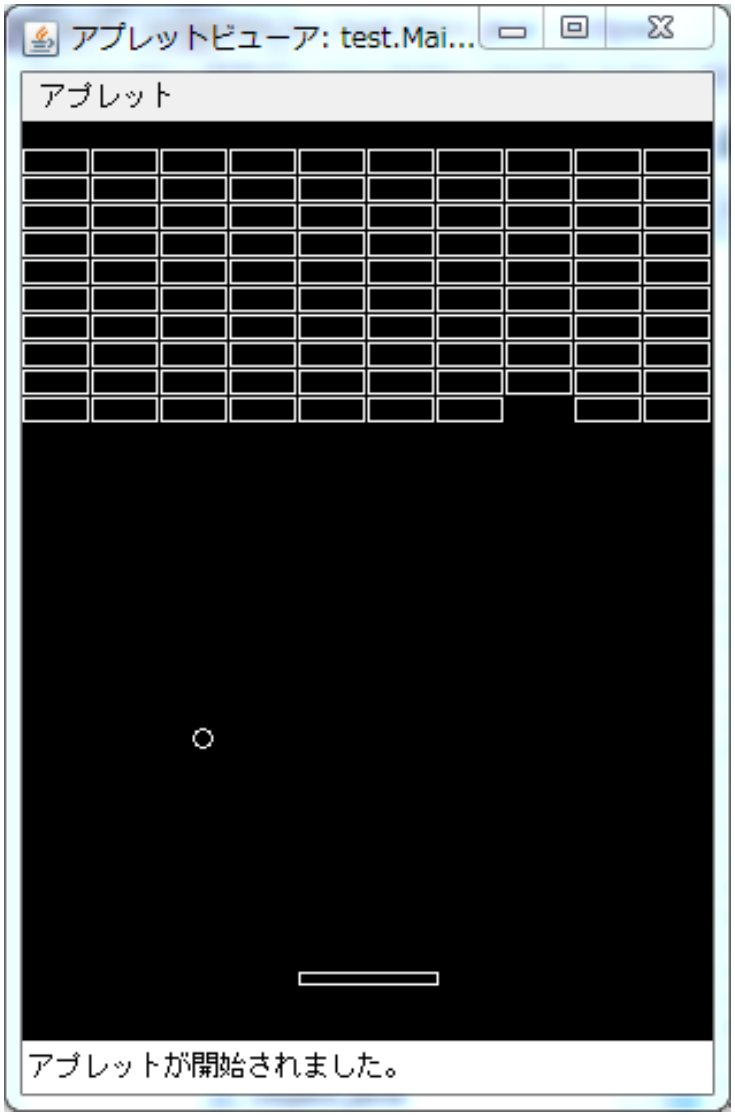
Java講座えくすとら

ブロック崩しゲームを作ろう

情報科学部コンピュータ科学科２年
蠧川内　努

この講座でできるもの



- Java applet を利用
- キーボード操作の受付
- タイトル画面、ゲーム画面、ゲームオーバー画面への状態遷移

はじめに

- 正直竹中のスライドの内容以上のことが書けないです
- より深く知りたいならば竹中のスライドを読むと良いです
- このスライドではどこをいじれば挙動が変わるのかを簡単に教えたいと思います

準備

- クラスの作成

- Mainクラス・・・ゲームの実行、画面の描画、ボール、ブロックなどの管理
- Ball、Bar、Block、Objectクラス・・・画面上に出てくる物体の、位置情報や大きさを持たせるクラス
- Keyクラス・・・キーボードからの入力を受け取るクラス

Mainクラス

- はじめに、ボールやバー、ブロックなどゲームに必要なものを生成する。後述のそれぞれのクラスで定義する変数に対してコンストラクターで値を与える

- ここで9行目の配列の値を変えると楽しい
- やってみよう　例えば　Block[][] blocks=**new Block[30][10];**　とか

```
public class Main extends Applet  implements Runnable {
    Bar bar=new Bar(300/2-30,370); //反射板を初期座標とともに生成
    Ball ball=new Ball(300/2-4,360); //ボールを初期座標とともに生成
    Block[][] blocks=new Block[10][10]; //ブロックの配列
    Thread gameThread;        // ゲームスレッド
    Image offImage;           // 仮想画面
    int scene;                // シーン（0:タイトル 1:メイン 2:ゲームオーバー）
    int blockDeaths;
```

