2015 – 後学期

プログラミング演習

グループプログラミングレポート

Shooting

|  |  |
| --- | --- |
| 学科 | 総合情報学科 |
| クラス | J1 |
| グループ番号 | 1 |
| 1410007 | 飯野　純平 |
| 1410019 | 江村　万里 |
| 1410055 | 佐藤　禎紀 |

1、概要説明

今回の課題で私達が作成したプログラムは「縦スクロール型シューティングゲーム」です。自機・敵機を画面上に配置しており、自機を操作し、上下左右に移動、弾を発射して敵機を倒すなどしながら画面上部に進んでゆき、時間経過(目的地到達)でクリア画面に移行する。敵機又は敵機が発射する弾に自機は触れるとライフが減り、ライフが0になるとゲームオーバー画面に移行する。

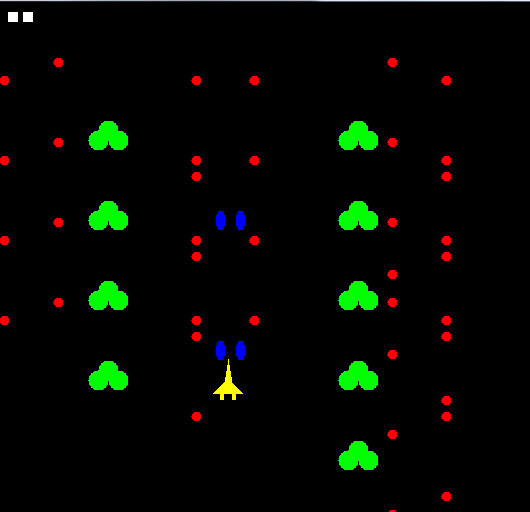


図.1　ゲーム画面

プログラム作成の分担は大まかに以下の通り

飯野：コントローラ部分一部

江村：ModelClass・オブジェクトの定義など

佐藤：TitleFrameClass・ゲームの大筋など

詳細は各担当部分の項にて記述。

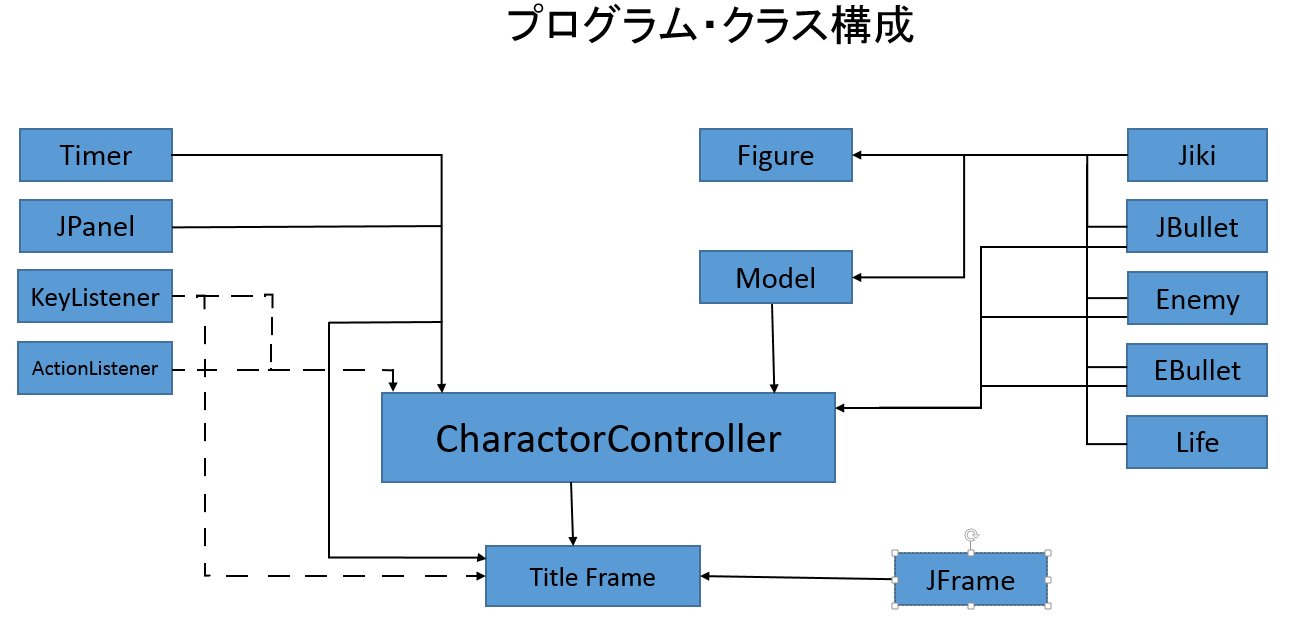
作業は、基になる最低限のプログラム作成後各自が担当分のプログラムを作成しそれを共有、つなぎ合わせる形で作成した。

文責：飯野

2、設計方針

まず第一にプログラムの作成にあたって、既存のプログラムを機能拡張する形で行うのではなく一から全て自分達でプログラムを作成することを目標としてプログラムを作成しました。

使用したclassは以下の通りである。



　　　　図,1　クラス相関図　プレゼンテーション用のスライドから

Jiki,JBullet,Enemy,EBullet,Lifeの各クラスで各オブジェクトを定義し、それらをFigureクラスによって描写する。またそれらはModelクラスで利用する。

ゲームの大筋はCharactorControllerクラスで成っており、Modelからオブジェクトを配置してゲームを進行する。

メインとして用いられているのはTitleFrameクラスでウィンドウやタイトル画面・ゲーム画面など必要なものを用意し、ゲーム画面でCharactorControllerを呼び出す。

文責：飯野

3、プログラム説明

3.1江村担当分

3.1.1 Figureクラス

ゲーム画面に表示させる物体のデータを格納するクラスの雛形。後述するJikiクラス、Enemyクラス、JBulletクラス、EBulletクラスLifeクラスにこれを継承させる。

1. フィールド

int型：x, y, width, height, speed

x, yは物体をゲーム画面に表示させる位置、width, height 物体の横と縦のの大きさ、speedは物体の移動する速さを表す。

Color型：color

　物体の色を示す。

1. コンストラクタ

public Figure (int x,int y,int w,int h,int s,Color c)

引数x, y, w, h, s, cをそれぞれフィールドの変数x, y, width, height, speed, colorに代入する。

1. メソッド

Public void draw(Graphics g)

　物体を描画する。ここではまだ内容を定義せず、子クラスでオーバーライドする。

3.1.2 Jikiクラス

自機のデータを格納する。Figureクラスを継承している。

(1) フィールド

Figyureクラスと同様

(2) コンストラクタ

public Jiki(int x,int y,int w,int h,int s,Color c)

自機のデータの初期化を行う。

1. メソッド

public void draw(Graphics g)

自機を描画する。図形を描画するメソッドを用いて戦闘機のような形になるようにした。

3.1.3 Enemyクラス

敵機のデータを格納する。Figureクラスを継承している。

(1) フィールド

ここではFigureクラスから追加した分について述べる。

int 型：type, shotcount

typeは敵機の動作パターンを表し、shotcountは敵機が弾を発射する時間の間隔を表す。どちらも後述するModelクラスのEnemyMoveメソッドで用いる。

boolean型：dflag

このオブジェクトを削除する場合はこの変数をtrueにする。後述するModelクラスのDeleteFegureクラスで用いる。

(2) コンストラクタ

public Enemy(int x,int y,int w,int h,int s,int t,Color c)

敵機のデータの初期化を行う。

(3) メソッド

public void draw(Graphics g)

敵機を描画する。今回は３つの円を組み合わせた形で敵機を表現した。

3.1.4 JBulletクラス

自機が発射した弾のデータを格納する。Figureクラスを継承している。

(1) フィールド

ここではFigureクラスから追加した分について述べる。

int 型：type

弾の動作パターンを表す。

boolean型：dflag

　　　Enemyクラスのものと同様である。

(2) コンストラクタ

public JBullet(int x,int y,int w,int h,int s,int t,Color c)

弾のデータの初期化を行う。

(3) メソッド

public void draw(Graphics g)

弾を描画する。今回はただの円形で弾を表現した。

3.1.5 EBulletクラス

敵機が発射した弾のデータを格納する。主な特徴はJBulletクラスと同様である。

3.1.6 Lifeクラス

自機のライフ（耐久値）の表示に関するデータを格納する。Figureクラスを継承している。

(1) フィールド

ここではFigureクラスから追加した分について述べる。

int型：count

自機の残りライフを表す。これが0になった時、ゲームオーバーとなる。

(2) コンストラクタ

public Life(int x,int y,int l)

ライフの初期化やゲーム画面に描画するライフ表示の初期化を行う。

(3) メソッド

public void draw(Graphics g)

ゲーム画面左上に自機の残りライフを表示させる。今回は自機の残りライフを白色の四角形の個数で表現した。

3.1.7 Modelクラス

ゲーム全体のデータを管理し、物体の移動等に関するメソッドを格納している。

1. フィールド

Jiki型：jiki

　自機のデータを表す。Figureクラスを継承している。主な特徴は

Life型：life

　自機のライフを表す。

配列：ArrayList<Enemy> enemy

ArrayList<JBullet> jbullet

ArrayList<EBullet> ebullet

それぞれ敵機、自機が発射した弾、敵機が発射した弾を表す。複数の敵機と弾を管理するためにArrayListを用いた。

1. コンストラクタ

public Model()

　自機とライフの初期設定と配列の生成を行う。

1. メソッド

主に自機の動作、敵機と弾の生成・動作、ライフの変動、当たり判定、ゲームオーバー判定などを管理する。Modelクラスはメソッドの数が多いが、ここでは特に説明が必要と思われるものについて述べる。

public void JikiMove(boolean r,boolean l,boolean d,boolean u)

自機の移動を表す変数がtrueの時、自機がその方向に移動する。変数の値はキーの操作によって変更される。佐藤担当のCharactorControllerクラスのactionPaformedメソッドから呼び出すことで、自機をスムーズに移動させることができる。

public void EnemyMove()

　敵機を移動させ、enemyの要素内のメンバー変数shotcountの値が一定値になると弾を発射させる。for文とArrayListを用いて複数の敵機をすべて動かせるようにした。またenemy要素内のメンバー変数typeの値によって敵機の動作パターンが変わるようにした。変数の値の種類を増やすことで動作パターンを追加できるのが特徴である。

　 public void BulletMove()

　自機や敵機が発射した弾を移動させる。主な特徴はEnemyMoveメソッドと同様である。

public void createBullet(int x,int y,int t)

　自機や敵機が発射する弾を生成する。x, yはJBullet型変数およびEBullet型変数のメンバー変数x, yに代入され、tはJBullet型変数およびEbullet型変数のメンバー変数typeに代入される。tの値によって自機が発射した弾か敵機が発射した弾かがわかるようになっている。

public void Hitcheck()

後述するHitboxメソッドを呼び出して自機、敵機、弾の当たり判定を行う。敵機や弾が衝突した場合、enemy, jbullet, ebulletの要素内のメンバー変数dflagをtrueにする。dlagがtrueになった要素は後述するDeleteFigureメソッドによって削除される。自機が衝突した場合、life変数のメンバー変数counntの値を1減少させる。これは自機のライフの減少を意味する。

public boolean Hitbox(int x1,int y1,int w1,int h1, int x2,int y2,int w2,int h2)

当たり判定に用いる。判定の方法は２つの四角形がかさなるかどうかで判定する。x, yは四角形の左上隅の点の座標、w, hは四角形の横と縦の大きさを表す。

public void DeleteFigure()

enemy, jbullet, ebulletの要素のうち、メンバー変数dflagがtrueの要素を削除する。衝突した敵や弾を消すときに用いる。

public boolean GameOver()

　自機のライフが0になったときにtrueを返す。ゲームオーバーの判定に用いる。

3.2佐藤担当分

3.2.1CharactorControllerクラス

　このクラスはJPanelを継承し、 ActionListenerを実装しているクラスで、主にゲーム自体の構成やゲームの描画などを行っているクラスである。工夫した点はTimerに対するactionPerformed内にModelクラス内の自機の動きに関するメソッドを入れることにより動きをスムーズにすることができたことである。

(1)コンストラクタ

29 public CharactorController()

30 {

31 timer = new javax.swing.Timer(50,this); //timerにtimer挿入(0.05秒おき)

32

33 model = new Model(); //modelオブジェクトの生成

34

35 this.setBackground(Color.BLACK); //背景設定

36 this.setFocusable(true); //キー入力をGUI部品が受け付ける

37 this.addKeyListener(this); //キー入力を追加

38 }

　このクラスのコンストラクタでは、Modelクラスのオブジェクトを変数modelに生成し、その他上のようにtimerの生成、背景の設定、キー入力が受け付けるように関数を指定した。

(2)timerstartメソッド、timerstopメソッド

　これらのメソッドはそれぞれコンストラクタで生成したtimerをスタート、ストップするメソッドである。これらは次のTitleFrameクラスで使う。

(3)paintComponentメソッド

　ここではModelクラスで作ったメソッドを利用して自機、ライフ、敵、そして自機、敵の弾の描画を実行できるようにした。

(4)actionerPerformedメソッド

　これはtimerが更新するごとに実行されるメソッドである。つまり、時間が経つごとにこのメソッドは実行される。

86 public void actionPerformed(ActionEvent e)

87 {

88 time++; //timerを0.1秒に換算

89

90 model.JikiMove(jrflag,jlflag,jdflag,juflag);

91 if(jshotflag==true && time%5==0)

92 {

93 model.createBullet(model.xJiki()+model.wJiki()/3,model.yJiki(),1); //自機弾の発射

94 model.createBullet(model.xJiki()-model.wJiki()/3,model.yJiki(),1); //自機弾の発射

95 }

96

97 Game();

98

99 /\*敵機の出現\*/

100 if(time == 10) //1秒たったら

101 {

102 for(int i=-320;i<=0;i+=80)

103 model.createEnemy(100,i,1);

104

105 for(int i=-320;i<=0;i+=80)

106 model.createEnemy(350,i,1);

107 }

108 if(time == 150)

109 {

110 for(int i=-320;i<=0;i+=80)

111 model.createEnemy(50,i,2);

112

113 for(int i=-320;i<=0;i+=80)

114 model.createEnemy(400,i,3);

115 }

116 if(time == 200)

117 for(int i=-160;i<=0;i+=80)

118 model.createEnemy(300,i,1);

119 if(time == 230)

120 for(int i=-160;i<=0;i+=80)

121 model.createEnemy(150,i,1);

122 if(time == 260)

123 for(int i=-160;i<=0;i+=80)

