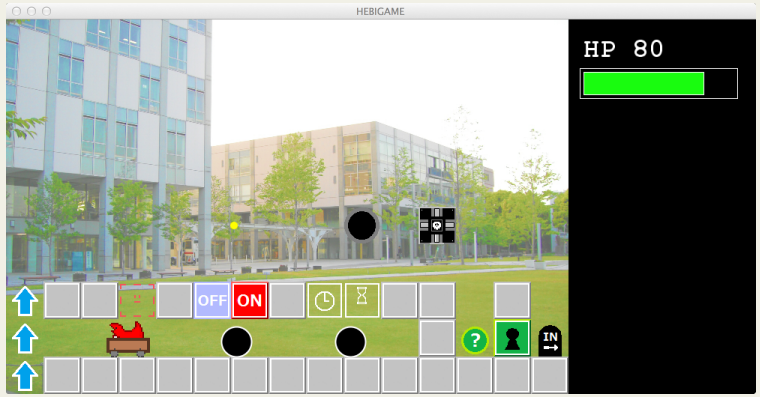
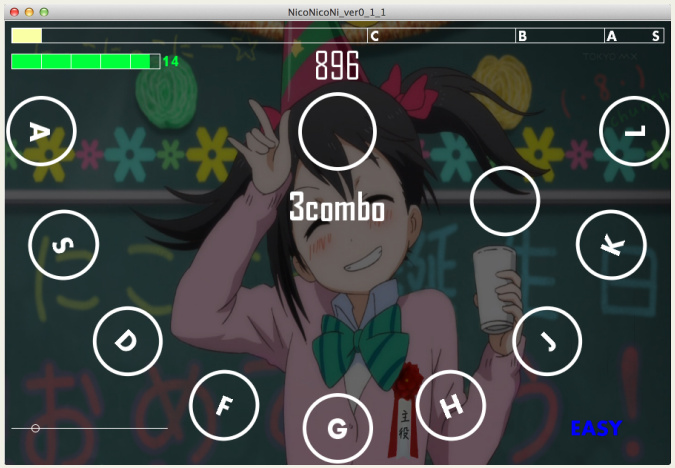
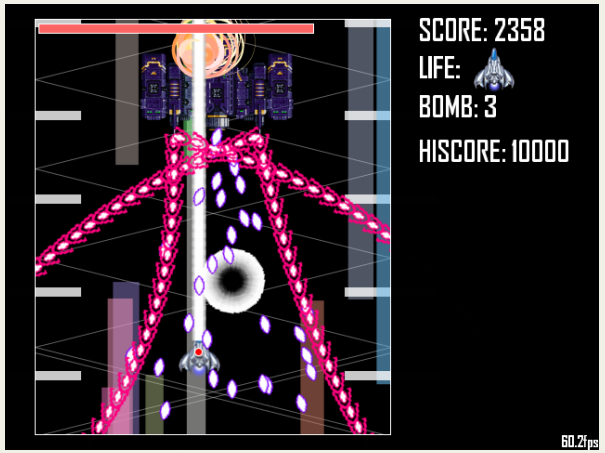
2014年度1年前期の最終課題







「ラブライブ！」とは？

- 2013年1月からスタートしたテレビアニメ
- 9人の女の子がスクールアイドルを結成
  - 高校の廃校を阻止するため
  - Love Liveに向けて特訓が始まる！

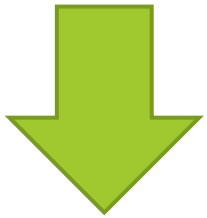
「スクフェス」とは？

- テレビアニメ「ラブライブ！」が原作のソーシャルゲーム
- リズムアクション＆アドベンチャー

- つまり音ゲー

作成理由

- PCでスクフェスをやりたい！
- ゲームを作りたい！



- 作ればええやん？

結果


ラブライブ!
スクールアイドルフェスティバル
画面をタッチしてください
©2013 プロジェクトラブライブ! ©KLabGames ©bushiroad All Rights Reserved.
JASRAC許諾 第9014302001Y43032号
JASRAC許諾 第9014302002Y43032号

工夫点

- CSVデータの活用
  - ランキング機能の実装
  - セーブの実現

今後の展望

- スキルの実装
  - スコアアップ
  - HP回復
  - 判定強化

今後の展望

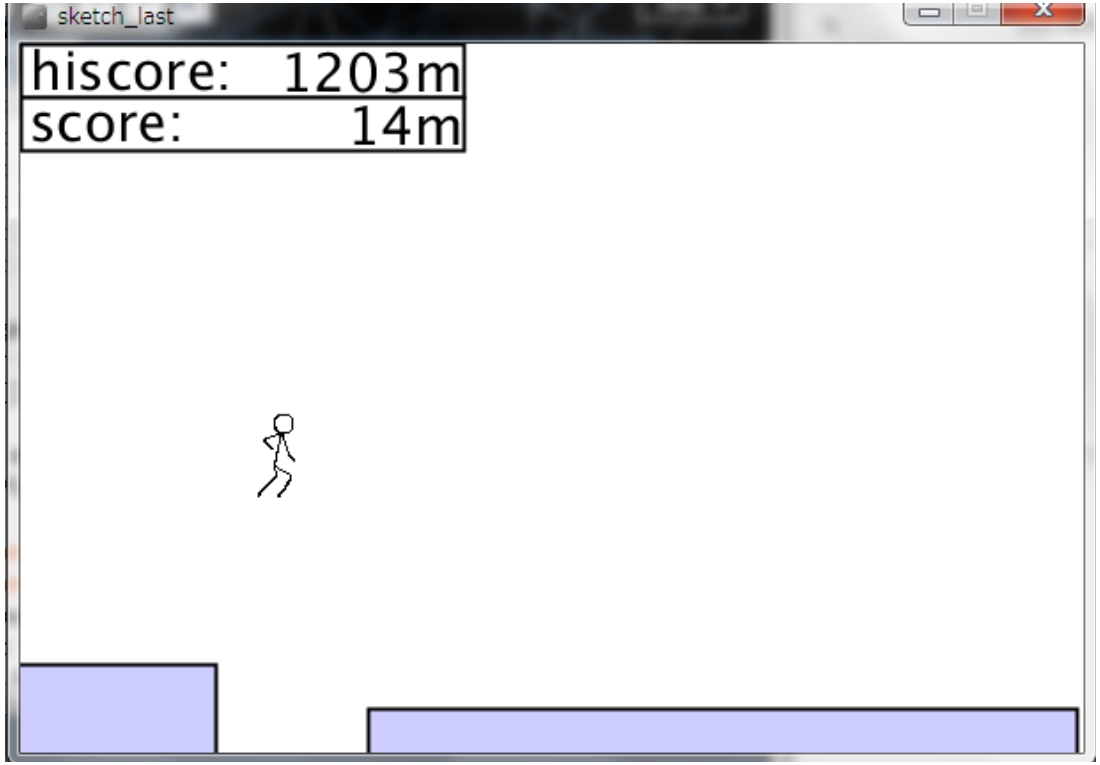
- 長押しの実装
  - keyPressed()とkeyReleased()
    - キーが押されたら判定開始
    - キーが離されたら判定終了

今後の展望

- 音ゲーにしたい！
  - プログラムと音楽の同期
    - ○が発生するタイミングを記憶
    - ソースコードが膨大に…

作品概要

・ジャンプアクションゲーム



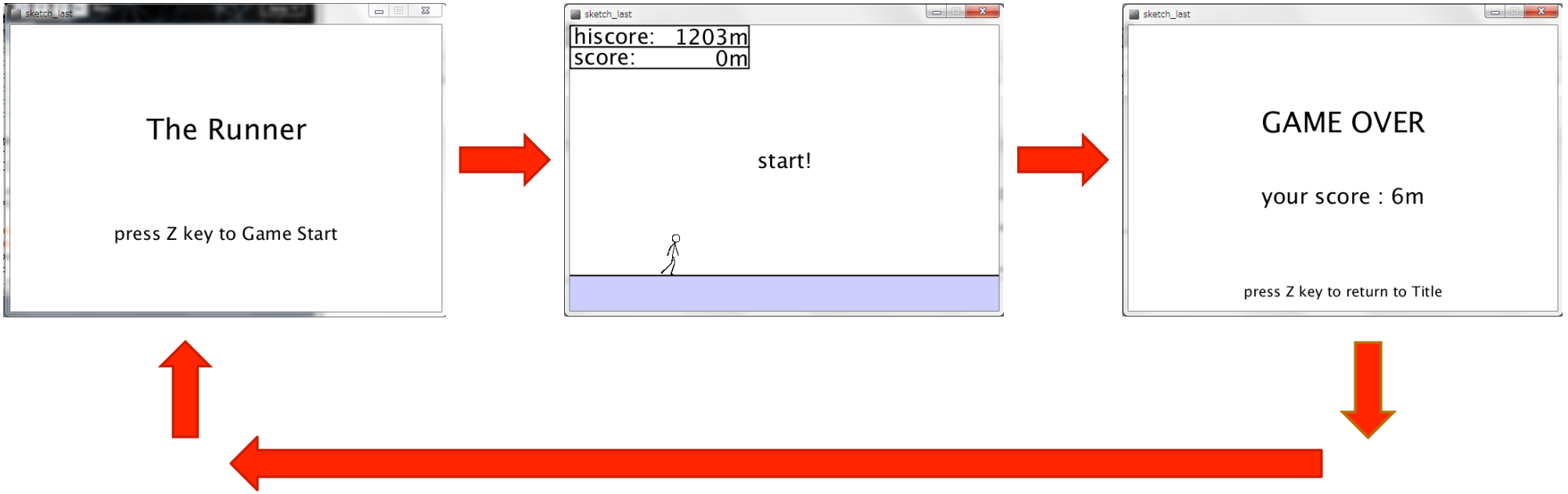
穴をZキーでジャンプして跳び越えながらどこまで走り続けられるか競う

2段ジャンプも可

走り続けているとだんだんスピードがアップする

ゲームの構成

このプログラムはタイトル画面、ゲーム画面、ゲームオーバー画面を遷移する



苦労した点１

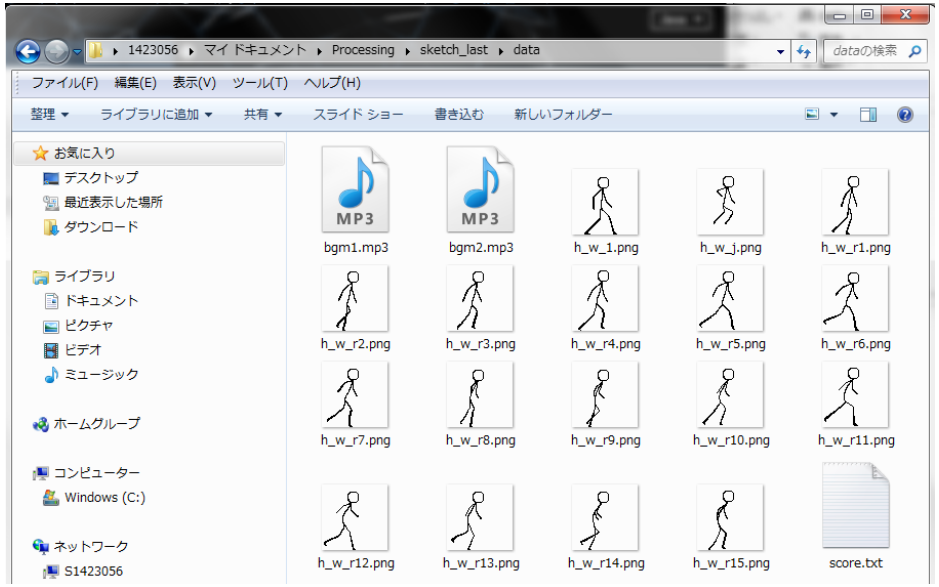
・オブジェクト指向プログラミング

　ーだいたいがクラスで構成されている。

・キャラクターの画像とBGMは自作

　ー画像は64×64を16枚ほど、

　　BGMは簡単なものを2つ



苦労した点２

・キャラクターと足場の当たり判定
　ーテストプレイを重ねて微調整する
　　のに時間がかかった

```
for(int i = 0;i<floor.length;i++){
  float fx = floor[i].posX;
  float fxl = floor[i].posX - floor[i].f_width/2;
  float fxr = floor[i].posX + floor[i].f_width/2;
  float fh = floor[i].posY - floor[i].f_height/2;
  if(posY+H_imgsize/2 > fh){
    if(posX > fx && posX < fxr+5){
      posX = fxr+5;
    }else if(posX < fx && posX > fxl-5){
      posX = fxl-5;
    }
  }else if(vY <= 0 && posY+H_imgsize/2 < fh +10 && posY+H_imgsize/2 > fh -10){
      if(posX > fxl-10 && posX < fxr+10){
        posY = fh - H_imgsize/2;
        isFRY = false;
        sjump = true;
        break;
      }else{
        isFRY = true;
    }
  }
}
```

プログラムのソースコード1

・メインタブのsetupとdraw

```
void setup(){
  size(600,400);

  minim = new Minim(this);
  bgm1 = minim.loadFile("bgm1.mp3");
  bgm2 = minim.loadFile("bgm2.mp3");

  textAlign(CENTER);
  textSize(40);
  fill(0);
  imageMode(CENTER);

  keyState = new boolean[256];
  p_keyState = new boolean[256];
  for(int t = 0;t<keyState.length;t++){
    keyState[t] = false;
    p_keyState[t] = false;
  }
  gamestate = new Title();
}
```

```
void draw(){
  background(255);
  gamestate.Update();
  gamestate.Draw();
  if(!p_keyState['q'%256] && keyState['q'%256]) debugmode
  for(int i = 0;i<256;i++){
    p_keyState[i] = keyState[i];
  }
}
```

注：draw関数のほうが短くなった。

他に変数宣言やキー入力の処理など

プログラムのソースコード2

Playerタブ

```
void Update(){
    counter++;

  if(!isFRY){
    vY = 0;
    if(keyState['z'%256]){
      isFRY = true;
      vY = 7.0;
    }
  }else{
    if(sjump && !p_keyState['z'%256] && keyState['z'%256]){
      sjump = false;
      vY = 7.0;
    }
    vY -= aY;
  }
  posY -= vY;
```

変数宣言、zキーを押したときの動作処理、
先ほど紹介した足場との当たり判定など

```
  if(counter >= pace){
    counter = 0;
    if(!isFRY && i < img.length){
      i++;
      if(i == img.length)i = 4;
    }
  }
}

void Draw(){
  if(isFRY){
    image(jmp,posX,posY);
  }else{
    image(img[i],posX,posY);
  }
}

void Paceup(){
  if(pace > 1) pace--;
  if(aY < 0.4) aY += 0.02;
}
```

画像の描画など

プログラムのソースコード3

キー入力について

メインタブ　上

```
boolean keyState[];
boolean p_keyState[];
```

メインタブsetup関数内

```
keyState = new boolean[256];
p_keyState = new boolean[256];
```

メインタブdraw関数内　最後の行

```
for(int i = 0;i<256;i++){
  p_keyState[i] = keyState[i];
}
```

メインタブ内

```
void keyPressed(){
  if(0<=key && key<256){
    keyState[key] = true;
  }else if(0<=keyCode && keyCode<256){
    keyState[keyCode] = true;
  }
}

void keyReleased(){
  if(0<=key && key<256){
    keyState[key] = false;
  }else if(0<=keyCode && keyCode<256){
    keyState[keyCode] = false;
  }
}
```

Playerタブ　Update関数内

```
if(sjump && !p_keyState['z'%256] && keyState['z'%256]){
  sjump = false;
  vY = 7.0;
}
```

ゲーム中ではzキーしか使用しないが、
実はすべてのキー入力を利用できる。

また、keyState配列とp_keyState配列を
両方使うことでキーを押された1フレーム
だけ処理を行うことができる。

注：複数キー入力はあるwebサイト
（http://d.hatena.ne.jp/sokamura/20100103/1262530219）を参考にした

今後なにか付け足していくとしたら

・障害物や敵キャラクター

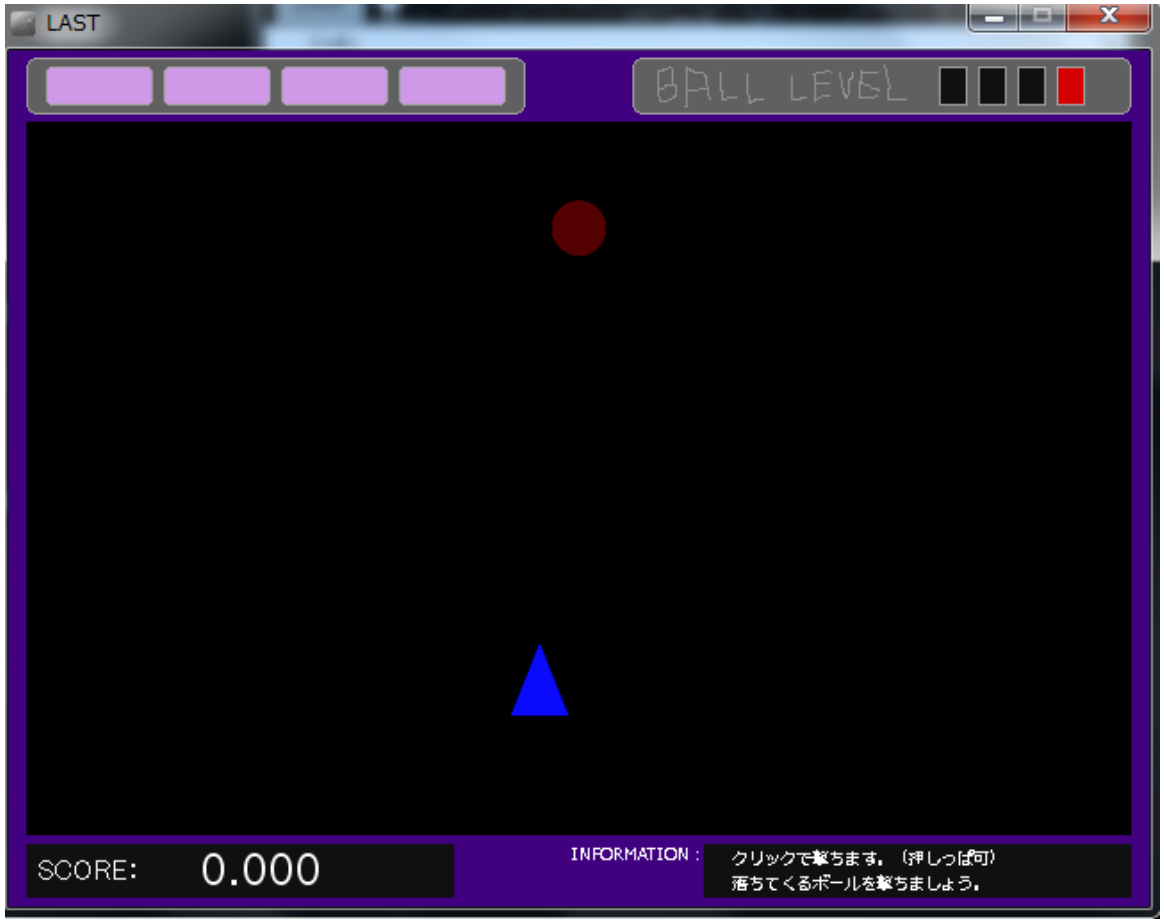
・得点アイテム

・難易度別のモード

などなど本当はやりたいことはたくさんあった…

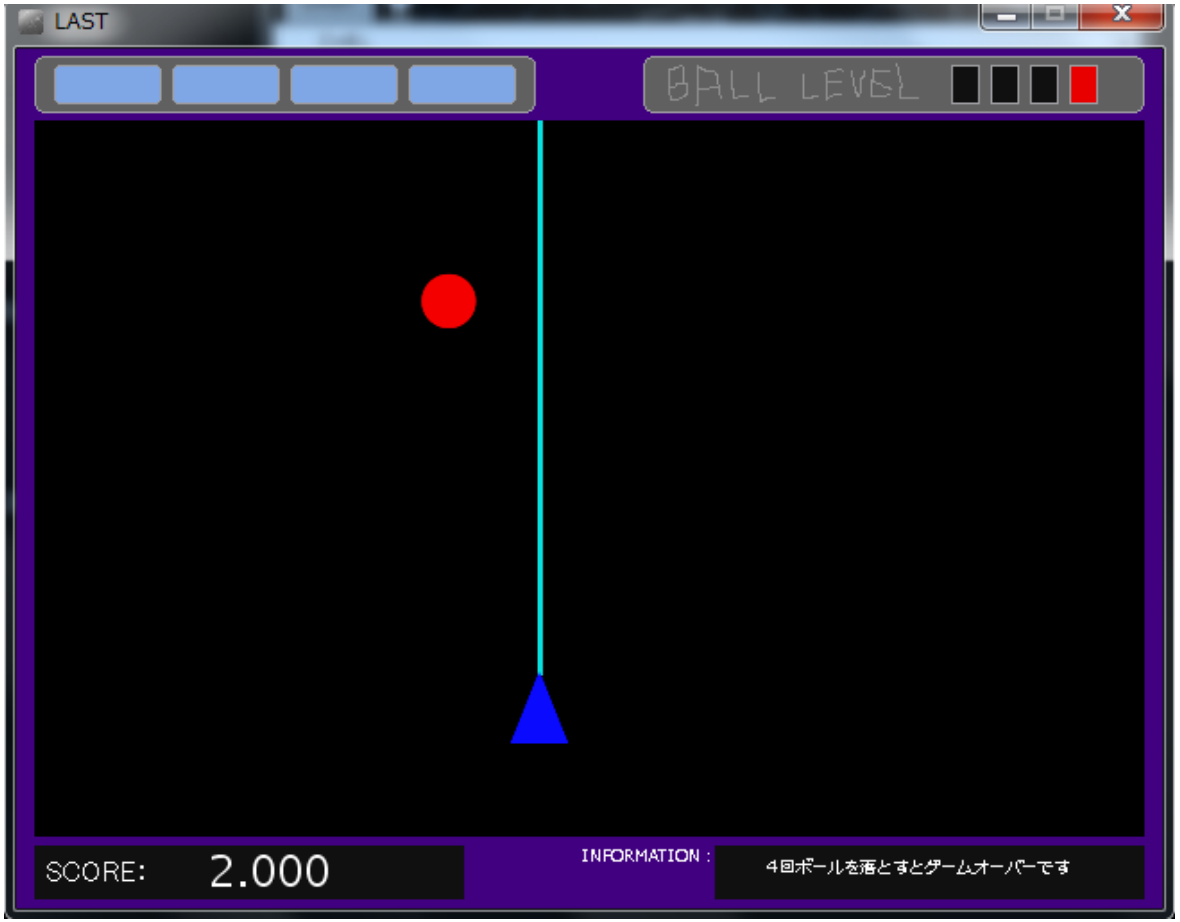
最初はこんな画面で始まります。
マウスをクリックするとゲーム開始です。

ゲームが始まるとボールが落ちてきます。
左下に操作方法とかが書かれていますね・。・

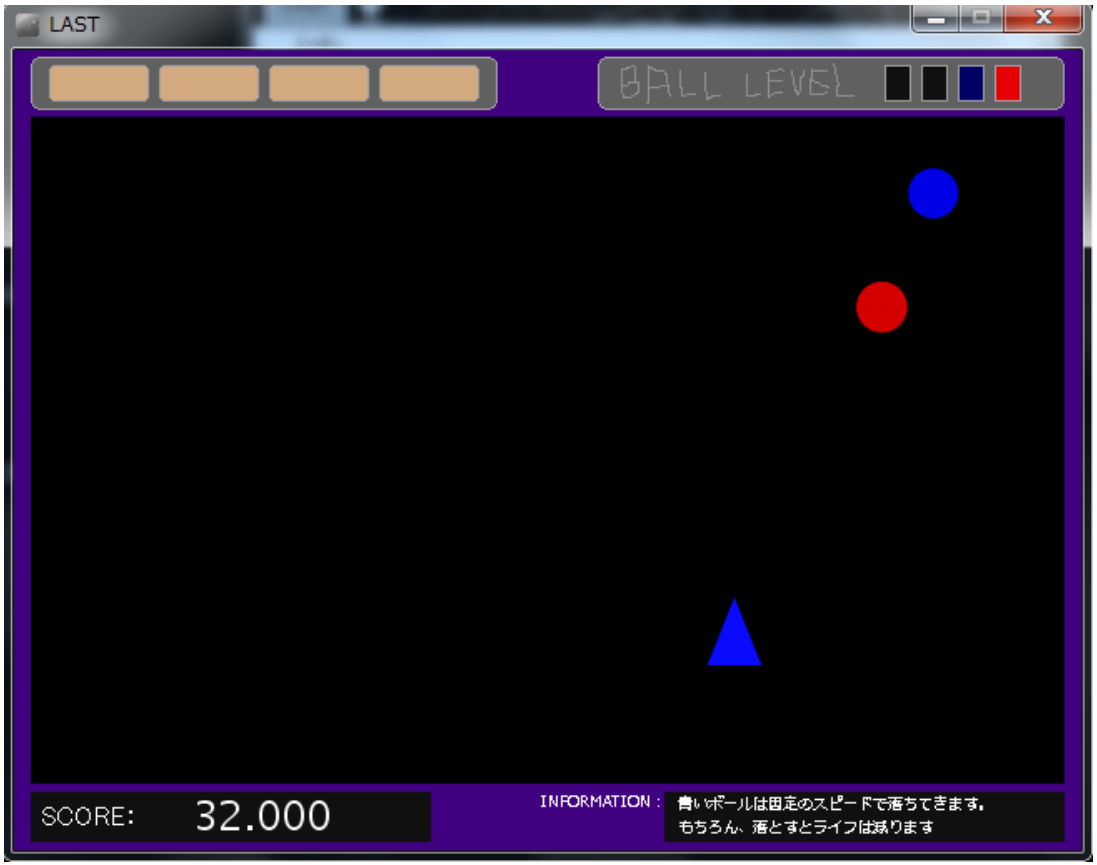


こんな感じのプログラム、実は授業でみなさん過去に制作しています。

覚えていますか？

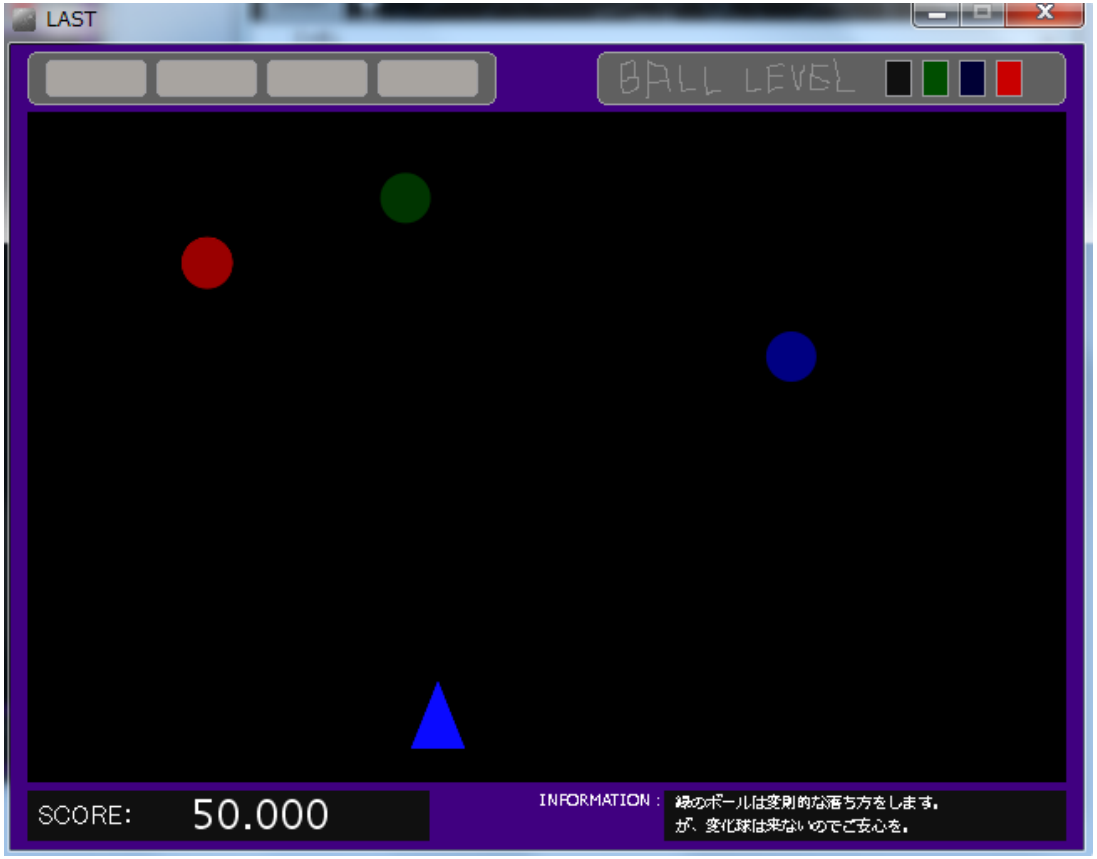


ここから、このプログラムについて軽く説明します。



- 光る円を撃つゲームです。
- ゲームが進むと円は増えます。
- マウスで操作します。
- 左上のゲージはライフです。
- ライフは電池制です。
- 左下の数字はスコアです。
- 右下に何らかの情報が。
- ライフが無くなると閉店。

続き。



- 右上のやつは
  光っているランプの色の
  円が落ちてくるよ。の表示。
- ドラッグ移動で撃つ場合は、
  円の下で一瞬止める感じにしないとスルーされるので注意。

- 電池なので回復は無い・。・

円に関する簡単な説明。

- 赤：始めだけ易しい。
  スコアに応じて
  徐々に速くなる。
  加速に際限はない。
- 青：固定スピード。
- 緑：変則スピード。
  変化球ではない。乱。
- 虹：たまに豪速球。乱。

後半戦

- 後半戦はこんな感じの絵図になるでしょう。
- 静止画だと伝わらないと思うが、赤いボールも１６０あたりからかなり速くなります。
- キーボード操作だと
  まず間に合いません・。・

お気付きだと思いますが

- ２枚抜き等、出来ます。
- ←の画面の場合は４枚抜き出来ますね。
- 円を撃った時は色によって異なるSEが鳴りますが、２枚抜きだと２つ同時に鳴りますよ。

ライフが無くなると・・・

- 左の画面のようになる。
- 閉店→ゲーム終了。
- クリックするとスコアが表示される。
- 閉店スキンの元ネタは
  提督なら分かる・。・
- リスタート機能は無い。
- ハイスコア機能も無い。

難易度に関してですが、
スコアに応じて以下の通りになります。

- ０～２０　　：初心者向け
- ２０～４０　　：初級
- ４０～６０　　：中級
- ６０～１００　：上級
- １００～１２０：超級
- １２０～２００：地獄級
- ２００～２６０：超地獄級
- ２６０～３００：哲学
- ３００～３６０：絶地獄級
- ３６０～　　：真理

ん？

- 売り・・・？
- 売りとしては・・・。

- オープンキャンパスで体験プレイさせれれば
- Processingでこんなゲーム作れんの？みたいな感じに
- OC訪問者Processingに興味を持ってくれるかも？な点

- シューティングゲームもどきのようなプログラムを作る際、
  「応用すればこんなのも作れます」みたいな感じの
  参考資料になればいいな～　な点

の２点ですね。

時間が余りましたね。

- 実際にプレイしてみましょうか(´◔ ‸ ◔)ﾉ
- （※自分の自己ベストは２４０点くらいです・。・）
- プレイしたいという方は、ポートフォリオの方から落とせるらしいので、そこからDLして、どうぞ。
- （一応、自分のポートフォリオのアイコンはスライム娘ですよとだけ）

