目　次

推進協議会の構成

組織名	代表者	都道府県
学校法人宮崎総合学院 宮崎情報ビジネス専門学校	理事長　川越　宏樹	宮崎県
学校法人北杜学園 仙台大原簿記情報公務員専門学校	理事長　鈴木　忠	宮城県
学校法人龍澤学館 盛岡情報ビジネス専門学校	理事長　龍澤　正美	岩手県
学校法人中央総合学園 中央情報経理専門学校高崎校	理事長　中島　利郎	群馬県
学校法人浦山学園 富山情報ビジネス専門学校	理事長　浦山　哲郎	富山県
学校法人麻生塾 麻生情報ビジネス専門学校	副理事長　古野　金廣	福岡県
学校法人コンピュータ総合学園 神戸電子専門学校	校長　福岡　壯治	兵庫県
学校法人三橋学園 船橋情報ビジネス専門学校	校長　鳥居　高之	千葉県
学校法人コンピュータ総合学園 神戸情報大学院大学	教授　田村　武志	兵庫県
宮崎県	商工観光労働部商業支援課 情報・サービス業担当主幹 津曲　雄二	宮崎県
宮崎県	県民政策部総合交通課 向畑　公俊	宮崎県
株式会社宮崎情報処理センター	代表取締役社長 川崎　友裕	宮崎県
株式会社宮崎コミュニティーカレッジ	部長　髙見　裕貴	宮崎県
株式会社インタープロ	代表取締役社長 南　克浩	宮崎県
社団法人組込みシステム技術協会	事務局　近森　満	東京都
株式会社クオリティ・エージェント	代表取締役社長 石川　和信	東京都

第0部　はじめに

第1章　事業概要

近年、スマートフォンをはじめとしたモバイル端末市場が急激に拡大している。また、東日本大震災発生直後から、スマートフォン向けの震災・防災関連アプリケーションが急増している。それに伴い、モバイル端末を対象としたIT技術者の需要拡大への対応が急務である。そこで本事業では、産学の連携によるコンソーシアムを構築し、モバイル技術者の育成に関する課題の抽出・解決を図った。そして、より具体的な人材評価基準を策定し、職業実践的教育の質の保証・向上のあり方、及び職業能力を育成する効果的学習体系の構築等について検討した。さらに、策定した人材評価基準をもとに、モバイルアプリケーション開発人材を育成する教育プログラムを構築した。構築した教育プログラムの一部を、実証講座として複数校で試行導入し、教育プログラムの有用性を検証した。本事業で策定した人材評価基準と構築した教育プログラムを、被災県の専門学校を初めとする全国の専門学校、大学等に普及を図り、震災からの復興を支援できるモバイルアプリケーション開発人材を育成していきたい。

第2章　事業目的

東日本大震災の発生直後から、スマートフォン向けの震災・防災関連のアプリケーションが急増した。例えば、地震速報を通知するものや、避難場所情報等を地図に表示するものなどが該当する[1]。このような状況から、スマートフォン等のモバイル端末を活用することで、震災からの復興や防災を支援しようという気運が高まっていることがわかる。

それに加えて、近年のスマートフォンをはじめとしたモバイル端末市場の成長はめざましい。MM 総研によるスマートフォン市場調査では、2010 年度の国内におけるスマートフォンの出荷台数は 855 万台に上り、前年比 3.7 倍になったと発表されている[2]。2011 年度も、国内各キャリアからスマートフォンが次々に発表されている。ジーエフケーマーケティングサービスジャパンの調査によると、震災が発生して 3 週間後の 2011 年 4 月第 1 週には、携帯電話販売台数に占めるスマートフォンの割合が 5 割を超えている[3]。今後も、タブレット PC も含めて、モバイル端末市場はさらに拡大していくことが考えられる。

これらの認識のもと、本事業では、産学の連携によるコンソーシアムを構築し、震災からの復興を支援できるモバイルアプリケーション開発人材の評価基準を策定することを目的とした。そして、策定した評価基準をもとに、モバイルアプリケーション開発人材の PBL 型教育プログラム[4]を構築した。さらにその実施及び普及を通して、モバイルアプリケーション開発人材を育成することで、震災からの復興を支援し、さらにはモバイル端末市場

---

1 代表的な例を巻末資料 1 に紹介しているので参照していただきたい。

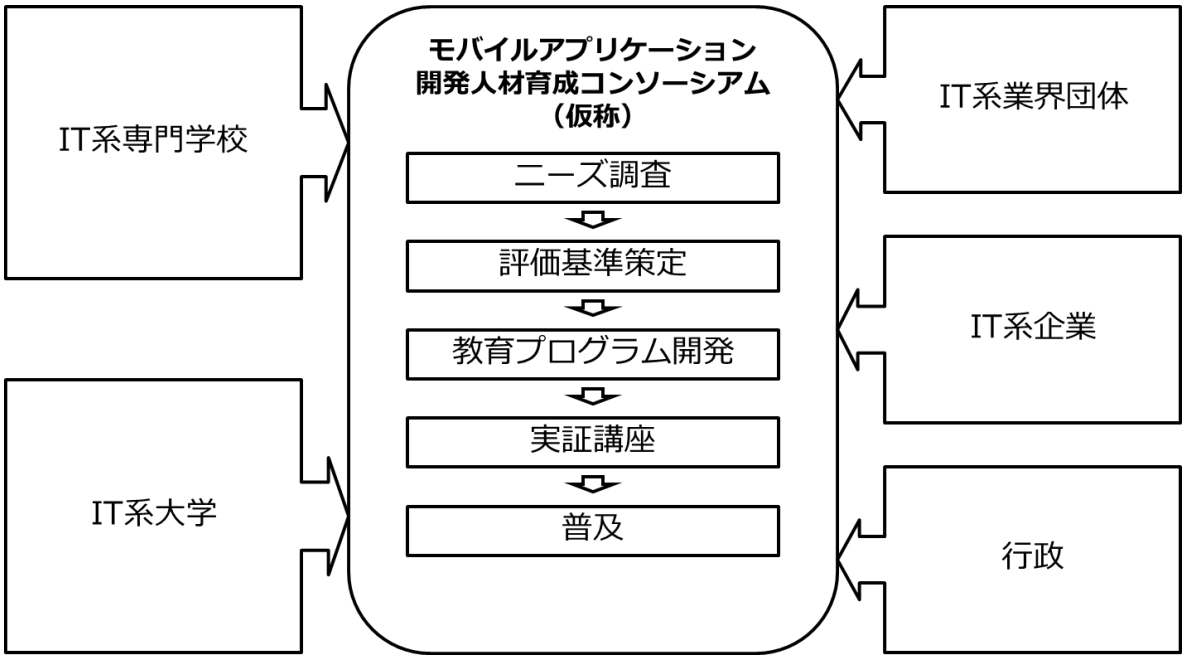
2 http://www.sem-r.com/news-2011/20110511040837.html　より

3 http://www.gfkjpn.co.jp/update_file/pdf/257.pdf　より

4 PBL：擬似的に再現された現場作業を通じて仕事を体得させる教育手法（Project Based Learning）

の発展に貢献することを目的とした。

本事業の全体スキームを図に表すと、次の図のようになる。



図表 1　本事業の全体スキーム

第3章　教育プログラム・教材の開発内容等

■開発の基本的な手順

本事業では、震災の復興を支援できるモバイルアプリケーション開発人材を育成するための教育プログラムを開発した。そこでは、まず、育成するモバイルアプリケーション開発人材の要件を明確にするために、評価基準を策定した。

（1）モバイルアプリケーション開発人材の評価基準策定

評価基準は、キャリアフレームワーク、能力評価基準、研修ロードマップから構成した。

a.キャリアフレームワーク

職種・専門分野とレベルを対応づけたマトリックス形式とした。学生レベルから社会人レベルへのキャリアパスがシームレスに接続する構成で検討した。

b.能力評価基準

各レベル／職種・専門分野において求められる能力項目、達成度・評価基準をまとめた。具体的な技術項目で記述し、教育機関や企業が、それぞれの状況に合わせて活用しやすくなるように工夫した。

c.研修ロードマップ

職種・専門分野ごとに、そのレベルを目指すために履修すべき研修を明確にした。

なお、我が国のエレクトロニクス・IT 産業の成長の活路とするため、クラウド・コンピューティング政策が推進されている（経済産業省『産業構造ビジョン 2010』）ことも考慮し、クラウド分野にも対応した枠組みとした。

（2）PBL 型教育プログラムの構築

本教育プログラムでは、Android を搭載したスマートフォン等のモバイル端末で動作する震災・防災関連アプリケーションの開発を題材とする。Android は、米 Google 社が中心となって開発したモバイル端末用のプラットフォームである。オープンソースソフトウェアであり、開発ツールも含めて全て無償で入手・利用することができるため、教育の題材としては非常に扱いやすい。開発言語は、多くの IT 系専門学校・大学等の授業で学習する Java を用いるため、学習者にとってもなじみやすい。また、個人・組織にかかわらず、開発したアプリケーションを Android Market[5]というアプリケーション配布・販売サイトで自由に配布（無償）・販売（有償）が可能であるため、学習者の動機づけになりやすい。さらに、Google 社のクラウド・コンピューティングを利用したサービスとの親和性が高いため、先述のクラウド・コンピューティング政策とも方向性が合致している。

実際のモバイルアプリケーション開発プロジェクトでは、個人ではなく、チームで取り組むことが多いことを考慮し、本教育プログラムでも、チーム内で密接にコミュニケーションを取りながら開発を進める PBL 形式を採用した。また、企画段階からプロジェクトに取り組ませることは、学習者にとって多大な負担を強いる場合があることから、最終的にターゲットとなるシステムを予め作成しておき、それに機能を追加・拡張する作業に取り組ませる、ミドルスタート方式で行った。

このような形式で、様々な震災・防災関連アプリケーションの開発を題材とし、テクノロジー分野のスキルをもととする実践力と、コミュニケーションやネゴシエーションなどのパーソナルスキルの向上を目指した。教育プログラム全体の講座時間数は 180 時間以上とするが、開発するアプリケーション等によっていくつかのユニットに分割し、一部を運用できるような構成にした。

[5] Android Market：https://market.android.com/

第4章　事業に対するニーズ

■事業の必要性

東日本大震災の発生から、スマートフォン等のモバイル端末を生活の中でより活用する人が、被災地を中心に増加している。電話やテレビの使えない被災地では、スマートフォンを利用して、ワンセグ放送[6]や twitter[7]、mixi[8]や facebook[9]に代表される SNS[10]等で震災に関する情報を得ていた、という事例が多数見受けられる。先述の震災・防災関連アプリケーションも含めて、モバイル端末が震災からの復興や防災に活用できることが明らかになった。モバイル端末市場がさらに拡大していくことからも、モバイルアプリケーション開発人材に対するニーズも、「量」と「質」の両面で高まることが予想される。

しかし、IT 人材全体では、「量」に対する不足感は緩和傾向にあるが、その一方で、「質」に対する不足感が高い状況が続いている。実際、独立行政法人情報処理推進機構が発表している『IT 人材白書 2011』によると、IT 企業を対象とした 2008 年の調査では、IT 人材の「量」について「特に不足はない」との回答が 17.1%、「やや過剰である」が 2.6%となっている。逆に、「大幅に不足している」が 16.2%、「やや不足している」が 59.4%と、不足しているという回答の方が大幅に上回っている。それが、2009 年の同じ調査では、「特に不足はない」が 35.7%、「やや過剰である」が 13.0%と、不足感が緩和されている。2010 年の調査でも、「特に不足はない」「やや過剰である」がそれぞれ 39.2%、10.1%となっており、不足感が緩和されている傾向にある。その一方で、IT 人材の「質」については、2010 年の調査では「大幅に不足している」「やや不足している」がそれぞれ 26.1%、59.7%である。質的不足感が高い状況は、図表 3 に示すように、ここ 3 年間続いている。

---

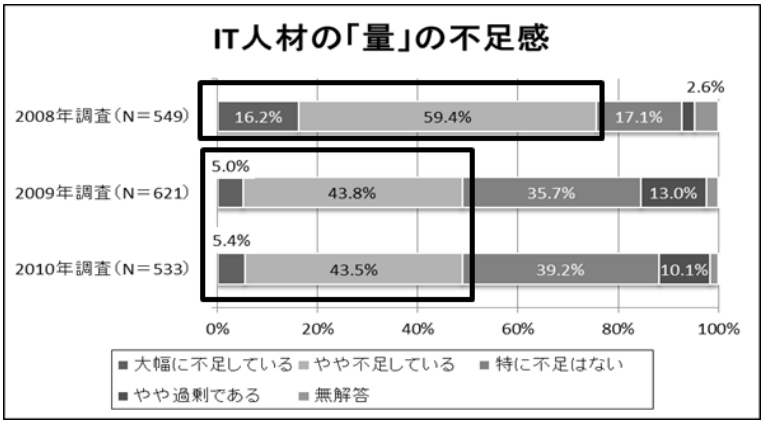
6 携帯電話やスマートフォンなどのモバイル端末を受信対象とする地上デジタルテレビ放送。

7 個々のユーザが「ツイート」と呼ばれる 140 字以内の短文を投稿・閲覧できるサービス。日本では、「ツイート」は「つぶやき」と呼ばれている。URL：http://twitter.com/

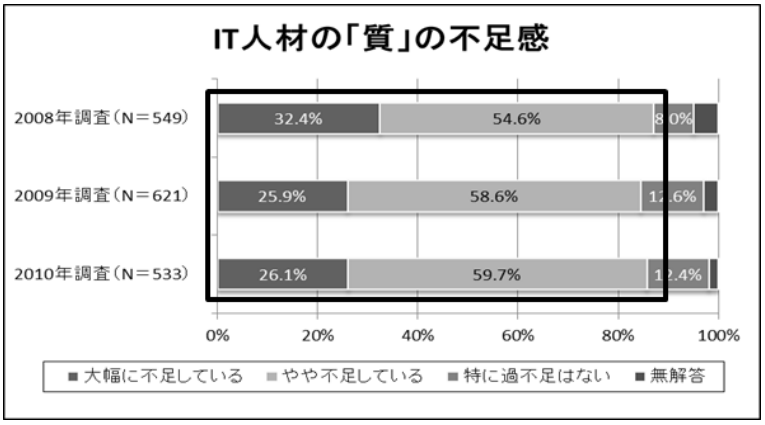
8 日本最大級の SNS。2010 年 4 月 14 日現在の有効 ID 数は約 2,000 万人。URL：http://mixi.jp/mixi

9 アメリカ合衆国でサービスが開始された世界最大級の SNS。2010 年 7 月 22 日に、アクティブユーザ数が全世界で 5 億人を突破した。URL：http://www.facebook.com/

10 ソーシャル・ネットワーキング・サービス。社会的なネットワークをインターネットで構築できるサービスのこと。



図表 2　IT 人材の「量」の不足感



図表 3　IT 人材の「質」の不足感

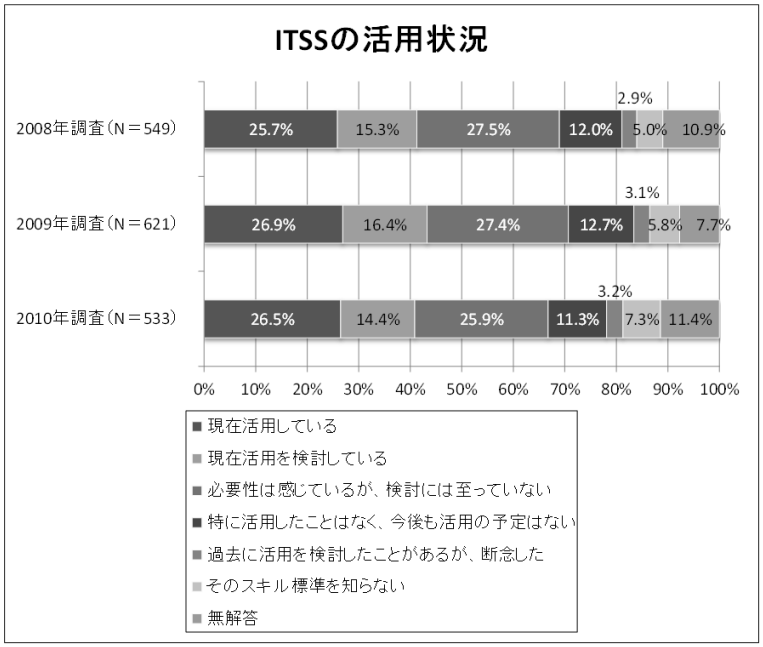
モバイル端末業界に限定すると、スマートフォン向けのモバイルアプリケーション開発人材のニーズが急速に高まっており[11]、「量」も「質」も不足感が高いことが考えられる。教育機関には、「量」と「質」の両面でモバイル端末業界の求める人材を育成することが求められているのである。

一方、教育機関に目を向けると、近年、産業界が求める人材の育成を重視していることがわかる。実際、『IT 人材白書 2011』によると、情報系専門学校や高等専門学校で 80%以上が、情報系大学でも 60%以上が、産業界が求める人材の育成を重視していると回答している。現に、グループ演習や PBL、コミュニケーション能力の強化を目的とする科目など、実践的な教育科目が普及しつつある。さらに、実践教育の受講生の 70.4%が、「システム開発手法や開発プロセス」の知識や経験が実務でも役立つと回答している。

11 テレビ東京『ワールドビジネスサテライト』2011 年 11 月 18 日の特集より。
http://www.tv-tokyo.co.jp/mv/wbs/feature/post_11083

教育機関での実践教育はある程度の成果を出しているが、モバイル端末業界では、特に「質」の面で人材の不足感が高いというギャップが存在している。そこで、産学の連携によって、こうしたギャップを解消し、より「質」の高い人材をより「多く」育成していく枠組みが必要である。

また、IT 人材の知識や技能を評価する指標として、IT スキル標準（ITSS）がある。IT 企業においては、企業戦略に沿った戦略的な人材育成・調達を行う際の指標として活用されることが期待されている。しかし、実際に ITSS を活用している IT 企業は 26.5%に留まっている（『IT 人材白書 2011』）。IT 企業が ITSS を活用していない理由としては、「ITSS が自社の業務内容と合わない」「メリットがわからない」「内容がわかりにくい」などというものが多い。そのため、ITSS や ETSS（組み込みスキル標準）等の各種 IT 系の能力評価基準を参照、分析し、より具体的な技術項目で評価する能力評価基準を策定する必要がある。



図表 4　ITSS の活用状況

さらに、産学連携の課題として、支援企業にとっての直接的なメリットが見えにくい、という点がある。しかし、企業が実務につながる知識やスキルを教育機関に求める一方で、産学連携による実践的な教育効果は明らかになりつつある。そこで、本事業により、産学双方が互いのメリットを理解し、産学がより連携を強められるコンソーシアムを構築する必要がある。また、コンソーシアムを基に、教育機関が企業の協力を得ることで、モバイル端末業界を主導する人材を輩出するための教育を推進し、継続していくことができる。

■ニーズ調査

本事業では、被災県の IT 技術者を中心に、幅広い範囲を対象にしたアンケート等の調査を実施した。主に次のような項目を意識して、それぞれに固有な課題を抽出し、解決の方向性を検討した。

（1）モバイルアプリケーション開発人材の育成に固有な課題の抽出・解決

a.IT 系の知識・技能の習得に関する課題

b.ヒューマンスキルの向上に関する課題

c.東日本大震災を教訓とした、防災や大規模災害からの復興に関わる課題

（2）教育の質保証・向上の方法に関する検討

a.カリキュラムに求められる要件

b.モバイルアプリケーション開発人材の技能に関する達成度水準の策定とその測定方法

c.アウトカムズの評価方法

第1部　調査

第1章　被災県の IT エンジニアに関するニーズ調査

1.1 調査の目的・方法

■目的

被災県における IT エンジニアの育成に固有な課題の抽出・解決を目的として、次の各項目に関する調査を行った。

a.IT 系の知識・技能の習得に関する課題

b.ヒューマンスキルの向上に関する課題

c.東日本大震災を教訓とした、防災や大規模災害からの復興に関わる課題

■方法

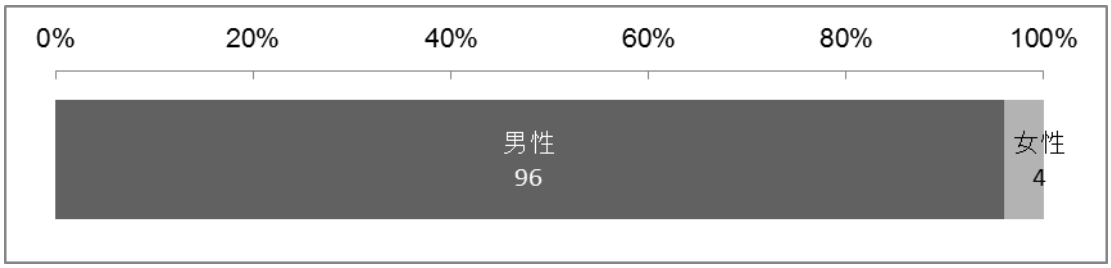
インターネットリサーチを利用して、岩手県、宮城県、福島県において働く IT エンジニア（男女 100 人）を対象とし、平成 24 年 2 月 18 日～28 日の期間で調査を実施した。

1.2 調査の結果

■基本的属性に関する回答状況

1　性別

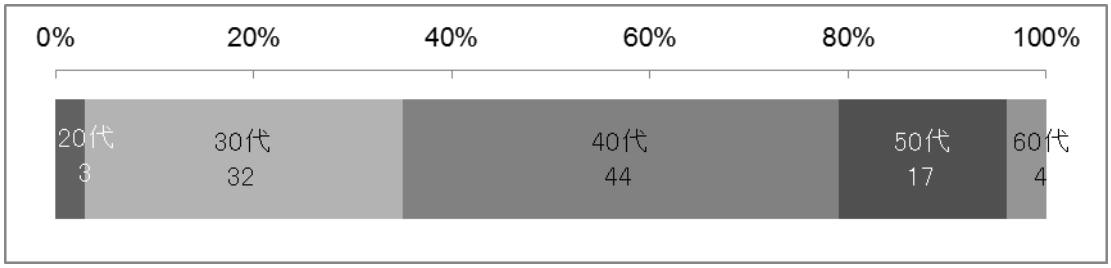
		n	%
	全体	100	100.0
1	男性	96	96.0
2	女性	4	4.0



圧倒的に男性が多かった。

2　年代

		n	%
	全体	100	100.0
1	20 代	3	3.0
2	30 代	32	32.0
3	40 代	44	44.0
4	50 代	17	17.0
5	60 代	4	4.0

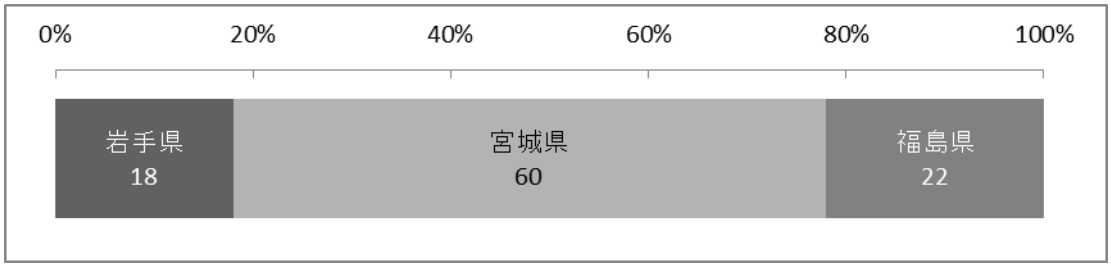


30 代、40 代で全体の 3/4 を占め、次が 50 代である。20 代が非常に少なかった。

■各質問項目に対する回答状況

Q1. あなたの勤務地を選んでください。

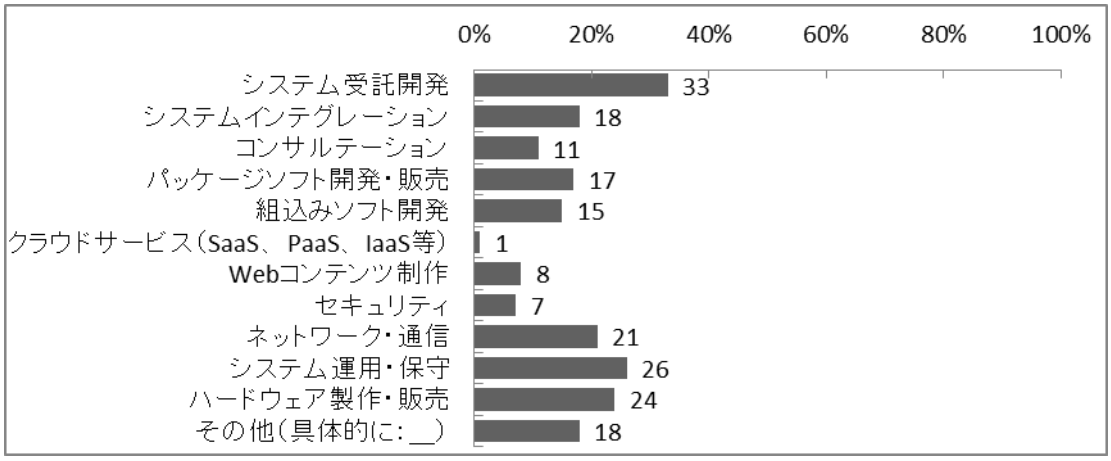
		n	%
	全体	100	100.0
1	岩手県	18	18.0
2	宮城県	60	60.0
3	福島県	22	22.0
4	その他	0	0.0



圧倒的に宮城県が多く 6 割を占め、他を岩手県と福島県がほぼ二分していた。

Q2. あなたの勤務している会社の主な業務は次のどれですか。（いくつでも）

		n	%
	全体	100	100.0
1	システム受託開発	33	33.0
2	システムインテグレーション	18	18.0
3	コンサルテーション	11	11.0
4	パッケージソフト開発・販売	17	17.0
5	組込みソフト開発	15	15.0
6	クラウドサービス（SaaS、PaaS、IaaS 等）	1	1.0
7	Web コンテンツ制作	8	8.0
8	セキュリティ	7	7.0
9	ネットワーク・通信	21	21.0
10	システム運用・保守	26	26.0
11	ハードウェア製作・販売	24	24.0
12	その他（具体的に：__）	18	18.0



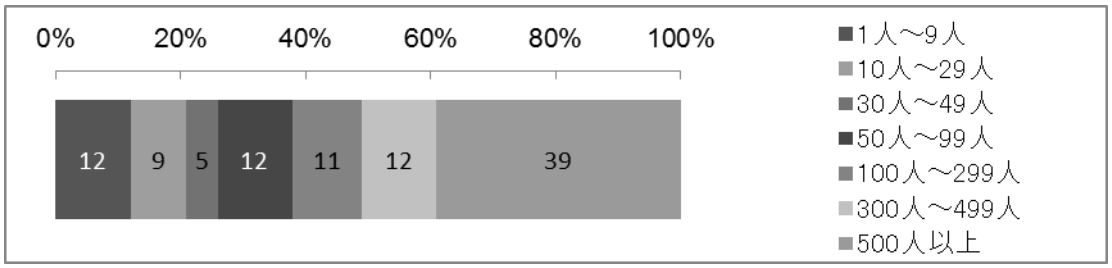
システム受託開発が最も多く、次にシステム運用保守が続くのは一般的傾向に近い。クラウドサービスがメインとなっているのがわずか１というのは意外な結果であった。パッケージソフト開発・販売の概念と重複した部分もあるからかと思われる。

その他の記述内容

コンテンツプロバイダー、電子部品製造、家電製造・販売、研究開発、建設業、ガス事業、設備の維持管理、製造業、アルゴリズム開発、エネルギー関係、エネルギー、電気機器製造業、飲料、サービスエンジニア、解析、

Q3. あなたの勤務している会社の従業員数は次のうちのどれですか。

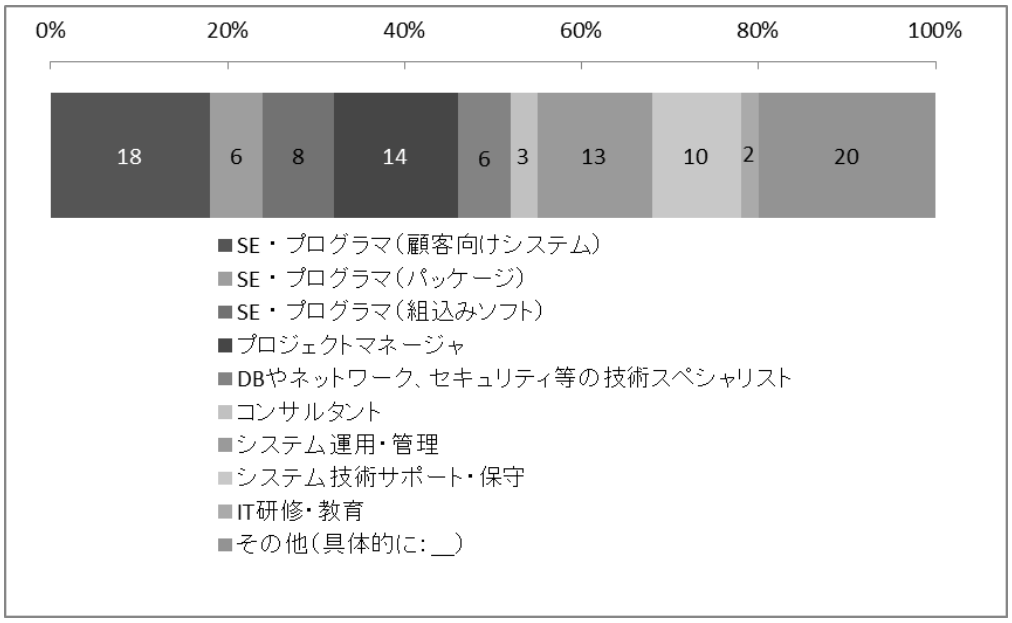
		n	%
	全体	100	100.0
1	1 人～9 人	12	12.0
2	10 人～29 人	9	9.0
3	30 人～49 人	5	5.0
4	50 人～99 人	12	12.0
5	100 人～299 人	11	11.0
6	300 人～499 人	12	12.0
7	500 人以上	39	39.0



圧倒的に大企業が多いことがわかる。

Q4. あなたの職種は次のどれですか。複数思い当たる場合は、もっともあてはまるものをひとつ選んでください。

		n	%
	全体	100	100.0
1	SE・プログラマ（顧客向けシステム）	18	18.0
2	SE・プログラマ（パッケージ）	6	6.0
3	SE・プログラマ（組込みソフト）	8	8.0
4	プロジェクトマネージャ	14	14.0
5	DB やネットワーク、セキュリティ等の技術スペシャリスト	6	6.0
6	コンサルタント	3	3.0
7	システム運用・管理	13	13.0
8	システム技術サポート・保守	10	10.0
9	IT 研修・教育	2	2.0
10	その他（具体的に:__）	20	20.0



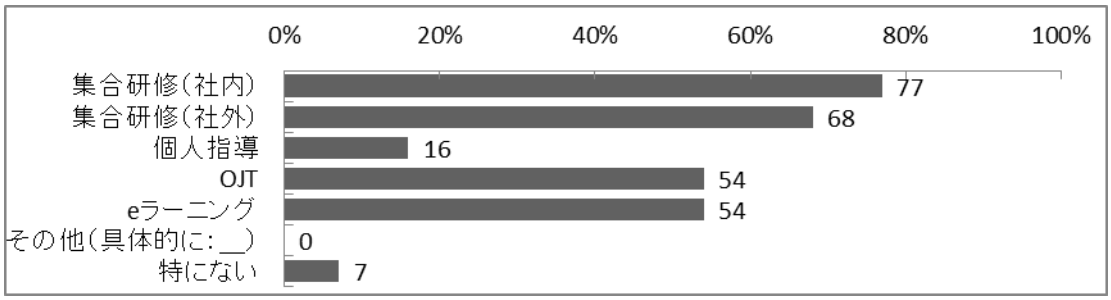
SE・プログラムで全体の 4 割。他の多くはマネジメント系。技術的なスペシャリストは少ない。

その他の記述内容

開発、ハード開発設計、製品開発、材料、機械設計、研究開発、機械・機構設計、設備の維持管理、設計、アルゴリズム開発、専門技術者、管理職、回路設計、ハード面の保守管理、解析

Q5. あなたが受けたことがある研修の形式は次のどれですか。（いくつでも）

		n	%
	全体	100	100.0
1	集合研修（社内）	77	77.0
2	集合研修（社外）	68	68.0
3	個人指導	16	16.0
4	OJT	54	54.0
5	e ラーニング	54	54.0
6	その他（具体的に:__）	0	0.0
7	特にない	7	7.0



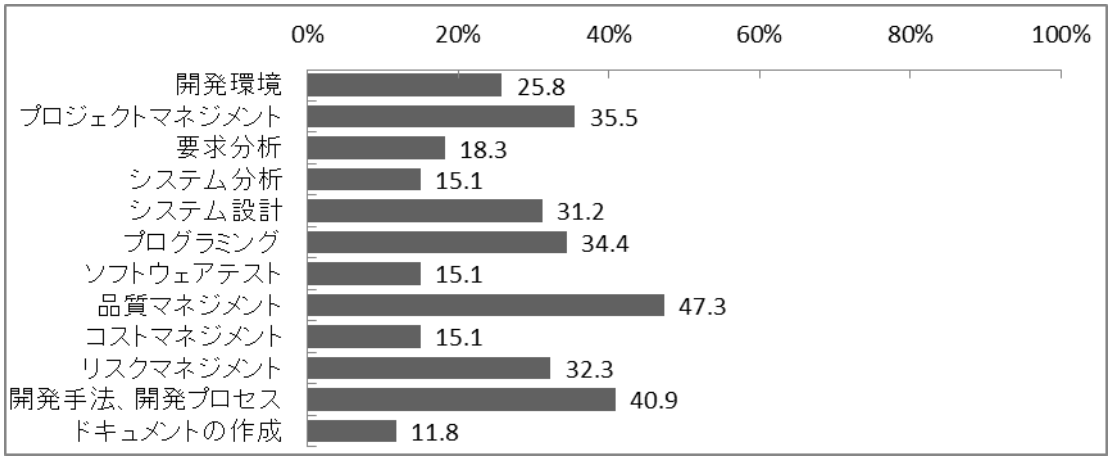
集合研修主体で、以下、OJT、e ラーニングの順である。ここまでを半分以上が経験しているが、これらはほぼ一般的な傾向と思われる。

その他

(なし)

Q6. あなたが受けたことがある研修の内容は以下の中ではどれですか。最も近いものを選択してください。（いくつでも）

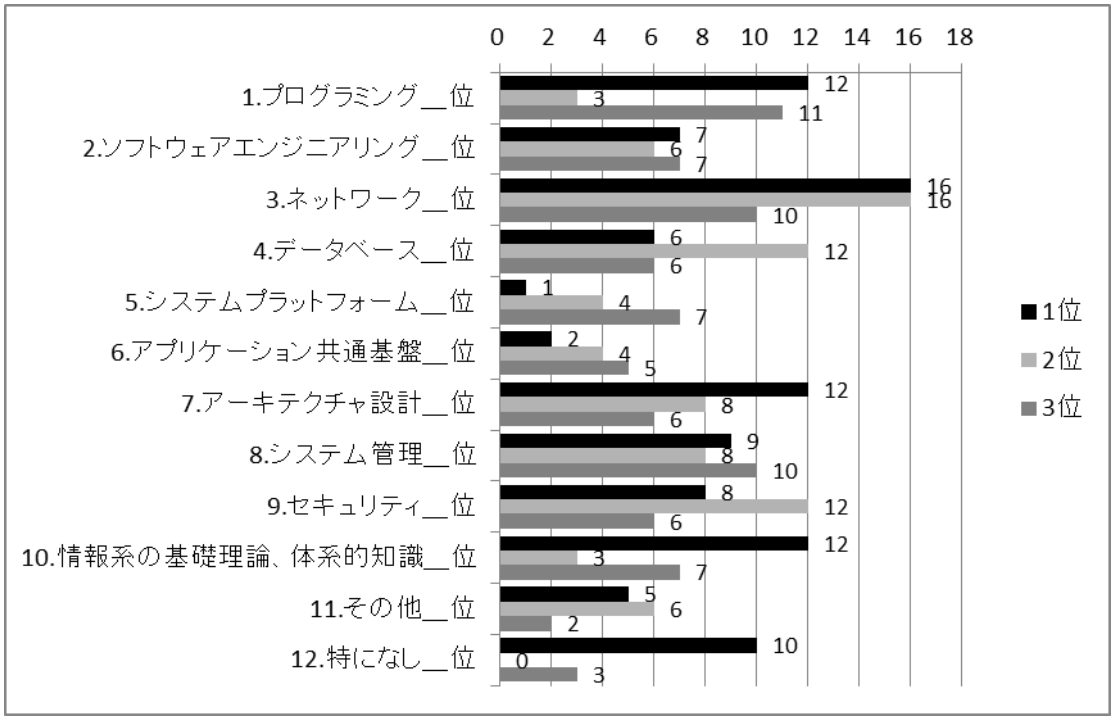
		n	%
	全体	93	100.0
1	開発環境	24	25.8
2	プロジェクトマネジメント	33	35.5
3	要求分析	17	18.3
4	システム分析	14	15.1
5	システム設計	29	31.2
6	プログラミング	32	34.4
7	ソフトウェアテスト	14	15.1
8	品質マネジメント	44	47.3
9	コストマネジメント	14	15.1
10	リスクマネジメント	30	32.3
11	開発手法、開発プロセス	38	40.9
12	ドキュメントの作成	11	11.8



分散しているが、どちらかというとマネジメント系の研修の受講例が多い。

Q7. 自分が身につけている技術面の知識やスキルで、特に向上させたいと考えているものは、以下の中ではどれですか。(1～3 の半角数字でご記入ください)

	n	1 位	2 位	3 位
1.プログラミング__位	26	12	3	11
	100.0	46.2	11.5	42.3
2.ソフトウェアエンジニアリング__位	20	7	6	7
	100.0	35.0	30.0	35.0
3.ネットワーク__位	42	16	16	10
	100.0	38.1	38.1	23.8
4.データベース__位	24	6	12	6
	100.0	25.0	50.0	25.0
5.システムプラットフォーム__位	12	1	4	7
	100.0	8.3	33.3	58.3
6.アプリケーション共通基盤__位	11	2	4	5
	100.0	18.2	36.4	45.5
7.アーキテクチャ設計__位	26	12	8	6
	100.0	46.2	30.8	23.1
8.システム管理__位	27	9	8	10
	100.0	33.3	29.6	37.0
9.セキュリティ__位	26	8	12	6
	100.0	30.8	46.2	23.1
10.情報系の基礎理論、体系的知識__位	22	12	3	7
	100.0	54.5	13.6	31.8
11.その他__位	13	5	6	2
	100.0	38.5	46.2	15.4
12.特になし__位	13	10	0	3
	100.0	76.9	0.0	23.1



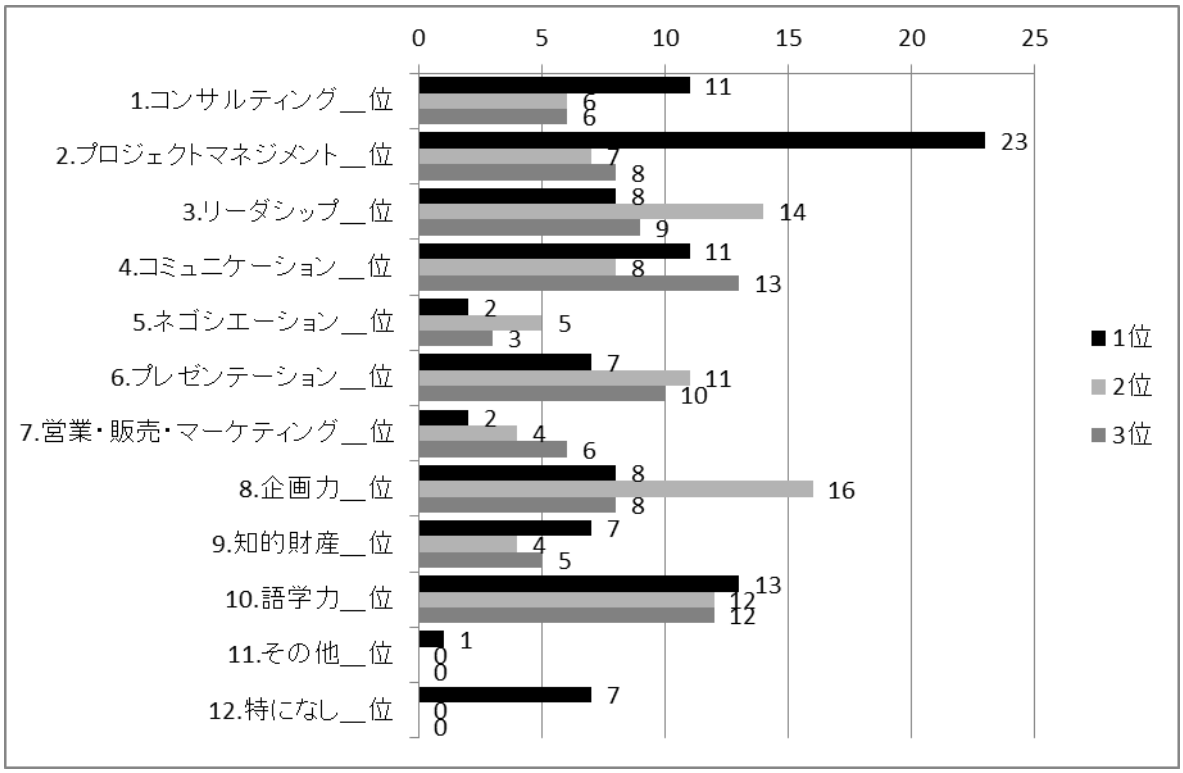
1位はネットワーク、次いで、プログラミング、アーキテクチャ設計、情報系の基礎理論の順であった。前三者は一般的な傾向とほぼ同様。それに加えて基礎的な部分をしっかり伸ばしておきたいというニーズが意外に多いことがわかる。

その他

マネジメント力。現在組織職として勤務しているが、その力・経験が不足していると感じている
物理、マネジメント
解析系処理が業務なので、対象の解析系の理論
Windows Server、顧客側の専門知識、技術、品質管理

Q8. 自分が身につけている技術面以外の知識やスキルで、特に向上させたいと考えているものは以下の中ではどれですか。(1～3 の半角数字でご記入ください)

	n	1 位	2 位	3 位
1.コンサルティング__位	23	11	6	6
	100.0	47.8	26.1	26.1
2.プロジェクトマネジメント__位	38	23	7	8
	100.0	60.5	18.4	21.1
3.リーダシップ__位	31	8	14	9
	100.0	25.8	45.2	29.0
4.コミュニケーション__位	32	11	8	13
	100.0	34.4	25.0	40.6
5.ネゴシエーション__位	10	2	5	3
	100.0	20.0	50.0	30.0
6.プレゼンテーション__位	28	7	11	10
	100.0	25.0	39.3	35.7
7.営業・販売・マーケティング__位	12	2	4	6
	100.0	16.7	33.3	50.0
8.企画力__位	32	8	16	8
	100.0	25.0	50.0	25.0
9.知的財産__位	16	7	4	5
	100.0	43.8	25.0	31.3
10.語学力__位	37	13	12	12
	100.0	35.1	32.4	32.4
11.その他__位	1	1	0	0
	100.0	100.0	0.0	0.0
12.特になし__位	7	7	0	0
	100.0	100.0	0.0	0.0



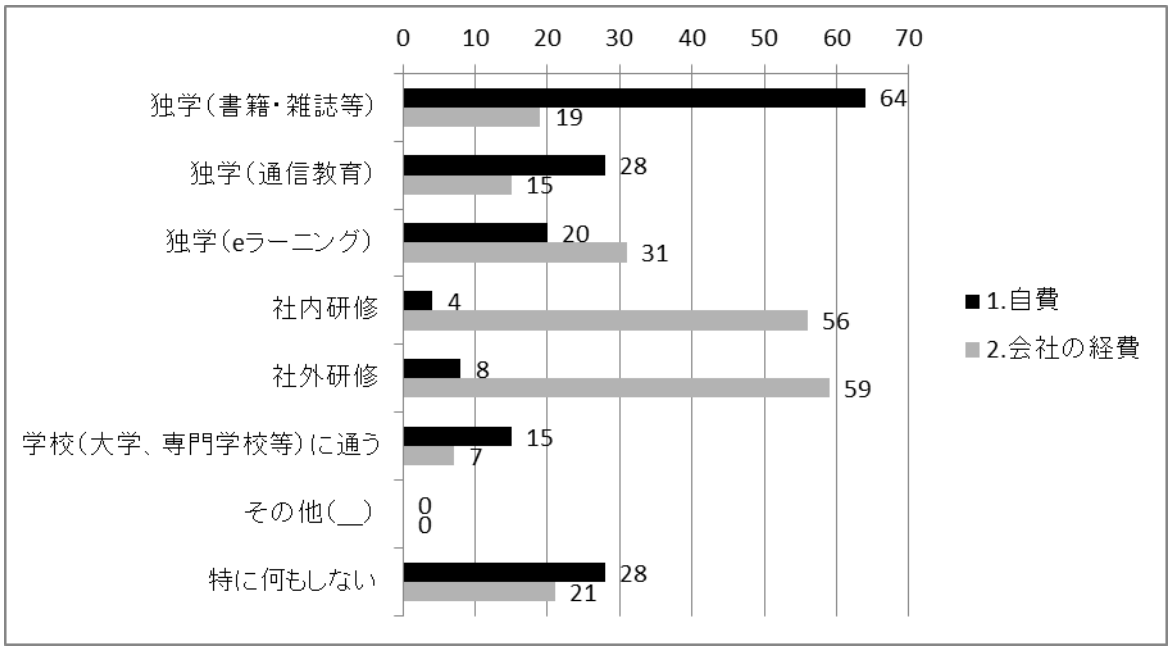
圧倒的にプロジェクトマネジメントが多かった。次いで企画力、リーダシップの順である。やはり、マネジメント系、より上流工程の部分でのニーズが大きい。その次が語学力で、1～3 位全体で見ると 4 割近くが重視しているのは興味深い。

その他

（なし）

Q9. 自分が身につけている知識やスキルを向上させるために、「自費」か「会社の経費」で学習をしようと考えているものをお答えください。（矢印方向にそれぞれいくつでも）

	1.自費		2.会社の経費	
n	100	100.0	100	100.0
独学（書籍・雑誌等）	64	64.0	19	19.0
独学（通信教育）	28	28.0	15	15.0
独学（e ラーニング）	20	20.0	31	31.0
社内研修	4	4.0	56	56.0
社外研修	8	8.0	59	59.0
学校（大学、専門学校等）に通う	15	15.0	7	7.0
その他（__）	0	0.0	0	0.0
特に何もしない	28	28.0	21	21.0



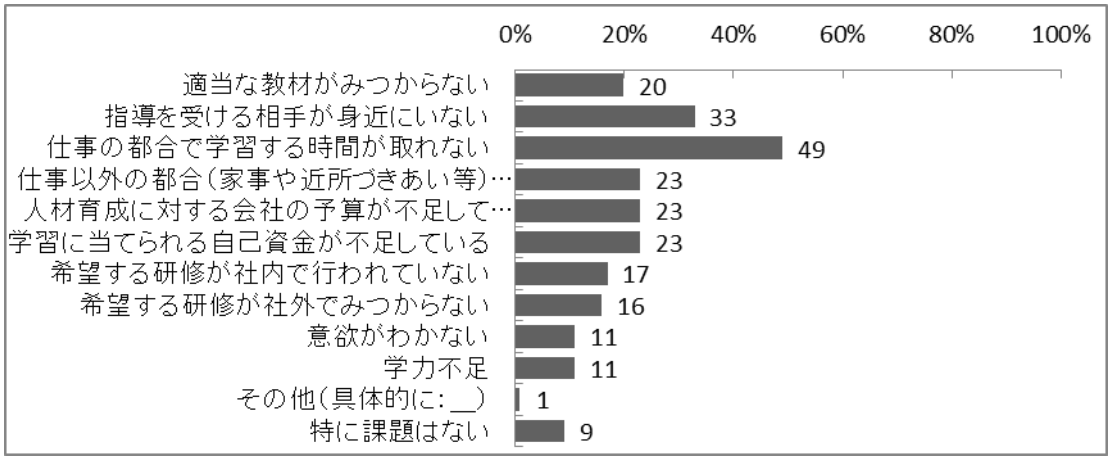
独学は自費で、研修は会社で、というハッキリとした傾向が見られる。独学だが会社の経費で、という回答は、e ラーニングの場合 3 割に上っている。

その他

（なし）

Q10. 自分が身につけている知識やスキルを向上させる上での課題として、どのようなものがありますか。（いくつでも）

		n	%
	全体	100	100.0
1	適当な教材がみつからない	20	20.0
2	指導を受ける相手が身近にいない	33	33.0
3	仕事の都合で学習する時間が取れない	49	49.0
4	仕事以外の都合(家事や近所づきあい等)で学習する時間が取れない	23	23.0
5	人材育成に対する会社の予算が不足している	23	23.0
6	学習に当てられる自己資金が不足している	23	23.0
7	希望する研修が社内で行われていない	17	17.0
8	希望する研修が社外でみつからない	16	16.0
9	意欲がわかない	11	11.0
10	学力不足	11	11.0
11	その他(具体的に:__)	1	1.0
12	特に課題はない	9	9.0



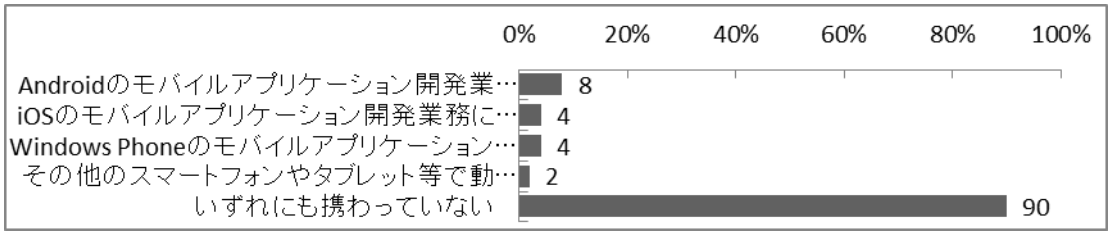
時間的な理由を挙げる者が多い(3、4)。2 の理由は被災地だからという事情からと考えられる。

その他

会社から費用が出ない

Q11. あなたは、Android や iOS、Windows Mobile を搭載したスマートフォンやタブレット等で動作するアプリケーション（モバイルアプリケーション）開発業務に携わっていますか。（いくつでも）

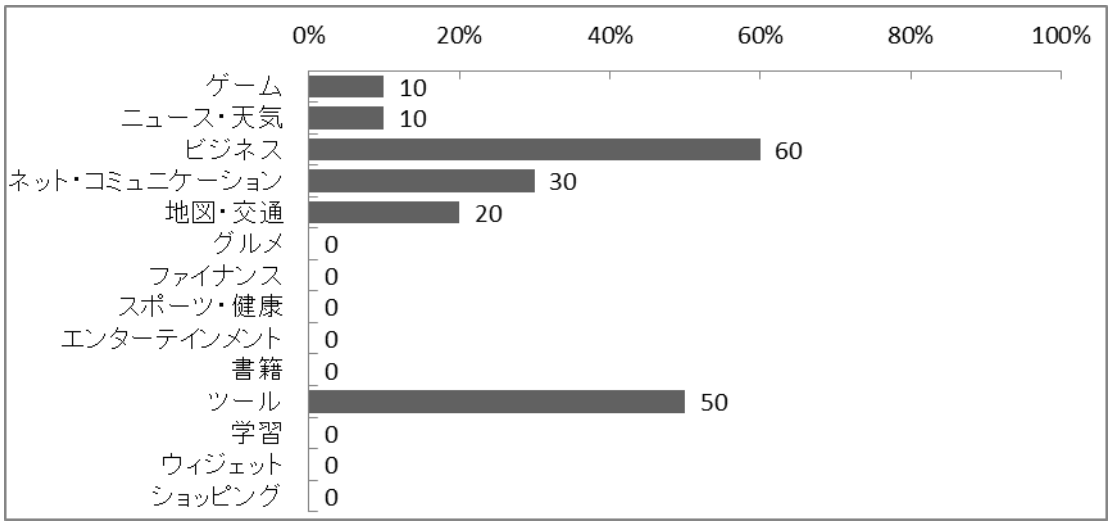
		n	%
	全体	100	100.0
1	Android のモバイルアプリケーション開発業務に携わっている	8	8.0
2	iOS のモバイルアプリケーション開発業務に携わっている	4	4.0
3	Windows Phone のモバイルアプリケーション開発業務に携わっている	4	4.0
4	その他のスマートフォンやタブレット等で動作するモバイルアプリケーションの開発に携わっている(フィーチャーフォン、ガラケーは含みません)	2	2.0
5	いずれにも携わっていない	90	90.0



非常に少ない。少ない中では Android が 8 割を占める。

Q12. どのようなモバイルアプリケーションの開発業務に携わっていますか。次の中から最も近いものを選択してください。（いくつでも）

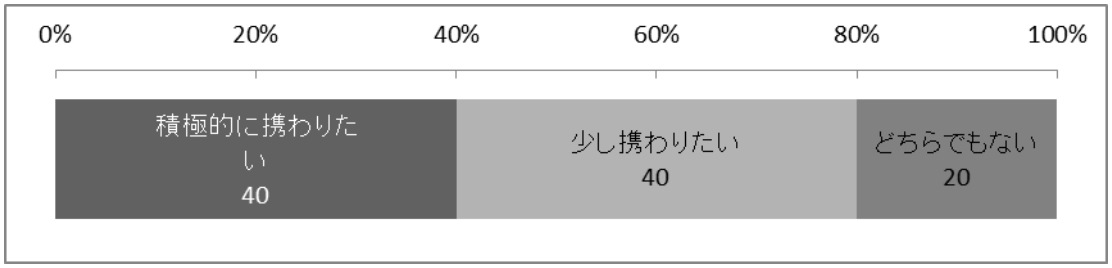
		n	%
	全体	10	100.0
1	ゲーム	1	10.0
2	ニュース・天気	1	10.0
3	ビジネス	6	60.0
4	ネット・コミュニケーション	3	30.0
5	地図・交通	2	20.0
6	グルメ	0	0.0
7	ファイナンス	0	0.0
8	スポーツ・健康	0	0.0
9	エンターテインメント	0	0.0
10	書籍	0	0.0
11	ツール	5	50.0
12	学習	0	0.0
13	ウィジェット	0	0.0
14	ショッピング	0	0.0



少ない中ではビジネスとツールが多い。

Q13. 今後も、モバイルアプリケーションの開発に携わりたいですか。

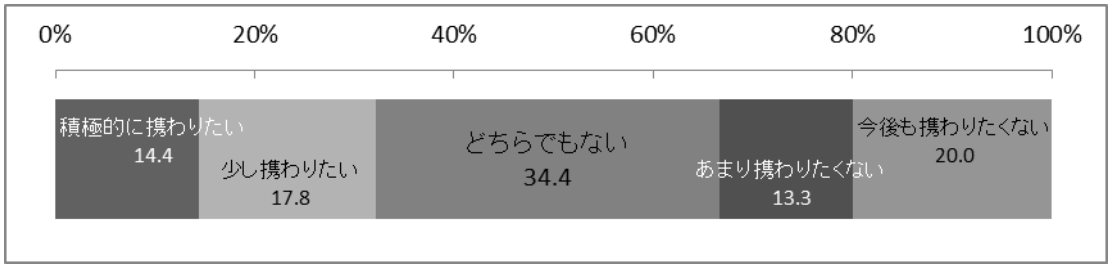
		n	%
	全体	10	100.0
1	積極的に携わりたい	4	40.0
2	少し携わりたい	4	40.0
3	どちらでもない	2	20.0
4	あまり携わりたくない	0	0.0
5	今後は携わりたくない	0	0.0



「積極的に」「少し」を問わず、携わりたい気持ちが 8 割出ている。

Q14. 今後、モバイルアプリケーションの開発に携わりたいですか。

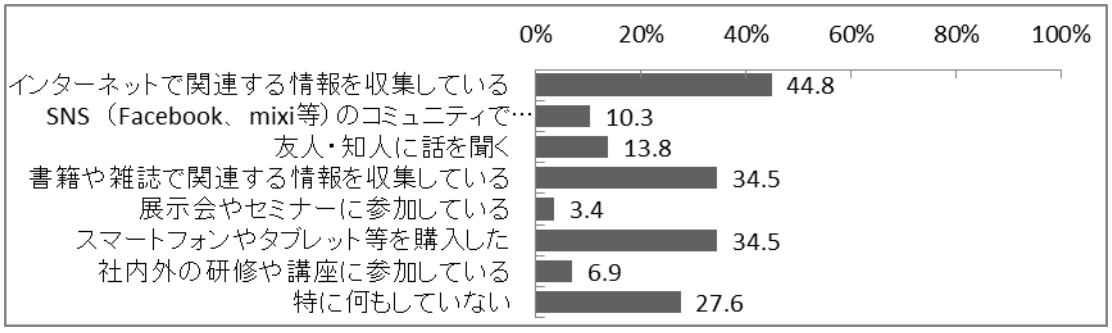
		n	%
	全体	90	100.0
1	積極的に携わりたい	13	14.4
2	少し携わりたい	16	17.8
3	どちらでもない	31	34.4
4	あまり携わりたくない	12	13.3
5	今後も携わりたくない	18	20.0



「携わりたい」、「どちらでもない」、「携わりたくない」に 3 等分されている。「どちらでもない」者に動機があれば、積極的に「携わりたい」のほうに動くような流動的状況であることがうかがえる。

Q15. モバイルアプリケーションの開発に携わるために行っていることはありますか。（いくつでも）

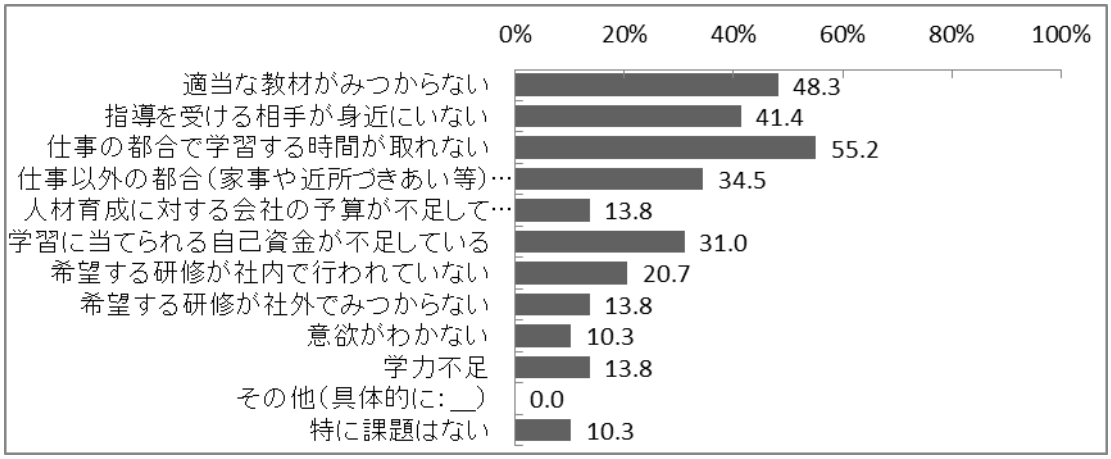
		n	%
	全体	29	100.0
1	インターネットで関連する情報を収集している	13	44.8
2	SNS（Facebook、mixi 等）のコミュニティで情報を収集している	3	10.3
3	友人・知人に話を聞く	4	13.8
4	書籍や雑誌で関連する情報を収集している	10	34.5
5	展示会やセミナーに参加している	1	3.4
6	スマートフォンやタブレット等を購入した	10	34.5
7	社内外の研修や講座に参加している	2	6.9
8	特に何もしていない	8	27.6



ネットや書籍等で情報を収集するなどの積極的な態度、自分でスマートフォンやタブレット PC に慣れるという姿などが少なくとも 1/3 以上垣間見られる。

Q16. モバイルアプリケーションの開発に携わるための準備で、何か課題はありますか。

	全体	29	100.0
1	適当な教材がみつからない	14	48.3
2	指導を受ける相手が身近にいない	12	41.4
3	仕事の都合で学習する時間が取れない	16	55.2
4	仕事以外の都合(家事や近所づきあい等)で学習する時間が取れない	10	34.5
5	人材育成に対する会社の予算が不足している	4	13.8
6	学習に当てられる自己資金が不足している	9	31.0
7	希望する研修が社内で行われていない	6	20.7
8	希望する研修が社外でみつからない	4	13.8
9	意欲がわかない	3	10.3
10	学力不足	4	13.8
11	その他(具体的に:__)	0	0.0
12	特に課題はない	3	10.3



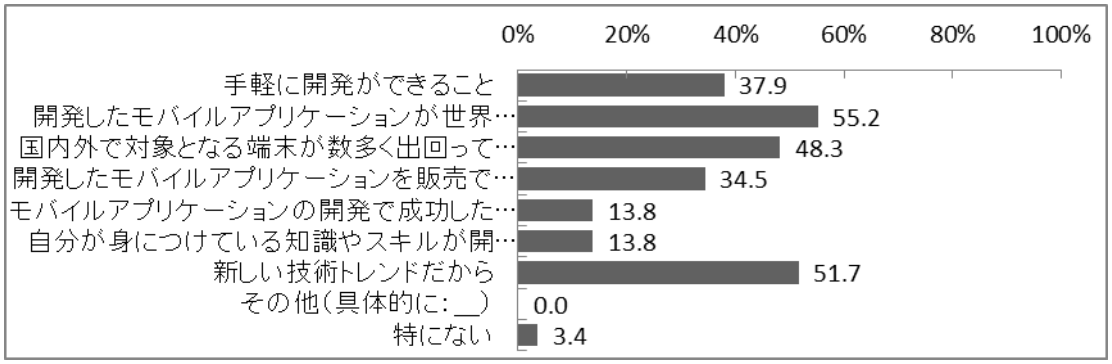
「時間」的ファクターを回答する者が多く、続いて「教材」のファクター、「人(教える人)」のファクターの順になっている。

その他

(なし)

Q17. モバイルアプリケーションの開発について、どのような点で魅力を感じていますか。（いくつでも）

		n	%
	全体	29	100.0
1	手軽に開発ができること	11	37.9
2	開発したモバイルアプリケーションが世界中で使用される可能性があること	16	55.2
3	国内外で対象となる端末が数多く出回っていること	14	48.3
4	開発したモバイルアプリケーションを販売できること	10	34.5
5	モバイルアプリケーションの開発で成功した人がたくさんいること	4	13.8
6	自分が身につけている知識やスキルが開発に活用できること	4	13.8
7	新しい技術トレンドだから	15	51.7
8	その他（具体的に:__）	0	0.0
9	特にない	1	3.4



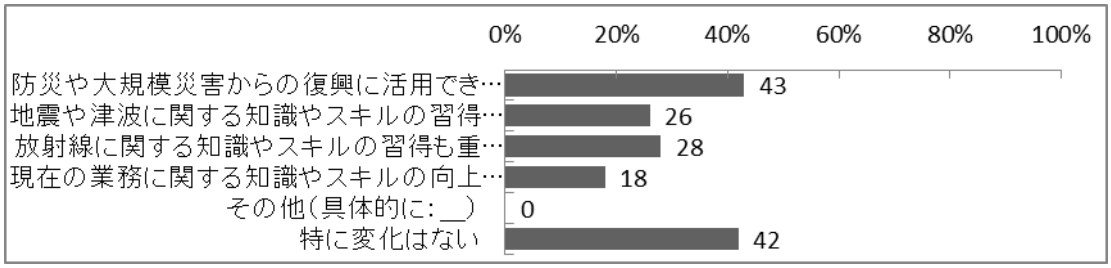
今後のマーケットの拡大、新規性に対して魅力を感じている傾向が見られる。

その他

(なし)

Q18. 東日本大震災の後、自身の IT の知識やスキルと、震災からの復旧・復興とを関連付けて考えることはありましたか。あなたの考えに近いものを以下から選んでください。(いくつでも)

		n	%
	全体	100	100.0
1	防災や大規模災害からの復興に活用できる知識やスキルの習得も重要だと考えるようになった	43	43.0
2	地震や津波に関する知識やスキルの習得も重要だと考えるようになった	26	26.0
3	放射線に関する知識やスキルの習得も重要だと考えるようになった	28	28.0
4	現在の業務に関する知識やスキルの向上をより重要視するようになった	18	18.0
5	その他(具体的に:__)	0	0.0
6	特に変化はない	42	42.0



半数に近い者が、「特に変化がない」と回答する一方で、復興に向けたニーズを感じていると思われる回答もほぼ同等かそれ以上ある傾向が見られる。

その他

(なし)

Q19. 以下に関して普段から感じていることがあればご自由にご記入ください。

・IT エンジニアのスキルアップ

・モバイルアプリケーションの今後の可能性

・IT を活用した被災地の復旧・復興

●IT エンジニアのスキルアップ

- コミュニケーション力
- 一番はやはりコミュニケーション。様々な人（部下）が居る為、何よりも難しい。
- IT 化が進むほどヒューマン的な対応ができるスキルが必要になると考えている。
- 上記のため学ぶ技術の幅が広い。
- 会社自体での人間に対する投資が少なすぎると思う
- スキルアップに欠かせない資金が不足しているのと、年齢に関係なく積極的な投資意欲が会社組織に不足していると特に感じる。人材を育てて使うという意識が全く見当たらない。
- 自身のスキルアップの必要性
- 勉強の時間はあまりとれない。
- モバイルアプリケーションのインターネットとの連携、双方向性についてもっと知る必要があると思います
- 若い人はスキルアップを目指すべき
- スキルアップはしたいが、歳を考えるとなかなか意欲がわかない。継続・飛躍の為には、新しい技術が必要だと思うが、正直生き残るのが優先でそこまで行かない
- 技術の種類が多すぎるためスキルアップが難しい
- 新しいものが多すぎる
- 開発現場にいると日々の作業に追われ、担当業務以外のスキルアップに時間を取ることができない
- スキルと成果の相関
- お先が見えないこの世の中、自分のやりたいことを集約し、いかに生き残っていくかを思案中である。
- 今はネットを検索すれば、それなりのコードを探す事が出来るため、自分で考えてアルゴリズムを組む開発者が少なくなった。そのため、自分で考え、効率のいいコードを書けない技術者がいないため、品質に問題が出てきている。

●モバイルアプリケーションの今後の可能性

- ユーザーサイドから見たアプリケーションの機能の重要さ。
- これからはモバイルアプリケーションが主流になると思う
- 被災時初期の IT インフラ消失時のバックアップなど、もっと早期に IT を活用する方

法について、色々と考えて置く必要がある事
- 携帯を衛星回線を通じてどこからでも何時でも繋がるようになれば良い
- 個人情報流出や GPS を利用した悪用の危惧
- 開発が難しい

●IT を活用した被災地の復旧・復興
- IT 開発が景気に左右されやすいと感じる。毎年仕事の量が増減するので、人員計画などを長期計画が立てにくく、どうしてもその場対応になりがち。
- 国や自治体がお金が無い
- 弊社は製造業のためソフト関連技術に関してファーム/アプリの隔てが無い。
- 災害時の通信確保
- IT 関連をどれだけスキルアップしても被災地復興にはさほど関係ないのでは？
- 部署ごとに仕事が分かれているので、被災地の復旧、復興の仕事はない。
- 個々の個性を伸ばして、結果それが彼らのスキルアップや会社業績への貢献、もしくは被災地（ここもそうです）の復旧・復興にどのような貢献ができるのか、難しい。
- 今まで以上に人と人を繋ぐ手助けが必要と感じた。
- 高齢の人が被災のケースが多い。これまで、携帯やスマートフォンに見向きもしなかった高齢の方々が、使うケースが増えている。
- こちらに重視を置いたアプリが、間接的に復旧復興への手助けになるのでは無いかと思う。
- 昨年の大震災以降、特に原発からの復興に IT を生かせないものかは思うところがある
- 被災地では、難聴者が続出しており会議に、集会に出席しても話がわからず引きこもっている人が大勢いる事を知っている。現在は要約筆記についてもっと活用出来ないか模索中である。

●その他
- 必要性と需要と今後の動向
- 事業継続計画をしっかり作りたい
- 一にも雇用、二にも雇用、三にも雇用と言ったこれはどこに行ったのかあ。
- 昨日、今日大手会社が倒産しているが明日はどこが倒産するのでしょうか。
- 現在メーカーの修理部門が主流で、あまりほかの事は考えていない

1.3 調査結果の分析

まず、ほとんどが 30 代以上の「男性」で、約 40%が従業員 500 名以上の比較的大きな企業の所属であったことが、回答者の特徴として挙げられる。このことに注意を払いながら、前節に掲げた回答結果を、本調査の目的に即して、次の a～c の観点から分析した。

a.IT 系の知識・技能の習得に関する課題

●マネジメント系が中心課題

Q8 で、向上させたいスキルの 1～3 位に挙がった項目の頻度に、それぞれ 3 点、2 点、1 点の重みを付けたスコアを計算し、60 点以上のスコアとなったスキル項目を点数の大きい順に並べると次のようになる。

・プロジェクトマネジメント（91 点）
・語学力（75 点）
・企画力（64 点）
・コミュニケーション（62 点）
・リーダシップ（61 点）

年齢層や現在就いている職種に対する回答を見ると、回答者の多くはプロジェクトマネジメントの経験があるか、または、そのレベルに近い層であることがうかがえる。したがって、差し迫ったスキルアップのニーズとしてプロジェクトマネジメントが突出して高いことは理解できる。一方、これまで受けてきた研修もマネジメント系の回答頻度が高く、このことを合わせて考えると、マネジメント系は研修の機会も多いが、スキルアップの必要性も高いことになり、それだけスキル要素が豊富で複雑であることをあらためて感じさせる結果となっている。

●語学力

Q7 で、語学力を 1 位に挙げた回答者の頻度は 4 番目であったが、2 位、3 位に挙げた回答者が多かったために､総合的に見たスコアでは語学力が 2 位であったことは注目される。IT 企業を取り巻くグローバル化の傾向が東北地方にも及んでいることがうかがえる。

●効率の高いスキルアップ方法が求められている

スキルアップの方法については､これまで受けてきた研修が､社内･社外研修､e ラーニングが中心（Q5）であり、今後受けたいと思っている研修も同様の傾向が見られる（Q9)。これらはオーソドックスな手法ではあるが、効率面を考えるとベターな方法であるとの認識がうかがえる。ただし、Q9 に対する回答を見る限り、一般に費用がかかるこれらの方法に対して自己資金を投入するところまで期待していないこともわかる。Q10 に対する回答

その他も合わせて考えれば、スキルアップに対する積極的な意思はあるものの、時間面と費用面の課題があって、これらを解決できる効率的なスキルアップ方法が求められているといえよう。

●モバイルアプリケーションの開発スキルの向上に対する期待も膨らんでいるが…

実際にモバイルアプリケーションの開発に携わっている回答者は 10%であった（Q11）が、そうでない回答者の 30%が携わってみたいと感じていた（Q14）。また、全体の 30%が何らかの形でモバイルアプリケーションの開発に関する情報収集を行っており（Q15）、「世界中で使用されている」「需要がある」「手軽に開発できる」「販売できる」といった点に魅力を感じている（Q17）ことも明らかになった。

一方で、これらのスキルに関する学習課題として、第一に時間的要素が、第二に教材の要素が、第三に金銭的要因が挙げられていた（Q16）。一般のスキル項目に関する課題の場合も第一は時間的要素であったが、第二は金銭的要因であった（Q10）。時間が足りないという要素は共通として、第二位が異なる点は注目すべきである。すなわり、モバイルアプリケーションの場合、新しい技術でかつ日進月歩の分野であるから、安定した定番の教材はなく、タイムリーにポイントを押さえた教材要素を供給し続けるような、また、そういう情報交換ができるような環境、場が求められているといえよう。また、そこでは、従来の方法にはない効果的な学習方法も求められている。

b.ヒューマンスキルの向上に関する課題

●コミュニケーション、リーダシップが求められている

前項の記述にある通り、スキルアップを図りたい項目のスコア値の第 4 位と第 5 位を、コミュニケーションとリーダシップが占めた。Q19 で求めたフリーアンサーでも、普段から「コミュニケーション」能力の向上を図る努力の必要性を訴える回答が散見される。

c.東日本大震災を教訓とした、防災や大規模災害からの復興に関わる課題

●東日本大震災は意識革命をもたらした

東日本大震災をきっかけとした意識等の変化をたずねた Q18 で、「防災や大規模災害からの復興に活用できる知識やスキルの習得も重要だと考えるようになった」と回答した者が 43%いたことは非常に注目される。その理由として第一に、社会的貢献を果たすという IT システム本来の目的に対して、自然に目が行く結果になったことが挙げられる。このような強烈な動機がないと、IT システムの社会的使命といってもどこか直接関係のない事柄のように感じるものである。それが、震災の被災地だからこそ、中には家族や友人、親戚といった身近な人の消息をたずねるような機会に接し、そこで必要な IT システムの要件を自

然に肌で感じた経験を持った者が多かったはずである。このように自然な流れによって、システム開発の目的志向性を高めたことは、技術者としてのレベル・ランクを一歩進めたぐらいのインパクトがあったはずである。

震災から 1 年が経過し、モバイルアプリケーションをソーシャルな存在の一部として位置づける意見や発想は Q19 のフリーアンサーに数多く見られる。この流れを、質の高いモバイルアプリケーションの開発につなげることが、社会的な課題であるともいえる。

第2章　相互評価の事例調査

2.1 調査の目的・方法

本事業では、第 0 章で述べたとおり、PBL 型の教育プログラムを企図し、Android マーケットのようなモデルの上でプロジェクトの成果を競う仕組みの構築を考えている。この仕組みを教育・学習面で応用することの狙いは、学習の動機づけが期待できることはもとより、マーケットの参加者同士の「相互評価」によって自然に技術を研鑽するマインドを醸成することにある。このような動機づけと技術研鑽のマインドが醸成されるならば、第 1 章で確認された「集合研修」や「e ラーニング」などの慣れ親しんだ研修方法に、相互評価の仕組みを加えることによって、より学習効果の高い仕組みにすることを期待できる。本事業ではこのような考え方に基づき、学習の仕組みとして「相互評価」が行われているか、行われているとしたらどのような局面でどのように行われていて、どのような成果が上がっているか、あるいは、どのような課題があるか、などに関する事例調査を試みた。

その方法は、Web サイトの上から、「相互評価」をキーワードとしてエントリーし、それに「専門学校」「大学」などの学校種を表すキーワードを加えた絞込みを行い、わかりやすい事例を収集した。このとき、分野は IT 関係に限定しなかった。学習者同士が成果を公開し、相互に評価する仕組みの中でどのような心理状態になるか、そこで生まれる効果や課題をとらえることを目的とした場合、分野を限定せず幅広く事例を収集したほうが、参考となる事項を把握しやすいと考えたからである。

Web サイトを検索した結果、大学・短期大学・高等専門学校等における学習機会から 13 の事例を集めることができた。2.2 でその概要を表し、それに基づいた分析結果を 2.3 でまとめた。

2.2 調査の結果

1. 愛媛大学医学部　解剖学第一講座、*動物実験施設

代　表　者
小林直人、齋藤正一郎、脇坂浩之、樅木勝巳*、松田正司
内　容・方　法
「骨学実習」と「解剖学実習」では本年 1999 年度より、学生の合否判定の中に教官による採点以外に、学生相互による採点を加えた。その目的は以下のような従来の面接試験に対する反省からである。 １）教官４人で学生 100 人の実習の採点をするとして一人５分必要とすると５ｘ25＝125 分となる。２時間使用して学生の発表時間は５分であり効率が悪い。 ２）学生としては問題の当たりはずれがあり、また教官による採点基準が異なるなど不公平感が残る。 そこで学生を４人ずつの班に分け（注：これは解剖実習の班である)、以下のような採点基準と採点用紙により採点を行わせた。発表は班対班の試合形式のように班毎に移動し別の班と順番に発表し合う。２班８人のうち１人ずつ各問題について３分間、骨標本又は人体標本を用いて発表し合う。別の班の４人が採点する。
結　果・考　察など
１）学生は５段階評価で１を付けることは無かった。この点を学生に問い正すと、「同級生には良い点は付け易いが、悪い点は付けにくい」との回答であった。この点は無理も無いと思われる。しかし、学生による採点と同時に行った教官の採点とはほぼ同じような傾向を示し、教官の５段階を学生が４段階で付けた結果になった。 ２）発表の時間的な効率はさすがに良く、１２組のそれぞれで１人が発表するので、全員が発表するのに２４分しかかからない。２時間で全員４問発表することになり、発表の練習としては効率的であった。　医学部の学生は、将来医者となり患者さんと話すことが必要である。筆記試験では良い点が取れても面接試験では手が震えて十分な実力を出せない学生が毎年 10 名以上いる。特に真面目な学生に多い。そのような学生に面接試験によるストレスを与えることは解剖実習以外では少ない。教官による面接試験に比較すると少しストレスは少ないようであるが、人前で発表する機会の少ない学生にとっては複数の学生に４問発表して採点されるというのは結構良い訓練と思われた。 ３）採点結果は学生に公開し、本試験方法についての感想を求めたところ、不満はほとんど無かった。学生同士の好き嫌いの影響が出ることも予想されたが、点数は複数の採点の合計として加算されていくのであまり関係無いようで、不公平感は少ない。むしろ勉強の

足りなかった学生にとっては甘くしてもらったという感じのほうが強いのかもしれない。

４）１）に書いたような「同級生には厳しい点は付けにくい」点も含め、十分な勉強をしていない学生も採点に加わることは、全体的に採点を甘くすることになる。学生の採点を過剰評価してはいけないと思われるので頻回にある小テストを学生同士、節目の中テストと最終テストを教官で行った。最終合否判定には学生採点 40%、教官採点 50%、主席点 10%程度で判断した。

５）この方法による予想外の収穫は、実習中の学生同士のディスカッションが多くなってきたことであろう。学生同士が教え合い、批判し合って解剖実習を進めており、良い雰囲気だと思われる。

出典：http://www.m.ehime-u.ac.jp/school/anatomy1/csl02.html

2.　別府大学短期大学部

代　表　者
後藤善友
内　容・方　法
相互評価活動は、講義中に前回の講義内容に関する設問に４００字程度の回答をする記述問題において実施した（「(教科）理科」３０人)。各相互評価において各学生は他学生３～５人分の回答を１０段階で評価した。評価結果は集計され、教師による評価とともに学生に提示された。評価を開始する前の注意事項として、教師や他の学生と著しく異なる評価をした場合は説明を求めることがあること・場合により評価者の評価を下げることもあることを確認した。
結　果・考　察など
簡単な記述課題について学生相互評価を実施し、教師評価との関連について調査した。相互評価と教師評価の結果を学生にフィードバックすることで、学生相互評価の平均値は教師評価と高い相関を示すようになった（０．５２→０．８１)。 また学生評価と教師評価の順位相関は、フィードバックの有無に大きな影響を受けず、高い相関を示していた（０．７→０．８)。 フィードバックによる学生評価と教師評価の接近は、学生が「どの程度の採点基準で教師評価に近づくか」を意識することで採点基準の均質化が進んだためだと考えられる。一方、順位相関がフィードバックの有無に関わらず高い値を示しているのは、もともと学生の評

価特性が教師と近く、学生による課題の善し悪しの判断は教師と大差なく行われていることを意味している。
今後、学生に対するフィードバックと、順位付け情報を組み合わせ、相互評価活動の頻度を高めることで、相互評価の結果はさらに高い精度で教師評価に一致することが期待できる。

備　考

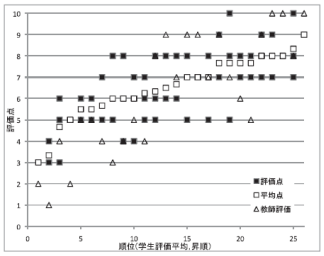


図3　フィードバックの有る場合の学生評価の分布と学生評価平均

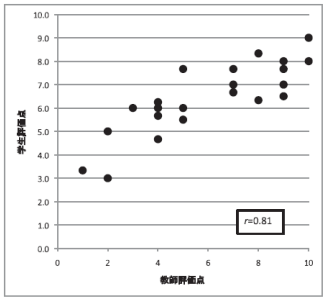


図4　フィードバックが有る場合の教師評価に対する学生評価平均の分布

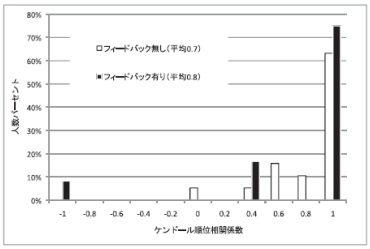


図5　フィードバックの有無による順位相関係数の分布変化

出典：http%3A//repo.beppu-u.ac.jp/modules/xoonips/download.php%3Ffile_id=3302

3.　愛知学院大学商学部青木ゼミの活動（学生相互評価によるゼミナール活性化）

代　表　者

名古屋マーケティング・インカレ

青木，秋本，大崎，太田，大塚，為広，濱

内　容・方　法

名古屋マーケティング・インカレの特色はいくつもありますが，その代表的なものに学生相互評価があげられます。通常，学生研究発表会における評価は教員やビジネスマンからなる審査員が行います。しかし，名古屋マーケティング・インカレでは，参加している学生チーム間で互いに評価して，優劣を決める方法を採っています。本大会では，ブロック予選において，相互評価得点に基づいてブロック代表の優秀チームを決定し，決勝において，参加全チームが得点をして最優秀賞を決定します。そして，コメントも同時につけます。今年度からは，中間発表会で相互に評価してコメントを送り合うことにしました。ただし，得点を与えるのは本大会のみです。

専門を同じくする研究者が相互に評価する方法はピアレビューといって，研究の世界では基本的な評価方法です。研究の世界では，専門化が進んでいて，専門家コミュニティーの外部にいる人が専門的研究を理解して評価することが難しくなっています。また，研究の世界では通説が破棄されて権威がひっくり返ることがままあります。専門家コミュニ

ティーの内部において，権威者が判定者になって研究の良し悪しを判定することは，通説とは違った研究を評価する際に困難さをもたらしがちです。そこで，専門を同じくする研究者が相互に評価する方法を用いるのです。学会などで日常的に行われています。

未熟な学生に研究者（それも上級者）と同じ流儀で相互評価をさせることにいろいろ問題が生じるのは承知しています。しかし，名古屋マーケティング・インカレでは，学生のモチベーション向上と評価眼の養成を目的として，あえて採用しています。モチベーションは評価されるとともに評価する立場に立つことで一層高まります。また，評価眼を持つことは，他人を評価するだけでなく，自分を評価することにも役立ちます。

今回，われわれは，学会発表のために，名古屋マーケティング・インカレを昨年度経験した学生と，今年度参加している学生全員を対象にヒヤリング調査を行いました。インカレに対する学生の本音を引き出して，大会運営・教育の改善につなげようと考えたのです。その結果を見て，われわれは相互評価の公正さに対する学生の認識に着目しました。

結　果・考　察など

昨年度経験者と今年度参加者に間に大きな認識の違いがありました。昨年度経験者のほとんどが相互評価に不公正さが存在していると考えているのですが，今年度参加者においては公正だという意見と不公正だという意見が拮抗しています。昨年度経験者は点数評価をし，優劣のつく競争を経験したので，不公正に過敏になっているのかもしれません。今年度参加者はチーム間でコメントを交換し合うことはしていますが，点数による評価を行っていないので，そのような「マイルド」な態度に落ち着いているのかもしれません。

昨年度経験者の公正さの認識に関する回答を検討してみると，興味深いことが分かりました。「自分たちが勝ちたいため，有望な他のチームの評価をわざと低くする」「同じ大学のチームには甘い評価をしてしまう」「中間発表会で仲が良くなったチームには厳しい評価ができない」というような評価の状況を不公正ととらえ，それが昨年度起きたのではないかという指摘が多くありました。これはわれわれも常に想定している不公正さです。一種の機会主義的行動といってもいいでしょう。しかし，つぎのような主旨の意見にわれわれは驚きました。

出典：http://blog.goo.ne.jp/aohito01/e/6280d35cc7d96d9040523341900bdc4e

4. 筑波大学　（口頭発表活動における学生同士の相互評価の役割）

代　表　者
高橋　純子
内　容・方　法
各発表、活動の後では質疑応答の時間を設けた。学生は、発表を聞きながら、B 5 用紙の半分くらいの用紙に自由記述で評価、感想を書いていった。評価項目は下記の通りである。 発表者の自己評価と感想、聴衆の学生評価に教師の評価、感想を加えフィードバックシートを作成し（資料参照)、翌週の授業の始めにクラス全体で話合いの時間を設けた。 【評価項目】 1）テーマ 2）構成（まず、次に、最後に、などの表現の使用の有無も含む) 3）内容 4）話し方（姿勢、目線、速さ、声の調子、間合い） 5）音声（発音、アクセント、イントネーション） 6）文法・語彙・表現 7）興味を引くための工夫（視覚資料の提示など） 8）全体的印象 9）その他気がついたこと 4．学生のコメント
結　果・考　察など
観察されたことをまとめると、 1）学生は他学生からもらう評価を肯定的に受け容れている。その評価はあまり厳しいものだとは感じていないようである。 2）聴衆の存在を意識し、興味を引きそうなテーマを選ぼうとしたり、より分かりやすい発表に取り組もうとしたりする動機付けになっている。 3）自分自身の行ったスピーチについては、失敗した点など意識しており、その時に何か肯定的なコメントをもらうことによって精神的に助けられていることがうかがえる。 4）あまり出来のよくなかったスピーチに対しては、学生は、相手を配慮した表現を用い、批判を緩和しようとしている。 学習者同士がどのように学ぶのかについて興味を持ち、学生の学び合う可能性とそれを引き出す工夫を探っている。相互学習が成功する条件としては、様々な要素が考えられる。学生の適性、教師の適性、目的意識の有無、自律的学習を助ける環境、雰囲気作りなどがあろう。授業の最適化を計るために、丁寧に事例を記述し、積み上げて行くことが肝要だ。 出典：http://www.tulips.tsukuba.ac.jp/limedio/dlam/M50/M509127/10.pdf

5. 愛知教育大学　プレゼンテーション演習における学生間相互評価の分析

代　表　者
竹田 尚彦　吉田 宏史　佐合 尚子
内　容・方　法（抄録のみ）
プレゼンテーション演習のようなコミュニケーション能力を高める授業では，情報の送り手が受け手を意識して発表し，それがどのように受け止められたかフィードバックする必要がある。少人数の授業であればフィードバック作業は容易だが，多人数では面倒である。そこで，我々の研究グループでは，評価点数と自由記述を入力できる相互評価システムを用いたプレゼンテーション演習を試みた。その結果，プレゼンテーション教育に効果的であることが分ったので報告する。 出典：http://ci.nii.ac.jp/naid/110002777169

6. 奈良工業高等専門学校　学生の相互評価を活用した短歌・俳句指導

代　表　者
鍵本有理
内　容・方　法
（1） 授業前日までの準備 学生から提出された課題について，まず全てについて氏名を隠す（終了後にはがせるよう，はがせるテープ，あるいは幅 12mm～25mm の付せんの類を利用）。 また，写真のあるもの（選択課題A），文字だけのもの（選択課題B）に分類し，黒板に掲示できるように 4～5 枚ずつ貼り合わせる。 （2） 掲示と閲覧時間の設定 授業開始前にマグネットバー等を利用して黒板に作品を貼っておく。その後，「1人3 票」として，気に入った作品を選んでもらうことを説明し，鑑賞時間を設ける。 ただし，写真のあるものとないもの，それぞれから最低 1 つ以上を選ぶという条件も付けた。さらに選択課題Bには五首ないし五句記載されているため，それについてはその中の一首あるいは一句だけを選んでもよいし，連作としてその人の作品をひとまとめとして評価してもよい，という旨を補足した。 （3） 相互評価 学生の鑑賞終了後，付せんに投票理由，つまり「どこがよかったか」を記し，また，最後に氏名を明記するよう指示する。そして先ほどの閲覧時と同様に，混雑しないように「投票」として、当該作品の余白に貼らせる。その際、得票数がわかりやすいよう、ずらして貼るようにする。その後，投票結果をふまえて教師が簡単にコメントを行う。
結　果・考　察など
選択課題としてではあるが，国語表現課題に写真を組み合わせるという方法により，教師のねらい通り，短歌・俳句に親しみ，また作品を通して学生同士のコミュニケーションも

深まるなど，学生にとっても印象深い授業を行うことができた。他の教員に作品を見せたところ，教員からも好評であった。
今回は短歌・俳句の選定・創作について自由度を持たせたが，現代俳句・短歌に限定する，あるいは古典の和歌・俳句に限定するなどすることで，教科書にある短歌・俳句の読解や古典の授業への応用も可能である。
また，複数の色の付せんを利用することにより，色分けで「最優秀作品」「佳作」等の区別を設けて学生に投票させる，あるいはディベートの板書に貼るなどして「賛成」「反対」の区別をわかりやすくする，といった方法に使うこともできる。
教師が細かいコメントを付さなくとも，同じクラスの学生のコメントをもらえることで，学生にとっても達成感を得ることができ，また優秀作品の選定などもたやすく行える，教師としても評価がしやすいという長所がある。

出典：
http://las.tut.ac.jp/~nakamori/engineer_japanese/pdf/h22koutoukyouiku_kagimoto.pdf

7. 久留米大学文学部　レポート相互評価法―大学における授業実践の試み

代　表　者
石橋潔
内　容・方　法
学生が書いたレポートの評価を学生自身の相互評価によって実施する．この相互評価法では，だれのレポートをだれが評価しているのか分からない形で相互評価を実施し，その評価を得点化する．このように匿名化することで 100 人規模の比較的大人数の授業でも実施できる．この方法を繰り返し実施することで，学生は評価者の視点に立ったレポート作成能力を学ぶことになる． ①課題設定 レポート相互評価法では，まず当然のことながら学生に対してレポート課題を提示しなくてはならない．レポート課題は，どのようなテーマ，内容でも構わない．ただし評価のための組み合わせを作るためには，あまりレポートのページ数が多いものはコピーして組み合わせを作るのが難しくなる．100 人程度の授業で相互評価を行う場合には，用紙 1 枚に収まるような分量に指定しておくほうがよいかもしれない． また学生に対して，相互評価法でレポートを評価すると伝えておく必要がある．筆者の授業では次のように学生に伝えている．「あなたのレポートを他の学生が読んで評価します．他の学生に“へぇ，なるほど”と思ってもらえるように書きなさい」． ②組み合わせ作成 提出されたレポートは，相互評価のために匿名化する．氏名などがレポート本文に記載さ

れている場合はその氏名などを消去し，その代わりレポート番号として連番を書き込む．そのうえで，学生に評価させるためのレポートをランダムに組み合わせて作成する．学生が時間内に評価可能なレポート数を設定し（おおよそ 5〜15 程度のレポート），そのランダムな組み合わせを作る．今までの経験では，1200 字程度のレポート 8 つを学生に評価させる場合，30 分ほどの時間を必要とするようである．レポート数を 8 つとした場合，100 人の授業での相互評価は，8 つのレポートのランダムな組み合わせを 100 個作ることになる．

③相互評価
組み合わせたレポートと評価用紙を学生に配布する．自分のレポートを自分で評価することがないように，配布したレポートの中に自分のレポートが入ってないことを確認させる．入っていた場合には，他の組み合わせと交換する．そのうえで学生にレポートを読ませ，そのレポートについての評価を数値で記入させる．持ち点 100 点を各レポートに割り振る方法などでもよいが，筆者の授業ではレポートに順位をつけさせている．8 つのレポートを相互評価させる場合，よいレポートだと思った順に 1 位から 8 位までの順位をつけさせている．

④評価の算出
③で実施した相互評価を表計算ソフトなどに入力し，レポートごとに平均得点を集計する．このとき，評価のばらつきの指標である標準偏差なども算出しておくとよい．

⑤評価のフィードバック
学生にレポートを評価点とともに返却する．筆者の授業では，④で算出した評価の平均順位を 100 点満点の得点に換算し，学生に返却している．またこのとき，評価のばらつきを示す標準偏差も記載して次のように学生に説明するとよいだろう．
「この標準偏差が大きい場合は，あなたのレポートをよいと評価した人と，よくないと評価した人がばらついていたことを示します．つまりあなたのレポートにはよい点もあったのでしょう．どのようなレポートをよいと評価するかは人によって，多少違いがあります．しかしできるだけ多くの人に，よいレポートだと思ってもらえるレポートを書くよう努力しましょう．」

結　果・考　察など

相互評価法では，平凡な内容のレポートではなく，へぇ，なるほどと思ってもらえるレポートが高く評価される．この点がこのレポート相互評価法を用いる場合の最大の利点である．このことを○×式の試験と比較すればこの利点は明確である．○×式のような教育評価は，学生に唯一の正解だけを学習する動機づけを与える．だがここからは，多様な発想やアイデアを生み出す動機づけを与えることは難しいだろう．相互評価法は，内容が興味深く，納得いくものを評価するという視点を学生に与える．そして，学生それぞれが，そうしたものを書きたいという動機づけを持つことができれば，レポート課題は独自の発想と説得力を競う場に変化するだろう．

1）相互評価法のメリット

学生は学生同士の仲間集団に強く帰属しているようである．教員がどんなに熱く学問的価値を語っても，学生集団内でその価値が共有されていなければ，教員の声は学生の耳には届かない．

このレポート相互評価法は，仲間に帰属する現代学生の性質を逆に利用する試みである．この方法は他の学生に自分がどのように見られているかを意識させる．教員の視線ではなく，同じ学生の視線を意識させることがこの方法の強みである．このことを通じて，よいレポートを書くという方向に学生の動機づけを誘導する．このようなレポート相互評価法がうまくいけば，レポートを書く上での学生の創意工夫を誘発する．人とは異なる意外な発想を書こうとする学生，資料を調べて書き込もうとする学生，大人びた専門用語を使って書こうとする学生．こうした学生の創意工夫は，高い評価を受ける場合もあるし，低い評価を受けることもある．しかし学生たちは，評価の結果を参考にしながら，次のレポートを書く際にさまざまな戦略をめぐらし，レポートを書く．つまり学生は評価者の視点を意識してレポートを書くようになる．この点が教育上の最大のメリットである。

またこのレポート相互評価法は，教員が学生の評価の視点を理解するのに役に立つ．つまり学生が何に影響され，よいものと悪いものを区別しているかを理解するのに役に立つ．学生の評価の視点は，必ずしも教員の評価の視点と同じではない．一般に大学の教員は自らの視点だけに立って，学生に対して一方的に知識や技能を伝達しようとしがちである．しかし教員が授業で伝えたと思っていることは，そのままの形で学生に消化されているわけではない．

このレポート相互評価法を実施すると，よいと評価されるレポートが教員の評価と一致しないことが生じる．このことを痛感するのは，教員が模範だと考える自分自身のレポートを相互評価に混ぜて学生に評価させた場合である．必ずしも教員が書いた模範レポートが高い評価とならずに，他の学生のレポートに得点で負けるケースが出てくる．教員としてはかなり悔しい思いをするが，この評価のズレこそ，授業の中で埋めていかなければならないズレである．

2）今後の課題

このレポート相互評価法はレポートの評価を計量化する．その点で学生の成績評価の重要な指標になるかもしれない．しかしこの点ではまだまだ検討していく問題も多い．特にこの点で重要なのはこの相互評価の信頼性と妥当性の検証である．

相互評価法を用いると，上位および下位レポートの評価結果はおおむね妥当だが，中位のレポートの評価結果は誤差が大きいという印象がある．おそらくこの問題は，ランダムに組みあわせたレポートを評価するという手法の持つサンプリング誤差の問題が一つはあるだろう．もう一つは，評価する学生が中位のレポートの良し悪しを区別できないということがあるかもしれない．つまり学生はいいレポートと悪いレポートを見分けることができるが，中位のレポートを区別することができないということがあるようだ．また学生は相互評価を繰り返し実施する中で，評価者の視点が育ち，定まっていく．とするとどのよう

な学習段階に到達した学生集団はどのようなテーマについて妥当な評価を下せるようになるかについても検証がいるだろう.
以上のようにレポート相互評価法には今後，検証していく課題も多い．しかしレポート相互評価法という方法は，学生に動機づけを与えて指導していくことが難しかったレポート指導という分野に，新たな教育上の手がかりを与える方法であることは間違いないだろうと思う.

