Министерство образования и науки Российской

**Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем

и программной инженерии

**Лабораторная работа № 6**

**по дисциплине**

**«Программирование компьютерной графики»**

**По теме: «Введение в программирование 2D графики»**

Выполнила:

ст. гр. ПРИ-118

Левченко В.П

Принял:

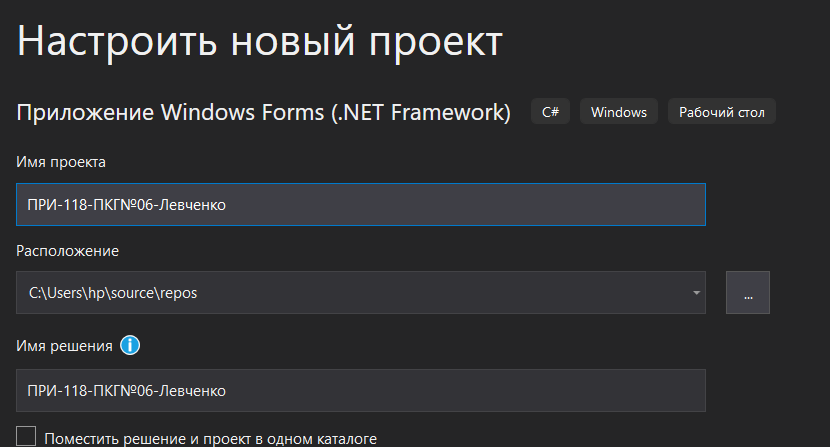
Жигалов И.Е

Владимир, 2021

**Цель работы**

На примере разработки простого растрового редактора, изучить принципы программирования 2D графики в C#.

**Выполнение работы**

  
Рисунок 1. Создание нового проекта

Класс Form1.cs

using System;

using System.Drawing;

using System.Windows.Forms;

using Tao.OpenGl;

using Tao.FreeGlut;

namespace ПРИ118\_ПКГ\_06\_Левченко

{

public partial class Form1 : Form

{

private anEngine ProgrammDrawingEngine;

private int ActiveLayer = 0;

private int LayersCount = 1;

private int AllLayrsCount = 1;

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

private void Form1\_Load(object sender, EventArgs e)

{

LayersControl.Items.Add("Главный слой", true);

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH);

Gl.glClearColor(255, 255, 255, 1);

Gl.glViewport(0, 0, AnT.Width, AnT.Height);

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

Glu.gluOrtho2D(0.0, AnT.Width, 0.0, AnT.Height);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

ProgrammDrawingEngine = new anEngine(AnT.Width, AnT.Height, AnT.Width, AnT.Height);

RenderTimer.Start();

}

private void checkedListBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void RenderTimer\_Tick(object sender, EventArgs e)

{

Drawing();

}

private void Drawing()

{

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

Gl.glLoadIdentity();

Gl.glColor3f(0, 0, 0);

ProgrammDrawingEngine.SwapImage();

Gl.glFlush();

AnT.Invalidate();

}

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

{

if (e.Button == MouseButtons.Left)

ProgrammDrawingEngine.Drawing(e.X, AnT.Height - e.Y);

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

ProgrammDrawingEngine.SetStandartBrush(4);

}

private void toolStripButton2\_Click(object sender, EventArgs e)

{

ProgrammDrawingEngine.SetSpecialBrush(0);

}

private void toolStripButton3\_Click(object sender, EventArgs e)

{

ProgrammDrawingEngine.SetBrushFromFile("brush-1.bmp");

}

private void color1\_Paint(object sender, PaintEventArgs e)

{

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Color tmp = color1.BackColor;

color1.BackColor = color2.BackColor;

color2.BackColor = tmp;

ProgrammDrawingEngine.SetColor(color1.BackColor);

}

private void color1\_MouseClick(object sender, MouseEventArgs e)

{

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

{

color1.BackColor = changeColor.Color;

ProgrammDrawingEngine.SetColor(color1.BackColor);

}

}

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

{

LayersCount++;

ProgrammDrawingEngine.AddLayer();

int AddingLayerNom = LayersControl.Items.Add("Слой" + LayersCount.ToString(), false);

