令和元年度　プログラミング演習

グループプログラミング　レポート

「ブロック崩しゲーム」

|  |  |
| --- | --- |
| グループ番号 | 3 |
| 1810181 | 鴨田恭佑 |
| 1810220 | 木元颯人 |
| 1810277 | 酒井佑旗 |

1.概要説明

私たちはブロック崩しゲームを作成した。ブロック崩しとは画面上を反射しながら移動するボールを、画面下部に落ちないようにバーを左右に移動して打ち返し、上部に並べられたブロックを破壊していくゲームである。ステージは６種類ある。あらかじめ用意したステージが５つ、ユーザーが自作することのできるステージが１つである。プレイヤーの残基は３で、３回ボールが画面下部に落ちる前にブロックを崩し切ることができればクリアとなる。ゲームプレイ中の操作は非常に簡単でマウスを左右に動かすだけである。ボールを跳ね返しブロックを崩すというシンプルなブロック崩しではゲーム性が欠けるためボールの速度上昇、アイテム、ステージ作成、BGM・SEなど様々な機能を追加した。ボールの速度はゲームのプレイ時間とともに加速していく。この機能により１回のプレイ時間が短縮されるだけでなく、なかなかブロックを消すことができないもどかしい時間を短縮できるようになる。ただ加速していくだけではクリアの難易度が高くなりすぎてしまうため、初期スピードに戻すアイテムなども存在している。アイテムは３種類存在し、それぞれバーの拡張、バレットの発射、スピード・能力の初期化という特徴を持つ。

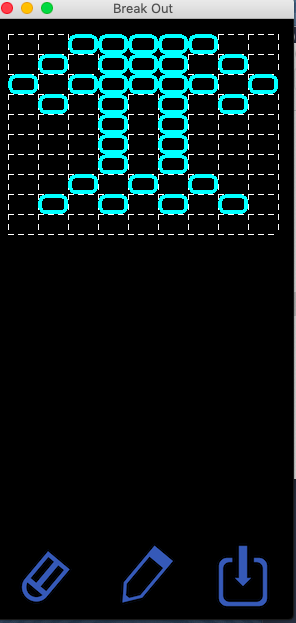
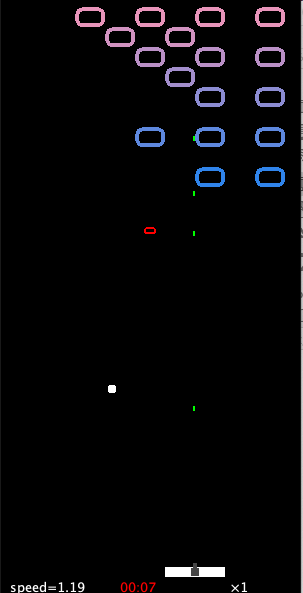
 

図1.1：ステージ制作画面　　　　　　　　図1.2：プレイ画面

　プロジェクト開始当初はMVCモデルに沿って役割を分担したが、完成が近づくにつれて各々が追加したい機能のMVCすべてを担当するようになっていった。そのため、一つのクラスを複数人が編集することも多くあった。複数人が同時に編集して、同時に同箇所の編集をしたりすることのないようにLINEなどのSNSを用いて各々の作業の進捗度合いなどを共有するようにした。鴨田が主に担当したのは画面上部のブロックと画面遷移など、木元が主に担当したのはボールの跳ね返り規則などを管理するクラスとSEやBGMなどの音楽の実装、酒井はアイテム、プレイヤー、画面遷移、統合などを担当した。ファイルなどはDropboxを用いて共有した。

文責:酒井　佑旗

2.設計方針

2-1 全体の流れ

全体の流れとしては、main関数から様々な関数を呼び出し、画面が関数によって遷移していく。ゲーム画面の部分にのみMVCモデルを用いたが、ViewとControllerは一つになっている。画面遷移は画面内のボタンが押されることにより対応する関数が呼び出される。View画面は全てJPanelを継承して作られている。

　以下、それぞれの流れの設計を画面遷移にしたがって説明する。

2-2 タイトル画面

タイトル画面はGridLayoutにより、PlayとMakeの二つのボタンを配置し、背景はPowerPointで作ったものをpngファイルに変換してJLabelを用いて貼り付けた。

2-3 ステージ選択画面

タイトル画面からPlayボタンを押されるとこの画面に遷移する。画面上部にはJPanelで用いられたボタンが6つ配置されており、そのボタンはステージの画像をpngファイルに保存し、その画像がボタンになるようにした。ボタンを押すことで画像に対応したステージのゲーム画面に遷移する。右下にあるボタンはステージmake画面で生成したステージをそのまま映し出して反映させている。

　画面下部はゲーム画面で出てくるアイテムの説明が書かれたpngファイルがJLabelを用いて貼り付けられている。

2-4 ステージmake画面

タイトル画面からMakeボタンを押されるとこの画面に遷移する。画面上部にはステージのブロックを配置できる場所が点線で記されていて、画面下部には3種類のアイコンがボタンとして配置されている。左からブロックを配置するボタン、ブロックを削除するボタン、ステージを保存しステージ選択画面に遷移するボタンである。この機能を実装するために他のviewでは使っていないMouseListenerを使用して、クリックされたかされていないかを記録している。

2-5 ゲーム画面

ゲーム画面の部分ではMVCモデルを用いており、またBlockクラス、Itemクラス、Playerクラス、CourtModelクラス、そしてGameViewクラスから成り立っている。ここではコート、ブロック、プレイヤー、アイテムの実装について順に説明する。

2-5-1 コートについて

コートとはボールが動く範囲であり、動ける範囲はView画面と同じ範囲で動けるようにした。ボールの実装もCourtModelクラス内で実装されている。ボールがコートの壁に当たったり、ブロックに当たると挙動が変化するので、その実装もこのクラスで行っている。

またボールは時間と共に速度が速くなっていく実装を行った。また、コートの下の画面に当たるとボールが初期位置に戻る。その回数が3回目になるとGame Over画面に移り、ブロックを全て消すことができるとGame Clear画面へと遷移する。

2-5-2 ブロックについて

　ブロックの実装はブロック一つのオブジェクトを生成するBlockクラスと実際にゲーム画面に配置されるブロックのまとまりをCourtModelとGemeViewに送るBlockModelクラスを用いて行った。BlockModelクラスで定義されているブロックのまとまりは配列で表していて、0ならブロックなし、1ならブロックありと表現した。0と1で表現することでわかりやすくなおかつ簡単に実装することができた。ブロック一つ一つに当たり判定があり、当たり判定にボールが一致すると対応する添字の配列の中身を0とし、またランダムでアイテムを生成するようにさせた。ブロックもMVCモデルを用いている。これはBlockは他のクラスでもたくさん使用されているので、独立していた方が利用しやすいと考えた。

2-5-3 プレイヤーについて

　プレイヤーは特別な工夫はないが、Playerクラスにもボールとの当たり判定に加えて、アイテムとの当たり判定が存在する。赤いアイテムと当たり判定が一致するとItemファイルで定義されたBalletクラスの機能が実装され、時間経過とともに元に戻る。緑のアイテムと当たり判定が一致すると、Playerクラスで定義されたPlayerの長さが変更されて一定時間長くなる。白いアイテムの場合はボールの速度を元に戻す物である。

2-5-4 アイテムについて

　2-5-3で述べたようにアイテムはこのゲームでは3種類ある。白と緑のアイテムはPlayerクラスで定義されたフィールドの値を変更するだけなので説明は省く。赤いアイテムを取ったときにBalletクラスで定義された機能が実装される。弾の実装はボールの当たり判定と同じにし、軌道は直線、ブロックと当たったら消えるようにしたものである。これにより再度関数を定義することなく機能を実装することができる。

2-6 終了画面

　終了画面にはGame Over画面とGame Clear画面がある。どちらもタイトル画面に戻るTitleボタンが実装されていて、他の実装の仕方は全てタイトル画面と同じである。

2-7 音について

　このゲームには音楽が実装されている。使用している音楽は2つあり、BGMとボールとブロックが当たった時に発生するSEである。音楽の実装はJavafxのAudioClipという機能を使い実装した。この機能は実装が楽でなおかつ音量を調整することができ、加えてmp3ファイルを使用することができる。BGMに関しては曲が終わったら再び関数を呼び出し常に音が流れる状態にした。SEは1秒に満たない音源であるため、ブロックに当たる時に呼び出される関数が呼び出されたタイミングで音を鳴らした。

main

game

View

・TitleView

・StageMakeView

・StageSelectView

・GameView

・FinishView

CourtModel

Player

Item

・Item

・Ballet

Block

・Block

・BlockModel

図2.1:クラス図

文責：木元　颯人

3.プログラムの説明

　ここでは、それぞれ担当した部分をそれぞれ説明する。

　3.1　酒井担当部分

3.1.1 Itemクラス

今回作成したブロック崩しにはいくつかのアイテムが存在する。その大元となるのがitemクラスである。Itemクラスが保有する変数はtype,size\_x,size\_y,x,yである。typeはアイテムのタイプを保存する。アイテムの種類は整数で区別される。size\_x,size\_yはアイテムのサイズ

を、x,yはアイテムの座標を保存する。Itemクラスは４つの関数を保有する。それぞれアイテムのを新たに生成する関数Item(int a, int b)、アイテムの移動に関する制御をおこなうmoveDown()、生成されたアイテムのタイプを返すgetType()、アイテムの描画を行うdraw(Graphics g)である。アイテムの種類はItem(int a, int b)で乱数を用いて決定される。それぞれのアイテムの挙動はItemクラスではなく、CourtModel、GameViewで定義される。

3.1.2 Balletクラス

いくつかあるアイテムのうちの１つであるバレットをプレイヤーが取得した場合に発射することができる弾を定義している。与えられた座標(a,b)に新たな弾を生成するBallet(int a, int b)、弾の移動を管理するmoveUp()、弾を描画するdraw(Graphics g)の３つの関数を保有する。

3.1.3 StageMakeViewクラス

　ステージメイキングに関するすべてを行っている。プログラムの流れを説明していく。まず、ユーザーが作成したステージ情報を保存するテキストデータ「uset.txt」を読み込む。「user.txt」はブロックの情報が０と１であらわされているcsvファイルである。読み込んだ情報をもとにブロックの情報を保存するblock\_listを作成する。

　画面の下部にはアイコンが「消しゴム」「鉛筆」「保存」の３種類ある。「消しゴム」と「鉛筆」は記述モード除去モードの切り替えを行う。それぞれのアイコンは自分で作成し、画像を描画させている。

アイコンがクリックされた時の処理は以下のプログラムで行っている。クラスにActionListenerを設置しているのでボタンがクリックされたらボタンの種類に合わせた処理を行う。「鉛筆」と「消しゴム」がクリックされたら変数typeを更新する。値は０と１の２種類で０が記述モード、１が除去モードを表す。初期値は０となっている。保存ボタンがクリックされたらブロックの情報を「user.txt」に保存し、次にスクリーンショットをプレビュー用に撮影し「userpng.jpg」の名前で保存している。

1  public void actionPerformed(ActionEvent e){

2     int i = 0;

3     if(e.getActionCommand() !=  null){

4       String cmd = e.getActionCommand();

5       if(cmd.equals("save")){

6         type = 3;

7         Point point = new Point(this.getLocationOnScreen());

8         Rectangle rect = new Rectangle(point.x+13 ,point.y+12, 274, 204);

9

10        this.repaint();

11

12        System.out.println("saved");

13        try{

14          File file = new File("user.txt");

15          FileWriter filewriter = new FileWriter(file);

16

17          for(Block b: block\_list){

18            i++;

19            filewriter.write("" + b.getCount());

20            if(i%9==0) filewriter.write("\n");

21          }

22          filewriter.close();

23        }catch(IOException ex){

24          System.out.println(ex);

25        }

26        try{

27          Robot r = new Robot();

28          BufferedImage img = r.createScreenCapture(rect);

29          ImageIO.write(img,"jpg",userpng);

30

31        } catch(Exception exo){

32        }

33        clicked = true;

34      }

35      if(cmd.equals("pen")){

36        type = 0;

37      }

38      if(cmd.equals("eraser")){

39        type = 1;

40      }

41    }

42  }

以下はブロックがされたときの処理を行うプログラムである。ブロックの表示・非表示は0と1で表現している。0が非表示、1が表示である。ある場所がクリックされたらその座標を変数x,yに保存する。その後、どのブロックの領域に属しているかを調べ、その時のモードに合わせてブロックの状態を更新している。

1 public void mouseClicked(MouseEvent e){

2     Point pt = e.getPoint();

3     int x = e.getX();

4     int y = e.getY();

5      // どのブロックがクリックされたか調べる。

6      // クリックされたブロックが見つかったらそのcountを1にする

7      for(Block b: block\_list){

8       if(b.x < x && x < b.x+b.width && b.y < y && y < b.y+b.height){

9          System.out.println("blockfound");

10         if(type == 0){

11          b.setCount(1);

12        }else if(type == 1){

13          b.setCount(0);

14        }

15        break;

16      }

17     repaint();

18    }

29  }

3.1.4 Playerクラス

　ゲームプレイ画面下部のユーザーが操作するバーに関するクラスである。バーの大きさ、位置、移動範囲、タイプなどの変数を持つ。タイプはアイテムの取得状況を表す。０は通常、１がロング、２がバレットである。主な関数にはバーにボールがあたっているか判定するcheckHit(double ball\_x, double ball\_y)、move(int mouse\_x)、setType(int x)、draw(Graphics g)などがある。それぞれについて説明していく。

　まずはcheckHit()。以下がソースコードである。ボールの座標を引数とし、バーの座標・大きさを考慮しながら当たり判定を行っている。

1 public boolean checkHit(double ball\_x,double ball\_y){

2     if (ball\_x>=x && ball\_x<x+width &&

3         ball\_y>=y && ball\_y<y+height) return true;

4     return false;

5   }

　次にmove()。整数を引数とし、その整数が移動可能な範囲内であればバーをその座標に移動する。

　次にsetType()。アイテムをゲットした際に変数typeを更新し、バーの長さも更新する。バーの長さによって移動可能な範囲が変化するのでそれに気を付けながらサイズを変化させている。

1 public void settype(int x){

2     this.type = x;

3     // typeが変化したらwidthを更新

4     if(x==0){

5       this.width = 60;

6       this.bounds\_x2 = 242;

7     }else if(x == 1){

8       this.width = 120;

9       this.bounds\_x2 = 183;

10     }else if(x == 2){

11       this.width = 60;

12       this.bounds\_x2 = 242;

13     }

14   }

最後にdraw(Graphics g)。バーの描画を行っている。以下がそのプログラムである。タイプによってバーの見た目が変化するので、タイプごとに処理を変化させている。タイプ０と１の場合は

1 public void draw(Graphics g){

2     if(type!=2) g.fillRect(x,y,width,height);

3     if(type==2){

4       g.fillRect(x,y,width,height);

5       g.setColor(new Color(70,70,70));

6       g.fillRect(x+width/2-4,y+1,8,8);

7       g.fillRect(x+width/2-2,y-4,4,8);

8       g.setColor(Color.WHITE);

9     }

10 }

文責:酒井　佑旗

　3.2 木元担当部分

3.2.1CourtModelクラスについて

3.2.1.1主要なフィールド(15-31行目)

　主な役割は付録2に対応するコメントが書かれている。ここではBlockModel、Player、Blockクラスがフィールドとして設定されているがこれはボールがブロックやプレイヤーに当たるとボールの挙動が変わるため、これらの情報が必要になってくる。

3.2.1.2関数 checkHit,checkHitBlock(62-90行目);