出典：http://bungaku.kurume-u.ac.jp/img/other/information/2010/2010-2.pdf

8. 大阪市立大学　医学情報学総括発表会における学生相互評価システム

代　表　者
不明
内　容・方　法
「21 世紀における医学･歯学教育の改善方策について」の中の「教員の教育業績評価ガイドライン」で述べられているように教員の教育者としての意識啓発のための学生による授業実習等の評価システムを次年度より導入する予定にしている。しかし学生が注意を受けたとかの個人的な感情で評価を正しく行わない可能性も想定しうる。そこで教員評価システムを構築する前のプレシステムとして、学生が互いの発表を統一化されたプロトコルに沿って公平で客観的な偏りのない相互評価をおこなう訓練のための投票システムを構築したので報告する。 ● 方法 学生が自由に選択した課題について独自で情報を収集分析し、その結果について学会形式の発表をおこなう訓練をおこなっている。発表会場は通常の授業で使用する講義室で、その講義室には昨年度より無線 LAN ネットワークを設置し、学生には年度初めに一人 1 台ずつ、無線 LAN 接続が可能なノート PC を貸与してある。そこで個々の発表終了後、評価システム用サーバにアクセスし、イントラネット上の Web Page で投票することとした。さらに各自が受けた評価を同じように Web 上で確認できるシステムを組み込んだ。 ● 今回のシステムの概要について 個人情報を扱うことを考慮して、SQL server は権限のある管理者以外はアクセスできないように設定し、Web Page での集計結果も、本人もしくは権限のある人間以外は見られないように設定した。権限の設定の問題は個人にそれぞれ固有のパスワードを発行することにより解決した。 まず、ログイン画面でユーザ名とパスワードを入力する。ユーザ名とパスワードがデータベースに登録されているものと一致しない場合ログインに失敗する。ユーザ名とパスワードが一致した場合には、投票ページ、もしくは集計表示ページを閲覧することができるようになる。投票ページにおいて、ユーザがデータを入力すると、データベースに登録され、

自動的に集計される。なお、その際に投票可能なページは時間などの要因により制限されている。また、二重投稿は受け付けない設定にしてある。

● 評価項目

一学年約 80 人が 36 演題に対して評価を行った。質問項目としては、
ａ.発表は興味深く聞けたか
ｂ.発表はよく準備されていたか
ｃ.先に提出したアブストラクトに沿った発表であるか
ｄ.明確で聞き取りやすい話し方であるか
ｅ.パワーポイントは理解しやすかったか
ｆ.発表内容に対する意欲が刺激されたか
ｇ.質疑応答は適切だったか
ｈ.コメント
を用意した。コメント以外の 7 項目について 5 段階で評価してもらい、集計用のサーバによりリアルタイムで集計をとった。コメントは自由記述欄を用意しており、投票者は匿名で収集できるように考慮した。

● 結果画面

結果画面のページでは、自分に対する 7 つの項目の評価の平均値と、それまでに投票された学生全員の平均点を見ることが出来、自分の発表に対するコメントはすべて表示させるようにした。

結　果・考　察など

今回の発表会では、計 2512 件の投票があった。投票は、午前 9 時から午後 4 時まで間断なく、続けられていた。この際、サーバが不安定になることはなく、また、過負荷がおこらなかったことは確認されている。データベースサイズは、1.6M であった。なお、システムフリーズやデットロックの発生などの問題点はまったく起こらなかった。これは、予定されていたよりもサーバに対する負荷が少ないことを示しており、この形式で作成したアンケートシステムであれば、利用人数が増えても、十分運用が可能であると考えられる。今回は、一覧表の形式でも集計を出力し、紙媒体でも結果が参照できるようにした。なお、投票の際には全ての情報を記録しているため、特定の情報のみを含むものも表示でき、それを利用して自分の発表に対する評価の平均点と全体の平均点を表示する結果表示ページを用意して結果を学生にフィードバックした。この学生相互評価システムは、学生に対する啓蒙の他に、来年以降の教員評価システムの実地を見越して、システムにかかる負荷、記録ファイルの容量、同時アクセスにより起こる問題点、不正アクセスの防止などのチェックの意義も含んでいる。来年以降予定されている教員評価システムでは、ユーザの権限により表示できる項目を可変する予定である。

出典：http%3A//www.med.osaka-cu.ac.jp/informatics/m14.htm

9.　東京経済大学　教師による評価と学生による相互評価の実践(英語教育)

代　表　者
中村 優治
内　容・方　法（抄録のみ）
本稿はオーラルプレゼンテーションクラスの評価方法として,教員による評価と学生による相互評価の両方を取り入れ,ラッシュモデルの FACET 理論を適用して項目応答理論によるデータ分析を行い,より科学的で実践的な評価方法の提言を試みたものである.結論として次の 3 点が検証された.1)学生相互評価は勉学への動機付けとなりうる.2)学生はある一定の信頼性を備えた評価者となりうる.3)ラッシュモデル(FACET 理論)により評価者の評価特性,学生の能力の特徴,および評価項目の適切さが明確に示され,プレゼンテーション能力の科学的な説明が可能となる. 出典：http%3A//ci.nii.ac.jp/naid/110007054235

10.　名古屋女子大学　学生の評価能力に関する考察

代　表　者
白井靖敏
内　容・方　法
「教育支援技法」（家政学部・教職３年）と「教育の方法と技術」（文学部・児童教育学科・幼児保育専攻４年）において、前者は学生の模擬授業、後者はＩＴを活用した教育方法のうち電子紙芝居の作品と実演についての学習成果の評価を次の方法で行った。 家政学部の教職科目「教育支援技法」における模擬授業では、指導案を含むパワーポイントプレゼンテーション教材を作成するとともに、それを使った１人 20 分程度の模擬授業を、電子教材の分かりやすさ、内容、構成、提示の仕方、話し方の観点別に６段階評価シート（５：すごくよい、４：よい、３：ややよい、２：ややよくない、１：よくない、０：まったくよくない）を配布して学生一人一人が自己評価および他者全員を評価した。 文学部の「教育の方法と技術」では、パワーポイントを使った動きのある電子紙芝居の作成と実演を行い、画面構成、画面の工夫、見やすさ、分かりやすさ、話し方の観点別に６段階評価シート（前述と同様）を配布して学生一人一人が自己評価および他者全員を評価した。
結　果・考　察など
家政学部の「教育支援技法」と文学部児童教育学科幼児保育専攻の「教育の方法と技術」の２つのクラスだけの分析であることと、サンプル数が少ないので統計的に意味のある分析の信頼性がやや乏しくなるので、一般解を求めることはできないが、対象クラスのおおまかな傾向は見て取れる。自分に「からく」、他者へは「あまい」評価傾向のある学生が２クラスとも７割を越えていることから、対象としたクラスの学生は自分に厳しい傾向があ

るといえるだろう。
対象学生の評価能力の点でみると、自分に「からい」傾向のある学生は、他者に「あまい」という、ひとつの評価基準を自分なりに持っていると考えられるので、他者への評価基準をやや「からく」しながら自己評価との調整訓練により比較的短期間で客観的な評価能力が身に付くと考えられる。
自己評価と他者から受けた評価がほぼ一致している学生は約１割と少ないが、客観的な評価がある程度できるものと見てよいだろう。ただし、他者への評価にばらつきがあるため、他者への評価訓練を積むことによって安定した評価ができていくと考えられる。
しかし、自分に「あまい」学生は、他者への評価のばらつきが大きく、他者への安定した評価ができていないため、自己評価や他者への評価訓練をかなり積まないと、客観的な評価ができる評価能力が身に付かないだろう。
一方、自己評価と他者から受ける評価との間にはほとんど相関がない。自分に「からい」場合と「あまい」場合によって、他者をみる評価基準が異なるので、一般的に見て他者評価の信頼性は乏しい。そのため、他者による評価は統計的に意味をもつ複数者によってなされ、その平均を採用するのが望ましい。あるいは、評価基準にゆらぎの少ない訓練された評価者でなければならないだろう。特に、学生同士による他者評価については変動が大きく、すぐにそのまま一般的な指標とはなりにくいことが分かった。
たとえば、学生による授業評価について、自己の学習についての自己評価と他者、いわゆる授業者への評価との関係で「あまい」のか「からい」のかによって、学生の評価基準が推し測れるとともに、評価能力もある程度推定できると考えられる。より信頼のある評価をおこなうためには、自己評価と他者への評価の差違を考慮した補正が必要となるだろう。また、教員相互による授業評価についても同様に、自己評価と他者への評価について分析し、かつ安定した評価基準をもつ訓練された評価者によって評価がなされることが必要ではないだろうか。

出典：http%3A//libweb.nagoya-wu.ac.jp/kiyo/kiyo52/jinbun/kojin/jinbun127-132.PDF

備　考

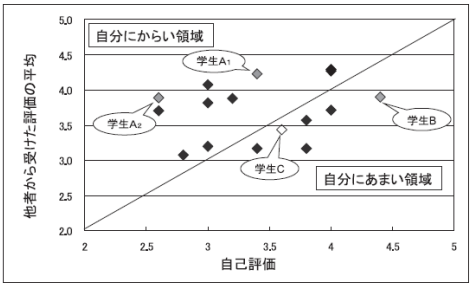
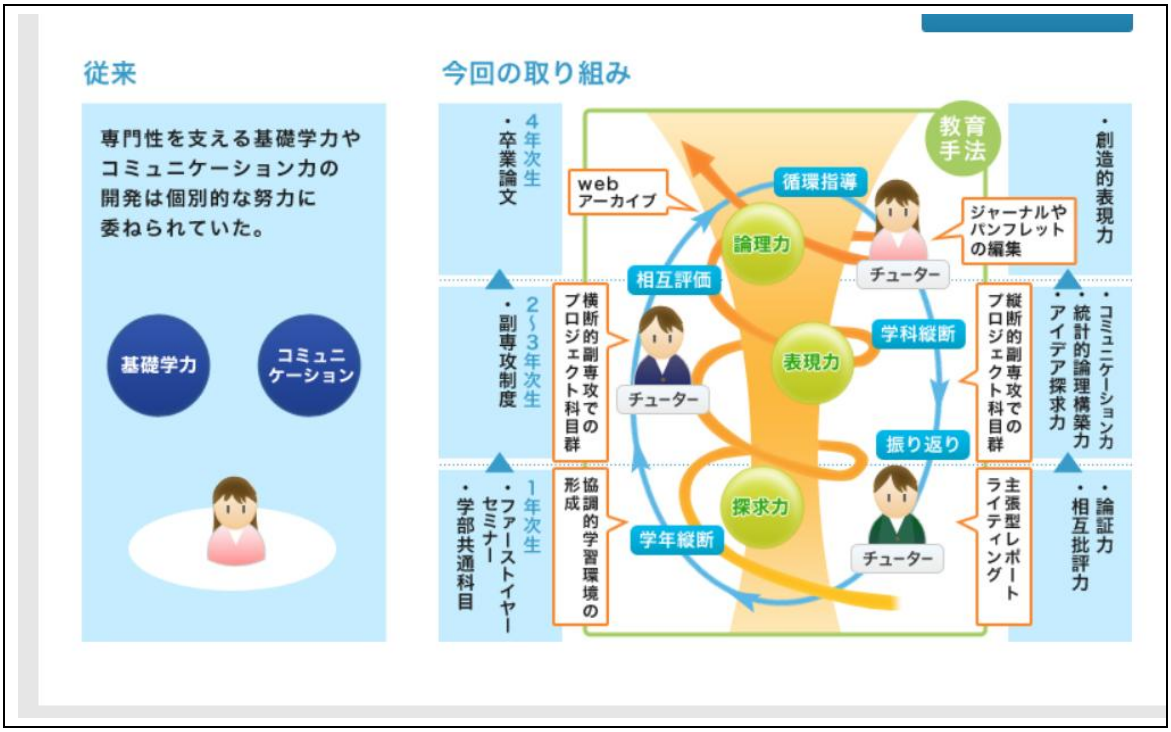


図１　自己評価と他者から受けた評価との関係（ｎ＝15）
線より上の領域は自分にからく（厳しく）、線より下の領域は自分にあまい。

11. 同志社大学社会学部　相互啓発による創造的学力育成カリキュラム

代　表　者
不明
内　容・方　法
高度な学力を育成するためには、学生が相互に啓発しあい成長できる環境が重要です。この取組では、同じ学部の先輩が後輩をクラスでサポートするチューター制を導入します。気軽に質問し相談できる先輩がいることで教育効果を高めます。また、チューターに選ばれた学生諸君にも、後輩にアドバイスする立場になることで自分の学力を深め、あるいはクラスをまとめるリーダシップやコミュニケーション力を身につけ、人間として成長してくれることを期待しています。これは広い意味でのキャリア教育となることも目指しています。 教育を効果的にするために、教員から学生への成績評価、および学生から教員への授業評価だけでなく、学生間の相互評価の導入に前向きに取り組んでいきたいと思います。自分の発表等に対する同級生や先輩後輩からのコメントは、時には教師による評価よりも大きな意味をもつでしょう。また、学生諸君はお互いに意見しあうことで、他人への言葉づかいや態度がいかに難しいものであるか、あらためて学ぶことでしょう。「他者への配慮」という人間教育を何より大切にしたいと思います。 私たちは、今後の様々な教育実践のためにできるだけ多くの記録やデータを残し、発表していきたいと思っております。各種のアンケート調査やチューターによる実践記録も、このＧＰホームページに逐次公開してまいります。 出典：http://www.doshisha.ac.jp/academics/activity/daigaku/index.html
備　考


従来
専門性を支える基礎学力やコミュニケーション力の開発は個別的な努力に委ねられていた。
基礎学力
コミュニケーション
今回の取り組み
4年次生
・卒業論文
2〜3年次生
・副専攻制度
横断的副専攻でのプロジェクト科目群
1年次生
・ファーストイヤーセミナー
・学部共通科目
協調的学習環境の形成
教育手法
循環指導
webアーカイブ
論理力
相互評価
チューター
ジャーナルやパンフレットの編集
学科縦断
表現力
チューター
縦断的副専攻でのプロジェクト科目群
振り返り
探求力
学年縦断
チューター
主張型レポートライティング
・創造的表現力
・コミュニケーション力
・統計的論理構築力
・アイデア探求力
・論証力
・相互批評力

12.仙台高等専門学校　Blackboard による e-learning と教室授業の融合

代　表　者
不明
内　容・方　法
仙台高専の保健授業は、学校のカリキュラム編成上、同年代の高校生と比べると４分の１程度しか授業時間を確保できていません。そこで、短い時間で学習効果をあげるための手だてとしてグループで学生が主体的に学習する課題解決型学習 Problem Based Learning（PBL）を取り入れました。この PBL 型授業に Blackboard を活用してみようと思いました。 グループ学習の進め方は、まず 1 クラス約 35 人を 8 つのグループ（1 グループ 4～5 名）に分け、課題を与えます。学生達は課題に沿って協力し、プレゼンテーション資料とレポートを作成します。 この作成過程で、Blackboard のディスカッション機能を用いた BBS 討論が学生同士や学生と教員間で行われます。　討論を行うことにより、資料のやり取りや確認をしながら協力して編集作業を進めることが可能になりました。Blackboard という ICT ツールを活用することでグループ内での相談が容易に行うことができるようになり、教員とのコミュニケーションもとても効率が良くなったと感じています。 プレゼンテーション資料が完成するまで教員による添削指導が継続されます。このディスカッションは PBL の授業時間外でも行われ、Blackboard を効果的に活用しています。 完成したプレゼンテーション資料を持ち寄り、授業内で学生に発表させ、Blackboard のアセスメント機能を使い相互評価をさせます。発表時間 15 分、質疑応答時間を 5 分として、毎回の授業で 2 グループずつ発表させます。 　このとき司会担当の学生は、発表のタイムキーピングの時間や質疑応答の内容を発表報告として、指定のフォーマットで Blackboard 上にアップします。これにより、学生は発表内容を集中して聞くことができるため、学習効果が高まるのではないかと考えています。
結　果・考　察など
プレゼンテーション後すぐに Blackboard のアセスメント機能を用いて、学生間での相互評価を行わせています。全員が評価を終えるとすぐにその結果が得点分布としてヒストグラムで表示されます。学生は自分達のグループ発表がクラスのほかの学生にどのように評価されたかを即座に知ることができるため、良い意味での刺激となります。これは自分の取り組み自体を改善しようとするモチベーションにもつながります。この点も、教育効果が期待できる学習活動だと考えています。 また、教員はグレードブック画面から一人ひとりの評価活動を確認することができます。指導と評価の一体化が図れる点も教育効果の向上に寄与できる点として注目できるのではないでしょうか？ これまで遠隔授業と考えていた e-learning ですが、Blackboard を活用することにより、教

室授業の中に e-learning を取り込んでいくというこれまでになかった授業形態が生み出され、学生主体の新しい授業スタイルが可能になったといえます。
Blackboard は教育効果の面のみならず、研究対象としても魅力があると感じます。

出典：http%3A//csklc.jp/case/sendainct.html

13. 東京女子大学
情報教育における学生のパフォーマンスに対する相互評価の試み

代 表 者
廣瀬　英子
内 容・方 法
東京都内の私立女子大学において，2～4 年生を対象とした「情報科学論」（通年科目）の授業の一部を使い，2002 年 11 月から 2003 年 1 月にかけて行った．この期間の授業の目的は，効果的な情報伝達の方法と技術を習得することであった．受講生は具体的な実習課題として，Microsoft 社のプレゼンテーションソフト PowerPoint を使い，効果的な発表スライドを作品として作ることに取り組んだ．受講生は 24 名であった． 指定期日までに 21 名の受講生が作品を提出した．提出後，受講生には，今度は評定者となって全作品の評価を行うことを課した．どの作品が誰によって作られたものかはわからないように，作成者を特定できるような情報（受講生の名前など）をすべて削除した．そして，g1 から g24 までの作品番号を振り，1 枚の CD-ROM に収めた．その CD-ROM と評価シートを各受講生に 1 枚ずつ配布し，評価シートに評価を記入させた． 評価にあたり，2.1.3.節の中に示されている作品づくりの条件を再度受講生に提示し，それらの条件に合っているかを評価基準として評価するように指示した．また，作品と作成者とを切り離し，作品そのものを客観的に評価すること，自分自身が作成した作品も，他の作品と同様に客観的に判断することを強調した． さらに，次のようなルールを設けた． ● 21 種類の作品について，相対的に良く出来ているほうから 5，4，3，2，1 の 5 段階で評定する． ● 3 以外の評定値，つまり，1，2，4，5 をつけた作品には，その理由を簡潔に文章で記す． ● 評定者が 5 や 1 にかたよった評定値をつけないように，5 が 2 作品，4 が 5 作品，3 が 7 作品，2 が 5 作品，そして 1 が 2 作品という配分を遵守する．この配分の比率は，5 段階評定の場合の原則（評定値の 1 と 5 に各 7%，2 と 4 に各 24%，3 に 38%）に沿うように決めた．

結　果・考　察など
本研究では，24 名の評定者は作品づくりを通して評価基準を熟知していたものと考えられる．結果として、ある程度評価基準をそろえたつもりでも，複数の評定者間の評価は必ずしも一致しなかったが，その一方で，もっとも一致した評定値から 2 段階以上外れる評定者はわずかであることも示された．このことから，一人の評定者による評価を絶対的に正しいと見なすよりも，複数の評定者の評定を総合する方が，全体として妥当な評価を行うことになるのではないかと考えられる． 大学では，ひとつの授業を複数の教員で担当したり評価したりすることは，稀である．今後，受講者による評価も活用すると良いのではないだろうか. 出典：http://www.ciec.or.jp/event/2003/papers/pdf/E00096.pdf

2.3 調査結果の分析

収集した事例について、主に次の 4 つの観点から、まず整理を行った。

・仕組み

相互評価の仕組みがどのようなものか

・従来・キッカケ

相互評価を行うまではどのように行っていて、どのようなキッカケで相互評価の仕組みを採り入れることになったかなど

・効果

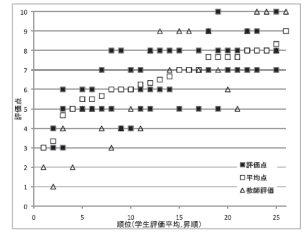
相互評価の仕組みを採り入れたことによる効果など

・課題

相互評価を行った結果としてどのような課題が生じたかなど

次のページから、整理の結果を表の形で表す。なお、整理のポイントとなる記述について下線を示している。

●学生(受講者・生徒)の相互評価　まとめ

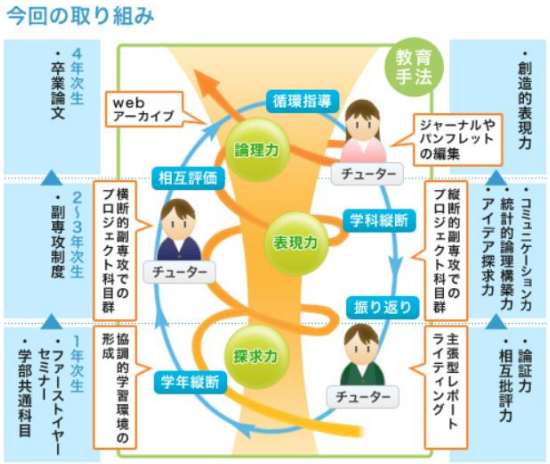
No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
1	大学	骨学実習、解剖学実習科目の合否判定	4 人の班。班対班。2 班 8 人のうち 1 人ずつ各問題について 3 分間、骨標本又は人体標本を用いて発表し合う。別の班の 4 人が採点。	従来の面接試験の反省。教官 4 名で 100 人の学生を試験＝1 人 5 分しかなく効率悪い。	効率の大幅な改善。コミュニケーションスキルの向上。自発的ディスカッション機会増加。	同級生には良い点は付け易いが、悪い点は付けにくい。しかし、相対的な位置関係は変わらない。	採点結果は学生に公開。感想を求めたが、不満ほとんど無し。学生同士の好き嫌いの影響が出なかった。学生の評価がすべてでなく、教官の評価と並立。両者の相関高。
2	短期大学	講義内容に関する記述問題の採点	各学生は他学生３～５人分の回答を１０段階で評価した。評価結果は集計され、教師による評価とともに学生に提示。	学生相互評価において、学生の評価基準や評価特性を教師の評価に近づける方法の検討 どれくらい評価に変化が生じるのか、またどのくらい教師の評価と一致するのかの検証。	相互評価と教師評価の結果を学生にフィードバックすることで、学生相互評価の平均値は教師評価と高い相関を示すようになった（0.52→0.81）。	今後、学生に対するフィードバックと、順位付け情報を組み合わせ、相互評価活動の頻度を高めることで、相互評価の結果はさらに高い精度で教師評価に一致することが期待できる。	 図3　フィードバックの有る場合の学生評価の分布と学生評価平均
3	大学	学生研究発表における学生相互評価	ブロック予選において、評価得点に基づいてブロック代表の優秀チームを決定し、決勝において、参加全チームが得点をして最優秀賞を決定し、コメントも同時につける。	研究の世界では、専門化が進んでいて、専門家コミュニティーの外部にいる人が専門的研究を理解して評価することが困難。 権威者が判定者になって研究の良し悪しを判定することは、困難さをもたらすので、専門を同じくする研究者が相互に評価する方法を採用。	未熟な学生に研究者と同じ流儀で相互評価をさせることの問題は承知しているが、学生のモチベーション向上と評価眼の養成を目的として、あえて採用。モチベーションは評価されるとともに評価する立場に立つことで一層高まり、評価眼を持つことは、他人を評価するだけでなく、自分を評価することにも有効。	「自分たちが勝ちたいため、有望な他のチームの評価をわざと低くする」「同じ大学のチームには甘い評価をしてしまう」「中間発表会で仲が良くなったチームには厳しい評価ができない」というような評価の不公平感。	アプローチや視点の違うチーム同士の評価では、低い評価が生まれてしまう例があったとしても、それは数ある評価の 1 つとして捉えればよい。同じ評価基準を用いても評価者によって捉え方や重視点が違から評価にぶれが出ることは当然。このような現象は社会のあらゆるところで散見され、それらを不公正と考えることには戸惑いを覚える。

No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
4	大学	口頭発表活動における学生同士の相互評価	各発表、活動の後で質疑応答の時間を設け、学生は発表を聞きながら B5 用紙の半分くらいの用紙に評価、感想を自由記述。 発表者の自己評価と感想、聴衆の学生評価に教師の評価、感想を加えフィードバックシートを作成し、翌週の授業の始めにクラス全体で話合いの時間を設けた。	学習者同士がどのように学ぶのかについて興味を持ち、学生の学び合う可能性とそれを引き出す工夫を探っている。	学生は他学生からもらう評価を肯定的に受け容れ、その評価はあまり厳しいものだとは感じていない。聴衆の存在を意識し、興味を引きそうなテーマを選ぼうとしたり、より分かりやすい発表に取り組もうとする動機付けになった。 自分自身の行ったスピーチについては、失敗した点など意識しており、その時に何か肯定的なコメントをもらうことによって精神的に助けられていることが伺える。 あまり出来のよくなかったスピーチに対しては、学生は、相手を配慮した表現を用いて、批判を緩和。	相互学習が成功する条件としては、様々な要素が考えられる。学生の適性、教師の適性、目的意識の有無、自律的学習を助ける環境、雰囲気作りなどがあろう。授業の最適化を計るために、丁寧に事例を記述し、積み上げて行くことが肝要。	
5	大学	プレゼンテーション授業などのコミュニケーション能力を高める授業での相互評価。	評価点数と自由記述を入力できる相互評価システムを用いたプレゼンテーション演習	情報の送り手が受け手を意識して発表し、 それがどのように受け止められたかフィードバックする必要があり、少人数の授業であればフィードバック作業は容易だが、 多人数では面倒。	プレゼンテーション教育に効果的であることが分った。		

No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
6	高等専門学校	国語の提出課題を相互評価	課題提出者の氏名を隠し、掲示。 学生は気に入った作品を選び、良かった点を記述しさらに氏名を明記して提出、採点。	高専の低学年を対象とした国語の授業において、学生に興味を持たせるのは容易ではないので、そこで、短歌・俳句を題材として、学生に作品を作らせるだけでなく、学生同士で作品評価を行わせ、作品創作についても「写真」を取り入れることにより、学生に「表現すること」の楽しさを実感させることが可能ではないかと考えた。	教師が細かいコメントを付さなくとも、同じクラスの学生のコメントをもらえることで、学生にとっても達成感を得ることができ、また優秀作品の選定などもたやすく行える、教師としても評価がしやすいという長所がある。		今回は短歌・俳句の選定・創作について自由度を持たせたが、現代俳句・短歌に限定する、あるいは古典の和歌・俳句に限定するなどすることで、教科書にある短歌・俳句の読解や古典の授業への応用も可能である。
7	大学	学生の書いたレポートを相互評価	レポートの氏名を匿名化し、レポート番号を付ける。1学生が時間内に評価できるレポート数を設定し、ランダムに組合せ、自身のレポートが入らないようにする。 1レポートを複数人が評価できるようにする。 100人分のレポートがあった場合、100人がそれぞれ8レポートぐらいを評価する。 評価は1位から順に順位をつける。 その後、すべてのレポートに対して、平均点（100点満点に換算）と標準偏差を算出。	教員がどんなに熱く学問的価値を語っても、学生集団内でその価値が共有されていなければ、教員の声は学生の耳には届かない。相互評価法は、仲間に帰属する現代学生の性質を逆に利用する試みである。	相互評価法は、内容が興味深く、納得いくものを評価するという視点を学生に与える。学生それぞれによい動機づけを持つことができれば、レポート課題は独自の発想と説得力を競う場に変化するだろう。	この相互評価の信頼性と妥当性の検証である。相互評価法を用いると、上位および下位レポートの評価結果はおおむね妥当だが、中位のレポートの評価結果は誤差が大きいという印象。 相互評価を繰り返し実施する中で、評価者の視点が育ち、定まっていく。とするとどのような学習段階に到達した学生集団はどのようなテーマについて妥当な評価を下せるようになるかについても検証がいるだろう。	学生の評価の視点は、必ずしも教員の評価の視点と同じではない。 教員が模範だと考える自分自身のレポートを相互評価に混ぜて学生に評価させた場合である。必ずしも教員が書いた模範レポートが高い評価とならずに、他の学生のレポートに得点で負けるケースが出てくる。教員としてはかなり悔しい思いをするが、この評価のズレこそ、授業の中で埋めていかなければならないズレである。

No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
8	大学	医学情報学統括発表における学生相互評価	学生が互いの発表を統一化されたプロトコルに沿って公平で客観的な偏りのない相互評価をおこなう訓練のための投票システムを利用。個々の発表終了後、評価システム用サーバにアクセスし、イントラネット上の Web Page で投票することとした。さらに各自が受けた評価を同じように Web 上で確認できるシステムを組み込んだ。	「21 世紀における医学・歯学教育の改善方策について」の中の「教員の教育業績評価ガイドライン」で述べられているように教員の教育者としての意識啓発のための学生による授業実習等の評価システムを次年度より導入する予定にしている。しかし学生が注意を受けたとかの個人的な感情で評価を正しく行わない可能性も想定しうる。そこで教員評価システムを構築する前のプレシステムとして、相互評価をおこなう訓練のための投票システムを構築。	今回の発表会では、計 2512 件の投票が午前 9 時から午後 4 時まで間断なく、続けられていたが、サーバが不安定になることも、過負荷がなかった。 利用人数が増えても、十分運用が可能であると考えられる。今回は、一覧表の集計出力や紙媒体での参照、特定の情報のみを含むものも表示でき、それを利用して自分の発表に対する評価の平均点と全体の平均点などの結果を学生にフィードバック可能。	この学生相互評価システムは、学生に対する啓蒙の他に、来年以降の教員評価システムの実地を見越して、システムにかかる負荷、記録ファイルの容量、同時アクセスにより起こる問題点、不正アクセスの防止などのチェックの意義も含んでいる。来年以降予定されている教員評価システムでは、ユーザの権限により表示できる項目を可変する予定。	
9	大学	オーラルプレゼンテーションクラスの相互評価	評価方法として、教員による評価と学生による相互評価の両方を取り入れ、ラッシュモデルの FACET 理論を適用して項目応答理論によるデータ分析を行い、より科学的で実践的な評価方法の提言を試みたものである。		1)学生相互評価は勉学への動機付けとなりうる。 2)学生はある一定の信頼性を備えた評価者となりうる。 3)ラッシュモデル(FACET 理論)により評価者の評価特性、学生の能力の特徴、および評価項目の適切さが明確に示され、プレゼンテーション能力の科学的な説明が可能となる。		

No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
10	大学	プレゼンテーション教材と20分程度の模擬授業の相互評価	模擬授業を、電子教材の分かりやすさ、内容、構成、提示の仕方、話し方の観点別に6段階評価シート（5：すごくよい、4：よい、3：ややよい、2：ややよくない、1：よくない、0：まったくよくない）を配布して学生一人一人が自己評価および他者全員を評価した。 パワーポイントを使った動きのある電子紙芝居の作成と実演を行い、画面構成、画面の工夫、見やすさ、分かりやすさ、話し方の観点別に6段階評価シート（前述と同様）を配布して学生一人一人が自己評価および他者全員を評価。			1つのクラス内でみたとき、自分に「からく」他者へは「あまい」傾向は74%、自分に「あまく」、他者へ「からい」は13%、ほぼ一致しているは、13%であった。 他方のクラス内では、自分に「からく」、他者へは「あまい」傾向は75%自分に「あまく」他者へ「からい」学生は23%、ほぼ一致しているは2%であった。 また、相関はほとんどみられないことから、客観的で正しい評価ができるような評価能力を育成する必要。	

No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
11	大学	同じ学部の先教育を効果的にするために、教員から学生への成績評価、および学生から教員への授業評価だけでなく、学生間の相互評価の導入		高度な学力を育成するためには、学生が相互に啓発しあい成長できる環境が重要。	先輩が後輩をクラスでサポートするチューター制を導入します。気軽に質問し相談できる先輩がいることで教育効果を高め、チューターに選ばれた学生諸君にも、後輩にアドバイスする立場になることで自分の学力を深め、あるいはクラスをまとめるリーダシップやコミュニケーション力を身につけ、人間として成長してくれることを期待。 これは広い意味でのキャリア教育となることも目指している。	自分の発表等に対する同級生や先輩後輩からのコメントは、時には教師による評価よりも大きな意味をもつ。また、学生諸君はお互いに意見しあうことで、他人への言葉づかいや態度がいかに難しいものであるか、あらためて学ぶことができる。「他者への配慮」という人間教育を何より大切にしたい。	

No	学校種	評価の対象	仕組み	従来・キッカケ	効果	課題	備考
12	高等専門学校	ICT ツールを利用したグループ作成課題の相互評価	課題解決型学習 Problem Based Learning（PBL）を取り入れ、授業に活用した。 1 グループ 4～5 名に分け、課プレゼンテーション資料とレポートを作成、その作成過程で、ディスカッション機能を用いた BBS 討論が学生同士や学生と教員間で行われ、プレゼンテーション資料が完成するまで教員による添削指導が継続される。このディスカッションは PBL の授業時間外でも行われ、完成した資料を持ち寄り、授業内で学生に発表させ、相互評価をさせる。	高等専門学校の保健授業は、学校のカリキュラム編成上、同年代の高校生と比べると 4 分の 1 程度しか授業時間を確保できていないので、短時間で効果的な PBL 型授業を取り入れた。	協力して編集作業を進めることが可能。 グループ内での相談が容易で、教員とのコミュニケーションも効率良く行える。 評価を終えるとすぐにその結果がヒストグラムで表示され、学生は、どのように評価されたかを即座に知ることで、教育効果を期待。 教員はそれらを確認することで、指導と評価の一体化が図れ、教育効果の向上に期待。	これまで遠隔授業と考えていた e-learning だが、教室授業の中に取り込んでいくことで、新たな授業形態が生み出され、学生主体の新しい授業スタイルが可能になったのでないか。 今後は、教育効果の面のみならず、研究対象としても魅力があると感じる。	
13	大学	授業の課題、プレゼンテーション資料を相互評価	提出されたどの作品からも作成者を特定できるような情報をすべて削除し、番号を振り、CD-ROM にして、評価シートを各受講生に配布し、評価を記入させた。 作品そのものを客観的に評価すること、自分自身が作成した作品も、他の作品と同様に客観的に判断することを強調。 5 段階評価で 3 以外の評定値、つまり、1、2、4、5 をつけた作品には、その理由を簡潔に文章で記す。 5 段階評定の場合の原則（評定値の 1 と 5 に各 7%、2 と 4 に各 24%、3 に 38%）に沿うように決めた。	大学では、ひとつの授業を複数の教員で担当したり評価したりすることは、稀である。 主観的な評価ばかりではなく、受講者による評価も活用すると良いのではないだろうか。	評定者は同時に作成者であり、評価基準を熟知していたものと考えられる。 もっとも一致した評定値から 2 段階以上外れる評定者はわずかであることも示された。 このことから、一人の評定者による評価を絶対的に正しいと見なすよりも、複数の評定者の評定を総合する方が、全体として妥当な評価を行うことになるのではないかと考えられる。	情報教育では他者に伝えたい情報を効果的に表現する力の育成がひとつの目的となるが、その学生のパフォーマンスを評価する際に、評定者の主観的判断が入りやすい。評価の客観性を高めるためには明確な評価基準を設定することが大切だが、さらに、複数の評定者による評価を活用すれば、評価の客観性が高まり、妥当な結果を得ることができるのではないか。	

整理の結果を各項目についてまとめると、次のようになる。

●仕組みについて

各事例とも、具体的な方法の違いはあるが、学生が他の学生の成果に対して何らかの評価を行う点で一致している。

●従来・キッカケについて

相互評価を行うことになった要因としては、大きく分けて 3 つあった。

・学生の動機づけを狙う

相互評価の実施が、「同じ立場にある他者から評価を受ける」ことによって学習者の動機づけになることを狙い、そのことが結果として学習者の学力やプレゼンテーション能力、コミュニケーション能力を高めることにつながることを期待するものである。

・効率性を高める

ここでいう「効率性」は、主として教員にとっての時間的な効率性である。すなわり、被教育者が多数の場合に、一人の教員の評価にかける工数が多大になる結果、評価の品質が低下することを避けるために、学生をグループ化するなどの方法と共に相互評価の仕組みを採り入れようとするものである。収集した事例では、この場合に、必ず、教員の評価と学生による評価の相関の有無が課題として上がっていた。

なお、一例であるが、カリキュラムの制約による絶対的な時間不足を理由に「相互評価」を採り入れるというものもあった。

・教員の教育者としての意識啓発

これも一例のみであるが、将来実施が予定されている「学生による授業評価」のプレシステムとして、学生による相互評価の仕組みを構築して実施したというケースがあった。

●効果について

相互評価を実施した効果についてまとめると、次のようになる。

・モチベーションが高まった

これは狙いどおりの成果・効果といえよう。多くの事例で、学習者の意欲が喚起されたことを示していた。

・効率が高まった

これも「効率を高める」狙いに即した結果が得られたことが、いくつかの事例で語られていた。この場合、「教員の評価と学生による相互評価の結果は高い相関を示した」とか、「中位の学生についてはぶれがあるものの上位、下位の内容については相関が高かった」など、全面的あるいは条件付きで、高い相関があることを肯定していた。

・コミュニケーション力、リーダシップなどの能力が高まった

相互評価結果をどのように表現するかはさまざまであるが、評価結果を意思として伝える局面、グループ作業をする場合にグループ内で行う議論の局面では、コミュニケーション力が必要である。相互評価はこのような局面が自然に発生するので、勢いコミュニケーションの機会が増え、その結果、その能力も向上する結果を期待できる。また、相互評価を行う多くのケースでグループなりチームなりが編成され、それらが組織的に機能する局面ではリーダーとなる学習者が存在し、これも自然発生的にリーダシップを発揮せざるをえない局面が登場する。その結果、リーダシップの醸成も期待できる。

●課題について

・学生による評価の妥当性の検証

各事例のほとんどが学生による評価の妥当性が保証されるかについての課題を挙げていたが、データを示してまでその妥当性を高く評価するものと、必ずしも学生の評価に客観性が保証できないとするものと、意見が分かれている。

特に、「自分には辛いが、他社には甘い」傾向があるとする事例がいくつかあったが、これを「他者への配慮」として肯定的にとらえる見方と、客観性を保証できない証拠の一つとして捉える見方に意見が分かれている。

学生による評価の妥当性は、評価の対象が何か、評価の方法がどのようなものか、教員と学生の立場の関係など複雑な要因が絡んでおり、検証結果を一般化するのはなかなか困難な作業であると思われる。実際に相互評価の仕組みを作る場合は、事例に示された検証例なども参考に、慎重に考える必要があると感じられた。

総合的にまとめると、初期的に、相互評価が学習者のモチベーションを喚起することには疑う余地がないと思われる。ただし、運用の中で、教員と学生との関係や学生同士の関係に配慮した公平性・客観性を保証する仕組みが存在しないと、長続きしないのではないかと結論付けることができる。

第2部　開発

第 2 部では、策定したモバイルアプリケーション開発人材の評価基準と、それを基に開発した教育プログラムの概要を述べる。

第1章　モバイルアプリケーション開発人材の評価基準策定

本事業では、モバイルアプリケーションの開発によって震災復興を支援できる人材の育成を目的としている。そこでまず、モバイルアプリケーション開発人材の要件を明確にするために、評価基準を策定した。

1.1 策定した評価基準の特徴

本事業で策定した評価基準は、組込みスキル標準（ETSS）[12]のフレームワークを参考に、専門学校生や大学生を想定した「レベル 0」を構築した。その内容に関しては、ETSS の他、IT スキル標準（ITSS）[13]、Android 技術スキル標準[14]等を参考にしている。また、本事業で策定した評価基準は、ETSS の職種「ソフトウェアエンジニア」の「レベル 1」の下位に当たることを想定している。そして、「レベル 0」に相当する専門学校生や大学生が学校を卒業後、学生レベルから社会人レベルへのキャリアパスがシームレスに接続するような構成を心がけた。

評価基準は、「キャリアフレームワーク」「能力評価基準」「研修ロードマップ」から構成される。

1.2 策定した評価基準の実際

策定した評価基準の、「キャリアフレームワーク」「能力評価基準」「研修ロードマップ」のそれぞれについて述べる。

1.2.1　キャリアフレームワーク

「レベル 0」に相当する者は、開発技術や開発手法の基礎を学習し、専門学校や大学等を卒業後、「レベル 1」以上にキャリアアップしていく者である。本評価基準の「レベル 0」は、ETSS における「ソフトウェアエンジニア」の「レベル 1」の下位に当たることを想定している。

ETSS のキャリアフレームワークは、職種・専門分野とキャリアレベルをまとめたものである。本評価基準では、ETSS の職種「ソフトウェアエンジニア」に、専門分野「モバイルアプリケーション開発」を新たに設け、その評価基準を策定することを目指した。この職種・専門分野の「レベル 0」に当たる人材を定義すると、「モバイルアプリケーションの各開発

[12] http://sec.ipa.go.jp/ETSS/index.html
[13] https://www.ipa.go.jp/jinzai/itss/index.html
[14] http://www.oesf.jp/modules/training/index.php?content_id=3

工程における開発・実装・テスト作業を学習する者」となる。

キャリアフレームワークの表を示すと、以下のようになる。表の塗りつぶしてある部分が、該当する職種・専門分野に規定するレベルである。将来的には、ETSS との整合性を取りながら、「レベル 1」以上も策定していく必要がある。

職種	ソフトウェアエンジニア
専門分野	モバイルアプリケーション開発
レベル 0	

図表 5　キャリアフレームワーク

次に、「モバイルアプリケーション開発」の「レベル 0」の人材に求められるスキル領域を以下の表に整理した。この表も、ETSS の「ソフトウェアエンジニア」を参考に、専門学校や大学等で教育すべきスキル領域をまとめたものである。また、震災復興を支援する人材を育成するという目的を果たすために、ETSS には含まれていない「災害対応」の領域を新たに設けている。

専門分野	スキル領域
モバイルアプリケーション開発	●開発技術 フレームワーク、システムアーキテクチャ、ユーザインタフェース、リソースファイル、アプリケーションコンポーネント、ストレージ、通信、GPS とセンサ、マルチメディア、テスト、アプリケーションの公開、開発手法 ●管理技術 統合マネジメント、スコープマネジメント、タイムマネジメント、コストマネジメント、品質マネジメント、組織マネジメント、コミュニケーションマネジメント、リスクマネジメント、調達マネジメント、開発プロセス設定、知財、開発環境マネジメント、構成管理・変更管理 ●パーソナルスキル コミュニケーション、リーダシップ、ネゴシエーション、問題解決 ●災害対応スキル 災害の基礎知識、放射線の基礎知識、災害対応マインド

図表 6　スキル領域

1.2.2 能力評価基準

「モバイルアプリケーション開発」の「レベル 0」に相当する人材に求められるスキルの評価基準を定めたものが、以下の表である。「開発技術」は、ETSS や「Android 技術スキル標準」の内容を参考に、より汎用的な内容に再構築した。「管理技術」は、ETSS や PMBOK[15]を参考にし、専門学校生や大学生に教育すべきレベルとして構成した。「パーソナルスキル」は、ETSS の内容を参考に構成したものである。また、「災害対応」に関しては、東日本大震災を教訓として、震災だけでなく、台風や火災等、他の災害も対象としている。さらに、震災発生から 1 年以上経過しても未だ収束の見えない原子力発電所の事故も鑑み、放射線に関する内容も扱うことにした。そして、知識やスキルだけでなく、自らの持てる知識やスキルを、防災や、災害からの復旧・復興に活用しようとするマインドも扱うこととした。

1.2.2.1 開発技術

開発技術は、モバイルアプリケーション開発に関わる技術的なスキル項目である。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●フレームワーク システムアーキテクチャ概要 ライセンス形態	レベル 0	開発しようとするアプリケーションを動作させるプラットフォームに関して、システムアーキテクチャの概要や関連するライセンス形態に関する基礎的な知識を有する。

15 「Project Management Body of Knowledge」。アメリカの非営利団体 PMI（Project Management Institute）が策定したプロジェクトマネジメントの知識体系。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●ユーザインタフェース メニュー ダイアログ イベント処理 ノティフィケーション スタイルとテーマ ビュー グラフィックス	レベル0	開発しようとするアプリケーションに応じて、適切なユーザインタフェースを選択し、構築するための基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●リソースファイル リソース XML ファイル セキュリティと許可	レベル0	開発しようとするアプリケーションに関し、リソースファイルや XML ファイル、セキュリティ関連の設定に関する基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●アプリケーションコンポーネント コンポーネントの構成 メッセージ	レベル0	開発しようとするアプリケーションを構成するコンポーネントや、コンポーネントどうしでやりとりするメッセージに関して、基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●フレームワーク システムアーキテクチャ概要 ライセンス形態	レベル0	開発しようとするアプリケーションを動作させるプラットフォームに関して、システムアーキテクチャの概要や関連するライセンス形態に関する基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●ストレージ ファイル入出力 ストレージ データベース	レベル0	ファイル入出力の概要や、ストレージ、データベースに関する基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●通信 ネットワークプロトコル Bluetooth Wi-Fi	レベル0	モバイル端末における通信方式（3G、Bluetooth、Wi-Fi）に関して、基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●GPS とセンサ GPS センサ	レベル0	GPS やセンサ（ジャイロセンサ、輝度センサ等）に関する基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●マルチメディア オーディオ ビデオ	レベル0	オーディオやビデオの扱い方に関する基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●テスト テスト手法	レベル0	開発したアプリケーションのテスト手法に関して、基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●アプリケーションの公開 アプリケーションの管理 アプリケーションの配布	レベル0	開発したアプリケーションのアップデートやバージョン管理、ユーザーへの配布方法に関する基礎的な知識やスキルを有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【開発技術】 ●開発手法 開発手法 開発ツール	レベル0	開発しようとするアプリケーションに応じて、適切な開発手法や開発ツールを選択するための、基礎的な知識やスキルを有する。

1.2.2.2 管理技術

管理技術は、PMBOK を参考に、プロジェクトマネジメントに必要な項目で構成した。但し、プロジェクトマネジメントのスキルは、専門学校生や大学生に相当する「レベル 0」では高すぎるため、ここでは主に「基礎的な知識を有する」というレベルとして規定した。

専門分野：モバイルアプリケーション開発	
スキル項目	評価基準
【管理技術】 ●統合マネジメント プロジェクト計画の策定 プロジェクト計画の実行	レベル 0 プロジェクトの各工程において、様々な要素を調和の取れた形に統合するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発	
スキル項目	評価基準
【管理技術】 ●スコープマネジメント プロジェクトの立ち上げ スコープ計画 スコープ定義 スコープ検収 スコープ変更管理	レベル 0 プロジェクトの目標である最終成果物を作成するために必要な作業が過不足なく確実に遂行されることを保証するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発	
スキル項目	評価基準
【管理技術】 ●タイムマネジメント 作業定義 作業順序設定 作業所要時間の見積もり スケジュール作成 スケジュール管理	レベル 0 プロジェクト開始前の計画段階に必要な作業を洗い出しスケジュールを作成し、そのスケジュールの予定と実績を管理していくための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●コストマネジメント 資源計画 コスト積算 予算設定 コスト管理	レベル0	承認された予算内でプロジェクトを完了させるための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●品質マネジメント 品質計画 品質保証 品質管理	レベル0	プロジェクトのニーズを確実に満足させるためことを目的とし、品質方針、目標および責任を定め、それら達成するために、品質計画、品質保証、品質管理、および、品質改善を実施していくための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●組織マネジメント 組織計画 要員調達 チーム育成	レベル0	プロジェクトに関与する人々を最も効果的に活用するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●コミュニケーションマネジメント コミュニケーション計画 情報配布 実績報告 完了手続き	レベル0	プロジェクト情報の生成、収集、配布、保管、廃棄をタイムリーかつ確実に行うための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●リスクマネジメント リスクマネジメント計画 リスク識別 定性的リスク分析 定量的リスク分析 リスク対応計画 リスクの監視・コントロール	レベル0	プロジェクトのリスクを識別し、分析し、リスクに対応することを目的とし、プロジェクトの目標に対してプラスに働く事象には、それが起こる確率とその発生結果が最大となるように、マイナスに働く事象については、逆に最小となるようにするための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●調達マネジメント 調達計画 引き合い計画 引き合い 発注先選定 契約管理 契約完了	レベル0	プロジェクト・スコープを達成する目的で、母体組織の外部から物品やサービスを取得するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●開発プロセス設定 システム開発プロセス設定 レビュー設定	レベル0	プロジェクトに応じて適切な開発プロセスを設定するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●知財マネジメント 関連法規 管理システム	レベル0	知財を管理・活用するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●開発環境マネジメント 開発環境企画 開発環境設計 開発環境構築 開発環境運用管理	レベル0	プロジェクトに応じて適切な開発環境を企画、設計、構築、運用するための、基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【管理技術】 ●構成管理・変更管理 識別 統制 記録 監査	レベル0	プロジェクトの質を向上させ、目標を達成することを目的とし、構成や変更を管理するための、基礎的な知識を有する。

1.2.2.3　パーソナルスキル

パーソナルスキルには、ETSS を参考に、コミュニケーション、ネゴシエーション、リーダシップ、問題解決という、業務を遂行していくために必要なスキル項目を設定した。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【パーソナルスキル】 ●コミュニケーション 聞く 話す 読む 書く	レベル0	プロジェクトの目標を達成するためのコミュニケーションスキルを身につけている。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【パーソナルスキル】 ●ネゴシエーション 質問 調査 主張	レベル0	プロジェクトの目標を達成するためのネゴシエーションに関する基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【パーソナルスキル】 ●リーダシップ 能力開発 時間管理 動機付け	レベル0	プロジェクトの目標を達成するためのリーダシップに関する基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【パーソナルスキル】 ●問題解決 着眼・発想 問題発見・分析 論理思考	レベル0	プロジェクトの目標を達成するための問題解決に関する基礎的な知識を有する。

1.2.2.4　災害対応

災害対応は、震災だけでなく、様々な災害に関する基礎的な知識や、放射線に関する基礎知識、及び、知識やスキルだけではなく、マインドに関わる項目も設定した。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【災害対応】 ●災害の基礎知識 自然災害 人災 防災 復旧・復興	レベル0	災害とその防止、復旧・復興に関する基礎的な知識を有する。

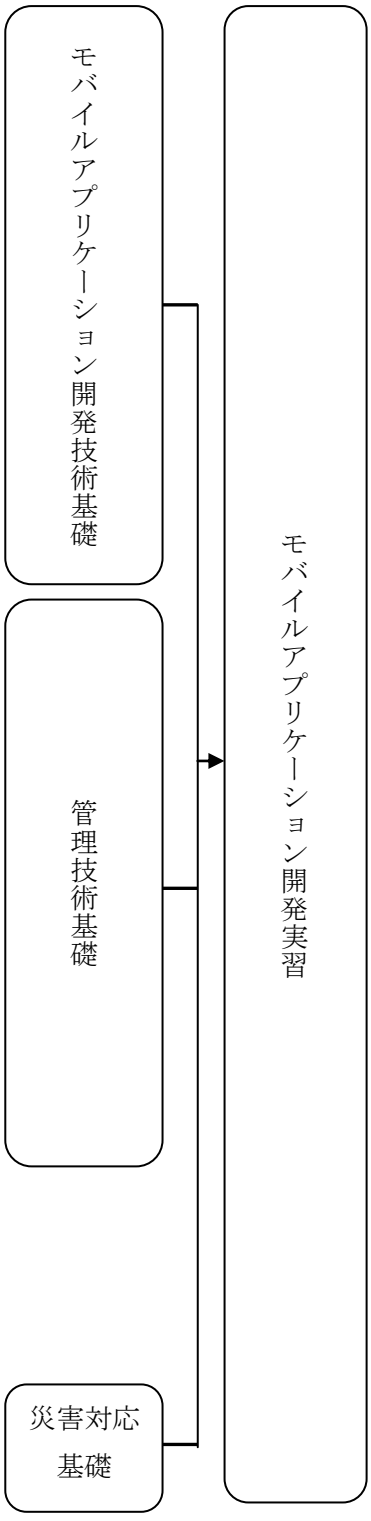
専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【災害対応】 ●放射線の基礎知識 放射線の概要 放射線の影響 法的知識 放射線の測定 除線	レベル0	放射線の概要、人体への影響、法的知識、測定、除線に関する基礎的な知識を有する。

専門分野：モバイルアプリケーション開発		
スキル項目	評価基準	
【災害対応】 ●災害対応マインド 防災マインド 復旧・復興マインド	レベル0	防災、復旧・復興に自分の身につけている専門知識やスキルを役立てようとするマインドを備えている。

1.2.3　研修ロードマップ

「レベル 0」では、モバイルアプリケーションの開発を一通り学習し、「レベル 1」以上へキャリアアップしていくための土台を固める。そのため、開発技術や管理技術に関する基礎的な内容を学習し、実際にモバイルアプリケーションの開発プロジェクトを体験しながら、実践力を向上させていくという方法で研修を進める。

以下の表は、「レベル 0」における各スキルカテゴリと、カリキュラムの科目の対応関係を示したものである。各科目の詳細は、カリキュラムで示す。

スキルカテゴリ		レベル 0
開発技術	フレームワーク	
	システムアーキテクチャ	
	ユーザインタフェース	
	リソースファイル	
	アプリケーションコンポーネント	
	ストレージ	
	通信	
	GPS とセンサ	
	マルチメディア	
	テスト	
	アプリケーションの公開	
	開発手法	
管理技術	統合マネジメント	
	スコープマネジメント	
	タイムマネジメント	
	品質マネジメント	
	組織マネジメント	
	コミュニケーションマネジメント	
	リスクマネジメント	
	調達マネジメント	
	開発プロセス設定	
	知財	
	開発環境マネジメント	
	構成管理・変更管理	
パーソナルスキル	コミュニケーション	
	リーダシップ	
	ネゴシエーション	
	問題解決	
災害対応	災害の基礎知識	
	放射線の基礎知識	
	災害対応マインド	

図表 7　研修ロードマップ

第2章　教育プログラム開発

策定した評価基準を元に、モバイルアプリケーションの開発で震災復興を支援する人材を育成するための教育プログラムを開発した。

2.1 開発した教育プログラムの特徴

本事業で開発した教育プログラムは、震災復興を支援するモバイルアプリケーションの開発人材育成を対象領域としている。本事業において被災地で実施したニーズ調査の結果からわかるように、被災地における IT エンジニアが向上させたいスキルとしては、プロジェクトマネジメントや語学力、企画力、コミュニケーション、リーダシップが上位を占めている。語学力を除き、これらのスキルを向上させるには、講義形式よりも、実際に開発プロジェクトを体験する PBL[16]形式の方がより効果的である。そこで本事業では、PBL を中心としたカリキュラムを構築した。なお、語学力は、モバイルアプリケーション開発だけでなく、より広範な領域に関わるスキル項目であるので、本教育プログラムでは割愛した。

本教育プログラムで実施する PBL は、主にミドルスタート方式で実施する。これは、白紙状態からアプリケーションを開発するのではなく、最低限の機能を備えたアプリケーションを題材として、そのソースコードを解析し、機能を変更したり、新たな機能を付け加えたりする方式である。白紙状態から一通り動作するアプリケーションを開発することは、「レベル 0」に相当する専門学校生には困難性が高いため、このような形式を採用した。

また、PBL で扱うアプリケーションのプラットフォームには、Android を採用した。Android は、プラットフォームや開発環境がオープンソースであり、無償で入手・利用でき、カスタマイズも自由にできる。そのため、Android は教育プログラムの題材として非常に扱いやすい。また、Android 上で動作するアプリケーションの開発言語は、IT 系の専門学校や大学等の授業で学習する Java を用いるため、学習者にとってもなじみやすい。さらに、個人・組織にかかわらず、開発したアプリケーションを Google Play[17]というアプリケーションの配布・販売サイトで自由に配布（無償）・販売（有償）が可能であるため、学習者の動機づけになりやすい。そして、Google 社のクラウド・コンピューティングを利用したサービスとの親和性が高いため、クラウド・コンピューティングの学習の題材としても活用できる。

本教育プログラムでは、このような形式で、モバイルアプリケーションの開発を題材とし、開発技術や管理技術をもととする実践力と、コミュニケーションやネゴシエーションなどのパーソナルスキルの向上を目指す。

---

16 Project Based Learning
17 https://play.google.com/store

2.2 構築したカリキュラム

前章に述べた評価基準を基に、カリキュラムを構築した。カリキュラム全体の学習時間数は 180 時間である。また、PBL 形式による「モバイルアプリケーション開発実習」が中心であるが、その前に、開発技術や管理技術、災害対応のスキル項目に関して学習し、基礎を固める構成とした。さらに、「モバイルアプリケーション開発実習」は、題材とするアプリケーションごとにユニットに分割し、部分的にも運用できるような構成としている。なお、「モバイルアプリケーション開発実習」で扱うアプリケーションは、ニーズ調査において、被災地の IT エンジニアが携わっているモバイルアプリケーションとして挙がっていることを考慮して選定した。

以下の表は、カリキュラムの全体構成を表したものである。

科目	時間数
モバイルアプリケーション開発技術基礎	30
管理技術基礎	30
災害対応基礎	15
モバイルアプリケーション開発実習	105
合計	180

図表 8　カリキュラム全体構成

以下、各科目のシラバスを列記する。

科目	モバイルアプリケーション開発技術基礎
時間数	30
目標	モバイルアプリケーション開発における開発技術の基礎的な知識やスキルを身につける
内容	フレームワーク システムアーキテクチャ ユーザインタフェース リソースファイル アプリケーションコンポーネント ストレージ 通信 GPS とセンサ マルチメディア テスト アプリケーションの公開 開発手法
備考	主に講義形式で行うが、必要に応じて e ラーニングや実習等も組み合わせる。

図表 9　モバイルアプリケーション開発技術基礎

科目	管理技術基礎
時間数	30
目標	モバイルアプリケーション開発を題材として、管理技術の基礎を学習する
内容	統合マネジメント スコープマネジメント タイムマネジメント コストマネジメント 品質マネジメント 組織マネジメント コミュニケーションマネジメント リスクマネジメント 調達マネジメント 開発プロセス設定 知財 開発環境マネジメント 構成管理・変更管理
備考	主に講義形式で行うが、グループワーク形式も取り入れ、学習した知識やスキルがどのように活用できるかを体験させる。

図表 10　管理技術基礎

科目	災害対応基礎
時間数	15
目標	災害の基礎知識や放射線の基礎知識を学習し、自らの知識やスキルを防災や、災害からの復旧・復興に活用しようとするマインドを醸成する。
内容	災害の基礎知識 　地震 　台風 　火災 　その他 放射線の基礎知識 　放射線の性質 　放射線の測定 　放射線からの防護 災害対応 　災害への IT の活用事例 　災害対応マインド
備考	講義形式に実習やケーススタディ等を組み合わせ、学習効果が高まるよう工夫する。

図表 11　災害対応基礎

科目	モバイルアプリケーション開発実習
時間数	105
目標	モバイルアプリケーション開発プロジェクトを体験し、開発技術、管理技術、パーソナルスキル、災害対応スキルの実践力を総合的に向上させる。
内容	ゲームアプリ開発 ニュース・天気アプリ開発 ビジネスアプリ開発 ネット・コミュニケーションアプリ開発 地図・交通アプリ開発 ツールアプリ開発 自由アプリ開発
備考	7 つの開発プロジェクトを題材とした PBL 形式で行う。1 つの開発プロジェクトは 15 時間程度で構成する。「自由アプリ開発」以外は、既存のアプリケーションの機能追加や変更等を題材とするミドルスタート方式で行う。「自由アプリ開発」は、これまでに学習した内容を活用し、チームで自由にモバイルアプリケーションを開発する形式とする。

2.3 開発した教材

本事業では、カリキュラムを構成する科目のうち、「モバイルアプリケーション開発実習」から「ゲームアプリ開発」の部分の教材を開発した。これは、本教育プログラムの一部を実証講座として試行導入した場合、学生にとって学習に対するモチベーションを保ちやすいモバイルアプリケーションはどのようなものかを考えた結果、普段からよく利用しているゲームが最適であると判断したからである。但し、単なるゲームを扱うだけでは震災復興に結びつかないことから、ゲームの内容に被災地である宮城県に関連する要素を取り入れ、震災からの復興を常に意識できるような内容としている。

以下、開発した教材の一覧である。

①Android アプリケーション開発環境構築マニュアル
②Android ゲームアプリケーション　ロールプレイングゲーム解説
③Miyagi Quest 追加機能　仕様解説書
④ロールプレイングゲーム「Miyagi Quest」プログラム一式

開発に当たっては、昨年度、本校が取り組んだ平成 22 年度文部科学省　産学連携による実践型人材育成事業―専門人材の基盤的教育推進プログラム―「クラウドサービスの設計や開発を担う中堅人材を育成するプログラムの開発と普及」で開発した教材をベースとして新たに開発した。具体的には、昨年度の教材開発以降に実施された Android や関連ソフトウェアのバージョンアップを反映させた。また、本事業は宮城県仙台市で実施することを考慮し、舞台を宮城県に移し、ゲームのストーリーや設定、敵キャラクターなどにも宮城県に関係のある要素を取り入れたものを開発した。もとのプログラムは、昨年度の事業内で実施したプログラミングコンテストの応募作品の一つである「真・MiyazakiQuest」（チーム CIC_AI、中央情報経理専門学校高崎校）をベースとしている。

2.3.1 Androidアプリケーション開発環境構築マニュアル

「Android アプリケーション開発環境構築マニュアル」は、Android アプリケーションを開発するために必要なツール類の入手方法、インストール方法、設定方法、エミュレータを用いた開発方法等を解説している。また、Android アプリケーションの開発に用いる統合開発環境の「Eclipse」に関しては、最初から日本語化されたバージョンである「Pleiades」をインストールする方法を解説した。これは、実際に講座として実施した場合、英語版のものをインストール・設定するよりも学生の負担が少なく、スムーズに講座を始められることを考慮した結果である。

以下は、「Android アプリケーション開発環境構築マニュアル」の内容である。なお、詳

細は巻末資料 2 を参照されたい。

2.3.2 Androidアプリケーション　ロールプレイングゲーム解説

「Android ゲームアプリケーション　ロールプレイングゲーム解説」は、ロールプレイングゲーム「Miyagi Quest」のソースコード、及びその解説をまとめたものである。また、一般的なゲーム開発の流れも紹介している。

ロールプレイングゲーム「Miyagi Quest」は、宮城県を模したフィールドを舞台とし、主人公を操作して敵と戦ったり、町で会話したり、アイテムを見つけたりしながら冒険を進め、特定の敵（ボスキャラクター）を倒すとゲームクリアになる、というゲームである。昨年度のベースプログラムである「Miyazaki Quest」にはストーリーやボスキャラクター、町、会話イベント等は実装されていなかったが、今年度は最初から実装して開発した。

以下、「Android ゲームアプリケーション　ロールプレイングゲーム解説」の内容である。なお、詳細は巻末資料 3 を参照されたい。

2.3.3 Miyagi Quest　追加機能　仕様解説書

「Miyazaki Quest」から「Miyagi Quest」に発展させる上で、追加した機能について解説した資料である。ソースコードのどこで設定しているかや、実際に設定方法も解説している。

以下、「Miyagi Quest　追加機能　仕様解説書」の内容である。なお、詳細は巻末資料 4 を参照されたい。

１ NPC（町にいるキャラクター）の設定方法
　１－１ マップ上の座標
　１－２ グラフィック
　１－３ 会話時のセリフ
　１－４ NPC（町にいるキャラクター）の設定時における注意点
２ マップ間移動の設定方法
　２－１ 移動先マップの設定
　２－２ 移動先マップでの初期位置
　２－３ 移動元マップの移動フラグ発生位置
　２－４ マップ作りにおける注意点
３ フラグ処理
　３－１ フラグ発生位置の設定
　３－２ イベント処理
　　３－２－１ センダイ：攻撃力アップイベント
　　３－２－２ センダイ：防御力アップイベント
　　３－２－３ センダイ：素早さアップイベント
　　３－２－４ ナナシ：カギの入手
　　３－２－５ センダイ：にじのしずくの入手
　　３－２－６ オオサキ：オーブの入手
　　３－２－７ ボスキャラまでの道
　３－３ フラグ処理に関する注意点
４ 戦闘中の特殊攻撃の処理
　４－１ 特殊攻撃
　４－２ 回復行動
　４－３ 戦闘中の特殊攻撃の処理に関する注意点

2.3.4 ロールプレイングゲーム「Miyagi Quest」プログラム一式

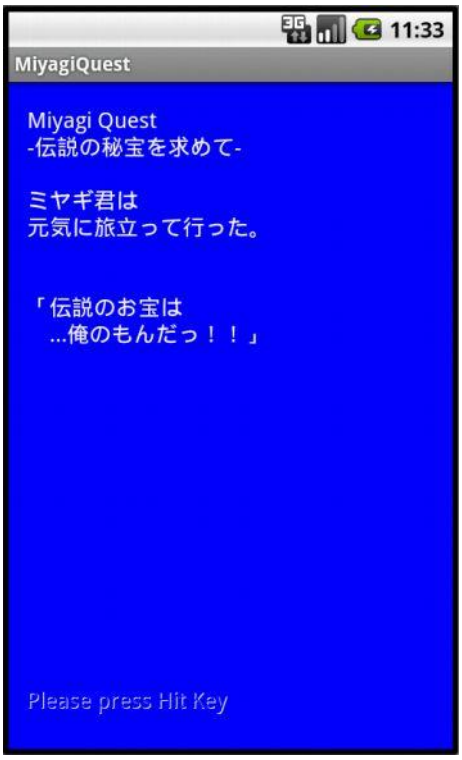
「Miyagi Quest」のプログラム本体である。ソースコードの他、BGM、画像ファイル等

全てが含まれている。統合開発環境 Eclipse でインポートすることで、利用可能である。

ゲームには、全体のフィールドマップに宮城県を模した形を用いている。また、ゲーム内に「センダイ」「オオサキ」という 2 つの町（それぞれ仙台市、大崎市から名前を取っている）がある。さらに、敵キャラクターとして「ミヤぷよ」という、伊達政宗の兜の前立てをモデルにしたものが登場するなど、宮城県に関連するものを取り入れている。

以下、「Miyagi Quest」の主な画面を列記する。

図表 12　「Miyagi Quest」タイトル画面



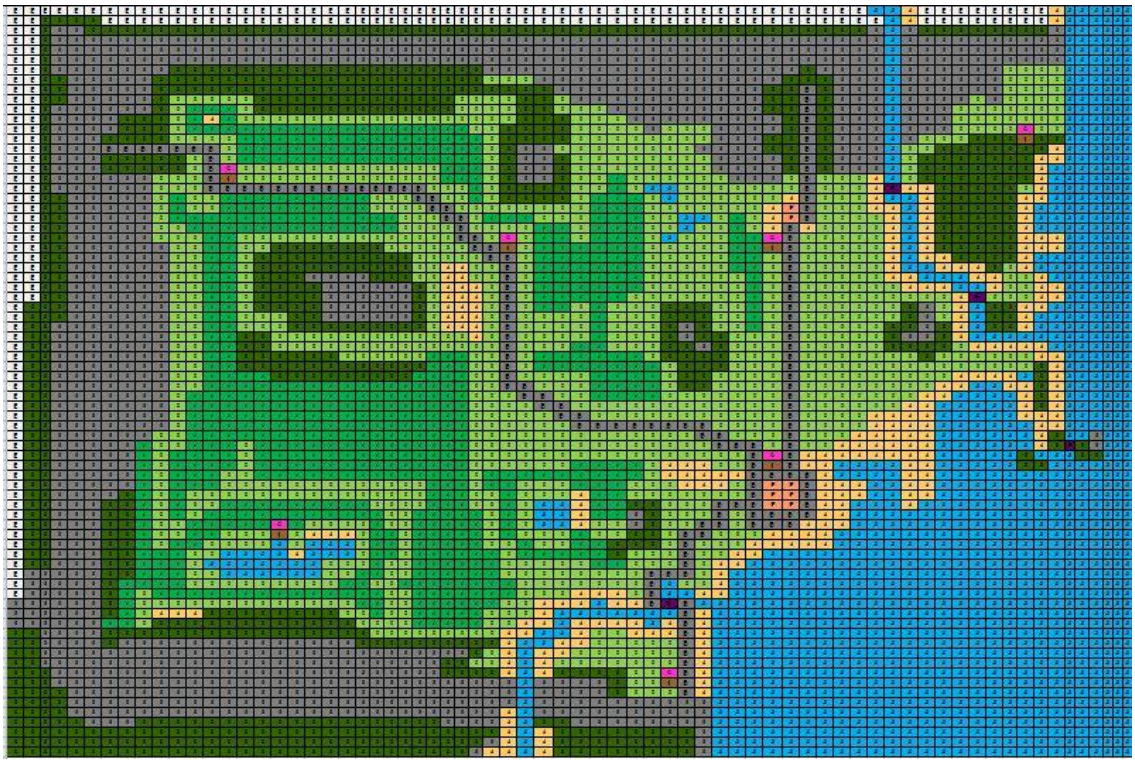
図表 13　プロローグ

図表 14　フィールド画面

図表 15　戦闘画面

図表 16　町

図表 17　センダイの町



図表 18　フィールドマップ全体

第3部　実証実験

第 3 部では、開発した教育プログラムの教育効果や妥当性を検証するために実施した実証実験について述べる。

第1章　実証実験の概要

1.1 実証実験の目的

本事業では､モバイルアプリケーションの開発によって震災復興を支援できる人材を育成するために、PBL を中心とした教育プログラムを開発した。この教育プログラムの教育効果や妥当性を検証するために、実証実験を実施した。

本実証実験は、開発した教育プログラムの中核となる､「モバイルアプリケーション開発実習」の一部を抽出して実施した。具体的には、学生にとってなじみの深い「ゲーム」の部分である。この「モバイルアプリケーション開発実習」は、チームで PBL を実施するという形式になっている。そこで、実証実験に参加する学生のモチベーションを高めるために、推進協議会参画校からチームで参加し、PBL の成果を競い合うコンテスト形式で実施することにした。また、事業期間の制約上、モバイルアプリケーションを完成させることは、例えミドルスタート方式でも困難性が高いことが予想された。そこで､「ゲーム」そのものの開発に限らず､キャラクターデザインや企画書等､得意な分野での参加も認めることとした。そのため、本コンテストでは、PBL 教材を用いた規定部門と、得意な分野で応募する自由部門を設けた｡但し､本事業は被災地である宮城県仙台市で実施することが前提であるため、コンテストの中心は仙台大原簿記情報公務員専門学校、及び、同じ学校法人である仙台デザイン専門学校としている。そのため、この 2 校からは複数の作品の応募を認めたが、他の学校からは、規定部門及び自由部門それぞれで 1 つまでとした。

1.2 実証実験の対象者

実証実験では､本事業の推進協議会に参画している専門学校の所属学生を対象とした｡規定部門は Android 上で動作するモバイルアプリケーションの開発であるので、ソフトウェア開発系や組込み開発系の学生を対象としたが､自由部門は多少範囲を広げている｡コンテスト参加には､前もってエントリーシートを提出して参加するという形態を取り､規定部門が 3 校 17 名、自由部門が 6 校 45 名、合計 7 校 60 名（2 名は規定部門と自由部門の両方に参加）が対象者となった。内訳は以下の表の通りである。

学校	人数
中央情報経理専門学校高崎校	6 名
富山情報ビジネス専門学校	4 名
麻生情報ビジネス専門学校	7 名
合計	17 名

図表 19　実証実験対象者（規定部門）

学校	人数
宮崎情報ビジネス専門学校	5 名
仙台大原簿記情報公務員専門学校	5 名
仙台デザイン専門学校	26 名
中央情報経理専門学校高崎校	3 名
富山情報ビジネス専門学校	2 名
船橋情報ビジネス専門学校	4 名
合計	45 名

図表 20　実証実験対象者（自由部門）

1.3 コンテストの概要

1.3.1　コンテスト名称

コンテストの名称は、「震災復興に貢献するモバイルアプリケーション等コンテスト」とした。これは、モバイルアプリケーションやその他の作品に、震災復興の要素を込め、自らの備える知識やスキルを震災からの復興に活用しようとする災害対応マインドを醸成させることを意図した名称である。

1.3.2　開催目的

専門学校生が日頃の学習成果を活かして､プログラミング技術やアイディア等を競うことを目的とした。また、参加者のチームワークや実現力を伸ばす機会となることを目指した。

1.3.3　コンテストの特徴

本コンテストには、以下の特徴を持たせた。

・取り組みやすい題材を通じて、プログラミング等、作品を創り出すことの面白さの体感
・世界最大級の SNS「Facebook」を活用し、グループで課題に取り組むことで、コミュニケーションや協調性の大切さを再認識

・制作した作品を学生どうしで相互に評価することで、他者のアイディアを参考にし、自己の技術向上に活用
・仙台市を中心に開催し、地域の専門学校を盛り上げ、震災復興に貢献

参加者が取り組みやすい題材とするため、参加者にも身近でかつなじみやすく、実際のプレイも可能なゲームを規定部門の題材とした。自由部門は、それに関連したものを中心に、PBL で学習した内容を活用できるように、プログラミング、キャラクターデザイン、音楽、企画書・レポート等の作品を募集した。

今回のコンテストでは、グループで課題に取り組み、コミュニケーションやリーダシップ等のパーソナルスキルの向上も目指した。そのため、Facebook を活用し、参加チームごとにグループを作って、ディスカッション等に活用できる環境を構築した。また、コンテスト事務局から全体への連絡等や、参加者からの質問とそのフォロー等にも、Facebook を利用している。

また、参加者のモチベーションの向上を目的とし、応募作品を参加者どうしで相互に評価する仕組みを取り入れた。具体的には、本コンテストに協賛している特定非営利活動法人教育支援システム研究機構の「オンライン評価システム[18]」を利用し、参加者どうしの相互評価の結果をコンテストの一次審査の一部に取り入れている。これによって、他の応募作品を利用・閲覧してアイディア等を参考にし、参加者の技術向上に活用してもらうこととした。さらには、コンテストの作品評価の効率性を高める効果も狙った。

また、コンテストの応募作品の開発や制作はそれぞれのチームごとに、所属する学校を中心に行うが、二次審査に取り入れた参加者によるプレゼンテーションは、仙台市内で実施した。全国からコンテスト参加者が仙台に集まることによって、学校間の学生レベルでの交流を深め、被災地の専門学校を支援する新たな形を探った。

1.3.4　実施体制

本コンテストは、本事業の推進協議会が主催して開催した。また、推進協議会参画校や、IT 企業、団体、NPO 法人等の協力・協賛によって実施した。

・主催：「モバイルアプルケーションの開発で震災復興を支援する人材の育成」推進協議会
（委員長：学校法人宮崎総合学院　理事長　川越宏樹）
・協力：学校法人宮崎総合学院（宮崎県）
学校法人北杜学園（宮城県）
学校法人龍澤学館（岩手県）
学校法人中央総合学園（群馬県）
学校法人浦山学園（富山県）

[18] 巻末資料 5 を参照

学校法人麻生塾（福岡県）
学校法人コンピュータ総合学園（兵庫県）
学校法人三橋学園（千葉県）
社団法人宮崎県情報産業協会
社団法人組込みシステム技術協会
株式会社宮崎コミュニティーカレッジ
株式会社インタープロ
株式会社クオリティ・エージェント
宮崎県

・協賛：特定非営利活動法人教育支援システム研究機構

1.3.5 作品テーマ

PBL 教材の「MiyagiQuest」を題材とした規定部門と、プログラミング、キャラクターデザイン、音楽、企画書・レポート等を応募する自由部門を設けた。

規定部門は、「MiyagiQuest」のソースコードを利用し、以下の 2 つの課題に取り組むものとした。

（1）キャラクターの追加、または差し替え

ゲーム内の町「センダイ」に、新たなキャラクターを追加する。または、既にいるキャラクターのグラフィックを差し替える。ゲームへの実装を前提とするが、追加、または差し替えるキャラクターのグラフィックデータのみでも評価を行う。ゲームに実装した場合は、さらに高得点が得られるものとする。

（2）操作性の改善

タッチパネルのみで操作する『MIYAGI QUEST』は、主人公やカーソルを移動させるためには、画面上で指をスライドさせて行う。しかし、スライドする方向が少しずれるだけで、意図した方向とは別の方向に移動してしまうことが多く、操作性にやや難がある。物理キーを搭載しない最近の端末でも快適な操作ができるような工夫をする。ゲームに実装し、その操作性を評価する。なお、操作性の改善案をまとめた企画書を提出しただけでも評価を行う。ゲームに実装した場合は、さらに高得点が得られものとする。

なお、課題（1）（2）以外にも、機能を追加するなど工夫が見られた場合にも、加点対象とした。

自由部門は、以下に該当する作品のいずれかを提出するものとした。

・プログラミング（ジャンル・プラットフォーム問わず）

・キャラクターデザイン（手書きのものはスキャナ等で取り込む）
・音楽（PC で再生可能なデータ形式）
・企画書、レポート等（PC で表示・印刷可能な形式にする）

なお、規定部門と同じ作品は提出できないこととした。

1.3.6 作品テーマ

本コンテストは、本教育プログラムの実証実験として開催するものであるため、参加者を推進協議会に参画する学校法人の学校に所属する学生に限定した。次年度以降は、本事業の成果をさらに発展させていき、被災地の専門学校を中心に全国から参加者を募集し、被災地においてコンテストを実施する予定である。

1.3.7 提出物

コンテスト実施期間中、その進行段階ごとに参加者は提出物の作成をすることとし、コンテスト事務局宛に提出する形式を取った。提出物は、以下の通りである。

1.3.7.1 エントリー時

エントリー時は、参加者それぞれが個人用エントリーシートに記入し、チーム参加の場合はチーム用エントリーシートも作成して、事務局宛にメールで提出するという形を採った。なお、コンテストに関する連絡用に、参加者一人ひとりに Facebook への登録を条件とし、事務局からの友達申請を承認した上で、事務局がコンテストのグループ、及びチームのグループ（チーム参加の場合）に登録をしている。

1.3.7.2 作品提出時

作品提出時には、作品紹介シート、作品、開発報告書、作品補足資料（任意）を提出させた。このうち作品は、参加した部門によって以下のものを「オンライン評価システム」にアップロードすることで提出を行った。

部門	課題	提出物
規定部門	（1）キャラクターの追加、または差し替え	プログラムファイル一式 ※（1）（2）の両方で一式提出 なお、実装まで至らなかった場合は、以下のものを提出する。 （1）キャラクターのデータファイル（PNG ファイルで、24×24 ピクセル） （2）改善案の企画書ファイル（PowerPoint 形式）
	（2）操作性の改善	
	（1）、（2）以外に機能を追加するなどの工夫が見られた場合は、さらに高得点が得られるものとする。	

図表 21　作品の提出（規定部門）

部門	サブカテゴリ	提出物
自由部門	プログラミング	プログラムファイル一式
	キャラクターデザイン	JPG、GIF 等、PC で表示可能な画像ファイル（手書きのものはスキャナ等で取り込むこと）
	音楽	MP3、MIDI 等、PC で再生可能な音楽ファイル（演奏を録音して上記のファイル形式に変換しても良い）
	企画書、レポート等	Word、PowerPoint、PDF 等

図表 22　作品の提出（自由部門）

また、作品以外の書類（作品紹介シート、開発報告書、作品補足資料）に関しては、Facebook のメッセージ機能を用いて事務局宛に送信するという方法で行った。

作品紹介シートは、作品のコンセプトや苦労した点等を簡単にまとめたものである。開発報告書は、二次審査のプレゼンテーションで用いる資料で、作品のコンセプトや工夫した点、苦労した点等を自由にまとめてもらった。なお、「企画書・レポート等」を提出する場合は、それ自体をプレゼンテーションの資料として用いるものとし、開発報告書の作成は免除した。作品補足資料は、開発報告書では説明しきれない内容があった場合に、任意で作成してよいことにした。

1.3.8　コンテストのスケジュール

実施に当たり、下記のスケジュールを作成し、コンテストを行った。

日程	内容	形態
平成 24 年 4 月 16 日	・コンテスト説明会実施	Ustream による配信
平成 24 年 4 月 20 日（締切）	・エントリーシート提出 ・その後、順次 Facebook への登録 ・Facebook 登録まで済んだら開発開始	事務局宛にメールで送信 Facebook 上での登録作業
平成 24 年 4 月 27 日	・オンライン評価システムのアカウント発行	事務局から Facebook のメッセージ機能で送付
平成 24 年 5 月 11 日（締切）	・作品提出	作品はオンライン評価システムへアップロード、作品紹介シート等の書類は Facebook のメッセージ機能で事務局宛に送信
平成 24 年 5 月 12 日～5 月 15 日	・一次審査	参加者による相互評価、及び事務局による提出物等評価
平成 24 年 5 月 16 日	・一次審査結果発表	事務局から参加チーム／参加者へ Facebook のメッセージ機能で通知
平成 24 年 5 月 22 日	・二次審査（プレゼンテーション） ・審査発表 ・表彰式	審査委員による審査

1.3.9　審査

提出された作品や書類に関して、以下のように審査を行った。

1.3.9.1　審査手順

審査は、一次審査と二次審査にわたり行った。一次審査では、提出された作品を参加者どうしによる相互評価を中心に、提出資料の内容や提出状況、相互評価による評価作品数、Facebook での活動状況等を加味して行った。一次審査の結果、上位 12 作品（規定部門 3 作品、自由部門 9 作品）を選抜し、二次審査の対象とした。二次審査は、仙台市内で作品評価会を開催して実施した。12 作品の応募者（チームの場合は代表者 1 名）を審査会場ま

で招待した。作品評価会では、事前に提出された開発報告書を基に、制作した作品についてのプレゼンテーションを行い、その内容を審査した。二次審査の審査委員は、本事業の推進協議会委員である。これらの一次審査、二次審査の評価を総合して、入賞作品を決定した。

1.3.9.2　審査基準、審査項目

一次審査の参加者による相互評価は、評価者の主観で 5 段階の点数をつけてもらう方法で行った。

事務局による審査項目は、以下の通りである。但し、相互評価、及びプレゼンテーションの評価に重きを置き、事務局による審査では大きく差が付かないように調整した。

項目	審査概要
エントリー手続き	エントリーシートの提出期日が守られていたか。エントリーシート提出後、速やかに Facebook 上の手続きを完了したか。
作品提出	作品提出の期日が守られていたか。
相互評価	期間内に評価を行った作品数（チームの場合はメンバーの平均)
Facebook の利用状況	Facebook を積極的に利用し、ディスカッションや事務局への質問等を活発に行っていたか。
提出書類	提出書類の記入に不備はなかったか。内容はわかりやすかったか。

図表 23　事務局審査の項目

二次審査では部門によって評価すべき観点が異なるため、審査項目も部門ごとに設定している。但し、自由部門では、「コンセプト」「作品ごとの項目」「努力の度合い」「発表」という項目で統一した。具体的には、以下の通りである。

項目	審査概要
課題 1 （デザイン面）	「センダイの町」に差し替え、または追加したキャラクターのデザイン、台詞、設定等に被災地復興の要素が込められているか。
課題 2 （機能面）	「操作性の改善」に工夫やアイディアが見られるか。
自由課題	課題 1・課題 2 以外の機能やキャラクターの追加に努力は見られたか。
発表	発表内容はわかりやすかったか。

図表 24　規定部門の審査項目（二次審査）

項目	審査概要
コンセプト	ゲームのコンセプトはユニークか。魅力を感じるか。
機能・操作性	ゲームの機能や操作性はわかりやすいか。
努力の度合い	作品の開発・制作に努力が見られるか。
発表	発表内容はわかりやすかったか。

図表 25　自由部門（ゲーム）の審査項目（二次審査）

項目	審査概要
コンセプト	ソフトウェアのコンセプトはユニークか。魅力を感じるか。
機能・操作性	ソフトウェアの機能や操作性はわかりやすいか。
努力の度合い	作品の開発・制作に努力が見られるか。
発表	発表内容はわかりやすかったか。

図表 26　自由部門（業務系ソフトウェア）の審査項目（二次審査）

項目	審査概要
設定・コンセプト	キャラクターの設定やコンセプトはユニークか。魅力を感じるか。
デザイン	キャラクターの見た目や色使いに魅力を感じるか。
努力の度合い	作品の開発・制作に努力が見られるか。
発表	発表内容はわかりやすかったか。

図表 27　自由部門（キャラクターデザイン）の審査項目（二次審査）

項目	審査概要
コンセプト	企画のコンセプトはユニークか。魅力を感じるか。
実現可能性	企画の実現可能性は高いと考えられるか。
努力の度合い	作品の開発・制作に努力が見られるか。
発表	発表内容はわかりやすかったか。

図表 28　自由部門（企画書・レポート等）の審査項目（二次審査）

なお、今回は「音楽」での応募はなかった。

二次審査では、審査委員がこれらの項目ごとに 5 段階で評価を付け、項目ごとの平均点を合計して審査結果とした。

1.3.9.3　審査発表、表彰

一次審査、二次審査に事務局からの審査の結果を合計して、最優秀作品賞（規定部門、自由部門１作品ずつ)、優秀作品賞（規定部門、自由部門合わせて３作品)、審査員特別賞（1作品）を選び、表彰を行った。審査員特別賞は、二次審査の対象となった全作品のうち、特に印象に残った作品を審査員に３つまで選んでもらい、最も選ばれた作品とした。

1.3.9.4　賞品

本コンテストでは、特定非営利活動法人教育支援システム研究機構の協賛により、賞品を用意した。最優秀作品賞には、チームメンバーに図書カード 10 万円分と、所属する学校にノートパソコン１台を進呈した。また、コンテスト参加者全員に記念品が贈られた。

1.3.10　コンテスト実施体制

コンテスト実施に当たって、審査委員会、及び事務局を組織した。

＜審査委員会＞

審査委員長	川越　宏樹	学校法人宮崎総合学院　理事長
審査委員	鈴木　忠	学校法人北杜学園　理事長
審査委員	龍澤　正美	学校法人龍澤学館　理事長
審査委員	中島　利郎	学校法人中央総合学園　理事長
審査委員	浦山　哲郎	学校法人浦山学園　理事長
審査委員	古野　金廣	学校法人麻生塾　副理事長
審査委員	福岡　壯治	学校法人コンピュータ総合学園 神戸電子専門学校　校長
審査委員	鳥居　高之	学校法人三橋学園 船橋情報ビジネス専門学校　校長
審査委員	田村　武志	学校法人コンピュータ総合学園 神戸情報大学院大学　教授
審査委員	津曲　雄二	宮崎県　商工観光労働部商業支援課
審査委員	向畑　公俊	宮崎県　県民政策部交通課
審査委員	川崎　友裕	株式会社宮崎情報処理センター 代表取締役社長
審査委員	髙見　裕貴	株式会社宮崎コミュニティーカレッジ 部長
審査委員	南　克浩	株式会社インタープロ 代表取締役社長

＜事務局＞

事務局長	岩村　聡志	宮崎情報ビジネス専門学校 教務部・事業推進部　部長
事務局	中村　徹	株式会社教育事業支援センター
事務局	荒島　健太郎	株式会社教育事業支援センター

1.3.11　その他（サポート方法、著作権、個人情報の取り扱い）

その他、コンテストの実施に当たって以下の事項を十分留意した運営をすることとした。

1.3.11.1　サポート方法について

不明点や問合せ事項のある場合、Facebook の「震災復興に貢献するモバイルアプリケーション等コンテスト」のグループのウォールに投稿することで受け付けることとした。その後、担当者から回答を投稿する形となっている。

1.3.11.2　著作権の取り扱いについて

応募作品の著作権は、規定部門の作品は文部科学省に、自由部門の作品は制作者に帰属するものとした。また、以下の資料等については教育目的のみに使用する場合があるとした。

・エントリーまたは作品提出時に提出された書類、データ

・作品評価会時に撮影した写真、ビデオ等

1.3.11.3　参加者の個人情報の取り扱いについて

学校法人宮崎総合学院、及び、推進協議会が、本コンテスト実施運営に際して取得した個人情報は、宮崎総合学院の個人情報取扱い方針に則り適正に扱い、実施事務局が責任を持って管理、運営するものとした。

第2章　実証実験の詳細

上記のコンテスト概要で示した内容をもって、「震災復興に貢献するモバイルアプリケーション等コンテスト」を開催した。

2.1 コンテスト参加

2.1.1　参加手続きの概要

コンテスト実施に当たり、推進協議会でその内容を議論した上で承認を得た。今回のコンテストは、本事業の推進協議会参画校で行われることとなったため、推進協議会委員から各学校の学生への通知がなされ参加者を募集した。コンテストの参加に当たっては、規定部門はチーム参加のみ、自由部門は個人でもチームでも参加可能とした。その結果、規定部門には3校3チーム、自由部門には6校5チーム及び個人参加26名の参加となった。エントリー方法は、推進協議会委員を通して学生にエントリーシートのWordファイルを配布し、必要事項を入力の上、事務局宛にメールで送信する方法を採った。さらに、Facebookのアカウントを用意し、事務局からの友達申請を承認することとした。事務局からの友達申請を承認した後は、事務局によってコンテスト全体のグループ、及びチームごとのグループに登録がなされた。なお、コンテスト運営に当たって作成したFacebookのグループは全て「秘密」の状態にしてあるため、グループに参加している者以外からはグループの存在がわからないようになっている。チームのグループも、チームのメンバーと事務局以外からは見えないようになっているため、作品の開発・制作に当たってのディスカッション内容も、他のチームのメンバーにはわからないようになっている。

2.1.2　説明会

コンテストの概要やエントリー手続き、作品の提出方法、審査方法等を説明した説明会を実施した。参加チームは全国に分散しており、また、必要に応じて後日再び確認できるように、Ustreamでの動画配信により行った。コンテスト説明会の概要は以下の通りである。

2.1.2.1　開催日時

説明会は、平成24年4月16日の16:00～16:30に開催した。また、配信した説明会の映像をUstreamのサイトに保存し、4月20日まで視聴できるようにした。

2.1.2.2　開催方法

Ustreamによる実況中継で行った。

2.1.2.3　配信内容

配信内容は、コンテスト概要として、規定部門とその課題、自由部門とその募集作品、提

出物、スケジュール、審査方法などを説明した。

2.1.2.4　使用機材等

Ustream 配信には、Android 搭載スマートブック「IS01」と、Ustream 配信・視聴用公式アプリ「Ustream」を用いて行った。具体的には、説明会資料をパソコンのディスプレイに表示し、それを IS01 のカメラ機能で撮影して、Ustream 上にリアルタイムで放送を流した。

2.1.3　エントリー

2.1.3.1　エントリーシート提出

説明会開催後、参加者は各自でエントリーシートを記入し、事務局へメールで送信するという方法で提出した。なお、チーム参加の場合は、チーム用のエントリーシートも提出した。

2.1.3.2　Facebook 登録

チームでのディスカッションや、事務局からの連絡を目的として、Facebook を活用した。参加者は Facebook にアカウントを用意し、事務局からの友達申請を承認することで、エントリーが完了する。事務局からの友達申請を承認した後、事務局がコンテストのグループ、及びチームのグループを用意し、それらに参加者を登録した。これらのグループの役割は、以下の通りである。

●コンテストのグループ

事務局からコンテスト参加者全員への連絡事項は、全てこちらのグループのウォールに投稿した。また、参加者から事務局への質問がある場合も、こちらのウォールに投稿し、担当者がそれに回答する、という形式を採った。

●チームのグループ

参加チームごとに設けられたグループで、作品の開発や制作に当たってのディスカッションに活用した。

図表 29　コンテストのグループ

また、事務局から参加者個人への連絡や、参加者から事務局への書類提出等に、メッセージ機能を活用している。

2.2 参加者、応募作品について

参加者は、推進協議会参画校のうち 7 校からの 60 名であった。チームは、8 チーム及び個人参加 26 名であった。以下の表は、規定部門、自由部門のそれぞれに参加したチームの内訳である。

