ECE 535: Mobile Development & Ubiquitous Computing Systems

Instructor: Dr. Paul Watta

Name: Srinivas Sambaraju

Date: Dec 16, 2013

Final Project Tutorial

Objective

The objective of this tutorial is to illustrate the usage of Animation in Android using Threading and Sensors with a simple example.

Thread

A thread can be used in a situation where continuous execution is required for a task. In this case if everything is done on the main UI thread, it will become unresponsive. So the repetitive and time consuming task can be done in the background using a Thread. Animation is a good scenario to use Threads since the screen needs to be redrawn multiple times.

Animation

The App sets Orientation to Landscape mode. When we start the App, we will see a blue bird flying across the screen from left to right. The coolest thing is that it also flaps its wings as it moves. In this example we create Animation using 'gif' files on a canvas. The thread enables us to draw the flying bird (which is a gif file) in increments along the canvas from one end of the phone to the other.

The Movie class can be used to read the 'gif' file as an input stream into a 'Movie' object and then it is played on the Canvas. A custom class keeps track of the bird progress. We check if the bird scrolled off the screen, so that we can create a new bird. The background need not be painted using the thread. Only the gif is painted again and again as it moves on. The Hardware Acceleration property should be set to false in the Manifest to play the 'gif' animation on the phone.

Orientation (Tilt) Sensor

We also have a SensorEventListener to keep track of the sensor changes. Using a simple calculation we can determine the Orientation or Tilt using the values of Acceleration and Magnetic field. Once we know the Tilt we need to use it to move the bird. In this App, when the Phone is Left side down – bird goes down, Right side down – bird goes up.

This App has been tested on a Samsung Galaxy S4. The Sensors are not available on the Emulator, so testing is possible only on a phone. The thread is created inside a view that implements a SensorEventListener.

Applications

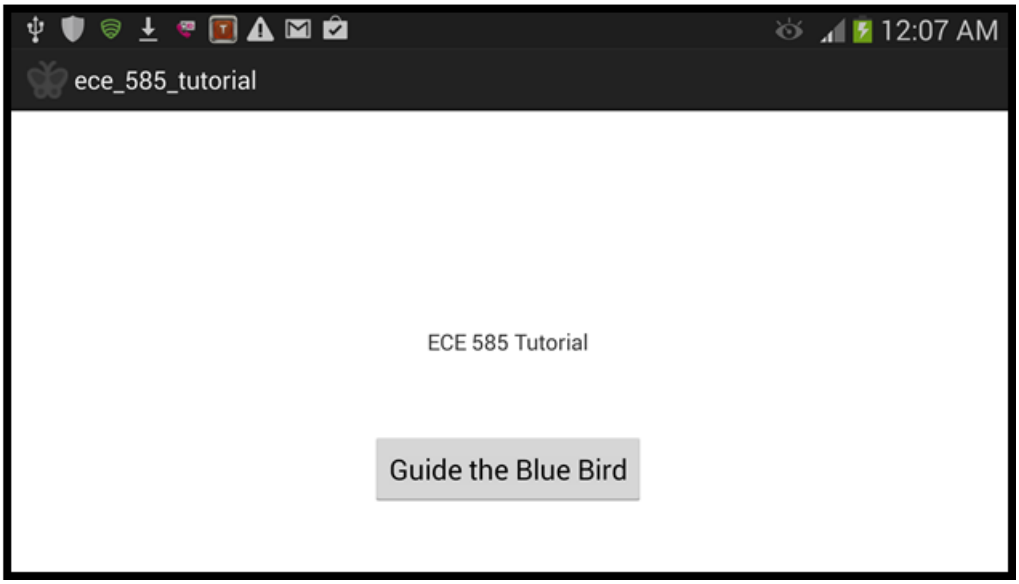
Applications of this example can be in games like getting the bird to fly by navigating obstacles in its path using the Screen tilts or an Arrow and Target game where the Arrow needs to fly towards a target.

Things not handled in the Tutorial

There are multiple things to keep in mind when using threads. The main idea of the Tutorial is to have an animation that plays continuously until the App is closed. Therefore, in this example, Thread stopping, and Memory optimization are not handled optimally. Some ideas are to use a Timer Task to automatically stop the thread after some time or to keep track of the bird's position and stop the thread after it scrolls off the screen etc.

