# 1. Scope

The tutorial will utilize 2D graphics in android to develop a full screen interactive music player application. It targets Android 2.1 (Eclair) handsets and assumes **Android 2.1-update1 (API Level 7) or above**. Proceeding from previous labs, it will also continue to strengthen the understanding of event handling and working with permissions.

# 2. Pre-requisites

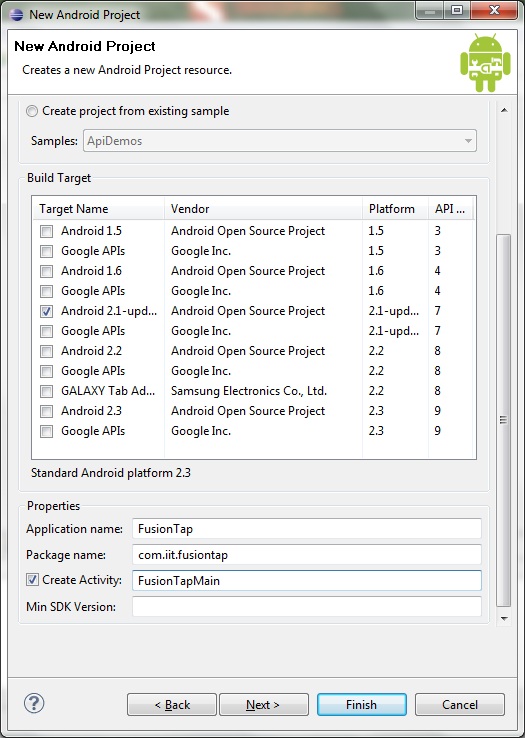
* Java JDK 6, Eclipse (v3.5 Galileo is preferred), Android SDK (2.1 or above) and a suitable AVD (Android Virtual Device).

## 2.1. Overview of the Application

The application will play different music samples based on the point at which the finger is placed on the display. Being a full screen application, it will use a virtual grid which is divided into 6 virtual compartments. First, it will attempt to locate the compartment at which the finger is placed and will play a music sample accordingly. Therefore the coordinates (in *x* and *y* axes) of each touch point is determined. At a given time a user can play multiple samples by touching at multiple locations on the screen. Hence, create their own music using the provided samples.

# 3. Create the Android Project

Create a new Android Project (**File -> New -> Other -> Android Project**). Click **Next** on the first dialog and continue to fill out project details. For project name and application name provide ‘***FusionTap***’ and for package name ‘***com.iit.fusiontap***’. Specify ‘***FusionTapMain***’ as the Activity name and select Android 2.1-update1 or above as the build target. Rest of the configurations can be left as default as shown on **Figure1**.

****

**Figure 1:** Project Details

# 4. Copy the Resource Files

Create a **new folder** named ‘***drawable***’ under res folder of the project and copy all the resource files. Change the icon file specified on the android manifest to use ‘***app\_ico.png***’ as the icon for the application.

# resoc.jpgcopied_resources.jpg

**Figure 2:** Copying the resource files

# 5. The Layout

Typically the applications that solely use 2D graphics, the layout file can be left empty (i.e. without any UI elements) and later they can be introduced via code when required. Also in contrast to the previous labs, the background for this application will be drawn instead of using ‘***android:background***’ property. Hence, the application will use an empty layout file and the default **TextView** element provided on the **main.xml** also needs to be **removed**. Once the TextView tag is removed the file will only have the following content.

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

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

</LinearLayout>

**Note:** Also include the following ‘***android:theme***’ property to the **AndroidManifest.xml** for it to work in full screen mode.

<activity android:name=*".FusionTapMain"*

android:label=*"@string/app\_name"*

android:screenOrientation=*"portrait"* android:theme=*"@android:style/Theme.NoTitleBar.Fullscreen"*>

# 6. Creating the Menu

The menu will hold two menu items, ‘***About***’ and ‘***Close***’. Create a new folder named **menu** under res folder and create a new XML file named ‘***menu.xml***’ and introduce the menu items via the Layout perspective of this file. The file will contain the following once the menu items are introduced.

<menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>

<item android:id=*"@+id/item1"* android:icon=*"@drawable/about"* android:title=*"About"*/>

<item android:id=*"@+id/item2"* android:icon=*"@drawable/close"* android:title=*"Close"*/>

</menu>

# 7. Implementing the Main Activity

The code for the Main Activity (i.e. FusionTapMain) along with all the imports and variables can be found below. It creates instances of the **PowerManager** to keep the phone without sleeping and sets its view to **SimulationView** that will make use of 2D graphics. SimulationView will be implemented as an **inner class** so that it **will reside within the FusionTapMain class** itself. Once they are complete, the code for handling the menu items can be worked out.

