プログラミング入門支援ゲームの試作と評価

09MH05 辻 大地

目次

概要

プログラミング教育の問題点として、プログラミングの授業を履修したにも関わらず、自力でプログラムを組めるようになる学生は多くないといった問題がある。プログラミング教育では、教員が学生に講義内容を理解させるだけでは不十分であり、学生が自発的に練習や問題解決に取り組むプログラミング実習を実施することが重要である。限られた時間内に教育効果を上げるためには、学生の”関心”や”自発性”を高めることが重要である。しかし、プログラミングを初めて学ぶ学生にとって、プログラムは非常に難解なものに見え、苦手意識を抱いてしまったり、関心をなくしてしまい自発性も養われない。本研究ではこの問題を改善する方法について検討した。

著者はこの”学生がプログラミングを習得出来ない”事の最大の要因は、「学生のモチベーション」にあると考えており、これを解決する方法を幾通りか検討した結果、本研究ではモチベーションを向上させる手段としてゲームを取り入れた。評価実験では、本研究により作られたプログラミング入門支援ゲームを授業において実際に使い、学生のモチベーションがどのように変化したかをアンケートにて調べた。

1 はじめに

本学の工学部情報システム工学科の学生の問題点の 1 つとして、プログラミングの講義を履修したにも関わらず、自力でプログラミングを組めるようになる学生は多くないという問題がある。今回この問題を解決する方法を検討した。

本学科の現状のプログラミングの講義の学習プロセスは以下の通りである。

1. 学生に学習目標を知らせる
2. 本授業時間で学ぶ命令構文の教示
3. テキストの例文の説明
4. プログラミング実習演習

他大学における、プログラミング指導の方法と成果を調査した結果、大半の学生は講義時間以外の自主的な学習を行わず [1]、構文理解の困難さからプログラミングに苦手意識を持ち [2]、結果一連の学習をしても、問題を与えられると解けなくなる学生が多くなる [3] という問題が明らかになった。これらの問題を改善する取り組みは以前から存在しており、例えば学習環境の構築提案や面接予約システム [4]、電子カルテの導入 [5]、Peer Review を用いての授業支援ツールの開発 [6] といった授業システムを改善しようといったものから、大学生に小学生のプログラミング指導をさせる [7] 等といった提案が数多く存在する。

しかし、効果的な方法は未だ確立されていない。そこで本研究ではこれらとは違った解決手段を検討した。解決手段を検討する手始めとして、プログラミングに苦手意識を抱く学生が多い原因を特定する事から始めた。

類似研究によると、学生がプログラミングの修得を困難とする原因は、学生のプログラミングに対する学習意欲やモチベーションの低さが主な要因であると提唱されている [8]。学生は学習意欲を損なわれた結果、効果的なプログラミング学習が困難になっているといわれている [3]。

上記の結果から、プログラミング初学者にプログラミングを習得させるには、モチベーションを向上させる事が重要であると仮定した。そこで、モチベーションを向上させる方法について調べた。結果、歴史を通して絶えず人類は競争状態に携わってきたことが分かった。例えばマネジメントやスポーツの世界では昔からモチベーションを向上させる為に競争心が利用されてきたし、原始人は、互いに、また他の種族と生存をかけて戦ってきた。また、リハビリや授業など固定化されたグループ内においても、競争心は意欲向上に役立つ事が分かっている [9]。続いて、競争心を高める方法について検討した。結果競争心を煽るにはゲームが最適であると分かった [10]。以上の経過により本研究では「プログラミングの講義の入門にゲームを応用させる事は学習者のプログラミング学習へのモチベーションを向上させることに対し有効である」と仮定し、それを検証する。

2 先行研究

プログラミング学習にゲームを応用させた研究はこれまでにも存在する。本章では、それらについて整理し本研究の特色を明確にする。

2.1 ゲームプログラミングによる情報教育の評価方法

この研究 [11] では、“プログラミング言語を中心とした教育ではなく、楽しくプログラミングが学習できる環境が必要である” とし、著者が制作した “だいこん 3 号” というゲーム作成学習ソフトを用いて学生にインベータ風のゲームを製作してもらい、その過程においてイベントドリブン的にプログラミング能力を育成することを目的としている。

2.2 ゲーム感覚でプログラミング可能なプログラミング言語に関する研究

この研究 [12] では、“Kazeoke” と呼ばれる視覚的プログラミング言語を開発し、“Kazeoke” を用いてプログラミング支援を行っている。“Kazeoke” はアイコンを並び替えることでプログラム可能なプログラミング言語であり、学習者はマウスを用いてアイコンを並び替えるだけでプログラミングが行えるとした研究。

2.3 ゲーム作成を課題にしたプログラミング教育とその分析方法の開発

この研究 [13] では、コンピュータ上に可視化したオブジェクトをダイナミックに組み合わせプログラムすることができる “IntelligentPad” [*1] システムを利用して、学習者にゲームを作成してもらいながらプログラミング学習を行うことを目的にしている。学習者は 200 以上もの部品を自由に組み合わせる等してゲームを作成することが可能である。

2.4 ボードゲームの戦略プログラミングを題材とした Java 演習の支援システムの開発

この研究 [14] では、五五ゲームと呼ばれるボードゲームを題材に学習者に戦略プログラムを製作させている。学習者は戦略プログラムを製作する過程においてプログラミングスキルを向上させられる。五五ゲームを取り上げた理由は身近な題材でとっつきやすいと考えた為であり、学生同士対戦させる事により競争心を刺激したとしている。講義の中間と終盤において大会を設け学生間の盛り上がりを高めているが、最終的な評価方法・成績の決定を課題としている。

2.5 ToonTalk - コンピュータを利用した理想的な学習環境へのステップ

この研究 [15] では、“ToonTalk” とよばれるツールを用いて、パズルゲームのようにプログラミングを楽しむものである。スクリーン内のオブジェクトをパズルのピースを並び替えるようにしてプログラミングライクな遊びを楽しむことを目的としている。

[*1] IntelligentPad とは、パッド (紙) と呼ぶ機能やデータが入った固まりをパソコンの画面上で目に見える形で表し、それらのパッドを直接操作によって組み合わせることにより、利用者が思いのままに、新たな編集物や道具を作り出すことができるシステムである。

2.6 先行研究のまとめ

これまでの先行研究は下記の 3 点に分類する事が出来る。

1. ゲームを製作する事を目的とした学習
2. 特定のゲームを攻略する為の AI の開発
3. 教育の為という色合いの濃いゲーム

１と２はある程度プログラミングが出来る事を前提としている。本研究ではこれらよりもっと初期の段階、プログラミングの入門段階で、学生のモチベーションを下げない事が重要であると考えており、これらよりもより初学者を対象にした支援が必要と考えた。

３について、これらの論文で言われているゲーム感覚と、著者が想定しているゲーム感覚は違ったものである。その相違点とは、上記研究でいうゲームとは、ゲーム黎明期のような単調なものであったり画面上にアイコンを配置し、それが特有の役割を果たすといったアナログゲームに近いものであるのに対し、著者が想定しているゲームは、ファミリーコンピュータに代表されるような家庭向けデジタルゲームに近いものであるといった違いがある。この違いにより、学生はより慣れ親しんだゲーム環境でゲームを楽しみながらプログラミング的思考 (後述 3) を体験する事が出来ると想定している。

3　本研究の目的

本研究の位置づけは、プログラミング学習の効果を高める研究の中でもゲームを用いた研究分野に所属しており、その中でも対象を初学者に限定した極めて入門向けのものである。本研究の目的は、「ゲームを通して学生のプログラミングへのモチベーションを向上させる事」 にあり、ゲームの中で身につけるプログラミングの知識の中にプログラミング的思考力の習得がある。本研究でいうプログラミング的思考の体験とは、プログラミングの基本 3 構造（逐次実行・条件分岐・繰り返し）を体感してもらう事を指しており、プログラミングの知識の全くいらないゲームをプレイしながら、プログラミングの基本 3 構造を体感してもらうことである。

上記のようにして、プログラミングの授業を始める前の段階でプログラミングの基本 3 構造を体感[*1] しておくことにより、後の授業における理解が促進されるとともに難解なものとしてとらえられていたプログラミングへの敷居が下がると仮定している。また同時に、本研究ではプログラミングで最も大切だとされている思考力 [16] を鍛えるトレーニングを行うことを兼ねている。

[*1] 受動的な体感ではなく、学生が自ら条件分岐や繰り返しという考え方を思いついてもらうことを期待している。これは自ら思いついた思考法は習得しやすい為である

4　ゲームの説明

どのような要素を折り込む必要があるか先行研究の結論や問題点を考慮し検討した。本研究目的に沿ったプログラミング支援ゲームに必要な要素は以下のとおりである。

- プログラミング学習への敷居が下がる
- 思考のトレーニングを行える
- 基本 3 構造への理解が促進する
- 競争心が高まる
- プログラミングの知識を必要とせず取り組める
- 教育色が濃くない

これらの要素を盛り込んだ (後述 3) ゲームを Python を用いて開発し、アンドリュークエスト (以下本ゲーム) と名づけた。本ゲームはキャラクター (以下 a4w) をゴールに到達させる事を目的としたゲームである。本ゲームのゲーム画面を**図** 1 に示す。