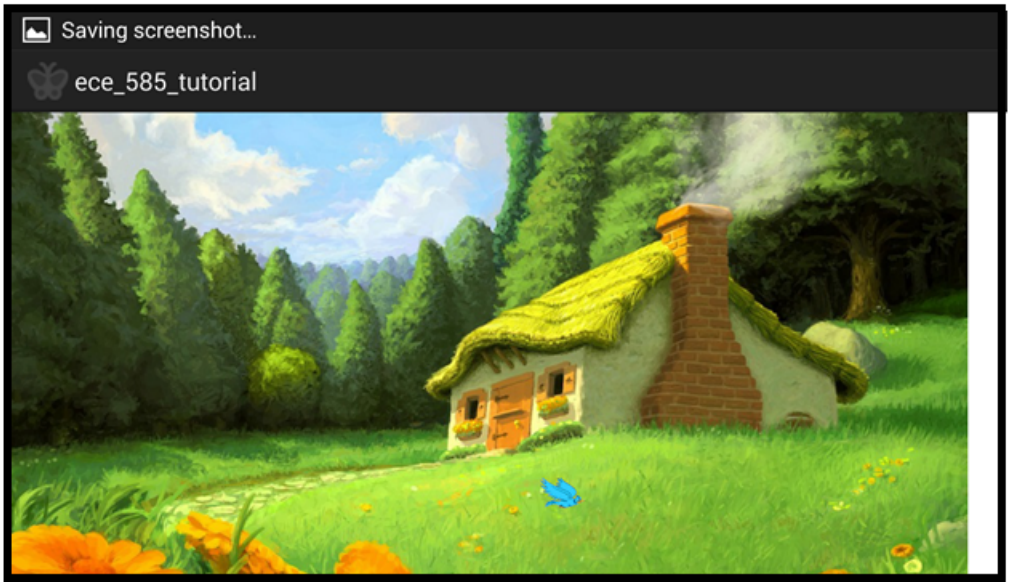
Screen Shots of the animation

Three Screen shots are presented here. First is the home screen to start the Animation. The second and third show the bird in flight. The bird is a gif, so its flaps its wings as well. The wing motion can be seen when screens two and three are compared.

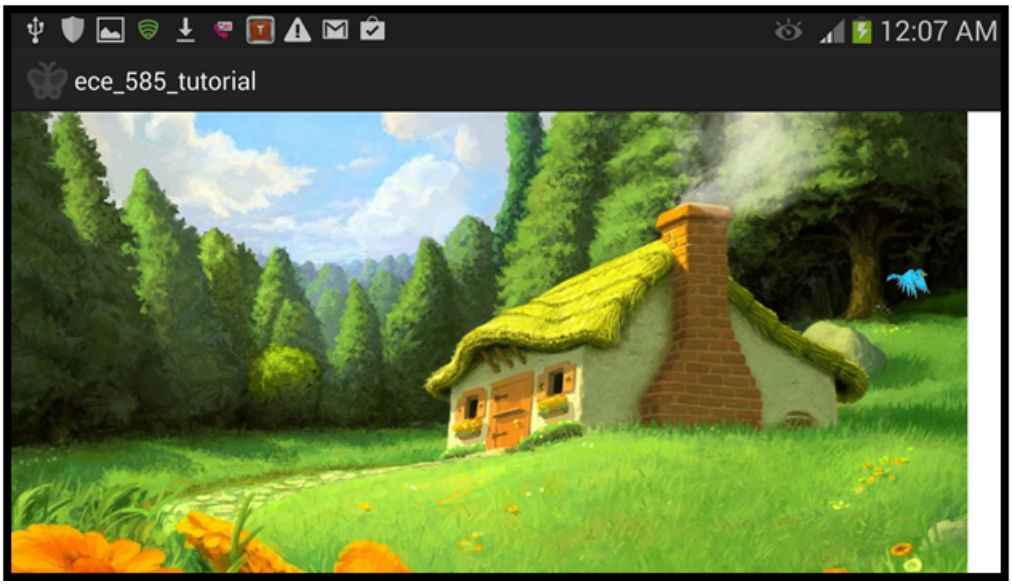


Home Screen

Animation in Android using Threading and Sensors



Blue bird is visible over the grass



Blue bird visible on the right of the house

Code

Programmed by – Srinivas Sambaraju
Date – Dec 16, 2013
Final Project Tutorial

Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.ece_585_tutorial"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/butterfly_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.ece_585_tutorial.TutorialActivity"
            android:label="@string/app_name"
            android:screenOrientation="landscape"
            android:hardwareAccelerated="false">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".TutorialActivity" >

