Daily Journal

Name Dennis Nguyen

Date April 27, 2017

## Required work:

**11. DrawView - w/Sprite\*** ****

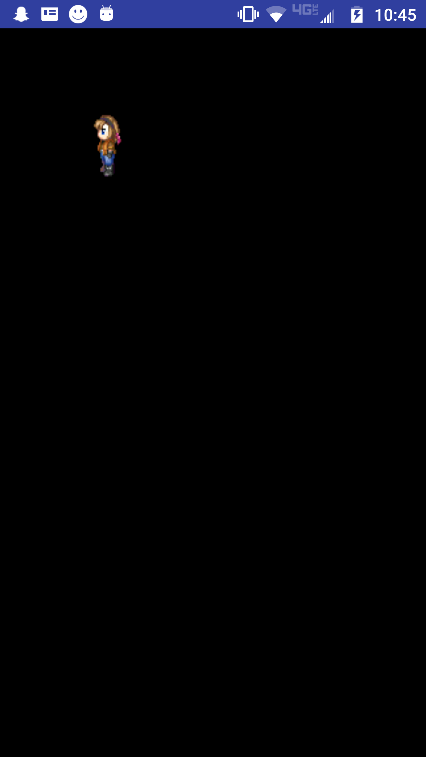
Add Sprite to DrawView while removing movement and drawing code

Remove x and xSpeed fields.

Declare Sprite object called sprite.

In onLayout(), instantiate a Sprite with heroBmp.(Don’t forget the 4 ints –left,top,right,bottom)

**public class** DrawView **extends** SurfaceView {  
  
 **private** Bitmap **heroBmp**; *// Declare space for Bitmap called heroBmp, global scope* **private** SurfaceHolder **holder**; *//Declares space for a SurfaceHolder called holder* **private** LoopThread **loopThread**; *//Declares space for a LoopThread called loopThread* **private** Sprite **sprite**; *//Creates space for character* **public** DrawView(Context context) { *// Constructor because it has the same name as the class* **super**(context); *//Calls View(context), Parent's constructor* **heroBmp** = BitmapFactory.*decodeResource*(getResources(), R.drawable.***bluejeans***); *//Instantiate heroBmp - assign to heroBmp for the first time* **holder** = getHolder();  
 **loopThread** = **new** LoopThread(**this**); *//Instantiate LoopThread with the current instance of DrawView* **holder**.addCallback(**new** SurfaceHolder.Callback() {  
 @Override  
 **public void** surfaceCreated(SurfaceHolder surfaceHolder) {  
 **loopThread**.setRunning(**true**); *//Sets the thread's running variable to true* **loopThread**.start(); *//Starts the thread* }  
  
 @Override  
 **public void** surfaceChanged(SurfaceHolder surfaceHolder, **int** i, **int** i1, **int** i2) {  
  
 }  
  
 @Override  
 **public void** surfaceDestroyed(SurfaceHolder surfaceHolder) {  
 **boolean** retry = **true**; *//Declares boolean called retry and instantiates to true, local variable* **loopThread**.setRunning(**false**); *//Sets the thread's running variable to false* **while** (retry) {  
 **try** {  
 **loopThread**.join(); *//Blocks the current thread until this instance's thread terminates* retry=**false**;  
 } **catch** (InterruptedException e) {  
  
 }  
 }  
 }  
 });  
 }  
  
 @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
  
 canvas.drawColor(Color.***BLACK***); *//Draws black over the canvas* **sprite**.draw(canvas); *//Modify and display sprite* }  
  
 @Override  
 **protected void** onLayout(**boolean** changed, **int** left, **int** top, **int** right, **int** bottom) {  
 **super**.onLayout(changed, left, top, right, bottom);  
 **sprite** = **new** Sprite(0, 0, (**float**).1 \* getWidth(), (**float**).1 \* getHeight(), **heroBmp**); *//Creates character with initial values* }  
}

**1.**       **12. Sprite - animation (parts of bitmap)** \* 

Use this [link](http://www.mmorpgmakerxb.com/p/characters-sprites-generator) to create a set of icons to simulate animation.