**package** com.iit.fusiontap;

**import** com.iit.fusiontap.R;

**import** android.app.Activity;

**import** android.app.AlertDialog;

**import** android.content.Context;

**import** android.content.DialogInterface;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.graphics.Canvas;

**import** android.graphics.Paint;

**import** android.graphics.Typeface;

**import** android.graphics.BitmapFactory.Options;

**import** android.media.MediaPlayer;

**import** android.media.MediaPlayer.OnCompletionListener;

**import** android.os.Bundle;

**import** android.os.PowerManager;

**import** android.os.PowerManager.WakeLock;

**import** android.util.DisplayMetrics;

**import** android.view.Menu;

**import** android.view.MenuInflater;

**import** android.view.MenuItem;

**import** android.view.MotionEvent;

**import** android.view.View;

**import** android.view.WindowManager;

**public** **class** FusionTapMain **extends** Activity {

**private** SimulationView mSimulationView;

**private** PowerManager mPowerManager;

**private** WindowManager mWindowManager;

**private** WakeLock mWakeLock;

**private** **static** String *eventName* = "";

**private** **static** **int** *x* = 0;

**private** **static** **int** *y* = 0;

**private** **static** **int** *points* = 0;

**private** **static** **int** *isPlayingOne* = 0;

**private** **static** **int** *isPlayingTwo* = 0;

**private** **static** **int** *isPlayingThree* = 0;

**private** **static** **int** *isPlayingFour* = 0;

**private** **static** **int** *isPlayingFive* = 0;

**private** **static** **int** *isPlayingSix* = 0;

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

// Get an instance of the PowerManager

mPowerManager = (PowerManager) getSystemService(*POWER\_SERVICE*);

// Get an instance of the WindowManager

mWindowManager = (WindowManager) getSystemService(*WINDOW\_SERVICE*);

// Create a bright wake lock

mWakeLock = mPowerManager.newWakeLock(

PowerManager.*SCREEN\_BRIGHT\_WAKE\_LOCK*, getClass().getName());

// instantiate our simulation view and set it as the activity's content

mSimulationView = **new** SimulationView(**this**);

setContentView(mSimulationView);

}

//**TODO:Create the inner class (SimulationView)**

//**TODO:Handle menu items**

}

# 8. Creating the View (uses 2D graphics)

Typically the ***onDraw()*** method will play a crucial role in applications that use 2D graphics. Here the method is used to draw the background along with the event type and coordinates at which the last event was triggered. This method will be called continuously to redraw/repaint the view. The **code for the View** (i.e. SimulationView) which is an inner class of FusionTapMain can be found below.

**class** SimulationView **extends** View {

**private** **float** mXDpi;

**private** **float** mYDpi;

**private** **float** mMetersToPixelsX;

**private** **float** mMetersToPixelsY;

**private** Bitmap mWood;

**public** SimulationView(Context context) {

**super**(context);

DisplayMetrics metrics = **new** DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(metrics);

mXDpi = metrics.xdpi;

mYDpi = metrics.ydpi;

mMetersToPixelsX = mXDpi / 0.0254f;

mMetersToPixelsY = mYDpi / 0.0254f;

Options opts = **new** Options();

opts.inDither = **true**;

opts.inPreferredConfig = Bitmap.Config.*RGB\_565*;

mWood = BitmapFactory.*decodeResource*(getResources(),

R.drawable.*fusion\_bg*, opts);

}

@Override

**public** **boolean** onTouchEvent(MotionEvent event) {

**int** action = event.getAction();

// get point of the triggered action

*x* = (**int**) event.getX();

*y* = (**int**) event.getY();

*points* = event.getPointerCount();

**switch** (action) {

**case** (MotionEvent.*ACTION\_DOWN*):

// touch pressed

*eventName* = "Pressed at ";

**if** (*x* > 0 && *x* <= 240 && *y* > 0 && *y* <= 220 && *isPlayingOne* == 0) {

playMusic(1);

} **else** **if** (*x* > 0 && *x* <= 240 && *y* > 220 && *y* <= 440

&& *isPlayingTwo* == 0) {

playMusic(2);

} **else** **if** (*x* > 0 && *x* <= 240 && *y* > 440 && *isPlayingThree* == 0) {

playMusic(3);

} **else** **if** (*x* > 240 && *y* > 0 && *y* <= 220 && *isPlayingFour* == 0) {

// X column 2

playMusic(4);

} **else** **if** (*x* > 240 && *y* > 220 && *y* <= 440 && *isPlayingFive* == 0) {

playMusic(5);

} **else** **if** (*x* > 240 && *y* > 440 && *isPlayingSix* == 0) {

playMusic(6);

}

**break**;

**case** (MotionEvent.*ACTION\_UP*):

// touch released

*eventName* = "Released";

**break**;

**case** (MotionEvent.*ACTION\_MOVE*):

// moved on screen

*eventName* = "Moved";

**break**;

**case** (MotionEvent.*ACTION\_CANCEL*):

// canceled

*eventName* = "Canceld";

**break**;

}

**return** **super**.onTouchEvent(event);

}

**private** **void** playMusic(**int** num) {

**if** (num == 1) {

*isPlayingOne* = 1;