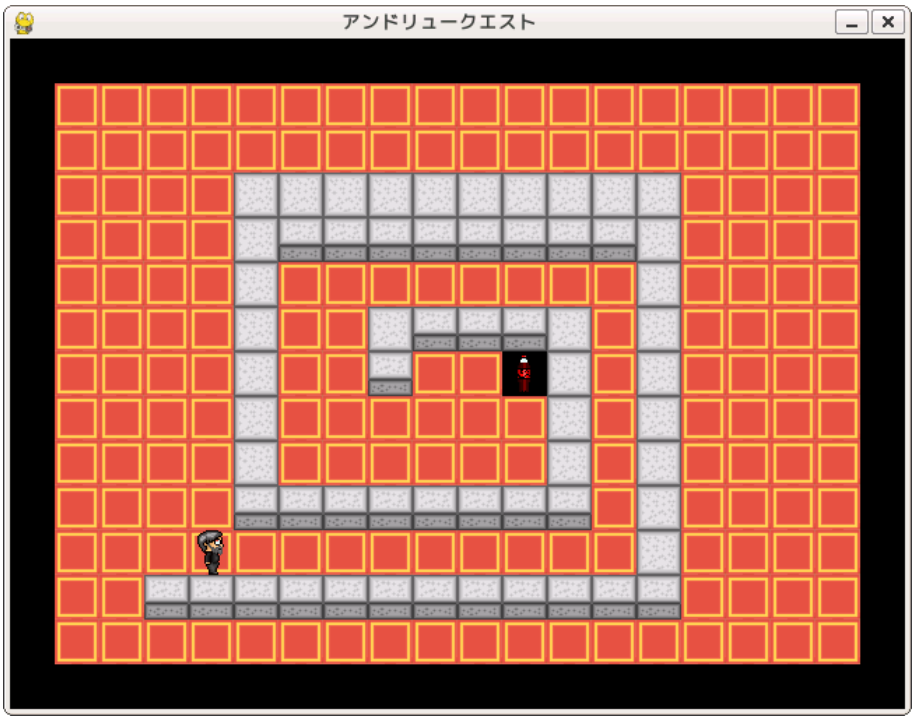

図 1　アンドリュークエストのプレイ画面。　画面内のキャラクターをゴールに辿り着かせる事が勝利条件であり、消費ステップ数を最小に留める事を目標にする。

本ゲームでは、a4w が行動する度に「消費ステップ数」という数値が増大する。プレイヤーはこれを最小限に抑える事を目標にする。a4w の行動には、ステージ毎[*1]に制約が発生する為、ゴールまで到達する為には思考が伴う。このゲームにより、学習者は思考のトレーニングを行いながら、ループや条件分岐という考え方を体感する。

[*1] ステージとは： 本ゲームには 10 のステージが用意されており、各ステージを攻略する毎に次のステージで遊べるようになっている。ステージは徐々に難易度を増し、遊びながら基本 3 原則を体感していけるように設定してある。

4.1 ゲーム詳細

ここから説明用のサンプルステージを用いて、本ゲームの詳細について説明をおこなう。本ゲームには、a4w が動き回るプレイ画面 (**図** 2) と、a4w を操作する操作画面 (**図** 3) の 2 つの画面が存在する。

図 2　プレイ画面：説明用に簡易化されたサンプル版

図 3　操作画面　：学習者からのコマンド (命令) はここに蓄積され、後で一括実行される

4.2 プレイ画面の説明

プレイ画面 (**図** 2) には a4w のスタート位置と到達目標であるゴールが表示されている。学習者は、a4w を操作しゴールを目指す。a4w の行動はひとつなぎである。つまり、学習者からのコマンド (命令) がある度に動くのではなく、スタートからゴールまでのすべてのコマンドを受けてからバッチ処理で行動する。a4w は以下の行動をとる事が出来る。

- 前進
- 右に 90 ° 旋回
- 左に 90 ° 旋回
- ジャンプ

4.3 操作画面の説明

操作画面 (**図** 3) ではキーボードからのキー入力により、a4w にコマンドを与える事が出来る。a4w に与えられる行動は、基本行動と、特殊行動に分けられる。

4.3.1 基本行動

基本動作はカーソルキーを用いて操作される。各カーソルキーは以下のように設定されている。

- [→]キー　：右に 90 ° 旋回
- [←]キー：左に 90 ° 旋回
- [↑]キー　：ジャンプ
- [↓]キー　：前進

コマンド (命令) を受け取った際のキー入力ログは内部に保持され (**図** 4)、操作画面上に表示される。スペースキーの押印により、a4w は一括行動を行う。基本的にこの 4 つのコマンドで a4w を操作する。

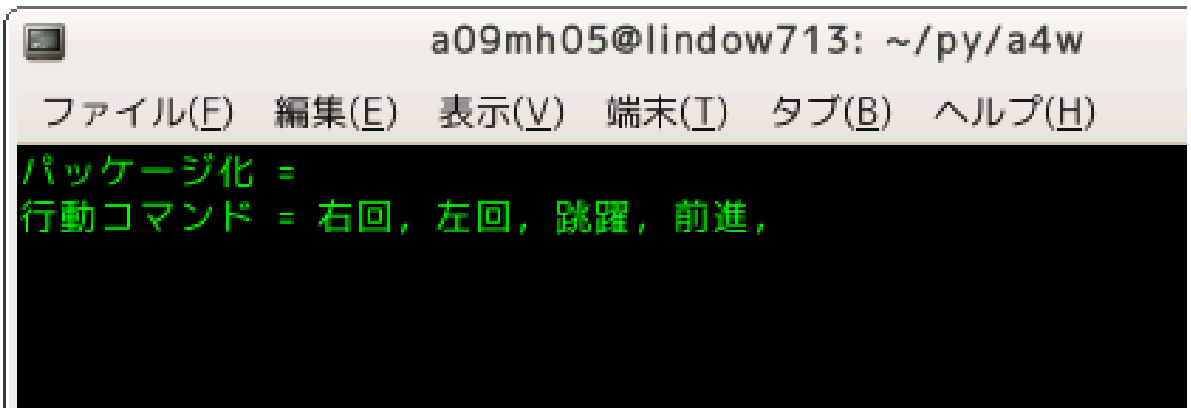

図 4　キーログ　学習者からのコマンドはこのように蓄積されていく

4.3.2 特殊行動

ステージの制約によっては、基本 4 行動だけではゴールまで到達することが出来ない場合がある。例えば、スタートからゴールまで 15 歩はかかるステージの制約が、“5 ステップしか命令出来ない” 等の場合である。この場合、最低でも 10 歩 (15 歩 - 5 ステップ) 分はなんらかの方法で省略する必要がある。このなんらかに該当するのが、“複数命令のパッケージ化” である。複数命令のパッケージ化とは、何度も繰り返している同じ行動を 1 つにパッケージ化する事を言う。このパッケージを呼び出すと、消費ステップを 1 つ消費する替わりに、複数歩進む事が出来る例えば仮に [↓][→][↓][→][↓][→][↓][→] をパッケージ化したとすると**図** 5 のように表示され、パッケージを呼び出す毎に a4w は周回運動を取り始める。この場合、消費ステップ数を 1 つ消費するだけで 8 歩分の行動を命令出来たことになる。これを活用すれば、より少ない消費ステップ数で攻略することが可能になる。パッ

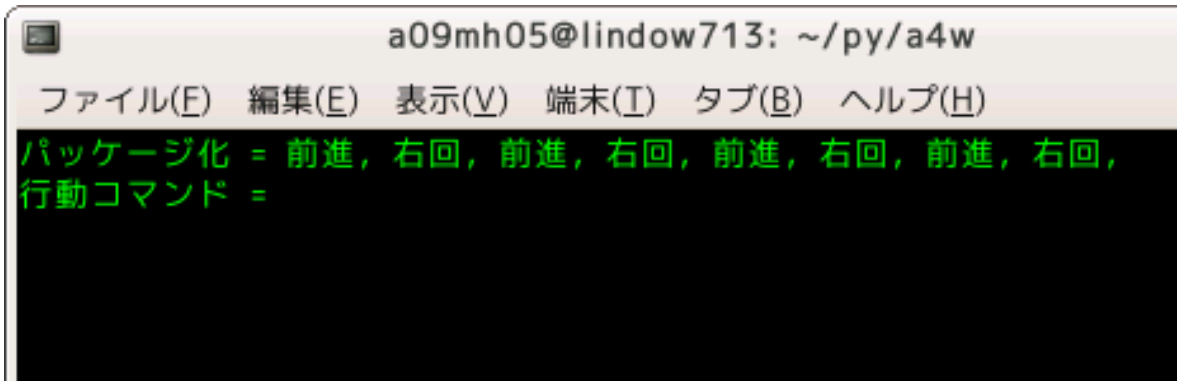

図 5 [↓][→][↓][→][↓][→][↓][→] をパッケージ化し、実行した場合の実行画面。消費ステップ数は 1 消費で済む。

ケージ化するときは、基本 4 行動のキーとは異なるキーを用いて a4w に命令を出す。

- R(ight) キー：右に 90 ° 旋回
- L(ight) キー：左に 90 ° 旋回
- J(ump) キー：ジャンプ
- F(orward) キー：前進

本ゲームを通して、プログラミングの基本 3 構造を体感してもらいながら思考のトレーニングを行ってもらう。

4.4 アンドリュークエストによって身につくプログラミングの知識

本ゲームで体感出来るプログラミング的思考に、繰り返しと条件分岐がある。

4.4.1 繰り返し

より少ないステップ数でゴールまで到達するには、工夫が必要であり、何度も繰り返し登場するコマンドはまとめておく事が必須になる。 これがパッケージ化の考え方である。学習者は目標（最短ステップ数による攻略）を達成する為に教えられずとも繰り返しの概念をつかむと期待している。また、そもそもコマンド数を減らす為、特定の条件の後に必ず発生するコマンドを自動化するという発想が生まれる。

