Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Уфимский государственный авиационный технический университет»

Факультет информатики и робототехники

Кафедра Вычислительной математики и кибернетики

**Лабораторная работа №6**

По дисциплине «Объектно-ориентированное программирование»

«Визуальный редактор»

Выполнил: студент группы ПРО-222

Вальшин Руслан Ильдусович

УФА 2020

1. **Цель работы:**

На основе Л.Р.4 реализовать простейший визуальный редактор векторных объектов (круг, квадрат, треугольник, отрезок, и т.д.)

1. **Задачи:**

* Создание графических объектов на экране:
  + меню, панель инструментов с доступными классами объектов
  + добавление нового объекта в рабочую область
* Манипуляции объектами в рабочей области:
  + выбор объекта для манипулирования (выделяется цветом или рамкой)
  + изменение цвета, размера, положения
  + удаление объекта из рабочей области
  + контроль выхода за рабочую область (при передвижении объект не должен выходить за границы ни одной своей частью)
* Программные требования:
  + манипуляции объектами выполняются с помощью клавиатуры, при этом выбор клавиш управления должен быть как можно ближе к используемым обычно для таких задач в среде Windows
  + «хорошая» иерархия классов
  + использование своего хранилища объектов из Л.Р.3
  + отделение взаимодействия с пользователем от логики работы классов

1. **Ход работы:**
2. Копирование класса Storage из предыдущей лабораторной работы.
3. Описание класса GraphicObject и его наследников.

public abstract class GraphicObject

{

public int x;

public int y;

public int width;

public int height;

public bool isSelected = false;

public Color color;

public int thickness = 5;

public GraphicObject(int x, int y, int width, int height, Color color)

{

this.x = x;

this.y = y;

this.width = width;

this.height = height;

this.color = color;

}

public abstract string className();

public abstract bool belongsTo(int x, int y);

}

public class Line : GraphicObject // В Line width и height становятся x2 и y2

{

public Line(int x, int y, int width, int height, Color color)

: base(x, y, width, height, color) { }

public override string className()

{

return "Line";

}

public override bool belongsTo(int x, int y)

{

double tgA = (double)(height) / (width) ;

int minX = Math.Min(this.x, this.x + width);

int maxX = Math.Max(this.x, this.x + width);

int minY = Math.Min(this.y, this.y + height);

int maxY = Math.Max(this.y, this.y + height);

double d = Math.Abs(tgA \* (this.x - x) + y - this.y) / Math.Sqrt(tgA \* tgA + 1);

if (d <= thickness && minX <= x && x <= maxX && minY <= y && y <= maxY)

return true;

return false;

}

}

public class Rectangle : GraphicObject

{

public Rectangle(int x, int y, int width, int height, Color color)

: base(x, y, width, height, color) { }

public override string className()

{

return "Rectangle";

}

public override bool belongsTo(int x, int y)

{

if (this.x - thickness <= x && x <= this.x + width + thickness)

if (this.y - thickness <= y && y <= this.y + height + thickness)

{

int result = 0;

if (Math.Abs(x - this.x) <= thickness)

result++;

if (Math.Abs(x - this.x - width) <= thickness)

result++;

if (Math.Abs(y - this.y) <= thickness)

result++;

if (Math.Abs(y - this.y - height) <= thickness)

result++;

if (result == 1)

return true;

}

return false;

}

}

public class Ellipse : GraphicObject

{

public Ellipse(int x, int y, int width, int height, Color color)

: base(x, y, width, height, color) { }

public override string className()

{

return "Ellipse";

}

public override bool belongsTo(int x, int y)

{

int a = width / 2;

int b = height / 2;

int centerX = this.x + a;

int centerY = this.y + b;

int lowR = (a - 3) \* (b - 3);

int highR = (a + 3) \* (b + 3);

int temp = b \* b \* (x - centerX) \* (x - centerX) + a \* a \* (y - centerY) \* (y - centerY);

return lowR \* lowR <= temp && temp <= highR \* highR;

