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

提出日：　2023 年　11　月　2　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

2　課題 3-3

黄色の背景色の画面中央に，半径 50 ピクセルの赤い円を描くプログラムを作成する．
作成した赤い円をアニメーションさせるプログラム（CanvasSample03）を作成せよ．
このプログラムは以下の仕様を満たすこととする．

- 初期状態では赤い円は画面中央にある
- スタートすると画面右方向に動き始め，画面の端に接触すると移動方向が反転する
- 画面をタッチすると左右方向の移動を停止し，下方向に移動を開始し，画面の端に接触すると移動方向が反転する
- 再度画面をタッチすると画面左右どちらかの方向（従前の方向）に動き始める ・ さらに再度画面をタッチすると画面上下どちらかの方向（従前の方向）に動き始める
- 以降，上記を繰り返す
-

2.1 実装内容

赤い円を画面中央に設定するため getWidth()，getHeight()を用いてその半分の値を取得する．
円の移動はスレッドによってアニメーションを実装させて行う．一定の時間間隔で画面を描画する処理を行うことでアニメーションを実装させる．円の移動を実装するために x に dx, y に dy という定数を加える処理を一定の時間間隔で繰り返し行う．円の移動方向を変えるために画面の端に接触すると，左右方向の場合 dx，上下方向の場合 dy の符号を変更する．画面をタッチすると発生する処理は onTouchEvent()を用いる．左右方向に動いている場合は dx を 0 にし，上下方向に動いている場合は dy を 0 にすることで動きを止める．また，従前の方向に移動させるために従前の方向を示す値を保持しておき動かす方向を決める．

2.2 ソースコードの説明

MySurfaceView.java の変更内容を一部抜粋する．

```
@Override
    public void surfaceCreated(SurfaceHolder holder) {
        thread = new Thread(this) ; /* 新しくスレッドを作成 */
        thread.start() ; /* スレッドをスタートさせる */
}

static final long FPS = 30 ;
static final long FTIME = 1000 / FPS ;
```

```
@Override
    public void run() {
        long loopC = 0 ; //  ループ数のカウンタ
        long wTime = 0 ; //  次の描画までの待ち時間（ミリ秒）
        long sTime = System.currentTimeMillis() ; //  開始時の現在時刻
        while (thread != null) {
            try {
                loopC++ ;
                drawCanvas() ; /* drawCanvas  メソッドで画面を描画  */
                wTime = (loopC * FTIME) - (System.currentTimeMillis() - sTime) ;
                if (wTime > 0) {
                    Thread.sleep(wTime) ;
                }
            } catch (InterruptedException e) {
         }
}
}
    private float x = 0 ;
    private float y = 0 ;
    private float dx = 15 ;//x 方向の変化量
    private float dy = 0 ;//y 方向の変化量
    float predx=15;
    float predy=15;
    private void drawCanvas() {
        Canvas c = sHolder.lockCanvas() ;
        c.drawColor(Color.YELLOW) ;
        Paint p = new Paint() ;
        p.setColor(Color.RED) ;
        x+=dx;//x 方向の移動
        y+=dy;//y 方向の移動
        c.drawCircle(c.getWidth()/2+x , c.getHeight() / 2+y ,50.0f, p);

        //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||x>=c.getWidth()/2){
            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||y>=c.getHeight()/2){
            dy*=-1;
        }
        sHolder.unlockCanvasAndPost(c) ;
    }

    //画面がタッチされた際の処理
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if(event.getAction()== MotionEvent.ACTION_DOWN) {
            //左右→上下
            if(dy==0){
                predx=dx;
                dx=0;
                dy=predy;
            }
            //上下→左右
```

```
            else{
                predy=dy;
                dx=predx;
                dy=0;
            }

        }
        return super.onTouchEvent(event) ; // 下位の View, Activity のタッチイベント
    }
```

drawCanvas()について説明する．
まず，SurfaceHolder クラスの lockCanvas()メソッドでほかのスレッドから操作されないように Canvas をロックする．drawColor で背景の色を設定し，Paint クラスのインスタンス p を生成する．
setColor で円の色を設定し drawCircle()で円を中央に配置する．x に定数 dx，y に定数 dy を加えることで移動を行う．左右の壁に衝突した際に移動方向を反転させるために if 文を用いて左右の壁に衝突すると dx，上下の壁に衝突すると dy の符号を変える．描画が終わると SurfaceHolder クラスの unlockCanvasAndPost メソッドで Canvas のロックを解除し，画面表示を更新する．これらの処理で画面を描画する drawCanvas()を実装する．

