2023 年度
実世界情報実験２
「モバイル端末プログラミング」
最終レポート

提出日：　　2023 年　11 月　18 日

学籍番号：26002201991

氏名：園山佳典

クラス：D1

担当教員：　柴田史久・藤井康之

1 実験環境

統合開発環境 Android Studio を利用し開発を行う．Android Studio は Android アプリケーションの効率的な開発とテストを支援するための環境である．
基本的な開発環境は以下のとおりである．
1 Android Studio のインストール
2 新しいプロジェクトの作成（今回は New Project の Empty Views Activity を選択）
3 レイアウトエディタでのデザインとレイアウトの設計
4 コードの記述
5 ビルド設定の確認
6 エミュレータで実行

デバイスの仕様

デバイス名	TABLET-IQ5DJ1PS
プロセッサ	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz　2.42 GHz
実装 RAM	16.0 GB (15.8 GB 使用可能)
デバイス ID	C1101403-C4B8-4F93-B86E-0FBCAF214F16
プロダクト ID	00356-06269-91493-AAOEM
システムの種類	64 ビット オペレーティング システム，x64 ベース プロセッサ
ペンとタッチ	10 タッチ ポイントでのペンとタッチのサポート

Windows の仕様

エディション	Windows 11 Home
バージョン	22H2
インストール日	2023/04/07
OS ビルド	22621.2715
シリアル番号	0F01H7P214701J
エクスペリエンス	Windows Feature Experience Pack 1000.22677.1000.0

2 概要

今回作成したのは瓦割りゲームで，上下左右の矢印のボタンで瓦を動かし迫ってくるチョップの手に当たらないように避けるゲームである．チョップの手は画面上部から速いスピードで迫ってくるため，避けるためには素早い判断が必要となる．瓦の移動速度は左右方向は遅いが，上下方向はチョップの手よりも速く，上手く利用すれば避けやすくなるだろう．瓦がチョップの手に当たった場合，両者の動きは停止し瓦が割れたことを表現する画像とリスタートボタンが表示される．

3 実装内容

プログラムの仕組みについて説明する．
まず，プログラムを実行すると画面に上下左右の矢印のボタンとユーザが操作する瓦と画面上部のランダムな位置にチョップの手が表示される．
チョップの手は画面上部のランダムな位置から画面下部に向けて移動し，画面外に行くと再度画面上部のランダムな位置に現れ画面下部に向けて移動という動作を繰り返す．
上下左右の矢印のボタンを押すと瓦がその矢印の方向に動く．
瓦は画面の端で反射し，移動方向を反転する．
瓦とチョップの手が当たると両者の動きは停止する．
衝突後，瓦が割れたことを表現する画像とリスタートボタンを表示する．
リスタートボタンを押すと，プログラム実行時と同じ状態にする．

4 ソースコードの説明

①　【MySurfaceView.java】drawcanvas メソッド

```
//画像の中心座標
Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.kawara);
Bitmap resized_bitmap=Bitmap.createScaledBitmap(bitmap,300,300,true);
c.drawBitmap(resized_bitmap, (imgX - imgR)+x, (2*imgY - 2*imgR)+y, p);

//相手画像の中心座標
Bitmap bitmap_a = BitmapFactory.decodeResource(getResources(), R.drawable.atyo1);
Bitmap resized_bitmap_a=Bitmap.createScaledBitmap(bitmap_a,200,200,true);
c.drawBitmap(resized_bitmap_a, rnd_x, -imgR_a + y1, p);
```

②【MySurfaceView.java】drawcanvas メソッド

```
y1+=dy1;
  //画像を再生成
    if(y1>2*imgY) {
      rnd_x=(float)Math.random()*(1080-2*imgR_a);
      y1=0;
    }
```

③【MainActivity.java】onCreate メソッド

```
private ImageButton Lbutton
  //ボタンを取得
        Lbutton = findViewById(R.id.Lbutton);
  //ボタンのクリックリスナーを設定
    Lbutton.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        MySurfaceView.dx=-15;
        MySurfaceView.dy=0;
      }
    });
```

④【MySurfaceView.java】drawcanvas メソッド

```
//円が左右の壁に接触した際に移動方向を反転
    if (x<=-c.getWidth()/2+imgR||x>=c.getWidth()/2-imgR){
      dx*=-1;
    }
    //円が上下の壁に接触した際に移動方向を反転
    if(y<=-c.getHeight()+2*imgR||y>0){
      dy*=-1;
    }
```

