**Министерство Образования и Науки РФ**

**Новосибирский Государственный Технический Университет**

**Кафедра ПМТ**

Практическая работа №1

по дисциплине «Компьютерная графика»

**ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ**

**С ИСПОЛЬЗОВАНИЕМ OPENGL**

|  |  |
| --- | --- |
| Факультет: | ПМИ |
| Группа: | ПМ-12 |
| Вариант: | 5 |
| Выполнили: | Овчинников И. Р.  Лойченко Д. С. |
| Преподаватель: | Задорожный А. Г. |

**Новосибирск**

**2024**

# Цель работы

Ознакомиться с основами использования библиотеки OpenGL и работе с примитивами.

# Задание

# Отобразить в окне множество примитивов (вершины которых задаются кликами мыши) в соответствии с вариантом задания (треугольники).

# Для завершения текущего (активного) набора (множества) примитивов и начала нового зарезервировать специальную клавишу (пробел или правый клик).

# Для текущего набора примитивов предоставить возможность изменения цвета и координат его вершин.

# Текущее множество примитивов выделять среди других.

# Предусмотреть возможность удаления последнего примитива и последнего набора примитивов.

# Продублировать команды в меню.

# Программа написана на языке C# с использованием библиотек SharpGL и WinForms.

# Используемые структуры хранения данных

Для реализации объектов программы были использованы следующие структуры:

1. Структура «точка», описывающий 2D-координаты вершины и ее цвет.

public struct Point

{

public short x, y;

public SimpleColor color;

public Point(short newX, short newY)

{

x = newX;

y = newY;

color = new SimpleColor();

}

public Point(short newX, short newY, SimpleColor newColor)

{

x = newX;

y = newY;

color = newColor;

}

public static Point operator +(Point a, Point b)

{

Point t;

t.x = (short)(a.x + b.x);

t.y = (short)(a.y + b.y);

t.color = a.color;

return t;

}

}

1. Структура «цвет».

public struct SimpleColor

{

public byte r, g, b;

public SimpleColor(byte newR, byte newG, byte newB)

{

r = newR;

g = newG;

b = newB;

}

public SimpleColor(Color color)

{

r = color.R;

g = color.G;

b = color.B;

}

}

1. Структура, хранящая информацию для отмены перемещения

public struct UndoInfo

{

public IRenderableObject obj;

public Point delta;

public UndoInfo(IRenderableObject movedObj, Point delta)

{

this.delta = delta;

obj = movedObj;

}

}

1. Интерфейс для взаимодействия с объектами

public interface IRenderableObject

{

bool hidden { set; get; }

void Draw(OpenGL gl, bool forceHidden);

void UpdatePivotPosition();

Point GetPivotPosition();

void Translate(Point delta);

bool CheckIfPointInside(Point p);

void HandleOutOfBounds(Control control);

void ChangeColor(SimpleColor newColor);

}

1. Класс «треугольник».

public class Triangle : IRenderableObject

{

public Point point1, point2, point3;

public Point pivot;

public bool hidden { get; set; }

public Triangle(Point newPoint1, Point newPoint2, Point newPoint3)

{

point1 = newPoint1;

point2 = newPoint2;

point3 = newPoint3;

hidden = false;

pivot = new Point((short)((point1.x + point2.x + point3.x) / 3), (short)((point1.y + point2.y + point3.y) / 3));

}

public void Draw(OpenGL gl, bool forceHidden)

{

if (hidden || forceHidden)

gl.Begin(OpenGL.GL\_LINE\_LOOP); // Без заливки, просто линии

else

gl.Begin(OpenGL.GL\_TRIANGLES); // С заливкой

gl.Color(point1.color.r, point1.color.g, point1.color.b);

gl.Vertex(point1.x, point1.y);

gl.Color(point2.color.r, point2.color.g, point2.color.b);

gl.Vertex(point2.x, point2.y);

gl.Color(point3.color.r, point3.color.g, point3.color.b);

gl.Vertex(point3.x, point3.y);

gl.End();

}

public void UpdatePivotPosition()

{

pivot.x = (short)((point1.x + point2.x + point3.x) / 3);

pivot.y = (short)((point1.y + point2.y + point3.y) / 3);

}

public void Translate(Point delta)

{

point1 += delta;

point2 += delta;

point3 += delta;

}

public Point GetPivotPosition()

{

return pivot;

}

public bool CheckIfPointInside(Point p)

{

double det = ((point2.y - point3.y) \* (point1.x - point3.x) + (point3.x - point2.x) \* (point1.y - point3.y));

double a = ((point2.y - point3.y) \* (p.x - point3.x) + (point3.x - point2.x) \* (p.y - point3.y)) / det;

double b = ((point3.y - point1.y) \* (p.x - point3.x) + (point1.x - point3.x) \* (p.y - point3.y)) / det;

double c = 1 - a - b;

return (a >= 0 && b >= 0 && c >= 0);

