Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ПМТ

Лабораторная работа № 1

по дисциплине «Компьютерная графика»

Факультет:	ПМИ
Группа:	ПМ-71
Вариант:	
Студенты:	Востриков Вячеслав
	Бурдуков Вадим
	Баштовой Павел
Преподаватели:	Задорожный А. Г.

Новосибирск

2020

1. **Цель работы**

   Ознакомиться с основами использования библиотеки OpenGL

2. **Используемые структуры хранения данных**

   Для реализации объектов программы были использован класс «group_of_objects», который описывает группу элементов (точек). Также каждая группа состоит из точек (содержит структуру type_point)

```
struct group_of_objects // Создание группы объектов
{
	group_of_objects(GLdouble rt, GLubyte R, GLubyte G, GLubyte B, GLubyte Size)
	{
		this->ColorB = B;
		this->ColorG = G;
		this->ColorR = R;
		this->PointSize = Size;
		this->rotation = rt;
	}
	vector <type_point> Points;
	GLubyte ColorR, ColorG, ColorB, PointSize;
	GLdouble rotation;
	unsigned int type_of_gr = 0;
};
vector <group_of_objects> groups;
struct type_point // задание контейнера вершин
{
	GLint x, y;
	type_point(GLint _x, GLint _y) { x = _x; y = _y;}
	 };
```

   Перед каждым выводом групп объектов вызывается функция get_point(), которая возвращает пересчитанные точки под соответствующую систему координат (у каждой группы своя «ск»).

```
type_point get_point(const GLdouble rot, const type_point point)
{
	type_point new_point(point.x, point.y);
	if (rot != 0) {
		new_point.x = point.x * cos(rot) - point.y * sin(rot);
		new_point.y = point.x * sin(rot) + point.y * cos(rot);
```

```
        }
        return new_point;
}
```

Также перед каждым движением вперед, назад, лево, право, вызывается фунция prepare_points(), которая обнуляет коэффициенты вращения системы координат, чтобы движения были «правильными»

```
void prepare_points()
{
type_point* p;
for (int i = 0; i < groups[i_groups].Points.size(); i++)
{
        p = &get_point(groups[i_groups].rotation, groups[i_groups].Points[i]);
        groups[i_groups].Points[i].x = p->x;
        groups[i_groups].Points[i].y = p->y;
}
groups[i_groups].rotation = 0;
}
```

3. Справка пользователя

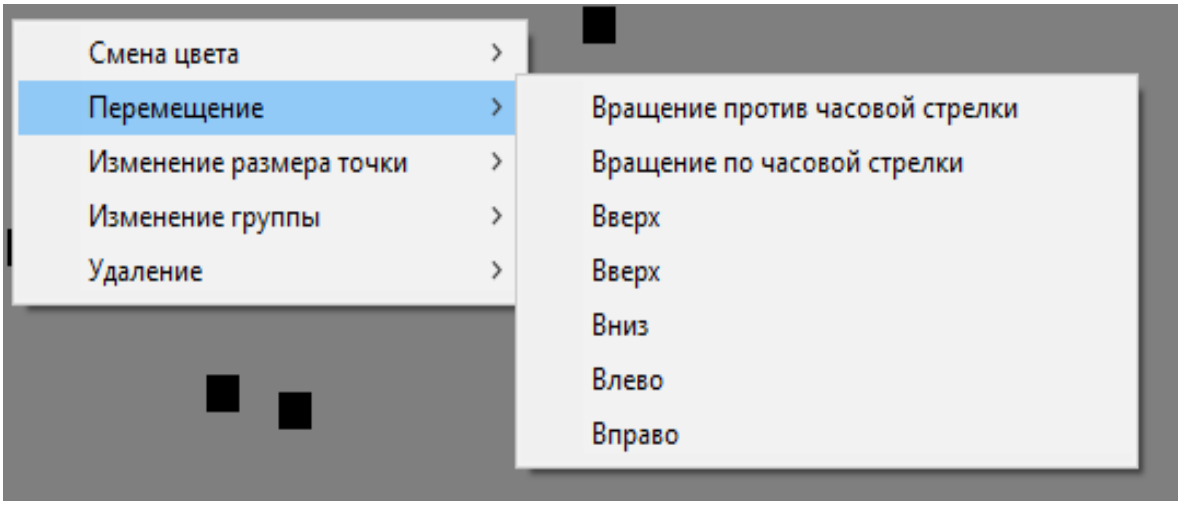
Управление данным графическим редактором осуществляется с помощью мыши и клавиатуры.