4.4.2 条件分岐

特定の条件によって、その後の行動に変化をもたらせる事が出来る。例えば、進行方向が穴であれば迂回する、敵であれば攻撃する等である。条件分岐の方法はステージの初めで a4w にアイテムを装備させる事で行う。本ゲームで実装されているのは、ジャンプシューズを装備させている間 a4w は目の前が段差であれば、自動的にジャンプするというものである。

繰り返しや条件分岐等のような (一般的に非日常的な) 考え方を、学習者が自ら生み出すことが後の授業 (条件分岐や繰り返しの概念) の理解を促進すると想定している。

5　評価実験

ゲームの評価手段として、本学の 1 回生を対象にしたプログラミングの授業に実際に導入し、学生にアンドリュークエストで遊んでもらい、アンケート調査を行った。

5.1　アンケート調査

アンケートは期間を開けて 2 度行った。
今回のアンケートで重要視する点は 2 点ある。1 点は本ゲームの有効性を数値化する事であり、もう 1 点は、学習者の内面にどのような変化が生じたかを調査することである。何故なら、本研究では、学習者のモチベーションや競争心、苦手意識といった数値に表わし難い度合いに重きをおいている為である。よって 1 度目は、初回プレイの直後に、メンタル面においてどのような変化が生じたかを訪ね
2 度目は、アンドリュークエストによってその後の授業における理解が促進したかを調査することを目的とし、1 度目の 7 週間後に行った。
次ページの**図** 6 にアンケート用紙を示す。

アンケート

プログラミング2

2010年9月21日

1　アンケートの前に

「1か月後にもう一度アンケートを予定しています。今回のアンケートと次回のアンケートを連携するためのキーとして以下のスペースに携帯電話の電話番号の下4桁だけを記入してください。携帯電話を持っていない人は自宅の電話番号の下4桁だけを記入してください。」[　　]

2　白板の問題を解けますか。(言語自由)

- □ 自力で解ける。
- □ テキスト見ながらなら解ける。
- □ わかりません。

3　プログラミング1を履修しましたか

- □ 毎回授業に出た。
- □ ほぼ授業に出た。
- □ あまり授業にでてない。
- □ ほとんど授業にでてない。
- □ 履修していない。

4　何面までクリア出来たか、以下の余白に記述してください。

[　　]

5　「先がどうなるか頭の中で推論する力」のトレーニングになりましたか。

- □ はい。かなりなった。頭痛い。
- □ まぁまぁ。なったような気がする。
- □ よくわからない。
- □ トレーニングになるとは思えない。
- □ いいや。まったくもって無意味。

6　より少ない手数で解こうと挑戦しましたか。

- □ はい。自分と戦ってました。
- □ まぁまぁ。目指したり目指さなかったり。
- □ どちらともいえない。
- □ あまり興味をもてなかった。
- □ それよりちゃっちゃと終わらせたかった。

7　周りの進み具合は気になりましたか。

- □ はい。友達と競争していた。
- □ まぁまぁ。友達の進み具合が気になった。
- □ どちらともいえない
- □ あまり効果を感じなかった。
- □ いいえ。競争心は生まれなかった。

8　プログラミングの入門段階でこのようなゲームを用いる事について、どう思いますか

- □ 通常の講義より面白い。
- □ まぁまぁ。そこそこ良いんじゃない。
- □ どちらとも言えない。
- □ あまり良い取り組みとは思えない。
- □ ゲームはいいからいつも通りの授業をやろう。

9　感想や意見を記述してください。

図6　初回ゲームプレイ直後に行ったアンケート

以下のグラフは、**図** 6 のアンケート結果である。

解答はすべて、左が肯定的な解答、右が否定的な解答となっている。

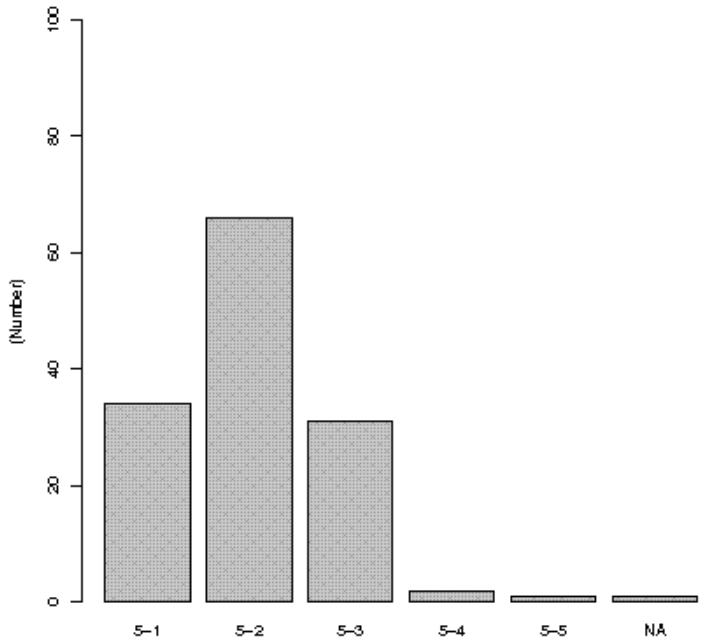

図 7　問 5 思考のトレーニングになったか という問いへの解答

2 度目のアンケートは、初回のアンケートから 7 週間後に行っており、本ゲームをプレイした結果、プログラミングの授業への理解度がどのように変化したかを調査した。

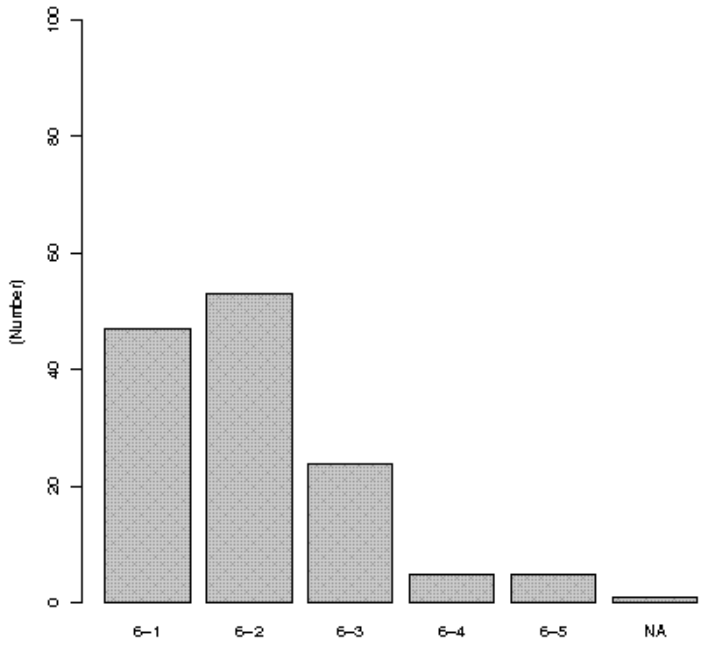

図 8　問 6 自分との競争 に関する問いへの解答

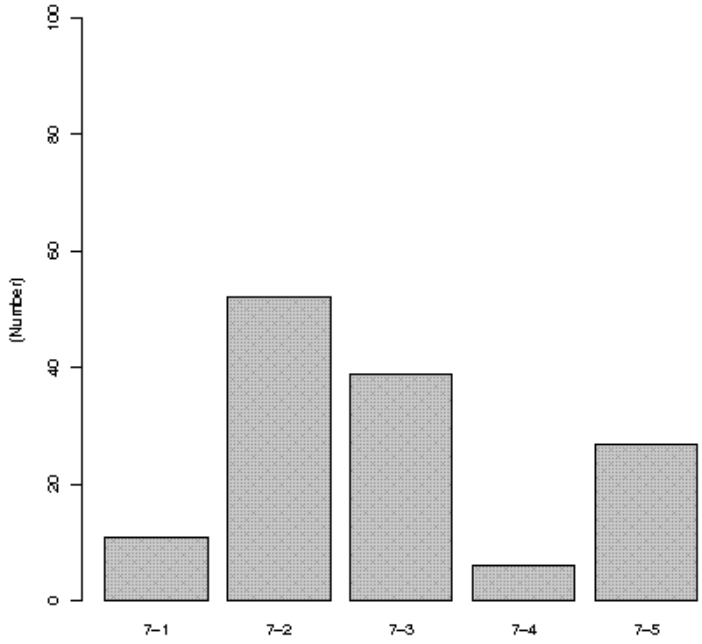

図 9　問 7 他者との競争 に関する問いへの解答

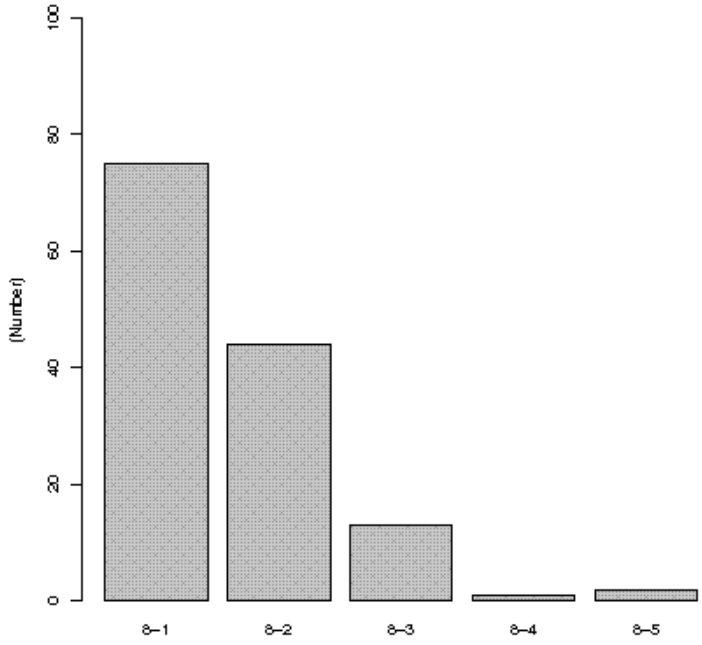

図 10　問 8 プログラミング入門にゲームを導入する是非 に関する問いへの解答

アンケート

プログラミング 2

2010 年 11 月 15 日

1　アンケートの前に

「前回のアンケートとリンクする為、前回と同じキー(携帯電話の電話番号の下 4 桁) を記入してください。」
[　　]