　本授業HPにある通信型テニスプログラムのコードをひな形として利用した。ここではボールの座標を引数として受け取り、その座標がPlayerや壁、ブロックの座標に一致しているかを判断している。また、Playerの右部分と左部分のように当たる場所でボールの挙動が変わってくるのでその場合に応じて異なる変数を返すようにした。そのため、変数の種類は壁の上下左右、ブロックの上下左右、プレイヤーの上辺左右、下辺左右、何にも当たらないときの13種類の変数が存在する。

3.2.1.3関数 moveball(109-221行目)

　この関数はchecnHit、checkHitBlock関数で返された値によってボールの挙動を決め、ボールを動かす役割の関数である。この関数も本授業HPにある通信型テニスプログラムのコードをひな形として利用した。ブロックの上下左右と壁の上下左右ではボールの挙動の変化は一致しているので、switch文で描くことでコードを簡潔にした。ただし、ブロックに当たった場合は当たったブロックをコート上から消去しなければならないため、ボールの変化を行うと同時にBlockModelクラス内のフィールドであるBlockの配列に対応するブロックの配列の要素を0に変更する作業も行なっている。

3.2.1.4関数 setBallSpeedUp、setBallSpeedUndo(105、107行目)

これらの関数はアイテムの取得によってボールに変化をつける関数であり、具体的な作業としてCourtModel内のフィールド値を変更を行なっている。

3.2.2 Viewクラスについて

設計方針のクラス図からわかるとおり、Viewクラスは5種類ある。GameView以外の4つはレイアウトの違いやボタンの代わりに画像やアイコンを読み込んでいる点などで違いが見られるが全て実装方法はほぼ同じである。そのため、ここではGameViewとGameView以外のViewクラスの2つにわけて説明を行う。

3.2.2.1 GameView以外のViewクラスについて

　ViewクラスではJPanelを継承して作られており、ボタンが複数ある場合はGridLayoutを用いてJPanelに貼り付けている。これらのViewクラスには全てボタンが実装されているため、ActionListnerをimplementsしてボタンが押されると画面が遷移するようになっている。ただしStageMakeViewクラスではペンアイコンや消しゴムアイコンをクリックすることでフィールド値の変更を行っているので、MouseListenerもimplementsしている。

3.2.2.2 GameViewクラスについて

　他のviewクラスと違い、ボタンが存在しないviewクラスである。フィールド値の特有のものとして残機数やアイテムの時間、そしてボールの速度を表すものがある。これはJLabelで宣言し、状況に応じて変化していく。また、ImplementsしたものとしてMouseMotionListnerが挙げられる。これは最初にボールの打ち出す方向を決めるとき、画面をクリックすることで発射するようにしたため、必要となった。Viewクラスであるためアイテムがプレイヤーに当たったとき、またアイテムの弾丸やボールがブロックに当たったときは画面からブロックやアイテムが消えるようにしなくてはならないので、他のクラスで定義した当たり判定を判断する関数を呼び出すことで行った。

　Gameが終了する条件は2つあり、ブロックが全て画面上から無くなることと残機数が3から0になることである。前者の場合、FinishクラスのGameClear画面が呼び出され、後者の場合はGameOver画面が呼び出されるが、これはこのクラス内で定義されてるわけではなく、main関数内で行われており、main関数ないではこのフィールド値であるgame\_stateの値によって判断している。

文責:木元　颯人

　3.3 鴨田担当部分

　　3.3.1 Blockクラス

　ブロックの大元となる情報があるクラスである。この関数が保持する変数はint型のcount, x, y, width, heightである。countは0なら非表示、1なら表示とし、後述するBlockModelクラスにおいて用いる。x, y, width, heightはそれぞれブロックの縦、横、幅、高さである。関数は10つである。それぞれ、色のrgbを受け取り新たにブロックを生成するBlock(int r, int g, int b)、縦、横、count、の情報を取得するgetBlockX(), getBlockY(),getCount()、それらの情報をセットするsetBlock(int x, int y), setCount(int count)、ボール、弾との当たり判定をジャッジするcheckhit(double ball\_x, double ball\_y), checkBalletHit(double ballet\_x, double ballet\_y)、ブロックを描くdraw(Graphics g)、StageMakeViewクラスで用いるdrawDotRect(Graphics g, int interval, int i)である。

　　3.3.2 BlockModelクラス

　テキストファイルから0,1の配列を受け取り、ブロックの配列を作るクラスである。変数はint型のx, y, max\_x, max\_y, i, r, g, b、ArryaList<Block>型のblock\_list, stage\_textである。関数は6つである。それぞれ各ステージに応じたブロックを生成するmake\_block(int a)、ブロックの情報を取得するgetBlock(int i)、弾と接触していたらそのブロックの配列の添え字返すcheck(double ballet\_x, double ballet\_y)、与えられた添え字のブロックのcountを0に消去するdelete(int i)、全てのブロックのcountが0ならtrueを返すclear\_check()、ブロックを描いていくdraw(Graphics g)である。以下のプログラムは関数make\_block(int a)である。引数のaは1~6の数字で各数字に応じてステージが構築される。テキストファイルにある0,1の配列を一文字ずつ読み取って、rgb値を変えながら、10\*10マスにブロックを作成していく。

1 public void make\_block(int a){/\*ブロックの生成\*/

2 i=-1;

3 Block block;

4 block\_list=new ArrayList<Block>();

5 String stage;

6 int j=0, k=0, n;

7

8 Path stage\_text\_path = Paths.get("stage1.txt");

9 switch(a){

10 case

11 stage\_text\_path = Paths.get("stage1.txt");

12 break;

13 case 2:

14 stage\_text\_path = Paths.get("stage2.txt");

15 break;

16 case 3:

17 stage\_text\_path = Paths.get("stage3.txt");

18 break;

19 case 4:

20 stage\_text\_path = Paths.get("stage4.txt");

21 break;

22 case 5:

23 stage\_text\_path = Paths.get("stage5.txt");

24 break;

25 case 6:

26 stage\_text\_path = Paths.get("user.txt");

27 break;

28 }

29 try {

30 stage\_text = Files.readAllLines(stage\_text\_path, StandardCharsets.UTF\_8);

31 System.out.println("print court");

32 for(String s : stage\_text) {

33 System.out.println(s);

34 }

35 } catch (IOException e) {

36 System.err.println( e);

37 }

38 r=255;

39 g=150;

40 b=180;

41 max\_y = 0;

42 for(y=10;max\_y<10;y+=20){

43 max\_x = 0;

44 max\_y++;

45 r-=23;

46 g-=2;

47 b+=6;

48 for(x=15;max\_x<9;x+=30){

49 block = new Block(r, g, b);

50 block.setBlock(x,y);

51 block.setCount(Integer.parseInt(stage\_text.get(j).substring(k,k+1),10));

52 block\_list.add(block);

53 max\_x++;

54 k++;

55 }

56 j++;

57 k=0;

58 }

59 }

文責:鴨田　恭佑

4.実行例

4.1 コンパイルした時の実行例

実行場所はファイルの一番最初の部分でなおかつmain.javaファイルをコンパイルし、mainクラスを実行する。するとゲーム画面が表示される



図4.1;コンパイル画面

4.2 タイトル画面でPlayボタンを押した時の実行例

画面がステージ選択画面に遷移しただけである。



図4.2: タイトル画面でPlayボタンを押した後の状態

4.3 ステージを選択した場合

ターミナル上ではステージ番号とブロックの状態が0と1の文字列で表示されている。この出力はステージによって異なり、対応した2進列が表示される。

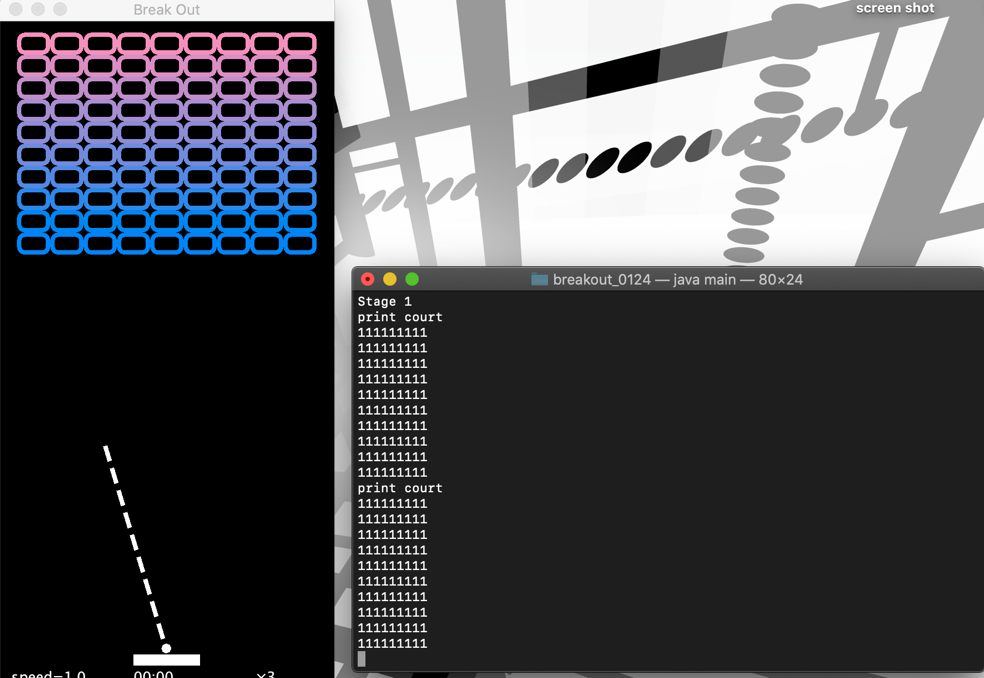


図4.3:ステージ選択画面で左上のボタンを押した後の状態

4.4 アイテムを取得した場合の実行例

　下の図はアイテムを取得して10秒がたった場合である。この時、ターミナル上では　Type1に更新されたと書かれており、残り時間が表示されている。白いアイテムを取得したらType 0、緑のアイテムを取得したらType 1、赤いアイテムを取得したらType 2と表示される。



図4.4:アイテムを取得した後の状態

4.5 ブロックが全て消えた時と残機3つを失ってしまった状態

　タイトル画面と同様に画面が遷移しただけである。



図4.5:ゲームクリアした状態



図4.6:残機を失った状態

4.6 タイトル画面でMakeボタンを押した時の実行例

この画面上で消しゴムのアイコンをクリックした後に画面に水色で表示されているブロックを押すと消え、ペンのアイコンをクリックした後に、点線で囲まれた範囲をクリックすると水色のブロックが現れる。ターミナルには最初に記憶されていたブロックの状態が0と1の2進列で表示される。そしてブロックを消すと0とfra,0と表示され、ブロックを増やすと1とfra,0と表示される。

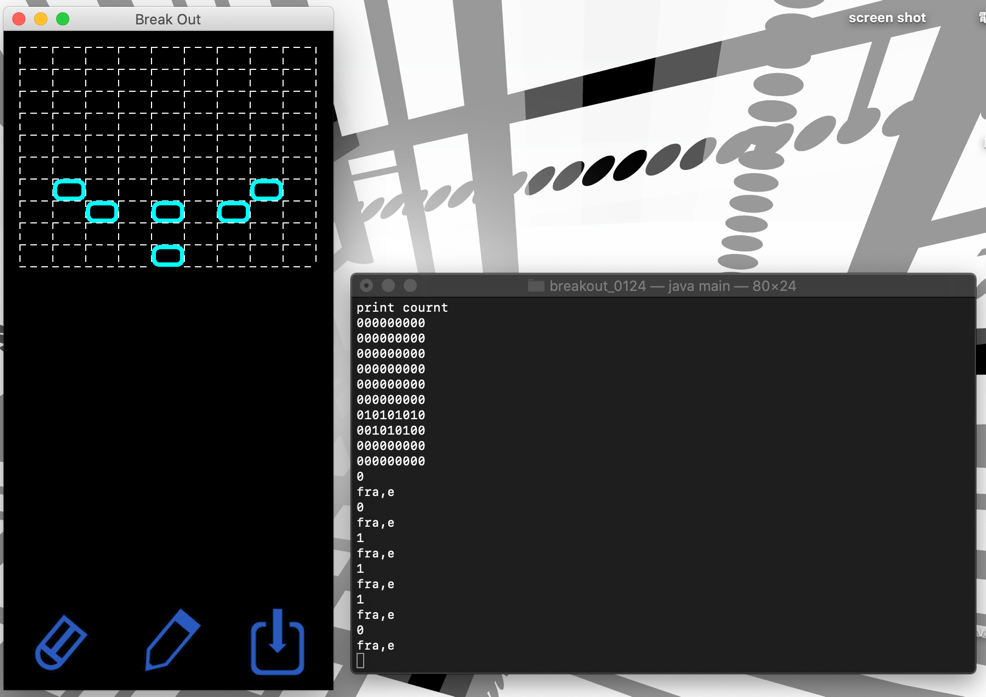


図4.7:make画面で実際に作成している状態

文責：木元　颯人

５．考察

　当初予定していたものは実装することができた。加えてアイテムの実装やステージ制作機能の実装、音楽の実装など追加の機能も実装することができた。このような予定以上の良い成果が得られたが、プログラムの作成にあたって、いくつか問題点が出てきてしまった。プロジェクト開始当初はMVCモデルに乗っ取って役割分担するつもりであったが、次第にMVCモデルから外れて行ってしまった。最終的には各々が実装したい機能についてMVCすべてを担当していた。そのため、同一のファイルを複数人で編集することがあった。コードの書き方が人によって異なったり、１ファイルにいくつものクラスがかかれていたりするため全体的に見づらくなってしまった。作業途中でプログラムが見づらくなってしまっているのを解消するために、1ファイルにつき１クラスのみを記述するようにプログラムを分割し、音楽ファイルやイメージファイルもフォルダにまとめるようにした。このようにファイルを整理することで幾分かは見やすくなったが、CourtModelなどゲームの根幹となる部分は分割することが難しかったためとても見づらい状態のままになってしまっている。今後これを改良するためには、記述が似たようなメソッドや分量が多いメソッドは他クラストして他ファイルに保存することで、ゲームの統括部分の記述量を減らす必要がある。また、プログラムを見やすくするために関数名に規則を設けるなどの方法も考えられる。関数の名付け方にはスネークケースやキャメルケースなどが存在する。グループ間で名付けかたを１種類に統一することでよりプログラムが見やすくなることが期待される。

　ほかの問題点として、OSの違いなどにより同一のファイルでもViewに差異がでてきてしまったことがある。その一つがゲームのプレイ中に流れるBGMやSEなどである。今回のグループでは２人がMacOS、１人がwindowsを使用していた。MacOSでは正常にファイルが読み込まれ再生されたが、Windowsではコンパイルの時点でエラーが出てしまったためBGMなしでしかプレイすることができなかった。もう１つがスクリーンの解像度の差によるViewの差異である。ステージ選択画面ではステージのキャプチャ画像が表示され選択しやすくなっている。ユーザーがステージを作成・編集した場合は保存する際にスクリーンショットを撮り、ステージ選択画面用にサイズ等を加工している。自作ステージはわかりやすいようにステージ制作画面の点線も一緒にキャプチャしているが環境によっては点線が一部欠けてしまう。次回作成するときはプロジェクトの使用環境をあらかじめ想定しておくことで、OSやPCの性能などの理由で差異がでないような機能を考える必要がある。