チーム名	学校	都道府県	人数
CIC_AI	中央情報経理専門学校高崎校	群馬県	6 名
1 ビットからの貢献	富山情報ビジネス専門学校	富山県	4 名
ABCC JK	麻生情報ビジネス専門学校	福岡県	7 名
	合計		17 名

図表 30　参加チーム一覧（規定部門）

チーム名	学校	都道府県	人数
宮情 IT	宮崎情報ビジネス専門学校	宮崎県	5 名
MOM5days	仙台大原簿記情報公務員専門学校	宮城県	5 名
（個人参加）	仙台デザイン専門学校	宮城県	26 名
UHI	中央情報経理専門学校高崎校	群馬県	3 名
1 ビットからの貢献	富山情報ビジネス専門学校	富山県	2 名
技術部	船橋情報ビジネス専門学校	千葉県	4 名
	合計		45 名

図表 31　参加チーム一覧（自由部門）

チームごとのプロフィール、応募作品は以下の通りである。

2.2.1 CIC_AI（中央情報経理専門学校高崎校）（規定部門）

No	学科	学年	性別
1	高度情報システム学科	3 年	男性
2	高度情報システム学科	3 年	男性
3	高度情報システム学科	3 年	男性
4	高度情報システム学科	3 年	男性
5	高度情報システム学科	3 年	男性
6	高度情報システム学科	3 年	男性

図表 32　CIC_AI（中央情報経理専門学校高崎校）のメンバー構成

タイトル	Miyagi Quest・改
作品紹介	「短いながらも最後まで遊べる RPG」への想い！ 前回「MiyazakiQuest」コンテストで培ったノウハウを活かし、トータルバランスを考慮して開発しました！！ ①キャラクター追加 ご当地ならではのキャラクターを追加！彼らからはご当地ならではの情報を聞き出せます！ ②操作性の改善 タップ操作・カーソル操作どちらにも対応！細かい操作もスムーズに実現！ ③オリジナルＢＧM追加 センダイ、オオサキ、ラスボス、エンディングの計 4 曲を新たに書き起こして追加！ ④画面バランスの調整 実記に合致したウィンドウサイズやメニューの実装！ ⑤エンディング・スタッフロールの追加 最後までプレイしてくれた方々への感謝！

図表 33　CIC_AI の作品紹介

CIC_AI は、昨年度実施した「クラウドを活用した Android アプリケーション開発コンテスト」の参加者を中心に構成されたチームである。Android アプリケーションの開発経験もあり、また、独自に BGM を作成できる、技術力の高いチームである。

2.2.2　1ビットからの貢献（富山情報ビジネス専門学校）（規定部門）

No	学科	学年	性別
1	高度情報システム学科	3 年	男性
2	高度情報システム学科	3 年	男性
3	高度情報システム学科	3 年	男性
4	高度情報システム学科	2 年	男性

図表 34　1 ビットからの貢献（富山情報ビジネス専門学校）のメンバー構成

タイトル	MiyagiQuest_Bitver.
作品紹介	操作性の悪い従来のスライド方式から、タッチ方式に変えたことで格段に操作性が向上しました。 また、震災復興に少しでも貢献できるように、「センダイ」の町の人のキャラクターデザインとセリフを変更しました。

図表 35　1 ビットからの貢献の作品紹介

1 ビットからの貢献は、高度情報システム学科の 3 年生と 2 年生で構成されたチームである。毎日 Facebook で作業内容を報告し合い、作業の進捗を確認しながら開発を進めていた。また、相互評価でも、メンバー全員がほぼ全ての応募作品の評価を行い、最も積極的に活動していたチームである。

2.2.3 ABCC JK（麻生情報ビジネス専門学校）（規定部門）

No	学科	学年	性別
1	情報工学科	4 年	男性
2	情報工学科	4 年	男性
3	情報工学科	4 年	男性
4	情報工学科	4 年	男性
5	情報工学科	4 年	男性
6	情報工学科	4 年	男性
7	情報工学科	4 年	男性

図表 36　ABCC JK（麻生情報ビジネス専門学校）のメンバー構成

タイトル	MiyagiQuest ABCC version
作品紹介	ミヤギ君のいる世界はみぞうの危機に見舞われた… 人々は疲弊し、いつ終わるのかわからない復興作業に追われる毎日… 疲弊した世界を救うため、ミヤギ君は不思議な力を持つと言われている伝説の秘宝を求めて旅立った。

図表 37　ABCC JK の作品紹介

ABCC JK は、全員が情報工学科の 4 年生であるチームである。登場するモンスター画像を全て差し替えている。また、課題 1 と絡めて奥行きのあるストーリーをしたところが、審査員からの評価が高かった。

2.2.4　宮情IT（宮崎情報ビジネス専門学校）（自由部門、企画書・レポート等）

No	学科	学年	性別
1	高度 IT 専門士学科	3 年	男性
2	高度 IT 専門士学科	3 年	男性
3	高度 IT 専門士学科	3 年	男性
4	高度 IT 専門士学科	3 年	女性
5	高度 IT 専門士学科	3 年	男性

図表 38　宮情 IT（宮崎情報ビジネス専門学校）のメンバー構成

タイトル	[企画書] Pray for Tohoku ～復興を願って～
作品紹介	震災の復興を願った、アプリケーションの企画書です。「東北の星空を通して全国と東北を繋ぐ、ハート ヒューマン コミュニケーション アプリ」とすることが、コンセプトになっています。

図表 39　宮情 IT の作品紹介

宮情 IT は、メンバー全員が高度 IT 専門士学科の 3 年生である。東北の復興を願ったコミュニケーションアプリの企画書を作成し、機能に盛り込まれたアイディアや、復興への願いに対する審査員からの評価が特に高かった。

2.2.5 MOM5days（仙台大原簿記情報公務員専門学校）（自由部門、企画書・レポート等）

No	学科	学年	性別
1	総合情報ビジネス学科	2 年	男性
2	総合情報ビジネス学科	2 年	男性
3	総合情報ビジネス学科	2 年	女性
4	総合情報ビジネス学科	2 年	女性
5	経理事務学科	2 年	女性

図表 40　MOM5days（仙台大原簿記情報公務員専門学校）のメンバー構成

タイトル	スマートフォン向けアプリケーション 企画提案書
作品紹介	震災前の美しい宮城を紹介するアプリ。宮城県を 4 つの地域に分け、それぞれの観光地を画像やレビュー文等で紹介する。

図表 41　MOM5days の作品紹介

MOM5days は、総合情報ビジネス学科と経理事務学科の 2 年生から構成されたチームである。地元の強みを活かし、観光の PR に活用できる内容となっている。また、プレゼンテーションの評価が高かった。

2.2.6　仙台デザイン専門学校（自由部門、キャラクターデザイン）

仙台デザイン専門学校は、26 名全員がグラフィックデザイン学科の 2 年生であり、個人参加であった。

No	性別	作品紹介
1	男性	「9 日後に大地震が来る」と地震予報が出された現代日本。 科学者たちはプレート層のズレによる地震と観測しているが、 実際には地震の神の封印が解かれたのが原因。 今や霊体ではあるが、その危機に偉大な武将「伊達政む…」 ってあれ？女の子？？
2	男性	孤児院のシスター。 荒廃した世界においての癒しの象徴。 優しさと強さの両立。 シスターなのに鎌を使って戦うというギャップ。
3	男性	県内にある竹駒神社をイメージして作成した。 清いイメージのある巫女の姿をベースとしてお稲荷様を描き、 人々への救いを表現した。 神様なので全体的に豪華な和服をデザインした。
4	男性	幼い頃に両親を「悪の根源」に惨殺された娘。 復讐の鬼となりシーフとして生きてきた彼女だったが、「悪の根源」 を倒す旅をしていた男と出会い、彼と行動を共にするうちに、濁って いたはずの彼女の眼に映る世界は徐々に変わっていく…。
5	男性	震災復興のためにMIYAGI内で歌を歌い、街の人たちに元気を あげる女の子。
6	男性	様々な災難を引き起こす化身。 右手で大地震を発生させ、左手からは障気を発生させて人々を 苦しめる。 死者の魂は球体の中におさめられ、一定量を超えると亡者のみが 住むことのできる世界になる。
7	男性	地震を起こす魔王ダイ・ジ・シンを倒すために突然現れた伝説の 幼女。 宮城にあふれる宮城パワー（通称；MP）を使い魔王を倒しに行く。
8	男性	宮城在住の「東原結衣」は普通の女子高生だったが、ささかまの 妖精「かきントス」から受け継いだ力で震災復興に立ち上がる！！ ささかまの妖精の力で変身した「東原結衣」は震災で傷ついた身体を 治療しながら、傷には身体に受けた傷だけではなく心に受けた傷も あることを学びながら進んでいくRPG。
9	男性	とある施設で開発されていたアンドロイド。 がれきの中に埋まっていたが、流れてきた被災者の血液を吸う ことで魂と意思を獲得した。
10	男性	①PON-CO-02 DT-10（ダーテン）← ハラキリ！カイシャク！の方 若者～子供にウケるようなデザインにした。『困ったときに現れる“ダテ”なロボット』をコンセプトとした。 ②海底大王シラー・カーンス 海底を支配する『真海生物』の王。地震を起こしたのもこいつのせい。 だが、かなりマヌケ。 人が行った事もないくらい深い海からはい上がってきた未知なる 生物。敵キャラです。

11	女性	伊達家の子孫という設定の主人公が民警として町の平和と住人を守るために戦うというのをテーマに作りました。 腕章の“天災地変”は、天災の恐ろしさを忘れないために、忘れてはならないという思いを込めて付けています。
12	女性	【味方側デザイン】 “復興” をモチーフとしてデザインし、看護をする救急箱と建設の 意味を込めてスコップを持たせました。 【敵側デザイン】 “火事” をモチーフとしてガスバーナーや炎を纏っている。
13	女性	地震により光を失った MIYAGI CITY を舞台にした RPG。攻略するのは宮城の有名な場所で「明かり＝希望」になっている。 【男性キャラ】 金華山黄金神社をモチーフにした主人公が白い剣で明かりを照らす。 【女性キャラ】 八木山ベニーランドのボスの少女。観覧車と融合してしまっている。 倒すと仲間となり、ヒロイン的な存在になる。
14	女性	伊達政宗をモチーフに描きました。 素早い動きができるようなキャラを描きたかったので、できるだけ 身軽な格好にしました。
15	女性	センダイ CITY を守るために魔法ステッキに選ばれた少年・泉くん。 ステッキを使うことで姿だけでなく身体能力や動体視力が上がり さらに魔法も使えるようになる。
16	女性	名前／猫宮姫（ねこみやひめ） 猫の妖怪、お姫様、お付きに火の玉の妖怪がいる。 昔、首を切られたため頭が浮いており、目は開かない。 大地震の影響で被害にあった妖怪を助けたりしている。 名前／大蛇（おろち） 大蛇の妖怪。捨てられた人形に憑いている。本体は背後の白蛇。 半壊した鞠に乗っている。頭の目玉リボンは猫宮姫の目玉。 大地震で混乱しているスキに悪だくみをする。
17	女性	2011.3.11 の震災により、被災地では化物が出現するようになった。 ダテマサキは行方不明になっている妹を探しつつ、化物を倒し、 被災地の現状の調査をしていた。 そしてムーという守護精霊を連れて今日も次の街へと 向かうのであった。
18	女性	被災地 MIYAGI の各避難所を舞台とした新感覚の避難所生活 RPG。 避難所を拠点としてアイテムを探したり、ボランティアをしたりして経験値を貯めて 1 ヵ月間生活をする。 最終に似ためた経験値に応じて奇跡が起こる…？ 主人公は男女選べるセレクトタイプ。
19	女性	各地に「星の木」を植えながら震災の元になった敵と戦う。 携帯ゲームのようにカードで召喚して戦う（苗も普段はカードに なっている）。 主人公は男・女選ぶことができる。 仙台は「七夕まつり」が有名なので、星をイメージしたデザインに しました。
20	女性	仙台で震災に遭い意識不明になった主人公「神谷大輝」。 目覚めたとき、彼の傍らには 1 匹の喋るクマがいた。 彼（？）が言うには、ここは現実の世界で

		はなく“復興できなかった ”未来の仙台だと言う。 街では死者の怨念なのか得体のしれないものが徘徊している。 クマはこいつらを倒し、復興の手助けをしてほしいと求め、「触れたものを武器にできる」能力をもらった大輝は戦いに挑む。
21	女性	アオバくん 杜都（トート）アカデミーに通っていて、作業用ゴーグルを首に下げ、頭は葉っぱのイメージ。 性格はまっすぐで行動力がある。
22	女性	名前はえだまー。 災害のために作られた調理ロボット。 食べ物を運んだり、口からミネラルウォーターとおいしいずんだ 餅を出してくれる。 苦手なことは会話。好きなことはみんながおいしく食べている時の 笑顔を見ること。
23	女性	杜の都を守る宮城の神様というイメージで作りました。 のんびりしているけれど、震災の時には全力で頑張る、 そんな神様です。
24	女性	幼い姉弟ががんばってボランティアをするファンタジー系のゲーム。 今までにない規模の地震が発生し、ボロボロになってしまった センダイ CITY。建物が崩れ、食料や水がない状態が長く続いていた ため住民たちは苦しんでいた。 いつまで経っても状況が良くならない事に我慢できなくなったオリは 弟のヒコを無理やり連れてセンダイ CITY の住民たちを助けにいく。
25	女性	このゲームは恋愛シュミレーションゲームです。 主人公とその兄が本屋で不思議な生き物を預かったことから、その 生き物の力で異世界へ飛ばされてしまう。 その世界では、人間が本を読まなくなったことで本の中のプリンセスたちがストライキを起こしていた。 主人公たちは全ての本の世界を旅する事になって…？
26	女性	舞台は壮大な時を刻むクロノリア大陸。 すべては大陸の象徴“クロノスの砂時計”によって均衡を保っていた。 しかし、ある日突然生まれた小さな時の歪みは月日と共に大きくなり、 千年の時を経た今、遂にその均衡は崩される。 消えゆく大陸、運命に翻弄される少女。騎士団は全てを取り戻すために立ち上がり、“クロノスの砂時計”の誕生、全ての元凶“逆転の日”の謎 に迫る。

図表 42　仙台デザイン専門学校の作品紹介

仙台デザイン専門学校からは、26 名がそれぞれ個人で制作したキャラクターデザインが応募された。ほとんどの作品で、震災からの復興への思いが込められている。その点が、審査員からの評価も高かった。

2.2.7 UHI（中央情報経理専門学校高崎校）（自由部門、ゲーム）

No	学科	学年	性別
1	高度情報システム学科	3 年	男性
2	高度情報システム学科	3 年	男性
3	高度情報システム学科	3 年	男性

図表 43 UHI（中央情報経理専門学校高崎校）のメンバー構成

タイトル	ReclamationStory
作品紹介	「震災からの復興」というコンセプトを、街（ステージ）の発展という形でシューティングゲームに取り入れています！ 敵を倒して素材（マテリアル）を入手し、街（ステージ）を発展させていくという、既存のシューティングゲームには無い要素が最大のアピールポイントです！ 自機も複数種類から選択でき、各々が異なるパワーアップをしていきます。また、遊び要素で多くの隠しコマンドを実装し、 初心者でも遊べる作りになっています！

図表 44 UHI の作品紹介

UHI は、規定部門に参加した CIC_AI とは別のチームである。全員が高度情報システム学科の 3 年生である。ゲームのコンセプトや、街を発展させていくというアイディアに対する評価が高かった。

2.2.8　1ビットからの貢献（富山情報ビジネス専門学校）（自由部門、業務系アプリ）

No	学科	学年	性別
1	高度情報システム学科	3 年	男性
2	高度情報システム学科	3 年	男性

図表 45　1 ビットからの貢献（富山情報ビジネス専門学校）のメンバー構成

タイトル	就職活動データベースシステム
作品紹介	Visual Basic にて作成した就職活動における報告書等をデータベースに保存できるシステム ・・・なんですが、実行形式にて作成できず Visual Basic 2008 がないと開けないと思います。 こちらの不手際で大変申し訳ありません。詳しくは zip ファイル内の説明書をご覧ください。

図表 46　1 ビットからの貢献の作品紹介

規定部門に参加した富山情報ビジネス専門学校のチーム「1 ビットからの貢献」から、2 名が自由部門にも参加した。身近なところにある課題を発見し、そこからソフトウェアの開発に入ったことの評価が高かった。さらに発展させて欲しいという意見もあった。

2.2.9　技術部（船橋情報ビジネス専門学校）（自由部門、キャラクターデザイン）

No	学科	学年	性別
1	情報ネットワーク科	2 年	女性
2	情報処理科	2 年	男性
3	IT エンジニア科	1 年	男性
4	IT エンジニア科	1 年	女性

図表 47　技術部（船橋情報ビジネス専門学校）のメンバー構成

タイトル	キャラクターデザイン
作品紹介	広い宇宙を舞台にした横スクロールアクション RPG(奥行のある)ゲームのキャラクターデザインです その世界の様々なキャラクター達をデザインしました

図表 48　技術部の作品紹介

技術部は、IT 系の学科に所属する 1 年生と 2 年生から構成されている。短い制作期間の中、55 ものキャラクター画像を提出した。世界観が統一されていた点や、プレゼンテーションで最初にコンセプトの説明があった点などが高く評価された。

2.3 二次審査の様子

二次審査の様子を紹介する。

図表 49　UHI の発表の様子

図表 50　MOM5days の発表の様子

2.4 審査結果

二次審査終了後、直ちにこれまでの審査結果を総合し、入賞チームを選抜した。その結果、入賞チームは以下のようになった。

賞	チーム	学校
規定部門 最優秀作品賞	1 ビットからの貢献	富山情報ビジネス専門学校
自由部門 最優秀作品賞	宮情 IT	宮崎情報ビジネス専門学校
優秀作品賞	CIC_AI	中央情報経理専門学校高崎校
	（個人参加）	仙台デザイン専門学校
	UHI	中央情報経理専門学校高崎校
審査員特別賞	MOM5days	仙台大原簿記情報公務員専門学校

図表 51　入賞チーム

第3章　アンケート

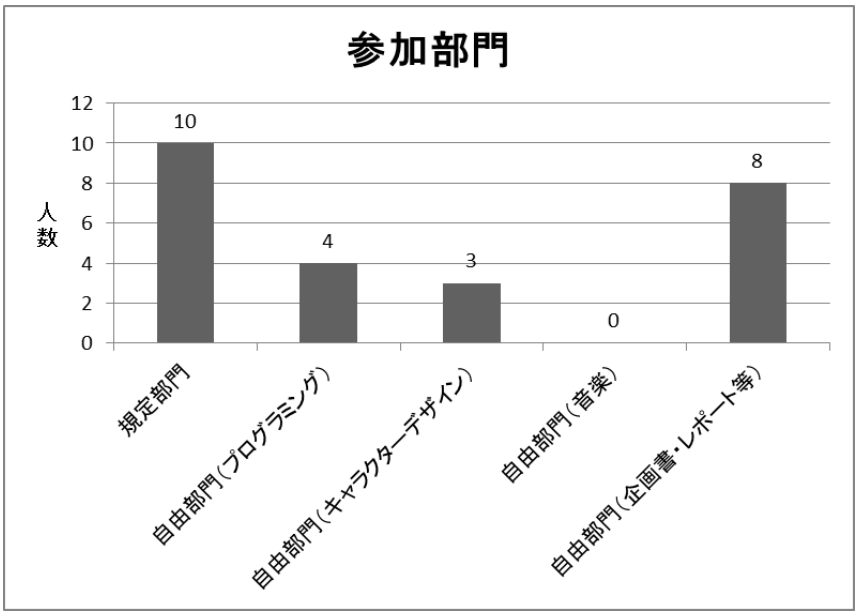
コンテストの作品提出、及び参加者による相互評価を終えた段階で、コンテスト参加者を対象にアンケートを実施した。このアンケートは、本事業で開発した教材や、コンテストの運営等に関して、課題や改善点を明らかにするために実施したものである。なお、アンケート調査票の回収は、Facebook を利用して参加者全員に協力を呼びかけ、それぞれで記入してもらい、Facebook のメッセージ機能で事務局に送信するという方法で行った。その結果、全部で 23 件の回答を得ることができた。

3.1 アンケート結果

3.1.1　回答者のコンテスト参加部門、参加形式

アンケートに先立ち、回答者のコンテストの参加部門、及び参加形式を質問した。まず、参加部門は以下の通りであった。

参加部門	人数
規定部門	10
自由部門（プログラミング）	4
自由部門（キャラクターデザイン）	3
自由部門（音楽）	0
自由部門（企画書・レポート等）	8
合計	23

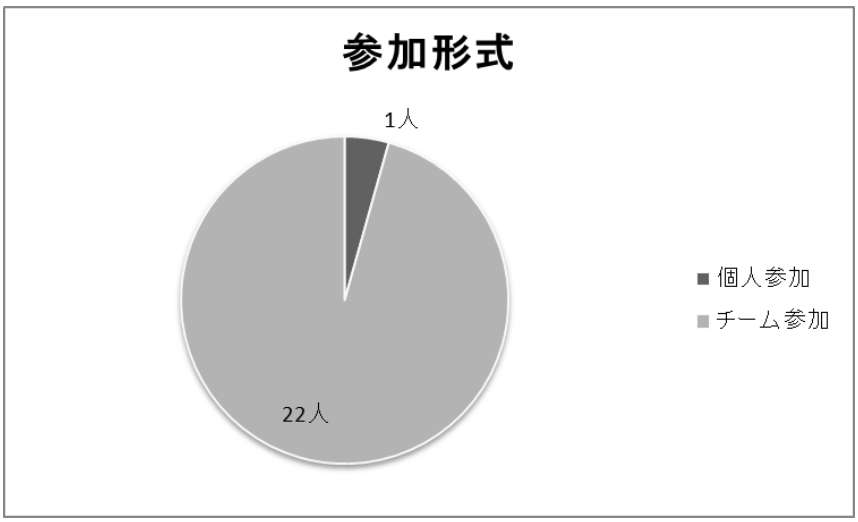


図表 52　参加部門

回答者の参加部門は、規定部門が 10 名、自由部門が 15 名であった。自由部門の内訳は、「プログラミング」4 名、「キャラクターデザイン」3 名、「企画書・レポート等」8 名であった。なお、回答者の中に規定部門と自由部門（プログラミング）の両方に参加した者が 2 名含まれているため、表の合計人数と内訳の人数の合計が合わない。

また、参加形式は以下の通りであった。

参加形式	人数
個人参加	1
チーム参加	22



図表 53　回答者の参加形式

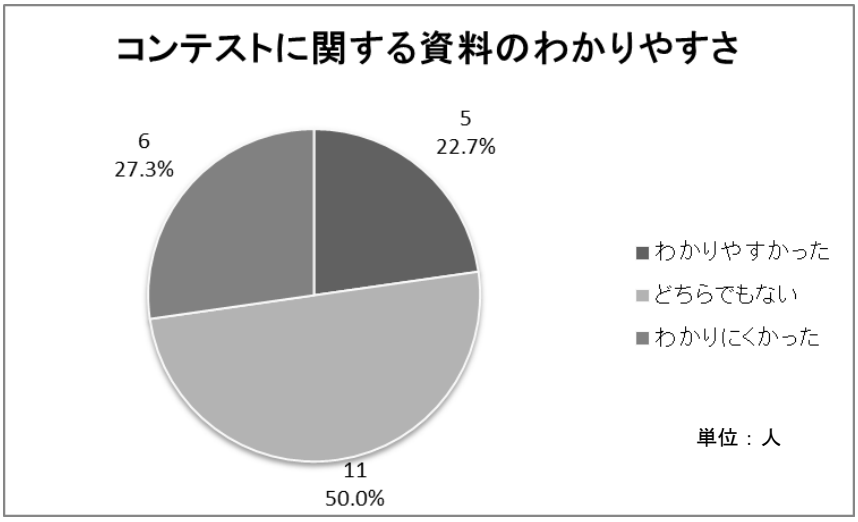
回答者の参加形式は、個人参加が 1 名、チーム参加が 22 名であった。

3.1.2　コンテストに関する資料のわかりやすさ

アンケートの問 1 では、事務局から配布したコンテストの要項や説明資料等のわかりやすさを質問している。

1．事務局から配布されたコンテストに関する資料はわかりやすかったですか。

回答	人数	割合
わかりやすかった	5	22.7%
どちらでもない	11	50.0%
わかりにくかった	6	27.3%
合計	22	



図表 54　コンテストに関する資料のわかりやすさ

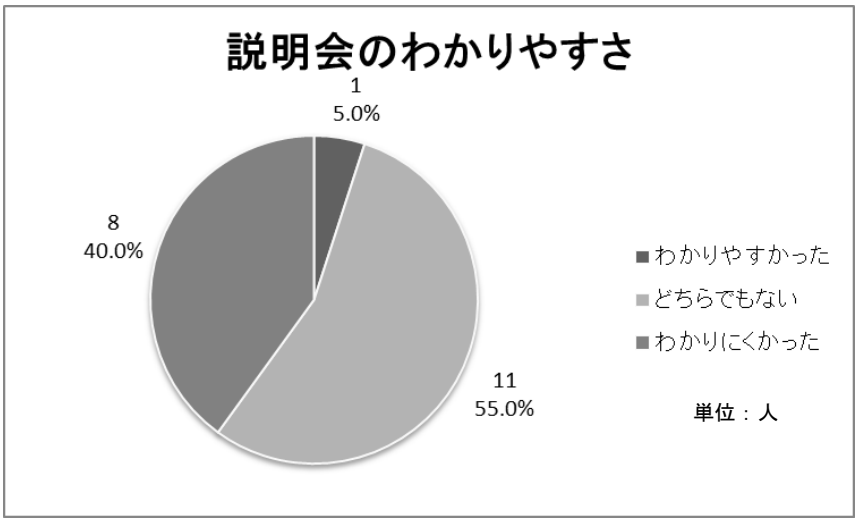
コンテストに関する資料のわかりやすさでは、半数が「どちらでもない」と回答し、「わかりやすかった」「わかりにくかった」で残りをほぼ二等分している。もう少しわかりやすい資料を作成する必要性が感じられる。

3.1.3　説明会のわかりやすさ

問 2 では、Ustream で配信した説明会のわかりやすさについて質問している。

２．Ustream で配信した説明会はわかりやすかったですか。

回答	人数	割合
わかりやすかった	1	5.0%
どちらでもない	11	55.0%
わかりにくかった	8	40.0%
合計	20	



図表 55　説明会のわかりやすさ

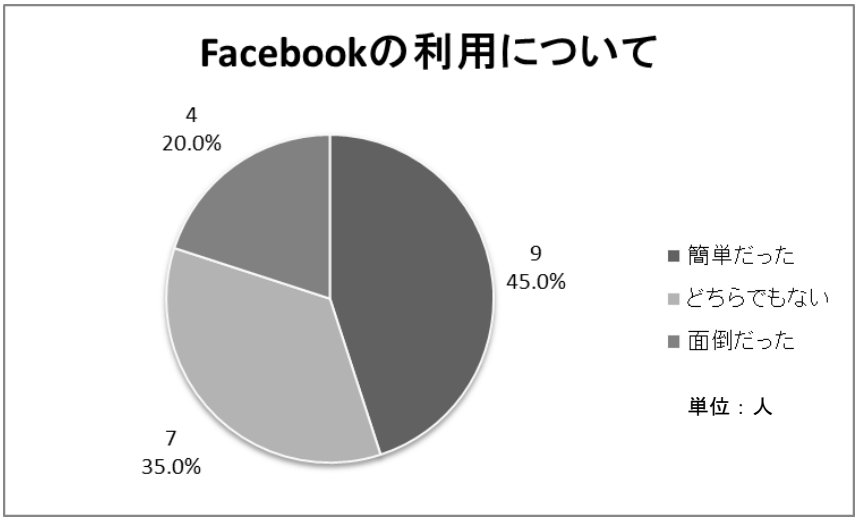
説明会が「わかりやすかった」という回答は 1 名に留まった。半数以上の 11 名は「どちらでもない」と回答し、「わかりにくかった」という回答は 4 割に上った。説明内容や使用機材、配信方式も含めて、再検討することが必要である。

3.1.4　Facebookの利用に関する印象

問 3 では、Facebook を利用した連絡や資料の配付に関する印象を質問した。

3．Facebook を利用したコンテストに関する連絡や資料の配付に関して、どのような印象を持ちましたか。

回答	人数	割合
簡単だった	9	45.0%
どちらでもない	7	35.0%
面倒だった	4	20.0%
合計	20	



図表 56　Facebook の利用に関する印象

Facebook の利用に関しては、「簡単だった」という回答が 9 名（45.0%）と最も多かった。一方で、「面倒だった」という回答は 2 割に達した。簡単だったという回答が比較的多かった。

3.1.5　Facebookの利用に関する感想や意見

問 4 では、前問の結果を掘り下げるために、コンテストで Facebook を利用した点について、感想や意見を自由記述で質問した。

4．コンテストで Facebook を利用した点について、感想や意見等があればご自由にご記入ください。

＜肯定的な意見＞

・今までは敬遠していたが、これを機に登録できてよかった。
・経過報告や、質問などしやすく良かった。
・いろいろな人と知り合いになれて楽しかったです。
・参加、情報の流れを facebook で行うということは、良いアイディアだと思いました。
・Facebook は利用したことが無かったが、操作が難しくなくやりやすいと思った
・情報交換がやりやすいと感じました。
・連絡をすぐに見る事が出来てよかったです。
・分かりやすかったので、気になる点は特にありません
・ログが残るという観点から使う価値はあったと思います。個人的に Facebook はよく使っていますので今回操作に戸惑うことは特になかったのでスムーズに使えたと考えております。
・Facebook を普段使ってたので、簡単にチェックできたのでよかったと思います。
・グループを利用した連絡など、とても良いと思いました。

＜否定的な意見＞

・プライベートのアカウントと同名になるので、使い分けや関連通知の使い分けがとても面倒でプライベートアカウントでは学校名などを公開したくないのに関連で公開することになった。
・Facebook を利用する利点があまりないように思いました。
・Facebook の登録に携帯電話会社のメールアドレスが必要で自分のアドレス（@mopera.net）では登録できず、支給されたアドレスを使って登録したのが不便だった。
・非常に手間がかかりました。

＜両方の意見＞

・未登録だったため、登録できたのは良かったと思います。しかし、可能であるならば複数手段があれば良かったかと思いました。

・今回のコンテストで初めて利用したので、慣れるまで時間がかかってしまいした。Facebook を利用することについては面白い発想だと思います。

肯定的な意見では、「質問などがしやすかった」「ログが残る点が良かった」「知り合いができた」という、Facebook のプラスの面を指摘する意見が見られた。また、コンテスト以前から既に利用中の参加者もいて、普段から慣れているという意見であった。あるいは、「コンテストを機に登録できて良かった」という意見もあった。

一方、否定的な意見では、「プライベートを明らかにしたくない」という意見が見られた。Facebook では実名での登録が前提であり、勤務先や学校等も明かすことになるという点で、まだ抵抗があるものと考えられる。また、登録に関する問題や、操作の手間がかかったという意見もあった。

その他、Facebook 以外の連絡手段があった方が良かったという意見もあった。

3.1.6 開発や制作に当たって苦労した点（個人参加）

問 5 では、個人での参加者を対象に、開発や制作に当たって苦労した点を、自由記述で質問した。

５．（個人参加の方）開発や制作に当たって、どのような点で苦労しましたか。（自由記述） （例：学校の授業が忙しく、制作にあてる時間がなかなか取れなかった。　等）

＜回答＞

・募集からエントリー、作品の提出、相互評価まで、告知されたタイミングからすべての〆切がタイトすぎたと思います。運営側の都合ではなく、参加者に配慮したスケジューリングが必要だったのではないでしょうか？と強く疑問に思います。

コンテスト全体のスケジュールがタイトであったという指摘があった。事業期間の関係もあったが、非常に厳しいスケジュールで実施せざるを得なかった点は、反省すべきである。次回実施する際は、準備期間も含めた全体のスケジュールを見直し、より参加しやすい環境を整えていく必要がある。

3.1.7　チームでの開発や制作に当たって苦労した点（チーム参加）

問 6 では、チームでの参加者を対象に、チームでの開発や制作に当たって苦労した点を、自由記述で質問している。

6．（チーム参加の方）チームでの開発や制作を進めるに当たり、どのような点で苦労しましたか。（自由記述） （例：メンバーの進捗に大きなズレが生じてしまった。メンバーのスケジュールがなかなか合わず、調整が大変だった。　等）

＜連絡、情報共有について＞

・連絡と情報共有。
・メンバー間の情報共有で、テスト段階になって実装を知らされた項目があった為、テスト項目の見直しが発生した。
・メンバーとの進捗度の報告などがあまりなく、全員の情報共有等がうまくできていなかったこと

＜時間、進捗について＞

・メンバー同士の進捗の度合いで、スケジュールがずれ、納期に間に合わせるのが大変だった。
・飛び入りで参加したので、どうしても開発に時間がかけられなかったのが、少し残念です。
・少ない時間の中で内容を濃いものにしていくことが大変でした。
・チームの人数がそこまで多くなかったため作業の進捗が遅めだった。
・少数のチームでの参加だったので、人員が足りず、資料作成が難航しました。"
・全員でソースを追う事に時間がかかってしまった。
・時間に余裕がなくて少し大変でしたが、面白かったです。

＜意見、アイディアのまとめ＞

・全員で一つのものを作り上げるので意見をまとめるのが大変だった
・一人一人の意見を尊重しつつ多くの人に利用して頂ける内容を考えるのが大変だった。
・皆の意見を一つにまとめることが大変だった。
・どんな作品にするのかで意見がなかなかまとまらず、土台が決まるまでが大変だった。
・アイディアを求めるのに苦労しました。
・アイディアがなかなか思い浮かばなかった所が大変でした。

＜メンバーの成果のまとめ＞

・個々の修正をまとめて同期するのが大変だった。

・画像や資料が期限までに中々揃わず完成させるのに苦労した。

＜その他＞

・メンバーのモチベーションと体調の管理

・自分の担当したところが悪い評価だとチーム全体に影響するので、作業時に悩みました。

・初期状態では汎用的なメソッドが少なく、それをいかに汎用化するか、どのようにした方が使いやすくなるか等、再設計する際はとても苦労しました。

・システムを一新した際に起きた、大幅な変更は、それなりに苦労したが、同時に達成感も味わえて、苦労しつつ楽しみながら出来ました。

＜特になし＞

・特にありませんでした。

・特にありません。

「連絡、情報共有」「時間、進捗」「意見、アイディアのまとめ」「メンバーの成果のまとめ」と、チームでの活動に特有の課題で苦労したという意見が多かった。

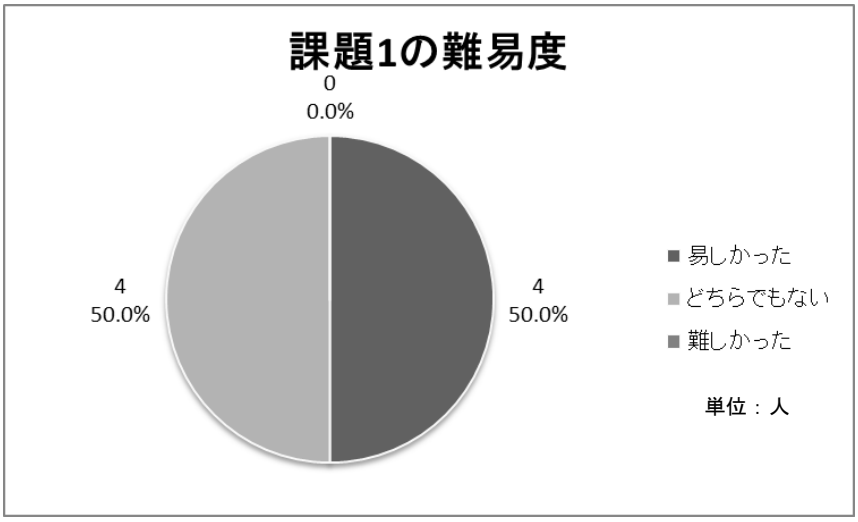
3.1.8　規定部門の課題の難易度（規定部門参加者）

問 7 では、規定部門の 2 つの課題について、難易度がどうだったかを質問した。また、課題に関するコメント等も自由記述で求めた。

7．（規定部門に参加した方）規定部門の課題に取り組んだ印象はどうでしたか。それぞれの課題についてお答えください。また、コメント等あればご自由にご記入ください。

●課題 1「キャラクターの差し替えまたは追加」について

回答	人数	割合
易しかった	4	50.0%
どちらでもない	4	50.0%
難しかった	0	0.0%
合計	8	



図表 57　課題 1 の難易度

＜コメント＞

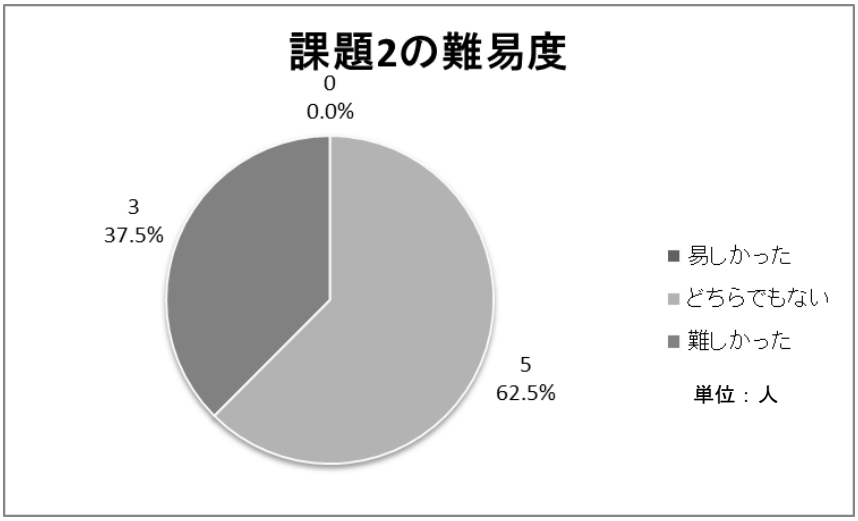
- キャラクターの差し替え、追加に関しては容易だと思ったが、震災復興というテーマが絡んでいたため設定の段階で少し苦労した
- 差し替え自体は簡単だと感じた。追加やバグ修正などを考えると易しくも感じなかった。
- 震災復興というテーマが絡んでいたため設定の段階で苦労しました。
- 今回は操作性に集中していたため追加は行わなかったが、差し替えだけなら簡単だと感じた

・追加、差し替えのための画像の用意や台詞を考えるのが大変でした。
・差し替える画像を用意するのは面倒だったけど、作業自体は易しかったと思います。
・画像を追加すること自体は簡単でしたが、セリフを考えるのが大変でした。

課題 1 の難易度は、「易しかった」と「どちらでもない」が二等分しており、「難しかった」という回答はなかった。しかし、コメントによると、「画像の差し替えは難しくなかった」としているものの、「震災復興のテーマを込めること」等で難しく感じたようである。

●課題 2「操作性の改善」について

回答	人数	割合
易しかった	0	0.0%
どちらでもない	5	62.5%
難しかった	3	37.5%
合計	8	



図表 58　課題 2 の難易度

＜コメント＞
・コーディングももちろん苦労したが、企画フェーズで良い発想が浮かばず苦労した
・固定値ばかり使われていたので動的に変えるのに面倒だった。
・現状の仕様を理解・利用しつつコードを変更していた為、理解するのが難しかった。
・企画段階で良い発想が浮かばず苦労しました。また、テスト段階で既存のバグと新たなバグの区別がつかず苦労しました。
・発想が出る前や出た時は苦労したが、実際にプログラムを組んでいくと意外と思った

通りにできた
・メニューの追加を担当しましたが、操作までは実装できなかったのが残念でした。
・操作性の改善の糸口が見えず難しかったです。
・ソースを追って研究することが大変でした。

課題 2 の難易度では、課題 1 とは対照的な結果となった。「易しかった」という回答はなく、「どちらでもない」が 6 割以上、「難しかった」が 4 割近くであった。コメントからもわかるように、課題 2 は主に機能面の課題であり、プログラミング技術に深く関わる内容であったため、難易度が高かったと考えられる。それでも、苦労しながら何とか改善案を考え、実装したという努力がうかがえる。

3.1.9 参加してみたいカテゴリ（自由部門参加者）

問 8 では、今回自由部門に設定した「プログラミング」「キャラクターデザイン」「音楽」「企画書・レポート等」以外に参加してみたいカテゴリについて、自由記述で質問した。

8．（自由部門に参加した方）今回のコンテストでは、自由部門として「プログラミング」「キャラクターデザイン」「音楽」「企画書・レポート等」という 4 つのサブカテゴリがありました。これ以外に、参加してみたいサブカテゴリにはどのようなものがありますか。（自由記述）

「特になし」や、今回のもの以外には、以下のような回答があった。

＜回答＞
・キャラデザのなかでもイラスト部門があると嬉しいです
・動画部門

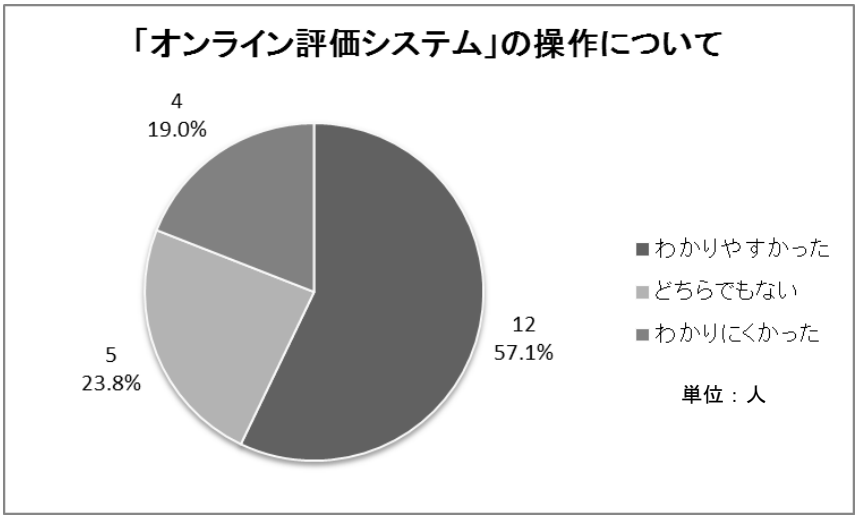
これらに関しては、今後のコンテストを実施していく上で検討する。

3.1.10 「オンライン評価システム」の操作について

問 9 では、作品の提出や相互評価で用いた「オンライン評価システム」の操作について質問した。

９.「オンライン評価システム」での作品の提出や評価の方法は、わかりやすかったですか。

回答	人数	割合
わかりやすかった	12	57.1%
どちらでもない	5	23.8%
わかりにくかった	4	19.0%
合計	21	



図表 59 「オンライン評価システム」の操作について

「オンライン評価システム」の操作については、半数を超える 12 名が「わかりやすかった」と回答している。一方で、「わかりにくかった」は 2 割近くあった。全体的には「わかりやすかった」という回答が多かったが、改善の余地はあると考えられる。

3.1.11 「オンライン評価システム」での相互評価について

問 10 では、「オンライン評価システム」を用いた相互評価に関する意見や感想を、自由記述で求めた。

１０.「オンライン評価システム」で他の応募者の作品を評価することについて、意見や感想を自由に記述してください。（自由記述）

＜相互評価の公平性・妥当性に関する課題について＞

・他の学生の作品を見られるのは良いと思ったが、学生同士の評価がそのまま１次審査の結果になってしまうため、同じ部門に応募したチームに対して評価するのは難しかった

・1 次のオンライン評価では、1 学校からの参加チームが多ければ多い程有利に、少ない学校は逆に不利になるかも知れない要因(贔屓)が発生するのではないか。

・同じ部門で審査しているとどうしても贔屓だとかそういう思いが出てしまうので評価が難しかった。また、評価するのはいいが、作品にかかわっていなければ誰でも評価できるので、同じ学校から自分たちで評価しあえるのでは問題があるのではないかと思った。後、自分の作品を自分で評価している人がいたので自分の作品は評価できないようにするなどが欲しかった

・参加者同士の感想を書けたり読んだりはできてもよいと思いますが、クリエイターとして第三者（一般の人や審査員など、このコンペの関係者ではない人）の公平な評価をしてもらわないと、コンペとして意味がないのではないかと思います。

＜相互評価の効果について＞

・皆さんの考えた作品を見るのがすごく面白かったと思います。

・ネットを通して様々な人の作品を見ることが出来て良かったです

・皆さんのとても素晴らしい作品を評価することができ、とてもよかったです。

・良くも悪くも自分、もしくは自分達で作った作品を多くの人に知ってもらい評価してもらえるシステムは素晴らしいと思おう。

・他のチームや個人の作品を見ることが出来てよかった。

・他の学生の作品を見られて良かったと思います。

・色々な作品があり、どれも凄いと感じました。また、他の人の作品を評価するのは良いと感じました。

・様々な人の感想が入るため正当に評価がされるため良いと思った

・他の応募者の作品を評価することはユーザーの意見を開発者に伝える機会だと思っています。

・自分も評価をもらって色々改善点が見えた所もありましたのでこういった機会は大切に

したいと考えています。

＜評価の基準について＞

・自由部門にはテーマがなかったので、評価する基準が見つけることができなかったので大変だった。
・自由部門がカテゴリの統一がされていなかったので評価が難しかったです。
・キャラクターデザインが多く、評価が難しかったです。
・自由部門の４つのサブカテゴリをどう評価するか悩みました。

＜操作、表示に関する問題＞

・スクリーンショットを一々ダウンロードしなければ見られない所が不便だと思った。
・いわゆる通常のデザインコンペの一般審査がそうであるように、すべての作品を一覧で見ることができるページが必要だと思いました。
・サムネイル機能が必要なかったと思います。

＜その他＞

・評価がぎりぎりになりましたが間に合ってよかったです

「オンライン評価システム」に関して、相互評価そのものや、システムに関する様々な意見が寄せられた。特に、相互評価の公平性・妥当性に関する課題については、本事業で実施した相互評価に関する調査結果でも指摘されていたように、運用の面で評価の客観性を保障する仕組みが必要である。今回はそれを、審査員による二次審査で補うこととしたが、参加者による相互評価と、審査員による評価との相関性を検証し、今後整備していく必要がある。

一方、相互評価の効果について、他の応募者の作品を見ることができる点や、同じような立場の他者（この場合、他の応募者）から評価を受けることでモチベーションにつながったという趣旨の意見があった。

また、自由部門の作品を評価する基準が難しかった、わからなかったという意見もあった。今回は参加者の主観に任せてしまったが、評価の公平性や妥当性を担保するためにも、ある程度参考になるような基準を提示する必要があった。

そして、操作や表示に関しても課題を指摘する意見があった。同様のシステムを研究し、より使いやすいものを備える必要がある。

3.1.12　コンテストに参加して最も勉強になったこと

問 11 では、コンテストに参加したことで最も勉強になったことを、自由記述で質問した。

１１．今回のコンテストに参加して、一番勉強になったことは何ですか。（自由記述）

＜チームでの活動について＞

・連絡・情報共有。
・団結力というものが大切だと思ったことです。
・たくさんの人と一つの大きなものを作るというのが難しくもあり楽しかったです
・仲間と協力すること。
・メンバー同士で情報交換しながら、プログラムを作っていく事が非常に難しいと感じました。
・ソースを追って研究、リーダーとしての責任感を感じる事が一番勉強になりました。
・チームで協力することです。

＜プロジェクトの関わり方＞

・プロジェクトで開発をするという経験が浅いため、経験を積めたという点についてはよかったと思う。
・企画やテストの担当をしたのが初めてだったので、作業工程を知ることができ、勉強になりました。
・プロジェクト単位での作業の仕方
・もっと時間に余裕をもちたいと思いました。
・時間配分は特に気をつける事。
・ゲームを作るのは勿論の事、企画する大変さも勉強になりました。
・拡張性を高めた状態で作成する事。

＜個別技術＞

・プログラミング技術、画像編集
・プログラミング

＜相互評価、他者のレベル等＞

・他校の作品を見る事が出来て、とても勉強になりました。
・人に評価されるものを作製する事がとても難しいことだと分かり、勉強になりました。
・ほかの学校の人たちがどのようなレベルであるのかが分からなかったので、今回のコンテストで自分の大体の位置が分かりました。

＜宮城県について＞

・宮城の有名な土地や物など勉強になった

・宮城県各地の名所・特産物のこと。

＜その他＞

・締め切りまでに描き、いろんな人の作品を見て評価する。自分の思いを書く勉強になれました。

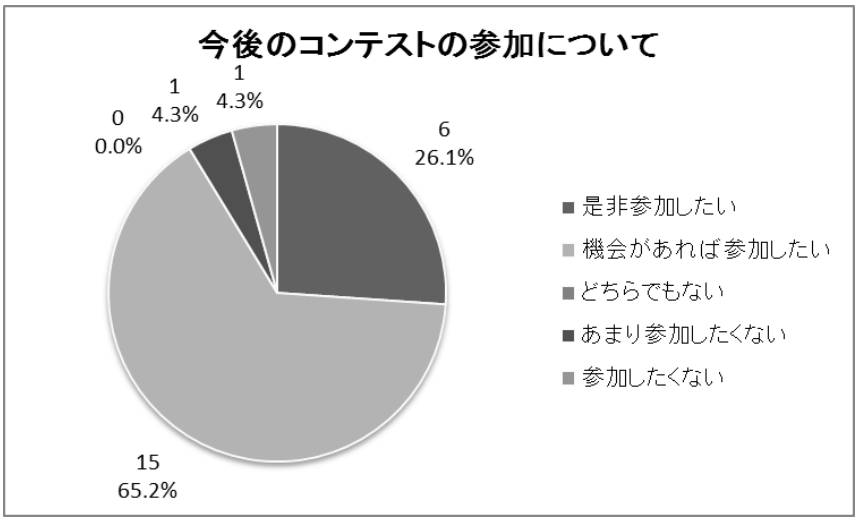
情報共有やチームワーク、企画力、タイムマネジメント等、個別の技術要素以外の、管理技術やパーソナルスキルの重要性に気付いたという意見が多かった。また、震災からの復興をテーマにしたことからか、宮城県について勉強になったという意見もあった。

3.1.13　今後のコンテストの参加について

問 12 では、今後も今回のようなコンテストへ参加意欲があるかを質問した。

１２．今後も、今回のようなコンテストに参加したいですか。

回答	人数	割合
是非参加したい	6	26.1%
機会があれば参加したい	15	65.2%
どちらでもない	0	0.0%
あまり参加したくない	1	4.3%
参加したくない	1	4.3%
合計	23	



図表 60　今後のコンテストの参加について

「是非参加したい」「機会があれば参加したい」を合わせると、9 割以上が「参加したい」という意欲を見せている。コンテストが開発や制作のモチベーションになるということが再認識できた。

3.1.14　コンテスト全体に関する感想、意見等

問 13 では、コンテスト全体に関する感想や意見等を、自由記述で求めた。

１３．コンテスト全体について感想、意見等があれば、ご自由にご記入ください。

＜規定部門についての指摘＞

・規定部門に参加したチームが３チームしかないのが残念だった。他校と競争できるよい機会だと思うので、次回からはもっと参加校が増えるような努力をしてほしい。

・規定部門に参加したチームが３チームしかないのが残念だった。他校と切磋琢磨できるよい機会だと思うので、次回からはもっと参加チームが増えていたら良いと思います。

＜自由部門についての指摘＞

・次回はキャラクターデザインは分けてしまった方がいいと思う

・自由部門の幅が広すぎるので、もう少し細分化した方が、評価がし易くなるので、キャラデザイン部門プログラム部門など、分けた方がいいと思います。

＜「オンライン評価システム」、相互評価についての指摘＞

・オンライン評価システムの操作性があまり良くなかった。1 度提出や評価をしてしまうと間違えていたとしても修正や削除ができないので、無駄に多く投稿してしまう可能性があった。また、作品の評価をしようとすると、説明文が見えなくなるのでちょっと分かりづらいと感じた。また、学生が評価するのであれば同じ学校からは評価できないなど公平性を保てるような仕組みにしたらいいと思った。

・オンライン評価システムで､ユーザーの所属しているチームが分かると良いと思います。

＜制作期間についての指摘＞

・個人的には、もう少し時間があれば改善出来た点などがあったように思えた。(改良：シナリオや仕様の変更・追加など)

・制作期間がもう少し長ければ良かったと思います。

＜その他の要望＞

・連絡に SNS を使いのは新鮮だったが Facebook などの実名などを使用したりするものを、使うのであればこのコンテストには今後あまり参加したくないと思った。

・２次審査が各チーム１人のプレゼンなので、２次審査の様子を動画でアップしてもらえればいいと思います。

＜コンテストへの参加意欲について＞

・楽しかったです。今度は千葉県でもやってほしいです。
・コンテストに参加する、ということだけでもいい経験になれました。これからも、機会があれば参加していきたいと思います。
・面白いコンテストでした。またしたいです。
・大変でしたが、面白く出来ました。また機会があれば参加したいと思います。
・このようなコンテストに参加出来たことはとても良い経験になるので、今後もこのようなコンテストがあれば参加していきたいです。

＜他者の作品について＞
・色々な作品があって面白いなと思いました。皆の個性が伝わっていて良かったなとおもいました。
・他の作品を見ることが出来て、参考になりました。"
・今回のコンテストでは学生同士での評価ができたのでとても良い経験が出来たと感じております。

＜その他＞
・震災の為に自分達が貢献出来る事はさほど多くはありません。なので、この様な形で少しでも復興に貢献出来る事を嬉しく思います。
・Android について色々と分かったのでいい機会だった。
・今回もこのようなコンテストを開いていただきありがとうございました。
・普段なかなかできない良い体験ができました。
・とても良い経験になりました。
・楽しかったです。

コンテストの部門や相互評価、期間等に関して様々な意見が寄せられた。特に部門や評価の方法、期間に関しては十分に検討し、次年度以降のコンテスト実施につなげていきたい。

第4章　実証実験のまとめ

本事業では、「震災復興に貢献するモバイルアプリケーション等コンテスト」を実証実験として実施した。このコンテストを終えて得られた成果と、今後の課題として検討すべきことを述べる。

まず、コンテストの応募作品として、規定部門、自由部門、ともにレベルの高い作品がそろった。特に、「震災からの復興」を意識した作品が多数応募されたことは、震災復興を支援する人材の育成という事業の目的とも合致し、この点では非常に大きな成果である。本事業で策定したモバイルアプリケーション開発人材の評価基準でも、開発技術や管理技術とは別に災害対応スキルを設定し、その中に災害対応マインドを設けた。これは、「防災、復旧・復興に自分の身につけている専門知識やスキルを役立てようとするマインド」のことである。本実証実験によって、開発した教育プログラムが災害対応マインドの醸成に効果があるということが確認できた。

また、コンテストのアンケート結果によると、プログラミング等の個別の技術要素のみならず、チームワークやタイムマネジメント等、管理技術やパーソナルスキルが勉強になったという回答が多数寄せられている。これによって、本教育プログラムが、開発技術や管理技術を基とする実践力と、パーソナルスキルの向上が見込めるということがわかった。

一方、課題としては、まず、コンテストの一次審査に取り入れた相互評価に関してである。調査で明らかになったように、参加者のモチベーションや、他の作品を参考にすることで自らの技術向上に活用するという点で高い効果が見られた。しかし、評価の公平性や妥当性という点では、今後も検証が必要である。今回は審査員による評価でその部分を補ってはいるが、学生の生み出した成果に対する評価として活用していくためには、先行事例も含めて、さらなる研究・検証を続けていく必要がある。

また、スケジュールや部門の設定等、運営面での課題は多い。アンケート結果にあるように、学生のコンテスト参加の意欲は非常に高かった。先述のような教育効果を活かすためにも、十分な時間をかけて準備を進め、より大きな成果を目指していきたい。

次年度以降は、今年度のコンテストを改善、発展させ、再び被災県周辺で開催する予定である。参加者は、被災県周辺を中心に全国の専門学校や大学等から広く募集し、成果発表会も被災県にて実施する。

また、モバイルアプリケーション開発人材の研修プログラムを収集、開発して、被災県周辺を中心に、全国の専門学校や大学、企業等に提供する。さらには、本事業で策定した評価基準も、「レベル 0」から「レベル 1」以上に広げ、精度を高めていく予定である。

巻末資料

巻末資料 1：震災・防災関連アプリケーションの具体的な事例集
巻末資料 2：Android アプリケーション開発環境構築マニュアル
巻末資料 3：Android ゲームアプリケーション　ロールプレイングゲーム解説
巻末資料 4：MiyagiQuest 追加機能　仕様解説書
巻末資料 5：オンライン評価システム　利用マニュアル

巻末資料１　震災・防災関連アプリケーションの具体的な事例集[19]

学校法人 宮崎総合学院
宮崎情報ビジネス専門学校

震災・防災関連アプリケーション事例集

「東日本大震災からの復旧・復興を担う専門人材育成支援事業」事業計画書
添付資料

19 本資料は本事業の計画書を提出した際の添付資料と同一のものである。

計画停電情報 with 節電メーター

作者：BTD STUDIO Co.,ltd.
最新バージョン：1.2.0
最終更新：2011年7月4日
Android要件：1.6以上

インストール数：10,000-50,000
サイズ：301KB
価格：無料

- 特徴
  - 首都圏の最新の消費電力と、計画停電の情報がチェックできる。
  - 都道府県→市町村→○丁目の3段階で停電情報の範囲を絞り込んで表示可能。
  - 停電情報をメール送信可能。
  - 一週間分の停電予定が一覧表示できる。

なまず速報β

作者：Stridge Project Inc.
最新バージョン：0.5.8
最終更新：2011年5月19日
Android要件：1.6以上

インストール数：
1,000,000-5,000,000
サイズ：431KB
価格：無料

- 特徴
  - 地震速報をプッシュ通知するアプリ。
  - 震源地までの距離やマグニチュード等で、通知する条件を設定できる。
  - キャリアの緊急地震速報がスマートフォンに実装される前から公開されている。
  - キャリアやテレビの緊急地震速報より早く通知が届くこともある。

東日本大震災情報

作者：osd
最新バージョン：1.0
最終更新：2011年3月18日
Android要件：1.6以上

インストール数：10,000-50,000
サイズ：18KB
価格：無料

- 特徴
  - GoogleMapコミュニティーの有志によって作られた、炊き出し実施場所、緊急避難場所、給水場所、携帯電話の充電可能場所などをマップ上で表したり、その場所までナビゲーションを行なったりする。
  - Googleの震災関連ページの中の安否情報をブラウザ上で表示。
  - Yahoo! Japanの震災関連情報をブラウザ上で表示。

ベクレルコンバーター

作者：JungleCruise Software
最新バージョン：1.3
最終更新：2011年4月2日
Android要件：1.6以上

インストール数：1,000-5,000
サイズ：23KB
価格：無料

- 特徴
  - ベクレル値（放射性物質が放射線を出す能力）をシーベルト値（放射線による人体への影響度合い）に変換して、実際、どのくらい身体に影響があるのか算出できる。

災害用伝言板

学校法人 宮崎総合学院
宮崎情報ビジネス専門学校

作者：ソフトバンクモバイル株式会社
最新バージョン：1.0.2
最終更新：2011年9月12日
Android要件：2.2以上

インストール数：10,000-50,000
サイズ：103KB
価格：無料

- 特徴
  - ソフトバンクモバイルが提供する災害用伝言板サービスへの接続専用アプリ。
  - ソフトバンクモバイルのAndroidスマートフォンのみにインストールでき、災害用伝言板を利用する際のパケット通信料は無料。

※NTTドコモ用、KDDI用は非公式版があり、キャリアによらず利用できる。

Androidアプリケーション
開発環境構築マニュアル

Ver. 2.00.00

更新履歴

Version	変更内容
1.00.00	初版発行
2.00.00	2012 年 3 月時点での SDK 等の最新バージョンに対応

目　次

1. はじめに

本書では、Android のアプリケーションを開発するに当たり、事前に準備しておかなければいけない開発環境の構築方法について記述しています。

1.1 インストール環境

本書では、以下の環境を使用しています。

OS	Windows Vista™ Business Service Pack 2
ブラウザ	Internet Explorer 8／Firefox 11.0
Android SDK	Android SDK
JDK	Java 7u3
Eclipse	Eclipse 3.7 Indigo Pleiades All in One

使用している環境によっては、本書に記載されている画面、メッセージとは異なっている場合がありますので、ご了承下さい。

1.2 対象読者

本書が対象にしている読者は、これから Android のアプリケーションを開発しようと考えている方を対象としています。

本書の記述範囲は、Android アプリケーションの開発環境構築であるため、必要とするスキルとしては、基本的な Windows の操作を行うことが出来れば十分なのですが、本書記述内容に基づき操作を行った場合でも、ネットワーク環境などにより、問題が発生する可能性はあります。その場合には、問題内容に応じた知識により、問題解決しなければいけません。

1.3 免責事項

本書に掲載する情報には、充分に注意を払って作成していますが、その内容について保証するものではありません。また、特定非営利活動法人教育支援システム研究機構は、本書の内容の使用ならびに閲覧によって生じたいかなる損害にも責任を負いかねますので、ご了承ください。

なお、本書の内容については、予告なく変更される場合があります。

2. Android とは？

Android（アンドロイド）とは、インターネット検索サービス大手である米グーグル社（Google Inc.）を中心として、台湾の HTC や Motorola などの端末メーカー、NTT ドコモや KDDI、T-Mobile といった通信事業者や、Intel や Texas Instruments などの半導体メーカーなどの会社が加盟する業界団体「Open Handset Alliance」（OHA）が発表した携帯電話用ソフトウェアプラットフォームです。

Android は、スマートフォンや小さな画面の携帯電話の両方をターゲットにしたソフトウェア群であり、OS やミドルウェア、ユーザインターフェイス、それに Web ブラウザや、カレンダー、電話帳、地図といったアプリケーションなど、次世代以降のスマートフォンに必要と考えられている携帯電話用ソフトウェアの多くはこの中に含まれています。

Android は、完全にオープンなライセンスで公開されていることが特徴で、どのメーカーやキャリアでも、自由に Android という環境を携帯電話端末に搭載して販売することが出来ます。

ただし、プラットフォームそのものはオープンで、かつフリーであっても、採用するメーカーが改変したり、端末そのものをオープンにしたりはせず、サードパーティアプリの利用を認めないなどの改造も可能となっています。

2.1 OHA（Open Handset Alliance）とは？

OHA（Open Handset Alliance）とは、2007 年 11 月に米グーグル社の呼びかけにより設立された携帯電話における共通のソフトウェアプラットフォームの開発および普及を推進する業界団体のことです。

OHA では、米グーグル社が推進する携帯電話端末向けのソフトウェア実行環境「Android」を基盤に、対応端末の開発や、関連するソフトウェアのサービスの普及などに取り組んでいます。

OHA は、緩やかな連合であり、Android 端末や関連サービスのリリースが義務化されているわけではないため、参加企業が、将来必ずしも Android への対応を行うとは限りません。

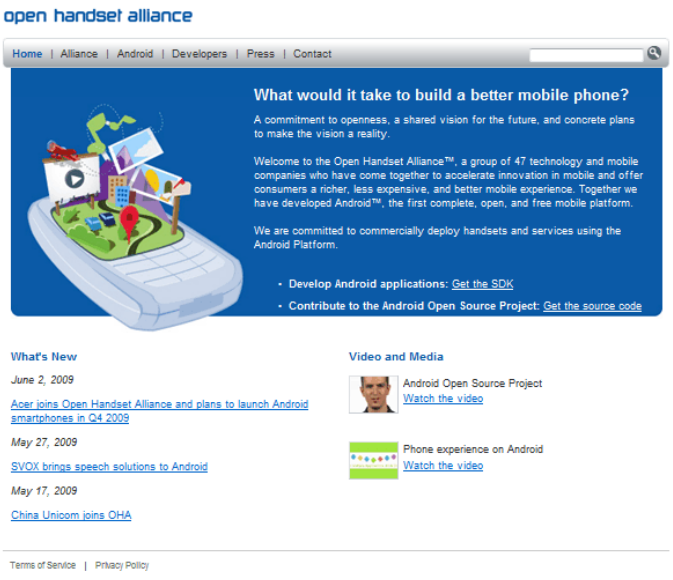


図 1 Open Handset Alliance HP

2.2 Android の内部構成

OS は、Linux カーネルをベースとし、その上に、ミドルウェアとして、MPEG4 や H.264、AAC、MP3 などの動画音声コーデックや、JPEG、PNG、GIF などの画像ライブラリ、OpenGL ES 1.0 対応の 3D グラフィックライブラリ、Bluetooth や無線 LAN に対応するソフトウェアスタック、SQL データベースエンジンの SQLite、JavaVM などのランタイム環境、アプリケーションマネージャなどが搭載され、さらに Webkit エンジンを搭載した Web ブラウザ、カレンダー、電話帳、地図などのアプリケーション、そして多くのサードパーティアプリケーションソフトなどが動作することになります。

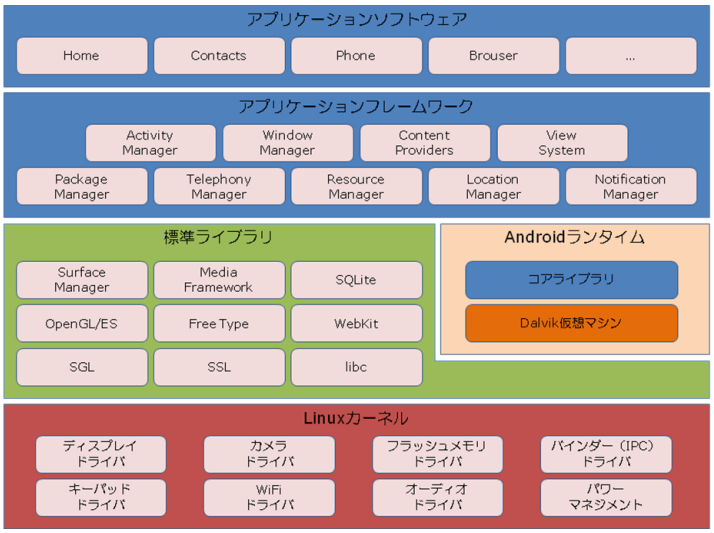


図 2 Android アーキテクチャ

Android 向けのアプリケーションは、すべて Java プログラミング言語で作成されています。例えば、前述の Web ブラウザや、カレンダー、電話帳、地図なども、すべて Java で作成されています。開発キットは、一般のユーザーにもフリーで公開されているので、誰もがアプリケーションを開発することが出来ます。

2.3 Android アプリケーション作成フロー

Android は、アプリケーションを Java、それも Java ME（Java Platform, Micro Edition）では無く、Java SE（Java Platform, Standard Edition）で開発出来るのが、開発者にとっての最大の特徴となっています。

開発者は、Java でアプリケーションを作成し、クラスファイルにコンパイルし、その後、コンパイルしたクラスファイルを Dalvik 実行形式にリコンパイルすることによって、Dalvik VM 上で動作させることが出来るようになります。Android で Dalvik 実行形式を動作させるためには、パッケージングする必要があります。

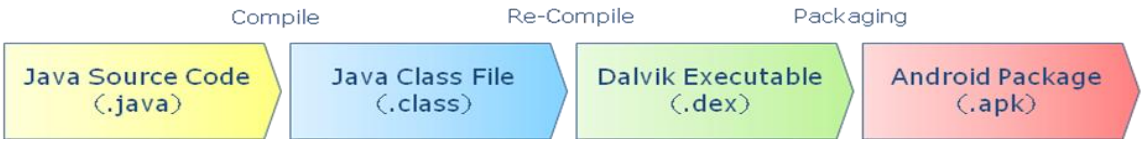


図 3 Android アプリケーション作成フロー

なお、携帯電話端末上でアプリケーションを実行させる場合には、最後のパッケージングの際に、署名ファイル（.keystore）を付加しなければいけません。（本書では、署名方法についての説明はありません）

3. Android アプリケーション開発環境の準備

Android アプリケーションは、Windows や Mac OS X、Linux などの OS で開発することが出来ます。本書では、Windows 環境での開発を行うことを前提として説明を行います。なお、Eclipse プラグイン「Android Development Tools Plug-in for the Eclipse IDE」を利用し、Eclipse 上で開発するのであれば、どの OS であっても開発方法に大きな違いはありません。

3.1 JDK・JRE のダウンロードとインストール

Eclipse を動作させるためには、JDK（Java Development Kit）か JRE（Java Runtime Environment）が必須となります。このため、事前に、インストールしておく必要があります。

ここでは、本書作成時点での最新版である Java 7u3 を使います。
以下の URL にアクセスして下さい。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

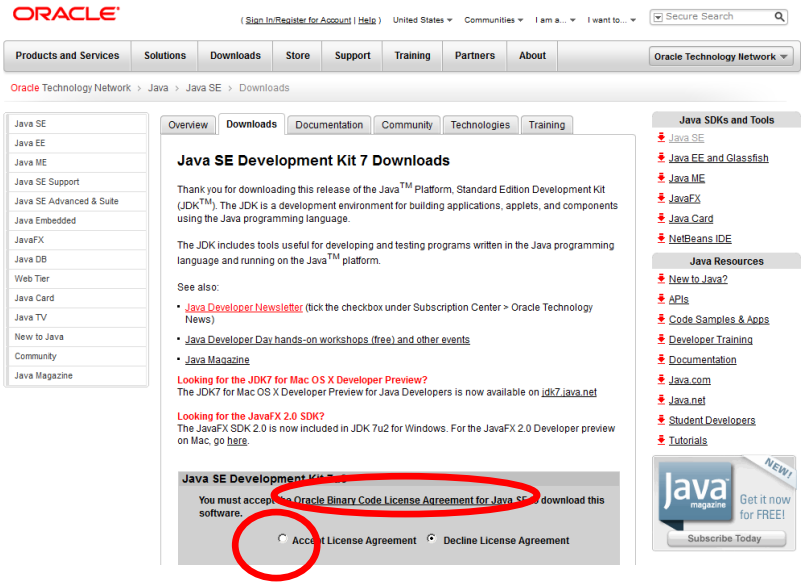
上記の URL にアクセスすると、下図のような画面が表示されます。

画面左側にある Java platform(JDK)を選択します。ボタンを押下すると、次画面に遷移します。

> JRE のみをインストールしたい場合には、上図のボタン下側にある「>>JRE」を選択し、インストールしても構いません。基本的なインストール手順は同じです。
> ちなみに、ボタン下側にある「>>JDK」は、ボタンでの動作と同じとなります。

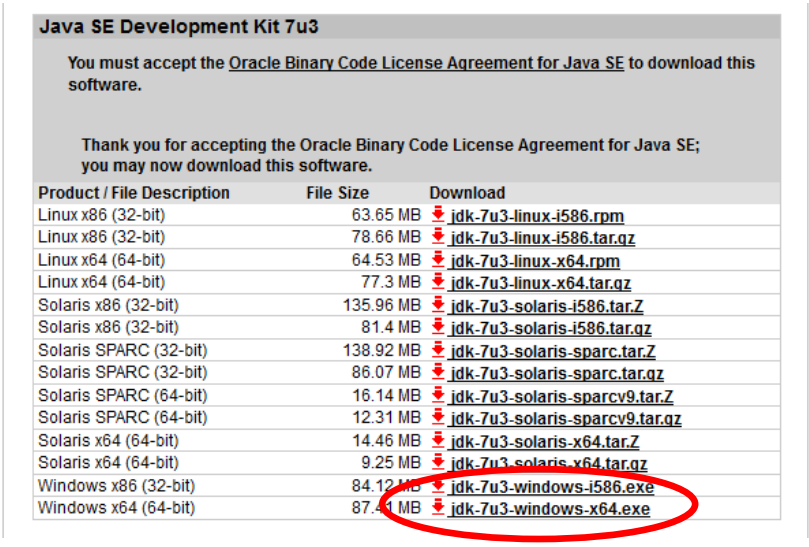
画面遷移後、使用する環境に合ったファイルをダウンロードします。その際、「Oracle Binary Code License Agreement for Java SE」という使用条件に同意しなければなりません。そのために

は、ページの上の方にある「Java SE Development Kit 7u3」のダウンロードファイルが並んでいるリストの上にあるラジオボタンを、「Accept License Agreement」にセットします。「Oracle Binary Code License Agreement for Java SE」は全文英語で記述されていますが、できるだけ目を通しておきましょう。

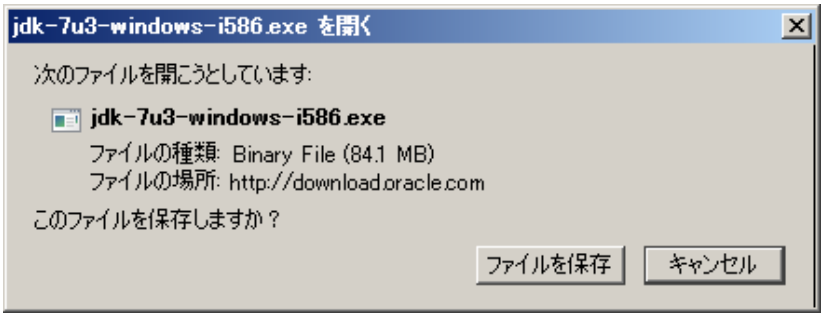


ファイルをダウンロードする前には、「Oracle Binary Code License Agreement for Java SE」のリンク先のライセンスを、必ず見るようにしましょう。
日本人は、比較的ライセンスに疎いと言われており、特に英語のライセンスを見ない人が多いのですが、使用するソフトウェアを、どのような場合に使って良いのかなどの条項が書かれています。もし、間違った使い方をした場合には、ライセンス違反として訴えられることもあるのです。このため、最初のうちは、難しいかも知れませんが、必ず見るクセを付けるようにしましょう。

次に、ラジオボタンの下にあるリストから、使用環境に合ったファイルをダウンロードします。Windows の 32bit 版であれば「jdk-7u3-windows-i586.exe」を、Windows の 32bit 版であれば「jdk-7u3-windows-x64.exe」をクリックします。



あとは、保存場所を指定してファイルをダウンロードします。ファイルサイズは 84.12MB（32bit 版）～87.41MB（64bit 版）もあるので、ダウンロードには少し時間がかかります。

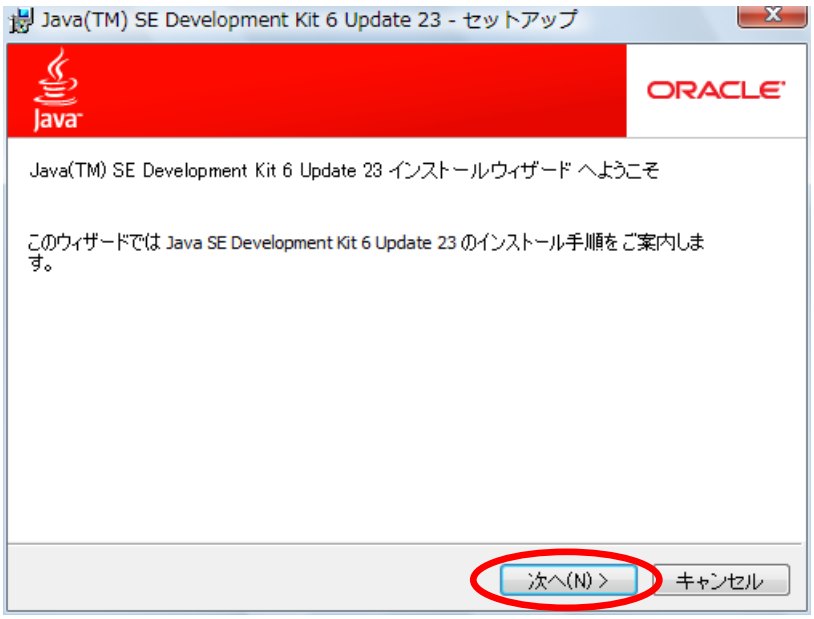


ファイルを保存せずに、そのままインストールしても構いません。その場合には、次へ進んで下さい。ただし、何らかの理由により、インストールが失敗した場合には、再度、インストールし直す必要があります。

ダウンロード後、ダウンロードしたファイルを実行します。セットアップウィザードにより、それほど迷うことは無いと思いますが、ここでは、順次説明していきたいと思います。

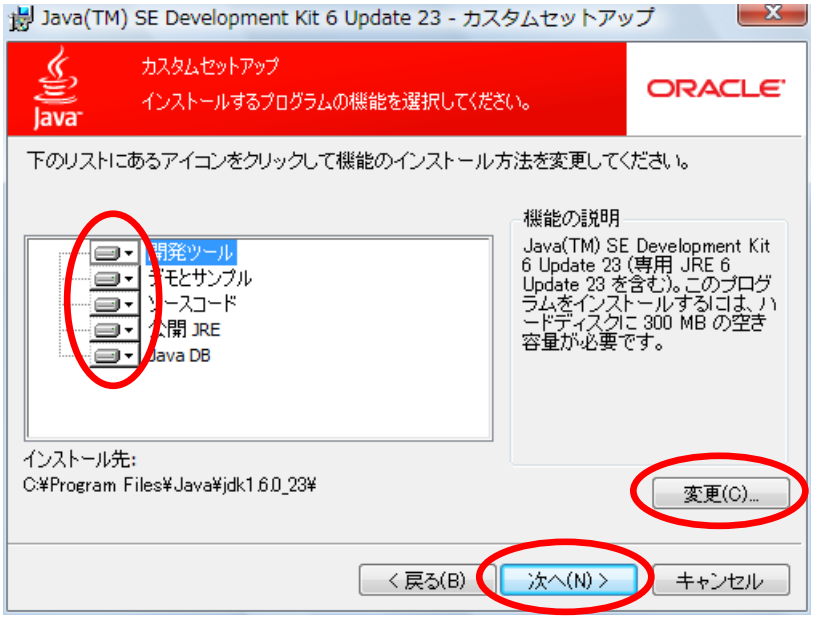
ダウンロードしたファイルを起動したときに、Windows のアカウント制御により、メッセージが表示される場合がありますが、その場合には、「続行」ボタンを押下して、処理を進めるようにして下さい。

※以下の画面は Java SE Development Kit 6 Update 23 のものですが、操作は基本的には同じです。



上図のような画面が表示されましたら、「次へ」ボタンを押下して下さい。

インストール先やインストールするプログラムの機能の選択画面が表示されます。

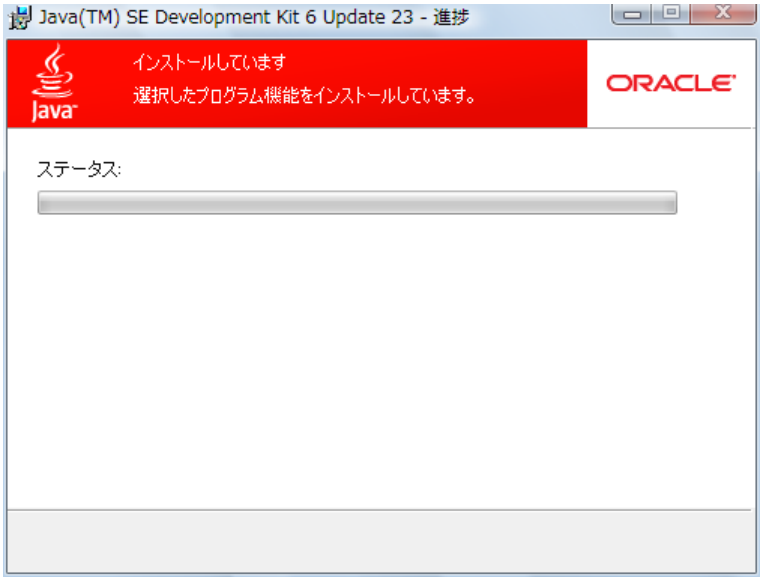


必要に応じ、インストールする機能を選択して下さい。基本的には、デフォルトのままで構いません。初期時には、すべてのプログラムの機能が選択された状態になっていますので、もし、不要なものがあるようであれば、各名称の前にあるアイコンを選択し、選択を解除して下さい。例えば、サンプルコードなどが不要な場合には、「デモとサンプル」の選択を解除するようにします。

それぞれの機能を選択すると、ダイアログ右側の機能の説明に、機能の説明と、必要となるディスク容量が表示されますので、参考にして下さい。

また、JDK のインストール先を変更する場合には、画面右下にある「変更」ボタンを押下し、インストール先のフォルダを指定して下さい。設定終了後（変更が無い場合にはそのまま）、「次へ」ボタンを押下して下さい。

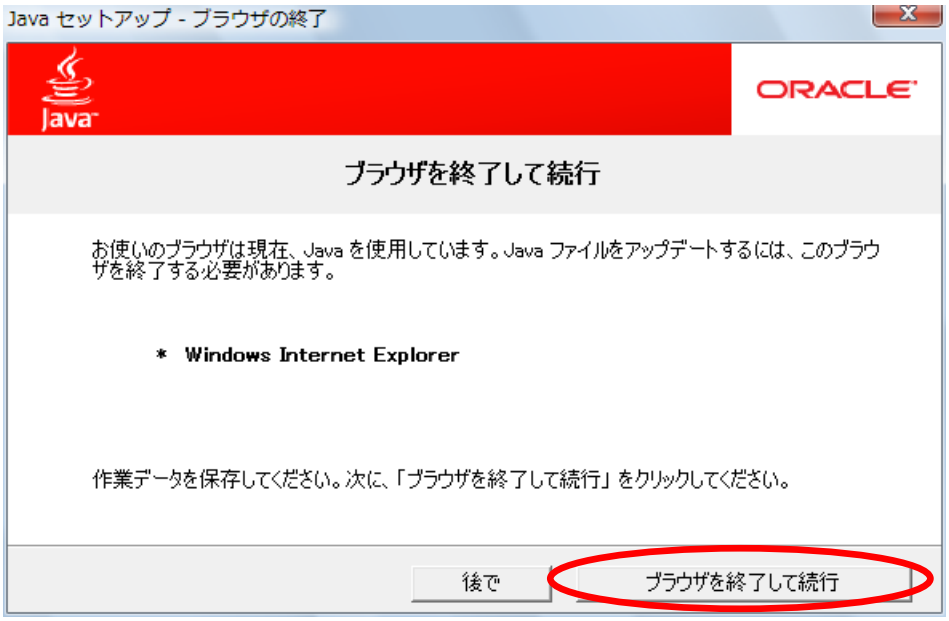
「次へ」ボタン押下後、インストールが開始されます。進捗状況は、画面にプログレスバーで表示されますので、しばらくお待ち下さい。



インストール後、JRE のインストール先フォルダの確認ダイアログが表示されます。

変更したい場合には、「変更」ボタンを押下し、インストール先フォルダを指定し直して下さい。
インストール先フォルダ指定後、「次へ」ボタンを押下し、次に進みます。

インストール開始時に、ブラウザを利用しており、かつそのブラウザが Java を使用している場合には、下図のようなメッセージが表示されます。
もし、作業中（例えば、何らかの入力を行っているような場合）のブラウザがある場合には、作業内容を保存した後で、「ブラウザを終了して続行」ボタンを押下して下さい。
このボタンを押下すると、強制的にブラウザを終了してしまうため、データが保存されることはありませんので、ご注意ください。もちろん、保存する必要が無い場合には、すぐに「ブラウザを終了して続行」ボタンを押下し、インストール作業を続行しても構いません。



再度、インストールが開始されます。

進捗状況は、画面にプログレスバーで表示されますので、しばらくお待ち下さい。

下図のダイアログが表示され、インストールが完了となります。

「完了」ボタンを押下して、終了となります。

3.1.1 JRE のダウンロードとインストールでの注意事項

先の説明では、JDK のインストール方法を示していますが、JRE も基本的に同じようにインストールすることが出来ます。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

上記の URL にアクセスし、対象となる OS のファイルをダウンロードし、インストールを進めて下さい。

なお、Windows 向けとして、オンライン版とオフライン版の 2 つのファイルが用意されています。

オンライン版は、最初に最低限のファイルをダウンロードし、その後に必要なファイルを、インターネットを通じ、順次ダウンロードしながらインストールします。インストール時には、インターネットへの接続が必要となりますが、インターネット接続回線の帯域が狭いような場合には、ダウンロードするファイル容量が小さいため、結果的にインストールを早く終わらせることが出来る場合があります。

ただし、企業や学校など大量の PC などへ展開するような場合には、すべての PC が一気にファイルをダウンロードすることになるため、ネットワークへの負荷が高くなってしまいますので、ご注意下さい。

オフライン版は、インストールするために必要なファイルがすべて含まれており、インターネット接続が無くてもインストールすることが可能となります。

3.2 Eclipse のダウンロードとインストール

Eclipse はもともと英語のソフトウェアですが、ここでは、日本語化されている「Eclipse 3.7 Indigo Pleiades All in One」を使います。

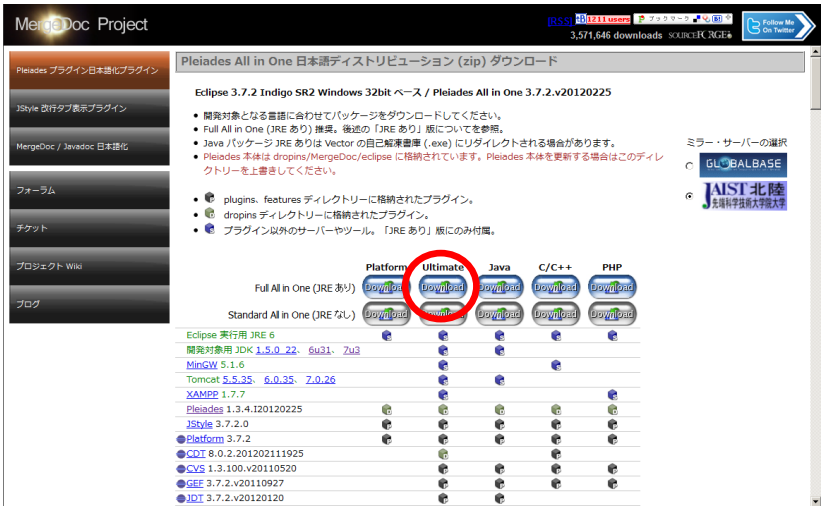
http://mergedoc.sourceforge.jp/

上記の URL にアクセスすると、下図のような画面が表示されます。

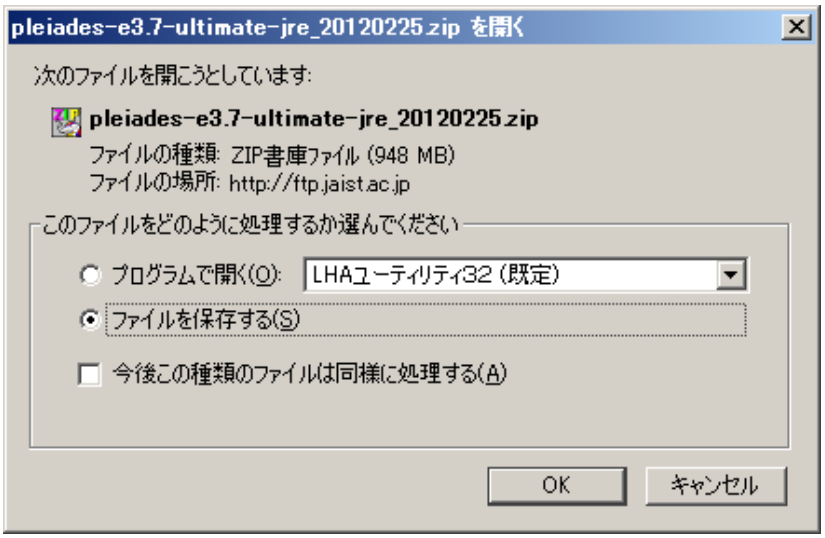
「Eclipse 3.7 Indigo Pleiades All in One」を選択します。

バージョンについては、必ず最新版である必要はありません。安定しているバージョンを使用した方が、様々なトラブルを回避することも出来ますので、その点を考慮した上で、バージョンを選ぶようにしましょう。

選択後、下図の画面が表示されます。



ここでは、必要なものが全てそろっている「Full All in One（JRE あり）」の「Ultimate」をダウンロードします。Android や Java の開発だけでなく、C/C++や PHP など、様々な言語で開発が可能になります。



ここでまた、保存場所を指定してファイルをダウンロードします。ファイルサイズは 948MB もありますので、ダウンロードにはしばらく時間がかかります。

ダウンロード後、任意の場所に展開して下さい。これで、インストールは終了です。

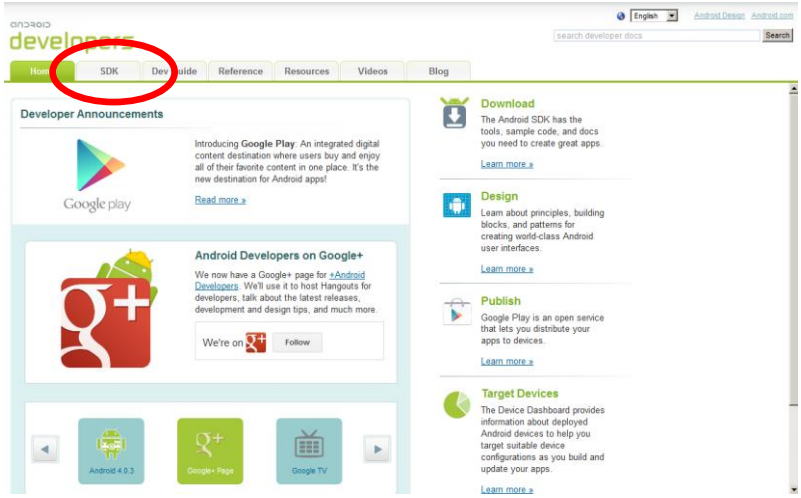
Eclipse のファイルは、ZIP 形式の圧縮ファイルになっていますので、別途、圧縮ファイルの解凍ソフトウェアが必要になります。あらかじめ準備しておいて下さい。

3.3 Android SDK のダウンロードとインストール

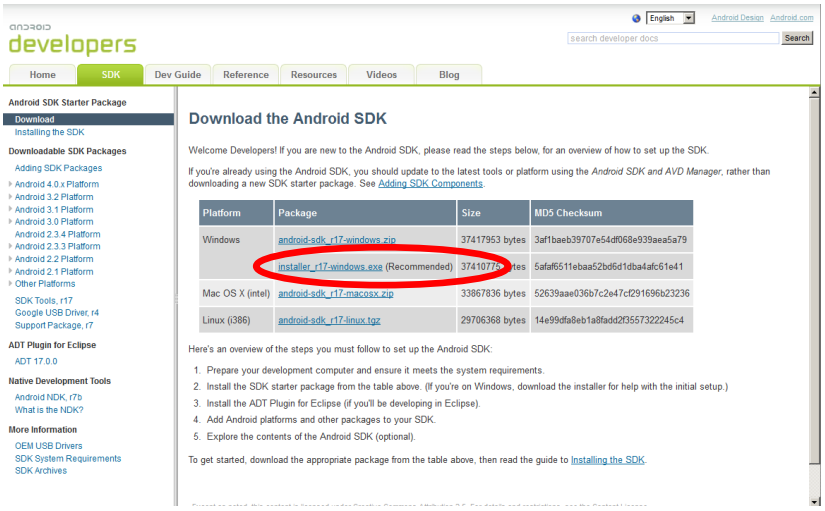
ここでは、Android SDK のダウンロードとインストールの手順について、説明します。

以下の URL にアクセスして下さい。

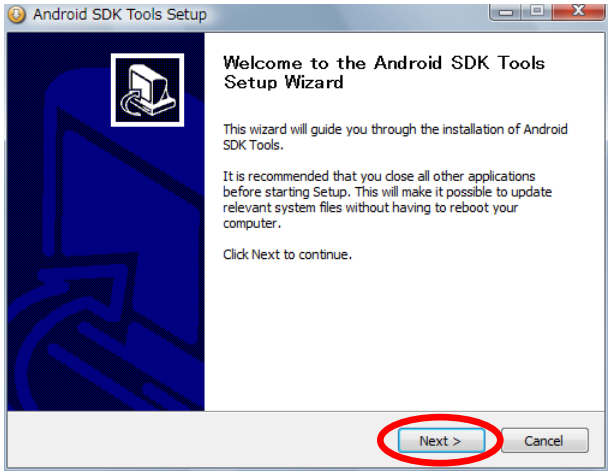
https://developer.android.com/index.html



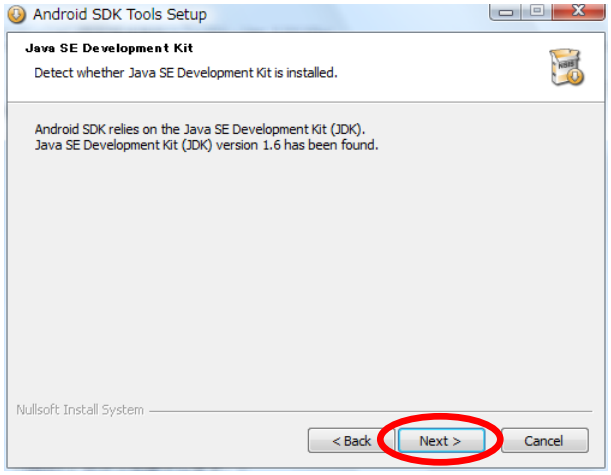
上記の URL にアクセスし、「SDK」タグを選択します。



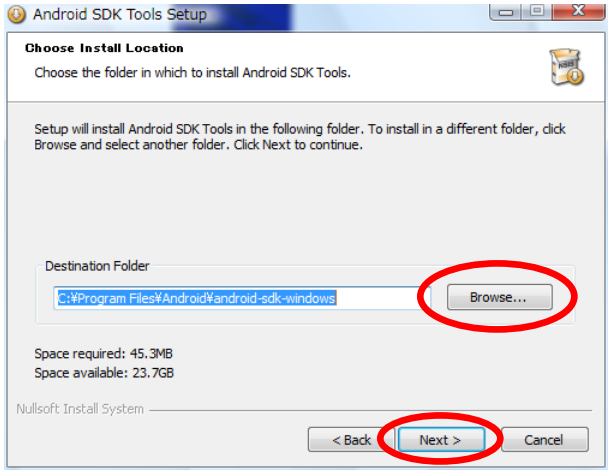
Android SDK のページに遷移します。「installer_r17-windows.exe」をクリックして、インストーラを、任意の場所に保存します。ZIP ファイルも用意されていますが、「（Recommended）」とあるように、インストーラによるインストールが推奨されています。そして、ファイル保存後、インストーラを起動します。



