МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра АСУ

**ОТЧЕТ**

по лабораторной работе №2

«Примитивы OpenGl»

Дисциплина «Компьютерная графика», 3 семестр

Преподаватель: Павенко Евгений Николаевич

Группа: АВТ-113

Студенты:

Осокин Дмитрий

Горшков Данил

Кашаев Аскер

г. Новосибирск

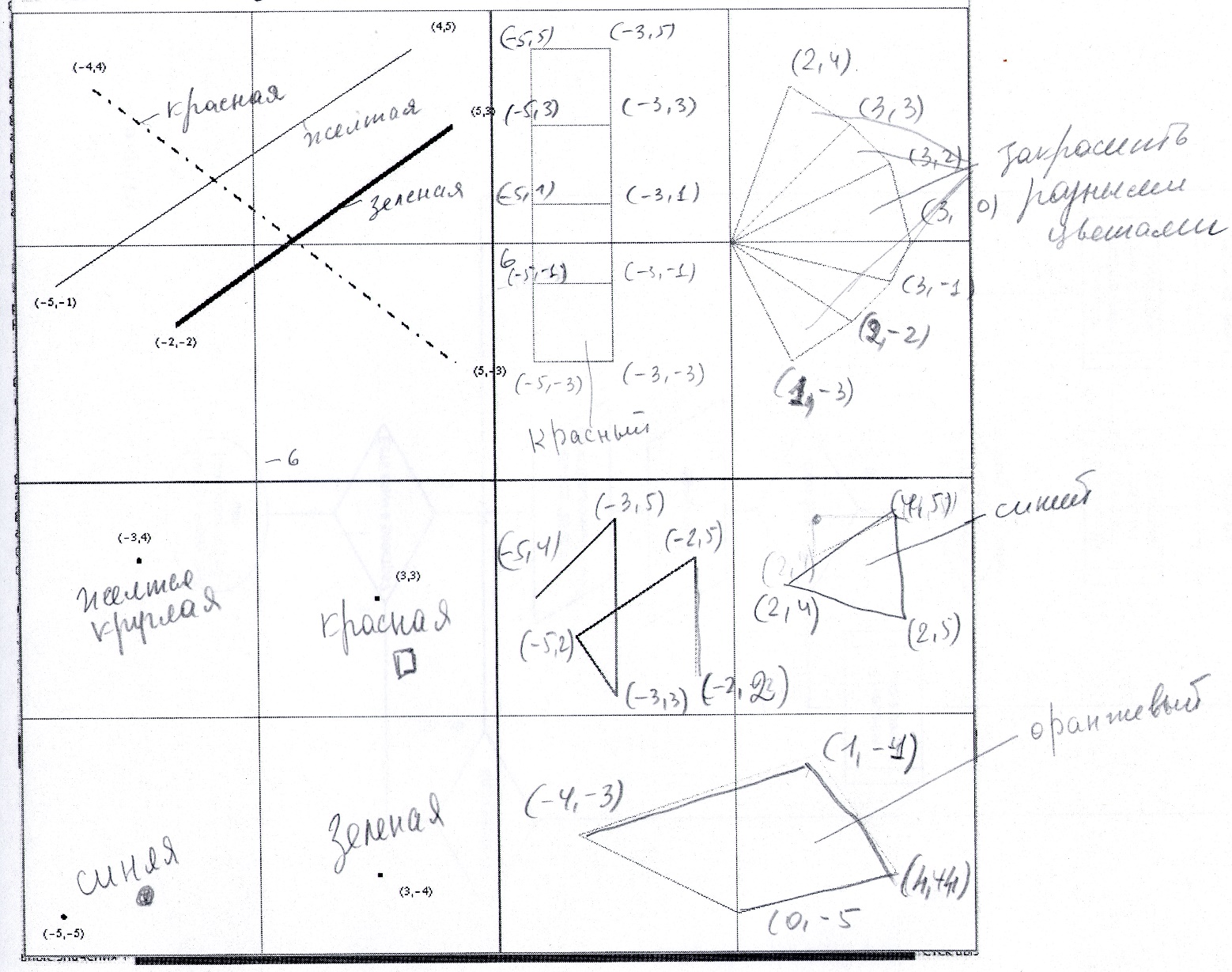
2022 год

**Цель работы**

Научиться работать с примитивами в OpenGL.