⑤【MySurfaceView.java】drawcanvas メソッド

```
//当たり判定　if((Math.abs(center_w-center_a_w)<imgR+imgR_a)&&(Math.abs(center_h-
center_a_h)<imgR+imgR_a)){
        dx=0;
        dy=0;
```

```
        dy1=0;
        flag=true;
```
⑥【MySurfaceView.java】drawcanvas メソッド
```
//衝突後，ボタンを表示
 MainActivity.restart_button.post(new Runnable() {
          @Override
          public void run() {
            if(MySurfaceView.flag){
               MainActivity.restart_button.setVisibility(View.VISIBLE);
            }

          }
        });
//衝突後画像を表示
if(flag){
  Bitmap bitmap_phonto = BitmapFactory.decodeResource(getResources(), R.drawable.phonto);
  Bitmap    resized_bitmap_phonto=Bitmap.createScaledBitmap(bitmap_phonto,1000,300,true);
  c.drawBitmap(resized_bitmap_phonto, 50, 400, p);

Bitmap bitmap_kawarawari=BitmapFactory.decodeResource
(getResources(),R.drawable.kawarawari);
  Bitmap resized_bitmap_kawarawari=Bitmap.createScaledBitmap(bitmap_kawarawari,1000,700,true);
        c.drawBitmap(resized_bitmap_kawarawari, 50, 700, p);
     }
```
⑦【MainActivity.java】onCreate メソッド
```
restart_button.setOnClickListener(new View.OnClickListener(){
       @Override
       public void onClick(View v) {
          MySurfaceView.x=0;
          MySurfaceView.y=0;
          MySurfaceView.rnd_x=(float)Math.random()*(1080-2*MySurfaceView.imgR_a);
          MySurfaceView.y1=0;
          MySurfaceView.dy1=45;
          MySurfaceView.flag=false;
       }
    }
     );
```

① でキャンパスに瓦とチョップの手の画像を描画する．BitmapFactory.decodeResource で画像のリソースを指定し Bitmap.createScaledBitmap で画像のサイズを指定する．c.drawBitmap でサイズを指定した画像を引数に画像を描画する．

② でチョップの手の移動と，画面外に行った際に画面上部のランダムな位置に再生成する動作を行う．

③ でボタンを押すと瓦が動くとう操作を行う．onClick 内にボタンが押された時の処理を書く．③では左の矢印のボタンのみ示したが，右上下も同様に行う．

④ で瓦が画面の端で反射し，移動方向を反転する操作を行う．壁に当たった際に移動の変化量の符号を反転させることで反射を実装する．

⑤ で瓦とチョップの手の当たり判定を実装する．瓦とチョップの手のそれぞれの中心座標の差の絶対値がそれぞれの画像の半径の和より小さい場合に変化量をすべて 0 にする．

⑥ 衝突後，瓦が割れたことを表現する画像とリスタートボタンを表示する．
MainActivity.restart_button.setVisibility(View.VISIBLE);でボタンを表示する．衝突時に flag=true になるため，if(flag)内で衝突後の処理を書く．
⑦でリスタートボタンを押した際の処理を書く．リスタートボタンを押すと最初にプログラムを実行したときと同じ状態にするために，変数の値を初期化する．

5 結果

本課題の実行結果を図に示す．
図 1 はアプリを起動するとランダムな位置にチョップの手が生成されることを示す．
図 2 は図 1 の位置にあったチョップの手が画面上部から下部の方へ移動することを示す．
図 3 は左矢印のボタンを押すと瓦が左に動いている様子を示す．
図 4 は上矢印のボタンを押すと瓦が上に動いている様子を示す．
図 5 は瓦とチョップの手が当たると両者の動きが停止し瓦が割れたことを示す画像が表示される．
図 6 はリスタートボタンを押すと再度チョップの手が画面上部のランダムな位置に現れゲームが再開する．