    <TextView
```

```
        android:id="@+id/myTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/myTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="48dp"
        android:onClick="buttonClick"
        android:text="@string/button_text" />

</RelativeLayout>
```

Strings

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">ece_585_tutorial</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">ECE 585 Tutorial</string>
    <string name="button_text">Guide the Blue Bird</string>

</resources>
```

Main Program

The Bitmaps are defined in the Constructor and the animations and sensors are read in the thread continuously.

```
package com.example.ece_585_tutorial;

import java.io.InputStream;
import java.util.Vector;

import android.app.Activity;
import android.content.Context;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Movie;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
```

```
import android.hardware.SensorManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.SystemClock;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.Menu;
import android.view.View;

public class TutorialActivity extends Activity {

	String TAGV = "Tutorial";
	boolean mInitialized = false;
	public long beginTime, currentTime;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu); // activity_thread_example
		return true;
	}

	public void buttonClick(final View view) {
		CustomView cv = new CustomView(view.getContext());
		setContentView(cv);
	}

	public class CustomView extends View implements SensorEventListener {

		private Movie mMovie;
		String TAGX = "TutAct";
		private long mMoviestart = 1000;

		Bitmap mBckgroundImage;
		Bitmap mBirdImage;
		Resources mRes;

		DisplayMetrics metrics;
		private int mCanvasWidth;
		private int mCanvasHeight;
		private int mArrTarLimitY;
		private int mPixelMoveX = 3;
		private Vector<Bird> mBirdVec;
		private boolean CreateNextBird = true;
		private int mBirdHeight;

		private SensorManager mSensorManager;
		private Sensor mAccelerometer;
		private Sensor mMagnetometer;
```

```
	float[] mGravity;
	float[] mGeomagnetic;
	float[] mOrientation;

	Handler m_handler;
	Runnable m_handlerTask;

	public CustomView(Context context) {
		super(context);
		// TODO Auto-generated constructor stub
		mRes = getResources();
		mBckgroundImage = BitmapFactory.decodeResource(mRes,
				R.drawable.for_butterfly);
		mBirdImage = BitmapFactory.decodeResource(mRes,
				R.drawable.blue_flying_bird);

		metrics = getResources().getDisplayMetrics();
		mCanvasWidth = metrics.widthPixels;
		mCanvasHeight = metrics.heightPixels;
		mArrTarLimitY = mCanvasHeight / 4;
		mBirdHeight = mBirdImage.getHeight();

		mSensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
		mAccelerometer = mSensorManager
				.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
		mMagnetometer = mSensorManager
				.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
		mOrientation = new float[3];
		mSensorManager.registerListener((SensorEventListener) this,
				mAccelerometer, SensorManager.SENSOR_DELAY_UI);
		mSensorManager.registerListener((SensorEventListener) this,
				mMagnetometer, SensorManager.SENSOR_DELAY_UI);

		m_handler = new Handler();
	}

	protected void onDraw(Canvas canvas) {
		super.onDraw(canvas);
		canvas.drawColor(Color.TRANSPARENT);
		canvas.drawBitmap(mBckgroundImage, 0, 0, null);
		move(canvas);
		this.invalidate();// Very important for animation
	}

	public void move(final Canvas canvas) {

		m_handlerTask = new Runnable() {
			@Override
			public void run() {

				//canvas.drawBitmap(mBckgroundImage, 0, 0, null);
				InputStream stream1 = null;
				try {
```

```
                                        stream1 =
getContext().getResources().openRawResource(
                                                R.drawable.blue_flying_bird);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                        Log.d(TAGV, " error: : " + e.getMessage());
                                }
                                if (!mInitialized) {
                                        setInitialGameState();
                                        mInitialized = true;
                                }
                                currentTime = SystemClock.uptimeMillis();
                                setGifAnimation(currentTime, stream1);