**Задание:**

Изобразить с помощью OpenGL данную картину.



**Приложение.**

****

#include <windows.h> /\*подключать библиотеку следует только под Windows

под Linux используем start.sh с кодом

g++ laba2main.cpp -lGL -lGLU -lglut && ./a.out

\*/

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glut.h>

#include <iostream>

#define WINDOW\_WIDTH 800

#define WINDOW\_HEIGHT 800

int viewportSize = WINDOW\_WIDTH;

void drawString(std::string str, double posX, double posY) {

int i = 0;

glColor3d(1, 1, 1);

while (str[i] != '\0') {

glRasterPos2d(posX, posY);

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, str[i]);

posX += 0.12;

i++;

}

}

void display(void) {

std::string title = "Gorshkov Osokin Kashaev ABT-113\0";

drawString(title, -4, 6.5);

glViewport(0, 0, 400, 400);

glPointSize(6);

// Bottom-left points

glEnable(GL\_POINT\_SMOOTH);

glBegin(GL\_POINTS);

glColor3d(0, 1, 0);

glVertex2d(-3, 4);

glEnd();

glDisable(GL\_POINT\_SMOOTH);

glBegin(GL\_POINTS);

glColor3d(0, 0, 1);

glVertex2d(-5, -5);

glColor3d(1, 0, 0);

glVertex2d(3, 3);

glColor3d(1, 1, 0);

glVertex2d(3, -4);

glEnd();

// Top-left lines

glViewport(0, 400, 400, 400);

glBegin(GL\_LINES);

glColor3d(1, 1, 0);

glVertex2d(-5, -1);

glVertex2d(4, 5);

glEnd();

glLineWidth(3);

glBegin(GL\_LINES);

glColor3d(0, 1, 0);

glVertex2d(-2, -2);

glVertex2d(5, 3);

glEnd();

glLineWidth(2);

glEnable(GL\_LINE\_SMOOTH | GL\_LINE\_STIPPLE);

glLineStipple(2, 58360);

glBegin(GL\_LINES);

glColor3d(1, 0, 0);

glVertex2d(-4, 4);

glVertex2d(5, -3);

glEnd();

// Bottom right strip lines

glViewport(400, 0, 400, 400);

glDisable(GL\_LINE\_SMOOTH | GL\_LINE\_STIPPLE);

glBegin(GL\_LINE\_STRIP);

glColor3d(1, 1, 0);

glVertex2d(-5, 4);

glVertex2d(-3, 5);

glVertex2d(-3, 3);

glVertex2d(-5, 2);

glVertex2d(-2, 5);

glVertex2d(-2, 2);

glEnd();

// Bottom right blue triangle

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

glBegin(GL\_TRIANGLES);

glColor3d(0, 0, 1);

glVertex2d(2, 4);

glVertex2d(4, 5);

glVertex2d(2, 5);

glEnd();

// Bottom right orange polygon

glBegin(GL\_QUADS);

glColor3d(1, 0.5, 0);

glVertex2d(-4, -3);

glVertex2d(1, -1);

glVertex2d(4, -4);

glVertex2d(0, -5);

glEnd();

