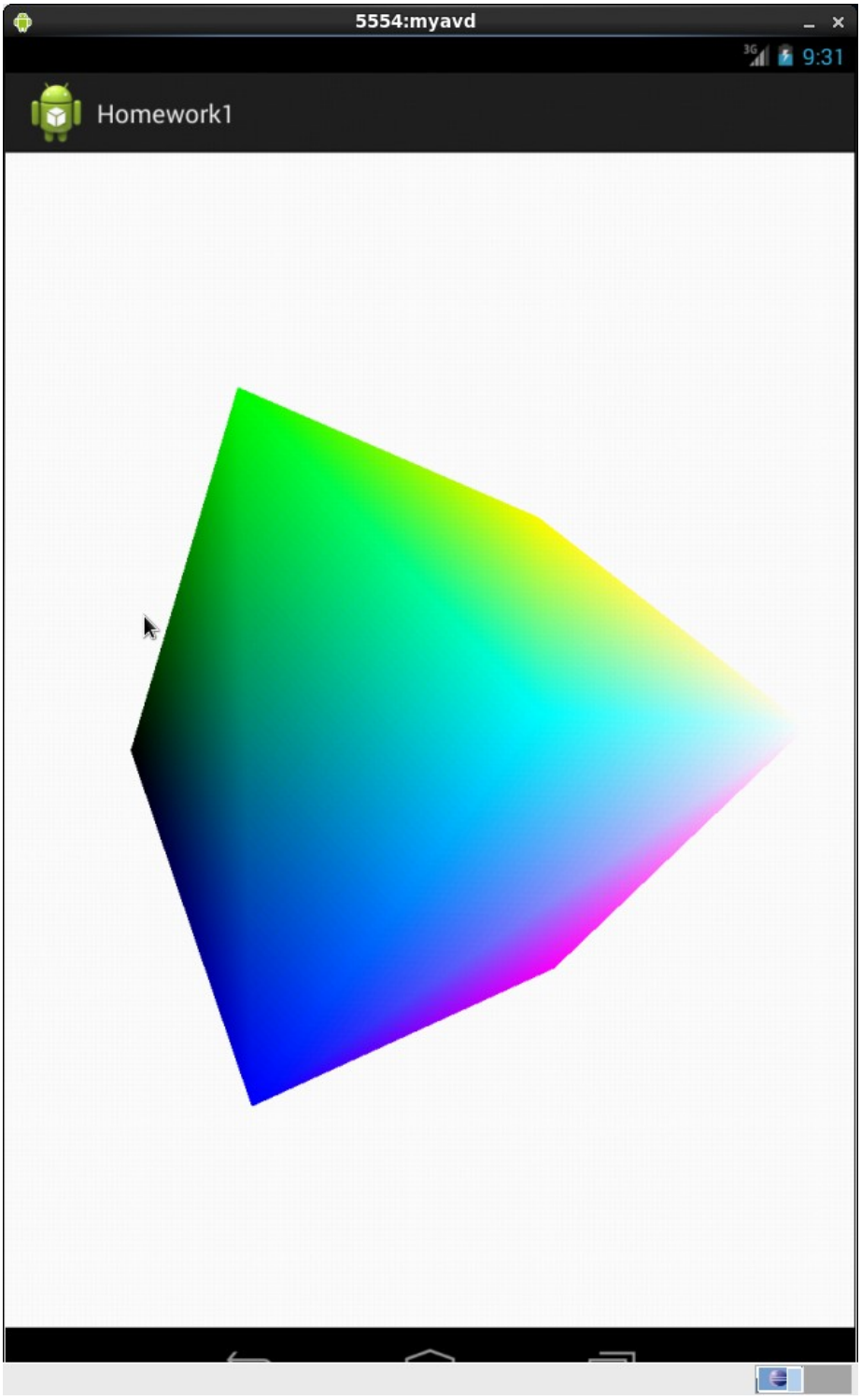
CSE520
Samuel Marrujo
Professor Yu
Homework 1

Draw 4 polygons using Eclipse

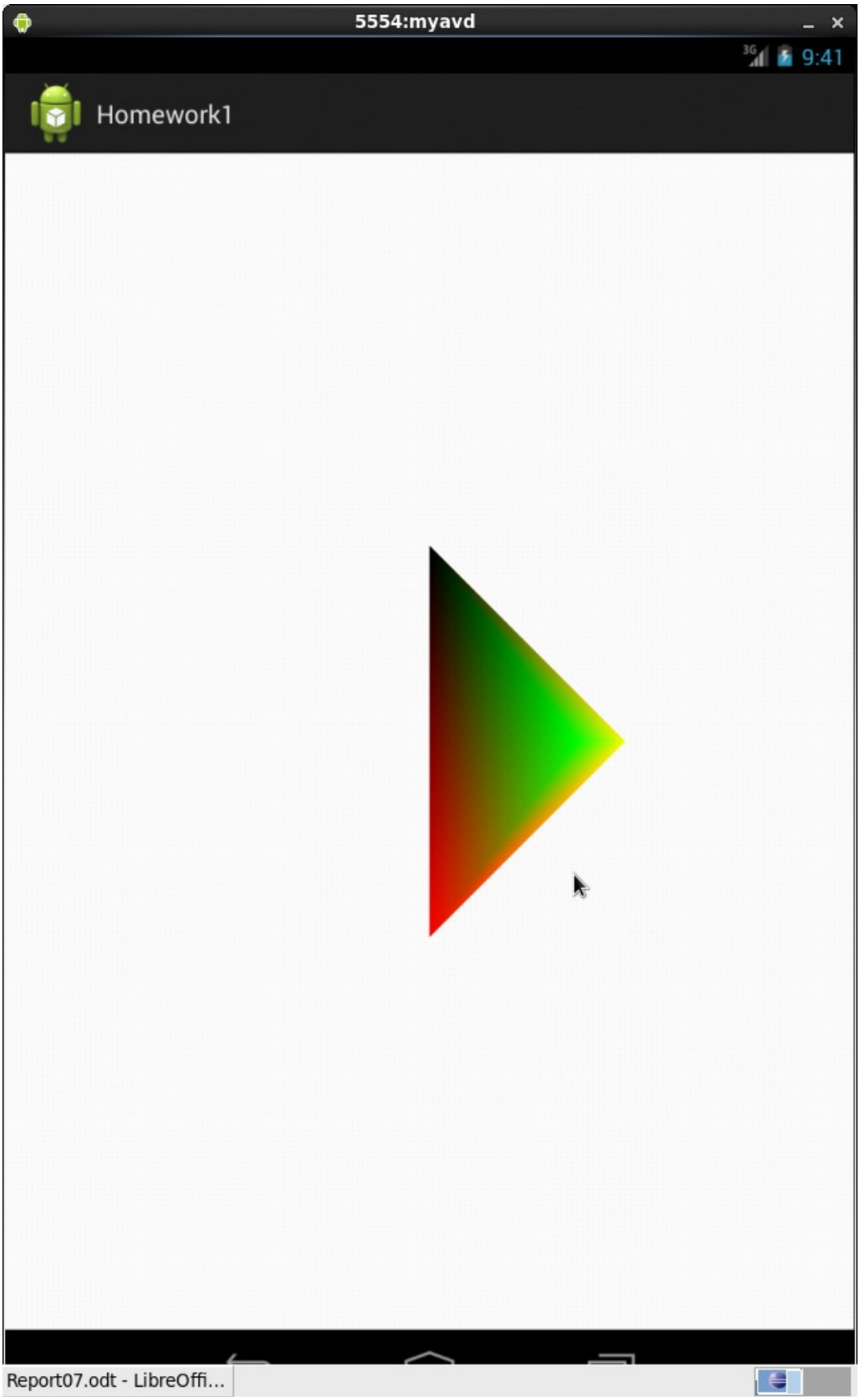
In this homework, we are to create a cube, tetrahedron, dodecahedron, and icosahedron using Eclipse to create an Android application. I have successfully completed this task, and here is a picture of the following:

Each picture will be on a separate page as shown below.

