Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Лабораторная работа по теме: «Примитивы OpenGL ES 1»

Выполнили:

студентки 4 курса

ИВТ, гр. ИП-712

Гервас А.В.

Онищенко А.В.

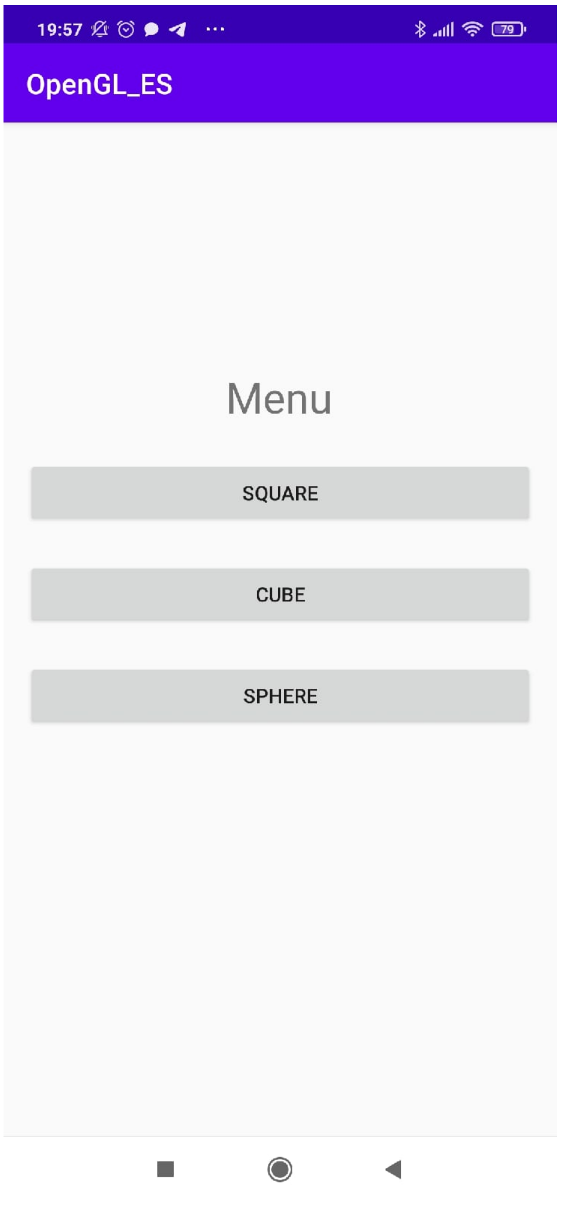
Новосибирск 2020

Оглавление

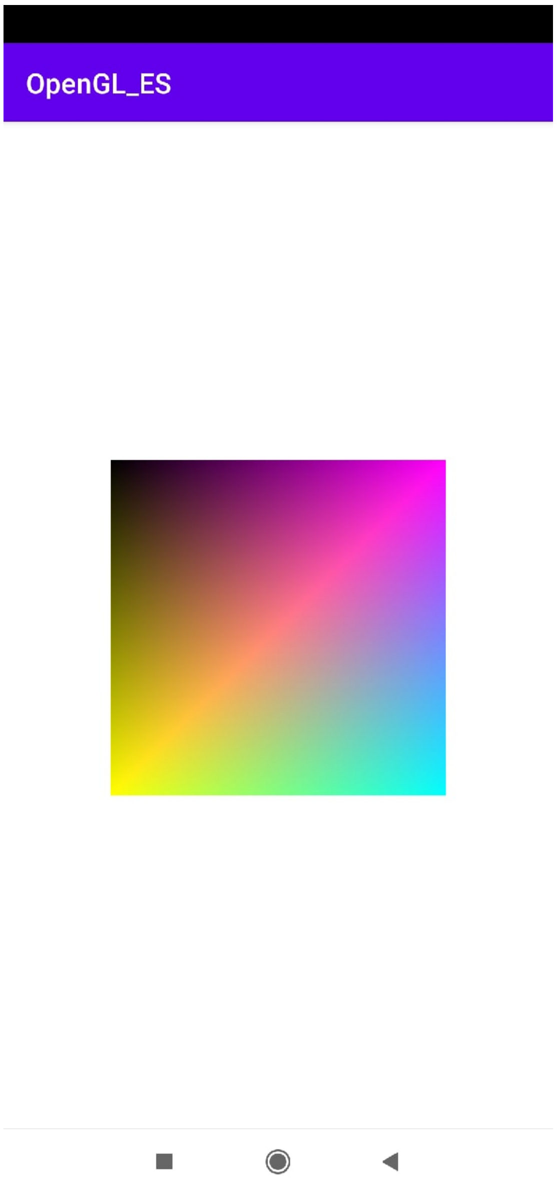
Задание

Необходимо создать классы прорисовки квадрата, куба, сферы.