Mainクラス

- 初期化
- 画面のサイズの設定、キーボード入力受付開始、ゲームの開始を行う
- ここをいじると画面サイズを自由に変えられる

→17行目、21行目をいじって変えてみよう

```
15      // 初期化
16      public void init() {
17          setSize(300, 400);//画面のサイズを指定(横, 縦)
18          ready();//開始準備呼び出し
19          setBackground(Color.black);// 背景色を黒色に設定
20          setForeground(Color.white);// 前景色を白色に設定
21          offImage = createImage(300, 400);// 仮想画面の生成
22          addKeyListener(new Key());// キー受け付けオブジェクト生成
23          requestFocusInWindow();// フォーカスを要求
24          scene = 0;// シーンをタイトルに
25      }
26
```

Mainクラス

- 開始準備メソッド
- ブロックの初期化、反射板、ボールの初期座標を設定する。
- ゲームを開始するたびリセットできるようにするためにこのメソッドを用意した。
- 初期座標をコンストラクターに入れても意味ないもんね

```
27    //開始準備
28    void ready(){
29        // ブロックの初期化
30        for(int i=0;i<blocks.length;i++){
31            for(int j=0;j<blocks[0].length;j++){
32                blocks[i][j]=new Block(j*30,(i+1)*12,10,28,0);
33            }
34        }
35        //反射板を初期座標へ
36        bar.x=300/2-30;
37        bar.y=370;
38
39        //ボールの初期化
40        ball.x=300/2-4;
41        ball.y=360;
42        ball.dx=2;
43        ball.dy =2;
44        ball.dead=0;
45
46        blockDeaths=0;
47    }
```