2　前回の講義以降で、a4wquest(以下本ゲーム) を自主的にプレイした回数

「本ゲームをプレイした回数を記入してください。」
[　　]

3　本ゲームは、ループを理解するのに役立ったか

- はい。かなりなった。
- まぁまぁ。なったような気がする。
- どちらともいえない。
- 役立つとは思えない。
- いいや。まったくもって無意味。

4　本ゲームは、条件分岐を理解するのに役立ったか

- はい。かなりなった。
- まぁまぁ。なったような気がする。
- どちらともいえない。
- 役立つとは思えない。
- いいや。まったくもって無意味。

5　本ゲームをプレイ後、授業への取り組み姿勢に変化は生じたか

- やる気出るので、今後もゲームを取り入れるべし。
- プログラミングにゲーム要素を感じるようになった。
- プログラミングに対する敷居が下がった。
- あまり効果を感じなかった。
- マイナス効果を生じたのでやめるべき。

マイナス効果 :

6　こんなゲームにしたらいい！と思うアイデアを自由に記述してください

7　感想や意見を思う存分好き放題に記述してください。

図 11　アンケート後期

結果をグラフ化したものを以下に示す。NA は無回答を示す。

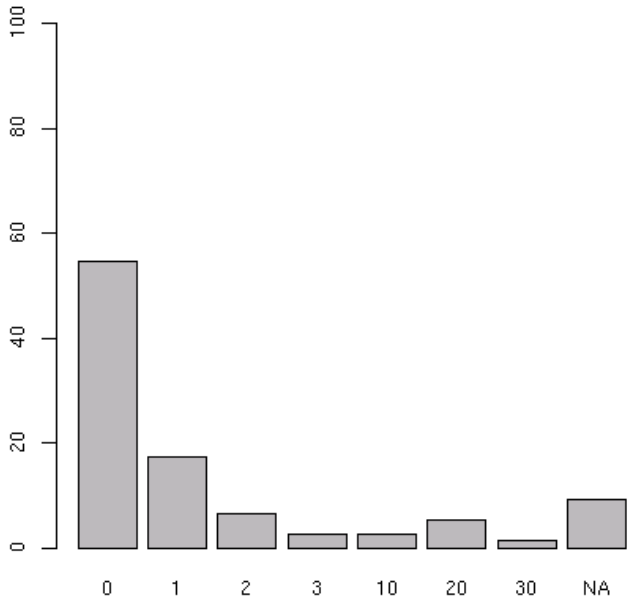

図 12　問 2 授業時間外で自主的に本ゲームをプレイした回数　10 より上は 10 以上 20 以上 30 以上としている

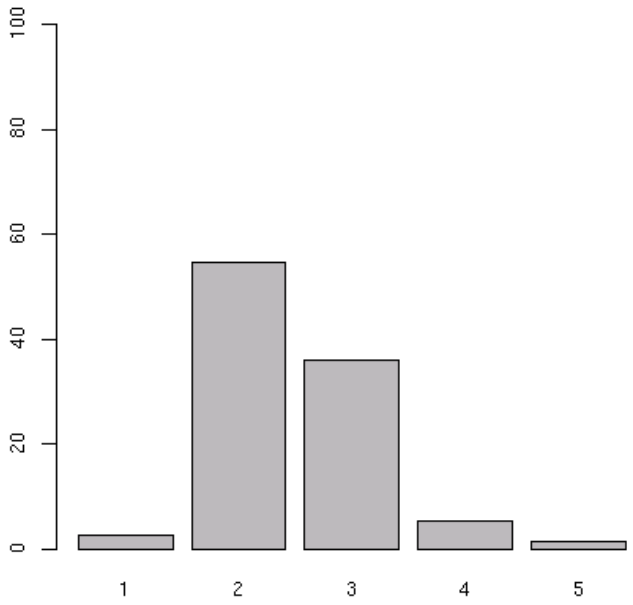

図 13　問 3 本ゲームは、ループの理解に役立つか

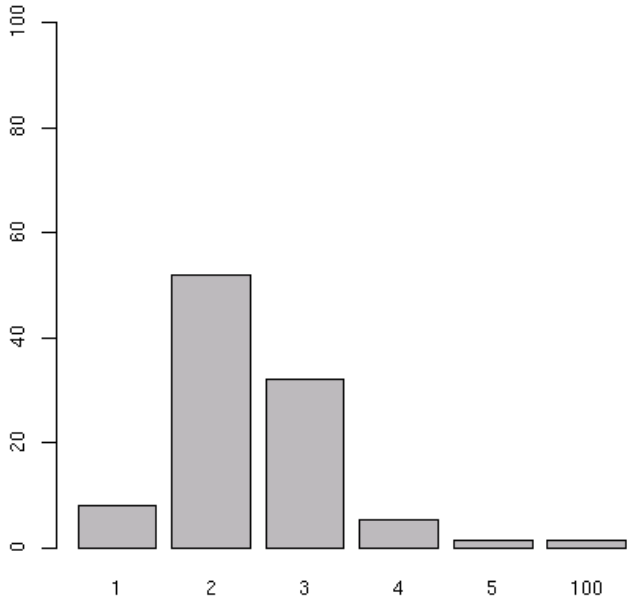

図 14　問 4 本ゲームは、条件分岐の理解に役立つか

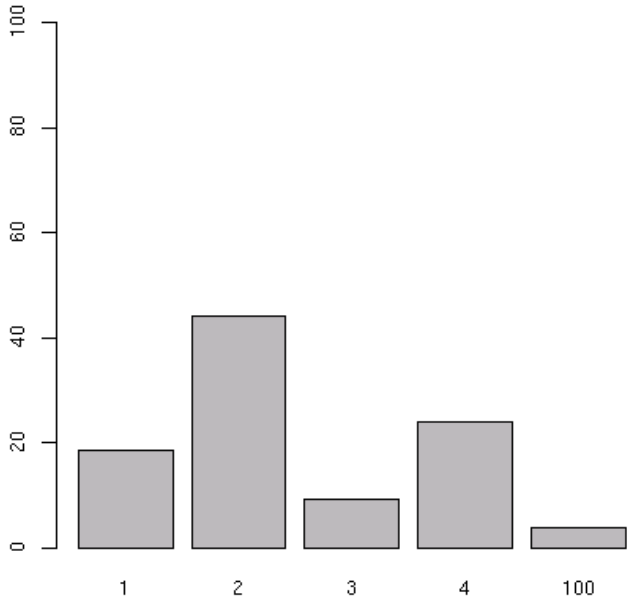

図 15　問 5 授業への取り組み姿勢に変化は生じたか

6　結果と考察

ここまで、ゲーム開発に着手した経緯から、ゲームの説明及び評価実験を行ってきた。ここでは、その結果に対する考察を述べていく。

6.1　メンタル面に対する効果について

前期アンケートはメンタル面でどのような変化が生じたかを中心に調べたものである。前期アンケートの問 5[*1]に対する解答は概ね肯定的な解答が得られている。この結果から、思考のトレーニングを行う効果があったことが分かる。

問 6[*2]に対する解答も概ね肯定的なものが多く、多くの学習者が競争心を刺激されていたことが分かる。

続いて問 7[*3]に対する解答を見ると、全般的には肯定的であるが、否定的な解答も見られる。このことから、本ゲームは自己への競争心は刺激されるが、他者への競争心についてはやや弱いことが分かる。

最後に問 8[*4]をみると、大多数の学習者が肯定的な解答をしており、楽しんで取り組んでいたことが分かる。以上の結果から、アンドリュークエストは思考のトレーニングとメンタル面においては効果的である事が示された。

6.2　アンドリュークエストの効果測定

1 度目のアンケートが示すように内面的な変化に関しては、ほとんどの学習者は肯定的な効果があると回答している。2 度目のアンケートでは、本ゲームをプレイした学生が、その後プログラミングの授業を受けてみて、実際にどれほど理解の支援になったのか、プログラミングに対する苦手意識や敷居の高さは克服されたのか、について調べた。

