【実習６】点数表示とシーン展開

１　ソースコード

（１）テンプレート　Jump の①の箇所に　Jump060View　を入力してください。

（２）次のアプリケーションを新規作成してください。

Jump050View をコピー＆ペーストして、ファイル名を「Jump060View」に変更してください。

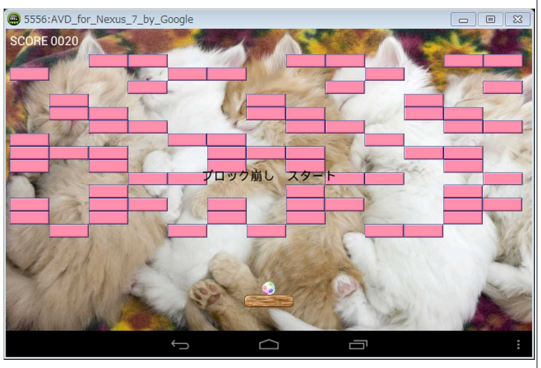
プロジェクト名：JumpPro????(年組席)　　アプリケーション名：Jump060View

```
/*    年    組    席  名前
 * Jump060View
 *     点数表示とシーン展開
 */
・・・

public class Jump060View extends SurfaceView
                    implements SurfaceHolder.Callback
{
    ・・・
    private int barHEIGHT;//バーの高さ

    //シーン定数
    private final static int
        S_TITLE=0,
        S_PLAY=1,
        S_CLEAR=2,
        S_GAMEOVER=3;
    //シーン
    private int scene=S_TITLE;//シーン
    private int init=S_TITLE;//初期化
    private String message=null;//メッセージ
    private int score=0;//スコア
    private int num;//ブロック数

   public Jump060View(Context context)
    {
        ・・・
    }
    //サーフェイス生成時に呼ばれる
    public void surfaceCreated(SurfaceHolder holder)
    {
        init=S_TITLE;
        //描画の設定
        ・・・
        executor=Executors.newSingleThreadScheduledExecutor();
        executor.scheduleAtFixedRate(
            new Runnable()
            {
                public void run()
```



```
                {
                    //シーンの初期化
                    if (init>=0)
                    {
                        scene=init;
                        init=-1;
                        //タイトルの初期化
                        if (scene==S_TITLE)
                        {
                            message="ブロック崩し　スタート";
                            for(int i=0;i<block.length;i++){block[i]=BLOCK[i];}
                            ballX =getWidth()/2-ballWIDTH/2;//X 座標
                            ballY=barY-ballHEIGHT;//Y 座標
                            ballVX=ballSPEED;
                            ballVY=-ballSPEED;
                            barX=getWidth()/2-barWIDTH/2;
                        }
                        //プレイの初期化
                        else if (scene==S_PLAY)
                        {
                            if (rand(5)>2)
                            {
                                ballVX=ballSPEED;
                            }else{
                                ballVX=-ballSPEED;
                            }
                            ballVY=-ballSPEED;
                            message=null;
                        }
                        //クリアの初期化
                        else if (scene==S_CLEAR)
                        {
                            message="おめでとうございます";
                        }
                        //ゲームオーバーの初期化
                        else if (scene==S_GAMEOVER)
                        {
                            message="ゲーム終了";
                        }
                    }
                    //ボールの移動
                    if (scene==S_PLAY)
                    {
                        for(int j=0;j<4;j++) moveBall();
                    }
                    draw(canvas);
                }
            }, 0, 1, TimeUnit.MILLISECONDS);
    }
    public void draw(Canvas canvas)
    {
        ・・・
        //ブロックの描画
        for(int i=0;i<BLOCK.length;i++)
        {
```

```
            if (block[i]!=0)
            {
                canvas.drawBitmap(bmp[1],
                        10+(i%13)*blockWIDTH,
                        62+(i/13)*blockHEIGHT,null);
                num++;
            }
        }
        //スコアの描画
        paint.setColor(Color.WHITE);
        paint.setTextSize(30);
        canvas.drawText("SCORE "+num2str(score,4),10,40,paint);
         if (scene==S_PLAY && num==0) init=S_CLEAR;
        //メッセージの描画
        if (message!=null)
        {
            paint.setColor(Color.BLACK);
            canvas.drawText(message,(getWidth()/2-
                        paint.measureText(message)/2),getHeight()/2,paint);
        }
        canvas.drawBitmap(bmp[3],barX,barY,null);//バーの描画
        canvas.drawBitmap(bmp[2],ballX,ballY,null);//ボールの描画
         holder.unlockCanvasAndPost(canvas);//実画面に反映
    }
    //ボールの移動
    private void moveBall()
    {
        ・・・
        if(ballY>getHeight()-ballHEIGHT)
        {
            ballY=getHeight()-ballHEIGHT;
            ballVY=-ballVY;
        }

        if(ballY>barY){init=S_GAMEOVER;}//バーで跳ね返りができなかった時

        //バーとの衝突判定
        ・・・
         //ブロックとの衝突判定
        for (int i=0;i<BLOCK.length;i++)
        {
            if (block[i]==0) continue;
            int r1=hitLineRect(
                ballX-ballVX,ballY-ballVY,ballX,ballY,(i%13)*blockWIDTH,
                                50+(i/13)*blockHEIGHT,blockWIDTH,blockHEIGHT);
            if (r1>=0)
            {
                if (r1==0 || r1==1) ballVY=-ballVY;//上、下
                if (r1==2 || r1==3) ballVX=-ballVX;//左、右
                block[i]=0;//ブロックの消去
                score+=10;
                break;
            }
        }
    }
```

```
    ・・・
    //サーフェイスの破棄
    public void surfaceDestroyed(SurfaceHolder holder){executor.shutdown();}

    //タッチイベントの処理
    public boolean onTouchEvent(MotionEvent event)
    {
        int action=event.getAction();
        switch (action&MotionEvent.ACTION_MASK)
        {
            case MotionEvent.ACTION_DOWN:
                if (scene==S_TITLE) {
                    init=S_PLAY;
                } else if (scene==S_PLAY) {
                    barX=(int)event.getX();
                } else {
                    init=S_TITLE;
                }
                break;
             case MotionEvent.ACTION_MOVE:
                if (scene==S_PLAY) {
                     barX=(int)event.getX();
                }
                break;
        }
        return true;
    }

    ・・・

    //乱数の取得
    private static Random rand=new Random();
    private static int rand(int num) { return (rand.nextInt()>>>1)%num; }
    //数値→文字列
    private static String num2str(int num,int len)
    {
        String str=""+num;
        while(str.length()<len) str="0"+str;
        return str;
    }
}
```