                                doBirdAnimation(canvas);
                                updateGameState();
                        }// run
                };// runnable
                m_handlerTask.run();
        }

        public void setGifAnimation(long now, InputStream stream) {
                Log.d(TAGX, " setGifAnimation: : ");

                try {
                        mMovie = Movie.decodeStream(stream);
                } catch (Exception e) {
                        e.printStackTrace();
                }
                if (mMoviestart == 0) {
                        mMoviestart = now;
                }
                int relTime = (int) ((now - mMoviestart) % mMovie.duration());
                mMovie.setTime(relTime);
        }

        public void setInitialGameState() {

                mBirdVec = new Vector<Bird>();
                doBirdCreation();
        }

        public void updateGameState() {

                if (CreateNextBird) {
                        doBirdCreation();
                }
                updateBird();
        }

        private void doBirdCreation() {

                Bird _bird = new Bird();
                _bird.mDrawY = mCanvasHeight / 2;
                _bird.mDrawX = 0;
                _bird.mStartTime = System.currentTimeMillis();
```

```
		mBirdVec.add(_bird);
	}

	protected void updateBird() {

		if (mBirdVec == null | mBirdVec.size() == 0)
			return;

		Bird curBird = mBirdVec.elementAt(0);
		double birdTiltAngle = (90 * mOrientation[1]) / 1.1;
		// mBirdImage.getHeight()
		double newYINcr = mBirdHeight
				* Math.tan(Math.toRadians(birdTiltAngle));
		int newYVal = curBird.mDrawY + (int) newYINcr;

		currentTime = SystemClock.uptimeMillis();
		if ((currentTime - beginTime) > 300) {

			beginTime = SystemClock.uptimeMillis();

			if (newYVal < mArrTarLimitY) {
				curBird.mDrawY = mArrTarLimitY;
			} else if (newYVal > 3 * mArrTarLimitY) {
				curBird.mDrawY = 3 * mArrTarLimitY;
			} else {
				curBird.mDrawY = newYVal;
			}
		}
		// Update the asteroids position, even missed ones keep moving
		curBird.mDrawX += mPixelMoveX; // -=

		if (curBird.mDrawX > (mCanvasWidth - 100)) {
			mBirdVec.removeElementAt(0);
			CreateNextBird = true;
		} else {
			CreateNextBird = false;
		}
	}

	private void doBirdAnimation(Canvas canvas) {

		if ((mBirdVec == null | mBirdVec.size() == 0))
			return;
		Bird curBird = mBirdVec.elementAt(0);
		mMovie.draw(canvas, curBird.mDrawX, curBird.mDrawY);
	}

	@Override
	public void onAccuracyChanged(Sensor arg0, int arg1) {
	}

	// sensor methods
	@Override
	public void onSensorChanged(SensorEvent event) {
```

```
                if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
                        mAccelVal = event.values;
                if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD)
                        mMagneticField = event.values;
                if (mAccelVal != null && mMagneticField != null) {
                        float RotMat[] = new float[9];
                        float InclMat[] = new float[9];

                        boolean success = SensorManager.getRotationMatrix(RotMat,
                                        InclMat, mAccelVal, mMagneticField);
                        if (success) {
                                SensorManager.getOrientation(RotMat, mOrientation);
                        }
                }
        }

        public void unregisterListener(SensorEventListener listener) {
                // Unregisters a listener for the sensors with which it is
                // registered.
                mSensorManager.unregisterListener(listener);
        }

        protected void onDestroy() {
                unregisterListener(this);
        }

    }
}
```

Code Auxiliary to main Program

This is used to define the location of the bird.

```
package com.example.ece_585_tutorial;
public class Bird {

       public int mDrawY = 0;
    public int mDrawX = 0;
    public long mStartTime = 0;
}
```

References

http://www.hdwallpapersplus.com/scenery-wallpapers.html
http://www.netanimations.net/Dog-and-puppy-running-playing-eating-animated-gif.htm
http://droid-blog.net/2011/10/14/tutorial-how-to-use-animated-gifs-in-android-part-1/
http://www.codingforandroid.com/2011/01/using-orientation-sensors-simple.html