Mainクラス

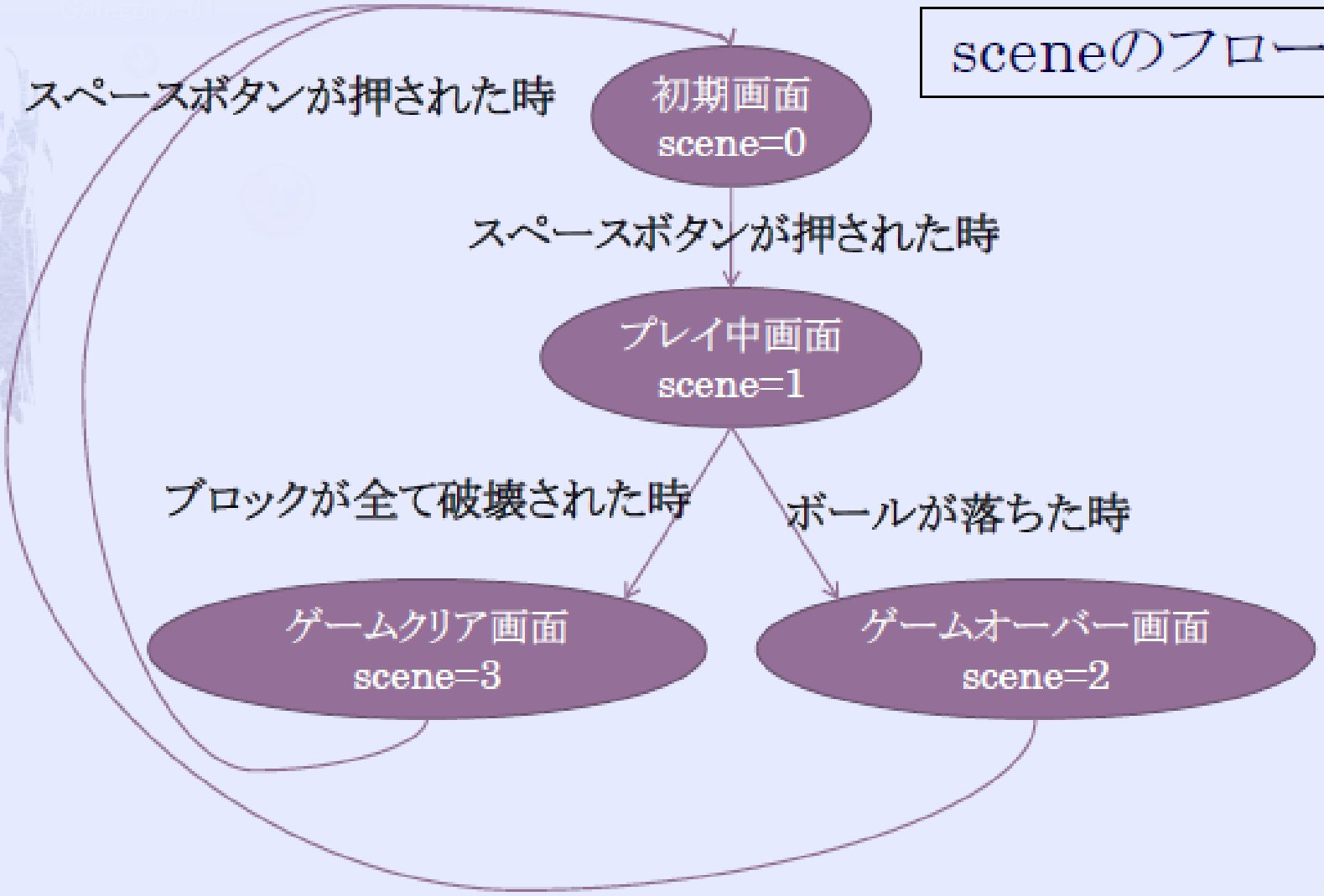
- ゲーム画面の遷移

- Switch文でのゲーム画面の切り替え
- While文で回してゲームを描画する

- 状態遷移図は竹中がいいのを書いてくれています

クラス設計　Mainクラス

sceneのフローチャート



Mainクラス

- サンプルコードにはゲームクリアー画面は用意されていませんがお好みで追加してください。
- 72行目の値を書き換えると描画間隔が変わります。
- 描画を早くするとより細かい入力が受け付けられるが速さをint型で定義したときボールの速度調整の関係で相性が悪い・・・
- とりあえずいじってみよう

```
61    // ゲームスレッドのメイン
62    public void run() {
63        while (gameThread == Thread.currentThread()) {
64            // ゲームメイン処理
65            switch (scene) {
66                case 0: gameTitle(); break;
67                case 1: gameMain(); break;
68                case 2: gameOver(); break;
69            }
70            // 描画間隔（ミリ秒単位）
71            try {
72                Thread.sleep(9);
73            } catch (InterruptedException e) {
74                break;
75            }
76        }
77    }
```

Mainクラス

- タイトル画面、ゲームオーバー画面

- ここではメッセージの表示と、ゲーム画面を選択させるためにキー入力の受付を行っている

- フォントの選択は
- gv.setFont(**new Font(“フォント名”, Font.文体, *大きさ*));**
- 文字の表示は
- gv.drawString(“表示させたい文章”, x座標, y座標);

- で行う

Mainクラス

- 適当にいじってみよう

```
79      // ゲームタイトル処理
80      void gameTitle() {
81          // エンターキーが押されたらシーンをゲームメインへ
82          if (Key.enter) scene = 1;
83          Graphics gv = offImage.getGraphics();
84          // 裏画面の消去
85          gv.clearRect(0, 0, 300, 400);
86          // タイトル文字列描画
87          gv.setFont(new Font("Arial", Font.BOLD, 35));
88          gv.drawString("Block break!", 50, 180);
89          gv.setFont(new Font("Arial", Font.PLAIN, 20));
90          gv.drawString("HIT ENTER KEY.", 70, 350);
91          // 再描画
92          repaint();
93      }
94
95      // ゲームメイン処理
96      void gameMain() {
123
124     // ゲームオーバー処理
125     void gameOver() {
126         Graphics gv = offImage.getGraphics();
127         gv.clearRect(0, 0, 300, 400);// 裏画面の消去
128         
129         // ゲームオーバー文字列描画
130         gv.setFont(new Font("Arial", Font.BOLD, 28));
131         gv.drawString("GAME OVER", 64, 180);
132         gv.setFont(new Font("Arial", Font.PLAIN, 20));
133         gv.drawString("Push space to return", 55, 250);
134         
135         repaint();// 再描画
136         
137         //再挑戦
138         if (Key.space) {
139             ready();// ゲーム準備処理を行い
140             scene = 0;// タイトル画面へ移行
141         }
```