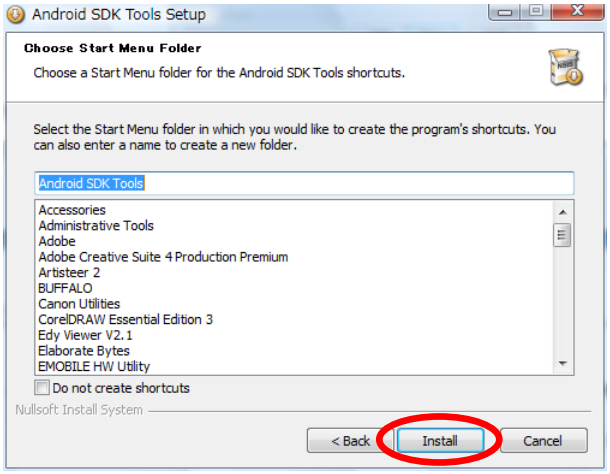
最初に、Android SDK Tools のインストールを行います。「Next>」ボタンを押下して下さい。



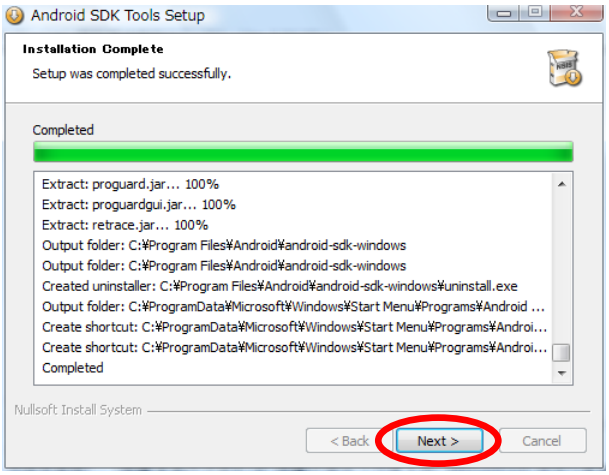
「Next>」ボタンを押下します。



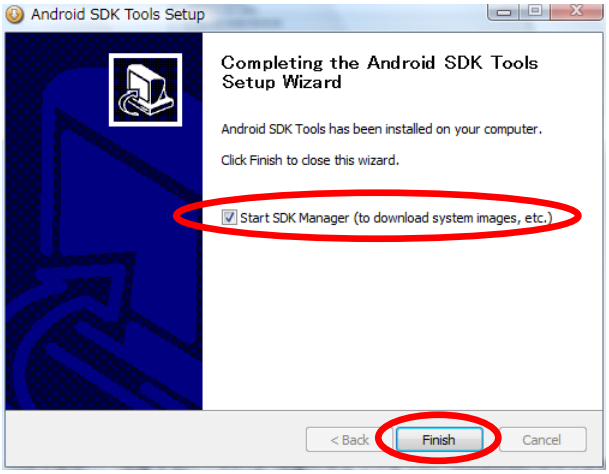
インストール先フォルダの指定ダイアログが表示されます。インストール先フォルダを変更する場合には、「Brouse...」ボタンを押下し、変更して下さい。変更後、「Next>」ボタンを押下します。



スタートメニューフォルダを指定します。指定後、「Install」ボタンを押下します。

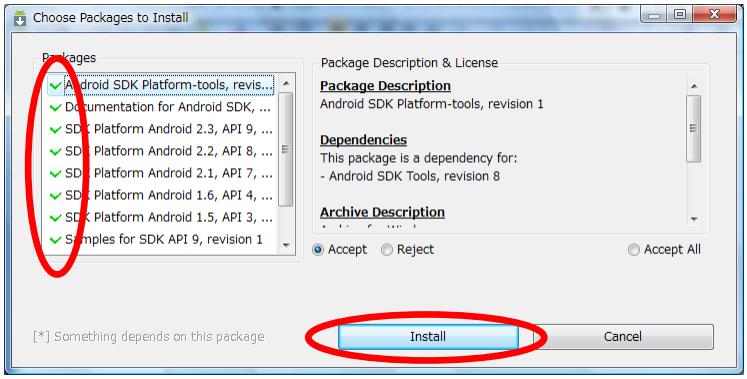


「Next>」を押下します。



Android SDK Tools のインストールが完了しました。続けて SDK Manager を起動する場合には、「Start SDK Manager」にチェックを入れ、「Finish」を押下して下さい。SDK Manager を開始しない場合には、チェックを外して下さい。

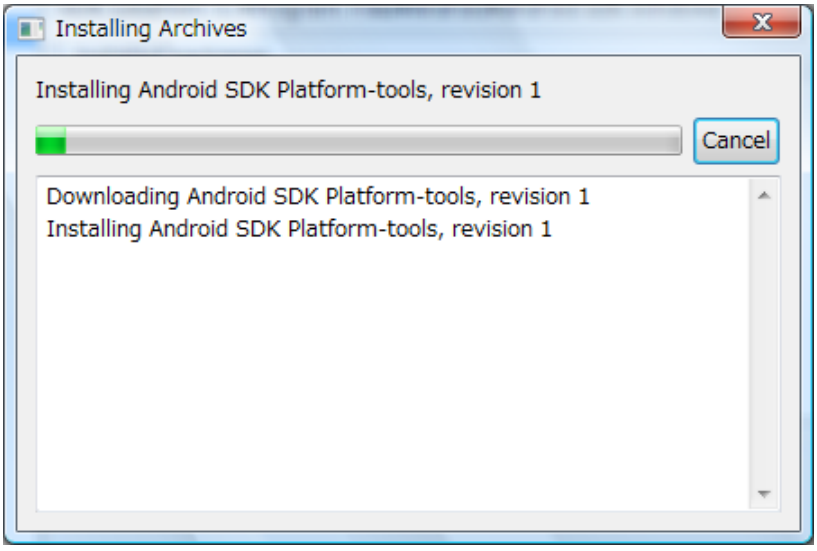
「Finish」押下後、SDK Manager が起動されます。



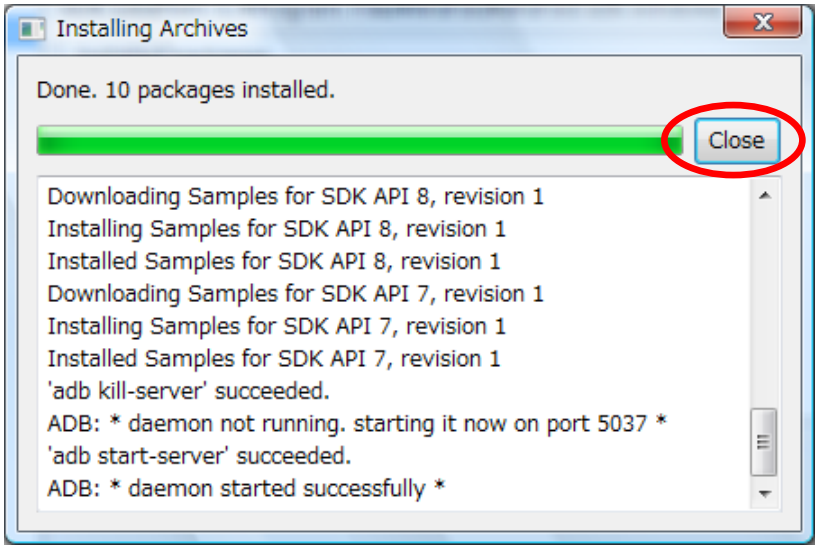
選択されたパッケージをインストールします。もし、不要なパッケージがある場合には、Packages の各項目の前にある ✔ をダブルクリックし、 ✘ に変更して下さい。

画面の右側にある「Accept」、「Reject」、「Accept All」でも、選択、解除の指定を行うことが出来ます。左側の Packages より、該当するパッケージを選択し、「Accept」あるいは「Reject」を押下することで、それぞれ「選択」、「解除」することが出来ます。

インストールしたいパッケージが決まり次第、「Install」ボタンを押下します。



「Install」ボタン押下後、インストールが始まります。
進捗状況は、画面にプログレスバーで表示されますので、しばらくお待ち下さい。



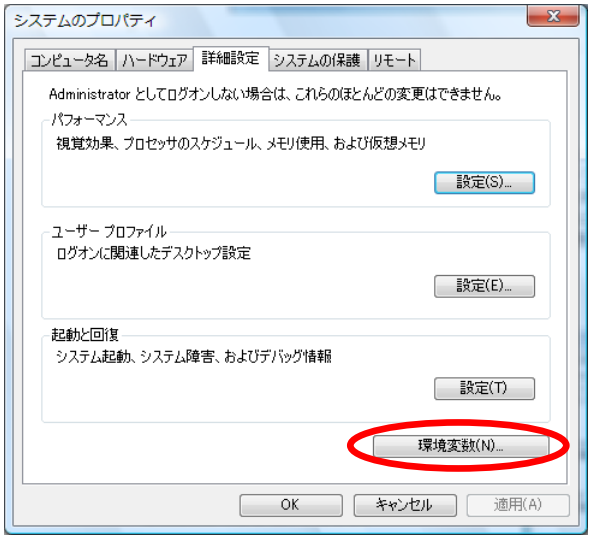
すべてのパッケージのインストールが終了すれば、完了です。「Close」ボタンを押下して、画面を閉じます。

3.3.1 環境変数 PATH への追加

Android SDK の中には、コマンドプロンプトで利用するコマンドがあります。Android SDK 内の Tools フォルダ内にあるコマンド類がこれに該当します。コマンドプロンプトで、これらのコマンドを実行する場合、コマンド類のプログラムが入っているフォルダへ移動するか、もしくは、フルパスでプログラムを指定しなければいけません。このため、事前に環境変数である PATH 変数に、このプログラムが入っているフォルダを設定しておくと便利に利用することが出来ます。

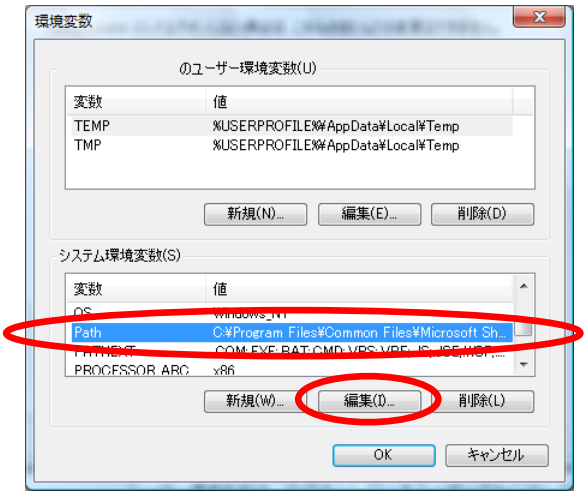
ここでは、環境変数への追加方法を記載します。

Windows のスタートメニューより「コントロールパネル」ー「システム」ー「システムの詳細設定」を選択します。



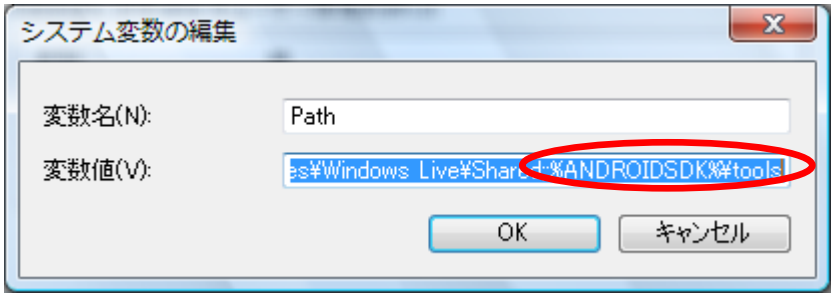
Windows Vista では、システムの詳細設定を選択したときに、ユーザーアカウント制御の確認メッセージが表示されますので、「続行」ボタンを選択し、処理を続けて下さい。

「環境変数」ボタンを押下します。環境変数ダイアログが表示されます。



ユーザー環境変数は、ログインしているユーザーのみに対して設定される変数となっており、システム環境変数は、そのコンピュータを利用する人すべてに対して設定される変数となります。

システム環境変数の「Path」の変数を選択し、その下側にある「編集」ボタンを押下します。



編集ボタン押下後、システム変数の編集ダイアログが表示されます。変数値の最後尾に、Android SDK のインストールフォルダ配下の tools フォルダ、及び platform-tools フォルダを追加します。

なお、上図では、「ANDROIDSDK」という変数を、別に追加しています。この変数には、Android SDK のインストールフォルダを設定しています。

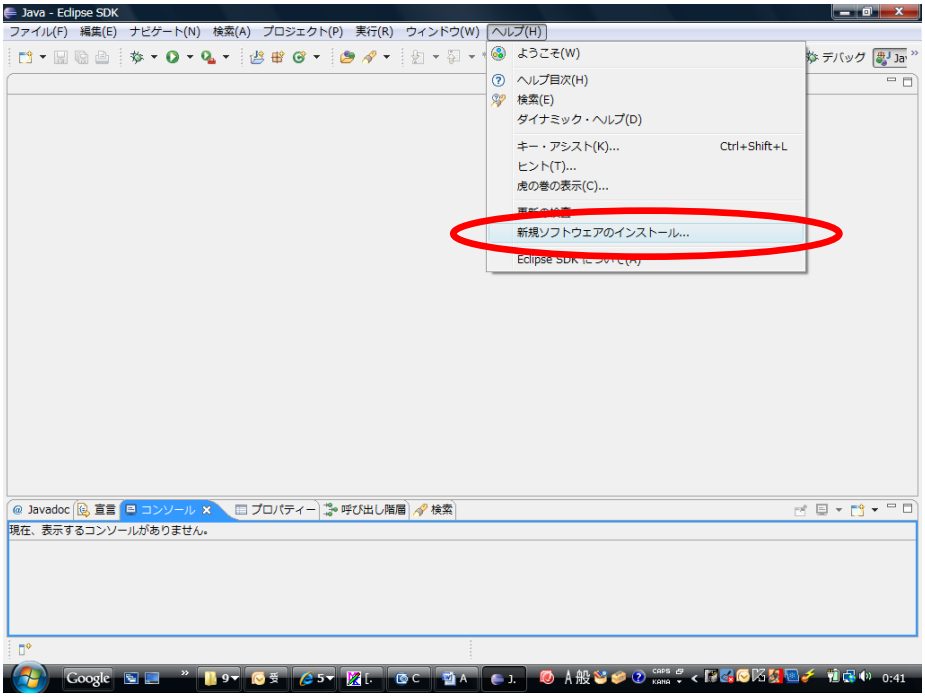
Path を指定する場合、それぞれのパスは、「;（セミコロン）」で区切って下さい。

このように設定しておくことで、コマンドプロンプトでは、プログラム名を指定するだけで動作させることが出来るようになります。

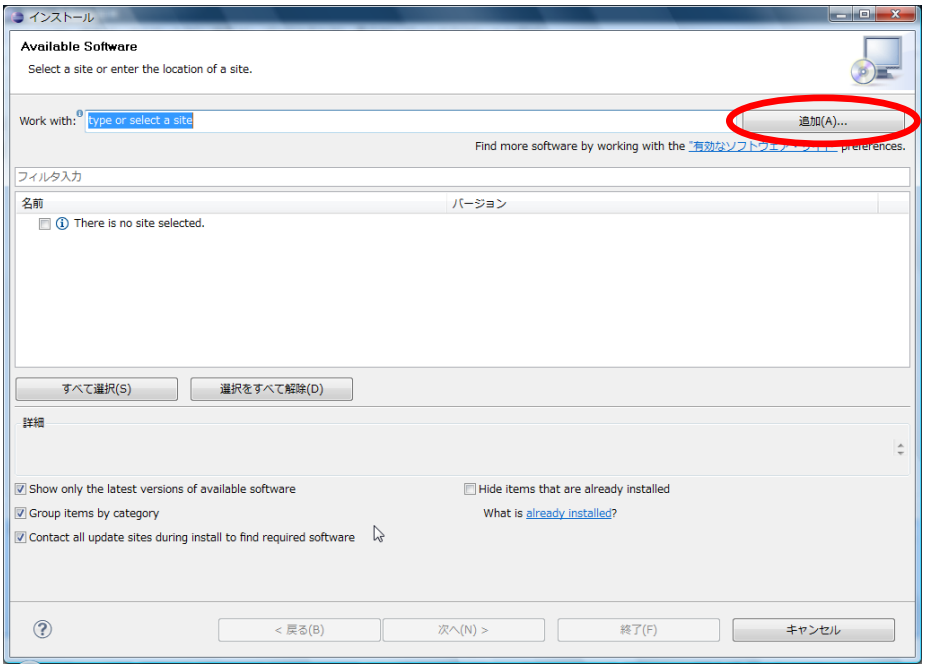
3.4 Android Development Tools Plug-in for the Eclipse IDE のインストール

ここでは、Android Development Tools Plug-in for the Eclipse IDE（以下 ADT）のインストール方法について説明します。

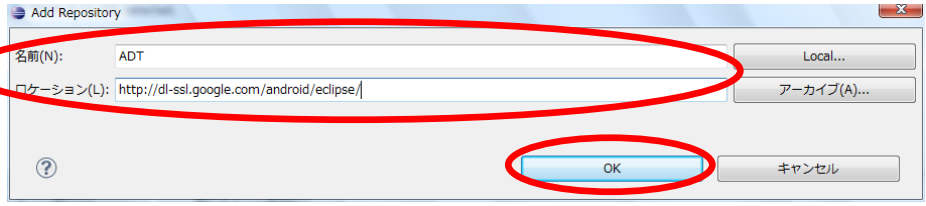
ADT は、Eclipse のプラグインであるため、Eclipse からインストールを行います。まずは、Eclipse を起動して下さい。



Eclipse 起動後、メニューの「ヘルプ」から「新規ソフトウェアのインストール...」を選択します。



インストール画面が表示されるので、「追加」ボタンを押下します。「追加」ボタン押下後、Add Repository 画面が表示されます。

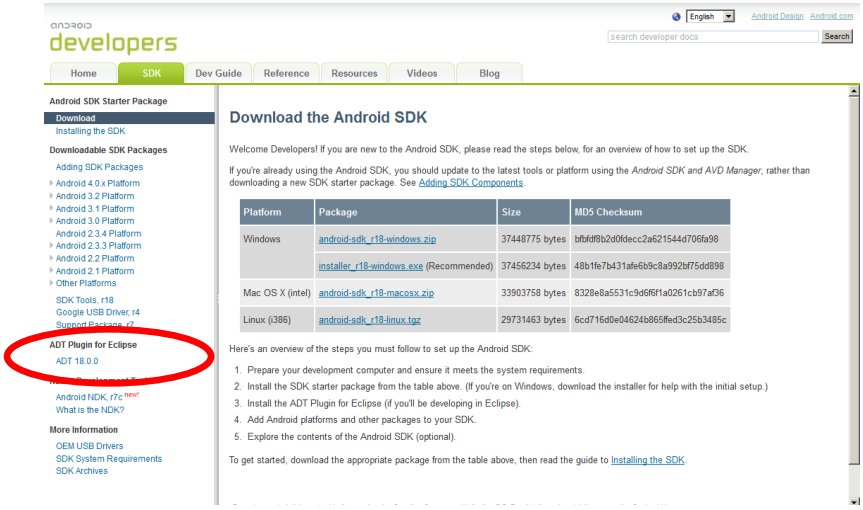


それぞれのテキストボックスに、以下の内容を設定して下さい。

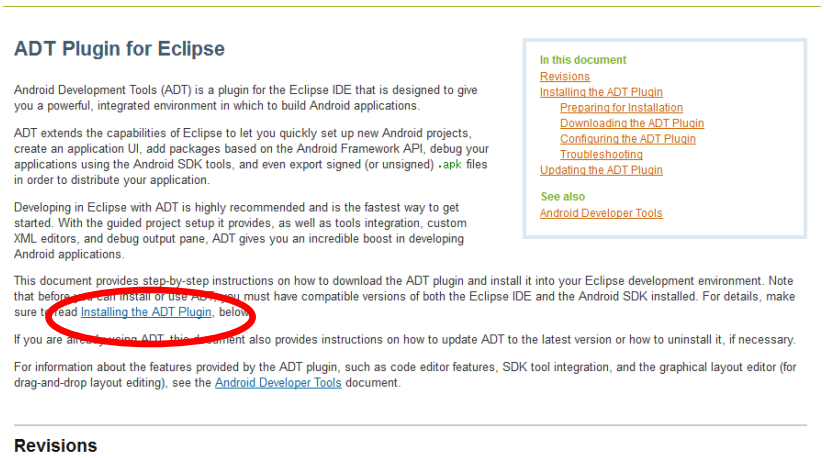
項目	内容
名前	任意の名前を付けて下さい。（例では、「ADT」という名前を付けています）
ロケーション	https://dl-ssl.google.com/android/eclipse/

入力後、「OK」ボタンを押下します。「OK」ボタン押下後、インストール画面に戻り、設定した情報によって、一覧が表示されます。

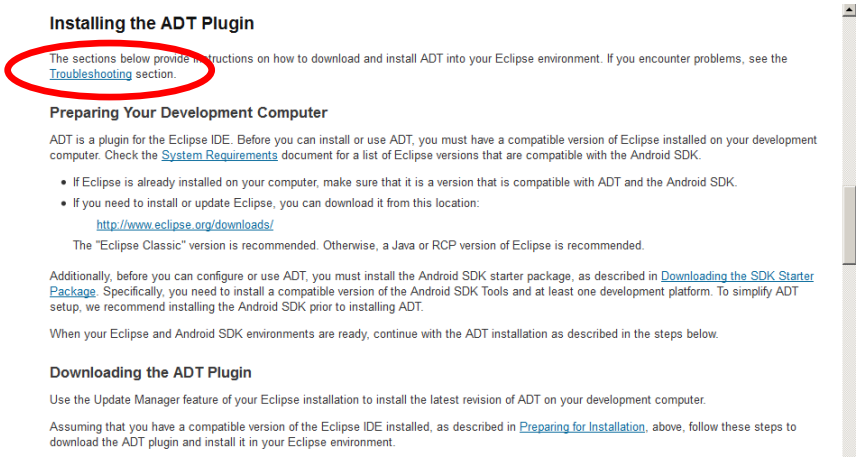
もしここでインストールがうまくいかない場合は、ZIP ファイルをダウンロードしてインストールする方法をお試し下さい。



「https://developer.android.com/index.html」にアクセスして、「SDK」のタグをクリックし、左側のメニューから「ADT Plugin for Eclipse」のメニューを選択します。



中程にある「Installing the ADT Plugin」をクリックします。



Installing the ADT Plugin

The sections below provide instructions on how to download and install ADT into your Eclipse environment. If you encounter problems, see the Troubleshooting section.

Preparing Your Development Computer

ADT is a plugin for the Eclipse IDE. Before you can install or use ADT, you must have a compatible version of Eclipse installed on your development computer. Check the System Requirements document for a list of Eclipse versions that are compatible with the Android SDK.

- If Eclipse is already installed on your computer, make sure that it is a version that is compatible with ADT and the Android SDK.
- If you need to install or update Eclipse, you can download it from this location:

  http://www.eclipse.org/downloads/

  The "Eclipse Classic" version is recommended. Otherwise, a Java or RCP version of Eclipse is recommended.

Additionally, before you can configure or use ADT, you must install the Android SDK starter package, as described in Downloading the SDK Starter Package. Specifically, you need to install a compatible version of the Android SDK Tools and at least one development platform. To simplify ADT setup, we recommend installing the Android SDK prior to installing ADT.

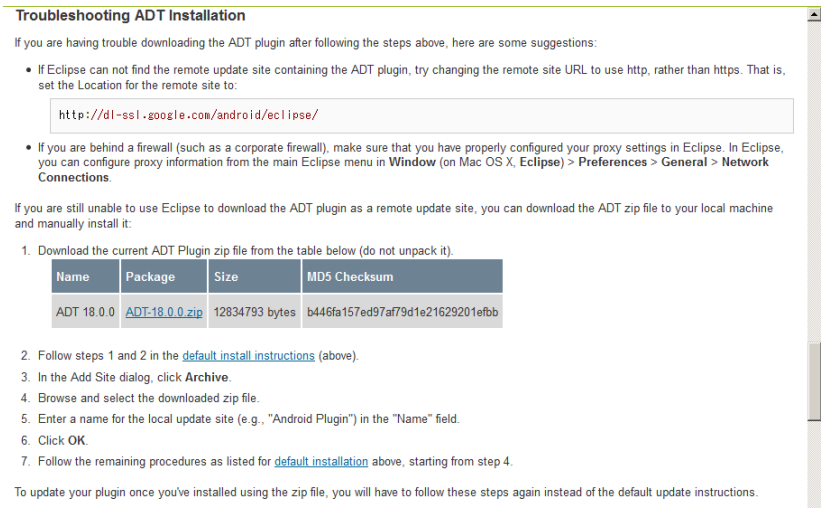
When your Eclipse and Android SDK environments are ready, continue with the ADT installation as described in the steps below.

Downloading the ADT Plugin

Use the Update Manager feature of your Eclipse installation to install the latest revision of ADT on your development computer.

Assuming that you have a compatible version of the Eclipse IDE installed, as described in Preparing for Installation, above, follow these steps to download the ADT plugin and install it in your Eclipse environment.

一番上の方にある「Troubleshooting」をクリックします。



Troubleshooting ADT Installation

If you are having trouble downloading the ADT plugin after following the steps above, here are some suggestions:

- If Eclipse can not find the remote update site containing the ADT plugin, try changing the remote site URL to use http, rather than https. That is, set the Location for the remote site to:

  `http://dl-ssl.google.com/android/eclipse/`

- If you are behind a firewall (such as a corporate firewall), make sure that you have properly configured your proxy settings in Eclipse. In Eclipse, you can configure proxy information from the main Eclipse menu in **Window** (on Mac OS X, **Eclipse**) > **Preferences** > **General** > **Network Connections**.

If you are still unable to use Eclipse to download the ADT plugin as a remote update site, you can download the ADT zip file to your local machine and manually install it:

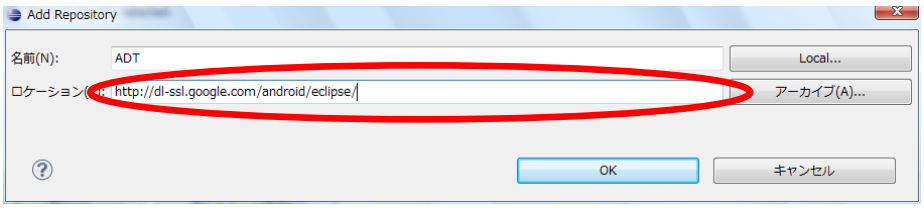
1. Download the current ADT Plugin zip file from the table below (do not unpack it).

Name	Package	Size	MD5 Checksum
ADT 18.0.0	ADT-18.0.0.zip	12834793 bytes	b446fa157ed97af79d1e21629201efbb

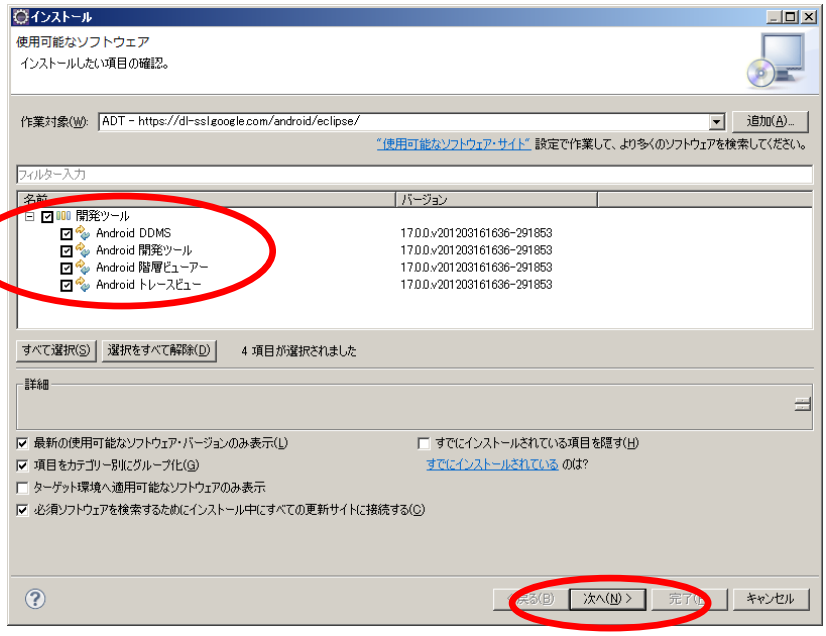
2. Follow steps 1 and 2 in the default install instructions (above).
3. In the Add Site dialog, click **Archive**.
4. Browse and select the downloaded zip file.
5. Enter a name for the local update site (e.g., "Android Plugin") in the "Name" field.
6. Click **OK**.
7. Follow the remaining procedures as listed for default installation above, starting from step 4.

To update your plugin once you've installed using the zip file, you will have to follow these steps again instead of the default update instructions.

中程に ZIP ファイルへのリンクがあるので、それをクリックし、ZIP ファイルを適当なフォルダにダウンロードします。



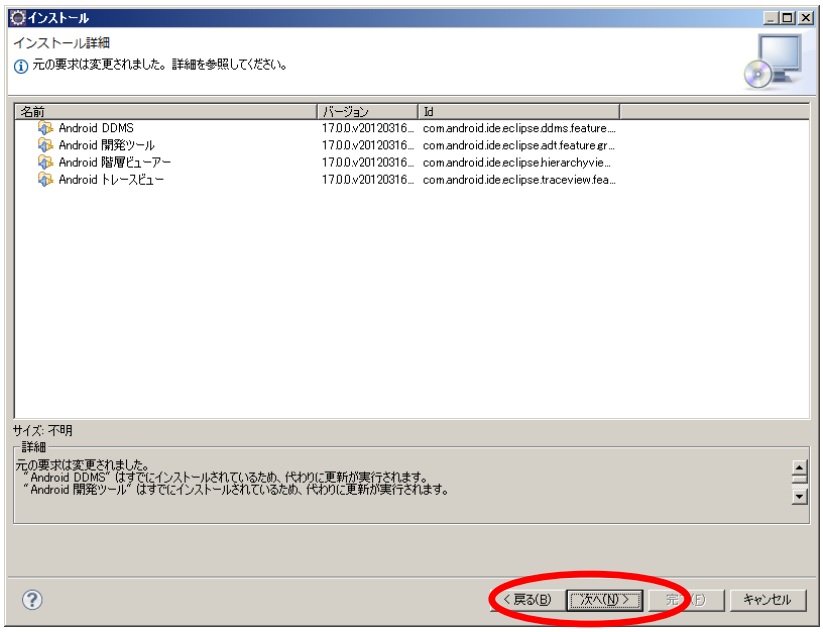
Eclipse でメニューの「ヘルプ」から「新規ソフトウェアのインストール...」を選択し、「アーカイブ」にダウンロードした ZIP ファイルを指定して「OK」ボタンをクリックします。あとは、Web 上からのインストール方法と同じです。



正常に追加されると、インストール画面に「開発ツール」が表示されます。その行の先頭にチェックを入れ、「次へ」ボタンを押下します。

「開発ツール」の先頭にある三角マークを選択すると、「開発ツール」に含まれている「Android DDMS」「Android 開発ツール」「Android 階層ビューアー」「Android トレースビュー」の 4 つの項目が表示されます。
それぞれの項目を選択すると、画面下部の詳細部分に、その項目の詳細が表示され、また、更に表示中に右側の「More...」を選択することで、より詳細な内容を見ることが出来ますので、確認をしましょう。

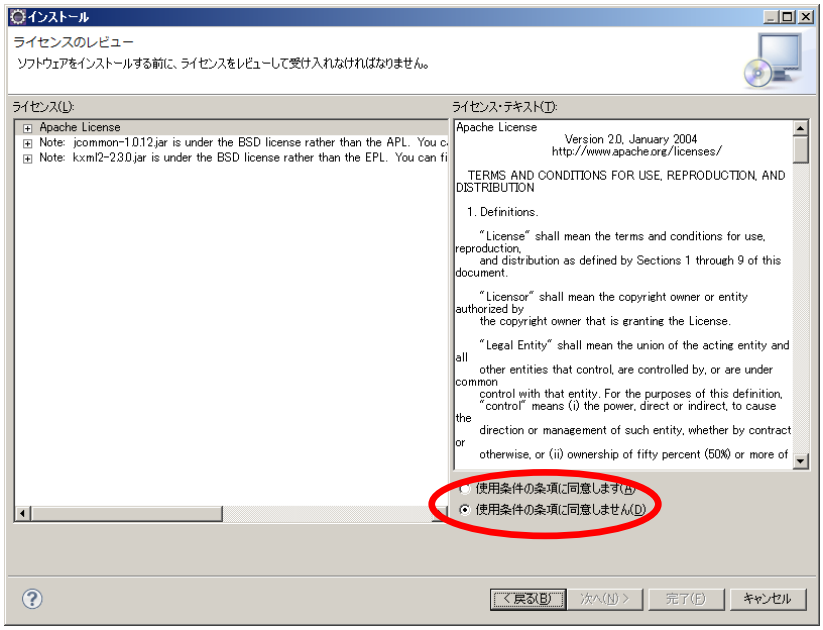
「次へ」ボタン押下後、インストール詳細画面が表示されます。



内容確認後、「次へ」ボタンを押下します。

この画面で、それぞれの項目選択時に表示される内容は、先の画面で確認した内容と同じです。必要に応じて、確認しましょう。

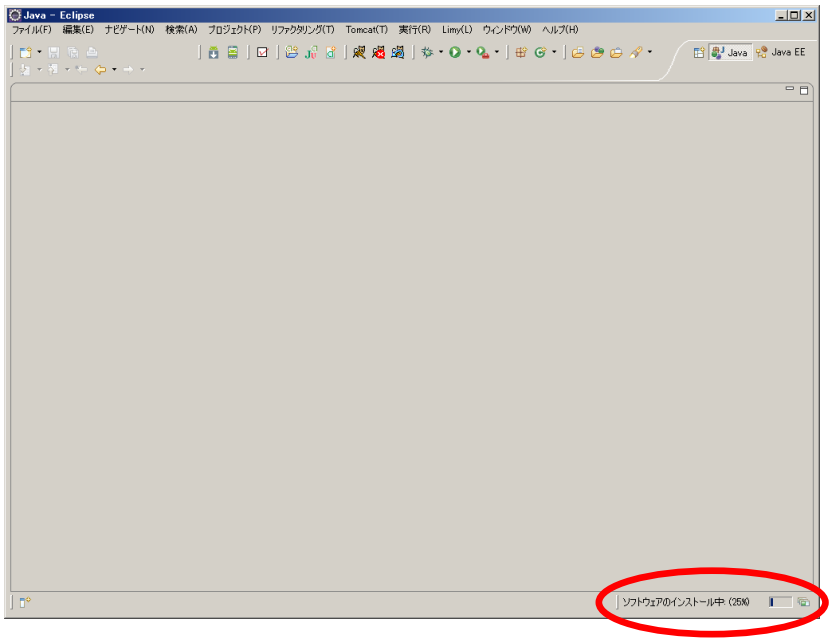
インストールされるソフトウェアのライセンス情報が表示されます。



ライセンスを確認し、（同意するのであれば）「使用条件の条項に同意します」を選択し、「完了」ボタンを押下します。

ライセンスは、非常に重要ですので、必ず内容を読むようにして下さい。

「完了」ボタン押下後、インストールが開始されます。進捗状況は、ウィンドウの右下にあるプログレスバーで表示されます。



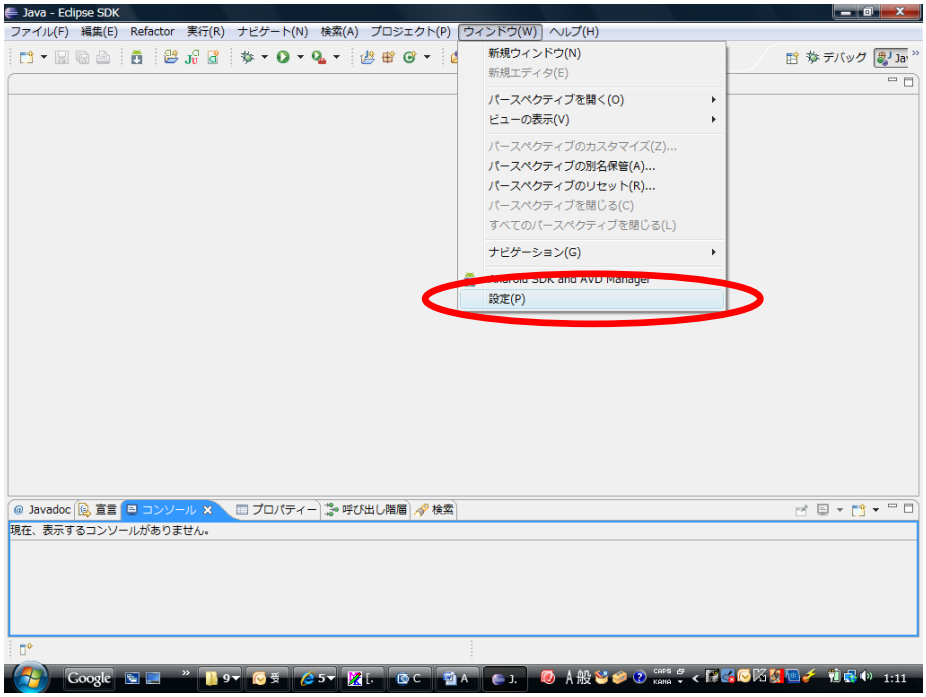
インストール終了後、Eclipse を再起動させて下さい。

「今すぐ再始動」ボタンを選択すると、Eclipse が再起動されます。

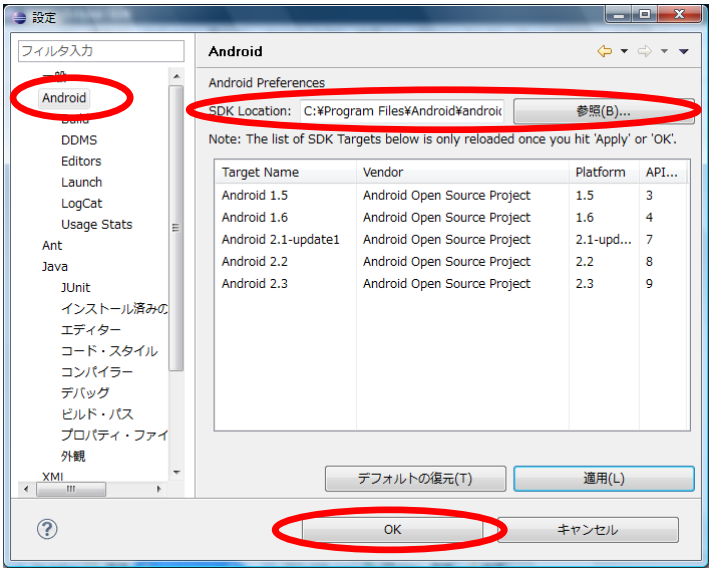
3.5 ADT の設定

ADT インストール後、Eclipse の設定を行います。

Eclipse のメニュー「ウィンドウ」ー「設定」を選択します。



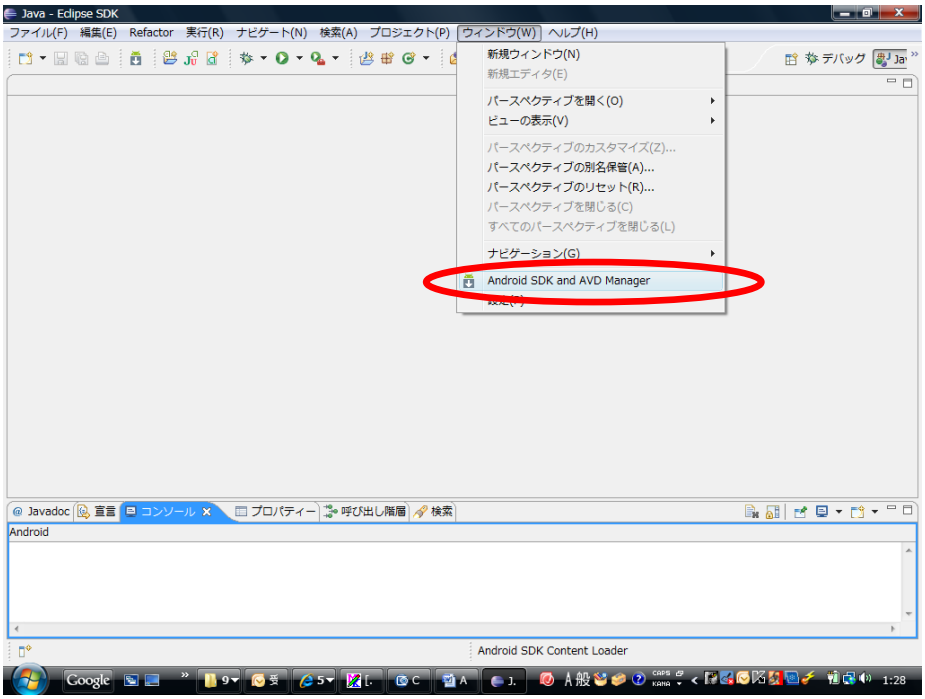
設定ダイアログが表示されます。



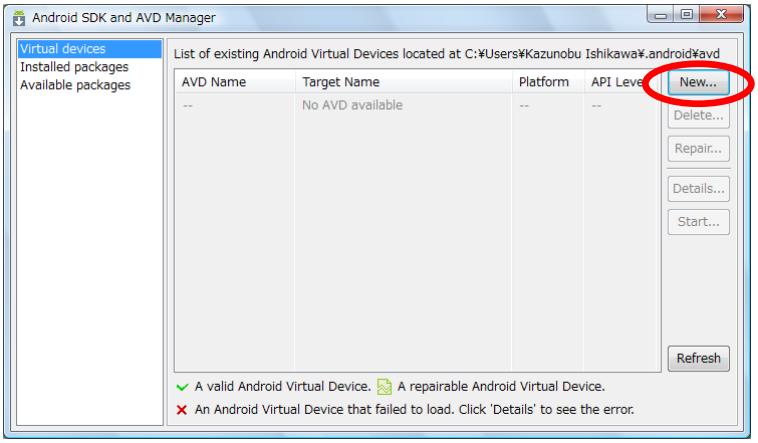
「Android」を選択し、設定内容を表示します。まずは、「SDK Location」に、先にインストールしている Android SDK のインストールフォルダを設定して下さい。

3.6 Android Virtual Device の作成

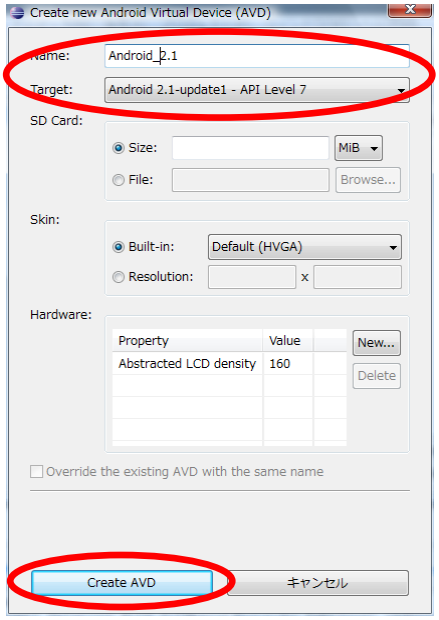
Android Virtual Device（以下 AVD）とは、PC 上での実行環境です。デフォルトでは、AVD は 1 つも設定されていないため、予め作成しておく必要があります。なお、実際には、AVD は、SDK のバージョンの違いや、各端末の画面の解像度の違いなどにより、様々なパラメータで AVD を作成しておく必要がありますが、ここでは、Android SDK 2.1 用の HVGA の解像度を持つ AVD を作成します。メニュー「ウィンドウ」ー「Android SDK and AVD Manager」を選択して下さい。



Android Virtual Device Manager の画面が表示されます。



画面右側にある「New...」ボタンを押下して下さい。Create new Android Virtual Device(AVD)の画面が表示されます。この画面で、AVD に必要な項目を設定します。



項目の詳細については、「表 1　Android Virtual Device Manager 設定項目一覧」を参照して下さい。

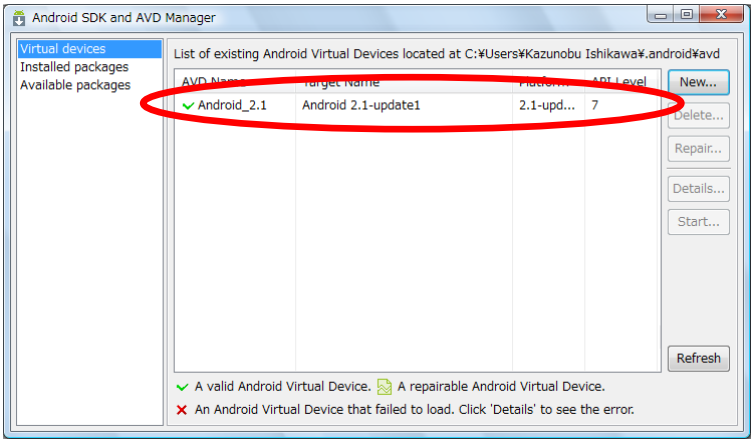
設定後、「create AVD」ボタンを押下し、AVD を作成して下さい。

項目		内容
Create AVD		
	Name	AVD の名前を設定します。入力出来る文字は、「a-z」、「A-Z」、「0-9」、「. 」、「_」、「-」です。
	Target	ターゲットとする SDK を指定します。
	SDCard	SD Card の情報を設定します。
	Skin	AVD が使用するスキンを指定します。
	Hardware	ハードウェアの情報を設定します。さまざまなハードウェアのサ

項目	内容
	ポートを行わせる場合には、ここで、各値を設定します。

表 1 Android Virtual Device Manager 設定項目一覧

「create AVD」ボタンを押下後、作成した AVD が表示されます。



ここでは、Android 2.1 用の AVD を作成しましたが、必要に応じて AVD を追加してみて下さい。

AVD の設定情報は、「C:¥ Documents and Settings¥アカウント名¥.android¥adv」に保存されます。このとき、アカウント名に日本語を利用している場合に、エミュレータを起動すると、エラーが発生し、起動することが出来ませんので、ご注意下さい。なお、対処方法としては、アカウント名に日本語を使わないようにするか、もしくは、上記のフォルダを、英字のみのフォルダに移動後、フォルダ下にある ini ファイルを編集し、Path の変更を行って下さい。

3.6.1 スキンの追加

現在では、すでに多くの Android 端末が発売されています。Android プログラムの開発を行う場合には、当然のことながら、対応させたい端末の仕様に合わせた AVD を利用して開発を行う必要があります。AVD の設定は、すべて自分自身で行うことは出来ますが、すべての端末の情報を設定するには大変な手間がかかってしまいます。

そこで利用したいのが、Android 端末を発売しているメーカーが、Android プログラムの開発者向けに提供しているスキンです。

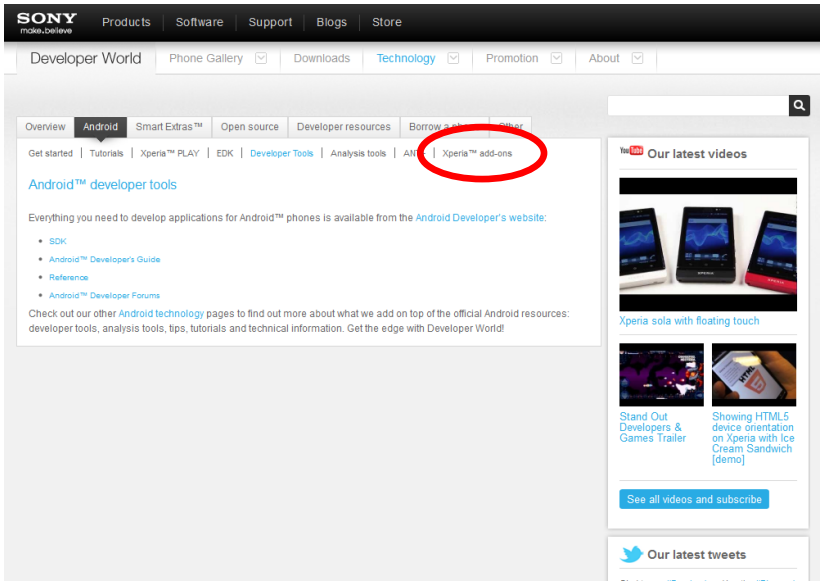
スキンとは、UI（ユーザーインターフェイス）に画像を張り付けて見た目を変更する仕組み、あるいは貼り付ける画像そのものを指します。このスキンを、Eclipse の環境へ設定し、AVD を作成しておけば、対応する端末そっくりの見た目や設定を行うことが出来ることでしょう。

ここでは、Xperia X10 のスキンを、導入する手順を示します。

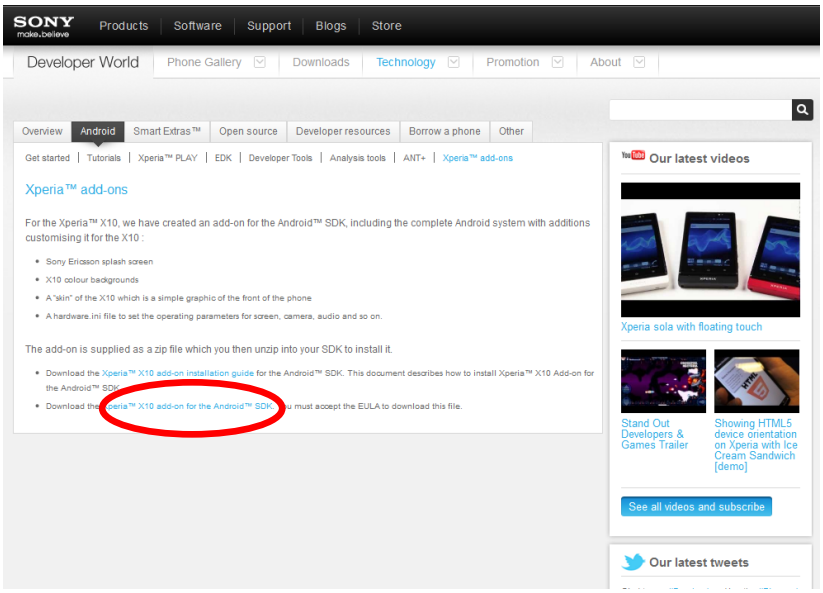
Sony Ericsson の Developer 向けのサイトである以下の URL にアクセスします。

http://developer.sonyericsson.com/wportal/devworld/technology/android/developer-tools

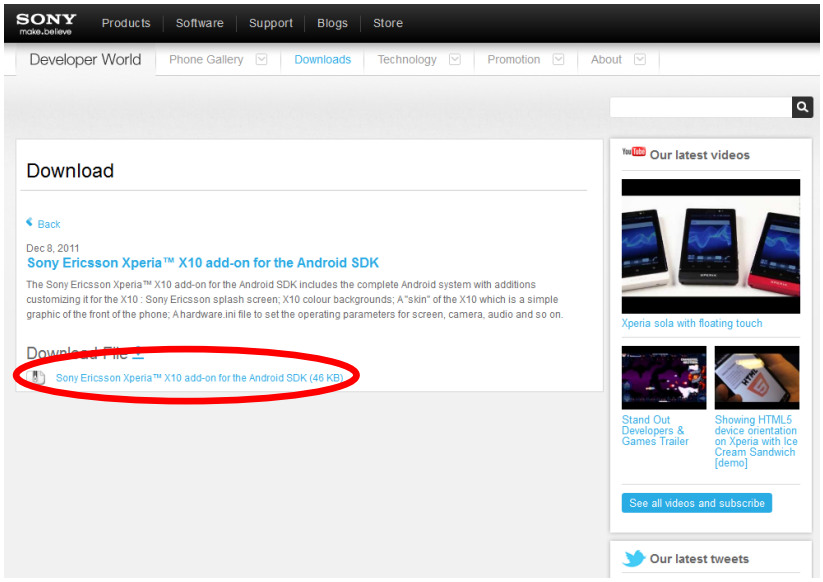
上記の URL にアクセスすると、次のような画面が表示されます。



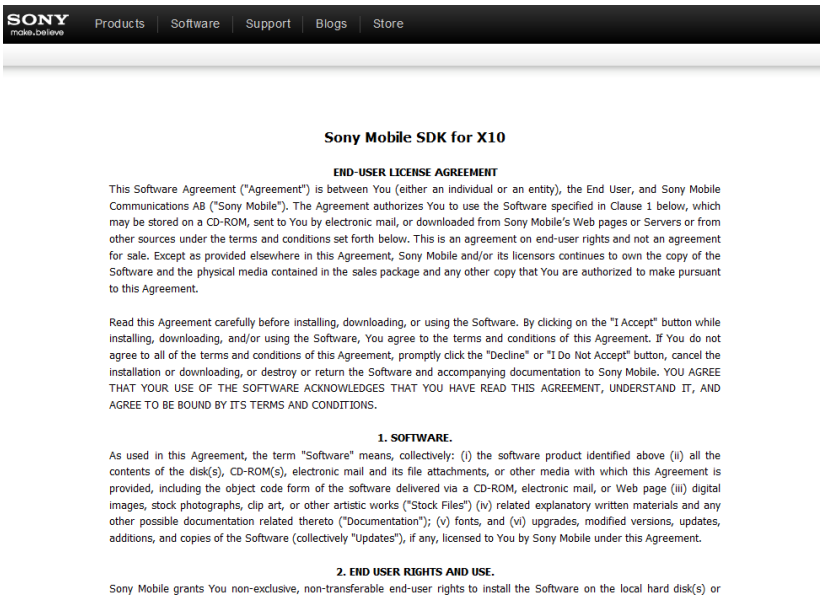
タブのすぐ下にある「Xperia™ X10 add-ons」をクリックすると、次のような画面が表示されます。



画面の下側にある「Xperia™ X10 add-on for the Android™ SDK」を選択します。



「Sony Ericsson Xperia™ X10 add-on for the Android SDK(46KB)」をクリックします。

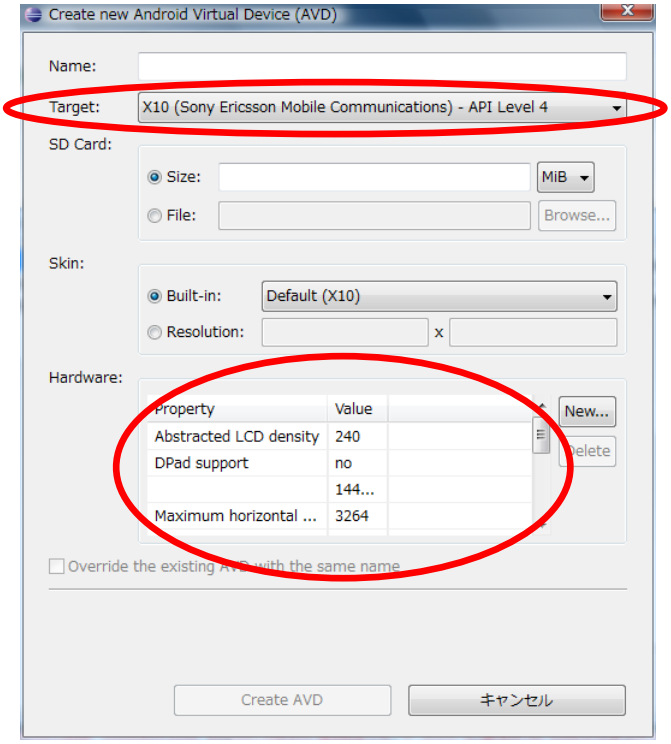


Sony Moboke SDK for X10 END-USER LICENSE AGREEMENT が表示されますので、内容を確認して下さい。読み進めていくと、画面の最下部に、「Reject」ボタンと「Accept」ボタンがあります。ライセンスの内容に問題がなければ、「Accept」ボタンを押下し、任意の場所にファイルをダウンロードして下さい。

ダウンロード後、任意の場所にファイルを解凍します。

ZIP 形式の圧縮ファイルになっていますので、別途、圧縮ファイルの解凍ソフトウェアが必要になりますので、あらかじめ準備しておいて下さい。

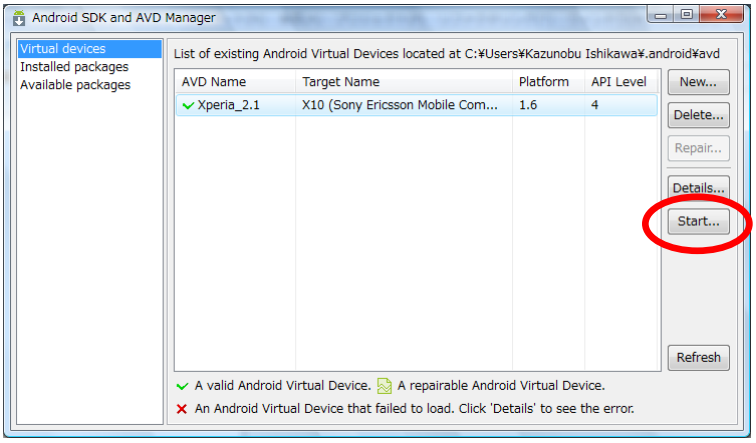
解凍すると、Xperia 用のフォルダが出来がります（例では、XPERIA-X10_r1 となります）。このフォルダを、Android SDK のインストールフォルダ配下にある add-ons 配下に保存します。



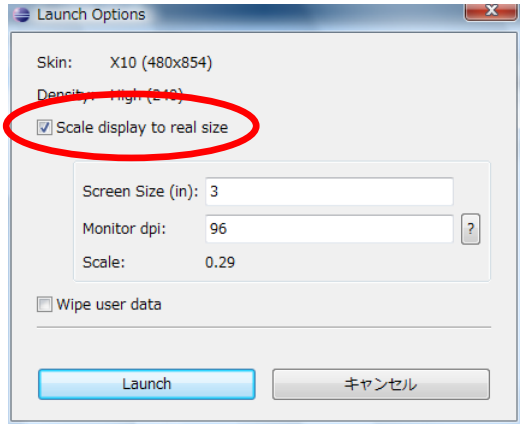
保存後、AVD の設定画面を開くと、「X10(Sony Ericsson Mobile Communications) – API Level 4」が選択することが出来るようになります。この Sony Ericsson SDK には、Hardware の詳細な設定についても設定されているため、そのまま利用することが可能です。

さて、これで、Xperia 用の AVD を作成することが出来ました。AVD の作成を行うと、「Android SDK

and AVD Manager」に、新たに作成した AVD が表示されます。



作成した AVD が、どのように設定されたのかを確認してみましょう。作成した AVD を選択し、画面右側にある「Start...」ボタンを押下します。「Start...」ボタンを押下すると、以下のダイアログが表示されます。



「Scale display to real size」をチェックし、「Launch」ボタンを押下し、エミュレータを起動すると、下図のようなエミュレータが起動されます。

何も設定せずに「Launch」ボタンを押下しても、Xperia のスキンのエミュレータを起動することは出来ますが、サイズが 480 x 854 となっているため、ほとんどの環境では巨大なエミュレータが起動されてしまいますので、「Scale display to real size」で、ディスプレイサイズの調整を行っておく必要があります。

なお、スクリーンの設定値に関しては、以下の URL のページに書かれている内容も、ご参照下さい。

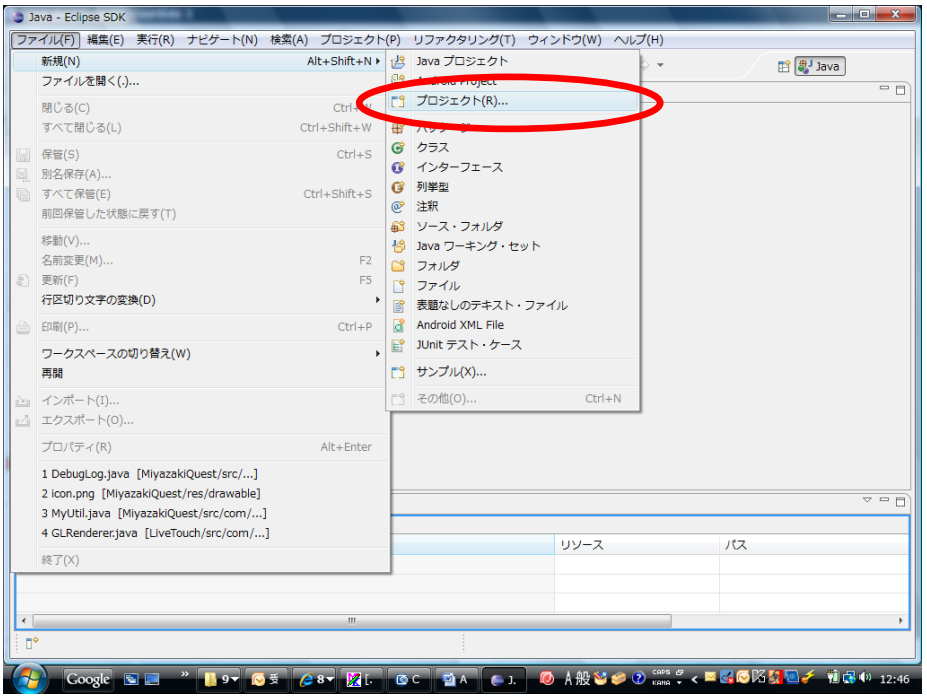
http://developer.android.com/guide/practices/screens_support.html

4. やっぱり最初は Hello World

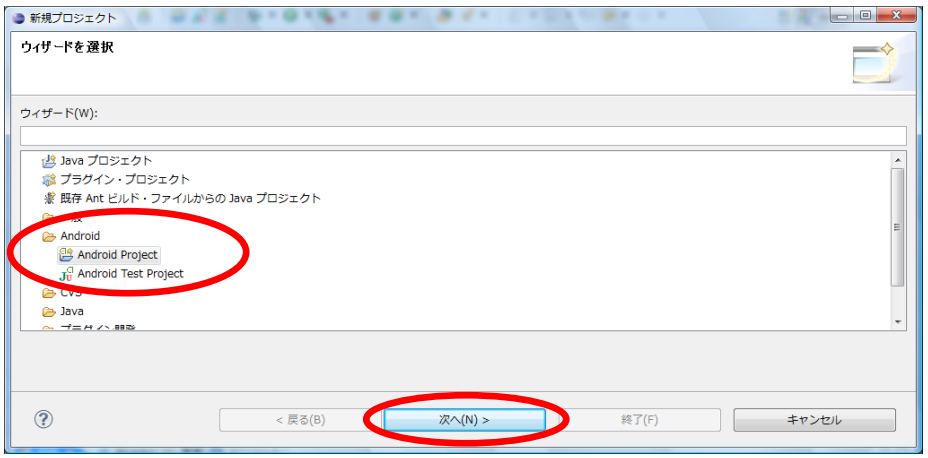
さて、ここまでの章で、インストールおよび設定が終了しました。これで、Eclipse 上で Android のアプリケーションを作成するための基本的な開発環境が構築出来ました。開発環境が構築出来たならば、動作確認も含めて、単純なアプリケーションを動作させてみましょう。ということで、最初はお約束の「Hello World」アプリケーションを作ってみましょう。

4.1 プロジェクト作成

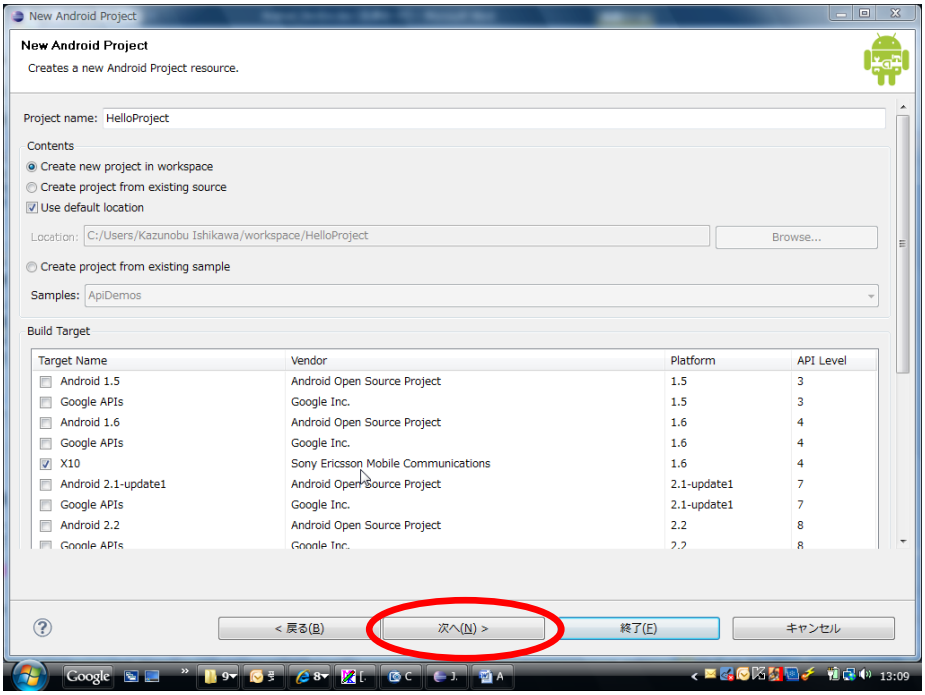
Eclipse のメニュー「新規」ー「プロジェクト...」を選択して下さい。



新規プロジェクトダイアログが表示されます。



リストの中から「Android」－「Android Project」を選択し、「次へ」ボタンを押下します。New Android Project ダイアログが表示されます。

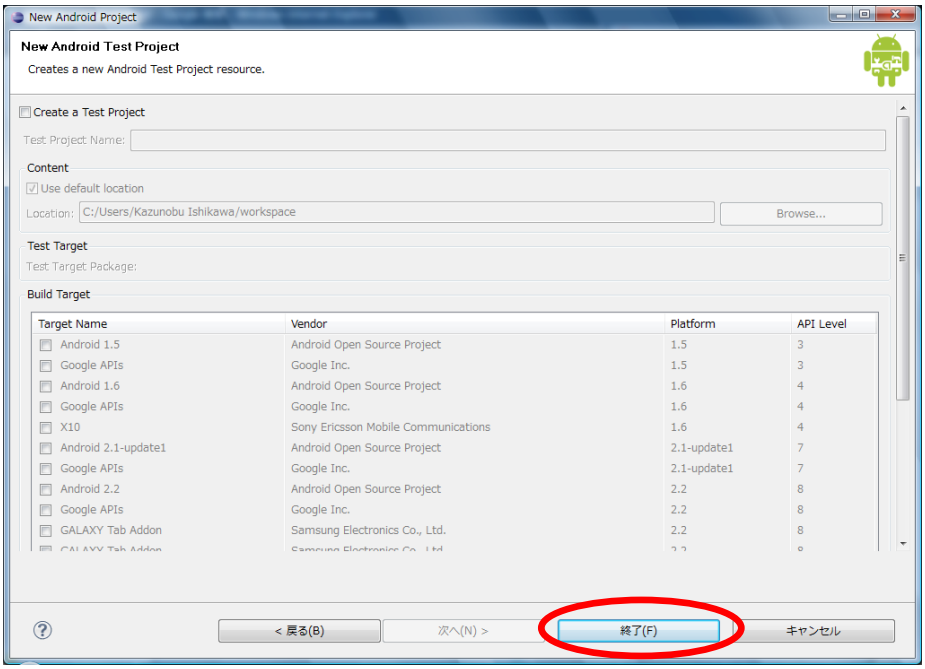


このダイアログで、Android プロジェクトに必要な事項を設定します。各項目設定後、「次へ」ボタンを押下します。

項目		内容	例
Project Name		プロジェクト名を設定します。	HelloProject
Contents			
	Create new project in workspace	ワークスペース内に新しいプロジェクトを生成します。	チェック
	Create project from existing source	既にあるソースからプロジェクトを生成します。	―
	Use default location	デフォルトの場所を使用する場合に、この項目をチェックします。別な場所を指定する場合には、このチェックを外し、下のテキストボックスに、任意の場所を指定します。	―
	Create project from existing sample	既存のサンプルからプロジェクトを生成する場合に、この項目をチェックします。選択した際には、その下のコンボボックスからサンプルプロジェクトを選択します。	―
Build Target		ターゲットとするプラットフォームを選択します。Android のバージョンの違いやメーカーから提供されているスキンによって、選択出来るプラットフォームが異なりますので、対応したいプラットフォームを選択して下さい。	X10
Properties			
	Application name	アプリケーション名を設定します。この名前で、Android の携帯に登録されることになります。	Hello World

項目		内容	例
	Package name	Java のパッケージ名を設定します。デフォルトパッケージは、使用することは出来ないため、使用可能なパッケージ名を確保しなければいけません。	com.exsample.world.hello
	Create Activity	Activity を継承するクラス名を指定します。Activity は、アプリケーションのライフサイクルを管理するクラスであり、Android アプリケーションでは、必ず実装する必要があります。	HelloWorld
	Min SDK Version	対応する Android SDK のレベルを指定します。Build Target 選択時に自動的に設定されますが、入力可能となっているため、基本的には、変更しなくて構いません。	—

表 2 New Android Project ダイアログ設定項目一覧



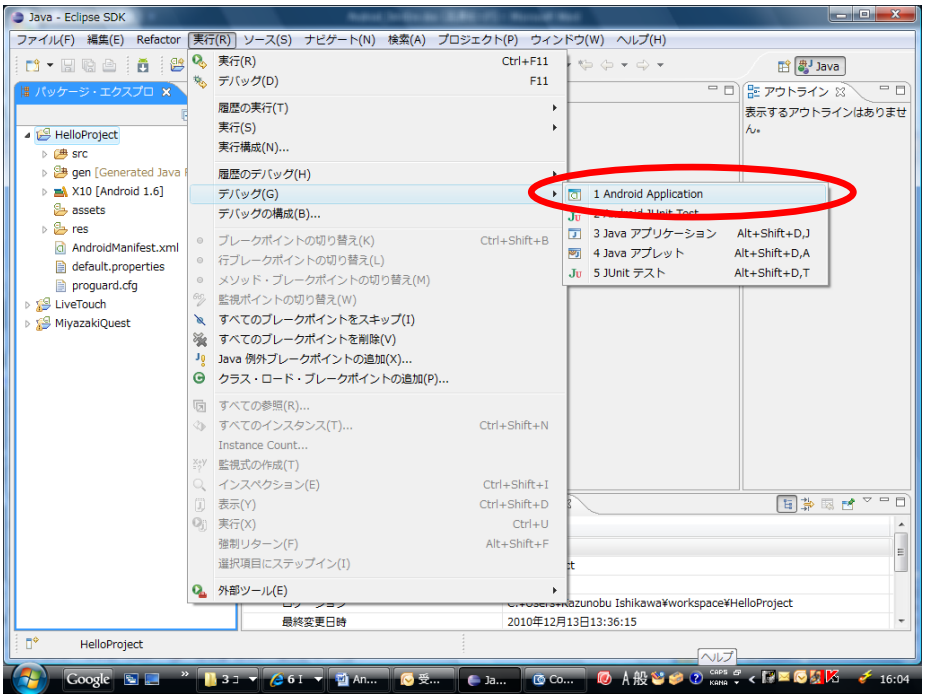
New Android Test Project ダイアログが表示されます。これは、テストプロジェクトを作成するためのものですので、ここでは、「Create Test Project」のチェックを解除してから、「終了」ボタンを押下して下さい。

テストプロジェクトを生成しない場合には、New Android Project ダイアログにて、「次へ」ボタンを押下せずに、「終了」ボタンを押下しても構いません。動作としては、同じ動作となります。

これで、プロジェクトが作成出来ました。

4.2 プロジェクト実行

プロジェクトが作成出来ましたので、それでは作ったばかりのプロジェクトを実行してみましょう。メニュー「実行」ー「デバッグ」ー「Android Application」を選択して下さい。



選択後、しばらくするとエミュレータが起動します。

エミュレータが起動するまでは、かなり時間がかかりますので、しばらくお待ち下さい。

画面上には、「Screen locked. Press Menu to unlock」とのメッセージが表示され、スクリーンロック状態となっています。スクリーンロックを解除するために、Menu キーを押下して下さい。

ロック解除後、画面上に、「Hello World, Hello World!」が表示されます。

これで、動作の確認も行うことが出来ました。

なお、コンパイル時に

”Error generating final archive: Debug certificate expired on　YY/MM/DD”

（「YY/MM/DD」には、環境によって適当な年月日が表示されます）

というエラーが発生した場合は、Windows のユーザフォルダにある

「C:¥User¥XXXX¥.android¥debug.keystore」

（「XXXX」にはユーザ名が入ります）

というファイルを削除し、Eclipse でクリーニングを行ってください。

※以前、Android の開発環境を設定してからしばらく使用していなかった場合に、このようなエラーが表示されるようです。

巻末資料 3　Android ゲームアプリケーション　ロールプレイングゲーム解説

平成 23 年度　東日本大震災からの復旧・復興を担う専門人材育成支援事業
「モバイルアプリケーションの開発で震災復興を支援する人材の育成」

Android ゲームアプリケーション
ロールプレイングゲーム解説

Ver. 2.00.00

更新履歴

Version	変更内容
1.00.00	初版発行
2.00.00	プログラム部分の大幅な改訂

目 次

1. はじめに

本書では、Android 上で動作するロールプレイングゲーム（以下 RPG）である MiyagiQuest についての解説をしています。おもに、プログラムの解説を行っているため、開発時に使用する環境、ツールや使用言語についての情報は最小限の情報しか記載されていませんので、必要に応じて、適宜、参考資料等を参照するようにして下さい。

また、本書で使用しているサンプルプログラムのソースコードには意図的にコピーライト表示をしていません。教育用途で使用され、学生が自由に変更出来ることを目的としています。

1.1 開発環境

本書では、以下の環境を使用しています。

OS	Windows Vista™ Home Service Pack 1
ブラウザ	Internet Explorer 8
Android SDK	Android 2.1 SDK
JDK	Java 6 Update 23
Eclipse	Eclipse 3.6.1

使用している環境によっては、本書に記載されている画面、メッセージとは異なっている場合がありますので、ご了承下さい。

1.2 対象読者

本書が対象にしている読者は、これから Android 上で、RPG を開発しようと考えている方を対象としています。必要となるスキルとしては、Eclipse の使用方法や Java 言語に対する知識は必須となります。

1.3 免責事項

本書に掲載する情報には、充分に注意を払って作成していますが、その内容について保証するものではありません。また、「モバイルアプリケーションの開発で震災復興を支援する人材の育成」推進協議会は、本書の内容の使用ならびに閲覧によって生じたいかなる損害にも責任を負いかねますので、ご了承ください。

なお、本書の内容については、予告なく変更される場合があります。

2. RPG とは？

RPG とは、Role Playing Game（ロールプレイングゲーム）の略称です。

Role（役割、職務）を Playing（演じる）Game（ゲーム）ということですね。

なお、日本で RPG として分類されているゲームは、アメリカではアドベンチャーゲームと呼ばれています。ゲームで使用されるキャラクターには、多くの場合、職業（クラスやジョブなど）や使命（生い立ちなど）をベースとした役割（Role）が決まっており、この役割に沿って、キャラクターを演じる（Play）ことになります。

RPG として有名なところでは、ドラゴンクエストやファイナルファンタジー（どちらも SQUARE ENIX）が有名です。マップ中を探索し、ストーリーを解いていく、変形アドベンチャーゲームとして知られています。

なお、「ロールプレイングゲーム」はホビージャパン、「RPG」はバンダイ、「ロープレ」はセガが商標登録しています。

よく RPG ゲームという方がいますが、前述のとおり RPG がロールプレイングゲームなので、「ゲームゲーム」と間違っていますね。が、本来は間違いなのですが、残念なことにゲーム制作会社の中にも、そのような書き方をしているところもあるようです。

2.1 RPG 制作の流れ

ここでは、実際に、どのように RPG が作成されていくのかを見ていきましょう。右図は、RPG 制作の流れですが、この流れは、一般的なものであり、ゲームの制作会社によって異なりますので、ご注意下さい。

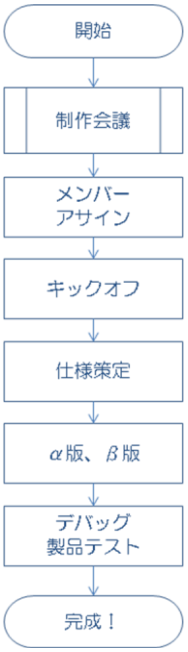


図 1 RPG 制作の流れ

さて、RPG に限ったものではありませんが、ゲームを作るときには、そのゲームにおける世界観が重要です。特に、RPG は、この世界観が、ゲームの楽しさを左右していきます。

制作会議では、この世界観を決めていくことになります。なお、この世界観のもとで、おおまかなストーリーも決められます。もちろん、世界観だけを決める訳ではなく、その他の作業についても整理しくのが、この制作会議になります。

さて、やるべきことが決まったら、その作業へメンバー（人員）をアサイン（割り当て）します。つまり、誰が、どんなことをやるのかを決めていきます。

ゲームでは、さまざまな役割の人たちが必要です。詳しくは、「2.2 RPG 制作体制」に記載していますが、総責任者であるプロデューサー、現場責任者であるディレクター、企画を考えるプランナー、デザインを考えるデザイナー、音楽（サウンド）を考えるサウンドクリエーター、RPG におけるさまざまな文章を考えるライター、プログラムを書くプログラマー、テストを実施するテストエンジニアなど（開発する規模によっては、もっと多くの職種があります）のメンバーが必要です。すべて専門職とな

りますので、これらの職種へ専門の知識を持つ人たちを割り当てていく訳です。

キックオフとは、ゲーム制作プロジェクト[1]を開始するときに、メンバーの責任や役割、スケジュールなど、プロジェクトに関する情報を共有するためのミーティングを指します。参加するメンバーは、基本的には、全員参加で行いますが、主要なメンバーだけで行う場合もあります。ゲーム制作に限らず、ソフトウェア開発のプロジェクトでは、情報を共有することが、とても重要で、情報を共有していないばっかりに、間違ったものを作ってしまったり、スケジュールに間に合わなかったりすることもあり得るのです。

さて、それぞれの役割も決まり、本格的にゲームを作っていきます。そこで、次に仕様策定を行います。おおもとの企画や制作会議での情報などから、もっと具体的な内容に落とし込んでいく作業になります。キャラクターをどのように動かすのか、画面はどのように遷移していくのかなどを決めていきます。また、その画面の遷移に合わせて、細かいシナリオも決められていきます。

仕様が決まれば、いよいよゲームの制作が始まります。

最近は、Flash によるゲームアプリも随分と出てきていますが、本格的なゲームは、やはり高級言語を使用したものとなります。日本の携帯電話のゲームアプリケーションの開発言語は、NTT ドコモとソフトバンクが Java 言語を使用し、au は C 言語が使われています。本書で取り扱っている Android 端末では、Java 言語を使用します。

au は、明らかに違うと分かると思いますが、NTT ドコモとソフトバンクの Java 言語も、使われるプロファイルが異なっているため、開発としては、別のアプリケーションとして作成していかなければいけません。NTT ドコモから Android 端末が発売されていますが、既存の NTT ドコモの Java 言語とは異なっているため、こちらもそのままでは動作しません。更には、ソフトバンクから発売されている iPhone も異なっています。

つまり、3 キャリア（NTT ドコモ、au、ソフトバンク）向けにゲームを作ろうと考えたとき、ゲーム制作では、3 つのプログラムを作らなければいけないことになります。更には、Android 端末、iPhone も含めると、5 つも別のプログラムを作らなければいけないことになります。

ちょっと話が脱線しちゃいましたね。

さて、ゲーム制作の現場では、プログラムを作ること以外にも、キャラクター制作、シナリオデータ制作なども、それぞれのメンバーが制作していくことになります。ゲームを制作していく中では、α 版や β 版といった途中段階のバージョンがリリースされます。このリリースされるバージョンを用いてテストが行われていきます。テストによって見つかったバグは、テストエンジニアから各開発担当メンバーへ通知され、すぐに修正されていきます。この作業を、デバッグ[2]と呼び、このデバッグ作業を繰り返してゲームの完成度を高めていきます。

ゲームの場合、動かなくなったり、画像や文字が間違っていたりと、すぐにバグと分かるものから、テストエンジニアが何となく違和感を感じるものなどの細かい点についても指摘していきます。この細かい点が指摘出来るかどうかについては、テストエンジニアのスキルによって左右されることも多いのが現状です。RPG では、文章を読ませるものも多いため、登場人物の言葉としておかしいものや話が繋がらないものなどについてもチェックされます。多くは、リソース[3]ファイルとして別ファイルで管理しており、文言をチェックするツールを使いますが、実際のゲームの流れの中で、文言の繋がりが大丈夫なのかなどのチェックも必要となります。

[1] プロジェクトとは、到達すべきゴールがあり、かつ、複数の人たちが関わる取り組みを指します。ゲーム制作プロジェクトとは、ゲームを制作するためのソフトウェア開発を指します。

[2] デバッグとは、プログラムの誤り（バグ）を取り除く作業のことを言います。

[3] リソースとは、資源という意味の英単語です。ここでは、プログラム内に埋め込まれているテキストやウィンドウの内容、メニューの情報といったプログラムが使用するデータをリソースと言います。ソースコード中にデータがハードコーディングされているものと違い、実行中にリソースの内容をメモリにロードしたり破棄したりすることが出来ます。

このようにデバッグ作業を繰り返しながら、ゲームの完成度を上げ、最終的に、製品としてのゲームソフトウェアが完成することになります。

2.2 RPG 制作体制

「2.1 RPG 制作の流れ」でも少し触れていますが、RPG に限らず、ゲームの制作現場では、さまざまな役割の人たちがいます。以下に、ゲーム制作における体制図を記載します。

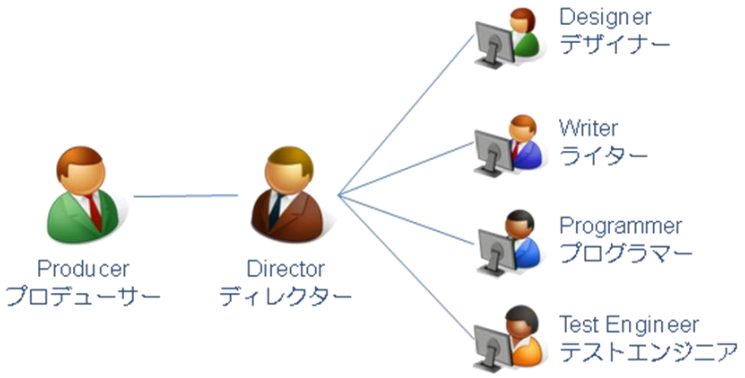


図 2 ゲーム制作体制図

この体制図は、あくまでも一般的なものです。

実際には、もっと細かく分類されており、プランナー（企画を考える人）、サウンドクリエーター（音楽制作）などの職種もあります。必要に応じて人材は追加され、大きなプロジェクトチームが形成されることもあります。ただ、このように人材を抱えることの出来ない小規模の会社では、いくつもの役割を兼務して行っていることも多い場合もあります。

ここでは、上図に記載されている代表的な職種について、説明をしていきたいと思います。

2.2.1 プロデューサー

プロデューサーとは、担当するゲームに対する全責任を負う統括責任者です。

実際には、次に説明をするディレクターと合わせてプロデューサー（あるいはディレクター）としている企業も多いのが実状ですが、企画決定、予算管理、スケジュール管理、宣伝、販売計画に至るまでの全責任を負っているのがプロデューサーとなります。このため、単にゲームや技術に詳しいというだけではなく、非常に高いコミュニケーション能力が求められることになります。

プロデューサーになるためには、やはりすぐになることは出来ず、経験値が必要です。

RPG でも同じですよね、つまりある程度のレベルに達していないとプロデューサーという職種になることは出来ないということです。ただ、小さなゲーム制作会社では、人がいないために、すぐにプロデューサーとして働ける場所も無いわけではありませんが、残念ながら、そのような中から真のプロデューサーになれる方は、ほんの一握りと言っても良いでしょう。ですので、プログラマーなどの経験をし、知識や技術の基礎をしっかりと固めてから、この職種を目指すようにするのが一般的なものとなります。

2.2.2 ディレクター

ディレクターは、いわゆる現場責任者です。

ゲームを制作するには、「図 2　ゲーム制作体制図」で記載しているような専門分野の人材が、ときには数百人規模になるまで必要になる場合もあります。このとき、ディレクターは、それぞれの担当している人たちのスケジュール管理、仕様決定、品質管理など、開発現場の状況を把握し、制作に関わるすべての方向性を決め、スタッフの総指揮を執る人となります。

ディレクターは、知識や技術の基礎知識はもちろんのこと、最新技術に至るまでの情報把握やプロデューサーと同様に、高いコミュニケーション能力が求められることになります。

ディレクターも、プロデューサー同様にすぐになることは出来ず、経験値が必要な職種です。

2.2.3 デザイナー

ここでは、一言でデザイナーという言葉を使いましたが、ゲームの世界では、いくつかのデザイナーがいます。ひとつは、キャラクターデザインなど、絵柄や図表を考案するデザイナーです。一般的に、デザイナーとして認知されているのは、こちらの方でしょう。もうひとつは、ゲームデザインです。とても紛らわしいのですが、ソフトウェア開発プロセスの中で、「設計」のことをデザインと呼び、その設計を行う人、いわゆる設計者のことをデザイナーと呼んでいます。

本書をお読みの方は、すでにプログラムを作ったことがあると思いますが、何かソフトウェアを作りたいと考えたとき、その「考え」自体は、まだ実現はされているものではありません。このため、その「考え」を、実際にコンピュータが理解し、実行出来るようにしていく必要があります。このときに必要となるのが、設計書ということになり、この設計書を作成するのが、ゲームデザイナーの仕事になります。

よく求人項目などを見ると、単にデザイナーと記載されている例も見られますが、このような場合には、上記の両方を求められることも多く、ゲームに必要となるビジュアルやグラフィック、場合によってはプログラミングに対する指示も与えることがあるため、デザインからプログラムまでの広範な知識が必要となります。

ゲーム制作の中で、デザインは、かなり重要な位置を占めています。

特に、ゲームのキャラクターデザインは、ゲームの人気に直結することが多いため、有名なデザイナーを起用してキャラクターのデザインを行ってもらうこともあります。（皆さんもご存じの有名 RPG では、多額のデザイン料でキャラクターデザインを依頼しています）

デザインには、キャラクターデザイン、フィールドデザイン、オープニングタイトルデザイン、背景デザインなど多くのデザインが必要になります。もちろん、ユーザーインターフェイスなどもデザインのひとつですね。

2.2.4 ライター

ライターとは、ゲームに関する文章を執筆する人のことを指します。

RPG では、デザインと共に、シナリオがゲームの面白さを左右することもありますので、シナリオライターの役割は、とても重要です。また、ライターは、操作説明書や、発売後のゲーム紹介、攻略本やゲーム題材の創作物などを執筆することもあります。

日本人であれば、誰でも日本語の文章を書くことは出来るのですが、ライターの方々は、その文

章を書くことを生業にしていますので、やはり出来映えがまったく違ったものとなります。

2.2.5 プログラマー

プログラマーは、言わずもがなですが、プログラムを作成する人を指します。

先ほど「2.1 RPG 制作の流れ」でも少し触れていますが、対象となる機器によって、異なる言語で開発を行わなければいけません。このため、プロデューサーやディレクターは、人選する際に、その対象となる機器によって、適したプログラマーを選ぶことになります。

プログラマーは、自分が受け持つ機能などによって、プログラムを作成していきます。

今回のテキスト内で使用しているプログラムは、かなり小さいものですので、機能別にプログラムを作成するようなことはありませんが、大規模になればなるほど、細かい機能ごとに分けて、プログラマーを割り当てることになります。RPG の場合には、例えば、フィールドマップを作成する人、村や洞窟などと言ったダンジョン部分を作成する人、キャラクターがフィールド上を動き回る部分を作成する人、キャラクター同士が戦う部分（いわゆる戦闘シーンですね）を作成する人などです。

また、プログラマーは、プログラムを作成すれば良いという訳ではありません。

特に大規模なプログラムであるほど、他の機能に大きな影響を与えることが考えられます。このため、自分自身で作成した部分のソースコードレビューや単体レベル（例えば、機能が動作するのか、作成した API[4]が正しく動作するのかなどの確認です）でのテストも実施しなければいけません。そこで、しっかりとした品質のソフトウェアを制作することで、次のデバッグ作業へ進むことが出来ることになります。

2.2.6 テストエンジニア

テストエンジニアは、テストを設計し実施する人を指します。テスターと呼ばれることも多いでしょう。

どのようなテストを行い、どのようにバグを出していくのかは、ゲームソフトウェアの品質を向上させていく上でとても重要なものです。ゲーム制作の中では、テストは大変な作業です。というのも、ゲームを使う人たちは、不特定多数の人たちであり、どのように、そのゲームを扱うのかは分からないのです。特に、RPG の場合には、他のゲームに比べて自由度が高く、ユーザーがどのように扱っても問題がないようにしなければいけません。フィールド上を決められたとおりに動くわけではありませんし、誰と会って話しかけても構いません。このため、ある特定の操作でのバグが発生するような状況を、少しでも減らしていく必要があるのです。このため、未だに、大量の人材を確保し、テストを行う場合も多く、規模によっては、数百人の人たちにテストを行わせている場合もあるのです。

テストエンジニアは、バグ[5]を発見すると、そのバグをすぐにプログラマーへと報告します（現在は、口頭や紙などではなく、しっかりと、バグトラッキングシステム[6]へ登録して、管理されている場合も多いです）。その報告結果から、プログラマーが修正を行うということを繰り返していきます。ゲーム制作会社では、この作業自体をデバッグ作業などと呼んでいます。

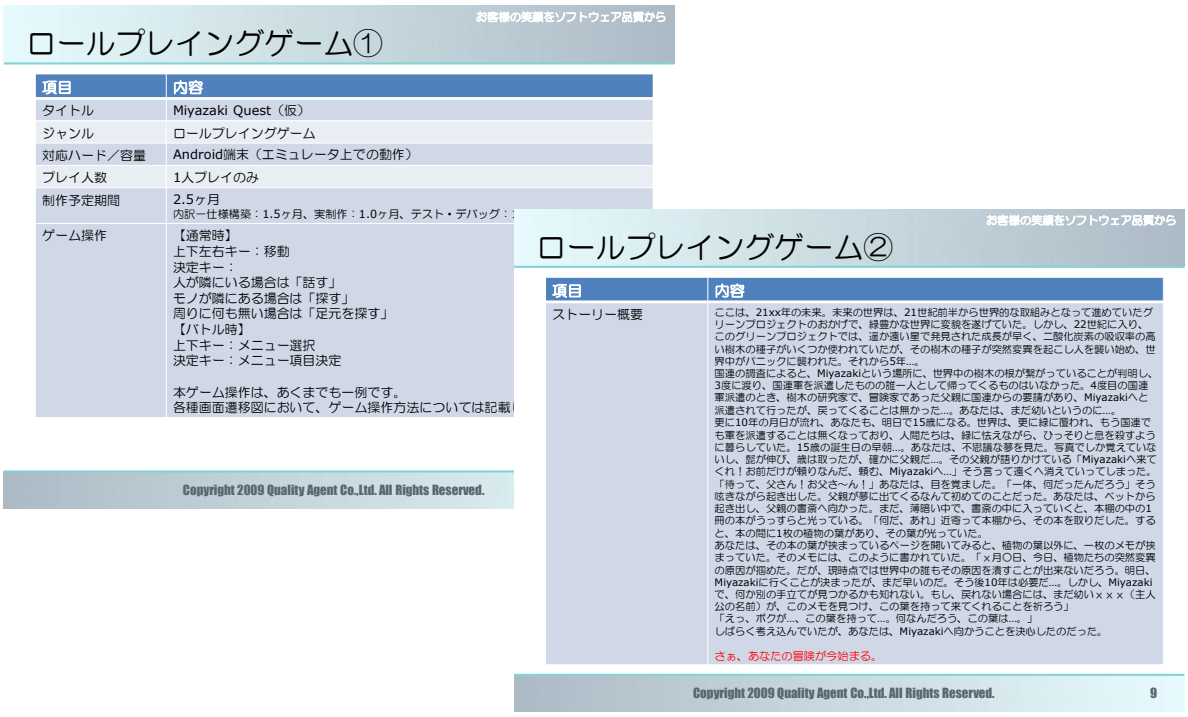
[4] Application Program Interface の略。ソフトウェアを開発する際に使用できる命令や関数の集合のことを指します。また、それらを利用するためのプログラム上の手続きを定めた規約の集合を指します。
[5] バグ（Bug）とは、プログラムに含まれる誤りや不具合のことです。
[6] バグトラッキングシステム（Bug Tracking System：バグ追跡システム）とは、プロジェクトのバグを登録し、修正状況を追跡するシステムです。

3. サンプルプログラムは、どのように作られたか？

それでは、実際に、サンプルプログラムが、どのような流れで作られたのかを解説していきたいと思います。

3.1 企画案作成

下記のファイルは、実際に、企画案として作成したものです。



お客様の笑顔をソフトウェア品質から

ロールプレイングゲーム①

項目	内容
タイトル	Miyazaki Quest（仮）
ジャンル	ロールプレイングゲーム
対応ハード／容量	Android端末（エミュレータ上での動作）
プレイ人数	1人プレイのみ
制作予定期間	2.5ヶ月 内訳ー仕様構築：1.5ヶ月、実制作：1.0ヶ月、テスト・デバッグ：
ゲーム操作	【通常時】 上下左右キー：移動 決定キー： 人が隣にいる場合は「話す」 モノが隣にある場合は「探す」 周りに何も無い場合は「足元を探す」 【バトル時】 上下キー：メニュー選択 決定キー：メニュー項目決定 本ゲーム操作は、あくまでも一例です。 各種画面遷移図において、ゲーム操作方法については記載

Copyright 2009 Quality Agent Co.,Ltd. All Rights Reserved.