}

public void HandleOutOfBounds(Control control)

{

if (pivot.x < 0)

{

point1.x -= pivot.x;

point2.x -= pivot.x;

point3.x -= pivot.x;

}

else if (pivot.x > control.Width)

{

point1.x -= (short)(pivot.x - control.Width);

point2.x -= (short)(pivot.x - control.Width);

point3.x -= (short)(pivot.x - control.Width);

}

if (pivot.y < 0)

{

point1.y -= pivot.y;

point2.y -= pivot.y;

point3.y -= pivot.y;

}

else if (pivot.y > control.Height)

{

point1.y -= (short)(pivot.y - control.Height);

point2.y -= (short)(pivot.y - control.Height);

point3.y -= (short)(pivot.y - control.Height);

}

}

public void ChangeColor(SimpleColor newColor)

{

point1.color = newColor;

point2.color = newColor;

point3.color = newColor;

}

}

1. Класс “группа треугольников”

public class TriangleGroup : IRenderableObject

{

public bool hidden { get; set; }

public bool isBeingEdited = false;

public List<IRenderableObject> tris = new List<IRenderableObject>();

public Point pivot;

public void ChangeColor(SimpleColor newColor)

{

foreach (IRenderableObject tri in tris)

{

tri.ChangeColor(newColor);

}

}

public bool CheckIfPointInside(Point p)

{

bool hit = false;

for (int i = tris.Count - 1; i >= 0 && !hit; i--)

{

hit = tris[i].CheckIfPointInside(p);

}

return hit;

}

public void Draw(OpenGL gl, bool forceHidden)

{

foreach (IRenderableObject tri in tris)

tri.Draw(gl, forceHidden || hidden);

}

public Point GetPivotPosition()

{

return pivot;

}

public void HandleOutOfBounds(Control control)

{

if (pivot.x < 0)

{

foreach(Triangle tri in tris)

{

tri.point1.x -= pivot.x;

tri.point2.x -= pivot.x;

tri.point3.x -= pivot.x;

}

}

else if (pivot.x > control.Width)

{

foreach(Triangle tri in tris)

{

tri.point1.x -= (short)(pivot.x - control.Width);

tri.point2.x -= (short)(pivot.x - control.Width);

tri.point3.x -= (short)(pivot.x - control.Width);

}

}

if (pivot.y < 0)

{

foreach(Triangle tri in tris)

{

tri.point1.y -= pivot.y;

tri.point2.y -= pivot.y;

tri.point3.y -= pivot.y;

}

}

else if (pivot.y > control.Height)

{

foreach(Triangle tri in tris)

{

tri.point1.y -= (short)(pivot.y - control.Height);

tri.point2.y -= (short)(pivot.y - control.Height);

tri.point3.y -= (short)(pivot.y - control.Height);

}

}

}

public void Translate(Point delta)

{

foreach (IRenderableObject tri in tris)

{

tri.Translate(delta);

}

}

public void UpdatePivotPosition()

{

foreach (IRenderableObject tri in tris)

tri.UpdatePivotPosition();

int xSum = tris.Sum(x => x.GetPivotPosition().x);

int ySum = tris.Sum(y => y.GetPivotPosition().y);

pivot.x = (short)(xSum / tris.Count);

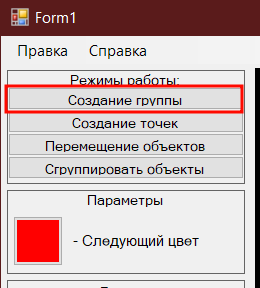
pivot.y = (short)(ySum / tris.Count);

}

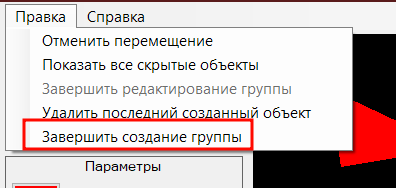
}

# Описание программы

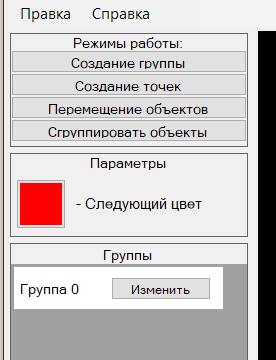
1. Режим “Создание группы”. Выбран по-умолчанию.



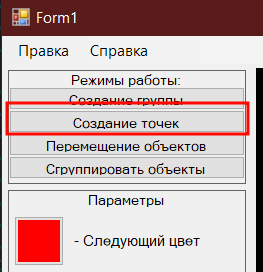
В данном режиме с помощью нажатия левой кнопкой мыши создаются точки. Каждая тройка точек формирует треугольник. При создании 2-х и более треугольников можно завершить формирование группы.



Созданная группа появится в списке.