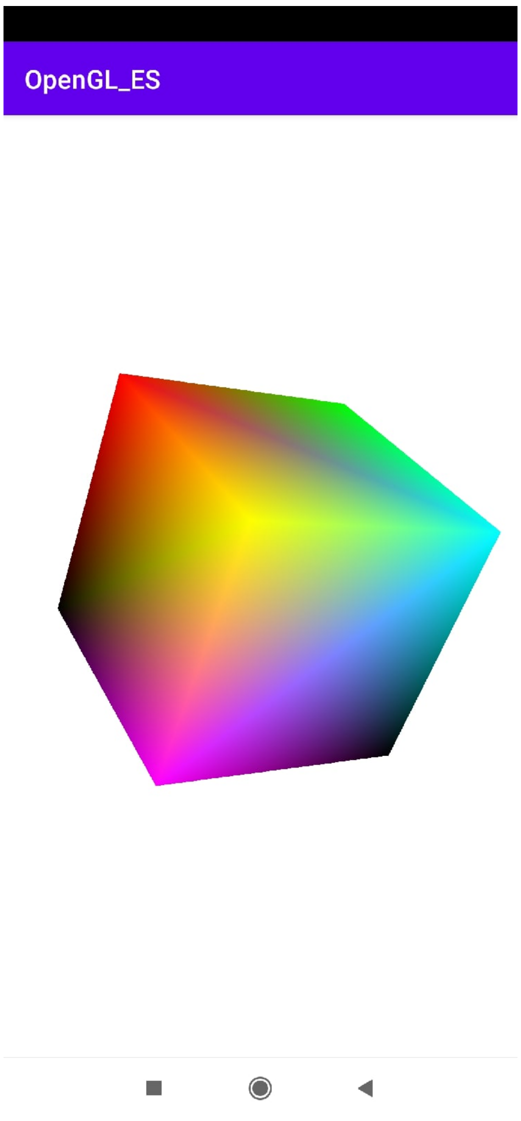
Скриншоты

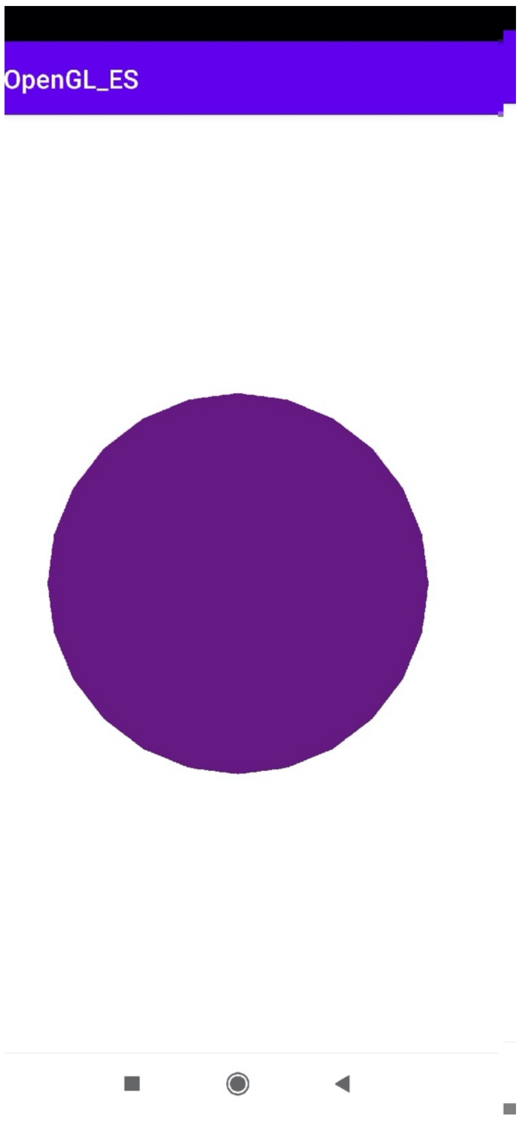


Стартовая активность



Прорисовка квадрата





Прорисовка куба	Прорисовка сферы

Листинг кода

Приложение написано на языке Java.