124 model.createEnemy(350,i,3);

125 if(time == 290)

126 for(int i=-160;i<=0;i+=80)

127 model.createEnemy(100,i,2);

128 if(time == 360)

129 for(int i=-160;i<=0;i+=80)

130 model.createEnemy(270,i,1);

131

132

133

121 model.createEnemy(150,i,1);

122 if(time == 260)

123 for(int i=-160;i<=0;i+=80)

124 model.createEnemy(350,i,3);

125 if(time == 290)

126 for(int i=-160;i<=0;i+=80)

127 model.createEnemy(100,i,2);

128 if(time == 360)

129 for(int i=-160;i<=0;i+=80)

130 model.createEnemy(270,i,1);

131

132

133

134

135 model.EnemyMove();

136 model.BulletMove();

137 model.HitCheck();

138 model.DeleteFigure();

139

140 repaint(); //そして再描画

141

142 }

　まず、timerの更新を視覚化するためにint型のtime変数を 更新ごとに1加算している。そして、model内のJikiMoveやcreateBulletを使って自機の動き、自機弾の発射を指定している。また、次に説明するGameメソッドを使用することにより、このあとのTitleFrameクラスでGameOver画面、GameClear画面に遷移できるようにしている。このあとの100行目から130行目に関しては、時間が経てばmodelクラス内の敵を出現するメソッドであるcreateEnemyメソッドを用いて敵を出現させるようにした。135行目から138行目のメソッドは当たり判定に関するメソッドで、詳しくはmodelクラスの説明内で解説してある。

最後、時間が経つごとに再描画してある。

(5)Gameメソッド

　このメソッドはゲームオーバーになったらint型変数iを1に、ゲームクリアになったらiを2にして、その後のTitleFrameクラスで画面の遷移ができるようにしている。

3.2.2TitleFrameクラス

　このクラスはTitle画面、GameOver画面、GameClear画面も全て含めたこのプログラム全体の構成を担っているクラスである。工夫した点はCardLayoutというレイアウトを使い、画面の遷移を可能にすることができたことである。このクラスはJFrameを継承し、ActionListener及びKeyListenerを実装している。キー操作に関するメソッドは飯野が担当分であるため、ここではそれ以外のプログラムに関して説明していく。

(1)コンストラクタ

21 /\*コンストラクタ(フレーム内の設定)\*/

22 public TitleFrame()

23 {

24 /\*JPanel設定\*/

25 p1 = new JPanel(); //大本のパネル

26 p2 = new JPanel(); //１ページ目のパネル（タイトル画面）

27 p3 = new JPanel(); //２ページ目のパネル（ゲーム画面）

28 p4 = new JPanel(); //3ページ目のパネル（ゲームオーバー画面）

29 p5 = new JPanel(); //4ページ目のパネル（クリア画面）

30 chara = new CharactorController(); //2ページ目に貼り付けるゲーム画面

31 timer = new javax.swing.Timer(1,this);

32 timer.start();

33

34 p2.setBackground(Color.BLACK); //背景設定

35 p4.setBackground(Color.BLACK); //背景設定

36 p5.setBackground(Color.BLACK); //背景設定

37

38

39 /\*CardLayoutの設定\*/

40 layout = new CardLayout();

41

42 /\*KeyListener追加\*/

43 p1.addKeyListener(this);

44 p1.setFocusable(true);

45

46 /\*p2,p4をBorderLayoutで設定\*/

47 p2.setLayout(new BorderLayout());

48 p4.setLayout(new BorderLayout());

49 p5.setLayout(new BorderLayout());

50

51 /\*p2にJLabelを貼り付ける\*/

52 JLabel l2 = new JLabel("<html>"

53 +"<span style='font-size:110pt;font-family:Impact,Charcoal;"

54 +"color:yellow;'>"

55 +"Shooting</span></html>",

56 +JLabel.CENTER);

57

58

59 JLabel l3 = new JLabel("<html>"

60 +"<span style='font-size:30pt;color:white;'>"

61 +"十字キーで移動、zで弾発射。<br/>"

62 +"<span style='font-size:30pt;color:white;'>"

63 +"Enterでゲームスタート!!"

64 +"</span></html>",

65 +JLabel.CENTER);

66

67 p2.add(l2,BorderLayout.CENTER);

68 p2.add(l3,BorderLayout.SOUTH);

69

70 /\*2ページ目\*/

71 p3.setLayout(new BorderLayout());

72 p3.add(chara,BorderLayout.CENTER);

73

74 /\*p4にJLabelを貼り付ける\*/

75 JLabel l4 = new JLabel("<html>"

76 +"<span style='font-size:90pt;font-family:Impact,Charcoal;"

77 +"color:blue;'>"

78 +"GAMEOVER</span></html>",

79 +JLabel.CENTER);

80

81

82 JLabel l5 = new JLabel("<html>"

83 +"<span style='font-size:30pt;color:white'>"

84 +"2秒後、タイトル画面を表示します。"

85 +"</span></html>",

86 +JLabel.CENTER);

87

88 p4.add(l4,BorderLayout.CENTER);

89 p4.add(l5,BorderLayout.SOUTH);

90

91 /\*p4にJLabelを貼り付ける\*/

92 JLabel l6 = new JLabel("<html>"

93 +"<span style='font-size:120pt;font-family:Impact,Charcoal;"

94 +"color:red;'>"

95 +"GAME<br/>CLEAR!!</span></html>",

96 +JLabel.CENTER);

97

98

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

55 +"Shooting</span></html>",

56 +JLabel.CENTER);

57

58

59 JLabel l3 = new JLabel("<html>"

60 +"<span style='font-size:30pt;color:white;'>"

61 +"十字キーで移動、zで弾発射。<br/>"

62 +"<span style='font-size:30pt;color:white;'>"

63 +"Enterでゲームスタート!!"

64 +"</span></html>",

65 +JLabel.CENTER);

66

67 p2.add(l2,BorderLayout.CENTER);

68 p2.add(l3,BorderLayout.SOUTH);

69

70 /\*2ページ目\*/

71 p3.setLayout(new BorderLayout());

72 p3.add(chara,BorderLayout.CENTER);

73

74 /\*p4にJLabelを貼り付ける\*/

75 JLabel l4 = new JLabel("<html>"

76 +"<span style='font-size:90pt;font-family:Impact,Charcoal;"

77 +"color:blue;'>"

78 +"GAMEOVER</span></html>",

79 +JLabel.CENTER);

80

81

82 JLabel l5 = new JLabel("<html>"

83 +"<span style='font-size:30pt;color:white'>"

84 +"2秒後、タイトル画面を表示します。"

85 +"</span></html>",

86 +JLabel.CENTER);

87

88 p4.add(l4,BorderLayout.CENTER);

89 p4.add(l5,BorderLayout.SOUTH);

90

91 /\*p4にJLabelを貼り付ける\*/

92 JLabel l6 = new JLabel("<html>"

93 +"<span style='font-size:120pt;font-family:Impact,Charcoal;"

94 +"color:red;'>"

97

98

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

95 +"GAME<br/>CLEAR!!</span></html>",

96 +JLabel.CENTER);

97

98

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER);

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

　JPanelは全体、タイトル画面、ゲーム画面、ゲームオーバー画面、ゲームクリア画面の5つのPanelを用意した。そして、timerの生成、背景設定、キー入力が受け付けるための設定、レイアウト設定をした。

　51行目から68行目に関してはTitle画面の生成をしている。JLabelを使い、HTMLでフォントや文字の大きさなどを指定しながらゲームタイトルと簡単なゲーム操作を記載した。また、74行目から89行目、91行目から107行目はそれぞれGameOver画面、GameClear画面をTitle画面と同様にJLabel、HTMLを使用しながらゲームオーバーもしくはゲームクリアである旨と画面が一定時間後に自動的に遷移するということを記載した。また、変数charaに前記のCharacterControllerクラスのオブジェクトを生成し、そのcharaをp3に貼り付けた。

　そして、p1に複数の部品を切り替えて表示できるCardLayoutを設定し、p2~p5を貼り付け、それが表示されるように115行目のようなメソッド記述をした。

　最後にはFrameのサイズなどの設定をした。

(2)actionPerformedメソッド

　このメソッドはtimerが更新されるごとに実行されるメソッドで、主にここでは画面の遷移について設定してある。

　前記のGameメソッドにより、ゲームオーバーになればi==1、ゲームクリアになればi==2となるように指定したある。このメソッドではこのi==1である時、layout.nextメソッドを使ってページを遷移し、GameOver画面に移行するように設定してある。i==2のときも同様にlayout.nextメソッドを使ってGameClear画面に移行するようにしている。そして、変数iを4に変えて、int型のtime変数を更新されるごとに加算。一定時間たったらsetVisibleメソッドでこのフレームを消し、新たなフレームを生成することによって 新たなゲームがプレイできるようにしている。

(3)mainメソッド

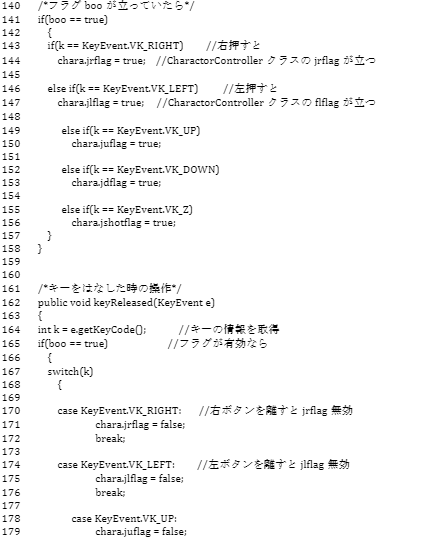
　このメソッドではフレームの生成しか行っていない。

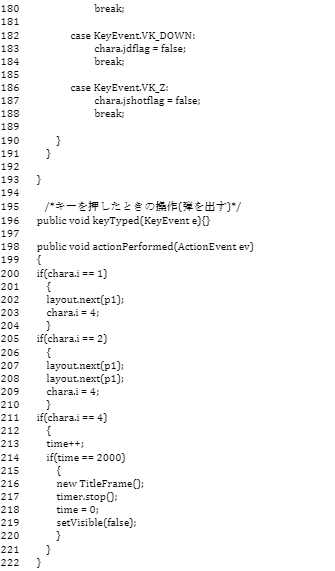
[文責：佐藤]

3.3飯野担当分

自機の操作等







まず各ボタンに関するフラグをboolean変数で定義し、ボタンが押されているときtrue、放されているときfalseとなるように関数を定義する。

また、自機を移動する関数、つまり自機の座標に一定値を加減算して再描画する関数を別で用意しておき、各ボタンのフラグがtrueのときのみ関数を実行するように条件分岐を設定する。またZキーに関しても、自機の弾を生成する関数を用意し、ボタンが押された場合に関数を実行するように設定している。

文責：飯野

4.実行例

　まず、プログラムの実行をする。Javaファイルで実行するにはメインクラスであるTitleFrameクラスを実行。Jarファイルで実行する場合には以下のように実行する。