LayersControl.SelectedIndex = AddingLayerNom;

ActiveLayer = AddingLayerNom;

}

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

{

DialogResult res = MessageBox.Show("Будет удален текущий активный слой, действительно продолжить?", "Внимание!", MessageBoxButtons.YesNo, MessageBoxIcon.Warning); // если пользователь нажал кнопку "ДА" в окне подтверждения

if (res == DialogResult.Yes)

{

if (ActiveLayer == 0)

{

MessageBox.Show("Вы не можете удалить нулевой слой.", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Stop);

}

else

{

LayersCount--;

int LayerNomForDel = LayersControl.SelectedIndex;

LayersControl.Items.RemoveAt(LayerNomForDel);

LayersControl.SelectedIndex = 0;

ActiveLayer = 0;

LayersControl.SetItemCheckState(0, CheckState.Checked);

ProgrammDrawingEngine.RemoveLayer(LayerNomForDel);

}

}

}

private void LayersControl\_SelectedValueChanged(object sender, EventArgs e)

{

if (LayersControl.SelectedIndex != ActiveLayer)

{

if (LayersControl.SelectedIndex != -1 && ActiveLayer < LayersControl.Items.Count)

{

LayersControl.SetItemCheckState(ActiveLayer, CheckState.Unchecked);

ActiveLayer = LayersControl.SelectedIndex;

LayersControl.SetItemCheckState(LayersControl.SelectedIndex, CheckState.Checked);

ProgrammDrawingEngine.SetActiveLayerNom(ActiveLayer);

}

}

}

private void toolStripButton5\_Click(object sender, EventArgs e)

{

добавитьСлойToolStripMenuItem\_Click(sender, e);

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

удалитьСлойToolStripMenuItem\_Click(sender, e);

}

private void toolStripButton6\_Click(object sender, EventArgs e)

{

ProgrammDrawingEngine.SetSpecialBrush(1);

}

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

{

toolStripButton1\_Click(sender, e);

}

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

{

toolStripButton3\_Click(sender, e);

}

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

{

toolStripButton6\_Click(sender, e);

}

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

{

DialogResult reslt = MessageBox.Show("В данный момент проект уже начат, сохранить изменения перед закрытием проекта?", "Внимание!", MessageBoxButtons.YesNoCancel);

switch (reslt)

{

case DialogResult.No:

{

ProgrammDrawingEngine = new anEngine(AnT.Width, AnT.Height, AnT.Width, AnT.Height);

LayersControl.Items.Clear();

ActiveLayer = 0;

LayersCount = 1;

AllLayrsCount = 1;

LayersControl.Items.Add("Главный слой", true);

break;

}

case DialogResult.Cancel:

{

return;

}

case DialogResult.Yes:

{

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

{

Bitmap ToSave = ProgrammDrawingEngine.GetFinalImage();

ToSave.Save(saveFileDialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

ProgrammDrawingEngine = new anEngine(AnT.Width, AnT.Height, AnT.Width, AnT.Height);

LayersControl.Items.Clear();

ActiveLayer = 0;

LayersCount = 1;

AllLayrsCount = 1;

LayersControl.Items.Add("Главный слой", true);

}

else

{

return;

}

break;

}

}

}

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

{

DialogResult reslt = MessageBox.Show("В данный момент проект уже начат, сохранить изменения перед закрытием проекта?", "Внимание!", MessageBoxButtons.YesNoCancel);

switch (reslt)

{

case DialogResult.No:

{

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

{

if (System.IO.File.Exists(openFileDialog.FileName))

{

Bitmap ToLoad = new Bitmap(openFileDialog.FileName);

if (ToLoad.Width > AnT.Width || ToLoad.Height > AnT.Height)

{

MessageBox.Show("Извините, но размер изображения превышает размеры области рисования", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

ProgrammDrawingEngine = new anEngine(AnT.Width, AnT.Height, AnT.Width, AnT.Height);

ProgrammDrawingEngine.SetImageToMainLayer(ToLoad);

LayersControl.Items.Clear();