文責：酒井　佑旗

６．各自の反省と感想

　グループプログラミングは１年生の時の基礎プログラミング演習以来であった。そのときはワンマンチームになり、グループプログラミングとして微妙な感じだった。今回のメディアプログラミング演習でメンバの実力も自分と近くやりたいことなどを共有し、作業分担も信頼して任せることができた。グループで作業するときはできるだけ同程度の実力の人と組むと作業がはかどることを実感した。また自分の担当分ではやろうとしたことは大体実現することができた。アイテムは５つ近く実装したかったが時間的な問題もあり３つしか実装することができなかった。この授業で初めてJavaに触れたため、実装力のすごい人には全然追い付けなかったが、オブジェクト指向の基礎的な部分は知ることができた。授業のプログラミングの部分にはあまり関係ないが、デザイン力の重要性を感じた。他グループの作品で自分でもプログラムが再現できそうなグループでも、イラストなどのデザインによってゲームなどのソフトウェアとしての完成度が段違いに高く感じた。

文責：酒井　佑旗

今回初めて3人でやるグループプログラミングを行った。一人で書く時とちがい、他の人のプログラムを理解していないとプログラムを書くことができないため、他のグループの人に伝わるようにもっとコメントを書くべきだったと感じた。また、この関数やフィールドの値が何を示していてどんな機能を持っているかをプログラムを見なくてもわかるようにわかりやすい変数名や関数名をつける必要性も感じた。他にも、グループの人のプログラムを見ることができるため、この書き方をすることでわかりやすくなったり、自分一人では思いつくことができないような実装の仕方があったのでとても勉強になった。

自分は主にCourtModel内の関数の実装と音楽について行った。CourtModel内の関数の実装はサンプルコードを拡張させて、グループのみんなと相談することでブロックが消えてなおかつボールがきれいに反射する実装をうまく作ることができた。しかし、この実装方法ではボールの速度が運が悪いと無限大に上がっていってしまい、当たり判定がうまく機能しないことがある。加えて、ボールの反射は乱数ではないので、ブロックの配置とあたりどころによってはボールがずっと同じ挙動しか行わないことがあったのでその2つは今後の課題であると感じた。

MVCモデルはわかりやすく実装の分担わけが簡単であるが、グループ内のプログラミングの技量がわかっていないと分担することが難しくなってしまい、自分たちのグループのように分担が混ざってしまうと感じた。MVCモデルで実装を行うときは、最初の計画の段階がとても重要になってくると感じた。

この演習を行うまでは自分はjavaに触れたことがなく、授業の課題も完璧に理解することができなかったが、この演習を終えて自分ができなかった任意課題に取り組んでみると解けるようになっていたので、自分のプログラミングの力がついたことを実感した。また、自分たちが作ったゲームの感想を貰えることが嬉しく、読むことで自分たちでは気付けなかった部分の問題点がわかったので他人に評価してもらうことがとても重要だと感じた。他のグループの人たちが優秀で追いつくことに必死だったがみんなで相談して作ることができたので、とてもやりがいを感じる授業だった。

文責:木元　颯人

　今回初めてグループプログラミングを行った。三人で分担してやるため、自分が負担する部分が少なくなる反面、色々難しい面もあった。まず、担当分担を決めるときである。私はJavaをやるのは今回が初で、他の二人の実力もあまり理解していない状況であったので、最初の話し合いに時間がかかった。次に、共有のやり方である。グループなので、自分だけではなく、他の人がパッと見て理解できるプログラムを書かなければならかった。また、二人が書いたプログラムを理解して、統合したり、機能を追加したりするのはなかなか骨の折れる作業だった。わかりやすい書き方やコメントを多く取り入れることが大事だと感じた。

　やりたいことが多く、各々がそれぞれ作りたいことを作っていたので、当初予定していたMVCモデルから大分外れてしまった。MVCモデルを実装する際には、計画を濃密にし、作りたい実装をメンバーに託し、しっかり共有すべきである。実装したかったことは大体実現することができた。しかし、細かい機能をまだ考えていたが、実装には至らなかった。例えば、スコア機能をつけ、ランキングを作成したり、である。

　この講義でJavaがかなり身についた。また、オブジェクト指向について基礎的な考え方、実装について理解することができた。さらに、グループプログラミングを行ったことで、グループでの作成や発表の難しさ、同じものを作って完成させる楽しさなどを経験することができた。

文責:鴨田　恭佑

付録1：操作法マニュアル

1.ブロック崩しとは

　画面上を反射しながら移動するボールを、画面下部に落ちないように、パドル（バー）を左右に操作して打ち返し、煉瓦上に並べられたブロックを消していくゲーム

（Ja.wikipedia.org から参照）

2.概要

　ステージは六種類。既存のステージ五種類に加え、後述する自作したステージをプレイすることが出来る。残基を3とし、3回下に落ちるとゲームオーバーとなり、それまでにブロックを全て消すことが出来ればゲームクリアとなる。操作は非常に簡単で、カーソルを左右に動かすだけである。

　従来のブロック崩しに加え、様々な機能を追加した。

2.1 ボールの速度上昇

ゲームをスタートさせてから時間が経過する毎にボールのスピードが速くなっていく。一回のプレイ時間がスピーディーになるだけでなく、ブロックが少なくなってきた終盤の、なかなか消せないときのもどかしい時間を短縮することにも繋がっている。ボールが落下したとき、又は後で記載するアイテムを取得することで初期化される。

　2.2 アイテム機能

ブロックを消したときに一定の確率でアイテムが落ちてくる。詳細は4.アイテムに記述する。

　2.3 ステージ作成機能

ユーザーが自分でブロックの配置を設定することができる。9×10マスの好きなところにブロックを置くことができ、実際にそれをプレイすることが可能。

3.ゲーム画面

ゲーム画面の遷移図は以下の図である。

スクリーンショット が含まれている画像

自動的に生成された説明

図1:画面遷移図

　3.1 タイトル画面

スクリーンショット が含まれている画像

自動的に生成された説明

画面下のボタンにより次の画面へ

・「Play」

3.2ステージセレクト画面へ

・「Make」

3.5ステージ作成画面へ

図2:タイトル画面

　3.2 ステージセレクト画面

スクリーンショット が含まれている画像

自動的に生成された説明

ステージの絵をクリックするとそのステージへ

既存のステージ5種類

ステージ作成画面で作成したステージが表示される。

図3:ステージセレクト画面

スクリーンショット が含まれている画像

自動的に生成された説明　3.3 ゲーム画面

ゲームスタート時、ボールが落下して新しく始める時、残基を一つ減らしてボールを発射することができる。そのとき、カーソルを動かすことで、ボールの軌道を変えて好きな方向に発射することができる（図5参照）。また、画面下には左から、現在のボールのスピード、アイテムの制限時間、残基が表示されている。（図6参照）。

ゲームオーバー又はクリアしたらキーボード が含まれている画像

自動的に生成された説明3.4フィニッシュ画面へと遷移する。

キーボード が含まれている画像

自動的に生成された説明

図4:ゲーム画面　　　　図5:ボールの軌道　　　　　　　　図6:画面下部の表示

　3.4 フィニッシュ画面

残基が0でボールが落下したときは画面中央に「GAME OVER」が、ブロックを全て消すことができれば「CLEAR」が表示される（図7及び図8）。画面左下の「Continue」ボタンでもう一度そのゲームを初めからスタートできる。画面右下の「Title」ボタンは文字通りタイトル画面へ戻る。

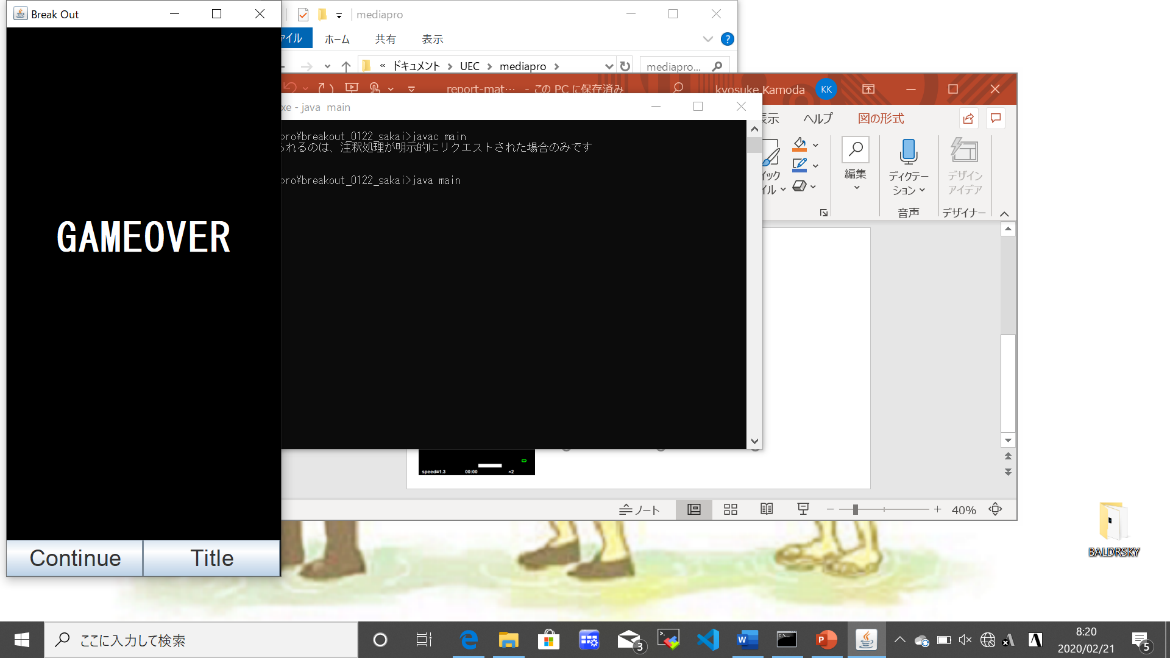
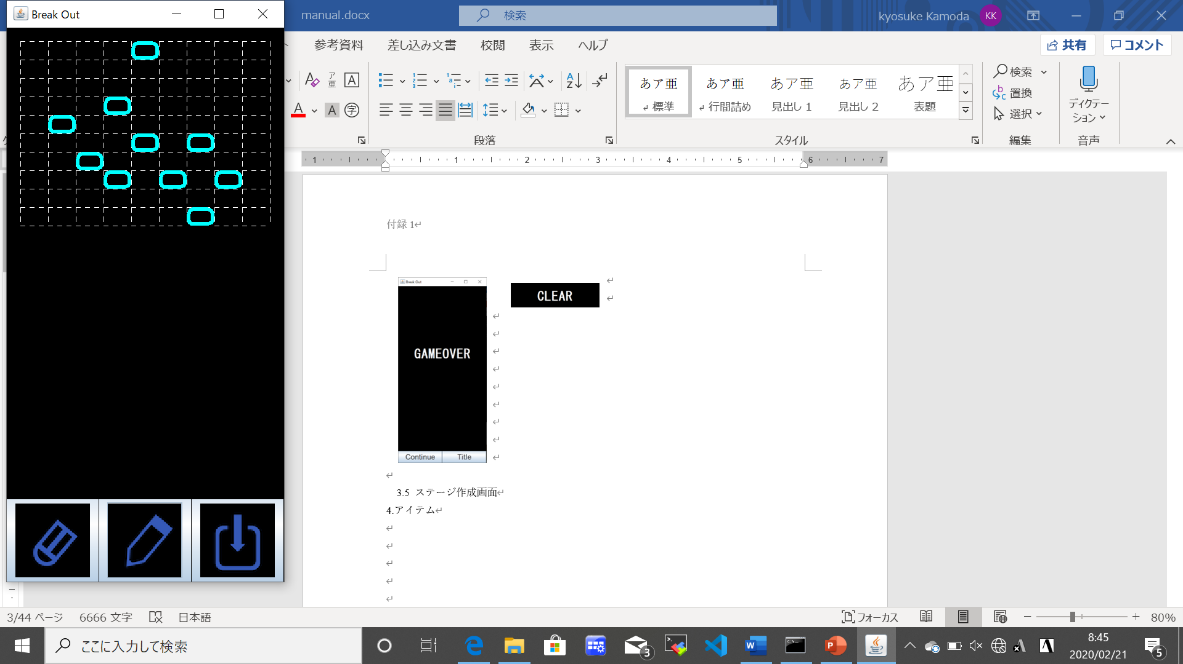
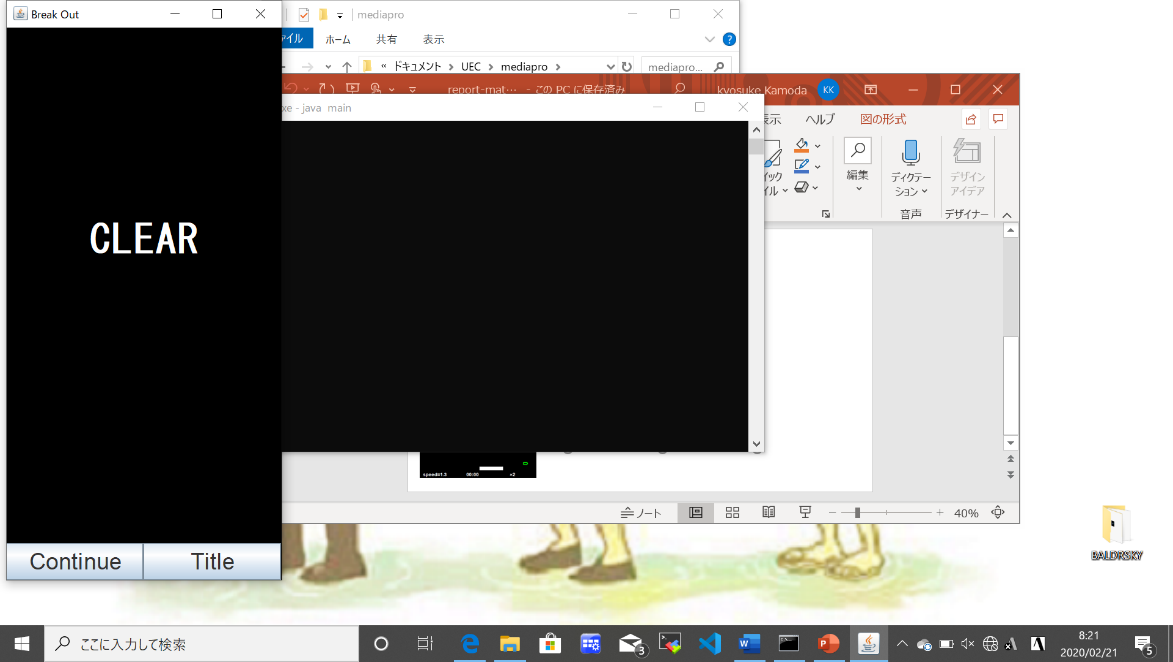