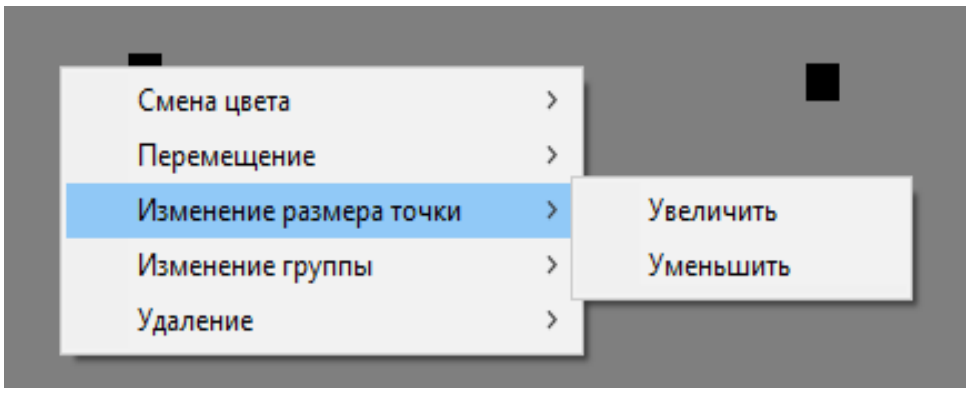
Управление	**Назначение**
Левая кнопка мыши	Добавление точки
q	Вращение против часовой стрелке
e	Вращение против часовой стрелки
8 или 7	Изменение примитива
l	Удаление текущей группы
0 и 9	Добавление новых групп/ изменение текущей группы **(ниже комментарий)**
u и l	Изменение размеров точек
Средняя кнопка мышки	Удаление последней точки текущей группы
g	Добавление зеленого цвета к группе
b	Добавление синего цвета к группе
r	Добавление красного цвета к группе

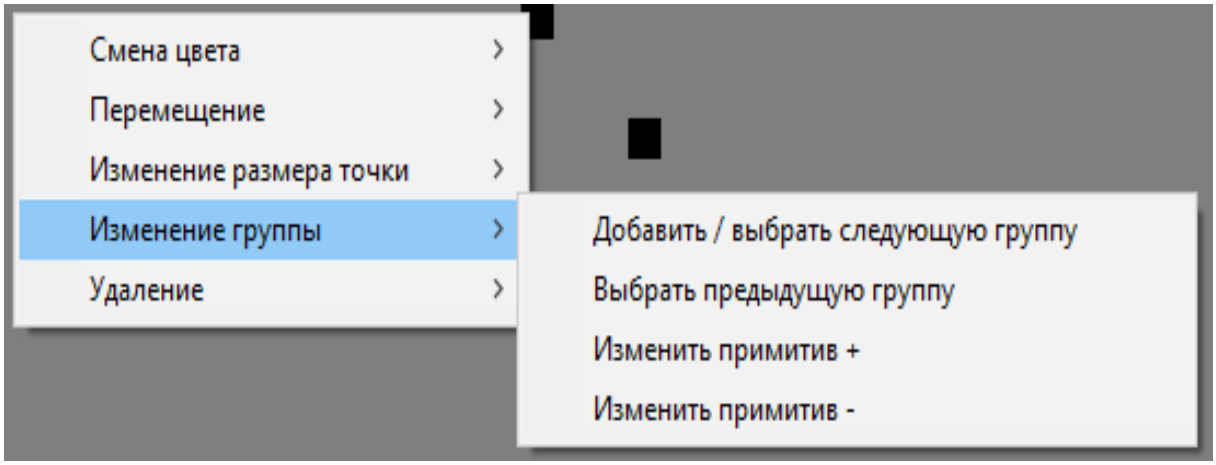
(	Переход от одной нарисованной фигуры к другой (от текущей к предыдущей)
)	Переход от одной нарисованной фигуры к другой (от текущей к следующей)
w	Смещение выбранной фигуры вверх
a	Смещение выбранной фигуры влево
s	Смещение выбранной фигуры вниз
d	Смещение выбранной фигуры направо