}

}

1. Описание класса Painter, который знает, как должна выглядеть фигура на экране.

public class Painter

{

private Graphics g;

private Color selectedColor;

public Painter(Graphics g, Color selectedColor)

{

this.g = g;

this.selectedColor = selectedColor;

}

public void drawObject(GraphicObject obj)

{

Pen pen;

if (obj.isSelected)

{

pen = new Pen(selectedColor, obj.thickness);

pen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;

}

else

pen = new Pen(obj.color, obj.thickness);

string className = obj.className();

switch(className)

{

case "Line":

g.DrawLine(pen, obj.x, obj.y, obj.x + obj.width, obj.y + obj.height);

break;

case "Rectangle":

{

int x1 = obj.width + obj.x;

int y1 = obj.height + obj.y;

System.Drawing.Rectangle rect = new System.Drawing.Rectangle(

Math.Min(obj.x, x1),

Math.Min(obj.y, y1),

Math.Abs(obj.width),

Math.Abs(obj.height));

g.DrawRectangle(pen, rect);

}

break;

case "Ellipse":

g.DrawEllipse(pen, obj.x, obj.y, obj.width, obj.height);

break;

}

}

}

1. Передача model данных графического интерфейса через события.

Model model;

public MainForm()

{

InitializeComponent();

this.DoubleBuffered = true;

SetStyle(ControlStyles.AllPaintingInWmPaint | ControlStyles.UserPaint, true);

SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

model = new Model();

}

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

{

if (e.Button == MouseButtons.Left)

{

model.mouseIsPressed = true;

model.doTheRightThing(e.X, e.Y);

pictureBox.Image = model.image;

}

}

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

{

if (e.Button == MouseButtons.Left)

if (model.creatingObject)

{

model.drawOnline(e.X, e.Y);

pictureBox.Image = model.image;

}

}

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

{

if (e.Button == MouseButtons.Left)

if (model.creatingObject == true)

{

model.mouseIsPressed = false;

model.creatingObject = false;

model.add();

pictureBox.Image = model.image;

}

}

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

{

model.setMode(0);

}

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

{

model.setMode(1);

}

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

{

model.setMode(2);

}

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

{

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

{

colorLabel.BackColor = colorDialog.Color;

model.setColor(colorDialog.Color);

}

}

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

{

switch(e.KeyCode)

{

case Keys.ControlKey:

model.ctrlIsPressed = true;

break;

case Keys.ShiftKey:

model.altIsPressed = true;

break;

case Keys.W:

model.wIsPressed = true;

model.move();

break;

case Keys.A:

model.aIsPressed = true;

model.move();

break;

case Keys.S:

model.sIsPressed = true;

model.move();

break;

case Keys.D:

model.dIsPressed = true;

model.move();

break;

case Keys.Delete:

model.delete();

break;

}

pictureBox.Image = model.image;

}

private void MainForm\_KeyUp(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.ControlKey:

model.ctrlIsPressed = false;

break;

case Keys.ShiftKey:

model.altIsPressed = false;

model.correct();

break;

case Keys.W:

model.wIsPressed = false;

break;

case Keys.A:

model.aIsPressed = false;

break;

case Keys.S:

model.sIsPressed = false;

break;

case Keys.D:

model.dIsPressed = false;

break;

}

model.velocity = 5;

}

1. Описание класса-модели, который отвечает за логику работы

public class Model

{

public Graphics g;

public Bitmap image;

private Painter painter;

private Storage<GraphicObject> storage;

private Storage<GraphicObject> selectedStorage;

private Storage<Color> originColorStorage;

private int creationMode;

private Color color = Color.Black;

public bool ctrlIsPressed;

public bool altIsPressed;

public bool wIsPressed;

public bool aIsPressed;

public bool sIsPressed;

public bool dIsPressed;

public bool mouseIsPressed;

public bool creatingObject;

public int velocity;

public Model()