Yoshiki$ java -jar Shooting.jar

すると以下のようなタイトル画面が表示される。

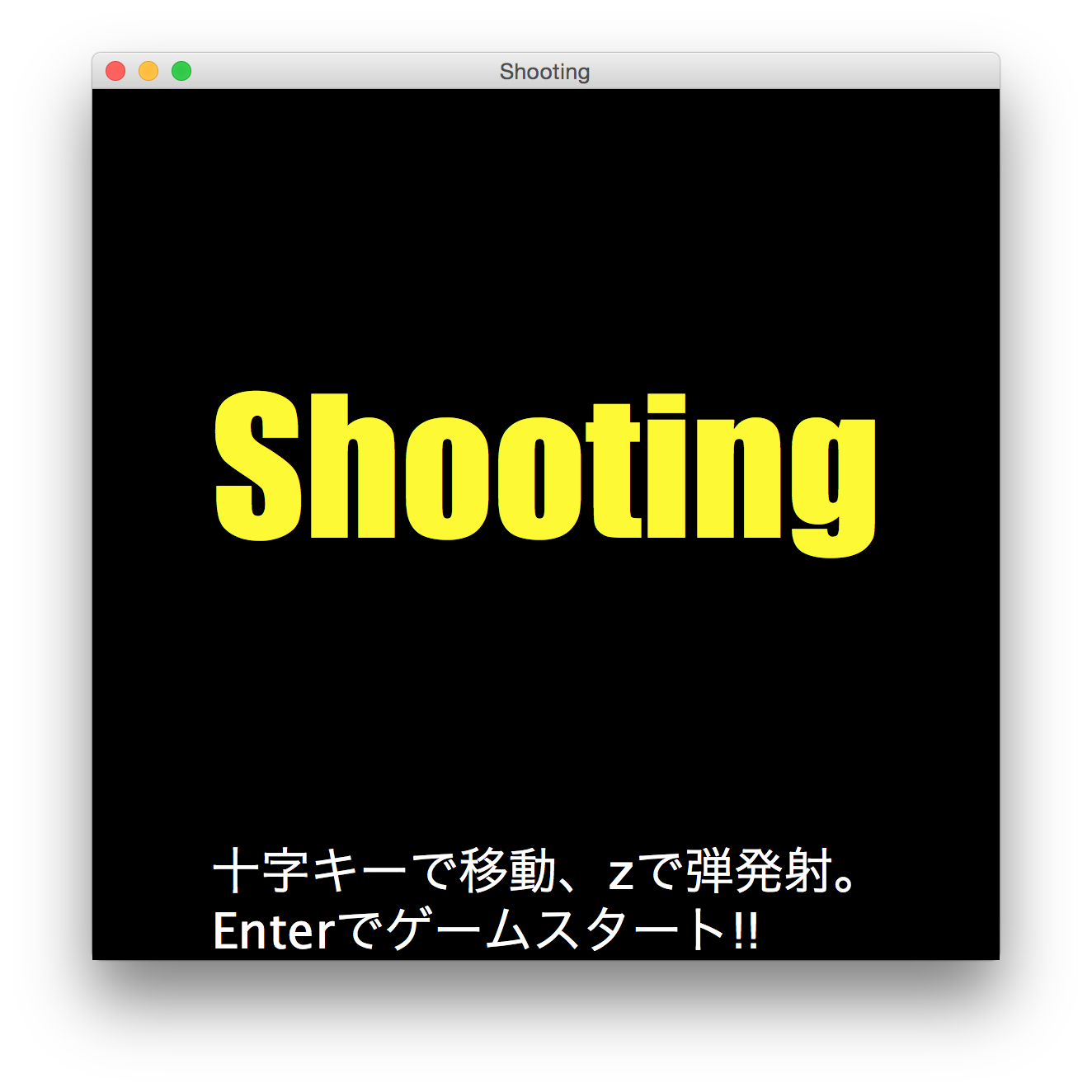


図１：タイトル画面

そして、Enterキーを押すとゲームがスタートし、次のようなゲーム画面に遷移する。

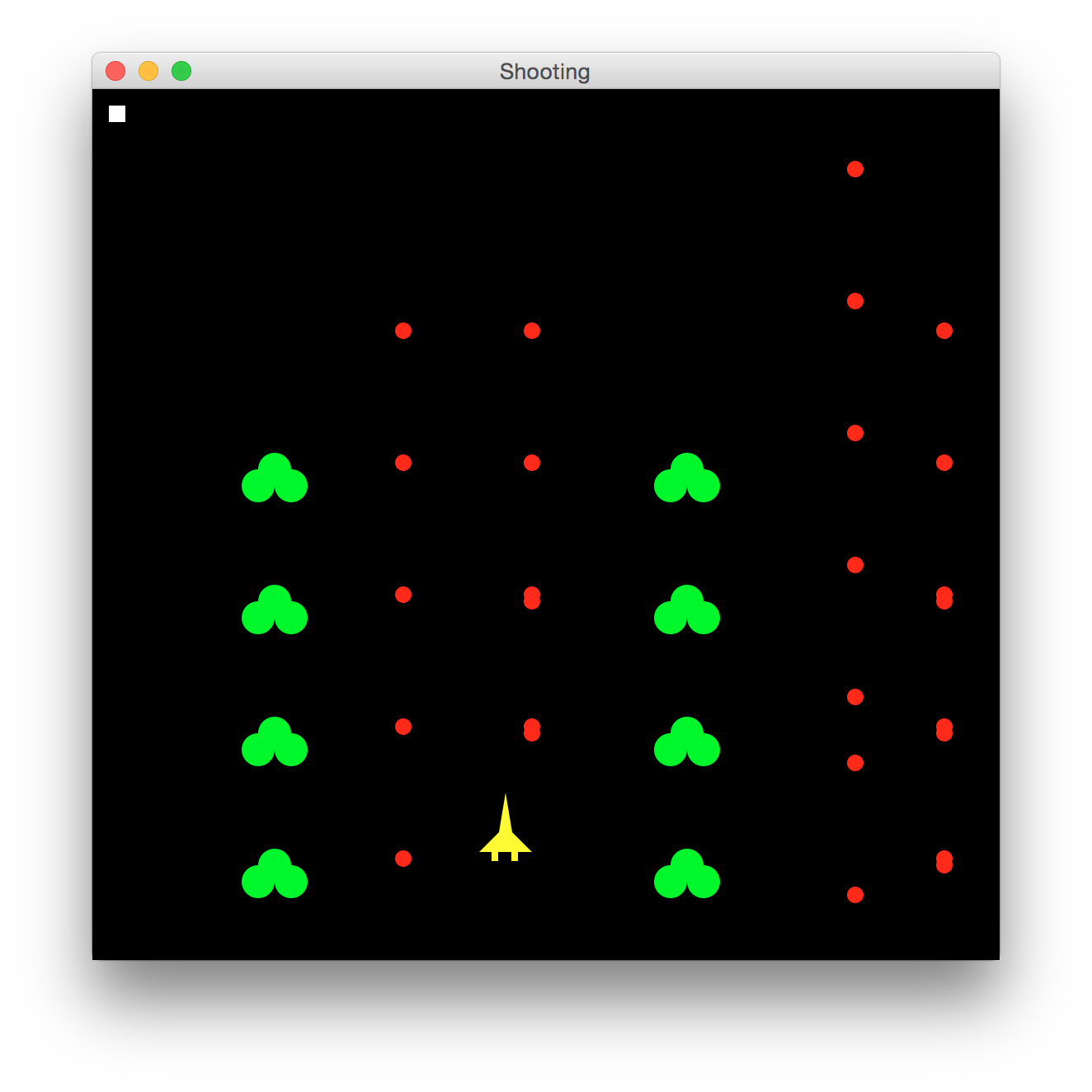
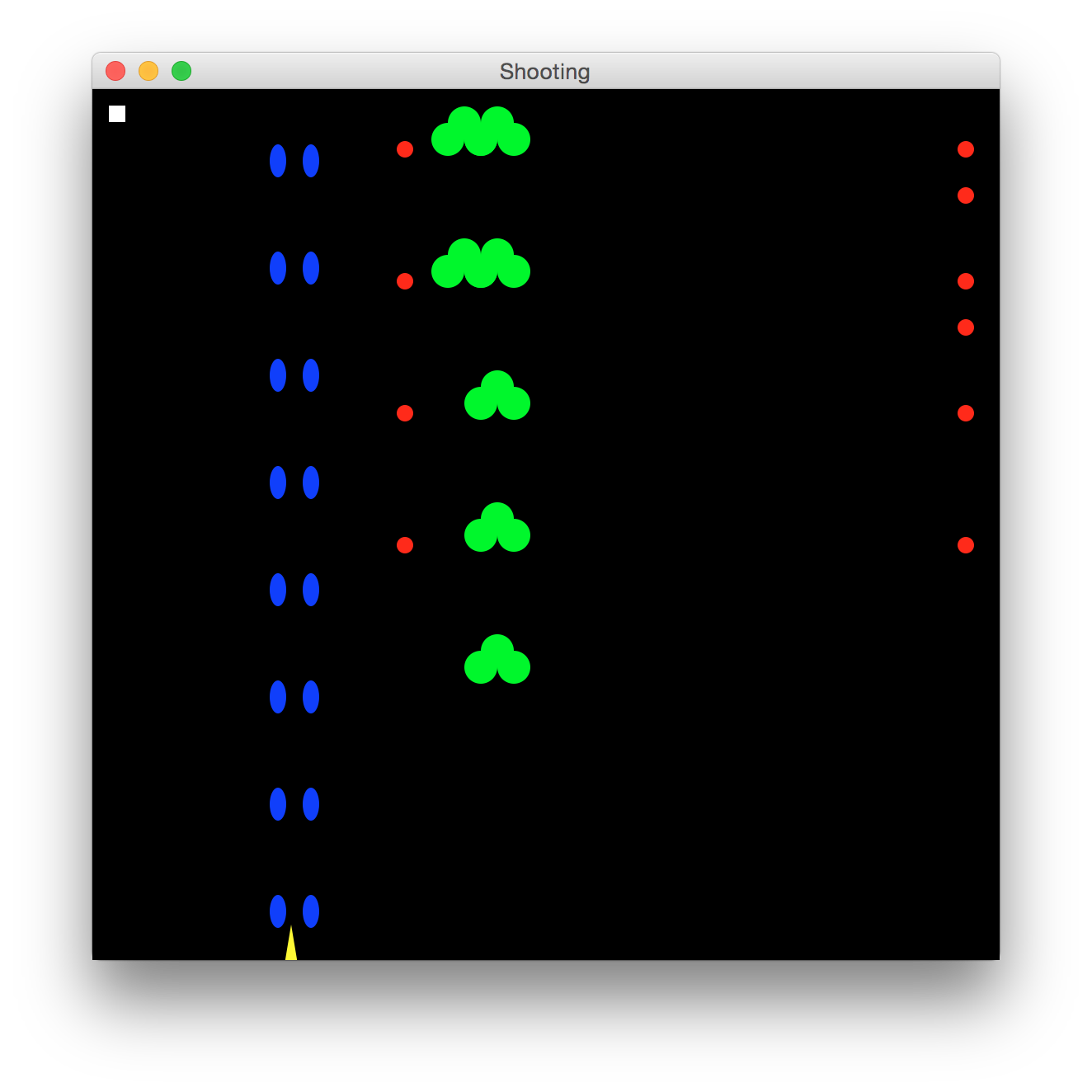


図２：ゲーム画面

　ゲーム画面に遷移したら、ゲームが開始。操作は付録１にも添付してあるように十字キーで上下左右の移動、zキーで自機弾の発射をする。ゲーム画面の右上にある白い四角は自機の残り残機で、最初は3機、0になるとゲームオーバーとなり、以下のようなゲームオーバー画面に遷移する。



図３：ゲームオーバー画面

　また、ゲームが始まってから一定時間（40秒）たったらゲームクリアとなり、ゲームクリア画面に遷移する。

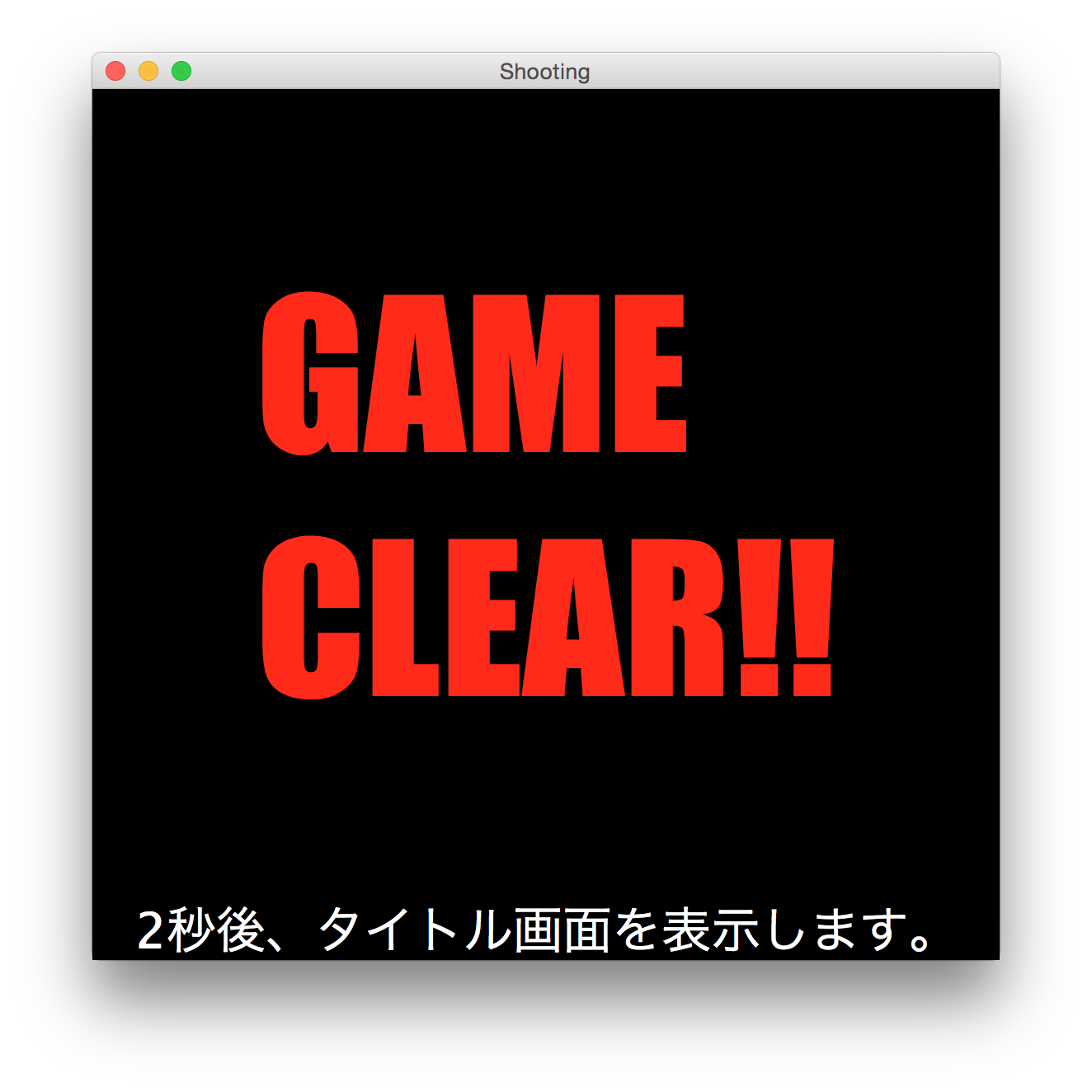


図４：ゲームクリア画面

ゲームオーバー画面、ゲームクリア画面はどちらも２秒たったら自動的に消えて、新たなタイトル画面が出てくる。

［文責：佐藤］

5.考察

6.感想

6.1飯野

まずグループ全体としては初期構想通り既存のプログラムに頼ることなく一からプログラムを組み立て、一応の形としてのゲームを完成させることが出来たのでその部分では満足しています。しかし、細かい部分を見ていくとあまり上手くいかなかったところも多く、一番大きなこととしては、段取りが悪かったと感じています。ある作業を行い始めるまでが長かったり、作業が一段落してからも次に取り掛かるのがスムーズに行かなかったりするなどいちいち作業がとまっている時間が長いように感じました。その結果として最後になって時間が押してしまったので、一応形になったとはいえ簡素なつくりになってしまい、ゲームボリュームとしても物足りないものになってしまったと思います。作業を始める前の構想の段階でもっと具体的な話を多めにして、明確な指標をもって作業を行うようにすればより円滑に作業が進み、時間に余裕もできてもっと完成度を高められたのではないかと思います。

また私個人としては、非常に要領悪く作業を行っておりグループメンバーに頼る部分が多くなってしまい、とても迷惑をかけたと思っています。私個人のスキルアップでも上の状況が少しは改善したのではと考えると、とても申し訳ないと思います。

この講義自体は上手くいけばここで終わってしまいますが、プログラミングは３年次以降も必要になることで、情報系の道では将来でも必要だと思うので何不自由なく自分の手足のようにプログラミングを動かせるようになれるよう精進していきたいと思います。

文責：飯野

6.2佐藤

(1)グループでの作業を通しての感想

　　グループ内での作業の感想として、全体的にあまり作業がはかどらなかったということである。最初、シューティングゲームを作ろうということまではすぐ決まったのだが、そこから実際にプログラムを作り出すまでに非常に時間がかかってしまった。また、プログラム書き始めても誰がどこを担当するかが大まかにしか決めていなかったため、自分の担当分が明確でないまま作業を進めてしまっていた。なので、大きなプログラムを書く際は、まず全体的にどのような構成のプログラムにするか紙に書き出し、誰がどこを担当するか明確にした上で、そのプログラムのどこまでをいつまでに終わらせるか期限をしっかり設ける必要性があったなと感じた。そうすれば自分が何をやればいいか明確になり、作業にも取り掛かりやすかったのではないかと感じている。