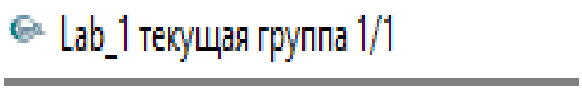
Клавиша 0 выбирает следующую группу в случае, если она существует, в ином случае добавляет новую. Клавиша 9 выбирает предыдущую, пока такая группа существует, и в ином случае выбирает последнюю.

Также программой можно управлять с помощью меню, которое появляется при нажатии на правую клавишу мыши



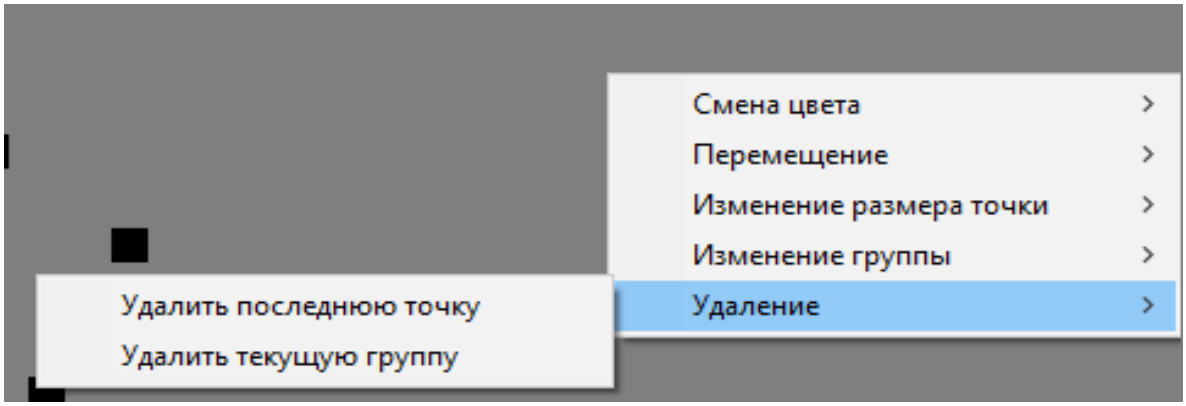






Последний скриншот показывает какая группа выбрана из общего количества (выбрали первую и единственную группу).