([Advanced icons](http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator/))

Replace your heroBmp with this .png file.

Add code to display one icon at a time. Then cycle through a row of icons.

Add static constants for the number of rows and columns (value will be constant for all sprites)

Declare and instantiate an int currentFrame at 0.

Declare two ints to represent the width and height of one icon (1 figure in bitmap of images)(iWidth, iHeight)

In constructor, calculate the iWidth of each icon by dividing the width of bitmap by number of columns.

In constructor, calculate the iHeight of each icon by dividing the height of bitmap by number of rows

In update(), increment the currentFrame by 1 (remember to use modular division to control your range)

{Source rectangle represents a portion of heroBmp that will be drawn. 1 icon of 12 available in heroBmp.}

In draw(), declare and instantiate int srcX to define the x value of the rectangle of each icon based on the currentFrame

                Declare and instantiate int srcY to define the y value for row 2.

                Declare and instantiate the source Rect using srcX and srcY.

Draw the heroBmp using the source Rect.

**public class** Sprite **extends** RectF {  
  
 **public** Sprite (**float** left, **float** top, **float** right, **float** bottom, Bitmap heroBMP) {  
 **super**(left, top, right, bottom);  
 **this**.**heroBMP** = heroBMP;  
 **iWidth** = heroBMP.getWidth() / *BMP\_COLS*; *//Calculate width of 1 icon* **iHeight** = heroBMP.getHeight() / ***BMP\_ROWS***; *//Calculate height of 1 icon* }  
 **public static final** Creator <RectF> ***CREATOR*** = **null**;  
 **private int xSpeed** = 5, **ySpeed** = 10; *//Declare and instantiate* Bitmap **heroBMP**; *//Received bitmap stores instance heroBMP* **private static final int *BMP\_ROWS*** = 4; *//Number of rows on sprite sheet* **private static int** *BMP\_COLS* = 4; *//Number of columns on sprite sheet* **private int currentFrame** = 0; *//Frame to be drawn* **private int iWidth**, **iHeight**; *//Dimensions of 1 icon on sprite sheet* **public void** update(Canvas c) {  
 **if** (**right** + **xSpeed** >= c.getWidth()) {**xSpeed** \*=-1;} *//Invert xSpeed and ySpeed when boundaries reached (bounce off walls)* **else if** (**left** + **xSpeed** <= 0) {**xSpeed** \*= -1;}  
 **if** (**bottom** + **ySpeed** >= c.getHeight()) {**ySpeed** \*= -1;}  
 **else if** (**top** + **ySpeed** <= 0) {**ySpeed** \*= -1;}  
 offset(**xSpeed**, **ySpeed**); *//Increment x and y directions* **currentFrame** = ++**currentFrame** % *BMP\_COLS*; *//Advanced to next frame, returns to 0 when past max* }  
  
 **public void** draw(Canvas c) {  
 **int** srcX = **currentFrame** \* **iWidth**; *//Set x of current icon* **int** srcY = 1 \* **iHeight**; *//Set y to 2 rows of icons* Rect src = **new** Rect(srcX, srcY, srcX + **iWidth**, srcY + **iHeight**); *//Define rectangle to be drawn (1 icon)* update(c); *//Modify Sprite* c.drawBitmap(**heroBMP**, src, **this**, **null**);  
 }  
  
}

**1.**       **13. Sprite - rnd speed\*** ****

Add random speed.

In the constructor, generate a random speed for each sprite.

**package** com.dragonfury.duy.p4a14nguyendennisanimatedgame;  
  
**import** android.graphics.Bitmap;  
**import** android.graphics.Canvas;  
**import** android.graphics.Rect;  
**import** android.graphics.RectF;  
  
*/\*\*  
 \* Created by 1383504 on 4/25/2017.  
 \*/***public class** Sprite **extends** RectF {  
  