2　仕様

①バーをタッチで左右に操作して、落ちてくるボールを跳ね返してブロックを壊していくゲームです。

②ブロックはボールに当たると消滅し、すべて消滅するとクリアになります。

③画面左上に表示されている数字はスコアです。

④ボールが画面下に落ちるとゲームオーバーになります。

３　プログラムの流れ

```
//シーン定数
private final static int
        S_TITLE=0,
        S_PLAY=1,
        S_CLEAR=2,
        S_GAMEOVER=3;
    ・
    ・
//シーン
private int scene=S_TITLE;//シーン
private int init=S_TITLE;//初期化
private String message=null;//メッセージ
private int score=0;//スコア
```

このゲームには４つのシーンがあり、現在のシーンは、int 型変数 scene で保持しています。次に遷移するシーンは、int 型変数 init で保持しています。遷移しない時は、init は－１を保持しています。

このゲームの処理のほとんどは、run()メソッド内の無限ループで行っています。最初に初期化変数 init を見て、必要なときは初期化を行い、その後はシーン変数 scene に応じて、各種内部処理およびキーイベント処理を行った後、描画処理を行い、最後にスリープしています。

3.1　０：タイトル

「ブロック崩し　スタート」という文字列を表示するシーンです。画面をタッチすると、シーンは「プレイ」に遷移します。

3.2　１：プレイ

実際にゲームをプレイするシーンです。ボールを下に落としたら、シーンは「ゲームオーバー」に遷移します。全ブロックを消すと、シーンは「クリア」に遷移します。

3.3　２：クリア

「おめでとうございます」という文字列を表示するシーンです。画面をタッチすると、シーンは「タイトル」に遷移します。

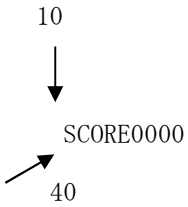
3.4　３：ゲームオーバー

「ゲーム終了」という文字列を表示するシーンです。画面をタッチすると、シーンは「タイトル」に遷移します。

４　詳細

(1)このプログラム中の moveBall メソッドでボールの移動を行っています。

(2)このプログラム中の num2str メソッドで数値から４桁の文字列に変換しています。
drawText メソッドで指定したＸＹ座標は、文字列の左下の座標となります。



(3)変数 num はブロックの数をカウントしています。num が０になると画面上のブロックがなくなり、クリアされたことになります。