4. Текст программы




```
#include <math.h>
#include <stdlib.h>
#include "glut.h"
#include <vector>
using namespace std;

GLint Width = 512, Height = 512;
enum keys { Empty, KeyR, KeyG, KeyB, KeyW, KeyA, KeyS, KeyD, KeyU, KeyI, KeyL, KeyQ, KeyE, Key0,
Key9, Key8, Key7, Key_middle_mouse };
int i_groups = 0;
```

```
struct type_point // задание контейнера вершин
{
	GLint x, y;
	type_point(GLint _x, GLint _y) { x = _x; y = _y;}
};

struct group_of_objects // Создание группы объектов
{
	group_of_objects(GLdouble rt, GLubyte R, GLubyte G, GLubyte B, GLubyte Size)
	{
		this->ColorB = B;
		this->ColorG = G;
		this->ColorR = R;
		this->PointSize = Size;
		this->rotation = rt;
	}
	vector <type_point> Points;
	GLubyte ColorR, ColorG, ColorB, PointSize;
	GLdouble rotation;
	unsigned int type_of_gr = 0;
};
vector <group_of_objects> groups;

type_point get_point(const GLdouble rot, const type_point point)
{
	type_point new_point(point.x, point.y);
	if (rot != 0) {
		new_point.x = point.x * cos(rot) - point.y * sin(rot);
		new_point.y = point.x * sin(rot) + point.y * cos(rot);
	}
	return new_point;
}

void prepare_points()
{
	type_point* p;
	for (int i = 0; i < groups[i_groups].Points.size(); i++)
	{
		p = &get_point(groups[i_groups].rotation, groups[i_groups].Points[i]);
		groups[i_groups].Points[i].x = p->x;
		groups[i_groups].Points[i].y = p->y;
```

```
        }
        groups[i_groups].rotation = 0;
}

void Display(void) // Функция вывода на экран
{
        glClearColor(0.5, 0.5, 0.5, 1); glClear(GL_COLOR_BUFFER_BIT);

        int size_group = groups.size();
        for (int k = 0; k < size_group; k++)
        {
                type_point* pointer;
                glColor3ub(groups[k].ColorR, groups[k].ColorG, groups[k].ColorB);
                glPointSize(groups[k].PointSize);
                //glMatrixMode();
                //glLoadIdentity();
                glBegin(groups[k].type_of_gr);
                for (int i = 0; i < groups[k].Points.size(); i++)
                {
                        pointer = &get_point(groups[k].rotation ,groups[k].Points[i]);
                        glVertex2i(pointer->x, pointer->y); // Надо менять
                }
                glEnd();
        }
        glFinish();
}

void Reshape(GLint w, GLint h) // Функция изменения размеров окна
{
        Width = w;    Height = h;
        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0, w, 0, h, -1.0, 1.0);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}

void Keyboard(unsigned char key, int x, int y) // Функция обработки сообщений от клавиатуры
{
        int i, n = groups[i_groups].Points.size();
```

```
/* Изменение примитива */
if (key == '8')
{
	if (groups[i_groups].type_of_gr == 9)
		groups[i_groups].type_of_gr = 0;
	else
		groups[i_groups].type_of_gr++;
}
if (key == '7')
{
	if (groups[i_groups].type_of_gr == 0)
		groups[i_groups].type_of_gr = 9;
	else
		groups[i_groups].type_of_gr--;
}

/* Изменение RGB-компонент цвета точек */
if (key == 'r') groups[i_groups].ColorR += 5;
if (key == 'g') groups[i_groups].ColorG += 5;
if (key == 'b') groups[i_groups].ColorB += 5;

if (key == 'w' || key == 's' || key == 'a' || key == 'd') prepare_points();

/* Изменение XY-кординат точек */
if (key == 'w') for (i = 0; i < n; i++) groups[i_groups].Points[i].y += 5;
if (key == 's') for (i = 0; i < n; i++) groups[i_groups].Points[i].y -= 5;
if (key == 'a') for (i = 0; i < n; i++) groups[i_groups].Points[i].x -= 5;
if (key == 'd') for (i = 0; i < n; i++) groups[i_groups].Points[i].x += 5;

/* Угол*/
if (key == 'q') groups[i_groups].rotation += 0.0436332306;
if (key == 'e') groups[i_groups].rotation -= 0.0436332306;

/* Изменение размера точек */
if (key == 'u')
	groups[i_groups].PointSize++;
if (key == 'i')
	groups[i_groups].PointSize--;
/* Управление группами */
if (key == '0')
```

```
	{
		if (i_groups < groups.size() - 1)
			i_groups++;
		else
		{
			groups.push_back(group_of_objects(0, 0, 0, 0, 15));
			i_groups++;
		}
	}
	if (key == '9')
	{
		if (i_groups - 1 < 0)
			i_groups = groups.size() - 1;
		else
			i_groups--;
	}

	/* Удаление текущей группы группы */
	if(key == 'l')
	{
		groups[i_groups].Points.clear();
		groups.erase(groups.begin() + i_groups);
		if (i_groups - 1 < 0)
			i_groups = groups.size() - 1;
		else
			i_groups--;
		if (i_groups == -1)
			i_groups = 0;
	}

	glutPostRedisplay();

	char v[50];
	int k = groups.size();
	sprintf(v, "Lab_1 текущая группа %d/%d", i_groups + 1, k);
	glutSetWindowTitle(v);

}

void Mouse(int button, int state, int x, int y) // Функция обработки сообщения от мыши
{
```