 **public** Sprite (**float** left, **float** top, **float** right, **float** bottom, Bitmap heroBMP) {  
 **super**(left, top, right, bottom);  
 **this**.**heroBMP** = heroBMP;  
 iWidth = heroBMP.getWidth() / *BMP\_COLS*; *//Calculate width of 1 icon* **iHeight** = heroBMP.getHeight() / ***BMP\_ROWS***; *//Calculate height of 1 icon* }  
 **public static final** Creator <RectF> ***CREATOR*** = **null**;  
 **private int xSpeed** = (**int**)(Math.*random*() \* 51 - 25), **ySpeed** = (**int**)(Math.*random*() \* 51 - 25); *//Random intSpeed from -25 to 25* Bitmap **heroBMP**; *//Received bitmap stores instance heroBMP* **private static final int *BMP\_ROWS*** = 4; *//Number of rows on sprite sheet* **private static int** *BMP\_COLS* = 4; *//Number of columns on sprite sheet* **private int currentFrame** = 0; *//Frame to be drawn* **private int** iWidth, **iHeight**; *//Dimensions of 1 icon on sprite sheet* **public void** update(Canvas c) {  
 **if** (**right** + **xSpeed** >= c.getWidth()) {**xSpeed** \*=-1;} *//Invert xSpeed and ySpeed when boundaries reached (bounce off walls)* **else if** (**left** + **xSpeed** <= 0) {**xSpeed** \*= -1;}  
 **if** (**bottom** + **ySpeed** >= c.getHeight()) {**ySpeed** \*= -1;}  
 **else if** (**top** + **ySpeed** <= 0) {**ySpeed** \*= -1;}  
 offset(**xSpeed**, **ySpeed**); *//Increment x and y directions* **currentFrame** = ++**currentFrame** % *BMP\_COLS*; *//Advanced to next frame, returns to 0 when past max* }  
  
 **public void** draw(Canvas c) {  
 **int** srcX = **currentFrame** \* iWidth; *//Set x of current icon* **int** srcY = 1 \* **iHeight**; *//Set y to 2 rows of icons* Rect src = **new** Rect(srcX, srcY, srcX + iWidth, srcY + **iHeight**); *//Define rectangle to be drawn (1 icon)* update(c); *//Modify Sprite* c.drawBitmap(**heroBMP**, src, **this**, **null**);  
 }  
  
}

Daily Journal

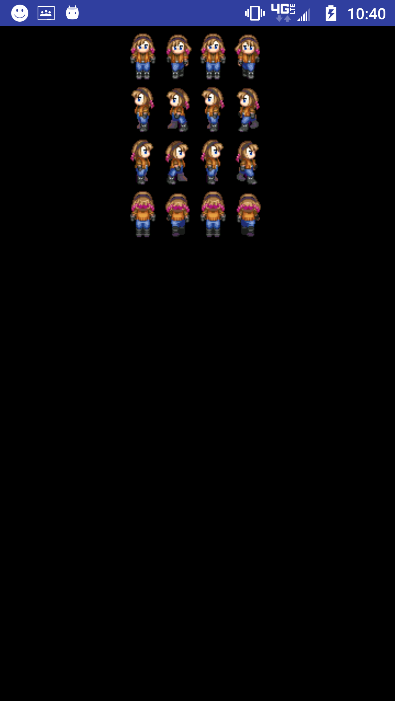
Name Dennis Nguyen

Date April 25, 2017

## Required work:

**8. LoopThread - w/FramesPerSecond** ****

1. Add a timer to maintain consistent timing between movements.
2. Declare and instantiate class constant called FPS =10.
3. In run(), declare and instantiate a long called ticksPS to 1000 divided by FPS.
   1. Declare longs called startTime and sleepTime.
   2. After c is declared, instantiate the startTime.
   3. Instantiate sleepTime as ticksPS minus the amount of time that has passed

**package** com.dragonfury.duy.p4a14nguyendennisanimatedgame;  
**import** android.graphics.Canvas;  
  
*/\*\*  
 \* Created by 1383504 on 4/21/2017.  
 \*/***public class** LoopThread **extends** Thread {   
  
 **private** DrawView **view**; *//Declares space for DrawView obj called view* **private boolean running** = **false**; *//Declare and instantiate a boolean called running to false* **public static int** *FPS* = 10; *//Declares int and instantiate to 10* **public** LoopThread(DrawView view){ *//Constructor, receives DrawView* **this**.**view** = view; *//Assigns and receives DrawView into the global view* }  
  