2 度目のアンケートの問 2 は、学習者が本ゲームを講義時間外に自主的にプレイした回数である。アンケート結果によると約半数の学習者は 0 回と答えているが、数回プレイした学習者や中には 10 回以上もプレイした学習者が存在する事が分かった。問 3 は本ゲームをプレイしたことによって、後の授業でループを理解するのに役立ったかを訊ねたものであり、やや肯定的な意見と中間に位置する意見が半々であった。問 4 は本ゲームをプレイしたことによって、後の授業で条件分岐を理解するのに役立ったかを訊ねたものであり、問 3 と同程度の回答を得られた。この 2 点に関しては、メンタル面に関する調査ほど肯定的な回答ではなかった。学習者の理解をより向上させる方法を検討する必要がある。

問 5 は授業へのプログラミングに対する苦手意識や敷居の高さは克服されたのかについて調べたものである。以下のような意見が多く、ほとんどの学習者が苦手意識を克服してくれた事が分かった。

- やる気出るので、今後もゲームを取り入れるべし。
- プログラミングにゲーム要素を感じるようになった。
- プログラミングに対する敷居が下がった。

[*1] 思考のトレーニングに関する質問
[*2] 競争心 (対自分) に関する質問
[*3] 競争心 (対他人) に関する質問
[*4] ゲームを用いたプログラミング教育の是非

7　結論

本研究では、「ゲームを用いて、プログラミング初学者を対象にプログラミング学習に対するモチベーションを向上させる方法」を考案した。学習の理解を支援するゲームを試作し、その学習範囲はプログラミング基本三構造と思考のトレーニングに限定した。支援ゲームの評価実験は、「学習者の内面に及ぼす効果」と「学習者の学習理解度向上に及ぼす効果」という 2 つの側面から調べており、それぞれに対し行った 2 度のアンケートから分かった結果は下記の通りである。

7.1　学習者の内面に及ぼす効果

学習者の内面に関する質問には主に以下のような項目を用意した。

- モチベーションの向上
- 思考のトレーニング
- プログラミングに対する敷居を下げる

7.1.1　モチベーションの向上に関する効果

第 5 章 **図** 8 からみてとれるように自己の内面で生じる自分との競争力を高めておりゲームを用いることで学習者にプラスの影響を与える事が確かめられた。また、これは数値化出来ないが、学習者同士が教え合い相談するなど授業における雰囲気は楽しげなものであり寝ているような学習者は見受けられなかった。

7.1.2　思考のトレーニングに関する効果

図 7 からみてとれるように本ゲームによる思考のトレーニングに対しては肯定的な回答を得られている。ヒアリング調査によると多くの学生は「数手先を推論」する経験はあまり積んだことがなかったらしく、悩まされたようである。

7.1.3　プログラミングに対する敷居を下げることの効果

図??は、本ゲームをプレイした後、プログラミングの講義を受けてみて取り組み姿勢にどのような変化があったかを調査したものであり、プログラミングにゲーム要素を感じるようになった、プログラミングに対する敷居が下がったという意見が大半であったことを示している。講義の初回は学生にとって今後の講義の雰囲気を感じ取る機会であり、ここで受講者の講義参加へのモチベーションを高めることが大切である。

7.2　学習者の学習理解度向上に及ぼす効果

学習理解度向上に関する質問には主に以下のような項目を用意した。

- ループを理解するのに役立ったか
- 条件分岐を理解するのに役立ったか

7.2.1　ループを理解するのに役立ったか

図 13 によると、肯定的な意見が 6 割を占める一方、4 割の学生が「わからない」という回答をしている。この中にはループを理解する項目までたどり着けなかった学生が少なからずいるにしても満足のいかなかった学習者は多いことが分かった。

7.2.2 条件分岐を理解するのに役立ったか

図 14 をみると分かるように、**図** 13 とほぼ同じ回答が返ってきている。理解度向上に関する 2 つの質問において同じ回答がされているということから、これはそのまま本ゲームが学習者の学習理解度向上に及ぼす効果に対する評価とみて良いと考えている。

8　まとめ

つまり、学習者の内面に及ぼす効果は肯定的な回答を得られたが、その学習理解度向上効果に対しては不十分であり、これについてはより理解を促進させるコンテンツを今後検討する必要がある。今後は、今回分かった課題を引き継いだうえでそれを修正し、より多くの実践を通して改良を繰り返し、プログラミング学習における問題に取り組んでいきたい。

今後の課題として、以下の項目が考えられる。

- モチベーション持続の為、褒めるシステム
- 音やエフェクト効果による惹き付け
- ウェブアプリ化
- 協力や対戦要素の追加
- ランキングの見える化
- ループや条件分岐の効果向上案の検討

8.1　モチベーション持続の為の褒めるシステム

学習者がゲームを遊び続けるうえで飽きが訪れると予測されるポイントにギミックを仕掛ける事はゲームシステムにとっては重要なことである。ギミックの例として、目的の変更、キャラの変化・音・動画・派手な演出・達成感…etc がある。例として、モチベーションを持続させる為に、要所要所学習者の行動をねぎらい・褒めるなどがある。

8.2　音やエフェクト効果による惹き付け

飽きの防止という意味合いでは、聴覚や視覚に刺激を与える事が重要である。音や視覚エフェクトが入ることで脳に異なる刺激を与える事が出来、単調さが軽減される為である。

8.3　ウェブアプリ化

現状、ゲームを始める為にはいくつかのデータを取得してくる等の環境設定が必要である。ここの敷居を下げる為に、ウェブアプリケーションにしてしまい、必要なデータはすべてサーバサイドで所持し、サーバーとクライアント間の通信でゲームを完結させてしまいたい。これにより、更なるサービスの向上も期待出来る。

8.4　協力や対戦要素の追加

周囲の学習者との競争が発生していないという問題がある。せっかく同時刻に同じ空間で遊んでいるプレイヤーが多くいるのだから、これを利用しないのはもったいない。2 人をペアにして協力し合ったり、対戦要素を考案してシステム化することでこの問題の解決が図れると予想する。

8.5　ランキングの見える化

周囲の学習者との競争というところで、上記以外にお互いの成績がランキング化され全員から見えるようにシステム化にすることで競争心が高まる可能性がある。

8.6 ループや条件分岐の効果向上案の検討

プログラミング基本 3 構造の理解度向上という項目に対して十分な評価が得られていないという問題がある。これを改善する為に改良を重ねるか、あるいは全く別の方法による支援を考える必要がある。

謝辞

本論文をまとめるにあたり、大垣 斉先生には本研究の実施の機会を与えて戴き，その遂行にあたって終始，ご指導を戴いたことに深く感謝致します。能勢和夫教授には主査として本論文の細部にわたり様々なご指導を頂いた事に感謝致します。杉村明彦教授, 並びに, 前川佳徳教授には副査として発表の場を通してご助言を頂き方向修正を重ねられてこられたことに感謝申し上げます。平松綾子准教授には、研究の要所要所でご助言を頂き、その度に工夫を凝らしてこれました。この場を借りまして感謝申し上げます。また、本ゲームのアルゴリズムやアンケートの統計法について度々助言頂いた東川 諒央氏, 藤田 康孝氏には感謝しております。共に過ごした日々が有意義であったと，将来語り合うことを楽しみにしております。最後になりましたが，他学科の博士過程への進学に理解を示して頂き、応援してくださった小林 信雄准教授と、博士課程に進学する機会を与えてくださり，ありとあらゆる場面で私を温かく見守り続けてくれた両親に深く深く感謝いたします．本研究の成果が「ゲームの学習への応用」といった一連の流れにどの程度影響を与えられるか甚だ疑問ではありますが，皆様の協力によりその有効性を示せられるまでになりました、ここに重ねて厚く謝意を表し，謝辞といたします．

参考文献

[1] 森下 博 「学習意欲向上を目指したプログラミング授業の実践的展開」, 平成 20 年度 全国大学 IT 活用教育方法研究発表会,D-8,(2008)

[2] 濱田 美奈子 玉田 春昭ら 5 名「プログラミング実習における自発性測定のための感情と自発性の関連分析」, 日本教育学会 年会論文集,v.21, pp.192-195(2005)

[3] 王文 涌　池田 満　李 峰宋　「プログラミング教育における動機付け教授方法の提案と評価」, 日本教育工学会論文誌 31(3),pp,349-357(2007)

[4] 松永 豊　「プログラミング演習支援システムの開発」, 愛知教育大学研究報告書,59(教育科学編),pp,169-174(2010)

[5] アブドサラム・ダウティ, 中山 洋, 山口 正二　「目標設定と評価教示による意欲向上を目的とした授業支援システム」日本教育情報学会学会誌 25(1), 3-13,(2009)

[6] 北 栄輔 山梨 樹里 「Peer Review に基づいたプログラミング実習授業支援ツールの開発」, 名古屋高等教育研究 第 7 号 (2007)

[7] 金塚 茉利子 山本 利一 本村 猛能　「プログラミング技術習得のための学習過程の提案： 大学生による小学生を対象としたプログラミング指導」, 日本教育情報学会学会誌 24(4), 37-43,(2009)

[8] 二宮 温子「プログラミングに苦手意識を抱く学生と対象としたプログラミング授業の提案」, 慶応義塾大学環境情報学部 卒業製作 (2003)

[9] Captain Clyde H.Bell、Jr.　「リハビリテーションにおける動機づけの誘因としての競争」日本障害者リハビリテーション協会発行「リハビリテーション研究」,23,pp,23-26(1977)

[10] 高山 草二　「ビデオゲームにおける内発的動機づけとメディア嗜好性の分析」, 日本教育情報学会学会誌 15(4),pp,11-19(2000)

[11] 栗山 裕 橋下 友茂 山下 利之 「ゲームプログラミングによる情報教育の評価方法」, 日本教育工学会論文誌 28,pp,181-184(2005)