お客様の笑顔をソフトウェア品質から

ロールプレイングゲーム②

項目	内容
ストーリー概要	ここは、21xx年の未来。未来の世界は、21世紀前半から世界的な取組みとなって進めていたグリーンプロジェクトのおかげで、緑豊かな世界に変貌を遂げていた。しかし、22世紀に入り、このグリーンプロジェクトでは、遥か遠い星で発見された成長が早く、二酸化炭素の吸収率の高い樹木の種子がいくつか使われていたが、その樹木の種子が突然変異を起こし人を襲い始め、世界中がパニックに襲われた。それから5年…。 国連の調査によると、Miyazakiという場所に、世界中の樹木の根が繋がっていることが判明し、3度に渡り、国連軍を派遣したものの唯一人として帰ってくるものはいなかった。4度目の国連軍派遣のとき、樹木の研究家で、冒険家であった父親に国連からの要請があり、Miyazakiへと派遣されて行ったが、戻ってくることは無かった…。あなたは、まだ幼いというのに…。 更に10年の月日が流れ、あなたも、明日で15歳になる。世界は、更に緑に覆われ、もう国連でも軍を派遣することは無くなっており、人間たちは、緑に怯えながら、ひっそりと息を殺すように暮らしていた。15歳の誕生日の早朝…。あなたは、不思議な夢を見た。写真でしか覚えていないし、髭が伸び、歳は取ったが、確かに父親だ…。その父親が語りかけている「Miyazakiへ来てくれ！お前だけが頼りなんだ。親父、Miyazakiへ…」そう言って遠くへ消えていってしまった。 「待って、父さん！お父さ～ん！」あなたは、目を覚ました。「一体、何だったんだろう」そう呟きながら起き出した。父親が夢に出てくるなんて初めてのことだった。あなたは、ベットから起き出し、父親の書斎へ向かった。まだ、薄暗い中で、書斎の中に入っていくと、本棚の中の1冊の本がうっすらと光っている。「何だ、あれ」近寄って本棚から、その本を取りだした。すると、本の間に1枚の植物の葉があり、その葉が光っていた。 あなたは、その本の葉が挟まっているページを開いてみると、植物の葉以外に、一枚のメモが挟まっていた。そのメモには、このように書かれていた。「ｘ月○日、今日、植物たちの突然変異の原因が掴めた。だが、現時点では世界中の誰もその原因を潰すことが出来ないだろう。明日、Miyazakiに行くことが決まったが、まだ早いのだ。そう後10年は必要だ…。しかし、Miyazakiで、何か別の手立てが見つかるかも知れない。もし、戻れない場合には、まだ幼いｘｘｘ（主人公の名前）が、このメモを見つけ、この葉を持って来てくれることを祈ろう」 「えっ、ボクが…、この葉を持って…。何なんだろう、この葉は…。」 しばらく考え込んでいたが、あなたは、Miyazakiへ向かうことを決心したのだった。 さぁ、あなたの冒険が今始まる。

Copyright 2009 Quality Agent Co.,Ltd. All Rights Reserved. 9

図 3 サンプルプログラム企画書

どこまでの企画を作るのかは、ゲーム制作会社によって異なりますが、このように概要を記載していく企画書もあります。

企画の目的は、その企画をお客様などに採用してもらい、ゲームとして実際に世の中へ出していくことになります。ボツになってしまえば、ただの案でしかありません。このため、いかに、この企画が面白いのかを説明しなければいけません。もちろん、そのゲームの世界観が、共有出来るように、企画の中に盛り込んでいくことも必要となります。

なお、残念ながら面白い企画書を書く簡単な方法はありません。

何度も企画書の提出を繰り返し、ダメ出しを貰って書けるようになっていくしかありません。ちなみに、ゲーム制作会社でも、企画書は、日々何度も提出されることも少なくありません。

3.2 企画が通った！

さあ、企画が通りました。

次に、この企画に対しての仕様を決定していきます。

サンプルプログラムでは、あまり大きなプログラムではありませんので、簡単な画面遷移図と、マップ上の動作のみが記載されています。もちろん、この内容では、まだまだ少ないので、実際には、もっと多くの仕様書が必要になります。（例えば、敵キャラクターと遭遇したときの動作などですね）

図 4 サンプルプログラム仕様書

仕様書は、プログラマーがプログラムを作成するための設計図です。

そのため、曖昧な記述がないようにしなければいけません。もちろん、それでも人間の作るものですから、そのような部分が絶対ないかというと、やはり曖昧な部分は残ってしまいます。そのようなときに、設計レビュー[7]を実施し、曖昧な記述がないか、プログラムを作る上で食い違いがないかなどをチェックしていきます。

[7] ソフトウェア開発における成果物（ここでは仕様書）を、複数の人たちにチェックしてもらう機会のことです。設計レビューの方法は、さまざまな方法があります。（本書では、詳しい設計レビューの説明はありませんので、調べてみましょう）

3.3 キャラクターを考えよう

さて、仕様書の作成と共に、並行してキャラクターのデザインも考えなければいけません。下図は、サンプルプログラムにおいて作成した敵キャラクターのラフデザインです。

※「MiyagiQuest」にはどちらも登場しません。

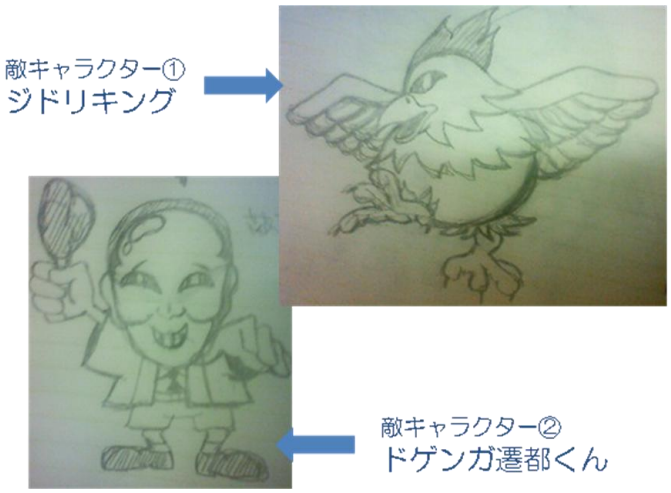


図 5 サンプルプログラムラフデザイン

さて、ここでも、やり方はさまざまですが、デザイナーの方は、デッサン用紙などにラフのデザインをたくさん書き出していきます。もちろん、このときのイメージは、常に、このゲーム内での世界観を考慮したものとなります。そのたくさんのデザインの中から、最もイメージの合うキャラクターデザインを決定していきます。「図 5 サンプルプログラムラフデザイン」では、敵キャラクターを例として出していますが、もちろん、主人公のデザインも存在します。これらのデザイン画からゲーム内で使用するために画像ファイルにしていかなければいけません。

画像ファイルを作る方法についても、いろいろやり方はありますが、本書でのやり方は、先のラフデザインを、スキャナーで画像ファイルとして読み込み、その画像ファイルを加工して作成しています。画像ファイルを取り扱うためのツールは、フリーソフト[8]から何万円もするような高機能のソフトウェアまでさまざまなツールが存在します。もちろん、自分に合った使い易いソフトウェアを使うのが良いでしょう。

さて、先ほどのキャラクターデザインからゲーム内で使うための画像ファイルへ作り変えていく必要があります。その画像が「図 6 画像ファイルへの変換」です。

この 2 つの画像のサイズは、48×48 ピクセルとなっています。もともとのデザイン画は、もっと大きなものですので、それを小さい画像の中で書き出す必要があります。今回のサンプルプログラムでは、デザインは、ゲーム内のキャラクターと同じようにデザインしていますが、異なる場合には、もっと複雑な手順があります。

図 6 画像ファイルへの変換

[8] フリーソフトとは、対価としてのお金を支払わずに使うことが出来るアプリケーションソフトウェアのことで、フリーウェアとも呼ばれています。ソフトウェアの作成者は、無償でソフトウェアを提供する代わりに、不具合の修正や使用法の解説といったユーザーサポートや開発の継続などの義務は負いません。

例えば、デザイナーが、キャラクターデザインを描くときに、八頭身のカッコイイ男性のキャラクターや綺麗な女性のキャラクターを書くときがあります。このとき、二頭身のキャラクターを作ってくれる場合もありますが、そうではない場合も多々あります。このとき、実際の画像ファイルへの変換を行う担当者が、八頭身キャラクターから二頭身キャラクターへ画像を作りながら変更をしていきます。このとき、八頭身キャラクターの特徴をきっちりと二頭身キャラクターへ取り込んでいかなければいけません。小さくなったら、全然誰なのか分からない状態というのは、ゲームとしてもおかしいですよね？

このように、各キャラクターのデザイン画を、ゲーム内で使えるように画像へと変換を行います。

なお、このサンプルプログラムでは、主人公を含めて 3 体のキャラクターしかありませんが、本当のゲームでは、もっと多くのキャラクターがあるのですから、大変な作業であることが分かると思います。

3.4 キャラクターを動かそう

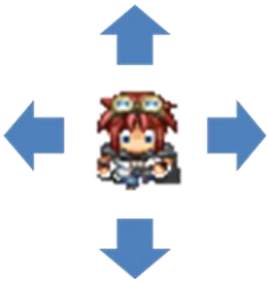
さて、「3.3 キャラクターを考えよう」では、それぞれ 1 つのキャラクターをゲーム内で作れるようにしてきましたが、実は、作り出すキャラクターは、1 つだけではありません。

ここでは、ちょっと主人公を中心に見ていきましょう。

右の図は、主人公のキャラクターを画像ファイルへと変換したものです。画像サイズは、32×32 ピクセルの小さい画像です。

さて、この主人公は、フィールド上を動きまわります。そのため、その動きを表現できるようにしなければいけないのです。

主人公



フィールド上で、この主人公は、上下左右に動作することが決まっています。現在の携帯電話では、8 方向（上下左右＋それぞれ斜め方向）へす進むことも出来ますが、昔ながらの RPG にするために 4 方向としています。では、この 4 方向だけの画像があれば良いかというと、そうではありません。4 方向へ移動するとき、キャラクター自身に動きが無ければ、動いているようには見えないのです。

では、どのようにすればよいのでしょうか？

皆さんは、昔、パラパラマンガを、教科書やノートの端に描かなかったでしょうか？

パラパラマンガは、重ねられた紙を使って作られるアニメーションの一種で、残像によって、あたかも連続して絵が動いているように見えるものです。これと同じようにすれば良いのです。しかし、動きをすべて画像にしてしまっては、メモリの容量に制限のある携帯電話では、搭載することは出来ません。そこで、2 つあるいは 3 つの画像だけを使用して動きを表現します。以下の画像は、2 つの画像を交互にしている状態ですが、いかがでしょうか？何となく動いているように見えますよね？

このように、各方向の画像を 2 つづつ計 8 つの画像ファイルを準備する必要があります。

ここでも難しい点があり、目の位置や特徴となる位置がずれないように、キャラクターに動きを与えていく必要があるのです。

3.5 フィールドマップを作ろう

さて、まだ準備しておかなければいけないものがあります。それが、フィールドマップです。

フィールドマップは地図ですから、このゲームの世界を表す重要なものです。作りだす RPG によって、大きなフィールドマップから、小さいフィールドマップまでさまざまですが、あまり大き過ぎると、ユーザーが迷ってしまうこともありますので、適度な大きさにしておく必要があります。（プログラムの中では、大きくても小さくても影響は小さいものですので、あくまでもユーザー目線でサイズを決めていく必要があります）

次にフィールドマップ上には、どのようなものを配置するのかも決めていく必要があります。また、その上にきたときに、どのような動作が行われるのかというのも決めなければいけません。

以下に、サンプルプログラム中で使用することの出来るフィールドの一覧を記載します。内容欄には、そのフィールドの名称と、そのフィールド上での動作するのかを示す仕様を表しています。（あくまでも、この仕様は、サンプルプログラム内で定めたものであり、別の意味を持たせても構いません）

表 1 フィールド一覧

No.	画像	内容	No.	画像	内容
0		芝 移動することが出来る	1		森 移動することが出来る
2		土 移動することが出来る	3		海 移動することは出来ない。 ただし、アイテム「船」があると、移動することが可能。
4		砂漠 移動することが出来る	5		山その１ 移動することが出来る
6		山その２ 移動することは出来ない	7		岩その１ 移動することは出来ない。 ただし、アイテム「ハンマー」があると壊すことが出来る
8		岩その２ 移動することが出来る	9		保存エリア 移動することが出来る。 この上で、SELECT キー押下により保存することが出来る
A		ブロック 移動することが出来ない	B		コンクリート 移動することが出来る
C		全回復エリア 移動することが出来る。 この上で、SELECT キーを押下し、全 HP/MP が全回復する	D		マグマ 移動することが出来る。 ただし、この上を移動すると HP が、徐々に減ってしまう
E		雪 移動することが出来る。	F		家 移動することが出来る。 この上に乗ることで、家の中に入ることが出来る

No.	画像	内容	No.	画像	内容
G		ドア 移動することが出来る。 この上に乗ることで、ドアの中に入ることが出来る	H		洞窟 移動することが出来る。 この上に乗ることで、洞窟の中に入ることが出来る
I		行き先表示 移動することが出来ない。 この前で、調べることによって、行き先を見ることが出来る	J		泉 移動することが出来る。 この上に乗ることで、泉の中に入ることが出来る
K		不思議ハウス 移動することが出来る。 何が起こるか分からない。	L		宿屋 移動することが出来る。 この上に乗ることで、宿屋の中に入ることが出来る
M		武器屋 移動することが出来る。 この上に乗ることで、武器屋の中に入ることが出来る	N		防具屋（鎧） 移動することが出来る。 この上に乗ることで、防具屋（鎧）の中に入ることが出来る
O		防具屋（盾） 移動することが出来る。 この上に乗ることで、防具屋（盾）の中に入ることが出来る	P		アイテム屋 移動することが出来る。 この上に乗ることで、アイテム屋の中に入ることが出来る
Q		教会 移動することが出来る。 この上に乗ることで、教会の中に入ることが出来る	R		橋 移動することが出来る。

これらのフィールドは、サンプルプログラム内で用意はされてはいますが、残念ながら、今回のプログラム中では、すべてのフィールドを利用してはいません。また、画像を変更するだけであれば、変更することは出来ますので、サンプルプログラムをいろいろと変更してみましょう。（「4.5.16.3 マップの描画」も参照してみて下さい）

このように、たくさん用意されているフィールドの画像ファイルは、どのように利用されているのでしょうか？

まず、最初に、「**エラー! 参照元が見つかりません。**」で示される各フィールドに番号を付けます。表内に記載されている No のように、例えば、0, 1, 2, …A, B, C, …, Z などで構いません。要は、各フィールドの画像を識別することが出来れば良いだけです。

次に、マップを作成します。例えば、Microsoft Excel のような表計算ソフトを使用し、そこに、どのぐらいの大きさのマップにするのかを決定します。右図では、8×8 マスのマップを生成することにしています。（サンプルプログラム内では、80×80 マスのマップです）マップ上の配列を示す各位置に、フィールドの画像を識別するための番号を指定していきます。

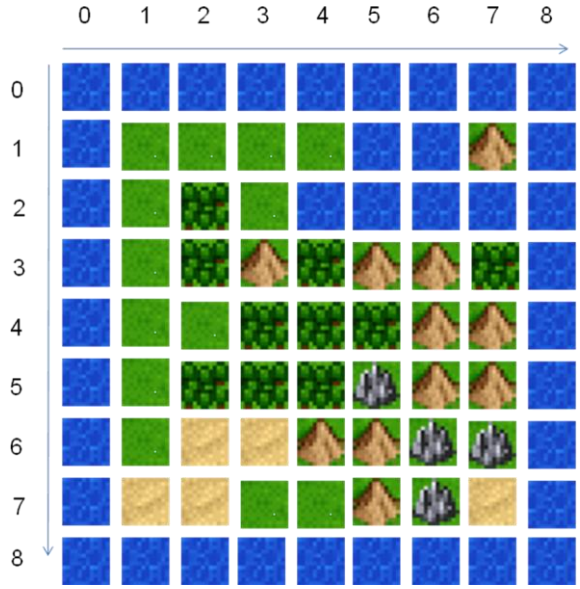


図 7 フィールドマップ生成方法

サンプルプログラムでは、まず、右方向へ進み、端まで進むと、次の行に進み、更に右方向へと進む形になります。もちろん、この進み方は、サンプルプログラム内での決めゴト（仕様）となりますので、変更しても構いません。

このように設定することで、各マップでの画像とが紐づけられることになり、フィールドマップ

を形成していくことが出来るようになります。

「図 7　フィールドマップ生成方法」では、ちょっと隙間が空いている状態にしていますが、これを隙間の無い状態にしたものを「図 8 フィールドマップ結合」になります。

なんとなく、RPG のマップらしくなってきましたね。このように、フィールドマップは作成していきます。

洞窟や山、街などのいわゆるダンジョンと呼ばれる部分のマップも、考え方としては同じように行います。

後は、各仕様に基づいて、どのようなものを配置していくのか、その場所でどのようなことが行われるのかを決めていくことによって、更に RPG としての完成度は上がっていくことになります。

図 8 フィールドマップ結合

4. プログラム解説

4.1 RPG のプログラムの構成

昔懐かしい初期のころのドラゴンクエストのような RPG を作成します。

主人公は、フィールドマップ上を移動し、敵と遭遇すると戦闘が始まるゲームとなっています。戦闘中は、「戦う」「特殊」「回復」「逃げる」のコマンドが使用可能となっています。フィールド上には、「HP・MP 全回復」や「データ保存」、町や洞窟等のフィールドが用意されており、様々な機能が実装されています。

このプログラムは、5 つのクラスで構成されています。

プロジェクトは、「MiyagiQuest」で作成して下さい。ファイルは、5 つのファイルに分けて作成してあります。

- MiyagiQuest（MiyagiQuest.java）
- MiyagiQuestView（MiyagiQuestView.java）
- Enemy（Enemy.java）
- EnemyComparator（EnemyComparator.java）
- MyUtl（MyUtil.java）

図 9 RPG の画面

4.2 画像ファイルの準備

このプログラムで使用している画像ファイルは、全部で 41 枚ありますが、分類すると、以下のような 5 つの画像に分類されます。なお、Android で利用出来る画像のファイル形式は PNG ですが、GIF、JPG は非推奨となっていますが利用することは出来ます。ここでは、この 3 つのファイル形式の中で準備することを前提とします。

- タイトル
- 主人公
- 敵キャラクター①
- 敵キャラクター②
- フィールド
- アイコンファイル（デフォルトのアイコンを使用）

図 10 タイトル画像

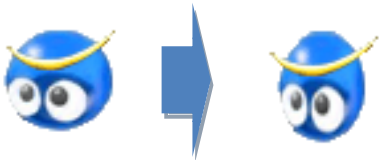
このうち、フィールドについては、前述してある「**エラー! 参照元が見つかりません。**」にある画像ファイル（28 枚）を準備します。（「**エラー! 参照元が見つかりません。**」を参照して下さい）

タイトルは、300×200 ピクセルのサイズの画像を準備します。この画像ファイルは、画面半分に配置させて使用します。

主人公の画像は、24×24 ピクセルのサイズの画像ファイルを準備します。また、背景と重ねたときに、キャラクターの周りが透けて見えるように、キャラクターの周囲は透過設定にします。

主人公は、フィールドを歩く必要があるため、上下左右それぞれの方向に向かった画像が必要となります。また、単に上下左右の方向だけではなく、それぞれが動いて見える（歩いて見える）ように、2 枚ずつ準備します。（「3.4 キャラクターを動かそう」も参照して下さい）

図 11 主人公画像



敵キャラクターは、9 体分準備してあります。敵キャラクターの画像は、48×48 ピクセル（主人公よりも大きいサイズです）のサイズの画像ファイルを準備します。

図 12 敵キャラクターの例

この敵キャラクターは、戦闘シーンでのみ使用しますので、画像は、それぞれ 1 枚だけでも良いのですが、戦闘シーンでも、動きがあるように見えるようにするために、2 枚ずつ準備します。（横幅が縮まって見える画像ファイルを準備）

画像ファイルは、Eclipse のパッケージエクスプローラーで、/res/drawable フォルダ配下に配置します。（Windows のエクスプローラーからドラッグ＆ドロップで配置することが可能です）

res フォルダの下に保存した画像ファイル（その他のファイルもあります）は、「リソース」と呼ばれ、ビルドしたときにアプリケーションの実行ファイルに付加されます。

4.3 サウンドファイルの準備

サンプルプログラムで使用しているサウンドファイルは、以下のように全部で 10 あります。Android で利用出来るサウンドファイルの形式は、3GP、MP4、M4A、MP3、OGG、WAV、MID となっています。サンプルプログラムでは、このうち、OGG ファイルを準備しています。

表 2 サウンドファイル一覧

	引数	内容
BGM	Sht_a02.ogg	タイトル画像表示の際に再生する BGM ファイル
	Sht_a03.ogg	マップ上で主人公が歩いている際に再生する BGM ファイル
	Sht_a06.ogg	戦闘中に再生する BGM ファイル
SE	Encount.ogg	敵と遭遇したときに再生する効果音
	Levelup.ogg	レベルアップ時に再生する効果音
	Attack_enemy.ogg	敵が攻撃するときに再生する効果音
	Attack_player.ogg	プレイヤーが攻撃するときに再生する効果音
	cure.ogg	回復行動時の効果音
	magic.ogg	特殊行動時の効果音
	in_out.ogg	建物出入り時の効果音

BGM ファイル、SE（Sound Effect）ファイルどちらも、プログラム内にリソースファイルとして取り込んで使用しています。

なお、BGM 用のサウンドファイルは、タイトル画面表示中、マップ移動中、戦闘中それぞれの処理が行われている間、継続して再生しなければいけません。しかし、長時間のサウンドファイルでは、リソースをたくさん使用してしまいますので、一定時間のサウンドファイルを準備し、そのファイルをループ再生するようにします。

サウンドファイルは、Eclipse のパッケージエクスプローラーの/res/raw フォルダ配下に配置します（画像ファイルと同様に、Windows のエクスプローラーからドラッグ＆ドロップで配置することが可能です）。なお、res フォルダ下には、初期状態のプロジェクトには、raw というフォルダは存在していませんので、サウンドファイルを配置する前に、raw フォルダを、res フォルダ下に作成するようにして下さい。

4.4 ソースコード

ここでは、まず、ソースコードの全文を記載します。ソースコードに対する解説については、次節にて説明します。

4.4.1 MiyagiQuest.java

このファイルは、ゲーム本体となります。以下のクラスが定義されています。

○ MiyagiQuest

```
package com.cgene.game;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.MotionEvent;
import android.view.GestureDetector;
import android.util.DisplayMetrics;

import android.util.Log;

import android.view.Display;
import android.view.WindowManager;

/**
 * RPG起動のActivityクラス
 */
public class MiyagiQuest extends Activity
                    implements GestureDetector.OnGestureListener{
          /**       定期処理ハンドラ変数 */
          private TickHandler tickHandler;

          /**       ゲーム表示用 view 変数 */
          private MiyagiQuestView view;
    GestureDetector mGestureDetector;
          private int TouchPosX_Down = 0;                         //        画面タッチ時のx座標
          private int TouchPosY_Down = 0;                         //        画面タッチ時ののy座標
          private int TouchPosX_Move = 0;                         //        画面長押し時のx座標
          private int TouchPosY_Move = 0;                         //        画面長押し時ののy座標
          @SuppressWarnings("unused")
          private int TouchPosX_UP = 0;                           //        画面リリース時のx座標
          @SuppressWarnings("unused")
          private int TouchPosY_UP = 0;                           //        画面リリース時のy座標
          private int TouchPosX_Fling = 0;                //        画面フリック時のx座標
          private int TouchPosY_Fling = 0;                //        画面フリック時のy座標

          /**
     * Called when the activity is first created.
           * Androidが、一番初めに実行する際に呼び出させる処理
           */
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        view = new MiyagiQuestView(getApplication(), this);

                    // 端末のディスプレイ情報取得
        WindowManager windowManager = getWindowManager();
                    Display display = windowManager.getDefaultDisplay();

                    DisplayMetrics displayMetrics = new DisplayMetrics();
                    display.getMetrics(displayMetrics);

        mGestureDetector = new GestureDetector(this);

        setContentView(view);
    }

    /*
     *    タッチ時処理
     */
         @Override
         public boolean onTouchEvent(MotionEvent event)
         {
                    DebugLog.println("onTouchEvent IN");

        //ジェスチャーディテクターの処理
        mGestureDetector.onTouchEvent(event);

        String action = "";

        switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            action = "ACTION_DOWN";
            break;
        case MotionEvent.ACTION_UP:
            action = "ACTION_UP";
                            view.Flag_LongTouch = false;
            break;
        case MotionEvent.ACTION_MOVE:
            action = "ACTION_MOVE";
            break;
        case MotionEvent.ACTION_CANCEL:
            action = "ACTION_CANCEL";
            break;
        }
        Log.v("MotionEvent", "view.TouchStatus = "
                                                + String.valueOf(view.TouchStatus));
        return true;
    }

    /**
     *    ダウン時処理
     */
    public boolean onDown(MotionEvent e)
          {
        DebugLog.println("Down("+(int)e.getX()+","+(int)e.getY()+")");
                    TouchPosX_Down = (int)e.getX();
                    TouchPosY_Down = (int)e.getY();
        return false;
    }
```

```
/**
 *    長押し時処理
 */
public void onLongPress(MotionEvent e)
      {
    DebugLog.printin("LongPress("+(int)e.getX()+","+(int)e.getY()+")");

    //          画面長押し中の座標を取得
                TouchPosX_Move = (int)e.getX();
                TouchPosY_Move = (int)e.getY();

                //        長押し位置のY座標が高さの中心点よりも上、かつ、
                //        X座標が両幅から1/3以内であれば、上スライドとして扱う
                if(TouchPosY_Move < ((view.displayHeight / 3))){
                          DebugLog.printin("LongPress:TouchStatus_UP");
                          view.TouchStatus = MiyagiQuestView.TouchStatus_UP;
                          view.Flag_LongTouch = true;
                //        長押し位置のY座標が高さの中心点よりも下、かつ、
                //        X座標が両幅から1/4以内であれば、下スライドとして扱う
                } else if(TouchPosY_Move > ((view.displayHeight / 3) * 2)) {
                          DebugLog.printin("LongPress:TouchStatus_DOWN");
                          view.TouchStatus = MiyagiQuestView.TouchStatus_DOWN;
                          view.Flag_LongTouch = true;
                //        長押し位置のX座標が幅の中心点よりも左、かつ、
                //        Y座標が上下1/3以内であれば、左スライドとして扱う
                } else if(TouchPosX_Move < ((view.displayWidth / 3))) {
                          DebugLog.printin("LongPress:TouchStatus_LEFT");
                          view.TouchStatus = MiyagiQuestView.TouchStatus_LEFT;
                          view.Flag_LongTouch = true;
                //        長押し位置のX座標が幅の中心点よりも右、かつ、
                //        Y座標が上下1/3以内であれば、右スライドとして扱う
                } else if(TouchPosX_Move > ((view.displayWidth / 3) * 2)) {
                          DebugLog.printin("LongPress:TouchStatus_RIGHT");
                          view.TouchStatus = MiyagiQuestView.TouchStatus_RIGHT;
                          view.Flag_LongTouch = true;
                }
           DebugLog.printin(
          "x = " + String.valueOf(e.getX()) + ", " +
          "y = " + String.valueOf(e.getY()) + ", " +
          "TouchPosX_Move = " + String.valueOf(TouchPosX_Move) + ", " +
          "TouchPosY_Move = " + String.valueOf(TouchPosY_Move) + ", " +
          "view.TouchStatus = " + String.valueOf(view.TouchStatus));
}

/**
 *    フリック時処理
 */
public boolean onFling(MotionEvent e0,MotionEvent e1,
                                          float velocityX,float velocityY)
      {
                DebugLog.printin("Fling("+(int)velocityX+","+(int)velocityY+")");
                TouchPosX_Fling = (int)e1.getX();
                TouchPosY_Fling = (int)e1.getY();

                // タッチ⇒リリース時のX座標差異が+40以上の場合、上スライドとして扱う
                if((TouchPosY_Down- TouchPosY_Fling) >= 40) {
                          view.TouchStatus = MiyagiQuestView.TouchStatus_UP;
               return false;
                // タッチ⇒リリース時のX座標差異が-40以下の場合、下スライドとして扱う
```

```
                } else if((TouchPosY_Down - TouchPosY_Fling) <= -40) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_DOWN;
                return false;
                // タッチ⇒リリース時のY座標差異が30以上の場合、左スライドとして扱う
                } else if((TouchPosX_Down - TouchPosX_Fling) >= 30) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_LEFT;
                return false;
                // タッチ⇒リリース時のY座標差異が-30以下の場合、右スライドとして扱う
                } else if((TouchPosX_Down - TouchPosX_Fling) <= -30) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_RIGHT;
                return false;
                }
        return false;
    }

    /**
     *    スクロール時処理
     */
    public boolean onScroll(MotionEvent e0,MotionEvent e1,
                                        float distanceX,float distanceY)
          {
        DebugLog.println("Scroll("+(int)distanceX+","+(int)distanceY+")");
        DebugLog.println("Scroll_e0("+(int)e0.getX()+","+(int)e0.getY()+")");
        DebugLog.println("Scroll_e1("+(int)e1.getX()+","+(int)e1.getY()+")");

        // 画面長押し中の座標を取得
                TouchPosX_Move = (int)e1.getX();
                TouchPosY_Move = (int)e1.getY();

                // 長押し位置のY座標が高さの中心点よりも上、かつ、
                // X座標が両幅から1/3以内であれば、上スライドとして扱う
                if(TouchPosY_Move < ((view.displayHeight / 3))) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_UP;
                        view.Flag_LongTouch = true;
                        view.sleep(100);            // 移動速度調整のためスリープ
                // 長押し位置のY座標が高さの中心点よりも下、かつ、
                // X座標が両幅から1/4以内であれば、下スライドとして扱う
                } else if(TouchPosY_Move > ((view.displayHeight / 3) * 2 )) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_DOWN;
                        view.Flag_LongTouch = true;
                        view.sleep(100);            // 移動速度調整のためスリープ
                // 長押し位置のX座標が幅の中心点よりも左、かつ、
                // Y座標が上下1/3以内であれば、左スライドとして扱う
                } else if( TouchPosX_Move < ((view.displayWidth / 3))) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_LEFT;
                        view.Flag_LongTouch = true;
                        view.sleep(100);            // 移動速度調整のためスリープ
                // 長押し位置のX座標が幅の中心点よりも右、かつ、
                // Y座標が上下1/3以内であれば、右スライドとして扱う
                } else if( TouchPosX_Move > ((view.displayWidth / 3) * 2)) {
                        view.TouchStatus = MiyagiQuestView.TouchStatus_RIGHT;
                        view.Flag_LongTouch = true;
                        view.sleep(500);            // 移動速度調整のためスリープ
                }
        return false;
    }

    /**
     *    プレス時処理
     */
```

```
	public void onShowPress(MotionEvent e)
		{
		DebugLog.println("ShowPress");
	}

	/**
	 *	シングルタップ時処理
	 */
	public boolean onSingleTapUp(MotionEvent e)
		{
			// シングルタップアップ時には、タッチ操作として扱う
			TouchPosX_UP = (int)e.getX();
			TouchPosY_UP = (int)e.getY();
			view.TouchStatus = MiyagiQuestView.TouchStatus_TAP;
		return false;
	}

	/**
	 * ダブルタップ時に呼ばれる
	 * @param e
	 * @return
	 */
	public boolean onDoubleTap(MotionEvent e)
		{
		DebugLog.println("DoubleTap");				//	ダブルタップは無効
		return false;
	}

	/**
	 * ダブルタップイベント時(down,move,up含む)に呼ばれる
	 * @param e
	 * @return
	 */
	public boolean onDoubleTapEvent(MotionEvent e)
		{
			DebugLog.println("DoubleTapEvent");		// ダブルタップは無効
		return false;
	}

	/**
	 *	シングルタップ時処理
	 *	@param e
	 *	@return
	 */
	public boolean onSingleTapConfirmed(MotionEvent e)
		{
		DebugLog.println("SingleTap");
		return false;
	}

		/**
		 *	アプリ再開時処理
		 */
		@Override
		public void onResume()
		{
			super.onResume();
		DebugLog.println("onResume");

		//	ハンドラを再開させます。
```

```
                tickHandler=new TickHandler();
                tickHandler.sleep(0);

                // BGM再開
                if(MyUtil.mp_Bgm != null) {
                        // 再生状況チェックをチェックし、再生中ならば、BGMを再生する
                        if(!MyUtil.mp_Bgm.isPlaying()) {
                                DebugLog.println("mp_Bgm:MediaPlayer.start");
                                MyUtil.mp_Bgm.start();
                        }
                }

                // SE再開
                if( MyUtil.mp_Se != null) {
                        // 再生状況チェックをチェックし、再生中ならば、SEを再生する
                        if(!MyUtil.mp_Bgm.isPlaying()) {
                                DebugLog.println("mp_Se:MediaPlayer.start");
                                MyUtil.mp_Se.start();
                        }
                }
        }

        /**
         *      アプリが一時停止時処理
         */
        @Override
        public void onPause()
        {
                super.onPause();

                tickHandler=null;  //      ハンドラを停止させます。
                pauseBGM();                     //      BGMの一時停止処理
                pauseSE();                      //      SEの一時停止処理
        }

        /**
         * アプリの終了ボタンが押された際と、他のActivityが起動された際に呼び出させる処理
         */
        @Override
        public void onStop()
        {
                super.onStop();

                tickHandler=null;  //      ハンドラを停止させます。
                pauseBGM();                     //      BGMの一時停止処理
                pauseSE();                      //      SEの一時停止処理
        }

        /**
         *  BGM停止処理
         *  MediaPlayerのインスタンスチェックし、再生中なら一時停止する
         */
        private void pauseBGM()
        {
                if(MyUtil.mp_Bgm != null) {
                        if(MyUtil.mp_Bgm.isPlaying()) {
                                MyUtil.mp_Bgm.pause();
                        }
                }
        }
```

```
	/**
	 *  SE停止処理
	 *  MediaPlayerのインスタンスチェックし、再生中なら一時停止する
	 */
	private void pauseSE()
	{
		if(MyUtil.mp_Se != null) {
			if(MyUtil.mp_Se.isPlaying()) {
				MyUtil.mp_Se.pause();
			}
		}
	}

	/**
	 * アプリの終了ボタンが押された際に呼び出させる処理
	 */
	@Override
	public void onDestroy()
	{
		super.onDestroy();

		//	ハンドラを停止させます。
		tickHandler=null;

		// MediaPlayerのインスタンスチェックし、BGM再生中ならBGMを停止する
		if(MyUtil.mp_Bgm != null) {
			MyUtil.mp_Bgm.stop();
			MyUtil.mp_Bgm.release();
			MyUtil.mp_Bgm = null;
		}

		// MediaPlayerのインスタンスチェック し、SE再生中ならSEを停止する
		if(MyUtil.mp_Se != null) {
			MyUtil.mp_Se.stop();
			MyUtil.mp_Se.release();
			MyUtil.mp_Se = null;
		}
	}

	/**
	 * 定期的な処理を行うメソッド
	 */
	public void onTick(){
		//	以下で、描写処理が呼び出され、view.onDrawが実行されます。
		view.invalidate();
		if (tickHandler!=null) tickHandler.sleep(view.getSleep());
	}
	/**
	 * 定期処理ハンドラクラス Handlerを継承して、作成する。
	 */
	public class TickHandler extends Handler {
		//ハンドルメッセージ
		@Override
		public void handleMessage(Message msg) {
		    onTick();
		}

		//スリープ
		public void sleep(long delayMills) {
```

```
                        removeMessages(0);
                        sendMessageDelayed(obtainMessage(0),delayMills);
                    }
            }
}
```

4.4.2 MiyagiQuestView.java

このファイルは、画面描写に関するクラスです。

```
package com.cgene.game;

import java.io.File;
import java.io.FileOutputStream;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;

import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.RectF;
import android.media.MediaPlayer;
import android.view.View;

/**
 * RPG描写のViewクラス
 *
 * 背景画像提供元：
 * 佐和多里様 URL：http://iyukiainosaka.kitunebi.com/
 */
class MiyagiQuestView extends View {

    public int displayWidth;          //画面幅
    public int displayHeight;         //画面高さ

    public boolean Flag_LongTouch = false;              // ロングタッチイベントステータス
    public int TouchStatus = -1;                        // タッチイベントステータス
    final static public int TouchStatus_UP = 1;         // タッチイベント上
    final static public int TouchStatus_DOWN = 2;       // タッチイベント下
    final static public int TouchStatus_LEFT = 3;       // タッチイベント左
    final static public int TouchStatus_RIGHT = 4;      // タッチイベント右
    final static public int TouchStatus_TAP = 5;        // タッチイベントタップ

    /** ステータス定義情報 */
    final static private int STATUS_WAIT = -1;          //何もしない。
    final static private int STATUS_START = 0;          //起動～タイトル表示まで
    final static private int STATUS_TITLE = 1;          //タイトル表示
    final static private int STATUS_ROLL = 2;           //ゲームロール
    final static private int STATUS_MAP = 3;            //フィールドマップ
    final static private int STATUS_MENU = 4;           //メニュー
```

```
    final static private int STATUS_BATTLE = 5;          //戦闘シーン
    final static private int STATUS_ENDING = 8;          //エンディング
    final static private int STATUS_GAMEOVER = 9;        //ゲームオーバー
    final static private int STATUS_ERROR = 9999;        //エラー

    /** ステータス情報 */
    private int status = STATUS_START;          //ゲームステータス
    @SuppressWarnings("unused")
        private int exStatus = -1;              //前のステータス（今回は利用せず）
    /** スリープ変数 */
        private long sleep = 100;               //処理を停止させるミリ秒

        /** プレイヤー情報 */
        private int level = 1;                  //レベル
        private int maxHp = 0;                  //最大ヒットポイント
        private int hp = 0;                     //ヒットポイント
        private int str = 0;                    //攻撃力
        private int def = 0;                    //防御力
        private int spd = 0;                    //スピード
        private int exp = 0;                    //現在の経験地
        private int nextExp = 0;                //次の経験値
        private int maxSp = 0;                  //最大スキルポイント
        private int sp = 0;                     //スキルポイント
        private int maxspup;                    //LVアップ時の成長ポイント

        private int FLAG_SENDAI_WEAPON = 0;     //フラグ：武器
        private int FLAG_SENDAI_ARMOR = 0;      //フラグ：鎧
        private int FLAG_SENDAI_SHIELD = 0;     //フラグ：盾
        private int FLAG_ITEM1 = 0;             //フラグ：アイテム１
        private int FLAG_ITEM2 = 0;             //フラグ：アイテム２
        private int FLAG_ITEM3 = 0;             //フラグ：アイテム３
        private int FLAG_BOSS1 = 0;             //フラグ：ボスキャラ１

        /** タイトル画面用パラメータ */
        private int titleSelectedIndex = 0;     //タイトルメニューの選択値
        private int playerDrawIndex = 0;        //プレイヤーのアニメーション値（マップでも利用）
    private Bitmap titleImage;                  //タイトル画像
    private Bitmap[][] playerImage = new Bitmap[4][2];   //プレイヤー画像配列（マップでも利用）
    private boolean startFlag = true;           //初回プレイ時のフラグ

        /** ロール画面用パラメータ */
        private String rollMsgTxt[] = new String[0];    //ロール用メッセージ
        private int rollMarginX = 10;                   //表示ｘ座標
        private int rollMsgPosY = 0;                    //表示ｙ座標

        /** フィールドマップ画面用パラメータ */
        final static private String MAP_DEF = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                                                        //地図定義
    final static private int PLAYER_DIRECTION_DOWN = 0;         //プレイヤーの向き（下）
    final static private int PLAYER_DIRECTION_LEFT = 1;         //プレイヤーの向き（左）
    final static private int PLAYER_DIRECTION_RIGHT = 2;        //プレイヤーの向き（右）
    final static private int PLAYER_DIRECTION_UP = 3;           //プレイヤーの向き（上）
    private Bitmap[] fldImage = new Bitmap[36];          //フィールド画像配列 28+NPC8=36
    private Bitmap mapImage = null;                     //地図描写用全体画像
        private int[][] filedMapInt = new int[0][0];    //地図情報配列
        private String filedMapCannotMoveTxt = "";      //移動不可地図情報テキスト
        @SuppressWarnings("unchecked")
        private HashMap mapActionMap = new HashMap();
                                            //地図上でのアクションを管理するMap変数
        private int cannotMoveImageInt = -1;  //地図の範囲外の場合に表示したい画像情報
```

```
	private int playerMapPosX = 0;                  //プレイヤーの地図上のｘ座標
	private int playerMapPosY = 0;                  //プレイヤーの地図上のｙ座標
	private int playerDrawPosX = 0;                 //プレイヤーの表示位置。必ず中央です。
	private int playerDrawPosY = 0;                 //プレイヤーの表示位置。必ず中央です。
	private int mapMarginX = 0;                     //画面上に描写する地図のマージンｘ座標
	private int mapMarginY = 0;                     //画面上に描写する地図のマージンｙ座標
	private int fieldMapImageWidth = 24;            //フィールド画像の幅
	private int fieldMapWidth = 80;//40;            //定義されている地図の最大横幅
	private int fieldMapHeight = 80;//40;           //定義されている地図の最大縦幅
	private int playerDirection = PLAYER_DIRECTION_DOWN;     //        プレイヤーの向き
	private int statusDrawPosY = 0;                 //ステータス表示をするｙ座標
	private int helpMsgTxtIndex = 0;                //補足テキストの、描写位置
	private String helpMsgTxt = "";                 //補足テキスト
	static int map = R.string.map;                  //初期マップ番号
	//private int mapNo = 0;     //現在マップ番号（現在マップ番号を保存するのであれば使用）
	private static int encount = 40;                //敵モンスター出現確率
	private boolean encFlag = true;                 //モンスター発生フラグ

	/** 戦闘シーン画面用パラメータ */
	public Enemy[] enemy = new Enemy[0];  //敵のクラス
	public int battleMarginX = 0;         //戦闘シーンのメッセージ枠描写のマージンｘ座標
	public int battleMarginY = 0;         //戦闘シーンのメッセージ枠描写のマージンｙ座標
	private String[] battleMsgTxts = new String[4];          //戦闘メッセージ
	private String[] battleMsgMasterTxts = new String[0];
                                          //戦闘メッセージマスタテキスト
	private int battleMsgTxtIndex = 0;            //戦闘メッセージマスタのインデックス値
	private int battleMenuSelectedIndex = 0;      //戦闘メニューの選択値
	private int battleMenuSelectedMinusX = 0;
                                          //戦闘メニュー選択表示のための、マージン値
	private int battleMenuSelectedPlusX = 0;
                                          //戦闘メニュー選択表示のための、マージン値
	private int battleEnemySelectedIndex = -1;
                                          //戦闘シーンで、敵の選択のインデックス値
	private String[] battleActions = new String[0];
                                          //戦闘シーンにおける、アクションテキスト配列
	private Bitmap[] batteImages;         //タイトル画像
	private Bitmap batteImage;
	public boolean selectFlag = false;            //全体攻撃用選択フラグ
	public boolean paintFlag = false;             //全体攻撃用選択フラグ
	public int reqSpec = 3;                       //特殊攻撃に必要なSP
	public int reqCure = 2;                       //回復に必要なSP

	/** エラー画面用パラメータ */
	private String errorTxt = "";                 //エラーメッセージ

	/** 共通変数 */
	private int drawFontSize = 14;                //描写用のフォントサイズ
	private int count = 0;                        //処理カウント変数
	@SuppressWarnings("unused")
	private long keyPressedTime = 0;
                    //キーイベント操作用変数。（押下時の時刻を保持させておく）

	/** BGM用変数 */
public MediaPlayer mp_Bgm = null;                 // BGM用インスタンス
public MediaPlayer mp_Se = null;                  // SE用インスタンス

	private MiyagiQuest activity = null;

	/**
```

```
     * Miyagi Quest の表示処理用クラス コンストラクタ
     * 初期化などの処理を行う。
          */
         @SuppressWarnings("unchecked")
         public MiyagiQuestView(Context context, MiyagiQuest act) {

                   super(context);
                   activity = act;

                   setFocusable(true);

                   //        タイトル画像読み込み
                   Resources r = getResources();
                   titleImage = BitmapFactory.decodeResource(r, R.drawable.titlemiyagi);

                   //TODO 戦闘時背景画像読み込み
                   batteImages = new Bitmap[4];
                   batteImages[0] = BitmapFactory.decodeResource(r, R.drawable.battle_01);
                   batteImages[1] = BitmapFactory.decodeResource(r, R.drawable.battle_02);
                   batteImages[2] = BitmapFactory.decodeResource(r, R.drawable.battle_03);
                   batteImages[3] = BitmapFactory.decodeResource(r, R.drawable.battle_04);

                   //マップ画像の読み込み
                   //フィールド名と、実際値が異なるので、reflectionを利用。
                   //reflection とは、フィールド名などから、その値を取得するための機能。
                   try {
                            //        どのクラスから取得するかを指定
                            Class cls = Class.forName("com.cgene.game.R$drawable");
                            for (int i = 1, idx = 0; i <= 7; i ++) {
                                     for (int j = 1; j <= 4; j ++) {
                                              String fldName = "field_" + i + j;
                                              fldImage[idx] = BitmapFactory.decodeResource(r,
getResouceInt(cls, fldName));
                                              idx ++;
                                     }
                            }

                            //TODO NPCキャラ（マップチップと同じ扱い）読み込み
                            fldImage[28] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_76" ));
                            fldImage[29] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_77" ));
                            fldImage[30] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_78" ));
                            fldImage[31] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_79" ));
                            fldImage[32] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_80" ));
                            fldImage[33] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_81" ));
                            fldImage[34] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_82" ));
                            fldImage[35] = BitmapFactory.decodeResource(r,
getResouceInt(cls,"field_83" ));

                            //上記の処理は、fldImage[idx] = BitmapFactory.decodeResource(r,
R.drawable.filed_11); としても同じだが、
                            //R.drawable.filed_11 を、28個も記述するのが、面倒なので、
リフレクションで簡易化している。
```

```
                        //プレイヤー画像の読み込み。フィールド名と、実際値が異なるので、
reflectionを利用。
                        char c[] = {'b', 'l', 'r', 'u'};
                        for (int i = 0; i < c.length; i ++) {
                                for (int j = 0; j < 2; j ++) {
                                        String fldName = "hero_" + c[i] + (j+1);
                                        playerImage[i][j] =
 BitmapFactory.decodeResource(r, getResouceInt(cls, fldName));
                                }
                        }

                } catch (Exception e) {
                        android.util.Log.e(DebugLog.TAG, e.toString());
                }
                //初期化が終わったので、ステータスを、タイトルに設定する。
                setStatus(STATUS_TITLE, true);

                TouchStatus = 0;
        }

        /**
     * リフレクションのための、サポートメソッド。指定されたフィールド名のint値が返る。
     * @param       cls                     対象クラス
     * @param       fieldName フィールド名
     * @return      設定されている、int値を返す。
         */
        @SuppressWarnings("unchecked")
        public int getResouceInt(Class cls, String fieldName) throws Exception {
                if (cls == null) {
                        cls = Class.forName("com.cgene.game.R$drawable");
                }
                Field fld = cls.getField(fieldName);
                return fld.getInt(cls.newInstance());
        }

        /**
     * 処理停止のミリ秒を返す。ステータスごとに描写の速度が異なるので、
     * ※public 変数にしてしまえば、必要なし。直接参照は可能。
     * @return      設定されている、停止ミリ秒を返す。
         */
        public long getSleep() {
                return sleep;
        }

        /**
     * ステータスをセットします。
     * @param       sts             設定するステータス値
     * @param       init    初期化するかのフラグ
         */
        private void setStatus(int sts, boolean init) {
                exStatus = status;
                status = sts;
                if (init) {
                        try {
                                init(sts);
                        } catch (Exception e) {
                                errorTxt = e.toString();
                                setStatus(STATUS_ERROR, false);
                        }
                }
```

```
	}

	/**
 * 処理を停止させる。
 * @param	time	停止させたいミリ秒
	 */
	public void sleep(long time) {
		try {
			Thread.sleep(time);
		} catch (Exception e) {
		}
	}

	/**
 * 初期化の処理
 * @param	sts		初期化を行うステータス
	 */
	@SuppressWarnings("unchecked")
	public void init(int sts) throws Exception {
		DebugLog.printin("init(" + sts + ") start");
		count = 0;
		Resources r = getResources();
	switch (sts) {
	case STATUS_TITLE	:	//タイトル描写のためのデータ初期化
	  sleep = 100;
	  drawFontSize = 20;
	  titleSelectedIndex = 0;
	  playerDrawIndex = 0;
	  break;
	case STATUS_ROLL	:	//ゲームロール描写のためのデータ初期化
	  rollMarginX = 20;
	  sleep = 50;
	  //	ゲームロールのメッセージは、roll_msg にて、リソース管理されています。
	  rollMsgTxt = MyUtil.split(r.getString(R.string.roll_msg), "|");
	  drawFontSize = 24;
	  rollMsgPosY = displayHeight;
	  break;
	case STATUS_MAP		:	//マップ描写のためのデータ初期化
	  //地図情報をリソースから、取得します。
		//String mapTxt = r.getString(R.string.map);
		String mapTxt = r.getString(map);	//マップ遷移対応！
	  //地図情報は、スペース区切りなので、各行ごとに分割させます。
		String mapTxts[] = MyUtil.split(mapTxt, " ");
		//フィールドマップの定義されている地図の最大幅を設定します。
		fieldMapWidth = mapTxts[0].length();
		fieldMapHeight = mapTxts.length;
			//地図情報の最後に、改行やスペースが入っていると変になるので注意。
		//地図情報から、実際の値（int値）を設定していきます。
		filedMapInt = new int[fieldMapHeight][fieldMapWidth];
		for (int y = 0; y < fieldMapHeight; y ++) {
			for (int x = 0; x < fieldMapWidth; x ++) {
				int idx = MAP_DEF.indexOf(mapTxts[y].charAt(x));
				if (idx == -1) idx = 0;
				else if (idx >= fldImage.length) idx = 0;
				filedMapInt[y][x] = idx;
			}
		}
		//	地図上のアクション（SAVE, CURE）などのアクションを設定していきます。
		if (true) {
			Class cls = Class.forName("com.cgene.game.R$string");
```

```
                        for (int i = 0; i < MAP_DEF.length(); i ++) {
                                try {
                                        int k = getResouceInt(cls,
"map_action_" + MAP_DEF.charAt(i));
                                        String act = r.getString(k);
                                        if (act != null) {
                                                mapActionMap.put(String.valueOf(i),
 act);
                                        }
                                } catch (Exception e) {
                                }
                        }
                }
                //      地図情報の中で、移動できないフィールドテキストを設定する。
                filedMapCannotMoveTxt = r.getString(R.string.map_cannot_move);
                cannotMoveImageInt = MAP_DEF.indexOf(filedMapCannotMoveTxt.charAt(0));

                //      マップ表示マージンの初期化
                mapMarginX = (displayWidth - fldImage[0].getWidth()*13)/2;
                mapMarginY = mapMarginX;

                //マップ表示時の効果音
                        try {
                                MyUtil.PlaySound_SE(activity, MyUtil.SE_IN_OUT);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }

                //      フィールド一つの画像幅を設定
                fieldMapImageWidth = fldImage[0].getWidth();

                //      プレイヤーの表示位置を設定。常に、マップ中央です。
                playerDrawPosX = mapMarginX + fieldMapImageWidth*6;
                playerDrawPosY = mapMarginY + fieldMapImageWidth*6;

                //      プレイヤーの初期化
                playerMapPosX = mapTxts[0].length()/2;
                playerMapPosY = mapTxts.length/2;
                playerDirection = PLAYER_DIRECTION_DOWN;
                playerDrawIndex = 0;

                //      プレイヤーステータス情報の初期化
                if(startFlag){
                        drawFontSize = 20;
                        level   = 1;
                        maxHp   = 35;                   //最大ヒットポイント
                        hp      = 35;                   //ヒットポイント
                        str     = 10;                   //攻撃力
                        def     = 8;                    //防御力
                        spd     = 6;                    //スピード
                        exp     = 0;                    //経験値
                        nextExp = 30;
                        maxSp   = 10;                   //最大スキルポイント
                        sp      = 10;                   //スキルポイント

                        //最初に表示するヘルプテキストをリソースから設定します。
                        helpMsgTxt = r.getString(R.string.map_help_msg);
                        helpMsgTxtIndex = -5;

                        //初回プレイフラグの変更
```

```
                        startFlag = false;
                }

                //      プレイヤーステータス（ＨＰなど）を表示するｙ座標を設定します。
                statusDrawPosY = mapMarginY*2 + 4 + fieldMapImageWidth*13;


                //      地図画像の初期化
                mapImage = Bitmap.createBitmap(fieldMapImageWidth*13,
fieldMapImageWidth*13, Bitmap.Config.ARGB_8888);
                Canvas canvas = new Canvas(mapImage);
                initMapImage(canvas);

        sleep = 100;

        break;
       case STATUS_MENU     :          //メニュー描写のためのデータ初期化（今回は、利用なし）
        break;
       case STATUS_BATTLE   :          //戦闘シーン描写のためのデータ初期化

        //      敵の数をランダムで設定します。
        int max = 2+level;
        if (max > 6) max = 6;
        int k = MyUtil.nextInt(max)+1;        //1 - 3 まで

                if(FLAG_BOSS1 == 1) k = 1;            //ボスキャラ用

        enemy = new Enemy[k];

        //戦闘メッセージを初期化します。
                battleMsgTxts = new String[4];
                battleMsgMasterTxts = new String[k];

                //敵の情報とメッセージを初期化します。
        for (int i = 0; i < enemy.length; i ++) {

                //レベルに応じた敵の選出
                //TODO モンスターの出現率などのバランスを調節！
                k = MyUtil.nextInt(level);  //レベルで乱数調節！
                if(FLAG_BOSS1 == 1) k = 99;           //ボスキャラ用
                String txt ="";
                switch(k){
                        case 0: case 1:
                                txt = r.getString(R.string.enemy_1);break;
                        case 2: case 3:
                                txt = r.getString(R.string.enemy_2);break;
                        case 4: case 5:
                                txt = r.getString(R.string.enemy_3);break;
                        case 6: case 7:
                                txt = r.getString(R.string.enemy_4);break;
                        case 8: case 9:
                                txt = r.getString(R.string.enemy_5);break;
                        case 10: case 11:
                                txt = r.getString(R.string.enemy_6);break;
                        case 12: case 13:
                                txt = r.getString(R.string.enemy_7);break;
                        case 14: case 15:
                                txt = r.getString(R.string.enemy_8);break;
                        case 99:
                                txt = r.getString(R.string.enemy_99);break;
```

```
                        default:
                                txt = r.getString(R.string.enemy_8);break;
                }

                enemy[i] = new Enemy(this, txt, i, enemy.length);
                battleMsgMasterTxts[i] = enemy[i].name + "が現れた！";
        }
        battleMarginX = mapMarginX;
        battleMarginY = -8;
                drawFontSize = 20;

                //      描写のための変数処理のため、paintを初期化します。
                Paint paint = new Paint();
          paint.setStyle(Paint.Style.FILL);
                paint.setStrokeWidth(1);
                paint.setTextSize(drawFontSize);

                //      戦闘メニューで必要な数値の初期化
                battleMenuSelectedIndex = 0;
                        battleMenuSelectedMinusX = 5 + (int)paint.measureText("≫");
                        battleMenuSelectedPlusX = 5 + (int)paint.measureText("戦　う");

                        battleEnemySelectedIndex = -1;
                        battleActions = new String[0];

                battleMsgTxtIndex = -1;
         break;

        case STATUS_ENDING   :          //ゲームオーバー描写のためのデータ初期化
         sleep = 1000;
         drawFontSize = 40;
         rollMarginX = 20;
         //ゲームロールのメッセージは、roll_msg にて、リソース管理されています。
         rollMsgTxt = MyUtil.split(r.getString(R.string.roll_msg2), "|");
         drawFontSize = 24;
         rollMsgPosY = displayHeight;
         break;

        case STATUS_GAMEOVER:           //ゲームオーバー描写のためのデータ初期化
         sleep = 1000;
         drawFontSize = 40;
         break;
        case STATUS_ERROR    :                  //エラー描写のためのデータ初期化
                drawFontSize = 20;
         break;
        }
                DebugLog.println("init(" + sts + ") done");
         }

         /**
     * マップの指定範囲内かどうかを判定する。
     * @param       x           地図のｘ座標
     * @param       y           地図のｙ座標
     * @return                  範囲内かどうかのフラグ
          */
         private boolean isDrawableMap(int x, int y) {
                if ( x < 0 || y < 0 || x >= fieldMapWidth || y >= fieldMapHeight) {
                        return false;
                } else {
                        int k = filedMapInt[y][x];
```

```
                        if (k < 0 || k >= fldImage.length) {
                                return false;
                        } else {
                                return true;
                        }
                }
        }

        /**
     * タイトルを描写します。
         * @param canvas    描写キャンバスクラス
         * @param paint     描写ペイントクラス
         */
        private void title(Canvas canvas, Paint paint) {
                //      タッチイベント処理
                switch (TouchStatus) {
                case TouchStatus_UP:
                        titleSelectedIndex--;
                        if(Flag_LongTouch != true) TouchStatus = -1;
                        break;
                case TouchStatus_DOWN:
                        titleSelectedIndex++;
                        if(Flag_LongTouch != true) TouchStatus = -1;
                        break;
                case TouchStatus_TAP:
                        //      選択キーでの処理
                        sleep(500);
                        TouchStatus = -1;
                        if (titleSelectedIndex == 1) {          //Continueが選択された場合
                                //      保存データを読み込み、[,]で分割。
                                String[] data = MyUtil.split(load(), ",");
                                //TODO 保存項目数により異なる！！
                                if (data.length < 19) {
                                        //データがないか、不正であれば、New Gameへ
                                        setStatus(STATUS_ROLL, true);
                                } else {
                                //データがあれば、ステータスをMAPに設定し、初期化）
                                        setStatus(STATUS_MAP, true);
                                        try {
                                                //保存データから、各値を初期化
                                                int k = 0;
                                                level = Integer.parseInt(data[k++]);
                                                maxHp = Integer.parseInt(data[k++]);
                                                hp = Integer.parseInt(data[k++]);
                                                maxSp = Integer.parseInt(data[k++]);
                                                sp = Integer.parseInt(data[k++]);
                                                str = Integer.parseInt(data[k++]);
                                                def = Integer.parseInt(data[k++]);
                                                spd = Integer.parseInt(data[k++]);
                                                exp = Integer.parseInt(data[k++]);
                                                nextExp = Integer.parseInt(data[k++]);
                                                map = Integer.parseInt(data[k++]);
                                                playerMapPosX =
Integer.parseInt(data[k++]);
                                                playerMapPosY =
Integer.parseInt(data[k++]);

                                                //TODO 各種イベントフラグ
                                                FLAG_SENDAI_WEAPON =
Integer.parseInt(data[k++]);
```

```
                                                FLAG_SENDAI_ARMOR  =
 Integer.parseInt(data[k++]);
                                                FLAG_SENDAI_SHIELD =
 Integer.parseInt(data[k++]);
                                                FLAG_ITEM1  =
 Integer.parseInt(data[k++]);
                                                FLAG_ITEM2  =
 Integer.parseInt(data[k++]);
                                                FLAG_ITEM3  =
 Integer.parseInt(data[k++]);

                                                //初回プレイ時フラグ
                                                startFlag = false;

                                        initMapImage(new Canvas(mapImage));
                                        } catch (Exception e) {
                                                //処理中にエラーが発生したら、
New Gameへ
                                                setStatus(STATUS_ROLL, true);
                                        }
                                }
                        } else {  //       New Gameが選択されました。
                                setStatus(STATUS_ROLL, true);
                        }
                        //       処理は、ここで終了。（キーイベント処理で終了です。）
                        return;
                }
                //       タイトル選択インデックスの処理
                if (titleSelectedIndex < 0) titleSelectedIndex = 1;
                else if (titleSelectedIndex > 2) titleSelectedIndex = 0;

                //       描写処理
                int x = (displayWidth-titleImage.getWidth())/2;
                int y = 10;
                canvas.drawBitmap(titleImage, x, y, paint);

                paint.setTextSize(42);
                y += 58 + titleImage.getHeight();

        paint.setColor(Color.WHITE);
        canvas.drawText("New Game", 80, y, paint);
        canvas.drawText("Continue", 80, y + 60, paint);
        //         選択されている側にプレイヤー画像を表示されます。
        if (titleSelectedIndex == 0) {
                canvas.drawBitmap(playerImage[2][playerDrawIndex], 20, y-36, paint);
        } else {
                canvas.drawBitmap(playerImage[2][playerDrawIndex], 20, y+60-36, paint);
        }
        //         プレイヤーのアニメーション変更。countで、早すぎないように変更させる。
        if (count%3 == 0) playerDrawIndex = (playerDrawIndex+1)%2;
         }

          /**
    * ゲームロールを描写します。
           * @param canvas    描写キャンバスクラス
           * @param paint     描写ペイントクラス
           */
         private void roll(Canvas canvas, Paint paint) {
                //       タッチイベント処理
                switch (TouchStatus) {
```

```
                    case TouchStatus_TAP:
                            if (rollMsgPosY < 50) {
                    //表示位置が、50pixより小さければ、選択キーで、次のステータス（マップ）へ
                                    sleep(500);
                                    TouchStatus = -1;
                                    setStatus(STATUS_MAP, true);
                                    return;
                            }
                    }
                    //描写位置の下限
                    if (rollMsgPosY < 50) rollMsgPosY = 50;

                    //ロールメッセージを表示させます。
                    paint.setTextSize(drawFontSize);
                    for (int i = 0; i < rollMsgTxt.length; i ++) {
                            paint.setColor(Color.WHITE);
                            canvas.drawText(rollMsgTxt[i],     rollMarginX,     rollMsgPosY     +
i*(drawFontSize+5), paint);
                    }
                    //ロールメッセージが、下限になった場合のみ、
「Hit Key」を表示させて、次を促します。
                    if (rollMsgPosY == 50) {
                            paint.setColor(Color.WHITE);
                            canvas.drawText("Please press Hit Key", rollMarginX,
displayHeight-drawFontSize-20, paint);
                            paint.setColor(Color.BLUE);
                            canvas.drawText("Please press Hit Key", rollMarginX-1,
displayHeight-drawFontSize-20-1, paint);
                    }
                    rollMsgPosY -= 5;
            }

            /**
        * 地図画像を初期化します。
             * @param canvas    描写キャンバスクラス
             */
            public void initMapImage(Canvas canvas) {
                    //地図の描写
                    for (int y = playerMapPosY - 6, j = 0; y < playerMapPosY + 7; y ++, j ++) {
                            for (int x = playerMapPosX - 6, i = 0; x < playerMapPosX + 7; x ++,
i ++) {
                                    if (isDrawableMap(x, y)) {
                                            canvas.drawBitmap(fldImage[filedMapInt[y][x]],
i*fieldMapImageWidth, j*fieldMapImageWidth, new Paint());
                                    } else {
                                            canvas.drawBitmap(fldImage[cannotMoveImageInt],
i*fieldMapImageWidth, j*fieldMapImageWidth, new Paint());
                                    }
                            }
                    }
            }

            /**
        * 地図上で、指定方向に動いたかを判定します。
             * @param direction 移動方向
             */
            public boolean mapMoved(int direction) {
                    boolean moved = true;
                    //        現在値をバックアップ
                    int prevx = playerMapPosX;
```

```
                int prevy = playerMapPosY;
                playerDirection = direction;
                switch (direction) {
                case PLAYER_DIRECTION_UP :              //上に移動したい場合
                        playerMapPosY --;
                        if (playerMapPosY < 0) {    //できませんでした。
                                playerMapPosY = 0;
                                moved = false;
                        }
                        break;
                case PLAYER_DIRECTION_DOWN :            //下に移動したい場合
                        playerMapPosY ++;
                        if (playerMapPosY >= fieldMapHeight) {//できませんでした。
                                playerMapPosY = fieldMapHeight-1;
                                moved = false;
                        }
                        break;
                case PLAYER_DIRECTION_LEFT :            //左に移動したい場合
                        playerMapPosX --;
                        if (playerMapPosX < 0) {    //できませんでした。
                                playerMapPosX = 0;
                                moved = false;
                        }
                        break;
                case PLAYER_DIRECTION_RIGHT :           //右に移動したい場合
                        playerMapPosX ++;
                        if (playerMapPosX >= fieldMapWidth) { //できませんでした。
                                playerMapPosX = fieldMapWidth-1;
                                moved = false;
                        }
                        break;
                }
                if (moved) {
                                //地図上では移動できた。ので、次に、
フィールドが移動可能なフィールドかどうかをチェックする。
                        int idx = filedMapInt[playerMapPosY][playerMapPosX];
                        char c = MAP_DEF.charAt(idx);

                //TODO 地形に応じた戦闘背景
                if      (c == '5') batteImage = batteImages[1]; //山
                else if(c == 'E') batteImage = batteImages[2]; //雪
                else if(c == '8' || c == 'K' || c == 'S')
                          batteImage = batteImages[3];        //夕日
                else      batteImage = batteImages[0];        //草原

                        if (filedMapCannotMoveTxt.indexOf(c) != -1) {
                                //移動できないフィールドなので、元に戻す。
                                playerMapPosX = prevx;
                                playerMapPosY = prevy;
                                moved = false;
                        } else {
                                //移動できるフィールドだったので、移動後の地図表示を行う
                                //地図の再描写
                                Bitmap bitmap = Bitmap.createBitmap(mapImage);
                        Canvas canvas = new Canvas(mapImage);
                        Paint paint = new Paint();
                  paint.setAntiAlias(true);
                  paint.setStyle(Paint.Style.FILL);

                  int x = 0;
```

```
                    int y = 0;
                    int i, j;
                    switch(direction) {
                    case PLAYER_DIRECTION_UP:                    //上を描写
                        canvas.drawBitmap(bitmap, 0, fieldMapImageWidth, paint);
                        y = playerMapPosY - 6;
                        for (x = playerMapPosX - 6, i = 0; x < playerMapPosX + 7; x ++, i ++)
{
                            if (isDrawableMap(x, y)) {
                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],
i*fieldMapImageWidth, 0, paint);
                            } else {
                                canvas.drawBitmap(fldImage[cannotMoveImageInt],
i*fieldMapImageWidth, 0, paint);
                            }
                                }
                        break;
                    case PLAYER_DIRECTION_DOWN:         //下を描写
                        canvas.drawBitmap(bitmap, 0, -fieldMapImageWidth, paint);
                        y = playerMapPosY + 7 - 1;
                        for (x = playerMapPosX - 6, i = 0; x < playerMapPosX + 7; x ++, i ++)
{
                            if (isDrawableMap(x, y)) {
                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],
i*fieldMapImageWidth, 12*fieldMapImageWidth, paint);
                            } else {
                                canvas.drawBitmap(fldImage[cannotMoveImageInt],
i*fieldMapImageWidth, 12*fieldMapImageWidth, paint);
                            }
                        }
                        break;
                    case PLAYER_DIRECTION_LEFT:                          //左を描写
                        canvas.drawBitmap(bitmap, fieldMapImageWidth, 0, paint);
                        x = playerMapPosX - 6;
                        for (y = playerMapPosY - 6, j = 0; y < playerMapPosY + 7; y ++, j ++)
{
                            if (isDrawableMap(x, y)) {
                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],
0, j*fieldMapImageWidth, paint);
                            } else {
                                canvas.drawBitmap(fldImage[cannotMoveImageInt],
0, j*fieldMapImageWidth, paint);
                            }
                        }
                        break;
                    case PLAYER_DIRECTION_RIGHT:                 //右を描写
                        canvas.drawBitmap(bitmap, -fieldMapImageWidth, 0, paint);
                        x = playerMapPosX + 7 - 1;
                        for (y = playerMapPosY - 6, j = 0; y < playerMapPosY + 7; y ++, j ++)
{
                            if (isDrawableMap(x, y)) {
                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],
12*fieldMapImageWidth, j*fieldMapImageWidth, paint);
                            } else {
                                canvas.drawBitmap(fldImage[cannotMoveImageInt],
12*fieldMapImageWidth, j*fieldMapImageWidth, paint);
                            }
                        }
                        break;
                }
```

```
		}

				DebugLog.println("X = " + playerMapPosX + " Y = " + playerMapPosY);

				//マップ遷移チェック
				try{
					mapChange();
				}catch(Exception e){
					//ignore
				}

				//マップオブジェクト上でのヘルプメッセージ
				int tileNo = filedMapInt[playerMapPosY][playerMapPosX];
				String act = (String)mapActionMap.get(String.valueOf(tileNo));
				if ("CURE".equals(act)) {
				helpMsgTxt = "画面タップ = HP・SP回復";
					helpMsgTxtIndex = -2;
				} else if ("SAVE".equals(act)) {
					helpMsgTxt = "画面タップ = セーブ";
					helpMsgTxtIndex = -2;
				} else if ("WEAPON".equals(act) || "ARMOR".equals(act) ||
"SHIELD".equals(act)) {
					helpMsgTxt = "画面タップ = ステータスアップ";
					helpMsgTxtIndex = -2;
				} else if ("NPC1".equals(act) ||
							"NPC2".equals(act) ||
							"NPC3".equals(act) ||
							"NPC4".equals(act) ||
							"NPC5".equals(act) ||
							"NPC6".equals(act) ||
							"NPC7".equals(act) ||
							"NPC8".equals(act)){
					helpMsgTxt = "画面タップ = 会話";
					helpMsgTxtIndex = -2;
				}
			}

			//マップ再作成
			this.initMapImage(new Canvas(mapImage));

			return moved;
		}

		/**
	 * マップフィールドの変更をします。
		 */
		private void mapChange() throws Exception{
			Resources r = getResources();

			//TODO マップ間移動処理
			switch (map) {
				//フィールド****************************************
				case R.string.map:
					//フィールド内にある入口は全てここで制御！
					if ((playerMapPosX == 46 && playerMapPosY == 48) ||
						(playerMapPosX == 46 && playerMapPosY == 49) ||
						(playerMapPosX == 46 && playerMapPosY == 50) ||
						(playerMapPosX == 47 && playerMapPosY == 48) ||
						(playerMapPosX == 47 && playerMapPosY == 49) ||
						(playerMapPosX == 47 && playerMapPosY == 50))
```

```
                {
                        //センダイへ
                        map = R.string.sendai;
                        //カギ所持ならばマップ変更
                        if(FLAG_ITEM1 == 1)map = R.string.sendai2;
                        encFlag = false;    //モンスター発生せず
                        this.init(STATUS_MAP);
                        playerMapPosX = 27;playerMapPosY = 29;
                        helpMsgTxt = "センダイの町です。";
                        helpMsgTxtIndex = -2;

                }else if((playerMapPosX == 47 && playerMapPosY == 20) ||
                         (playerMapPosX == 47 && playerMapPosY == 21)){
                        //オオサキへ
                        map = R.string.oosaki;
                        //カギ所持ならばマップ変更
                        if(FLAG_ITEM2 == 1)map = R.string.oosaki2;
                        encFlag = false;    //モンスター発生せず
                        this.init(STATUS_MAP);
                        playerMapPosX = 28;playerMapPosY = 7;
                        helpMsgTxt = "オオサキの町です";
                        helpMsgTxtIndex = -2;

                }else if(playerMapPosX == 12 && playerMapPosY == 11){
                        //ナナシへ
                        map = R.string.nanasi;
                        encFlag = false;    //モンスター発生せず
                        this.init(STATUS_MAP);
                        playerMapPosX = 16;playerMapPosY = 14;
                        helpMsgTxt = "？？？";
                        helpMsgTxtIndex = -2;

                }else if(playerMapPosX == 17 && playerMapPosY == 55){
                        //レイクへ
                        map = R.string.lake;
                        //オーブ所持ならばマップ変更
                        if(FLAG_ITEM3 == 1)map = R.string.lake2;
                        encFlag = true;     //モンスター発生
                        this.init(STATUS_MAP);
                        playerMapPosX = 7;playerMapPosY = 5;
                        helpMsgTxt = "レイク・ダンジョン";
                        helpMsgTxtIndex = -2;
                }
                break;    //CAUTION!

        //センダイ****************************************
        case R.string.sendai:
        case R.string.sendai2:
                if ((playerMapPosX <=  6) ||
                        (playerMapPosX >= 43) ||
                        (playerMapPosY <=  6) ||
                        (playerMapPosY >= 30))
                {
                        encFlag = true;     //モンスター発生
                        map = R.string.map;
                        this.init(STATUS_MAP);
                        playerMapPosX = 47;playerMapPosY = 51;
                        helpMsgTxt = "フィールドに出ます。";
                        helpMsgTxtIndex = -2;
                }
```

```
                                break;    //CAUTION!

                        //オオサキ*****************************************
                        case R.string.oosaki:
                        case R.string.oosaki2:
                                if ((playerMapPosX <=  6) ||
                                        (playerMapPosX >= 43) ||
                                        (playerMapPosY <=  6) ||
                                        (playerMapPosY >= 30))
                                {
                                        encFlag = true;    //モンスター発生
                                        map = R.string.map;
                                        this.init(STATUS_MAP);
                                        playerMapPosX = 47;playerMapPosY = 22;
                                        helpMsgTxt = "フィールドに出ます。";
                                        helpMsgTxtIndex = -2;
                                }
                                break;    //CAUTION!

                        //ナナシ*******************************************
                        case R.string.nanasi:
                                if ((playerMapPosX <=  6) ||
                                        (playerMapPosX >= 24) ||
                                        (playerMapPosY <=  6) ||
                                        (playerMapPosY >= 17))
                                {
                                        encFlag = true;    //モンスター発生
                                        map = R.string.map;
                                        this.init(STATUS_MAP);
                                        playerMapPosX = 12;playerMapPosY = 12;
                                        helpMsgTxt = "フィールドに出ます。";
                                        helpMsgTxtIndex = -2;
                                }
                                break;    //CAUTION!

                        //レイク*******************************************
                        case R.string.lake:
                        case R.string.lake2:
                                if (playerMapPosX == 5 && playerMapPosY == 5)
                                {
                                        encFlag = true;    //モンスター発生
                                        map = R.string.map;
                                        this.init(STATUS_MAP);
                                        playerMapPosX = 17;playerMapPosY = 54;
                                        helpMsgTxt = "フィールドに出ます。";
                                        helpMsgTxtIndex = -2;
                                }
                                break;    //CAUTION!
                }
        }

        /**
     * マップフィールドを描写します。
         * @param canvas    描写キャンバスクラス
         * @param paint     描写ペイントクラス
         */
        private void map(Canvas canvas, Paint paint) {
                boolean moved = false;

                //      タッチイベント処理
```

```
                    switch (TouchStatus) {
                    case TouchStatus_UP:
                            moved= mapMoved(PLAYER_DIRECTION_UP);
                            if(Flag_LongTouch != true) TouchStatus = -1;
                            break;
                    case TouchStatus_DOWN:
                            moved= mapMoved(PLAYER_DIRECTION_DOWN);
                            if(Flag_LongTouch != true) TouchStatus = -1;
                            break;
                    case TouchStatus_LEFT:
                            moved= mapMoved(PLAYER_DIRECTION_LEFT);
                            if(Flag_LongTouch != true) TouchStatus = -1;
                            break;
                    case TouchStatus_RIGHT:
                            moved= mapMoved(PLAYER_DIRECTION_RIGHT);
                            if(Flag_LongTouch != true) TouchStatus = -1;
                            break;
                    case TouchStatus_TAP:
                            //選択キーの処理
                            int k = filedMapInt[playerMapPosY][playerMapPosX];
                            String act = (String)mapActionMap.get(String.valueOf(k));
                            android.util.Log.e(DebugLog.TAG, "map value = " + k + ":" + act);
                            //フィールド値が、アクションマップで定義されているかをチェック。
                            if ("CURE".equals(act)) {
                                    this.hp = this.maxHp;
                                    this.sp = this.maxSp;
                                    helpMsgTxt = "HPとSPが全回復しました！";
                                    try {
                                            MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                    } catch (Exception e) {
                                            e.printStackTrace();
                                    }
                                    helpMsgTxtIndex = -2;
                            }
                            if ("SAVE".equals(act)) {
                                    if (save()) {
                                            helpMsgTxt = "保存しました。";
                                            try {
                                            MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                            } catch (Exception e) {
                                                    e.printStackTrace();
                                            }
                                    } else {
                                            helpMsgTxt = "保存に失敗しました。";
                                    }
                                    helpMsgTxtIndex = -2;
                            }
                            TouchStatus = -1;

                            //TODO 各種タッチイベント（マップ毎に設定）
                            switch (map) {
                                    //センダイ**************************************
                                    case R.string.sendai:
                                    case R.string.sendai2:
                                            if ("NPC1".equals(act))
                                                    /*S*/ helpMsgTxt =
"「ここはセンダイの町！秘宝を求めてハンターがたくさん集まる町だ！」";
                                            if ("NPC2".equals(act))
                                                    /*T*/ helpMsgTxt =
"「レベルに応じて強いモンスターが出るぜっ！」";
```

```
                                        if ("NPC3".equals(act))
                                                /*U*/ helpMsgTxt =
 "「武器や防具はタップするだけで入手可能…」";
                                        if ("NPC4".equals(act))
                                                /*V*/ helpMsgTxt =
 "「秘宝はどこにあるのだろう？？」";
                                        if ("NPC5".equals(act))
                                                /*W*/ helpMsgTxt =
 "「レベル上げは重要だな！」";
                                        if ("NPC6".equals(act))
                                                /*X*/ helpMsgTxt =
 "「宿屋ではHP・SPがタダで回復できるぞっ！」";
                                        if ("NPC7".equals(act))
                                                /*Y*/ helpMsgTxt =
 "「北の方に町が有ると聞いたが…」";
                                        if ("NPC8".equals(act))
                                                /*Z*/ helpMsgTxt =
 "「俺が最初に秘宝を入手してやるっ！」";
                                        if ("WEAPON".equals(act)){
                                                if(FLAG_SENDAI_WEAPON == 0) {
                                                        this.str = this.str + this.level
+ 10;
                                                        FLAG_SENDAI_WEAPON = 1;
                                                        helpMsgTxt = "攻撃力がアップ！
";
                                                        try {
                MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                                        } catch (Exception e) {
                                                                e.printStackTrace();
                                                        }
                                                } else {
                                                        helpMsgTxt = "効果がないようだ
";
                                                }
                                                helpMsgTxtIndex = -2;
                                        }
                                        if ("ARMOR".equals(act)){
                                                if(FLAG_SENDAI_ARMOR == 0) {
                                                        this.def = this.def + this.level
+ 10;
                                                        FLAG_SENDAI_ARMOR = 1;
                                                        helpMsgTxt = "防御力がアップ！
";
                                                        try {
                MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                                        } catch (Exception e) {
                                                                e.printStackTrace();
                                                        }
                                                } else {
                                                        helpMsgTxt = "効果がないようだ
";
                                                }
                                                helpMsgTxtIndex = -2;
                                        }
                                        if ("SHIELD".equals(act)){
                                                if(FLAG_SENDAI_SHIELD == 0) {
                                                        this.spd = this.spd + this.level
+ 10;
```

```
                                                        FLAG_SENDAI_SHIELD = 1;
                                                        helpMsgTxt = "素早さがアップ！
";
                                                        try {

        MyUtil.playSound_SE(activity, MyUtil.SE_CURE);
                                                        } catch (Exception e) {
                                                                e.printStackTrace();
                                                        }
                                                } else {
                                                        helpMsgTxt = "効果がないようだ
";
                                                }
                                                helpMsgTxtIndex = -2;
                                        }
                                        if ("TREASURE".equals(act)){
                                                if(FLAG_ITEM2 != 1){
                                                        helpMsgTxt = "";
                                                        sleep(500);
                                                        helpMsgTxt = "    にじのしずく
を手に入れた！！";
                                                        FLAG_ITEM2 = 1;
                                                        try {

        MyUtil.playSound_SE(activity, MyUtil.SE_CURE);
                                                        } catch (Exception e) {
                                                                e.printStackTrace();
                                                        }
                                                }else{
                                                        helpMsgTxt = "…もう何も無い。
";
                                                }
                                        }
                                        break;    //CAUTION!

                                //オオサキ****************************************
                                case R.string.oosaki:
                                case R.string.oosaki2:
                                        if ("NPC1".equals(act))
                                                /*S*/ helpMsgTxt = "「ここはオオサキの
町！！」";
                                        if ("NPC2".equals(act))
                                                /*T*/ helpMsgTxt = "「西の行き止まりが気
になる…」";
                                        if ("NPC3".equals(act))
                                                /*U*/ helpMsgTxt = "「向こう岸へはどうや
って行くんだ？」";
                                        if ("NPC4".equals(act))
                                                /*V*/ helpMsgTxt = "「センダイの扉の先に
は何が？」";
                                        if ("NPC5".equals(act))
                                                /*W*/ helpMsgTxt = "「センダイでステータ
スアップはしたか？」";
                                        if ("NPC6".equals(act))
                                                /*X*/ helpMsgTxt = "「センダイの…西の湖
…」";
                                        if ("NPC7".equals(act))
                                                /*Y*/ helpMsgTxt = "「古い言い伝え…」";
                                        if ("NPC8".equals(act))
                                                /*Z*/ helpMsgTxt = "「オーブを捧げると…」
```

```
";
                                        if ("TREASURE".equals(act)){
                                                if(FLAG_ITEM3 != 1){
                                                        helpMsgTxt = "";
                                                        sleep(500);
                                                        helpMsgTxt = "    オーブを手に
入れた！！";
                                                        FLAG_ITEM3 = 1;
                                                        try {

        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                                        } catch (Exception e) {
                                                                e.printStackTrace();
                                                        }
                                                }else{
                                                        helpMsgTxt = "…もう何も無い。
";
                                                }
                                        }
                                        break;    //CAUTION!

                                //レイク**************************************
                                case R.string.lake:
                                case R.string.lake2:
                                        if ("NPC1".equals(act))
                                                /*S*/ helpMsgTxt = "「…ヒホウハオレノモ
ノダッ！！」";
                                        if ("TREASURE".equals(act) && FLAG_BOSS1 == 1){
                                                helpMsgTxt = "";
                                                sleep(500);
                                                helpMsgTxt = "    伝説の秘宝を手に入れ
た！！";
                                                try {
                                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_CURE);
                                                        setStatus(STATUS_ENDING,
true);
                                                        init(STATUS_ENDING);
                                                } catch (Exception e) {
                                                        e.printStackTrace();
                                                }
                                        }
                                        if ("TREASURE".equals(act) && FLAG_BOSS1 == 0){
                                                FLAG_BOSS1 = 1;//ボスキャラフラグ
                                                TouchStatus = -1;
                                                setStatus(STATUS_BATTLE,        true);//
        戦闘シーンへ
                                        }
                                        break;

                                case R.string.nanasi:
                                        helpMsgTxt = "…怪しい！";
                                        if (playerMapPosX == 15 && playerMapPosY == 10){
                                                if ("GRASS".equals(act)){
                                                        if(FLAG_ITEM1 != 1){
                                                                helpMsgTxt = "";
                                                                sleep(500);
                                                                helpMsgTxt = "    カ
ギを手に入れた！！";
                                                                FLAG_ITEM1 = 1;
```

```
                                                                try {

            MyUtil.playSound_SE(activity, MyUtil.SE_CURE);
                                                                } catch (Exception e)
{

            e.printStackTrace();
                                                                }
                                                            }else{
                                                                helpMsgTxt = "…もう
何も無い。";
                                                            }
                                                        }
                                                    }
                                                    break;    //CAUTION!
                                            }
                                        }

                                        //        地図の描写
                                        try {
                                                canvas.drawBitmap(mapImage, mapMarginX, mapMarginY, paint);
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                                android.util.Log.e(DebugLog.TAG, "map map" + playerMapPosY + ", " +
playerMapPosX + ", " + fieldMapWidth + ", " + fieldMapHeight);
                                                sleep(2000);
                                        }

                                        try {
                                                //        プレイヤーの描写（必ず中央）
                                                canvas.drawBitmap(playerImage[playerDirection][playerDrawIndex],
playerDrawPosX, playerDrawPosY, paint);

                                                //        ステータス枠を描写
                                                paint.setStyle(Paint.Style.STROKE);
                                                paint.setStrokeWidth(4);
                                                paint.setColor(Color.WHITE);
                                                canvas.drawRoundRect(new
RectF(mapMarginX+2,statusDrawPosY,displayWidth-mapMarginX*2+2,statusDrawPosY+60),10,10,paint);
                                                canvas.drawRoundRect(new
RectF(mapMarginX+2,statusDrawPosY+68,displayWidth-mapMarginX*2+2,statusDrawPosY+68+30),10,10,pai
nt);

                                                //        ステータス枠の中を描写
                                        paint.setStyle(Paint.Style.FILL);
                                                paint.setStrokeWidth(1);
                                                paint.setTextSize(drawFontSize); //drawFontSize
                                                String line1 = "Lv:" + level + " HP:" + hp + "/" + maxHp + " SP:" + sp
+ "/" + maxSp;
                                                canvas.drawText(line1,  mapMarginX + 10, statusDrawPosY+25, paint);

                                                paint.setTextSize(13);
                                                String line2 = "経験値:" + exp + " / " + nextExp + "  攻:" + str + "
防:" + def + "  速:" + spd;
                                                canvas.drawText(line2, mapMarginX + 10, statusDrawPosY+50, paint);

                                                //        ヘルプテキストを描写
                                                paint.setTextSize(drawFontSize);
                                                int idx = helpMsgTxtIndex;
                                                if (idx < 0) idx = 0;
```

```
                        for (int i = idx, x = mapMarginX + 10; i < helpMsgTxt.length(); i ++)
{
                                String c = helpMsgTxt.substring(i, i+1);
                                float f = paint.measureText(c);
                                if (x+f >= displayWidth-mapMarginX*2-14) {
                                        break;
                                }
                                canvas.drawText(c, x, statusDrawPosY+90, paint);
                                x += f;
                        }
                        if (count%5 == 0) {
                                helpMsgTxtIndex ++;
                                if (helpMsgTxtIndex >= helpMsgTxt.length()) {
                                        helpMsgTxtIndex = -2;
                                        helpMsgTxt = "";
                                }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                        android.util.Log.e(DebugLog.TAG, "map player");
                }

                if (moved) {
                        //      敵の出現をランダムで設定。（encount分の1で、敵出現）
                        if (encFlag == true && MyUtil.nextInt(encount) == 0){
                                //      フィールドBGMを停止して、エンカウントSE鳴動
                                try {
                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_MAP_ENCOUNT);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                                sleep(500);
                                TouchStatus = -1;
                                setStatus(STATUS_BATTLE, true);       //      戦闘シーン
へ
                                return;
                        }
                }

        if (count%3 == 0) playerDrawIndex = (playerDrawIndex+1)%2;
         }

         /**
    * メニューを描写します。
         * @param canvas    描写キャンバスクラス
         * @param paint     描写ペイントクラス
         */
        private void menu(Canvas canvas, Paint paint) {
                //TODO 実装されていません！
        }

         /**
    * 戦闘シーンを描写します。
         * @param canvas    描写キャンバスクラス
         * @param paint     描写ペイントクラス
         */
        @SuppressWarnings("unchecked")
        private void battle(Canvas canvas, Paint paint) throws Exception {
```

```
                    //特殊攻撃対応
                    boolean damagein = false;
                    if(selectFlag == true) damagein = true;

                    //        戦闘アクション中は、キー操作を無効とする。
                    if (battleActions.length != 0 && battleActions[0] != null) TouchStatus = -1;
          //        戦闘アクションがあれば、キー操作無効
                    //        タッチイベント処理
                    switch (TouchStatus){
                    case TouchStatus_UP:
                              if(Flag_LongTouch != true) TouchStatus = -1;
                              if (battleEnemySelectedIndex == -1) {
                                        //        敵選択中ではない場合＝戦闘メニュー選択中
                                        battleMenuSelectedIndex = (battleMenuSelectedIndex+2)%4;
                              } else {
                                        //        敵選択中だったので、戦闘メニューに戻す。
                                        battleEnemySelectedIndex = -1;
                              }
                              break;
                    case TouchStatus_DOWN:
                              if(Flag_LongTouch != true) TouchStatus = -1;
                              if (battleEnemySelectedIndex == -1) {
                                        //        敵選択中ではない場合＝戦闘メニュー選択中
                                        battleMenuSelectedIndex = (battleMenuSelectedIndex+2)%4;
                              } else {
                                        //        敵選択中だったので、戦闘メニューに戻す。
                                        battleEnemySelectedIndex = -1;
                              }
                              break;
                    case TouchStatus_LEFT                :
                              if(Flag_LongTouch != true) TouchStatus = -1;
                              if (battleEnemySelectedIndex == -1) {
                                        //        敵選択中ではない場合＝戦闘メニュー選択中
                                        battleMenuSelectedIndex                                        =
battleMenuSelectedIndex/2*2+(battleMenuSelectedIndex+1)%2;
                              } else {
                                        //        敵選択中なので、敵の選択を行う
                                        int idx = battleEnemySelectedIndex;
                                        battleEnemySelectedIndex --;
                                        if (battleEnemySelectedIndex < 0) battleEnemySelectedIndex =
enemy.length-1;
                                        while (enemy[battleEnemySelectedIndex].hp <= 0) {
                                                  battleEnemySelectedIndex --;
                                                  if       (battleEnemySelectedIndex       <       0)
battleEnemySelectedIndex = enemy.length-1;
                                                  if (battleEnemySelectedIndex == idx) break;
                                        }
                              }
                              break;
                    case TouchStatus_RIGHT:
                              if(Flag_LongTouch != true) TouchStatus = -1;
                              if (battleEnemySelectedIndex == -1) {
                                        //        敵選択中ではない場合＝戦闘メニュー選択中
                                        battleMenuSelectedIndex                                        =
battleMenuSelectedIndex/2*2+(battleMenuSelectedIndex+1)%2;
                              } else {
                                        //        敵選択中なので、敵の選択を行う
                                        int idx = battleEnemySelectedIndex;
                                        battleEnemySelectedIndex ++;
                                        if      (battleEnemySelectedIndex      >=      enemy.length)
```

```
battleEnemySelectedIndex = 0;
                                        while (enemy[battleEnemySelectedIndex].hp <= 0) {
                                                battleEnemySelectedIndex ++;
                                                if  (battleEnemySelectedIndex  >=  enemy.length)
battleEnemySelectedIndex = 0;
                                                if (battleEnemySelectedIndex == idx) break;
                                        }
                                }
                                break;
                        case TouchStatus_TAP:
                                //      選択キーの処理
                                if (battleEnemySelectedIndex == -1) {
                                        //      敵選択中ではない場合＝戦闘メニュー選択中
                                        if (battleMenuSelectedIndex == 0) {
                                                //      「戦う」が選択されました。
                                                TouchStatus = -1;
                                                for (int i = 0; i < enemy.length; i ++) {
                                                        if (enemy[i].hp > 0) {
                                                                battleEnemySelectedIndex = i;
                                                                break;
                                                        }
                                                }
                                                battleMsgTxts = new String[4];
                                                battleMsgTxts[0] = "攻撃対象を選択してください。";
                                                battleMsgMasterTxts = new String[0];
                                                count = 1;

                                                selectFlag = true;//特殊攻撃対応

                                        } else if (battleMenuSelectedIndex == 1) {
                                                //      「特殊」が選択されました。
                                                if (sp <= reqSpec){
                                                        battleMsgTxts = new String[4];
                                                        battleMsgTxts[0] = "SPが足りません";
                                                }else {
                                                        TouchStatus = -1;
                                                        sp -= reqSpec;
                                                        for (int i = 0; i < enemy.length; i ++) {
                                                                if (enemy[i].hp > 0) {

        battleEnemySelectedIndex = i;
                                                                        break;
                                                                }
                                                        }
                                                        battleMsgTxts = new String[4];
                                                        battleMsgTxts[0] = "";
                                                        battleMsgMasterTxts = new String[0];
                                                        count = 1;
                                                        damagein = true;
                                                        paintFlag = true;
                                                }
                                        } else if (battleMenuSelectedIndex == 2) {
                                                //      「回復」が選択されました。
                                                if (sp < reqCure){
                                                        battleMsgTxts = new String[4];
                                                        battleMsgTxts[0] = "SPが足りません";
                                                }else{
                                                        TouchStatus = -1;
                                                        sp -= reqCure;
                                                        for (int i = 0; i < enemy.length; i ++) {
```

```
                                                if (enemy[i].hp > 0) {

        battleEnemySelectedIndex = i;
                                                        break;
                                                }
                                        }
                                        battleMsgTxts = new String[4];
                                        battleMsgTxts[0] = "";
                                        battleMsgMasterTxts = new String[0];
                                        count = 1;
                                        damagein = true;
                                }
                        } else if (battleMenuSelectedIndex == 3) {
                                //      「逃げる」が選択されました。
                                //      今回は、必ず、逃げれます。
                                //      ただし、ボス戦時は選択不可とする！
                                if(FLAG_BOSS1 != 1){
                                        TouchStatus = -1;
                                        battleEnemySelectedIndex = -1;
                                        sleep(500);
                                        setStatus(STATUS_MAP,           false);
        //      地図へ。
                                        return;
                                }
                        }
                //} else {
                }
                //特殊攻撃・回復対応
                if (damagein == true) {
                        selectFlag = false;

                        //      敵が選択されました。
                        count = 1;
                        //      戦闘の順番を決めます。（素早さから判断する）
                        ArrayList list = new ArrayList();
                        for (int i = 0; i < enemy.length; i ++) {
                                list.add(enemy[i]);
                        }
                        Enemy player = new Enemy(this, new int[0], "ミヤギ君", hp, str,
def, spd, -1, 0, 0);
                        list.add(player);
                        Object obj[] = list.toArray();
                        Arrays.sort(obj, new EnemyComparator(null));    //
        ここで、ソートされます。
                        ArrayList battleActionList = new ArrayList();
                        Enemy skip = null;

                        //      ソートされた結果で、戦闘アクションを追加していきま
す。
                        for (int i = 0; i < obj.length; i ++) {
                                Enemy e = (Enemy)obj[i];
                                if (e == player) {
                                        //      自分の番になった場合の処理

                                        //戦う
                                        if(battleMenuSelectedIndex == 0){
                                                int             idx             =
battleEnemySelectedIndex;
                                                int             damage          =
enemy[idx].getAttackDamage(player);
```

```
                                                    battleActionList.add("MESSAGE:
ミヤギ君の攻撃");

            battleActionList.add("SLEEP:500");
                                                    battleActionList.add("ATTACK:"
+ idx + ":"+damage+":"

                    + enemy[idx].name + "に、¥n"

                    + damage + "のダメージを与えた！");

            battleActionList.add("SLEEP:500");
                                                    int enemyHp = enemy[idx].hp -
damage;
                                                    if (enemyHp <= 0) {
                                                            enemy[idx].skip     =
true;
                                                            //        敵 を 倒 し
た！！

            battleActionList.add("MESSAGE:"+ enemy[idx].name + "を倒した！");
                                                    }

                                                //特殊攻撃
                                                }else if(battleMenuSelectedIndex == 1){
                                                    battleActionList.add("MESSAGE:
ミヤギ君の特殊攻撃");

            battleActionList.add("SLEEP:500");
                                                    for(int l = 0; l < enemy.length;
l++){
                                                            if(enemy[l].hp <= 0){
                                                                    continue;
                                                            }else{
                                                                    int damage2
= enemy[l].getAttackDamage(player);

            battleActionList.add("MAGIC:" + l + ":"+damage2+":"

                                    + enemy[l].name + "に、¥n"

                                    + damage2 + "のダメージを与えた！");

            battleActionList.add("SLEEP:500");
                                                                    int
enemyHp2 = enemy[l].hp - damage2;

                                                                    if
(enemyHp2 <= 0) {

            enemy[l].skip = true;

            //        敵を倒した！！

            battleActionList.add("MESSAGE:"

                                    + enemy[l].name + "を倒した！");
                                                                    }
                                                            }
                                                    }
```

```
                                                //回復行動
                                                }else if(battleMenuSelectedIndex == 2){
                                                        battleActionList.add("MESSAGE:
ミヤギ君の回復");

        battleActionList.add("SLEEP:500");

        battleActionList.add("RECOVER:");

        battleActionList.add("MESSAGE:" + "ミヤギ君はHPを回復した");

        battleActionList.add("SLEEP:500");
                                                }
                                        } else {
                                                //        敵の番になった場合の処理
                                                if (e.hp <= 0) continue;
        //        すでに倒している敵では、処理せず。
                                                //else  if  (e.equals(skip))  continue;
        //        自分の番の時に、倒した敵なので、処理せず。
                                                else if (e.skip == true) continue;
                                                int damage = player.getAttackDamage(e);
                                                battleActionList.add("MESSAGE:" + e.name
+ "の攻撃！");
                                                battleActionList.add("SLEEP:500");
                                                battleActionList.add("ATTACKED:" + e.idx
+ ":"+damage

                + ":" + damage + "のダメージ！");
                                                battleActionList.add("SLEEP:500");
                                        }
                                }

                                //        敵を全部倒したかの判定
                                int enemyExp = 0;
                                int cnt = 0;
                                for (int i = 0; i < enemy.length; i ++) {
                                        //if (enemy[i].hp <= 0 || enemy[i].equals(skip)) {
                                        if (enemy[i].hp <= 0 || enemy[i].skip == true) {
                                                cnt ++;
                                                enemyExp += enemy[i].exp;
                                        }
                                }
                                if (cnt == enemy.length) {
                                        battleActionList.add("MESSAGE:敵を全部倒した！¥n"

                + enemyExp + "の経験値を手に入れた！");
                                        battleActionList.add("SLEEP:1000");
                                        exp += enemyExp;
                                        //        レベルアップ値の決定
                                        if (exp >= nextExp) {
                                                int maxHpUp = MyUtil.nextInt(15)+5;
                                                int maxSpUp = MyUtil.nextInt(level)+10;
                                                int strUp = MyUtil.nextInt(5)+2;
                                                int defUp = MyUtil.nextInt(5)+2;;
                                                int spdUp = MyUtil.nextInt(5)+2;
                                                maxspup = maxSpUp;
                                                battleActionList.add("MESSAGE:レベルア
ップ！！！");
                                                battleActionList.add("SLEEP:1000");
```

```
                                                battleActionList.add("LEVELUP:"        +
maxHpUp + "." + strUp + "." + defUp + "." + spdUp

                    + ":最大HP・SPが" + maxHpUp + "・"+ maxSpUp + "アップ！"

                    + "¥n攻撃力が、"+ strUp + "アップ！"

                    + "¥n防御力が、"+ defUp + "アップ！"

                    + "¥n素早さが、"+ spdUp + "アップ！");
                                                battleActionList.add("SLEEP:1000");
                                        }
                                        battleActionList.add("FINISH:");
                                } else {
                                        battleActionList.add("CLEAR:");
                                }
                                battleActions = MyUtil.a2s(battleActionList);
                        }
                }

                if (count%5 == 0) {
                        if (battleActions.length != 0 && battleActions[0] != null) {
                                //      戦闘アクションごとに処理を行っていく。
                                String action = battleActions[0];
                                if (action.startsWith("ATTACK:")) {
        //      「攻撃した」
                                        battleMsgTxts = new String[4];
                                        String                  msgs[]                  =
MyUtil.split(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] =
msgs[i];
                                        }
                                        String acts[] = MyUtil.split(action, ":");
                                        int idx = Integer.parseInt(acts[1]);
                                        int damage = Integer.parseInt(acts[2]);
                                        int enemyHp = enemy[idx].setDamage(damage);
                                        if (enemyHp <= 0) {
                                android.util.Log.e(DebugLog.TAG, "Done ![" + enemyHp + "]");
                                        }
                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_BATTLE_ATTACK_PLAYER);

                                }else if(action.startsWith("MAGIC:")){
                                        //特殊攻撃
                                        battleMsgTxts = new String[4];
                                        String                  msgs[]                  =
MyUtil.split(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] =
msgs[i];
                                        }
                                        String acts[] = MyUtil.split(action, ":");
                                        int idx = Integer.parseInt(acts[1]);
                                        int damage = (int) (Integer.parseInt(acts[2]) *
0.8);
                                        int enemyHp = enemy[idx].setDamage(damage);
                                        if (enemyHp <= 0) {
                                android.util.Log.e(DebugLog.TAG, "Done ![" + enemyHp + "]");
                                        }
```

```
                                        MyUtil.playSound_SE(activity, MyUtil.SE_MAGIC);
                                        paintFlag = false;

                                }else if(action.startsWith("RECOVER:")){

                                        //回復行動
                                        int recovery = (int)(maxHp * 0.3);
                                        if(hp + recovery > maxHp){
                                                hp = maxHp;
                                        }else{
                                                hp += recovery;
                                        }
                                        MyUtil.playSound_SE(activity, MyUtil.SE_CURE);

                                } else if (action.startsWith("ATTACKED:")) {    //
        「攻撃された」
                                        battleMsgTxts = new String[4];
                                        String                  msgs[]                  =
MyUtil.split(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] =
msgs[i];
                                        }
                                        String acts[] = MyUtil.split(action, ":");
                                        int damage = Integer.parseInt(acts[2]);
                                        hp = hp - damage;
                                        MyUtil.playSound_SE(activity,
MyUtil.SE_BATTLE_ATTACK_ENEMY);

                                        if (hp <= 0) {
                                                hp = 0;
                                                //        ゲームオーバーへ。
                                                ArrayList    battleActionList    =    new
ArrayList();

        battleActionList.add(action.substring(action.lastIndexOf(":")+1));
                                                battleActionList.add("SLEEP:500");
                                                battleActionList.add("MESSAGE:ミヤギ君
は...¥n力尽きた...。");
                                                battleActionList.add("SLEEP:1000");
                                                battleActionList.add("MESSAGE:ミヤギ君
は...¥n力尽きた...。¥n¥nむっ、無念だ！！！");
                                                battleActionList.add("SLEEP:2000");
                                                battleActionList.add("GAMEOVER:");
                                                battleActions                           =
MyUtil.a2s(battleActionList);
                                        android.util.Log.e(DebugLog.TAG, "GAMEOVER[" + hp + "]");
                                        }
                                } else if (action.startsWith("MESSAGE:")) {
        //        「メッセージ表示」
                                        battleMsgTxts = new String[4];
                                        String                  msgs[]                  =
MyUtil.split(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] =
msgs[i];
                                        }
                                } else if (action.startsWith("SLEEP:")) {
        //        「処理一時停止」
                                        long                    sleep                   =
```

```
Long.parseLong(action.substring(action.lastIndexOf(":")+1));
                                        sleep(sleep);
                                } else if (action.startsWith("LEVELUP:")) {
        //      「レベルアップ」
                                        String up[] = MyUtil.split(action, ":");
                                        up = MyUtil.split(up[1], ".");
                                        int maxHpUp = Integer.parseInt(up[0]);
                                        int strUp = Integer.parseInt(up[1]);
                                        int defUp = Integer.parseInt(up[2]);
                                        int spdUp = Integer.parseInt(up[3]);
                                        level ++;
                                        maxHp += maxHpUp;
                                        hp = maxHp;
                                        maxSp += maxspup;
                                        sp = maxSp;
                                        str += strUp;
                                        def += defUp;
                                        spd += spdUp;
                                        nextExp += 100*(level-1);
                                        battleMsgTxts = new String[4];
                                        String                  msgs[]                  =
MyUtil.split(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] =
msgs[i];
                                        }
                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_BATTLE_LEVELUP);
                                } else if (action.startsWith("CLEAR:")) {
        //      「クリア」
                                        battleMsgTxts = new String[4];
                                } else if (action.startsWith("FINISH:")) {
        //      「終了」
                                        TouchStatus = -1;
                                        battleEnemySelectedIndex = -1;
                                        sleep(500);
                                        setStatus(STATUS_MAP, false);
                //      地図へ
                                        return;
                                } else if (action.startsWith("GAMEOVER:")) {   //
        「ゲームオーバー」
                                        TouchStatus = -1;
                                        battleEnemySelectedIndex = -1;
                                        sleep(500);
                                        setStatus(STATUS_GAMEOVER, true);
        //      ゲームオーバーへ
                                        return;
                                }
                                DebugLog.println("battle action[" + action + "]");
                                //      戦闘アクションのスワップ（差し替え）
                                for (int i = 0; i < battleActions.length-1; i ++) {
                                        battleActions[i] = battleActions[i+1];

        DebugLog.println("battleActions[i]:"+battleActions[i]);
                                }
                                battleActions[battleActions.length-1] = null;
                                count = 1;
                                if (battleActions[0] == null) {
                                        //      元に戻る。
                                        battleEnemySelectedIndex = -1;
```

```
                                        }
                                }
                        }

                        //戦闘背景の描画
                        canvas.drawBitmap(batteImage, 0, 0,paint);

                        //        敵の描写
                        paint.setStyle(Paint.Style.STROKE);
                        paint.setStrokeWidth(2);
                        paint.setColor(Color.RED);
                        for (int i = 0; i < enemy.length; i ++) {
                                //enemy[i].drawEnemy(canvas,      paint,      count%4==0,      i      ==
battleEnemySelectedIndex);
                                if(battleMenuSelectedIndex == 0){
                                        enemy[i].drawEnemy(canvas,    paint,    count%4==0,    i    ==
battleEnemySelectedIndex);
                                }else if(paintFlag == true){
                                        enemy[i].drawEnemy(canvas, paint, count%4==0, true);
                                }else{
                                        enemy[i].drawEnemy(canvas, paint, count%4==0, false);
                                }
                        }

                        //        戦闘シーンの枠を描写
                        paint.setStyle(Paint.Style.STROKE);
                        paint.setStrokeWidth(4);
                        paint.setColor(Color.WHITE);
                        canvas.drawRoundRect(new
RectF(battleMarginX+2,displayHeight/2+battleMarginY,displayWidth-battleMarginX*2+2,displayHeight
/2+60+battleMarginY),10,10,paint);
                        canvas.drawRoundRect(new
RectF(battleMarginX+2,displayHeight/2+68+battleMarginY,displayWidth-battleMarginX*2+2,displayHei
ght/2+68+110+battleMarginY),10,10,paint);
                        canvas.drawRoundRect(new
RectF(mapMarginX+2,statusDrawPosY+68,displayWidth-mapMarginX*2+2,statusDrawPosY+68+30),10,10,pai
nt);

                        //        戦闘シーンの枠の中を描写
                paint.setStyle(Paint.Style.FILL);
                        paint.setStrokeWidth(1);
                        paint.setTextSize(drawFontSize);
                        canvas.drawText("戦　う", battleMarginX + 50,
                                                                                displayHeight / 2 + 25
+ battleMarginY, paint);

                        canvas.drawText("特　殊", displayWidth / 2 + 45,
                                                                                displayHeight / 2 + 25
+ battleMarginY, paint);
                        canvas.drawText("回　復", battleMarginX + 50,
                                                                                displayHeight / 2 + 50
+ battleMarginY, paint);
                        //ボス戦時は選択不可とする！
                        if(FLAG_BOSS1 != 1){
                                canvas.drawText("逃げる", displayWidth / 2 + 45,
                                                                                displayHeight / 2 + 50
+ battleMarginY, paint);
                        }
                        canvas.drawText("HP　: " + hp + "/" + maxHp,  mapMarginX + 10, statusDrawPosY +
90, paint);
```

```
                    canvas.drawText("SP  : " + sp + "/" + maxSp,  mapMarginX + 150, statusDrawPosY
+ 90, paint);

                    //        選択されている戦闘メニューの描写
                    if (battleMenuSelectedIndex == 0) {
                              canvas.drawText(">", battleMarginX + 50 - battleMenuSelectedMinusX,
                                                                                  displayHeight/2  +
25 + battleMarginY, paint);
                              canvas.drawText("<", battleMarginX + 50 + battleMenuSelectedPlusX,
                                                                                  displayHeight/2  +
25 + battleMarginY, paint);
                    } else if (battleMenuSelectedIndex == 1) {
                              canvas.drawText(">", displayWidth / 2 + 45 - battleMenuSelectedMinusX,
                                                                                  displayHeight / 2 +
25 + battleMarginY, paint);
                              canvas.drawText("<", displayWidth / 2 + 45 + battleMenuSelectedPlusX,
displayHeight/2+25+battleMarginY, paint);
                    } else if (battleMenuSelectedIndex == 2) {
                              canvas.drawText(">", battleMarginX + 50 - battleMenuSelectedMinusX,
                                                                                  displayHeight / 2 +
50 + battleMarginY, paint);
                              canvas.drawText("<", battleMarginX + 50 + battleMenuSelectedPlusX,
                                                                                  displayHeight / 2 +
50 + battleMarginY, paint);
                    } else if (battleMenuSelectedIndex == 3) {
                              //ボス戦時は選択不可とする！
                              if(FLAG_BOSS1 != 1){
                                        canvas.drawText(" > ",  displayWidth  /  2  +  45  -
battleMenuSelectedMinusX,
                                                                                  displayHeight / 2 +
50 + battleMarginY, paint);
                                        canvas.drawText(" < ",  displayWidth  /  2  +  45  +
battleMenuSelectedPlusX,
                                                                                  displayHeight / 2 +
50 + battleMarginY, paint);
                              }
                    }

                    //        戦闘メッセージの描写
                    for (int i = 0, j = 0; i < battleMsgTxts.length; i ++, j ++) {
                              if (battleMsgTxts[i] == null) continue;
                              canvas.drawText(battleMsgTxts[i],  battleMarginX + 10,
                                                  displayHeight / 2 + 68 + 25 + j * 25
+battleMarginY, paint);
                              DebugLog.println("canvas.drawText:"+battleMsgTxts[i]);
                    }

                    if (count%5 == 0) {
                              //        戦闘メッセージのスワップ（入れ替え）
                              for (int i = 0; i < battleMsgTxts.length-1; i ++) {
                                        battleMsgTxts[i] = battleMsgTxts[i+1];
                              }
                              battleMsgTxtIndex ++;
                              if    (battleMsgTxtIndex    <    0    ||    battleMsgTxtIndex    >=
battleMsgMasterTxts.length) {
                                        battleMsgTxts[battleMsgTxts.length-1] = null;
                              } else {
                                        battleMsgTxts[battleMsgTxts.length-1]                    =
battleMsgMasterTxts[battleMsgTxtIndex];
                              }
```

```
                }
        }

        /**
     * ゲームオーバーを描写します。
         * @param canvas    描写キャンバスクラス
         * @param paint     描写ペイントクラス
         */
        private void gameover(Canvas canvas, Paint paint) {
                //      キーイベント処理

                switch (TouchStatus) {
                case TouchStatus_TAP                    :
                        //      選択キーの処理
                        sleep(500);
                        TouchStatus = -1;
                        setStatus(STATUS_TITLE, true);          //      タイトルへ
                        //return;
                        //TODO 強制終了した方が良い
                        System.exit(0);

                }
                //      ゲームオーバーを描写
                paint.setTextSize(drawFontSize);
                paint.setColor(Color.WHITE);
                canvas.drawText("Game Over", 20, displayHeight/2-drawFontSize/2, paint);
        }

        private void ending(Canvas canvas, Paint paint) {

                //      描写位置の下限
                if (rollMsgPosY < 50) rollMsgPosY = 50;

                //      ロールメッセージを表示させます。
                paint.setTextSize(drawFontSize);
                for (int i = 0; i < rollMsgTxt.length; i ++) {
                        paint.setColor(Color.WHITE);
                        canvas.drawText(rollMsgTxt[i],    rollMarginX,    rollMsgPosY    +
i*(drawFontSize+5), paint);
                }
                //      ロールメッセージが、下限になった場合のみ、「Hit Key」を表示させて、次
を促します。
                if (rollMsgPosY == 50) {
                        paint.setColor(Color.WHITE);
                        canvas.drawText("Please    press    Hit    Key",    rollMarginX,
displayHeight-drawFontSize-20, paint);
                        paint.setColor(Color.BLUE);
                        canvas.drawText("Please    press    Hit    Key",    rollMarginX-1,
displayHeight-drawFontSize-20-1, paint);
                }
                rollMsgPosY -= 5;

                switch (TouchStatus) {
                case TouchStatus_TAP                    :
                        //      選択キーの処理
                        sleep(500);
                        TouchStatus = -1;
                        setStatus(STATUS_TITLE, true);          //      タイトルへ
                        return;
                }
```

```
        }

        /**
         * エラーを描写します。
         * @param canvas    描写キャンバスクラス
         * @param paint     描写ペイントクラス
         */
        private void error(Canvas canvas, Paint paint) {
                paint.setColor(Color.WHITE);
                canvas.drawText(errorTxt, 20, 20, paint);
        }

        /**
         * 描写処理
         * @param canvas    描写キャンバスクラス
         */
        @Override
        protected void onDraw(Canvas canvas) {
                if (status == STATUS_WAIT) return;

                super.onDraw(canvas);
        // 背景色の設定
        canvas.drawColor(Color.BLUE); //DKGRAY
        // 描画オブジェクトの生成
        Paint paint = new Paint();
        paint.setAntiAlias(true);
        paint.setStyle(Paint.Style.FILL);

        try {
         //       ステータスごとに、描写を振り分ける
                switch (status) {
                case STATUS_TITLE:
                 MyUtil.PlaySound_BGM(activity, MyUtil.BGM_TITLE);
                                title(canvas, paint);
                                break;
                case STATUS_ROLL:
                                roll(canvas, paint);
                                break;
                case STATUS_MAP:
                 MyUtil.PlaySound_BGM(activity, MyUtil.BGM_MAP);
                                map(canvas, paint);
                                break;
                case STATUS_MENU:
                 menu(canvas, paint);
                 break;
                case STATUS_BATTLE:
                 MyUtil.PlaySound_BGM(activity, MyUtil.BGM_BATTLE);
                                battle(canvas, paint);
                                break;
                case STATUS_GAMEOVER:
                 gameover(canvas, paint);
                 break;
                case STATUS_ENDING:
                 ending(canvas, paint);
                 break;
                case STATUS_ERROR:
                 error(canvas, paint);
                 break;
                default:
                 break;
```

```
                }
        } catch (Exception e) {
          android.util.Log.e(DebugLog.TAG, e.toString());
        }
        count ++;
          }

          /**
           *  画面サイズが変更されたときに呼び出されるメソッド
           **/
    @Override
          protected void onSizeChanged(int w, int h, int oldw, int oldh){
        displayWidth  = w;
        displayHeight = h;
                DebugLog.println("onSizeChanged(" + displayWidth + ", " + displayHeight + ")");
    }

    /**
     * 保存ファイル名を返す
     * @return      保存ファイル名
     **/
    public String getFileName() {
          return "/data/data/" + getContext().getPackageName() + "/save.txt";
    }

    /**
     * データを保存する
     * @return      保存結果
     **/
          public boolean save() {
                // 保存先の決定
                String fname = getFileName();

                StringBuffer buf = new StringBuffer();
                buf.append(level);                              //        レベル
                buf.append(",");
                buf.append(maxHp);                              //        最大HP
                buf.append(",");
                buf.append(hp);                                 //        現在のHP
                buf.append(",");
                buf.append(maxSp);                              //        最大SP
                buf.append(",");
                buf.append(sp);                                 //        現在のSP
                buf.append(",");
                buf.append(str);                                //        攻撃
                buf.append(",");
                buf.append(def);                                //        防御
                buf.append(",");
                buf.append(spd);                                //        素早さ
                buf.append(",");
                buf.append(exp);                                //        経験値
                buf.append(",");
                buf.append(nextExp);                            //        次の経験値
                buf.append(",");
                buf.append(map);                                // マップ番号
                buf.append(",");
                buf.append(playerMapPosX);              //        プレイヤーのマップ上のX座標
                buf.append(",");
                buf.append(playerMapPosY);              //        プレイヤーのマップ上のY座標
                buf.append(",");
```

```
                    //TODO 各種イベントフラグ
                    buf.append(FLAG_SENDAI_WEAPON);
                    buf.append(",");
                    buf.append(FLAG_SENDAI_ARMOR);
                    buf.append(",");
                    buf.append(FLAG_SENDAI_SHIELD);
                    buf.append(",");
                    buf.append(FLAG_ITEM1);
                    buf.append(",");
                    buf.append(FLAG_ITEM2);
                    buf.append(",");
                    buf.append(FLAG_ITEM3);

                    // 保存データをファイルに書き込む
                    DebugLog.println("saveToFile(" + buf.toString() + ") to [" + fname + "]");
                    try {
                            File file = new File(fname);
                            file.getParentFile().mkdirs();

                            FileOutputStream out = new FileOutputStream(file);
                            out.write(buf.toString().getBytes("UTF-8"));
                            out.flush();
                            out.close();
                            return true;
                    } catch(Exception e) {
                            android.util.Log.e(DebugLog.TAG, e.toString());
                            return false;
                    }
            }

            /**
        * データをロードする
        * @return      ロードされた情報
        **/
            public String load() {
                    DebugLog.println("load()");
                    try {
                            String fname = getFileName();
                            DebugLog.println("loadFromFile[" + fname + "]");
                            String data = new String(MyUtil.getFileByteArray(fname), "UTF-8");
                            DebugLog.println(data);
                            return data;
                    } catch (Exception e) {
                            android.util.Log.e(DebugLog.TAG, e.toString());
                            return "";
                    }
            }
}
```

