Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Кафедра ПМиК

Программирование мобильных устройств

Лабораторная работа №3

Матрицы модели-вида OpenGL ES 1

Выполнил: студент 4 курса

Ф. ИВТ, группа: ИП-711

Мартасов И. О.

Проверил: доцент кафедры ПМиК

Нечта Иван Васильевич

Новосибирск, 2020

**Содержание**

1. **Постановка задачи..………………………………………………………3**
2. **Исходный код……..……………………………………………………….3**
3. **Результаты работы…..………………………………………………….10**

**Постановка задачи**

Необходимо создать модель Солнце и вращающиеся Земля и Луна. Текстуры взять из интеренета.

**Исходный код**

**MainActivity.java – основной класс**

public class MainActivity extends AppCompatActivity

{

@RequiresApi(api = Build.VERSION\_CODES.Q)

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

GLSurfaceView gl\_surface\_view = new GLSurfaceView(this);

gl\_surface\_view.setRenderer(new MatrixRenderer(this));

gl\_surface\_view.setRenderMode(GLSurfaceView.RENDERMODE\_CONTINUOUSLY);

setContentView(gl\_surface\_view);

}

}

**MatrixRenderer.java – создание и отрисовка матриц**

@RequiresApi(api = Build.VERSION\_CODES.N)

class MatrixRenderer implements GLSurfaceView.Renderer

{

int counter = 0;

Activity new\_activity;

ArrayList<Object3D> matrices = new ArrayList<>();

@RequiresApi(api = Build.VERSION\_CODES.Q)

public MatrixRenderer(Activity activity)

{

new\_activity = activity;

}

@RequiresApi(api = Build.VERSION\_CODES.Q)

@Override

public void onSurfaceCreated(GL10 gl10, EGLConfig eglConfig)

{

matrices.addAll(Arrays.asList(

new QuadrilateralTextured(

gl10,

new float[] {

-1.5f, -1.5f, 0f,

-1.5f, 1.5f, 0f,

1.5f, -1.5f, 0f,

1.5f, 1.5f, 0f,

},

(Bitmap) BitmapFactory.decodeStream(new\_activity.getResources().openRawResource(R.raw.sun))

),

new QuadrilateralTextured(

gl10,

new float[] {

-1.5f, -1.5f, 0f,

-1.5f, 1.5f, 0f,

1.5f, -1.5f, 0f,

1.5f, 1.5f, 0f,

},

(Bitmap) BitmapFactory.decodeStream(new\_activity.getResources().openRawResource(R.raw.earth))

),

new QuadrilateralTextured(

gl10,

new float[]{

-1.5f, -1.5f, 0f,

-1.5f, 1.5f, 0f,

1.5f, -1.5f, 0f,

1.5f, 1.5f, 0f,

},

(Bitmap) BitmapFactory.decodeStream(new\_activity.getResources().openRawResource(R.raw.moon))

)

));

}

@Override

public void onSurfaceChanged(GL10 gl10, int i, int i1)

{

}

@Override

public void onDrawFrame(GL10 gl10)

{

counter = ++counter % 360;

float dx\_earth = (float) Math.cos(Math.toRadians((float) counter)) \* 4;

float dy\_earth = (float) Math.sin(Math.toRadians((float) counter)) \* 4;

float dx\_moon = (float) Math.cos(Math.toRadians((float) counter \* 2)) \* 2 + dx\_earth;

float dy\_moon = (float) Math.sin(Math.toRadians((float) counter \* 2)) \* 2 + dy\_earth;

gl10.glClear(GL10.GL\_COLOR\_BUFFER\_BIT | GL10.GL\_DEPTH\_BUFFER\_BIT);

gl10.glLoadIdentity();

gl10.glScalef(0.25f, 0.25f, 0.25f);

gl10.glEnable(GL10.GL\_DEPTH\_TEST);

gl10.glDepthFunc(GL10.GL\_LESS);

gl10.glEnable(GL10.GL\_BLEND);

gl10.glBlendFunc(GL10.GL\_SRC\_ALPHA, GL10.GL\_ONE\_MINUS\_SRC\_ALPHA);

gl10.glEnable(GL10.GL\_TEXTURE\_2D);

matrices.get(0).Draw(gl10);

matrices.get(1).Draw(gl10);

matrices.get(2).Draw(gl10);

matrices.add(1, new QuadrilateralTextured(gl10,

new float[]

{

-1.5f + dx\_earth, -1.5f + dy\_earth, 0f,

-1.5f + dx\_earth, 1.5f + dy\_earth, 0f,

1.5f + dx\_earth, -1.5f + dy\_earth, 0f,

1.5f + dx\_earth, 1.5f + dy\_earth, 0f,

},

(Bitmap) BitmapFactory.decodeStream(new\_activity.getResources().openRawResource(R.raw.earth))

