Introduction

Two-dimensional graphics in Android are implemented differently than dynamic, interactive, or three-dimensional graphics. Android has 2D drawing APIs for creating this kind of graphics. Using these APIs, there are two ways of implementing graphics:

- •Drawing to a View
- •Drawing on a Canvas

Drawing to a View

To draw a simple graphic, one which does not require dynamic changes in the application, extend the View class and define an onDraw() callback method.

The onDraw() method is passed with the Canvas, which can be used to draw a simple graphics using Canvas methods.

Displaying an Image and Text in a Normal View

To display an image and text in a normal View, create a new Project with the activity named ViewGraphicsActivity

```
public class ViewGraphicsActivity extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                requestWindowFeature(Window.FEATURE_NO_TITLE);
                setContentView(new MyView(this));
        }
}
```

The requestWindowFeature(Window.FEATURE_NO_TITLE) method is added to show a window without a title. Next, create an inner class named MyView which extends from the View class and overrides the method onDraw(Canvas)

```
class MyView extends View {
    public MyView(Context context) {
        super(context);
    }

    @Override
    public void onDraw(Canvas canvas) {
    }
}
```

The next step is to implement an onDraw(Canvas) callback method. For drawing you need a Paint object through which you can set Text properties like color, size, font, and style:

```
Paint paint = new Paint();
paint.setColor(Color.WHITE);
```

```
paint.setTextSize(20);
paint.setAntiAlias(true);
```
Now make canvas background color blue using the drawColor(Color) method.

```
canvas.drawColor(Color.BLUE);
```
Then using the paint object, we draw the text as follows:

```
canvas.drawText("Hello Android", 10, 20, paint);
```
To draw a bitmap image on the canvas, get the bitmap image from the resource:

```
Bitmap image = BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
canvas.drawBitmap(image, 40, 80, null);
```
Example

The given example illustrates the display of an image and text in a normal View.

```
package com.view.graphics;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Bundle;
import android.view.View;
import android.view.Window;

public class ViewGraphicsActivity extends Activity {
         @Override
         public void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  requestWindowFeature(Window.FEATURE_NO_TITLE);
                  setContentView(new MyView(this));
         }

         class MyView extends View {
                  public MyView(Context context) {
                           super(context);
                  }

                  @Override
                  public void onDraw(Canvas canvas) {
                           Paint paint = new Paint();
```

```
                        paint.setColor(Color.WHITE);
                        paint.setTextSize(20);
                        paint.setAntiAlias(true);
                        canvas.drawColor(Color.BLUE);
                        canvas.drawText("Hello Android", 10, 20, paint);
                        Bitmap image = BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher);
                        canvas.drawBitmap(image, 40, 80, null);
                }
        }
}
```

Drawing on a Canvas

To draw dynamic 2D graphics where in your application, draw on the Canvas by extending SurfaceView. SurfaceView is a subclass of View where a dedicated drawing surface is offered within the View hierarchy. Here drawing is done in the application's secondary thread instead of waiting to be drawn by the system's view hierarchy.

Your class should also implement a SurfaceHolder.Callback interface to get notification of the underlying surface events such as surface created, surface changed, and surface destroyed. These events help your application to determine when to start and stop drawing.

Displaying an Image and Text in a Normal View

Create a new Project with the activity named MySurfaceViewActivity. This project displays an image and text on the Canvas. The example shows how an image can be moved dynamically in the canvas. The sample code listens for the touch events and renders an image on the touched coordinates of the screen.

```
public class MySurfaceViewActivity extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                requestWindowFeature(Window.FEATURE_NO_TITLE);
                setContentView(new MySurface(this));
        }
}
```

Next, create an inner class that extends SurfaceView and implements SurfaceHolder.callback and implement the three methods as shown below:

```
class MySurface extends SurfaceView implements SurfaceHolder.Callback {
    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {}

    @Override
    public void surfaceCreated(SurfaceHolder holder) {}
```

```
    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {}
}
```

Next, implement the callback methods and call the application’s secondary thread inside a constructor

```
class MySurface extends SurfaceView implements SurfaceHolder.Callback {
       private SecondThread thread;
       //Initial position of the image
       private int x = 100;
       private int y = 200;
       public MySurface(Context context) {
                   super(context);
                   getHolder().addCallback(this);
                   thread = new SecondThread(getHolder(), this);
       }

         @Override
         public void surfaceChanged(SurfaceHolder holder, int format, int width, int height)
{
         }

         @Override
         public void surfaceCreated(SurfaceHolder holder) {
                  thread.setRunning(true);
                  thread.start();
         }

         @Override
         public void surfaceDestroyed(SurfaceHolder holder) {
                  boolean retry = true;
                  thread.setRunning(false);
                  while (retry) {
                           try {
                                    thread.join();
                                    retry = false;
                           } catch (InterruptedException e) {}
                  }
         }
}
```

Create the secondary thread in your application where the constructor requires parameters like mySurface and the SurfaceHolder.

```
class SecondThread extends Thread {
```

```
      private SurfaceHolder surfaceHolder;
      private MySurface mySurface;
      private boolean _run = false;

      public SecondThread(SurfaceHolder surfaceHolder, MySurface mySurface) {
            this.surfaceHolder = surfaceHolder;
            this.mySurface = mySurface;
      }

      public void setRunning(boolean run) {
                  _run = run;
      }

      @Override
      public void run() {
                   Canvas c;
                   while (_run) {
                                  c = null;
                                  try {
                                           c = surfaceHolder.lockCanvas(null);
                                           synchronized (surfaceHolder) {
                                                    mySurface.onDraw(c);
                                           }
                                  } finally {
                                           if (c != null) {
                                                    surfaceHolder.unlockCanvasAndPost(c);
                                           }
                                  }
                    }
        }
}
```