　また、ゲーム自体の反省点としては敵の種類や敵弾と自機弾の種類をもっと増やせればよかったなと感じている。具体的には自機の弾としてビームを出したり、敵弾の動きのパターンを多くしたり、敵として最後にボスを設定するなどができればさらに面白いゲームになったのではないかとかに感じている。

(2)今後の各自の担当分の課題とやり残したこと

　　自分の担当分において、課題として感じていることは、プログラムが非常に長くなってしまったということである。プログラム自体が非常に長くなり、それらを一つクラスにしてしまったため、読みにくいプログラムとなってしまった。よって今後自分の課題としてはJavaでプログラムを書くときはプログラムをできるだけ多くのクラスに細分化して書くように工夫したいと思った。また、機能として、経過時間やスコア、自機の残機を分かりやすく表示できればよかったなと感じた。

(3)「プログラミング演習」に関する感想と学習内容に対する感想

　　個人的にはJavaに関してもっと色々なことを知りたかったと感じている。具体的にはもっと色々なパッケージやメソッドなどの説明や、ネットワークプログラミングの説明などである。また、MVCモデルに関してもちょっと難しく、最初はよくわからなかったのでもう少し詳しく説明してもらえるとありがたかったかなと感じた。なので、難しいとは思うが もっと時間数と単位数を増やしてくれると嬉しいと思った。

[文責:佐藤]

6.3江村

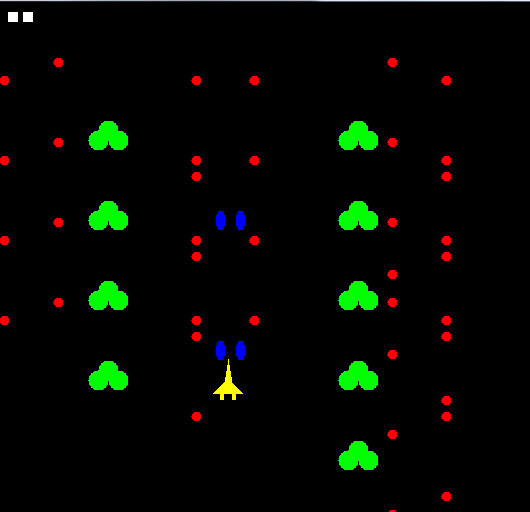
付録1、操作法マニュアル

ゲームはshooting.jarファイルにまとめられている。起動するとタイトル画面が表示される。



　　　　　　　　　　　図1、タイトル画面

タイトル画面での操作：Enterキー　：ゲーム画面に移行



③ライフ

②敵機

①自機

　　　　　　　　　　　図２、ゲーム画面

画面上青い楕円が自機の弾、赤い円が敵機の弾

自機の弾を敵機に当てると該当敵機消滅、敵機の弾が自機に触れるとライフ１減少

ゲーム画面での操作：矢印キー　：自機移動

　　　　　　　　　　　　　　　　矢印キーを入力した方向に自機が移動する。

　　　　　　　　　　Zキー　　：自機弾発射

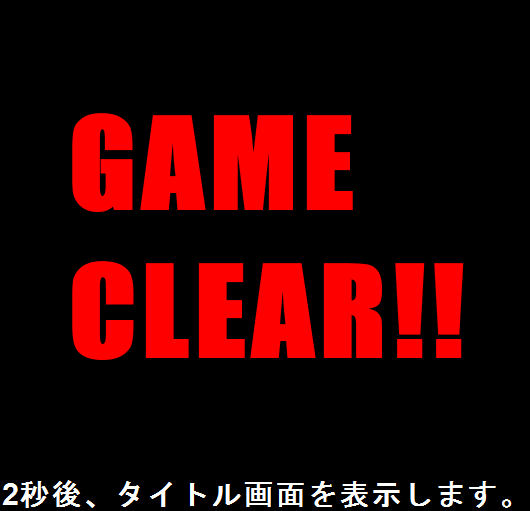
　　　　　　　　　　　　　　　　自機から上方向に弾を発射する。

ライフが０になるとゲームオーバー画面に移行、ライフを保ったまま一定時間経過でゲームクリア画面に移行



　　　　　　　　　　　図３、ゲームオーバー画面

一定時間経過後自動的にタイトル画面に移動する。



一定時間経過後自動的にタイトル画面に移行する。

文責：飯野

**付録2:プログラムリスト**

1.Model.java

1 import java.awt.\*;

2 import java.awt.event.\*;

3 import javax.swing.\*;

4 import java.util.\*;

5

6 //MVCのMの部分をつくる

7

8 /\*Figureクラス\*/

9 class Figure {

10 protected int x,y,width,height,speed; //ローカル変数定義

11 protected Color color;

12

13 public Figure(int x,int y,int w,int h,int s,Color c) {

14 this.x = x; this.y = y; // this.x, this.y はフィールド変数を指します．

15 width = w; height = h; // ローカル変数で同名の変数がある場合は，this

16 speed = s; color = c; // を付けると，フィールド変数を指すことになります．

17 }

18

19

20 /\*drawメソッド定義\*/

21 public void draw(Graphics g) {}

22 }

23

24 class Life extends Figure{ //ライフ

25 protected int count;

26

27 public Life(int x,int y, int l){

28 super(x,y,10,10,0,Color.white);

29 count = l;

30 }

31

32 public void draw(Graphics g){

33 g.setColor(color);

34 for(int i=0;i<count;i++)

35 g.fillRect(x+i\*15,y,width,height);

36 }

37 }

38

39 class Jiki extends Figure{ //自機クラス

40 public Jiki(int x,int y,int w,int h,int s,Color c){

41 super(x,y,w,h,s,c); //親コンストラクタの呼び出し

42 }

43

44

45 /\*自機の描画\*/

46 public void draw(Graphics g){

47 g.setColor(color);

48 int xPoints[] = {x,x-width/8,x-width/2,x+width/2,x+width/8};

49 int yPoints[] = {y-height/2,y,y+height/4,y+height/4,y};

50 g.fillPolygon(xPoints, yPoints,5);

51

52 g.fillRect(x+width/8,y+height/4,width/8,height/8);

53 g.fillRect(x-width/4,y+height/4,width/8,height/8);

54 }

55 }

56

57

58

59 class Enemy extends Figure{ //敵機クラス

60 /\*フィールド\*/

61 protected int type;

62 protected boolean dflag; //消滅フラグ

63 protected int shotcount;

64

65 /\*コンストラクタ\*/

66 public Enemy(int x,int y,int w,int h,int s,int t,Color c){

67 super(x,y,w,h,s,c); //親クラス呼び

68 type = t; //敵のタイプ

69 dflag = false;

70 shotcount = 0;

71 }

72

73 /\*敵機の描画\*/

74 public void draw(Graphics g){

75 g.setColor(color);

76 g.fillOval(x,y,width,height);

77 g.fillOval(x+width/2,y+height/2,width,height);

78 g.fillOval(x-width/2,y+height/2,width,height);

79 }

80

81 }

82

83 class JBullet extends Figure{ //弾クラス

84 /\*フィールド\*/

85 protected boolean dflag; //消滅フラグ

86 protected int type; //敵弾の種類

87

88 /\*コンストラクタ\*/

89 public JBullet(int x,int y,int t){

90 super(x-3,y-3,10,20,13,Color.blue);

91 type = t;

92 dflag = false;

93 }

94

95 /\*弾の描画\*/

96 public void draw(Graphics g){

97 g.setColor(color);

98 g.fillOval(x,y,width,height);

99 }

100 }

101

102

103 class EBullet extends Figure{ //弾クラス

104 /\*フィールド\*/

105 protected boolean dflag; //消滅フラグ

106 protected int type; //敵弾の種類

107

108 /\*コンストラクタ\*/

109 public EBullet(int x,int y,int t){

110 super(x-3,y-3,10,10,4,Color.red);

111 type = t;

112 dflag = false;

113 }

114

115 /\*弾の描画\*/

116 public void draw(Graphics g){

117 g.setColor(color);

118 g.fillOval(x,y,width,height);

119 }

120 }

121

122

123 /\*Modelクラス\*/

124 class Model {

125 protected Jiki jiki; //自機

126 protected Life life; //自機のライフ

127 protected ArrayList<Enemy> enemy; //敵機格納用の配列

128 protected ArrayList<JBullet> jbullet; //自機弾格納用の配列

129 protected ArrayList<EBullet> ebullet; //敵機弾格納用の配列

130 protected Enemy drawEnemy;

131 protected boolean gameover;

132

133 /\*オブジェクト\*/

134 public Model(){

135 jiki = new Jiki(200,200,32,48,10,Color.yellow); //自機の設定

136 life = new Life(10,10,3); //ライフ3からスタート

137 /\*配列格納\*/

138 enemy = new ArrayList<Enemy>(); //Enemy型の配列(敵)

139 jbullet = new ArrayList<JBullet>(); //Bullet型の配列(自機弾)

140 ebullet = new ArrayList<EBullet>(); //Bullet型の配列(敵機弾)

141 drawEnemy = null;

142 }

143

144 /\*自機関連のメソッド\*/

145 /\*自機のdrawメソッド\*/

146 public void drawJiki(Graphics g){

147 jiki.draw(g);

148 }

149 /\*自機の移動\*/

150 public void JikiMove(boolean r,boolean l,boolean d,boolean u){

151 if(r==true && jiki.x <= 540) jiki.x += jiki.speed;

152 if(l==true && jiki.x >= 10) jiki.x -= jiki.speed;

153 if(d==true && jiki.y <= 530) jiki.y += jiki.speed;

154 if(u==true && jiki.y >= 10) jiki.y -= jiki.speed;

155 }

156 public int xJiki(){return jiki.x;} //自機の中心x座標を返す

157 public int yJiki(){return jiki.y;} //自機の中心y座標を返す

158 public int wJiki(){return jiki.width;} //自機の横幅を返す

159

160 /\*敵機関連のメソッド\*/

161 /\*敵を記録している配列を返す関数\*/

162 public ArrayList<Enemy> getEnemys() {

163 return enemy;

164 }

165

166 /\*何番目かを指定して敵を取り出すメソッド\*/

167 public Enemy getEnemy(int idx) {

168 return enemy.get(idx);

169 }

170

171 public int xEnemy(int idx) {

172 return enemy.get(idx).x; //敵機の中心x座標を返す

173 }

174 public int yEnemy(int idx) {

175 return enemy.get(idx).y; //敵機の中心y座標を返す

176 }

177

178 /\*敵機の生成\*/

179 public void createEnemy(int x,int y,int type){

180 Enemy e = new Enemy(x,y,20,20,5,type,Color.green); //敵を生成して

181 enemy.add(e); //配列に格納して

182 }

183

184 /\*敵機の動作の管理\*/

185 public void EnemyMove(){

186 for(int i=0; i<enemy.size(); i++){

187 switch(enemy.get(i).type){

188

189 case 1: //敵のタイプが1のとき

190 enemy.get(i).y+=enemy.get(i).speed+1; //敵機のy座標を増やしていく

191 enemy.get(i).shotcount++; //shotcountも増やす

192

193 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

194 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

195 {

196 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

197 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

198 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

199 }

200 break;

201

202 case 2: //敵のタイプが2のとき

203 enemy.get(i).x+=enemy.get(i).speed; //敵機のx座標を増やしていく

204 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

205

206 enemy.get(i).shotcount++; //shotcountを増やす

207 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

208 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

209 {

210 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

211 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

212 }

213 break;

214

215 case 3: //敵のタイプが3のとき

216 enemy.get(i).x-=enemy.get(i).speed; //敵機のx座標を増やしていく

217 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

218

219 enemy.get(i).shotcount++; //shotcountを増やす

220 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

221 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

222 {

223 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

224 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

225 }

226 break;

227

228 }

229 }

230 }

231

232 /\*弾関連メソッド\*/

233 /\*弾を記録している配列を返す関数\*/

234 public ArrayList<EBullet> getBullets() {

235 return ebullet;

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

36 }

37 }

38

39 class Jiki extends Figure{ //自機クラス

40 public Jiki(int x,int y,int w,int h,int s,Color c){

41 super(x,y,w,h,s,c); //親コンストラクタの呼び出し

42 }

43

44

45 /\*自機の描画\*/

46 public void draw(Graphics g){

47 g.setColor(color);

48 int xPoints[] = {x,x-width/8,x-width/2,x+width/2,x+width/8};

49 int yPoints[] = {y-height/2,y,y+height/4,y+height/4,y};

50 g.fillPolygon(xPoints, yPoints,5);

51

52 g.fillRect(x+width/8,y+height/4,width/8,height/8);

53 g.fillRect(x-width/4,y+height/4,width/8,height/8);

54 }

55 }

56

57

58

59 class Enemy extends Figure{ //敵機クラス

60 /\*フィールド\*/

61 protected int type;

62 protected boolean dflag; //消滅フラグ

63 protected int shotcount;

64

65 /\*コンストラクタ\*/

66 public Enemy(int x,int y,int w,int h,int s,int t,Color c){

67 super(x,y,w,h,s,c); //親クラス呼び

68 type = t; //敵のタイプ

69 dflag = false;

70 shotcount = 0;

71 }

72

73 /\*敵機の描画\*/

74 public void draw(Graphics g){

75 g.setColor(color);

76 g.fillOval(x,y,width,height);

77 g.fillOval(x+width/2,y+height/2,width,height);

78 g.fillOval(x-width/2,y+height/2,width,height);

79 }

80

81 }

82

83 class JBullet extends Figure{ //弾クラス

84 /\*フィールド\*/

85 protected boolean dflag; //消滅フラグ

86 protected int type; //敵弾の種類

87

88 /\*コンストラクタ\*/

89 public JBullet(int x,int y,int t){

90 super(x-3,y-3,10,20,13,Color.blue);

91 type = t;

92 dflag = false;

93 }

94

95 /\*弾の描画\*/

96 public void draw(Graphics g){

97 g.setColor(color);

98 g.fillOval(x,y,width,height);

99 }

100 }

101

102

103 class EBullet extends Figure{ //弾クラス

104 /\*フィールド\*/

105 protected boolean dflag; //消滅フラグ

106 protected int type; //敵弾の種類

107

108 /\*コンストラクタ\*/

109 public EBullet(int x,int y,int t){

110 super(x-3,y-3,10,10,4,Color.red);

111 type = t;

112 dflag = false;

113 }

114

115 /\*弾の描画\*/

116 public void draw(Graphics g){

117 g.setColor(color);

118 g.fillOval(x,y,width,height);