4.4.3 Enemy.java

このファイルは、敵を定義するクラスです。

```
package com.cgene.game;

import android.content.res.Resources;
```

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;

/**
 * 敵を定義するためのクラス
 **/
class Enemy {

	private MiyagiQuestView view;			//	MiyagiQuestViewクラス

	private Bitmap img[];					//	画像

	/** ステータス関連 */
	public int maxHp = 0;					//	最大ヒットポイント
	public int hp = 0;						//	ヒットポイント
	public int str = 0;						//		攻撃力
	public int def = 0;						//		防御力
	public int spd = 0;						//		スピード
	public int exp = 0;						//		経験値
	public String name = "";				//	名称

	public int idx = 0;						//		戦闘シーン中の敵グループ内のインデックス値
	public int maxIndex = 0;				//	戦闘シーン中の敵グループ内の最大インデックス値
	public int drawPosX = 0;				//	描写位置ｘ座標
	public int drawPosY = 0;				//	描写位置ｙ座標

	private int imgIndex = 0;				//	アニメーション用インデックス値

	public boolean skip;					//	スキップフラグ

	/**
	 * 敵を定義します。
	 * @param view		MiyagiQuestView
	 * @param info		敵のステータス値が、[,]区切りで保持されたテキスト
	 * @param idx			戦闘シーン中の敵グループ内のインデックス値
	 * @param maxIndex	戦闘シーン中の敵グループ内の最大インデックス値
	 **/
	@SuppressWarnings("unchecked")
	public Enemy(MiyagiQuestView view, String info, int idx, int maxIndex) throws Exception {
		this.view = view;
		//	ステータス情報を分割
		String data[] = MyUtil.split(info, ",");
		int k = 0;
		//	リソース情報を分割
		String enemyRes[] = MyUtil.split(data[k++], "+");

		//	reflectionを使って、リソースから、画像を読み込みます。
		Class cls = Class.forName("com.cgene.game.R$drawable");
		img = new Bitmap[enemyRes.length];
		Resources r = view.getResources();
		for (int i = 0; i < img.length; i ++) {
			img[i]  =  BitmapFactory.decodeResource(r,  view.getResouceInt(cls,
enemyRes[i]));
		}
```

```
                //        各ステータスを初期化
                this.name= data[k++];
                this.maxHp = Integer.parseInt(data[k++]);
                this.hp = this.maxHp;
                this.str = Integer.parseInt(data[k++]);
                this.def = Integer.parseInt(data[k++]);
                this.spd = Integer.parseInt(data[k++]);
                this.exp = Integer.parseInt(data[k++]);

                this.idx = idx;
                this.maxIndex = maxIndex;
                init();
        }

        /**
         * 敵を定義します。
         * @param view              MiyagiQuestView
         * @param res                       リソースのint値配列
         * @param name              名称
         * @param hp                        ヒットポイント
         * @param str                       攻撃力
         * @param def                       防御力
         * @param spd                       スピード
         * @param exp                       経験値
         * @param idx                       戦闘シーン中の敵グループ内のインデックス値
         * @param maxIndex  戦闘シーン中の敵グループ内の最大インデックス値
         **/
        public Enemy(MiyagiQuestView view, int res[], String name, int hp, int str, int def, int
spd, int exp, int idx, int maxIndex) throws Exception {
                this.view = view;
                img = new Bitmap[res.length];
                Resources r = view.getResources();
                for (int i = 0; i < img.length; i ++) {
                        img[i] = BitmapFactory.decodeResource(r, res[i]);
                }
                this.name = name;
                this.maxHp = hp;
                this.hp = this.maxHp;
                this.str = str;
                this.def = def;
                this.spd = spd;
                this.exp = exp;

                this.idx = idx;
                this.maxIndex = maxIndex;
                init();
        }

        /**
         * 基準値から、微調整された値を返す。
         * @param value     基準値
         * @param rate      1000分率での微調整率
         * @param min               下限値
         * @param minus     正規分布にさせるかどうかのフラグ。trueの場合は、基準値から、上下する。
falseの場合は、基準値から上になる
         * @return                  微調整された値
         **/
        private int getTweakedValue(int value, int rate, int min, boolean minus) {
                int tweak;
                if (minus) {
```

```
                        tweak = value * (MyUtil.nextInt(rate)-rate / 2) / 1000;
                } else {
                        tweak = value * MyUtil.nextInt(rate) / 1000;
                }
                int k = value + tweak;
                if (k < min) k = min;
                return k;
        }

        /**
         * 攻撃するattackerから、ダメージを計算させて、返す。
         * @param attacker 攻撃者
         * @return                              ダメージ
         */
        public int getAttackDamage(Enemy attacker) {
                //      ダメージを算出します。
                int power1 = attacker.str * 100 / this.def;     //      強さレベル
                int power2 = attacker.str - this.def;           //      攻撃力 - 防御力
                if (power2 < 10) power2 = 10 - MyUtil.nextInt(8);
                return power1*power2/100;
        }

        /**
         * ダメージをセットします。
         * @param damage   ダメージ
         * @return                              結果のヒットポイント
         */
        public int setDamage(int damage) {
                hp = hp - damage;
                if (hp < 0) hp = 0;
                return hp;
        }

        /**
         * 各パラメータを基準値から、微調整させる処理と、描写位置を初期化します。
         * @param damage   ダメージ
         * @return                              結果のヒットポイント
         */
        public void init() {
                //      以下、パラメータの初期化。ランダムに上下させる。
                if (maxIndex > 0) {
                        maxHp = getTweakedValue(maxHp, 120, 10, true);
                        hp = maxHp;
                        str = getTweakedValue(str, 100, 1, true);
                        def = getTweakedValue(def, 100, 1, true);
                        spd = getTweakedValue(spd, 100, 1, true);
                }

                //      以下、描写位置の初期化
                if (maxIndex <= 0) {
                        //      何もしない。
                } else if (maxIndex == 1) {
                        drawPosX = view.displayWidth/2 - img[0].getWidth() / 2;
                        drawPosY = 100;
                } else {
                        int w = (view.displayWidth - view.battleMarginX * 2)/maxIndex;
                        drawPosX = w / 2 - img[0].getWidth() / 2 + view.battleMarginX + w * idx;
                        drawPosY = 100;
                }
        }
```

```
	/**
	 * アニメーション処理を行った結果の画像を返す。
	 * @param inc			アニメーションを変更するかどうかのフラグ
	 * @return				画像
	 */
	public Bitmap getEnemyImage(boolean inc) {
		if (inc) imgIndex ++;
		if (imgIndex >= img.length) imgIndex = 0;
		return img[imgIndex];
	}

	/**
	 * 描写処理
	 * @param canvas		描写キャンバスクラス
	 * @param paint			描写ペイントクラス
	 * @param inc			アニメーションを変更するかどうかのフラグ
	 * @param isSelected	現在、選択されているかのフラグ
	 */
	public void drawEnemy(Canvas canvas, Paint paint, boolean inc, boolean isSelected) {
		if (hp <= 0) return;
		canvas.drawBitmap(getEnemyImage(inc), drawPosX, drawPosY, paint);
		if (isSelected) {
			canvas.drawRect(new RectF(drawPosX - 1, drawPosY - 1,
					drawPosX + img[0].getWidth() +
2,
					drawPosY+img[0].getHeight()  +
2), paint);
		}
	}
}
```

4.4.4 EnemyComparator.java

このファイルは、戦闘中の行動の順番を決めるクラスです。

```
package com.cgene.game;

/**
 * ソートクラス
 **/
@SuppressWarnings("unchecked")
class EnemyComparator implements java.util.Comparator{

	// どの値で比較するかのタイプ
	private String type;

	/**
	 *		コンストラクタ。タイプを指定。
	 */
	public EnemyComparator(String type) {
		super();
		this.type = type;
	}

	/**
```

```
		 * 比較処理
		 */
		public int compare(Object o1, Object o2){
			Enemy e1 = (Enemy)o1;
			Enemy e2 = (Enemy)o2;
			if ("maxHp".equals(type)) {
				return e2.maxHp - e1.maxHp;
			} else if ("hp".equals(type)) {
				return e2.hp - e1.hp;
			} else if ("str".equals(type)) {
				return e2.str - e1.str;
			} else if ("def".equals(type)) {
				return e2.def - e1.def;
			} else {
				//		デフォルトは、素早さで比較します。
				return e2.spd - e1.spd;
			}
		}
}
```

4.4.5 MyUtil.java

このファイルは、ユーティリティクラスです。

```
/**
 *	Android RPG MiyagiQuest<br>
 *	<br>
 *	Copyright (c) CGENE Inc. All Right Reserved.<br>
 *	http://cgene.jp/<br>
 *
 *	BGM	http://www.skipmore.com/sound/
 */

package com.cgene.game;

import java.io.*;
import java.util.*;

//import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

//import android.view.KeyEvent;
import android.util.Log;
import android.media.*;

@SuppressWarnings("unused")
public class MyUtil {

	static MediaPlayer mp_Bgm  = null;
	static int BGM_STATUS      = -1;
	static int Bgm_Playing_Now = -1;

	private static final String[]	BGM_Connection_URL =
	{
		"http://httptest.web.fc2.com/sht_a02.ogg",
		"http://httptest.web.fc2.com/sht_a03.ogg",
```

```
                "http://httptest.web.fc2.com/sht_a06.ogg",
        };

    final static public int BGM_TITLE  = 0;
    final static public int BGM_MAP    = 1;
    final static public int BGM_BATTLE = 2;

    static MediaPlayer mp_Se  = null;
    static int SE_STATUS      = -1;
    static int Se_Playing_Now = -1;

        private static final String[]         SE_Connection_URL =
        {
                "R.raw.encount",
                "R.raw.attack_player",
                "R.raw.attack_player2",
                "R.raw.attack_enemy",
                "R.raw.levelup",
                "R.raw.in_out",
                "R.raw.magic"
        };

    final static public int SE_MAP_ENCOUNT                  = 0;
    final static public int SE_BATTLE_ATTACK_PLAYER     = 1;
    final static public int SE_BATTLE_ATTACK_ENEMY      = 2;
    final static public int SE_BATTLE_LEVELUP           = 3;

    final static public int SE_IN_OUT                       = 4;
    final static public int SE_CURE                                 = 5;
    final static public int SE_MAGIC                        = 6;

    final static public int PlaySound_NG         = -1;
    final static public int PlaySound_OK         = 0;
    final static public int PlaySound_PLAYING = 1;

        public static String[] split(String strTarget, String token) {
                return strTarget.split("¥¥Q" + token + "¥¥E", -1);
        }

        @SuppressWarnings("unchecked")
        public static String replace(String strTarget, HashMap conv) {
                for (Iterator it = conv.entrySet().iterator(); it.hasNext();) {
                        Map.Entry entry = (Map.Entry) it.next();
                        strTarget    =    MyUtil.replace(strTarget,    (String)entry.getKey(),
(String)entry.getValue());
                }
                return strTarget;
        }

        public static String replace(String strTarget, String strOldStr, String strOldNew) {
                StringBuffer strResult = new StringBuffer();
                String[] strSplit = split(strTarget, strOldStr);
                strResult.append(strSplit[0]);
                for (int i = 1; i < strSplit.length; i ++) {
                        strResult.append(strOldNew);
                        strResult.append(strSplit[i]);
                }
                return strResult.toString();
        }
```

```
	public static String s2s(String s[], String separator) {
		StringBuffer buf = new StringBuffer();
		for (int i = 0; i < s.length; i ++) {
			if (i != 0) buf.append(separator);
			buf.append(s[i]);
		}
		return buf.toString();
	}

	@SuppressWarnings("unchecked")
	public static String a2s(ArrayList a, String separator) {
		StringBuffer buf = new StringBuffer();
		for (int i = 0; i < a.size(); i ++) {
			if (i != 0) buf.append(separator);
			buf.append((String)a.get(i));
		}
		return buf.toString();
	}

	@SuppressWarnings("unchecked")
	public static String[] a2s(ArrayList a) {
		String s[] = new String[a.size()];
		for (int i = 0; i < s.length; i ++) {
			s[i] = a.get(i).toString();
		}
		return s;
	}

	private static long cnt = 0;

	public static int nextInt(int k) {
		java.util.Date d = new java.util.Date();
		Random r = new Random(d.getTime()+cnt);
		long p = r.nextInt();
		cnt = p;
		return java.lang.Math.abs((int)p)%k;
	}

	@SuppressWarnings("unchecked")
	public static String getDebugText(Map map, String demi) {
		StringBuffer buf = new StringBuffer();
		for (Iterator it = map.entrySet().iterator(); it.hasNext();) {
			Map.Entry entry = (Map.Entry) it.next();
			if (buf.length() != 0) buf.append(demi);
			Object o1 = entry.getKey();
			Object o2 = entry.getValue();
			if (o2 instanceof List) {
				buf.append(o1 + " = {" + getDebugText((List)o2, ", ") + "}");
			} else {
				buf.append(o1 + " = " + o2);
			}
		}
		return buf.toString();
	}

	@SuppressWarnings("unchecked")
	public static String getDebugText(java.util.List list, String demi) {
		StringBuffer buf = new StringBuffer();
		for (int i = 0; i < list.size(); i ++) {
			Object obj = list.get(i);
```

```
                        if (buf.length() != 0) buf.append(demi);
                        if (obj instanceof String[]) {
                                String tmp[] = (String[])obj;
                                for (int j = 0; j < tmp.length; j ++) {
                                        buf.append("[" + tmp[j] + "]");
                                }
                        } else {
                                buf.append(obj);
                        }
                }
                return buf.toString();
        }

        public static String getDebugText(String args[], String demi) {
                StringBuffer buf = new StringBuffer();
                for (int i = 0; i < args.length; i ++) {
                        if (buf.length() != 0) buf.append(demi);
                        buf.append(args[i]);
                }
                return buf.toString();
        }

        public static byte[] getFileByteArray(String file) throws Exception {
                return getFileByteArray(new File(file));
        }

        public static byte[] getFileByteArray(File file) throws Exception {
                BufferedInputStream bis = null;
                ByteArrayOutputStream baos = null;
                try {
                        bis = new BufferedInputStream(new FileInputStream(file));
                        baos = new ByteArrayOutputStream();
                        byte buf[] = new byte[1024];
                        int count;
                        while ((count = bis.read(buf, 0, 1024)) != -1) {
                                baos.write(buf, 0, count);
                        }
                        if (baos != null) {
                                baos.flush();
                                baos.close();
                        }
                        if (bis != null) {
                                bis.close();
                                bis = null;
                        }
                } catch (Exception e) {
                        try {
                                if (baos != null) {
                                        baos.flush();
                                        baos.close();
                                }
                                if (bis != null) {
                                        bis.close();
                                        bis = null;
                                }
                        } catch (Exception ee) {
                        }
                }
                if (baos != null) return baos.toByteArray();
                else return new byte[0];
```

```
	}

	public static boolean writeFile(String file, String body) throws Exception {
		return writeFile(file, body, "Shift_JIS", null);
	}

	public static boolean mkdirs(String file) throws Exception {
		int idx1 = file.indexOf(File.separatorChar);
		int idx2 = file.lastIndexOf(File.separatorChar);
		String dirs = new String();
		if (idx1 == -1) return true;
		else if (idx1 == 0 && idx1 == idx2) dirs = file;
		else if (idx1 == 0) dirs = file.substring(0, idx2);
		else dirs = file.substring(0, idx2);
		File dir = new File(dirs);
		boolean ret = dir.mkdirs();
		return ret;
	}

	public static boolean writeFile(String file, byte buf[]) throws Exception {
		boolean ret = false;
		FileOutputStream fos = null;
		mkdirs(file);
		try {
			fos = new FileOutputStream(new File(file));
			fos.write(buf);
			fos.flush();
			ret = true;
		} catch (Exception e) {
			ret = false;
		}
		if (fos != null) {
			try {
				fos.close();
				fos = null;
			} catch (Exception e) {
			}
		}
		return ret;
	}

	@SuppressWarnings("unchecked")
	public static boolean writeFile(String file, Object obj, String encoding, String sepa) throws
Exception {
		boolean ret = false;
		mkdirs(file);

		BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(new
FileOutputStream(file), encoding));
		try {
			if (obj instanceof ArrayList) {
				bw.write(a2s((ArrayList)obj, sepa));
			} else if (obj instanceof Map) {
				bw.write(getDebugText((Map)obj, sepa));
			} else if (obj instanceof String[]) {
				bw.write(s2s((String[])obj, sepa));
			} else if (obj instanceof StringBuffer) {
				bw.write(obj.toString());
			} else {
				bw.write(obj.toString());
```

```
                        }
                        ret = true;
                } catch (Exception e) {
                        ret = false;
                }
                if (bw != null) {
                        try {
                                bw.close();
                                bw = null;
                        } catch (Exception e) {
                        }
                }
                return ret;
        }

        public static int PlaySound_BGM(MiyagiQuest activity, int BGM_STATUS) throws Exception
        {
                int ret = PlaySound_NG;

                if(BGM_STATUS == Bgm_Playing_Now) {
                        ret = PlaySound_PLAYING;
        return ret;
                }

                switch(BGM_STATUS)
                {
                case BGM_TITLE:
                case BGM_MAP:
                case BGM_BATTLE:
                        Bgm_Playing_Now = BGM_STATUS;
                        break;
                default:
                        DebugLog.println("BGM_STATUS:Error");
        return ret;
                }

/*              if(mp_Bgm != null){
                        DebugLog.println("MediaPlayer:mp_Bgm.Release");
                        mp_Bgm.stop();
                        mp_Bgm.setOnPreparedListener(null);
                        mp_Bgm.release();
                        mp_Bgm = null;
                }

                URLConnection                connection                =                new
URL(BGM_Connection_URL[BGM_STATUS]).openConnection();
                InputStream is = connection.getInputStream();
                File file = new File(activity.getCacheDir(), "tmp");
                FileOutputStream fos = new FileOutputStream(file);
                byte buf[] = new byte[1024];
                do
                {
                        int numread = is.read(buf);
                        if (numread == -1) {
                                break;
                        }
                        fos.write(buf, 0, numread);
                } while (true);
                fos.flush();
                fos.close();
```

```
                mp_Bgm = new MediaPlayer();

            DebugLog.println("new mp_Bgm = "+mp_Bgm);

                mp_Bgm.setDataSource(new FileInputStream(file).getFD());
                mp_Bgm.setOnPreparedListener(new MediaPlayer.OnPreparedListener()
                {
                        //@Override
                        public void onPrepared(MediaPlayer mp_Bgm)
                        {
                                try {
                                        DebugLog.println("STATUS_TITLE");
                                        mp_Bgm.start();
                                        mp_Bgm.setLooping(true);
                                } catch (Exception e) {
                                        Log.d("Error", e.getMessage());
                                        if(mp_Bgm != null) {

        DebugLog.println("catch:MediaPlayer.release");
                                                mp_Bgm.stop();
                                                mp_Bgm.release();
                                                mp_Bgm = null;
                                        }
                                }
                        }
                });
                mp_Bgm.prepareAsync();
*/
                ret = PlaySound_OK;
                return ret;
        }

        public static int PlaySound_SE(MiyagiQuest activity, int SE_STATUS) throws Exception
        {
                int ret = PlaySound_NG;

                if(mp_Se != null) {
                        DebugLog.println("PlaySound_SE:MediaPlayer.release");
                        mp_Se.stop();
                        mp_Se.release();
                        mp_Se = null;
                }

                switch(SE_STATUS)
                {
                case SE_MAP_ENCOUNT:
                        if(mp_Bgm != null) {
                                DebugLog.println("MediaPlayer:mp_Bgm.Release");
                                mp_Bgm.stop();
                                mp_Bgm.setOnPreparedListener(null);
                                mp_Bgm.release();
                                mp_Bgm = null;
                        }
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.encount);
                        break;
                case SE_BATTLE_ATTACK_PLAYER:
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.attack_player);
```

```
                        break;
                case SE_BATTLE_ATTACK_ENEMY:
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.attack_enemy);
                        break;
                case SE_BATTLE_LEVELUP:
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.levelup);
                        break;

                case SE_IN_OUT:
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.in_out);
                        break;
                case SE_CURE:
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.cure);
                        break;
                case SE_MAGIC:
                        Se_Playing_Now = SE_STATUS;
                        mp_Se = MediaPlayer.create(activity, R.raw.magic);
                        break;

                default:
                        DebugLog.println("SE_STATUS:Error");
        return ret;
                }
                mp_Se.start();

                ret = PlaySound_OK;
                return ret;
        }
}
```

4.5 ソースコードの解説

この節では、各ソースコードを解説します。

4.5.1 パッケージ宣言

行頭にある「package」は、そのクラス自身のパッケージ名です。

「import」は、利用するクラスを指定する予約語[9]となります。

MiyagiQuest は、パッケージ名「com.cgene.game」、利用するパッケージは、「java.io.* 」以下の各種パッケージを指定しています。

4.5.2 アクセス修飾子

クラスの先頭に書かれている「public」は、アクセス修飾子と呼ばれるものです。

他のクラスや他のパッケージからのアクセスを許可するかどうかを指定します。「public」以外にも、「指定なし」、「private」、「protected」があり、全部で 4 種類あります。

「public」は、すべてのアクセスを許可します。

「指定なし」は、同じパッケージ内のクラスからのみアクセスを許可します。

「private」は、同じクラス内からのアクセスのみを許可します。

「protected」は、同じクラスかサブクラス（継承先のクラス）からのアクセスのみ許可します。

これらを一覧にまとめると、以下のようになります。

表 3 アクセス修飾子別のアクセス可否

呼び出し元のクラス	public	指定なし	private	pretected
同クラス	○	○	○	○
同パッケージ内サブクラス	○	○	×	○
同パッケージ内非サブクラス	○	○	×	×
他パッケージ内サブクラス	○	×	×	○
他パッケージ内非サブクラス	○	×	×	×

[9] 予約語とは、プログラミング言語の仕様で定められた変数名や関数名として定義できない単語のことを言います。プログラミング言語の一部として定義されている命令語は、同名のユーザー定義変数が存在するとソースコードを正しく解釈できなくなるため、あらかじめ「予約」されていてプログラマが自由に使えないようになっています。

4.5.3 クラスの定義

Java では、プログラムの最小単位をクラスと呼びます。クラス定義の書式は、以下のとおりです。

```
アクセス修飾子 class クラス名 extends 継承元クラス名 {
    クラスの中身を記述します
}
```

Android アプリケーションでは、Activity クラスが、Android アプリケーションの土台となるクラスであり、すべての Android アプリケーションは、このクラスを継承しています。

継承とは、Java のようなオブジェクト指向プログラミングにおいて、既に定義されているクラスをもとに、拡張や変更を加えた新しいクラスを定義することを言います。元になるクラスをスーパークラスあるいは基底クラス、基本クラスなどと呼び、新たに定義されたクラスをサブクラスあるいは派生クラスと呼びます。サブクラスでは、スーパークラスの性質は、すべてサブクラスに受け継がれ、サブクラスでは、スーパークラスとの違いを定義するだけで良いものとなります。

ここで、MiyagiQuest クラスの定義を見てみましょう。

```
public class MiyagiQuest extends Activity {
  :
}
```

このように、MiyagiQuest クラスは、Activity クラスのサブクラスであることが分かります。

4.5.4 メソッドの定義

クラスの中には、そのクラスで行う処理を記述します。Java では、クラスで行う処理のことをメソッドと呼びます。そのクラスで何か処理を行わせたい場合には、このメソッドを呼び出します。

単純な Android アプリケーションのプログラムの場合には、onCreate という名前のメソッドが 1 つだけ存在しています（すごい簡単なプログラムの場合ですよ）。Android アプリケーションが起動されると、この onCreate メソッドが呼び出され、このメソッド内に記述してある処理を実行します。

MiyagiQuest では、以下のように定義されています。

```
@Override
public void onCreate(Bundle savedInstanceState) {

    //    スーパークラス呼び出し
    super.onCreate(savedInstanceState);

    //    ビュー生成
    view = new MiyagiQuestView(getApplication(), this);
```

```
    // 端末のディスプレイ情報取得
    WindowManager windowManager = getWindowManager();
    Display display = windowManager.getDefaultDisplay();

    DisplayMetrics displayMetrics = new DisplayMetrics();
    display.getMetrics(displayMetrics);

    //  モーションイベントを利用します。
    mGestureDetector = new GestureDetector(this);

    //    ビューの割り当て
    setContentView(view);
}
```

onCreate メソッドは、スーパークラス（Activity クラス）でも定義されているメソッドです。

サンプルプログラムでは、そのメソッドをオーバーライドして定義しています。オーバーライドとは、クラスを継承した元になっているスーパークラスで定義されているメソッドを継承したサブクラスを同じメソッド名（および同じ引数）で置き換えることが出来ます。つまり、メソッドを上書きするイメージです。これを、メソッドの**オーバーライド**と言います。

オーバーライドを行った場合には、上書きされるため、サブクラスのメソッドしか呼び出されることがありません。このため、super.onCreate メソッドを呼び出し、親クラスの onCreate メソッドを明示的に呼び出して処理を行っています。その後、Android 端末の実画面上へ表示するためのビューを作成し、そのビューを指定しています。（ビューの設定については、次の「4.5.5 ビューの設定」で説明します）

さて、ここで、メソッドの定義を見てみましょう。

```
アクセス修飾子 戻り値の型 メソッド名(引数の型 引数の名前, ...) {
    メソッドの処理
    return 戻り値
}
```

メソッドは、処理を実行した結果として戻り値を返します。

どの値を戻り値として返すのかは、メソッド内の return 文に記述します。戻り値が無い場合には、戻り値の型に void と記述します。

4.5.5 ビューの設定

実画面に表示するビューを指定するには、Activity クラスの setContentView メソッドを使用します。サンプルプログラムでは、MiyagiQuestView クラスのオブジェクトを表示するビューとして使用しています。

以下に、setContentView メソッドを記載します。

- Acvitity クラス
  void setContentView(View view)

説明：実画面に表示するビューを指定します。
引数：view　ビュー

4.5.6 アノテーション

JDK1.5 から**アノテーション（annotation：注釈）**という機能が追加されました。
Java アノテーションの特徴としては、以下のようなものがあります。

a). アノテーションとは、その名の通り注釈のことを指します。
クラスやメソッドなどに付与された関連情報（メタデータ[10]）のことを言います。
b). アノテーションは付加情報ですので、その有無は、ソースコードに書かれているプログラムの動作には影響を与えません。
c). package, class, method, field などに付加することが出来ます。
d). コンパイルされた class ファイルの中にもアノテーションは残ります。

サンプルプログラムの中には、いくつかのアノテーションが記載されています。以下に、使用されているアノテーションの説明を記載します。

○ @Override
スーパークラスのメソッドをオーバーライドしていることを示すアノテーションです。オーバーライドしていない場合、コンパイラが警告してくれるようになります。このアノテーションを記載することによって、ミスを少なくすることが出来ます。

○ @SuppressWarnings
指定した警告メッセージの表示を抑制することが出来るアノテーションです。設定出来るパラメータには、以下のようなものがあります。

表 4 @SuppressWarnings への引数一覧

引数	内容
deprecation	使用すべきではないクラスやメソッドを使用したときの警告を無視します。
fallthrough	switch ブロックに break などが無い場合の警告を無視します。
finally	finally ブロックは正常に完成出来ないときの警告を無視します。
path	コンパイルオプションの classpath や sourcepath において、パスが存在しない場合の警告を無視します。
serial	直列化可能なクラス[11]に、serialVersionUID が未定義のときの警告を無視します。
unchecked	安全でない型の階層化などの場合の警告を無視します。
unused	使用されていない変数やメソッドが存在した場合の警告を無視します。
all	すべての警告を無視します。

サンプルプログラム内では定義されていませんが、その他の標準のアノテーションには、以下のアノテーションがあります。

[10] メタデータとは、データについてのデータのことを指します。あるデータそのものではなく、そのデータに関連する情報のことです。データの作成日時や作成者、データ形式、タイトル、注釈などがあります。
[11] 特定のデータを保存することであり、クラス等のデータをファイルなどに保存することを直列化と言います。また、保存することが出来るクラスを直列化可能なクラスと呼びます。直接化のことをシリアライズとも言います。

○ @Deprecated
推奨されないクラスやメソッドであることを示します。

4.5.7 Activity のライフサイクル

Activity とは、Android アプリケーションの画面に相当します。ボタンやリストが配置された画面や Web ページが表示されている画面、3D グラフィックスが表示されている画面などは、それぞれが Activity となります。Activity は、さまざまな表示を行ったり、ユーザーと対話したり、イベントを受け取ったりすることが出来ます。ここで、Activity のライフサイクル[12]を見てみましょう。

Activity のライフサイクルは、以下のようになっています。

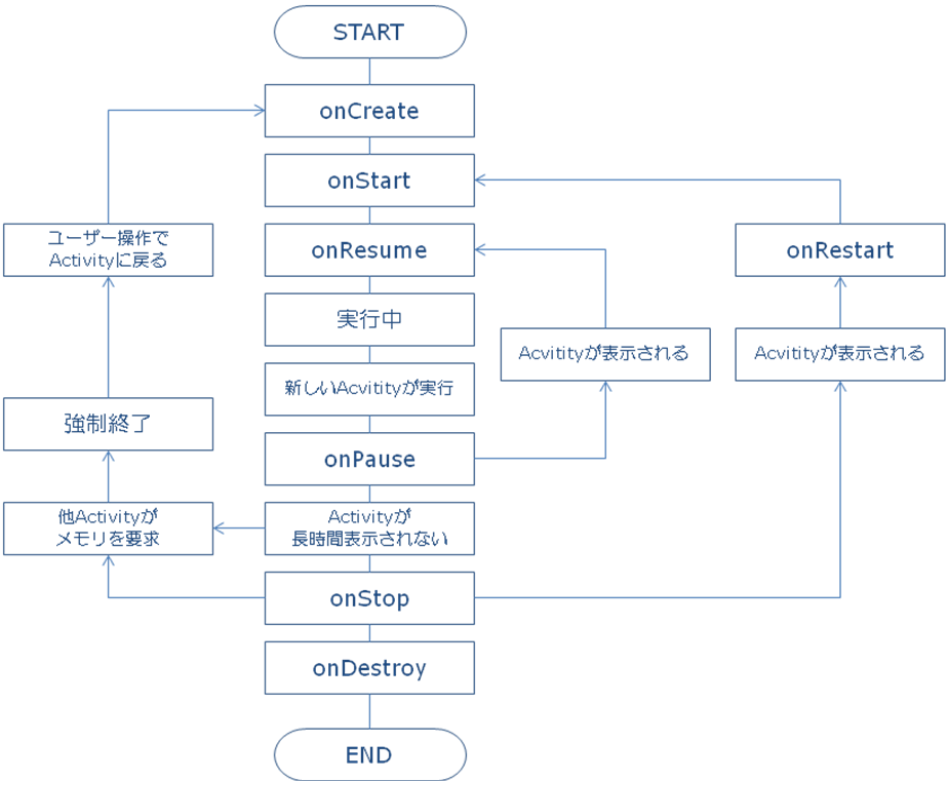


図 13 Activity のライフサイクル

「図 13 Activity のライフサイクル」にも記載されていますが、Activity に関わるメソッドには、以下のようなメソッドがあります。

[12] 生まれてから死ぬまでの流れを表す用語で、ソフトウェアライフサイクルと言えば、ソフトウェアが構想され、開発、運用・保守、廃棄されるまでの流れを指しています。ここでは、Activity が、作られてから破棄されるまでの流れを指します。

表 5 Activity 関連のメソッド

Method	機能
onCreate	Activity 生成時に呼ばれます。
onStart	Activity の表示が開始されたときに呼ばれます。
onResume	ユーザーの操作受付を開始したときに呼ばれます。（一時停止からの再開）
onPause	ユーザーの操作受付を中断したときに呼ばれます。（一時停止）
onStop	Activity の表示を中断したときに呼ばれます。
onRestart	停止状態からの表示再開に遷移するときに呼ばれます。
onDestroy	Activity が破棄されたときに呼ばれます。

他の Activity に遷移しても、明示的に Activity を終了していない場合は、起動し続けます。他の Activity がメモリを要求してきたときに初めて Activity は終了します。

サンプルプログラムの中では、これらのメソッドのうち、onStart 以外の onCreate、onResume、onPause、onStop、onRestart、onDestory のメソッドを利用しています。

4.5.8 定期処理ハンドラ作成

Android では、任意の時間に何か処理を行いたい場合に、Handler クラスを使用します。

サンプルプログラムでは、Handler クラスを継承した内部クラスの TickHandler というクラスを利用しています。ちょっとコードを見てみましょう。

```
/**
 * 定期処理ハンドラクラス Handlerを継承して、作成する。
 */
public class TickHandler extends Handler {
    // ハンドルメッセージ
    @Override
    public void handleMessage(Message msg) {
        onTick();
    }

    //スリープ
    public void sleep(long delayMills) {
        removeMessages(0);
        sendMessageDelayed(obtainMessage(0),delayMills);
    }
}
```

サンプルプログラムでは、上記のように、sleep メソッドと handleMessage メソッドを持つ TickHandler クラスを作成しています。

sleep メソッドは、指定ミリ秒後に handleMessage メソッドを実行するためのメソッドです。

内部的には、removeMessages メソッドでメッセージの初期化を行い、sendMessageDelayed メソッドで指定ミリ秒後にメッセージを送信するように記述しています。

handleMessage メソッドでは、定期的に呼び出されたときに、MiyagiQuest クラスの onTick メソッドを呼び出すようになっています。

○ Handler クラス
void removeMessages(int what)
説明：メッセージの削除
引数：what　メッセージ ID

○ Handler クラス
boolean sendMessageDelayed(Message msg, long delayMillis)
説明：メッセージの送信
引数：msg　メッセージ
　　delay メッセージを送信するまでの時間（ミリ秒）
戻り値：True – 成功、Flase – 失敗

メッセージは、obtainMessage メソッドで取得します。

○ Handler クラス
Message obtainMessage(int what)
説明：メッセージの取得
引数：what　メッセージ ID
戻り値：メッセージ

handleMessage メソッドは、Handler クラスが持つメソッドで、メッセージを受信したときに実行する処理を記述します。今回は、MiyagiQuestView オブジェクトの invalidate メソッドと、TickHandler オブジェクトの sleep メソッドを呼んでいます。

invalidate メソッドは、View クラスのメソッドで再描画を行います。プログラムの内部的には、onDraw メソッドが呼ばれるようになります。

○ View クラス
void invalidate()
説明：ビューの再描画

sleep メソッドは、MiyagiQuestView クラスの sleep メンバ変数により設定しています。（メンバ変数は、定数として 100（ミリ秒）が設定されています）

○ Handler クラス
void handleMessage(Message msg)
説明：メッセージの受信
引数：msg　メッセージ

4.5.9 定期処理ハンドラ再開

定期ハンドラの再開時には、onResume メソッドが呼び出されます。このメソッドでは、TickHandler を生成後、sleep メソッドを呼び出すことにより定期処理を再開しています。

4.5.10 定期処理ハンドラ停止

定期ハンドラの停止時には、onPause メソッドが呼び出されます。このメソッドでは、tickHandler メンバ変数に null を代入することによって、handleMessage メソッド内で sleep メソッドが呼ばれ

なくなり、定期処理ハンドラが停止します。

4.5.11 フォーカス

Android 端末のキーを押下したときに、キーイベントが通知されるのは、フォーカスを持っているビューのみとなっています。そこで起動時に、フォーカスを有効にしておかなければいけません。

フォーカスの有効には、View クラスの setFocusable メソッドのほかに、タッチモードを有効にする setFocusableTouchMode メソッドもありますが、サンプルプログラムでは、setFocusable のみを有効にしています。

- View クラス
  void setFocusable(Boolean focusable)
  説明：フォーカスの指定
  引数：focusable　有効／無効フラグ

4.5.12 キーイベント処理

※サンプルプログラムには実装されていません。

View クラスの onKeyDown メソッド、onKeyUp メソッドをオーバーライドすることによってキーダウンイベント、キーアップイベントを、それぞれ取得することが出来ます。これらのメソッドは、Android 端末のキーが押下されたり、話されたりするたびに、これらのメソッドが呼ばれます。

- View クラス
  boolean onKeyDown(int keycode, KeyEvent event)
  説明：キーダウンイベント発生時に呼ばれる
  引数：keycode　キーコード
  　　event　　キーイベント
  戻り値：処理が完了したかどうかを返します。

- View クラス
  boolean onKeyUp(int keycode, KeyEvent event)
  説明：キーアップイベント発生時に呼ばれる
  引数：keycode　キーコード
  　　event　　キーイベント
  戻り値：処理が完了したかどうかを返します。

キーコードには、Android 端末のそれぞれのキーに割り当てられている ID が渡されます。

この ID は、KeyEvent クラスの「KEYCODE_」から始まる定数として定義されています。つまり、押下されたボタンの種類によって、このキーコードが渡されます。例えば、Android 端末の 1 キーを押下された場合には、KEYCODE_1 が渡されるといった具合です。（定数の一覧は、ここでは取り扱いませんが、KeyEvent.class を、自分で調べてみましょう）

戻り値は、処理が完了したかどうかを返し、次の Listener[13]に処理をまかせるときは「false」を指定します。

[13] リスナーとは、ボタンをクリックしたり、メニューを実行したり、ウィンドウをリサイズしたり、マウスを動かしたり、キーを押したりなどのイベント監視し、イベント発生時に対応するアクションを実行するオブジェクトのことを指します。

4.5.13 タッチイベント処理

View クラスの onTouchEvent メソッドをオーバーライドすることによって、タッチイベントを取得することが出来ます。画面上のタッチや移動するたびに、このメソッドが呼ばれます。

- View クラス
  boolean onTouchEvent(MotionEvent event)
  説明：タッチイベント発生時に呼ばれる
  引数：event　　タッチイベント
  戻り値：処理が完了したかどうかを返します。

- View クラス
  boolean onFling (MotionEvent e1, MotionEvent e2, float velocity, float velocityY)
  説明：スクリーン上で指をスライドさせたときに呼ばれる
  引数：e1　　タッチイベント 1
  　　e2　　タッチイベント 2
  　　velocityX　　X 方向移動量
  　　velocity　　Y 方向移動量
  戻り値：処理が完了したかどうかを返します。

- View クラス
  boolean onSingleTapUp (MotionEvent e)
  説明：タップしたときに呼ばれる
  引数：e1　　タッチイベント
  戻り値：処理が完了したかどうかを返します。

これらのタッチイベントによって、キー操作と同じような動きをさせます。

onFling メソッドは、Android 端末のタッチパネル上を操作されたときに呼び出され、指定されているベクトル[14]から、どのようにタッチパネル上を操作されたのかを判断し、その判断結果から処理を行っていきます。まずは、onFling メソッドの流れを、以下に示します。

[14] ベクトルとは、向きを持った量のことを言います。逆に向きを持たない量のことをスカラーと言います。

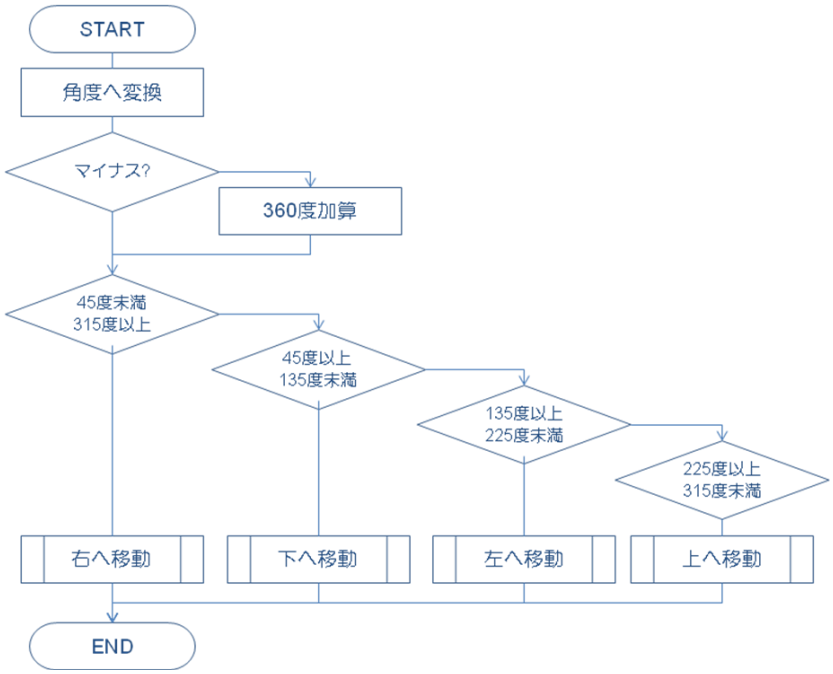


図 14 addFling メソッド流れ

addFling メソッドでは、最初に、指定されているベクトルから角度へと変換します。このとき、Math クラスの atan2 メソッドおよび toDegrees メソッドを利用します。

- Math クラス
  double atan2 (double x, double y)
  説明：アークタンジェントの計算
  引数：x　X 方向ベクトル値
  　　y　Y 方向ベクトル値
  戻り値：ラジアン角を返します。

- Math クラス
  double toDegrees (double angrad)
  説明：ラジアン角から角度への変換
  引数：angrad　ラジアン角
  戻り値：角度を返します。

さて、ここで、アークタンジェント（atan）という言葉が出てきましたので、ちょっと説明しておきたいと思います。このアークタンジェント（atna）以外にも、アークサイン（asin）やアークコサイン（acos）といった言葉もあります。

本書を読んでいる皆さんは、サイン（sin）、コサイン（cos）、タンジェント（tan）というのは、すでに聞いたことがあるでしょう。このサイン、コサイン、タンジェントというのは、角度に対する関数のことです。例えば、tan 60°というのは、√3 ですが、このように角度を与えると数値を求めることが出来ます。逆に、アークタンジェントなどは、数値に対する関数のことです。atan √3 というと、60°という角度を求めることが出来ます。このように、タンジェントとアークタンジェントの関係は、逆関数という関係があります。逆関数というのは、原因と結果が逆になるような関数のことを言います。

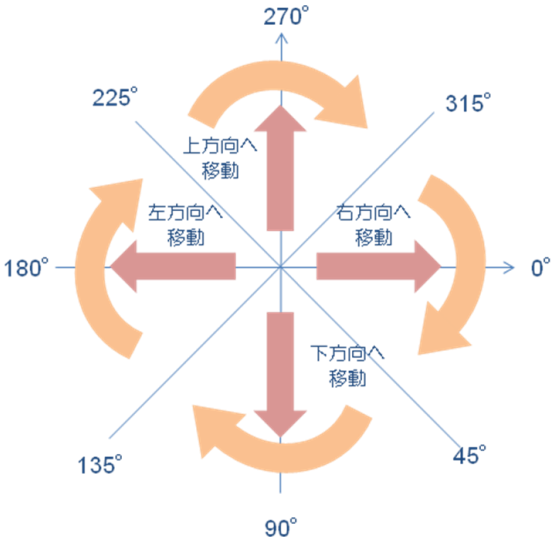


図 15 キャラクター移動処理

このようにして角度が求められます。

求められた角度は、マイナス値となっていることがありますので、マイナス値の場合には、360°を加算し、プラス値となるようにしておきます。（このようにすることで、不要なチェックが要りません）

この求められた角度から、移動処理を行います。

「図 15 キャラクター移動処理」は、角度と移動処理の方法の関係について示しています。

Canvas 上では、3 時方向が 0°となり、時計回りにぐるっと一周回ることになります。

そこで、「図 15 キャラクター移動処理」に記載されているとおり、315°以上か、45°未満の場合には、ブロックを右方向へ移動させ、135°以上 225°未満の場合には、左方向へ移動させます。

45°以上 135°未満の場合には、下方向へ移動させます。225°以上 315°未満の場合には、上方向へ移動させます。

4.5.14 画像ファイルの読み込み

※MiyagiQuestView.java に記述されています。

リソースの画像ファイルを読み込むには、BitmapFactory クラスの decodeResource メソッドを使用します。

○ BitmapFactory クラス
static Bitmap decodeResource(Resources res, int id)
説明：リソース画像ファイルの読み込み
引数：res　リソースオブジェクト
　　id　リソース ID
戻り値：Bitmap オブジェクト

更に、リソースオブジェクトは、Context クラスの getResources メソッドで取得します。

○ Context クラス
Resources getResources()
説明：リソースオブジェクトの取得
引数：なし

戻り値：リソースオブジェクト

リソース ID は、res/drowable フォルダに配置したファイルを、以下のような書式で指定します。

```
R.drawable.ファイル名（拡張子なし）
```

例えば、サンプルプログラムでは、タイトルの画像に「title.gif」というファイルを利用していますが、この場合のリソース ID は、「R.drawable.title」となります。

戻り値としては、Bitmap オブジェクトが返ってきます。

サンプルプログラムでは、MiyagiQuestView クラスにおいて、初めにタイトル画像、フィールド画像、主人公画像をそれぞれ取得しています。ちょっと該当箇所を見てみましょう。

```
//マップ画像の読み込み
//フィールド名と、実際値が異なるので、reflectionを利用。
//reflection とは、フィールド名などから、その値を取得するための機能。
try {
        //どのクラスから取得するかを指定
        Class cls = Class.forName("com.cgene.game.R$drawable");
        for (int i = 1, idx = 0; i <= 7; i ++) {
                for (int j = 1; j <= 4; j ++) {
                        String fldName = "field_" + i + j;
                        fldImage[idx]  =  BitmapFactory.decodeResource(r,  getResouceInt(cls,
fldName));
                        idx ++;
                }
        }

        //TODO NPCキャラ（マップチップと同じ扱い）読み込み
        fldImage[28] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_76" ));
        fldImage[29] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_77" ));
        fldImage[30] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_78" ));
        fldImage[31] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_79" ));
        fldImage[32] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_80" ));
        fldImage[33] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_81" ));
        fldImage[34] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_82" ));
        fldImage[35] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_83" ));

        //上記の処理は、fldImage[idx] = BitmapFactory.decodeResource(r, R.drawable.filed_11); とし
ても同じだが、
        //R.drawable.filed_11 を、28個も記述するのが、面倒なので、リフレクションで簡易化している。

        //プレイヤー画像の読み込み。フィールド名と、実際値が異なるので、reflectionを利用。
        char c[] = {'b', 'l', 'r', 'u'};
        for (int i = 0; i < c.length; i ++) {
                for (int j = 0; j < 2; j ++) {
                        String fldName = "hero_" + c[i] + (j+1);
                        playerImage[i][j] = BitmapFactory.decodeResource(r, getResouceInt(cls,
fldName));
                }
        }

} catch (Exception e) {
        android.util.Log.e(DebugLog.TAG, e.toString());
```

```
}
```

リフレクションという技術を利用して、各リソースを取得しています。リフレクションとは、プログラムの実行過程で、プログラム自身の構造を読み取ったり、書き換えたりする技術のことを言います。

例えば、フィールド画像の取得の部分では、インデックスの値を利用しながら、すべての画像ファイルを取得しています。フィールドの画像は、Field_11、Field_12、Field_13、Field_14、Field_21、….と続き、Field_74 までの合計 28 枚（7×4）の画像ファイルがあり、これを、for 文を用いて繰り返し名前を生成して、その後、decodeResource メソッドを利用しています。（もちろん、画像ファイルを、Field_1 から Field_28 として利用することも可能です）

ここで、for 文などのループ処理について、説明しておきましょう。

ループ処理は、条件によって処理を繰り返すときに利用します。for 文以外にも、while 文などがあります。ここでは、その 2 つの方法について説明します。

- for 文

  for 文は、ループの回数が決まっているときに使用する命令です。
  for 文の書式は、以下のとおりです。

```
for(初期化式; 条件式; 増減式) {
    処理
}
```

  Field_1 から Field_28 までの文字列を生成し出力するには、次のように記述します。

```
for(int i = 1; i <= 28; i++) {
    String strName = “Field_” + i;
    printf(“%s”, strName);
}
```

  まず、最初に、「int i = 1」で、変数 i に 1 を代入しています。これを for 文における初期化式と呼びます。次の「i <= 28」では、ループを繰り返す条件を指定しており、これは、i は 28 以下であるという意味になります。つまり、28 を超えるまで、この for 文を繰り返し実行することになります。最後の「i++」は、i に 1 を加算（インクリメント）します。これは、ループの最後に毎回実行されることになります。

- while 文

  while 文は、条件式が真（True）となるまで処理を行う命令です。
  while 文の書式は、以下のとおりです。

```
while(条件式) {
    処理
}
```

  先ほどの for 文と同じ処理を、while 文で記述すると、以下のように記述します。

```
int i = 1;
while(i <= 28) {
    String strName = “Field_” + i;
    printf(“%s”, strName);
    i++;
}
```

4.5.15 サウンドファイルの再生

リソースのサウンドファイルを再生するためには、MediaPlayer クラスを使用します。
この MediaPlayer で利用できるファイルフォーマットは、以下のとおりです。

表 6 MediaPlayer で再生可能なフォーマット

拡張子	説明
.3gp	3GPP で記録された動画ファイルで、MP4 形式がベースとなっています。
.mp4	MPEG4 形式で圧縮された動画ファイルです。国際標準化機構（ISO）の下部組織である MPEG（Motion Picture Experts Group）により制定されたものです。
.m4a	AAC（Advanced Audio Coding）規格で圧縮された音声ファイルです。AAC 規格は、MP3 規格が進化したものであると言える。ただし、MP3 が MPEG1 の拡張形式である（MPEG1 Audio Layer-3）のに対し、AAC は、MPEG4 形式がベースとなっています。
.mp3	MP3 形式で圧縮されたファイルのことです。厳密には、MPEG1 Audio Layer-3、MPEG2.5 Audio Layer-3 という規格が存在するが、一般的に MP3 と呼ばれる場合には、MPEG1 Audio Layer-3 形式で圧縮された音声ファイルを指す。音楽 CD に遜色のない音質を保ちながら、WAV 形式のファイルに比べてデータサイズが約 12 分の 1 になります。
.ogg	Ogg Project によって開発された音声圧縮方式（Ogg Vorbis）で用いられるファイルです。ogg 形式は、MP3 形式のライセンス権に関する問題を避けるために開発された規格であるとされています。音質は、MP3 ファイルよりも優れており、規格はオープンソースでロイヤリティフリーとなっています。
.wav	WAVE 形式で保存された音声ファイルのことです。Windows の標準的な音声ファイル形式となっています。
.mid	シンセサイザーなどの電子楽器をパソコンと接続する方式である MIDI（Musical Instrument Digital Interface）規格で作成されたデータファイルのことです。

サウンドを再生するためには、まずメディアプレイヤーを生成します。メディアプレイヤーを生成するには、MediaPlayer クラスの create メソッドを利用します。create メソッドは、指定方法により、以下の 3 つのメソッドがあります。

○ MediaPlayer クラス
static MediaPlayer create(Context context, int resid)
説明：メディアプレイヤーの生成
引数：context　　コンテキスト
　　resid　　リソース ID
戻り値：MediaPlayer オブジェクト

○ MediaPlayer クラス
static MediaPlayer create(Context context, Uri uri, SurfaceHolder holder)
説明：メディアプレイヤーの生成
引数：context　　コンテキスト
　　uri　　データソース取得用 URI
　　resid　　メディア表示のために使用する SurfaceHolder
戻り値：MediaPlayer オブジェクト

○ MediaPlayer クラス
static MediaPlayer create(Context context, Uri uri)
説明：メディアプレイヤーの生成
引数：context　　コンテキスト
　　uri　　データソース取得用 URI

戻り値：MediaPlayer オブジェクト

/res/raw フォルダにあるリソースを利用する場合、リソース ID は、「R.raw.ファイル名」（拡張子は付きません）の書式で指定します。

その後、prepare メソッドを呼び出し、再生の準備を行います。

○ MediaPlayer クラス
void prepare()
説明：サウンド再生の準備
引数：なし
戻り値：なし

サウンドの再生には、start メソッドを呼び出します。サンプルプログラムでは使用していませんが、再生位置を一定にするために、seekTo メソッドを利用し、再生位置を先頭に戻しても良いかもしれません。

○ MediaPlayer クラス
void seekTo(int msec)
説明：サウンド再生位置の指定
引数：msec　　サウンド再生位置（ミリ秒）
戻り値：なし

○ MediaPlayer クラス
void start()
説明：サウンドの再生開始
引数：なし
戻り値：なし

「4.3 サウンドファイルの準備」でも少し触れていますが、BGM の場合、該当する処理が継続している場合、常に再生を行っておく必要があります。このため、ループ再生の指定も行っておく必要があります。ループ再生の指示は、setLooping メソッドを利用します。

○ MediaPlayer クラス
void setLooping(boolean looping)
説明：ループ再生指定
引数：looping　　True – ループ再生する、Flase – ループ再生しない
戻り値：なし

さて、基本的には、ここまでの説明で、サウンドファイルの再生を行うことは出来ることになります。res/raw フォルダ配下に含めているサウンドファイルであれば、これらのメソッドを組み合わせることにより、再生することまでは出来ると思います。

しかし、このサンプルプログラムの中では、BGM を、外部のサーバからプログラム内に読み込み、読み込み後にメディアプレイヤーに設定することで再生する方法をとっています。

以下に、この処理のプログラムを抜粋してみましょう。

※MyUtil.java に記述

```
//    指定URL接続
URLConnection connection =  new URL(BGM_Connection_URL[BGM_STATUS]).openConnection();
InputStream is = connection.getInputStream();
File file = new File(activity.getCacheDir(), "tmp");
FileOutputStream fos = new FileOutputStream(file);
byte buf[] = new byte[1024];

do {
    int numread = is.read(buf);
    if (numread == -1) {
        break;
    }
    fos.write(buf, 0, numread);
} while (true);
fos.flush();
fos.close();

//    MediaPlayerインスタンス生成
mp_Bgm = new MediaPlayer();
mp_Bgm.setDataSource(new FileInputStream(file).getFD());
mp_Bgm.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer mp_Bgm) {
        try {
            mp_Bgm.start();
            mp_Bgm.setLooping(true);
        } catch (Exception e) {
            Log.d("Error", e.getMessage());
            if( mp_Bgm != null ) {
                mp_Bgm.stop();
                mp_Bgm.release();
                mp_Bgm = null;
            }
        }
    }
});
mp_Bgm.prepareAsync();
```

BGM は、タイトル画像表示中、マップ歩行中、それと戦闘中の全部で 3 種類あります。その URL は、BGM_Connection_URL に入っており、状態（BGM_STATUS）によって切り分けるようになっています。サンプルプログラムの中では、状態によって毎回読み込むようになっていますが、メモリに余裕があるのであれば、最初に読み込むようにしても良いでしょうね。

読み込まれたファイルは、setDataSource メソッドにより MediaPlayer に設定され、その後、start メソッドによって再生されます。BGM は、サウンドファイルをずっと再生させておく必要があるため、setLooping メソッドによりループ再生されるようにしています。

- MediaPlayer クラス
  void setDataSource(String path)
  説明：データソースの設定

引数：path　　ファイルパス名あるいは URL
戻り値：なし

○ MediaPlayer クラス
void setDataSource(FileDescriptor fd, long offset, long length)
説明：データソースの設定
引数：fd　　ファイルディスクリプター[15]
　　offset　　オフセット
　　length　　長さ
戻り値：なし

○ MediaPlayer クラス
void setDataSource(FileDescriptor fd)
説明：データソースの設定
引数：fd　　ファイルディスクリプター[16]
戻り値：なし

4.5.16 ステータスによる処理の分岐

RPG に限らず、アプリケーションを作成する際に、良く利用される方法が、ステータスによって処理を分岐させる方法があります。

サンプルプログラムでも、このステータスを使用し、さまざまな処理を行わせています。
以下に、サンプルプログラムで使用しているステータスを記述します。

表 7 サンプルプログラム内ステータス一覧

Status	内容
STATUS_WAIT	何もしない
STATUS_START	開始ステータス（起動～タイトル表示までを行います）
STATUS_TITLE	タイトル表示ステータス（タイトル画面の描画を行います）
STATUS_ROLL	テキストロールステータス（ゲーム開始時の説明文を描画し、下から上へロールさせます）
STATUS_MAP	マップ描画ステータス（フィールドマップを描画します）
STATUS_MENU	メニュー描画ステータス（メニューを描画します。ただし、サンプルプログラムでは未使用です）
STATUS_BATTLE	戦闘ステータス（戦闘シーンを描画します）
STATUS_GAMEOVER	ゲームオーバーステータス（ゲームオーバー画面を描画します）
STATUS_ERROR	エラーステータス（エラー画面を描画します）

ステータスは、MiyagiQuestView クラスの status メンバ変数によって、現在のステータスを保持しています。サンプルプログラムにおけるステータスの遷移を、「図 16　ステータス遷移」に示します。

[15] プログラムがアクセスするファイルや標準入出力などを OS が識別するために用いる識別子。
[16] プログラムがアクセスするファイルや標準入出力などを OS が識別するために用いる識別子。

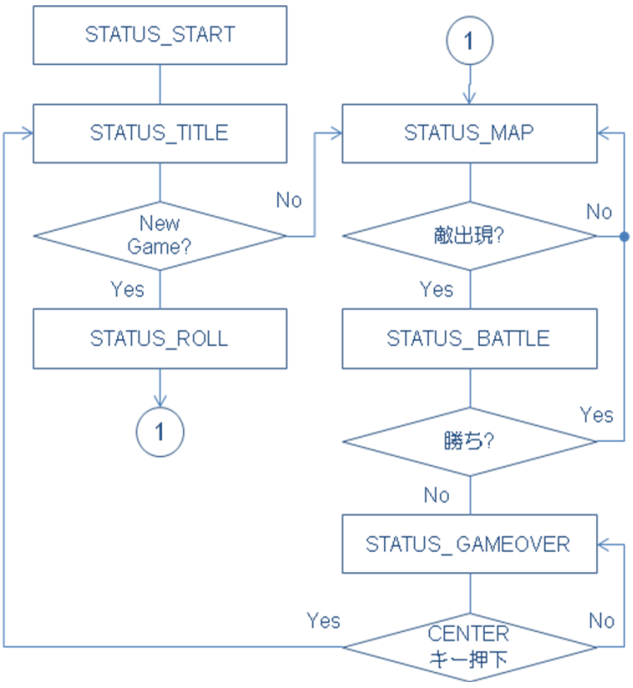


図 16 ステータス遷移

初期化終了後、STATUS_START に設定されており、その後、タイトルの描画（STATUS_TITLE）が行われます。STATUS_TITLE では、タイトル画像の描画以外にも、選択メニューとして「New Game」と「Continue」が表示されており、ユーザーは、そのどちらかを選択することになります。ここで、ユーザーが「New Game」を選択した場合には、テキストのロール表示（STATUS_ROLL）に遷移します。ユーザーが「Continue」を選択した場合には、データが保存されているかをチェックし、データが保存されている場合には、そのまま STATUS_MAP へと遷移します。データが存在していない場合には、「New Game」が選択されたものとしてテキストロール表示（STATUS_ROLL）へと遷移します。

STATUS_ROLL は、リソースファイル内で定義されているテキストを描画し、画面下から画面上方向へロールさせていきます。その後、ステータスを STATUS_MAP へと遷移させます。

STATUS_MAP は、主人公がマップ上を移動するステータスであり、この状態が、サンプルプログラムにおける通常の状態です。マップ移動中には、ランダムに敵が出現する場合があり、敵が出現すると、ステータスを STATUS_BATTLE へと遷移させます。敵が出現しなければ、STATUS_MAP のまま、推移していきます。

STATUS_BATTLE は、敵キャラクターとの戦闘シーンを描画しますが、敵キャラクターとの戦闘に負けてしまった場合には、STATUS_GAMEOVER となります。敵キャラクターとの戦闘に勝った場合には、STATUS_MAP へと遷移し、再度、マップ描画の処理が始まります。

STATUS_GAMEOVER では、CENTER キーが押下されるまで、ゲームオーバー画面が表示されます。CENTER キー押下により、STATUS_TITLE のステータスへと遷移し、再度、タイトル表示へと戻っていきます。

このように、ステータスを利用することによって、ゲームを進めていきます。もちろん、これらのステータス以外にも、プログラムにステータスを追加することによって、別の処理を行わせるようにすることも可能です。例えば、STATUS_BATTLE と同様に、ランダムに敵以外のキャラクターを登場させる（例：旅人に出会う、流浪の商人に出会うなどなど）ようにすることも可能です。

サンプルプログラムでは、「4.5.8 定期処理ハンドラ作成」にも記載していますが、定期的に、MiyagiQuestView クラスの onDraw メソッドが呼び出されます。

このメソッドの中で、status メンバ変数の値を利用し、処理を分岐させています。

```
/*
 * 描写処理
 * @param    canvas    描写キャンバスクラス
 */
protected void onDraw(Canvas canvas) {
    if (status == STATUS_WAIT) return;

    super.onDraw(canvas);
    // 背景色の設定
    canvas.drawColor(Color.BLUE);
    // 描画オブジェクトの生成
    Paint paint = new Paint();
    paint.setAntiAlias(true);
    paint.setStyle(Paint.Style.FILL);

    try {
        //    ステータスごとに、描写を振り分ける
        switch (status) {
        case STATUS_TITLE:
            MyUtil.PlaySound_BGM(activity, MyUtil.BGM_TITLE);
            title(canvas, paint);
            break;
        case STATUS_ROLL:
            roll(canvas, paint);
            break;
        case STATUS_MAP:
            MyUtil.PlaySound_BGM(activity, MyUtil.BGM_MAP);
            map(canvas, paint);
            break;
        case STATUS_MENU:
            menu(canvas, paint);
            break;
        case STATUS_BATTLE:
            MyUtil.PlaySound_BGM(activity, MyUtil.BGM_BATTLE);
            battle(canvas, paint);
            break;
        case STATUS_GAMEOVER:
            gameover(canvas, paint);
            break;
        case STATUS_ERROR:
            error(canvas, paint);
            break;
        default:
            break;
        }
    } catch (Exception e) {
        android.util.Log.e(DebugLog.TAG, e.toString());
```

```
    }
    count++;
}
```

ここで、ちょっと View クラスの onDraw メソッドについて解説しましょう。

View クラスを継承したクラスは、onDraw メソッドをオーバーライドすることによって、さまざまな描画処理を行うことが出来ます。onDraw メソッドは、アプリ起動時や再描画が必要なときに呼び出されます。このメソッドへ引数として渡される Canvas クラスのオブジェクトを操作することによって、画面上に絵や文字列などを描画させることが出来ます。

○ View クラス
void onDraw(Canvas canvas)
説明：描画時に呼び出されます
引数：canvas　キャンバス

背景色の指定は、View クラスの setBackgroundColor メソッドで行います。

○ View クラス
void setBackgroundColor(int color)
説明：背景色を指定します
引数：color　背景色

色には、以下の定数を指定します。

表 8 setBackgroundColor メソッドでの指定定数一覧

定数	意味
Color.BLACK	黒
Color.BLUE	青
Color.CYAN	シアン
Color.DKGRAY	ダークグレー
Color.GRAY	グレー
Color.GREEN	緑
Color.LTGRAY	ライトグレー
Color.MAGENTA	マゼンタ
Color.RED	赤
Color.TRANSPARENT	透明色
Color.WHITE	白
Color.YELLOW	黄

サンプルプログラムでは、「Color.DKGRAY」を指定しています。
続けて、描画オブジェクトの生成を行い、その後、Paint クラスの setAntiAlias メソッドおよび setStyle メソッドを呼び出しています。

○ Paint クラス

void setAntiAlias(Boolean aa)
説明：アンチエイリアスの有効／無効の指定
引数：aa　アンチエイリアスの有効／無効

アンチエイリアスとは、文字やラインを滑らかに見せる処理のことを言います。

○ Paint クラス
void setStyle(Paint.Style style)
説明：描画スタイルの指定
引数：style　描画スタイル

描画スタイルで指定出来る定数には、以下のようなものがあります。

表 9 描画スタイルでの指定定数一覧

定数	意味
Paint.Style.FILL	塗り潰し
Paint.Style.FILL_AND_STROKE	塗り潰し+ライン
Paint.Style.STROKE	ライン

サンプルプログラムでは、塗り潰ししか使わないため、Paint.Style.FILL を指定しています。

4.5.16.1 タイトル描画

ここでは、ステータス「STATUS_TITLE」での処理を見ていきましょう。
ステータス「STATUS_TITLE」での処理は、すべて title メソッド内で行っています。

※MiyagiQuestView.java に記述

```
/**
     * タイトルを描写します。
 * @param canvas    描写キャンバスクラス
 * @param paint     描写ペイントクラス
 */
private void title(Canvas canvas, Paint paint) {
        //タッチイベント処理
        switch (TouchStatus) {
        case TouchStatus_UP:
                titleSelectedIndex--;
                if(Flag_LongTouch != true) TouchStatus = -1;
                break;
        case TouchStatus_DOWN:
                titleSelectedIndex++;
                if(Flag_LongTouch != true) TouchStatus = -1;
                break;
        case TouchStatus_TAP:
                //        選択キーでの処理
                sleep(500);
                TouchStatus = -1;
                if (titleSelectedIndex == 1) {        //Continueが選択された場合
                        //保存データを読み込み、[,]で分割。
                        String[] data = MyUtil.split(load(), ",");
                        //TODO 保存項目数により異なる！！
                        if (data.length < 19) {
                                //データがないか、不正であれば、New Gameへ
                                setStatus(STATUS_ROLL, true);
                        } else {
                                //データがあれば、ステータスをMAPに設定し、初期化）
                                setStatus(STATUS_MAP, true);
                                try {
                                        //        保存データから、各値を初期化
                                        int k = 0;
                                        level = Integer.parseInt(data[k++]);
                                        maxHp = Integer.parseInt(data[k++]);
                                        hp = Integer.parseInt(data[k++]);
                                        maxSp = Integer.parseInt(data[k++]);
                                        sp = Integer.parseInt(data[k++]);
                                        str = Integer.parseInt(data[k++]);
                                        def = Integer.parseInt(data[k++]);
                                        spd = Integer.parseInt(data[k++]);
                                        exp = Integer.parseInt(data[k++]);
                                        nextExp = Integer.parseInt(data[k++]);
                                        map = Integer.parseInt(data[k++]);
                                        playerMapPosX = Integer.parseInt(data[k++]);
                                        playerMapPosY = Integer.parseInt(data[k++]);

                                        //TODO 各種イベントフラグ
                                        FLAG_SENDAI_WEAPON = Integer.parseInt(data[k++]);
                                        FLAG_SENDAI_ARMOR  = Integer.parseInt(data[k++]);
                                        FLAG_SENDAI_SHIELD = Integer.parseInt(data[k++]);
                                        FLAG_ITEM1  = Integer.parseInt(data[k++]);
```

```
                                        FLAG_ITEM2  = Integer.parseInt(data[k++]);
                                        FLAG_ITEM3  = Integer.parseInt(data[k++]);

                                        //初回プレイ時フラグ
                                        startFlag = false;

                                    initMapImage(new Canvas(mapImage));
                                    } catch (Exception e) {
                                        //処理中にエラーが発生したら、New Gameへ
                                        setStatus(STATUS_ROLL, true);
                                    }
                                }
                            } else { //      New Gameが選択されました。
                                setStatus(STATUS_ROLL, true);
                            }
                            //処理は、ここで終了。（キーイベント処理で終了です。）
                            return;
                        }
                        //タイトル選択インデックスの処理
                        if (titleSelectedIndex < 0) titleSelectedIndex = 1;
                        else if (titleSelectedIndex > 2) titleSelectedIndex = 0;

                        //描写処理
                        int x = (displayWidth-titleImage.getWidth())/2;
                        int y = 10;
                        canvas.drawBitmap(titleImage, x, y, paint);

                        paint.setTextSize(42);
                        y += 58 + titleImage.getHeight();

                    paint.setColor(Color.WHITE);
                    canvas.drawText("New Game", 80, y, paint);
                    canvas.drawText("Continue", 80, y + 60, paint);
                    //選択されている側にプレイヤー画像を表示されます。
                    if (titleSelectedIndex == 0) {
                            canvas.drawBitmap(playerImage[2][playerDrawIndex], 20, y-36, paint);
                    } else {
                            canvas.drawBitmap(playerImage[2][playerDrawIndex], 20, y+60-36, paint);
                    }
                    //プレイヤーのアニメーション変更。countで、早すぎないように変更させる。
                    if (count%3 == 0) playerDrawIndex = (playerDrawIndex+1)%2;
}
```

title メソッドでは、最初に TouchStatus メンバ変数を調べます。（キー押下によるキーコードも、このメンバ変数に設定されます）

タイトル画面には、タイトル画像の描画以外にも、「New Game」と「Continue」が表示されています。ユーザーは、このうち、一方を選択するようになっています。現状、どちらが選択されているのかについては、titleSelectedIndex メンバ変数で管理されています。

TouchStatus _UP または TouchStatus _DOWN が選択された場合には、アイコンを移動させるようになっています。

TouchStatus_TAP が選択された場合には、現在、選択されているメニューにより、処理を行うようになっています。この処理が行われた後には、ステータスを STATUS_MAP（Continue の場合）、STATUS_ROLL（New Game あるいは何らかのエラーが発生した場合）へと遷移します。なお、こ

れらの処理が行われた場合には、タイトル画面を描画する必要がないため、このメソッドから復帰するようになっています。

単に上下方向キー押下あるいは画面上で上下方向へタッチし、メニューの選択を変更しただけの場合には、現状設定されている各種情報によって、タイトル画面を描画します。

図 17 タイトル画像

タイトル画像の描画には、Canvas クラスの drawBitmap メソッドを利用して描画を行います。

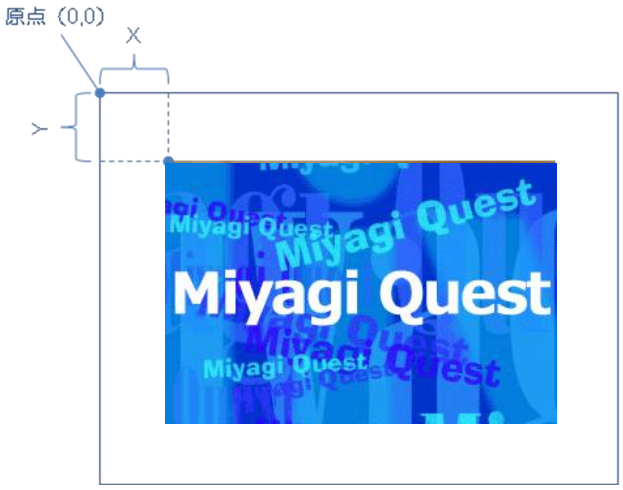


図 18 drawBitmap での描画座標

○ Canvas クラス
void drawBitmap(Bitmap bitmap, int left, int top, Paint paint)

説明：イメージの描画
引数：bitmap　ビットマップオブジェクト
　　left　　X 座標
　　top　　Y 座標
　　paint　　描画オブジェクト

引数となっている X 座標（left）および Y 座標（top）は、イメージの左上の座標となります。

※MiyagiQuestView.java に記述

```
//    描画処理
int x = (displayWidth - titleImage.getWidth()) / 2;
int y = 10;
canvas.drawBitmap(titleImage, x, y, paint);
```

サンプルプログラムでは、X 座標および Y 座標は、画面サイズを考慮した配置位置になっています。

X 座標は、画面幅からタイトル画像幅を引いた値を 2 で割ることから、画像を横方向の中心に配置することを意味しています。同様に、Y 座標は 10 になっているので、画像を画面上端から縦方向で 10 の位置に配置することを意味しています。

X 座標と Y 座標を、このような方法で設定することによって、画面内で適切な位置に画像を配置させることが出来ます。この方法は、画像を配置する際に良く使う方法ですので、是非、覚えておきましょう。

続いて、文字列の描画を行います。

文字列を描画するためには、Canvas クラスの drawText メソッドを使用します。

また、文字列を描画する前に、テキストのサイズ設定（Paint クラスの setTextSize メソッド）と色設定（Paint クラスの setColor メソッド）を使用します。

○ Paint クラス
void setTextSize(float textSize)
説明：文字サイズの指定
引数：textSize　文字サイズ

○ Paint クラス
void setColor(int color)
説明：色の指定
引数：color　色の値

○ Canvas クラス
void drawText(String text, int x, int y, Paint paint)
説明：文字列の描画
引数：text　　文字列
　　x　　描画 X 座標
　　y　　描画 Y 座標
　　paint　描画オブジェクト

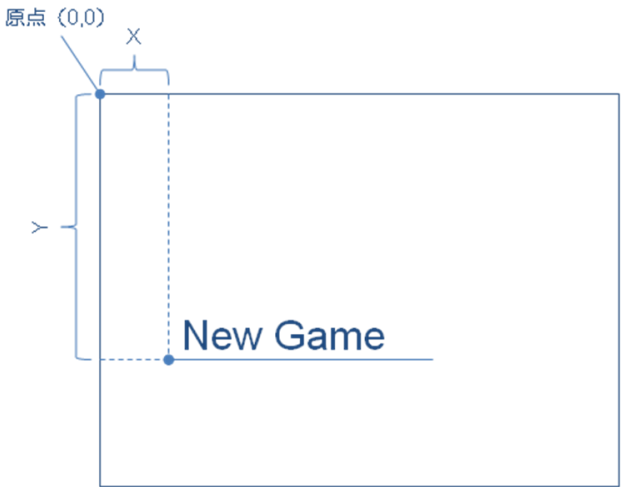


図 19 drawText の描画位置

X 座標は、画面左からのピクセル数、Y 座標は、画面上からのピクセル数を示します。

ただし、注意しなければいけない点として、このメソッドで指定する座標は、文字列のベースラインの左隅の座標となります。文字列の左上ではないことに注意して下さい。

また、端末の種類によっては、「g」や「j」、「q」などの文字については、ベースラインよりも下に表示されることもありますので、何行にも渡って文字列を描画する際には、座標を考えながら設計していかなければいけません。

最後に、titleSelectedIndex メンバ変数で示されている位置に、主人公の画像を描画しています。なお、このときに、アニメーションとして動作することを考慮し、playerDrawIndex メンバ変数によって、次に描画する画像を切り替えています。

4.5.16.2 テキストロールの描画

ここでは、ステータス「STATUS_ROLL」での処理を見ていきましょう。

※MiyagiQuestView.java に記述

```
/**
* ゲームロールを描写します。
* @param canvas    描写キャンバスクラス
* @param paint     描写ペイントクラス
*/
private void roll(Canvas canvas, Paint paint) {
        //      タッチイベント処理
        switch (TouchStatus) {
        case TouchStatus_TAP:
                if (rollMsgPosY < 50) {
                        //表示位置が、50pixより小さければ、選択キーで、次のステータス（マップ）
へ
                        sleep(500);
                        TouchStatus = -1;
                        setStatus(STATUS_MAP, true);
                        return;
                }
        }
        //描写位置の下限
        if (rollMsgPosY < 50) rollMsgPosY = 50;
```

```
            //        ロールメッセージを表示させます。
            paint.setTextSize(drawFontSize);
            for (int i = 0; i < rollMsgTxt.length; i ++) {
                    paint.setColor(Color.WHITE);
                    canvas.drawText(rollMsgTxt[i], rollMarginX, rollMsgPosY + i*(drawFontSize+5),
paint);
            }
            //        ロールメッセージが、下限になった場合のみ、「Hit Key」を表示させて、次を促します。
            if (rollMsgPosY == 50) {
                    paint.setColor(Color.WHITE);
                    canvas.drawText("Please press Hit Key", rollMarginX,
displayHeight-drawFontSize-20, paint);
                    paint.setColor(Color.BLUE);
                    canvas.drawText("Please press Hit Key", rollMarginX-1,
displayHeight-drawFontSize-20-1, paint);
            }
            rollMsgPosY -= 5;
}
```

roll メソッドでは、最初に TouchStatus メンバ変数を調べます。（キー押下によるキーコードも、このメンバ変数に設定されます）

TouchSTatus_TAP が選択された場合で、rollMsgPosY メンバ変数が 50 よりも小さくなった場合に、ステータスを STATUS_MAP へと変更します。

テキストロール用の文字列は、rollMsgTxt[]という配列に保持されています。おおもとの文字列は、リソースファイル内に設定されており、その文字列を、以下のように取扱い保持しています。

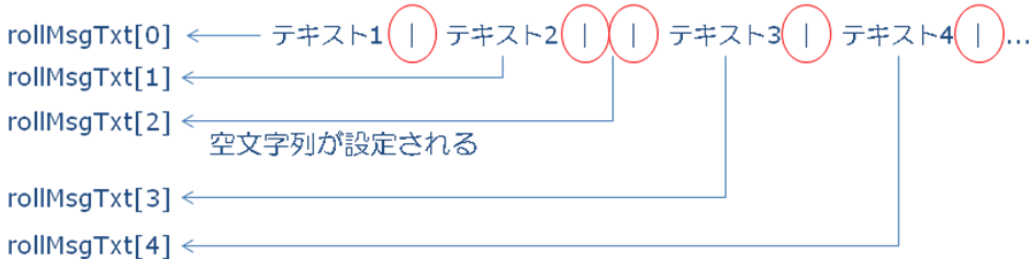


図 20 テキストロール用文字列の保持方法

区切り文字として「｜」を使用しています。

この区切り文字で区切られた文字列を、それぞれ配列へ保持していきます。区切り文字が、二重になっている場合には、配列には、空文字列が配列に保持されます。

これらの文字列を、配列の 1 つを 1 行として、画面上に書き出していきます。画面への書き出し位置は、画面の高さの位置（つまりは、一番下からの書き出し）を行い、5 ピクセルづつ上へ移動させています。

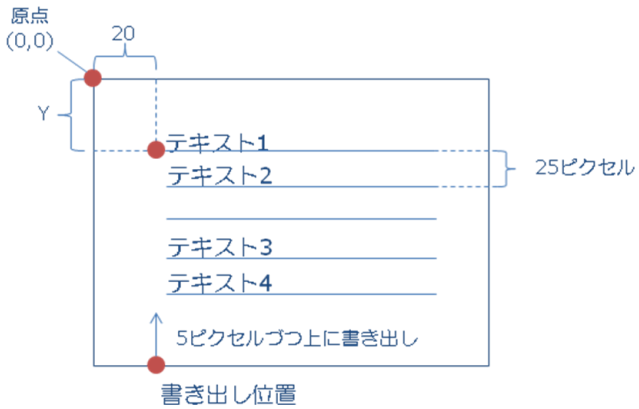


図 21 テキストロールの書き方

1 行あたり 25 ピクセル（テキストサイズ：20 ピクセル+5 ピクセル）となっており、下方向へ改行しながら文字列を描画します。このとき、文字列が画面からは

み出して描画をしていますが、画面には表示されないだけですので、描画していても何ら問題ありません。

4.5.16.3 マップの描画

ここでは、ステータス「STATUS_MAP」での処理を見ていきましょう。

※MiyagiQuestView.java に記述

```
/**
* マップフィールドを描写します。
 * @param canvas    描写キャンバスクラス
 * @param paint     描写ペイントクラス
 */
private void map(Canvas canvas, Paint paint) {
        boolean moved = false;

        //      タッチイベント処理
        switch (TouchStatus) {
        case TouchStatus_UP:
                moved= mapMoved(PLAYER_DIRECTION_UP);
                if(Flag_LongTouch != true) TouchStatus = -1;
                break;
        case TouchStatus_DOWN:
                moved= mapMoved(PLAYER_DIRECTION_DOWN);
                if(Flag_LongTouch != true) TouchStatus = -1;
                break;
        case TouchStatus_LEFT:
                moved= mapMoved(PLAYER_DIRECTION_LEFT);
                if(Flag_LongTouch != true) TouchStatus = -1;
                break;
        case TouchStatus_RIGHT:
                moved= mapMoved(PLAYER_DIRECTION_RIGHT);
                if(Flag_LongTouch != true) TouchStatus = -1;
                break;
        case TouchStatus_TAP:
                        //      選択キーの処理
                int k = filedMapInt[playerMapPosY][playerMapPosX];
                String act = (String)mapActionMap.get(String.valueOf(k));
                android.util.Log.e(DebugLog.TAG, "map value = " + k + ":" + act);
                //      フィールド値が、アクションマップで定義されているかをチェック。
                if ("CURE".equals(act)) {
                        this.hp = this.maxHp;
                        this.sp = this.maxSp;
                        helpMsgTxt = "HPとSPが全回復しました！";
                        try {
                                MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                        helpMsgTxtIndex = -2;
                }
                if ("SAVE".equals(act)) {
                        if (save()) {
                                helpMsgTxt = "保存しました。";
                                try {
                                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
```

```
                        } else {
                                helpMsgTxt = "保存に失敗しました。";
                        }
                        helpMsgTxtIndex = -2;
                }
                TouchStatus = -1;

                //TODO 各種タッチイベント（マップ毎に設定）
                switch (map) {
                        //センダイ**************************************
                        case R.string.sendai:
                        case R.string.sendai2:
                                if ("NPC1".equals(act))
                                /*S*/ helpMsgTxt = "「ここはセンダイの町！秘宝を求めてハンターがたくさん集まる町だ！」";
                                if ("NPC2".equals(act))
                                /*T*/ helpMsgTxt = "「レベルに応じて強いモンスターが出るぜっ！」";
                                if ("NPC3".equals(act))
                                /*U*/ helpMsgTxt = "「武器や防具はタップするだけで入手可能…」";
                                if ("NPC4".equals(act))
                                /*V*/ helpMsgTxt = "「秘宝はどこにあるのだろう？？」";
                                if ("NPC5".equals(act))
                                /*W*/ helpMsgTxt = "「レベル上げは重要だな！」";
                                if ("NPC6".equals(act))
                                        /*X*/ helpMsgTxt = "「宿屋ではHP・SPがタダで回復できるぞっ！」";
                                if ("NPC7".equals(act))
                                /*Y*/ helpMsgTxt = "「北の方に町が有ると聞いたが…」";
                                if ("NPC8".equals(act))
                                /*Z*/ helpMsgTxt = "「俺が最初に秘宝を入手してやるっ！」";
                                if ("WEAPON".equals(act)){
                                        if(FLAG_SENDAI_WEAPON == 0) {
                                                this.str = this.str + this.level + 10;
                                                FLAG_SENDAI_WEAPON = 1;
                                                helpMsgTxt = "攻撃力がアップ！";
                                                try {
                                                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                                } catch (Exception e) {
                                                        e.printStackTrace();
                                                }
                                        } else {
                                                helpMsgTxt = "効果がないようだ";
                                        }
                                        helpMsgTxtIndex = -2;
                                }
                                if ("ARMOR".equals(act)){
                                        if(FLAG_SENDAI_ARMOR == 0) {
                                                this.def = this.def + this.level + 10;
                                                FLAG_SENDAI_ARMOR = 1;
                                                helpMsgTxt = "防御力がアップ！";
                                                try {
                                                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                                                } catch (Exception e) {
                                                        e.printStackTrace();
                                                }
                                        } else {
```

```
                                        helpMsgTxt = "効果がないようだ";
                                }
                                helpMsgTxtIndex = -2;
                        }
                        if ("SHIELD".equals(act)){
                                if(FLAG_SENDAI_SHIELD == 0) {
                                        this.spd = this.spd + this.level + 10;
                                        FLAG_SENDAI_SHIELD = 1;
                                        helpMsgTxt = "素早さがアップ！";
                                        try {
                                                MyUtil.PlaySound_SE(activity,
MyUtil.SE_CURE);
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                        }
                                } else {
                                        helpMsgTxt = "効果がないようだ";
                                }
                                helpMsgTxtIndex = -2;
                        }
                        if ("TREASURE".equals(act)){
                                if(FLAG_ITEM2 != 1){
                                        helpMsgTxt = "";
                                        sleep(500);
                                        helpMsgTxt = "    にじのしずくを手に入れ
た！！";
                                        FLAG_ITEM2 = 1;
                                        try {
                                                MyUtil.PlaySound_SE(activity,
MyUtil.SE_CURE);
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                        }
                                }else{
                                        helpMsgTxt = "…もう何も無い。";
                                }
                        }
                        break;    //CAUTION!