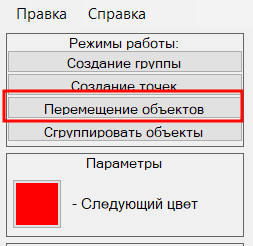


1. Режим “Создание точек”.



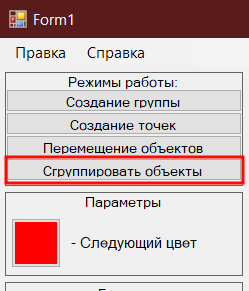
В данном режиме с помощью нажатия левой кнопкой мыши создаются точки. Каждая тройка точек формирует треугольник. При этом создание группы не происходит.

1. Режим “Перемещение объектов”.

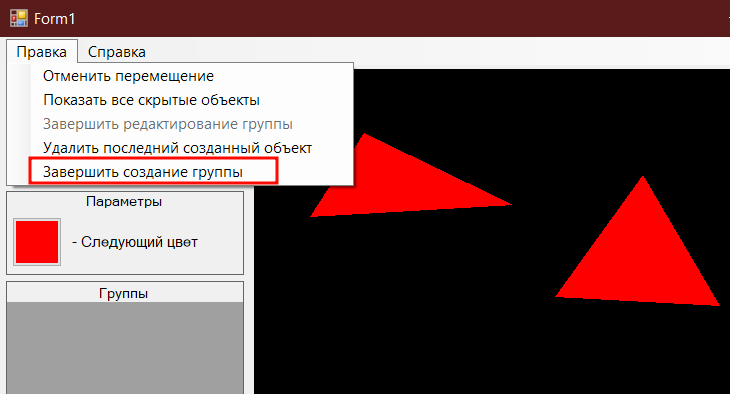


В данном режиме можно перемещать объекты с помощью зажатия левой кнопки мыши на объекте.

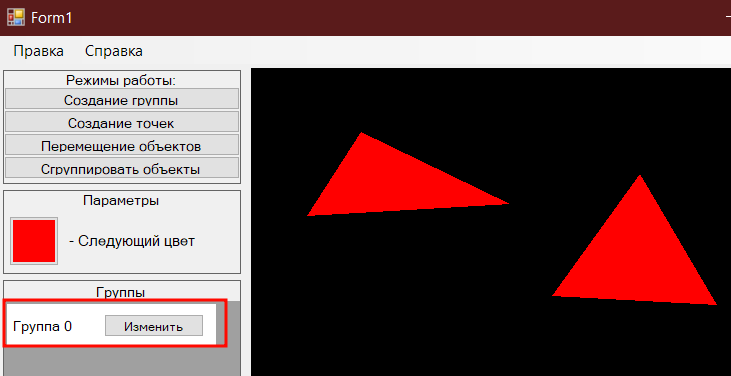
1. Режим “Сгруппировать объекты”.



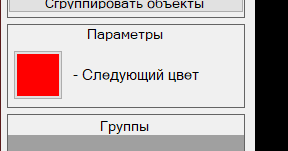
В данном режиме можно группировать уже существующие объекты. Для этого нужно нажать левой кнопкой мыши на нужные объекты и завершить создание группы. При этом группу можно создать только если выбраны 2 и более объектов.



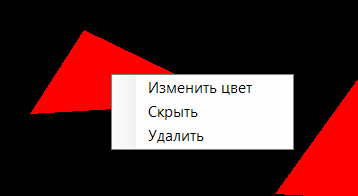
После этого новая группа появится в списке.



1. С помощью параметра “Следующий цвет” можно выбрать какого цвета будет следующая созданная точка.

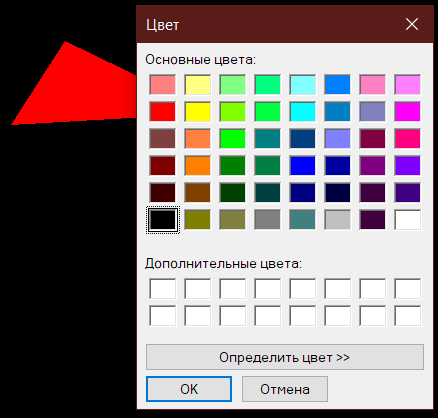


1. Для взаимодействия с конкретным, уже созданным, объектом нужно нажать на него правой кнопкой мыши, после чего откроется контекстное меню.