119 }

120 }

121

122

123 /\*Modelクラス\*/

124 class Model {

125 protected Jiki jiki; //自機

126 protected Life life; //自機のライフ

127 protected ArrayList<Enemy> enemy; //敵機格納用の配列

128 protected ArrayList<JBullet> jbullet; //自機弾格納用の配列

129 protected ArrayList<EBullet> ebullet; //敵機弾格納用の配列

130 protected Enemy drawEnemy;

131 protected boolean gameover;

132

133 /\*オブジェクト\*/

134 public Model(){

135 jiki = new Jiki(200,200,32,48,10,Color.yellow); //自機の設定

136 life = new Life(10,10,3); //ライフ3からスタート

137 /\*配列格納\*/

138 enemy = new ArrayList<Enemy>(); //Enemy型の配列(敵)

139 jbullet = new ArrayList<JBullet>(); //Bullet型の配列(自機弾)

140 ebullet = new ArrayList<EBullet>(); //Bullet型の配列(敵機弾)

141 drawEnemy = null;

142 }

143

144 /\*自機関連のメソッド\*/

145 /\*自機のdrawメソッド\*/

146 public void drawJiki(Graphics g){

147 jiki.draw(g);

148 }

149 /\*自機の移動\*/

150 public void JikiMove(boolean r,boolean l,boolean d,boolean u){

151 if(r==true && jiki.x <= 540) jiki.x += jiki.speed;

152 if(l==true && jiki.x >= 10) jiki.x -= jiki.speed;

153 if(d==true && jiki.y <= 530) jiki.y += jiki.speed;

154 if(u==true && jiki.y >= 10) jiki.y -= jiki.speed;

155 }

156 public int xJiki(){return jiki.x;} //自機の中心x座標を返す

157 public int yJiki(){return jiki.y;} //自機の中心y座標を返す

158 public int wJiki(){return jiki.width;} //自機の横幅を返す

159

160 /\*敵機関連のメソッド\*/

161 /\*敵を記録している配列を返す関数\*/

162 public ArrayList<Enemy> getEnemys() {

163 return enemy;

164 }

165

166 /\*何番目かを指定して敵を取り出すメソッド\*/

167 public Enemy getEnemy(int idx) {

168 return enemy.get(idx);

169 }

170

171 public int xEnemy(int idx) {

172 return enemy.get(idx).x; //敵機の中心x座標を返す

173 }

174 public int yEnemy(int idx) {

175 return enemy.get(idx).y; //敵機の中心y座標を返す

176 }

177

178 /\*敵機の生成\*/

179 public void createEnemy(int x,int y,int type){

180 Enemy e = new Enemy(x,y,20,20,5,type,Color.green); //敵を生成して

181 enemy.add(e); //配列に格納して

182 }

183

184 /\*敵機の動作の管理\*/

185 public void EnemyMove(){

186 for(int i=0; i<enemy.size(); i++){

187 switch(enemy.get(i).type){

188

189 case 1: //敵のタイプが1のとき

190 enemy.get(i).y+=enemy.get(i).speed+1; //敵機のy座標を増やしていく

191 enemy.get(i).shotcount++; //shotcountも増やす

192

193 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

194 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

195 {

196 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

197 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

198 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

199 }

200 break;

201

202 case 2: //敵のタイプが2のとき

203 enemy.get(i).x+=enemy.get(i).speed; //敵機のx座標を増やしていく

204 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

205

206 enemy.get(i).shotcount++; //shotcountを増やす

207 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

208 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

209 {

210 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

211 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

212 }

213 break;

214

215 case 3: //敵のタイプが3のとき

216 enemy.get(i).x-=enemy.get(i).speed; //敵機のx座標を増やしていく

217 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

218

219 enemy.get(i).shotcount++; //shotcountを増やす

220 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

221 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

222 {

223 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

224 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

225 }

226 break;

227

228 }

229 }

230 }

231

232 /\*弾関連メソッド\*/

233 /\*弾を記録している配列を返す関数\*/

234 public ArrayList<EBullet> getBullets() {

235 return ebullet;

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

76 g.fillOval(x,y,width,height);

77 g.fillOval(x+width/2,y+height/2,width,height);

78 g.fillOval(x-width/2,y+height/2,width,height);

79 }

80

81 }

82

83 class JBullet extends Figure{ //弾クラス

84 /\*フィールド\*/

85 protected boolean dflag; //消滅フラグ

86 protected int type; //敵弾の種類

87

88 /\*コンストラクタ\*/

89 public JBullet(int x,int y,int t){

90 super(x-3,y-3,10,20,13,Color.blue);

91 type = t;

92 dflag = false;

93 }

94

95 /\*弾の描画\*/

96 public void draw(Graphics g){

97 g.setColor(color);

98 g.fillOval(x,y,width,height);

99 }

100 }

101

102

103 class EBullet extends Figure{ //弾クラス

104 /\*フィールド\*/

105 protected boolean dflag; //消滅フラグ

106 protected int type; //敵弾の種類

107

108 /\*コンストラクタ\*/

109 public EBullet(int x,int y,int t){

110 super(x-3,y-3,10,10,4,Color.red);

111 type = t;

112 dflag = false;

113 }

114

115 /\*弾の描画\*/

116 public void draw(Graphics g){

117 g.setColor(color);

118 g.fillOval(x,y,width,height);

119 }

120 }

121

122

123 /\*Modelクラス\*/

124 class Model {

125 protected Jiki jiki; //自機

126 protected Life life; //自機のライフ

127 protected ArrayList<Enemy> enemy; //敵機格納用の配列

128 protected ArrayList<JBullet> jbullet; //自機弾格納用の配列

129 protected ArrayList<EBullet> ebullet; //敵機弾格納用の配列

130 protected Enemy drawEnemy;

131 protected boolean gameover;

132

133 /\*オブジェクト\*/

134 public Model(){

135 jiki = new Jiki(200,200,32,48,10,Color.yellow); //自機の設定

136 life = new Life(10,10,3); //ライフ3からスタート

137 /\*配列格納\*/

138 enemy = new ArrayList<Enemy>(); //Enemy型の配列(敵)

139 jbullet = new ArrayList<JBullet>(); //Bullet型の配列(自機弾)

140 ebullet = new ArrayList<EBullet>(); //Bullet型の配列(敵機弾)

141 drawEnemy = null;

142 }

143

144 /\*自機関連のメソッド\*/

145 /\*自機のdrawメソッド\*/

146 public void drawJiki(Graphics g){

147 jiki.draw(g);

148 }

149 /\*自機の移動\*/

150 public void JikiMove(boolean r,boolean l,boolean d,boolean u){

151 if(r==true && jiki.x <= 540) jiki.x += jiki.speed;

152 if(l==true && jiki.x >= 10) jiki.x -= jiki.speed;

153 if(d==true && jiki.y <= 530) jiki.y += jiki.speed;

154 if(u==true && jiki.y >= 10) jiki.y -= jiki.speed;

155 }

156 public int xJiki(){return jiki.x;} //自機の中心x座標を返す

157 public int yJiki(){return jiki.y;} //自機の中心y座標を返す

158 public int wJiki(){return jiki.width;} //自機の横幅を返す

159

160 /\*敵機関連のメソッド\*/

161 /\*敵を記録している配列を返す関数\*/

162 public ArrayList<Enemy> getEnemys() {

163 return enemy;

164 }

165

166 /\*何番目かを指定して敵を取り出すメソッド\*/

167 public Enemy getEnemy(int idx) {

168 return enemy.get(idx);

169 }

170

171 public int xEnemy(int idx) {

172 return enemy.get(idx).x; //敵機の中心x座標を返す

173 }

174 public int yEnemy(int idx) {

175 return enemy.get(idx).y; //敵機の中心y座標を返す

176 }

177

178 /\*敵機の生成\*/

179 public void createEnemy(int x,int y,int type){

180 Enemy e = new Enemy(x,y,20,20,5,type,Color.green); //敵を生成して

181 enemy.add(e); //配列に格納して

182 }

183

184 /\*敵機の動作の管理\*/

185 public void EnemyMove(){

186 for(int i=0; i<enemy.size(); i++){

187 switch(enemy.get(i).type){

188

189 case 1: //敵のタイプが1のとき

190 enemy.get(i).y+=enemy.get(i).speed+1; //敵機のy座標を増やしていく

191 enemy.get(i).shotcount++; //shotcountも増やす

192

193 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

194 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

195 {

196 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

197 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

198 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

199 }

200 break;

201

202 case 2: //敵のタイプが2のとき

203 enemy.get(i).x+=enemy.get(i).speed; //敵機のx座標を増やしていく

204 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

205

206 enemy.get(i).shotcount++; //shotcountを増やす

207 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

208 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

209 {

210 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

211 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

212 }

213 break;

214

215 case 3: //敵のタイプが3のとき

216 enemy.get(i).x-=enemy.get(i).speed; //敵機のx座標を増やしていく

217 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

218

219 enemy.get(i).shotcount++; //shotcountを増やす

220 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

221 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

222 {

223 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

224 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

225 }

226 break;

227

228 }

229 }

230 }

231

232 /\*弾関連メソッド\*/

233 /\*弾を記録している配列を返す関数\*/

234 public ArrayList<EBullet> getBullets() {

235 return ebullet;

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

116 public void draw(Graphics g){

117 g.setColor(color);

118 g.fillOval(x,y,width,height);

119 }

120 }

121

122

123 /\*Modelクラス\*/

124 class Model {

125 protected Jiki jiki; //自機

126 protected Life life; //自機のライフ

127 protected ArrayList<Enemy> enemy; //敵機格納用の配列

128 protected ArrayList<JBullet> jbullet; //自機弾格納用の配列

129 protected ArrayList<EBullet> ebullet; //敵機弾格納用の配列

130 protected Enemy drawEnemy;

131 protected boolean gameover;

132

133 /\*オブジェクト\*/

134 public Model(){

135 jiki = new Jiki(200,200,32,48,10,Color.yellow); //自機の設定

136 life = new Life(10,10,3); //ライフ3からスタート

137 /\*配列格納\*/

138 enemy = new ArrayList<Enemy>(); //Enemy型の配列(敵)

139 jbullet = new ArrayList<JBullet>(); //Bullet型の配列(自機弾)

140 ebullet = new ArrayList<EBullet>(); //Bullet型の配列(敵機弾)

141 drawEnemy = null;

142 }

143

144 /\*自機関連のメソッド\*/

145 /\*自機のdrawメソッド\*/

146 public void drawJiki(Graphics g){

147 jiki.draw(g);

148 }

149 /\*自機の移動\*/

150 public void JikiMove(boolean r,boolean l,boolean d,boolean u){

151 if(r==true && jiki.x <= 540) jiki.x += jiki.speed;

152 if(l==true && jiki.x >= 10) jiki.x -= jiki.speed;

153 if(d==true && jiki.y <= 530) jiki.y += jiki.speed;

154 if(u==true && jiki.y >= 10) jiki.y -= jiki.speed;

155 }

156 public int xJiki(){return jiki.x;} //自機の中心x座標を返す

157 public int yJiki(){return jiki.y;} //自機の中心y座標を返す

158 public int wJiki(){return jiki.width;} //自機の横幅を返す

159

160 /\*敵機関連のメソッド\*/

161 /\*敵を記録している配列を返す関数\*/

162 public ArrayList<Enemy> getEnemys() {

163 return enemy;

164 }

165

166 /\*何番目かを指定して敵を取り出すメソッド\*/

167 public Enemy getEnemy(int idx) {

168 return enemy.get(idx);

169 }

170

171 public int xEnemy(int idx) {

172 return enemy.get(idx).x; //敵機の中心x座標を返す

173 }

174 public int yEnemy(int idx) {

175 return enemy.get(idx).y; //敵機の中心y座標を返す

176 }

177

178 /\*敵機の生成\*/

179 public void createEnemy(int x,int y,int type){

180 Enemy e = new Enemy(x,y,20,20,5,type,Color.green); //敵を生成して

181 enemy.add(e); //配列に格納して

182 }

183

184 /\*敵機の動作の管理\*/

185 public void EnemyMove(){

186 for(int i=0; i<enemy.size(); i++){

187 switch(enemy.get(i).type){

188

189 case 1: //敵のタイプが1のとき

190 enemy.get(i).y+=enemy.get(i).speed+1; //敵機のy座標を増やしていく

191 enemy.get(i).shotcount++; //shotcountも増やす

192

193 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

194 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

195 {

196 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

197 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

198 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

199 }

200 break;

201

202 case 2: //敵のタイプが2のとき

203 enemy.get(i).x+=enemy.get(i).speed; //敵機のx座標を増やしていく

204 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

205

206 enemy.get(i).shotcount++; //shotcountを増やす

207 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

208 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

209 {

210 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

211 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

212 }

213 break;

214

215 case 3: //敵のタイプが3のとき

216 enemy.get(i).x-=enemy.get(i).speed; //敵機のx座標を増やしていく

217 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

218

219 enemy.get(i).shotcount++; //shotcountを増やす

220 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

221 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

222 {

223 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

224 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

225 }

226 break;

227

228 }

229 }

230 }

231

232 /\*弾関連メソッド\*/

233 /\*弾を記録している配列を返す関数\*/

234 public ArrayList<EBullet> getBullets() {

235 return ebullet;

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

156 public int xJiki(){return jiki.x;} //自機の中心x座標を返す

157 public int yJiki(){return jiki.y;} //自機の中心y座標を返す

158 public int wJiki(){return jiki.width;} //自機の横幅を返す

159

160 /\*敵機関連のメソッド\*/

161 /\*敵を記録している配列を返す関数\*/

162 public ArrayList<Enemy> getEnemys() {

163 return enemy;

164 }

165

166 /\*何番目かを指定して敵を取り出すメソッド\*/

167 public Enemy getEnemy(int idx) {

168 return enemy.get(idx);

169 }

170

171 public int xEnemy(int idx) {

172 return enemy.get(idx).x; //敵機の中心x座標を返す

173 }

174 public int yEnemy(int idx) {

175 return enemy.get(idx).y; //敵機の中心y座標を返す

176 }

177

178 /\*敵機の生成\*/

179 public void createEnemy(int x,int y,int type){

180 Enemy e = new Enemy(x,y,20,20,5,type,Color.green); //敵を生成して

181 enemy.add(e); //配列に格納して

182 }

183

184 /\*敵機の動作の管理\*/

185 public void EnemyMove(){

186 for(int i=0; i<enemy.size(); i++){

187 switch(enemy.get(i).type){

188

189 case 1: //敵のタイプが1のとき

190 enemy.get(i).y+=enemy.get(i).speed+1; //敵機のy座標を増やしていく

191 enemy.get(i).shotcount++; //shotcountも増やす

192

193 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

194 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

195 {

196 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

197 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

198 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

199 }

200 break;