発表の流れ

- 作成理由
- 目的
- 制作物の紹介
- 作成方法
- 考察
- 感想

制作理由･目的

- 制作理由

タイピングの練習をしたいと思い、自分で作ったらやっていて楽しいものが作れると考えたため。

- 目的

私がこのタイピングゲームで練習してタイピングが正確で打つ速度が上昇するようになるものを作る。

制作物

- 初めの画面からキーボードの1・2・3で難易度選択をした後スペースキーでタイピングを始める。
- 制限時間と何個間違えずにタイプできたかが点数でわかるものがついていて、まちがえると点数が一点減る。
- タイピングが終わると何点取れたかとその点数による評価に画面が変わる。
- 0を押すと初めの画面に戻り何回もタイピングができる。

制作方法

- 制限時間はint型やboolean 型を使いtであらわして、ソースコードから簡単に変えられるようにした。
- ローマ字をランダムに表示するのにはchar型を使用した。
- 難易度選択を作るためにswitch～case～if~breakを使用した。
- ゲームの説明の部分を日本語にするためにTeraPadで保存し、吸い出した。

考察

- よくできた点：制限時間を設けられた点・textファイルを読み込んで、日本語で説明するようにした点・0を押せば何度もやり直せるようにした点・何点取れたかによって・表示する結果の場面を変えられた点・難易度によって何点とれば、いい点数になるか別別に作れた点・難易度を作れた点。

- 改良したい点：難易度の2と3は文字大きさが変わるか、かわらないかだけなので、右から流れてきたり、一定時間たつと消えるなどもっと難易度の高いものを作りたい。

  そして、難易度1のものより簡単な、すでに出ているローマ字を打っていくものを増やして

  3つしかない難易度の種類を増やしていきたい。

制限時間を変えたいとき、ソースコードのtを変えれば変えられるが、そうでなく、設定のような

ものから変えられるようにすれば便利だと思う・キャラクターを使ったが配所の画面でしか使っていないので、タイピングしている最中に話したり、他にも活用していきたい・見た目が地味な点。

感想

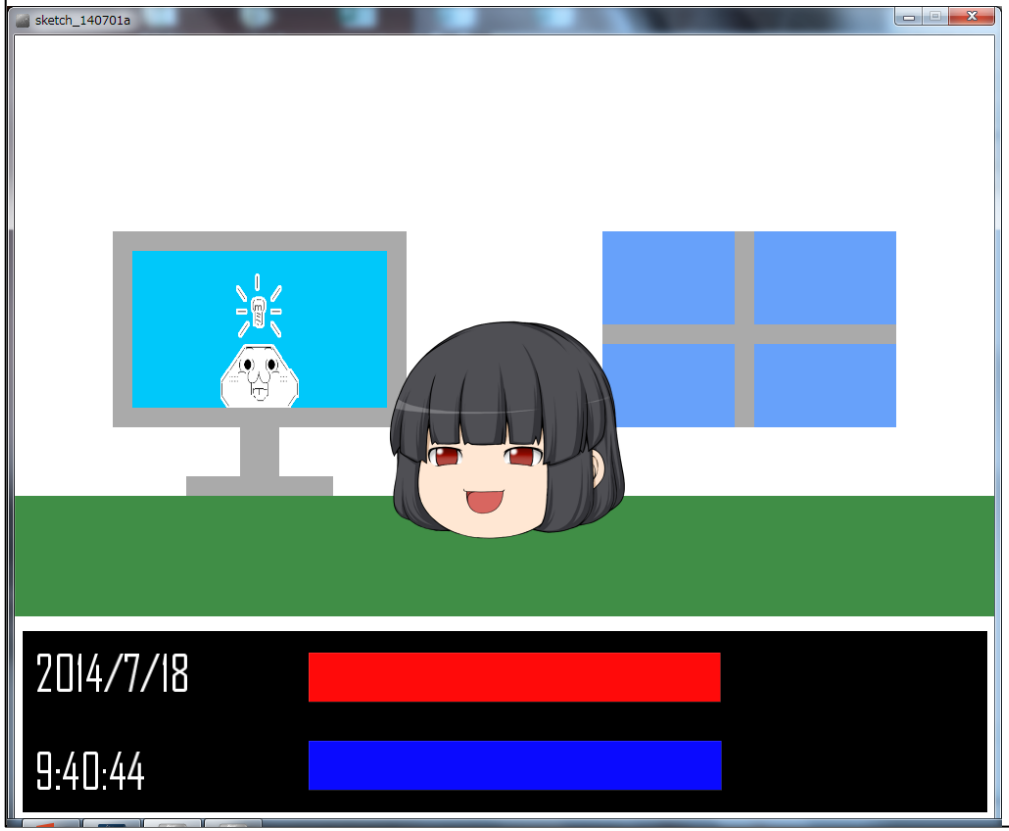
- タイピングゲームを作るのは難しかったけれどとても楽しくできた。特に制限時間をどうすればいいか初めはまったくわからなかった。
- 難易度わけをどうしたらいいか考えていたところ見つけたのがswitch～case～if~breakという関数だった。
- 心残りは、どうにかタイピングゲームになるようにするので精一杯で、外見画とても地味な物になってしまった点。
- なので、これから私のタイピング練習がてら、改良を加えていけたらいいと思う。

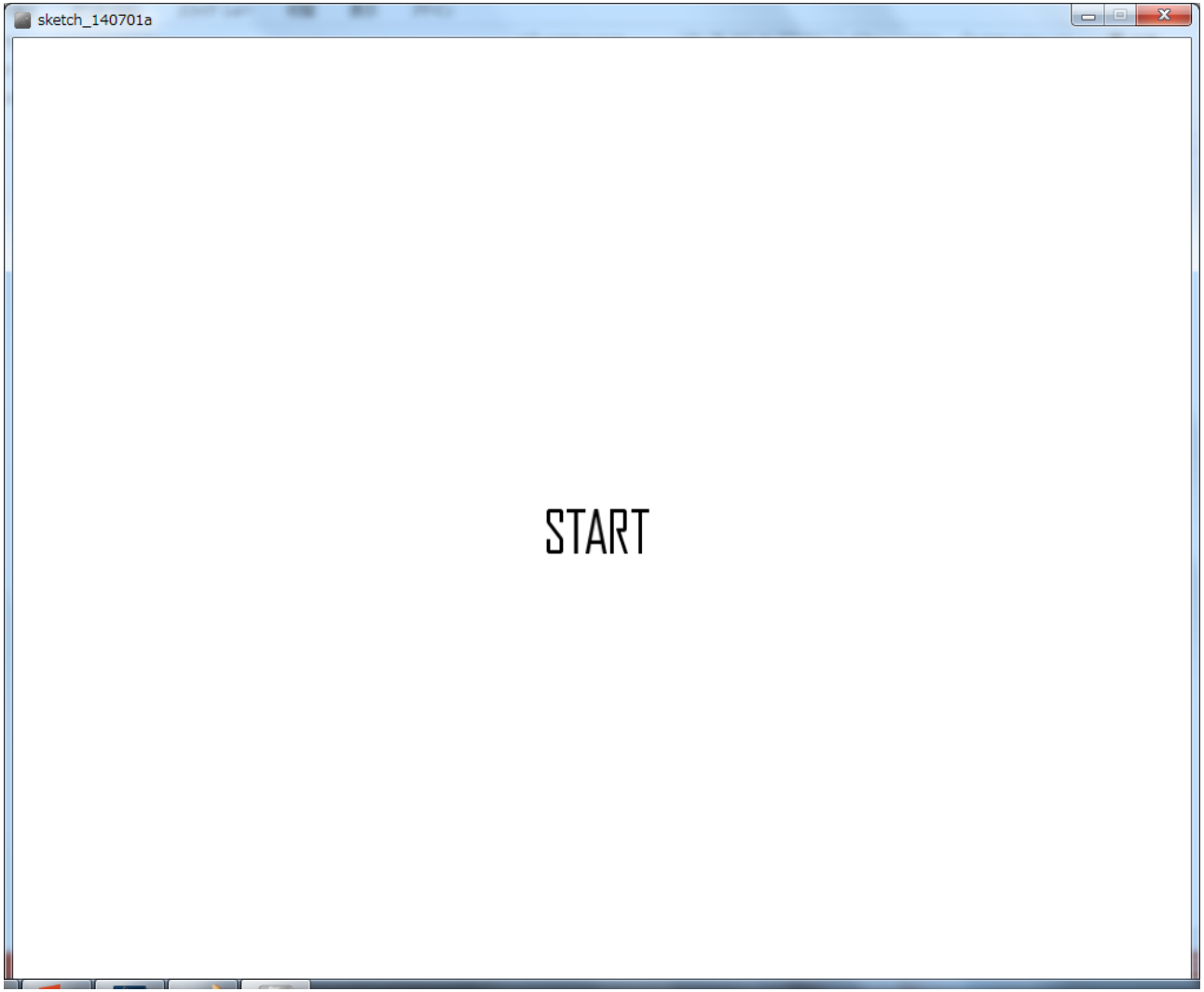
目的

- 現在の日時を確認しつつ何かを育成していくようなちょっとした暇（1～2分）をつぶしたい人向けにこのプログラムを作成しました。

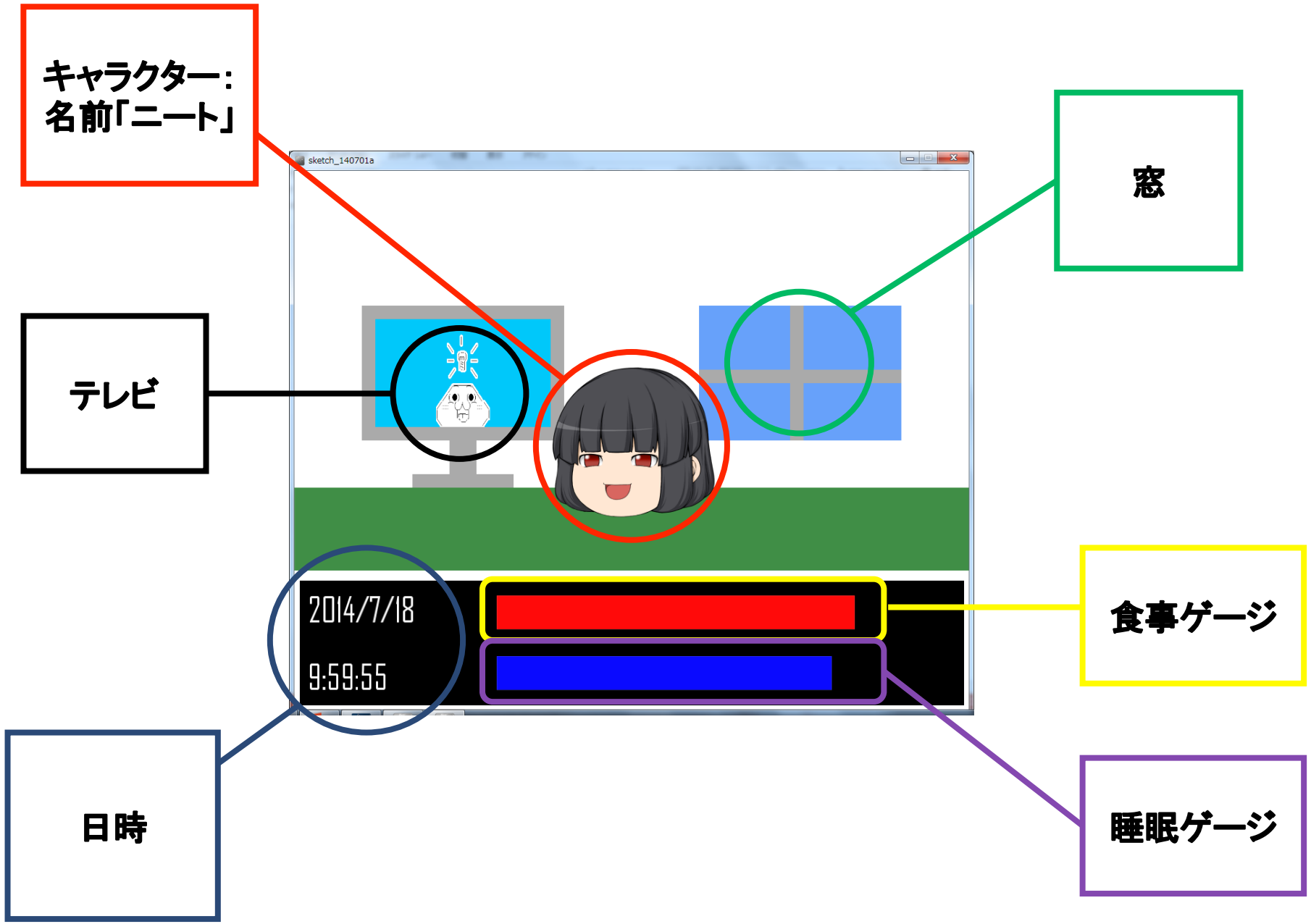
内容

たまご〇ちみたいな簡単な育成ゲームです。





最初に表示される画面は真ん中に表示されている「START」をクリックすると別の画面に切り替わります。


キャラクター：
名前「ニート」
窓
テレビ
sketch_140701a
2014/7/18
9:59:55
食事ゲージ
日時
睡眠ゲージ

「ニート」について

- 「ニート」はこのゲームの主人公です。（登場人がこいつしかいないので必然的にこいつが主人公です。）
- 「ニート」はキーボードのキーを押すことで姿を変えることができます。

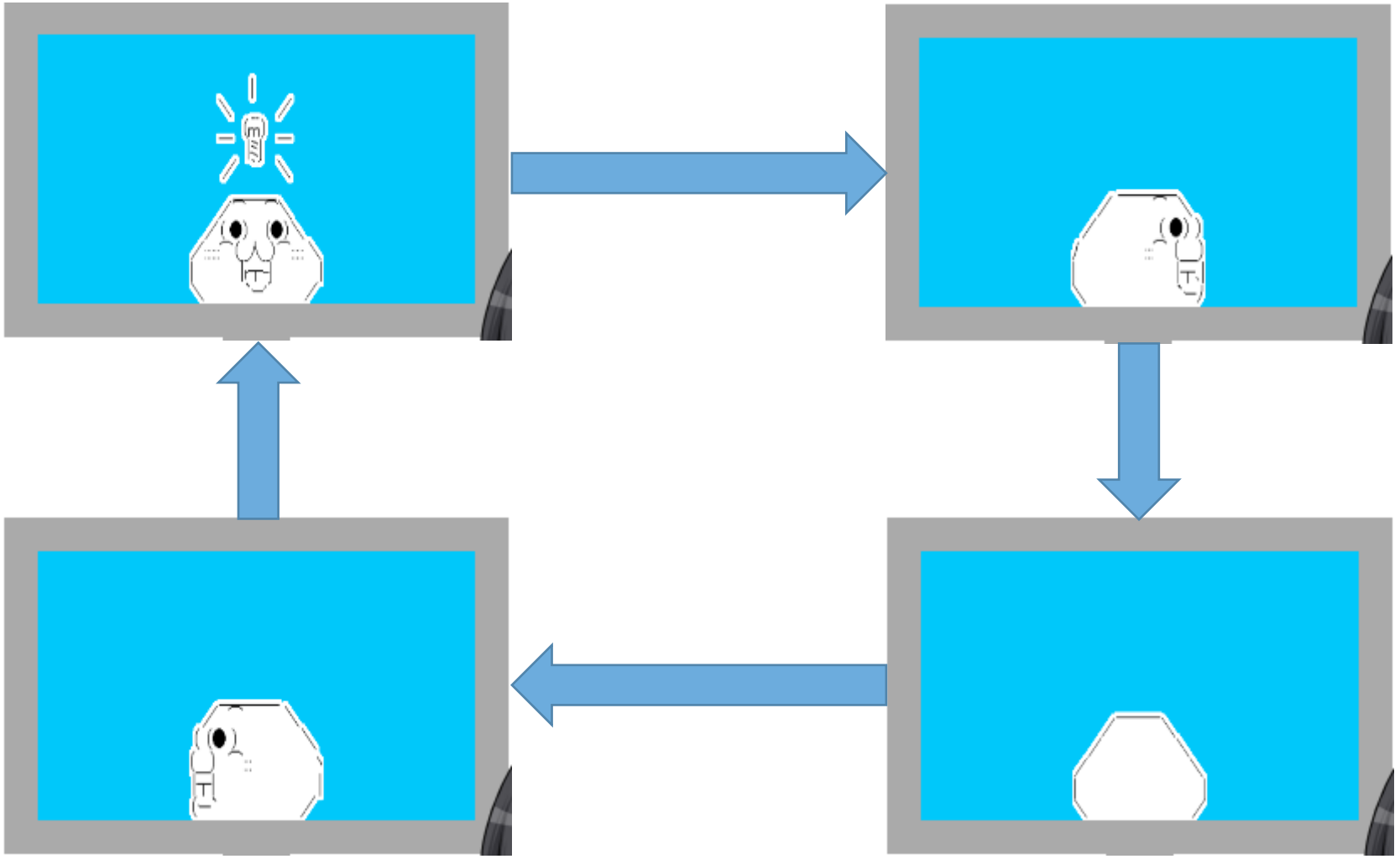
「通常ニート」
・普段の状態の「ニート」。初期状態、または「Q」キーを押すとこの状態になる。
“一応”不老不死。

「猫耳ニート」
・猫耳を付けている状態の「ニート」。「W」キーを押すとこの状態になる。
猫耳は付けているが尻尾はついていない。

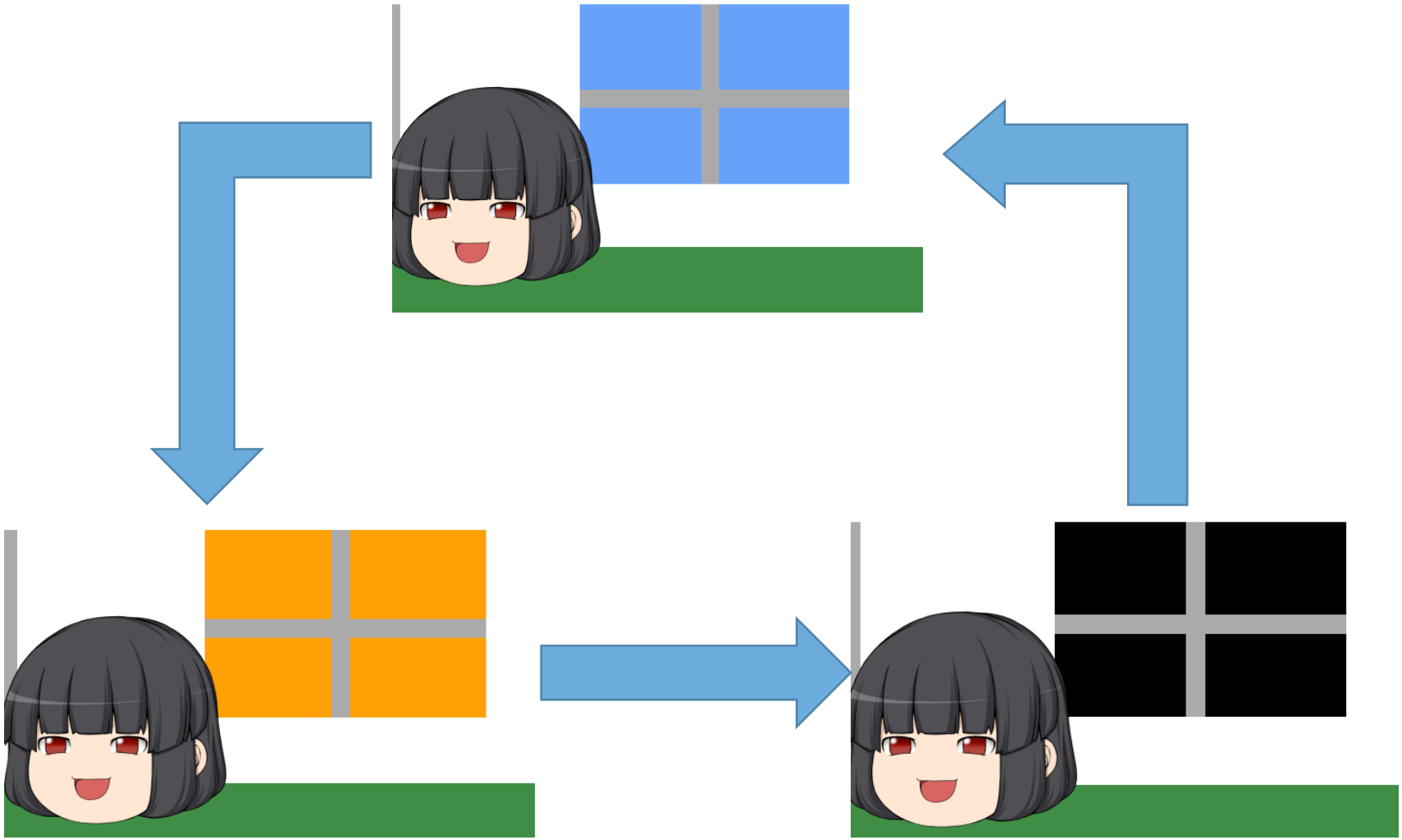
「死にかけニート」
・死にかけている状態の「ニート」。「E」キーを押すか、食事・睡眠ゲージが0になるとこの状態になる。
死にかけのくせに血色がいい。

テレビ

・テレビの画面は時間の経過によって動くぞ！



窓



食事・睡眠ゲージ

- 睡眠・食事ゲージは「ニート」の体調を整えるのに必要なゲージだぞ！
- ゲージは時間の経過とともに徐々に減っていき、ある一定の数値になると「ニート」は体調を崩してしまうぞ！
- 食事ゲージは「Z」キーを押して食事することで回復することができるぞ！
- 睡眠ゲージは「C」キーを押して睡眠することで回復することができるぞ！

食事をしている「ニート」の顔
とてもうれしそうだ

睡眠をしている「ニート」の顔
ぐっすり眠っている。

食事・睡眠ゲージが0の場合

食事・睡眠ゲージが0の場合

工夫した点

- キャラクターの表情を変化させるために変数をいろいろ使用した。
- 最初の画面をクリックしたら明度を0にすることで次の画面が写るようにしました。

自己評価

- 自分が作りたいものができたので個人的には満足している。
- プログラム内容が手探りに作成していたのでぐちゃぐちゃになってしまった。
- もう少し機能を付け加えておきたかった。

素材提供場所

- nicotalk&キャラ素材配布所で以下の方が作成した素材を使用させていただきました！！
  - きつねさん
    - かぐや
  - ヤマアラシさん
    - AAやる夫

●作品の概要

ジャンル:

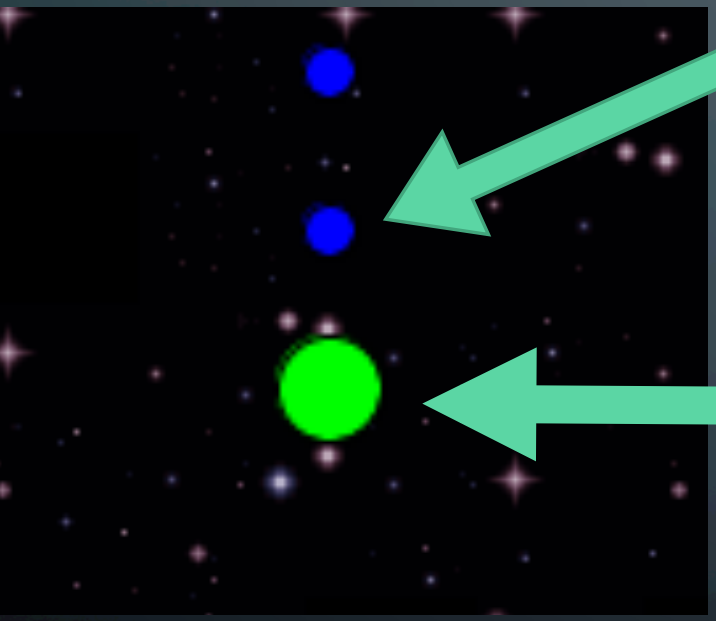
シューティングゲーム

最終目的:

最後まで生き残りハイスコアを残す

●基本操作説明

自機の発射した弾



自機

自機の移動

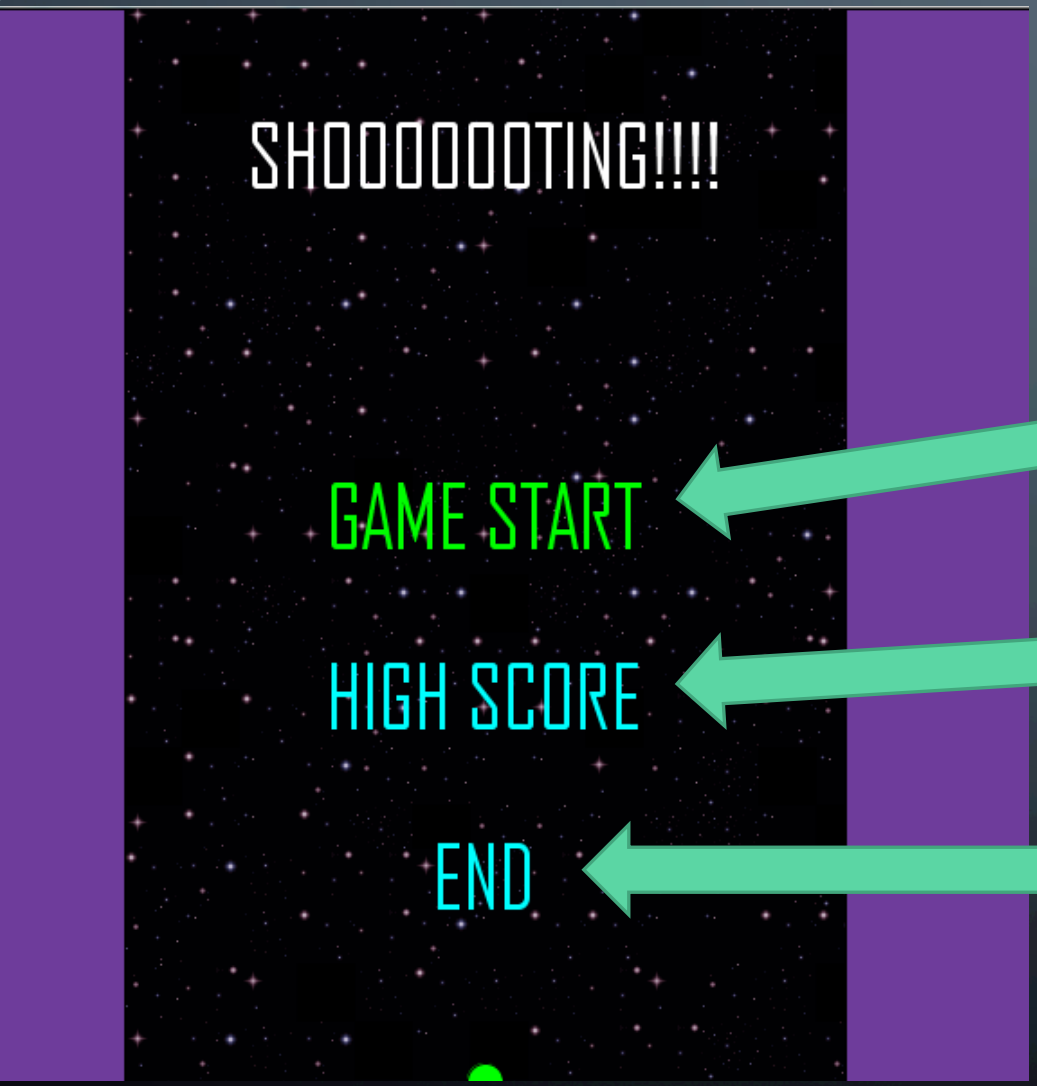
↑「P」

「L」←●→「:」

↓「;」

弾の発射:「Z」

●画面の説明（タイトル）



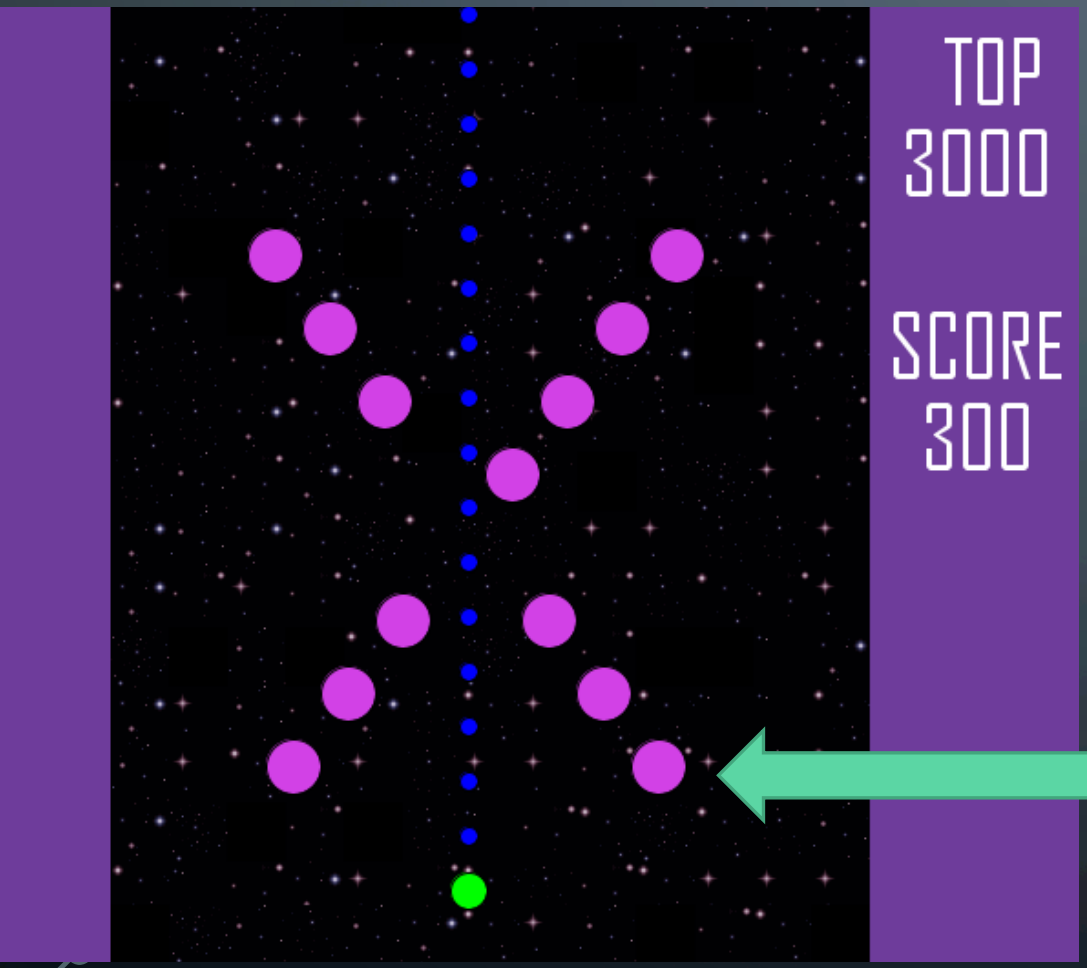
緑色になっているメニューが現在選択中
「↑」「↓」・・・メニューの選択
「Enter」・・・メニューの決定