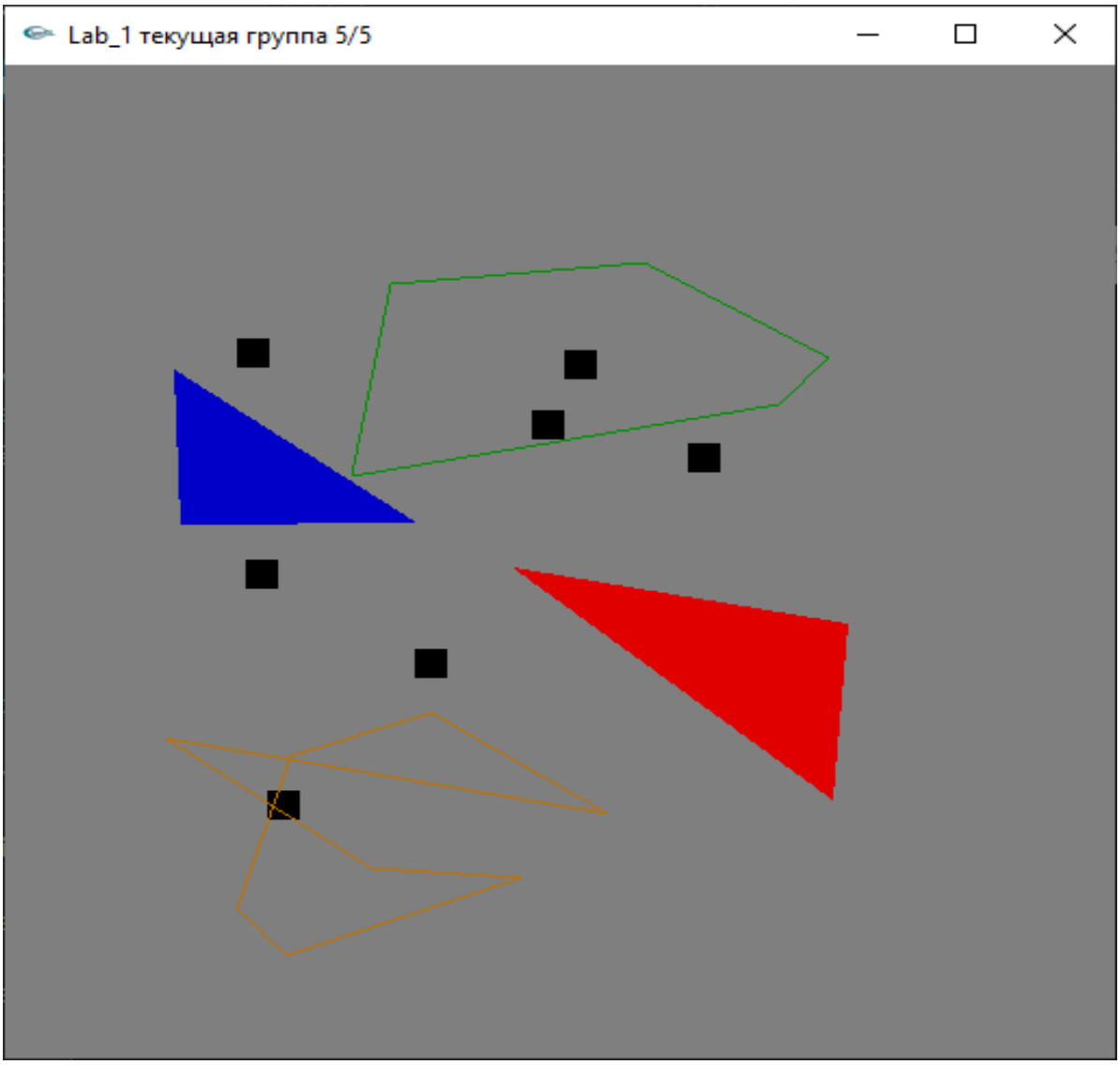
```
	/* клавиша была нажата, но не отпущена */
	if (state != GLUT_DOWN) return;

	/* новая точка по левому клику */
	if (button == GLUT_LEFT_BUTTON)
	{
		type_point p(x, Height - y);
		groups[i_groups].Points.push_back(p);
	}
	/* удаление последней точки по центральному клику */
	if (button == GLUT_MIDDLE_BUTTON)
	{
		if (groups[i_groups].Points.size() != 0)
			groups[i_groups].Points.pop_back();
	}

	glutPostRedisplay();
}

void Menu(int pos)
{
	int key = (keys)pos;

	switch (key)
	{
	case KeyR: Keyboard('r', 0, 0); break;
	case KeyG: Keyboard('g', 0, 0); break;
	case KeyB: Keyboard('b', 0, 0); break;
	case KeyW: Keyboard('w', 0, 0); break;
	case KeyS: Keyboard('s', 0, 0); break;
	case KeyA: Keyboard('a', 0, 0); break;
	case KeyD: Keyboard('d', 0, 0); break;
	case KeyU: Keyboard('u', 0, 0); break;
	case KeyI: Keyboard('i', 0, 0); break;
	case KeyQ: Keyboard('q', 0, 0); break;
	case KeyE: Keyboard('e', 0, 0); break;

	case Key7: Keyboard('7', 0, 0); break;
	case Key8: Keyboard('8', 0, 0); break;
	case Key9: Keyboard('9', 0, 0); break;
	case Key0: Keyboard('0', 0, 0); break;
```

```
case KeyL: Keyboard('l', 0, 0); break;
case Key_middle_mouse: Mouse(GLUT_MIDDLE_BUTTON, 0, 0, 0); break;

default:
        int menu_color = glutCreateMenu(Menu);
        glutAddMenuEntry("Компонента R", KeyR);
        glutAddMenuEntry("Компонента G", KeyG);