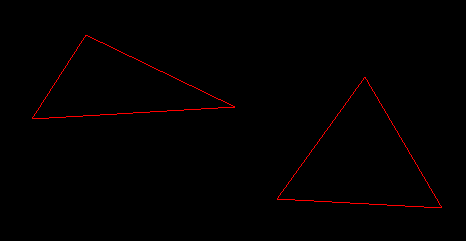
* “Изменить цвет”

Меняет цвет данного объекта в открывшемся меню



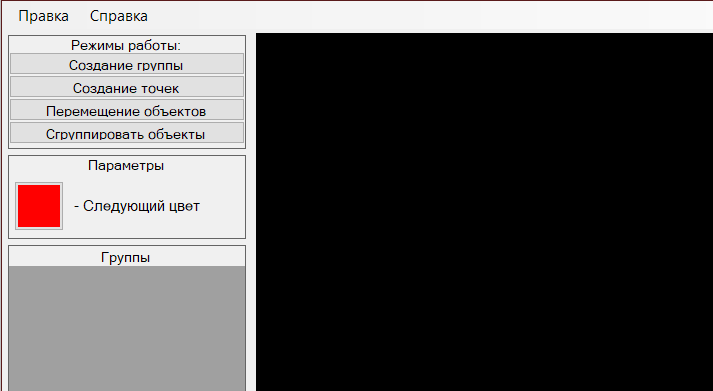
* “Скрыть”

Скрывает объект. Взаимодействовать со скрытыми объектами невозможно, при этом визуально они отображаются в виде каркаса.

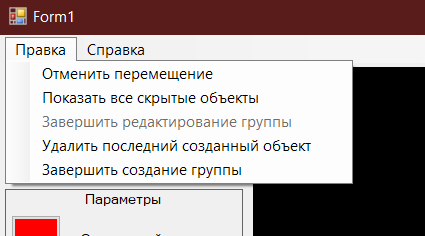


* “Удалить”

Удаляет объект



1. В меню “Правка” представлены следующие команды.



* “Отменить перемещение” (горячая клавиша: Ctrl+Z)

Отменяет последнее перемещение объекта, возвращая его на предыдущую позицию.

* “Показать все скрытые объекты” (горячая клавиша: Alt+H)

Отменяет скрытие у всех объектов

* “Завершить редактирование группы” (горячая клавиша: F)

Позволяет завершить редактирование группы, если пользователь находится в режиме редактирования.

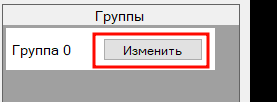
* “Удалить последний созданный объект”

Удаляет последний объект, который был создан. Последним созданным объектом в том числе может являться группа.

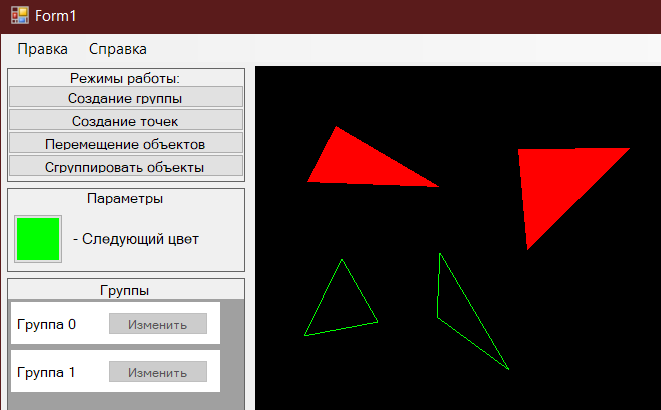
* “Завершить создание группы” (горячая клавиша: Enter)

Позволяет создать группу, если пользователь создал несколько треугольников в режиме “Создание группы” или выбрал несколько объектов в режиме “Сгруппировать объекты”

1. Созданные группы можно редактировать. Для этого в списке групп у соответствующей группы нужно нажать кнопку “Изменить”.



В данном режиме все элементы группы рассматриваются как независимые объекты. Для взаимодействия с ними и самой группой доступны все инструменты (перемещение, создание точек, создание группы и т.д.). При этом объекты, не относящиеся к редактируемой группе, скрываются.



После завершения редактирования все изменения остаются внутри группы, а сама она снова рассматривается как единое целое.

# Тесты программы

|  |  |
| --- | --- |
| Действие | Результат |
| Создание треугольника по точкам без группировки в режиме “Создание точек” |  |
| Создание треугольников по точкам с последующей группировкой в режиме “Создание группы” |  |
| Группировка уже существующих объектов (красный и зеленый треугольники). В режиме “Сгруппировать объекты” выбираются объекты и завершается создание группы |  |
| В режиме “Перемещение объектов” передвижение объектов мышью |  |
| Отмена последнего перемещения объекта |  |
| Скрытие объекта через контекстное меню |  |
| Изменение цвета объекта через контекстное меню |  |
| Удаление выбранного объекта через контекстное меню |  |
| Создание треугольника с вершинами разного цвета путем изменения “Следующего цвета” |  |
| Показать все скрытые объекты |  |
| Удалить последний созданный объект |  |
| Изменение цвета группы |  |
| Редактирование группы. При редактировании, все объекты независимы друг от друга и могут быть изменены непосредственно, как обычные треугольники |  |
| Переполнение списка групп |  |
| Попытка переместить треугольник за экран | Треугольник не двигается дальше точки своей середины |