ActiveLayer = 0;

LayersCount = 1;

AllLayrsCount = 1;

LayersControl.Items.Add("Главный слой", true);

}

}

break;

}

case DialogResult.Cancel:

{

return;

}

case DialogResult.Yes:

{

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

{

Bitmap ToSave = ProgrammDrawingEngine.GetFinalImage();

ToSave.Save(saveFileDialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

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

{

if (System.IO.File.Exists(openFileDialog.FileName))

{

Bitmap ToLoad = new Bitmap(openFileDialog.FileName);

if (ToLoad.Width > AnT.Width || ToLoad.Height > AnT.Height)

{

MessageBox.Show("Извините, но размер изображения превышает размеры области рисования", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Stop);

return;

}

ProgrammDrawingEngine = new anEngine(AnT.Width, AnT.Height, AnT.Width, AnT.Height);

ProgrammDrawingEngine.SetImageToMainLayer(ToLoad);

LayersControl.Items.Clear();

ActiveLayer = 0;

LayersCount = 1;

AllLayrsCount = 1;

LayersControl.Items.Add("Главный слой", true);

}

}

break;

}

else

{

return;

}

}

}

}

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

{

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

{

Bitmap ToSave = ProgrammDrawingEngine.GetFinalImage();

ToSave.Save(saveFileDialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

}

}

private void выходToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void toolStripButton7\_Click(object sender, EventArgs e)

{

ProgrammDrawingEngine.SetBrushFromFile("hv-1.bmp");

}

private void toolStripButton8\_Click(object sender, EventArgs e)

{

ProgrammDrawingEngine.SetBrushFromFile("pri-1.bmp");

}

}

}

Класс AnBrush:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.IO;

namespace ПРИ118\_ПКГ\_06\_Левченко

{

class anBrush

{

public Bitmap myBrush;

private bool IsErase = false;

public anBrush(int Value, bool Special)

{

if (!Special)

{

myBrush = new Bitmap(Value, Value);

for (int ax = 0; ax < Value; ax++)

for (int bx = 0; bx < Value; bx++)

myBrush.SetPixel(0, 0, Color.Black);

IsErase = false;

}

else

{

switch (Value)

{

default:

{

myBrush = new Bitmap(5, 5);

for (int ax = 0; ax < 5; ax++)

for (int bx = 0; bx < 5; bx++)

myBrush.SetPixel(ax, bx, Color.Red);

myBrush.SetPixel(0, 2, Color.Black);

myBrush.SetPixel(1, 2, Color.Black);

myBrush.SetPixel(2, 0, Color.Black);

myBrush.SetPixel(2, 1, Color.Black);

myBrush.SetPixel(2, 2, Color.Black);

myBrush.SetPixel(2, 3, Color.Black);

myBrush.SetPixel(2, 4, Color.Black);

myBrush.SetPixel(3, 2, Color.Black);

myBrush.SetPixel(4, 2, Color.Black);

IsErase = false;

break;

}

case 1:

{

myBrush = new Bitmap(5, 5);

for (int ax = 0; ax < Value; ax++)

for (int bx = 0; bx < Value; bx++)

myBrush.SetPixel(0, 0, Color.Black);

IsErase = true;

break;

}

}

}

}

public anBrush(string FromFile)

{

string path = Directory.GetCurrentDirectory();

path += "\\" + FromFile;

myBrush = new Bitmap(path);

}

public bool IsBrushErase()

{

return IsErase;

}

}

}

Class AnEngine:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Collections;

namespace ПРИ118\_ПКГ\_06\_Левченко

{

class anEngine

{

private int picture\_size\_x, picture\_size\_y;

private int scroll\_x, scroll\_y;

private int screen\_width, screen\_height;

private int ActiveLayerNom;

private ArrayList Layers = new ArrayList();

private anBrush standartBrush;

private Color LastColorInUse;

public anEngine(int size\_x, int size\_y, int screen\_w, int screen\_h)

{

standartBrush = new anBrush(3, false);

picture\_size\_x = size\_x;

picture\_size\_y = size\_y;

screen\_width = screen\_w;

screen\_height = screen\_h;

scroll\_x = 0;

scroll\_y = 0;