                //オオサキ****************************************
                case R.string.oosaki:
                case R.string.oosaki2:
                        if ("NPC1".equals(act))
                        /*S*/ helpMsgTxt = "「ここはオオサキの町！！」";
                        if ("NPC2".equals(act))
                        /*T*/ helpMsgTxt = "「西の行き止まりが気になる…」";
                        if ("NPC3".equals(act))
                        /*U*/ helpMsgTxt = "「向こう岸へはどうやって行くんだ？」";
                        if ("NPC4".equals(act))
                        /*V*/ helpMsgTxt = "「センダイの扉の先には何が？」";
                        if ("NPC5".equals(act))
                        /*W*/ helpMsgTxt = "「センダイでステータスアップはしたか？」";
                        if ("NPC6".equals(act))
                        /*X*/ helpMsgTxt = "「センダイの…西の湖…」";
                        if ("NPC7".equals(act))
                        /*Y*/ helpMsgTxt = "「古い言い伝え…」";
                        if ("NPC8".equals(act))
                        /*Z*/ helpMsgTxt = "「オーブを捧げると…」";
                        if ("TREASURE".equals(act)){
                                if(FLAG_ITEM3 != 1){
```

```
                                                helpMsgTxt = "";
                                                sleep(500);
                                                helpMsgTxt = "　　オーブを手に入れた！！";
                                                FLAG_ITEM3 = 1;
                                                try {
                                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_CURE);
                                                } catch (Exception e) {
                                                        e.printStackTrace();
                                                }
                                        }else{
                                                helpMsgTxt = "…もう何も無い。";
                                        }
                                }
                                break;    //CAUTION!

                        //レイク**************************************
                        case R.string.lake:
                        case R.string.lake2:
                                if ("NPC1".equals(act))
                                /*S*/ helpMsgTxt = "「…ヒホウハオレノモノダッ！！」";
                                if ("TREASURE".equals(act) && FLAG_BOSS1 == 1){
                                        helpMsgTxt = "";
                                        sleep(500);
                                        helpMsgTxt = "　　伝説の秘宝を手に入れた！！";
                                        try {
                                                MyUtil.PlaySound_SE(activity,
MyUtil.SE_CURE);
                                                setStatus(STATUS_ENDING, true);
                                                init(STATUS_ENDING);
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                        }
                                }
                                if ("TREASURE".equals(act) && FLAG_BOSS1 == 0){
                                        FLAG_BOSS1 = 1;//ボスキャラフラグ
                                        TouchStatus = -1;
                                        setStatus(STATUS_BATTLE, true);//　　戦闘シーンへ
                                }
                                break;

                        case R.string.nanasi:
                                helpMsgTxt = "…怪しい！";
                                if (playerMapPosX == 15 && playerMapPosY == 10){
                                        if ("GRASS".equals(act)){
                                                if(FLAG_ITEM1 != 1){
                                                        helpMsgTxt = "";
                                                        sleep(500);
                                                        helpMsgTxt = "　　カギを手に入れ
た！！";
                                                        FLAG_ITEM1 = 1;
                                                        try {
                                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_CURE);
                                                        } catch (Exception e) {
                                                                e.printStackTrace();
                                                        }
                                                }else{
                                                        helpMsgTxt = "…もう何も無い。";
                                                }
```

```
                                        }
                                    }
                                    break;    //CAUTION!
                        }
                }

                //        地図の描写
                try {
                        canvas.drawBitmap(mapImage, mapMarginX, mapMarginY, paint);
                } catch (Exception e) {
                        e.printStackTrace();
                        android.util.Log.e(DebugLog.TAG, "map map" + playerMapPosY + ", " + playerMapPosX +
", " + fieldMapWidth + ", " + fieldMapHeight);
                        sleep(2000);
                }

                try {
                        //        プレイヤーの描写（必ず中央）
                        canvas.drawBitmap(playerImage[playerDirection][playerDrawIndex],  playerDrawPosX,
playerDrawPosY, paint);

                        //        ステータス枠を描写
                        paint.setStyle(Paint.Style.STROKE);
                        paint.setStrokeWidth(4);
                        paint.setColor(Color.WHITE);
                        canvas.drawRoundRect(new
RectF(mapMarginX+2,statusDrawPosY,displayWidth-mapMarginX*2+2,statusDrawPosY+60),10,10,paint);
                        canvas.drawRoundRect(new
RectF(mapMarginX+2,statusDrawPosY+68,displayWidth-mapMarginX*2+2,statusDrawPosY+68+30),10,10,paint);

                        //        ステータス枠の中を描写
                paint.setStyle(Paint.Style.FILL);
                        paint.setStrokeWidth(1);
                        paint.setTextSize(drawFontSize); //drawFontSize
                        String line1 = "Lv:" + level + " HP:" + hp + "/" + maxHp + " SP:" + sp + "/" + maxSp;
                        canvas.drawText(line1,  mapMarginX + 10, statusDrawPosY+25, paint);

                        paint.setTextSize(13);
                        String line2 = "経験値:" + exp + " / " + nextExp + "  攻:" + str + "  防:" + def +
"  速:" + spd;
                        canvas.drawText(line2, mapMarginX + 10, statusDrawPosY+50, paint);

                        //        ヘルプテキストを描写
                        paint.setTextSize(drawFontSize);
                        int idx = helpMsgTxtIndex;
                        if (idx < 0) idx = 0;
                        for (int i = idx, x = mapMarginX + 10; i < helpMsgTxt.length(); i ++) {
                                String c = helpMsgTxt.substring(i, i+1);
                                float f = paint.measureText(c);
                                if (x+f >= displayWidth-mapMarginX*2-14) {
                                        break;
                                }
                                canvas.drawText(c, x, statusDrawPosY+90, paint);
                                x += f;
                        }
                        if (count%5 == 0) {
                                helpMsgTxtIndex ++;
                                if (helpMsgTxtIndex >= helpMsgTxt.length()) {
                                        helpMsgTxtIndex = -2;
                                        helpMsgTxt = "";
```

```
                        }
                    }
            } catch (Exception e) {
                    e.printStackTrace();
                    android.util.Log.e(DebugLog.TAG, "map player");
            }

            if (moved) {
                    //      敵の出現をランダムで設定。（encount分の1で、敵出現）
                    if (encFlag == true && MyUtil.nextInt(encount) == 0){
                            //      フィールドBGMを停止して、エンカウントSE鳴動
                            try {
                                    MyUtil.PlaySound_SE(activity, MyUtil.SE_MAP_ENCOUNT);
                            } catch (Exception e) {
                                    e.printStackTrace();
                            }
                            sleep(500);
                            TouchStatus = -1;
                            setStatus(STATUS_BATTLE, true);       //      戦闘シーンへ
                            return;
                    }
            }

           if (count%3 == 0) playerDrawIndex = (playerDrawIndex+1)%2;
}
```

サンプルプログラムでは、主人公がマップを移動し、戦闘を行うのが中心になっているプログラムですので、この STATUS_MAP というステータスが、サンプルプログラムにおける中心となるステータスとなります。

まずは、他のメソッドと同様に、TouchStatus メンバ変数を調べます。

マップ上を移動させるため、上下左右方向への移動が有効となります。それぞれの方向を引数として、mapMoved メソッドを呼び出します。

※MiyagiQuestView.java に記述

```
/**
 * 地図上で、指定方向に動いたかを判定します。
 * @param direction 移動方向
 */
public boolean mapMoved(int direction) {
        boolean moved = true;
        //      現在値をバックアップ
        int prevx = playerMapPosX;
        int prevy = playerMapPosY;
        playerDirection = direction;
        switch (direction) {
        case PLAYER_DIRECTION_UP :          //      上を移動したい場合
                playerMapPosY --;
                if (playerMapPosY < 0) {    //      できませんでした。
                        playerMapPosY = 0;
                        moved = false;
                }
                break;
        case PLAYER_DIRECTION_DOWN :        //      下を移動したい場合
                playerMapPosY ++;
                if (playerMapPosY >= fieldMapHeight) {      //      できませんでした。
```

```
                        playerMapPosY = fieldMapHeight-1;
                        moved = false;
                }
                break;
        case PLAYER_DIRECTION_LEFT :            //      左に移動したい場合
                playerMapPosX --;
                if (playerMapPosX < 0) {        //      できませんでした。
                        playerMapPosX = 0;
                        moved = false;
                }
                break;
        case PLAYER_DIRECTION_RIGHT :           //      右に移動したい場合
                playerMapPosX ++;
                if (playerMapPosX >= fieldMapWidth) { //      できませんでした。
                        playerMapPosX = fieldMapWidth-1;
                        moved = false;
                }
                break;
        }
        if (moved) {
                //      地図上では移動できた。ので、次に、フィールドが移動可能なフィールドかど
うかをチェックする。
                int idx = filedMapInt[playerMapPosY][playerMapPosX];
                char c = MAP_DEF.charAt(idx);

                //TODO 地形に応じた戦闘背景
                if      (c == '5') batteImage = batteImages[1]; //山
                else if(c == 'E') batteImage = batteImages[2]; //雪
                else if(c == '8' || c == 'K' || c == 'S')
                                                batteImage = batteImages[3];        //夕日
                else                            batteImage = batteImages[0];        //草原

                if (filedMapCannotMoveTxt.indexOf(c) != -1) {
                        //      移動できないフィールドなので、元に戻す。
                        playerMapPosX = prevx;
                        playerMapPosY = prevy;
                        moved = false;
                } else {
                //      移動できるフィールドだったので、移動後の地図表示を行う
                //      地図の再描写
                        Bitmap bitmap = Bitmap.createBitmap(mapImage);
                        Canvas canvas = new Canvas(mapImage);
                        Paint paint = new Paint();
                        paint.setAntiAlias(true);
                        paint.setStyle(Paint.Style.FILL);

                        int x = 0;
                        int y = 0;
                        int i, j;
                        switch(direction) {
                        case PLAYER_DIRECTION_UP:                       //      上を描写
                                canvas.drawBitmap(bitmap, 0, fieldMapImageWidth, paint);
                                y = playerMapPosY - 6;
                                for (x = playerMapPosX - 6, i = 0; x < playerMapPosX + 7; x ++,
i ++) {
                                if (isDrawableMap(x, y)) {
                                        canvas.drawBitmap(fldImage[filedMapInt[y][x]],
i*fieldMapImageWidth, 0, paint);
                                } else {
                                        canvas.drawBitmap(fldImage[cannotMoveImageInt],
```

```
i*fieldMapImageWidth, 0, paint);
                                        }
                                        }
                                        break;
                        case PLAYER_DIRECTION_DOWN:                             //        下を描写
                                        canvas.drawBitmap(bitmap, 0, -fieldMapImageWidth, paint);
                                        y = playerMapPosY + 7 - 1;
                                        for (x = playerMapPosX - 6, i = 0; x < playerMapPosX + 7; x ++,
i ++) {
                                        if (isDrawableMap(x, y)) {
                                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],
i*fieldMapImageWidth, 12*fieldMapImageWidth, paint);
                                        } else {
                                                canvas.drawBitmap(fldImage[cannotMoveImageInt],
i*fieldMapImageWidth, 12*fieldMapImageWidth, paint);
                                        }
                                        }
                                        break;
                        case PLAYER_DIRECTION_LEFT:                             //        左を描写
                                        canvas.drawBitmap(bitmap, fieldMapImageWidth, 0, paint);
                                        x = playerMapPosX - 6;
                                        for (y = playerMapPosY - 6, j = 0; y < playerMapPosY + 7; y ++,
j ++) {
                                        if (isDrawableMap(x, y)) {
                                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],  0,
j*fieldMapImageWidth, paint);
                                        } else {
                                                canvas.drawBitmap(fldImage[cannotMoveImageInt], 0,
j*fieldMapImageWidth, paint);
                                        }
                                        }
                                        break;
                        case PLAYER_DIRECTION_RIGHT:                    //        右を描写
                                canvas.drawBitmap(bitmap, -fieldMapImageWidth, 0, paint);
                                x = playerMapPosX + 7 - 1;
                                for (y = playerMapPosY - 6, j = 0; y < playerMapPosY + 7; y ++, j ++) {
                                        if (isDrawableMap(x, y)) {
                                                canvas.drawBitmap(fldImage[filedMapInt[y][x]],
12*fieldMapImageWidth, j*fieldMapImageWidth, paint);
                                        } else {
                                                canvas.drawBitmap(fldImage[cannotMoveImageInt],
12*fieldMapImageWidth, j*fieldMapImageWidth, paint);
                                        }
                                }
                                break;
                                }
                }

                DebugLog.println("X = " + playerMapPosX + " Y = " + playerMapPosY);

                //マップ遷移チェック
                try{
                        mapChange();
                }catch(Exception e){
                        //ignore
                }

                //マップオブジェクト上でのヘルプメッセージ
                int tileNo = filedMapInt[playerMapPosY][playerMapPosX];
                String act = (String)mapActionMap.get(String.valueOf(tileNo));
```

```
                    if ("CURE".equals(act)) {
                            helpMsgTxt = "画面タップ = HP・SP回復";
                            helpMsgTxtIndex = -2;
                    } else if ("SAVE".equals(act)) {
                            helpMsgTxt = "画面タップ = セーブ";
                            helpMsgTxtIndex = -2;
                    } else if ("WEAPON".equals(act) || "ARMOR".equals(act) || "SHIELD".equals(act))
{
                            helpMsgTxt = "画面タップ = ステータスアップ";
                            helpMsgTxtIndex = -2;
                    } else if ("NPC1".equals(act) ||
                               "NPC2".equals(act) ||
                               "NPC3".equals(act) ||
                               "NPC4".equals(act) ||
                               "NPC5".equals(act) ||
                               "NPC6".equals(act) ||
                               "NPC7".equals(act) ||
                               "NPC8".equals(act)){
                            helpMsgTxt = "画面タップ = 会話";
                            helpMsgTxtIndex = -2;
                    }
            }

            //マップ再作成
            this.initMapImage(new Canvas(mapImage));

            return moved;
}
```

mapMoved メソッドでは、まず、フィールドマップの大きさで移動可能であるかを判定します。フィールドマップの大きさは、fieldMapHeight メンバ変数および fieldMapWidth メンバ変数にて保持されており、移動出来る範囲は（0, 0）から（fieldMapWidth, fieldMapHeight）の範囲で移動することが出来ます。（「図 22　フィールドマップの大きさ」参照）

このため、上下左右キーの方向キー押下によって、現在位置（playerMapPosX, playerMapPosY）を移動させていきます。

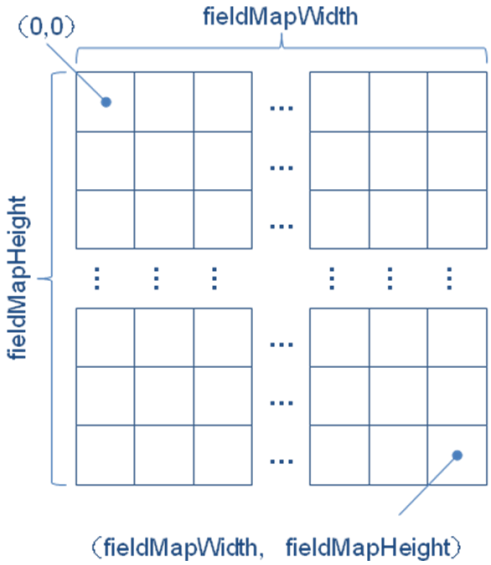


図 22 フィールドマップの大きさ

フィールドマップの大きさで移動可能であると判断出来た後、次に、マップ内で指定されている

フィールドが移動可能なフィールドであるかをチェックしています。

移動可能ではないフィールドの場合（移動可／不可については、「表 1 フィールド一覧」を参照して下さい）には、直前に保持していた位置に戻します。

移動可能である場合には、移動後のフィールドを描画します。

以下に、このフィールドの描画方法を、上方向の場合を例に説明しましょう。

```
Bitmap bitmap = Bitmap.createBitmap(mapImage);
Canvas canvas = new Canvas(mapImage);
Paint paint = new Paint();
paint.setAntiAlias(true);
paint.setStyle(Paint.Style.FILL);
    :
canvas.drawBitmap(bitmap, 0, fieldMapImageWidth, paint);
int y = playerMapPosY - 6;
for (int x = playerMapPosX - 6, i = 0; x < playerMapPosX + 7; x ++, i ++) {
    if (isDrawableMap(x, y)) {
        work_bmp = fldImage[filedMapInt[y][x]];
    } else {
        work_bmp = fldImage[cannotMoveImageInt];
    }
    canvas.drawBitmap(work_bmp, i * fieldMapImageWidth, 0, paint);
}
```

まず、フィールドマップのイメージを作成し、そのイメージをキャンバス上に描画します。

このとき、描画する位置は、先頭から 1 行分（fieldMapImageWidth：1 行というのは、1 つのフィールド画像のサイズ分を指します）下から描画しています。これは、上方向へ移動しているため、画面上では、既存のフィールドマップが 1 行分下へ移動しているように見せています。（逆に言うと、1 行単位でキャラクターは移動していくことになります）

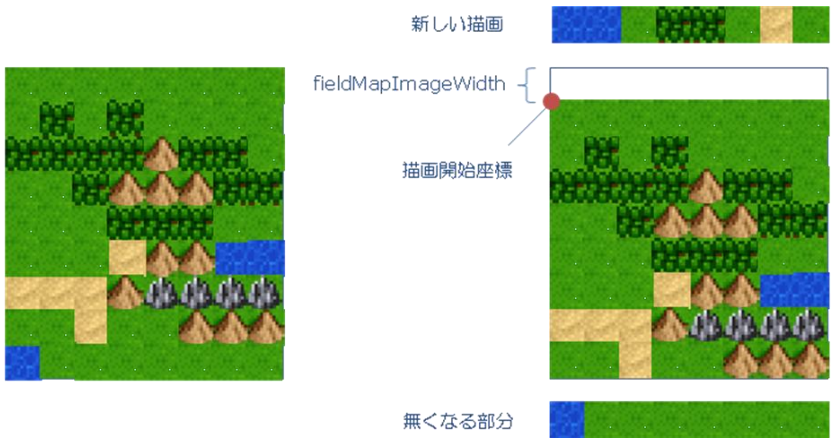


図 23 上方向への移動の描画方法

このように、上下左右方向への移動により、フィールドマップをそれぞれの方向へ移動させ、消えてしまった部分のフィールドマップを描画するようにしています。

上下左右方向以外にも、タップ処理（ENTER キー押下）も有効になっており、このとき、下に

あるフィールドによって処理を行うようになっています。

サンプルプログラムでは、以下の 2 つのフィールドの場合に、特殊な処理を行います。

表 10 特殊フィールド

画像	内容	画像	内容
	全回復エリア 移動することが出来る。 この上で、ENTER キーを押下し、全 HP/MP が全回復する。		保存エリア 移動することが出来る。 この上で、ENTER キー押下により保存することが出来る

全回復エリアの場合は、最大 HP／MP を、HP および MP に設定し、回復します。

保存エリアの場合には、MiyagiQuestView クラスの save メソッドを呼び出し、データの保存処理を行います。保存するデータとしては、現在のレベル、最大 HP、現 HP、攻撃力、防御力、スピード、経験値、次レベルまでの経験値、プレイヤーX 座標値、プレイヤーY 座標値を保存します。保存したデータは、OutputStream クラスを使用してファイルへ保存するようにします。

- File クラス
  boolean mkdirs()
  説明：ディレクトリの作成
  引数：なし
  戻り値：True – 成功、False – 失敗

- FileOutputStream クラス
  FileOutputStream(String file)
  説明：FileOutputStream クラスのコンストラクタ[17]
  引数：file　ファイル名

ファイル名は、パスの先頭を「/data/」で始めます。write メソッドを利用し、バイトデータを書き込み、close メソッドでクローズします。

- OutputStream クラス
  void write(data)
  説明：バイトデータの全データの書き込み
  引数：data　バイトデータ

- OutputStream クラス
  void flush()
  説明：書き込んだバイトデータを明示的に送信
  引数：なし

サンプルプログラムでは、OutputStream クラスの flush メソッドを呼び出した後、すぐに close メソッドを呼び出しているので、本来は flush メソッドは必要ありません。

それでは、どのような場合に、flush メソッドを利用した方が良いのでしょうか？

出力ストリームでは、write メソッド呼び出し時に、すぐに物理デバイス上への書き込みは行われません。出力ストリームの内部バッファがいっぱいになるか、flush メソッドあるいは close メソッドの呼び出しによって、初めて物理デバイスへ書き込まれます。このため、close メソッドの呼び

[17] コンストラクタとは、C++や Java 言語などのオブジェクト指向言語で作成したプログラムにおいて、データとそれを操作するための手続きを一体化したオブジェクトの生成時に呼び出される特殊な関数のことを言います。

出しまでに別の処理があり、その処理中に何らかのエラーでプログラムが続行出来なくなったときに、物理デバイスへデータが書き込まれていない場合があります。（物理デバイスへ書き込まれているか、書き込まれていないかは状況によって異なります）

このようなことの無いように、データ保存処理等を行う場合には、flush メソッドを利用して、明示的に物理デバイスへ書き込みを行うようにした方が良い場合もあります。

ちなみに、close メソッドが呼び出された場合には、まず、flush メソッドが呼び出され、内部バッファのデータを物理デバイスへ書き込みを行い、その後、出力ストリームがクローズされます。

- OutputStream クラス
  void close()
  説明：出力ストリームのクローズ
  引数：なし

さて、次に、主人公のキャラクターの描画と、ステータス欄（HP、最大 HP、経験値、攻撃力、防御力、スピード）、ヘルプ欄の描画を行います。

キャラクターの配置位置は、playerDrawPosX、playerDrawPosY に保存されており、この座標値を利用します。（キャラクターは、必ず画面の中央に表示します）

ステータス欄およびヘルプ欄には、角丸矩形の描画には、Canvas クラスの drawRoundRect メソッドを使用します。このときに描画する矩形のライン幅は、Paint クラスの setStrokeWidth メソッドを使用します。

- Paint クラス
  void setStrokeWidth(float width)
  説明：ライン幅の指定
  引数：width　ライン幅

- Canvas クラス
  void drawRoundRect(RectF rect, float rx, float ry, Paint)
  説明：角丸矩形の描画
  引数：rect　矩形のデータ構造
  　　rx　　コーナーの X 方向半径
  　　ry　　コーナーの Y 方向半径
  　　paint　描画オブジェクト

RectF クラスは、float 型の矩形情報のクラスです。

- RectF クラス
  RectF(float left, float top, float right, float bottom)
  説明：RectF クラスのコンストラクタ
  引数：left　　矩形の左上の X 座標
  　　top　　矩形の左上の Y 座標
  　　right　　矩形の右下の X 座標
  　　bottom　矩形の右下の Y 座標

移動処理終了後、乱数を利用して敵キャラクターとランダムで遭遇をチェックします。

MyUtil クラスに、このランダムに遭遇させるための nextInt メソッドを使用します。

```
/**
```

```
 *    指定された整数値までのランダム数を返す
 *    @param    k    ランダム係数
 *    @return        ランダム数
 */
public static int nextInt(int k) {
    java.util.Date d = new java.util.Date();
    Random r = new Random(d.getTime()+cnt);
    long p = r.nextInt();
    cnt = p;
    return java.lang.Math.abs((int)p) % k;
}
```

このメソッドでは、Random オブジェクトを作成して、nextInt メソッドを呼び出します。乱数に使用するシード値[18]には、日付を利用します。この日付に、前回の乱数生成時に取得した値を加算して、更にランダムに数値を得られるようにしています。

なぜ、このような方法をとっているかというと、コンピュータで乱数を生成する場合、ある数値をシード値として用い、その値をもとにして特定の演算により乱数を生成します。このため、同じシード値を使用した場合、発生する乱数がまったく同じ値になる可能性が高いためです。

取得した乱数は、Math クラスの abs メソッドを利用し、絶対値が得られ、その値を任意の値（引数で指定されている値）で割った余りを求めれば、0 から任意の値までの乱数を得ることが出来ます。

このようにして得られた乱数が、0 になったときに敵と遭遇したと判定します。サンプルプログラムでは、任意の値を 20 としており、20 分の 1 の確率で敵と遭遇することになります。敵と遭遇した場合には、ステータスを STATUS_BATTLE へと変更します。

4.5.16.4 戦闘シーンの描画

続いて、ステータス「STATUS_BATTLE」の処理を見ていきましょう。以下に、battle メソッドを抜粋します。

※MiyagiQuestView.java に記述

```
/**
 * 戦闘シーンを描写します。
 * @param canvas    描写キャンバスクラス
 * @param paint     描写ペイントクラス
 */
@SuppressWarnings("unchecked")
private void battle(Canvas canvas, Paint paint) throws Exception {

        //特殊攻撃対応
        boolean damagein = false;
        if(selectFlag == true) damagein = true;

        //        戦闘アクション中は、キー操作を無効とする。
        if (battleActions.length != 0 && battleActions[0] != null) TouchStatus = -1;
        //        戦闘アクションがあれば、キー操作無効
        //        タッチイベント処理
        switch (TouchStatus){
```

[18] 乱数を生成する際に、その元となる数値をシード値（Seed：種）と呼びます。

```
		case TouchStatus_UP:
			if(Flag_LongTouch != true) TouchStatus = -1;
			if (battleEnemySelectedIndex == -1) {
				//	敵選択中ではない場合＝戦闘メニュー選択中
				battleMenuSelectedIndex	=
(battleMenuSelectedIndex+2)%4;
			} else {
				//	敵選択中だったので、戦闘メニューに戻す。
				battleEnemySelectedIndex = -1;
			}
			break;
		case TouchStatus_DOWN:
			if(Flag_LongTouch != true) TouchStatus = -1;
			if (battleEnemySelectedIndex == -1) {
				//	敵選択中ではない場合＝戦闘メニュー選択中
				battleMenuSelectedIndex = (battleMenuSelectedIndex+2)%4;
			} else {
				//	敵選択中だったので、戦闘メニューに戻す。
				battleEnemySelectedIndex = -1;
			}
			break;
		case TouchStatus_LEFT		:
			if(Flag_LongTouch != true) TouchStatus = -1;
			if (battleEnemySelectedIndex == -1) {
				//	敵選択中ではない場合＝戦闘メニュー選択中
				battleMenuSelectedIndex = battleMenuSelectedIndex/2*2
+(battleMenuSelectedIndex+1)%2;
			} else {
				//	敵選択中なので、敵の選択を行う
				int idx = battleEnemySelectedIndex;
				battleEnemySelectedIndex --;
				if (battleEnemySelectedIndex < 0) battleEnemySelectedIndex
= enemy.length-1;
				while (enemy[battleEnemySelectedIndex].hp <= 0) {
					battleEnemySelectedIndex --;
					if (battleEnemySelectedIndex < 0) battleEnemySelectedIndex
= enemy.length-1;
					if (battleEnemySelectedIndex == idx) break;
				}
			}
			break;
		case TouchStatus_RIGHT:
			if(Flag_LongTouch != true) TouchStatus = -1;
			if (battleEnemySelectedIndex == -1) {
				//	敵選択中ではない場合＝戦闘メニュー選択中
				battleMenuSelectedIndex = battleMenuSelectedIndex/2*2
+(battleMenuSelectedIndex+1)%2;
			} else {
				//	敵選択中なので、敵の選択を行う
				int idx = battleEnemySelectedIndex;
				battleEnemySelectedIndex ++;
				if (battleEnemySelectedIndex >= enemy.length)
 battleEnemySelectedIndex = 0;
				while (enemy[battleEnemySelectedIndex].hp <= 0) {
					battleEnemySelectedIndex ++;
					if (battleEnemySelectedIndex >= enemy.length)
 battleEnemySelectedIndex = 0;
					if (battleEnemySelectedIndex == idx) break;
				}
			}
```

```
                break;
        case TouchStatus_TAP:
                //      選択キーの処理
                if (battleEnemySelectedIndex == -1) {
                        //      敵選択中ではない場合＝戦闘メニュー選択中
                        if (battleMenuSelectedIndex == 0) {
                                //      「戦う」が選択されました。
                                TouchStatus = -1;
                                for (int i = 0; i < enemy.length; i ++) {
                                        if (enemy[i].hp > 0) {
                                                battleEnemySelectedIndex = i;
                                                break;
                                        }
                                }
                                battleMsgTxts = new String[4];
                                battleMsgTxts[0] = "攻撃対象を選択してください。";
                                battleMsgMasterTxts = new String[0];
                                count = 1;

                                selectFlag = true;//特殊攻撃対応

                        } else if (battleMenuSelectedIndex == 1) {
                                //      「特殊」が選択されました。
                                if (sp <= reqSpec){
                                        battleMsgTxts = new String[4];
                                        battleMsgTxts[0] = "SPが足りません";
                                }else {
                                        TouchStatus = -1;
                                        sp -= reqSpec;
                                        for (int i = 0; i < enemy.length; i ++) {
                                                if (enemy[i].hp > 0) {
                                                        battleEnemySelectedIndex = i;

        break;
                                                        }
                                        }
                                        battleMsgTxts = new String[4];
                                        battleMsgTxts[0] = "";
                                        battleMsgMasterTxts = new String[0];
                                        count = 1;
                                        damagein = true;
                                        paintFlag = true;
                                }
                        } else if (battleMenuSelectedIndex == 2) {
                                //      「回復」が選択されました。
                                if (sp < reqCure){
                                        battleMsgTxts = new String[4];
                                        battleMsgTxts[0] = "SPが足りません";
                                }else{
                                        TouchStatus = -1;
                                        sp -= reqCure;
                                        for (int i = 0; i < enemy.length; i ++) {
                                                if (enemy[i].hp > 0) {
                                                        battleEnemySelectedIndex = i;
                                                        break;
                                                }
                                        }
                                        battleMsgTxts = new String[4];
                                        battleMsgTxts[0] = "";
                                        battleMsgMasterTxts = new String[0];
```

```
                                        count = 1;
                                        damagein = true;
                                }
                        } else if (battleMenuSelectedIndex == 3) {
                                //      「逃げる」が選択されました。
                                //      今回は、必ず、逃げれます。
                                //      ただし、ボス戦時は選択不可とする！
                                if(FLAG_BOSS1 != 1){
                                        TouchStatus = -1;
                                        battleEnemySelectedIndex = -1;
                                        sleep(500);
                                        setStatus(STATUS_MAP, false);
        //      地図へ。
                                        return;
                                }
                        }
                //} else {
                }
                //特殊攻撃・回復対応
                if (damagein == true) {
                        selectFlag = false;

                        //      敵が選択されました。
                        count = 1;
                        //      戦闘の順番を決めます。（素早さから判断する）
                        ArrayList list = new ArrayList();
                        for (int i = 0; i < enemy.length; i ++) {
                                list.add(enemy[i]);
                        }
                        Enemy player = new Enemy(this, new int[0], "ミヤギ君", hp, str,
 def, spd, -1, 0, 0);
                        list.add(player);
                        Object obj[] = list.toArray();
                        Arrays.sort(obj, new EnemyComparator(null));
        //      ここで、ソートされます。
                        ArrayList battleActionList = new ArrayList();
                        Enemy skip = null;

                        //      ソートされた結果で、戦闘アクションを追加していきます。
                        for (int i = 0; i < obj.length; i
++) {
                                Enemy e = (Enemy)obj[i];
                                if (e == player) {
                                        //      自分の番になった場合の処理

                                        //戦う
                                        if(battleMenuSelectedIndex == 0){
                                                int idx = battleEnemySelectedIndex;
                                                int                 damage                  =
enemy[idx].getAttackDamage(player);

                                                battleActionList.add("MESSAGE:ミヤギ君の攻
撃");

                                                battleActionList.add("SLEEP:500");
                                                battleActionList.add("ATTACK:"   +   idx   +
":"+damage+":"

                                                        + enemy[idx].name + "に、¥n"
                                                        + damage + "のダメージを与えた！");
                                                battleActionList.add("SLEEP:500");
                                                int enemyHp = enemy[idx].hp - damage;
                                                if (enemyHp <= 0) {
```

```
                                                enemy[idx].skip = true;
                                                //        敵を倒した！！
                                                battleActionList.add("MESSAGE:"+
enemy[idx].name + "を倒した！");
                                        }

                                        //特殊攻撃
                                        }else if(battleMenuSelectedIndex == 1){
                                                battleActionList.add
("MESSAGE:ミヤギ君の特殊攻撃");
                                                battleActionList.add("SLEEP:500");
                                                for(int i = 0; i < enemy.length; i++){
                                                        if(enemy[i].hp <= 0){
                                                                continue;
                                                        }else{
                                                                int damage2 =
 enemy[i].getAttackDamage(player);
                                        battleActionList.add("MAGIC:" + i + ":"+damage2+":"

                                + enemy[i].name + "に、¥n"

                                                + damage2 + "のダメージを与えた！");
                                                battleActionList.add("SLEEP:500");
                                                int enemyHp2 = enemy[i].hp - damage2;

                                                if (enemyHp2 <= 0) {
                                                        enemy[i].skip = true;
                                                        //        敵を倒した！！
                                                        battleActionList.add("MESSAGE:"

                                + enemy[i].name + "を倒した！");
                                                                }
                                                        }
                                                }

                                        //回復行動
                                        }else if(battleMenuSelectedIndex == 2){
                                                battleActionList.add
("MESSAGE:ミヤギ君の回復");
                                                battleActionList.add("SLEEP:500");
                                                battleActionList.add("RECOVER:");
                                                battleActionList.add("MESSAGE:"
 + "ミヤギ君はHPを回復した");
                                                battleActionList.add("SLEEP:500");
                                        }
                                } else {
                                        //        敵の番になった場合の処理
                                        if (e.hp <= 0) continue;
                                //        すでに倒している敵では、処理せず。
                                        //else if (e.equals(skip)) continue;
                //        自分の番の時に、倒した敵なので、処理せず。
                                        else if (e.skip == true) continue;
                                        int damage = player.getAttackDamage(e);
                                        battleActionList.add("MESSAGE:" + e.name +
 "の攻撃！");
                                        battleActionList.add("SLEEP:500");
                                        battleActionList.add("ATTACKED:" + e.idx
+ ":"+damage
                                                + ":" + damage + "のダメージ！");
                                        battleActionList.add("SLEEP:500");
```

```
                                        }
                                }

                                //      敵を全部倒したかの判定
                                int enemyExp = 0;
                                int cnt = 0;
                                for (int i = 0; i < enemy.length; i ++) {
                                        //if (enemy[i].hp <= 0 || enemy[i].equals(skip)) {
                                        if (enemy[i].hp <= 0 || enemy[i].skip == true) {
                                                cnt ++;
                                                enemyExp += enemy[i].exp;
                                        }
                                }
                                if (cnt == enemy.length) {
                                        battleActionList.add("MESSAGE:敵を全部倒した！¥n"
                                                + enemyExp + "の経験値を手に入れた！");
                                        battleActionList.add("SLEEP:1000");
                                        exp += enemyExp;
                                        //      レベルアップ値の決定
                                        if (exp >= nextExp) {
                                                int maxHpUp = MyUtil.nextInt(15)+5;
                                                int maxSpUp = MyUtil.nextInt(level)+10;
                                                int strUp = MyUtil.nextInt(5)+2;
                                                int defUp = MyUtil.nextInt(5)+2;;
                                                int spdUp = MyUtil.nextInt(5)+2;
                                                maxspup = maxSpUp;
                                                battleActionList.add("MESSAGE:レベルアップ！！！");
                                                battleActionList.add("SLEEP:1000");
                                                battleActionList.add("LEVELUP:" + maxHpUp + "."
 + strUp + "." + defUp + "." + spdUp
                                                + ":最大HP・SPが" + maxHpUp + "・"+ maxSpUp + "アップ！"
                                                + "¥n攻撃力が、"+ strUp + "アップ！"
                                                + "¥n防御力が、"+ defUp + "アップ！"
                                                + "¥n素早さが、"+ spdUp + "アップ！");
                                                battleActionList.add("SLEEP:1000");
                                        }
                                        battleActionList.add("FINISH:");
                                } else {
                                        battleActionList.add("CLEAR:");
                                }
                                battleActions = MyUtil.a2s(battleActionList);
                        }
                }

                if (count%5 == 0) {
                        if (battleActions.length != 0 && battleActions[0] != null) {
                                //      戦闘アクションごとに処理を行っていく。
                                String action = battleActions[0];
                                if (action.startsWith("ATTACK:")) {
                                        //      「攻撃した」
                                        battleMsgTxts = new String[4];
                                        String msgs[] = MyUtil.split
(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] = msgs[i];
                                        }
                                        String acts[] = MyUtil.split(action, ":");
                                        int idx = Integer.parseInt(acts[1]);
                                        int damage = Integer.parseInt(acts[2]);
                                        int enemyHp = enemy[idx].setDamage(damage);
```

```
                                        if (enemyHp <= 0) {
                                android.util.Log.e(DebugLog.TAG, "Done ![" + enemyHp + "]");
                                        }
                                        MyUtil.PlaySound_SE(activity,
MyUtil.SE_BATTLE_ATTACK_PLAYER);
                                        }else if(action.startsWith("MAGIC:")){
                                        //特殊攻撃
                                        battleMsgTxts = new String[4];
                                        String msgs[] = MyUtil.split
(action.substring(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] = msgs[i];
                                        }
                                        String acts[] = MyUtil.split(action, ":");
                                        int idx = Integer.parseInt(acts[1]);
                                        int damage = (int) (Integer.parseInt(acts[2]) * 0.8);
                                        int enemyHp = enemy[idx].setDamage(damage);
                                        if (enemyHp <= 0) {
                                android.util.Log.e(DebugLog.TAG, "Done ![" + enemyHp + "]");
                                        }
                                        MyUtil.PlaySound_SE(activity, MyUtil.SE_MAGIC);
                                        paintFlag = false;

                                }else if(action.startsWith("RECOVER:")){

                                        //回復行動
                                        int recovery = (int)(maxHp * 0.3);
                                        if(hp + recovery > maxHp){
                                                hp = maxHp;
                                        }else{
                                                hp += recovery;
                                        }
                                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);

                                } else if (action.startsWith("ATTACKED:")) {   //        「攻撃された」
                                        battleMsgTxts = new String[4];
                                        String msgs[] = MyUtil.split(action.substring
(action.lastIndexOf(":")+1), "¥n");
                                        for (int i = 0; i < battleMsgTxts.length; i ++) {
                                                if (i < msgs.length) battleMsgTxts[i] = msgs[i];
                                        }
                                        String acts[] = MyUtil.split(action, ":");
                                        int damage = Integer.parseInt(acts[2]);
                                        hp = hp - damage;
                                        MyUtil.PlaySound_SE(activity, MyUtil.SE_BATTLE_ATTACK_ENEMY);

                                        if (hp <= 0) {
                                                hp = 0;
                                                //        ゲームオーバーへ。
                                                ArrayList battleActionList = new ArrayList();
                                                battleActionList.add(action.
substring(action.lastIndexOf(":")+1));
                                                battleActionList.add("SLEEP:500");
                                                battleActionList.add
("MESSAGE:ミヤギ君は...¥n力尽きた...。");
                                                battleActionList.add("SLEEP:1000");
                                                battleActionList.add
("MESSAGE:ミヤギ君は...¥n力尽きた...。¥n¥nむっ、無念だ！！！");
                                                battleActionList.add("SLEEP:2000");
                                                battleActionList.add("GAMEOVER:");
```

```
                                        battleActions = MyUtil.a2s(battleActionList);
                                android.util.Log.e(DebugLog.TAG, "GAMEOVER[" + hp + "]");
                                }
                            } else if (action.startsWith("MESSAGE:")) {
                    //        「メッセージ表示」
                                battleMsgTxts = new String[4];
                                String msgs[] = MyUtil.split
(action.substring(action.lastIndexOf(":")+1), "¥n");
                                for (int i = 0; i < battleMsgTxts.length; i ++) {
                                        if (i < msgs.length) battleMsgTxts[i] = msgs[i];
                                }
                            } else if (action.startsWith("SLEEP:")) {
                    //        「処理一時停止」
                                long sleep = Long.parseLong(action.substring
(action.lastIndexOf(":")+1));
                                sleep(sleep);
                            } else if (action.startsWith("LEVELUP:")) {
            //        「レベルアップ」
                                String up[] = MyUtil.split(action, ":");
                                up = MyUtil.split(up[1], ".");
                                int maxHpUp = Integer.parseInt(up[0]);
                                int strUp = Integer.parseInt(up[1]);
                                int defUp = Integer.parseInt(up[2]);
                                int spdUp = Integer.parseInt(up[3]);
                                level ++;
                                maxHp += maxHpUp;
                                hp = maxHp;
                                maxSp += maxspup;
                                sp = maxSp;
                                str += strUp;
                                def += defUp;
                                spd += spdUp;
                                nextExp += 100*(level-1);
                                battleMsgTxts = new String[4];
                                String msgs[] = MyUtil.split(action.substring
(action.lastIndexOf(":")+1), "¥n");
                                for (int i = 0; i < battleMsgTxts.length; i ++) {
                                        if (i < msgs.length) battleMsgTxts[i] = msgs[i];
                                }
                                MyUtil.PlaySound_SE(activity, MyUtil.SE_BATTLE_LEVELUP);
                            } else if (action.startsWith("CLEAR:")) {
                    //        「クリア」
                                battleMsgTxts = new String[4];
                            } else if (action.startsWith("FINISH:")) {
                    //        「終了」
                                TouchStatus = -1;
                                battleEnemySelectedIndex = -1;
                                sleep(500);
                                setStatus(STATUS_MAP, false);
                                //        地図へ
                                return;
                            } else if (action.startsWith("GAMEOVER:")) {
            //        「ゲームオーバー」
                                TouchStatus = -1;
                                battleEnemySelectedIndex = -1;
                                sleep(500);
                                setStatus(STATUS_GAMEOVER, true);
                            //        ゲームオーバーへ
                                return;
                            }
```

```
                                DebugLog.println("battle action[" + action + "]");
                                //        戦闘アクションのスワップ（差し替え）
                                for (int i = 0; i < battleActions.length-1; i ++) {
                                        battleActions[i] = battleActions[i+1];
                                        DebugLog.println("battleActions[i]:"+battleActions[i]);
                                }
                                battleActions[battleActions.length-1] = null;
                                count = 1;
                                if (battleActions[0] == null) {
                                        //        元に戻る。
                                        battleEnemySelectedIndex = -1;
                                }
                        }
                }

                //戦闘背景の描画
                canvas.drawBitmap(batteImage, 0, 0,paint);

                //        敵の描写
                paint.setStyle(Paint.Style.STROKE);
                paint.setStrokeWidth(2);
                paint.setColor(Color.RED);
                for (int i = 0; i < enemy.length; i ++) {
                        //enemy[i].drawEnemy(canvas, paint, count%4==0, i == battleEnemySelectedIndex);
                        if(battleMenuSelectedIndex == 0){
                                enemy[i].drawEnemy(canvas, paint, count%4==0, i
 == battleEnemySelectedIndex);
                        }else if(paintFlag == true){
                                enemy[i].drawEnemy(canvas, paint, count%4==0, true);
                        }else{
                                enemy[i].drawEnemy(canvas, paint, count%4==0, false);
                        }
                }

                //        戦闘シーンの枠を描写
                paint.setStyle(Paint.Style.STROKE);
                paint.setStrokeWidth(4);
                paint.setColor(Color.WHITE);
                canvas.drawRoundRect(new RectF(battleMarginX+2,displayHeight/2+battleMarginY
,displayWidth-battleMarginX*2+2,displayHeight/2+60+battleMarginY),10,10,paint);
                        canvas.drawRoundRect(new RectF(battleMarginX+2,displayHeight/2+68+battleMarginY,
displayWidth-battleMarginX*2+2,displayHeight/2+68+110+battleMarginY),10,10,paint);
                        canvas.drawRoundRect(new
RectF(mapMarginX+2,statusDrawPosY+68,displayWidth-mapMarginX*2+2,statusDrawPosY+68+30),10,10,paint);

                //        戦闘シーンの枠の中を描写
               paint.setStyle(Paint.Style.FILL);
                paint.setStrokeWidth(1);
                paint.setTextSize(drawFontSize);
                canvas.drawText("戦　う", battleMarginX + 50,
                                        displayHeight / 2 + 25 + battleMarginY, paint);

                canvas.drawText("特　殊", displayWidth / 2 + 45,
                                        displayHeight / 2 + 25 + battleMarginY, paint);
                canvas.drawText("回　復", battleMarginX + 50,
                                        displayHeight / 2 + 50 + battleMarginY, paint);
                //ボス戦時は選択不可とする！
                if(FLAG_BOSS1 != 1){
                        canvas.drawText("逃げる", displayWidth / 2 + 45,
                                        displayHeight / 2 + 50 + battleMarginY, paint);
```

```
		}
		canvas.drawText("HP  : " + hp + "/" + maxHp,  mapMarginX + 10, statusDrawPosY + 90, paint);
		canvas.drawText("SP  : " + sp + "/" + maxSp,  mapMarginX + 150, statusDrawPosY + 90, paint);

		//		選択されている戦闘メニューの描写
		if (battleMenuSelectedIndex == 0) {
			canvas.drawText(">", battleMarginX + 50 - battleMenuSelectedMinusX,
						displayHeight/2 + 25 + battleMarginY, paint);
			canvas.drawText("<", battleMarginX + 50 + battleMenuSelectedPlusX,
						displayHeight/2 + 25 + battleMarginY, paint);
		} else if (battleMenuSelectedIndex == 1) {
			canvas.drawText(">", displayWidth / 2 + 45 - battleMenuSelectedMinusX,
						displayHeight / 2 + 25 + battleMarginY, paint);
			canvas.drawText(" < ",  displayWidth  /  2  +  45  +  battleMenuSelectedPlusX,
displayHeight/2+25+battleMarginY, paint);
		} else if (battleMenuSelectedIndex == 2) {
			canvas.drawText(">", battleMarginX + 50 - battleMenuSelectedMinusX,
						displayHeight / 2 + 50 + battleMarginY, paint);
			canvas.drawText("<", battleMarginX + 50 + battleMenuSelectedPlusX,
						displayHeight / 2 + 50 + battleMarginY, paint);
		} else if (battleMenuSelectedIndex == 3) {
			//ボス戦時は選択不可とする！
			if(FLAG_BOSS1 != 1){
				canvas.drawText(">", displayWidth / 2 + 45 - battleMenuSelectedMinusX,
						displayHeight / 2 + 50 + battleMarginY, paint);
				canvas.drawText("<", displayWidth / 2 + 45 + battleMenuSelectedPlusX,
						displayHeight / 2 + 50 + battleMarginY, paint);
			}
		}

		//		戦闘メッセージの描写
		for (int i = 0, j = 0; i < battleMsgTxts.length; i ++, j ++) {
			if (battleMsgTxts[i] == null) continue;
			canvas.drawText(battleMsgTxts[i],  battleMarginX + 10,
				  displayHeight / 2 + 68 + 25 + j * 25 +battleMarginY, paint);
			DebugLog.println("canvas.drawText:"+battleMsgTxts[i]);
		}

		if (count%5 == 0) {
			//		戦闘メッセージのスワップ（入れ替え）
			for (int i = 0; i < battleMsgTxts.length-1; i ++) {
				battleMsgTxts[i] = battleMsgTxts[i+1];
			}
			battleMsgTxtIndex ++;
			if (battleMsgTxtIndex < 0 || battleMsgTxtIndex >= battleMsgMasterTxts.length) {
				battleMsgTxts[battleMsgTxts.length-1] = null;
			} else {
				battleMsgTxts[battleMsgTxts.length-1]                                    =
battleMsgMasterTxts[battleMsgTxtIndex];
			}
		}
}
```

まずは、他のメソッドと同様に、TouchStatus メンバ変数を調べます。

ただし、すでに戦闘中である場合には、スクリーンへのタッチ操作やキー操作は無効にしておかなければいけないため、TouchStatus メンバ変数に「-1」を設定し、処理が行われないようにします。

この STATUS_BATTLE では、以下のような戦闘前のメニュー選択、戦闘中の敵キャラクター選択の 2 つのモードでのキー操作があります（以下の画面は、イメージ画面であり、サンプルプログラムで描画される画面とは異なりますので、ご注意ください）。

図 24 STATUS_BATTLE での選択モード

メニュー選択モードでは、「戦う」「特殊」「回復」「逃げる」の 4 つのメニューが存在しています。「逃げる」については、サンプルプログラムでは、必ず逃げることが出来るようになっていますが、敵キャラクターのスピードと主人公のスピードから乱数などを利用して、逃げることが出来るかどうかをチェックするようにしても良いかも知れませんね。

なお、「特殊」「回復」については、別冊の「MiyagiQuest 追加機能　使様解説」で解説しています。

「戦う」が選択された場合、次に、敵キャラクターの選択モードへと移行します。

敵キャラクター選択後、「図 25　作成流れ」以下のようにして、戦闘シーンを作り上げます。

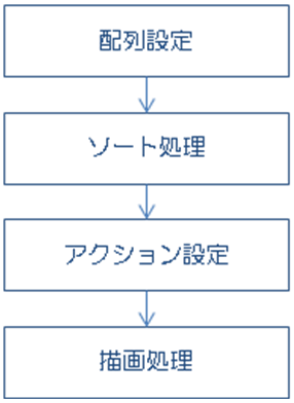


図 25 作成流れ

配列設定というのは、まず、主人公キャラクターと敵キャラクターの各種属性値と共に保存していきます。このとき使用するのが、Enemy クラスです。この Enemy クラスを配列として保持します。

その後、EnemyComparator クラスを使用して、配列内のデータを比較します。このとき、比較対象とするのは、各キャラクターが持つスピードによって順序を決定します。この順序は、攻撃を行う順序であり、先に攻撃が出来るかどうかは、ゲーム上、非常に重要な意味を持ってきます。このため、サンプルプログラムでは、単純にスピードだけを比較対象としていますが、ここでも乱数などを利用して、スピードが遅いキャラクターも順序が偏らないようにした方が、よりゲームとしては面白いものとなることでしょう。

次に、アクションの設定です。アクションの設定には、battleActions メンバ変数によって、文字列配列として保持されており、設定値は、以下のようになっています。

表 11 アクションメッセージ一覧

文字列	内容	その他パラメータ
MESSAGE	メッセージを表示します。	メッセージ内容
ATTACK	主人公⇒敵へ攻撃を行います。	敵配列のインデックス

文字列	内容	その他パラメータ
		ダメージ
		メッセージ
ATTACKED	敵⇒主人公へ攻撃を行います。	敵配列のインデックス
		ダメージ
		メッセージ
CLEAR	敵を倒した	なし
FINISH	敵をすべて倒した	なし
SLEEP	スリープ処理を行います。	時間（ミリ秒）
GAMEOVER	主人公が倒れてしまった	なし

例として、ATTACK の場合には、以下のような文字列が設定されます。

```
ATTACK:1:34:敵に、¥n43のダメージを与えた！
```

各パラメータは、「:」（コロン）で区切られています。（このため、メッセージ内に「:」（コロン）を使用してはいけません）このように、アクションが設定された上で、描画処理が行われることになります。

○ ArrayList クラス
ArrayList()
説明：ArrayList オブジェクトのコンストラクタ

○ ArrayList クラス
boolean add(Object object)
説明：ArrayList への追加
引数：object　追加するオブジェクト
戻り値：True – 成功、False – 失敗

○ ArrayList クラス
boolean add(Object object)
説明：ArrayList への追加
引数：object　追加するオブジェクト
戻り値：True – 成功、False – 失敗

4.5.16.5 ゲームオーバーの描画

ステータス「STATUS_GAMEOVER」の処理を見ていきましょう。

```
/**
 * ゲームオーバーを描写します。
 * @param    canvas     描写キャンバスクラス
 * @param    paint      描写ペイントクラス
 */
private void gameover(Canvas canvas, Paint paint) {
    switch (TouchStatus) {
    case TouchStatus_TAP:
```

```
            //    選択キーの処理
            sleep(500);
            TouchStatus = -1;
            setStatus(STATUS_TITLE, true);    //    タイトルへ
            return;
        }
        //    ゲームオーバーを描写
        paint.setTextSize(drawFontSize);
        paint.setColor(Color.WHITE);
        canvas.drawText("Game Over", 20, displayHeight / 2 - drawFontSize / 2, paint);
    }
```

gameover メソッドでも、他のメソッドと同様に、TouchStatus メンバ変数のチェックを行います。このメソッドでは、TouchStatus_TAP ステータスのみ有効となります。タップや ENTER キー押下により、ステータスを STATUS_TITLE に変更し、タイトル画面描画へと遷移します。

ゲームオーバー文字列を描画します。使用しているメソッドは、これまでにも説明しているメソッドですので、改めて説明はしませんが、文字列を描画位置の計算方法（画像の描画位置については、「4.5.16.1 タイトル描画」も参照して下さい）は、さまざまなプログラムでも使用することが出来ると思いますので覚えておきましょう。

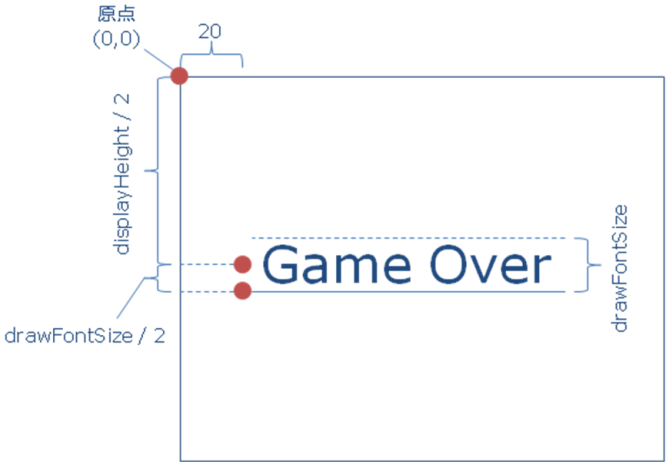


図 26 GameOver 描画位置

携帯電話アプリケーションの場合に、良く画面中央（中心）、縦方向での中央、横方向での中央などは使われることが多く使われますので、ちょっとだけ説明をしておきたいと思います。

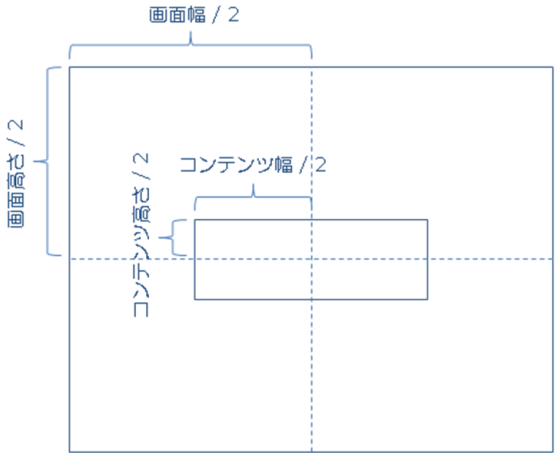


図 27 基準位置の計算方法

どの場合であっても、画面のサイズと文字列や画像などのサイズがあれば、その中央を求めることは可能です。右図に記載されている内容からも求めることは可能です。

コンテンツの左上が基準点である場合には、以下の式で求めます。

X = （画面幅 – コンテンツ幅）/ 2
Y = （画面高さ – コンテンツ高さ）/ 2

基準点が左下であれば、以下の式となります。

X = （画面幅 – コンテンツ幅）/ 2
Y = （画面高さ + コンテンツ高さ）/ 2

その他にも、基準点によって計算方法は変わりますが、これらの情報から求めることが出来ることは分かりますよね？

4.5.16.6 エラーの描画

ステータス「STATUS_ERROR」の処理を見ていきましょう。

```
/**
 * エラーを描写します。
 * @param    canvas    描写キャンバスクラス
 * @param    paint     描写ペイントクラス
 */
private void error(Canvas canvas, Paint paint) {
    paint.setColor(Color.WHITE);
    canvas.drawText(errorTxt, 20, 20, paint);
}
```

これは、単純なメソッドですね。

単に、errorTxt メンバ変数に設定されている文字列を、（20, 20）の位置に描画するだけのメソッドとなります。

5. おわりに

さて、ここまでロールプレイングゲームである MiyagiQuest のサンプルプログラムを見てきましたが、このサンプルプログラムは、道具や魔法を使うことは出来ませんし、街や洞窟などのダンジョンなども存在していません。敵のキャラクターも 2 体しか出てきませんし、他のキャラクターとの出会いなどもありません。つまり、このサンプルプログラムは、ロールプレイングゲームとしては、まだまだ未完成のプログラムなのです。

このため、もっと多くの機能を追加していくことで、もっと楽しいロールプレイングゲームに育て上げていくことが出来ますので、是非、本書を読んでいる方々には、どんどんプログラムを追加して頂ければと思います。そのためには、「2.1 RPG 制作の流れ」にもあるように、設計なども行っていく必要もありますし、キャラクター作成なども行っていかなければいけません。もちろん、Java のプログラミング言語を覚えていくことも必要ではありますが、その知識だけでは、ロールプレイングゲームを作っていくことは出来ませんので、頑張って下さいね。

平成 23 年度　文部科学省

東日本大震災からの復旧・復興を担う専門人材育成支援事業

モバイルアプリケーションの開発で
震災復興を支援する人材の育成
「MiyagiQuest 追加機能 仕様解説書」

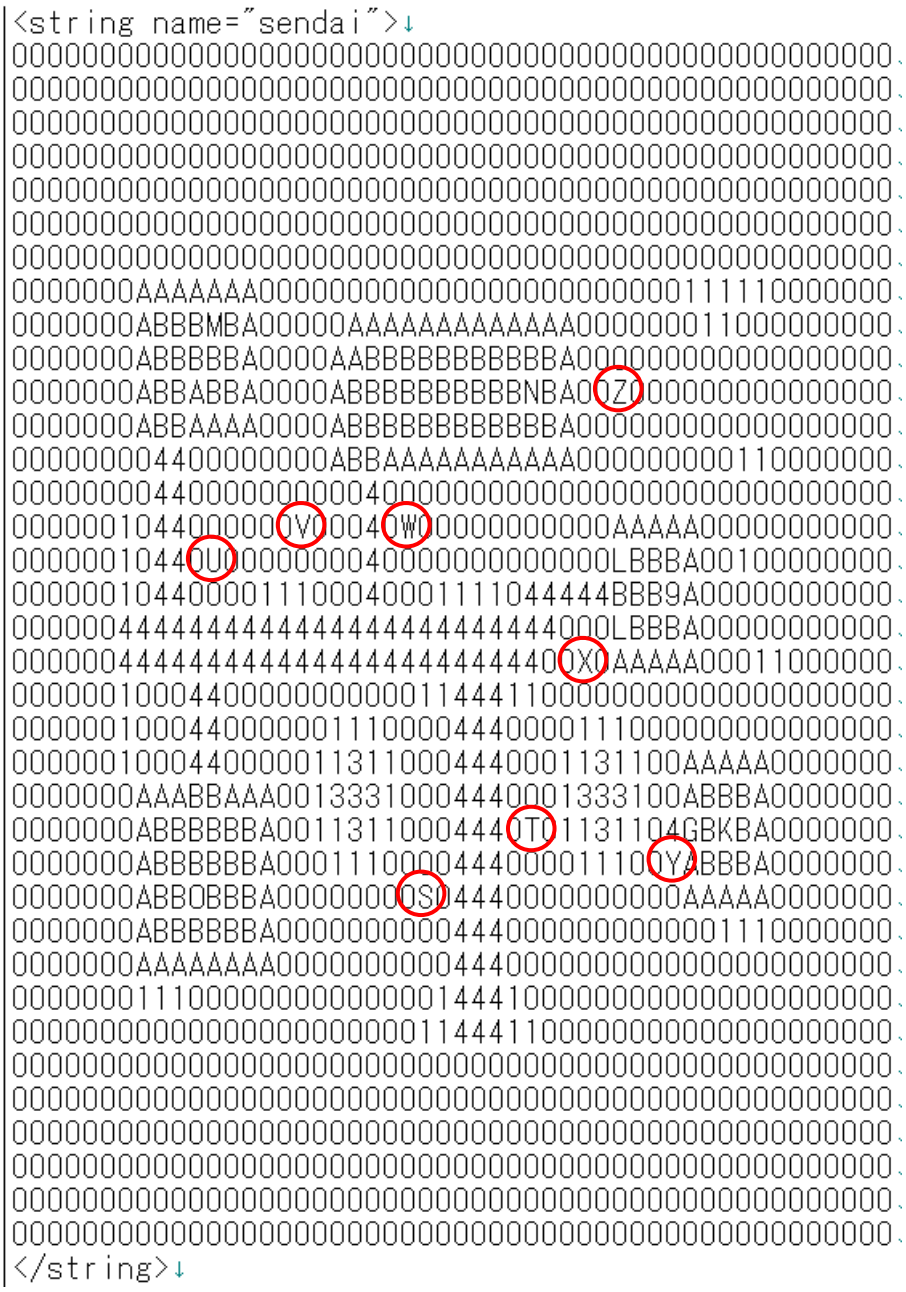
平成 24 年　3 月 15 日　第 1 版記

目　次

1　NPC（町にいるキャラクター）の設定方法

１－１　マップ上の座標

基本的には他のマップチップオブジェクトと同様の扱いで設定しています。
その為、strings.xml 内の各マップに該当文字を設定する事で、表示位置を設定する事になります。



NPC チップ（S-Z）までを埋めたセンダイマップの例（strings.xml）

元々、0～R までのキャラ文字がマップチップとして予約済である為、S～Z の 8 種類を新たに NPC オブジェクトとし strings.xml に記述します。

```
<string name="map_action_U">GRASS</string>↓
<string name="map_action_4">SAND</string>↓
<string name="map_action_9">CURE</string>↓
<string name="map_action_C">SAVE</string>↓
<string name="map_action_D">DAMEGE</string>↓
<string name="map_action_E">SNOW</string>↓
<string name="map_action_F">TOWN</string>↓
<string name="map_action_G">DOOR</string>↓
<string name="map_action_H">DUNGEON</string>↓
<string name="map_action_I">SIGNBOARD</string>↓
<string name="map_action_J">SPECIAL</string>↓
<string name="map_action_K">TREASURE</string>↓
<string name="map_action_L">INN</string>↓
<string name="map_action_M">WEAPON</string>↓
<string name="map_action_N">ARMOR</string>↓
<string name="map_action_O">SHIELD</string>↓
<string name="map_action_P">DRUG</string>↓
<string name="map_action_Q">CROSS</string>↓
<string name="map_action_R">BRIDGE</string>↓
<string name="map_action_S">NPC1</string>↓
<string name="map_action_T">NPC2</string>↓
<string name="map_action_U">NPC3</string>↓
<string name="map_action_V">NPC4</string>↓
<string name="map_action_W">NPC5</string>↓
<string name="map_action_X">NPC6</string>↓
<string name="map_action_Y">NPC7</string>↓
<string name="map_action_Z">NPC8</string>↓
```

NPC オブジェクトとして S-Z までを定義する例（strings.xml）

１－２　グラフィック

１－１　で位置を設定したら、次は、その位置に表示するグラフィックの設定を行います。こちらも、先述したマップチップと同様の処理となります。その為、NPC 画像もフィールド fldImage にロードして使用します。

```
/** フィールドマップ画面用パラメータ */
final static private String MAP_DEF = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";   //  地図定義
final static private int PLAYER_DIRECTION_DOWN = 0;      //  プレイヤーの向き（下）
final static private int PLAYER_DIRECTION_LEFT = 1;      //  プレイヤーの向き（左）
final static private int PLAYER_DIRECTION_RIGHT = 2;     //  プレイヤーの向き（右）
final static private int PLAYER_DIRECTION_UP = 3;        //  プレイヤーの向き（上）
private Bitmap[] fldImage = new Bitmap[36];              //  フィールド画像配列 28+NPC8=36
private Bitmap mapImage = null;                          //  地図描写用全体画像
private int[][] filedMapInt = new int[0][0];             //  地図情報配列
private String filedMapCannotMoveTxt = "";               //  移動不可地図情報テキスト
```

NPC 画像はフィールド画像と同じフィールドで扱う

（MiyagiQuestView.java@フィールド定義）

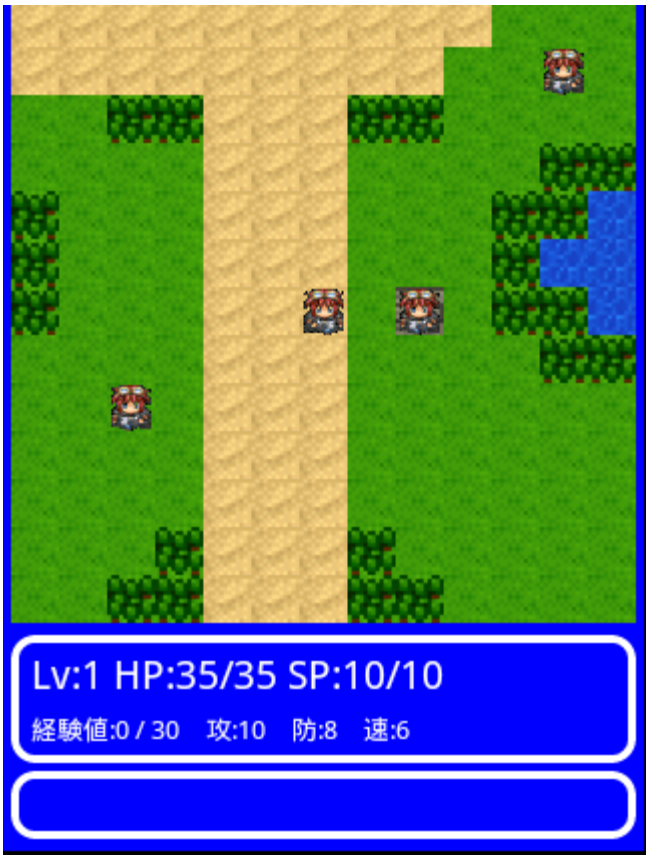
実際に画像をファイルから読み込むタイミングは、マップチップ画像読み込みの直後に行っています。

```
        //  マップ画像の読み込み
        //  フィールド名と、実際値が異なるので、reflectionを利用。
        //  reflection とは、フィールド名などから、その値を取得するための機能。
        try {
            //  どのクラスから取得するかを指定
            Class cls = Class.forName("com.cgene.game.R$drawable");
            for (int i = 1, idx = 0; i <= 7; i ++) {
                for (int j = 1; j <= 4; j ++) {
                    String fldName = "field_" + i + j;
                    fldImage[idx] = BitmapFactory.decodeResource(r, getResouceInt(cls, fldName));
                    idx ++;
                }
            }

            //TODO NPCキャラ（マップチップと同じ扱い）読み込み
            fldImage[28] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_76" ));
            fldImage[29] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_77" ));
            fldImage[30] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_78" ));
            fldImage[31] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_79" ));
            fldImage[32] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_80" ));
            fldImage[33] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_81" ));
            fldImage[34] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_82" ));
            fldImage[35] = BitmapFactory.decodeResource(r, getResouceInt(cls,"field_83" ));

```

NPC 画像ファイル（field_76～field83 まで）を読み込んでいる

（MiyagiQuestView.java@コンストラクタ）

これで使用する準備は整いました。このまま実行すれば、マップの指定位置に NPC 画像が表示されます。しかし、このままでは何のイベントも発生しません。次に、タップにより会話イベントを行えるように設定します。



NPC 画像ファイルは正しく表示されるが何もイベントは起こらない

１－３　会話時のセリフ

最初に、プレイヤーキャラクタがマップ上で NPC と重なった場合、タップする事で会話する事が可能であるというヘルプメッセージ「画面タップ = 会話」を表示するようにします。プレイヤーキャラクタが移動に成功したかを判断するメソッド mapMoved メソッド内に記述します。移動成功後の移動先が NPC キャラクタであった場合、直ぐにメッセージを表示したいからです。

```
            //マップオブジェクト上でのヘルプメッセージ↓
            int tileNo = filedMapInt[playerMapPosY][playerMapPosX];↓
            String act = (String)mapActionMap.get(String.valueOf(tileNo));↓
            if ("CURE".equals(act)) {↓
                helpMsgTxt = "画面タップ = HP・SP回復";↓
                helpMsgTxtIndex = -2;↓
            } else if ("SAVE".equals(act)) {↓
                helpMsgTxt = "画面タップ = セーブ";↓
                helpMsgTxtIndex = -2;↓
            } else if ("WEAPON".equals(act) || "ARMOR".equals(act) || "SHIELD".equals(act)) {↓
                helpMsgTxt = "画面タップ = ステータスアップ";↓
                helpMsgTxtIndex = -2;↓
            } else if ("NPC1".equals(act) ||↓
                       "NPC2".equals(act) ||↓
                       "NPC3".equals(act) ||↓
                       "NPC4".equals(act) ||↓
                       "NPC5".equals(act) ||↓
                       "NPC6".equals(act) ||↓
                       "NPC7".equals(act) ||↓
                       "NPC8".equals(act)){↓
                helpMsgTxt = "画面タップ = 会話";↓
                helpMsgTxtIndex = -2;↓
            }↓
        }↓
↓
```

いずれかの NPC と座標が重なった場合にヘルプメッセージを表示

(MiyagiQuestView.java@mapMoved メソッド)

「画面タップ ＝ 会話」のヘルプメッセージを表示

次に、実際にタップした際の処理を記述します。
マップフィールドの描写を行っている map メソッド内に記述します。
NPC 座標マスとプレイヤーキャラクタ座標が重なっており、かつタップ操作された時のタッチイベントとして処理しています。その際、「どのマップの、どの NPC(S～Z)か」を分岐処理により判断し、対応するメッセージを画面に出力しています。

簡潔にまとめると、下記４ステップでの判断処理です。

① タッチイベント処理　　：画面にタッチされているか
② タッチ種類判定処理　　：そのタッチはタップ操作か
③ マップ判定処理　　：プレイヤーキャラクタの居るマップは何処か
④ イベント判定処理　　：NPC キャラクタか

③以降の処理個所を抜粋した実際のソースを見てみましょう。ソースはセンダイマップの例です。

マップ毎の分岐

分岐先 sendai、sendai2 を同じにしているのは、同一マップ扱いの為。

```
//TODO 各種タッチイベント（マップ毎に設定）↓
switch (map){↓
    //センダイ**********************************↓
    case R.string.sendai:↓
    case R.string.sendai2:↓
        if ("NPC1".equals(act))↓
            /*S*/ helpMsgTxt = "「ここはセンダイの町！秘宝を求めてハンターがたくさん集まる町だ！」";↓
        if ("NPC2".equals(act))↓
            /*T*/ helpMsgTxt = "「レベルに応じて強いモンスターが出るぜっ！」";↓
        if ("NPC3".equals(act))↓
            /*U*/ helpMsgTxt = "「武器や防具はタップするだけで入手可能…」";↓
        if ("NPC4".equals(act))↓
            /*V*/ helpMsgTxt = "「秘宝はどこにあるのだろう？？」";↓
        if ("NPC5".equals(act))↓
            /*W*/ helpMsgTxt = "「レベル上げは重要だな！」";↓
        if ("NPC6".equals(act))↓
            /*X*/ helpMsgTxt = "「宿屋ではHP・SPがタダで回復できるぞっ！」";↓
        if ("NPC7".equals(act))↓
            /*Y*/ helpMsgTxt = "「北の方に町が有ると聞いたが…」";↓
        if ("NPC8".equals(act))↓
            /*Z*/ helpMsgTxt = "「俺が最初に秘宝を入手してやるっ！」";↓
        if ("WEAPON".equals(act)){↓
            if(FLAG_SENDAI_WEAPON == 0) {↓
                this.str = this.str + this.level + 10;↓
                FLAG_SENDAI_WEAPON = 1;↓
                helpMsgTxt = "攻撃力がアップ！";↓
                try {↓
```

タップされた際のオブジェクト毎にメッセージを表示

（MiyagiQuestView.java@map メソッド）

１－１～１－３の処理が正しく行われると、プレイヤーキャラクタがそのマップにいる NPC と会話するイベントを作成する事が出来ます。

NPC６との会話メッセージ表示の例

１－４　NPC（町にいるキャラクター）の設定時における注意点

NPC キャラクタの会話を実装する上で、注意すべき点が２つあります。

1 つ目は、「メッセージ表示とスクロールスピードがゆっくりである為、長文メッセージには不向きである」という点です。現在の仕様では、会話メッセージはヘルプメッセージとして画面下部に表示されます。そして、メッセージが 1 画面に収まらない長さである場合、自動的に左スクロールする事で全体メッセージが表示されていきます。しかし、1 度表示が開始されてしまえば、プレイヤーは次の操作に移行する事が可能です。その為、長いメッセージの場合、最後まで表示し終わる前に次のイベント（例：別 NPC との会話等）を実行させてしまうと、前のメッセージ表示がキャンセルされて、次のメッセージ表示がされてしまいます。

2 つ目は、「今回の実装方法では、1 マップあたり最大 8 人の NPC しか配置出来ない」という点です。先述した様に、NPC はマップチップと同様に S-Z の 8 種類 を予約文字として設定しています。そして、それらの NPC は１－３で説明した様に、マップ毎に処理が区別されています。つまり、同一マップ内で S という NPC を複数配置した際、全て同じイベントしか設定出来ないという事になります。その為、より複雑なイベントや NPC 設定を行いたいのであれば、

① 町の中であっても建物等に入ったら別マップに切り替わる様に作成する
（1 マップを複数マップに細かく分割する）
② S～Z 以外に予約済のマップチップを NPC キャラとして再定義する

等の工夫が必要になるでしょう。

２　マップ間移動の設定方法

２－１　移動先マップの設定

全てのマップは、strings.xml 内で設定しています。0~9、A－Z の１文字を１つのマップチップに割り当てて、それらを２次元に並べる事でマップを構築しています。

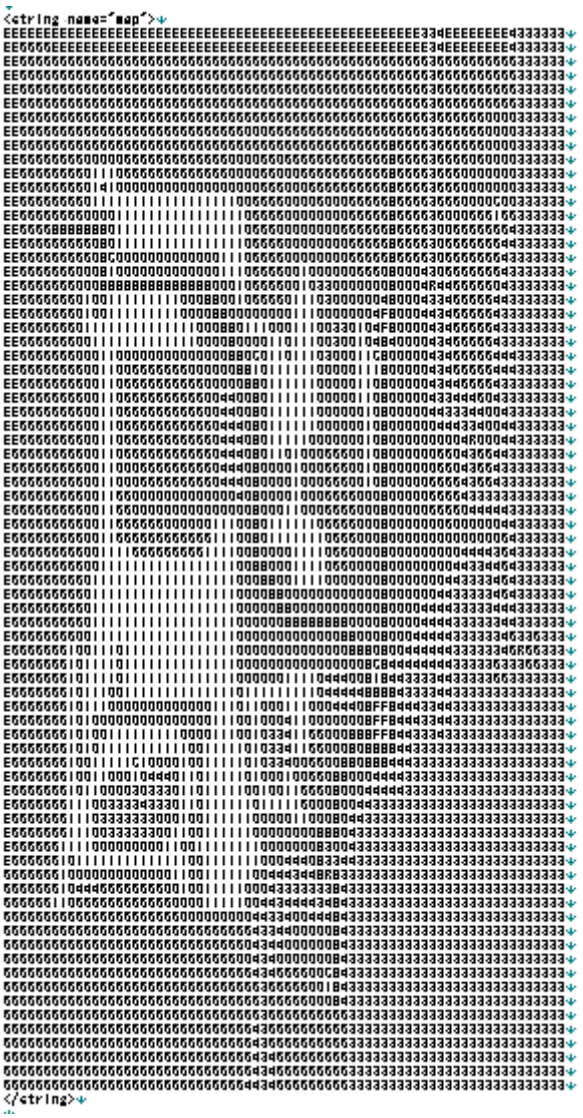
```
<string name="map_action_0">GRASS</string>
<string name="map_action_1">FOREST</string>
<string name="map_action_2">BOGLAND</string>
<string name="map_action_3">WATER</string>
<string name="map_action_4">SAND</string>
<string name="map_action_5">MOUNTAIN</string>
<string name="map_action_6">ROCK</string>
<string name="map_action_7">STONE</string>
<string name="map_action_8">FLOOR</string>
<string name="map_action_9">CURE</string>
<string name="map_action_A">WALL</string>
<string name="map_action_B">ROAD</string>
<string name="map_action_C">SAVE</string>
<string name="map_action_D">DAMEGE</string>
<string name="map_action_E">SNOW</string>
<string name="map_action_F">TOWN</string>
<string name="map_action_G">DOOR</string>
<string name="map_action_H">DUNGEON</string>
<string name="map_action_I">SIGNBOARD</string>
<string name="map_action_J">SPECIAL</string>
<string name="map_action_K">TREASURE</string>
<string name="map_action_L">INN</string>
<string name="map_action_M">WEAPON</string>
<string name="map_action_N">ARMOR</string>
<string name="map_action_O">SHIELD</string>
<string name="map_action_P">DRUG</string>
<string name="map_action_Q">CROSS</string>
<string name="map_action_R">BRIDGE</string>
<string name="map_action_S">NPC1</string>
```

マップチップとマップアクションを対応させている例（strings.xml）

また、１つのマップにおける最大幅・高さも設定されています。こちらは、MiyagiQuestView.java のフィールド値として設定されています。fieldMapWidh が最大幅で fieldMapHeight が高さです。デフォルトは８０×８０となっていますが、より大きなマップを構築したいのであれば、こちらの値を変更する事になります。

```
    private int playerMapPosX = 0;                  //  プレイヤーの地図上のｘ座標
    private int playerMapPosY = 0;                  //  プレイヤーの地図上のｙ座標
    private int playerDrawPosX = 0;                 //  プレイヤーの表示位置。必ず中央です。
    private int playerDrawPosY = 0;                 //  プレイヤーの表示位置。必ず中央です。
    private int mapMarginX = 0;                     //  画面上に描写する地図のマージンｘ座標
    private int mapMarginY = 0;                     //  画面上に描写する地図のマージンｙ座標
    private int fieldMapImageWidth = 24;            //  フィールド画像の幅
    private int fieldMapWidth = 80;//40;            //  定義されている地図の最大横幅
    private int fieldMapHeight = 80;//40;           //  定義されている地図の最大縦幅
    private int playerDirection = PLAYER_DIRECTION_DOWN;    //  プレイヤーの向き
    private int statusDrawPosY = 0;                 //  ステータス表示をするｙ座標
```

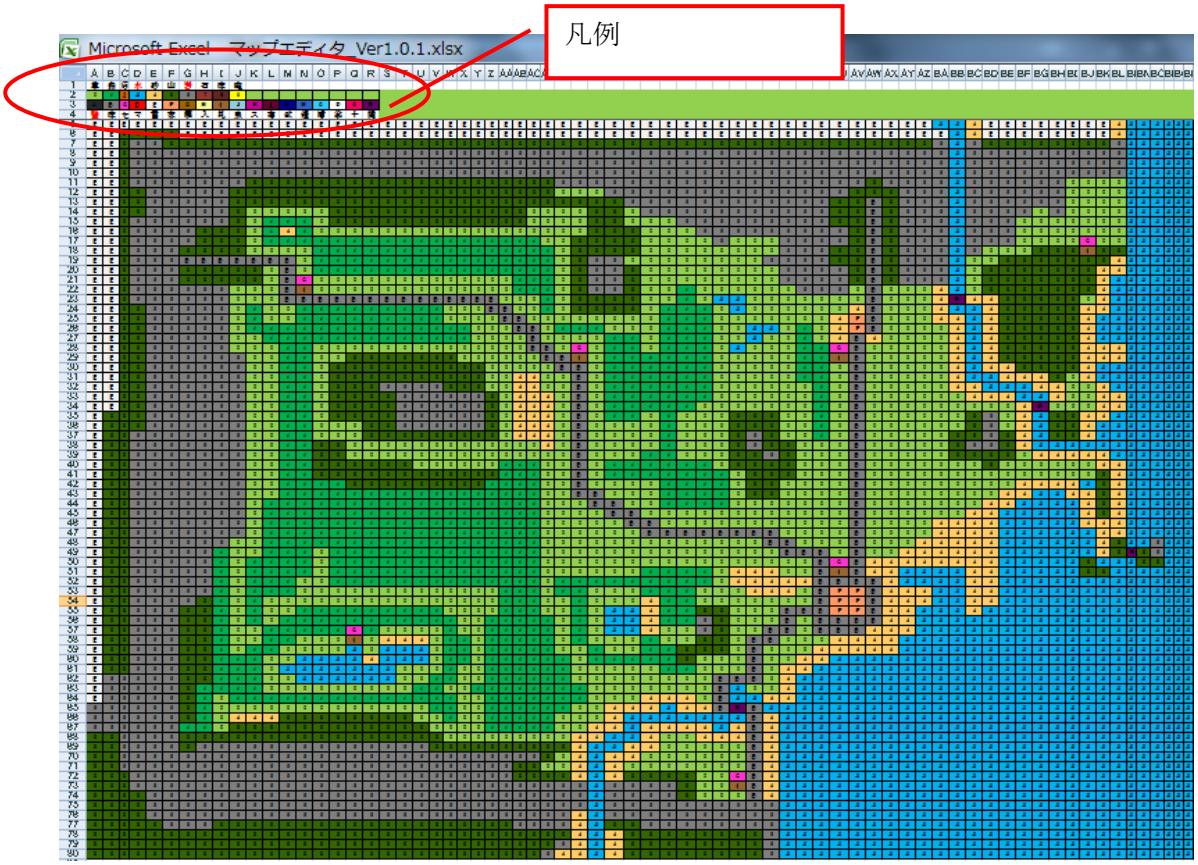
マップの最大幅・高さの設定値（MiyagiQuestView.java）



マップチップを８０×８０に並べている例（strings.xml）

文字とマップの対応が分かりましたので、実際にマップ作成に取りかかる事は可能です。しかし、何のツールも用いずにマップ作製を行う事は、は具体的なイメージも湧かず、大変な作業となります。そこで、表計算ソフト EXCEL 等の条件付き書式等を利用した簡易

マップエディタを利用します。0-9、A-Z までの文字をセル内に入力するとその意味に近似した色に変更する様に予め条件付き書式として設定しておきます。

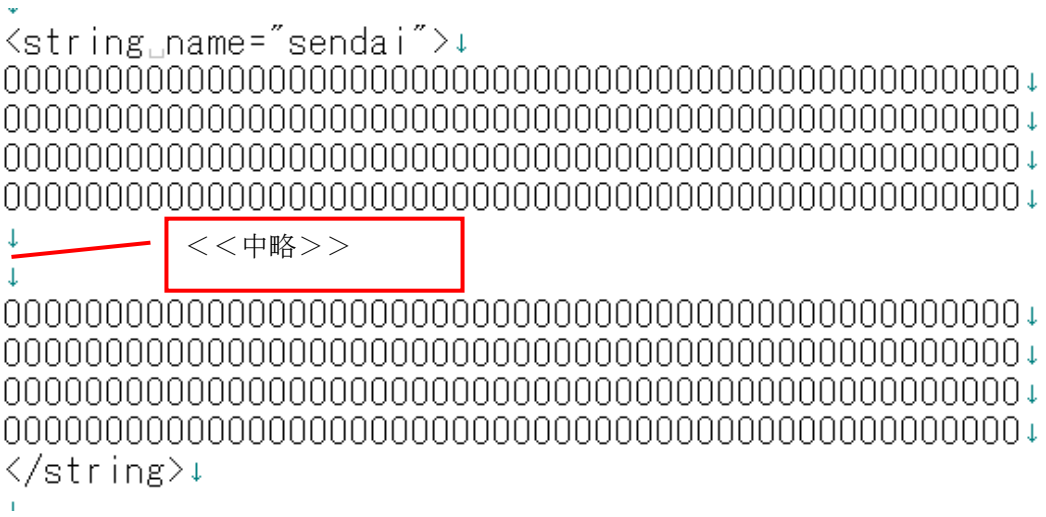


マップチップを８０×８０をマップエディタで作成（マップエディタ_Ver1.0.1.xlsx）

このエディタを利用すれば、マップデザインはとてもやり易くなります。また、後述するマップ間移動を実現する上で、「マップのどの座標に重なると、どこに移動するか」といった移動元⇔移動先の関連付けも、とてもやり易くなります。

エディタの利用について、１点だけ注意が必要です。このエディタからマップデータをコピーして貼り付けると、タブ文字(¥t)区切りのデータとして貼り付けられてしまいます。その為、貼り付けた後にタブ文字(¥t)を“”（空欄）に一括置換して区切り文字を全て取り除く事を忘れないで下さい。

以上の作業を行う事で、マップを作成する事が出来ます。実際に作成されたマップは、strins.xml 内に記述する事になります。



センダイマップの定義部例（strings.xml）

この方法で複数のマップを作製していきます。MiyagiQuest では、予め８種類のマップが作成してあります。

<string name="map">…<<中略>>…</string>　　：フィールドマップ
<string name="sendai">…<<中略>>…</string>　　：センダイの町
<string name=”sendai2">…<<中略>>…</string>　　：センダイの町（鍵入手後）
<string name="oosaki">…<<中略>>…</string>　　：オオサキの町
<string name="oosaki2">…<<中略>>…</string>　　：オオサキの町（にじのしずく入手後）
<string name="nanasi">…<<中略>>…</string>　　：ナナシ
<string name="lake">…<<中略>>…</string>　　：レイクダンジョン
<string name="lake2">…<<中略>>…</string>　　：レイクダンジョン（オーブ入手後）

似ている名前のマップが存在しているのは、イベントフラグ ON・OFF により、マップ内容を変更しているからです。詳細は、「３　フラグ処理」にて説明します。

２－２　移動先マップでの初期位置

２－１で複数のマップが準備出来たら、それらのマップ間を移動出来る様に処理を記述します。