# Текст программы

using SharpGL;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace Lab1

{

// Режим работы

enum Mode : byte

{

Drawing,

Dragging,

Grouping,

DrawingGroup

}

public struct Point

{

public short x, y;

public SimpleColor color;

public Point(short newX, short newY)

{

x = newX;

y = newY;

color = new SimpleColor();

}

public Point(short newX, short newY, SimpleColor newColor)

{

x = newX;

y = newY;

color = newColor;

}

public static Point operator +(Point a, Point b)

{

Point t;

t.x = (short)(a.x + b.x);

t.y = (short)(a.y + b.y);

t.color = a.color;

return t;

}

}

// В c# и так есть Color, но он слишком комплексный

public struct SimpleColor

{

public byte r, g, b;

public SimpleColor(byte newR, byte newG, byte newB)

{

r = newR;

g = newG;

b = newB;

}

public SimpleColor(Color color)

{

r = color.R;

g = color.G;

b = color.B;

}

}

public struct UndoInfo

{

public IRenderableObject obj;

public Point delta;

public UndoInfo(IRenderableObject movedObj, Point delta)

{

this.delta = delta;

obj = movedObj;

}

}

public interface IRenderableObject

{

bool hidden { set; get; }

void Draw(OpenGL gl, bool forceHidden);

void UpdatePivotPosition();

Point GetPivotPosition();

void Translate(Point delta);

bool CheckIfPointInside(Point p);

void HandleOutOfBounds(Control control);

void ChangeColor(SimpleColor newColor);

void Delete();

}

public partial class Form1 : Form

{

OpenGL gl;

Mode mode = Mode.DrawingGroup;

List<IRenderableObject> objects = new List<IRenderableObject>();

List<IRenderableObject> backupPointer;

List<IRenderableObject> groupingBuffer = new List<IRenderableObject>();

List<UndoInfo> undoBuffer = new List<UndoInfo>();

Point[] pointsBuffer = new Point[3];

Point prevMouseLocation;

Point startMovingLocation;

byte pointsInBuffer = 0;

int selectedObject = -1;

int maxUndoBufferSize = 256;

TriangleGroup editedGroup;

public Form1()

{

InitializeComponent();

}

// Update функция OGL, вызывается каждый кадр

private void openGLControl1\_OpenGLDraw(object sender, RenderEventArgs args)

{

gl.Clear(OpenGL.GL\_COLOR\_BUFFER\_BIT | OpenGL.GL\_DEPTH\_BUFFER\_BIT); // Очистка экрана и буфера глубины

// Цикл отрисовки всех треугольников

if (editedGroup is null)

{

foreach (IRenderableObject tri in objects)

{

tri.Draw(gl, false);

}

}

else

{

foreach (IRenderableObject tri in backupPointer)

{

if (tri != editedGroup)

tri.Draw(gl, true);

}

foreach (IRenderableObject tri in objects)

{

tri.Draw(gl, false);

}

}

// Отрисовка точек из буфера

for (int i = 0; i < pointsInBuffer; i++)

{

gl.PointSize(20);

gl.Enable(OpenGL.GL\_POINT\_SMOOTH);

gl.Begin(OpenGL.GL\_POINTS);

gl.Color(pointsBuffer[i].color.r, pointsBuffer[i].color.g, pointsBuffer[i].color.b);

gl.Vertex(pointsBuffer[i].x, pointsBuffer[i].y);

gl.End();

gl.Disable(OpenGL.GL\_POINT\_SMOOTH);

}

gl.Finish();

}

// Вызывается при создании объекта OGL

private void openGLControl1\_Load(object sender, EventArgs e)

{

gl = openGLControl1.OpenGL;

}

// Ивент нажания мыши

private void openGLControl1\_MouseDown(object sender, MouseEventArgs e)

{

// Если нажата ЛКМ

if (e.Button == MouseButtons.Left)

{

if (mode == Mode.Drawing)

{

// Создаем точку с координатами клика и добавляем в буфер

Point clickPoint = new Point((short)e.Location.X, (short)e.Location.Y, new SimpleColor(colorButton.BackColor.R, colorButton.BackColor.G, colorButton.BackColor.B));

pointsBuffer[pointsInBuffer] = clickPoint;

pointsInBuffer++;

// Если буфер полный создаем треугольник и добавляем его

if (pointsInBuffer > 2)

{

pointsInBuffer = 0;

Triangle newTri = new Triangle(pointsBuffer[0], pointsBuffer[1], pointsBuffer[2]);

objects.Add(newTri);

}

}

else if (mode == Mode.DrawingGroup)

{

// Создаем точку с координатами клика и добавляем в буфер

Point clickPoint = new Point((short)e.Location.X, (short)e.Location.Y, new SimpleColor(colorButton.BackColor.R, colorButton.BackColor.G, colorButton.BackColor.B));

pointsBuffer[pointsInBuffer] = clickPoint;

pointsInBuffer++;

// Если буфер полный создаем треугольник и добавляем его

if (pointsInBuffer > 2)