201

202 case 2: //敵のタイプが2のとき

203 enemy.get(i).x+=enemy.get(i).speed; //敵機のx座標を増やしていく

204 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

205

206 enemy.get(i).shotcount++; //shotcountを増やす

207 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

208 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

209 {

210 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

211 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

212 }

213 break;

214

215 case 3: //敵のタイプが3のとき

216 enemy.get(i).x-=enemy.get(i).speed; //敵機のx座標を増やしていく

217 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

218

219 enemy.get(i).shotcount++; //shotcountを増やす

220 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

221 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

222 {

223 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

224 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

225 }

226 break;

227

228 }

229 }

230 }

231

232 /\*弾関連メソッド\*/

233 /\*弾を記録している配列を返す関数\*/

234 public ArrayList<EBullet> getBullets() {

235 return ebullet;

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

196 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

197 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

198 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

199 }

200 break;

201

202 case 2: //敵のタイプが2のとき

203 enemy.get(i).x+=enemy.get(i).speed; //敵機のx座標を増やしていく

204 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

205

206 enemy.get(i).shotcount++; //shotcountを増やす

207 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

208 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

209 {

210 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

211 this.createBullet(this.xEnemy(i),this.yEnemy(i),12);

212 }

213 break;

214

215 case 3: //敵のタイプが3のとき

216 enemy.get(i).x-=enemy.get(i).speed; //敵機のx座標を増やしていく

217 enemy.get(i).y+=enemy.get(i).speed\*2; //敵機のy座標も増やしていく

218

219 enemy.get(i).shotcount++; //shotcountを増やす

220 if(enemy.get(i).shotcount%50==0||enemy.get(i).shotcount==1)

221 //shotcountが50で割り切れるようになった時、敵機の場所に弾生成

222 {

223 this.createBullet(this.xEnemy(i),this.yEnemy(i),10);

224 this.createBullet(this.xEnemy(i),this.yEnemy(i),11);

225 }

226 break;

227

228 }

229 }

230 }

231

232 /\*弾関連メソッド\*/

233 /\*弾を記録している配列を返す関数\*/

234 public ArrayList<EBullet> getBullets() {

235 return ebullet;

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

236 }

237 public ArrayList<JBullet> getjBullets() {

238 return jbullet;

239 }

240

241 /\*弾の動作の管理\*/

242 public void BulletMove(){

243 for(int i=0; i<jbullet.size(); i++){

244 switch(jbullet.get(i).type){

245 case 1:

246 jbullet.get(i).y -= jbullet.get(i).speed;

247 break;

248

249 case 2:

250 jbullet.get(i).y -= jbullet.get(i).speed;

251 jbullet.get(i).x += jbullet.get(i).speed;

252 break;

253

254 case 3:

255 jbullet.get(i).y -= jbullet.get(i).speed;

256 jbullet.get(i).x -= jbullet.get(i).speed;

257 break;

258

259 }

260 }

261 for(int i=0; i<ebullet.size(); i++){

262 switch(ebullet.get(i).type){

263 case 10:

264 ebullet.get(i).x += ebullet.get(i).speed;

265 ebullet.get(i).y += ebullet.get(i).speed\*2;

266 break;

267

268 case 11:

269 ebullet.get(i).x += ebullet.get(i).speed;

270 ebullet.get(i).y += ebullet.get(i).speed;

271 break;

272

273 case 12:

274 ebullet.get(i).x -= ebullet.get(i).speed;

275 ebullet.get(i).y += ebullet.get(i).speed;

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

276 break;

277 }

278 }

279 }

280

281

282 /\*弾の生成\*/

283 public void createBullet(int x, int y, int t){

284 switch(t){

285 case 1:

286 JBullet jb1 = new JBullet(x,y,t);

287 jbullet.add(jb1);

288 break;

289

290 case 2:

291 JBullet jb2 = new JBullet(x,y,t);

292 jbullet.add(jb2);

293 break;

294

295 case 3:

296 JBullet jb3 = new JBullet(x,y,t);

297 jbullet.add(jb3);

298 break;

299

300

301

302 case 10:

303 EBullet eb10 = new EBullet(x,y,t);

304 ebullet.add(eb10);

305 break;

306

307 case 11:

308 EBullet eb11 = new EBullet(x,y,t);

309 ebullet.add(eb11);

310 break;

311

312 case 12:

313 EBullet eb12 = new EBullet(x,y,t);

314 ebullet.add(eb12);

315 break;

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

316 }

317 }

318

319 /\*その他\*/

320 /\*衝突時の処理\*/

321 public void HitCheck(){

322 for(int i=0; i<enemy.size(); i++){

323 for(int j=0; j<jbullet.size(); j++){

324 if(HitBox(enemy.get(i).x,enemy.get(i).y,

325 enemy.get(i).width,enemy.get(i).height,

326 jbullet.get(j).x,jbullet.get(j).y,

327 jbullet.get(j).width,jbullet.get(j).height)

328 == true){

329 jbullet.get(j).dflag = true; //敵機に当たった自機弾の消滅フラグを立てる

330 enemy.get(i).dflag = true; //自機弾に当たった敵機の消滅フラグを立てる

331 }

332 }

333 }

334 for(int i=0; i<ebullet.size(); i++){

335 if(HitBox(jiki.x,jiki.y,

336 jiki.width,jiki.height,

337 ebullet.get(i).x,ebullet.get(i).y,

338 ebullet.get(i).width,ebullet.get(i).height)

339 == true){

340 ebullet.get(i).dflag = true; //自機に当たった敵機弾の消滅フラグを立てる

341 life.count--; //ライフの減少

342 }

343 }

344 for(int i=0; i<enemy.size(); i++){

345 if(HitBox(jiki.x,jiki.y,

346 jiki.width,jiki.height,

347 enemy.get(i).x,enemy.get(i).y,

348 enemy.get(i).width,enemy.get(i).height)

349 == true){

350 enemy.get(i).dflag = true; //自機に当たった敵機の消滅フラグを立てる

351 life.count--; //ライフの減少

352 GameOver();

353 }

354 }

355 }

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

356

357 /\*矩形当たり判定\*/

358 public boolean HitBox(int x1,int y1,int w1,int h1,

359 int x2,int y2,int w2,int h2){

360 if((x1 < x2 + w2) && (x2 < x1 + w1) &&

361 (y1 < y2 + h2) && (y2 < y1 + h1))

362 return true;

363 else

364 return false;

365 }

366

367

368 /\*消滅の処理\*/

369 public void DeleteFigure(){

370 for(int i=enemy.size()-1; i>=0; i--)

371 if(enemy.get(i).dflag==true)

372 enemy.remove(i);

373

374 for(int i=ebullet.size()-1; i>=0; i--)

375 if(ebullet.get(i).dflag==true)

376 ebullet.remove(i);

377

378 for(int i=jbullet.size()-1; i>=0; i--)

379 if(jbullet.get(i).dflag==true||jbullet.get(i).y<=0)

380 jbullet.remove(i);

381 }

382

383 /\*ゲームオーバー\*/

384 public boolean GameOver()

385 {

386 if(life.count < 1)

387 {

388 return true;

389 }

390 else

391 return false;

392 }

393

394 /\*ライフの初期化\*/

395 public void LifeInit(){

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

396 life.count = 3;

397 }

398 /\*ライフの描画\*/

399 public void drawLife(Graphics g){

400 life.draw(g);

401 }

402 }

403

404

2.STG.java

1 import java.awt.\*;

2 import java.awt.event.\*;

3 import javax.swing.\*;

4 import java.util.\*;

5

6 /\*ゲーム画面に関するクラス\*/

7 class CharactorController extends JPanel implements ActionListener

8 {

9 /\*フィールド\*/

10 private Model model; //Modelのデータを取り込む

11 private ArrayList<Enemy> enemy; //敵格納配列定義

12 private ArrayList<EBullet> ebullet; //敵弾格納用配列定義

13 private ArrayList<JBullet> jbullet; //自機弾格納用配列定義

14 private javax.swing.Timer timer; //timer(敵機の動作に関するtimer)

15 public int time =0; //timerを以下でtimeに換算。初期値は0

16 protected int i = 0;

17

18 boolean jshotflag = false;

19 boolean jrflag = false; //右ボタンに関するフラグ

20 boolean jlflag = false; //左ボタンに関するフラグ

21 boolean jdflag = false; //下ボタンに関するフラグ

22 boolean juflag = false; //上ボタンに関するフラグ

23 boolean flag\_timer = false; //timerに関するフラグ

24 boolean gameover = false; //gameoverか否か判定するフラグ

25

26

27

28 /\*コンストラクタ\*/

29 public CharactorController()

30 {

31 timer = new javax.swing.Timer(50,this); //timerにtimer挿入(0.05秒おき)

32

33 model = new Model(); //modelオブジェクトの生成

34

35 this.setBackground(Color.BLACK); //背景設定

36 this.setFocusable(true); //キー入力をGUI部品が受け付ける

37

38 }

39

40 /\*timerをスタートさせるメソッド\*/

41 public void timerstart()

42 {

43 timer.start(); //timerをスタート

44 }

45

46 /\*timerを止めるメソッド\*/

47 public void timerstop()

48 {

49 timer.stop();

50 }

51

52

53 /\*描画\*/

54 protected void paintComponent(Graphics g)

55 {

56 super.paintComponent(g);

57

58 /\*自機の描画\*/

59 model.drawJiki(g);

60

61 /\*ライフの描画\*/

62 model.drawLife(g);

63

64 enemy = model.getEnemys();

65 /\*敵機の描画\*/

66 for(Enemy e:enemy){

67 e.draw(g);

68 }

69

70 ebullet = model.getBullets();

71 /\*敵弾の描画\*/

72 for(EBullet eb:ebullet){

73 eb.draw(g);

74 }

75

76 jbullet = model.getjBullets();

77 /\*自機の弾の描画\*/

78 for(JBullet jb:jbullet){

79 jb.draw(g);

80 }

81

82 }

83

84

85 /\*timerに対するactionPerformed(敵の生成)\*/

86 public void actionPerformed(ActionEvent e)

87 {

88 time++; //timer1を0.1秒に換算

89

90 model.JikiMove(jrflag,jlflag,jdflag,juflag);

91 if(jshotflag==true && time%5==0)

92 {

93 model.createBullet(model.xJiki()+model.wJiki()/3,model.yJiki(),1); //自機弾の発射

94 model.createBullet(model.xJiki()-model.wJiki()/3,model.yJiki(),1); //自機弾の発射

95 }

96

97 Game();

98

99 /\*敵機の出現\*/

100 if(time == 10) //1秒たったら

101 {

102 for(int i=-320;i<=0;i+=80)

103 model.createEnemy(100,i,1);

104

105 for(int i=-320;i<=0;i+=80)

106 model.createEnemy(350,i,1);

107 }

108 if(time == 150)

109 {

110 for(int i=-320;i<=0;i+=80)

111 model.createEnemy(50,i,2);

112

113 for(int i=-320;i<=0;i+=80)

114 model.createEnemy(400,i,3);

115 }

116 if(time == 200)

117 for(int i=-160;i<=0;i+=80)

118 model.createEnemy(300,i,1);

119 if(time == 230)

120 for(int i=-160;i<=0;i+=80)

121 model.createEnemy(150,i,1);

122 if(time == 260)

123 for(int i=-160;i<=0;i+=80)

124 model.createEnemy(350,i,3);

125 if(time == 290)

126 for(int i=-160;i<=0;i+=80)

127 model.createEnemy(100,i,2);

128 if(time == 360)

129 for(int i=-160;i<=0;i+=80)

130 model.createEnemy(270,i,1);

131

132

133

134

135 model.EnemyMove();

136 model.BulletMove();

137 model.HitCheck();

138 model.DeleteFigure();

139

140 repaint(); //そして再描画

141

142 }

143

144 /\*GameOver画面に遷移するためのメソッド\*/

145 public int Game()

146 {

147 if(model.GameOver() == true && i==0)

148 {

149 i++;

150 }

151 else if(time == 400)

152 {

153 i++;

154 i++;

155 }

156 return i;

157 }

158

159 }

160

161

162

163

29 public CharactorController()

30 {

31 timer = new javax.swing.Timer(50,this); //timerにtimer挿入(0.05秒おき)

32

33 model = new Model(); //modelオブジェクトの生成

34

35 this.setBackground(Color.BLACK); //背景設定

36 this.setFocusable(true); //キー入力をGUI部品が受け付ける

37

38 }

39

40 /\*timerをスタートさせるメソッド\*/

41 public void timerstart()

42 {

43 timer.start(); //timerをスタート

44 }

45

46 /\*timerを止めるメソッド\*/

47 public void timerstop()

48 {

49 timer.stop();

50 }

51

52

53 /\*描画\*/

54 protected void paintComponent(Graphics g)

55 {

56 super.paintComponent(g);

57

58 /\*自機の描画\*/

59 model.drawJiki(g);

60

61 /\*ライフの描画\*/

62 model.drawLife(g);

63

64 enemy = model.getEnemys();

65 /\*敵機の描画\*/

66 for(Enemy e:enemy){

67 e.draw(g);

68 }

69

70 ebullet = model.getBullets();

71 /\*敵弾の描画\*/

72 for(EBullet eb:ebullet){

73 eb.draw(g);

74 }

75

76 jbullet = model.getjBullets();

77 /\*自機の弾の描画\*/

78 for(JBullet jb:jbullet){

79 jb.draw(g);

80 }

81

82 }

83

84

85 /\*timerに対するactionPerformed(敵の生成)\*/

86 public void actionPerformed(ActionEvent e)

87 {

88 time++; //timer1を0.1秒に換算

89

90 model.JikiMove(jrflag,jlflag,jdflag,juflag);

91 if(jshotflag==true && time%5==0)

92 {

93 model.createBullet(model.xJiki()+model.wJiki()/3,model.yJiki(),1); //自機弾の発射

94 model.createBullet(model.xJiki()-model.wJiki()/3,model.yJiki(),1); //自機弾の発射

95 }

96

97 Game();

98

99 /\*敵機の出現\*/

100 if(time == 10) //1秒たったら

101 {

102 for(int i=-320;i<=0;i+=80)

103 model.createEnemy(100,i,1);

104

105 for(int i=-320;i<=0;i+=80)

106 model.createEnemy(350,i,1);

107 }

108 if(time == 150)

109 {

110 for(int i=-320;i<=0;i+=80)

111 model.createEnemy(50,i,2);

112

113 for(int i=-320;i<=0;i+=80)

114 model.createEnemy(400,i,3);