[12] 澤田 崇 山之上 卓 堤 宏智 山根 真人 「ゲーム感覚でプログラミング可能なプログラミング言語に関する研究」, 社団法人情報処理学会全国大会講演論文集 (56),pp,348-349(1998)

[13] 野口 孝文 「ゲーム作成を課題にしたプログラミング教育とその分析方法の開発」, 電子情報通信学会技術研究報告 ET,104(222),pp,1-6(2004)

[14] 尾崎 浩和 富永 浩之 林 敏浩 山崎 敏範 「ボードゲームの戦略プログラミングを題材とした Java 演習の支援システムの開発」, コンピュータと教育研究会報告 108,pp,1-8(2006)

[15] 松澤 芳昭 大岩 元「ToonTalk を利用したプログラミング教育環境の構築と実践」, 日本教育工学会第 17 回全国大会, 発表資料,(2002)

[16] 永野　和男教授　インタビュー記事「プログラミングを通して学ぶ　論理的な思考力と問題解決能力」,CUBE LAND WEB(2005)

9 ソース

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import pygame
from pygame.locals import *
import sys
import os
#import readline

x,y = 12,5  # プレイヤーの開始位置（単位：マス） #new
SCR_RECT = Rect(0, 0, 640, 480) #メインスクリーンのサイズ
GS = 32          #マップチップのサイズ
jGS = 0          #ジャンプした時に y 軸を変える為の変数
DOWN,LEFT,RIGHT,UP = 0,3,1,2    #a4w の初期向き  マップチップによりこの番宛て
key = [0]        #ユーザー入力 box
subkey = [0]     #ユーザー入力 subbox
i = 0            #キーの押された回数
s = 0            #サブ関数でキーの押された回数
goout = 0        #入力関数からの脱出 ESCkey 等 s
count = 0        #移動回数
scount = 0       #サブ関数内の移動回数
kai = "回目"
rakka = 0        #落下フラグ
str = 'I\'ve saved!!\n\n[  main BOX ]\n\n'      #セーブ用変数
substr = '\n\n[ sub BOX ]\n\n'
jump = 1         #ジャンプフラグ
#subflag = 0     #サブ関数用フラグ
before = 0
after = 0
mainn = '行動コマンド ='
sub = '魔法の書 ='
Next = 0
Previous = 0
item1 = 0


def main():
        pygame.init()
        import __main__
        global goout
        global kai
```

```
        global str
        global key
        global i
        global count
        global scount
        global DOWN,LEFT,RIGHT,UP
        global rakka
        global jump
        global jGS
        global after,before
        global subkey
        global s
        global mainn
        global sub
        global substr
        global Next
        global Previous
        global item1
#        global subflag
        screen = pygame.display.set_mode(SCR_RECT.size) #メイン画面作成
        pygame.display.set_caption("アンドリュークエスト")#タイトルバー
        Map.images[0] = load_image("hole.png")         # 穴
        Map.images[1] = load_image("carpet.png")         # 床
        Map.images[2] = load_image("dansa1.png")         # 壁
        Map.images[3] = load_image("dansa2.png")         # 壁
        Map.images[4] = load_image("dansa3.png")         # 壁
        Map.images[5] = load_image("dansa4.png")         # 壁
        Map.images[6] = load_image("dansa6.png")         # 壁
        Map.images[9] = load_image("treasure.png", -1)          #階段
# マップとプレイヤー作成
        map = Map("04")
        player = Player("andrew1", (x,y), RIGHT)
        clock = pygame.time.Clock()
        clock.tick(60)          #60 FPS (Frame Per Second)
        player.update()         # 経過フレーム数に応じて表示する画像を変える
        map.draw(screen)        # マップを描画
        player.draw(screen)     # a4w 描画
        pygame.display.update() #画面の更新
#        for event in pygame.event.get():
#                if event.type == QUIT:
#                        sys.exit()
#                if event.type == KEYDOWN and event.key == K_a:
#                        goout = 1               # 終了フラグ
```

```
#               print "何かアイテムを装備しますか？"
#               if event.type == KEYDOWN and event.key == K_1:
#                       item1 == 1
#               if item1 == 1:
#                       print Player.direction
#       item=raw_input(' 何かアイテムを装備しますか？ ')
#       if item == "1":
#               print "ジャンプシューズを装備した！"
#               item1 = 1
        print "\n\n 今回 a4w は 5 回までしか言うことを聞かないようだ"
        print mainn     # メイン BOX の中身（親切設計）
        print sub               # サブ BOX の中身（親切設計）

        while True:
                clock.tick(60)          #60 FPS（Frame Per Second）
                player.update()         # 経過フレーム数に応じて表示する画像を変える
                map.draw(screen)        # マップを描画
                player.draw(screen)     # a4w 描画
                for event in pygame.event.get():
                        if event.type == QUIT:
                                sys.exit()
                        if event.type == KEYDOWN and event.key == K_SPACE:
                                goout = 1               # 終了フラグ
#"""ユーザーからの入力を 入力 BOX(key[]) に放り込んでいく"""
# [ ユーザー入力 BOX ]
                        if event.type == KEYDOWN and event.key == K_DOWN:
                                key = key + [0] #下を押したら 0 が入る
                                i += 1                  #回数にプラス１
#                                print i+s,      kai,"  FORWARD" #ターミナルに表示
#                                print "メイン BOX =",key[1:]        # メイン BOX の中身
#                                print "サブ BOX =",subkey[1:]      # サブ BOX の中身
                                mainn += ' 前進,'
                                print mainn     # メイン BOX の中身（親切設計）
                                print sub               # サブ BOX の中身（親切設計）
                                str += '  FORWARD\n'            #tmp ファイルに書き込み

                        if event.type == KEYDOWN and event.key == K_RIGHT:
                                key = key + [1] #右を押したら１が入る
                                i += 1                  #回数にプラス１
#                                print i+s       ,kai,"  RIGHT"  #ターミナルに表示
#                                print "メイン BOX =",key[1:]
#                                print "サブ BOX =",subkey[1:]
                                mainn += ' 右回,'
```

```
                print mainn
                print sub
                str += '  RIGHT\n'              #tmp ファイルに書き込み

        if event.type == KEYDOWN and event.key == K_LEFT:
                key = key + [2] #左を押したら 2 が入る
                i += 1
#                print i+s       ,kai,"  LEFT"
#                print "メイン BOX =",key[1:]
#                print "サブ BOX =",subkey[1:]
                mainn += ' 左回,'
                print mainn
                print sub
                str += '  LEFT\n'

        if event.type == KEYDOWN and event.key == K_UP:
                key = key + [5] #上を押したら 5 が入る
                i += 1
#                print i+s       ,kai,"  JUMP"
#                print "メイン BOX =",key[1:]
#                print "サブ BOX =",subkey[1:]
                mainn += ' 跳躍,'
                print mainn
                print sub
                str += '  JUMP\n'

        if event.type == KEYDOWN and event.key == K_m:
                key = key + [6] #s を押したら 6 が入る
                i += 1
#                print i+s       ,kai,"  サブ関数突入"
#                print "メイン BOX =",key[1:]
#                print "サブ BOX =",subkey[1:]
                mainn += ' 魔法を唱えた,'
                print mainn
                print sub
                str += '  >>insert [ sub function ] \n'

        if event.type == KEYDOWN and event.key == K_SPACE:
                goout = 1               # 終了フラグ

        if event.type == KEYDOWN and event.key == K_f and s < 6:
                subkey = subkey + [0]   #f を押したら 0 が入る
                s += 1                  #回数にプラス１
```

```
#                               print i+s,      kai,"  forward" #ターミナルに表示
#                               print "メイン BOX =",key[1:]
#                               print "サブ BOX =",subkey[1:]
                               sub += ' 前進,'
                               print mainn
                               print sub
                               substr += '  forward\n'          #tmp ファイルに書き込み

                       if event.type == KEYDOWN and event.key == K_r and s < 6:
                               subkey = subkey + [1]    #r を押したら１が入る
                               s += 1                   #回数にプラス１
#                               print i+s,      kai,"  right"   #ターミナルに表示
#                               print "メイン BOX =",key[1:]
#                               print "サブ BOX =",subkey[1:]
                               sub += ' 右回,'
                               print mainn
                               print sub
                               substr += '   right\n'           #tmp ファイルに書き込み

                       if event.type == KEYDOWN and event.key == K_l and s < 6:
                               subkey = subkey + [2]    #l を押したら 2 が入る
                               s += 1
#                               print i+s,      kai,"  left"
#                               print "メイン BOX =",key[1:]
#                               print "サブ BOX =",subkey[1:]
                               sub += ' 左回,'
                               print mainn
                               print sub
                               substr += '  left\n'

                       if event.type == KEYDOWN and event.key == K_j and s < 6:
                               subkey = subkey + [5]    #j を押したら 5 が入る
                               s += 1
#                               print i+s,      kai,"  jump"
#                               print "メイン BOX =",key[1:]
#                               print "サブ BOX =",subkey[1:]
                               sub += ' 跳躍,'
                               print mainn
                               print sub
                               substr += '  jump\n'

#"""一時ファイルに保存"""
               f = open('tmp.txt', 'w') # 書き込みモードで開く
```

```
                f.write(str) # 引数の文字列をファイルに書き込む
                f = open('subtmp.txt', 'w') # 書き込みモードで開く
                f.write(substr) # 引数の文字列をファイルに書き込む
                if goout == 1: break            # Esc キーが押されたら終了
                f.close() # ファイルを閉じる