「GAME　START」
ゲームを開始します

「HIGH　SCORE」
ハイスコアを見ます

「END」
ゲームを終了します

●画面の説明（プレイ中）

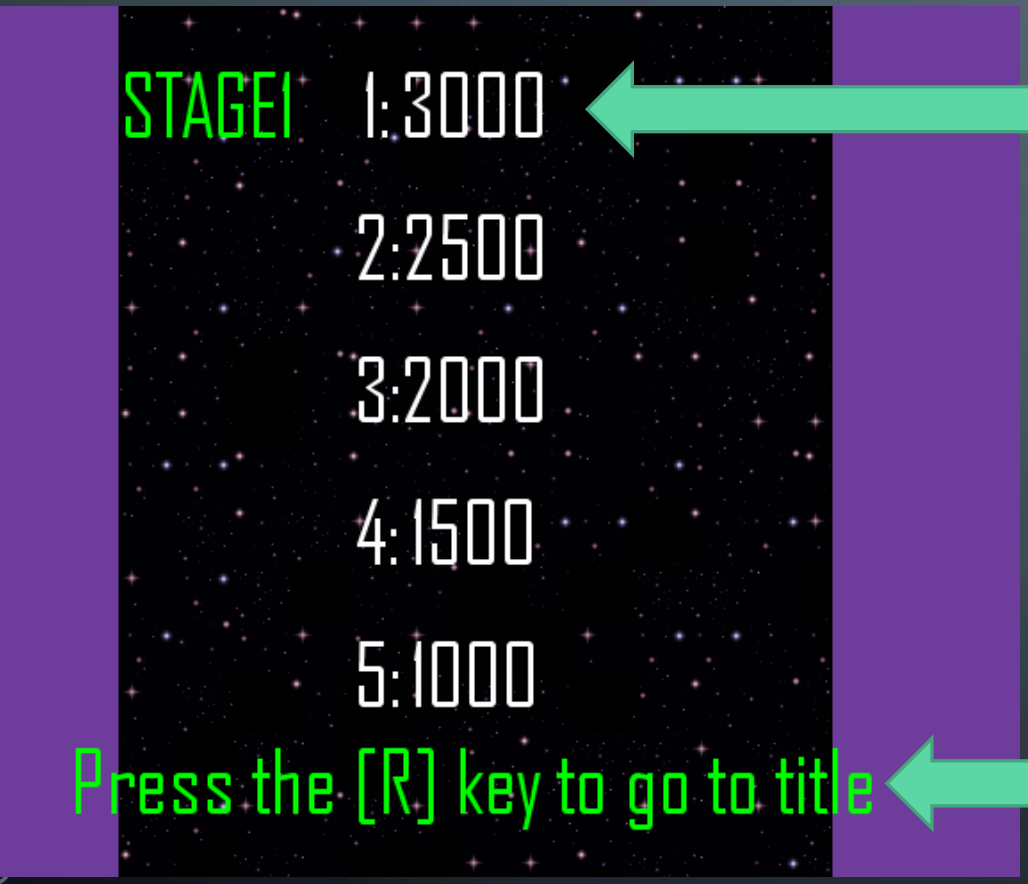


現在トップのスコア

現在の自分スコア

敵
（HPを0にすると撃破）

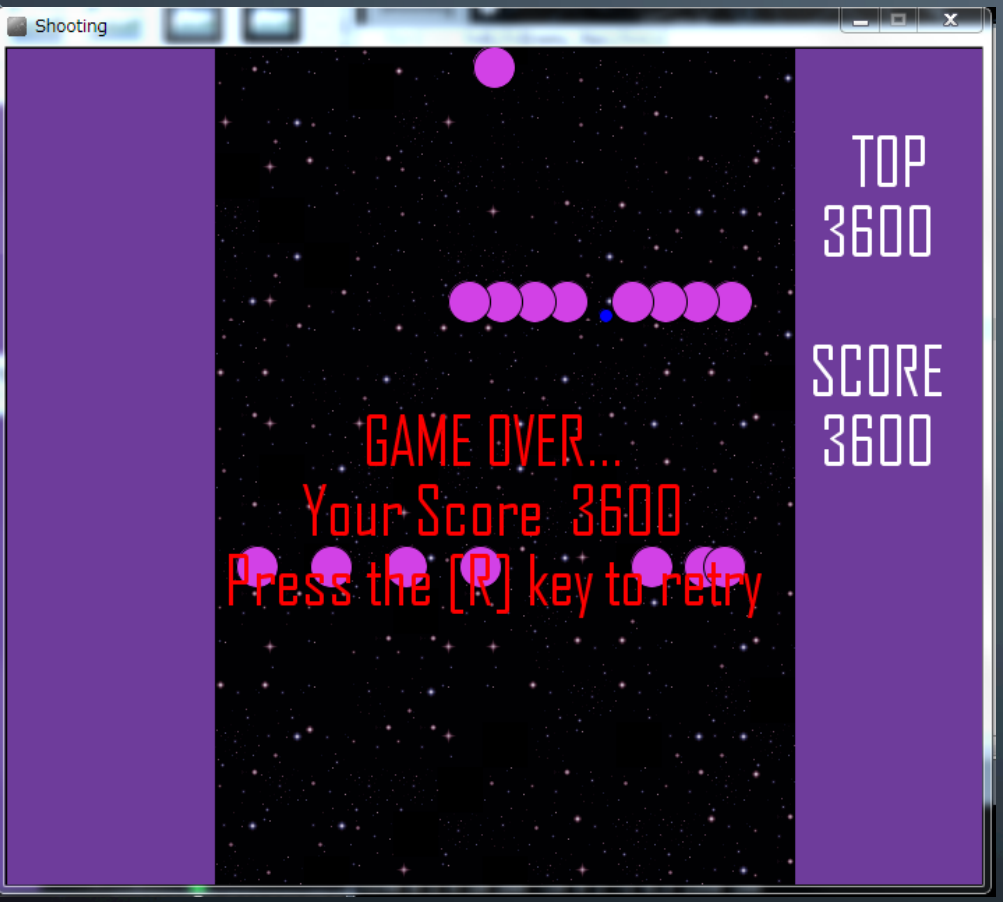
●画面の説明（ハイスコア）



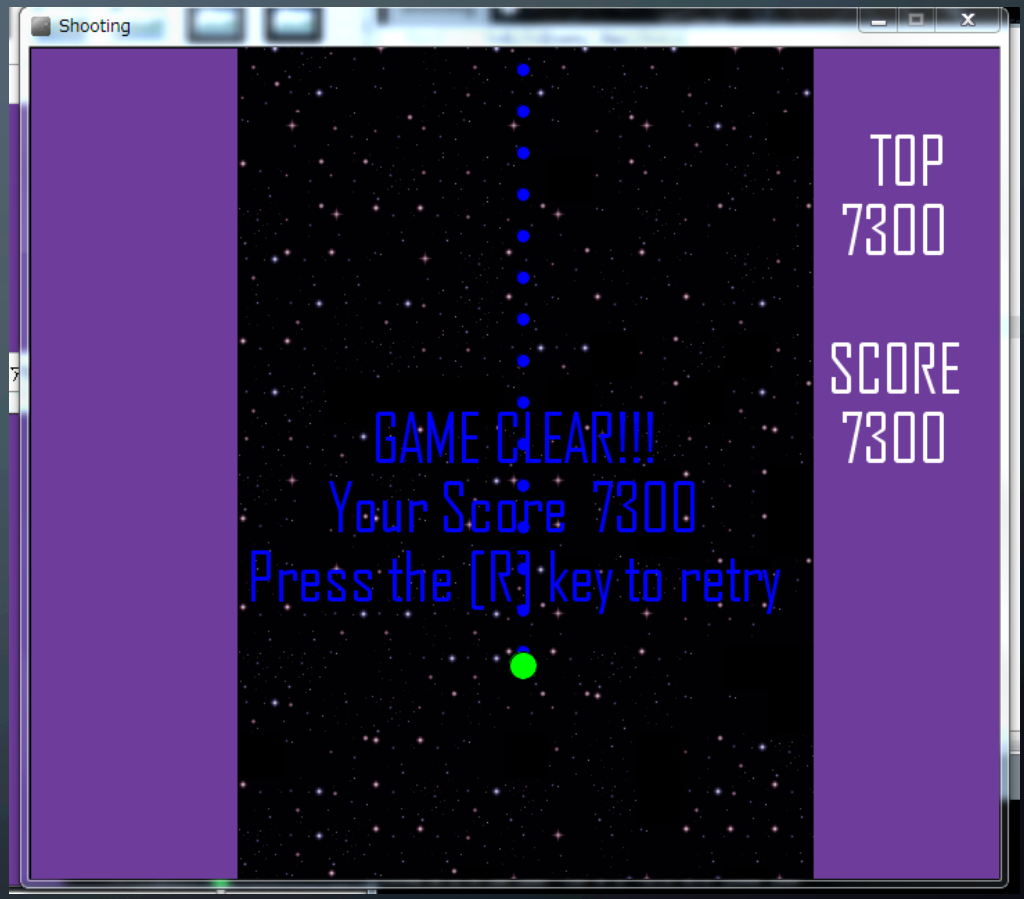
現在トップのスコア

Rキーでタイトルに戻る

●画面の説明（ゲームオーバー・クリア）



ゲームオーバー画面



ゲームクリア画面

●ステージの読み込み

時間	状態フラッ	種類	行動パター	HP	x	y	vx1	vy1	vx2	vy2	行動の変わるタイミング
0	0	1	1	300	20	-30	3	3	5	10	0
0	0	1	1	300	380	-30	3	3	5	10	0
20	0	1	1	300	40	-30	3	3	5	10	0
20	0	1	1	300	360	-30	3	3	5	10	0
40	0	1	1	300	60	-30	3	3	5	10	0
40	0	1	1	300	340	-30	3	3	5	10	0
60	0	1	1	300	80	-30	3	3	5	10	0
60	0	1	1	300	320	-30	3	3	5	10	0
80	0	1	1	300	100	-30	3	3	5	10	0
80	0	1	1	300	300	-30	3	3	5	10	0
100	0	1	1	300	120	-30	3	3	5	10	0
100	0	1	1	300	280	-30	3	3	5	10	0
120	0	1	2	300	100	-30	3	1	5	10	360
150	0	1	2	300	130	-30	3	1	5	10	360
180	0	1	2	300	160	-30	3	1	5	10	360
210	0	1	2	300	190	-30	3	1	5	10	360
240	0	1	2	300	220	-30	3	1	5	10	360
270	0	1	2	300	250	-30	3	1	5	10	360
300	0	1	2	300	280	-30	3	1	5	10	360
330	0	1	2	300	310	-30	3	1	5	10	360
120	0	1	2	300	300	-30	3	1	5	10	360
150	0	1	2	300	270	-30	3	1	5	10	360
180	0	1	2	300	240	-30	3	1	5	10	360
210	0	1	2	300	210	-30	3	1	5	10	360
240	0	1	2	300	180	-30	3	1	5	10	360
270	0	1	2	300	150	-30	3	1	5	10	360

csvを変更すれば自分でステージを作れる

読み込みはcsvファイルから行
設定されているのは
・「出現する時間」
・「敵の状態のフラグ」
・「敵の種類」
・「行動パターン」
・「HP」
・「x」・・・x座標
・「y」・・・y座標
・「vx1」・・・状態1の時のx速度
・「vy1」・・・状態1の時のy速度
・「vx2」・・・状態2の時のx速度
・「vy2」・・・状態2の時のy速度
・「行動の変わるタイミング」
・・・状態1から2に移行する

●自分でステージを作る場合（設定方法）

- 出現時間・・・60で1秒換算

  例：3秒後に出現させたい場合180に設定

- 状態フラグ・・・基本は0＝（存在しない）に設定
- 行動パターン

  「1」・・・vx1,vy1の速度で移動

  「2」・・・vx1,vy1の速度で移動（状態1）vx2vy2の速度で移動（状態2）

  「3」・・・移動しない（状態1）vy2の速度で移動（状態2）

  「4」・・・vx1,vy1の速度で移動（状態1）vx2vy2の速度で移動（状態2）

  状態2の時壁に当たると反射する

  「5」・・・基本行動は4と同じ、状態2の時加速する

注意点

・このキャラクターはへびまるといいます。→ 

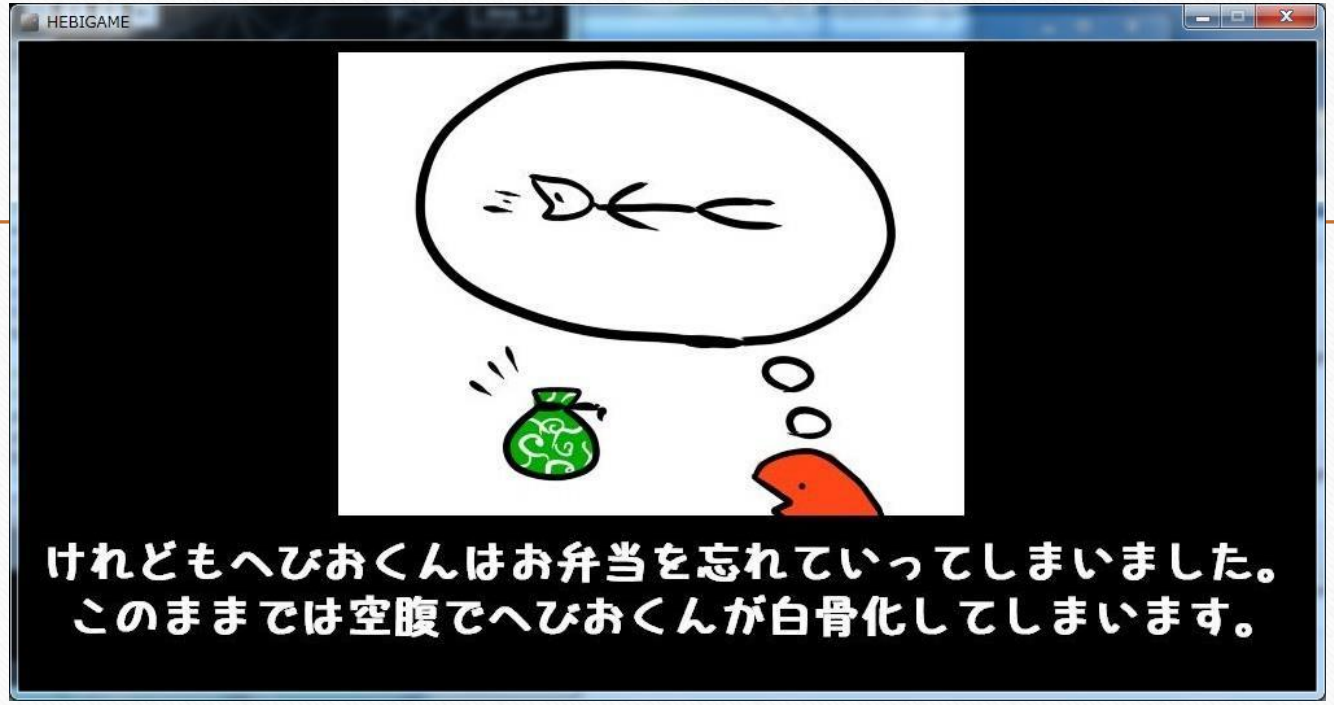
・マップ作成モードとステータス確認画面は右クリックでメニューに戻れます。

・メニューで終了した場合のみステータスや作ったマップがセーブされます。

「へびゲーム」の工夫した点

- マウスだけを使って遊べるようにしました。
- ゲーム本編だけではつまらないのでいろんなモードを入れました。
- セーブとロードをできるようにしました。
- 容量をできるだけ軽くしました。

ストーリー



飼い主がお弁当を持たずに学校へ行ってしまったので、届けに行く物語です。

メニュー画面

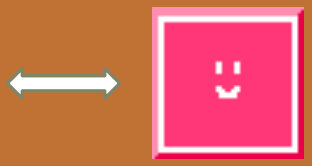
ここからやりたいモードを選択して遊びます。
まずはゲームスタートから説明していきます。

ゲームスタート

全11ステージのゲーム本編です。最初のステージに操作方法が書いてあります。
全てクリアすればエンディングがあります。
つぎに、このゲームに出てくるギミックを一部紹介します。

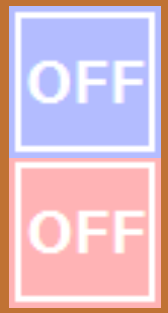
ゲームに出てくる基本的なギミック1

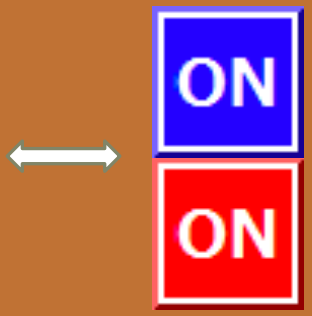
ほとんどはクリックすると変化します
基本的に、色が薄いブロックは通れて、色が濃いブロックは通れません



←クリックすると
出たり消えたりします。

←一定時間ごとに
出たり消えたりします。
クリックしても意味がないです。





←クリックすると
出たり消えたりします。
青がONの時は赤がOFF
赤がONの時は赤がOFF
になります。

←クリックすると一定時間だけ出現します。

ゲームに出てくる基本的なギミック2

少し特殊なブロックたちです。

←？をとると右に迷路が出現します。マウスが迷路内のカギに触れている場合のみがギアナのブロックが消えます。

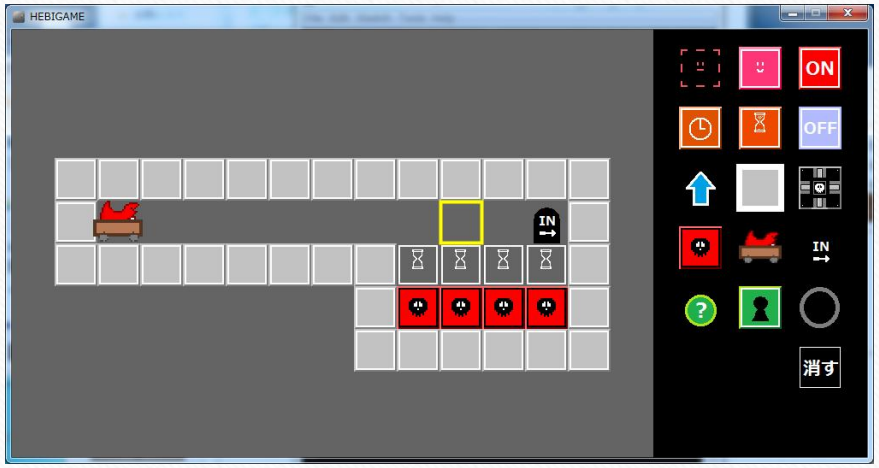
←一定時間ごとに鉄球を落とします。鉄球はへびまると同じように反射したり落ちたりします。

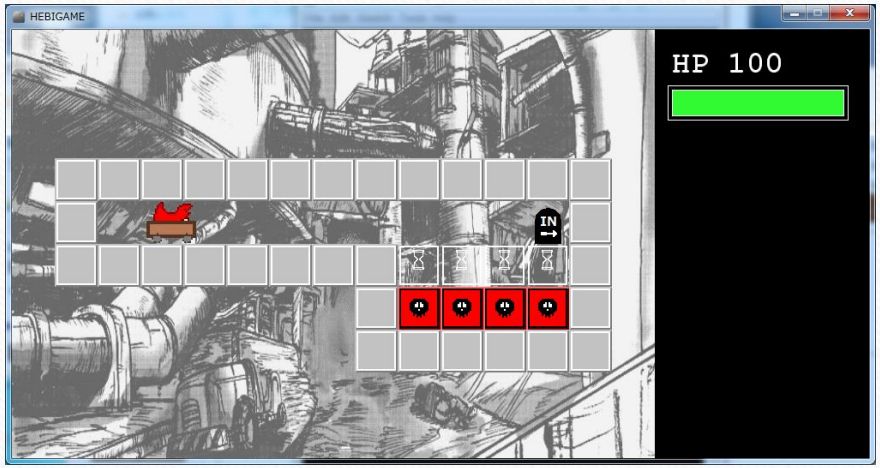
←一定時間ごとに4方へ弾を撃ちます。触れるとダメージを受けます。

←このマークに触れている場合、へびまるが上昇していきます。

マップ作成モード/作ったマップで遊ぶ

マップ作成モードで右クリックをするとセーブしてメニューへ戻ります。



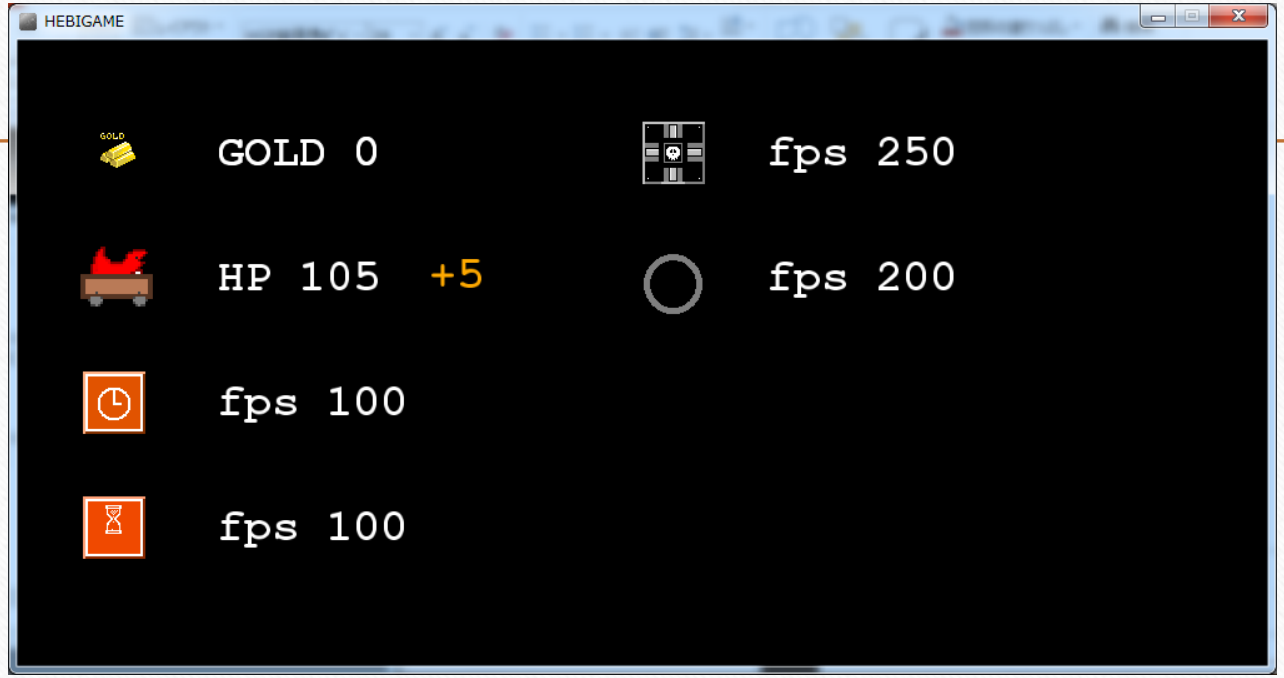


さきほど紹介したものブロックをつかって
自由にマップを作って遊べます。常識の範囲内でブロックをおいてください

ガチャ　一回300GOLD

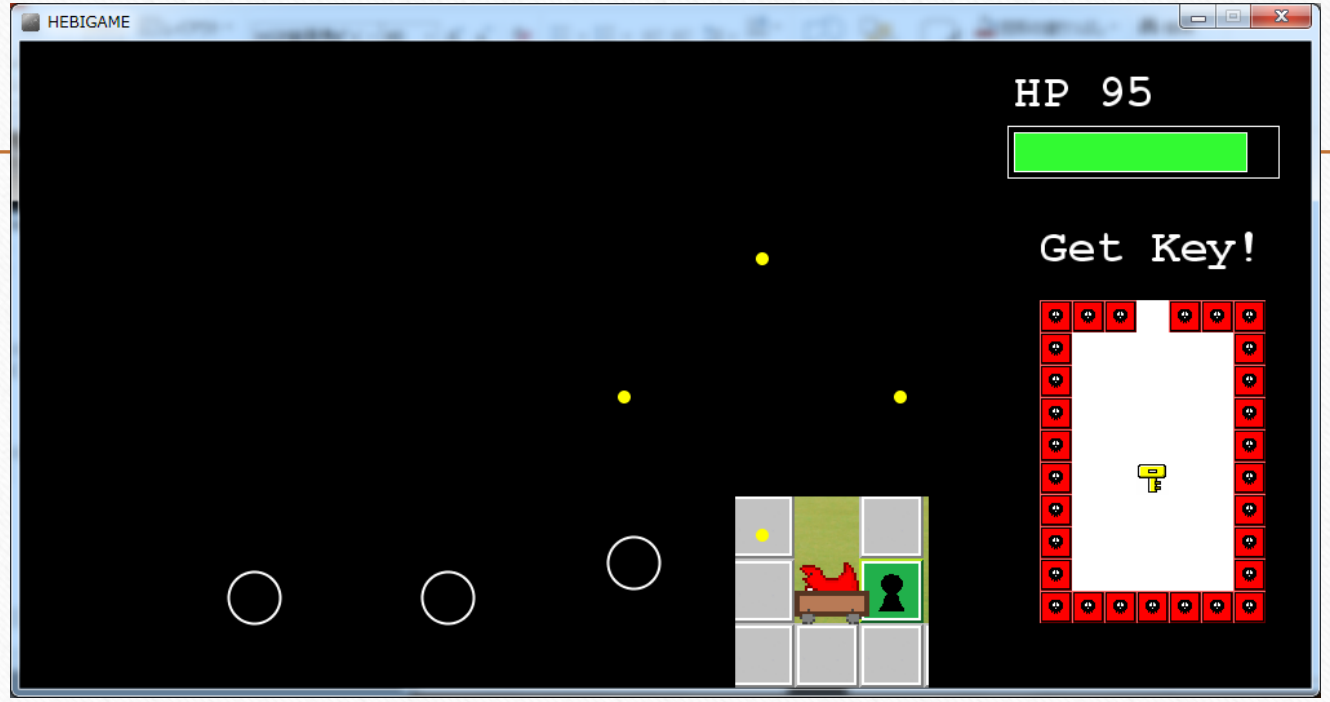
ゲーム本編で金塊をとるとガチャを引けます。
引いた景品によってゲームが有利になります。課金はできません。

ステータス確認



ステータスを確認できます。
ガチャで強くなったステータスの分は虹色で表示しています。

初心者お断り　一寸先は闇モード



ステージはゲーム本編と同じですが、
視界がとても悪くなります。

感想＆反省点

・この程度のゲームでも作っているときに何度もあきてやめてしまったり
してしまったので、根性がたりてないとおもいました。

・不必要な変数を使ってしまっているので
今後はなるべく変数を節約していきたいです。

・全く計画を立てないでゲームを作り始めたので
新しいギミックを追加するたびに過去のプログラムを編集してくたびれました。
次回からはちゃんと計画を立ててからプログラムを作りたいです。

使った音楽素材

- ［ファミコンBGM集］熱血高校ドッジボール部
- http://www.nicovideo.jp/watch/sm1499051
- **SFCダウンタウン熱血べーすぼーる物語 BGM03ミーティング**
- http://www.nicovideo.jp/watch/sm20514654
- **パズドラ BGM 1(ロビー)**
- https://www.youtube.com/watch?v=B2298gEmlgc
- **風来のシレン3　BGM 「特殊モンスターハウス」sm3680869**
- https://www.youtube.com/watch?v=kYHuafAXigc

ゲーム内容

- 上部にあるブロックに球体をぶつけて消滅させていくゲーム。
- ブロックをすべて消すことでブロックの後ろの隠れる背景をすべて見ることができる。

仕　様

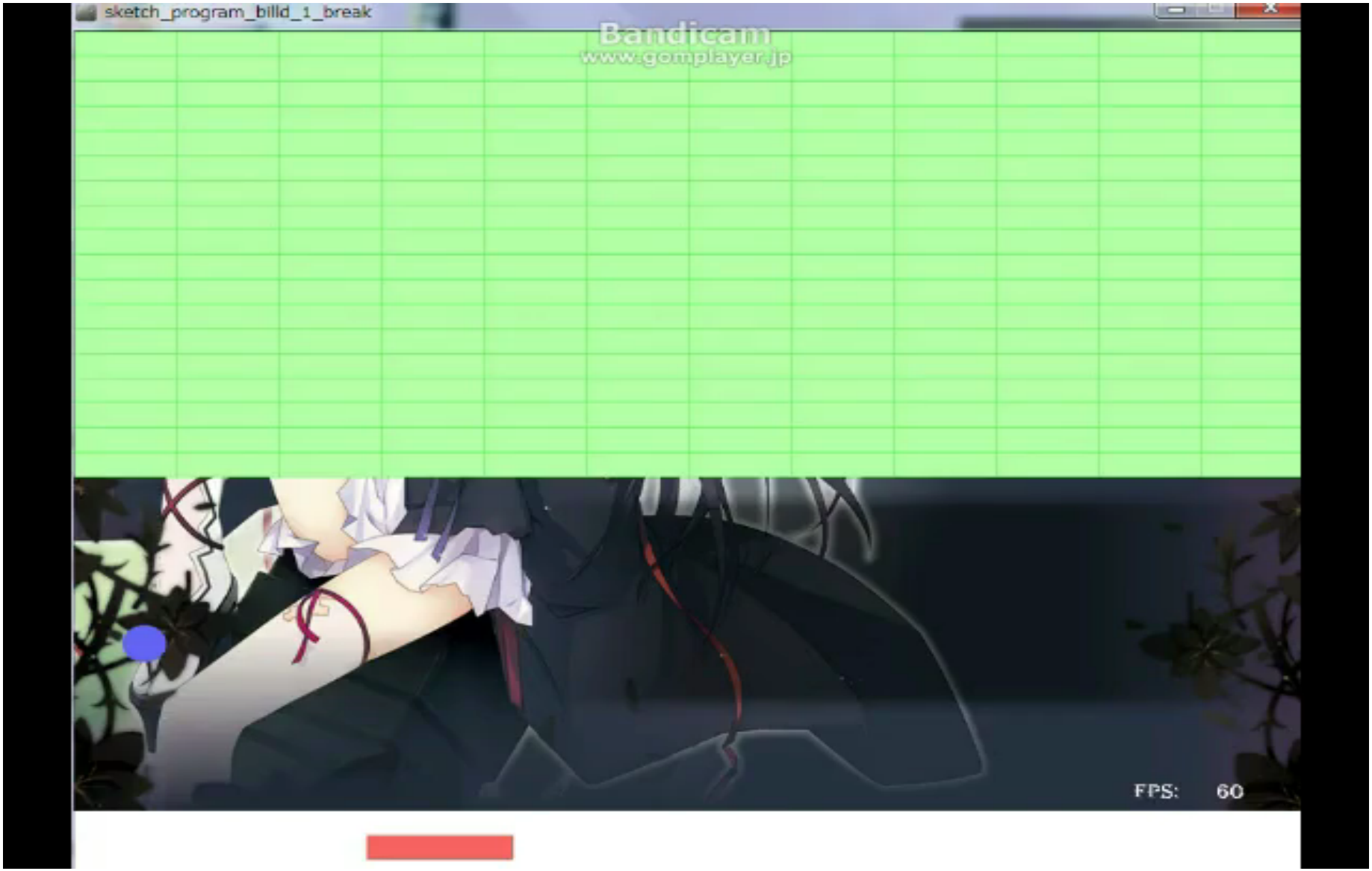
- 画面の左端に球体が当たるとはね返る。
- 画面の上部に球体が当たるとはね返る。
- 画面の右端に球体が当たるとはね返る。
- 画面の下部に球体が当たるとはね返らず、クリア不可となる。
- プレイヤーが操作するバーに球体が当たるとはね返る。
- ブロックに球体が当たるとはね返る。
  ー　ブロックは消滅する。
- ブロックをすべて消すと、「COMPLETE!!」という文字を表示する。