{

storage = new Storage<GraphicObject>();

selectedStorage = new Storage<GraphicObject>();

originColorStorage = new Storage<Color>();

image = new Bitmap(1920, 1080);

g = Graphics.FromImage(image);

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

painter = new Painter(g, Color.Red);

velocity = 5;

}

public void setMode(int mode)

{

creationMode = mode;

}

public void setColor(Color color)

{

this.color = color;

if(selectedStorage.getSize()!=0)

{

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

selectedStorage.getCurrent().color = color;

}

}

public GraphicObject temp;

public void doTheRightThing(int x, int y)

{

if(check(x,y)) // Если попали по элементу, выбор

{

redrawAll();

}

else // Если попали по пустому месту, создание

{

deselectAll();

creatingObject = true;

if (creationMode == 0)

temp = new Line(x, y, 0, 0, color);

else if (creationMode == 1)

temp = new Rectangle(x, y, 0, 0, color);

else if (creationMode == 2)

temp = new Ellipse(x, y, 0, 0, color);

}

}

public void drawOnline(int x, int y)

{

if (x < 0)

x = 0;

if (y < 0)

y = 0;

g.Clear(Color.White);

temp.width = x - temp.x;

temp.height = y - temp.y;

redrawAll();

painter.drawObject(temp);

}

public void redrawAll()

{

g.Clear(Color.White);

storage.setFirst();

for (int i = 0; i < storage.getSize(); i++, storage.next())

painter.drawObject(storage.getCurrent());

}

private bool check(int x,int y)

{

bool found = false;

storage.setFirst();

for (int i = 0; i < storage.getSize(); i++, storage.next())

if (storage.getCurrent().belongsTo(x, y))

{

temp = storage.getCurrent();

if (ctrlIsPressed)

{

if (temp.isSelected)

{

temp.isSelected = false;

selectedStorage.checkAndSetCurrent(temp);

selectedStorage.del();

}

else

{

temp.isSelected = true;

selectedStorage.add(temp);

}

}

else

{

deselectAll();

if (temp.isSelected == false)

{

selectedStorage.add(storage.getCurrent());

temp.isSelected = true;

}

}

found = true;

}

return found;

}

private void deselectAll() // Убираем все элементы из списка выбранных

{

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

selectedStorage.getCurrent().isSelected = false;

selectedStorage = new Storage<GraphicObject>();

}

public void add()

{

if (Math.Abs(temp.width) < 10 && Math.Abs(temp.height) < 10)

{

redrawAll();

return;

}

if (temp.className() != "Line")

if (temp.width < 0 || temp.height < 0)

{

temp.x = Math.Min(temp.x, temp.x + temp.width);

temp.y = Math.Min(temp.y, temp.y + temp.height);

temp.width = Math.Abs(temp.width);

temp.height = Math.Abs(temp.height);

}

storage.add(temp);

}

public void correct()

{

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

{

temp = selectedStorage.getCurrent();

if (temp.className() != "Line")

if (temp.width < 0 || temp.height < 0)

{

temp.x = Math.Min(temp.x, temp.x + temp.width);

temp.y = Math.Min(temp.y, temp.y + temp.height);

temp.width = Math.Abs(temp.width);

temp.height = Math.Abs(temp.height);

}

}

}

public void move()

{

if (altIsPressed)

{

changeSize();

return ;

}

if (selectedStorage.getSize() == 0)

return ;

int x = 0;

int y = 0;

velocity++;

if (wIsPressed)

y--;

if (aIsPressed)

x--;

if (sIsPressed)

y++;

if (dIsPressed)

x++;

x \*= velocity / 5;

y \*= velocity / 5;

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

{

temp = selectedStorage.getCurrent();

if (temp.x + x >= 0 && temp.x + temp.width + x >= 0)

temp.x += x;

if (temp.y + y >= 0 && temp.y + temp.height + y >= 0)

temp.y += y;

}

redrawAll();

}

public void changeSize()

{

if (selectedStorage.getSize() == 0)

return;

velocity++;

int currentVelocity = velocity / 5;