        glutAddMenuEntry("Компонента B", KeyB);

        int menu_move = glutCreateMenu(Menu);
        glutAddMenuEntry("Вращение против часовой стрелки", KeyQ);
        glutAddMenuEntry("Вращение по часовой стрелки", KeyE);
        glutAddMenuEntry("Вверх", KeyW);
        glutAddMenuEntry("Вверх", KeyW);
        glutAddMenuEntry("Вниз", KeyS);
        glutAddMenuEntry("Влево", KeyA);
        glutAddMenuEntry("Вправо", KeyD);

        int menu_size = glutCreateMenu(Menu);
        glutAddMenuEntry("Увеличить", KeyU);
        glutAddMenuEntry("Уменьшить", KeyI);

        int menu_group = glutCreateMenu(Menu);
        glutAddMenuEntry("Добавить / выбрать следующую группу", Key0);
        glutAddMenuEntry("Выбрать предыдущую группу", Key9);
        glutAddMenuEntry("Изменить примитив +", Key8);
        glutAddMenuEntry("Изменить примитив -", Key7);

        int menu_del = glutCreateMenu(Menu);
        glutAddMenuEntry("Удалить последнюю точку", Key_middle_mouse);
        glutAddMenuEntry("Удалить текущую группу", KeyL);

        int menu = glutCreateMenu(Menu);
        glutAddSubMenu("Смена цвета", menu_color);
        glutAddSubMenu("Перемещение", menu_move);
        glutAddSubMenu("Изменение размера точки", menu_size);
        glutAddSubMenu("Изменение группы", menu_group);
        glutAddSubMenu("Удаление", menu_del);

        glutAttachMenu(GLUT_RIGHT_BUTTON);
        Keyboard(Empty, 0, 0);
```