操 作 方 法

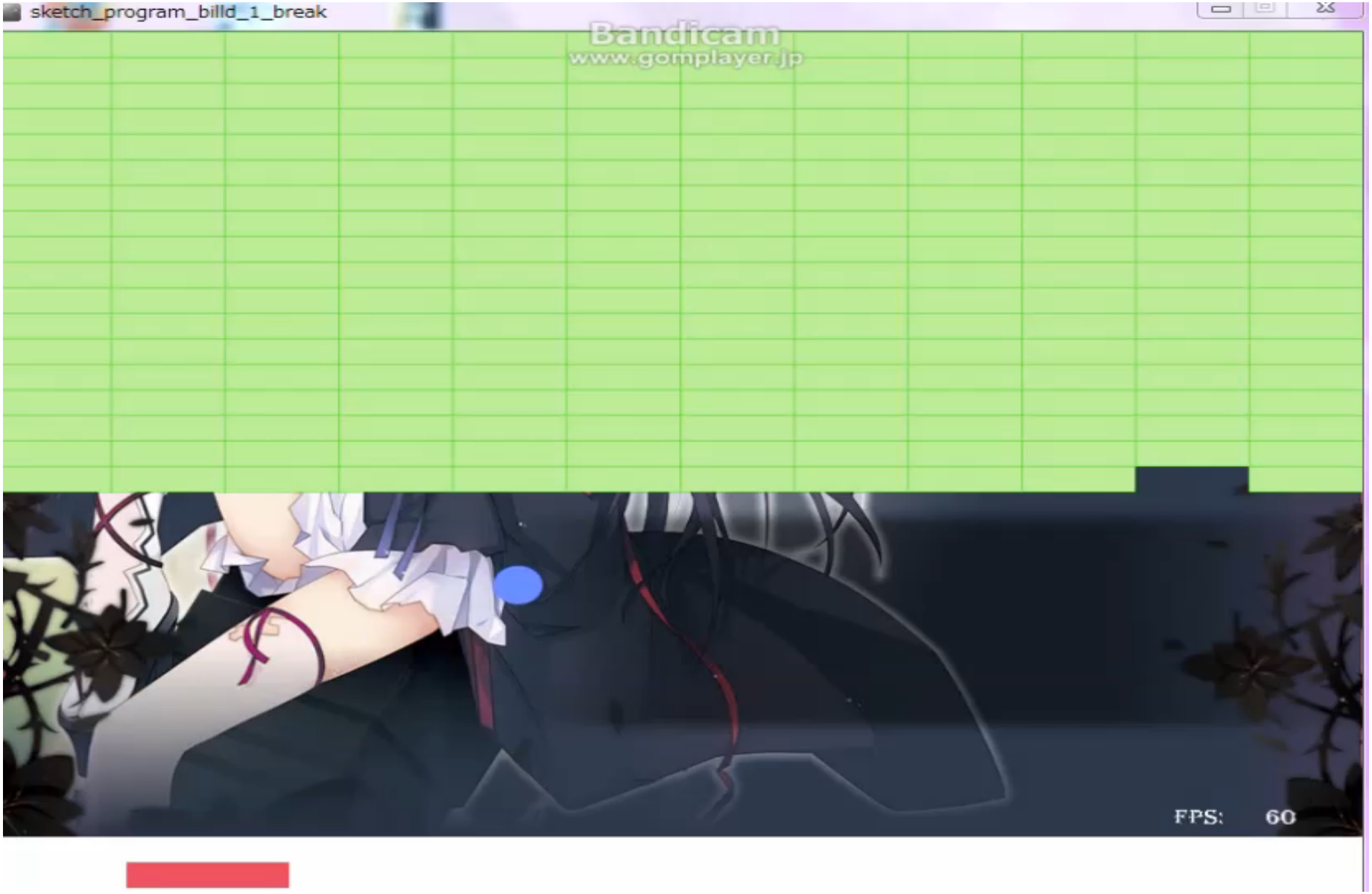
- 「←」 バーを左に移動。
- 「→」 バーを右に移動。
- 「↑」 フレームレートを5上げる。
- 「↓」 フレームレートを5下げる。
- 「Shift」 最初からやりなおす。

※フレームレートは60から始まり、上限320、下限60。
　画面にはFPSとして現在のフレームレートを表示します。

プレイ動画


sketch_program_billd_1_break
Bandicam
www.gomplayer.jp
FPS: 60

おまけ(Shift restart)



使用した資料・ソフト

- 背景画像：機巧少女は傷つかないキービジュアル
- BGM：機巧少女は傷つかない 「To Victory」

- コンパイラソフト：processing 2.1.1
- 動画撮影ソフト ：Bandicam
- 動画編集ソフト： VideoPad

実際の戦闘機（FA-18F）のデータを再現

- 最大速度 M1.6
- 大きさ　13.62m, 18.38m
- 固定武装　M61A1 20mmバルカン砲　装弾数：400発　速度：6,000発/分
- ミサイル　AIM-9 サイドワインダー
- これらを画面比率を考慮し、再現。

素材も実際のデータを再現

エンジン始動音

エンジン音

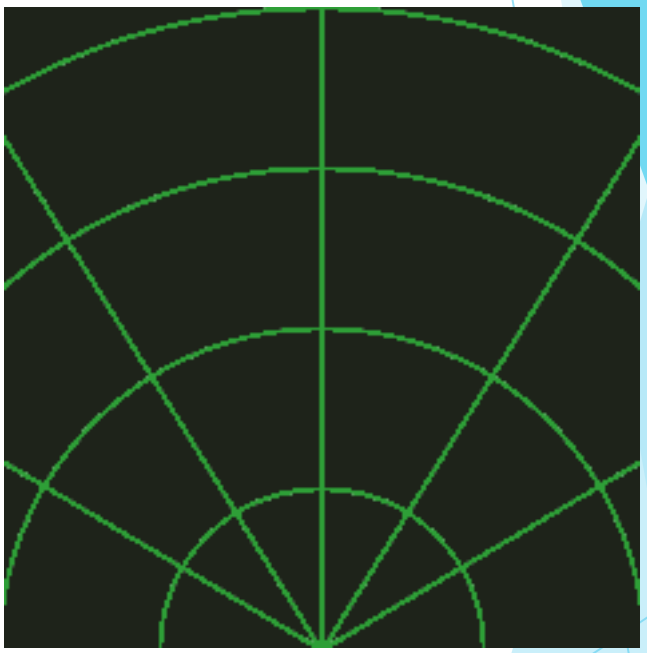
機銃発砲音

遠くの機銃発砲音

機銃着弾音

素材も実際のデータを再現

FA-18F スーパーホーネット



実際に使用されるレーダーの形状

素材も実際のデータを再現

AIM-120 AMRAAM

AIM-9X サイドワインダー

操作方法

- ↑ ボタンで自機の加速
- ↓ ボタンで自機の減速
- → ボタンで、自機が右に旋回
- ← ボタンで、自機が左に旋回

操作方法

- TAB ボタンでレーダーの大きさを変更
- Spaceボタンで機銃を発砲
- C ボタンで右のミサイルを発射
- X ボタンで左のミサイルを発射

状態説明

- 機銃、ミサイル、他の戦闘機と接触した場合、戦闘機が破損し、最悪の場合墜落する。
- 他の戦闘機との接触時は、相対速度が遅い場合、破損しない。
- 遠くの敵をレーダーを見ながら追尾し、ミサイル、機銃などで撃ち落とすことが可能

プレイ実演

概要…どんなゲームか

「ベリーのにわ」

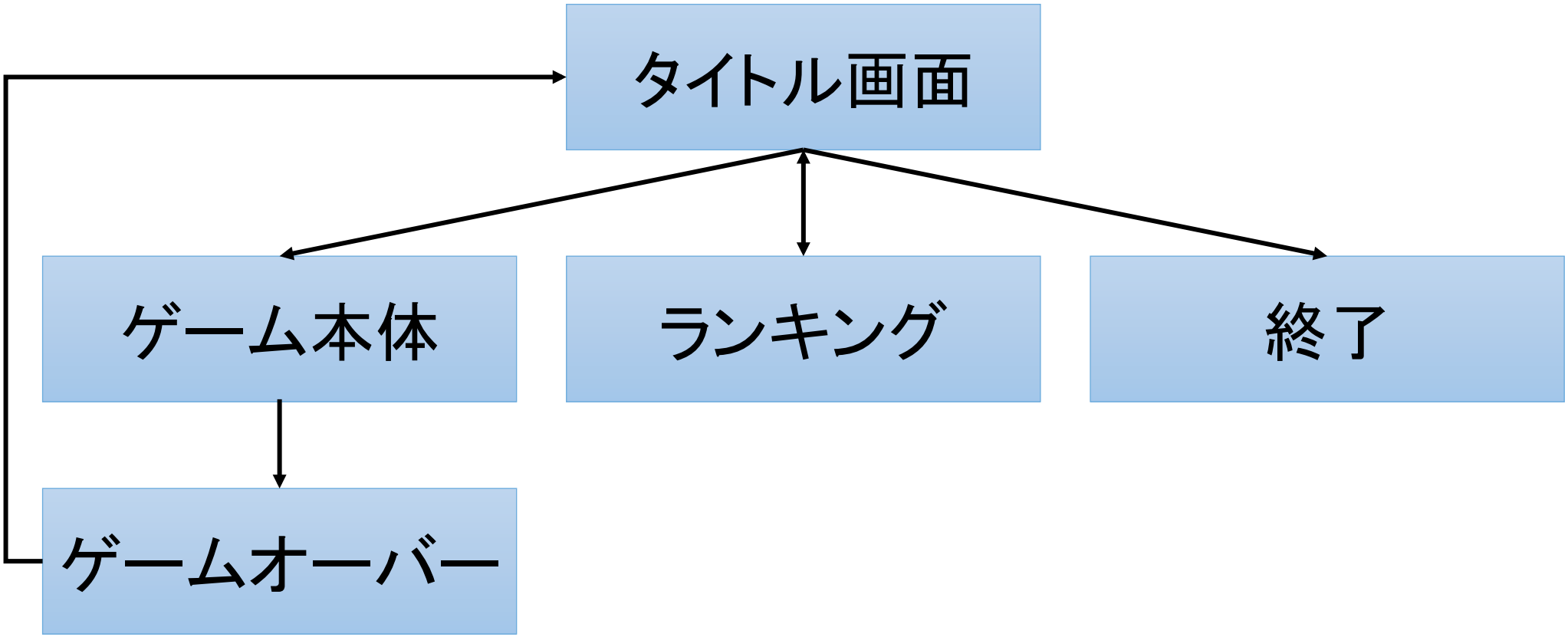
好感度システム付き、問題カスタム可能な
四択式クイズゲーム

ストーリー

薄暗い森の奥で、貴方は謎の少女「ベリー」と出会います。
博学なベリーは“貴方へ向けた、貴方のための”
様々な問題を出してくれますので、
一緒に遊んであげましょう。

「ベリーのことがしりたいの？」
「それなら、ベリーとたくさん遊んでね！」

構造



詳細システム

- 一問出題ごとにゲージが最初に一度のみ回復する。
  - 正解すれば残り時間分がスコアに移され次の問題へ

  - 間違えた場合やゲージが0になった場合
    →ペナルティが付き次の問題へ進む。
- ペナルティが3たまった時点でゲームオーバー。

工夫点

- 好感度システムを導入し、プレイヤーがゲームを進めるごとにキャラクターが変わっていく
- CSVから自分で問題を追加することができる。

…昔匿名でほしいゲームについてのアンケートを取った際に
「クイズ形式の、勉強に使えて余り勉強しているような外見ではないゲームがほしい」
「自分でゲームを追加ができるクイズゲームがほしい」
という意見があり、それを反映させたものになる。

工夫点

- 重複なし四択
- 簡易的ばらつき出題システム
  …問題数全体の前半後半から交互に出題することで、
  なるべく重複を避ける。

好感度システム

- 最大値スコアを元に、一定以上のラインを超える
  →タイトルの少女のグラフィックが変わっていく。
- 内部の値は（ファイルを参照しない限りは）見えない
  →プレイヤーの想像を掻き立てる。

問題追加システム

- 自分のほしい問題を追加できるシステム。
- ファイル内のquiz.csvに問題と回答を追加して記述すれば
  新しい問題も出題する。

１，　はじめに

私の最終課題のプログラムはいわゆる弾幕 STG というジャンルに分類されるゲームです。
素材画像を作る時間がなかったため、一部素材はオンライン上から拝借しました。
素材サイト一覧は data フォルダの中の memo に入っています。

２，　簡単な動作説明

起動後タイトルが表示されます。Z キーを押すことでゲームを開始することが出来ます。
敵を倒しきるとクリア、３回ミスするとゲームオーバーです。
クリア後には残機および残ボムでスコアにボーナスが入ります。
クリア後、およびゲームオーバー後に Z キー入力でタイトルに戻ることが出来ます。
この時更新されているハイスコアはプログラムを閉じるまで継続されます。
操作方法は以下の通りです

■操作方法■

Z キー・・・レーザーショット
　　敵を貫通するレーザーを打ちます。
　　自機はその時、通常速度の半分以下の速度で移動します。
X キー・・・ボム
　　画像上の敵弾を削除するだけであり、敵にダメージが入ることはありません。
　　ボム発射直後一定時間無敵になります
C キー・・・ショット
　　通常の連続ショットです。
　　レーザーより多少火力が高めになっています。
十字キー・・・移動

敵は一定のパターンに基づいた攻撃を仕掛けてきます。
ある程度の弾幕 STG 経験者であればクリアできる程度の難易度です。
ミスをしてもある程度の長さの無敵時間を用意しています。
自機の周りに灰色の円状のバーがあるうちは無敵です。
また、コンティニューは実装しておりません。
S キーでスクリーンショットが取れます。

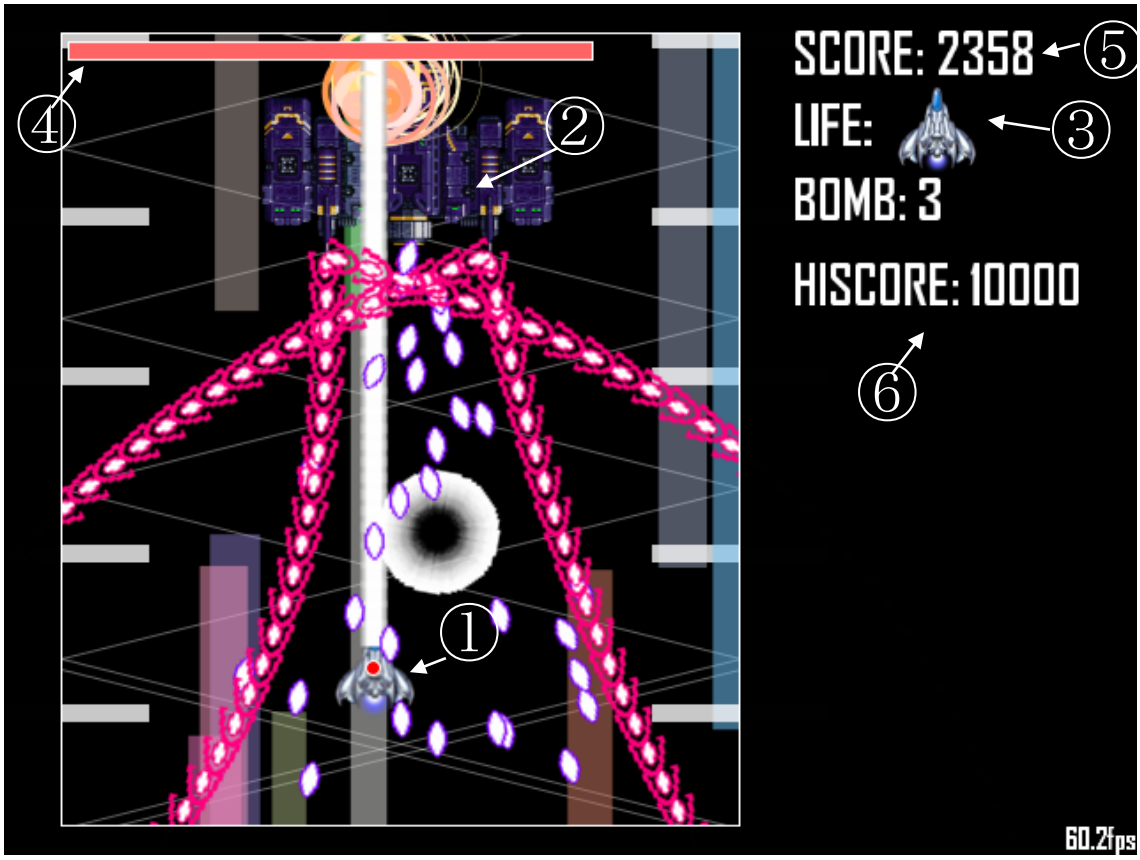


図 1 ゲーム画面

■画面内解説■

①・・・自機です。中心の赤丸が弾の当たり判定に入らなければミスになりません。

②・・・敵です。このプログラムではこの敵と戦うことになります。

③・・・残機です。何もない状態でミスをするとゲームオーバーです。

④・・・敵の体力です。赤のバーがなくなれば撃破です。

⑤・・・スコアです。敵の体力を削ればスコアが入ります。

⑥・・・ハイスコアです。アプリケーション起動中に出た最高得点を記録します

３，　プログラム解説

このプログラムにおいて私が工夫した点は次のとおりです。

1. 画像処理
2. 無駄な処理の回避（メモリ確保）
3. 敵の行動の大まかなブロック分け

これらの３つの項目について順に説明します。

またこのプログラムの詳細は後述されているのでそちらを参照してください。

1. 画像の処理

弾幕ゲーム限らず、画像データは複数のデータを用意するより一つのデータを利用したほうが必要な画像数やアクセスする回数が少なく住みます。

なので、このゲームで利用している弾画像はもともと一枚の画像データとして配布されているものを用いています。

アルファ値が当てられていなかったため、別のプログラムでアルファの値を加えた画像に改変しました。

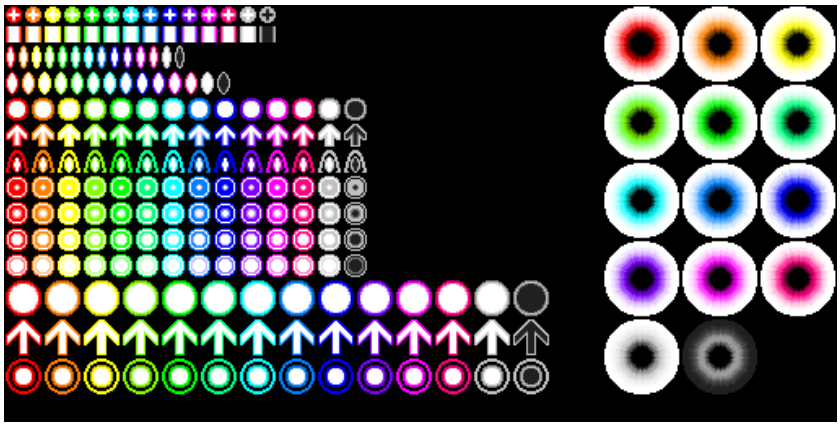


図 2　元弾画像

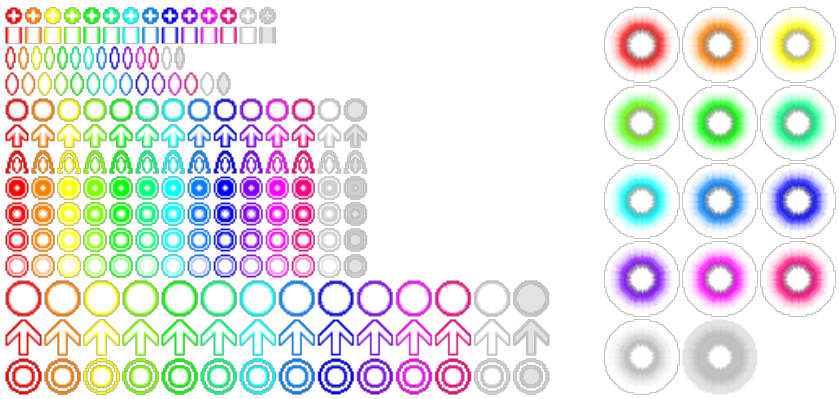


図 3　アルファ値を与えた弾画像

また、このままではゲームで扱えないので CVS ファイルに弾画像の分割位置とサイズのデータを入力し、それを元に画像を別の配列変数に分割保存するようになっています。

```
/*class Image_c*/
/*弾画像と自機画像、及びボス画像取得のアルゴリズム*/
  void Setup(){
    PImage []bulletImageType;
    bulletImageType=new PImage[15];
    bulletImage=new PImage[15][14];//[type][color];
    String GetLine="setup";//仮に適当な値を入れて for に入らない危険を回避
    loadimg=loadImage("data/alpha_shot.png");
    loadImageData=createReader("data/alpha_shot_pixellist.csv");
    for(int Z=0;GetLine!=null;Z++){//弾画像取得
      try{//弾データ取得
        GetLine=loadImageData.readLine();
      }catch(IOException e){
        e.printStackTrace();
        GetLine=null;
      }//弾データ取得ここまで
      if(Z==0||GetLine==null)continue;//読むこむデータが無くなったら以下の処理をすべてスキップする

      String[] GetData=split(GetLine,",");

      int []INI=new int[6];
      for(int i=0;i<GetData.length-1;i++){
        INI[i]=int(GetData[i+1]);
      }

      bulletImageType[Z-1]=createImage(INI[2]-INI[0],INI[3]-INI[1],ARGB);
      for(int y=INI[1];y<INI[3];y++){
        for(int x=INI[0];x<INI[2];x++){
          bulletImageType[Z-1].pixels[(y-INI[1])*(INI[2]-INI[0])+(x-INI[0])]
                                    =loadimg.pixels[y*loadimg.width+x];
        }
      }
      for(int i=0;i<bulletImage[Z-1].length;i++){
        bulletImage[Z-1][i]=createImage(INI[4],INI[5],ARGB);
```

```
        if(Z-1!=14){
          for(int y=0;y<INI[5];y++){
            for(int x=0;x<INI[4];x++){
              bulletImage[Z-1][i].pixels[y*INI[4]+x]=bulletImageType[Z-1].pixels[y*INI[2]+x+i*INI[4]];
            }
          }
        }else{
          for(int y=0;y<INI[5];y++){
            for(int x=0;x<INI[4];x++){
              INI[2]=bulletImageType[Z-1].width;
              bulletImage[Z-1][i].pixels[y*INI[4]+x]=bulletImageType[Z-1].
                                                  pixels[y*INI[2]+x+i%3*INI[4]+(INI[4]*INI[5]*3)*int(i/3)];
            }
          }
        }
      }
    }//弾画像取得ここまで
    loadimg=loadImage("data/cannon-a.png");//ボス画像取得
    BossImg=loadimg; //ボス画像取得ここまで
    PlayerImage=new PImage[2];//プレイヤー画像取得
    for(int i=0;i<PlayerImage.length;i++){
      String[] data={"data/k6_p1a.png","data/k6_p1b.png"};
      PlayerImage[i]=loadImage(data[i]);
    }//プレイヤー画像取得ここまで
    loadimg=null; //読み込みに使用した変数を空にする
  }
```

name	xIni	yIni	xEn	yEn	Wid	Hei
cross	0	0	168	12	12	12
laser	0	12	168	24	12	12
SlimRice	0	24	112	40	8	16
FatRice	0	40	140	56	10	16
WBall	0	56	224	72	16	16
Arrow	0	72	224	88	16	16
wind	0	88	224	104	16	16
Ball1	0	104	224	120	16	16
Ball2	0	120	224	136	16	16
Ball3	0	136	224	152	16	16
Ball4	0	152	224	168	16	16
LWBall	0	168	336	192	24	24
LArrow	0	192	336	216	24	24
Lball	0	216	336	240	24	24
BIGBall	368	0	512	240	48	48

図 44　画像分割用 CSV の中身

2.　無駄な処理の回避

弾幕 STG では大量の弾が画面上をうめつくすため膨大な処理が必要となり、メモリ容量が足りず処理が遅くなることも頻繁にあります。

そのため、できるだけ無駄な処理は省いておく必要があります。

例えば見えてもいない弾に画像をいつまでも当てはめておくのは非常に処理の無駄になります。

たとえば弾が１０発、１００発ならまだいいでしょう。

でも１０００発の弾の処理になると処理的にはギリギリ速度を保てている程度になっていることがあります。

さらに、画像データはサイズによっても処理の重さが変わってくるので巨大な画像を使う時だと１０００発扱うときに処理しきれずメモリオーバーしてしまうかもしれません。

そのため、このプログラムでは次のような処理をしています。

```
/*class bullet_c*/
/*弾の軌道演算処理及び描画処理*/
  void Draw(){
    if(alpha>0)
    Update();
    if(img!=null){
      pushMatrix();
      translate(xPos,yPos);
      rotate(angle+radians(90)+((Spin)?SpinAngle:0));
      tint(255,alpha);
      image(img,0,0,img.width*1.5,img.height*1.5);
```

```
        if(img.width>16&&Spin)
          SpinAngle=random(2*PI);
        else if(Spin)
          SpinAngle+=(SpinAngle+1)/60;
        popMatrix();
      }
      for(int i=0;i<EXP.length;i++)
      EXP[i].Draw();
    }

    void Update(){
      xPos+=speed*cos(angle);
      yPos+=speed*sin(angle);
      speed+=AdSpeed;
      angle+=AdAngle;
      alpha-=(alpha>0)?8:0;
      if(live&&!laser)
        alpha=255;
      else if(live)
        alpha=255;
      if((abs(xPos-GetMinX-GetWidth/2)>GetWidth/2+50||abs(yPos-GetHeight/2-GetMinY)>GetHeight/2+50))
      live=false;
      if(alpha<0){
        img=null;
      }
      laserT--;
      if(laser&&KEYS.GetKey(4)&&laserC==0)
        xPos=PL.PosX;
      if(!KEYS.GetKey(4))
        laserC++;
    }
```

/*画面上に存在する弾を爆破消し*/

```
for(int i=0;i<BL.length;i++){
    if(!BL[i].live)continue;
    BL[i].Broke();
 }
```

3. 敵の行動の大まかなブロック分け

ボスの処理を switch で分岐させることで個人的に記述が楽になる。

具体的には好きなタイミングで分岐変数を変えることで好きな行動に移すことが可能になる。

また、行動一覧を見やすくできる。

```
/*class Boss_c*/
/*switch で処理を分岐させる。具体的に何をするのかをブロック分けしている*/
  void Update(){
    if(HP>0){
      float imh=img.height,ima=IMGANGLE;
      if(PL.HIT(PosX,PosY,imh/2)||
      PL.HIT(PosX-30*cos(ima)-imh/6*sin(ima),PosY-30*sin(ima)-imh/6*cos(ima),imh/3)||
      PL.HIT(PosX+30*cos(ima)-imh/6*sin(ima),PosY+30*sin(ima)-imh/6*cos(ima),imh/3)){
        HP-=PL.shotDm*DRate;
        STB.SCORE+=(DRate!=0)?PL.shotDm*20:0;
      }
    }

    if(HP<0)HP=0;
    switch(Style){
      case 0:
        DamageRate(0.9);
        ACT1();//弾幕を張る
        break;
      case 5:
        Style=500;
        for(int i=0;i<BL.length;i++){
          if(!BL[i].live)continue;
          BL[i].Broke();
        }
        actCont=0;
        break;//ここから未実装部分
        /*
        actCont=0;
        DamageRate(0);
        if(IMGANGLE==0)
        Style++;
```

```
    IMGANGLE+=PI/60;
    IMGANGLE%=PI*2;
    if(abs(IMGANGLE)<PI/60)IMGANGLE=0;
    break;*/
  case 6:
    IMGANGLE=0;
    Move1(GetCenterX,GetMaxY-90,120-actCont+1);
    FixX=GetCenterX;
    FixY=GetMaxY-90;
    if(!(120-actCont+1>0))
    Style=9;
    break;
  case 9:
    Setup(300,GetCenterX,GetMaxY-90);
    Style++;
    break;
  case 10:
    DamageRate(0.9);
    ACT2();
    break;
  case 15:
    for(int i=0;i<BL.length;i++)
    BL[i].live=false;
    actCont=0;
    DamageRate(0);
    if(IMGANGLE==0)
    Style++;
    IMGANGLE+=PI/60;
    IMGANGLE%=PI*2;
    if(abs(IMGANGLE)<PI/60)IMGANGLE=0;
    break;
  case 16:
    IMGANGLE=0;
    Move1(GetCenterX,GetCenterY,120-actCont+1);
    FixX=GetCenterX;
    FixY=GetCenterY;
```

```
            if(!(120-actCont+1>0))
            Style=19;
            break;
        case 19:
            Setup(400,GetCenterX,GetCenterY);
            Style++;
            break;
        case 20:
            STB.POSE();
            break;//ここまで未実装部分
        case 100:
            if(actCont<60){
                Move1(GetCenterX,GetMinY+100,60-actCont);
                FixX=GetCenterX;
                FixY=GetMinY+100;
                if(PL.invisibleTime<120)actCont-=1;
            }else{
                Setup(500,GetCenterX,GetMinY+100);
                actCont=-1;
                Style=0;
                MOVINANGEL=0;
            }
            break;
        case 500:
            DEATH();//死んだ時の処理
        break;
        default:
            actCont=-1;
            DamageRate(0);
            Style=100;
    }
    actCont++;
}
```

４，　今後の出来る範囲で実装したい内容

・音楽及び効果音を入れる

音に関わる記述を書ききれなかった。

実装自体はすぐにでも可能。

・道中ステージの追加

ボスを登場させるのを遅らせて雑魚を沸かせる。

背景などをうまく変えれば自由に変わる。

・雑魚の追加

class Enemy_c を作りそれに雑魚の命令を記述する

・ステージの複数化

ボス記述自体を 1 ステージとして発展させれば可能

・ボス攻撃の多様化

3－3 にあげている内容を利用すれば可能

・リプレイ保存

ゲームの最初に時間などでシード値を決め、

そのシード値やフレームごとの入力したコマンドを CSV などで保存すれば

仮想的なリプレイ機能が使えると思われる

５，　素材提供サイト

素材提供してもらったサイト様一覧です（管理主敬称略）

・自機画像

夢蒼/musou （フリーゲーム素材＆無料ゲーム http://game.yu-nagi.com/htm/senntouki1.htm ）

k6_p1a.png

k6_p1b.png

・弾画像

ryooooooyan （いろいろ倉庫(主に東方弾幕風）

http://anystorage.blog53.fc2.com/blog-category-3.html ）

_shot.png （透過加工後 alpha_shot.png）

STG2.pde

```
import ddf.minim.spi.*;
import ddf.minim.signals.*;
import ddf.minim.*;
import ddf.minim.analysis.*;
import ddf.minim.ugens.*;
import ddf.minim.effects.*;

final int GetMinX=32,GetMaxX=416;
final int GetMinY=16,GetMaxY=464;
final int GetWidth=GetMaxX-GetMinX,GetHeight=GetMaxY-GetMinY;
final int GetCenterX=GetWidth/2+GetMinX,GetCenterY=GetHeight/2+GetMinY;
Boss_c BS;
Image_c IMG;
StatusBar_c STB;
Keys_c KEYS;
Player_c PL;
BackGround_c BG;
void setup(){
  size(640,480);
  frameRate(60);
  imageMode(CENTER);
  IMG=new Image_c();
  BS=new Boss_c();
  STB=new StatusBar_c();
  KEYS=new Keys_c();
  PL=new Player_c();
  BG=new BackGround_c();
  IMG.Setup();
  BS.Setup(500,GetCenterX,GetMinY-50);
  STB.Setup(BS.HP);
  PL.Setup();
}
void draw(){
    STB.Setup(BS.HPMAX);
    STB.GetBar(BS.HP);
```

```
  if(!STB.title){
    BG.Draw();
    BS.Draw();
    PL.Draw();
  }
    STB.Draw();
    if(KEYS.GetKey(7))
      save("Screenshot/"+nf(month(),2)+"-"+nf(day(),2)
          +"-"+nf(hour(),2)+"-"+nf(minute(),2)+"-"+nf(second(),2)+".png");
    noCursor();
}
void ReSet(){
  BS.Setup(500,GetCenterX,GetMinY-50);
  BS.ReSetup();
  STB.ReSetup();
  PL.ReSetup();
  PL.Setup();
}
void keyPressed(){
  KEYS.Pressed(key,keyCode);
}
void keyReleased(){
  KEYS.Released(key,keyCode);
}
```