115 }

116 if(time == 200)

117 for(int i=-160;i<=0;i+=80)

118 model.createEnemy(300,i,1);

119 if(time == 230)

120 for(int i=-160;i<=0;i+=80)

121 model.createEnemy(150,i,1);

122 if(time == 260)

123 for(int i=-160;i<=0;i+=80)

124 model.createEnemy(350,i,3);

125 if(time == 290)

126 for(int i=-160;i<=0;i+=80)

127 model.createEnemy(100,i,2);

128 if(time == 360)

129 for(int i=-160;i<=0;i+=80)

130 model.createEnemy(270,i,1);

131

132

133

134

135 model.EnemyMove();

136 model.BulletMove();

137 model.HitCheck();

138 model.DeleteFigure();

139

140 repaint(); //そして再描画

141

142 }

143

144 /\*GameOver画面に遷移するためのメソッド\*/

145 public int Game()

146 {

147 if(model.GameOver() == true && i==0)

148 {

149 i++;

150 }

151 else if(time == 400)

152 {

153 i++;

154 i++;

155 }

156 return i;

157 }

158

159 }

160

161

162

163

68 }

69

70 ebullet = model.getBullets();

71 /\*敵弾の描画\*/

72 for(EBullet eb:ebullet){

73 eb.draw(g);

74 }

75

76 jbullet = model.getjBullets();

77 /\*自機の弾の描画\*/

78 for(JBullet jb:jbullet){

79 jb.draw(g);

80 }

81

82 }

83

84

85 /\*timerに対するactionPerformed(敵の生成)\*/

86 public void actionPerformed(ActionEvent e)

87 {

88 time++; //timer1を0.1秒に換算

89

90 model.JikiMove(jrflag,jlflag,jdflag,juflag);

91 if(jshotflag==true && time%5==0)

92 {

93 model.createBullet(model.xJiki()+model.wJiki()/3,model.yJiki(),1); //自機弾の発射

94 model.createBullet(model.xJiki()-model.wJiki()/3,model.yJiki(),1); //自機弾の発射

95 }

96

97 Game();

98

99 /\*敵機の出現\*/

100 if(time == 10) //1秒たったら

101 {

102 for(int i=-320;i<=0;i+=80)

103 model.createEnemy(100,i,1);

104

105 for(int i=-320;i<=0;i+=80)

106 model.createEnemy(350,i,1);

107 }

108 if(time == 150)

109 {

110 for(int i=-320;i<=0;i+=80)

111 model.createEnemy(50,i,2);

112

113 for(int i=-320;i<=0;i+=80)

114 model.createEnemy(400,i,3);

115 }

116 if(time == 200)

117 for(int i=-160;i<=0;i+=80)

118 model.createEnemy(300,i,1);

119 if(time == 230)

120 for(int i=-160;i<=0;i+=80)

121 model.createEnemy(150,i,1);

122 if(time == 260)

123 for(int i=-160;i<=0;i+=80)

124 model.createEnemy(350,i,3);

125 if(time == 290)

126 for(int i=-160;i<=0;i+=80)

127 model.createEnemy(100,i,2);

128 if(time == 360)

129 for(int i=-160;i<=0;i+=80)

130 model.createEnemy(270,i,1);

131

132

133

134

135 model.EnemyMove();

136 model.BulletMove();

137 model.HitCheck();

138 model.DeleteFigure();

139

140 repaint(); //そして再描画

141

142 }

143

144 /\*GameOver画面に遷移するためのメソッド\*/

145 public int Game()

146 {

147 if(model.GameOver() == true && i==0)

148 {

149 i++;

150 }

151 else if(time == 400)

152 {

153 i++;

154 i++;

155 }

156 return i;

157 }

158

159 }

160

161

162

163

105 for(int i=-320;i<=0;i+=80)

106 model.createEnemy(350,i,1);

107 }

108 if(time == 150)

109 {

110 for(int i=-320;i<=0;i+=80)

111 model.createEnemy(50,i,2);

112

113 for(int i=-320;i<=0;i+=80)

114 model.createEnemy(400,i,3);

115 }

116 if(time == 200)

117 for(int i=-160;i<=0;i+=80)

118 model.createEnemy(300,i,1);

119 if(time == 230)

120 for(int i=-160;i<=0;i+=80)

121 model.createEnemy(150,i,1);

122 if(time == 260)

123 for(int i=-160;i<=0;i+=80)

124 model.createEnemy(350,i,3);

125 if(time == 290)

126 for(int i=-160;i<=0;i+=80)

127 model.createEnemy(100,i,2);

128 if(time == 360)

129 for(int i=-160;i<=0;i+=80)

130 model.createEnemy(270,i,1);

131

132

133

134

135 model.EnemyMove();

136 model.BulletMove();

137 model.HitCheck();

138 model.DeleteFigure();

139

140 repaint(); //そして再描画

141

142 }

143

144 /\*GameOver画面に遷移するためのメソッド\*/

145 public int Game()

146 {

147 if(model.GameOver() == true && i==0)

148 {

149 i++;

150 }

151 else if(time == 400)

152 {

153 i++;

154 i++;

155 }

156 return i;

157 }

158

159 }

160

161

162

163

144 /\*GameOver画面に遷移するためのメソッド\*/

145 public int Game()

146 {

147 if(model.GameOver() == true && i==0)

148 {

149 i++;

150 }

151 else if(time == 400)

152 {

153 i++;

154 i++;

155 }

156 return i;

157 }

158

159 }

160

161

162

163

3.Title.java

1 import javax.swing.\*;

2 import java.awt.\*;

3 import java.awt.event.\*;

4 import java.awt.CardLayout;

5 import java.util.\*;

6

7

8 /\*mainクラス\*/

9 class TitleFrame extends JFrame implements KeyListener,ActionListener

10 {

11 /\*フィールド\*/

12 JPanel p1,p2,p3,p4,p5; //パネル定義

13 CardLayout layout; //cardlayoutの定義

14 boolean boo = false; //操作を変えるためのフラグ

15 boolean flag = false;

16 private CharactorController chara; //ゲーム画面

17 int a = 20;

18 javax.swing.Timer timer;

19 private int time;

20

21 /\*オブジェクト(フレーム内の設定)\*/

22 public TitleFrame()

23 {

24 /\*JPanel設定\*/

25 p1 = new JPanel(); //大本のパネル

26 p2 = new JPanel(); //１ページ目のパネル（タイトル画面）

27 p3 = new JPanel(); //２ページ目のパネル（ゲーム画面）

28 p4 = new JPanel(); //3ページ目のパネル（ゲームオーバー画面）

29 p5 = new JPanel(); //4ページ目のパネル（クリア画面）

30 chara = new CharactorController(); //2ページ目に貼り付けるゲーム画面

31 timer = new javax.swing.Timer(1,this);

32 timer.start();

33

34 p2.setBackground(Color.BLACK); //背景設定

35 p4.setBackground(Color.BLACK); //背景設定

36 p5.setBackground(Color.BLACK); //背景設定

37

38

39 /\*CardLayoutの設定\*/

40 layout = new CardLayout();

41

42 /\*KeyListener追加\*/

43 p1.addKeyListener(this);

44 p1.setFocusable(true);

45

46 /\*p2,p4をBorderLayoutで設定\*/

47 p2.setLayout(new BorderLayout());

48 p4.setLayout(new BorderLayout());

49 p5.setLayout(new BorderLayout());

50

51 /\*p2にJLabelを貼り付ける\*/

52 JLabel l2 = new JLabel("<html>"

53 +"<span style='font-size:110pt;font-family:Impact,Charcoal;"

54 +"color:yellow;'>"

55 +"Shooting</span></html>",

56 +JLabel.CENTER);

57

58

59 JLabel l3 = new JLabel("<html>"

60 +"<span style='font-size:30pt;color:white;'>"

61 +"十字キーで移動、zで弾発射。<br/>"

62 +"<span style='font-size:30pt;color:white;'>"

63 +"Enterでゲームスタート!!"

64 +"</span></html>",

65 +JLabel.CENTER);

66

67 p2.add(l2,BorderLayout.CENTER);

68 p2.add(l3,BorderLayout.SOUTH);

69

70 /\*2ページ目\*/

71 p3.setLayout(new BorderLayout());

72 p3.add(chara,BorderLayout.CENTER);

73

74 /\*p4にJLabelを貼り付ける\*/

75 JLabel l4 = new JLabel("<html>"

76 +"<span style='font-size:90pt;font-family:Impact,Charcoal;"

77 +"color:blue;'>"

78 +"GAMEOVER</span></html>",

79 +JLabel.CENTER);

80

81

82 JLabel l5 = new JLabel("<html>"

83 +"<span style='font-size:30pt;color:white'>"

84 +"2秒後、タイトル画面を表示します。"

85 +"</span></html>",

86 +JLabel.CENTER);

87

88 p4.add(l4,BorderLayout.CENTER);

89 p4.add(l5,BorderLayout.SOUTH);

90

91 /\*p4にJLabelを貼り付ける\*/

92 JLabel l6 = new JLabel("<html>"

93 +"<span style='font-size:120pt;font-family:Impact,Charcoal;"

94 +"color:red;'>"

95 +"GAME<br/>CLEAR!!</span></html>",

96 +JLabel.CENTER);

97

98

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

124

125 /\*keyListener内のメソッド、keyPressedの設定\*/

126 public void keyPressed(KeyEvent e)

127 {

128

129 /\*押されたボタンを読み取る\*/

130 int k = e.getKeyCode();

131

132 /\*押されたボタンがEnterキーかつフラグbooが降りていたら\*/

133 if(k == KeyEvent.VK\_ENTER && boo ==false)

134 {

135 layout.next(p1); //次のページへ移行し、

136 boo = true; //フラグを有効にして操作をゲームの操作にする

137 chara.timerstart(); //timerをスタート

138 }

139

140 /\*フラグbooが立っていたら\*/

141 if(boo == true)

142 {

143 if(k == KeyEvent.VK\_RIGHT) //右押すと

144 chara.jrflag = true; //CharactorControllerクラスのjrflagが立つ

145

146 else if(k == KeyEvent.VK\_LEFT) //左押すと

147 chara.jlflag = true; //CharactorControllerクラスのflflagが立つ

148

149 else if(k == KeyEvent.VK\_UP)

150 chara.juflag = true;

151

152 else if(k == KeyEvent.VK\_DOWN)

153 chara.jdflag = true;

154

155 else if(k == KeyEvent.VK\_Z)

156 chara.jshotflag = true;

157 }

158 }

159

160

161 /\*キーをはなした時の操作\*/

162 public void keyReleased(KeyEvent e)

163 {

164 int k = e.getKeyCode(); //キーの情報を取得

165 if(boo == true) //フラグが有効なら

166 {

167 switch(k)

168 {

169

170 case KeyEvent.VK\_RIGHT: //右ボタンを離すとjrflag無効

171 chara.jrflag = false;

172 break;

173

174 case KeyEvent.VK\_LEFT: //左ボタンを離すとjlflag無効

175 chara.jlflag = false;

176 break;

177

178 case KeyEvent.VK\_UP:

179 chara.juflag = false;

180 break;

181

182 case KeyEvent.VK\_DOWN:

183 chara.jdflag = false;

184 break;

185

186 case KeyEvent.VK\_Z:

187 chara.jshotflag = false;

188 break;

189

190 }

191 }

192

193 }

194

195 /\*キーを押したときの操作(弾を出す)\*/

196 public void keyTyped(KeyEvent e){}

197

198 public void actionPerformed(ActionEvent ev)

199 {

200 if(chara.i == 1)

201 {

202 layout.next(p1);

203 chara.i = 4;

204 }

205 if(chara.i == 2)

206 {

207 layout.next(p1);

208 layout.next(p1);

209 chara.i = 4;

210 }

211 if(chara.i == 4)

212 {

213 time++;

214 if(time == 2000)

215 {

216 new TitleFrame();

217 timer.stop();

218 time = 0;

219 setVisible(false);

220 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }

17 int a = 20;

18 javax.swing.Timer timer;

19 private int time;

20

21 /\*オブジェクト(フレーム内の設定)\*/

22 public TitleFrame()

23 {

24 /\*JPanel設定\*/

25 p1 = new JPanel(); //大本のパネル

26 p2 = new JPanel(); //１ページ目のパネル（タイトル画面）

27 p3 = new JPanel(); //２ページ目のパネル（ゲーム画面）

28 p4 = new JPanel(); //3ページ目のパネル（ゲームオーバー画面）

29 p5 = new JPanel(); //4ページ目のパネル（クリア画面）

30 chara = new CharactorController(); //2ページ目に貼り付けるゲーム画面

31 timer = new javax.swing.Timer(1,this);

32 timer.start();

33

34 p2.setBackground(Color.BLACK); //背景設定

35 p4.setBackground(Color.BLACK); //背景設定

36 p5.setBackground(Color.BLACK); //背景設定

37

38

39 /\*CardLayoutの設定\*/

40 layout = new CardLayout();

41

42 /\*KeyListener追加\*/

43 p1.addKeyListener(this);

44 p1.setFocusable(true);

45

46 /\*p2,p4をBorderLayoutで設定\*/

47 p2.setLayout(new BorderLayout());

48 p4.setLayout(new BorderLayout());

49 p5.setLayout(new BorderLayout());

50

51 /\*p2にJLabelを貼り付ける\*/

52 JLabel l2 = new JLabel("<html>"

53 +"<span style='font-size:110pt;font-family:Impact,Charcoal;"

54 +"color:yellow;'>"

55 +"Shooting</span></html>",

56 +JLabel.CENTER);

57

58

59 JLabel l3 = new JLabel("<html>"

60 +"<span style='font-size:30pt;color:white;'>"

61 +"十字キーで移動、zで弾発射。<br/>"

62 +"<span style='font-size:30pt;color:white;'>"

63 +"Enterでゲームスタート!!"

