ADDING CONDITIONAL STATEMENTS PART 1

This document complements the Steps provided in Week 3 for adding conditional statements.

Code referred to is from 'TheGame.java' file

Defining New Variables

It is needed to define new variables to implement the functionality we require, along with other variables such as mBallX and mBallY.

To do this, add a Bitmap to store mPaddle:

```
private Bitmap mPaddle;
```

Then add a variable to hold the X position of the Paddle (Note: it does not move in Y direction so we do not need a variable for the Y position):

```
private float mPaddleX = 0;
```

Constructing the Bitmap

This piece of code will have to be included in the 'TheGame(GameView gameView)' method:

```
mPaddle = BitmapFactory.decodeResource
      (gameView.getContext().getResources(),
            R.drawable.yellow_ball);
```

Tip: If you search for 'BitmapFactory' you will be able to find the right location to insert the code.

Set the initial position of the Paddle

Set the mPaddleX to the middle of the screen on the X axis:

```
mPaddleX = mCanvasWidth / 2;
```

This statement will need to be placed in the 'setupBeginning' method.

Tip: Search for 'setupBeginning' to locate the place in the file.

Drawing the Paddle

The statement below will need to be placed in the 'doDraw' method. You can place it after the first drwBitmap statement.

```
canvas.drawBitmap(mPaddle, mPaddleX - mPaddle.getWidth() / 2, mCanvasHeight -
mPaddle.getHeight() / 2, null);
```

The values placed inside the brackets are explained in the video.

The order of the parameters is: the bitmap, X position, Y position, null.

Note: Bitmaps are use the top left corner as the 0 point on the x and y axis (0,0). Therefore to have the paddle at the bottom of the screen Y value should be mCanvasHeight – mPaddle.getHeight()/2 .

Responding to Touch or Click

Insert this code to the 'actionOnTouch' method:

```
mPaddleX = x;
```

Responding to Phone Moving

Place this code in 'actionWhenPhoneMoved' method.

```
mPaddleSpeedX = mPaddleSpeedX + 70f * xDirection;
```

However, we have not defined mPaddleSpeedX. So define mPaddleSpeedX at the top along with other variables.

```
private float mPaddleSpeedX = 0;
```

Assign a value of zero to mPaddleSpeedX in the 'setupBeginning' method.

```
mPaddleSpeedX = 0;
```

Similar to the small ball we do not want the paddle going out of the screen either. So we add this code to the 'actionWhenPhoneMoved' method to keep it within the visible area.

```
if (mPaddleX < 0) mPaddleSpeedX = 0;
if (mPaddleX > mCanvasWidth) mPaddleSpeedX = 0;
```

Add this line of code to 'updateGame' method

```
mPaddleX = mPaddleX + secondsElapsed * mPaddleSpeedX;
```

Now add these statements to make sure mPaddleX will never be below zero or greater than mCanvasWidth.

After adding those statements the conditions will look similar to:

```
if (mPaddleX < 0){
        mPaddleSpeedX = 0;
        mPaddleX = 0;
}
   if (mPaddleX > mCanvasWidth){
          mPaddleSpeedX = 0;
          mPaddleX = mCanvasWidth;
   }
```

Notice the use of curly brackets to group statements. If there is just one statement following the if or else condition (for example mPaddleX = 0; in the case if(mPaddleX < 0)) it can be put in without using curly brackets. But if there is more than one statement, the use of curly brackets is essential – otherwise the program will not work as you would expect it to.

Check this code on your device or emulator.

You will see the paddle is a bit wobbly. Now let us correct that by checking the speed of the paddle along with its position as we did in the case of the little ball.

```
if ((mPaddleX <= 0) && (mPaddleSpeedX < 0)){
           mPaddleSpeedX = 0;
           mPaddleX = 0;
}
if ((mPaddleX >= mCanvasWidth) && (mPaddleSpeedX > 0)){
           mPaddleSpeedX = 0;
           mPaddleX = mCanvasWidth;
 }
```