Enemy.pde

```
class Boss_c{
  int actCont,Style,bulletConter,EXPT;
  float PosX,PosY,FixX,FixY,HP,HPMAX,MOVINANGEL,IMGANGLE,DRate;
  PImage img;
  Bullet_C[] BL;
  Exprotion_c[] EXP;
  Boss_c(){
    BL= new Bullet_C[1024];
    for(int i=0;i<BL.length;i++)
      BL[i]= new Bullet_C();
    EXP= new Exprotion_c[16];
    for(int i=0;i<EXP.length;i++)
      EXP[i]= new Exprotion_c();
    actCont=0;
    bulletConter=0;
    Style=2;
  }
  void Setup(float hp,float x,float y){
    HP=hp;
    HPMAX=hp;
    PosX=x;
    PosY=y;
    actCont=0;
    img=IMG.BOSS();
  }
  void ReSetup(){
    Style=2;
    bulletConter=0;
    MOVINANGEL=0;
    IMGANGLE=0;
    for(int i=0;i<BL.length;i++)
    BL[i]= new Bullet_C();
  }
  void Draw(){
    Update();
```

```
    pushMatrix();
    tint(255,255);
    imageMode(CENTER);
    translate(PosX,PosY);
    rotate(IMGANGLE);
    if(img!=null)
    image(img,0,0);
    popMatrix();
    for(int i=0;i<EXP.length;i++)
      EXP[i].Draw();
}
void Update(){
    if(HP>0){
      float imh=img.height,ima=IMGANGLE;
      if(PL.HIT(PosX,PosY,imh/2)||
      PL.HIT(PosX-30*cos(ima)-imh/6*sin(ima),PosY-30*sin(ima)-imh/6*cos(ima),imh/3)||
      PL.HIT(PosX+30*cos(ima)-imh/6*sin(ima),PosY+30*sin(ima)-imh/6*cos(ima),imh/3)){
        HP-=PL.shotDm*DRate;
        STB.SCORE+=(DRate!=0)?PL.shotDm*20:0;
      }
    }

    if(HP<0)HP=0;
    switch(Style){
      case 0:
        DamageRate(0.9);
        ACT1();
        break;
      case 5:
        Style=500;
        for(int i=0;i<BL.length;i++){
          if(!BL[i].live)continue;
          BL[i].Broke();
        }
        actCont=0;
        break;
```

```
    /*
    actCont=0;
    DamageRate(0);
    if(IMGANGLE==0)
    Style++;
    IMGANGLE+=PI/60;
    IMGANGLE%=PI*2;
    if(abs(IMGANGLE)<PI/60)IMGANGLE=0;
    break;*/
  case 6:
    IMGANGLE=0;
    Move1(GetCenterX,GetMaxY-90,120-actCont+1);
    FixX=GetCenterX;
    FixY=GetMaxY-90;
    if(!(120-actCont+1>0))
    Style=9;
    break;
  case 9:
    Setup(300,GetCenterX,GetMaxY-90);
    Style++;
    break;
  case 10:
    DamageRate(0.9);
    ACT2();
    break;
  case 15:
    for(int i=0;i<BL.length;i++)
    BL[i].live=false;
    actCont=0;
    DamageRate(0);
    if(IMGANGLE==0)
    Style++;
    IMGANGLE+=PI/60;
    IMGANGLE%=PI*2;
    if(abs(IMGANGLE)<PI/60)IMGANGLE=0;
    break;
```

```
        case 16:
          IMGANGLE=0;
          Move1(GetCenterX,GetCenterY,120-actCont+1);
          FixX=GetCenterX;
          FixY=GetCenterY;
          if(!(120-actCont+1>0))
          Style=19;
          break;
        case 19:
          Setup(400,GetCenterX,GetCenterY);
          Style++;
          break;
        case 20:
          STB.POSE();
          break;
        case 100:
          if(actCont<60){
            Move1(GetCenterX,GetMinY+100,60-actCont);
            FixX=GetCenterX;
            FixY=GetMinY+100;
            if(PL.invisibleTime<120)actCont-=1;
          }else{
            Setup(500,GetCenterX,GetMinY+100);
            actCont=-1;
            Style=0;
            MOVINANGEL=0;
          }
          break;
        case 500:
          DEATH();
        break;
        default:
          actCont=-1;
          DamageRate(0);
          Style=100;
    }
```

```
    actCont++;
}
void ACT1(){
    int firstFream;
    int cont;
    if(HP<=0)
    Style=5;
    firstFream=0;

    /*start*/
    /*this exmple is bullet pattern
    BL[bulletConter].CreateShot01(PosX,PosY,5,angle,14,10);
    bulletConter++;
    bulletConter%=BL.length;
    */

    if(actCont<60*(5+2)){
        float ine=radians(MOVINANGEL);
        Move2(FixX+50*cos(ine*3),FixY+50*sq(sin(ine)),1);
        /*end*/
    }else if(actCont<60*(8.5+15)&&actCont>60*8.5){
        float ine=radians(MOVINANGEL);
        Move2(FixX+20*cos(ine*3),FixY-30+20*sq(sin(ine)),1);
    }

    float angle;
    //println(actCont);
    if(actCont<60*5){
        angle=GetAngleTo(PL.PosX,PL.PosY,0);
        if(actCont%30==0){
            for(int i=0;i<8;i++){
                BL[bulletConter].CreateShot01(PosX,PosY,5,angle+i*360/8,14,0);
                bulletConter++;
                bulletConter%=BL.length;
            }
        }
```

```
    }else if(actCont<60*(5+2)){
      angle=random(360);
      if(actCont%20==0){
        for(int i=0;i<8;i++){
          BL[bulletConter].CreateShot01(PosX-(img.width-20)/2,PosY+20,3,angle+i*360/8,5,5);
          bulletConter++;
          bulletConter%=BL.length;
        }
      }
      Move1(GetCenterX,PosY,5);
      if(actCont%20==10){
        for(int i=0;i<8;i++){
          BL[bulletConter].CreateShot01(PosX+(img.width-20)/2,PosY+20,3,angle+i*360/8,5,10);
          bulletConter++;
          bulletConter%=BL.length;
        }
      }
    }else if(actCont<60*(7+1.5)){
      angle=degrees(IMGANGLE)-90;
      for(int i=0;i<2;i++){

BL[bulletConter].CreateShot01(PosX+50.*cos(radians(angle+i*180)),PosY+50.*sin(radians(angle+i*180)),3+0.1*(actCo
nt-60*7),angle+i*360/2,10,7);
        bulletConter++;
        bulletConter%=BL.length;
      }
    }else if(actCont<60*(8.5+15)){
      angle=90;
      if(actCont%2==0){
        for(int i=0;i<2;i++){
          BL[bulletConter].CreateShot01(PosX+46,PosY+52,9,angle+75*i-(sin(radians(actCont-60*8.5))+1)*25,6,11);
          bulletConter++;
          bulletConter%=BL.length;
          BL[bulletConter].CreateShot01(PosX-46,PosY+52,9,angle-75*i+(sin(radians(actCont-60*8.5))+1)*25,6,11);
          bulletConter++;
          bulletConter%=BL.length;
```

```
            }
        }
        if(actCont%(60*4)==60*2){
            BL[bulletConter].CreateShot01(PosX,PosY+50,1,GetAngleTo(PL.PosX,PL.PosY,0),14,12);
            bulletConter++;
            bulletConter%=BL.length;
        }
        if(actCont%5==0){

BL[bulletConter].CreateShot01(PosX,PosY+50,random(1.5,3),GetAngleTo(PL.PosX,PL.PosY,0)+random(-25,25),3,9);
            bulletConter++;
            bulletConter%=BL.length;
        }
    }else if(actCont<60*(23.5+1.5)){
        angle=degrees(IMGANGLE)-90;
        if(actCont%5==0){
          float ad=random(360);
          for(int i=0;i<8;i++){

BL[bulletConter].CreateShot01(PosX+50.*cos(radians(angle+i*45)),PosY+50.*sin(radians(angle+i*45)),3,angle+i*360/
8+ad,0,0);
            bulletConter++;
            bulletConter%=BL.length;
          }
        }
    }else actCont=0;

    if(actCont>60*(7+0.5)&&actCont<60*(7+1)||actCont>60*(28+0.5-5)&&actCont<60*(28+1-5))
        IMGANGLE+=PI/29;
    else if(actCont==60*(7+1))
        IMGANGLE=PI;
    MOVINANGEL++;
    //actCont%=120;
  }
  void ACT2(){
    if(HP<=0)
```

```
    Style=15;
  }

  void DEATH(){
    if(actCont%3==0&&actCont<120){
      if(img!=null)

EXP[EXPT].Exprotion(PosX+random(img.width)-img.width/2,PosY+random(img.height)-img.height/2,20,random(20,6
0));
      EXPT++;
      EXPT%=EXP.length;
    }
    if(actCont==120){
      EXP[EXPT].Exprotion2(PosX,PosY,180,GetHeight*2);
      EXPT++;
      EXPT%=EXP.length;
    }
    if(actCont==180){
      img=null;
    }
    if(actCont==360){
      STB.setTime=0;
      STB.CLEAR();
    }
  }

  void Move1(float x,float y,int freams){
    if(freams>0&&(x!=PosX||y!=PosY)){
      PosX+=(x-PosX)/freams;
      PosY+=(y-PosY)/freams;
    }
  }
  void Move2(float x,float y,float speed){
    float angle= GetAngleTo(x,y,1);
    if(sq(x-PosX)+sq(y-PosY)<sq(speed))
    speed=sqrt(sq(x-PosX)+sq(y-PosY));
```

```
    PosX+=cos(angle)*speed;
    PosY+=sin(angle)*speed;
  }
  float GetAngleTo(float x, float y,int type){
    if(type==0)
    return degrees(atan2(y-PosY,x-PosX));
    else
    return atan2(y-PosY,x-PosX);
  }
  boolean HIT(float tx,float ty,float tr){
    boolean hits=true;
    for(int i=0;i<BL.length;i++){
      if(!hits)continue;
      hits=(BL[i].HIT(tx,ty,tr))?false:true;
    }
    return hits;
  }
  void DamageRate(float rate){//
    DRate=rate;
  }
}
```

Image.pde

```
class Image_c{
  PImage loadimg;
  PImage [][]bulletImage;
  PImage []PlayerImage;
  PImage BossImg;
  BufferedReader loadImageData;
  Image_c(){
  }
  void Setup(){
    PImage []bulletImageType;
    bulletImageType=new PImage[15];
    bulletImage=new PImage[15][14];//[type][color];
    String GetLine="setup";
    loadimg=loadImage("data/alpha_shot.png");
    loadImageData=createReader("data/alpha_shot_pixellist.csv");
    for(int Z=0;GetLine!=null;Z++){
      try{
        GetLine=loadImageData.readLine();
      }catch(IOException e){
        e.printStackTrace();
        GetLine=null;
      }
      if(Z==0||GetLine==null)continue;

      String[] GetData=split(GetLine,",");

      int []INI=new int[6];
      for(int i=0;i<GetData.length-1;i++){
        INI[i]=int(GetData[i+1]);
      }

      bulletImageType[Z-1]=createImage(INI[2]-INI[0],INI[3]-INI[1],ARGB);
      for(int y=INI[1];y<INI[3];y++){
        for(int x=INI[0];x<INI[2];x++){
          bulletImageType[Z-1].pixels[(y-INI[1])*(INI[2]-INI[0])+(x-INI[0])]=loadimg.pixels[y*loadimg.width+x];
```

```
          }
        }
        for(int i=0;i<bulletImage[Z-1].length;i++){
          bulletImage[Z-1][i]=createImage(INI[4],INI[5],ARGB);

          if(Z-1!=14){
            for(int y=0;y<INI[5];y++){
              for(int x=0;x<INI[4];x++){
                bulletImage[Z-1][i].pixels[y*INI[4]+x]=bulletImageType[Z-1].pixels[y*INI[2]+x+i*INI[4]];
              }
            }
          }else{
            for(int y=0;y<INI[5];y++){
              for(int x=0;x<INI[4];x++){
                INI[2]=bulletImageType[Z-1].width;

bulletImage[Z-1][i].pixels[y*INI[4]+x]=bulletImageType[Z-1].pixels[y*INI[2]+x+i%3*INI[4]+(INI[4]*INI[5]*3)*int(i/3)];
              }
            }
          }
        }
      }
      loadimg=loadImage("data/cannon-a.png");
      BossImg=loadimg;
      PlayerImage=new PImage[2];
      for(int i=0;i<PlayerImage.length;i++){
        String[] data={"data/k6_p1a.png","data/k6_p1b.png"};
        PlayerImage[i]=loadImage(data[i]);
      }
      loadimg=null;
    }
    PImage BULLET(int type,int col){
      return bulletImage[type][col];
    }
    /*PImage ENEMY(int type,int act){
      return enemyImage[type][col];
```

```
  }*/
  PImage BOSS(){
    return BossImg;
  }
  PImage PLAYER(int cont){
    return PlayerImage[cont%PlayerImage.length];
  }
}
```

Player.pde

```
class Player_c{
  int actCont,invisibleTime,graCont,bulletConter,LIFE,BombC,waitTime;
  float PosX,PosY,hit,speed,alpha,shotDm,BOMBANM;
  Bullet_C[] BL;
  boolean live;
  Player_c(){
    BL=new Bullet_C[64];
    for(int i=0;i<BL.length;i++)
    BL[i]=new Bullet_C();
    LIFE=4;
    BombC=3;
    BOMBANM=60;
  }
  void ReSetup(){
    LIFE=4;
    BombC=3;
    BOMBANM=60;
    for(int i=0;i<BL.length;i++)
    BL[i]=new Bullet_C();
  }
  void Setup(){
    live=true;
    if(LIFE>0){
      waitTime=30;
      hit=8;
      speed=5;
      PosX=GetCenterX;
      PosY=GetMaxY-50;
      alpha=128;
      invisibleTime=0;
      for(int i=0;i<BS.BL.length;i++){
        if(!BS.BL[i].live)continue;
          BS.BL[i].live=false;
      }
      graCont=0;
```

```
        shotDm=1.1;
        BombC=3;
        LIFE--;
      }else {
        PosX=GetCenterX;
        PosY=height*3;
        STB.GAMEOVER();
      }
    }
    void Draw(){
      if(live&&LIFE>0){
        Update();
        tint(255,alpha);
        imageMode(CENTER);

image(IMG.PLAYER(graCont),PosX+1,PosY+1,IMG.PLAYER(graCont).width+1,IMG.PLAYER(graCont).height+1);
        graCont++;
      }else {
        int waittime=30;
        if(actCont>waittime){
          actCont=0;
          alpha=0;
        }else if(actCont==waittime){
          Setup();
        }else if(LIFE>0){
          for(int i=0;i<BS.BL.length;i++){
            if(!BS.BL[i].live)continue;
              BS.BL[i].live=false;
          }
        }
      }

      for(int i=0;i<BL.length;i++)
        BL[i].Draw();
      for(int i=0;i<BS.BL.length;i++)
        BS.BL[i].Draw();
```

```
    actCont++;
  }
  void Update(){
    if(live&&invisibleTime==120)
      live=BS.HIT(PosX,PosY,hit/2);
    if(alpha<1024*4&&actCont%8==0)
      alpha*=-2;
    else if(invisibleTime<120){
      noFill();
      strokeWeight(5);
      stroke(255,255-invisibleTime);
      arc(PosX,PosY,100,100,radians(map(invisibleTime,0,119,-150,270)),PI/2*3);
      strokeWeight(1);
      invisibleTime++;
    }
    if(waitTime<=0){
      if(KEYS.GetKey(0))PosY-=speed;
      if(KEYS.GetKey(1))PosY+=speed;
      if(KEYS.GetKey(2))PosX+=speed;
      if(KEYS.GetKey(3))PosX-=speed;
      speed=4;
      if(KEYS.GetKey(4)){
        BL[bulletConter].PLCS02(PosX,PosY-9,11,-90);
        bulletConter++;
        bulletConter%=BL.length;
        speed=1.7;
      }else if(actCont%5==0&&KEYS.GetKey(6)){
        BL[bulletConter].PLCS01(PosX+5,PosY-8,15,-90);
        bulletConter++;
        bulletConter%=BL.length;
        BL[bulletConter].PLCS01(PosX-5,PosY-8,15,-90);
        bulletConter++;
        bulletConter%=BL.length;
      }
      if(KEYS.GetKey(5)&&(BombC>0&&invisibleTime==120)){
        BOMBANM=0;
```

```
            Bomb();
          }
        }else waitTime--;

        if(PosX<GetMinX+hit/2)PosX=GetMinX+hit/2;
        if(PosX>GetMaxX-hit/2)PosX=GetMaxX-hit/2;
        if(PosY<GetMinY+hit/2)PosY=GetMinY+hit/2;
        if(PosY>GetMaxY-hit/2)PosY=GetMaxY-hit/2;
        if(BOMBANM<60){
          for(int i=0;i<BS.BL.length;i++){
            if(!BS.BL[i].live)continue;
              BS.BL[i].live=false;
          }

          fill(255,map(BOMBANM,0,59,255,50));
          ellipse(PosX,PosY-50,map(BOMBANM,0,59,0,height*2),map(BOMBANM,0,59,0,height*2));
          BOMBANM++;
        }
    }
    boolean HIT(float tx,float ty,float tr){
        boolean hits=false;
        for(int i=0;i<BL.length;i++){
          if(hits)continue;
          hits=(BL[i].HIT(tx,ty,tr))?true:false;
          PL.shotDm=BL[i].DM;
        }
        return hits;
    }
    void Bomb(){

        invisibleTime=-100;
        BombC--;
    }
}
class Keys_c{
    Key_c[] KY;
```

```
  Keys_c(){
    KY=new Key_c[8];
    KY[0]=new Key_c((char)CODED,UP);
    KY[1]=new Key_c((char)CODED,DOWN);
    KY[2]=new Key_c((char)CODED,RIGHT);
    KY[3]=new Key_c((char)CODED,LEFT);
    KY[4]=new Key_c('z',0);
    KY[5]=new Key_c('x',0);
    KY[6]=new Key_c('c',0);
    KY[7]=new Key_c('s',0);
  }
  boolean GetKey(int type){
    return KY[type%KY.length].isPressed;
  }
  void Pressed(char k,int kt){
    for(int i=0;i<KY.length;i++)
    KY[i].Pressed(k,kt);
  }
  void Released(char k,int kt){
    for(int i=0;i<KY.length;i++)
    KY[i].Released(k,kt);
  }
}
class Key_c{
  int KT;
  char K;
  boolean isPressed;
  Key_c(char k,int kt){
    KT=kt;
    K=k;
    isPressed=false;
  }
  void Pressed(char k,int kt){
    if(Check(k,kt))
    isPressed=true;
  }
```

```
  void Released(char k,int kt){
    if(Check(k,kt))
    isPressed=false;
  }
  boolean Check(char k,int kt){
    if(int(k)<int('a'))k+=32;
    if(k!=CODED){
      if(k==K)
        return true;
    }else{
      if(kt==KT)
        return true;
    }
    return false;
  }
}
```

background.pde

```
class BackGround_c{
  float []PosX,PosY,SizX,SizY;
  color []col;
  BackGround_c(){
    PosX=new float[34];
    PosY=new float[34];
    SizX=new float[10];
    SizY=new float[10];
    col =new color[10];
    for(int i=0;i<SizX.length;i++)
      Setup(i);
    for(int i=0;i<SizX.length*2+4;i++){
      PosX[i+10]=(i%2==0)?GetMinX:GetMaxX;
      PosY[i+10]=int(abs(i-10)/2)*height/5;
    }
  }
  void Setup(int i){
    SizY[i]=random(GetCenterY,GetMaxY);
    SizX[i]=random(15,30);
    PosX[i]=random(GetMinX,GetMaxX);
    PosY[i]=GetMinY-SizY[i]/2;
    col[i]=color(random(100,255),random(100,255),random(100,255),random(100,150));
  }
  void Draw(){
    background(0);
    rectMode(CENTER);
    for(int i=0;i<7;i++){
      stroke(255,100);
      line(PosX[i*2+20],PosY[i*2+20]-height/5,PosX[i*2+20+1],PosY[i*2+20]);
      line(PosX[i*2+20],PosY[i*2+20],PosX[i*2+20+1],PosY[i*2+20]-height/5);
      Update2(i*2+20,3,height/5*6);
    }
    for(int i=0;i<SizX.length;i++){

      fill(col[i]);
```

```
        noStroke();
        rect(PosX[i],PosY[i],SizX[i],SizY[i]);
        Update(i);

        fill(255,200);
        rect(PosX[i+10],PosY[i+10],100,10);
        Update2(i+10,10,GetMaxY);
      }
    }
    void Update(int i){
      if(PosY[i]-SizY[i]/2>GetMaxY)
        Setup(i);
      PosY[i]+=SizY[i]/SizX[i]/3;
    }
    void Update2(int i,int j,int max){
      if(PosY[i]-10/2>max)
      PosY[i]=0;
      PosY[i]+=j;
    }
}
```

bullet.pde

```
class Bullet_C{
  private int alpha,laserT,laserC,EXP_C;
  private float xPos,yPos,speed,angle,AdSpeed,AdAngle,hit,SpinAngle,DM;
  private PImage img;
  boolean live,Spin,laser;
  Exprotion_c[] EXP;
  Bullet_C(){
    live=false;
    DM=0;
    laserT=0;
    EXP = new Exprotion_c[10];
    for(int i=0;i<EXP.length;i++)
    EXP[i] = new Exprotion_c();
  }
  void CreateShot01(float x,float y, float s,float a,int type,int col){
    if(!live){
      xPos=x;
      yPos=y;
      speed=s;
      angle=radians(a);
      AdSpeed=0;
      AdAngle=0;
      alpha=255;
      live=true;
      img=IMG.BULLET(type,col);

      Spin=(type==0||type==14)?true:false;
      hit=(img.width<img.height)?img.width:img.height;
      hit/=2;
      hit*=0.6;
      laser=false;
    }
  }
  void CreateShot02(float x,float y, float s,float as,float a,float aa,int type,int col){
    if(!live){
```

```
      xPos=x;
      yPos=y;
      speed=s;
      angle=radians(a);
      AdSpeed=as;
      AdAngle=radians(aa);
      alpha=255;
      live=true;
      img=IMG.BULLET(type,col);

      Spin=(type==0||type==14)?true:false;
      hit=(img.width<img.height)?img.width:img.height;
      hit/=2;
      hit*=0.6;
      laser=false;
    }
  }
  void PLCS01(float x,float y, float s,float a){
    if(!live){
      xPos=x;
      yPos=y;
      speed=s;
      angle=radians(a);
      AdSpeed=0;
      AdAngle=0;
      alpha=255;
      live=true;
      img=IMG.BULLET(6,13);
      Spin=false;
      hit=(img.width<img.height)?img.width:img.height;
      hit/=2;
      laser=false;
      DM=1.1;
    }
  }
  void PLCS02(float x,float y, float s,float a){
```

```
    if(!live){
      xPos=x;
      yPos=y;
      speed=s;
      angle=radians(a);
      AdSpeed=0;
      AdAngle=0;
      alpha=255;
      live=true;
      img=IMG.BULLET(1,12);
      Spin=false;
      hit=(img.width<img.height)?img.width:img.height;
      hit/=2;
      laser=true;
      DM=0.3;
      laserC=0;
    }
  }
  void Draw(){
    if(alpha>0)
    Update();
    if(img!=null){
      pushMatrix();
      translate(xPos,yPos);
      rotate(angle+radians(90)+((Spin)?SpinAngle:0));
      tint(255,alpha);
      image(img,0,0,img.width*1.5,img.height*1.5);
      if(img.width>16&&Spin)
        SpinAngle=random(2*PI);
      else if(Spin)
        SpinAngle+=(SpinAngle+1)/60;
      popMatrix();
    }
    for(int i=0;i<EXP.length;i++)
    EXP[i].Draw();
  }
```

```
  void Broke(){
    live=false;
    EXP[EXP_C].Exprotion(xPos,yPos,10,hit*6);
    EXP_C++;
    EXP_C%=EXP.length;
  }
  void Update(){
    xPos+=speed*cos(angle);
    yPos+=speed*sin(angle);
    speed+=AdSpeed;
    angle+=AdAngle;
    alpha-=(alpha>0)?8:0;
    if(live&&!laser)
      alpha=255;
    else if(live)
      alpha=255;
    if((abs(xPos-GetMinX-GetWidth/2)>GetWidth/2+50||abs(yPos-GetHeight/2-GetMinY)>GetHeight/2+50))
    live=false;
    if(alpha<0){
      img=null;
    }
    laserT--;
    if(laser&&KEYS.GetKey(4)&&laserC==0)
      xPos=PL.PosX;
    if(!KEYS.GetKey(4))
      laserC++;
  }
  boolean HIT(float tx,float ty,float tr){
    if(sq(xPos-tx)+sq(yPos-ty)<sq(hit+tr)&&live){
      if(!laser){
        EXP[EXP_C].Exprotion(xPos,yPos,30,hit*12);
        EXP_C++;
        EXP_C%=EXP.length;
        live=false;
        speed=1;
        return true;
```

```
        }else if(laserT<=0){
          EXP[EXP_C].Exprotion(xPos,yPos,30,hit*12);
          EXP_C++;
          EXP_C%=EXP.length;
          laserT=5;
          return true;
        }
      }
      return false;
    }
}
class Exprotion_c{
    int time,tMax,type;
    float x,y,siz;
    Exprotion_c(){
    }
    void Exprotion(float _x,float _y,int _time,float _siz){
      x=_x+random(-10,10);
      y=_y+random(-10,10);
      time=_time;
      tMax=_time;
      siz=_siz;
      type=0;
    }
    void Exprotion2(float _x,float _y,int _time,float _siz){
      x=_x;
      y=_y;
      time=_time;
      tMax=_time;
      siz=_siz;
      type=1;
    }
    void Draw(){
      Update();
      if(time>0){
        float a=random(150,255);
```

```
        if(type==0){
          noFill();
          stroke(255,a,a-random(100),(time+100)/float(tMax)*255);
          strokeWeight(time/float(tMax)*10);
        }else {
          noStroke();
          fill(255);
          if(tMax-time>60)
          fill(255,time/float(tMax-60)*255);
        }
        a=map(time,tMax,0,0,siz);
        ellipse(x,y,a,a);
        strokeWeight(1);
      }
    }
    void Update(){
      if(time!=0)
      time--;
    }
}
```

music.pde

```
class Music_c{//未実装
    Minim music;
    Music_c(){
    }
}
class SE_c{
    Minim[] Se;
    SE_c(){
    }
}
```

statusbar.pde

```
class StatusBar_c{
  PFont font;
  PImage img;
  float HP,MAXHP,frames,f;
  int actTime,setTime,SCORE,HISCORE,setSCORE;
  boolean pose=false,gameover=false,title=true,Clear=false;

  StatusBar_c(){
    font=loadFont("data/AgencyFB-Bold-48.vlw");
    img=loadImage("data/Frame.png");
    HISCORE=10000;
  }
  void ReSetup(){
    SCORE=0;
    actTime=0;
    setTime=0;
  }
  void Setup(float BOSSHP){
    MAXHP=BOSSHP;
    textFont(font,16);
    f=0;
  }
  void GetBar(float E_HP){
    HP=E_HP;
  }
  void Draw(){

    fill(255,0,0,PL.alpha);
    stroke(255,PL.alpha);
    ellipse(PL.PosX,PL.PosY,PL.hit,PL.hit);

    imageMode(CORNER);
    tint(255,255);
    image(img,0,0);
    stroke(255);
```

```
fill(255,100,100);
rectMode(CORNER);
if(!title)
rect(GetMinX+4,GetMinY+5,map(HP,0,MAXHP,0,GetWidth-8),10);

f+=frameRate;
if(actTime%30==0){
  frames=f;
  f=0;
}
textAlign(RIGHT,BOTTOM);
fill(255);
textSize(16);
text(nf(frames,2,1)+"fps",width,height);
textSize(32);
textAlign(LEFT,BOTTOM);
text("SCORE: "+str(SCORE),GetMaxX+30,GetMinY+30);
textAlign(LEFT,TOP);
text("LIFE: ",GetMaxX+30,GetMinY+40);
imageMode(CORNER);
for(int i=0;i<PL.LIFE-1;i++)
image(IMG.PLAYER(0),GetMaxX+30+textWidth("LIFE: ")+60*i,GetMinY+25,60,60);
textAlign(LEFT,TOP);
text("BOMB: "+str(PL.BombC),GetMaxX+30,GetMinY+40+42);
text("HISCORE: "+str(HISCORE),GetMaxX+30,GetMinY+130);
if(HISCORE<SCORE)HISCORE=SCORE;
actTime++;
if(pose)
  POSE();

if(gameover)
  GAMEOVER();

if(Clear)
  CLEAR();
```

```
    if(title)
      TITLE();

    noFill();
    stroke(255);
    rect(GetMinX,GetMinY,GetWidth-1,GetHeight-1);
    if(!Clear)
    setSCORE=SCORE;
  }
  void POSE(){
    pose=true;
    noLoop();
    textSize(100);
    pushMatrix();
    translate(width/2,height/2);
    rotate(PI/7);
    textAlign(CENTER,CENTER);
    text("Development!!!",0,0);
    popMatrix();
  }
  void GAMEOVER(){
    gameover=true;
    fill(255,200,200);
    textSize(22);
    textAlign(CENTER,CENTER);
    text("-GAMEOVER-",GetCenterX,GetCenterY);
    if(setTime>5&&setTime%2==0)
    text("Please Z kay",GetCenterX,GetCenterY+20);
    if(actTime%30==0)
    setTime++;
    if(setTime>5&&KEYS.GetKey(4)){
      title=true;
      setTime=0;
      actTime=0;
    }
  }
```

```
void CLEAR(){
  int enSCORE;
  Clear=true;
  PL.waitTime=100;
  fill(0,100);
  noStroke();
  rectMode(CORNER);
  rect(GetMinX,GetMinY,GetWidth-1,GetHeight-1);
  fill(255,255);
  textSize(22);
  textAlign(CENTER,CENTER);
  text("-GAMECLEAR-",GetCenterX,GetMinY+100);
  textAlign(LEFT,CENTER);
  text("LIFE BONUS: 2000 x "+str(PL.LIFE-1)+" = "+str((PL.LIFE-1)*2000),GetMinX+90,GetMinY+150);
  text("BOMB BONUS:    500 x "+str(PL.BombC)+" = "+str(PL.BombC*2000),GetMinX+90,GetMinY+200);
  enSCORE=setSCORE+2000*PL.LIFE+500*PL.BombC;
  textSize(30);
  text("TOTAL SCORE: "+str(enSCORE),GetMinX+90,GetMinY+300);
  if(enSCORE>SCORE){
    if(KEYS.GetKey(4)){
      SCORE=enSCORE;
      setTime=2;
    }else
      SCORE+=10;
  }
  if(SCORE==enSCORE){
    textSize(22);
    textAlign(CENTER,CENTER);
    if(setTime>2&&setTime%2==1)
      text("-Please Z kay-",GetCenterX,GetMinY+340);
    if(actTime%60==0)
    setTime++;
    if(setTime>2&&KEYS.GetKey(4)){
      title=true;
      setTime=0;
      actTime=0;
```

```
            Clear=false;
          }
        }
      }
    void TITLE(){
      gameover=false;
      fill(0);
      noStroke();
      rectMode(CORNER);
      rect(GetMinX,GetMinY,GetWidth-1,GetHeight-1);
      textAlign(CENTER,CENTER);
      fill(255);
      textSize(22);
      text("TODAY HIGH SCORE : "+str(HISCORE),GetCenterX,GetCenterY-50);
      text((setTime%2==1)?"-Please Z kay-":"-GAMESTART-",GetCenterX,GetCenterY+50);
      if(actTime%60==0)
        setTime++;

      if(KEYS.GetKey(4)&&setTime>1){
        title=false;
        setTime=0;
        ReSet();
      }
    }
}
```