MainActivity.java

```
package ru.sibsutis.opengl_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import ru.sibsutis.opengl_es.R;

public class MainActivity extends AppCompatActivity {

   Button squareButton;
   Button cubeButton;
   Button sphereButton;

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

      squareButton = (Button) findViewById(R.id.squareButton);
      cubeButton = (Button) findViewById(R.id.cubeButton);
      sphereButton = (Button) findViewById(R.id.sphereButton);

      squareButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this, SquareActivity.class);
            startActivity(intent);
         }
      });

      cubeButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this, CubeActivity.class);
            startActivity(intent);
         }
      });

      sphereButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this, SphereActivity.class);
            startActivity(intent);
         }
      });
   }
}
```

SquareActivity.java

```
package ru.sibsutis.opengl_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.opengl.GLSurfaceView;
import android.os.Bundle;
import android.view.WindowManager;

import ru.sibsutis.opengl_es.entity.Square;

public class SquareActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);


getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
      GLSurfaceView view = new GLSurfaceView(this);
      view.setRenderer(new Square());
      setContentView(view);
   }
}
```

Square.java

```
package ru.sibsutis.opengl_es.entity;

import android.opengl.GLSurfaceView;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import javax.microedition.khronos.opengles.GL11;

/**
 * A vertex shaded square.
 */
public class Square implements GLSurfaceView.Renderer {

   private FloatBuffer mFVertexBuffer;
   private ByteBuffer mColorBuffer;
   private ByteBuffer mIndexBuffer;

   private boolean mTranslucentBackground;
   private float mTransY;

   public Square() {
      float[] vertices = {
            -1.0f, -1.0f,
            1.0f, -1.0f,
            -1.0f, 1.0f,
            1.0f, 1.0f
      };

      byte maxColor = (byte) 255;

      byte[] colors = {
            maxColor, maxColor, 0, maxColor,
            0, maxColor, maxColor, maxColor,
            0, 0, 0, maxColor,
            maxColor, 0, maxColor, maxColor
      };

      byte[] indices = {
            0, 3, 1,
            0, 2, 3
      };

      ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
      vbb.order(ByteOrder.nativeOrder());
      mFVertexBuffer = vbb.asFloatBuffer();
      mFVertexBuffer.put(vertices);
      mFVertexBuffer.position(0);

      mColorBuffer = ByteBuffer.allocateDirect(colors.length);
      mColorBuffer.put(colors);
      mColorBuffer.position(0);

      mIndexBuffer = ByteBuffer.allocateDirect(indices.length);
      mIndexBuffer.put(indices);
      mIndexBuffer.position(0);
```

```
    }

    public void draw(GL10 gl) {
        gl.glFrontFace(GL11.GL_CW);
        gl.glVertexPointer(2, GL11.GL_FLOAT, 0, mFVertexBuffer);
        gl.glColorPointer(4, GL11.GL_UNSIGNED_BYTE, 0, mColorBuffer);
        gl.glDrawElements(GL11.GL_TRIANGLES, 6,
GL11.GL_UNSIGNED_BYTE, mIndexBuffer);
        gl.glFrontFace(GL11.GL_CCW);
    }

    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glDisable(GL10.GL_DITHER);
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
GL10.GL_FASTEST);
        if (mTranslucentBackground) {
            gl.glClearColor(0, 0, 0, 0);
        } else {
            gl.glClearColor(1, 1, 1, 1);
            gl.glEnable(GL10.GL_CULL_FACE);
            gl.glShadeModel(GL10.GL_SMOOTH);
            gl.glEnable(GL10.GL_DEPTH_TEST);
        }
    }

    @Override
    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);
    }

    @Override
    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(0.0f, (float) Math.sin(mTransY), -3.0f);

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

        draw(gl);
        mTransY += .075f;
    }
}
```

CubeActivity.java

```
package ru.sibsutis.opengl_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.opengl.GLSurfaceView;
import android.os.Bundle;
import android.view.WindowManager;

import ru.sibsutis.opengl_es.R;
import ru.sibsutis.opengl_es.entity.Cube;
import ru.sibsutis.opengl_es.entity.Square;

public class CubeActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);


getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        GLSurfaceView view = new GLSurfaceView(this);
        view.setRenderer(new Cube());
        setContentView(view);
    }
}
```