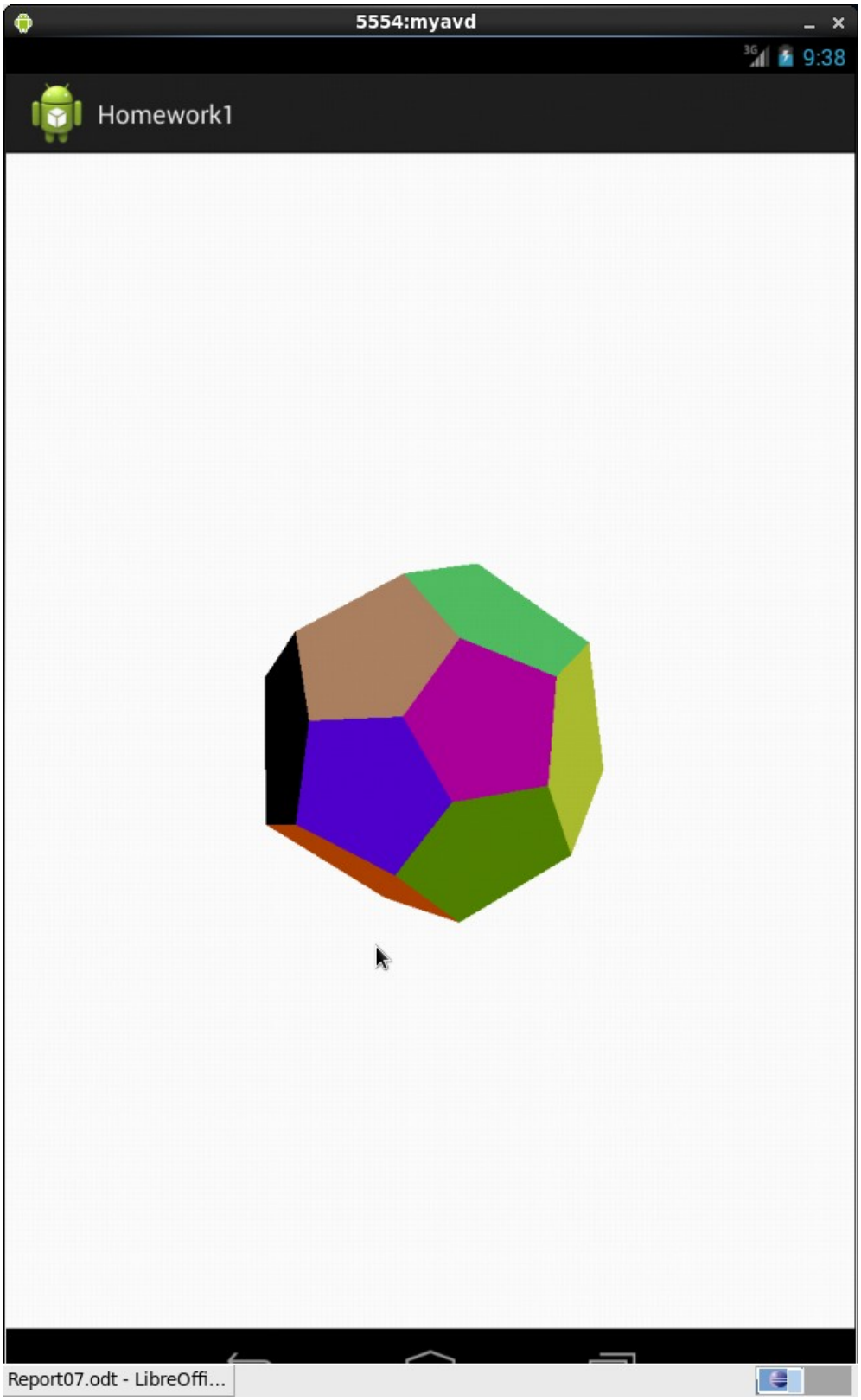
Cube:



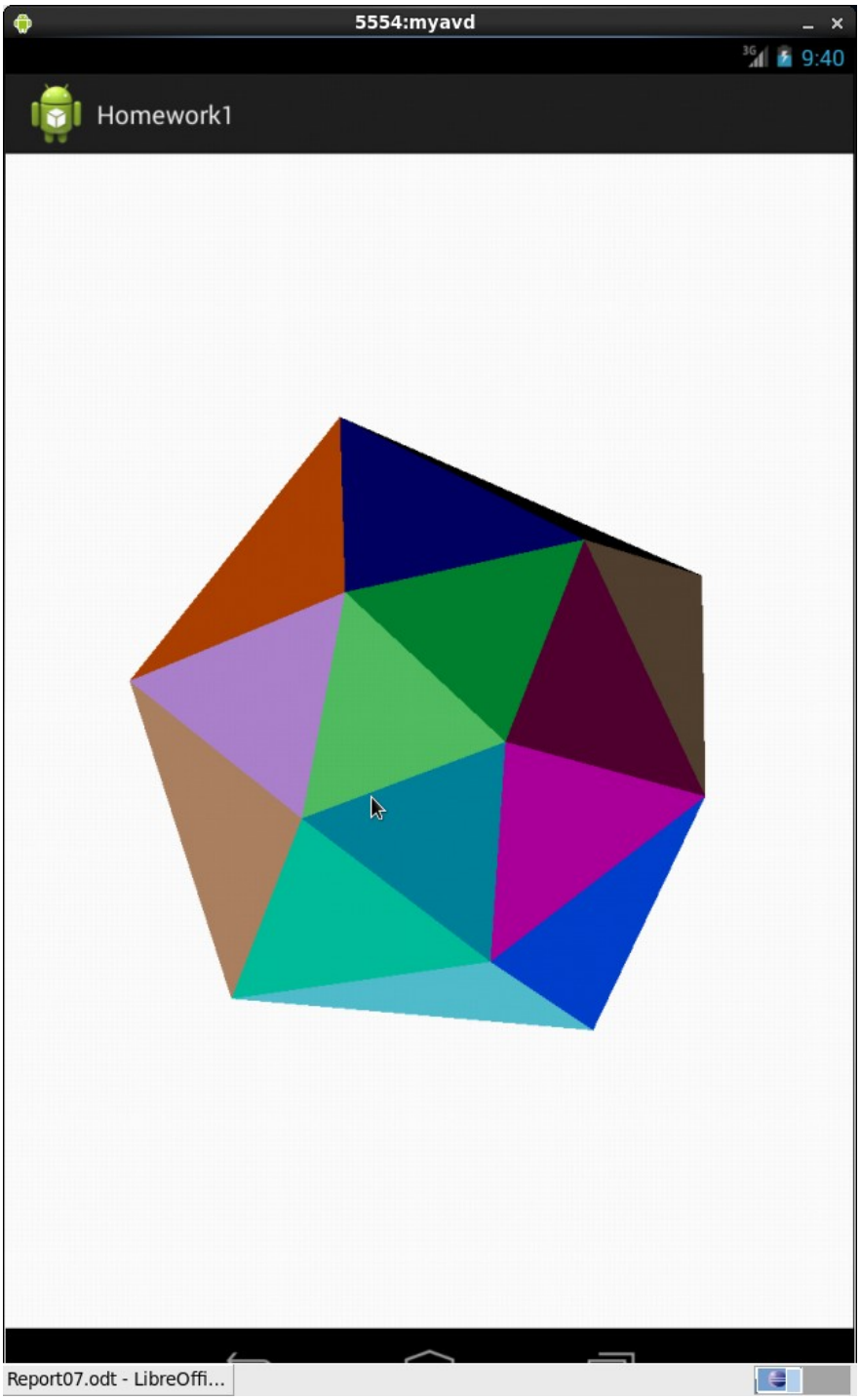
Tetrahedron:



Dodecahedron:



Icosahedron:



Code:
For each render, I change the “setRenderer(mSOMERenderer);” code to each respective render. This way, it's a simple change into each render I want to show. In the future I want to implement this with a button, but it became incredibly difficult to implement.