{

pointsInBuffer = 0;

Triangle newTri = new Triangle(pointsBuffer[0], pointsBuffer[1], pointsBuffer[2]);

objects.Add(newTri);

groupingBuffer.Add(newTri);

}

}

else if (mode == Mode.Dragging)

{

// Создаем точку в месте клика

Point p = new Point((short)e.X, (short)e.Y);

selectedObject = -1;

// Перебор всех треугольников, проверка попали ли внутрь какого-нибудь. Обход с конца, поэтому выбирается самый верхний при нескольких

selectedObject = CheckForObject(p);

// Если нашелся треугольник, запоминаем его и запоминаем координаты клика

if (selectedObject != -1)

{

prevMouseLocation = p;

startMovingLocation = p;

}

}

else if (mode == Mode.Grouping)

{

// Создаем точку в месте клика

Point p = new Point((short)e.X, (short)e.Y);

selectedObject = CheckForObject(p);

if (selectedObject != -1 && !groupingBuffer.Contains(objects[selectedObject]))

groupingBuffer.Add(objects[selectedObject]);

selectedObject = -1;

}

}

else if (e.Button == MouseButtons.Right) // Если нажата ПКМ

{

Point p = new Point((short)e.X, (short)e.Y);

selectedObject = -1;

selectedObject = CheckForObject(p);

// Если попали в треугольник, запоминаем его и открываем в месте клика контекстное меню

if (selectedObject != -1)

contextMenuStrip1.Show(openGLControl1, e.Location);

}

}

private int CheckForObject(Point mousePos)

{

int result = -1;

for (int i = objects.Count - 1; i >= 0 && result == -1; i--)

{

if (objects[i].CheckIfPointInside(mousePos) && !objects[i].hidden)

{

// Если попали в треугольник, возвращаем его

result = i;

}

}

return result;

}

private void openGLControl1\_Resized(object sender, EventArgs e)

{

gl = openGLControl1.OpenGL; // Вызывается раньше "load", поэтому самостоятельно присваиваем

Control control = (Control)sender;

gl.Viewport(0, 0, control.Width, control.Height); // Задаем вьюпорт: координаты левого нижнего угла (0,0 по дефолту) и разрешение

gl.MatrixMode(OpenGL.GL\_PROJECTION); // Задаем с какой матрицей работать

gl.LoadIdentity();

gl.Ortho2D(0, control.Width, control.Height, 0); // Задаем ортографическую проекцию, теперь коодринаты совпадают с пикселями с центром в левом верхнем углу

gl.MatrixMode(OpenGL.GL\_MODELVIEW); // Возвращаю дефолтное значение

}

// Ивент нажатия кнопки

private void openGLControl1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Back)

{

DeleteLastObject();

}

else if (e.KeyCode == Keys.H && e.Modifiers == Keys.Alt)

{

e.SuppressKeyPress = true;

UnhideAllObjects();

}

else if (e.KeyCode == Keys.Enter)

{

FinishMakingGroup();

}

else if (e.KeyCode == Keys.F)

{

FinishEditingGroup();

}

else if (e.KeyCode == Keys.Z && e.Modifiers == Keys.Control)

{

Undo();

}

}

private void DeleteLastObject()

{

// Удаляю последний треугольник, если есть

if (objects.Count > 0)

{

IRenderableObject temp = objects.Last();

objects.RemoveAt(objects.Count - 1);

groupingBuffer.Remove(temp);

temp.Delete();

}

}

private void DeleteObject(int index)

{

IRenderableObject temp = objects[index];

objects.RemoveAt(index);

groupingBuffer.Remove(temp);

temp.Delete();

}

private void ChangeMode(Mode newMode)

{

if (mode != newMode)

{

mode = newMode;

groupingBuffer.Clear();

}

}

private void FinishMakingGroup()

{

if (groupingBuffer.Count > 1)

{

TriangleGroup group = new TriangleGroup();

foreach (IRenderableObject obj in groupingBuffer)

group.tris.Add(obj);

foreach (IRenderableObject obj in groupingBuffer)

objects.Remove(obj);

groupingBuffer.Clear();

objects.Add(group);

TrianglesGroupUC newGroup = new TrianglesGroupUC((TriangleGroup)objects.Last(), $"Группа {groupsContainer.Controls.Count}");

newGroup.onStartEdit += StartEditingGroup;

groupsContainer.Controls.Add(newGroup);

if (editedGroup != null)

newGroup.SwitchButton(false);

}

else

{

// Error here

MessageBox.Show("Для создания группы должно быть выдлено как минимум 2 объекта!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void UnhideAllObjects()

{

// Прохожу по всем треугольникам и убираю скрытие

for (int i = 0; i < objects.Count; i++)

objects[i].hidden = false;

}

private void FinishEditingGroup()