『Click Circle』

作品紹介

『Click Circle』とは

上から落ちてくる円を
ひたすらクリックするゲーム

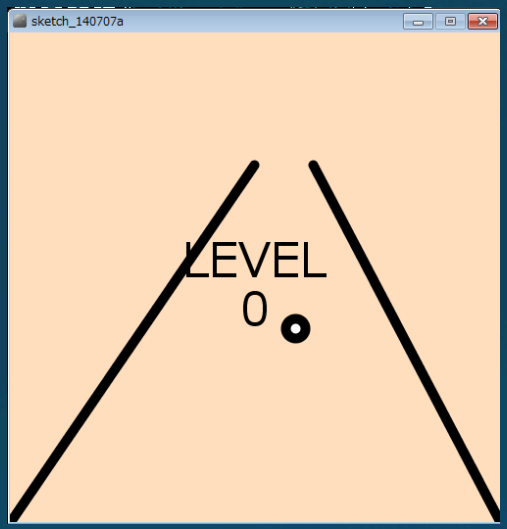
ゲーム自体はマウス一つでできる

ゲームのルール

- 上から落ちてくる円をクリックして消す
- レベル３５０でクリア
- 円が画面下を越えるとゲームオーバー

レベルによる変化

- 難易度の変化
- 背景色の変化



レベルが上がると背景が変わり、
円の挙動が激しくなる

『Click Circle』

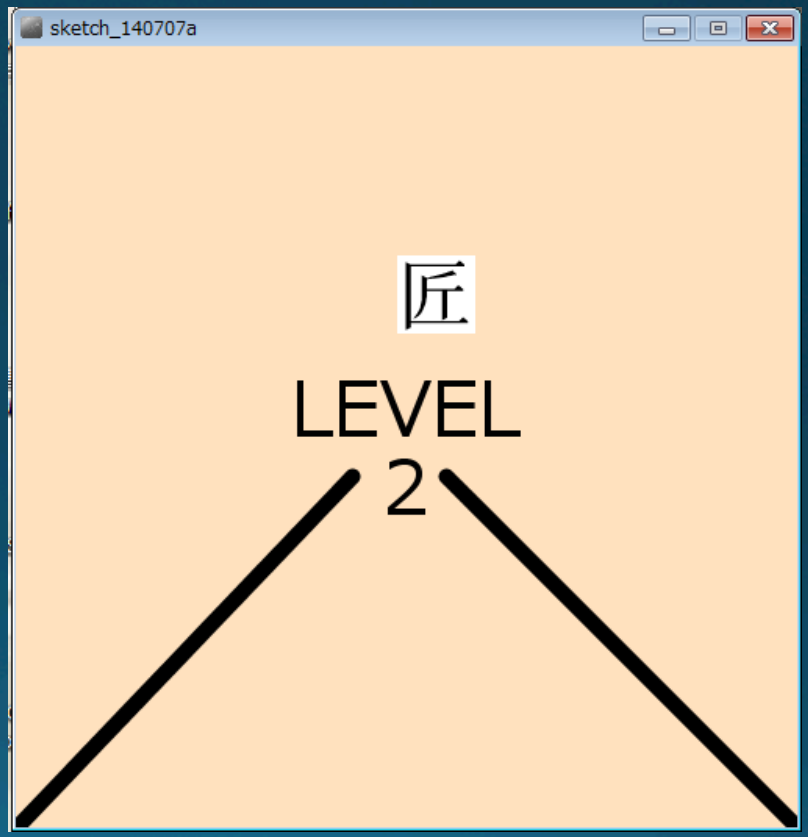
その他の機能

収録ＢＧＭ

- 『楔』　奥華子
- 『天体観測』BUMP OF CHICKEN
- 『Inscrutable Battle』松谷卓
- 『光芒の大地』秋山裕和

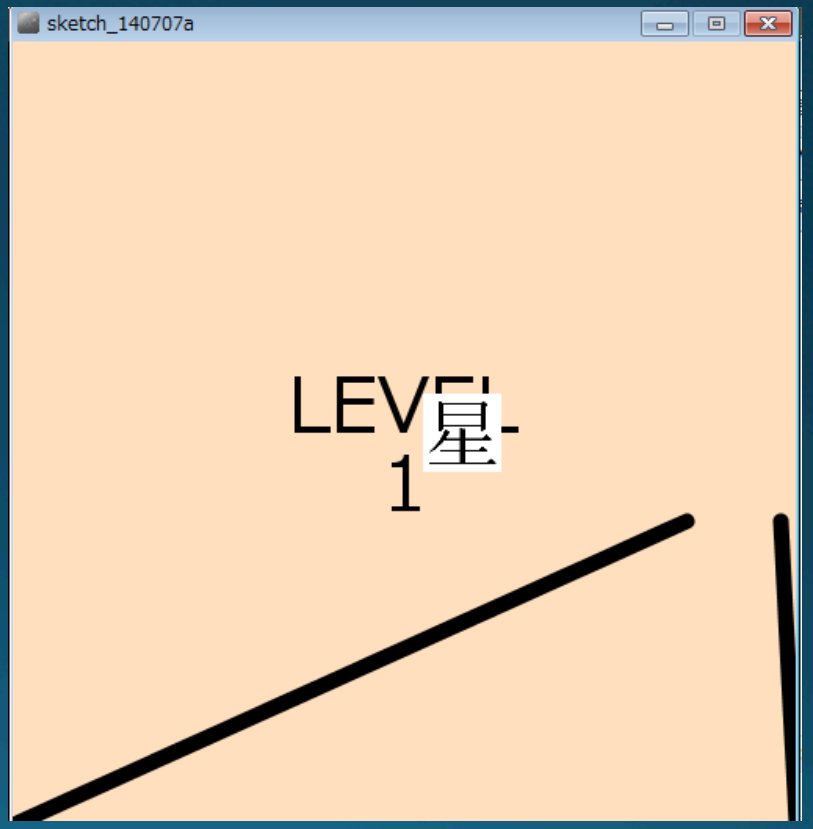
「２」キーを押した場合

- BGMの変更
- 円が匠になる



「３」キーを押した場合

- BGMの変更
- 円が星になる



星の連続撮影

- 特殊な技法を用いることで星を連続で撮影することが出来ます

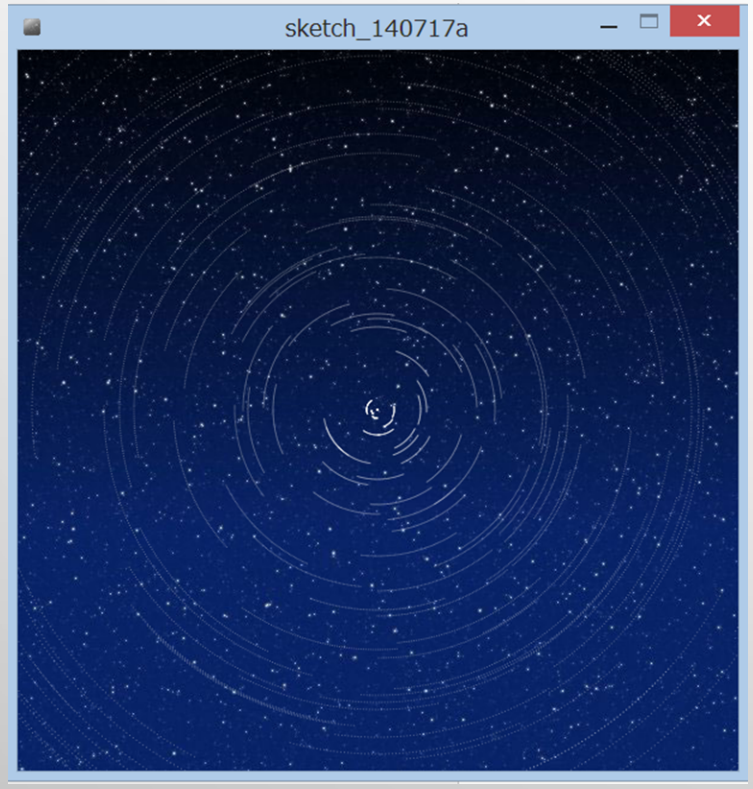
今回使用した主な命令

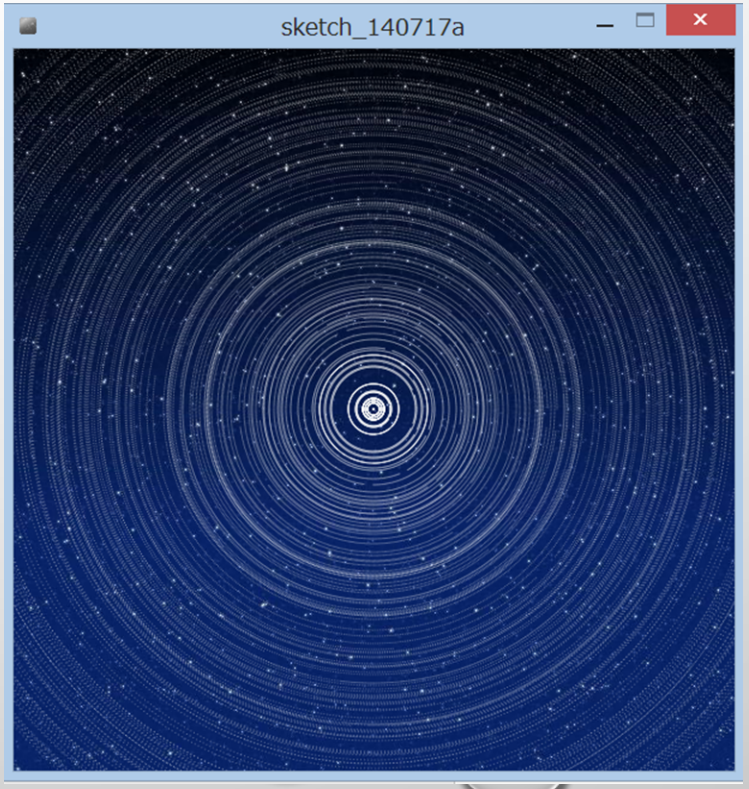
- FRAMERATE（円が移動する速さの設定）
- DISTANCE（距離などの設定）
- ANGLE（角度などの設定）
- IMAGE（画像）
- MINIM（音声再生）

などなど・・・

実際に動かしてみた

- 実行例





お借りした画像・音源

- 画像

キャノン

HTTP://CWEB.CANON.JP/CAMERA/DCAM/LINEUP/POWERSHOT/G16/FEATURE-MODE.HTML

HTTP://CWEB.CANON.JP/CAMERA/DCAM/LINEUP/POWERSHOT/G1XMK2/FEATURE-MODE.HTML

- 音楽

夢幻のオルゴール工房

HTTP://MIDWAY.SAKURA.NE.JP/MBS/DIARY.CGI?NO=347

作品コンセプト

- 映画のラストに流れるスタッフロールのようなものを作ってみたい。

- 映画『マトリックス』やほかの色々な作品でも見かけたことのあるような数字がランダムに変化しながら移動するものを作ってみたい。

作品解説・数字の羅列

- ‘randam()を使用して数を延々と変化させている。
- 数字の向きの変更に関してはrotate()を、数字の移動に関してはtranslate()をそれぞれ利用している。
- 画面外に出た時点でmillis()を利用して移動を止めている。

作品解説・エンドクレジット①

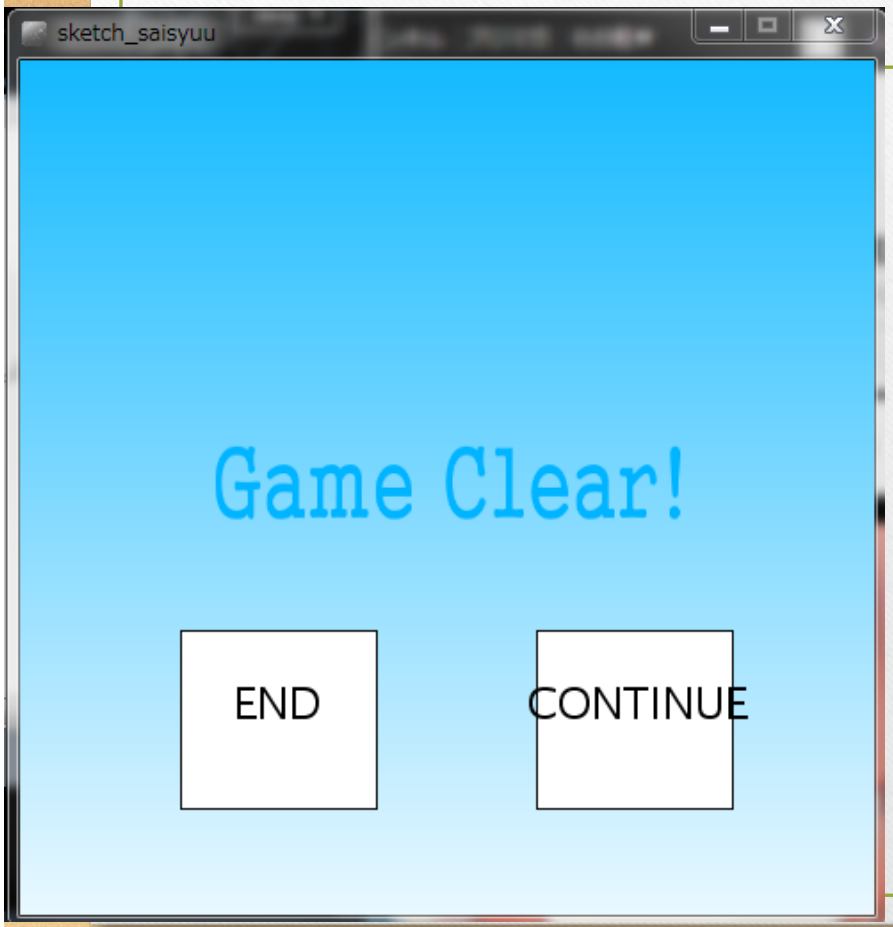
- resetMatrix()を利用して数字の移動に利用していたtranslate()の影響をなくしている。

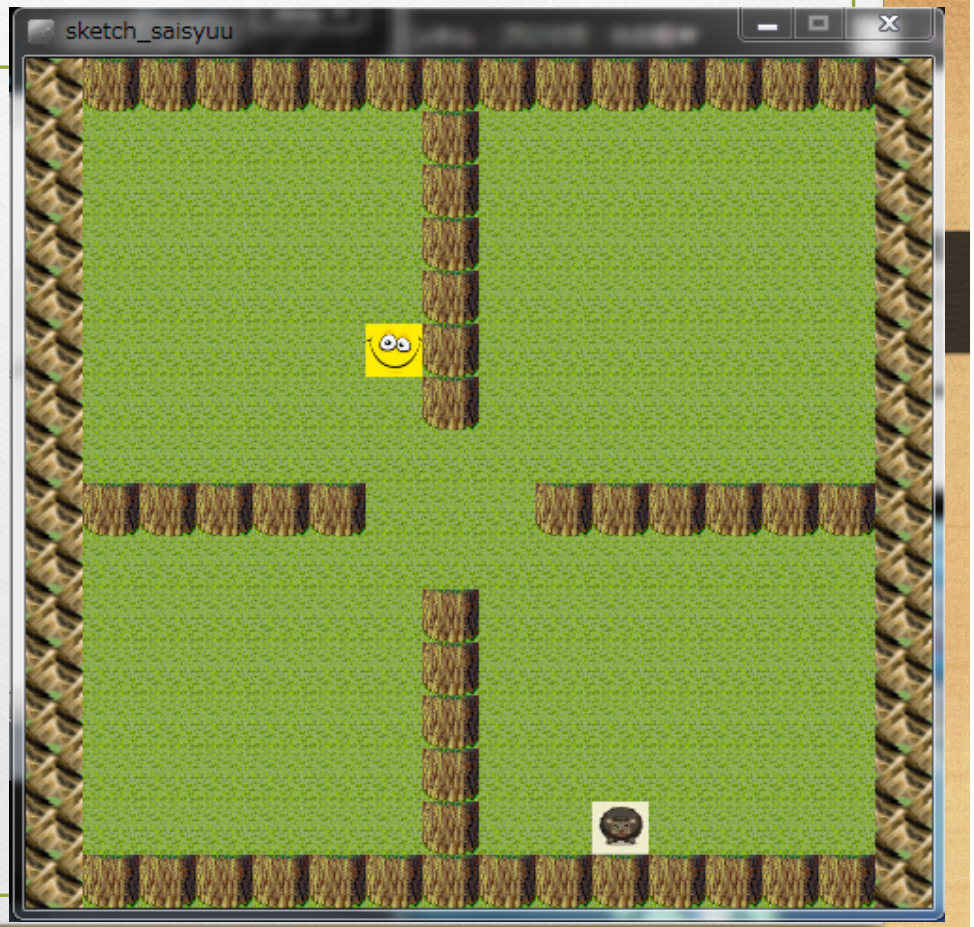
- 文字に関しては、外部のUTF-8Nに文字コードを変換した.txtファイルから読み込むことで、日本語の表示とプログラムの簡易性を高めている。

作品解説・エンドクレジット②

- 変数の値を参照するやり方と、millis()関数を参照するやり方をうまく使い分けることで文字を止めたり動かしたりを自由に行っている。

ゲーム概要





プログラム解説

マップ作成

```
int[][] map = {{2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2}
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2},
{2,1,1,1,1,1,0,0,0,1,1,1,1,1,1,2},
{2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,0,0,0,0,0,0,1,0,0,0,0,0,0,0,2},
{2,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2},
```

マップ描画

```
for (int i = 0; i < masu_y; i++) {
for (int k = 0; k < masu_x; k++) {
if(map[i][k] == 0) image(kusa, k * CHIP, i * CHIP,
CHIP,CHIP);
if(map[i][k] == 1) image(wall, k * CHIP, i * CHIP,
CHIP,CHIP);
if(map[i][k] == 2) image(yama, k * CHIP, i * CHIP,
CHIP,CHIP);
}
}
```

ターゲット移動

```
int f = int(random(1,5));
   if((f==1)&&(aa<14)){}
   if((f==2)&&(aa>1)){}
   if((f==3)&&(bb<14)){}
   if((f==4)&&(bb>1)){}
```

プレイヤー移動

- if((keyPressed==true)&&(keyCode==RIGHT)&&(a<14)&&((map[b][a+1] == 0))){
- a=a+1;
- }
- 残りも同様

終了画面

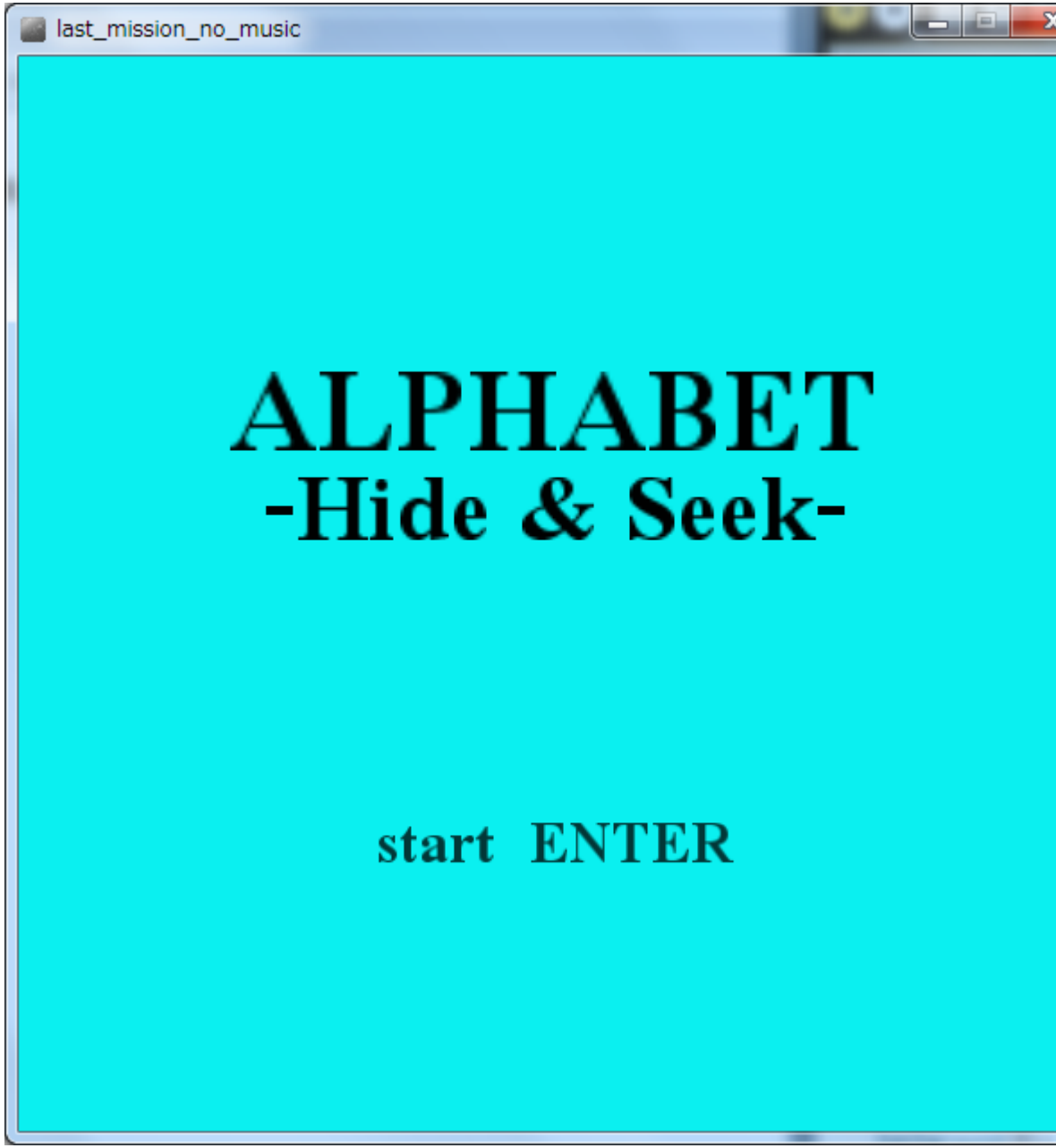
```
if(key=='e'){
  exit();
 }
 if(key=='c'){
//初期化
}
 }
```

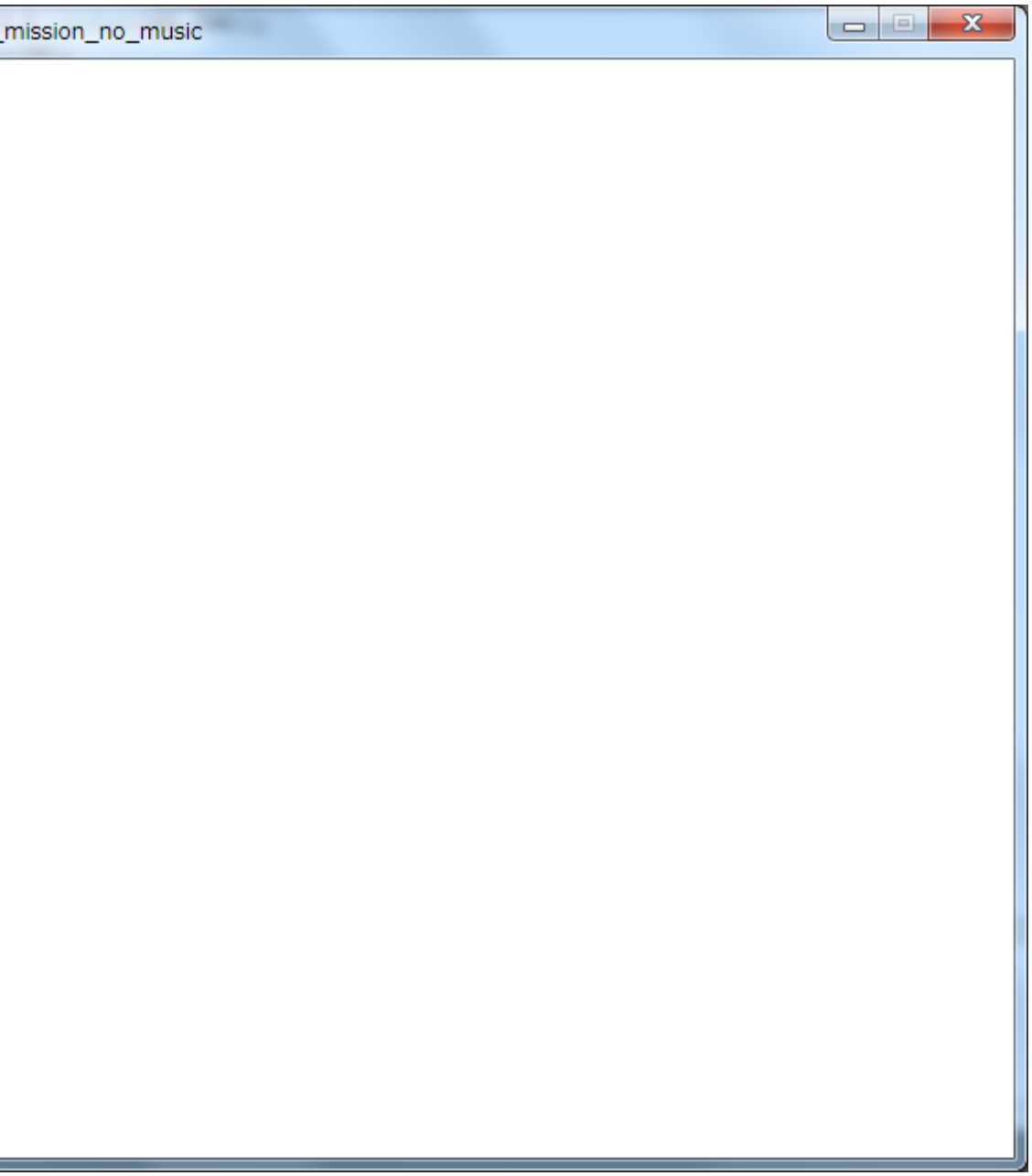
ゲームの仕様

- 画面内に隠れたアルファベットを探す
- 全5ステージ
- ステージによって出てくるアルファベットや図形が変わる
- クリア―画面にトータルタイムとミスをした回数が表示される

