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.

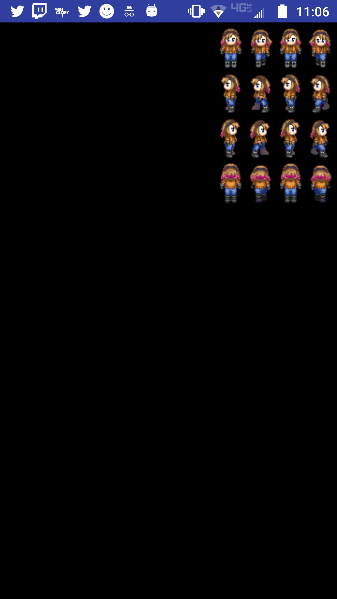
                Set loopThread’s running to false

                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)* }  
}