 **public void** setRunning(**boolean** running) { *//Setter to allow public access to boolean running* **this**.**running** = running;  
 }  
  
 **public void** run() { *//Every thread must have a run method* **long** tickPS = 1000/*FPS*; *//Number of seconds to complete each loop* **long** startTime = System.*currentTimeMillis*(); *//Current system time in milliseconds* **long** sleepTime;  
  
 **while**(**running**){ *//Loop as long as running is true* Canvas c = **null**; *//Declares space for Canvas called c, local variable* **try** {  
 c = **view**.getHolder().lockCanvas(); *//Locks canvas* **synchronized** (**view**.getHolder()) {  
 **view**.onDraw(c); *//Calls drawView's onDraw()* }  
 }**finally** {  
 **if**(c != **null**) { *//Unlocks the canvas* **view**.getHolder().unlockCanvasAndPost(c);  
 }  
 }  
  
 sleepTime = tickPS - (System.*currentTimeMillis*() - startTime); *//Calculate the amount of time requird to use up tickPS* **try** {  
 **if** (sleepTime>0) *sleep*(sleepTime); *//Pause to use up sleepTime***else** *sleep*(10);  
 }**catch** (InterruptedException e) {}  
 }  
  
 }  
}

1. Use sleep() to pause the thread until the required ticksPS has passed.  If ticksPS has already expired, pause for 10 ms
2. **9. DrawView - w/xSpeed** \* 
3. Add speed and border check.
4. Declare and instantiate global int called xSpeed to 5.
5. Invert xSpeed when bmp reaches left and right boundaries.
6. **package** com.dragonfury.duy.p4a14nguyendennisanimatedgame;  
     
   **import** android.graphics.BitmapFactory;  
   **import** android.graphics.Canvas;  
   **import** android.graphics.Color;  
   **import** android.view.SurfaceView;  
   **import** android.view.SurfaceHolder;  
   **import** android.content.Context;  
   **import** android.graphics.Bitmap;  
     
   */\*\*  
    \* Created by 1383504 on 4/19/2017.  
    \*/***public class** DrawView **extends** SurfaceView {  
     
    **private** Bitmap **heroBmp**; *// Declare space for Bitmap called heroBmp, global scope* **private** SurfaceHolder **holder**; *//Declares space for a SurfaceHolder called holder* **private** LoopThread **loopThread**; *//Declares space for a LoopThread called loopThread* **private int x** = 0;  
    **private int xSpeed** = 5;  
     
    **public** DrawView(Context context) { *// Constructor because it has the same name as the class* **super**(context); *//Calls View(context), Parent's constructor* **heroBmp** = BitmapFactory.*decodeResource*(getResources(), R.drawable.***bluejeans***); *//Instantiate heroBmp - assign to heroBmp for the first time* **holder** = getHolder();  
    **loopThread** = **new** LoopThread(**this**); *//Instantiate LoopThread with the current instanc eof DrawView* **holder**.addCallback(**new** SurfaceHolder.Callback() {  
    @Override  
    **public void** surfaceCreated(SurfaceHolder surfaceHolder) {  
    **loopThread**.setRunning(**true**); *//Sets the thread's running variable to true* **loopThread**.start(); *//Starts the thread* }  
     
    @Override  
    **public void** surfaceChanged(SurfaceHolder surfaceHolder, **int** i, **int** i1, **int** i2) {  
     
    }  
     
    @Override  
    **public void** surfaceDestroyed(SurfaceHolder surfaceHolder) {  
    **boolean** retry = **true**; *//Declares boolean called retry and instantiates to true, local variable* **loopThread**.setRunning(**false**); *//Sets the thread's running variable to false* **while** (retry) {  
    **try** {  
    **loopThread**.join(); *//Blocks the current thread until this instance's thread terminates* } **catch** (InterruptedException e) {  
     