図8:フィニッシュ画面

（クリア時）

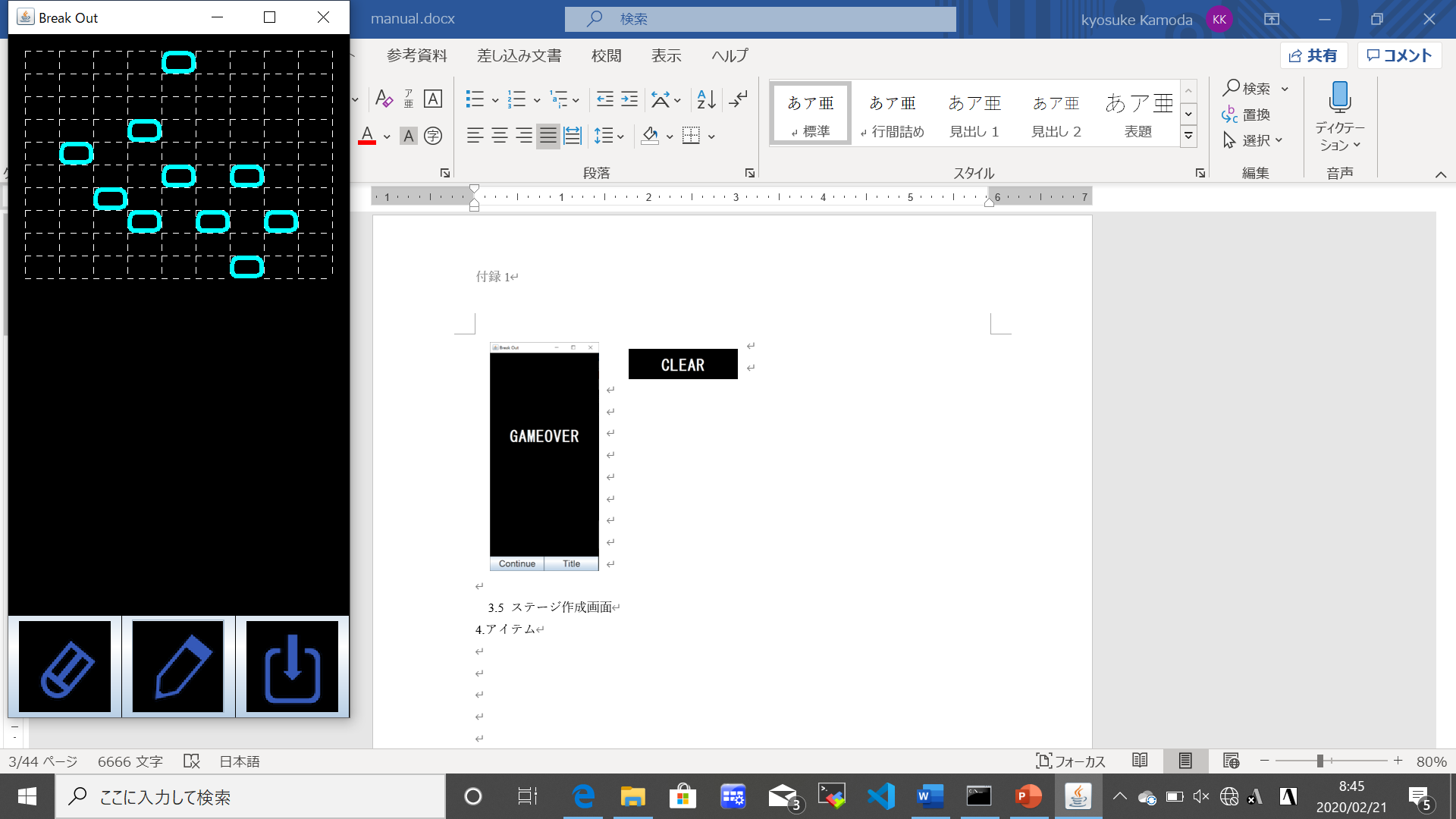


図9:アイコン

　図7:フィニッシュ画面　　　　　　　　　　　　　　　　図10:ステージ作成画面

（ゲームオーバー時）

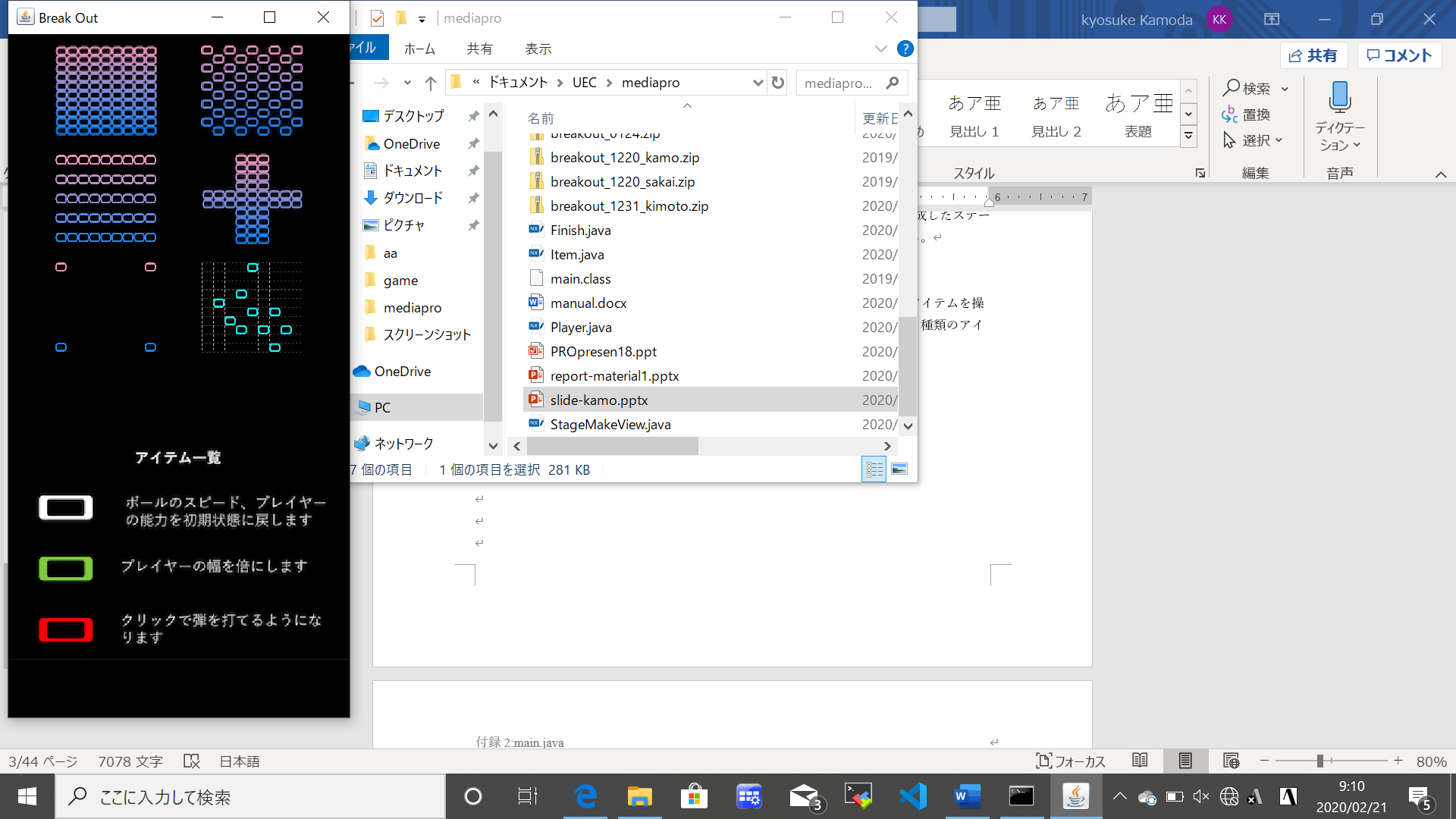
　3.5 ステージ作成画面

ユーザーが自分でステージを作成できる画面（図10）。画面下にアイコンが3つ

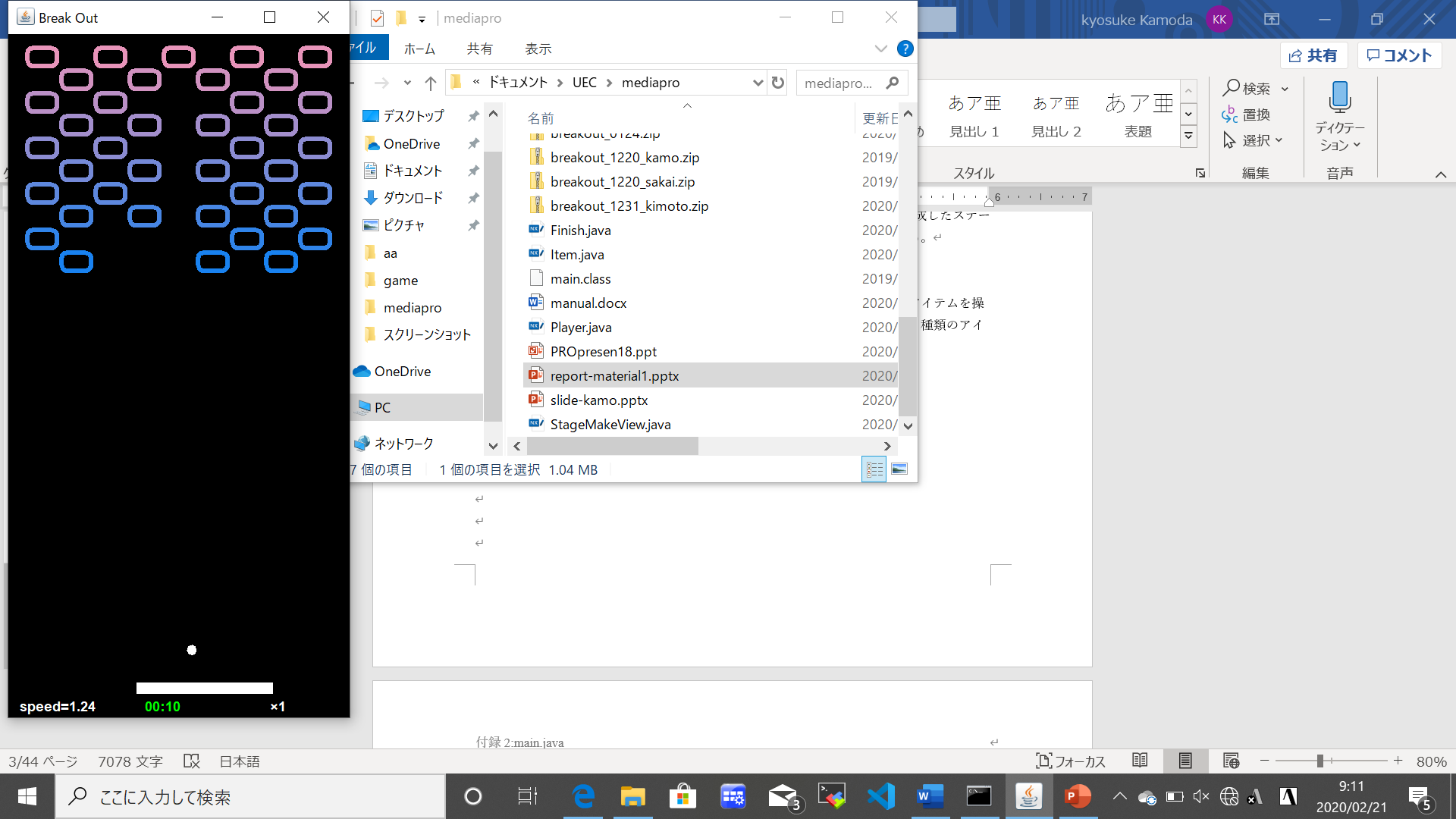
り、それぞれ押すことでそのモードになる（図9）。真ん中のペンのアイコンにより上部の点線の枠の好きなところをクリックするとそこにブロックが置ける。左の消しゴムのアイコンは現在置いているブロックをクリックすることで消去することができる。右のアイコンをクリックするとそのブロックの配置を保存し、タイトル画面へ戻る。作成したステージはステージセレクト画面の中央右にあり、そのステージを遊ぶことができる。

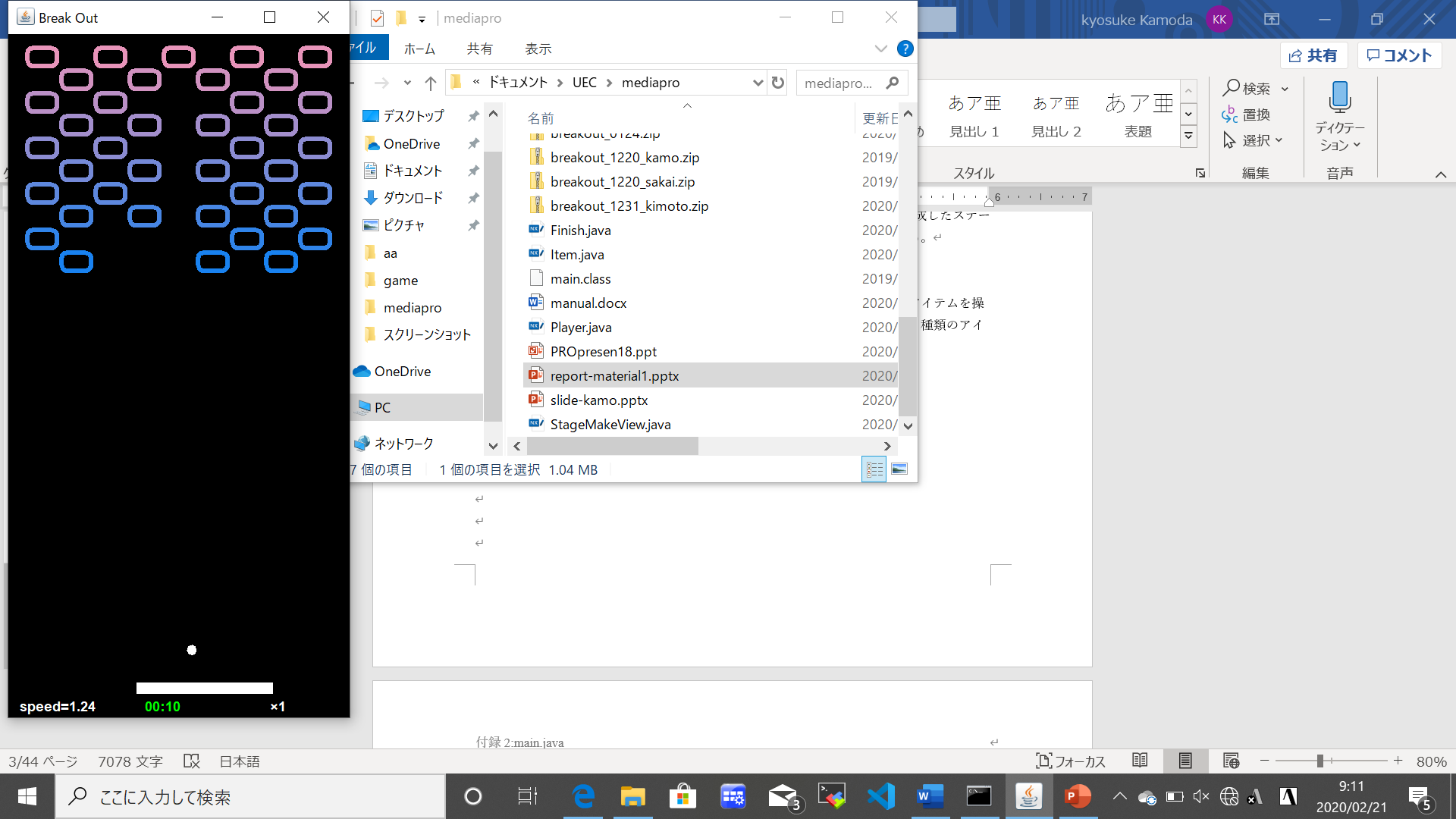
4.アイテム

　ブロックを消したときに一定の確率でアイテムが出現する。落下してきたアイテムを操作しているバーで上手くとると効果が現れる。アイテムの種類は3種類。違う種類のアイテムを取得すると、上書きされる。