```
package opengl.homework1;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.view.View.OnClickListener;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Button;
import android.graphics.Bitmap;
//import android.graphics.BitmapFactory;
import opengl.homework1.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.opengl.GLSurfaceView;
import android.opengl.GLSurfaceView.Renderer;
import android.view.MotionEvent;
import android.content.Context;

public class MainActivity extends Activity {
  Button imageButton;
  Button imageButton2;
  Button imageButton3;
  Button imageButton4;
  Bitmap bitmap;
  Bitmap bitmap2;
  ImageView image;
  TextView message;
  private GLSurfaceView mGLSurfaceView;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mGLSurfaceView = new TouchSurfaceView(this);
    setContentView(mGLSurfaceView);
    mGLSurfaceView.requestFocus();
    mGLSurfaceView.setFocusableInTouchMode(true);

    //bitmap = BitmapFactory.decodeResource(this.getResources(),
R.drawable.picture);
    //bitmap2 = BitmapFactory.decodeResource(this.getResources(),
R.drawable.picture2);

    //addListenerOnButton();
  }

  class TouchSurfaceView extends GLSurfaceView {

          public TouchSurfaceView(Context context) {
              super(context);
              mRenderer = new CubeRenderer();
              mDodecahedronRenderer = new DodecahedronRenderer();
              mTetrahedronRenderer = new TetrahedronRenderer();
```

```
            mIcosahedronRenderer = new IcosahedronRenderer();

            setRenderer(mTetrahedronRenderer);
            //RENDERMODE_WHEN_DIRTY means "do not call onDrawFrame() unless
            // something explicitly requests rendering with requestRender()
            setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
        }

        @Override public boolean onTrackballEvent(MotionEvent e) {
            mRenderer.mAngleX += e.getX() * TRACKBALL_SCALE_FACTOR;
            mRenderer.mAngleY += e.getY() * TRACKBALL_SCALE_FACTOR;
            /*
            mDodecahedronRenderer.anglex += e.getX() * TRACKBALL_SCALE_FACTOR;
            mDodecahedronRenderer.anglez += e.getY() * TRACKBALL_SCALE_FACTOR;
            mTetrahedronRenderer.mAngleX += e.getX() * TRACKBALL_SCALE_FACTOR;
            mTetrahedronRenderer.mAngleY += e.getY() * TRACKBALL_SCALE_FACTOR;
            mIcosahedronRenderer.anglex += e.getX() * TRACKBALL_SCALE_FACTOR;
            mIcosahedronRenderer.anglez += e.getY() * TRACKBALL_SCALE_FACTOR;
            */
            requestRender();
            return true;
        }

        @Override public boolean onTouchEvent(MotionEvent e) {
            float x = e.getX();
            float y = e.getY();
            switch (e.getAction()) {
            case MotionEvent.ACTION_MOVE:
                float dx = x - mPreviousX;
                float dy = y - mPreviousY;
                mRenderer.mAngleX += dx * TOUCH_SCALE_FACTOR;
                mRenderer.mAngleY += dy * TOUCH_SCALE_FACTOR;
                /*
                mDodecahedronRenderer.anglex += dx * TOUCH_SCALE_FACTOR;
                mDodecahedronRenderer.anglez += dy * TOUCH_SCALE_FACTOR;
                mTetrahedronRenderer.mAngleX += dx * TOUCH_SCALE_FACTOR;
                mTetrahedronRenderer.mAngleY += dy * TOUCH_SCALE_FACTOR;
                mIcosahedronRenderer.anglex += dx * TOUCH_SCALE_FACTOR;
                mIcosahedronRenderer.anglez += dy * TOUCH_SCALE_FACTOR;
                */
                requestRender();
            }
            mPreviousX = x;
            mPreviousY = y;
            return true;
        }

    private class CubeRenderer implements Renderer {
        public CubeRenderer() {
            mCube = new Cube();
        }

        public void onDrawFrame(GL10 gl) {


            gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

            gl.glMatrixMode(GL10.GL_MODELVIEW);
```