Layers.Add(new anLayer(picture\_size\_x, picture\_size\_y));

ActiveLayerNom = 0;

standartBrush = new anBrush(1, false);

}

public void SetActiveLayerNom(int nom)

{

((anLayer)Layers[ActiveLayerNom]).CreateNewList();

((anLayer)Layers[nom]).SetColor(((anLayer)Layers[ActiveLayerNom]).GetColor());

ActiveLayerNom = nom;

}

public void SetWisibilityLayerNom(int nom, bool visible)

{

}

public void Drawing(int x, int y)

{

((anLayer)Layers[ActiveLayerNom]).Draw(standartBrush, x, y);

}

public void SwapImage()

{

for (int ax = 0; ax < Layers.Count; ax++)

{

if (ax == ActiveLayerNom)

{

((anLayer)Layers[ax]).RenderImage(false);

}

else

{

((anLayer)Layers[ax]).RenderImage(true);

}

}

}

public void SetStandartBrush(int SizeB)

{

standartBrush = new anBrush(SizeB, false);

}

public void SetSpecialBrush(int Nom)

{

standartBrush = new anBrush(Nom, true);

}

public void SetBrushFromFile(string FileName)

{

standartBrush = new anBrush(FileName);

}

public void SetColor(Color NewColor)

{

((anLayer)Layers[ActiveLayerNom]).SetColor(NewColor);

LastColorInUse = NewColor;

}

public void AddLayer()

{

int AddingLayer = Layers.Add(new anLayer(picture\_size\_x, picture\_size\_y));

SetActiveLayerNom(AddingLayer);

}

public void RemoveLayer(int nom)

{

if (nom < Layers.Count && nom >= 0)

{

Layers.RemoveAt(nom);

SetActiveLayerNom(0);

}

}

public Bitmap GetFinalImage()

{

Bitmap resaultBitmap = new Bitmap(picture\_size\_x, picture\_size\_y);

for (int ax = 0; ax < Layers.Count; ax++)

{

int[,,] tmp\_layer\_data = ((anLayer)Layers[ax]).GetDrawingPlace();

for (int a = 0; a < picture\_size\_x; a++)

{

for (int b = 0; b < picture\_size\_y; b++)

{

if (tmp\_layer\_data[a, b, 3] != 1)

{

resaultBitmap.SetPixel(a, b, Color.FromArgb(tmp\_layer\_data[a, b, 0], tmp\_layer\_data[a, b, 1], tmp\_layer\_data[a, b, 2]));

}

else

{

if (ax == 0)

{

resaultBitmap.SetPixel(a, b, Color.FromArgb(255, 255, 255));

}

}

}

}

}

resaultBitmap.RotateFlip(RotateFlipType.Rotate180FlipX);

return resaultBitmap;

}

public void SetImageToMainLayer(Bitmap layer)

{

layer.RotateFlip(RotateFlipType.Rotate180FlipX);

for (int ax = 0; ax < layer.Width; ax++)

{

for (int bx = 0; bx < layer.Height; bx++)

{

SetColor(layer.GetPixel(ax, bx));

Drawing(ax, bx);

}

}

}

}

}

Class AnLayer:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using Tao.OpenGl;

namespace ПРИ118\_ПКГ\_06\_Левченко

{

class anLayer

{

public int Width, Heigth;

private int[,,] DrawPlace;

private bool isVisible;

private Color ActiveColor;

private int ListNom;

public int[,,] GetDrawingPlace()

{

return DrawPlace;

}

public anLayer(int s\_W, int s\_H)

{

Width = s\_W;

Heigth = s\_H;

DrawPlace = new int[Width, Heigth, 4];

for (int ax = 0; ax < Width; ax++)

{

for (int bx = 0; bx < Heigth; bx++)

{

DrawPlace[ax, bx, 3] = 1;

}

}

isVisible = true;

ActiveColor = Color.Black;

}

public void SetVisibility(bool visiblityState)

{

isVisible = visiblityState;

}

public bool GetVisibility()

{

return isVisible;