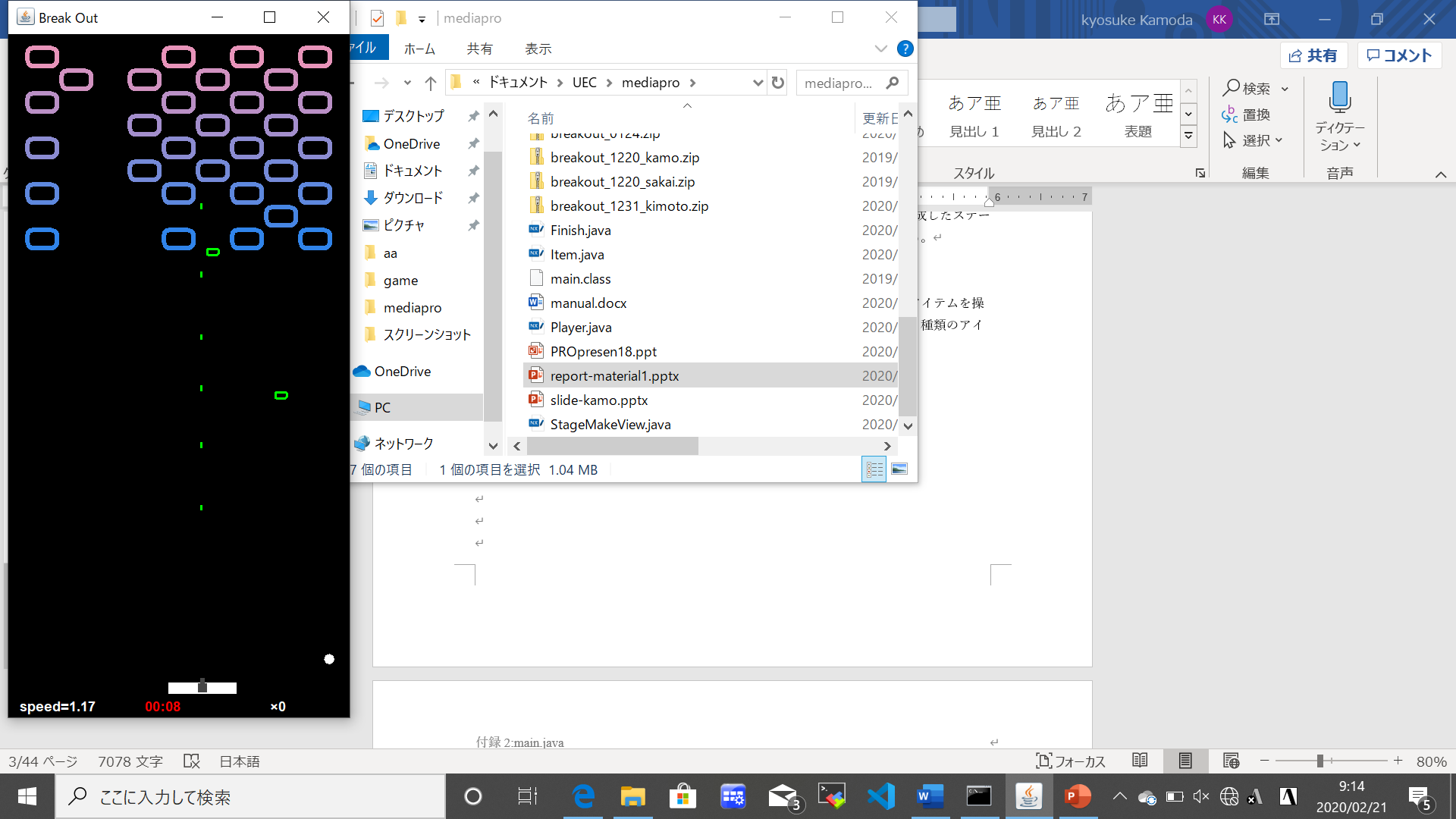
　4.1 バーの拡張

緑色のアイテムを取得すると操作しているバーが伸び　　　　　　　　　　　る。制限時間は10秒。

 図11

スクリーンショット が含まれている画像

自動的に生成された説明 図12 図13

　4.2 バレット機能

赤いアイテムを取得するとクリックすると弾を撃つことができ

る、バレット機能が付与される。この弾でもブロックを消すことができる。制限時間は10秒間。

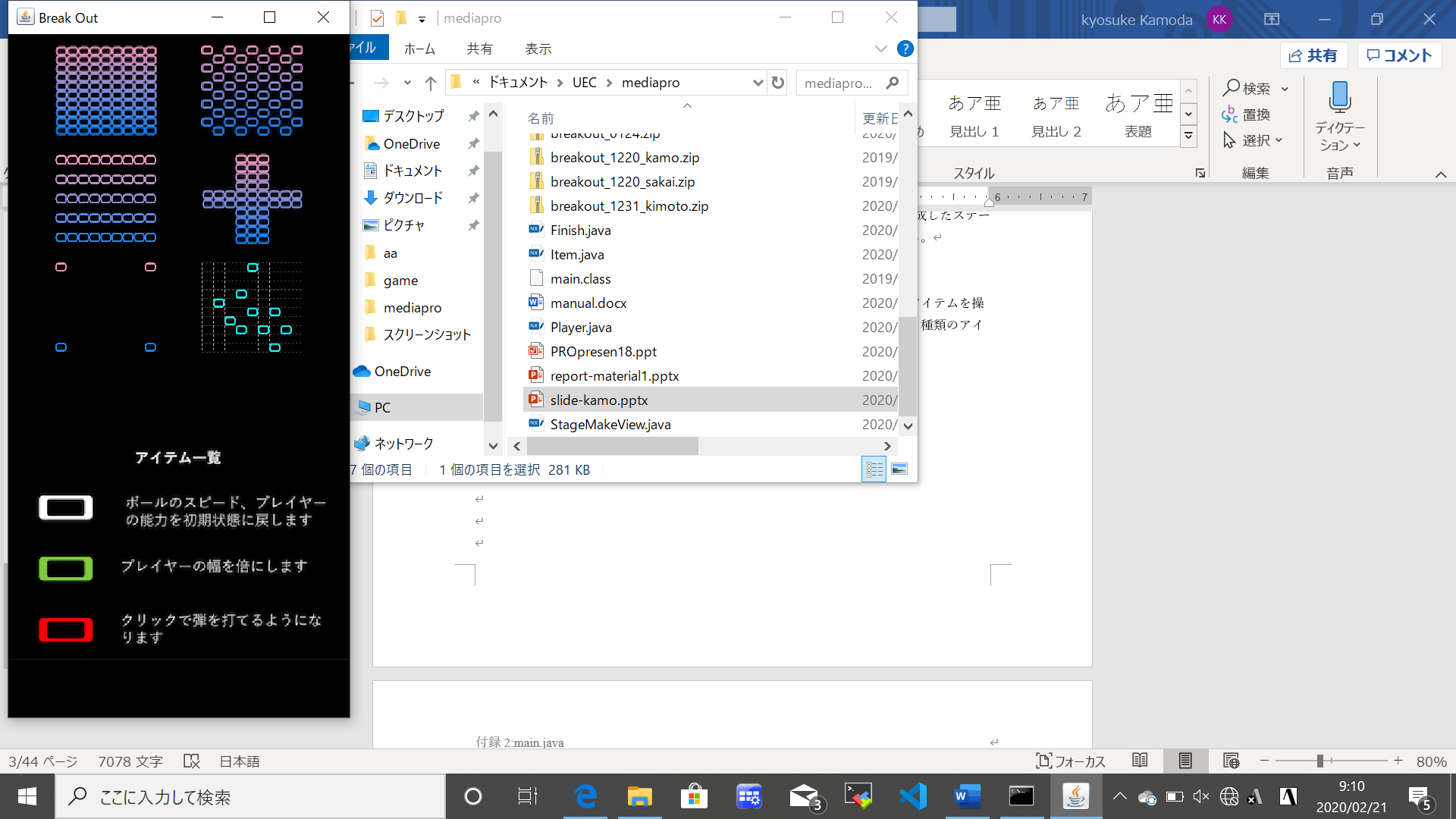
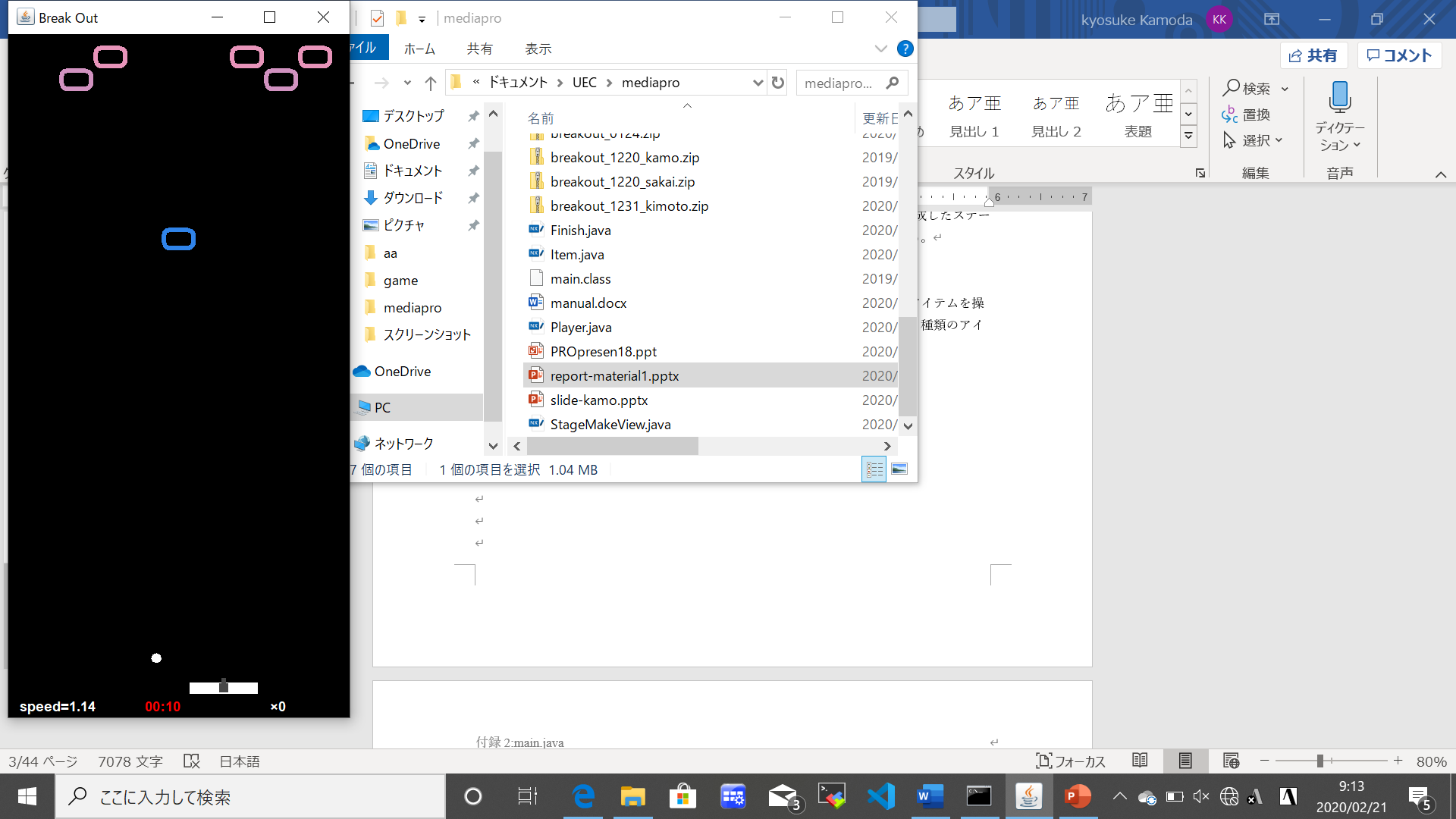


図14 図15

図16

4.3 スピードの初期化

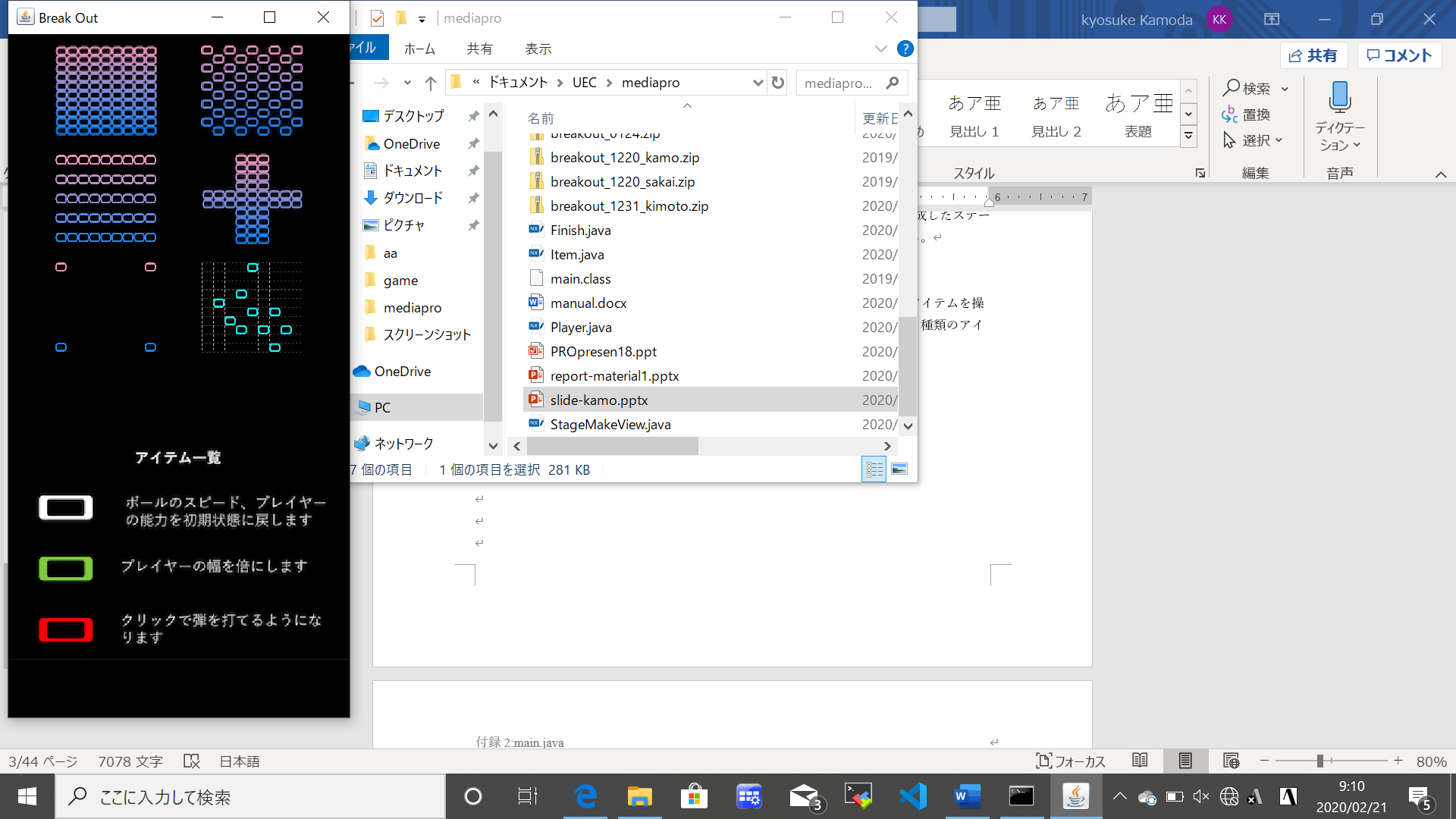
 白いアイテムを取得するとスピードが初期化され、倍率1.0になる。

図17

　アイテムを取捨選択をして、いかに上手く取得できるかが、ゲームクリアの鍵となる

文責：鴨田　恭佑

付録2:プログラムコード

・main.java

1 import java.awt.\*;

2 import java.awt.event.\*;

3 import javax.swing.\*;

4 import game.\*;

5 import game.block.\*;

6 import java.util.\*;

7 import java.io.File;

8 import java.net.MalformedURLException;

9 import javafx.scene.media.AudioClip;

10

11 class main extends JFrame {　//画面の遷移

12 public static void main(String[] args) throws Exception{

13 String str;

14 CourtModel tm;

15 BlockModel bm;

16 AudioClip bgm

17 = new AudioClip(new File("bgm\_maoudamashii\_8bit14.mp3").toURI().toString());

