Федеральное агентство связи  
Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

**Лабораторная работа по теме:**

**«Водная гладь»**

Выполнили:

студентки 4 курса

ИВТ, гр. ИП-712

Гервас А.В.

Онищенко А.В.

Новосибирск 2020

Оглавление

[Задание 3](#_Toc56735192)

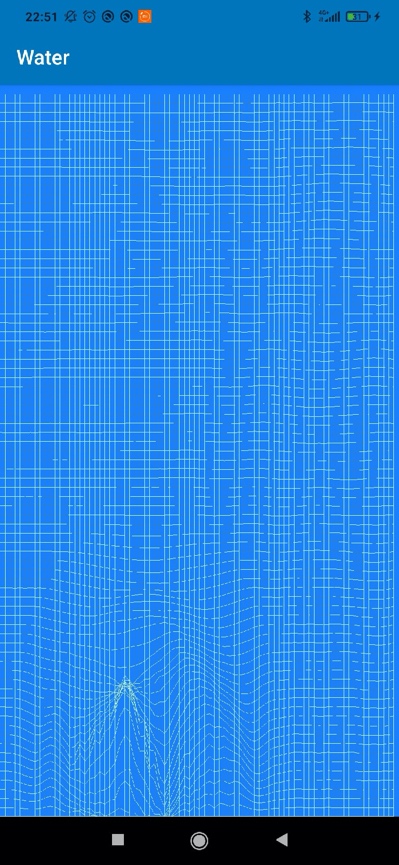
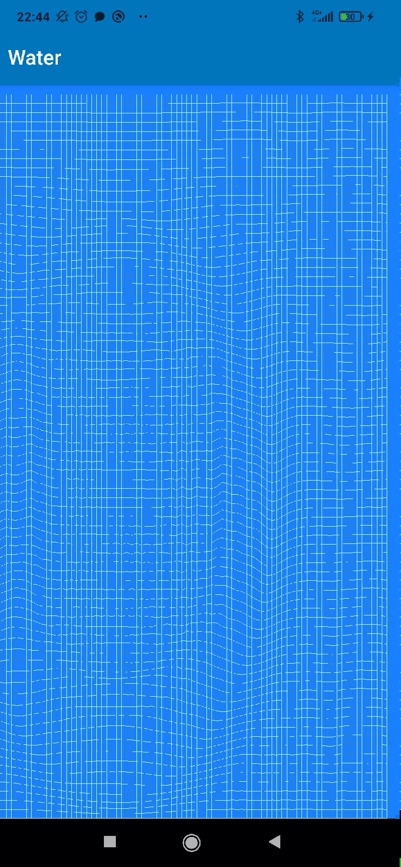
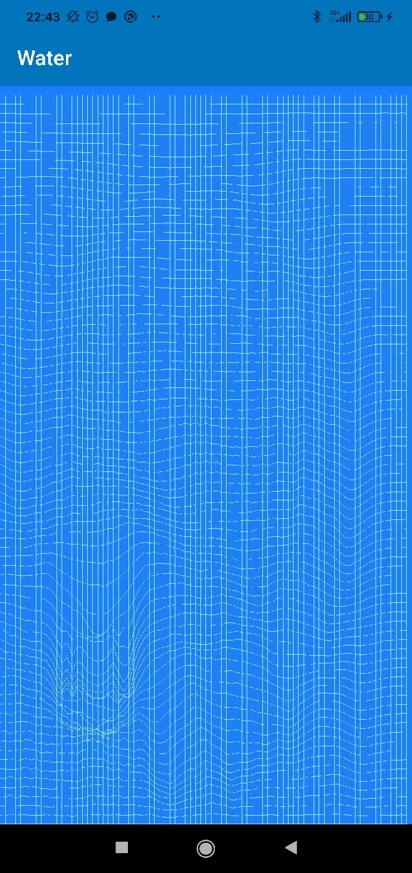
[Скриншоты 3](#_Toc56735193)

[Листинг кода 3](#_Toc56735194)

**Задание**

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

**Скриншоты**

**Листинг кода**

Приложение написано на языке Java.

**MainActivity.java**

package ru.sibsutis.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 view = new GLSurfaceView(this);

view.setRenderer(new WaterRenderer());

view.setRenderMode(GLSurfaceView.RENDERMODE\_CONTINUOUSLY);

setContentView(view);

}

}

**P.java**

package ru.sibsutis.water;

public class P {

P() {

z = 0;

vz = 0;

}

public float x, y, z, vz;

}

**WaterRenderer.java**

package ru.sibsutis.water;

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;

public class WaterRenderer implements GLSurfaceView.Renderer {

final int N = 60; // длина распространения волны

float K = 0.06f; // коэф. деформации поверхности при попадании капли о поверхность

float DT = 0.1f; // клэф. распространение волны по поверхности

int offs = 0; // смещение

public P[][] p;

float time;

float[] a; // массив вершин

FloatBuffer f;

ByteBuffer b;

float sqr(float x) {

return x \* x;

}

void NioBuff() {

// создадим буфер для хранения координат вершин бассейна

b = ByteBuffer.allocateDirect(2 \* 2 \* 3 \* N \* N \* 4);

b.order(ByteOrder.nativeOrder());

f = b.asFloatBuffer();

// перепишем координаты вершин из массива в буфер

f.put(a);

f.position(0);

}

// перемещение волны по поверхности

void Init1() {

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 = 2.0f \* j / N;

(p[i][j]).y = 2.0f \* i / N;

(p[i][j]).z = 0;

(p[i][j]).vz = 0;

}

}

}

// ф-ция отрисовки дна (сетка)

void display() {

offs = 0;

// цикл отрисовки горизонтальных линий

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;

}

}

// цикл отрисовки вертикальных линий

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]; // массив вершин

p = new P[N][N]; // позиций

}

@Override

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

Init1(); // перемещение волны по поверхности

}

// ф-ция рандомного попадания капли на поверхность воды

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)); // расстояние между координатами

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(); // двидение волны по поверхности

}

@Override

public void onDrawFrame(GL10 gl) {

gl.glClearColor(0.1f, 0.5f, 1, 1); // фон дна бассейна

gl.glClear(GL10.GL\_COLOR\_BUFFER\_BIT); //очищаем кадр

gl.glLoadIdentity(); // заменяет текущую матрицу на единичную матрицу

gl.glTranslatef(-1f, -1f, 0); // умножает текущую матрицу на матрицу перевода для дальнейшего смещения

gl.glScalef(2f, 4f, 0);

gl.glRotatef(60, 1, 0, 0);

gl.glColor4f(0.8f, 1f, 1f, 1);

gl.glEnableClientState(GL10.GL\_VERTEX\_ARRAY); // устанавливает состояние клиентской части

MyTimer();

NioBuff();

gl.glVertexPointer(3, GL10.GL\_FLOAT, 0, f); // определяет массив данных вершин

gl.glDrawArrays(GL10.GL\_LINES, 0, 4 \* N \* (N));

gl.glDisableClientState(GL10.GL\_VERTEX\_ARRAY); // отключение массивов

}

}