    }  
    }  
    }  
    });  
    }  
     
    @Override  
    **protected void** onDraw(Canvas canvas) {  
    **super**.onDraw(canvas);  
     
    canvas.drawColor(Color.***BLACK***); *//Draws black over the canvas  
    //if (x < getWidth() - heroBmp.getWidth()) x++; //If x left of edge of screen minus width of bmp, increase 1* **if** (**x** + **heroBmp**.getWidth() + **xSpeed** > getRight()) **xSpeed**\*=-1;*//If x and width of bmp passes right edge, invert xSpeed* **if** (**x** < getLeft()) **xSpeed** \*=-1; *//If x passes left edge, invert xSpeed* **x**+=**xSpeed**; *//Increment x by xSpeed* canvas.drawBitmap(**heroBmp**, **x**, 10 \* 2560 / getHeight(), **null**); *// Draw heroBmp at (10, 10)* }  
   }

**1.**      **10. Sprite** ****

Creating the sprite class and letting it control its’ movement and drawing.

Create the Sprite class which extends RectF

                Implement the static CREATOR field:  public static final Creator<RectF> CREATOR=null;

Declare and instantiate an int called xSpeed and ySpeed to 5 and 10 respectively.

Create a constructor that receives four int’s, and a Bitmap.  Store the Bitmap.

Create the update() method.

Enter the boundary bouncing conditions (top,left,bottom,right).

Offset the instance by xSpeed and ySpeed.

Create the draw() method.

                Call the update() method

                Draw the bitmap into this sprite

**package** com.dragonfury.duy.p4a14nguyendennisanimatedgame;  
  
**import** android.graphics.Bitmap;  
**import** android.graphics.Canvas;  
**import** android.graphics.RectF;  
  
*/\*\*  
 \* Created by 1383504 on 4/25/2017.  
 \*/***public class** Sprite **extends** RectF {  
  
 **public** Sprite (**float** left, **float** top, **float** right, **float** bottom, Bitmap heroBMP) {  
 **super**(left, top, right, bottom);  
 **this**.**heroBMP** = heroBMP;  
 }  
 **public static final** Creator <RectF> ***CREATOR*** = **null**;  
 **private int xSpeed** = 5, **ySpeed** = 10; *//Declare and instantiate* Bitmap **heroBMP**; *//Received bitmap stores instance heroBMP* **public void** update(Canvas c) {  
  
 }  
  
 **public void** draw(Canvas c) {  
 update(c); *//Modify Sprite* c.drawBitmap(**heroBMP**, **null**, **this**, **null**);  
 }  
  
}

Daily Journal

Name: Dennis Nguyen

Date: April 21, 2017

## Required work:

**DrawView -> SurfaceView**

Convert DrawView to a SurfaceView for faster rendering.  SurfaceView will allow complete control over the screen.

Add SurfaceHolder to lock and unlock the canvas.

                In constructor, instantiate holder.

                Add Callback() to holder.

                                Implement abstract methods: surfaceDestroyed(), surfaceCreated(), and surfaceChanged().

**public class** DrawView **extends** SurfaceView {  
  
 **private** Bitmap **heroBmp**; *// Declare space fir Bitmap called heroBmp, global scope* **private** SurfaceHolder **holder**; *//Declares space for a SurfaceHolder called holder* **public** DrawView(Context context) { *// Constructor because it has the same name as the class* **super**(context); *//Calls View(context), Parent's constructor* **heroBmp** = BitmapFactory.*decodeResource*(getResources(), R.drawable.***bluejeans***); *//Instantiate heroBmp - assign to heroBmp for the first time* **holder** = getHolder();  
 **holder**.addCallback(**new** SurfaceHolder.Callback() {  
 @Override  
 **public void** surfaceCreated(SurfaceHolder surfaceHolder) {  
  
 }  
  
 @Override  
 **public void** surfaceChanged(SurfaceHolder surfaceHolder, **int** i, **int** i1, **int** i2) {  
  
 }  
  
 @Override  
 **public void** surfaceDestroyed(SurfaceHolder surfaceHolder) {  
  
 }  
 });  
 }  
  
 @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
 canvas.drawColor(Color.***BLACK***);  
 canvas.drawBitmap(**heroBmp**, 10 \* 1440 / getWidth(), 10 \* 2560 / getHeight(), **null**); *// Draw heroBmp at (10, 10)* }  
}

**LoopThread**

Create LoopThread extending Thread to allow multiple sprites to behave simultaneously.