18 bgm.setVolume(0.3);

19 int state=0; // 1:crear, -1:game over

20 int stage\_num=1; //各ステージの番号

21 int nextView = 0; // 0:Title, 1:StageSelect, 2:Game, 3:Finish, 4:StageMake

22 boolean click\_state; //StageMakeから受け取る、true:保存

23 int click\_state\_int; //Title,Finishから受け取る

24 JFrame frame = new JFrame("Break Out");

25 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

26

27 TitleView title = new TitleView();

28

29 while(true){

30 switch(nextView){

31 case 0:

32 {

33 title = new TitleView();

34 //タイトル画面を表示

35

36 frame.add(title, BorderLayout.CENTER);

37 frame.pack();

38 frame.setVisible(true);

39 if(bgm.isPlaying()==false)bgm.play();

40 while(true){

41 if(bgm.isPlaying()==false)bgm.play();

42 click\_state\_int = title.getClicked(); //1:StageSelect, 2:StageMake

43 System.out.printf("");

44 if(click\_state\_int != 0){

45 System.out.printf("");

46 frame.remove(title);

47 break;

48 }

49 }

50 }

51 if(click\_state\_int == 1) nextView = 1; //StageSelect

52 else if(click\_state\_int == 2) nextView = 4; //StageMake

53 break;

54

55 case 1:

56 {

57 StageSelectView stage = new StageSelectView();

58 //ステージセレクト画面を表示

59

60 frame.add(stage, BorderLayout.CENTER);

61 frame.setVisible(true);

62 if(bgm.isPlaying()==false)bgm.play();

63 while(true){

64 if(bgm.isPlaying()==false)bgm.play();

65 stage\_num = stage.getStage();

66 System.out.printf("");

67 if(stage\_num!=0){

68 System.out.println("Stage " + stage\_num);

69 frame.remove(stage);

70

71 break;

72 }

73 }

74 }

75 nextView = 2; //Game

76 break;

77

78 case 2:

79 {

80 tm = new CourtModel(300,600,30,stage\_num);

81 GameView tv = new GameView(tm);

82 tv.setBlockModel(stage\_num);

83 //ゲーム画面を表示

84

85 frame.add(tv, BorderLayout.CENTER);

86 frame.setVisible(true);

87 if(bgm.isPlaying()==false)bgm.play();

88 while(true){

89 if(bgm.isPlaying()==false)bgm.play();

90 state = tv.getState();

91 System.out.printf("");

92 if(state != 0){

93 frame.remove(tv);

94 break;

95 }

96 }

97 }

98 nextView = 3; //Finish

99 break;

100

101 case 3:

102 {

103 Finish finish = new Finish(state);

104 //フィニッシュ画面を表示

105 frame.add(finish);

106 frame.setVisible(true);

107 if(bgm.isPlaying()==false)bgm.play();

108 while(true){

109 if(bgm.isPlaying()==false)bgm.play();

110 click\_state\_int = finish.getClicked(); //1:Game, 2:Title

111 System.out.printf("");

112 if(click\_state\_int != 0){

113 title = new TitleView();

114 frame.remove(finish);

115 break;

116 }

117 }

118 }

119 if(click\_state\_int == 1) nextView = 2; //Game

120 else if(click\_state\_int == 2) nextView = 0; //Title

121 break;

122

123 case 4:

124 {

125 StageMakeView stagemake = new StageMakeView();

126 //ステージ作成画面を表示

127

128 frame.add(stagemake, BorderLayout.CENTER);

129 frame.pack();

130 frame.setVisible(true);

131 if(bgm.isPlaying()==false)bgm.play();

132 while(true){

133 if(bgm.isPlaying()==false)bgm.play();

134 click\_state = stagemake.getClicked();

135 System.out.printf("");

136 if(click\_state){

137 frame.remove(stagemake);

138 break;

139 }

140 }

141 nextView = 0; //Title

142 break;

143 }

144 }

145 }

146 }

147 }

・CourtModel.java

1 package game;

2 import java.awt.\*;

3 import javax.swing.\*;

4 import game.block.\*;

5 import java.io.File;

6 import java.net.MalformedURLException;

7 import javafx.scene.media.AudioClip;

8 import java.util.Random;

9 import java.lang.Math;

10 import java.util.\*;

11 import game.item.\*;

12 import java.awt.event.\*;

13

14 public class CourtModel implements ActionListener,MouseListener{

15 public int court\_size\_x,court\_size\_y; // テニスコートの大きさ

16 public Player myself; // Player 自分と対戦相手

17 public Block b;

18 public BlockModel bm;

19 public StageSelectView stage;

20 private AudioClip se;

21 private double ball\_x,ball\_y; // ballの位置 (斜めに移動するので，doubleにする)

22 private double ball\_radius; // ボールの半径

23 private double ball\_moving\_dir; // ボールの移動方向 (0-360)

24 private double ball\_moving\_x,ball\_moving\_y; // ボールの移動方向

25 private double ball\_speed; // ボールのスピード

26 private int save\_x, save\_y; //プレイヤーの初期値を保存

27 public ArrayList<Item> item\_list = new ArrayList<Item>();

28 public ArrayList<Ballet> ballet\_list = new ArrayList<Ballet>();

29

30 private javax.swing.Timer timer = new javax.swing.Timer(1000,this);

31 private int timer\_count; //アイテムの制限時間

32

33 public CourtModel(int x,int y,int offset,int a){

34 court\_size\_x=x; court\_size\_y=y;

35 myself = new Player(x/2-30,y-offset,0,x);

36 save\_x = x/2-30; save\_y = y-offset;

37 stage = new StageSelectView();

38 bm=new BlockModel();

39 b=new Block();

40 bm.make\_block(a);

41 se = new AudioClip(new File("pi.mp3").toURI().toString());

42 se.setVolume(0.1);

43 ball\_x=x/2-30; ball\_y=y-offset-10;

44 ball\_moving\_dir=245;

45 calcMovingVector();

46 ball\_speed=1;

47 ball\_radius=5;

48 timer\_count = 0;

49 }

50

51 private void calcMovingVector(){

52 while (ball\_moving\_dir<0) ball\_moving\_dir+=360;

53 while (ball\_moving\_dir>360) ball\_moving\_dir-=360;

54 ball\_moving\_x=Math.cos(Math.toRadians(ball\_moving\_dir));

55 ball\_moving\_y=Math.sin(Math.toRadians(ball\_moving\_dir));

56 }

57

58 /\* ボールがコート内かどうかチェック

59 返り値: 0: コート内，1: 上方向で接触，2: 下で接触，3:左で接触，4:右で接触

60 yが583以上の時はコントローラーとの接触とする

61 5:左辺, 6:上辺左側, 7:上辺右側, 8:右辺 \*/

62 public int checkHit(double x,double y,double playerX){

63 if (y<568){

64 if (x<=ball\_radius && ball\_moving\_x<=0) return 3;

65 if (x>=court\_size\_x-ball\_radius && ball\_moving\_x>=0) return 4;

66 if (y<=ball\_radius && ball\_moving\_y<=0) return 1;

67 if (y>=court\_size\_y-ball\_radius && ball\_moving\_y>=0) return 2;

68 } else {

69 if (x>=playerX && x<=playerX+myself.getWidth()){

70 if (ball\_moving\_y>=0 && playerX+myself.getWidth()/2 > x) return 6;

71 if (ball\_moving\_y>=0 && playerX+myself.getWidth()/2 < x) return 7;

72 return 2;

73 }

74 if (x<=playerX+myself.getWidth()/2+ball\_radius &&

75 x>playerX+myself.getWidth()/2 && ball\_moving\_x<=0) return 8;

76 if (x>=playerX-myself.getWidth()/2-ball\_radius &&

77 x<playerX-myself.getWidth()/2 && ball\_moving\_x>=0) return 5;

78 }

79 return 0;

80 }

81

82 public int checkHitBlock(double x,double y){ //ブロックの辺り判定

83 double b\_x=b.getBlockX();double b\_y=b.getBlockY();

84 if (b\_x<=x && ball\_moving\_x>=0 && ball\_x<=b\_x) return 9;

85 if (b\_x+myself.getWidth()/2>=x &&

86 ball\_moving\_x<=0 && ball\_x>=b\_x+myself.getWidth()/2) return 10;

87 if (b\_y<=y && ball\_moving\_y>=0 &&ball\_y<=b\_y) return 11;

88 if (b\_y+20>=y && ball\_moving\_y<=0 &&ball\_y>=b\_y+20) return 12;

89 return 0;

90 }

91

92 public void typeUpdate(int type){ //アイテム取得

93 if(type==0) setBallSpeedUndo();

94 myself.settype(type);

95 timer.start();

96 System.out.println("typeを" + type + "に更新しました。");

97 timer\_count = 10;

98 }

99

100 public boolean isInCourt(){ //コート内なら1、違うなら0を返す

101 if(ball\_y<580)return true;

102 return false;

103 }

104

105 public void setBallSpeedUp(){ ball\_speed\*=1.03; }

106 public void setBallSpeedDown(){ ball\_speed\*=0.9; }

107 public void setBallSpeedUndo(){ ball\_speed=1; }

108

109 public void moveBall(){ // ボールを1ステップ進める．

110 double x0=ball\_x + ball\_moving\_x\*ball\_speed\*5;

111 double y0=ball\_y + ball\_moving\_y\*ball\_speed\*5;

112 int c=0,off=0;

113 int i=bm.check(x0,y0);

114 if(i!=-1){

115 b=bm.getBlock(i);

116 if(b.getCount()==1){

117 c=checkHitBlock(x0,y0);

118 b.setCount(0);

119 if(Math.random()<0.2)

120 item\_list.add(new Item((int)b.getBlockX()+b.width/2-5,

121 (int)b.getBlockY()+b.height/2-3));

122 bm.delete(i);

123 se.play();

124 }

125 }

126 if(c==0){

127 c=checkHit(x0,y0,myself.getPlayerX());

128 }

129 switch (c){

130 case 0:

131 ball\_x=x0; ball\_y=y0;

132 return;

133 case 1: // 上に接触

134 case 2:

135 case 11:

136 case 12: // 下に接触

137 ball\_moving\_dir=360-ball\_moving\_dir;

138 calcMovingVector();

139 ball\_x=x0;

140 break;

141 case 3: // 右に接触

142 case 4:

143 case 9:

144 case 10: // 左に接触

145 ball\_moving\_dir=180-ball\_moving\_dir;

146 calcMovingVector();

147 ball\_y=y0;

148 break;

149 case 5: // プレーヤーの左に接触(下)

150 ball\_moving\_dir+=180;

151 calcMovingVector();

152 break;

153 case 6: /\* プレーヤーの上左側に接触(上)

154 ボールが右側からきたら左に、左からきたら左に\*/

155 if(ball\_moving\_x>0){

156 ball\_moving\_dir+=180;

157 }else{

158 ball\_moving\_dir=(360-ball\_moving\_dir);

159 }

160 calcMovingVector();

161 break;

162 case 7: /\* プレーヤーの上右側に接触(上)

163 ボールが右側からきたら右に、左からきたら右に\*/

164 if(ball\_moving\_x>0){

165 ball\_moving\_dir=540-ball\_moving\_dir;

166 }else{

167 ball\_moving\_dir+=180;

168 }

169 calcMovingVector();

170 break ;

171 case 8: // プレーヤーの右に接触(下)

172 ball\_moving\_dir+=180;

173 calcMovingVector();

174 break;

175 /\*正確には，プレーヤーが打ち返した場合は，打ち返した瞬間に

176 ランダムで方向が変化するので，それを考慮する必要があるが,

177 ここではそれは考慮せずに簡易的に実装\*/

178 }

179 switch (c){

180 case 1: // 上に接触

181 ball\_y=2\*ball\_radius-y0;

182 break;

183 case 2: // 下に接触

184 ball\_y=(court\_size\_y-ball\_radius)\*2-y0;

185 break;

186 case 3: // 左に接触

187 ball\_x=2\*ball\_radius-x0;

188 break;

189 case 4: // 右に接触

190 ball\_x=(court\_size\_x-ball\_radius)\*2.0-x0;

191 break;

192 case 5: // プレーヤーの左に接触(下)

193 ball\_x=(court\_size\_x-ball\_radius)\*2.0-x0;

194 ball\_y=(court\_size\_y-ball\_radius)\*2-y0;

195 break;

196 case 6: /\* プレーヤーの上左側に接触(上)

197 ボールが右側からきたら左に、左からきたら左に\*/

198 if(ball\_moving\_x<0){

199 ball\_y=(570-ball\_radius)\*2-y0;

200 ball\_x = x0;

201 }else{

202 ball\_y=(570-ball\_radius)\*2-y0;

203 ball\_x = x0;

204 }

205 break;

206 case 7: /\* プレーヤーの上右側に接触(上)

207 ボールが右側からきたら右に、左からきたら右に\*/

208 if(ball\_moving\_x>0){

209 ball\_y=(570-ball\_radius)\*2-y0;

210 ball\_x = ball\_x - Math.abs(x0 - ball\_x);

211 }else{

212 ball\_y=(570-ball\_radius)\*2-y0;

213 ball\_x= x0;

214 }

215 break ;

216 case 8: // プレーヤーの右に接触(下)

217 ball\_x=2\*ball\_radius-x0;

218 ball\_y=(court\_size\_y-ball\_radius)\*2-y0;

219 break;

220 }

221 }

222

223 public void setBall(double x,double y,double dir,double speed){

224 // ボールの移動情報のセット

225 ball\_x=x; ball\_y=y; ball\_moving\_dir=dir;

226 calcMovingVector();

227 ball\_speed=speed;

228 }

229 public double getx0(){

230 return ball\_x + ball\_moving\_x\*ball\_speed\*5;

231 }

232

233 public double gety0(){

234 return ball\_y + ball\_moving\_y\*ball\_speed\*5;

235 }

236

237 public void draw(Graphics g){

238 g.setColor(Color.WHITE);

239 g.fillOval((int)(ball\_x-ball\_radius),(int)(ball\_y-ball\_radius),

240 (int)(ball\_radius\*2-1),(int)(ball\_radius\*2-1));

241 myself.draw(g);

242 for(Item i: item\_list) i.draw(g);

243 for(Ballet bal: ballet\_list) bal.draw(g);

244 }

245

246 public void setLine(double x, double y){

247 ball\_x = myself.getX()+myself.getWidth()/2;

248 ball\_y = myself.getY()-myself.getHeight()/2;

249 ball\_moving\_dir = Math.toDegrees(Math.acos(

250 (x-ball\_x)/Math.sqrt( (x-ball\_x)\*(x-ball\_x)+(y-ball\_y)\*(y-ball\_y) ) ));

251 ball\_moving\_dir -= 2\*ball\_moving\_dir;

252 calcMovingVector();

253 }

254

255 public void Linedraw(Graphics g){ //スタート時のボールの発射の軌道

256 g.setColor(Color.WHITE);

257 int startX = (int)ball\_x;

258 int startY = (int)ball\_y;

259 int endX = (int)(ball\_x+ball\_moving\_x);

260 int endY = (int)(ball\_y+ball\_moving\_y);

261

262 int count=0;

263 while(count<10){

264 g.drawLine(startX, startY, endX, endY);

265 startX = (int)(endX + ball\_moving\_x\*10);

266 startY = (int)(endY + ball\_moving\_y\*10);

267 endX = (int)(startX + ball\_moving\_x\*10);

268 endY = (int)(startY + ball\_moving\_y\*10);

269 count++;

270 }

271 }

272

273 public void initialize(){ //プレイヤーの初期化

274 setBallSpeedUndo();

275 myself.setPlayer(save\_x, save\_y);

276 ball\_x = myself.getX()+myself.getWidth()/2;

277 ball\_y = myself.getY()-myself.getHeight()/2;