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

{

temp = selectedStorage.getCurrent();

if (wIsPressed)

if (temp.y - currentVelocity >= 0 && temp.y + temp.height - currentVelocity >= 0)

{

temp.height -= currentVelocity;

}

if (aIsPressed)

if (temp.x - currentVelocity >= 0 && temp.x + temp.width - currentVelocity >= 0)

{

temp.width -= currentVelocity;

}

if (sIsPressed)

if (temp.y + currentVelocity >= 0 && temp.y + temp.height + currentVelocity >= 0)

temp.height += currentVelocity;

if (dIsPressed)

if (temp.x + currentVelocity >= 0 && temp.x + temp.width + currentVelocity >= 0)

temp.width += currentVelocity;

}

redrawAll();

}

public void delete()

{

storage.setFirst();

for (int i = 0; i < storage.getSize(); i++)

if (storage.getCurrent().isSelected)

{

storage.del();

i--;

}

else

storage.next();

selectedStorage = new Storage<GraphicObject>();

redrawAll();

}

}

1. **Описание интерфейса программы:**
2. Создание объекта:

* На панели справа выбирает желаемый объект.
* Зажав ЛКМ на полотне и потянув в любую сторону, задаются параметры объекта.

1. Выбор объекта:

* При нажатии на созданный объект он выделяется пунктиром красным цветов.
* Если зажать клавишу ctrl, можно выделить несколько объектов.

1. Изменение цвета объекта:

* При нажатии на кнопку «Выбор цвета» открывается диалоговое окно выбора цвета.
* Выбранный цвет отображается рядом с кнопкой.
* Если перед изменением цвета был выбран объект, цвет объекта изменится (после снятия выбора).

1. Изменение положения:

* Клавишами WASD можно двигать объект или группу объектов по полотну.

1. Изменение размера:

* Зажимая клавишу Shift и кнопки WASD, можно менять размер объекта или группы объектов.

1. Удаление объекта:

* Нажатием кнопки Delete удаляются все выбранные объекты.

Приложение А. Ссылка на репозиторий и скриншот истории коммитов

<https://github.com/Ruslan376M/OOP_Lab6>

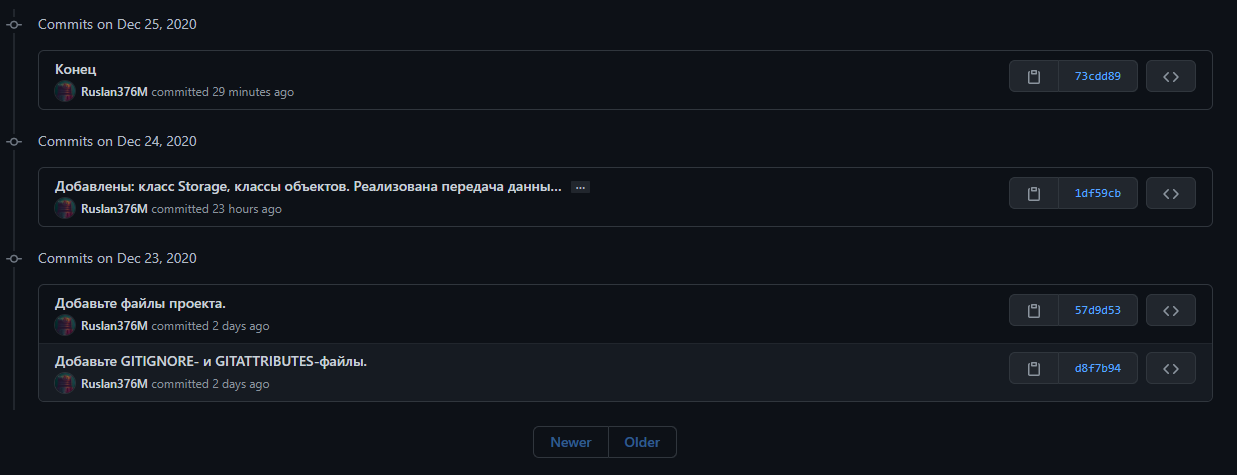


Рисунок 1 - Скриншот истории коммитов

Приложение B. Примеры работы программы.