タイトル画面

ここでENTERを押す
ことで、
Stage1に移動します。

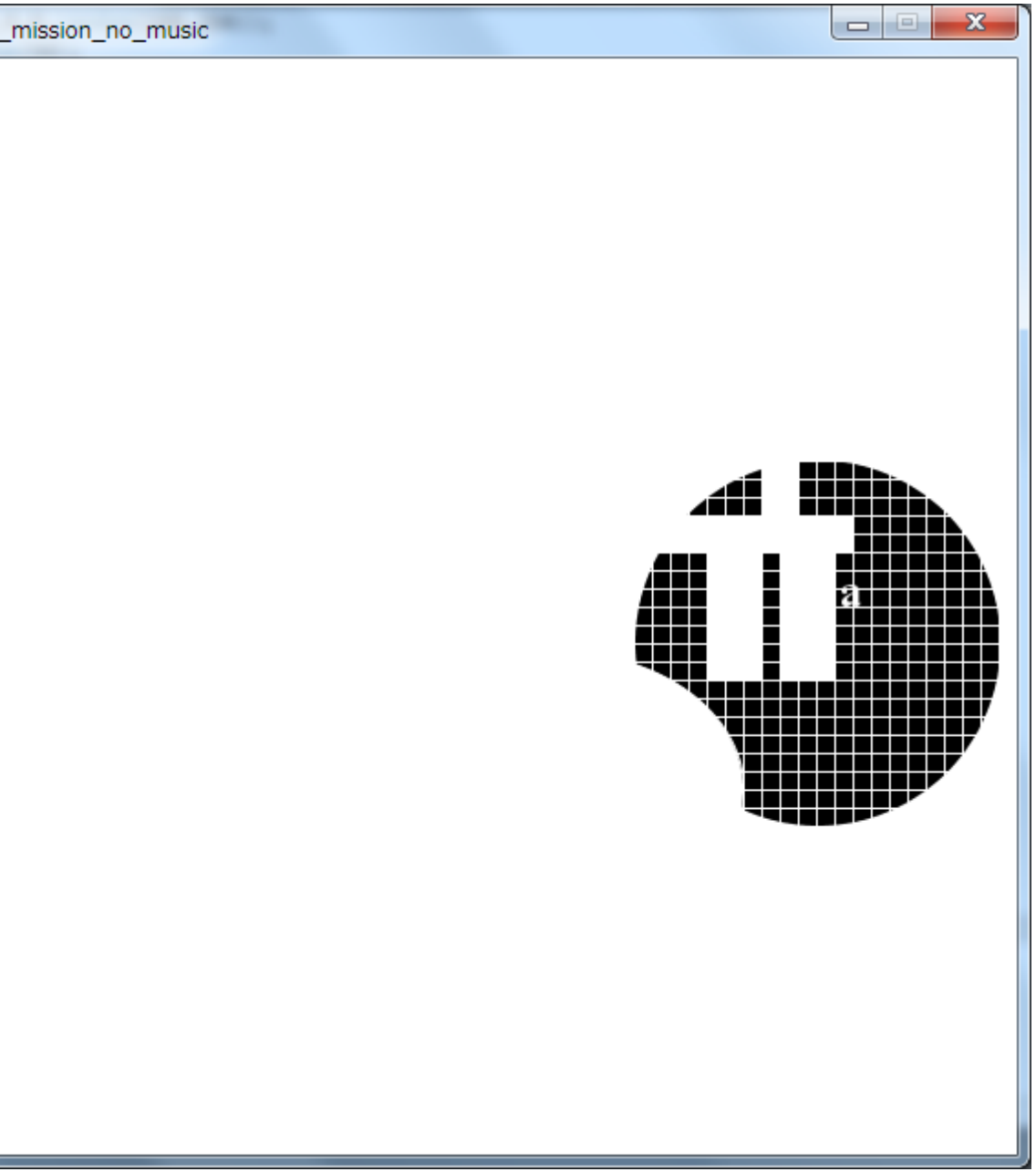




Stage1

一見何もないように見え
るが、

マウスをクリックしてみ
と・・・



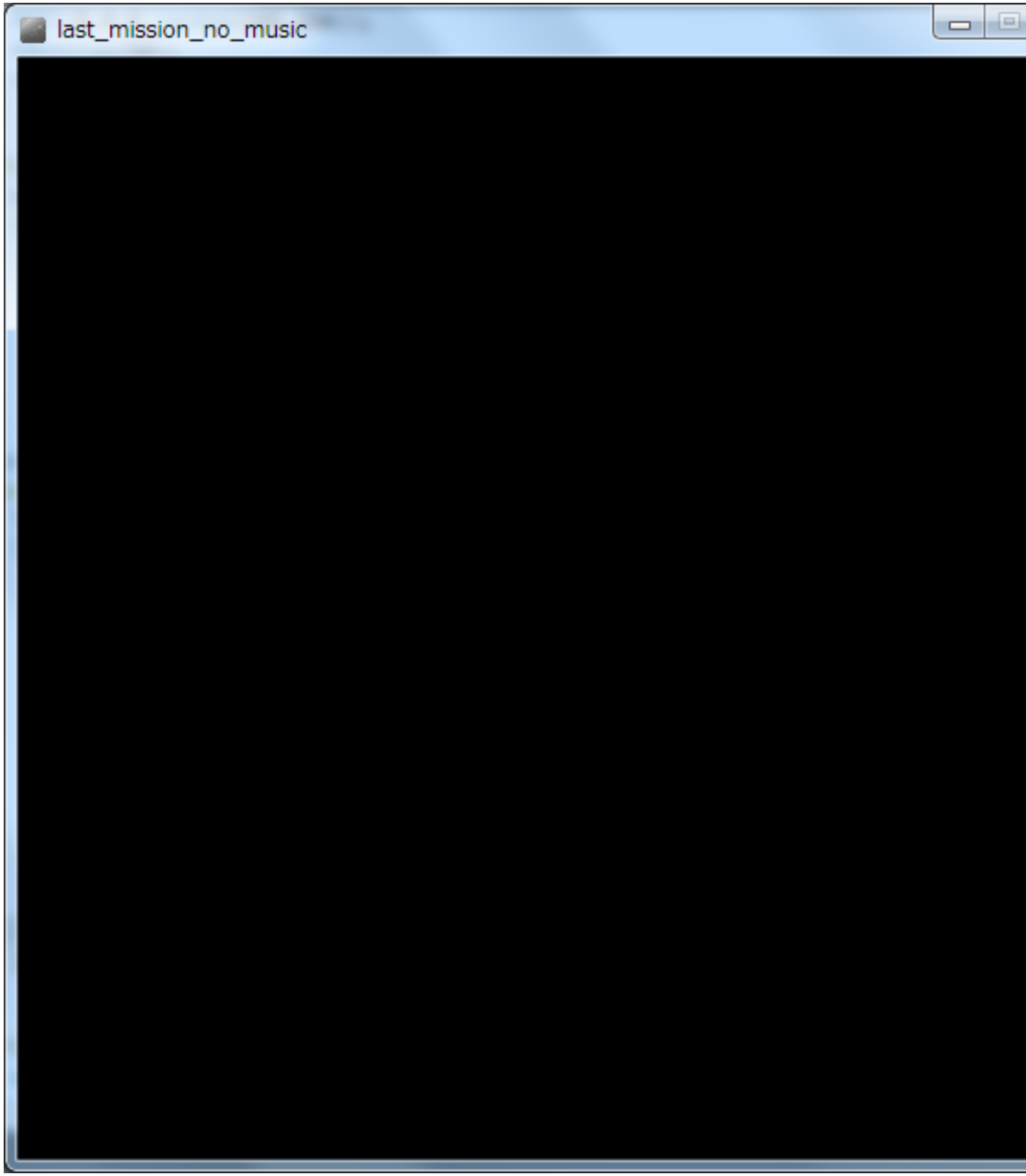
Stage1

画面内に図形が浮かび上がり、
よくみるとアルファベット小文字の「a」が表示されている
「a」を押してみると・・・

tage2

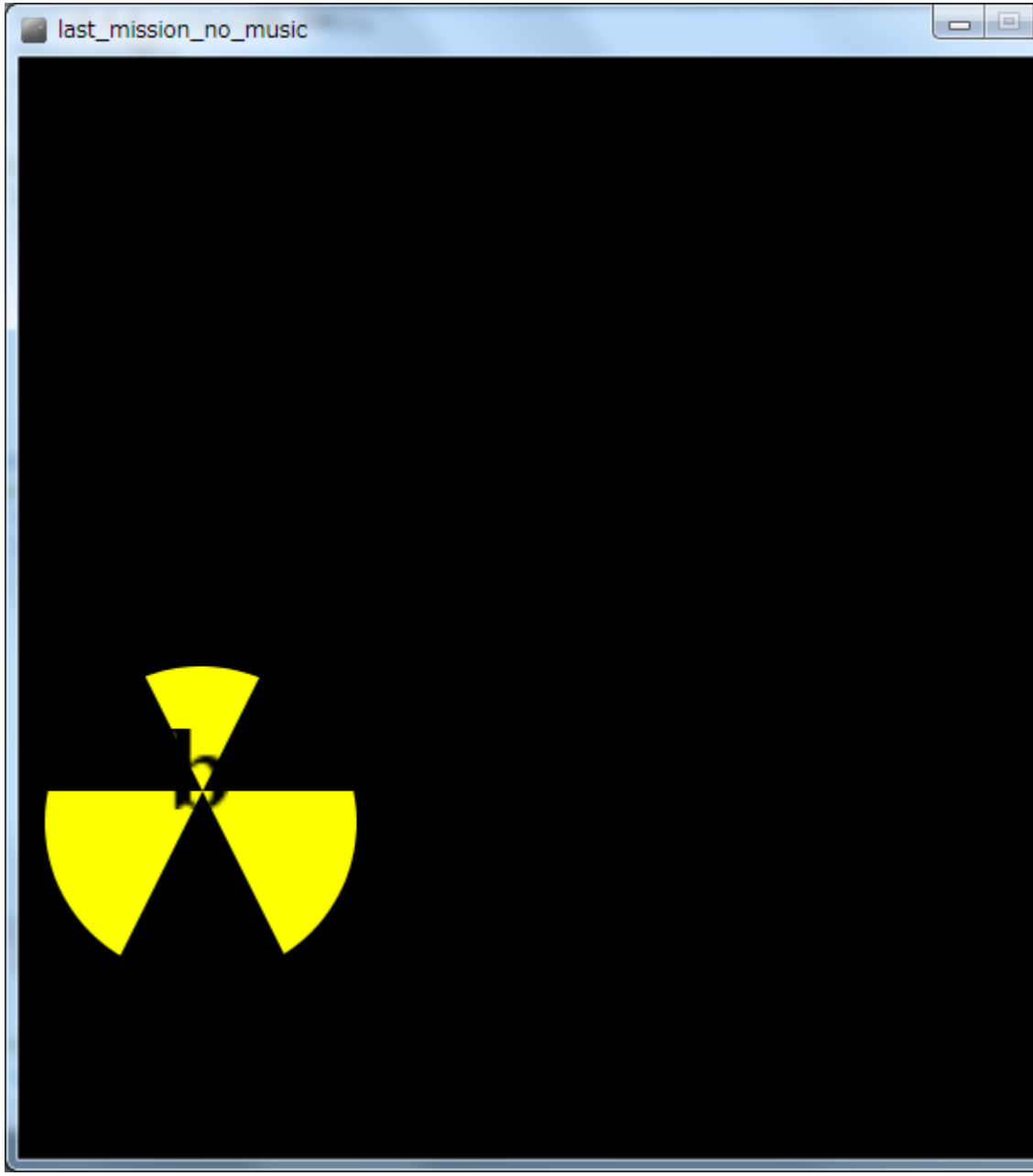
tage2に移動した

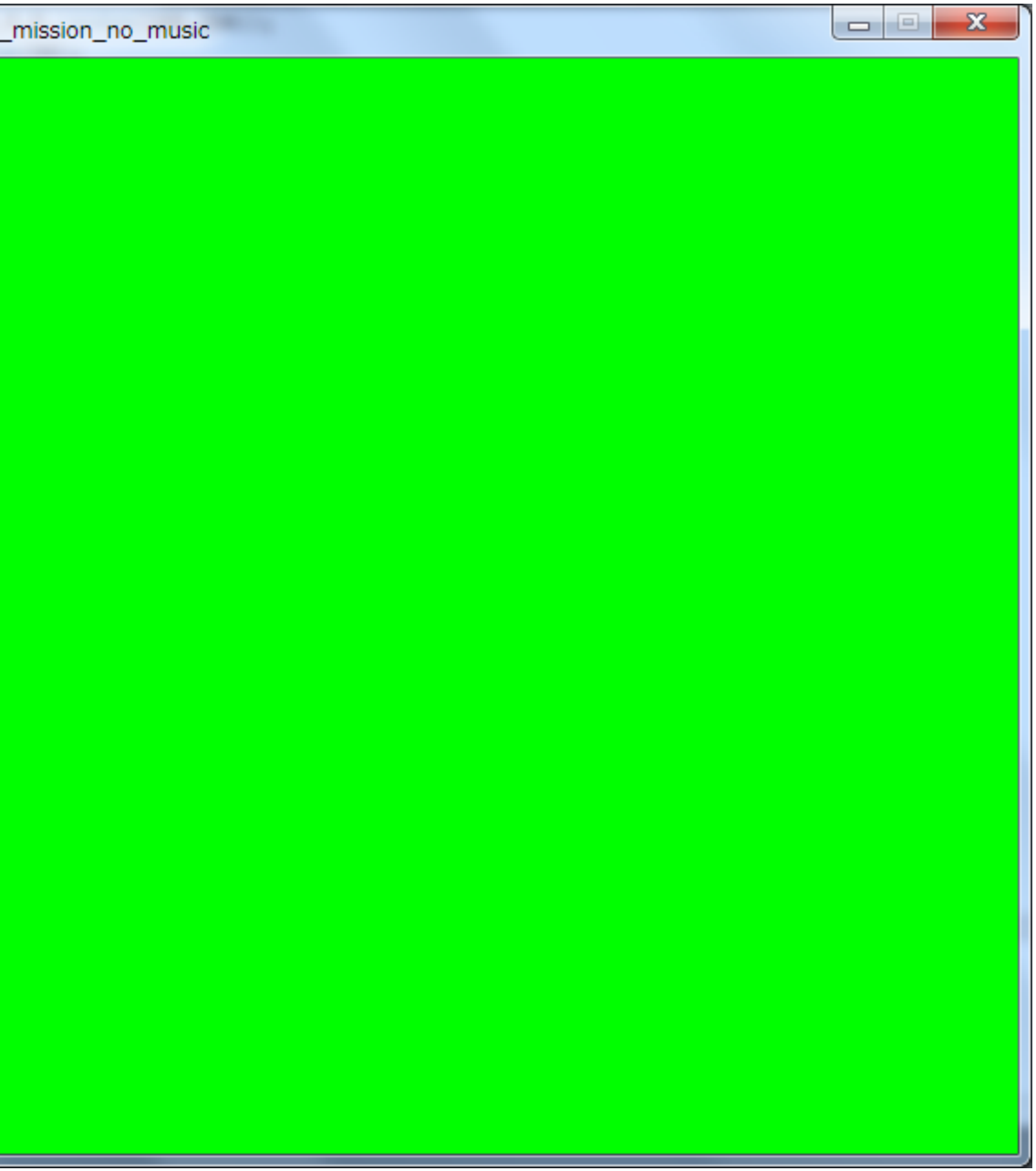
tage2では背景の色を黒
を塗りつぶした



tage2

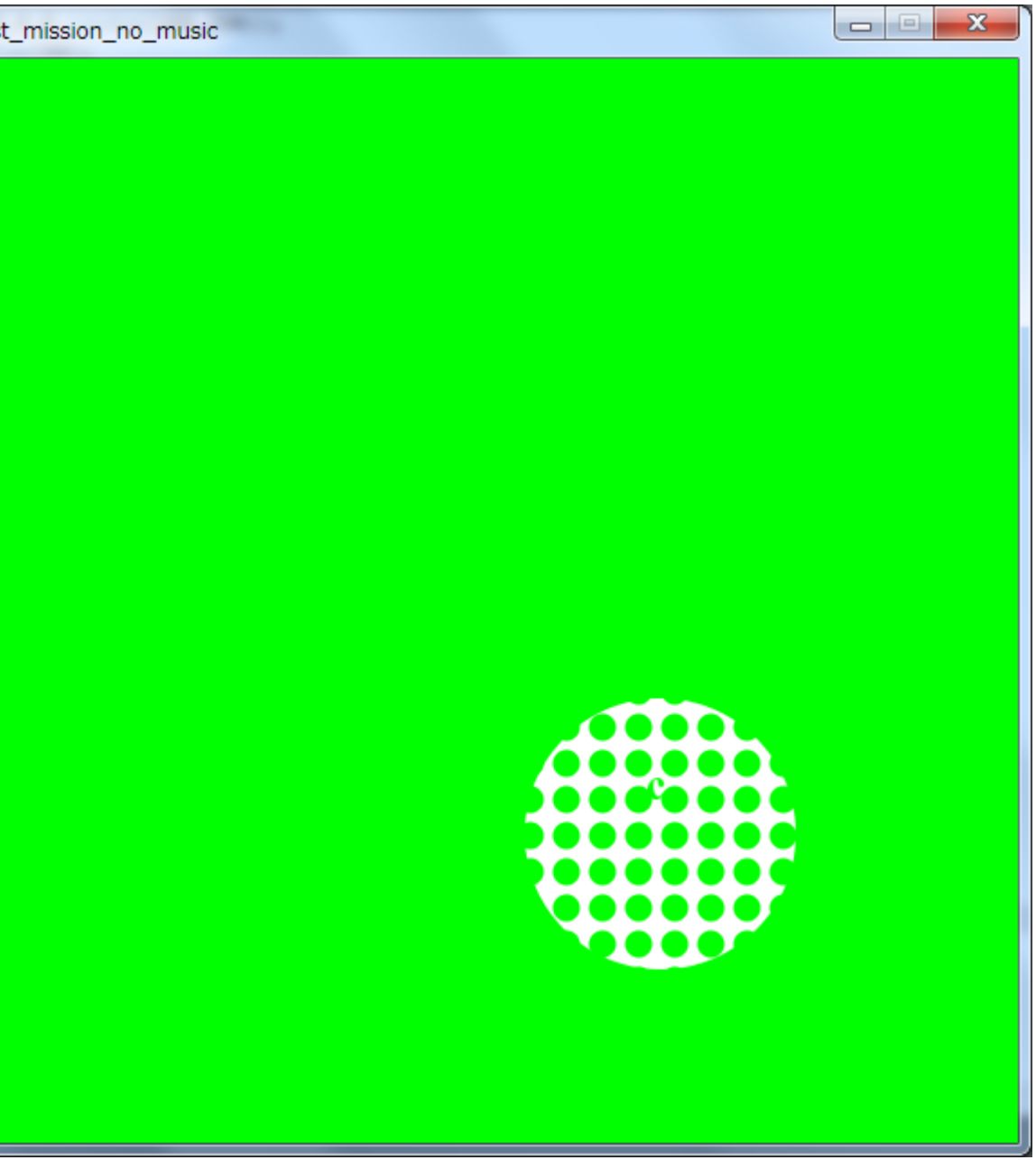
tage2で出てきたアル
ファベットは小文字の「b」
表示されている図形が変
わっているということに気
づいていただけるとうれし
いです





Stage3

Stage3では背景の色を
緑色で塗りつぶしました

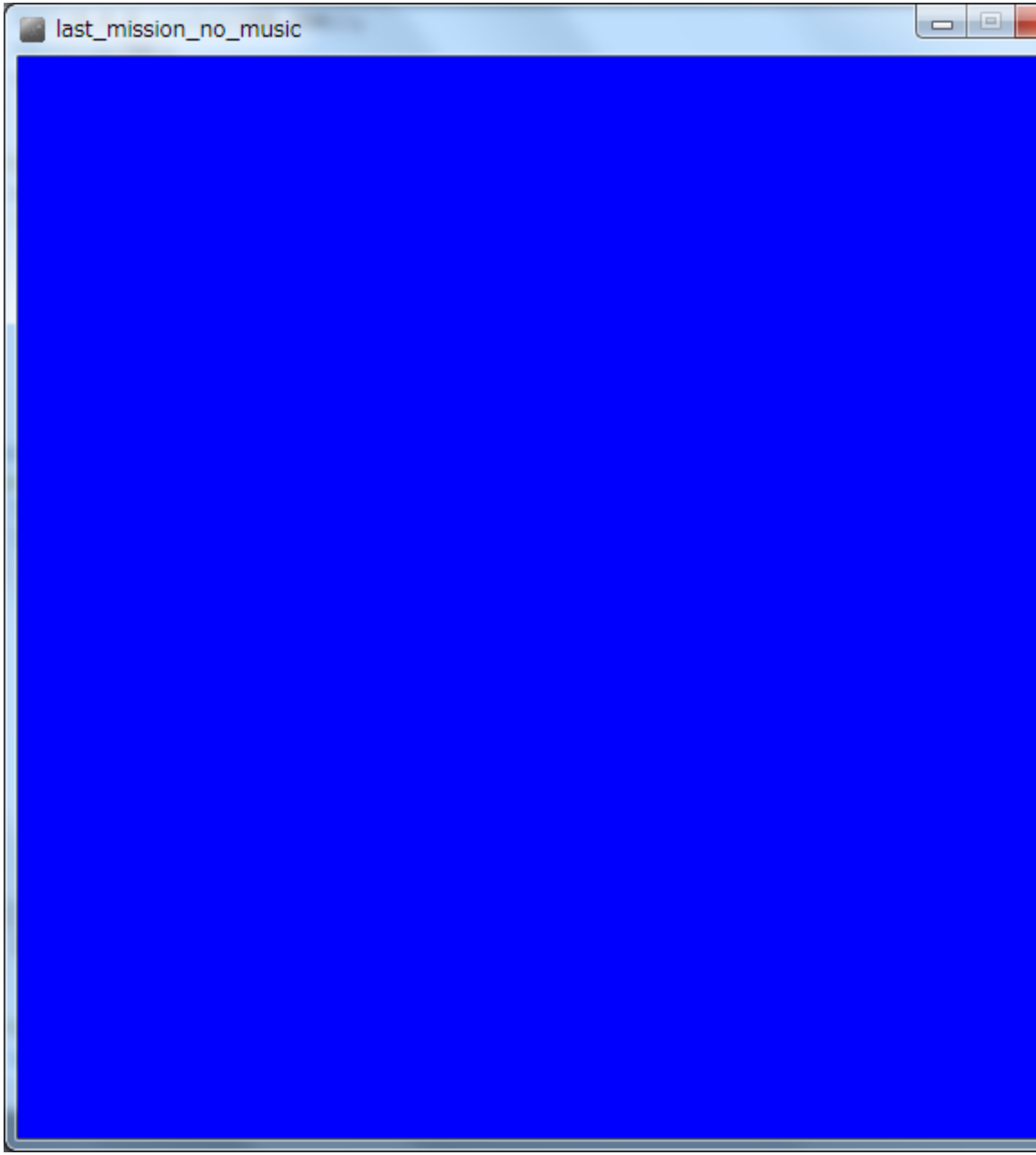


Stage3

Stage3では小文字の「c」
というアルファベットが画
面内に隠されていました

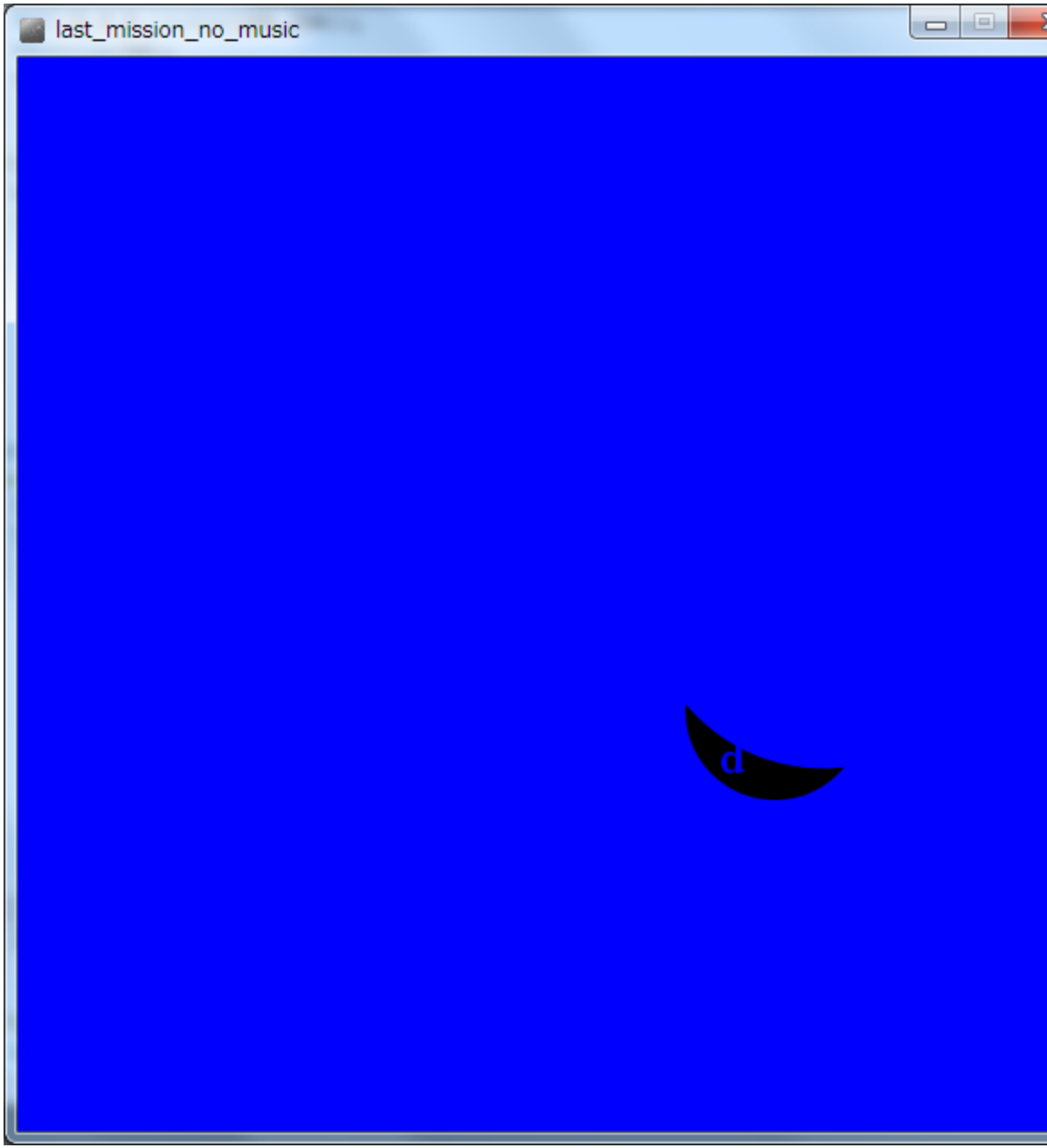
tage4

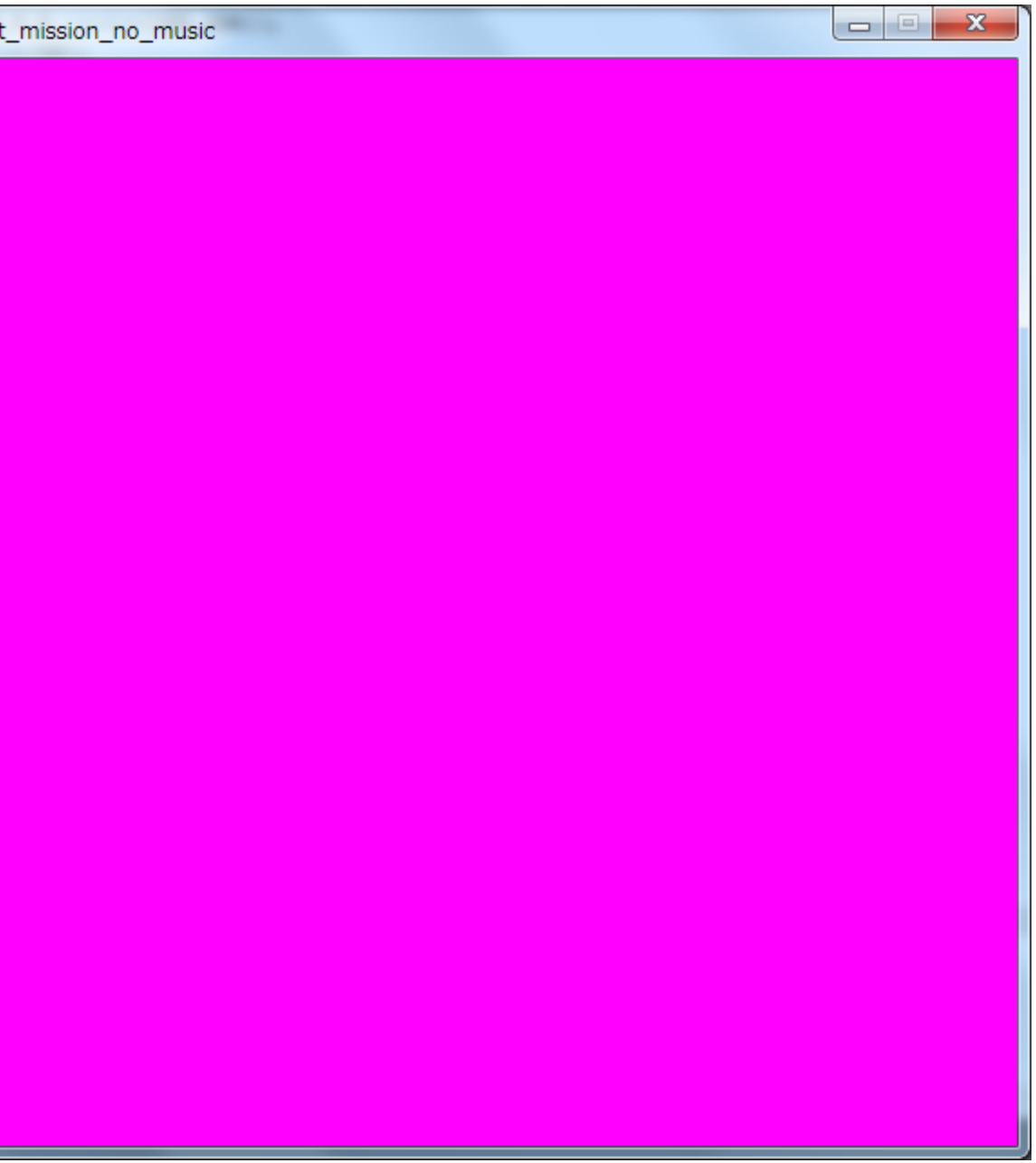
age4では背景の色を青
で塗りつぶしました



Stage4

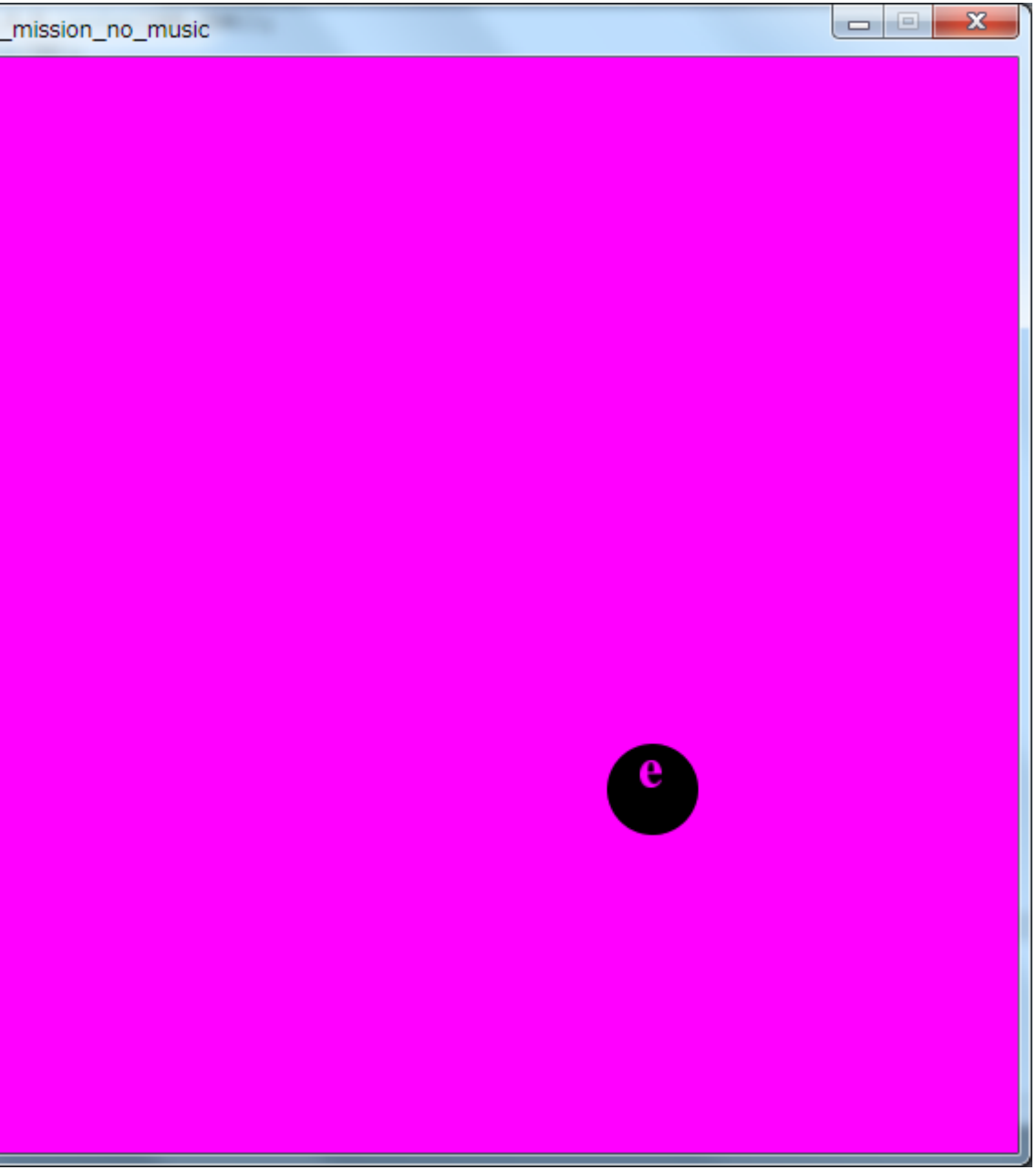
Stage4からは障害物とし
ては少し変わった、
動く障害物を取り入れて
作っていきました
画面内に表示されている
アルファベットは小文字
の「d」です