```
            gl.glLoadIdentity();
            gl.glTranslatef(0, 0, -3.0f);
            gl.glRotatef(mAngleX, 0, 1, 0);
            gl.glRotatef(mAngleY, 1, 0, 0);

            gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
            gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
            mCube.draw(gl);

        }

        public void onSurfaceChanged(GL10 gl, int width, int height) {
             gl.glViewport(0, 0, width, height);

             float ratio = (float) width / height;
             gl.glMatrixMode(GL10.GL_PROJECTION);
             gl.glLoadIdentity();
             gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);
        }

        public void onSurfaceCreated(GL10 gl, EGLConfig config) {

            gl.glDisable(GL10.GL_DITHER);

             gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
                     GL10.GL_FASTEST);


             gl.glClearColor(1,1,1,1);
             gl.glEnable(GL10.GL_CULL_FACE);
             gl.glShadeModel(GL10.GL_SMOOTH);
             gl.glEnable(GL10.GL_DEPTH_TEST);
        }
        private Cube mCube;
        public float mAngleX;
        public float mAngleY;
    }

private class TetrahedronRenderer implements Renderer {
        public TetrahedronRenderer() {
            mTetrahedron = new Tetrahedron();
        }

        public void onDrawFrame(GL10 gl) {


            gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);



            gl.glMatrixMode(GL10.GL_MODELVIEW);
            gl.glLoadIdentity();
            gl.glTranslatef(0, 0, -3.0f);
            gl.glRotatef(mAngleX, 0, 1, 0);
            gl.glRotatef(mAngleY, 1, 0, 0);

            gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
            gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
```

```
            mTetrahedron.draw(gl);

        }

        public void onSurfaceChanged(GL10 gl, int width, int height) {
             gl.glViewport(0, 0, width, height);

             float ratio = (float) width / height;
             gl.glMatrixMode(GL10.GL_PROJECTION);
             gl.glLoadIdentity();
             gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);
        }

        public void onSurfaceCreated(GL10 gl, EGLConfig config) {

            gl.glDisable(GL10.GL_DITHER);

             gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
                     GL10.GL_FASTEST);


             gl.glClearColor(1,1,1,1);
             gl.glEnable(GL10.GL_CULL_FACE);
             gl.glShadeModel(GL10.GL_SMOOTH);
             gl.glEnable(GL10.GL_DEPTH_TEST);
        }
        private Tetrahedron mTetrahedron;
        public float mAngleX;
        public float mAngleY;
    }
public class DodecahedronRenderer implements Renderer
{
    GL10 gl;
    Dodecahedron dodecahedron = new Dodecahedron();
    private float anglex;
    private float anglez;
    private final int nfaces = 12;
    //@Override
    //Refresh automatically
    public void onDrawFrame(GL10 gl)
    {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
        gl.glTranslatef(0.0f, 0.0f, -3.0f);
        gl.glRotatef( anglex, 1.0f, 0.0f, 0.0f ); // Rotate about  x-axis
        gl.glRotatef( anglez, 0.0f, 0.0f, 1.0f); // Rotate about z-axis
        dodecahedron.draw(gl);
        anglex += 1.0f;
        anglez += 2.0f;
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
    }

    public void onSurfaceChanged(GL10 gl, int width, int height)
```

```
  {
      gl.glViewport(0, 0, width, height);
      float ratio = (float) width / height;
      gl.glMatrixMode(GL10.GL_PROJECTION);
      gl.glLoadIdentity();
      gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);
  }

  public void onSurfaceCreated(GL10 gl, EGLConfig config)
  {
      gl.glDisable(GL10.GL_DITHER);
      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
      gl.glClearColor(1, 1, 1, 0);