// Top right triangles

glViewport(400, 400, 400, 400);

glShadeModel(GL\_FLAT);

glBegin(GL\_TRIANGLE\_FAN);

glColor3d(1, 0, 0);

glVertex2d(0, 0);

glVertex2d(2, 4);

glVertex2d(3, 3);

glColor3d(0, 1, 0);

glVertex2d(4, 2);

glColor3d(1, 1, 1);

glVertex2d(5, 0);

glColor3d(0, 0, 1);

glVertex2d(4, -1);

glVertex2d(3, -2);

glColor3d(1, 1, 0);

glVertex2d(2, -3);

glEnd();

glShadeModel(GL\_FLAT);

glBegin(GL\_QUAD\_STRIP);

glColor3d(1, 0, 0);

glVertex2d(-5, 5);

glVertex2d(-3, 5);

glVertex2d(-5, 3);

glVertex2d(-3, 3);

glColor3d(0, 1, 0);

glVertex2d(-5, 1);

glVertex2d(-3, 1);

glColor3d(0, 0, 1);

glVertex2d(-5, -1);

glVertex2d(-3, -1);

glColor3d(1, 1, 1);

glVertex2d(-5, -3);

glVertex2d(-3, -3);

glEnd();

glutSwapBuffers();

// glFlush();

}

void init(void) {

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

//glEnable(GL\_DEPTH\_TEST);

glClearColor(0.0, 0.0, 0.0, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-6.0, 6.0, -6.0, 6.0, 2.0, 12.0);

gluLookAt(0, 0, 5, 0, 1, 0, 0, 1, 0);

glMatrixMode(GL\_MODELVIEW);

}

static void reshape(int w, int h) {

viewportSize = min(w, h);

int x = (w - viewportSize) / 2;

int y = (h - viewportSize) / 2;

glViewport(x, y, viewportSize, viewportSize);

init();

}

int main(int argc, char \*\*argv) {

glutInit(&argc, argv);

// glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition(50, 10);

glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

glutCreateWindow("Lab 2");

glutReshapeFunc(reshape);

init();

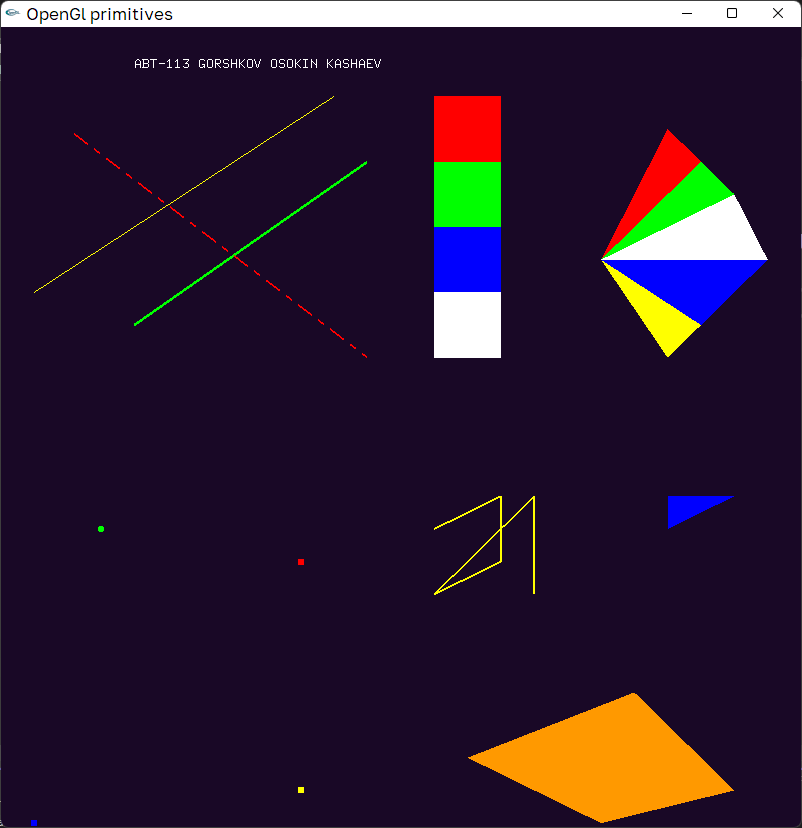
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**Скриншот**



**Дополнительное задание**

Изобразить двумерный узор Серпинского.

Вписанный треугольник делит стороны внешнего треугольника на равные части. Выполнить не менее 10 итераций.