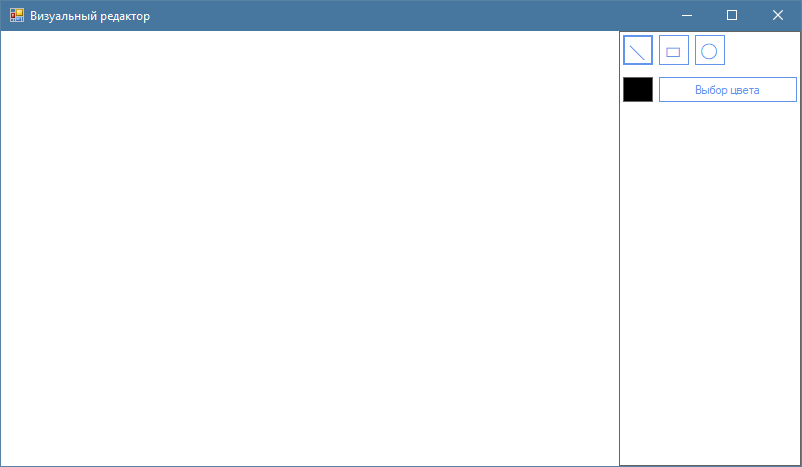


Рисунок 2 - Пример работы программы

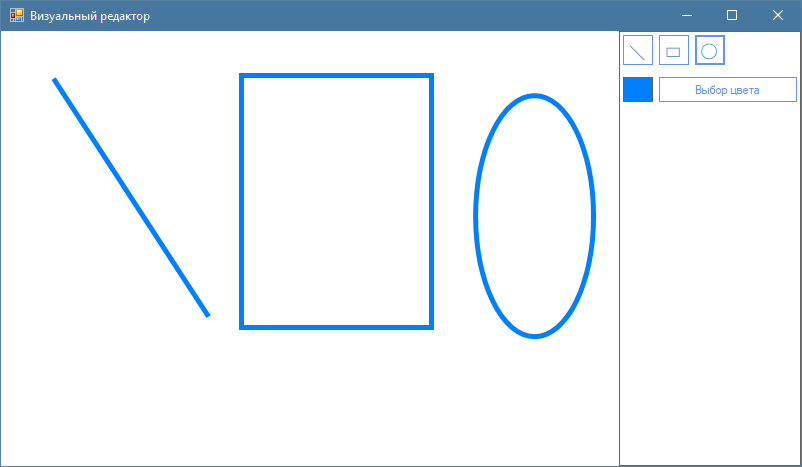


Рисунок 3 - Пример работы программы

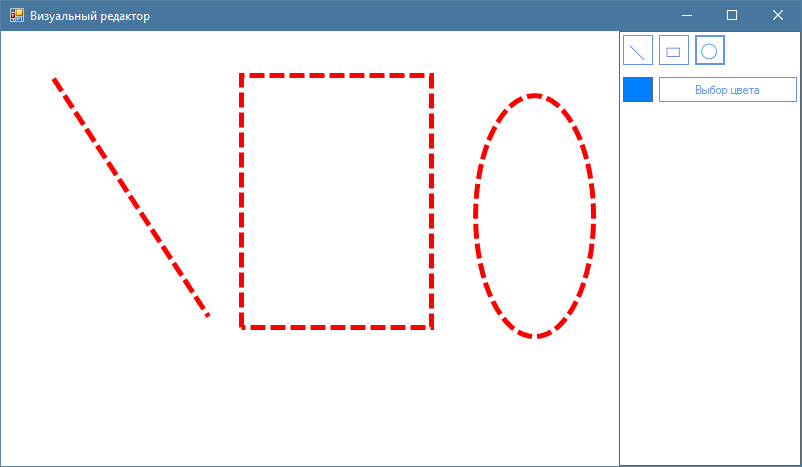


Рисунок 4 - Пример работы программы