));

matrices.add(2, new QuadrilateralTextured(

gl10,

new float[] {

-1.5f + dx\_moon, -1.5f + dy\_moon, 0f,

-1.5f + dx\_moon, 1.5f + dy\_moon, 0f,

1.5f + dx\_moon, -1.5f + dy\_moon, 0f,

1.5f + dx\_moon, 1.5f + dy\_moon, 0f,

},

(Bitmap) BitmapFactory.decodeStream(new\_activity.getResources().openRawResource(R.raw.moon))

));

gl10.glDisableClientState(GL10.GL\_COLOR\_ARRAY);

gl10.glDisableClientState(GL10.GL\_VERTEX\_ARRAY);

gl10.glDisableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY);

gl10.glDisable(GL10.GL\_TEXTURE\_2D);

gl10.glDisable(GL10.GL\_BLEND);

gl10.glDisable(GL10.GL\_DEPTH\_TEST);

}

}

**QuadriatelTexture.java – создание текстур**

interface Object3D

{

void Draw(GL10 gl10);

}

class QuadrilateralTextured implements Object3D

{

private final int[] textures = new int[1];

private final ShortBuffer order\_buffer;

private final FloatBuffer coordinates\_buffer;

private final FloatBuffer texture\_coordinates\_buffer;

@RequiresApi(api = Build.VERSION\_CODES.N)

private FloatBuffer new\_coordinates\_buffer(float[] coordinates\_buffer)

{

ByteBuffer byte\_coordinates\_buffer = ByteBuffer.allocateDirect(Float.BYTES \* coordinates\_buffer.length);

byte\_coordinates\_buffer.order(ByteOrder.nativeOrder());

FloatBuffer vertex\_coordinates\_buffer = byte\_coordinates\_buffer.asFloatBuffer();

vertex\_coordinates\_buffer.put(coordinates\_buffer);

vertex\_coordinates\_buffer.position(0);

return vertex\_coordinates\_buffer;

}

@RequiresApi(api = Build.VERSION\_CODES.N)

private FloatBuffer new\_texture\_buffer()

{

ByteBuffer byte\_texture\_buffer = ByteBuffer.allocateDirect(Float.BYTES \* 8);

byte\_texture\_buffer.order(ByteOrder.nativeOrder());

FloatBuffer vertex\_texture\_buffer = byte\_texture\_buffer.asFloatBuffer();

vertex\_texture\_buffer.put(new float[] {

0f, 0f, 0f, 1f, 1f, 0f, 1f, 1f

});

vertex\_texture\_buffer.position(0);

return vertex\_texture\_buffer;

}

@RequiresApi(api = Build.VERSION\_CODES.N)

private ShortBuffer new\_order\_buffer()

{

ByteBuffer byte\_order\_buffer = ByteBuffer.allocateDirect(Short.BYTES \* 4);

byte\_order\_buffer.order(ByteOrder.nativeOrder());

ShortBuffer vertex\_order\_buffer = byte\_order\_buffer.asShortBuffer();

vertex\_order\_buffer.put(new short[]{

0, 1, 2, 3

});

vertex\_order\_buffer.position(0);

return vertex\_order\_buffer;

}

@RequiresApi(api = Build.VERSION\_CODES.N)

QuadrilateralTextured(GL10 gl10, float[] vertices, Bitmap bitmap)

{

coordinates\_buffer = this.new\_coordinates\_buffer(vertices);

texture\_coordinates\_buffer = this.new\_texture\_buffer();

order\_buffer = this.new\_order\_buffer();

gl10.glGenTextures(1, textures, 0);

gl10.glBindTexture(GL10.GL\_TEXTURE\_2D, textures[0]);

gl10.glTexParameterf(GL10.GL\_TEXTURE\_2D, GL10.GL\_TEXTURE\_MIN\_FILTER, GL10.GL\_NEAREST);

GLUtils.texImage2D(GL10.GL\_TEXTURE\_2D,0, bitmap,0);

}

@Override

public void Draw(GL10 gl10)

{

gl10.glEnableClientState(GL10.GL\_VERTEX\_ARRAY);

gl10.glVertexPointer(3, GL10.GL\_FLOAT, GL10.GL\_ZERO, this.coordinates\_buffer);

gl10.glBindTexture(GL10.GL\_TEXTURE\_2D, textures[0]);

gl10.glEnableClientState(GL10.GL\_TEXTURE\_COORD\_ARRAY);

gl10.glTexCoordPointer(2, GL10.GL\_FLOAT, GL10.GL\_ZERO, this.texture\_coordinates\_buffer);

gl10.glDrawElements(GL10.GL\_TRIANGLE\_STRIP, 4, GL10.GL\_UNSIGNED\_SHORT, this.order\_buffer);

}

}

**Результаты работы**

****