#""" 入力 BOX を舐めていく"""
# [ 実行 BOX ]
        while count < i:        #入力された回数分　回す
                count += 1
                map.draw(screen)  # マップ描画
                player.draw(screen)# プレイヤー描画
                pygame.display.update()
                player.update()
#                print "key[]=",key[1:],"count= ",count                          #入 力 BOX
の中身　確認用
                if count > 5:           #消費ステップ数オーバー
                        break
                if rakka == 1:
                        break
                for event in pygame.event.get():        #強制終了用
                        if event.type == QUIT:                  #強制終了用
                                sys.exit()                                      #強制終了用
                        if event.type == KEYDOWN and event.key == K_ESCAPE: #強制終了用
                                sys.exit()                                      #強制終了用
                if item1 == 1:  #ジャンプシューズ装備しているなら
                        player.move(6,map)
                if key[count] == 0:             #直進って事
#                         print "入力 BOX の値=",key[count]     #ターミナルに表示する　テスト用
                        player.move(0,map)
                if key[count] == 1:             #回れ右って事
#                         print "入力 BOX の値=",key[count]     #ターミナルに表示する　テスト用
                        player.move(1,map)
                if key[count] == 2:             #回れ左って事
#                         print "入力 BOX の値=",key[count]     #ターミナルに表示する　テスト用
                        player.move(2,map)

                if key[count] == 5:             #jump って事
#                         print "入力 BOX の値=",key[count]     #ターミナルに表示する　テスト用
                        player.move(5,map)
                player.update()
                pygame.display.update()
                if key[count] == 6:             # サブ関数
```

```
#                        print "関数入力 BOX の中身 = ",subkey[1:]
                       while scount < s:
#                               print "scount <  s = ",scount,"<",s
                               scount += 1
                               map.draw(screen)  # マップ描画
                               player.draw(screen)# プレイヤー描画
#                               print "subkey[]=",subkey[1:],"scount= ",scount
入力 BOX の中身　確認用
                               if rakka == 1:
                                       break
                               if subkey[scount] == 0:          #直進って事
#                                       print "関数入力 BOX の値=",subkey[scount]       #
ターミナルに表示する　テスト用
                                       player.move(0,map)
                               if subkey[scount] == 1:          #回れ右って事
#                                       print "関数入力 BOX の値=",subkey[scount]       #
ターミナルに表示する　テスト用
                                       player.move(1,map)
                               if subkey[scount] == 2:          #回れ左って事
#                                       print "関数入力 BOX の値=",subkey[scount]       #
ターミナルに表示する　テスト用
                                       player.move(2,map)
                               if subkey[scount] == 5:          #jump って事
#                                       print "関数入力 BOX の値=",subkey[scount]       #
ターミナルに表示する　テスト用
                                       player.move(5,map)
                       scount = 0              #またサブ関数を使えるように
               player.update()
               pygame.display.update()

               if item1 == 1:  #ジャンプシューズ装備しているなら
                       player.move(6,map)
#                if item1 == 1:  #ジャンプシューズ装備しているなら
#                        player.move(6,map)
               player.update()
               pygame.display.update()

#***落下***
       if rakka == 1:
               map.draw(screen)  # マップ描画
               player.draw(screen)  # プレイヤー描画
               pygame.display.update()
               sysfont = pygame.font.SysFont(None, 80)
```

```
                text = sysfont.render("a 4 w  D e a d!", True, (80,0,0)) #第３引数文字色
                while True:
        # screen.fill((0,0,255)) #別画面でゴール表示するなら on
        # テキストを描画する
                        screen.blit(text, (230,170))
                        pygame.display.update()
                        for event in pygame.event.get():
                                if event.type == QUIT:
                                        sys.exit()
                                if event.type == KEYDOWN and event.key == K_ESCAPE:
                                        sys.exit()


        if i > 5:
                print "「a4w が命令に従わない。疲れたようだ...」"
#***階段に到達していればゴール***
        print "\n 最終座標 (x,y)",player.x,player.y
        print "最終床番号",map.map[player.y][player.x]
        print "消費ステップ数 = ",i," 回"

        if map.map[player.y][player.x] == 9:
                map.draw(screen)  # マップ描画
                player.draw(screen)  # プレイヤー描画
                pygame.display.update()
                sysfont = pygame.font.SysFont(None, 80)
                sysfont2 = pygame.font.SysFont(None, 40)
                text = sysfont.render("Get the Jumpshoes!", True, (100,0,250)) #第 3 引 数 文
字色
                text2 = sysfont2.render("Go to the URL!!", True, (100,0,200))
        #***セーブするかどうか***
                print "セーブしますか？"
                save=raw_input(' y か n を入力してください')

                if save == 'y':
                        f = open('save04.txt', 'w') # 書き込みモードで開く
                        f.write(__main__.str) # 引数の文字列をファイルに書き込む
                        f = open('save04.txt', 'a+') # 書き込みモードで開く
                        f.write(__main__.substr) # 引数の文字列をファイルに書き込む
                        print "セーブしたよん。おつかれ"
                else:
                        print "セーブしませんでした。a4w クエストの画面にカーソル合わせて Esc"
                print "\n\n 下の URL を開くと幸せになれる。"
                print "\n\n   http://www0.ise.osaka-sandai.ac.jp/~a09mh05/jump.html\n\n"
                print "\nNext Stage code = python 07ruby.py"
```

```
                print "このステージの最短攻略ステップ数 = 消費ステップ数 : 4 \n"
                shoesImg = pygame.image.load("./data/shoes.png").convert()

                while True:
        # screen.fill((0,0,255)) #別画面でゴール表示するなら on
        # テキストを描画する
                        screen.blit(text, (70,120))
#                        screen.blit(shoesImg, (200,150))
                        screen.blit(text2, (260,400))
                        pygame.display.update()
                        for event in pygame.event.get():
                                if event.type == QUIT:
                                        sys.exit()
                                if event.type == KEYDOWN and event.key == K_ESCAPE:
                                        sys.exit()
#       if map.map[player.y][player.x] == 9 and i >= 6:
#               print "消費ステップオーバーｏｏｏ"
#               print "考えろ、君ならクリア出来る。"
#***Esckey 入力があるまで待つ***(ゴール未到達用)　なかったらすぐ消えるから
        while True:
                map.draw(screen)  # マップ描画
                player.draw(screen)  # プレイヤー描画
                pygame.display.update()
        #    print x,y
                for event in pygame.event.get():
                        if event.type == QUIT:
                                sys.exit()
                        if event.type == KEYDOWN and event.key == K_ESCAPE:
                                sys.exit()
                        else:
                                pygame.time.wait(10)

#***ファイル読み込み***
def load_image(filename, colorkey=None):
        filename = os.path.join("data", filename)
        try:
                image = pygame.image.load(filename)
        except pygame.error, message:
                print "Cannot load image:", filename
                raise SystemExit, message
        image = image.convert()
        if colorkey is not None:
                if colorkey is -1:
```

```
                colorkey = image.get_at((0,0))
            image.set_colorkey(colorkey, RLEACCEL)
    return image

def split_image(image):                    #画像を切る
    """128x128 のキャラクターイメージを 32x32 の 16 枚のイメージに分割
    分割したイメージを格納したリストを返す"""
    imageList = []
    for i in range(0, 128, GS):
        for j in range(0, 128, GS):
            surface = pygame.Surface((GS,GS))
            surface.blit(image, (0,0), (j,i,GS,GS))
            surface.set_colorkey(surface.get_at((0,0)), RLEACCEL)
            surface.convert()
            imageList.append(surface)
    return imageList

class Map:
    images = [None] * 256  # マップチップ（番号->イメージ）
    def __init__(self, name):
        self.name = name
        self.row = -1   #       行数
        self.col = -1   #       列数
        self.map = []   #       2 次元リストのマップ受け取り
        self.load()

    def draw(self, screen):
        """マップを描画する"""
        for r in range(self.row):
            for c in range(self.col):
                screen.blit(self.images[self.map[r][c]], (c*GS,r*GS))

#***移動判定***
    def is_movable(self, x, y):
        global before,after
        global jump
        nextpoji = self.map[x][y]       # 次の床の値を nextpoji に入れる
        #行き先は移動可能か
#           if x < 0 or x > self.col-1 or y < 0 or y > self.row-1:          #マップの
範囲内かどうか
#               print "いけまへん"
#               return False
        if self.map[x][y] == 9:         # 行き先が cola なら行ってよし
```

```
                        return True
                if jump < nextpoji:                     # 次の床の値　と　ジャンプフラグの値を
比べる。越えてなければぶつかる
                        print "「イタッ!!!」"
                        return False
                if self.map[x][y] == 0:         # 数字のところには移動出来ない
#                        print "rakkax",y,"rakkay",x             #位置確認
                        global rakka
                        rakka = 1
#                print "ismovable"
                return True                                     # 移動出きるなら True を返す

        def is_jumpable(self, x, y):
                global before,after
                global jump
                global jGS
                nextpoji = self.map[x][y]       # 次の床の値を nextpoji に入れる
                if nextpoji == jump or nextpoji == jump - 1:    # 同じ階層でジャンプした場合
                        print "「その場ジャンプ」"
                        return False
                if nextpoji == jump + 1 or nextpoji == jump + 2:        # 1段だけ上なら ジャ
ンプ実行
                        jump += 2
                if self.map[x][y] == 9:         # 行き先が cola なら行ってよし
                        return True
#                if self.map[x][y] == 3 or self.map[x][y] == 5:  # 床の値が 3 か 5 なら マイ
ナス 1 する
#                        nextpoji = self.map[x][y] - 1
                if jump < nextpoji:                     # 次の床の値　と　ジャンプフラグの値を
比べる。越えてなければぶつかる
                        print "「イタッ!!!」"
                        return False


                else:
                        return True                                     # 移動出きるなら
True を返す

        def jumpshoes(self, x, y):
                global jump
                global jGS
                nextpoji = self.map[x][y]
                if nextpoji == jump + 1 or nextpoji == jump + 2:        # 1段だけ上なら ジャ
ンプ実行
```

