**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

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

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 /\*コンストラクタ\*/

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

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

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

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

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

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

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

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

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

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

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 }

19 private int time;

20

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 }

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 }

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 }

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 }

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 }