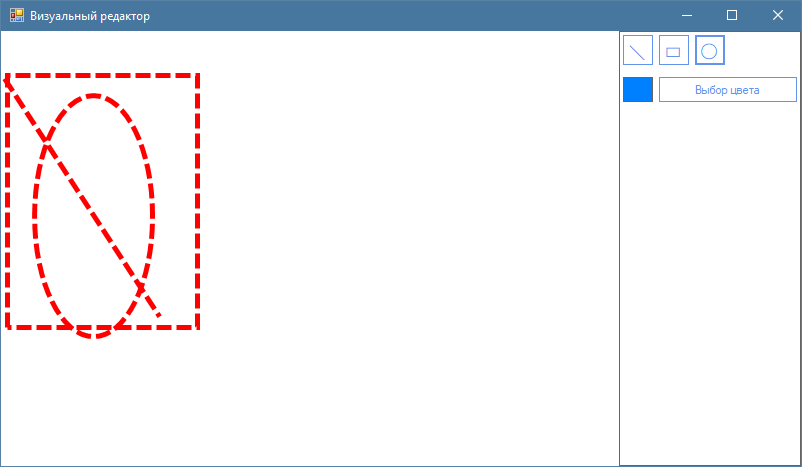


Рисунок 5 - Пример работы программы

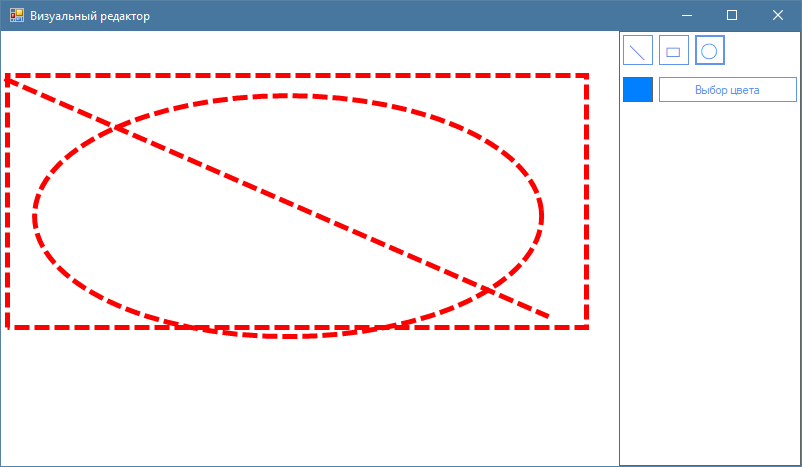


Рисунок 6 - Пример работы программы

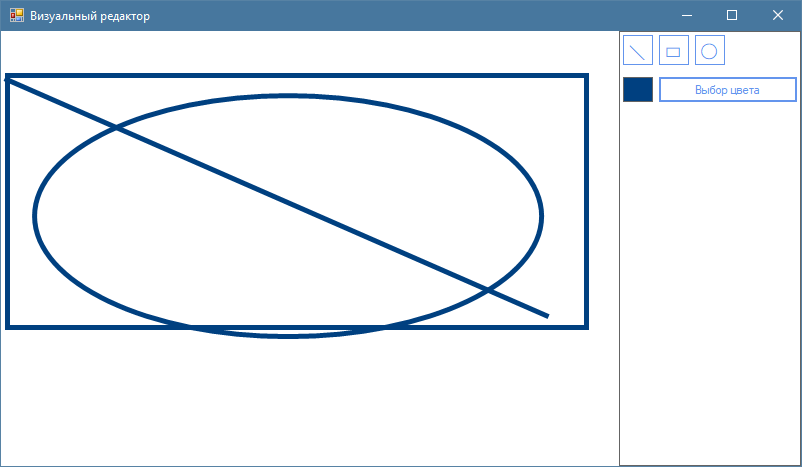


Рисунок 7 - Пример работы программы