}

public void Draw(anBrush BR, int x, int y)

{

int real\_pos\_draw\_start\_x = x - BR.myBrush.Width / 2;

int real\_pos\_draw\_start\_y = y - BR.myBrush.Height / 2;

if (real\_pos\_draw\_start\_x < 0)

real\_pos\_draw\_start\_x = 0;

if (real\_pos\_draw\_start\_y < 0)

real\_pos\_draw\_start\_y = 0;

int boundary\_x = real\_pos\_draw\_start\_x + BR.myBrush.Width;

int boundary\_y = real\_pos\_draw\_start\_y + BR.myBrush.Height;

if (boundary\_x > Width)

boundary\_x = Width;

if (boundary\_y > Heigth)

boundary\_y = Heigth;

int count\_x = 0, count\_y = 0;

for (int ax = real\_pos\_draw\_start\_x; ax < boundary\_x; ax++, count\_x++)

{

count\_y = 0;

for (int bx = real\_pos\_draw\_start\_y; bx < boundary\_y; bx++, count\_y++)

{

if (BR.IsBrushErase())

{

Color ret = BR.myBrush.GetPixel(count\_x, count\_y);

if (!(ret.R == 255 && ret.G == 0 && ret.B == 0))

{

DrawPlace[ax, bx, 3] = 1;

}

}

else

{

Color ret = BR.myBrush.GetPixel(count\_x, count\_y);

if (!(ret.R == 255 && ret.G == 0 && ret.B == 0))

{

DrawPlace[ax, bx, 0] = ActiveColor.R;

DrawPlace[ax, bx, 1] = ActiveColor.G;

DrawPlace[ax, bx, 2] = ActiveColor.B;

DrawPlace[ax, bx, 3] = 0;

}

}

}

}

}

public void RenderImage()

{

Gl.glBegin(Gl.GL\_POINTS);

for (int ax = 0; ax < Width; ax++)

{

for (int bx = 0; bx < Heigth; bx++)

{

if (DrawPlace[ax, bx, 3] != 1)

{

Gl.glColor3f(DrawPlace[ax, bx, 0], DrawPlace[ax, bx, 1], DrawPlace[ax, bx, 2]);

Gl.glVertex2i(ax, bx);

}

}

}

Gl.glEnd();

}

public void SetColor(Color NewColor)

{

ActiveColor = NewColor;

}

public Color GetColor()

{

return ActiveColor;

}

public void ClearList()

{

if (Gl.glIsList(ListNom) == Gl.GL\_TRUE)

{

Gl.glDeleteLists(ListNom, 1);

}

}

public void CreateNewList()

{

if (Gl.glIsList(ListNom) == Gl.GL\_TRUE)

{

Gl.glDeleteLists(ListNom, 1);

ListNom = Gl.glGenLists(1);

}

Gl.glNewList(ListNom, Gl.GL\_COMPILE);

RenderImage(false);

Gl.glEndList();

}

public void RenderImage(bool FromList)

{

if (FromList)

{

Gl.glCallList(ListNom);

}

else

{

int count = 0;

for (int ax = 0; ax < Width; ax++)

{

for (int bx = 0; bx < Heigth; bx++)

{

if (DrawPlace[ax, bx, 3] != 1)

{

count++;

}

}

}

int[] arr\_date\_vertex = new int[count \* 2];

float[] arr\_date\_colors = new float[count \* 3];

int now\_element = 0;

for (int ax = 0; ax < Width; ax++)

{

for (int bx = 0; bx < Heigth; bx++)

{

if (DrawPlace[ax, bx, 3] != 1)

{

arr\_date\_vertex[now\_element \* 2] = ax;

arr\_date\_vertex[now\_element \* 2 + 1] = bx;

arr\_date\_colors[now\_element \* 3] = (float)DrawPlace[ax, bx, 0] / 255.0f;

arr\_date\_colors[now\_element \* 3 + 1] = (float)DrawPlace[ax, bx, 1] / 255.0f;

arr\_date\_colors[now\_element \* 3 + 2] = (float)DrawPlace[ax, bx, 2] / 255.0f;

now\_element++;