Now draw the graphics using the Canvas object:

```
public void onDraw(Canvas canvas) {
  Paint paint = new Paint();
  paint.setColor(Color.WHITE);
  paint.setTextSize(20);
  paint.setAntiAlias(true);
  canvas.drawColor(Color.BLUE);
  canvas.drawText("Hello Android", 10, 20, paint);

  Bitmap image = BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher);
  //renders image using x and y parameter x and y value is filled by the touch //event
```

```
  canvas.drawBitmap(image, x, y, null);
}
```

The code renders the image using x, y parameters. The x, y parameters values are filled using touch events listener as shown in the following code:

```
@Override
public boolean onTouchEvent(MotionEvent event) {
if (event.getAction() == MotionEvent.ACTION_MOVE) {
                x = (int) event.getX();
                y = (int) event.getY();
        }
        return true;
}
```

Example

The given example illustrates the display of an image and text on Canvas:

```
package com.surface.view;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Bundle;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.Window;

public class MySurfaceViewActivity extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                requestWindowFeature(Window.FEATURE_NO_TITLE);
                setContentView(new MySurface(this));
        }

        class MySurface extends SurfaceView implements SurfaceHolder.Callback {
                private SecondThread thread;
                private int x = 100;
                private int y = 200;
                private boolean isMoving;
```

```
                public MySurface(Context context) {
                        super(context);
                        getHolder().addCallback(this);
                        thread = new SecondThread(getHolder(), this);
                }

                @Override
                public boolean onTouchEvent(MotionEvent event) {
                        if (event.getAction() == MotionEvent.ACTION_MOVE) {
                                x = (int) event.getX();
                                y = (int) event.getY();
                                isMoving = true;
                        } else if (event.getAction() == MotionEvent.ACTION_UP) {
                                isMoving = false;
                        }
                        return true;
                }

                @Override
                public void onDraw(Canvas canvas) {
                        Paint paint = new Paint();

                        paint.setColor(Color.WHITE);

                        paint.setTextSize(20);

                        paint.setAntiAlias(true);
                        canvas.drawColor(Color.BLUE);
                        canvas.drawText("Hello Android", 10, 20, paint);
                        canvas.drawText("Exampe to draw dynamically on canvas", 10, 60,
paint);
                        canvas.drawText("DRAG ICON TO MOVE", 10, 100, paint);
                        Bitmap image = BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher);
                        if (isMoving)
                         canvas.drawText("ICON IS MOVING", 10, 130, paint);
                        else
                         canvas.drawText("ICON IS NOT MOVING", 10, 130, paint);
                        canvas.drawText("x is " + x + ": y is " + y, x - 20, y - 20,
paint);
                        canvas.drawBitmap(image, x, y, null);
        }
```

```
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
        int height) {}

        @Override
        public void surfaceCreated(SurfaceHolder holder) {
                 thread.setRunning(true);
                 thread.start();
        }

        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
                 boolean retry = true;
                 thread.setRunning(false);
                 while (retry) {
                          try {
                                   thread.join();
                                   retry = false;
                          } catch (InterruptedException e) {}
                          }
                 }
        }

class SecondThread extends Thread {
        private SurfaceHolder surfaceHolder;
        private MySurface mySurface;
        private boolean _run = false;

        public SecondThread(SurfaceHolder surfaceHolder, MySurface mySurface) {
                          this.surfaceHolder = surfaceHolder;
                          this.mySurface = mySurface;
        }

        public void setRunning(boolean run) {
                          _run = run;
        }

    @Override
        public void run() {
                 Canvas c;
                 while (_run) {
                          c = null;
                          try {
```

```
                                c = surfaceHolder.lockCanvas(null);
                                synchronized (surfaceHolder) {
                                        mySurface.onDraw(c);
                                }
                        } finally {
                                if (c != null) {
                                surfaceHolder.unlockCanvasAndPost(c);
                                }
                        }
                }
        }
    }
}
```

The following figure shows drawing static text on canvas as well as dynamic data on canvas. The initial values for x and y parameters are set to 100, 200.

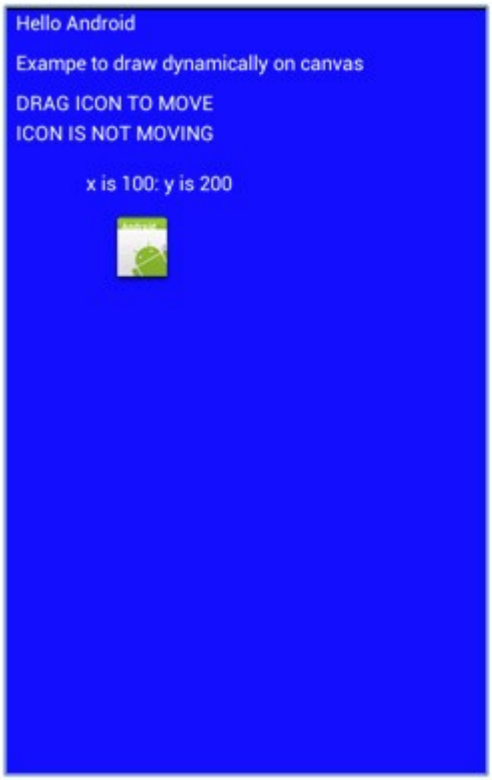

Figure 1: Drawing Graphics

Figure 2: Dynamic Drawing on Canvas

ref:http://developer.samsung.com/android/technical-docs/2D-Graphics-in-Android