Declare global space for a DrawView.

Declare a global Boolean called running and instantiate to false.

Set the constructor to receive a DrawView.

Create a settor for running.

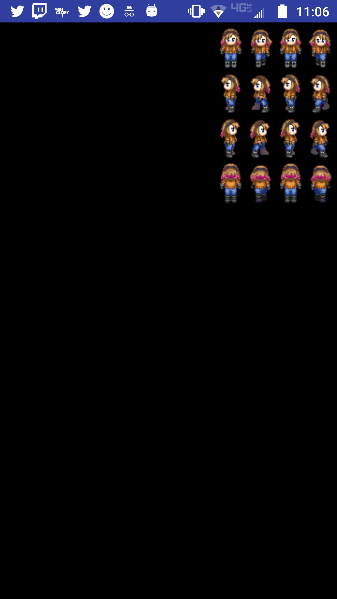
Implement the run() method

                Set a loop to repeat until running is false

                Declare space for a Canvas

                Try to get and lock the canvas from the view, call the view’s onDraw()

                                In the end unlock the canvas

**public class** LoopThread **extends** Thread {  
  
 **private** DrawView **view**; *//Declares space for DrawView obj called view* **private boolean running** = **false**; *//Declare and instantiate a boolean called running to false* **public** LoopThread(DrawView view){ *//Constructor, receives DrawView* **this**.**view** = view; *//Assigns and receives DrawView into the global view* }  
  
 **public void** setRunning(**boolean** running) { *//Setter to allow public access to boolean running* **this**.**running** = running;  
 }  
  
 **public void** run() { *//Every thread must have a run method* **while**(**running**){ *//Loop as long as running is true* Canvas c = **null**; *//Declares space for Canvas called c, local variable* **try** {  
 c = **view**.getHolder().lockCanvas(); *//Locks canvas* **synchronized** (**view**.getHolder()) {  
 **view**.onDraw(c);  
 }  
 }**finally** {  
 **if**(c != **null**) {  
 **view**.getHolder().unlockCanvasAndPost(c);  
 }  
 }  
 }  
  
 }  
}

**DrawView - w/LoopThread and horizontal increment \***

Declare space for a LoopThread called loopThread and an int called x.  Instantiate the x to 0.

In surfaceCreated(), set loopThread’s running to true and start loopThread.

In surfaceDestroyed(), declare and instantiate Boolean retry to true.

                Set loopThread’s running to false

                Use the join() to pause the current thread until loopThread terminates (must use try{}catch{} to handle exception)

                                Set retry to false

In onDraw(), draw the heroBmp at (x,10).

                Move the bitmap 1 to the right until bitmap reaches edge of screen

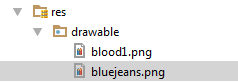
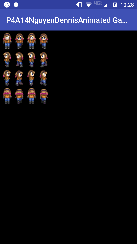
**public class** DrawView **extends** SurfaceView {  
  
 **private** Bitmap **heroBmp**; *// Declare space fir Bitmap called heroBmp, global scope* **private** SurfaceHolder **holder**; *//Declares space for a SurfaceHolder called holder* **private** LoopThread **loopThread**; *//Declares space for a LoopThread called loopThread* **private int x** = 0;  
  
 **public** DrawView(Context context) { *// Constructor because it has the same name as the class* **super**(context); *//Calls View(context), Parent's constructor* **heroBmp** = BitmapFactory.*decodeResource*(getResources(), R.drawable.***bluejeans***); *//Instantiate heroBmp - assign to heroBmp for the first time* **holder** = getHolder();  
 **loopThread** = **new** LoopThread(**this**);  
  
 **holder**.addCallback(**new** SurfaceHolder.Callback() {  
 @Override  
 **public void** surfaceCreated(SurfaceHolder surfaceHolder) {  
 **loopThread**.setRunning(**true**);  
 **loopThread**.start();  
 }  
  