Cube.java

```
package ru.sibsutis.opengl_es.entity;

import android.opengl.GLSurfaceView;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import javax.microedition.khronos.opengles.GL11;

public class Cube implements GLSurfaceView.Renderer {

   private FloatBuffer mFVertexBuffer;
   private ByteBuffer mColorBuffer;
   private ByteBuffer mTfan1;
   private ByteBuffer mTfan2;

   private boolean mTranslucentBackground;
   private Cube mCube;
   private float mTransY;
   private float mAngle;

   public Cube() {

      float[] vertices = {
            -1.0f, 1.0f, 1.0f,
            1.0f, 1.0f, 1.0f,
            1.0f, -1.0f, 1.0f,
            -1.0f, -1.0f, 1.0f,

            -1.0f, 1.0f, -1.0f,
            1.0f, 1.0f, -1.0f,
            1.0f, -1.0f, -1.0f,
            -1.0f, -1.0f, -1.0f
      };

      byte maxColor = (byte) 255;

      byte[] colors = {
            maxColor, maxColor, 0, maxColor,
            0, maxColor, maxColor, maxColor,
            0, 0, 0, maxColor,
            maxColor, 0, maxColor, maxColor,

            maxColor, 0, 0, maxColor,
            0, maxColor, 0, maxColor,
            0, 0, maxColor, maxColor,
            0, 0, 0, maxColor
```

```
    };

    byte[] tfan1 = {
            1, 0, 3,
            1, 3, 2,
            1, 2, 6,
            1, 6, 5,
            1, 5, 4,
            1, 4, 0
    };

    byte[] tfan2 = {
            7, 4, 5,
            7, 5, 6,
            7, 6, 2,
            7, 2, 3,
            7, 3, 0,
            7, 0, 4
    };

    ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
    vbb.order(ByteOrder.nativeOrder());
    mFVertexBuffer = vbb.asFloatBuffer();
    mFVertexBuffer.put(vertices);
    mFVertexBuffer.position(0);

    mColorBuffer = ByteBuffer.allocateDirect(colors.length);
    mColorBuffer.put(colors);
    mColorBuffer.position(0);

    mTfan1 = ByteBuffer.allocateDirect(tfan1.length);
    mTfan1.put(tfan1);
    mTfan1.position(0);

    mTfan2 = ByteBuffer.allocateDirect(tfan2.length);
    mTfan2.put(tfan2);
    mTfan2.position(0);

}

public void draw(GL10 gl) {
    gl.glVertexPointer(3, GL11.GL_FLOAT, 0, mFVertexBuffer);
    gl.glColorPointer(4, GL11.GL_UNSIGNED_BYTE, 0, mColorBuffer);
```

```
      gl.glDrawElements(gl.GL_TRIANGLE_FAN, 6 * 3,
gl.GL_UNSIGNED_BYTE, mTfan1);
      gl.glDrawElements(gl.GL_TRIANGLE_FAN, 6 * 3,
gl.GL_UNSIGNED_BYTE, mTfan2);
   }

   @Override
   public void onSurfaceCreated(GL10 gl, EGLConfig config) {
      gl.glDisable(GL11.GL_DITHER);

      gl.glHint(GL11.GL_PERSPECTIVE_CORRECTION_HINT,
GL11.GL_FASTEST);

      if (mTranslucentBackground) {
         gl.glClearColor(1, 0, 0, 0);
      } else {
         gl.glClearColor(1, 1, 1, 1);
      }

      gl.glEnable(GL11.GL_CULL_FACE);
      gl.glShadeModel(GL11.GL_SMOOTH);
      gl.glEnable(GL11.GL_DEPTH_TEST);
   }

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

      float aspectRatio;
      float zNear = .1f;
      float zFar = 1000;
      float fieldOfView = 30.0f / 57.3f;
      float size;

      gl.glEnable(GL10.GL_NORMALIZE);

      aspectRatio = (float) width / (float) height;

      gl.glMatrixMode(GL10.GL_PROJECTION);

      size = zNear * (float) (Math.tan((double) (fieldOfView / 2.0f)));

      gl.glFrustumf(-size, size, -size / aspectRatio,
            size / aspectRatio, zNear, zFar);
```

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

    @Override
    public void onDrawFrame(GL10 gl) {
        gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
        gl.glClear(GL11.GL_COLOR_BUFFER_BIT |
GL11.GL_DEPTH_BUFFER_BIT);

        gl.glMatrixMode(GL11.GL_MODELVIEW);
        gl.glLoadIdentity();
        gl.glTranslatef(0.0f, (float) Math.sin(mTransY), -7.0f);

        gl.glRotatef(mAngle, 0.0f, 1.0f, 0.0f);
        gl.glRotatef(mAngle, 1.0f, 0.0f, 0.0f);

        gl.glEnableClientState(GL11.GL_VERTEX_ARRAY);
        gl.glEnableClientState(GL11.GL_COLOR_ARRAY);

        draw(gl);

        mTransY += .075f;
        mAngle += .4;
    }
}
```

s

SphereActivity.java

```
package ru.sibsutis.opengl_es.activity;

import androidx.appcompat.app.AppCompatActivity;

import android.opengl.GLSurfaceView;
import android.os.Bundle;
import android.view.WindowManager;

import ru.sibsutis.opengl_es.entity.Sphere;

public class SphereActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        GLSurfaceView view = new GLSurfaceView(this);
        view.setRenderer(new Sphere(0.5f));
        setContentView(view);
    }
}
```

Sphere.java

