Mobile 3D Graphics

Introduction to Android graphics

Android graphics

- **Android** provides a huge set of **2D-drawing APIs** that allow you to create **graphics**.
- **Android framework** provides a rich set of powerful **APIS** for applying **animation** to **UI elements** and **graphics** as well as drawing custom **2D** and **3D** graphics.

Animation systems

- **Three animation systems used in Android applications:**

  1. **View** Animation

  2. **Drawable** Animation

  3. **Property** Animation

View Animation

- **View Animation** is also called as **TweenAnimation.**
- The ***android.view.animation*** provides classes which handle **view animation**.
- This animation can be used to **animate** the **content of a view**.
- It is limited to simple transformation such as **moving**, **re-sizing** and **rotation**, but **not** its background color.

Drawable Animation

- **Drawable animation** is **implemented** using the ***AnimationDrawable*** class.
- This animation works by displaying a running sequence of **'Drawable'** **resources** that is **images**, frame by frame inside a **view** object.

Property Animation

- **Property animation** is the preferred method of animation in Android.
- Which lets you animate any **properties** of any objects, **view** or **non-view** objects.
- The ***android.animation*** provides classes which handle **property animation**.

2D Graphics
Canvas

- **Android graphics** provides **low level graphics tools** such as **canvases**, **color**, **filters**, **points** and **rectangles** which handle drawing to the screen directly.

Ways to draw 2D graphics

1. Draw your animation into a **View object** from your **layout**.
2. Draw your animation **directly** to a **Canvas**.

Some of the important methods of Canvas Class are as follows

i) **drawText()**

ii) **drawRoundRect()**

iii) **drawCircle()**

iv) **drawRect()**

v) **drawBitmap()**

vi) **drawARGB()**

- You can use these methods in **onDraw() method** to create your own custom user interface.

3D Graphics
OpenGL ES

- **"OpenGL ES" APIs** supported by the Android framework.
- powerful tools for manipulating and displaying **high-end animated 3D graphics** that can be benefited from the hardware acceleration of graphics processing units (**GPUs**) provided on many Android devices.

Example

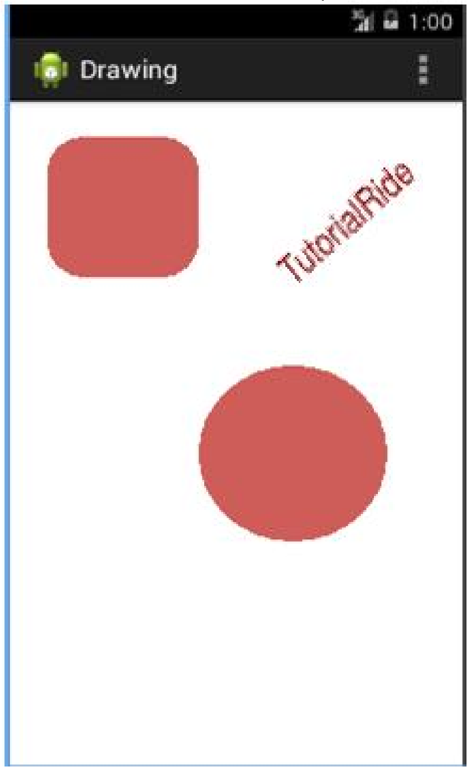
- Create a new Java class that should extend from **View class**.
- Override the **onDraw()** method. In this method, you can use **Canvas** class to draw the different shapes.

MyView.java

```
public class MyView extends View
{
      public MyView(Context context)
      {
          super(context);
          // TODO Auto-generated constructor stub
       }
       @Override
       protected void onDraw(Canvas canvas)
       {
            // TODO Auto-generated method stub
            super.onDraw(canvas);
            int radius;
            radius = 50;
            Paint paint = newPaint();
            paint.setStyle(Paint.Style.FILL);
            paint.setColor(Color.parseColor("#CD5C5C"));
            canvas.drawCircle(150,200, radius, paint);
            canvas.drawRoundRect(newRectF(20,20,100,100), 20, 20, paint);
            canvas.rotate(-45);
            canvas.drawText(“ITMC401", 40, 180, paint);
            canvas.restore();
        }
}
```

MainActivity.java

- **Note:** You have to pass the object of subclass that extends from View class in **setContentView()** method as given below. In our case the name of the subclass is **MyView**.



- Public class **MainActivity** extends **Activity**

```
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new MyView(this));
    }
    @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);
        return true;
    }
}
```

References