}

}

}

Gl.glEnableClientState(Gl.GL\_VERTEX\_ARRAY);

Gl.glEnableClientState(Gl.GL\_COLOR\_ARRAY);

Gl.glColorPointer(3, Gl.GL\_FLOAT, 0, arr\_date\_colors);

Gl.glVertexPointer(2, Gl.GL\_INT, 0, arr\_date\_vertex);

Gl.glDrawArrays(Gl.GL\_POINTS, 0, count);

Gl.glDisableClientState(Gl.GL\_VERTEX\_ARRAY);

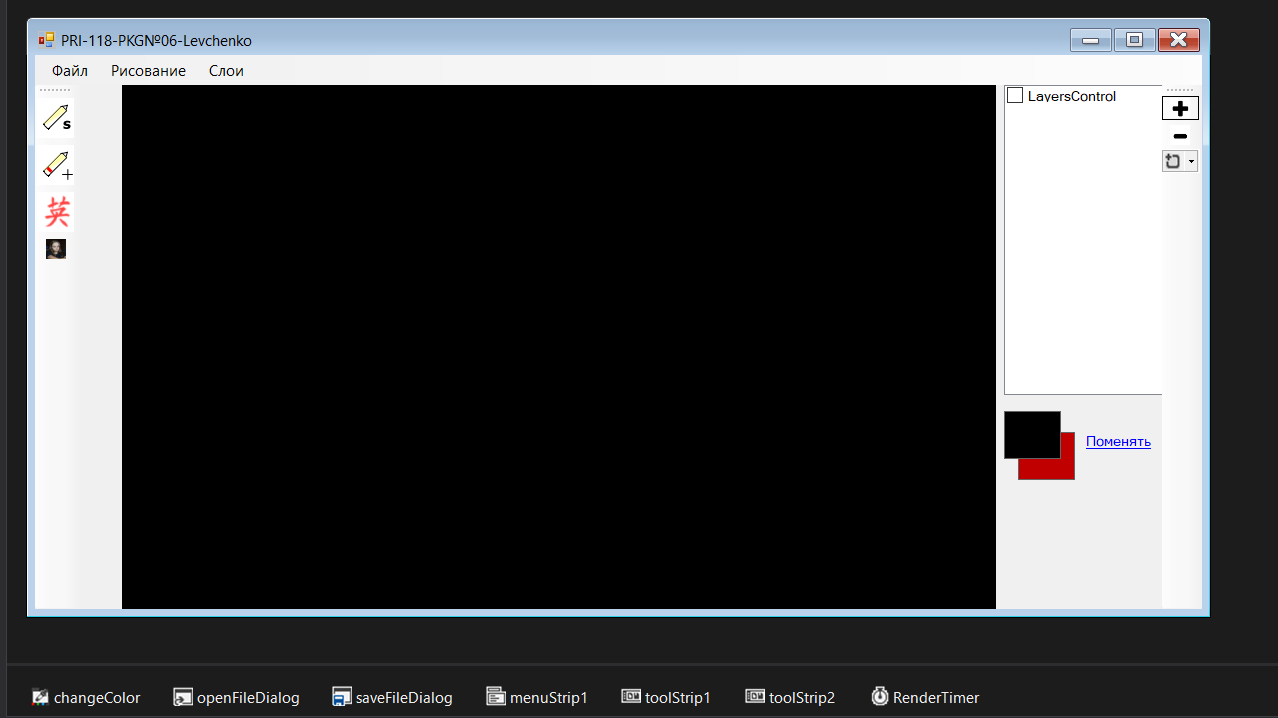
Gl.glDisableClientState(Gl.GL\_COLOR\_ARRAY);