```
package ru.sibsutis.opengl_es.entity;

import android.opengl.GLSurfaceView;
import android.opengl.GLU;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.FloatBuffer;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

public class Sphere implements GLSurfaceView.Renderer {

    public FloatBuffer mVertexBuffer;
    private ByteBuffer mColorBuffer;
    private boolean mTranslucentBackground;

    public int n = 0, sz = 0;

    public Sphere(float R) {

        float i = 0;
        int dtheta = 15, dphi = 15;

        int theta, phi;
        float DTOR = (float) (Math.PI / 180.0f);

        ByteBuffer byteBuf = ByteBuffer.allocateDirect(5000 * 3 * 4);
        byteBuf.order(ByteOrder.nativeOrder());
        mVertexBuffer = byteBuf.asFloatBuffer();
        byteBuf = ByteBuffer.allocateDirect(5000 * 2 * 4);
        byteBuf.order(ByteOrder.nativeOrder());
        for (theta = -90; theta <= 90 - dtheta; theta += dtheta) {
            for (phi = 0; phi <= 360 - dphi; phi += dphi) {
                sz++;
                mVertexBuffer.put((float) (Math.cos(theta * DTOR) * Math.cos(phi *
DTOR)) * R);
                mVertexBuffer.put((float) (Math.cos(theta * DTOR) * Math.sin(phi *
DTOR)) * R);
                mVertexBuffer.put((float) (Math.sin(theta * DTOR)) * R);

                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) *
Math.cos(phi * DTOR)) * R);
                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) *
Math.sin(phi * DTOR)) * R);
                mVertexBuffer.put((float) (Math.sin((theta + dtheta) * DTOR)) * R);

                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) *
Math.cos((phi + dphi) * DTOR)) * R);
                mVertexBuffer.put((float) (Math.cos((theta + dtheta) * DTOR) *
Math.sin((phi + dphi) * DTOR)) * R);
                mVertexBuffer.put((float) (Math.sin((theta + dtheta) * DTOR)) * R);
                n += 3;

                mVertexBuffer.put((float) (Math.cos(theta * DTOR) * Math.cos((phi +
dphi) * DTOR)) * R);
                mVertexBuffer.put((float) (Math.cos(theta * DTOR) * Math.sin((phi +
dphi) * DTOR)) * R);
                mVertexBuffer.put((float) (Math.sin(theta * DTOR)) * R);
                n++;
            }
        }

        mVertexBuffer.position(0);

    }

    public void draw(GL10 gl) {

        gl.glFrontFace(GL10.GL_CCW); // Front face in counter-clockwise
orientation
        gl.glEnable(GL10.GL_CULL_FACE); // Enable cull face
        gl.glCullFace(GL10.GL_BACK); // Cull the back face (don't display)
        gl.glEnable(GL10.GL_BLEND);
        gl.glBlendFunc(GL10.GL_SRC_ALPHA,
GL10.GL_ONE_MINUS_SRC_ALPHA);
```

```
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mVertexBuffer);


        int i = 0;
        for (i = 0; i < n; i += 4) {
            gl.glColor4f(0.4f, 0.1f, 0.5f, 1.0f);
            gl.glDrawArrays(GL10.GL_TRIANGLE_FAN, i, 4);
        }

        gl.glColor4f(0.4f, 0.1f, 0.5f, 1.0f);
    }

    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glDisable(GL10.GL_DITHER);
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
GL10.GL_FASTEST);
        if (mTranslucentBackground) {
            gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
        } else {
            gl.glClearColor(1, 1, 1, 1);
            gl.glEnable(GL10.GL_CULL_FACE);
            gl.glShadeModel(GL10.GL_SMOOTH);
            gl.glEnable(GL10.GL_DEPTH_TEST);
        }

        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
    }

    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        float ratio = (float) width / height;
        GLU.gluPerspective(gl, 45.0f, ratio, 1f, 100f);
    }

    @Override
    public void onDrawFrame(GL10 gl) {
        gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT |
GL10.GL_DEPTH_BUFFER_BIT);
```

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

        gl.glTranslatef(0f, 0f, -3.0f);
        draw(gl);
    }
}
```