      gl.glEnable(GL10.GL_CULL_FACE);
      gl.glShadeModel(GL10.GL_SMOOTH);
      gl.glEnable(GL10.GL_DEPTH_TEST);
  }
}
public class IcosahedronRenderer implements Renderer
{
  GL10 gl;
  Icosahedron icosahedron = new Icosahedron();
  private float anglex;
  private float anglez;
  private final int nfaces = 20;
  //@Override
  //Refresh automatically
  public void onDrawFrame(GL10 gl)
  {
      gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
      gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
      gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
      gl.glMatrixMode(GL10.GL_MODELVIEW);
      gl.glLoadIdentity();
      gl.glTranslatef(0.0f, 0.0f, -2.0f);
      gl.glRotatef( anglex, 1.0f, 0.0f, 0.0f ); // Rotate about  x-axis
      gl.glRotatef( anglez, 0.0f, 0.0f, 1.0f); // Rotate about z-axis
      icosahedron.draw(gl);
      anglex += 1.0f;
      anglez += 2.0f;
      gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
      gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
  }

  public void onSurfaceChanged(GL10 gl, int width, int height)
  {
      gl.glViewport(0, 0, width, height);
      float ratio = (float) width / height;
      gl.glMatrixMode(GL10.GL_PROJECTION);
      gl.glLoadIdentity();
      gl.glFrustumf(-ratio, ratio, -1, 1, 1, 10);
  }

  public void onSurfaceCreated(GL10 gl, EGLConfig config)
  {
      gl.glDisable(GL10.GL_DITHER);
      gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
```

```
        gl.glClearColor(1, 1, 1, 0);
        gl.glEnable(GL10.GL_CULL_FACE);
        gl.glShadeModel(GL10.GL_SMOOTH);
        gl.glEnable(GL10.GL_DEPTH_TEST);
    }
}

    private final float TOUCH_SCALE_FACTOR = 180.0f / 320;
    private final float TRACKBALL_SCALE_FACTOR = 36.0f;
    private CubeRenderer mRenderer;

    private DodecahedronRenderer mDodecahedronRenderer;
    private TetrahedronRenderer mTetrahedronRenderer;
    private IcosahedronRenderer mIcosahedronRenderer;

    private float mPreviousX;
    private float mPreviousY;



}


    //define Listeners
    public void addListenerOnButton() {
      imageButton = (Button) findViewById(R.id.imageButton1);
      imageButton2 = (Button) findViewById(R.id.imageButton2);
      imageButton3 = (Button) findViewById(R.id.imageButton3);
      imageButton4 = (Button) findViewById(R.id.imageButton4);

      imageButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
         message = (TextView) findViewById(R.id.message);
         image = (ImageView) findViewById(R.id.carimage);
         message.setText("Cube");
         //image.setImageBitmap(bitmap);
        }
      });

      imageButton2.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
          message = (TextView) findViewById(R.id.message);
          image = (ImageView) findViewById(R.id.carimage);
          message.setText("Tetrahedron");
          //image.setImageBitmap(bitmap2);
        }
      });
      imageButton3.setOnClickListener(new OnClickListener() {
          @Override
          public void onClick(View view) {
           message = (TextView) findViewById(R.id.message);
           image = (ImageView) findViewById(R.id.carimage);
           message.setText("Dodecahedron");
           //image.setImageBitmap(bitmap);
          }
        });
```

```
    imageButton4.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
         message = (TextView) findViewById(R.id.message);
         image = (ImageView) findViewById(R.id.carimage);
         message.setText("Icosahedron");
         //image.setImageBitmap(bitmap);
        }
      });
  }
  @Override
  protected void onPause() {
      super.onPause();
      // The following call pauses the rendering thread.
      mGLSurfaceView.onPause();
  }

  @Override
  protected void onResume() {
      super.onResume();
      // The following call resumes a paused rendering thread.
      mGLSurfaceView.onResume();
  }
}
```