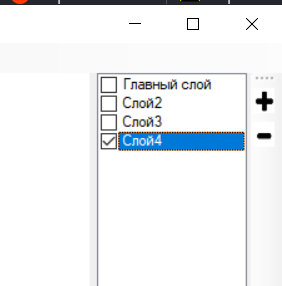
}

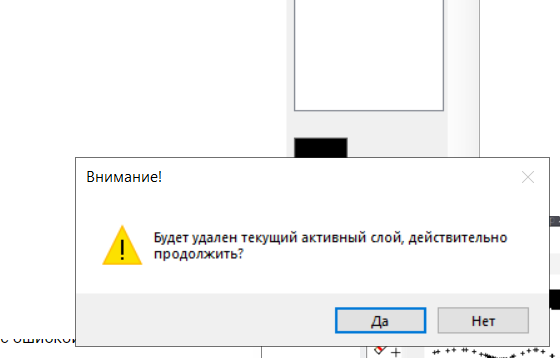
}

}

}

  
Рисунок 2. Вид приложения в конструкторе

  
Рисунок 3. Результат создания слоёв

  
Рисунок 4. Предупреждение при удалении

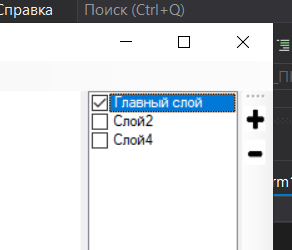
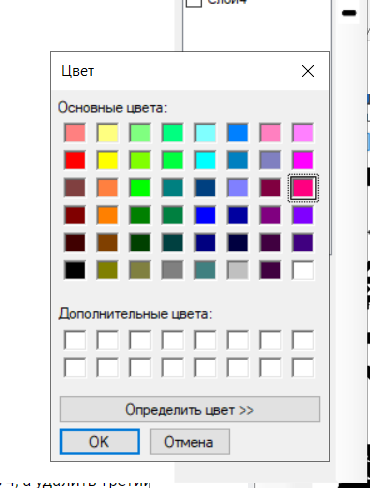
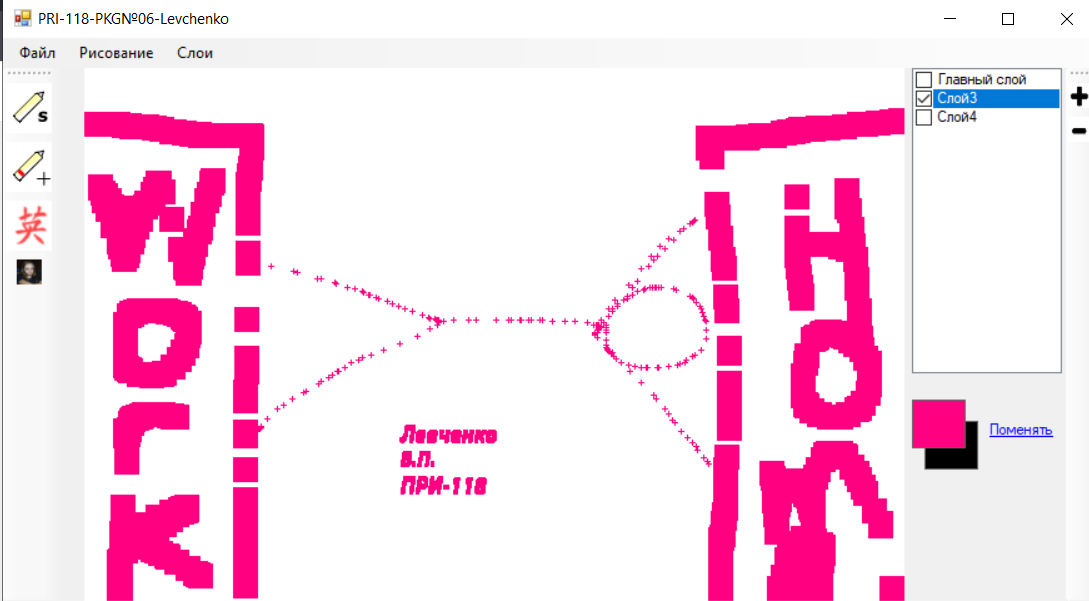
  
Рисунок 5. Результат удаления слоя 3

  
Рисунок 6. Результат работы

  
Рисунок 7. Выбор цветов

  
Рисунок 8. Проверка изменения цвета

**Вывод**

На примере разработки простого растрового редактора, изучен принцип программирования 2D графики в C#.