{

if (editedGroup is null)

return;

finishEditingTheGroupToolStripMenuItem.Enabled = false;

objects = backupPointer;

editedGroup.isBeingEdited = false;

editedGroup.UnhideAll();

editedGroup = null;

foreach (TrianglesGroupUC groupPanel in groupsContainer.Controls)

groupPanel.SwitchButton(true);

}

private void StartEditingGroup(TriangleGroup group)

{

foreach (TrianglesGroupUC groupPanel in groupsContainer.Controls)

groupPanel.SwitchButton(false);

finishEditingTheGroupToolStripMenuItem.Enabled = true;

ChangeMode(Mode.Drawing);

backupPointer = objects;

objects = group.tris;

editedGroup = group;

editedGroup.isBeingEdited = true;

editedGroup.hidden = false;

}

private void Undo()

{

if (undoBuffer.Count > 0)

{

UndoInfo undoInfo = undoBuffer.Last();

undoBuffer.RemoveAt(undoBuffer.Count - 1);

undoInfo.obj.Translate(undoInfo.delta);

}

}

// Ивент движения мыши по области OGL

private void openGLControl1\_MouseMove(object sender, MouseEventArgs e)

{

if (selectedObject != -1 && e.Button == MouseButtons.Left && mode == Mode.Dragging)

{

// Дельта перемещения мыши

Point delta = new Point((short)(e.X - prevMouseLocation.x), (short)(e.Y - prevMouseLocation.y));

IRenderableObject curObj = objects[selectedObject];

// Просто добавляю дельту ко всем координатам

curObj.Translate(delta);

curObj.UpdatePivotPosition();

// Проверка выхода за экран

curObj.HandleOutOfBounds(this);

prevMouseLocation.x = (short)e.X;

prevMouseLocation.y = (short)e.Y;

}

}

private void drawButton\_Click(object sender, EventArgs e)

{

ChangeMode(Mode.Drawing);

}

private void dragButton\_Click(object sender, EventArgs e)

{

ChangeMode(Mode.Dragging);

}

private void openGLControl1\_MouseUp(object sender, MouseEventArgs e)

{

if (mode == Mode.Dragging && e.Button == MouseButtons.Left)

{

if (selectedObject != -1)

{

IRenderableObject temp = objects[selectedObject];

Point delta = new Point((short)(startMovingLocation.x - temp.GetPivotPosition().x), (short)(startMovingLocation.y - temp.GetPivotPosition().y));

UndoInfo newUndo = new UndoInfo(temp, delta);

undoBuffer.Add(newUndo);

if (undoBuffer.Count > maxUndoBufferSize)

undoBuffer.RemoveAt(0);

}

selectedObject = -1;

}

}

// Ивент кнопки смены цвета из контекстного меню

private void changeColorToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

objects[selectedObject].ChangeColor(new SimpleColor(colorDialog1.Color));

}

// Ивент кнопки скрыть из контекстного меню

private void hideToolStripMenuItem\_Click(object sender, EventArgs e)

{

objects[selectedObject].hidden = true;

selectedObject = -1;

}

// Ивент кнопки удалить из контекстного меню

private void deleteToolStripMenuItem\_Click(object sender, EventArgs e)

{

DeleteObject(selectedObject);

}

private void colorButton\_Click(object sender, EventArgs e)

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

colorButton.BackColor = colorDialog1.Color;

}

private void groupButton\_Click(object sender, EventArgs e)

{

ChangeMode(Mode.Grouping);

}

private void editToolStripMenuItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Backspace - Удалить последний созданный объект\nAlt+H - Показать все объекты\nF - завершить редактирование группы", "Help", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void groupButton\_MouseHover(object sender, EventArgs e)

{

toolTip1.Show("Выделите несколько групп и нажмите ENTER чтобы сгруппировать их", groupButton);

}

private void unhideAllObjectsToolStripMenuItem\_Click(object sender, EventArgs e)

{

UnhideAllObjects();

}

private void finishEditingTheGroupToolStripMenuItem\_Click(object sender, EventArgs e)

{

FinishEditingGroup();

}

private void deleteTheLastCreatedObjectToolStripMenuItem\_Click(object sender, EventArgs e)

{

DeleteLastObject();

}

private void undoToolStripMenuItem\_Click(object sender, EventArgs e)

{

Undo();

}

private void drawGroupButton\_Click(object sender, EventArgs e)

{

ChangeMode(Mode.DrawingGroup);

}

private void finishMakingGroupToolStripMenuItem\_Click(object sender, EventArgs e)

{

FinishMakingGroup();

}

}

public class Triangle : IRenderableObject

{

public Point point1, point2, point3;

public Point pivot;

public bool hidden { get; set; }

public Triangle(Point newPoint1, Point newPoint2, Point newPoint3)

{

point1 = newPoint1;

point2 = newPoint2;

point3 = newPoint3;

hidden = false;

pivot = new Point((short)((point1.x + point2.x + point3.x) / 3), (short)((point1.y + point2.y + point3.y) / 3));