MediaPlayer mp = MediaPlayer.*create*(getBaseContext(),

R.drawable.*a1*);

mp.start();

mp.setOnCompletionListener(**new** OnCompletionListener() {

@Override

**public** **void** onCompletion(MediaPlayer mp) {

*isPlayingOne* = 0;

mp.release();

}

});

} **else** **if** (num == 2) {

//**TODO**

} **else** **if** (num == 3) {

//**TODO**

}

**if** (num == 4) {

*isPlayingFour* = 1;

MediaPlayer mp = MediaPlayer.*create*(getBaseContext(),

R.drawable.*b1*);

mp.start();

mp.setOnCompletionListener(**new** OnCompletionListener() {

@Override

**public** **void** onCompletion(MediaPlayer mp) {

*isPlayingFour* = 0;

mp.release();

}

});

} **else** **if** (num == 5) {

//**TODO**

} **else** **if** (num == 6) {

//**TODO**

}

}

@Override

**protected** **void** onDraw(Canvas canvas) {

canvas.drawBitmap(mWood, 0, 0, **null**);

Paint pt = **new** Paint();

pt.setAntiAlias(**true**);

pt.setColor(0xFFFFFFFF);

pt.setTextSize(24);

pt.setTypeface(Typeface.*SERIF*);

canvas.drawText(*eventName* + " x:" + *x* + " y:" + *y*, 160, 765, pt);

canvas.drawText("Developeed at IIT", 160, 800, pt);

// and make sure to redraw as soon as possible

invalidate();

}

}

# 9. Handling the Menu Items

The code below is used for handling the menu items. It allows to close the application or to display an About dialog when the menu items are pressed.

**public** **boolean** onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.*menu*, menu);

**return** **true**;

}

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**switch** (item.getItemId()) {

**case** R.id.*item1*:

AlertDialog.Builder alert = **new** AlertDialog.Builder(**this**);

alert.setCancelable(**true**);

alert.setTitle("Fusion Tap");

alert.setMessage("An interactive music player."

+ " Developed at IIT.");

alert.setPositiveButton("OK",

**new** DialogInterface.OnClickListener() {

**public** **void** onClick(DialogInterface dialog,

**int** whichButton) {

}

});

alert.show();

**break**;

**case** R.id.*item2*:

finish();

**break**;

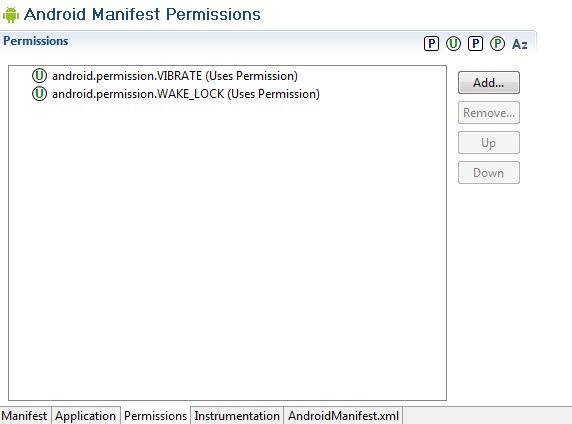
}

**return** **true**;

}

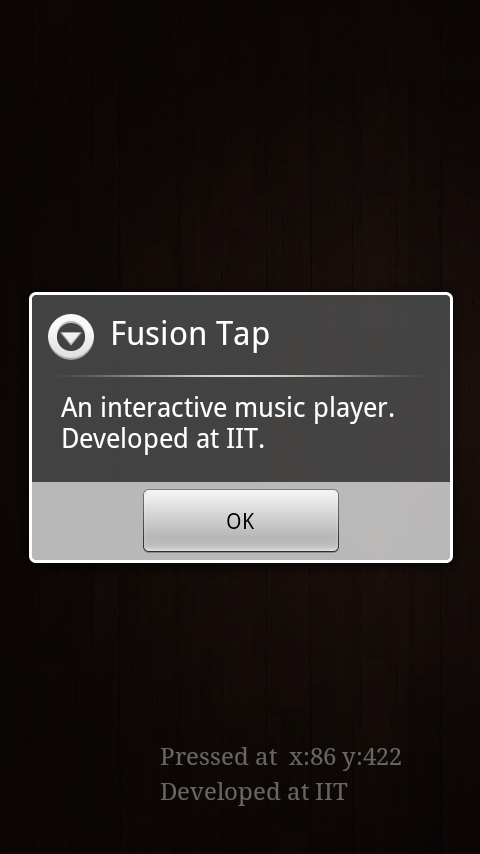
# 9. Permissions

The application requires two special permissions to keep the phone active and make it vibrate. These can be added using permissions perspective of the android manifest as shown in **Figure 3**.



**Figure 3:** Adding the permissions required for the application

# 10. Screen shot

**Figure 4:** A screen shot of ‘*Fusion Tap*’