```
                jump += 2
                return True
        if self.map[x][y] == 9:         # 行き先が cola なら行ってよし
                return True
        else:
                return False                                    # 移動出きるなら
True を返す

#***外部ファイルからマップをロード***  #map という 2 次元リストを生成
    def load(self):
        file = os.path.join("data", self.name + ".map")
        fp = open(file)
        lines = fp.readlines()  #行単位で読み込んでリスト lines に格納
        row_str, col_str = lines[0].split()
        self.row, self.col = int(row_str), int(col_str)
        for line in lines[1:]:
                line = line.rstrip()    #行末の改行コードを取り除く
                self.map.append([int(x) for x in list(line)])   # .append で map リス
トにデータを追加
                                                                # 1 行ずつ読み込ん
で、int に変えて追加
        fp.close()
#        print map

class Player:
    animcycle = 4           #何枚でチェンジさせるか
    frame = 0                       #frame 変数
    def __init__(self, name, pos, dir):
        self.name = name        # プレイヤー名
        self.images = split_image(load_image("%s.png" % name))
        self.image = self.images[0]     #  描画中のイメージ
        self.x, self.y = pos[0], pos[1] #キャラの座標受け取り（単位:マス）
        self.rect = self.image.get_rect(topleft=(self.x*GS, self.y*GS))
        self.direction = dir
    def update(self):
        #frame 変数の値によって絵（a4w の向き）を変える
        self.frame += 1
        self.image = self.images[(self.direction%4)*4+self.frame/self.animcycle%4]    #
縦*横

# ［行動］
    def move(self,dir,map):
        global jGS
```

```
                global jump
                global after,before
#                print "player move x",self.x,"y",self.y                    #現在の x 座標 y 座
標
#                print "move 中 現在地",map.map[self.y][self.x]                 #現在の床の値
#                print "move 中 jump フラグ",jump
                before = map.map[self.y][self.x]                           # 別名　現在位置の床
番号
                self.update()             # 経過フレーム数に応じて表示する画像を変える
                """入力 BOX が 1 の時"""
                if dir == 1:
                        self.direction -= 1              #回れ右した
                        print "右に回った"
#                        print "回れ右 dir=",self.direction
                        pygame.time.wait(300)   #ちょい停止

                """入力 BOX が 2 の時"""
                if dir == 2:
                        self.direction += 1              #回れ左した
                        print "左に回った"
#                        print "回れ左 dir=",self.direction
                        pygame.time.wait(300)   #ちょい停止

                """入力 BOX が 0 の時"""
                if dir == 0:
#                        print "move","direction",self.direction
#                """a4w が右を向いていたら"""
                        if self.direction%4 == 1:
                                if map.is_movable(self.y, self.x+1):
                                        self.x += 1
                                        self.rect.left += GS
                                        print "東に進んだ"
                                pygame.time.wait(300)   #ちょい停止

#                """a4w が下を向いていたら"""
                        if self.direction%4 == 0:
                                if map.is_movable(self.y+1, self.x):
                                        self.y += 1
                                        self.rect.top += GS
                                        print "南に進んだ"
                                pygame.time.wait(300)   #ちょい停止
#                """a4w が左を向いていたら"""
                        if self.direction%4 == 3:
```

```
                        if map.is_movable(self.y, self.x-1):
                                self.x -= 1
                                self.rect.left -= GS
                                print "西に進んだ"
                        pygame.time.wait(300)   #ちょい停止

#               """a4w が上を向いていたら"""
                        if self.direction%4 == 2:
                                if map.is_movable(self.y-1, self.x):
                                self.y -= 1
                                self.rect.top -= GS
                                print "北に進んだ"
                        pygame.time.wait(300)   #ちょい停止


                if dir == 5:
#                        print "before =",before,"after =",after
#                        if before == after -1:
#                        jump += 2                       #jump フラグ  をたてる. 2 段目に 2 と 3
が混在している為、+2
                        jGS = 15                        #y 座標変更
#                        print "jump move"

#               """a4w が右を向いていたら"""
                        if self.direction%4 == 1:
                                if map.is_jumpable(self.y, self.x+1):
                                self.x += 1
                                self.rect.left += GS
                                if before == 1:         # Grand Floor からのジャンプ
は視覚的にもジャンプ
                                        self.rect.top -= jGS
                                        jGS = 0                         # 羽 ば た き
防止
                                print "東に跳んだ"
                        pygame.time.wait(300)   #ちょい停止

#               """a4w が下を向いていたら"""
                        if self.direction%4 == 0:
                                if map.is_jumpable(self.y+1, self.x):
                                self.y += 1
                                self.rect.top += GS
                                if before == 1:
                                        self.rect.top -= jGS
                                        jGS = 0
```

```
                                print "南に跳んだ"
                        pygame.time.wait(300)   #ちょい停止

#               """a4w が左を向いていたら"""
                if self.direction%4 == 3:
                        if map.is_jumpable(self.y, self.x-1):
                                self.x -= 1
                                self.rect.left -= GS
                                if before == 1:
                                        self.rect.top -= jGS
                                        jGS = 0
                                print "西に跳んだ"
                        pygame.time.wait(300)   #ちょい停止

#               """a4w が上を向いていたら"""
                if self.direction%4 == 2:
                        if map.is_jumpable(self.y-1, self.x):
                                self.y -= 1
                                self.rect.top -= GS
                                if before == 1:
                                        self.rect.top -= jGS
                                        jGS = 0
                                print "北に跳んだ"
                        pygame.time.wait(300)   #ちょい停止

            after = map.map[self.y][self.x]
#               print "before= ",before,"after= ",after                                 # before after
い確認用  <違い有り>

            if before > after:                                                  # 落下して
きたら
#                       print "jumpbefore",jump,"現在地",map.map[self.y][self.x]
                        if map.map[self.y][self.x] % 2 == 0:
                                jump = map.map[self.y][self.x] + 1              # jump フ
ラグは着地した床の番号にする = 浮遊はダメよ
                        else:
                                jump = map.map[self.y][self.x]
                        if after == 1:                          # 落下先が Grand Floor なら
                                self.rect.top += 15             # a4w の足を床に着ける = 地
に足つけてね
#                       print "jumpafter",jump
                        self.update()
            self.update()
```

```
                pygame.display.update()

                if dir == 6:    #ジャンプシューズ装備時
                        jGS = 15                        #y 座標変更

#                """a4w が下を向いていたら"""
                        if self.direction%4 == 0:
                                if map.jumpshoes(self.y+1,self.x):
                                        self.y += 1
                                        self.rect.top += GS
                                        if before == 1:
                                                self.rect.top -= jGS
                                                jGS = 0
                                        print "南に跳んだ　ジャンプシューズ　"
                                pygame.time.wait(300)   #ちょい停止

#                """a4w が右を向いていたら"""
                        if self.direction%4 == 1:
                                if map.jumpshoes(self.y, self.x+1):
                                        self.x += 1
                                        self.rect.left += GS
                                        if before == 1:         # Grand Floor からのジャンプ
は視覚的にもジャンプ
                                                self.rect.top -= jGS
                                                jGS = 0                         # 羽 ば た き
防止
                                        print "東に跳んだ　ジャンプシューズ　"
                                pygame.time.wait(300)   #ちょい停止

#                """a4w が左を向いていたら"""
                        if self.direction%4 == 3:
                                if map.jumpshoes(self.y, self.x-1):
                                        self.x -= 1
                                        self.rect.left -= GS
                                        if before == 1:
                                                self.rect.top -= jGS
                                                jGS = 0
                                        print "西に跳んだ　ジャンプシューズ　"
                                pygame.time.wait(300)   #ちょい停止

#                """a4w が上を向いていたら"""
                        if self.direction%4 == 2:
                                if map.jumpshoes(self.y-1, self.x):
```

```
                                        self.y -= 1
                                        self.rect.top -= GS
                                        if before == 1:
                                                self.rect.top -= jGS
                                                jGS = 0
                                        print "北に跳んだ　ジャンプシューズ　"
                                pygame.time.wait(300)    #ちょい停止

#               after = map.map[self.y][self.x]
#               if before > after:                                              # 落下して
きたら
#                       print "jumpbefore",jump,"現在地",map.map[self.y][self.x]
#                       if map.map[self.y][self.x] % 2 == 0:
#                               jump = map.map[self.y][self.x] + 1              # jump フ
ラグは着地した床の番号にする = 浮遊はダメよ

#                       else:
#                               jump = map.map[self.y][self.x]
#                       if after == 1:                          # 落下先が Grand Floor なら

#                               self.rect.top += 15             # a4w の足を床に着ける = 地
に足つけてね
#                       print "jumpafter",jump
#                       self.update()
#               pygame.display.update()

       def draw(self, screen):
               screen.blit(self.image, self.rect)
if __name__ == "__main__":

   main()
```