Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Лабораторная работа №5

«Матрицы модели-вида OpenGL ES»

Выполнил: студент 4 курса ИВТ,

гр. ИП-713

Михеев Н.А.

Проверил: ассистент кафедры ПМиК,

Павлова У.В.

Новосибирск, 2020 г.

Задание

Создать водную поверхность, прозрачную до дна (взять произвольный рисунок). По поверхности должна идти волна.

Решение поставленной задачи

Был реализован класс MyRenderer в котором происходит просчет модели воды и ее отрисовка, появление всплесков воды по таймеру.

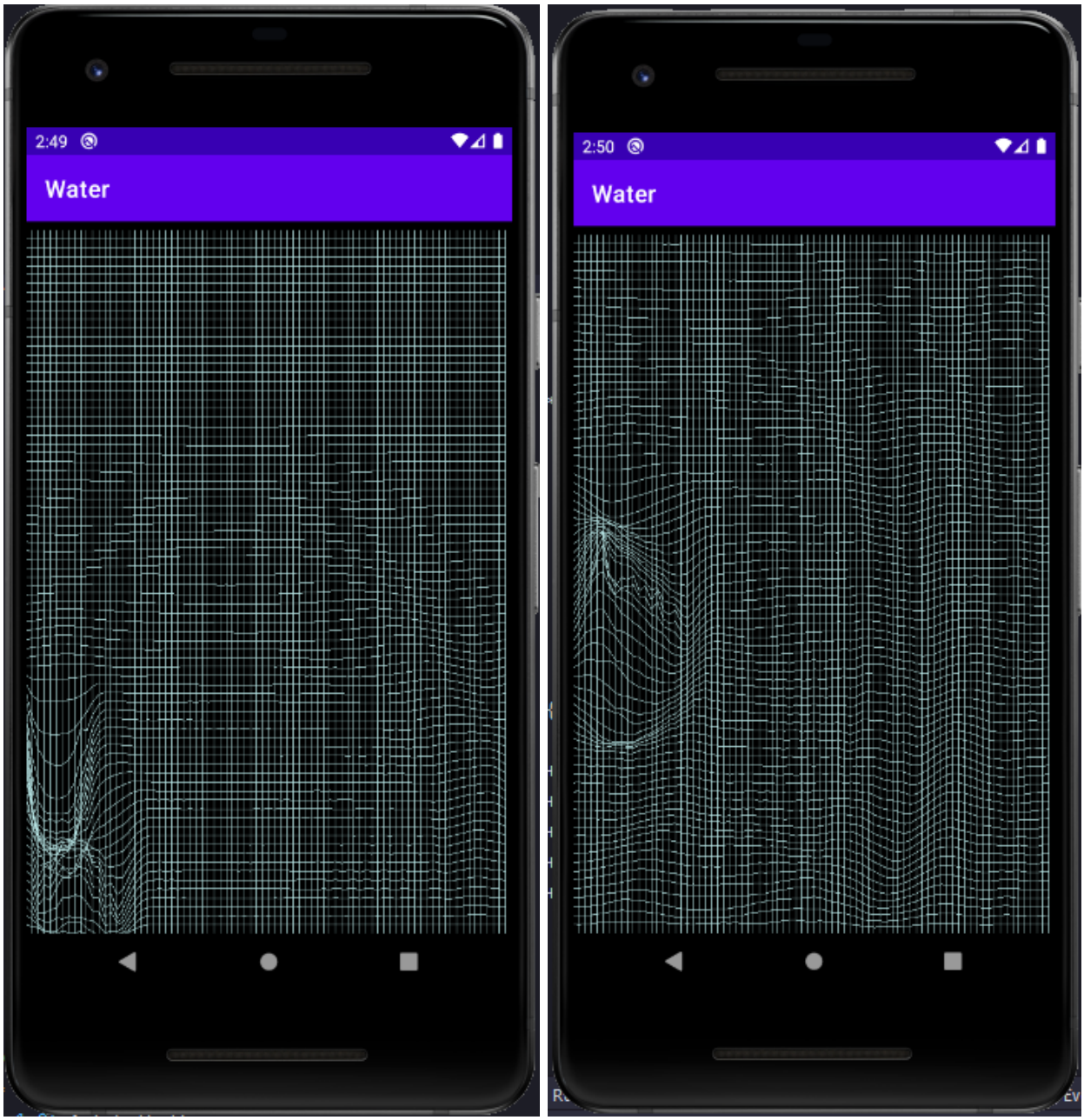

Рис. 1 – демонстрация работы программы.