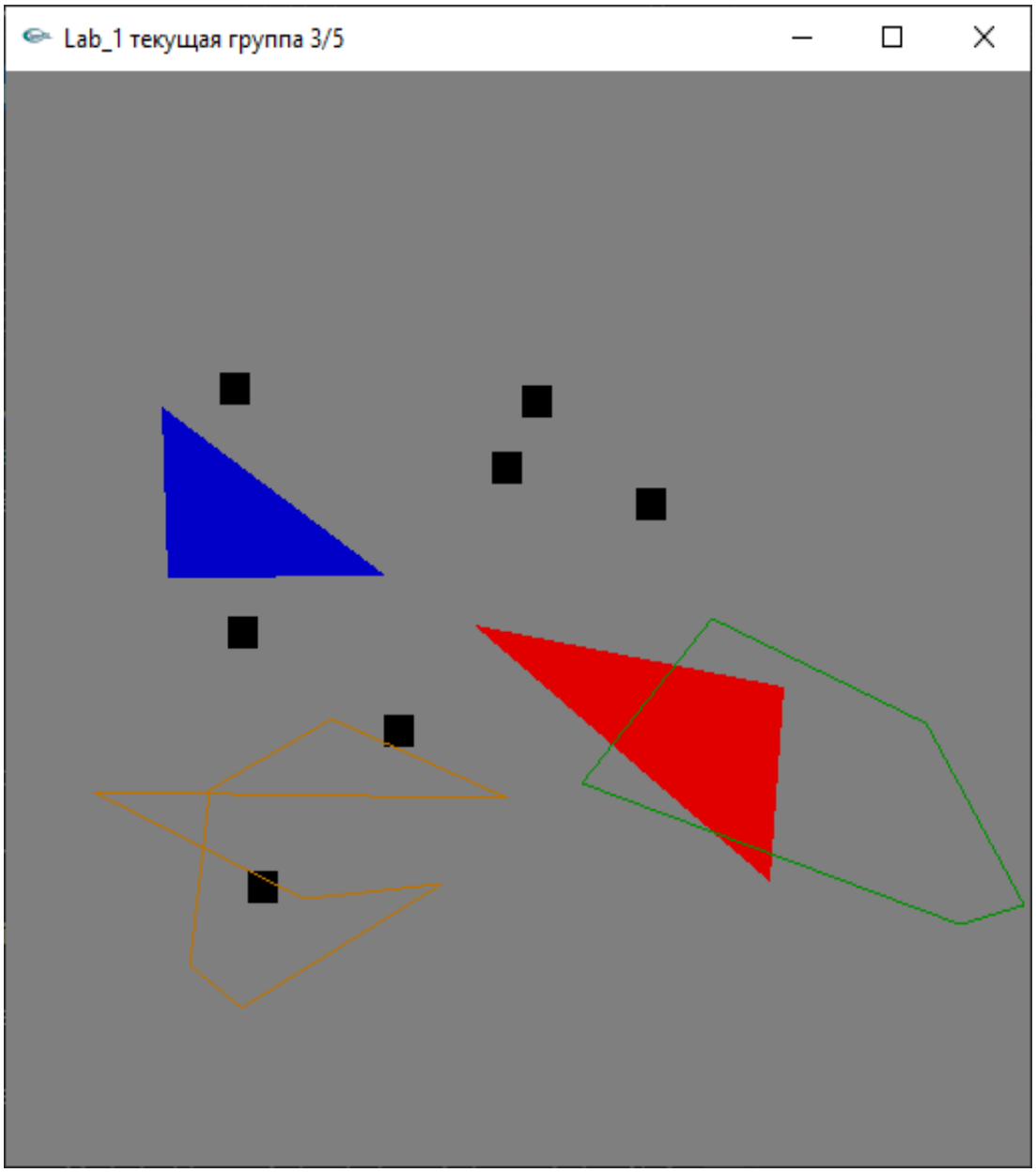
```
        }
}

void main(int argc, char* argv[]) // Головная программа
{
        groups.push_back(group_of_objects(0, 0, 0, 0, 15));
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_RGB);
        glutInitWindowSize(Width, Height);
        glutCreateWindow("Текущий цвет всех точек:");
        Menu(Empty);
        glutDisplayFunc(Display);
        glutReshapeFunc(Reshape);
        glutKeyboardFunc(Keyboard);
        glutMouseFunc(Mouse);

        glutMainLoop();
}
```