64 +"</span></html>",

65 +JLabel.CENTER);

66

67 p2.add(l2,BorderLayout.CENTER);

68 p2.add(l3,BorderLayout.SOUTH);

69

70 /\*2ページ目\*/

71 p3.setLayout(new BorderLayout());

72 p3.add(chara,BorderLayout.CENTER);

73

74 /\*p4にJLabelを貼り付ける\*/

75 JLabel l4 = new JLabel("<html>"

76 +"<span style='font-size:90pt;font-family:Impact,Charcoal;"

77 +"color:blue;'>"

78 +"GAMEOVER</span></html>",

79 +JLabel.CENTER);

80

81

82 JLabel l5 = new JLabel("<html>"

83 +"<span style='font-size:30pt;color:white'>"

84 +"2秒後、タイトル画面を表示します。"

85 +"</span></html>",

86 +JLabel.CENTER);

87

88 p4.add(l4,BorderLayout.CENTER);

89 p4.add(l5,BorderLayout.SOUTH);

90

91 /\*p4にJLabelを貼り付ける\*/

92 JLabel l6 = new JLabel("<html>"

93 +"<span style='font-size:120pt;font-family:Impact,Charcoal;"

94 +"color:red;'>"

95 +"GAME<br/>CLEAR!!</span></html>",

96 +JLabel.CENTER);

97

98

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

124

125 /\*keyListener内のメソッド、keyPressedの設定\*/

126 public void keyPressed(KeyEvent e)

127 {

128

129 /\*押されたボタンを読み取る\*/

130 int k = e.getKeyCode();

131

132 /\*押されたボタンがEnterキーかつフラグbooが降りていたら\*/

133 if(k == KeyEvent.VK\_ENTER && boo ==false)

134 {

135 layout.next(p1); //次のページへ移行し、

136 boo = true; //フラグを有効にして操作をゲームの操作にする

137 chara.timerstart(); //timerをスタート

138 }

139

140 /\*フラグbooが立っていたら\*/

141 if(boo == true)

142 {

143 if(k == KeyEvent.VK\_RIGHT) //右押すと

144 chara.jrflag = true; //CharactorControllerクラスのjrflagが立つ

145

146 else if(k == KeyEvent.VK\_LEFT) //左押すと

147 chara.jlflag = true; //CharactorControllerクラスのflflagが立つ

148

149 else if(k == KeyEvent.VK\_UP)

150 chara.juflag = true;

151

152 else if(k == KeyEvent.VK\_DOWN)

153 chara.jdflag = true;

154

155 else if(k == KeyEvent.VK\_Z)

156 chara.jshotflag = true;

157 }

158 }

159

160

161 /\*キーをはなした時の操作\*/

162 public void keyReleased(KeyEvent e)

163 {

164 int k = e.getKeyCode(); //キーの情報を取得

165 if(boo == true) //フラグが有効なら

166 {

167 switch(k)

168 {

169

170 case KeyEvent.VK\_RIGHT: //右ボタンを離すとjrflag無効

171 chara.jrflag = false;

172 break;

173

174 case KeyEvent.VK\_LEFT: //左ボタンを離すとjlflag無効

175 chara.jlflag = false;

176 break;

177

178 case KeyEvent.VK\_UP:

179 chara.juflag = false;

180 break;

181

182 case KeyEvent.VK\_DOWN:

183 chara.jdflag = false;

184 break;

185

186 case KeyEvent.VK\_Z:

187 chara.jshotflag = false;

188 break;

189

190 }

191 }

192

193 }

194

195 /\*キーを押したときの操作(弾を出す)\*/

196 public void keyTyped(KeyEvent e){}

197

198 public void actionPerformed(ActionEvent ev)

199 {

200 if(chara.i == 1)

201 {

202 layout.next(p1);

203 chara.i = 4;

204 }

205 if(chara.i == 2)

206 {

207 layout.next(p1);

208 layout.next(p1);

209 chara.i = 4;

210 }

211 if(chara.i == 4)

212 {

213 time++;

214 if(time == 2000)

215 {

216 new TitleFrame();

217 timer.stop();

218 time = 0;

219 setVisible(false);

220 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }

58

59 JLabel l3 = new JLabel("<html>"

60 +"<span style='font-size:30pt;color:white;'>"

61 +"十字キーで移動、zで弾発射。<br/>"

62 +"<span style='font-size:30pt;color:white;'>"

63 +"Enterでゲームスタート!!"

64 +"</span></html>",

65 +JLabel.CENTER);

66

67 p2.add(l2,BorderLayout.CENTER);

68 p2.add(l3,BorderLayout.SOUTH);

69

70 /\*2ページ目\*/

71 p3.setLayout(new BorderLayout());

72 p3.add(chara,BorderLayout.CENTER);

73

74 /\*p4にJLabelを貼り付ける\*/

75 JLabel l4 = new JLabel("<html>"

76 +"<span style='font-size:90pt;font-family:Impact,Charcoal;"

77 +"color:blue;'>"

78 +"GAMEOVER</span></html>",

79 +JLabel.CENTER);

80

81

82 JLabel l5 = new JLabel("<html>"

83 +"<span style='font-size:30pt;color:white'>"

84 +"2秒後、タイトル画面を表示します。"

85 +"</span></html>",

86 +JLabel.CENTER);

87

88 p4.add(l4,BorderLayout.CENTER);

89 p4.add(l5,BorderLayout.SOUTH);

90

91 /\*p4にJLabelを貼り付ける\*/

92 JLabel l6 = new JLabel("<html>"

93 +"<span style='font-size:120pt;font-family:Impact,Charcoal;"

94 +"color:red;'>"

95 +"GAME<br/>CLEAR!!</span></html>",

96 +JLabel.CENTER);

97

98

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

124

125 /\*keyListener内のメソッド、keyPressedの設定\*/

126 public void keyPressed(KeyEvent e)

127 {

128

129 /\*押されたボタンを読み取る\*/

130 int k = e.getKeyCode();

131

132 /\*押されたボタンがEnterキーかつフラグbooが降りていたら\*/

133 if(k == KeyEvent.VK\_ENTER && boo ==false)

134 {

135 layout.next(p1); //次のページへ移行し、

136 boo = true; //フラグを有効にして操作をゲームの操作にする

137 chara.timerstart(); //timerをスタート

138 }

139

140 /\*フラグbooが立っていたら\*/

141 if(boo == true)

142 {

143 if(k == KeyEvent.VK\_RIGHT) //右押すと

144 chara.jrflag = true; //CharactorControllerクラスのjrflagが立つ

145

146 else if(k == KeyEvent.VK\_LEFT) //左押すと

147 chara.jlflag = true; //CharactorControllerクラスのflflagが立つ

148

149 else if(k == KeyEvent.VK\_UP)

150 chara.juflag = true;

151

152 else if(k == KeyEvent.VK\_DOWN)

153 chara.jdflag = true;

154

155 else if(k == KeyEvent.VK\_Z)

156 chara.jshotflag = true;

157 }

158 }

159

160

161 /\*キーをはなした時の操作\*/

162 public void keyReleased(KeyEvent e)

163 {

164 int k = e.getKeyCode(); //キーの情報を取得

165 if(boo == true) //フラグが有効なら

166 {

167 switch(k)

168 {

169

170 case KeyEvent.VK\_RIGHT: //右ボタンを離すとjrflag無効

171 chara.jrflag = false;

172 break;

173

174 case KeyEvent.VK\_LEFT: //左ボタンを離すとjlflag無効

175 chara.jlflag = false;

176 break;

177

178 case KeyEvent.VK\_UP:

179 chara.juflag = false;

180 break;

181

182 case KeyEvent.VK\_DOWN:

183 chara.jdflag = false;

184 break;

185

186 case KeyEvent.VK\_Z:

187 chara.jshotflag = false;

188 break;

189

190 }

191 }

192

193 }

194

195 /\*キーを押したときの操作(弾を出す)\*/

196 public void keyTyped(KeyEvent e){}

197

198 public void actionPerformed(ActionEvent ev)

199 {

200 if(chara.i == 1)

201 {

202 layout.next(p1);

203 chara.i = 4;

204 }

205 if(chara.i == 2)

206 {

207 layout.next(p1);

208 layout.next(p1);

209 chara.i = 4;

210 }

211 if(chara.i == 4)

212 {

213 time++;

214 if(time == 2000)

215 {

216 new TitleFrame();

217 timer.stop();

218 time = 0;

219 setVisible(false);

220 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }

99 JLabel l7 = new JLabel("<html>"

100 +"<span style='font-size:30pt;color:white'>"

101 +"2秒後、タイトル画面を表示します。"

102 +"</span></html>",

103 +JLabel.CENTER);

104

105

106 p5.add(l6,BorderLayout.CENTER);

107 p5.add(l7,BorderLayout.SOUTH);

108

109 /\*p1をCardLayoutで設定し、p1にパネルを貼り付ける。\*/

110 p1.setLayout(layout);

111 p1.add(p2,BorderLayout.CENTER);

112 p1.add(p3,BorderLayout.CENTER);

113 p1.add(p4,BorderLayout.CENTER);

114 p1.add(p5,BorderLayout.CENTER);

115 getContentPane().add(p1,BorderLayout.CENTER); //これがないと表示されない(はず)

116

117

118 /\*Frame設定\*/

119 this.setSize(550,550); //Frameは550\*550(仮)

120 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

121 this.setTitle("Shooting");

122 this.setVisible(true);

123 }

124

125 /\*keyListener内のメソッド、keyPressedの設定\*/

126 public void keyPressed(KeyEvent e)

127 {

128

129 /\*押されたボタンを読み取る\*/

130 int k = e.getKeyCode();

131

132 /\*押されたボタンがEnterキーかつフラグbooが降りていたら\*/

133 if(k == KeyEvent.VK\_ENTER && boo ==false)

134 {

135 layout.next(p1); //次のページへ移行し、

136 boo = true; //フラグを有効にして操作をゲームの操作にする

137 chara.timerstart(); //timerをスタート

138 }

139

140 /\*フラグbooが立っていたら\*/

141 if(boo == true)

142 {

143 if(k == KeyEvent.VK\_RIGHT) //右押すと

144 chara.jrflag = true; //CharactorControllerクラスのjrflagが立つ

145

146 else if(k == KeyEvent.VK\_LEFT) //左押すと

147 chara.jlflag = true; //CharactorControllerクラスのflflagが立つ

148

149 else if(k == KeyEvent.VK\_UP)

150 chara.juflag = true;

151

152 else if(k == KeyEvent.VK\_DOWN)

153 chara.jdflag = true;

154

155 else if(k == KeyEvent.VK\_Z)

156 chara.jshotflag = true;

157 }

158 }

159

160

161 /\*キーをはなした時の操作\*/

162 public void keyReleased(KeyEvent e)

163 {

164 int k = e.getKeyCode(); //キーの情報を取得

165 if(boo == true) //フラグが有効なら

166 {

167 switch(k)

168 {

169

170 case KeyEvent.VK\_RIGHT: //右ボタンを離すとjrflag無効

171 chara.jrflag = false;

172 break;

173

174 case KeyEvent.VK\_LEFT: //左ボタンを離すとjlflag無効

175 chara.jlflag = false;

176 break;

177

178 case KeyEvent.VK\_UP:

179 chara.juflag = false;

180 break;

181

182 case KeyEvent.VK\_DOWN:

183 chara.jdflag = false;

184 break;

185

186 case KeyEvent.VK\_Z:

187 chara.jshotflag = false;

188 break;

189

190 }

191 }

192

193 }

194

195 /\*キーを押したときの操作(弾を出す)\*/

196 public void keyTyped(KeyEvent e){}

197

198 public void actionPerformed(ActionEvent ev)

199 {

200 if(chara.i == 1)

201 {

202 layout.next(p1);

203 chara.i = 4;

204 }

205 if(chara.i == 2)

206 {

207 layout.next(p1);

208 layout.next(p1);

209 chara.i = 4;

210 }

211 if(chara.i == 4)

212 {

213 time++;

214 if(time == 2000)

215 {

216 new TitleFrame();

217 timer.stop();

218 time = 0;

219 setVisible(false);

220 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }

139

140 /\*フラグbooが立っていたら\*/

141 if(boo == true)

142 {

143 if(k == KeyEvent.VK\_RIGHT) //右押すと

144 chara.jrflag = true; //CharactorControllerクラスのjrflagが立つ

145

146 else if(k == KeyEvent.VK\_LEFT) //左押すと

147 chara.jlflag = true; //CharactorControllerクラスのflflagが立つ

148

149 else if(k == KeyEvent.VK\_UP)

150 chara.juflag = true;

151

152 else if(k == KeyEvent.VK\_DOWN)

153 chara.jdflag = true;

154

155 else if(k == KeyEvent.VK\_Z)

156 chara.jshotflag = true;

157 }

158 }

159

160

161 /\*キーをはなした時の操作\*/

162 public void keyReleased(KeyEvent e)

163 {

164 int k = e.getKeyCode(); //キーの情報を取得

165 if(boo == true) //フラグが有効なら

166 {

167 switch(k)

168 {

169

170 case KeyEvent.VK\_RIGHT: //右ボタンを離すとjrflag無効

171 chara.jrflag = false;

172 break;

173

174 case KeyEvent.VK\_LEFT: //左ボタンを離すとjlflag無効

175 chara.jlflag = false;

176 break;

177

178 case KeyEvent.VK\_UP:

179 chara.juflag = false;

180 break;

181

182 case KeyEvent.VK\_DOWN:

183 chara.jdflag = false;

184 break;

185

186 case KeyEvent.VK\_Z:

187 chara.jshotflag = false;

188 break;

189

190 }

191 }

192

193 }

194

195 /\*キーを押したときの操作(弾を出す)\*/

196 public void keyTyped(KeyEvent e){}

197

198 public void actionPerformed(ActionEvent ev)

199 {

200 if(chara.i == 1)

201 {

202 layout.next(p1);

203 chara.i = 4;

204 }

205 if(chara.i == 2)

206 {

207 layout.next(p1);

208 layout.next(p1);

209 chara.i = 4;

210 }

211 if(chara.i == 4)

212 {

213 time++;

214 if(time == 2000)

215 {

216 new TitleFrame();

217 timer.stop();

218 time = 0;

219 setVisible(false);

220 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }

180 break;

181

182 case KeyEvent.VK\_DOWN:

183 chara.jdflag = false;

184 break;

185

186 case KeyEvent.VK\_Z:

187 chara.jshotflag = false;

188 break;

189

190 }

191 }

192

193 }

194

195 /\*キーを押したときの操作(弾を出す)\*/

196 public void keyTyped(KeyEvent e){}

197

198 public void actionPerformed(ActionEvent ev)

199 {

200 if(chara.i == 1)

201 {

202 layout.next(p1);

203 chara.i = 4;

204 }

205 if(chara.i == 2)

206 {

207 layout.next(p1);

208 layout.next(p1);

209 chara.i = 4;

210 }

211 if(chara.i == 4)

212 {

213 time++;

214 if(time == 2000)

215 {

216 new TitleFrame();

217 timer.stop();

218 time = 0;

219 setVisible(false);

220 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }

221 }

222 }

223

224 /\*mainメソッド(Frame生成のみ)\*/

225 public static void main(String argv[])

226 {

227 new TitleFrame();

228 }

229 }