278 item\_list = new ArrayList<Item>();

279 if(timer\_count>0) timer.stop();

280 timer\_count=0;

281 }

282

283 public int getItem(){

284 java.util.Iterator<Item> iter = item\_list.iterator();

285 while (iter.hasNext()) {

286 Item item\_iter = iter.next();

287 if(myself.getPlayerX()<item\_iter.x &&

288 item\_iter.x<myself.getPlayerX()+myself.getWidth()){

289 if(myself.getPlayerY()<item\_iter.y+5 &&

290 item\_iter.y<myself.getPlayerY()+myself.getHeight()){

291 typeUpdate(item\_iter.type);

292 iter.remove();

293 return item\_iter.type;

294 }

295 }

296 }

297 return -1;

298 }

299 public BlockModel balletHit(BlockModel b\_model){

300 //アイテムの弾のブロックへの辺り判定

301 java.util.Iterator<Ballet> iter = ballet\_list.iterator();

302 while (iter.hasNext()) {

303 Ballet ballet\_iter = iter.next();

304 int i=b\_model.check(ballet\_iter.x,ballet\_iter.y-5);

305 if(i!=-1){

306 b=b\_model.getBlock(i);

307 if(b.getCount()==1){

308 b.setCount(0);

309 if(Math.random()<0.1)

310 item\_list.add(new Item((int)b.getBlockX()+b.width/2-5,

311 (int)b.getBlockY()+b.height/2-3));

312 b\_model.delete(i);

313 bm.delete(i);

314 se.play();

315 iter.remove();

316 }

317 }

318 }

319 return b\_model;

320 }

321

322 public void item\_isInCourt(){

323 java.util.Iterator<Item> iter = item\_list.iterator();

324 while (iter.hasNext()) {

325 if(iter.next().y > 600){

326 iter.remove();

327 }

328 }

329 }

330

331 public void ballet\_isInCourt(){

332 java.util.Iterator<Ballet> iter = ballet\_list.iterator();

333 while (iter.hasNext()) {

334 if(iter.next().y < 0){

335 iter.remove();

336 }

337 }

338 }

339

340 public void actionPerformed(ActionEvent e){

341 //アイテムを取得したらタイマー開始

342 System.out.println("残り"+timer\_count+"秒");

343 timer\_count-=1;

344 if(timer\_count==0){

345 timer\_count = 0;

346 myself.settype(0);

347 timer.stop();

348 }

349 }

350

351 public int getTimer\_count(){

352 return timer\_count;

353 }

354 public void setTimer\_count(int time){

355 timer\_count = 0;

356 timer.stop();

357 }

358

359 public String getSpeed(){

360 double sp = ball\_speed;

361 sp\*=100;

362 if(sp - (int)sp >= 0.5){

363 sp += 1;

364 sp = (int)(sp+1);

365 }else{

366 sp = (int)(sp);

367 }

368 sp /= 100;

369 String ball\_speed\_str = (sp + "");

370 return ball\_speed\_str;

371 }

372

373 public void mouseClicked(MouseEvent e){}

374 public void mouseEntered(MouseEvent e){}

375 public void mouseExited(MouseEvent e){}

376 public void mousePressed(MouseEvent e){}

377 public void mouseReleased(MouseEvent e){}

378 }

・TitleView.java

1 package game;

2 import java.awt.\*;

3 import javax.swing.\*;

4 import java.awt.event.\*;

5

6 public class TitleView extends JPanel implements ActionListener{

7 private javax.swing.Timer timer;

8 private int clicked=0;//0:押してない、1:play, 2:make

9 public TitleView(){

10 JPanel panel = new JPanel(new GridLayout(1,2));

11 JButton playButton = new JButton("Play");

12 JButton makeButton = new JButton("Make");

13 JLabel title = new JLabel(new ImageIcon("./game/title-1.png"));

14 playButton.addActionListener(this);

15 playButton.setActionCommand("play");

16 makeButton.addActionListener(this);

17 makeButton.setActionCommand("make");

18 playButton.setFont(new Font("Arial", Font.PLAIN, 25));

19 makeButton.setFont(new Font("Arial", Font.PLAIN, 25));

20 panel.add(playButton); panel.add(makeButton);

21 setPreferredSize(new Dimension(301,601));

22 setFocusable(true);

23 setBackground(Color.BLACK);

24 panel.setBackground(Color.BLACK);

25 panel.setPreferredSize(new Dimension(300, 50));

26 setLayout(new BorderLayout());

27 add(title,BorderLayout.CENTER);

28 add(panel,BorderLayout.PAGE\_END);

29 }

30

31 public int getClicked(){

32 return this.clicked;

33 }

34

35 public void actionPerformed(ActionEvent e){

36 if(e.getActionCommand() != null){

37 String cmd = e.getActionCommand();

38 if(cmd.equals("play")){

39 clicked = 1;

40 }

41 if(cmd.equals("make")){

42 clicked = 2;

43 }

44 }

45 }

46 }

・StageSelectView.java

1 package game;

2 import java.awt.\*;

3 import javax.swing.\*;

4 import java.awt.event.\*;

5 import javax.swing.border.\*;

6

7 public class StageSelectView extends JPanel implements ActionListener{

8 private JPanel cardPanel;

9 private int stage;// 0:未選択、1,2,3,..ステージ番号

10 public StageSelectView(){

11 cardPanel = new JPanel();

12 setBackground(Color.BLACK);

13 JButton Stage1 =

14 new JButton("<html><img src='file:stage1.png' width=90 height=80></html>");

15 JButton Stage2 =

16 new JButton("<html><img src='file:stage2.png' width=90 height=80></html>");

17 JButton Stage3 =

18 new JButton("<html><img src='file:stage3.png' width=90 height=80></html>");

19 JButton Stage4 =

20 new JButton("<html><img src='file:stage4.png' width=90 height=80></html>");

21 JButton Stage5 =

22 new JButton("<html><img src='file:stage5.png' width=90 height=80></html>");

23 JButton Stage6 =

24 new JButton("<html><img src='file:user.png' width=90 height=80></html>");

25 JLabel intro =

26 new JLabel("<html><img src='file:introduce.png' width=300 height=200></html>");

27

28 Stage1.setBackground(Color.BLACK);

29 Stage2.setBackground(Color.BLACK);

30 Stage3.setBackground(Color.BLACK);

31 Stage4.setBackground(Color.BLACK);

32 Stage5.setBackground(Color.BLACK);

33 Stage6.setBackground(Color.BLACK);

34 intro.setBackground(Color.BLACK);

35 cardPanel.setBackground(Color.BLACK);

36

37 Stage1.setBorderPainted(false);

38 Stage2.setBorderPainted(false);

39 Stage3.setBorderPainted(false);

40 Stage4.setBorderPainted(false);

41 Stage5.setBorderPainted(false);

42 Stage6.setBorderPainted(false);

43

44 Stage1.addActionListener(this);

45 Stage2.addActionListener(this);

46 Stage3.addActionListener(this);

47 Stage4.addActionListener(this);

48 Stage5.addActionListener(this);

49 Stage6.addActionListener(this);

50

51 Stage1.setActionCommand("1");

52 Stage2.setActionCommand("2");

53 Stage3.setActionCommand("3");

54 Stage4.setActionCommand("4");

55 Stage5.setActionCommand("5");

56 Stage6.setActionCommand("6");

57

58 setPreferredSize(new Dimension(301,601));

59 setFocusable(true);

60 stage = 0;

61

62 cardPanel.add(Stage1);

63 cardPanel.add(Stage2);

64 cardPanel.add(Stage3);

65 cardPanel.add(Stage4);

66 cardPanel.add(Stage5);

67 cardPanel.add(Stage6);

68 setLayout(new GridLayout(2,1));

69 add(cardPanel);

70 add(intro);

71 }

72

73 public int getStage(){ return this.stage; }

74 public void setStage(int a){ stage=a; }

75

76 public void actionPerformed(ActionEvent e){

77 if(e.getActionCommand() != null){

78 String cmd = e.getActionCommand();

79 if(cmd.equals("1")){

80 setStage(1);

81 }

82 if(cmd.equals("2")){

83 setStage(2);

84 }

85 if(cmd.equals("3")){

86 setStage(3);

87 }

88 if(cmd.equals("4")){

89 setStage(4);

90 }

91 if(cmd.equals("5")){

92 setStage(5);

93 }

94 if(cmd.equals("6")){

95 setStage(6);

96 }

97 }

98 }

99 }

・GameView.java

1 package game;

2 import java.awt.\*;

3 import java.awt.event.\*;

4 import javax.swing.\*;

5 import game.\*;

6 import game.block.\*;

7 import java.util.\*;

8 import game.item.\*;

9

10 public class GameView extends JPanel

11 implements MouseListener,MouseMotionListener,ActionListener{

12 private javax.swing.Timer timer;

13 private CourtModel tm;

14 private BlockModel bm;

15 private boolean ballMoving;

16 final static int court\_size\_x=300;

17 final static int court\_size\_y=600;

18 private int game\_state = 0; //-1:gameover,0:playing,1:clear

19 private int click=0; //スタート時クリックしたら1

20 private int residue=3; //残基

21 private boolean click\_state=false;

22 private JLabel res\_label;

23 private JLabel item\_label;

24 private JLabel speed\_label;

25 private int time\_count; /\*ゲームスタートしてから開始、

26 100=1秒でスピードアップ\*/

27 public GameView(CourtModel tm){

28 this.tm=tm;

29 this.bm = new BlockModel();

30 ballMoving=false;

31 setBackground(Color.BLACK);

32 setPreferredSize(new Dimension(tm.court\_size\_x,tm.court\_size\_y));

33 setFocusable(true);

34 addMouseMotionListener(this);

35 addMouseListener(this);

36 timer = new javax.swing.Timer(10,this); // 10ミリ秒ごとにボールが移動

37 this.setLayout(null);

38 res\_label = new JLabel(""+residue);

39 item\_label = new JLabel();

40 speed\_label = new JLabel();

41 res\_label.setForeground(Color.WHITE);

42 item\_label.setForeground(Color.WHITE);

43 speed\_label.setForeground(Color.WHITE);

44 res\_label.setBounds(230, 490, 200, 200);

45 item\_label.setBounds(120,490,200,200);

46 speed\_label.setBounds(10,490,200,200);

47 this.add(res\_label);

48 this.add(item\_label);

49 this.add(speed\_label);

50 }

51

52 public void setBlockModel(int a){

53 bm.make\_block(a);

54 timer.start();

55 }

56

57 public void mouseDragged(MouseEvent e){}

58 public void mouseMoved(MouseEvent e){

59 /\* アプレットの上でマウスが動くのに合わせてコントローラを移動

60 マウスのx座標を取得する\*/

61 Point point = e.getPoint();

62 if(click\_state){

63 tm.myself.move(point.x);

64 }else{

65 tm.setLine(point.x, point.y);

66 }

67 repaint();

68 }

69

70 protected void paintComponent(Graphics g){

71 super.paintComponent(g);

72 tm.draw(g);

73 bm.draw(g);

74 if(!click\_state){

75 tm.Linedraw(g);

76 }

77 }

78

79 public int getState(){

80 return this.game\_state;

81 }

82

83 public void actionPerformed(ActionEvent e){

84 if(!tm.isInCourt()){// ボールが落下

85 tm.myself.settype(0);

86 click=0;

87 click\_state=false;

88 tm.initialize();

89 setColor(0);

90 if(residue==0){　//残基が0

91 game\_state = -1;

92 timer.stop();

93 }

94 }else{ //ボールが当たったブロックの配列の添字を返す

95 int i = bm.check(tm.getx0(),tm.gety0());

96 if(i >= 0)bm.delete(i);

97 }

98

99 if(bm.clear\_check()){ //クリアチェック

100 game\_state = 1;

101 timer.stop();

102 }

103

104 for(Item i: tm.item\_list){

105 i.moveDown();

106 }

107

108 for(Ballet bal: tm.ballet\_list){

109 bal.moveUp();

110 }

111

112 int type = tm.getItem();

113 bm = tm.balletHit(bm);

114 setColor(type);

115 if(type!=-1) System.out.println(type);

116 if(type==0) tm.setTimer\_count(0);

117 if(tm.getTimer\_count()<10)

118 item\_label.setText("00:0"+tm.getTimer\_count());

119 else

120 item\_label.setText("00:"+tm.getTimer\_count());

121

122 if(click\_state){

123 tm.moveBall();

124 time\_count++;

125 }

126

127 if(time\_count==100){

128 tm.setBallSpeedUp();

129 time\_count=0;

130 }

131

132 speed\_label.setText("speed="+tm.getSpeed());

133 repaint();

134 }

135

136 public void setColor(int type){

137 // アイテムの種類に応じて色を決める

138 if(type==0) item\_label.setForeground(Color.WHITE);

139 else if(type==1) item\_label.setForeground(Color.GREEN);

140 else if(type==2) item\_label.setForeground(Color.RED);

141 }

142

143

144 public void mousePressed(MouseEvent e) {

145 if(!click\_state) residue -=1;

146 else if(tm.myself.getType()==2)

147 tm.ballet\_list.add(new Ballet(tm.myself.getX()+tm.myself.getWidth()/2-1,

148 tm.myself.getY()-tm.myself.getHeight()/2-6));

149 res\_label.setText(""+residue);

150 click += 1;

151 if(click==1) click\_state=true;

152 }

153 public void mouseClicked(MouseEvent e) { }

154 public void mouseReleased(MouseEvent e){ }

155 public void mouseEntered(MouseEvent e) { }

156 public void mouseExited(MouseEvent e) { }

157 }

・Finish.java

1 package game;

2 import java.awt.\*;

3 import javax.swing.\*;

4 import java.awt.event.\*;

5

6 public class Finish extends JPanel implements ActionListener{

7 private javax.swing.Timer timer;

8 private int clicked=0;

9 //0:押されてない, 1:continue, 2: StageSelect

10

11 public Finish(int state){

12 JButton Continue = new JButton("Continue");

13 JButton StageSelect = new JButton("Title");

14 JLabel str\_state = new JLabel();

15 str\_state.setHorizontalAlignment(JLabel.CENTER);

16 JPanel Panel1 = new JPanel(new GridLayout(1,2));

17 if(state == -1) str\_state.setText("GAMEOVER");

18 if(state == 1) str\_state.setText("CLEAR");

19 str\_state.setFont(new Font("ＭＳ ゴシック", Font.BOLD, 45));

20 str\_state.setForeground(Color.WHITE);

21 setBackground(Color.BLACK);

22 Panel1.setBackground(Color.BLACK);

23

24 Continue.addActionListener(this);

25 Continue.setFont(new Font("Arial", Font.PLAIN, 25));

26 Continue.setActionCommand("Continue");

27

28 StageSelect.addActionListener(this);

29 StageSelect.setFont(new Font("Arial", Font.PLAIN, 25));

30 StageSelect.setActionCommand("Title");

31

32 Panel1.add(Continue);

33 Panel1.add(StageSelect);

34 setPreferredSize(new Dimension(301,601));

35 setFocusable(true);

36 setLayout(new BorderLayout(100,100));

37 add(str\_state,BorderLayout.CENTER);

38 add(Panel1,BorderLayout.PAGE\_END);

39 }

40

41 public int getClicked(){

42 return this.clicked;

43 }

44

45 public void actionPerformed(ActionEvent e){

46 if(e.getActionCommand() != null){

47 String cmd = e.getActionCommand();

48 if(cmd.equals("Continue")){

49 clicked = 1;

50 }

51 if(cmd.equals("Title")){

52 clicked = 2;

53 }

54 }

55 }

56 }

・StageMakeView.java

1 package game;

2 import java.awt.\*;

3 import javax.swing.\*;

4 import game.block.\*;

5 import java.util.\*;