5. Результат работы программы

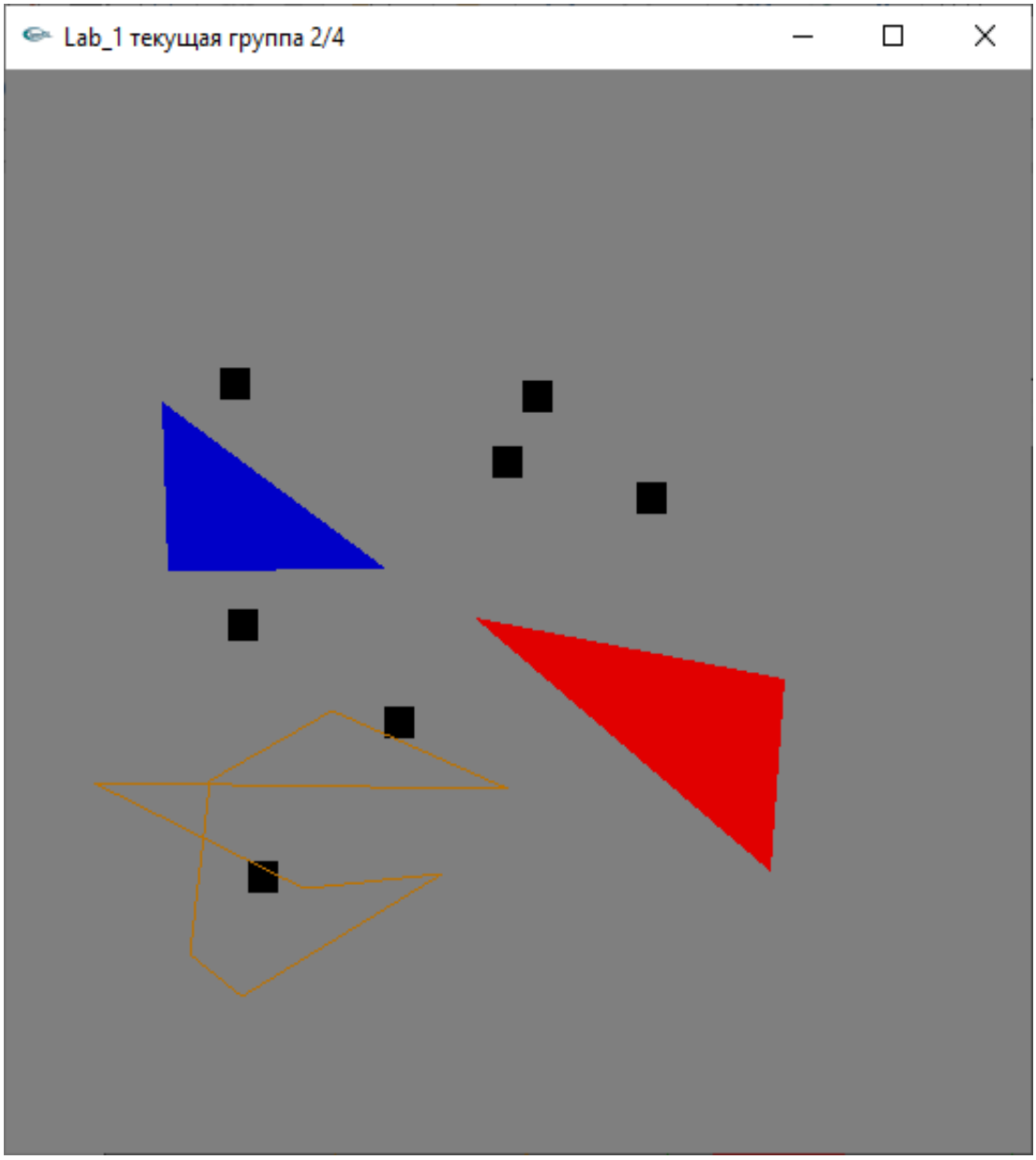
1. Создание нескольких групп



2. Вращение двух групп (зеленая и оранжевая группы)



3. Удаление одной группы (зеленая группа)



4. Удаление четырех точек первой группы