Mainクラス

- ゲームメイン処理
- ようやく本題、ゲームのメイン処理です
- ここでは後で定義する移動メソッドと当たり判定メソッドを呼び出し、その移動結果を画面に書き出す処理を行います
- この処理が62行目から始まっているメソッド内のwhile文でループされることによってゲームが進行します

- ゲームオーバー画面への遷移条件は、ボールが画面外に落ちてしまうか、ブロックがすべて壊されるかに設定しました

- **if (ball.dead==1 || blockDeaths==blocks.length*blocks[0].length) scene = 2;//ボールが死んだらゲームオーバー画面へ**

Mainクラス

- ブロック、ボール、バーの描画には以下の方法を使います

- 四角形
- gv.drawRect(x座標, y座標, 幅, 高さ);
- まる
- gv.drawOval(x座標, y座標, 幅, 高さ);
- このプログラムでは座標、大きさの情報は板、ボールのクラスがそれぞれ持っているのでそこから呼び出しています

Mainクラス

- このクラス最後のほうには当たり判定の記述がありますが、これの解説は竹中のスライドのほうが分かりやすいのでそっちを見ましょう

- このメソッドの役割は、ボールが反射板とぶつかったらボールを跳ね返す、ボールがブロックとぶつかったら跳ね返してブロックを消す処理をすることです

- このあたり判定メソッドがメイン処理メソッドで呼び出され、当たり判定を行っています

Objectクラス

- このクラスではどの物体でも持っている座標、大きさの変数を宣言する

- このクラスを親クラスにすることで新しく物体のクラスを作るときの手間を少し減らせる

Blockクラス

- ブロックには座標、大きさの基本変数のほかに、壊されたかどうかを判定する変数が必要

- よって判定用の変数を追加する

Ballクラス

- このクラスには、ボールが落ちてしまったときの判定用の変数を追加する
- ボールは動くので速度成分の変数も必要
- 独自のメソッドとして壁との反射を行うメソッドを追加する

```
3  public class Ball extends Object{
4      double dx;
5      double dy;
6      
7      int dead=0;
8      
9      Ball(int x,int y){
10         super(x, y, 8, 8);
11     }
12     
13     void ballMove(){
14         //壁での反射
15         if(this.dead==0){
16             if(this.x>300-this.width || this.x<0){
17                 this.dx=-this.dx;
18             }
19             if(this.y<0){
20                 this.dy=-this.dy;
21             }
22             if(this.y>400){
23                 this.dead=1;
24             }
25             
26             this.x += dx;
27             this.y += dy;
28         }
29     }
30 }
31
```

Barクラス

- このクラスには何が必要か？

- 移動を行う必要がある
- キー入力を受けたら座標を動かすメソッドを書く
- 10行目、15行目をいじると楽しい
- やってみよう
- 例・**this.x -= 100;**　など

Keyクラス

- じゃあさっきから言ってるキー入力ってどうやって受けてるのよ？
- →java.awtのKeyAdapterとKeyEventを利用
- はじめにキーの名前をboolean型で宣言し、その真偽の条件をKeyEvent.VK_キーの名前 で押されたときに変化するように設定する。

Keyクラス

```
class Key extends KeyAdapter {
    static boolean left, right, up, down, space, enter, b;
    // キーが押されたときの処理
    public void keyPressed(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_LEFT:  left = true; break; // ←
            case KeyEvent.VK_RIGHT:  right = true; break;   // →
            case KeyEvent.VK_SPACE:  space = true; break;   // SPACE
            case KeyEvent.VK_ENTER:  enter = true; break;   // Enter
        }
    }
    // キーが離されたときの処理
    public void keyReleased(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_LEFT:  left = false; break;// ←
            case KeyEvent.VK_RIGHT:  right = false; break;  // →
            case KeyEvent.VK_SPACE:  space = false; break;  // SPACE
            case KeyEvent.VK_ENTER:  enter = false; break;  // Enter
        }
    }
    // キーがタイプされたときの処理
    public void keyTyped(KeyEvent e) {}
}
```

作ってみよう

- 以上のことを踏まえてオリジナルのゲームを作ろう
- テンプレートとなるファイルを用意したので自由に書いてみてね
- ブロック崩しのだけじゃなくてホッケーゲームとかインベーダーゲームとかも考えれば作れるはずだよっ☆