}

public void Draw(OpenGL gl, bool forceHidden)

{

if (hidden || forceHidden)

gl.Begin(OpenGL.GL\_LINE\_LOOP); // Без заливки, просто линии

else

gl.Begin(OpenGL.GL\_TRIANGLES); // С заливкой

gl.Color(point1.color.r, point1.color.g, point1.color.b);

gl.Vertex(point1.x, point1.y);

gl.Color(point2.color.r, point2.color.g, point2.color.b);

gl.Vertex(point2.x, point2.y);

gl.Color(point3.color.r, point3.color.g, point3.color.b);

gl.Vertex(point3.x, point3.y);

gl.End();

}

public void UpdatePivotPosition()

{

pivot.x = (short)((point1.x + point2.x + point3.x) / 3);

pivot.y = (short)((point1.y + point2.y + point3.y) / 3);

}

public void Translate(Point delta)

{

point1 += delta;

point2 += delta;

point3 += delta;

}

public Point GetPivotPosition()

{

return pivot;

}

public bool CheckIfPointInside(Point p)

{

double det = ((point2.y - point3.y) \* (point1.x - point3.x) + (point3.x - point2.x) \* (point1.y - point3.y));

double a = ((point2.y - point3.y) \* (p.x - point3.x) + (point3.x - point2.x) \* (p.y - point3.y)) / det;

double b = ((point3.y - point1.y) \* (p.x - point3.x) + (point1.x - point3.x) \* (p.y - point3.y)) / det;

double c = 1 - a - b;

return (a >= 0 && b >= 0 && c >= 0);

}

public void HandleOutOfBounds(Control control)

{

if (pivot.x < 0)

{

point1.x -= pivot.x;

point2.x -= pivot.x;

point3.x -= pivot.x;

}

else if (pivot.x > control.Width)

{

point1.x -= (short)(pivot.x - control.Width);

point2.x -= (short)(pivot.x - control.Width);

point3.x -= (short)(pivot.x - control.Width);

}

if (pivot.y < 0)

{

point1.y -= pivot.y;

point2.y -= pivot.y;

point3.y -= pivot.y;

}

else if (pivot.y > control.Height)

{

point1.y -= (short)(pivot.y - control.Height);

point2.y -= (short)(pivot.y - control.Height);

point3.y -= (short)(pivot.y - control.Height);

}

}

public void ChangeColor(SimpleColor newColor)

{

point1.color = newColor;

point2.color = newColor;

point3.color = newColor;

}

public void Delete() { }

}

public class TriangleGroup : IRenderableObject

{

public bool hidden { get; set; }

public bool isBeingEdited = false;

public List<IRenderableObject> tris = new List<IRenderableObject>();

public Point pivot;

public delegate void OnDeleted();

public OnDeleted onDeleted;

public void ChangeColor(SimpleColor newColor)

{

foreach (IRenderableObject tri in tris)

{

tri.ChangeColor(newColor);

}

}

public bool CheckIfPointInside(Point p)

{

bool hit = false;

for (int i = tris.Count - 1; i >= 0 && !hit; i--)

{

hit = tris[i].CheckIfPointInside(p);

}

return hit;

}

public void Draw(OpenGL gl, bool forceHidden)

{

foreach (IRenderableObject tri in tris)

tri.Draw(gl, forceHidden || hidden);

}

public Point GetPivotPosition()

{

return pivot;

}

public void HandleOutOfBounds(Control control)

{

if (pivot.x < 0)

{

foreach(IRenderableObject tri in tris)

{

Point p = new Point((short)(-pivot.x), 0);

tri.Translate(p);

}

}

else if (pivot.x > control.Width)

{

foreach(IRenderableObject tri in tris)

{

Point p = new Point((short)(control.Width - pivot.x), 0);

tri.Translate(p);

}

}

if (pivot.y < 0)

{

foreach(IRenderableObject tri in tris)

{

Point p = new Point(0, (short)(-pivot.y));

tri.Translate(p);

}

}

else if (pivot.y > control.Height)

{

foreach(IRenderableObject tri in tris)

{

Point p = new Point(0, (short)(control.Height - pivot.y));

tri.Translate(p);

}

}

}

public void Translate(Point delta)

{

foreach (IRenderableObject tri in tris)

{

tri.Translate(delta);

}

}

public void UnhideAll()

{

foreach (IRenderableObject obj in tris)

obj.hidden = false;

}

public void UpdatePivotPosition()

{

foreach (IRenderableObject tri in tris)

tri.UpdatePivotPosition();

int xSum = tris.Sum(x => x.GetPivotPosition().x);

int ySum = tris.Sum(y => y.GetPivotPosition().y);

pivot.x = (short)(xSum / tris.Count);

pivot.y = (short)(ySum / tris.Count);

}

public void Delete()

{

onDeleted?.Invoke();

}

}

}