Stage5

Stage5では背景の色を
色で塗りつぶしました

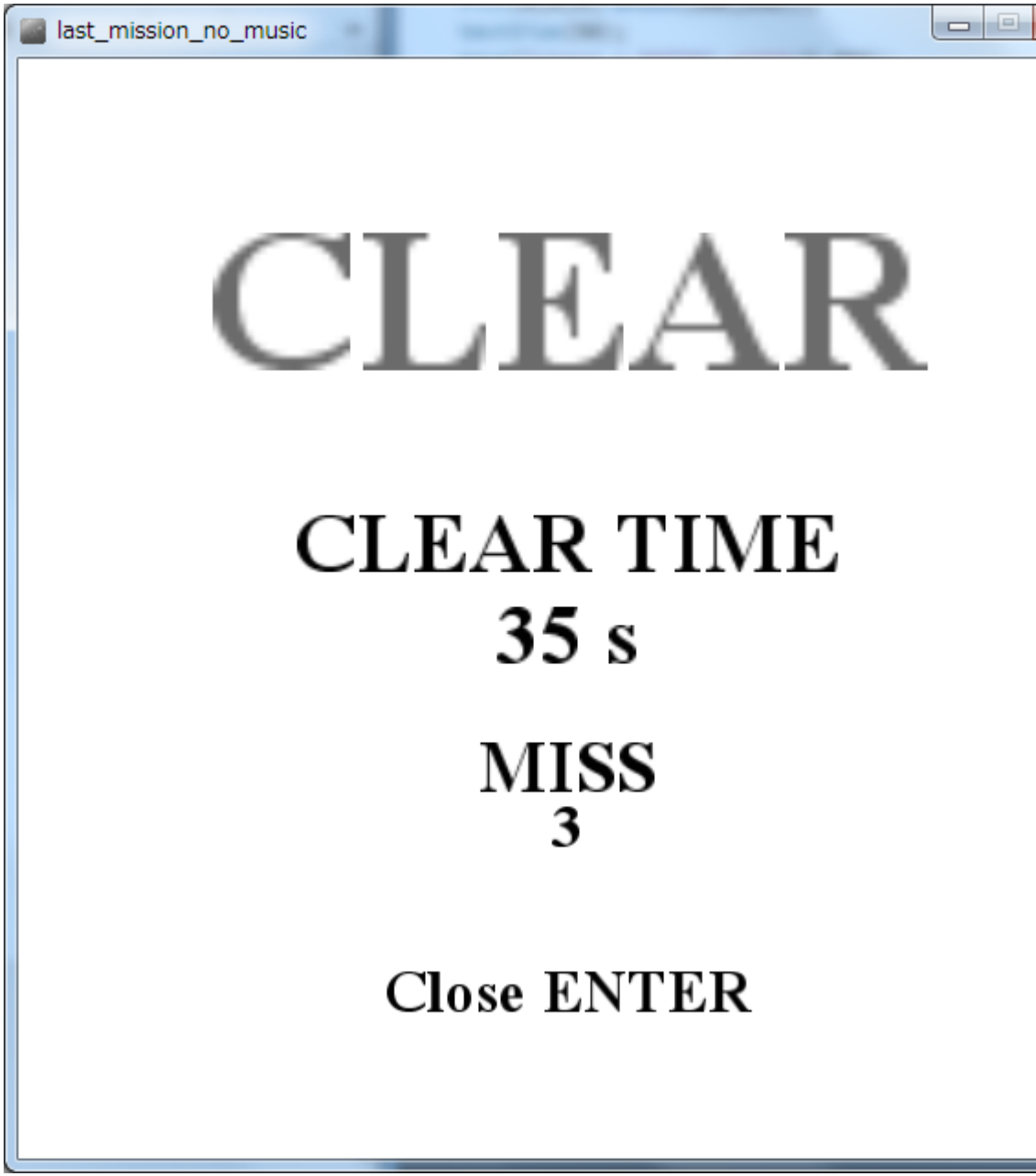


Stage5

Stage5の障害物は、
徐々にそのサイズを大き
くしていくものです
Stage5で表示されたアル
ファベットは小文字の「e
でした

lear

ear画面
こではクリアーまでに
かったトータルタイムと、
分が何回ミスをしたの
というものを表示でき
ようにしました



最後に・・・

- 今回このパワーポイントで紹介させていただいたものは「紹介用」であり、実際のものとは多少の違い（アルファベット、表示位置など）があります。
- 紹介を見て少しでも興味がわいた方は、実際にプレイしてみてください。

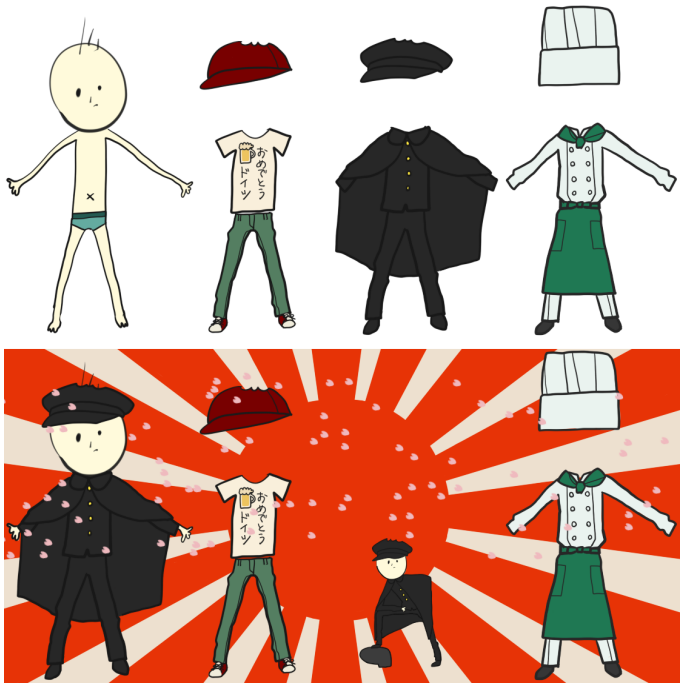
製作するに当たって

- 視覚的にも聴覚的にも楽しいもの。
- 自分が作っていて楽しいもの。

プログラム

- 特定の範囲をマウスプレスすることで音楽を再生する。

- また、音楽が再生されている間、視覚的にちょっとした動きを加えている。



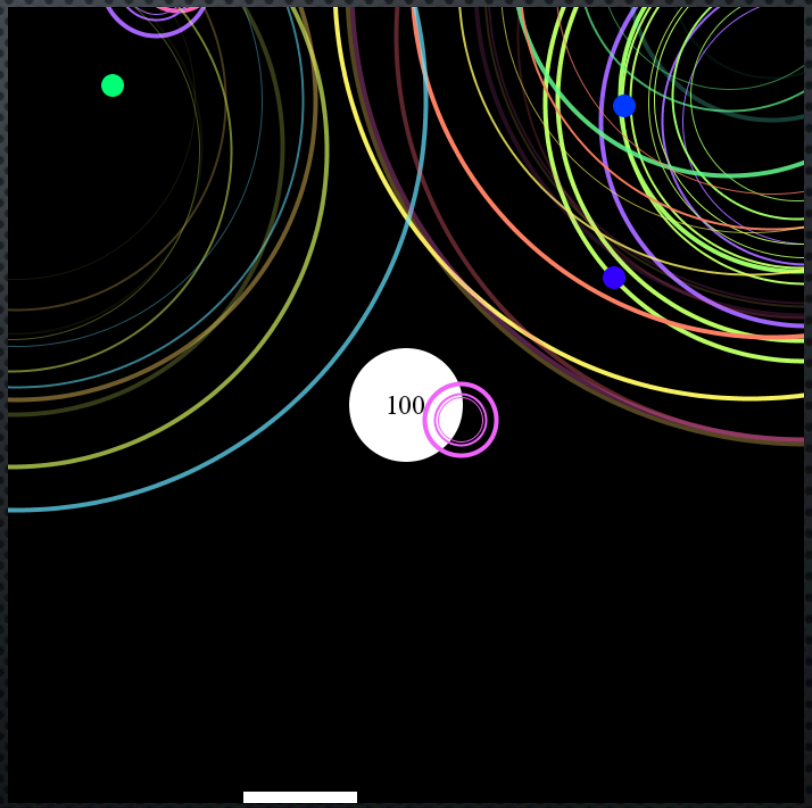
情報メディア基盤ユニット
最終課題

1323197

安澤輝

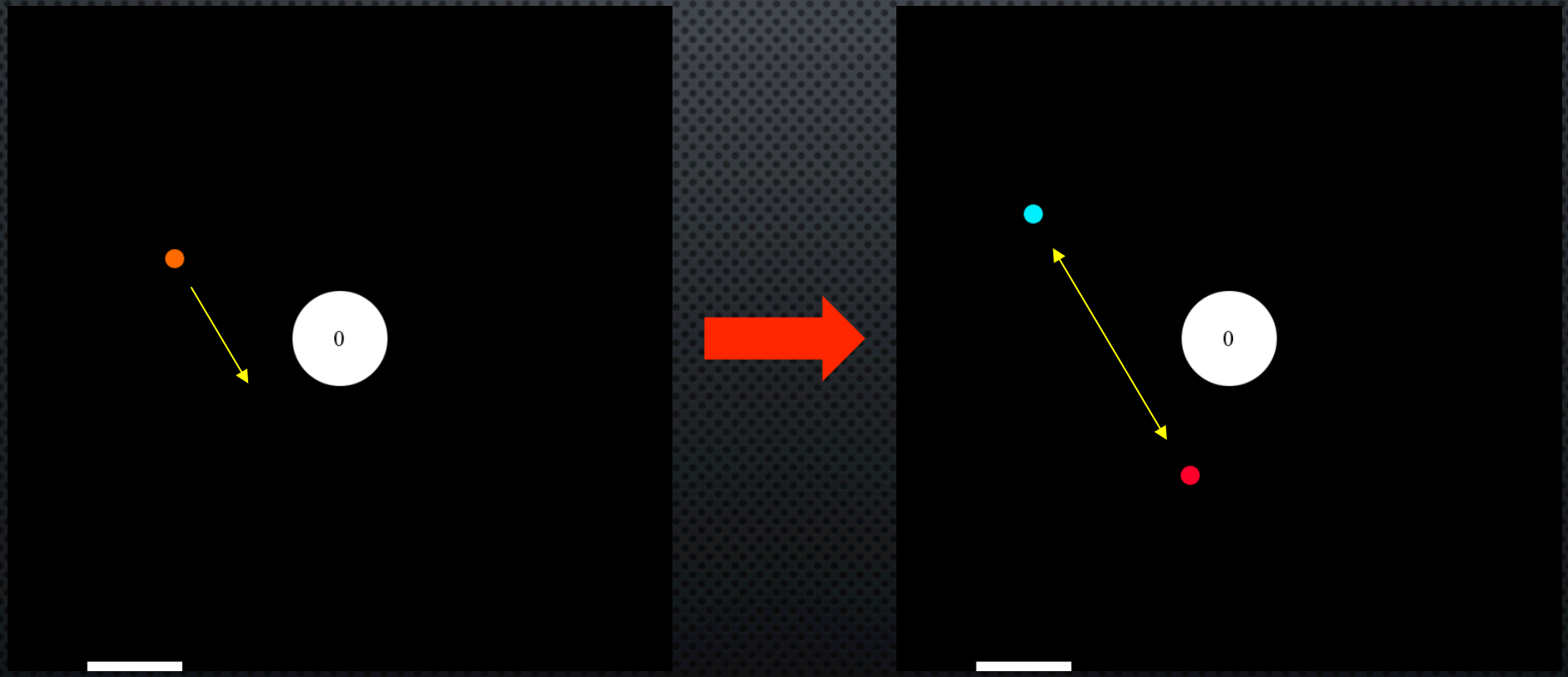
作品紹介

ボールを落とさないようにしながら、中央の円にボールを当てて、得点を稼ぐミニゲーム。プレイヤーはボールの分裂と画面下の板の操作することができる。



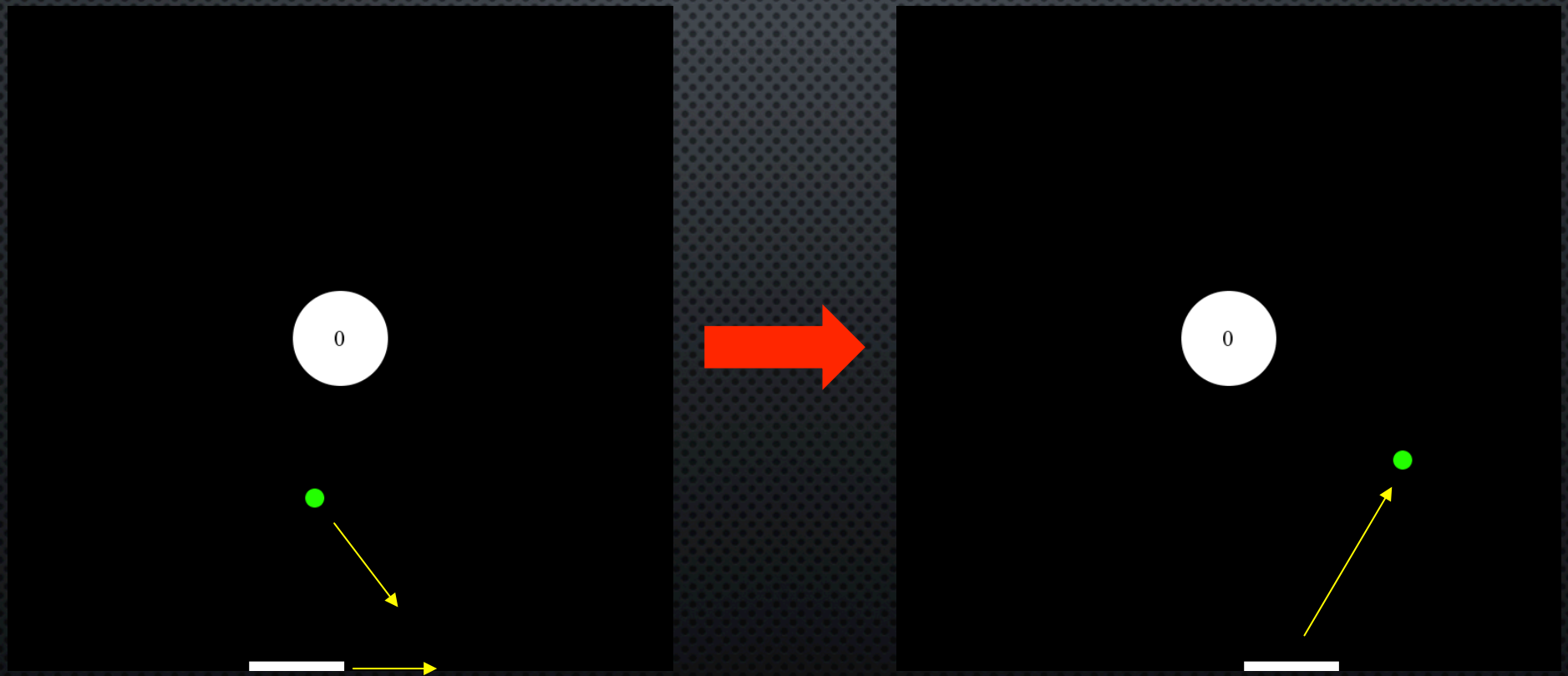
ルール詳細①（プレイヤーの操作）

- ボールを分裂させる（SPACEキー）



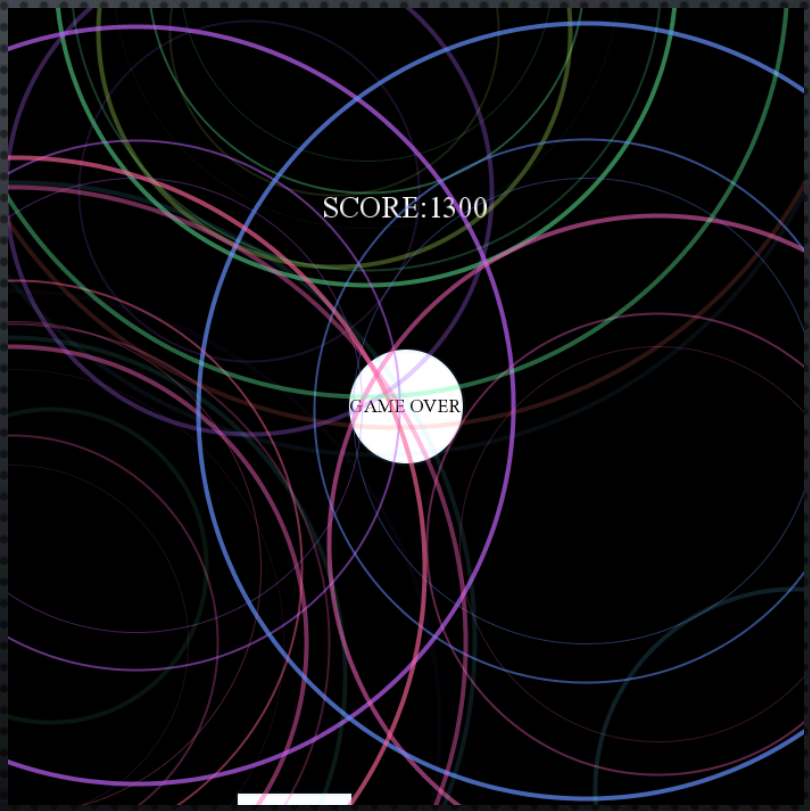
ルール詳細①（プレイヤーの操作）

- ボールを打ち返す（下のバーを移動させる）



ルール詳細②（ゲームオーバーについて）

- ☹ すべてのボールが画面から消えた場合
- ☹ プレイ時間に対して一定以上得点できなかった場合



参考文献

[1] MYY*MAKE*BLOG / PROCESSINGによる波紋アニメーション

HTTP://MYY.GITHUB.IO/BLOG/2013/05/20/RIPPLE-ANIMATION/

[2]無料効果音

HTTP://TAIRA-KOMORI.JPN.ORG/FREESOUND.HTML

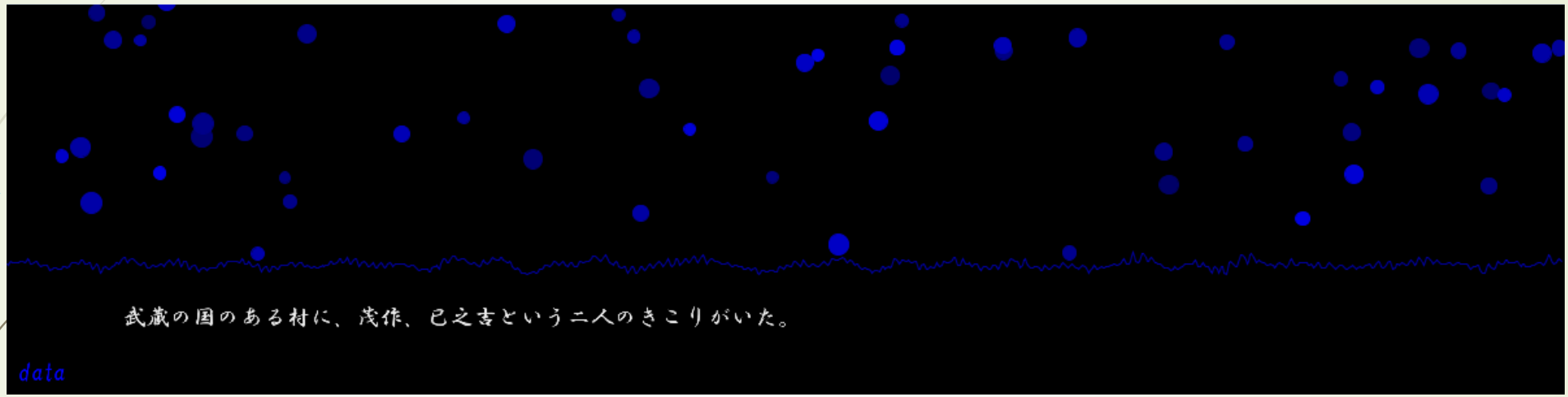
目的

- 今までに習ったことをなるべく全て使い、プログラムを完成させる
- 波形データの表示を利用して、面白いものを作る

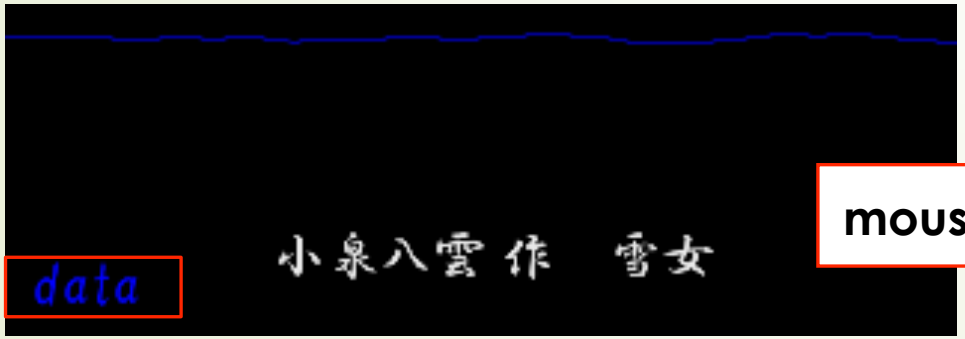
動作内容

- 「雪女」のラジオドラマを再生・波形データを表示
- 「雪女」の再生にほぼ合うように、data.txtから読み込んだ字幕を表示
- 円のx座標、y座標、半径、移動速度、線色・塗りつぶし色の濃淡は乱数で決定し、ウィンドウ上部から下降させる
- 円が波形データを表示する線のあたりまできたらウィンドウ上部に戻し、再び下降させる
- ウィンドウの左下隅に"data"という文字列を表示させ、これをマウスで押すと、タイトル・出典・再生時間を表示
- 再生終了７秒後に、プログラムの実行を終了する

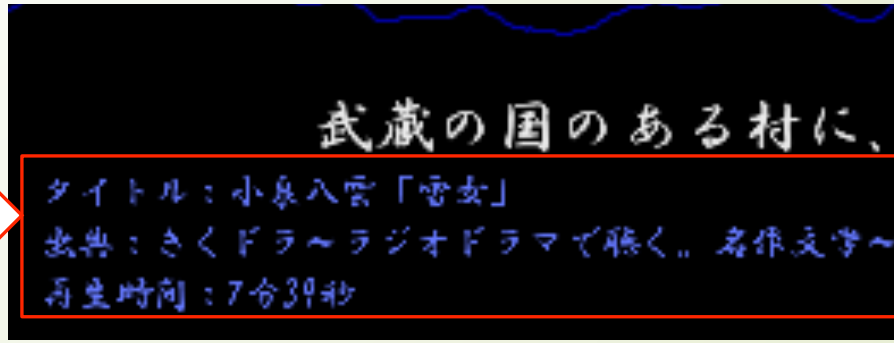
実行画面



円の移動方向　　文字列の移動方向



mousePressed



工夫・努力した点

- 曲や歌などではなく、ラジオドラマを使用した点
- ラジオドラマの字幕を表示した点
- 円の大きさや濃淡、移動速度などを乱数で決め、雪のように見せた点

反省・感想

- 字幕を一行ずつウィンドウに表示する方法しかわからなかったので、もう少し時間をかけて、次々と表示する方法を探し出すべきだった。
- 　座標変換も取り入れられればよかった。
- 見てくれは綺麗なものが作れたので安心した。
- もっと複雑なプログラムを作れるように努力していきたい。

製作物概要

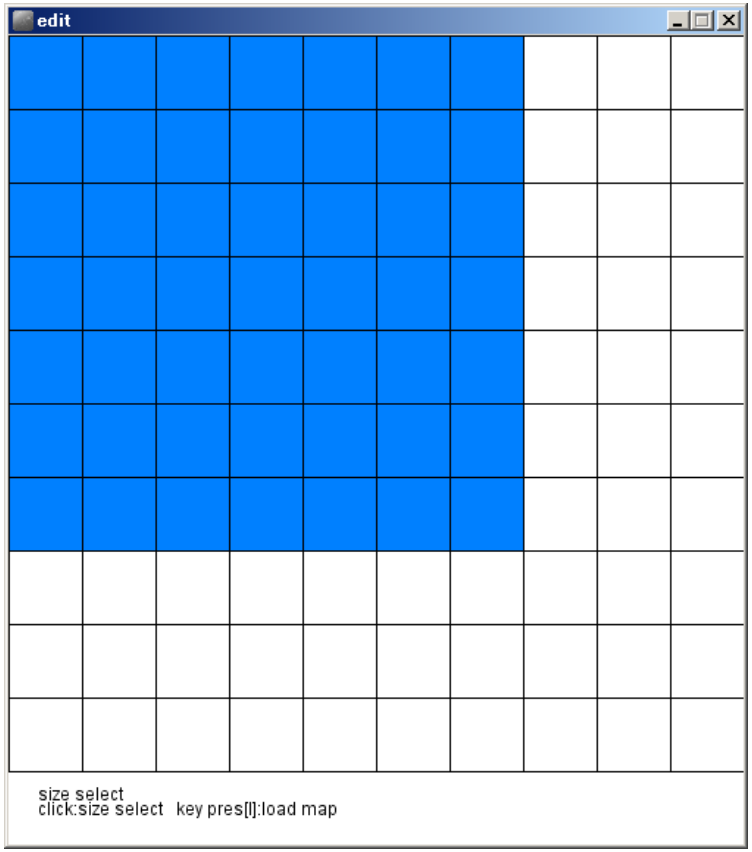
- 今回作ったのは「倉庫番」です。
- 倉庫番のルール
- （１）１個の荷物を押して動かすことができます。
- （２）２個以上の荷物を押して動かすことはできません。
- （３）荷物を引いて動かすことはできません。

（倉庫番オフィシャルサイトhttp://www.sokoban.jp/ より）

- このルールに従って荷物を目標地点に運ぶのが目的です。
- 今回エディットプログラムとゲームプログラムを作りました。

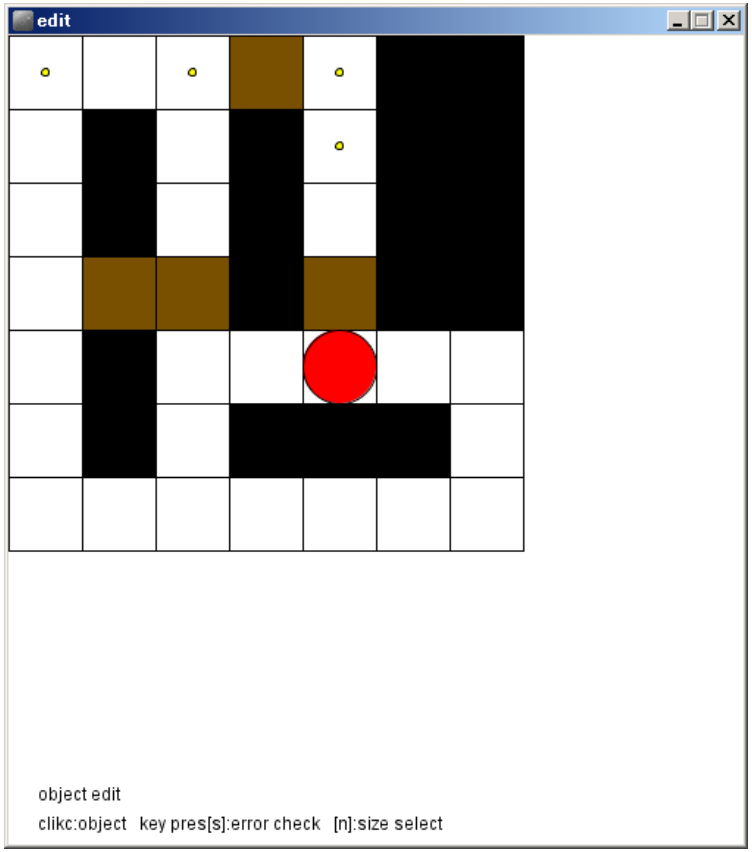
エディットプログラム１

- 起動してはじめは大きさを決定します
- １０×１０の大きさが決められます。
- ファイル内のステージデータをロードすることもできます。



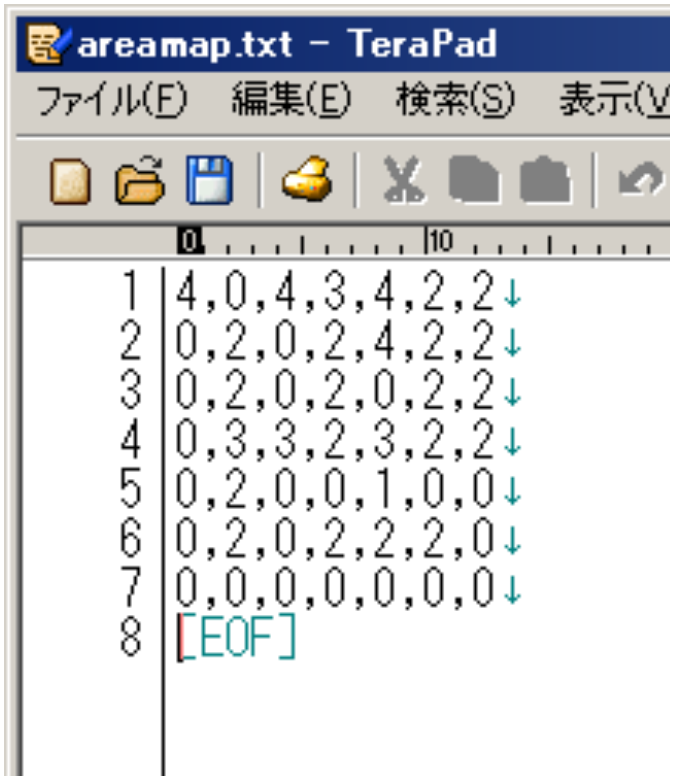
エディットプログラム２

- 大きさを決めたら、物を配置します。
- マス目をクリックするたび配置物が変わります。
- Sキーでエラーチェック・ステージの保存ができます。



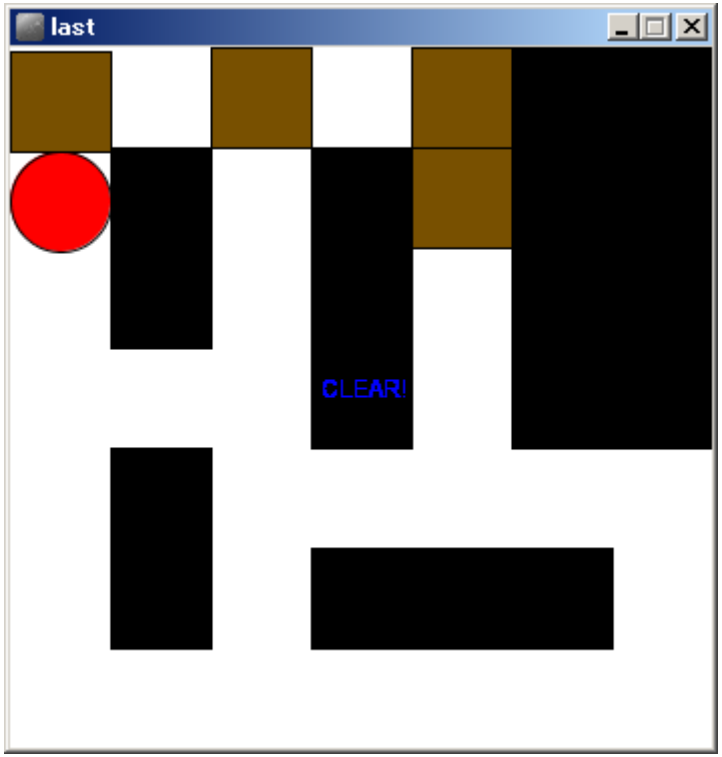
ステージデータ

- データはテキストファイルで保存。
- ０：空白　１：プレイヤー
  ２：壁　３：荷物　４：目標地点
- 「，」と改行で区切ってます。



ゲームプログラム

- ゲームは十字キーで移動、Rキーでステージを初期化します
- エディットプログラムを使わず、自分で書くことで１０×１０より大きいステージも描画されます。
- 目標地点にすべての荷物を運ぶと画面中央に「CREAR!」と表示されます。



引用

- 倉庫番オフィシャルサイト　http://www.sokoban.jp/