次にスレッドについて説明する．
surfaceCreated()内にスレッドを作成し，Surface が作成されるとスレッドを開始させる．スレッドの処理の中で run()メソッドで一定の時間間隔で画面を描画する処理を実行する．

run()メソッドについて説明する．
FPS は 1 秒あたりのフレーム数を表し，FTIME は 1 フレームあたりの時間を表す．
run()メソッドはスレッドが null でない限りループする．run()メソッド内で drawCanvas()を呼び出し画面描画を繰り返し行う．FPS を 30 に設定しているためそれを維持するために次のフレームまでの待機時間を計算する．

最後に onTouchEvent について説明する．
画面をタッチした際に行う処理を実装する．
If 文を用いて x 方向に動いている(y 方向の変化量が 0)場合とそれ以外(y 方向に動いている)で条件分岐する．x 方向に動いている場合は，動いている方向を記録しておくために predx=dx という操作を行う．その後，x 方向の動きを停止させ y 方向の変化量にに従前の方向の変化量を記録した predy を代入する．
x と同様に y 方向に動いている場合も行う．

2.3 結果

本課題の実行結果を図に示す．
図１はアプリを実行した際の初期状態であり画面中央に赤い円が位置していることがわかる．
図２はアプリを実行すると初期状態から x 方向の移動が行われていることを示す．
図３は図２で示した x 方向の移動の後壁に反射し反対方向に動いている様子を示す．
図４は画面をタッチすると x 方向の移動を停止し y 方向の移動を開始したことを示す．

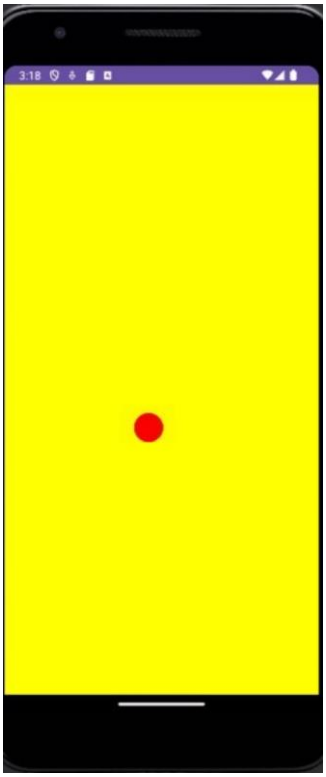

図１　アプリの実行結果　初期状態

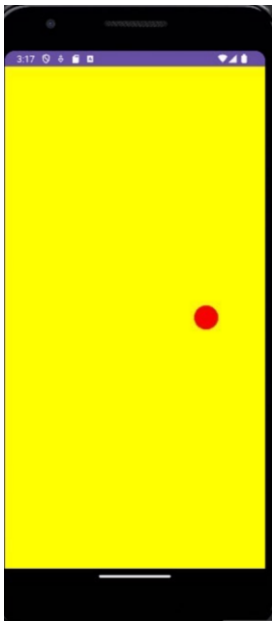

図２　アプリの実行結果　x 方向の移動

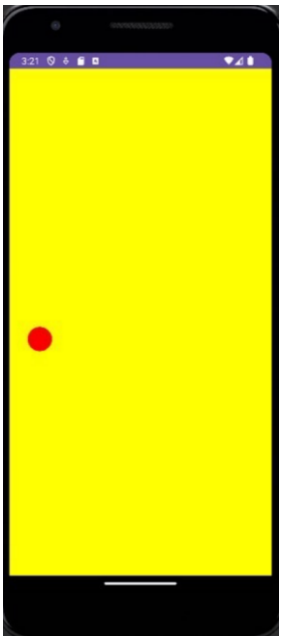

図３　アプリの実行結果　壁に反射後

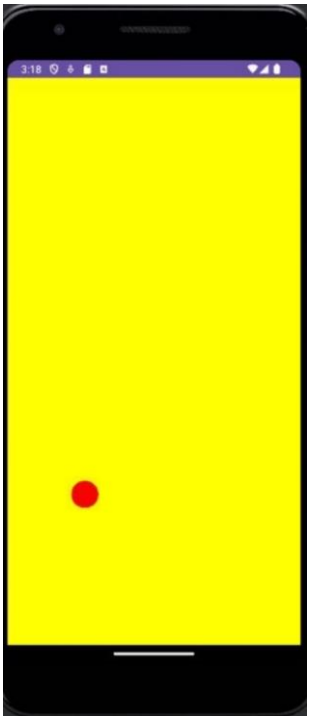

図４　アプリの実行結果　y 方向の移動

3　考察

今回の課題を通してアニメーションを実装する方法を学んだ.
また，30FPS で動かすにはどうするかなど具体的なコードの書き方を学んだ.
はじめ x 方向の変化量 dx と y 方向の変化量 dy を初期化しループごとに+1 をしてアニメーションを実装させようとしていて意図通りにいかなかった. その原因は 30FPS，つまり 1 秒間に 30 回画面を描画することでアニメーションを実装しているため変化量は定数にする必要があった. そこで dx，dy の値を定数にすることで滑らかに円が動くようになった.