**Приложение.**

point.h

#pragma once

class Point {

public:

Point() : x(0), y(0) {}

Point(double \_x, double \_y) : x(\_x), y(\_y) {}

double getX() const;

double getY() const;

private:

double x;

double y;

};

double Point::getX() const { return x; }

double Point::getY() const { return y; }

main.cpp

****

#include <windows.h> /\*подключать библиотеку следует только под Windows

под Linux используем start.sh с кодом

g++ laba2.1main.cpp -lGL -lGLU -lglut && ./a.out

\*/

#include "point.h"

#include <GL/gl.h>

#include <GL/glu.h>

#include <GL/glut.h>

#include <iostream>

#include <vector>

#define ITERATIONS\_COUNT 10

#define WINDOW\_WIDTH 800

#define WINDOW\_HEIGHT 800

std::string title = "Gorshkov Osokin Kashaev ABT-113\0";

std::vector<Point> startPoints = { Point(-5.0, -5.0), Point(0.0, 5.0),

Point(5.0, -5.0) };

int viewportSize = WINDOW\_WIDTH;

Point getMiddle(Point p1, Point p2) {

Point middlePoint =

Point((p1.getX() + p2.getX()) / 2, (p1.getY() + p2.getY()) / 2);

return middlePoint;

}

// Отрисовка треугольника по заданным координатам

void drawTriangle(std::vector<Point> points) {

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

glBegin(GL\_TRIANGLES);

glColor3d(1, 1, 1);

glVertex2d(points[0].getX(), points[0].getY());

glVertex2d(points[1].getX(), points[1].getY());

glVertex2d(points[2].getX(), points[2].getY());

glEnd();

}

void drawString(std::string str, double posX, double posY) {

int i = 0;

while (str[i] != '\0') {

glRasterPos2d(posX, posY);

glutBitmapCharacter(GLUT\_BITMAP\_8\_BY\_13, str[i]);

posX += 0.95/(viewportSize / 100);

i++;

}

}

// Отрисовка треугольника Серпинского рекурсивным способом

void drawSerpinsky(std::vector<Point> points, int iterationsCount) {

if (iterationsCount == 0)

return;

// Отрисовываем треугольник по заданными точкам

drawTriangle(points);

// Левый треугольник

std::vector<Point> middles;

middles.push\_back(points[0]);

middles.push\_back(getMiddle(points[0], points[1]));

middles.push\_back(getMiddle(points[0], points[2]));

drawSerpinsky(middles, iterationsCount - 1);

middles.clear();

// Верхний треугольник

middles.push\_back(points[1]);

middles.push\_back(getMiddle(points[0], points[1]));

middles.push\_back(getMiddle(points[1], points[2]));

drawSerpinsky(middles, iterationsCount - 1);

middles.clear();

// Правый треугольник

middles.push\_back(points[2]);

middles.push\_back(getMiddle(points[2], points[1]));

middles.push\_back(getMiddle(points[0], points[2]));

drawSerpinsky(middles, iterationsCount - 1);

middles.clear();

}

void display(void) {

// Отображение треугольника Серпинского

drawTriangle(startPoints);

drawSerpinsky(startPoints, ITERATIONS\_COUNT);

drawString(title, -5.0 / (viewportSize / 250.0), 6.0);

glutSwapBuffers();

}

void init(void) {

glEnable(GL\_COLOR\_MATERIAL);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

//glEnable(GL\_DEPTH\_TEST);

glClearColor(0.19, 0.12, 0.27, 0.0);

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-6.0, 6.0, -6.0, 6.0, 2.0, 12.0);

gluLookAt(0, 0, 5, 0, 1, 0, 0, 1, 0);

glMatrixMode(GL\_MODELVIEW);

}

static void reshape(int w, int h) {

viewportSize = min(w, h);

int x = (w - viewportSize) / 2;

int y = (h - viewportSize) / 2;

glViewport(x, y, viewportSize, viewportSize);

init();

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); //UPD: deleted GLUT\_DEPTH

glutInitWindowSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

glutInitWindowPosition(50, 50);

glutCreateWindow("Lab 2 : Serpinsky");

init();

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**Скриншот дополнительного задания:**