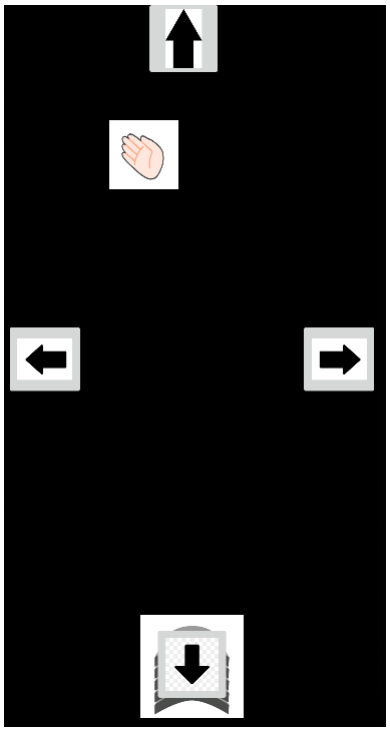

図 1　アプリの実行結果　初期状態

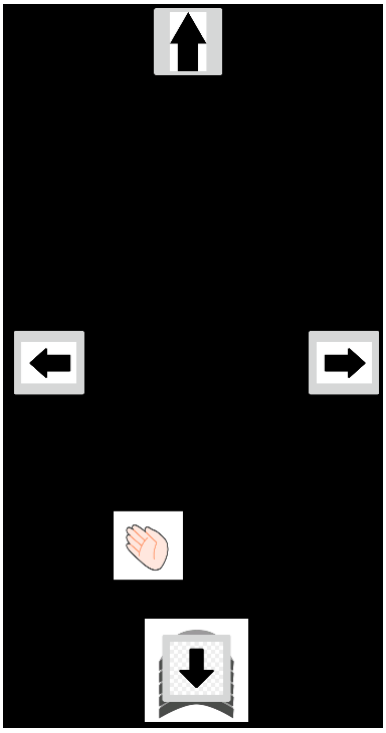

図２　アプリの実行結果　チョップの手の移動

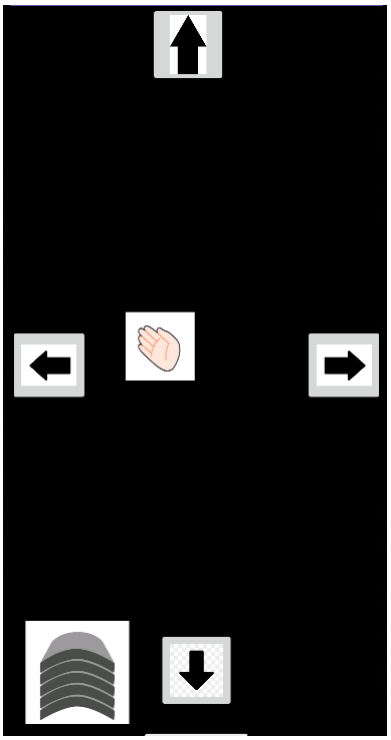

図３　アプリの実行結果　瓦の左右移動

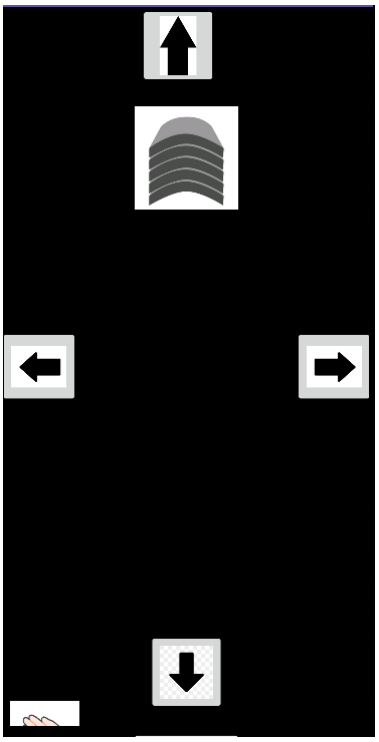

図 4　アプリの実行結果　瓦の上下移動

図 5　アプリの実行結果　衝突後の画面

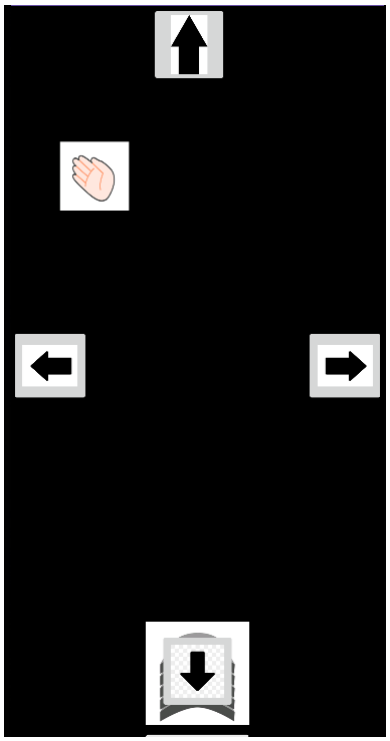

図 6 アプリの実行結果　ランダムな位置に再生成

6 考察

オリジナルアプリを作る際は作るアプリの方向性を決め実際に開発を進め，開発途中に様々な機能を追加していくと開発がスムーズになると感じた．Androidstudio のエミュレータがうまく起動しなかったり API が一致していないなどのエラーへの対処方法が身についた．

7 まとめ

7.1 モバイル端末プログラミングについての考察

Androidstudio でのアプリ開発では XML ファイルと Java ファイルを用いて開発し，両者の特徴や必要性を理解した．Androidstudio では様々な機能があり画像に動きをつけたり動画を用いたりすることができ使いこなすと開発できるアプリの幅は広いと感じた．

7.2 感想

実際にアプリを開発したことで，今まで学んできたプログラミングの知識や技術が役にたったことを実感した．モバイル端末プログラミングの授業を通して自分で一から時間をかけてアプリを作って完成したときの達成感や，アプリを作ることの面白さを感じた．

参考文献

https://qiita.com/QiitaD/items/3e14291a75599bda8409
https://codeforfun.jp/android-studio-catch-the-ball-5/