Листинг программы

MainActivity:

```
package ru.mikheev.water;

import androidx.appcompat.app.AppCompatActivity;

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

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        GLSurfaceView gLView = new GLSurfaceView(this);
        gLView.setBackgroundResource(R.drawable.water);
        gLView.setZOrderOnTop(true);
        gLView.setRenderer(new MyRenderer());
        setContentView(gLView);
    }
}
```

MyGLRenderer:

```
package ru.mikheev.water;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.opengl.GLES10;
import android.opengl.GLES20;
import android.opengl.GLSurfaceView;
import android.opengl.GLUtils;

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

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

class P {
    P() {z=0; vz=0;}
    public float x, y, z, vz;
}

class MyRenderer implements GLSurfaceView.Renderer {
    final int N = 80;
    float K = 0.06f;
    float DT = 0.1f;
```

```
int offs = 0;
public P [][] p;
float time;
float [] a;
FloatBuffer f;
ByteBuffer b;

float sqr(float x) {
    return x * x;
}

void NioBuff() {
    b = ByteBuffer.allocateDirect(2 * 2 * 3 * N * N * 4);
    b.order(ByteOrder.nativeOrder());
    f = b.asFloatBuffer();
    f.put(a);
    f.position(0);
}

void display() {
    offs = 0;
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N - 1; j++) {
            a[N * i * 3 * 2 + j * 3 * 2] = 1.0f * j / N;
            a[N * i * 3 * 2 + j * 3 * 2 + 1] = 1.0f * i / N;
            a[N * i * 3 * 2 + j * 3 * 2 + 2] = 1.0f * (p[i][j]).z;
            a[N * i * 3 * 2 + j * 3 * 2 + 3] = 1.0f * (j + 1) / N;
            a[N * i * 3 * 2 + j * 3 * 2 + 4] = 1.0f * (i) / N;
            a[N * i * 3 * 2 + j * 3 * 2 + 5] = 1.0f * (p[i][j + 1]).z;
            offs += 6;
        }
    }
    for (int i = 0; i < N - 1; i++) {
        for (int j = 0; j < N; j++) {
            a[offs + N * i * 3 * 2 + j * 3 * 2] = 1.0f * j / N;
            a[offs + N * i * 3 * 2 + j * 3 * 2 + 1] = 1.0f * i / N;
            a[offs + N * i * 3 * 2 + j * 3 * 2 + 2] = 1.0f * (p[i][j]).z;
            a[offs + N * i * 3 * 2 + j * 3 * 2 + 3] = 1.0f * (j) / N;
            a[offs + N * i * 3 * 2 + j * 3 * 2 + 4] = 1.0f * (i + 1) / N;
            a[offs + N * i * 3 * 2 + j * 3 * 2 + 5] = 1.0f * (p[i + 1][j]).z;
        }
    }
}

@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    a = new float [12 * N * N];
    p = new P[N][N];
}

@Override
public void onSurfaceChanged(GL10 gl, int width, int height) {
    time = 0;
    for (int i = 0; i < N; i++) {
```

```
            for (int j = 0; j < N; j++) {
                p[i][j] = new P();
                (p[i][j]).x = 1.0f * j / N;
                (p[i][j]).y = 1.0f * i / N;
                (p[i][j]).z = 0;
                (p[i][j]).vz = 0;
            }
        }
    }

    void Push1() {
        if (Math.random() * 500 > 10) { return; }
        int x0 = (int)(Math.random() * N / 2 + 1);
        int y0 = (int)(Math.random() * N / 2 + 1);
        for(int y = y0 - 5; y < y0 + 5; y++) {
            if ((y < 1) || (y >= N - 1)) continue;
            for (int x = x0 - 5; x < x0 + 5; x++) {
                if ((x < 1) || (x >= N - 1)) continue;
                p[x][y].z = 10.0f / N - (float)(Math.sqrt(sqr(y - y0) + sqr(x - x0))
* 1.0 / N);
            }
        }
    }

    void MyTimer() {
        final int []dx = { -1, 0, 1, 0 };
        final int []dy = { 0, 1, 0, -1 };
        Push1();
        for(int y = 1; y < N - 1; ++y) {
            for(int x = 1; x < N - 1; ++x) {
                P p0 = p[x][y];
                for (int i = 0; i < 4; ++i) {
                    P p1 = p[x + dx[i]][y + dy[i]];
                    float d = (float)Math.sqrt(sqr(p0.x - p1.x) + sqr(p0.y - p1.y) +
sqr(p0.z-p1.z));
                    p0.vz += K * (p1.z - p0.z) / d * DT;
                    p0.vz *= 0.99f;
                }
            }
        }
        for(int y = 1; y < N - 1; ++y)
            for(int x = 1; x < N - 1; ++x) {
                P p0 = p[x][y];
                p0.z += p0.vz;
            }
        display();
    }

    @Override
    public void onDrawFrame(GL10 gl) {
        gl.glClearColor(0, 0, 0, 1);
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT);

        gl.glLoadIdentity();
```

```
        gl.glTranslatef(-1f, -1f, 1f);
        gl.glScalef(2f, 4f, 0);
        gl.glRotatef(60, 1, 0, 0);
        gl.glColor4f(0.8f, 1f, 1f, 1);
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        //gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);

        gl.glEnable(GL10.GL_ALPHA_TEST);
        gl.glEnable(GL10.GL_BLEND);
        gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE_MINUS_SRC_ALPHA);

        MyTimer();
        NioBuff();
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, f);
        gl.glDrawArrays(GL10.GL_LINES, 0, 4 * N * (N));
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        //gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
    }
}
```