 @Override  
 **public void** surfaceChanged(SurfaceHolder surfaceHolder, **int** i, **int** i1, **int** i2) {  
  
 }  
  
 @Override  
 **public void** surfaceDestroyed(SurfaceHolder surfaceHolder) {  
 **boolean** retry = **true**;  
 **loopThread**.setRunning(**false**);  
 **while** (retry) {  
 **try** {  
 **loopThread**.join();  
 } **catch** (InterruptedException e) {  
  
 }  
 }  
 }  
 });  
 }  
  
 @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
  
 canvas.drawColor(Color.***BLACK***);  
 **if** (**x** < getWidth() - **heroBmp**.getWidth()) {  
 **x**++;  
 }  
 canvas.drawBitmap(**heroBmp**, x, 10 \* 2560 / getHeight(), **null**); *// Draw heroBmp at (10, 10)* }  
}

Daily Journal Name Dennis Nguyen Date April 19, 2017

## Required work:

1. MainActivity
   * Create MainActivity class extending AppCompatActivity.
     + In onCreate(), set the content view to the new DrawView instance.

**public class** MainActivity **extends** AppCompatActivity {  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) { *//Android lifecycle* **super**.onCreate(savedInstanceState);  
 setContentView(**new** DrawView(**this**)); *//Instantiate DrawView* }  
}

1. **DrawView – View\*** ****
   * Copy your personal images to the res/drawable folder. (See Assignment Button to download the images)
     + Remember file names must contain only lowercase **a-z, 0-9, or \_**.
   * Create the DrawView class.
     + In the constructor, process the bluejeans.png to bitmap.
     + In onDraw() draw black over the view and draw the bitmap.

**public class** DrawView **extends** SurfaceView {  
  
 **private** Bitmap **heroBmp**; *// Declare space fir Bitmap called heroBmp, global scope*  
  
 **public** DrawView(Context context) { *// Constructor because it has the same name as the class* **super**(context); *//Calls View(context), Parent's constructor* **heroBmp** = BitmapFactory.*decodeResource*(getResources(), R.drawable.***bluejeans***); *//Instantiate heroBmp - assign to heroBmp for the first time*   
 @Override  
 **protected void** onDraw(Canvas canvas) {  
 **super**.onDraw(canvas);  
  
 canvas.drawColor(Color.***BLACK***);  
 canvas.drawBitmap(**heroBmp**, **10**, 10 \* 2560 / getHeight(), **null**); *// Draw heroBmp at (10, 10)* }  
}

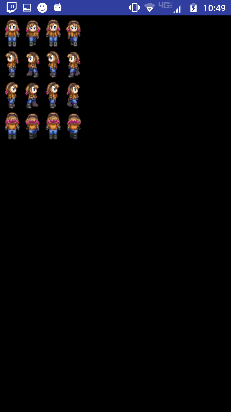
* + Test App! (Screenshot into journal) [Do this after steps 2,3, and 4.]

1. **AndroidManifest.xml – Locking Orientation to Portrait** \* 

<**activity android:name=".MainActivity" android:screenOrientation="portrait" android:configChanges="keyboardHidden|orientation|screenSize"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"**/>  
  
 <**category android:name="android.intent.category.LAUNCHER"**/>  
 </**intent-filter**>  
</**activity**>

* + In res subdirectory, open AndroidManifest.xml
  + In <**activity** **android:name=".MainActivity"**> insert android:screenOrientation="portrait" and android:configChanges="keyboardHidden|orientation|screenSize".

1. **styles.xml – Removing the title bar** \* 

<**style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar"**>

<**resources**>  
  
 *<!-- Base application theme. -->* <**style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar"**>  
 *<!-- Customize your theme here. -->* <**item name="colorPrimary"**>@color/colorPrimary</**item**>  
 <**item name="colorPrimaryDark"**>@color/colorPrimaryDark</**item**>  
 <**item name="colorAccent"**>@color/colorAccent</**item**>  
 </**style**>  
  
</**resources**>

* + In res/values subdirectory, open styles.xml
  + Change **.DarkActionBar** to **.NoActionBar**.