6 import java.awt.event.\*;

7 import java.awt.event.MouseListener;

8 import java.awt.event.MouseEvent;

9 import java.nio.charset.StandardCharsets;

10 import java.io.File;

11 import java.nio.file.Files;

12 import java.io.FileWriter;

13 import java.io.IOException;

14 import javax.imageio.ImageIO;

15 import java.awt.image.BufferedImage;

16 import java.nio.file.Path;

17 import java.nio.file.Paths;

18

19 public class StageMakeView extends JPanel implements MouseListener,ActionListener{

20 ArrayList<Block> block\_list = new ArrayList<Block>();

21 private int type = 0,j=0,k=0; // 0:pen, 1:eraser

22 private boolean clicked=false;//false:押してない,ture:押した

23 private boolean result;

24 File userpng = new File("user.png");

25 public java.util.List<String> stage\_text = new ArrayList<String>();

26

27 public StageMakeView(){

28 setBackground(Color.BLACK);

29 setPreferredSize(new Dimension(301,601));

30 setFocusable(true);

31

32 try {

33 stage\_text = Files.readAllLines(Paths.get("user.txt"), StandardCharsets.UTF\_8);

34 System.out.println("print cournt");

35 for(String s : stage\_text) {

36 System.out.println(s);

37 }

38 } catch (IOException e) {

39 System.err.println( e);

40 }

41

42 //　ブロックをおける位置を点線で表示

43 int max\_y = 0;

44 for(int y=15;max\_y<10;y+=20){

45 int max\_x = 0;

46 max\_y++;

47 for(int x=15;max\_x<9;x+=30){

48 Block block = new Block(0, 255, 255);

49 block.setBlock(x,y);

50 block.setCount(Integer.parseInt(stage\_text.get(j).substring(k,k+1),10));

51 block\_list.add(block);

52 max\_x++;

53 k++;

54 }

55 j++;

56 k=0;

57 addMouseListener(this);

58 }

59

60

61

62 // 画面下のアイコンを定義、表示

63 {

64 JPanel icons = new JPanel(new GridLayout(1,3));

65 ImageIcon icon\_save = new ImageIcon("game/icons/icons-1.png");

66 ImageIcon icon\_pen = new ImageIcon("game/icons/icons-2.png");

67 ImageIcon icon\_eraser = new ImageIcon("game/icons/icons-3.png");

68 JButton bt\_save = new JButton(icon\_save);

69 JButton bt\_pen = new JButton(icon\_pen);

70 JButton bt\_eraser = new JButton(icon\_eraser);

71 icons.setBackground(Color.BLACK);

72 bt\_save.setBorderPainted(false);

73 bt\_pen.setBorderPainted(false);

74 bt\_eraser.setBorderPainted(false);

75 bt\_save.addActionListener(this);

76 bt\_pen.addActionListener(this);

77 bt\_eraser.addActionListener(this);

78 bt\_save.setActionCommand("save");

79 bt\_pen.setActionCommand("pen");

80 bt\_eraser.setActionCommand("eraser");

81 setLayout(new BorderLayout());

82 icons.add(bt\_eraser);

83 icons.add(bt\_pen);

84 icons.add(bt\_save);

85 add(icons,BorderLayout.SOUTH);

86 }

87 }

88

89 protected void paintComponent(Graphics g){

90 super.paintComponent(g);

91 System.out.println(type);

92 if(type == 3){

93 drawReal(g);

94 System.out.println("real");

95 }else{

96 drawFrame(g);

97 System.out.println("fra,e");

98 }

99 }

100

101 public void mouseEntered(MouseEvent e){}

102 public void mouseExited(MouseEvent e){}

103 public void mousePressed(MouseEvent e){}

104 public void mouseReleased(MouseEvent e){}

105 public void mouseClicked(MouseEvent e){

106 Point pt = e.getPoint();

107 int x = e.getX();

108 int y = e.getY();

109 /\* どのブロックがクリックされたか調べる。

110 クリックされたブロックが見つかったらそのcountを1にする\*/

111 for(Block b: block\_list){

112 if(b.x < x && x < b.x+b.width && b.y < y && y < b.y+b.height){

113 if(type == 0){

114 b.setCount(1);

115 }else if(type == 1){

116 b.setCount(0);

117 }

118 break;

119 }

120 repaint();

121 }

122 }

123

124 public void drawFrame(Graphics g){

125 for(Block b: block\_list){

126 if(b.getCount() == 1){

127 b.draw(g);

128 }else{

129 b.drawDotRect(g, 5, 0);

130 }

131 }

132 }

133

134 public void drawReal(Graphics g){

135 for(Block b: block\_list){

136 if(b.getCount() == 1){

137 b.draw(g);

138 }else{

139 b.drawDotRect(g, 5, 1);

140 }

141 }

142 }

143

144 public boolean getClicked(){ return this.clicked; }

145

146 public void actionPerformed(ActionEvent e){

147 int i = 0;

148 if(e.getActionCommand() != null){

149 String cmd = e.getActionCommand();

150 if(cmd.equals("save")){

151 type = 3;

152 Point point = new Point(this.getLocationOnScreen());

153 Rectangle rect = new Rectangle(point.x+13 ,point.y+12, 274, 204);

154

155 this.repaint();

156

157 System.out.println("saved");

158 try{

159 File file = new File("user.txt");

160 FileWriter filewriter = new FileWriter(file);

161

162 for(Block b: block\_list){

163 i++;

164 filewriter.write("" + b.getCount());

165 if(i%9==0) filewriter.write("\n");

166 }

167 filewriter.close();

168 }catch(IOException ex){

169 System.out.println(ex);

170 }

171 try{

172 Robot r = new Robot();

173 BufferedImage img = r.createScreenCapture(rect);

174 ImageIO.write(img,"jpg",userpng);

175

176 } catch(Exception exo){

177 }

178 clicked = true;

179 }

180 if(cmd.equals("pen")){ type = 0; }

181 if(cmd.equals("eraser")){ type = 1; }

182 }

183 }

184 }

・Player.java

1 package game;

2 import java.awt.\*;

3 import javax.swing.\*;

4

5 public class Player {

6 private int x,y; // プレーヤーの位置 (プレーヤ図形の左上の座標)

7 private int width,height; // プレーヤーの幅と高さ (長方形で表現)

8 private int bounds\_x1,bounds\_x2; // プレーヤの移動範囲

9 private int type; // 0:ノーマル、　1:ロング

10

11 public Player(int x,int y,int bounds\_x1,int bounds\_x2){

12 this.x=x; this.y=y;

13 this.bounds\_x1=bounds\_x1; this.bounds\_x2=bounds\_x2-30;

14 width=60; height=10;

15 type=0;

16 }

17 public double getPlayerX(){ return this.x; }

18 public double getPlayerY(){ return this.y; }

19 public void move(int mouse\_x){

20 if(mouse\_x>bounds\_x1 && mouse\_x<bounds\_x2)

21 x=mouse\_x;

22 }

23

24 // ボールが当たったかチェック

25 public boolean checkHit(double ball\_x,double ball\_y){

26 if (ball\_x>=x && ball\_x<x+width &&

27 ball\_y>=y && ball\_y<y+height) return true;

28 return false;

29 }

30

31 public void settype(int x){

32 this.type = x;

33 // typeがロングに変化したらwidthを更新

34 if(x==0){

35 this.width = 60;

36 this.bounds\_x2 = 242;

37 }else if(x == 1){

38 this.width = 120;

39 this.bounds\_x2 = 183;

40 }else if(x == 2){

41 this.width = 60;

42 this.bounds\_x2 = 242;

43 }

44 }

45

46 public int getType(){

47 return this.type;

48 }

49

50 // プレーヤーの描画

51 public void draw(Graphics g){

52 if(type!=2) g.fillRect(x,y,width,height);

53 if(type==2){

54 g.fillRect(x,y,width,height);

55 g.setColor(new Color(70,70,70));

56 g.fillRect(x+width/2-4,y+1,8,8);

57 g.fillRect(x+width/2-2,y-4,4,8);

58 g.setColor(Color.WHITE);

59 }

60 }

61

62 public int getX() { return x; }

63 public int getY() { return y; }

64 public int getWidth() { return width; }

65 public int getHeight() { return height; }

66

67 public void setPlayer(int x, int y){

68 this.x=x; this.y=y;

69 }

70 }

・Block.java

1 package game.block;

2 import java.awt.\*;

3 import javax.swing.\*;

4

5 public class Block{

6 private int count; //0:非表示,1:表示

7 public int x,y,width,height;//ブロックの縦、横、幅、高さ

8 public Block(){}

9 public Block(int r, int g, int b) {

10 color = new Color(r, g, b);

11 width=30; height=20;

12 }

13 public double getBlockX(){ return this.x; }

14 public double getBlockY(){ return this.y; }

15

16 public void setBlock(int x, int y){ this.x=x;this.y=y; }

17 public void setCount(int count){ this.count=count; }

18

19 public int getCount(){ return count; }

20

21 public boolean checkhit(double ball\_x, double ball\_y){//ボールとの当たり判定

22 if(ball\_x>=x && ball\_x<x+width && ball\_y>=y && ball\_y<y+height)

23 return true;

24 return false;

25 }

26 public boolean checkBalletHit(double ballet\_x, double ballet\_y){//弾との接触の判定

27 if(ballet\_x>=x && ballet\_x<x+width && ballet\_y>=y && ballet\_y<y+height)

28 return true;

29 return false;

30 }

31

32 public void draw(Graphics g){

33 Graphics2D g2 = (Graphics2D)g;

34 g2.setColor(color);

35 g2.setStroke(new BasicStroke(4.0f));

36 g2.drawRoundRect(x+2, y+2, width-4, height-4, 10, 10);

37 }

38

39

40 public void drawDotRect(Graphics g, int interval, int i){

41 int startX = x;

42 int startY = y;

43 int endX = startX + interval;

44 int endY = startY + interval;

45 Graphics2D g2 = (Graphics2D)g;

46 if(i==0) g2.setColor(Color.WHITE);

47 if(i==1) g2.setColor(Color.BLACK);

48 g2.setStroke(new BasicStroke(1.0f));

49 while(endX < x+width) {

50 g2.drawLine(startX, startY, endX, startY);

51 g2.drawLine(startX, startY+height, endX, startY+height);

52 startX = endX + interval;

53 endX = startX + interval;

54 }

55

56 startX = x;

57 startY = y;

58 endX = startX + interval;

59 endY = startY + interval;

60

61 while(endY < y+height) {

62 g2.drawLine(startX, startY, startX, endY);

63 g2.drawLine(startX+width, startY, startX+width, endY);

64 startY = endY + interval;

65 endY = startY + interval;

66 }

67 }

68 }

・BlockModel.java

1 package game.block;

2 import java.awt.\*;

3 import java.awt.event.\*;

4 import javax.swing.\*;

5 import game.\*;

6 import java.util.\*;

7 import java.nio.file.Path;

8 import java.nio.file.Paths;

9 import java.nio.file.Files;

10 import java.nio.charset.StandardCharsets;

11 import java.io.IOException;

12

13 public class BlockModel{

14 private int x, y, max\_x, max\_y,i, r, g, b;

15 //ブロックをグラデーションにするためにrgbを採用

16 private ArrayList<Block> block\_list;

17 public java.util.List<String> stage\_text = new ArrayList<String>();

18 public BlockModel(){}

19 public void make\_block(int a){/\*ブロックの生成\*/

20 i=-1;

21 Block block;

22 block\_list=new ArrayList<Block>();

23 String stage;

24 int j=0, k=0, n;

25

26 Path stage\_text\_path = Paths.get("stage1.txt");

27 switch(a){

28 case

29 stage\_text\_path = Paths.get("stage1.txt");

30 break;

31 case 2:

32 stage\_text\_path = Paths.get("stage2.txt");

33 break;

34 case 3:

35 stage\_text\_path = Paths.get("stage3.txt");

36 break;

37 case 4:

38 stage\_text\_path = Paths.get("stage4.txt");

39 break;

40 case 5:

41 stage\_text\_path = Paths.get("stage5.txt");

42 break;

43 case 6:

44 stage\_text\_path = Paths.get("user.txt");

45 break;

46 }

47 try {

48 stage\_text =

49 Files.readAllLines(stage\_text\_path, StandardCharsets.UTF\_8);

50 System.out.println("print court");

51 for(String s : stage\_text) {

52 System.out.println(s);

53 }

54 } catch (IOException e) {

55 System.err.println( e);

56 }

57 r=255;

58 g=150;

59 b=180;

60 max\_y = 0;

61 for(y=10;max\_y<10;y+=20){

62 max\_x = 0;

63 max\_y++;

64 r-=23;

65 g-=2;

66 b+=6;

67 for(x=15;max\_x<9;x+=30){

68 block = new Block(r, g, b);

69 block.setBlock(x,y);

70 block.setCount(

71 Integer.parseInt(stage\_text.get(j).substring(k,k+1),10));

72 block\_list.add(block);

73 max\_x++;

74 k++;

75 }

76 j++;

77 k=0;

78 }

79 }

80

81 public void draw(Graphics g){

82 for(Block b: block\_list){

83 if(b.getCount() == 1){

84 b.draw(g);

85 }

86 }

87 }

88

89 public int check(double ballet\_x,double ballet\_y){

90 //弾と接触していたらそのブロックの配列の添え字を返す

91 int i=0;

92 for(Block b: block\_list){

93 if(b.checkhit(ballet\_x,ballet\_y)){

94 return i;

95 }

96 i++;

97 }

98 return -1;

99 }

100

101 public Block getBlock(int i){

102 return block\_list.get(i);

103 }

104

105 public void delete(int i){

106 if(i==-1){

107 return;

108 }

109 Block b = block\_list.get(i);

110 b.setCount(0);

111 }

112

113 public boolean clear\_check(){/\*クリアチェック

114 全てのブロックがCount=0ならばクリア\*/

115 for(Block b: block\_list){

116 if(b.getCount() == 1) return false;

117 }

118 return true;

119 }

120

121 }

・Item.java

1 package game.item;

2 import java.awt.\*;

3 import javax.swing.\*;

4 import java.lang.Math;

5

6 public class Item{

7 public int type;

8 // 0:ニュートラル, 1:long, 2:ミサイルをうつ:

9 public int size\_x, size\_y;

10 public int x,y;

11 public Item(){}

12 public Item(int a, int b){

13 double r = Math.random();

14 if(r < 0.3) this.type = 0;

15 else if(r < 0.6) this.type = 1;

16 else this.type = 2;

17 size\_x = 10;

18 size\_y = 5;

19 x = a; y = b;

20 }

21

22 public void moveDown(){　y += 2;　}

23 public int getType(){　return type;　}

24

25 public void draw(Graphics g){

26 Graphics2D g2 = (Graphics2D)g;

27 if(type==0) g2.setColor(Color.WHITE);

28 else if(type==1) g2.setColor(Color.GREEN);

29 else if(type==2) g2.setColor(Color.RED);

30 g2.setStroke(new BasicStroke(2.0f));

31 g2.drawRoundRect(x, y, size\_x, size\_y, 2, 2);

32 }

33 }

・Ballet.java

1 package game.item;

2 import java.awt.\*;

3 import javax.swing.\*;

4

5 public class Ballet{

6 public int size\_x, size\_y;

7 public int x,y;

8 public Ballet(){}

9 public Ballet(int a, int b){

10 size\_x = 2;

11 size\_y = 5;

12 x = a; y = b;

13 }

14 public void moveUp(){　y -= 5;　}

15

16 public void draw(Graphics g){

17 Graphics2D g2 = (Graphics2D)g;

18 g2.setColor(Color.GREEN);

19 g.fillRect(x-1,y-5,size\_x,size\_y);

20 }

21 }

文責：鴨田　恭佑