```
    private void mapChange() throws Exception{
        Resources r = getResources();

        //TODO マップ間移動処理
        switch (map) {
            //フィールド**********************************************
            case R.string.map:
                //フィールド内にある入口は全てここで制御！
                if ((playerMapPosX == 46 && playerMapPosY == 48) ||
                    (playerMapPosX == 46 && playerMapPosY
                    (playerMapPosX == 46 && playerMapPosY
                    (playerMapPosX == 47 && playerMapPosY == 48) ||
                    (playerMapPosX == 47 && playerMapPosY == 49) ||
                    (playerMapPosX == 47 && playerMapPosY == 50))
                {
                    //センダイへ
                    map = R.string.sendai;
                    //カギ所持ならばマップ変更
                    if(FLAG_ITEM1 == 1)map = R.string.sendai2;
                    encFlag = false;    //モンスター発生せず
                    this.init(STATUS_MAP);
                    playerMapPosX = 27;playerMapPosY = 29;
                    helpMsgTxt = "センダイの町です。";
                    helpMsgTxtIndex = -2;

                }else if((playerMapPosX == 47 && playerMapPosY == 20) ||
```

現在マップの判断

現在座標の判断

map 値の代入変更

マップ間遷移を行っている処理部

(MiyagiQuestView.java@mapChange メソッド)

処理の流れは以下の通りです。mapMoved⇒mapChanges⇒mapMoved の流れで行われています。

① mapMoved メソッドにより指定方向に動けているか判定
② ①の判定により、正常に移動可能であれば内部的に mapChange メソッド呼出し
　(ア) 現在プレイヤーキャラクタが居るマップ毎に処理分岐
　(イ) 現在プレイヤーキャラクタが居るマップ内の座標位置が指定の位置であれば map の値を代入し直して変更
　(ウ) mapChange メソッドを抜ける
③ 再び mapMoved メソッドに戻り、
　this.initMapImage(new Canvas(mapImage));
　部分で現在の map 値でマップ内容を書き変える。map 値に変更が無ければ以前と同じ

マップ表示に。変更されていれば、変更後の map を表示します。

マップの特定座標に差し掛かると（左）
マップ間遷移が行われます（右）

移動に成功する為には、移動先マップの初期位置も設定して上げる必要が有ります。
なお、今回言う座標（X 座標，Y 座標）というのは、左上を（０，０）として、

右に行けば X 座標が増加
下に行けば Y 座標が増加

という、いわゆるコンピュータグラフィックスの座標軸と同じルールに基づいています。

```
    private void mapChange() throws Exception{
        Resources r = getResources();

        //TODO マップ間移動処理
        switch (map) {
            //フィールド**************************************
            case R.string.map:
                //フィールド内にある入口は全てここで制御！
                if ((playerMapPosX == 46 && playerMapPosY == 48) ||
                    (playerMapPosX == 46 && playerMapPosY == 49) ||
                    (playerMapPosX == 46 && playerMapPosY == 50) ||
                    (playerMapPosX == 47 && playerMapPosY == 48) ||
                    (playerMapPosX == 47 && playerMapPosY == 49) ||
                    (playerMapPosX == 47 && playerMapPosY == 50))
                {
                    //センダイへ
                    map = R.string.sendai;
                    //カギ所持ならばマップ変更
                    if(FLAG_ITEM1 == 1)map = R.string.sendai2;
                    encFlag = false;    //モンスター発生せず
                    this.init(STATUS_MAP);
                    playerMapPosX = 27;playerMapPosY = 29;
                    helpMsgTxt = "センダイの町です。";
                    helpMsgTxtIndex = -2;

                }else if((playerMapPosX == 47 && playerMapPosY == 20) ||
```

移動先マップの座標位置

もう一度、先ほど提示した mapChange メソッドの内部を見てみましょう。
mapChange メソッドでは、

(ア) 現在プレイヤーキャラクタが居るマップ毎に処理分岐
(イ) 現在プレイヤーキャラクタが居るマップ内の座標位置が指定の位置であれば map の値を代入し直して変更
(ウ) mapChange メソッドを抜ける

という流れで処理が行われていますが、(イ) の map 値の代入し直した後の処理を細かく見ていきましょう。

```
encFlag = false;  //モンスター発生せず
this.init(STATUS_MAP);
playerMapPosX = 27;playerMapPosY = 29;
helpMsgTxt = "センダイの町です。";
```

という処理が書かれています。
まず、encFlag というのは、モンスターを出現させるかどうかの判断フラグです。町の中でモンスターを発生させるのはあまり好ましくありませんので、このフラグを true にしておく事で、そのマップ内では戦闘が発生しない事になります。次の this.init(STATUS_MAP); の部分は、「ゲームステータスをマップにいる状態とする」という事なので、ここではあま

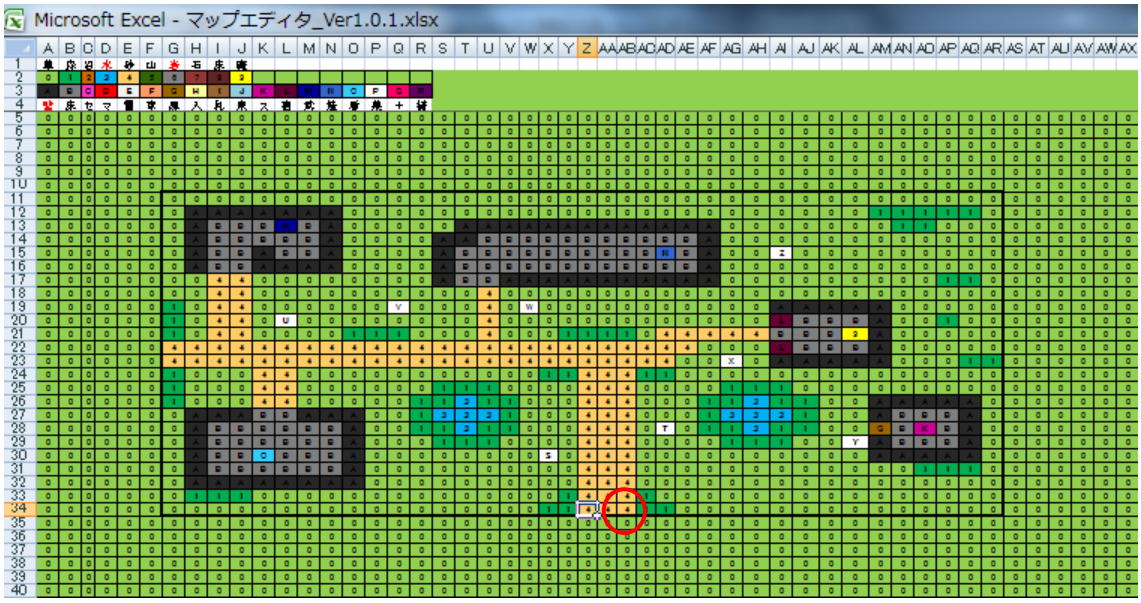
り気にしなくても良いです。次の

playerMapPosX = 27;

playerMapPosY = 29;

の２つの座標設定が、移動先のプレイヤーキャラクタ位置を決める値になります。

センダイマップの X 座標が２７、Y 座標が２９の位置というのは、下記に示す位置になります。



町に入って直ぐに、町の中央位置から始まるとプレイヤーは混乱してしまいますので、出入口付近に位置を指定するのが良いでしょう。

細かい座標位置は、マップエディタで細かく確認した上で設定し、一度テストプレイなどで確認すると良いと思います。１つ座標位置が変わっただけでもプレイしている側はとても違和感を覚えるからです。

また、移動した先がどこなのかをプレイヤーに教える意味で、helpMsgTxt = "センダイの町です。";という部分で、ヘルプメッセージを設定しています。色々な所でプレイヤーに対してアナウンスメッセージを出すことも、プレイを快適にして貰う上では重要な事です。

マップ遷移後の位置とヘルプメッセージを確認

２－３　移動元マップの移動フラグ発生位置

２－２でマップ間遷移の仕組みを理解したら、移動元⇔移動先の設定の詳細方法を見ていきます。基本的には、「プレイヤーキャラクタのＸ座標・Ｙ座標が、そのマップ内の指定位置Ｘ座標・Ｙ座標と重なったら（または指定範囲を超えたら）」という判断で行われています。

① フィールド　⇒　センダイ

フィールドからセンダイに移動する座標は６マス指定してあります。

Ｘ座標が４６で、かつ、Ｙ座標が４８、又は４９、又は５０の位置、

又は、

Ｘ座標が４７で、かつ、Ｙ座標が４８、又は４９、又は５０の位置、

のいずれかの位置にプレイヤーキャラクタの座標が重なると、センダイに移動したことになります。

MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

```
if ((playerMapPosX == 46 && playerMapPosY == 48) ||
        (playerMapPosX == 46 && playerMapPosY == 49) ||
        (playerMapPosX == 46 && playerMapPosY == 50) ||
        (playerMapPosX == 47 && playerMapPosY == 48) ||
        (playerMapPosX == 47 && playerMapPosY == 49) ||
        (playerMapPosX == 47 && playerMapPosY == 50))
{
```

② フィールド　⇒　オオサキ

フィールドからオオサキに移動する座標は２マス指定してあります。

X 座標が４７で、かつ、Y 座標が２０の位置、

又は、

X 座標が４７で、かつ、Y 座標が２１の位置、

のいずれかの位置にプレイヤーキャラクタの座標が重なると、オオサキに移動したことになります。

MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

}else if((playerMapPosX == 47 && playerMapPosY == 20) ||
(playerMapPosX == 47 && playerMapPosY == 21)){

③ フィールド ⇒ ナナシ

フィールドからナナシに移動する座標は１マスのみ指定してあります。
X 座標が１２で、かつ、Y 座標が１１の位置、
にプレイヤーキャラクタの座標が重なると、ナナシに移動したことになります。
センダイやオオサキと異なり、隠し入口的な扱いにしたかったので、外見は単なる砂漠のマップチップですが、移動してみると別マップへの入り口だったという事になっています。
MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

}else if(playerMapPosX == 12 && playerMapPosY == 11){

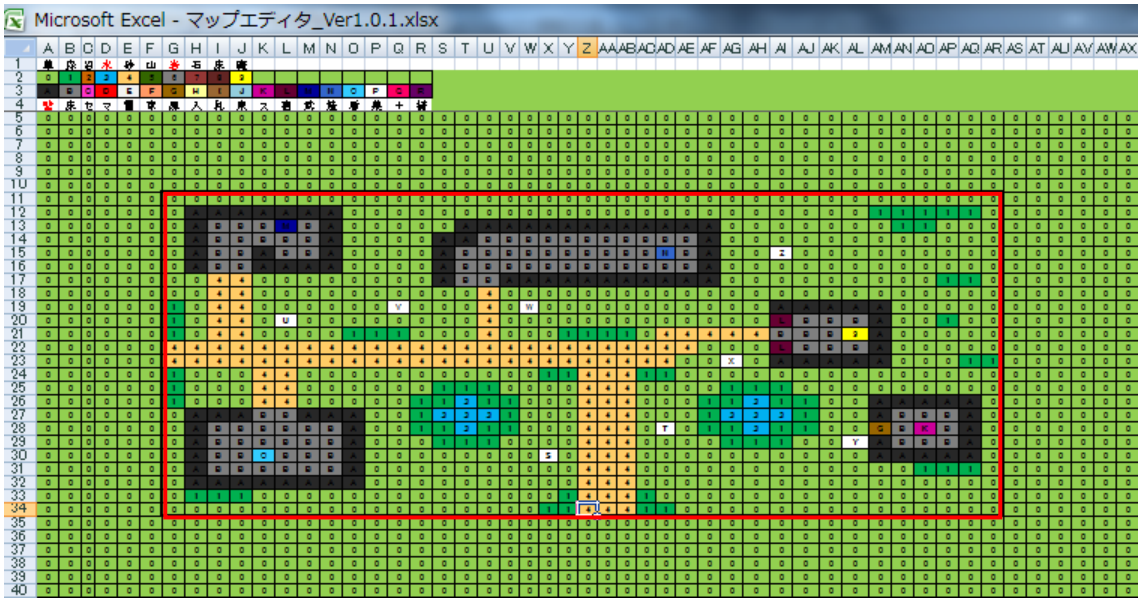
④ フィールド　⇒　レイクダンジョン

フィールドからレイクダンジョンに移動する座標も１マスのみ指定してあります。
　X 座標が１７で、かつ、Y 座標が５５の位置、
にプレイヤーキャラクタの座標が重なると、ナナシに移動したことになります。
ナナシと同様に、隠し入口的な扱いにしてあります。
MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

}else if(playerMapPosX == 17 && playerMapPosY == 55){

①~④までは全てフィールドから別マップへの移動処理を説明してきました。次に、各移動先マップからフィールドへの移動を説明します。

⑤ センダイ　⇒　フィールド



センダイからフィールドに移動する座標は範囲で指定してあります。

X 座標が 6 以下、又は、４３以上の位置、

又は、

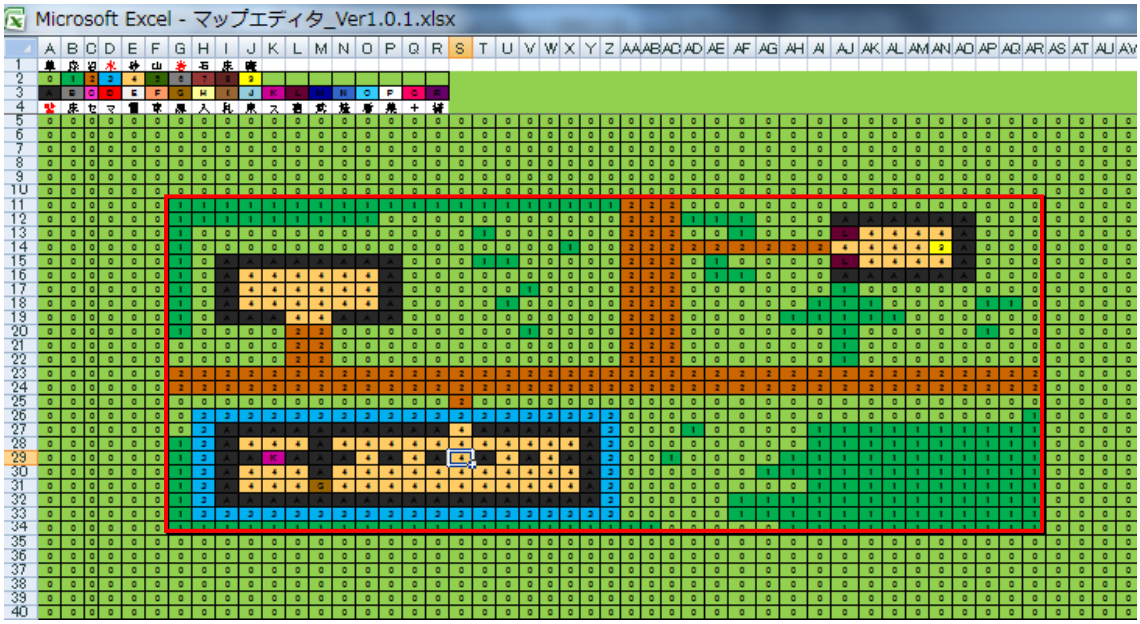
Y 座標が 6 以下、又は、Y 座標が３０以上の位置、

のいずれかの位置にプレイヤーキャラクタの座標が重なると、フィールドに移動したことになります。つまり、赤線の枠外に出ると移動、という事です。

MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

```
if ((playerMapPosX <=  6) ||
        (playerMapPosX >= 43) ||
        (playerMapPosY <=  6) ||
        (playerMapPosY >= 30))
{
```

⑥ オオサキ　⇒　フィールド



オオサキからフィールドに移動する座標は範囲で指定してあります。

X 座標が 6 以下、又は、４３以上の位置、

又は、

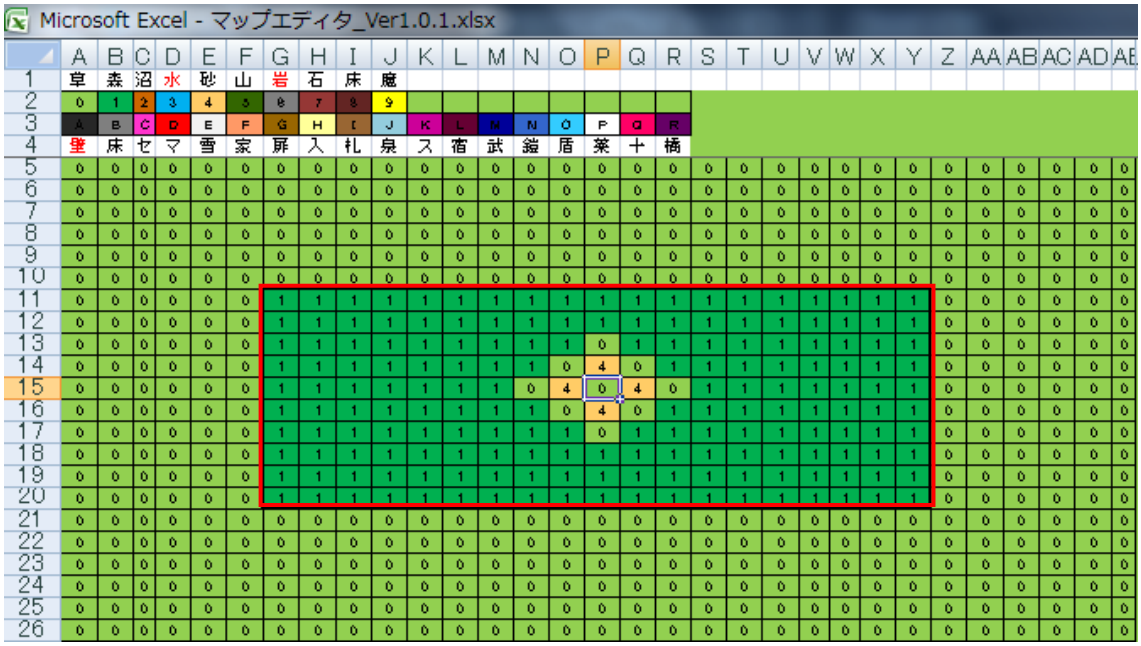
Y 座標が 6 以下、又は、Y 座標が３０以上の位置、

のいずれかの位置にプレイヤーキャラクタの座標が重なると、フィールドに移動したことになります。つまり、赤線の枠外に出ると移動、という事です。

MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

```
if ((playerMapPosX <=  6) ||
        (playerMapPosX >= 43) ||
        (playerMapPosY <=  6) ||
        (playerMapPosY >= 30))
{
```

⑦ ナナシ　⇒　フィールド



ナナシからフィールドに移動する座標は範囲で指定してあります。

X 座標が 6 以下、又は、２４以上の位置、

又は、

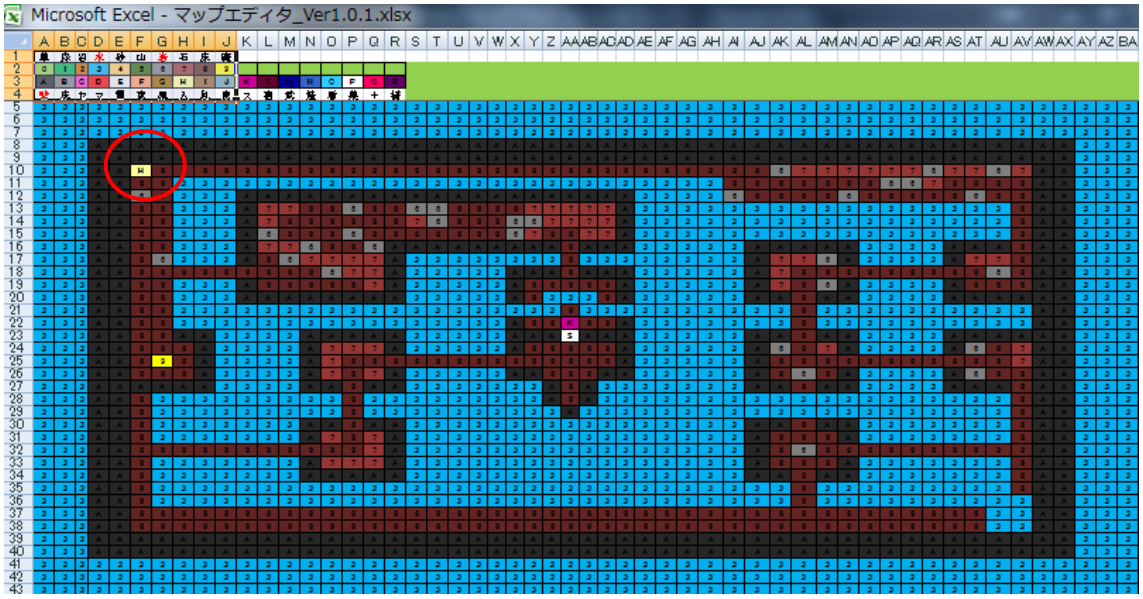
Y 座標が 6 以下、又は、Y 座標が１７以上の位置、

のいずれかの位置にプレイヤーキャラクタの座標が重なると、フィールドに移動したことになります。つまり、赤線の枠外に出ると移動、という事です。

MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

```
if ((playerMapPosX <=  6) ||
        (playerMapPosX >= 24) ||
        (playerMapPosY <=  6) ||
        (playerMapPosY >= 17))
{
```

⑧ レイクダンジョン　⇒　フィールド



レイクダンジョンからフィールドに移動する座標は１マスのみ指定してあります。

　X 座標が５、かつ、Y 座標が５の位置、

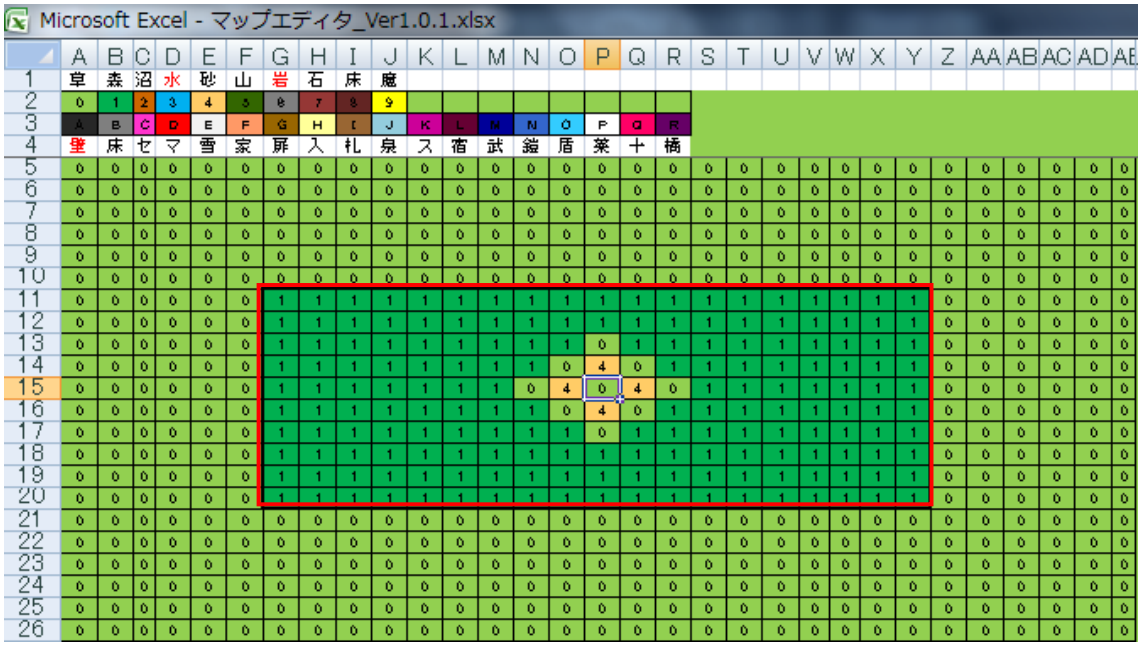
にプレイヤーキャラクタの座標が重なると、フィールドに移動したことになります。

MiyagiQuestView.java@mapChange メソッド内では、下記の様な分岐条件になっています。

```
if (playerMapPosX == 5 && playerMapPosY == 5)
{
```

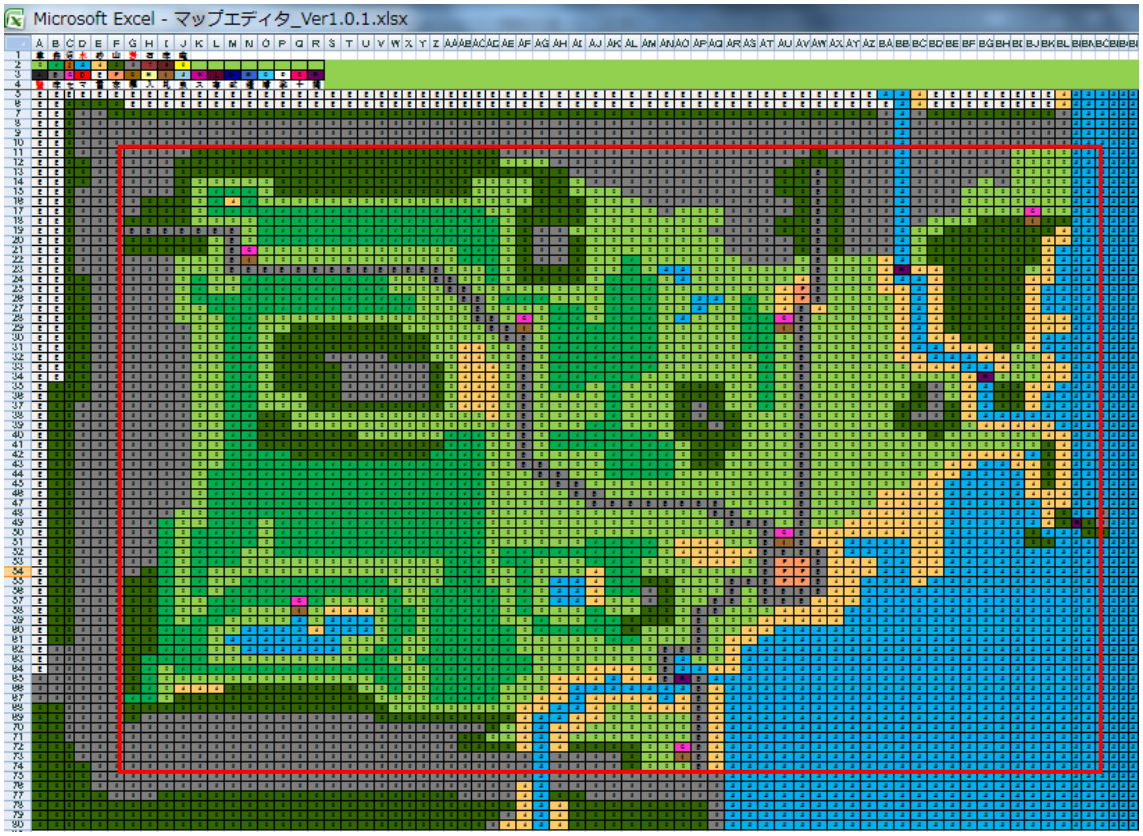
２－４　マップ作りにおける注意点

マップ間遷移を実装する上で、画面に入りきらない部分への配慮が必要になってきます。
フィールドマップも、センダイ等の各町マップ等も、マップを良く見ると外周に不要なマップチップが設定されている事が分かります。
ナナシマップを例に、もう一度確認してみると、



赤枠の外に６マス分の余計な部分が存在しています。
この部分の設定は必須ではありませんが、設定しないとマップの外は水フィールドで埋め尽くされてしまいます。標準の作りで、マップの範囲外は水フィールドとされているからです。
プレイヤーキャラクタは常に画面の中央に位置し、移動（マップの書き換え）を行っています。その位置からギリギリ画面内に入る範囲が６マス分となります。その為、町等の出入口ギリギリまでプレイヤーキャラクタを移動させた際に、６マス分余計に外周を作成しておく事でその外側の部分（水）部分を画面に入らない様にしているのです。

フィールドマップも同様です。今回、フィールドマップでは、移動不可なマスにより外周６マス以内に近づけない様にマップ作成時に工夫を行っています。



周囲が海のフィールドを除いて、概ね赤枠内を移動可能とする事で、その外側を表示しない様にしています。マップを拡げていく際の参考にして下さい。

３　フラグ処理

３－１　フラグ発生位置の設定

フラグとはオン・オフのスイッチの様なもので、そのイベント条件を満たしたか・満たさないかを判断できる値を保持する為に使用します。本来は、論理値(boolean 型:true、false)で値を保持する事が望ましいのですが、データ保存・読込時に余計な変換作業を行う事を避けたかった為、今回は整数値（int 型）の１、０でオン、オフを保持しています。

```
    private int FLAG_SENDAI_WEAPON = 0;          //  フラグ：武器↓
    private int FLAG_SENDAI_ARMOR = 0;           //  フラグ：鎧↓
    private int FLAG_SENDAI_SHIELD = 0;          //  フラグ：盾↓
    private int FLAG_ITEM1 = 0;                  //  フラグ：アイテム１↓
    private int FLAG_ITEM2 = 0;                  //  フラグ：アイテム２↓
    private int FLAG_ITEM3 = 0;                  //  フラグ：アイテム３↓
    private int FLAG_BOSS1 = 0;                  //  フラグ：ボスキャラ１↓
```

各種イベントフラグのフィールド

(MiyagiQuestView.java@フィールド)

FLAG_SENDAI_WEAPON　　：センダイの剣マップチップに対応するフラグ

FLAG_SENDAI_AROMOR　　：センダイの鎧マップチップに対応するフラグ

FLAG_SENDAI_SHIELD　　：センダイの盾マップチップに対応するフラグ

FLAG_ITEM1　　：ナナシに落ちているアイテム「カギ」に対応するフラグ

FLAG_ITEM2　　：センダイのアイテム「にじのしずく」に対応するフラグ

FLAG_ITEM3　　：オオサキのアイテム「オーブ」に対応するフラグ

FLAG_BOSS1　　：ボスバトルか否かを判断するフラグ

また、これらのフラグは、ゲームの開始・保存・読込時にもそれぞれ設定・再設定される必要がある為、それらの処理部分にも記述が影響してきます。

具体的には、MiyagiQuestView.java 内で、ゲーム開始処理を行う title メソッド内で、再開「Continue」を選択された時に読み込まれるタイミングと、セーブポイントでゲームの途中経過を保存する際に呼び出させる save メソッド内に記述が必要となります。

```
            // データがあれば、ステータスをMAPに設定し、初期化）
            setStatus(STATUS_MAP, true);
            try {
                // 保存データから、各値を初期化
                int k = 0;
                level = Integer.parseInt(data[k++]);
                maxHp = Integer.parseInt(data[k++]);
                hp = Integer.parseInt(data[k++]);
                maxSp = Integer.parseInt(data[k++]);
                sp = Integer.parseInt(data[k++]);
                str = Integer.parseInt(data[k++]);
                def = Integer.parseInt(data[k++]);
                spd = Integer.parseInt(data[k++]);
                exp = Integer.parseInt(data[k++]);
                nextExp = Integer.parseInt(data[k++]);
                map = Integer.parseInt(data[k++]);
                playerMapPosX = Integer.parseInt(data[k++]);
                playerMapPosY = Integer.parseInt(data[k++]);

                //TODO 各種イベントフラグ
                FLAG_SENDAI_WEAPON = Integer.parseInt(data[k++]);
                FLAG_SENDAI_ARMOR  = Integer.parseInt(data[k++]);
                FLAG_SENDAI_SHIELD = Integer.parseInt(data[k++]);
                FLAG_ITEM1  = Integer.parseInt(data[k++]);
                FLAG_ITEM2  = Integer.parseInt(data[k++]);
                FLAG_ITEM3  = Integer.parseInt(data[k++]);
```

Continue 時に各種イベントフラグを読み込んで値を再設定

（MiyagiQuestView.java@title メソッド）

```
        buf.append(",");
        buf.append(nextExp);        // 次の経験値
        buf.append(",");
        buf.append(map);            // マップ番号
        buf.append(",");
        buf.append(playerMapPosX);  // プレイヤーのマップ上のX座標
        buf.append(",");
        buf.append(playerMapPosY);  // プレイヤーのマップ上のY座標
        buf.append(",");

        //TODO 各種イベントフラグ
        buf.append(FLAG_SENDAI_WEAPON);
        buf.append(",");
        buf.append(FLAG_SENDAI_ARMOR);
        buf.append(",");
        buf.append(FLAG_SENDAI_SHIELD);
        buf.append(",");
        buf.append(FLAG_ITEM1);
        buf.append(",");
        buf.append(FLAG_ITEM2);
        buf.append(",");
        buf.append(FLAG_ITEM3);

        // 保存データをファイルに書き込む
        DebugLog.println("saveToFile(" + buf.toString() + ") to [" + fname + "]");
        try {
```

セーブポイントで各種イベントフラグの値を保存

（MiyagiQuestView.java@save メソッド）

今回の MiyagiQuest においては、下記の８つのフラグイベントを実装しています。

① センダイにおいて、一度だけ攻撃力を増加する事が出来る（ヘルプメッセージ有）
② センダイにおいて、一度だけ防御力を増加する事が出来る（ヘルプメッセージ有）
③ センダイにおいて、一度だけ素早さを増加する事が出来る（ヘルプメッセージ有）
④ ナナシにおいて、「カギ」を入手する
⑤ センダイにおいて「カギ」を所持していれば、「にじのしずく」を入手可能となる
⑥ オオサキにおいて「にじのしずく」を所持していれば、「オーブ」を入手可能となる
⑦ レイクダンジョンにおいて「オーブ」を所持していれば、ボスまで辿り着く事が出来る
⑧ ボスとの戦闘シーンに突入した後マップに戻ってきていれば（＝勝利している）「秘宝」を入手する事が出来る

３－２　イベント処理

①~③の各種ステータスアップイベントについては、NPC やセーブポイント、回復ポイント等と同様に、その座標上にプレイヤーキャラクタが重なると、自動的にヘルプメッセージを表示する様にしてあります。

```
        //マップオブジェクト上でのヘルプメッセージ
        int tileNo = filedMapInt[playerMapPosY][playerMapPosX];
        String act = (String)mapActionMap.get(String.valueOf(tileNo));
        if ("CURE".equals(act)) {
            helpMsgTxt = "画面タップ = HP・SP回復";
            helpMsgTxtIndex = -2;
        } else if ("SAVE".equals(act)) {
            helpMsgTxt = "画面タップ = セーブ";
            helpMsgTxtIndex = -2;
        } else if ("WEAPON".equals(act) || "ARMOR".equals(act) || "SHIELD".equals(act)) {
            helpMsgTxt = "画面タップ = ステータスアップ";
            helpMsgTxtIndex = -2;
        } else if ("NPC1".equals(act) ||
                   "NPC2".equals(act) ||
                   "NPC3".equals(act) ||
                   "NPC4".equals(act) ||
                   "NPC5".equals(act) ||
                   "NPC6".equals(act) ||
                   "NPC7".equals(act) ||
                   "NPC8".equals(act)){
            helpMsgTxt = "画面タップ = 会話";
            helpMsgTxtIndex = -2;
        }
    }
```

各種ステータスアップと重なった場合にヘルプメッセージを表示
（MiyagiQuestView.java@mapMoved メソッド）

各種ステータスアップと重なった場合にヘルプメッセージを表示する例

また、ボスキャラ戦闘に勝利したかどうかのフラグ⑧以外は、１章の NPC の設定「１－３会話時のセリフ」と同様の場所、すなわち MiyagiQuestView.java の map メソッド内に処理の大半が書かれています。

現在マップの判断

マップ毎の分岐

```
//TODO 各種タッチイベント
switch (map) {
    //センダイ**********
    case R.string.sendai:
    case R.string.sendai2:
        if ("NPC1".equals(act))
            /*S*/ helpMsgTxt = "「ここはセンダイの町！秘                まる町だ！」";
        if ("NPC2".equals(act))
            /*T*/ helpMsgTxt = "「レベルに応じて強いモンスターが出るぜっ！」";
        if ("NPC3".equals(act))
            /*U*/ helpMsgTxt = "「武器や防具はタップするだけで入手可能…」";
        if ("NPC4".equals(act))
            /*V*/ helpMsgTxt = "「秘宝はどこにあるのだろう？？」";
        if ("NPC5".equals(act))
            /*W*/ helpMsgTxt = "「レベル上げは重要だな！」";
        if ("NPC6".equals(act))
            /*X*/ helpMsgTxt = "「宿屋ではHP・SPがタダで回復できるぞっ！」";
        if ("NPC7".equals(act))
            /*Y*/ helpMsgTxt = "「北の方に町が有ると聞いたが…」";
        if ("NPC8".equals(act))
            /*Z*/ helpMsgTxt = "「俺が最初に秘宝を入手してやるっ！」";
        if ("WEAPON".equals(act)){
            if(FLAG_SENDAI_WEAPON == 0) {
                this.str = this.str + this.level + 10;
                FLAG_SENDAI_WEAPON = 1;
                helpMsgTxt = "攻撃力がアップ！";
                try {
```

マップ毎に各種フラグイベントの処理が書かれている
（MiyagiQuestView.java@map メソッド）

それでは、一つ一つのフラグイベントを見ていきましょう。

３－２－１　センダイ：攻撃力アップイベント

```
        if ("WEAPON".equals(act)){
            if(FLAG_SENDAI_WEAPON == 0) {
                this.str = this.str + this.level + 10;
                FLAG_SENDAI_WEAPON = 1;
                helpMsgTxt = "攻撃力がアップ！";
                try {
                    MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else {
                helpMsgTxt = "効果がないようだ";
            }
            helpMsgTxtIndex = -2;
        }
```

ここまで処理が到達しているという事は、少なくともプレイヤーキャラクタはセンダイマップ内に居る事が保障されています。後は何のマップアクションか、既に一度実行されているか、等を判断していけば良いのです。

①□マップアクションが WEAPON（武器）かどうかを判定

②①が真であるなら FLAG_SENDA_WEAPON が０．すなわち既に攻撃力アップを行ったかどうかを判定

③②が真であれば、効果音と共に、ステータスアップ処理（現在の攻撃力値に、現在レベル値＋１０を加算する）を行い、メッセージを表示、偽であれば、既に一度ステータスアップが行われている為、「効果がないようだ」というメッセージを表示

武器マップチップと重なる事で、ヘルプメッセージが自動的に表示される

タップする事で、効果音と共に「攻撃力がアップ！」の表示。

実際に攻撃力が増加している

１度実行したステータスアップイベントは

再実行出来ない

これでステータスアップイベントの流れが理解出来ました。続けて、防御力アップイベント、素早さアップイベントも見ていきましょう。

３－２－２　センダイ：防御力アップイベント

```
            if␣("ARMOR".equals(act)){↓
                if(FLAG_SENDAI_ARMOR␣==␣0)␣{↓
                    this.def␣=␣this.def␣+␣this.level␣+␣10;↓
                    FLAG_SENDAI_ARMOR␣=␣1;↓
                    helpMsgTxt␣=␣"防御力がアップ！";↓
                    try␣{↓
                        MyUtil.PlaySound_SE(activity,␣MyUtil.SE_CURE);↓
                    }␣catch␣(Exception␣e)␣{↓
                        e.printStackTrace();↓
                    }↓
                }␣else␣{↓
                    helpMsgTxt␣=␣"効果がないようだ";↓
                }↓
                helpMsgTxtIndex␣=␣-2;↓
            }↓
```

基本的には、３－２－１と同様の処理となる為、スクリーンショットは割愛します。

① マップアクションが ARMOR（鎧）かどうかを判定
② ①が真であるなら FLAG_SENDA_AROMOR が０．すなわち既に防御力アップを行ったかどうかを判定
③ ②が真であれば、効果音と共に、ステータスアップ処理（現在の防御力値に、現在レベル値＋１０を加算する）を行い、メッセージを表示、偽であれば、既に一度ステータスアップが行われている為、「効果がないようだ」というメッセージを表示

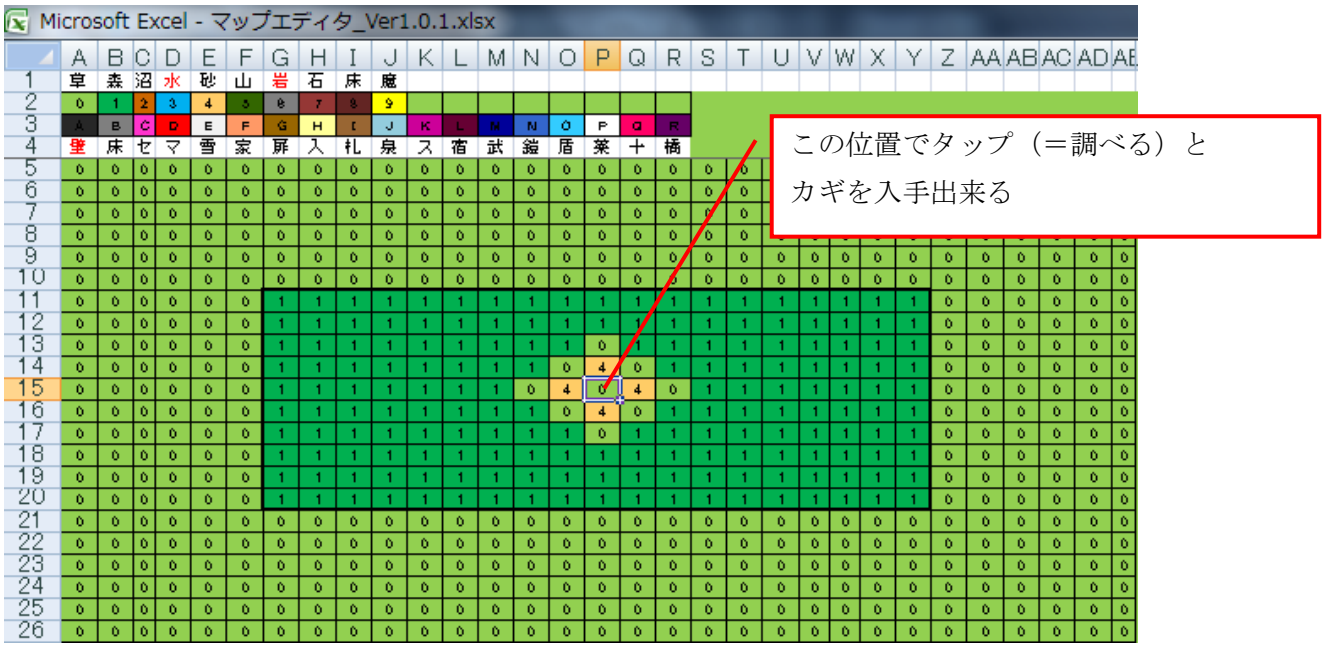
３－２－３　センダイ：素早さアップイベント

```
            if (″SHIELD″.equals(act)){
                if(FLAG_SENDAI_SHIELD == 0) {
                    this.spd = this.spd + this.level + 10;
                    FLAG_SENDAI_SHIELD = 1;
                    helpMsgTxt = ″素早さがアップ！″;
                    try {
                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                } else {
                    helpMsgTxt = ″効果がないようだ″;
                }
                helpMsgTxtIndex = -2;
            }
```

こちらも基本的には、３－２－１、３－２－２と同様の処理となります。
① マップアクションが ARMOR（盾）かどうかを判定
② ①が真であるなら FLAG_SENDA_SHIELD が０．すなわち既に素早さアップを行ったかどうかを判定
③ ②が真であれば、効果音と共に、ステータスアップ処理（現在の素早さ値に、現在レベル値＋１０を加算する）を行い、メッセージを表示、偽であれば、既に一度ステータスアップが行われている為、「効果がないようだ」というメッセージを表示

３－２－４　ナナシ：カギの入手

先述のステータスアップイベントとは異なり、こちらはプレイヤーに「怪しい所があるから調べてみよう」という気持ちで、「周辺を探させる」という行為を誘発する、「隠されたアイテムを発見するイベント」となっています。



それでは、実際のコードを見ていきましょう。

```
                case R.string.nanasi:↓
                    helpMsgTxt = "…怪しい！";↓
                    if (playerMapPosX == 15 && playerMapPosY == 10){↓
                        if ("GRASS".equals(act)){↓
                            if(FLAG_ITEM1 != 1){↓
                                helpMsgTxt = "";↓
                                sleep(500);↓
                                helpMsgTxt = "□□カギを手に入れた！！";↓
                                FLAG_ITEM1 = 1;↓
                                try {↓
                                    MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);↓
                                }catch (Exception e) {↓
                                    e.printStackTrace();↓
                                }↓
                            }else{↓
                                helpMsgTxt = "…もう何も無い。";↓
                            }↓
                        }↓
                    }↓
                    break;  //CAUTION!↓
            }↓
```

ここまで処理が到達しているという事は、少なくともプレイヤーキャラクタはナナシマッ

プ内に居る事が保障されています。後は何のマップアクションか、既に一度実行されているか、等を判断していけば良いのですが、今回はマップアクションだけの指定だと上手くいきません。というのも。外見上は他のマップチップと同様の草原（GRASS）に、イベントを仕込んでいるからです。その為、下記の様な流れになります。

① プレイヤーキャラクタが、ナナシマップ内のどの座標位置でタップ（調べる）を行っても、メッセージ「…怪しい！」をヒントとして表示する
② プレイヤーキャラクタが指定の座標位置（ナナシマップの中心）にいるかどうか
③ マップアクションが GRASS（草原）の位置でタップ（調べる）されているかどうか
④ ②も③も共に真であるなら FLAG_ITEM_1 が０．すなわち既にカギを所持しているかどうかを判定
⑤ ④が真であれば、効果音と共に、メッセージを表示し、FLAG_ITEM_1 を１（入手済）に変更する。偽であれば、既にカギを所持している為、「…もう何も無い。」というメッセージを表示。

画面タップ（調べる）をする事で、
この周辺に何かある事を示唆するヒントメッセージを表示する

指定座標を画面タップ（調べる）をする事で、
カギを入手する事が可能

同じ場所を調べてももう何も入手する物は無い。

３－２－５　センダイ：にじのしずくの入手

こちらのイベントは、プレイヤーに「目に見えるけど入手する為には何かが必要」という事が、予め分かっているイベントになります。「どこかで何らかのアイテム入手、又はイベントを終わらせてから入手する」という事を、プレイヤーに示唆しているイベントになります。

最初から見えているけど、このままでは入手出来ない

今回、実際に入手する為には、３－２－４で説明した「カギ」が必要になります。本来ですと、アイテム⇒カギを使用…等と選んで、扉が開き、アイテムを入手する事が出来る…というのが理想の形なのですが、今回の実装ではアイテムを使用するというコマンドが存在しません。その為、カギを所持していれば、マップそのものを書き変えてしまう事で、アイテム入手を実現しています。

つまり、

カギ無し（FLAG_ITEM1 が 0）　⇒　マップに扉があるセンダイ(sendai)

カギ有り（FLAG_ITEM1 が 1）　⇒　マップから扉が消えているセンダイ(sendai2)

という事です。
実際のコードは下記の様になります。最初は、マップを作りかえている所を見てみましょう。マップ間遷移で使用している MiyagiQuestView.java の mapChange メソッドになります。フィールドからセンダイに入る時に、センダイ（sendai）マップを用意している処理になるのですが、この時、カギ所持で有れば、センダイ２（sendai2）マップを用意するようにしてあります。

```
        //TODO マップ間移動処理
        switch (map) {
            //フィールド**************************************
            case R.string.map:
                //フィールド内にある入口は全てここで制御！
                if ((playerMapPosX == 46 && playerMapPosY == 48) ||
                    (playerMapPosX == 46 && playerMapPosY == 49) ||
                    (playerMapPosX == 46 && playerMapPosY == 50) ||
                    (playerMapPosX == 47 && playerMapPosY == 48) ||
                    (playerMapPosX == 47 && playerMapPosY == 49) ||
                    (playerMapPosX == 47 && playerMapPosY == 50))
                {
                    //センダイへ
                    map = R.string.sendai;
                    //カギ所持ならばマップ変更
                    if(FLAG_ITEM1 == 1)map = R.string.sendai2;
                    encFlag = false;    //モンスター発生せず
                    this.Init(STATUS_MAP);
                    playerMapPosX = 27;playerMapPosY = 29;
                    helpMsgTxt = "センダイの町です。";
                    helpMsgTxtIndex = -2;

```

FLAG_ITEM1 が１であれば、扉無しの別マップを用意
（MiyagiQuestView.java@mapChange メソッド）

```
0000000AAABBAAA00133310004440001333100ABBBA0000000
0000000ABBBBBBA00113110004440T01131104GBKBA0000
0000000ABBBBBBA00011100004440000111100YABBBA0000
```

G：扉は、通行不可です

センダイ（sendai）マップの１部（strings.xml）

↓

```
0000000AAABBAAA00133310004440001333100ABBBA0000000
0000000ABBBBBBA00113110004440T01131104BBKBA0000000
0000000ABBBBBBA00011100004440000111100YABBBA0000000
```

センダイ（sendai2）マップの１部（strings.xml）

後は、他のイベントと同様に、MiyagiQuestView.java の map メソッド内に入手する為の処理を記述すれば良い事になります。

```
                if ("TREASURE".equals(act)){
                    if(FLAG_ITEM2 != 1){
                        helpMsgTxt = "";
                        sleep(500);
                        helpMsgTxt = "□□にじのしずくを手に入れた！！";
                        FLAG_ITEM2 = 1;
                        try {
                            MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    }else{
                        helpMsgTxt = "…もう何も無い。";
                    }
                }
                break;  //CAUTION!
```

ここまで処理が到達しているという事は、少なくともプレイヤーキャラクタはセンダイマップ内に居る事が保障されています。後は何のマップアクションか、既に一度実行されているか、等を判断していけば良いのです。

① マップアクションが TRESURE（宝）かどうかを判定

② ①が真であるなら FLAG_ITEM2 が０．すなわち既ににじのしずくを入手したかどうかを判定

③ ②が真であれば、効果音と共に、メッセージを表示し、FLAG_ITEM_2 を１（入手済）に変更する。偽であれば、既ににじのしずくを所持している為、「…もう何も無い。」というメッセージを表示。

初めてセンダイに立ち寄ったプレイヤーは
「何かが必要なのだろう」と推測しながら冒険に出ます

冒険しながらナナシにてカギを入手
その後再びセンダイに戻ってみると…

マップが書き変わって扉が消えている為、
怪しい場所を調べる事が出来ます

タップ操作で調べてみる事で、

無事ににじのしずくを入手する事が出来ました

このアイテムフラグは、オオサキの町で使用する事になります

一度入手してしまえば、同じ場所を調べても何もありません

３－２－６　オオサキ：オーブの入手

こちらのイベントも、３－２－５と同様で「どこかで何らかのアイテム入手、又はイベントを終わらせてから入手する」という事を、プレイヤーに示唆しているイベントになります。

最初から見えているけど、このままでは入手出来ない
NPC から「対岸に渡る」という発想のヒントを得る事になります

今回、実際に入手する為には、３－２－５で説明した「にじのしずく」が必要になります。カギを所持時と同様に、にじのしずくを所持していれば、マップそのものを書き変えてしまう事で、アイテム入手を実現しています。
つまり、
　にじのしずく無し（FLAG_ITEM2 が 0）　⇒　対岸に渡れないオオサキ(oosaki)
　にじのしずく有り（FLAG_ITEM2 が 1）　⇒　対岸に渡れる事が出来るオオサキ(oosaki2)
という事です。

実際のコードは下記の様になります。最初は、マップを作りかえている所を見てみましょう。こちらも、カギと同様にマップ間遷移で使用している MiyagiQuestView.java の mapChange メソッドになります。フィールドからオオサキに入る時に、オオサキ（oosaki）マップを用意している処理になるのですが、この時、にじのしずく所持で有れば、オオサキ２（oosaki2）マップを用意するようにしてあります。

```
        }else if((playerMapPosX == 47 && playerMapPosY == 20) ||
                (playerMapPosX == 47 && playerMapPosY == 21)){
            //オオサキへ
            map = R.string.oosaki;
            //にじのしずく所持ならばマップ変更
            if(FLAG_ITEM2 == 1)map = R.string.oosaki2;
            encFlag = false;    //モンスター発生せず
            this.init(STATUS_MAP);
            playerMapPosX = 28;playerMapPosY = 7;
            helpMsgTxt = "オオサキの町です";
            helpMsgTxtIndex = -2;
```

FLAG_ITEM2 が１であれば、対岸に渡る事が出来る別マップを用意

（MiyagiQuestView.java@mapChange メソッド）

```
000000000000000000U2000000000000000000000W00000000
00000003333333333333333333300000000000000000
00000003AAAAAAAAAA4AAAAAA3000100001111111110000
00000013A444A44444444444A3000000001111111110000
00000013AAKAAA4A4A4A4A4AA3001000011111111110000
```

３：水は、通行不可です

オオサキ（oosaki）マップの１部（strings.xml）

↓

```
000000000000000000U2000000000000000000000W00000000
00000003333333333343333333000000000000000000010000
00000003AAAAAAAAAA4AAAAAA3000100001111111110000
00000013A444A44444444444A3000000001111111110000
00000013AAKAAA4A4A4A4A4AA3001000011111111110000
```

オオサキ（oosaki2）マップの１部（strings.xml）

後は、他のイベントと同様に、MiyagiQuestView.java の map メソッド内に入手する為の処理を記述すれば良い事になります。

```
            if ("TREASURE".equals(act)){
                if(FLAG_ITEM3 != 1){
                    helpMsgTxt = "";
                    sleep(500);
                    helpMsgTxt = "□□オーブを手に入れた！！";
                    FLAG_ITEM3 = 1;
                    try {
                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }else{
                    helpMsgTxt = "…もう何も無い。";
                }
            }
            break;  //CAUTION!
```

ここまで処理が到達しているという事は、少なくともプレイヤーキャラクタはオオサキマップ内に居る事が保障されています。後は何のマップアクションか、既に一度実行されているか、等を判断していけば良いのです。

④ マップアクションが TRESURE（宝）かどうかを判定

⑤ ①が真であるなら FLAG_ITEM3 が０．すなわち既にオーブを入手したかどうかを判定

⑥ ②が真であれば、効果音と共に、メッセージを表示し、FLAG_ITEM_3 を１（入手済）に変更する。偽であれば、既にオーブを所持している為、「…もう何も無い。」というメッセージを表示。

NPC のヒントを頼りに、

プレイヤーはこれからナナシでカギを入手する事になります

センダイでカギを使用してにじのしずくを入手していれば
対岸への道が出来ています

無事にオーブを入手する事が出来ました
この後、プレイヤーはNPCのヒントを頼りに
最後のレイクダンジョンに向かう事になります

３－２－７　ボスキャラまでの道

こちらのイベントは、今回の MiyagiQuest における最終目的「秘宝」入手まで繋がるイベントとなります。カギ、にじのしずく、オーブとフラグアイテムを 3 種類入手してきたプレイヤーは、レイクダンジョンの再深部まで到達する事が可能になっており、そこでのボス線に勝利する事で、エンディングを迎える事になります。仮にボスキャラに負けてしまうようであれば、プレイヤーの強さが足らないという事になり、もう少し雑魚モンスターを倒してレベルを上げる必要があるかもしれません。

最終目的地レイクダンジョンは、
その入口も一見すると見落としてしまう場所に隠されています

初めて到達したプレイヤーは、
途中で道が途切れている為、引き返す事になります

今回、ダンジョンの再深部に進む為には、３－２－６で説明した「オーブ」が必要になります。他のフラグアイテムと同様に、オーブを所持していれば、マップそのものを書き変えてしまう事で、先に進める様になっています。
つまり、
　オーブ無し（FLAG_ITEM3 が 0）　⇒　途中で道が無くなるレイクダンジョン(lake)
　オーブ有り（FLAG_ITEM3 が 1）　⇒　ボスまで到達出来るレイクダンジョン(lake2)
という事です。
実際のコードは下記の様になります。最初は、マップを作りかえている所を見てみましょう。こちらも、他のフラグアイテムと同様にマップ間遷移で使用している MiyagiQuestView.java の mapChange メソッドになります。フィールドからレイクに入る時に、レイク（lake）マップを用意している処理になるのですが、この時、オーブ所持で有れば、レイク２（lake2）マップを用意するようにしてあります。

```
        }else if(playerMapPosX == 17 && playerMapPosY == 55){
            //レイクへ
            map = R.string.lake;
            //オーブ所持ならばマップ変更
            if(FLAG_ITEM3 == 1)map = R.string.lake2;
            encFlag = true; //モンスター発生
            this.init(STATUS_MAP);
            playerMapPosX = 7;playerMapPosY = 5;
            helpMsgTxt = "レイク・ダンジョン";
            helpMsgTxtIndex = -2;
        }
        break;  //CAUTION!
```

FLAG_ITEM3 が１であれば、ボスまで到達出来る別マップを用意

（MiyagiQuestView.java@mapChange メソッド）

```
3333333333333333333333333333333333333333333333333333333333
333AAAAAAA333AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA333
333AAAAAAAA333AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA333
333AAH888833388888888888888888888888677777767767AA333
```

３：水は、通行不可です

レイク（lake）マップの１部（strings.xml）

↓

```
3333333333333333333333333333333333333333333333333333333333
333AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA333
333AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA333
333AAH888888888888888888888888888677777767767AA333
```

レイク（lake2）マップの１部（strings.xml）

後は、他のイベントと同様に、MiyagiQuestView.java の map メソッド内にボス戦、及び秘宝を入手する為の処理を記述すれば良い事になります。

```
            //レイク****************************************↓
            case R.string.lake:↓
            case R.string.lake2:↓
                if ("NPC1".equals(act))↓
                    /*S*/ helpMsgTxt = "「…ヒホウハオレノモノダッ！！」";↓
                if ("TREASURE".equals(act) && FLAG_BOSS1 == 1){↓
                    helpMsgTxt = "";↓
                    sleep(500);↓
                    helpMsgTxt = "□□伝説の秘宝を手に入れた！！";↓
                    try {↓
                        MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);↓
                        setStatus(STATUS_ENDING, true);↓
                        init(STATUS_ENDING);↓
                    } catch (Exception e) {↓
                        e.printStackTrace();↓
                    }↓
                }↓
                if ("TREASURE".equals(act) && FLAG_BOSS1 == 0){↓
                    FLAG_BOSS1 = 1;//ボスキャラフラグ↓
                    TouchStatus = -1;↓
                    setStatus(STATUS_BATTLE, true);//   戦闘シーンへ↓
                }↓
                break;↓
```

ここまで処理が到達しているという事は、少なくともプレイヤーキャラクタはレイクマップ内に居る事が保障されています。後は何のマップアクションか、ボス戦前なのか後なのか等を判断していけば良いのです。

① マップアクションが NPC1（ボスキャラクタ）であるかどうかを判定
② ①が真であるならメッセージを表示

③ マップアクションが TREASURE（秘宝）で、かつ、FLAG_BOSS の値が 1（ボス戦後であるかどうかを判定
④ ③が真であるならば、効果音と共に、メッセージを表示しゲームステータスをエンディングに変更します

⑤ マップアクションが TREASURE（秘宝）で、かつ、FLAG_BOSS1 の値が 0（ボス戦前であるかどうかを判定
⑥ ⑤が真であるならば、FLAG_BOSS1 の値を 1 に変更し、ゲームステータスをバトル（戦闘）に変更します

今回のイベントは、ボスキャラクタとの戦いやエンディングをイベントに含んでいる為、他処理への影響もあります。FLAG_BOSS1 というイベントフラグによって、出現するモンス

ターのボスモンスター固定や、戦闘時に「逃げる」コマンドを非表示&選択不可にしています。下記に、それらのコードをまとめて記しておきます。

```
        if(FLAG_BOSS1 == 1) k = 1;      //ボスキャラ用
↓
        enemy = new Enemy[k];↓
↓
        //  戦闘メッセージを初期化します。↓
        battleMsgTxts = new String[4];↓
        battleMsgMasterTxts = new String[k];↓
↓
        //  敵の情報とメッセージを初期化します。↓
        for (int i = 0; i < enemy.length; i ++) {↓
↓
            //レベルに応じた敵の選出↓
            //TODO モンスターの出現率などのバランスを調節！↓
            k = MyUtil.nextInt(level);  //レベルで乱数調節！↓
            if(FLAG_BOSS1 == 1) k = 99;      //ボスキャラ用↓
            String txt="";↓
            switch(k){↓
                case 0: case 1:↓
                    txt = r.getString(R.string.enemy_1);break;↓
```

敵モンスター数を１に、敵モンスター種類をボスキャラクタ（99）に固定しています

（MiyagiQuestView.java@init メソッド）

```
            } else if (battleMenuSelectedIndex == 3) {↓
                //  「逃げる」が選択されました。↓
                //  今回は、必ず、逃げれます。↓
                //  ただし、ボス戦時は選択不可とする！↓
                if(FLAG_BOSS1 != 1){↓
                    TouchStatus = -1;↓
                    battleEnemySelectedIndex = -1;↓
                    sleep(500);↓
                    setStatus(STATUS_MAP, false);   //  地図へ。↓
                    return;↓
                }↓
            }↓
```

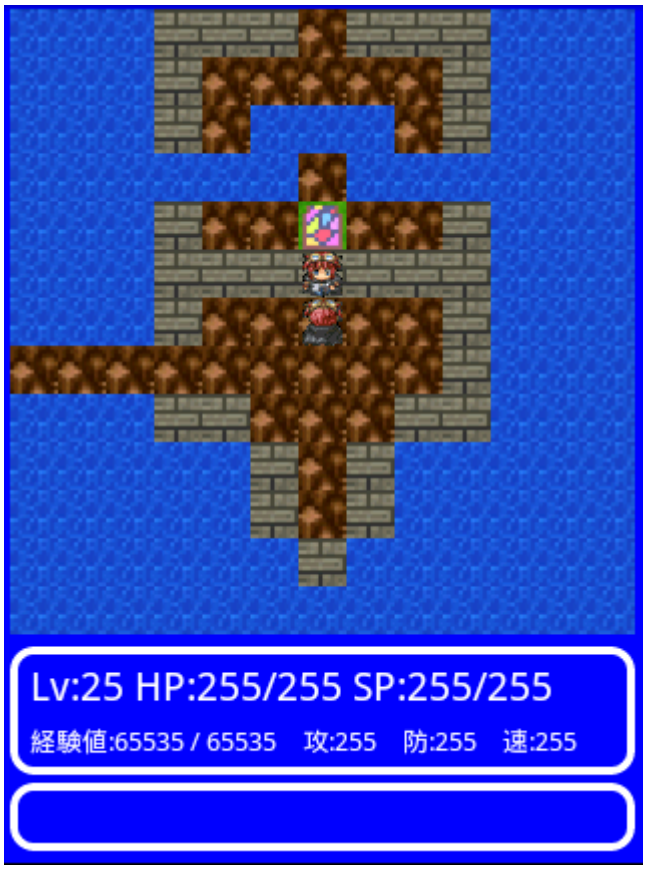
戦闘時における「逃げる」コマンドを選択不可としています

（MiyagiQuestView.java@battle メソッド）

```
    //ボス戦時は選択不可とする！↓
    if(FLAG_BOSS1 != 1){↓
        canvas.drawText("逃げる", displayWidth / 2 + 45,↓
                        displayHeight / 2 + 50 + battleMarginY, paint);↓
    }↓
    canvas.drawText("HP  : " + hp + "/" + maxHp,  mapMarginX + 10, statusDrawPosY + 90, paint
    canvas.drawText("SP  : " + sp + "/" + maxSp,  mapMarginX + 150, statusDrawPosY + 90, pain
```

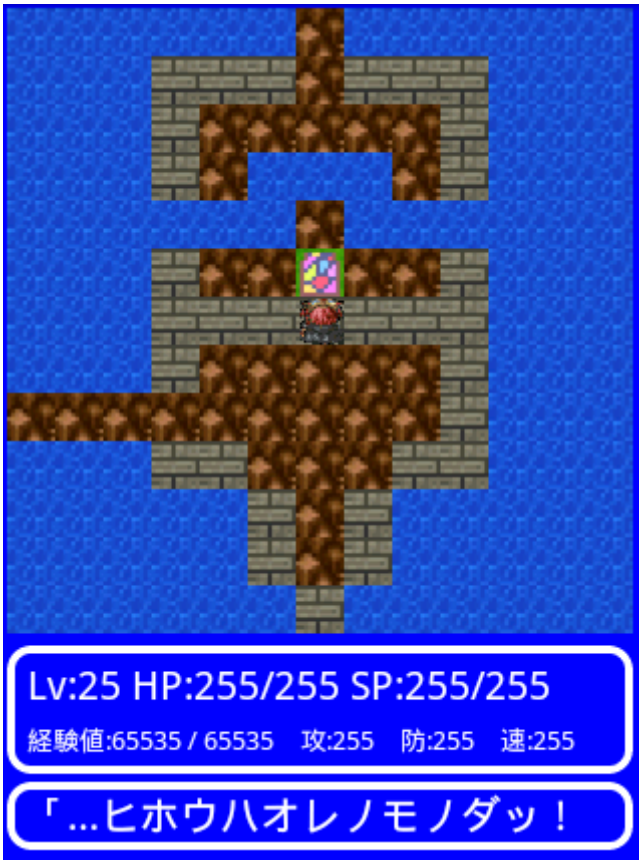
戦闘時における「逃げる」コマンドを非表示としています

（MiyagiQuestView.java@battle メソッド）



オーブの使用で道は拓け、

いよいよボスキャラクターとご対面です



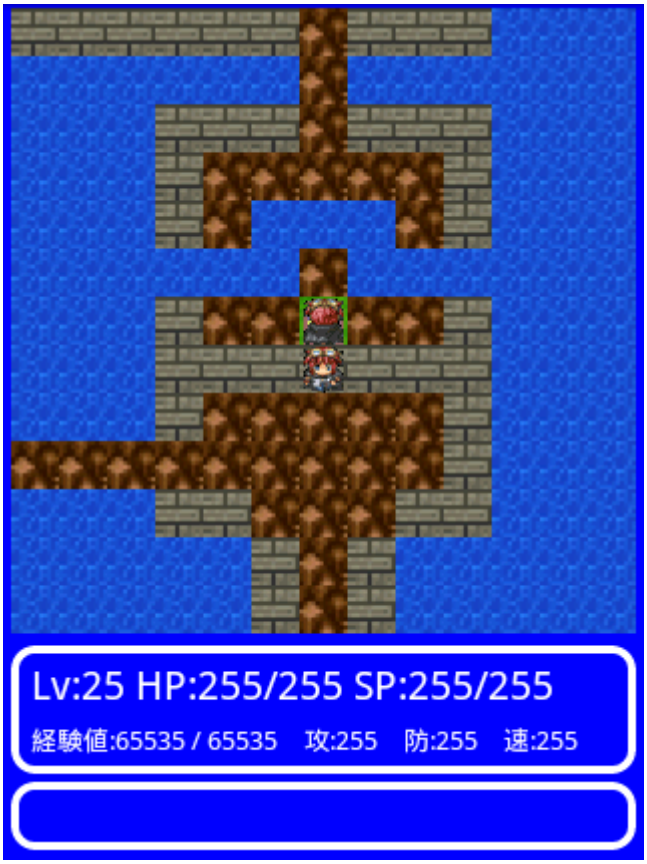
NPC との会話も用意しています

強制バトルは秘宝を入手しようとした瞬間

（TRESURE をタップ）で行われます

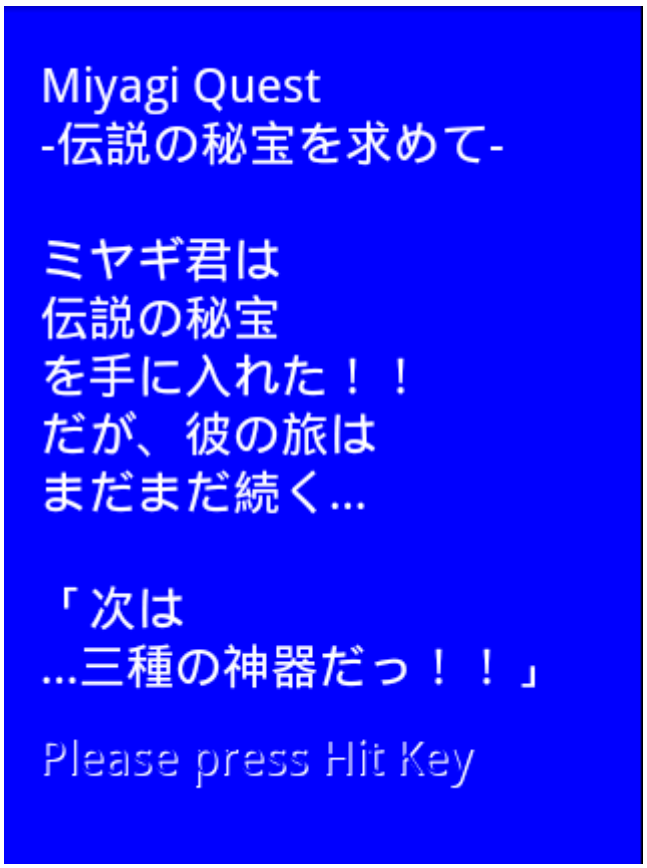
「逃げる」が
選択不可＆非表示と
なっています

秘宝を入手しようと画面をタップすると
いよいよボスキャラクタとのバトルです
最後の戦いから「逃げる」事は許されません



無事に勝利する事が出来ました

改めて秘宝をタップ操作で入手しようとすると…



もうボスキャラクタとの戦いはありません
無事にエンディングとなります

３－３　フラグ処理に関する注意点

今回の MiyagiQuest では、大きく分けて 2 種類のフラグイベントを作成してあります。

画面上に表示されている物に対してアクションを起こす事で発生するイベント
（アイテム入手等の）フラグの値によって結果が自動的に変更されるイベント

イベント内容や演出等によって、上記 2 つを使い分けると良いでしょう。

合わせて、今回の仕様では、プレイヤー自身にはフラグが見えていません。通常のゲームですと、何らかのアイテムを入手すれば、メニューコマンド等で自分の所持しているアイテム等を確認する事が出来る筈です。そうしたメニュー画面やアイコン表示などで、プレイヤーにフラグを表現して上げると、よりゲームとしての完成度は高まる事でしょう。

また、ボスキャラクタとの戦い等、「倒さないと先に進めないイベント」を設定する場合は、開発側は常に「バランス」を意識する必要が有ります。ボスキャラクタが強過ぎても弱過ぎても、プレイヤーにとっては面白くありません。その地点まで到達したプレイヤーが、あと少し努力をする事でボスキャラクタを倒せる…というのが理想のバランスだと思います。こうしたバランス調整の為に、デバッグやテストプレイが繰返し必要になってきます。

４　戦闘中の特殊攻撃の処理

ロールプレイングゲームにおいて、戦闘シーンというのは、プレイ時間の大半を費やすことになる為、非常に重要なウェイトを占めます。本来ですと、複数人パーティによる戦闘シーンなどで、戦闘ターンの順番や得意・不得意な相手、属性などのパラメータにより、より複雑に楽しめる戦闘シーンが望ましいのですが、今回の MiyagiQuest では、主人公パーティは存在せず、あくまでソロ（1 人）での冒険となっています。その為、最低限の駆け引きを行える「全体攻撃」と「回復行動」の 2 つの特殊コマンドを実装しています。
どちらのコマンドもリスク無しで選択出来てしまっては、戦闘バランスを崩してしまいます。その為、SP（スキルポイント）と呼ばれる特殊行動に必要なポイントを設定し、特殊攻撃・回復行動どちらを行う上でも一定の SP を消費するようにしてあります。
消費 SP の設定は、MiyagiQuestView.java のフィールド reqSpec、reqCure で設定しています。

```
    public int reqSpec = 3;                    //  特殊攻撃に必要なSP↓
    public int reqCure = 2;                    //  回復に必要なSP↓
'
```

特殊攻撃・回復行動に必要な消費 SP を設定
（MiyagiQuestView.java@フィールド）

４－１　特殊攻撃

先述した通り、プレイヤーが操る主人公パーティはソロです。しかし、敵モンスターは徒党を組んで襲ってきます。その為、1 対他という状況に陥ってしまいます。それらを効率的に打破する為に、SP を消費して全体攻撃を行う事が出来ます。

こちら側は１人なのに、敵モンスターは大勢です

ソースを見ながら順に見ていきましょう。下記は、MiyagiQuestView.java の戦闘シーン描写処理を行う battle メソッドです。コマンド選択により、分岐する中で、「特殊」が選ばれた際にこちらに処理が移動します。

```
        }else if (battleMenuSelectedIndex == 1) {
            //  「特殊」が選択されました。
            if (sp <= reqSpec){
                battleMsgTxts = new String[4];
                battleMsgTxts[0] = "SPが足りません";
            }else {
                TouchStatus = -1;
                sp -= reqSpec;
                for (int i = 0; i < enemy.length; i ++) {
                    if (enemy[i].hp > 0) {
                        battleEnemySelectedIndex = i;
                        break;
                    }
                }
                battleMsgTxts = new String[4];
                battleMsgTxts[0] = "";
                battleMsgMasterTxts = new String[0];
                count = 1;
                damagein = true;
                paintFlag = true;
            }
```

特殊行動コマンド選択時の初期処理

(MiyagiQuestView.java@battle メソッド)

最初に、現在の SP が消費 SP 分有るかどうかの確認をしています。無ければ、特殊攻撃を行う事が出来ません。もし、足りているのであれば消費 SP を現在 SP から減算し、その後、出現モンスターの全てを対象として選択する for 文を行っています。本来は単体攻撃である為、複数選択は不可能なのですが、特殊攻撃においては、強制的に全モンスターを対象としています。

```
            //特殊攻撃
            }else if(battleMenuSelectedIndex == 1){
                battleActionList.add("MESSAGE:ミヤギ君の特殊攻撃");
                battleActionList.add("SLEEP:500");
                for(int l = 0; l < enemy.length; l++){
                    if(enemy[l].hp <= 0){
                        continue;
                    }else{
                        int damage2 = enemy[l].getAttackDamage(player);
                        battleActionList.add("MAGIC:" + l + ":"+damage2+":"
                                + enemy[l].name + "に、¥n"
                                + damage2 + "のダメージを与えた！");
                        battleActionList.add("SLEEP:500");
                        int enemyHp2 = enemy[l].hp - damage2;

                        if (enemyHp2 <= 0) {
                            enemy[l].skip = true;
                            //  敵を倒した！！
                            battleActionList.add("MESSAGE:"
                                    + enemy[l].name + "を倒した！");
                        }
                    }
                }
```

特殊行動コマンド選択時のメッセージ作成処理

(MiyagiQuestView.java@battle メソッド)

次の戦闘メッセージ作成処理においても、同様に全モンスターの数を for 文で作成してい

ます。この時、ダメージ計算の結果、そのモンスターを倒せているのであれば、そのモンスターの skip フラグを true とし、選択対象外とする様にしています。今回の battle メソッドは繰返し何度も呼び出される為に、この様な対応となっています。

```
        }else if(action.startsWith("MAGIC:")){
            //特殊攻撃
            battleMsgTxts = new String[4];
            String msgs[] = MyUtil.split(action.substring(action.lastIndexOf(":")+1), "¥n");
            for (int i = 0; i < battleMsgTxts.length; i ++) {
                if (i < msgs.length) battleMsgTxts[i] = msgs[i];
            }
            String acts[] = MyUtil.split(action, ":");
            int idx = Integer.parseInt(acts[1]);
            int damage = (int) ((double)Integer.parseInt(acts[2]) * 0.8);
            int enemyHp = enemy[idx].setDamage(damage);
            if (enemyHp <= 0) {
                android.util.Log.e(DebugLog.TAG, "Done ![" + enemyHp + "]");
            }
            MyUtil.PlaySound_SE(activity, MyUtil.SE_MAGIC);
            paintFlag = false;
```

特殊行動コマンド選択時のダメージ値設定処理

(MiyagiQuestView.java@battle メソッド)

次は、実際のダメージ計算の部分です。通常の攻撃ダメージとして産出されていた値に、0.8 を掛けた値を特殊攻撃のダメージとしています。そうして計算されたダメージ値を setDamage メソッドで敵モンスター1 対毎に設定していきます。

```
    // 敵の描写
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(2);
    paint.setColor(Color.RED);
    for (int i = 0; i < enemy.length; i ++) {
        if(battleMenuSelectedIndex == 0){
            enemy[i].drawEnemy(canvas, paint, count%4==0, i == battleEnemySelectedIndex);
        }else if(paintFlag == true){
            enemy[i].drawEnemy(canvas, paint, count%4==0, true);
        }else{
            enemy[i].drawEnemy(canvas, paint, count%4==0, false);
        }
    }
```

特殊行動コマンド選択時の複数モンスター枠選択処理

(MiyagiQuestView.java@battle メソッド)

最後に、複数の敵モンスターをまとめて選択している事を表す赤枠線を描写している部分です。Enemy クラスの drawEnemy メソッドの第 4 引数が「選択されているかどうか」の意味なので、全ての敵モンスターに true（選択されている）を渡す事で、全モンスター上に選択されている事を意味する赤枠線が描写されることになります。

特殊行動コマンド選択時には、
複数モンスターを同時に全て選択されている事になります

４－２　回復行動

先述した通り、プレイヤーが操る主人公パーティはソロです。しかし、敵モンスターは徒党を組んで襲ってきますので、大量のモンスターから一度にダメージを受ける事になります。その時、自分の HP がピンチになりそうだと判断したら、1 ターン消費する事にはなりますが、SP を消費して自分の HP を回復する事が出来ます。

こちらも、ソースを見ながら順に見ていきましょう。下記は、MiyagiQuestView.java の戦闘シーン描写処理を行う battle メソッドです。コマンド選択により、分岐する中で、「回復」が選ばれた際にこちらに処理が移動します。

```
        } else if (battleMenuSelectedIndex == 2) {
            //  「回復」が選択されました。
            if (sp < reqCure){
                battleMsgTxts = new String[4];
                battleMsgTxts[0] = "SPが足りません";
            }else{
                TouchStatus = -1;
                sp -= reqCure;
                for (int i = 0; i < enemy.length; i ++) {
                    if (enemy[i].hp > 0) {
                        battleEnemySelectedIndex = i;
                        break;
                    }
                }
                battleMsgTxts = new String[4];
                battleMsgTxts[0] = "";
                battleMsgMasterTxts = new String[0];
                count = 1;
                damagein = true;
            }
```

回復行動コマンド選択時の初期処理

(MiyagiQuestView.java@battle メソッド)

最初に、現在の SP が消費 SP 分有るかどうかの確認をしています。無ければ、特殊攻撃と同様に回復行動を行う事が出来ません。足りているのであれば、消費 SP を現在 SP から減算し、その後、出現モンスターの全てを対象として選択する for 文を行っています。処理上、意味は無いのですが、戦闘ルーチンが共通なので、仮に攻撃対象を設定している事にしてあります。

```
            //回復行動
            }else if(battleMenuSelectedIndex == 2){
                battleActionList.add("MESSAGE:ミヤギ君の回復");
                battleActionList.add("SLEEP:500");
                battleActionList.add("RECOVER:");
                battleActionList.add("MESSAGE:" + "ミヤギ君はHPを回復した");
                battleActionList.add("SLEEP:500");
            }
```

回復行動コマンド選択時のメッセージ作成処理

(MiyagiQuestView.java@battle メソッド)

次の戦闘メッセージ作成処理においては、単純に回復した旨を表現するメッセージを作成しているだけです。

```
            }else if(action.startsWith("RECOVER:")){
↓
                //回復行動↓
                int recovery = (int)(maxHp * 0.3);↓
                if(hp + recovery > maxHp){↓
                    hp = maxHp;↓
                }else{↓
                    hp += recovery;↓
                }↓
                MyUtil.PlaySound_SE(activity, MyUtil.SE_CURE);↓
```

回復行動コマンド選択時の回復値設定処理

(MiyagiQuestView.java@battle メソッド)

次は、実際の回復値計算の部分です。回復値は現在の最大 HP＊０．３の計算で産出された値を、現在 HP に加算します。仮に、加算後の値が最大 HP を上回ってしまうのであれば、加算後の値を現在の最大 HP としてあります。

SP を消費して HP を回復しています

回復時の加算値が最大 HP を超えてしまったので、現在の最大 HP まで回復しています

４－３　戦闘中の特殊攻撃の処理に関する注意点

今回、主人公１人での冒険であり、最低限の行動として全体攻撃と回復行動を実装しています。ロールプレイングゲームの戦闘シーンにおいては、主人公パーティの行動に複数の選択肢が有る事で、奥深さや戦略性が高まってきます。SP 消費での攻撃力アップや防御力アップ、一定確率で敵モンスターを一撃で倒す技や敵モンスターを眠らせる技など、応用の幅は多岐に渡ります。是非、コマンドを追加して多くの選択肢を実現出来る様、発展させてみて下さい。

また、今回は通常攻撃時においても、効果音のみで、アニメーション等の戦闘エフェクトを実装していません。コマンドを追加して選択肢の幅を増やせる事に成功出来たのであれば、是非、戦闘時エフェクト等を追加して、より臨場感ある戦闘シーンを実現してみて下さい。

オンライン評価システム
利用マニュアル

Ver. 1.00.00

更新履歴

Version	変更内容
1.00.00	初版発行

目 次

1. オンライン評価システムとは

「オンライン評価システム」は、登録者が自分のファイルをアップロードしたり、他の登録者がアップロードしたファイルをダウンロードして評価したりするためのシステムです。「震災復興に貢献するモバイルアプリケーション等コンテスト」では、参加者が制作した作品をアップロードしたり、一次審査における相互評価を行ったりするのに用います。

1.1 利用環境

本書では、以下の環境を使用しています。

OS	Windows Vista™ Business Service Pack 2
ブラウザ	Internet Explorer 8

使用している環境によっては、本書に記載されている画面、メッセージとは異なっている場合がありますので、ご了承下さい。

1.2 対象読者

本書が対象にしている読者は、Windows やブラウザの基本的な操作ができる方です。これ以外に特別なスキルは必要ありません。

1.3 免責事項

本書に掲載する情報には、充分に注意を払って作成していますが、その内容について保証するものではありません。また、特定非営利活動法人教育支援システム研究機構は、本書の内容の使用ならびに閲覧によって生じたいかなる損害にも責任を負いかねますので、ご了承ください。

なお、本書の内容については、予告なく変更される場合があります。

2. オンライン評価システムの利用方法

「オンライン評価システム」は、その名の通り、インターネット上に設置されているシステムです。基本的には、インターネット・ブラウザから全ての操作を行うことができます。

2.1 オンライン評価システムのアクセス方法

「オンライン評価システム」の URL は、以下の通りです。PDF ファイル等でこのマニュアルをご覧になっている場合は、以下の URL をマウスでクリックすると、ブラウザが起動して「オンライン評価システム」のトップページが表示されます。

URL：http://203.138.125.240/users/login

もしブラウザが起動しなかったり、上の URL がクリックできなかったりする場合は、ブラウザのアドレスバーにコピー＆ペーストするか、直接キーボードで入力してください。

異常がなければ、以下のトップページが表示されます。

図 1 「オンライン評価システム」トップページ

2.2 ログイン

作品のアップロードや相互評価を行うためには、ログイン ID とパスワードを入力して、ログインします。ログイン ID とパスワードは、コンテスト事務局からご連絡しているものを使ってください。ログインに成功すると、以下のような画面が表示されます。

※ログイン ID とパスワードはなくさないように注意してください。

図 2 マイページ

初めてログインした時は、まだ作品をアップロードしていないので、上の図のように、「マイ作品」のページには何も表示されません。

2.3 マイページの説明

マイページの画面構成は、以下の通りです。

図 3 マイページの画面構成

項目	説明
①マイ作品	オンライン評価システムにアップロードした作品を確認するページです。
②評価一覧	オンライン評価システムにアップロードされている全ての作品の評価を一覧で表示するページです。
③作品提出	作品を提出するページです。
④作品評価	他の作品を評価する時に使うページです。
⑤マイ作品の説明	アップロードした作品の説明を確認します。
⑥ユーザ名	自分のユーザ名が表示されています。
⑦ログアウト	操作を終了してログアウトする時にこのボタンをクリックします。

各項目の詳しい説明は、以下で行います。

2.3.1 マイ作品のアップロード

自分の作品をアップロードするには、マイページで「作品提出」のボタンをクリックします。

※チームの作品は、チームのリーダーがアップロードしてください。

図 4「作品提出」をクリック

すると、次のような画面が表示されます。

図 5「作品提出」ページ

操作方法を順番に説明します。

①提出部門

作品を提出する部門を、「規定部門」「自由部門」の中から 1 つだけ選択します。部門を間違えないようにしてください。

②タイトル

提出する作品のタイトルを入力します。他の方から見て、わかりやすいタイトルにしましょう。

③内容

作品の内容を説明します。自由部門で提出する方は、「業務系ソフトウェア」「ゲーム」「キャラクターデザイン」「音楽」「企画書・レポート等」のどの作品かを、最初に記入してください。また、パソコンで表示したり使用したりする際の注意点（ファイルの種類等）があれば、ここに記入してください。他の方が見た時に、できるだけわかりやすく記入してください。

④ファイル

提出する作品のファイルを選択します。提出する作品の部門・内容によって、以下のようなファイルを用意してください。なお、アップロードできるファイルは 1 つだけですので、**ファイルが複数になる場合は、1 つの ZIP ファイルにまとめてアップロードしてください**。

部門	課題	提出ファイル
規定部門	（1）キャラクターの追加、または差し替え	・Apk ファイル（課題を実装した場合） ※課題をゲームに実装しない場合は、以下のものを用意してください。 （1）キャラクターのデータファイル（PNG、JPG、GIF 等のファイルで、24×24 ピクセル） （2）改善案の企画書ファイル（PowerPoint 形式） ※（1）（2）のいずれかのみを実装した場合も、APK ファイルを用意してください。
	（2）操作性の改善	

部門	サブカテゴリ	提出ファイル
自由部門	業務系ソフトウェア ゲーム	・実行形式ファイル、使用方法を説明したファイル（TXT 形式等）、その他実行に必要なファイル
	キャラクターデザイン	・JPG、GIF 等、PC で表示可能な画像ファイル ※手書きのものはスキャナ等で取り込むこと。
	音楽	・MP3、MIDI 等、PC で再生可能な音楽ファイル ※演奏を録音して上記のファイル形式に変換しても良い。
	企画書、レポート等	・Word、PowerPoint、PDF 等のファイル

※これ以外に、以下の資料を作成し、事務局（中村徹）宛に、Facebook のメッセージ機能で送信してください。

- 作品紹介シート（1 作品につき 1 つ）
- プログラムファイル一式（規定部門、及び自由部門の「業務系ソフトウェア」「ゲーム」）

  ソースコードのファイルやその他必要なファイル全て。ファイルやフォルダが複数になる場合は、1 つの ZIP ファイルにまとめてください。
- 開発報告書（PowerPoint 形式）【必須】

  作品の開発に当たってのコンセプトや工夫点、苦労した点等をまとめてください。フォーマットは自由です。二次審査では、この資料を用いて発表していただきます。

  ただし、「企画書、レポート等」を提出する場合は不要です。二次審査の発表では、作品そのものを使ってください。
- 作品補足資料【任意】

  作成する場合は、フォーマットは自由です。

⑤スクリーンショット

規定部門や、自由部門の「業務系ソフトウェア」「ゲーム」を提出する場合は、特徴的な画面のスクリーンショットを用意してください。その他の作品は必須ではありませんが、イメージ画像のようなものを作成してアップロードしていただいてもかまいません。

⑥「登録」ボタン

以上の内容を入力して間違いがなければ、「登録」ボタンを 1 回だけクリックしてください。

ファイルのアップロードが成功すると、以下の画面が表示されます。

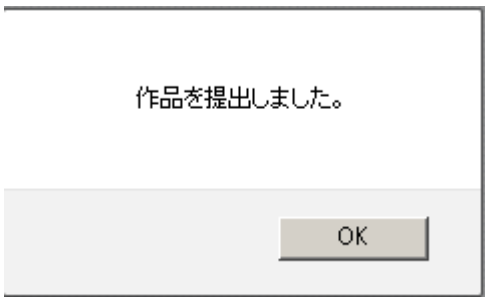


図 6 アップロード成功

また、「マイ作品」のページで登録された情報が確認できます。

図 7「マイ作品」に作品が追加された

上の画面で「詳細」ボタンをクリックすると、以下のような画面が表示されます。

図 8 作品の詳細

「ダウンロード」ボタンをクリックすると、作品のダウンロードが可能です。また、他の方から評価を受けると、「評価」の欄に評価の値が表示されます。

2.3.2 作品の評価

提出された作品を評価するには、「作品評価」のボタンをクリックします。すると、以下のような作品の一覧が表示されます。

図 9「作品評価」のページ

作品を評価するには、評価したい作品の「評価」ボタンをクリックします。すると、次のような画面が表示されます。

図 10 評価を入力するページ

①ダウンロード

作品をダウンロードして、確認します。作品紹介シート、及び作品そのものを表示・使用してみて、評価をつけてください。

②スクリーンショット

登録されている場合は、スクリーンショットを表示することもできます。スクリーンショットが登録されていない場合は、クリックしても画像は表示されません。

③評価（平均）

これまでの評価の平均値が表示されています。

④評価

評価は★の数でつけます。★5 つが最高です。1 つだけ選択してください。

⑤コメント

作品に対する感想等があれば、この欄に入力してください。

⑥「登録」ボタン

入力した内容に間違いがなければ、このボタンを 1 回だけクリックします。

⑦評価レビュー

これまでに評価を行った方の評価点、及びコメントが一覧で表示されます。

※最低限、次の数だけ作品を評価してください。なお、自分の（チームの）作品は数えません。

提出した部門	評価する作品数
規定部門	規定部門：2 作品 自由部門：3 作品以上
自由部門	規定部門・自由部門合計で 5 作品以上

規定部門と自由部門の両方に作品を提出した方は、「規定部門：2 作品、自由部門：3 作品以上」の作品を評価してください。

評価した作品数が多い場合は、審査で加点になります。評価の期間は短いですが、できるだけ多くの作品の評価を行ってください。

2.3.3 評価一覧

「評価一覧」のページでは、これまでに登録された評価点や評価コメントの一覧が、新しい順に表示されます。評価をする作品を選ぶのにも使えます。

図 11 評価一覧