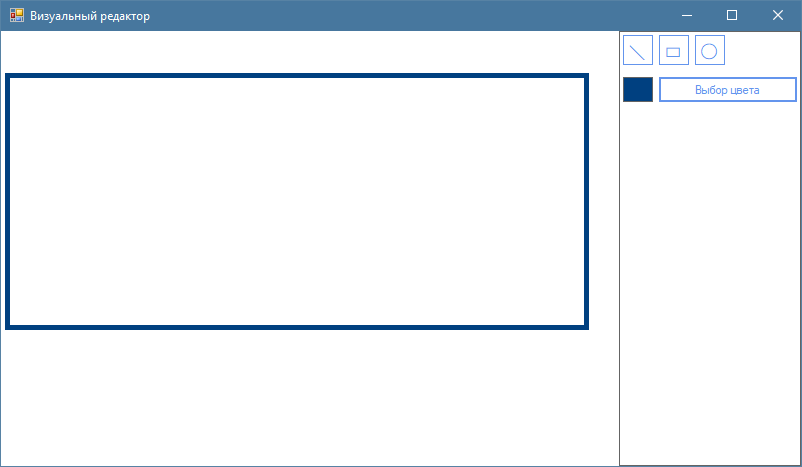


Рисунок 8 - Пример работы программы

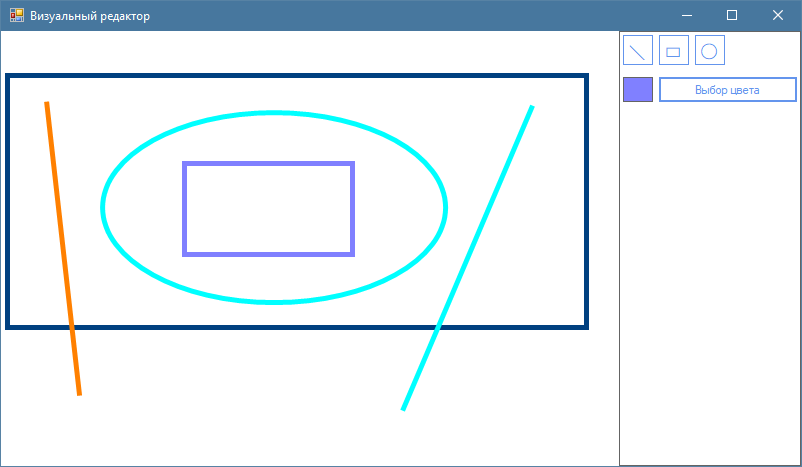


Рисунок 9 - Пример работы программы

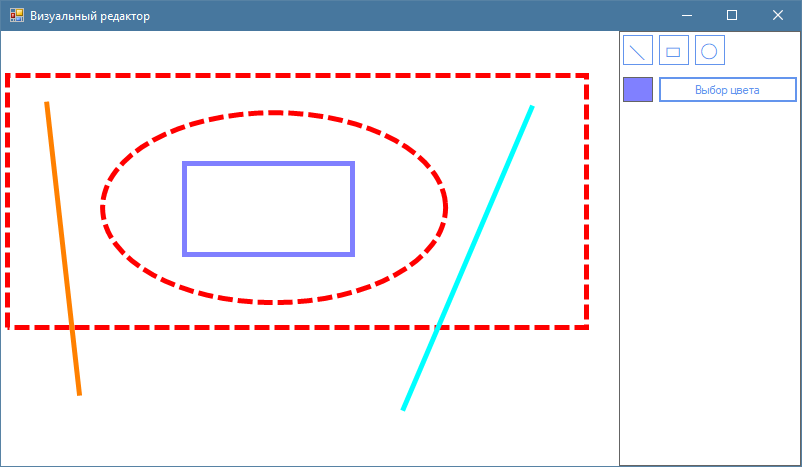


Рисунок 10 - Пример работы программы

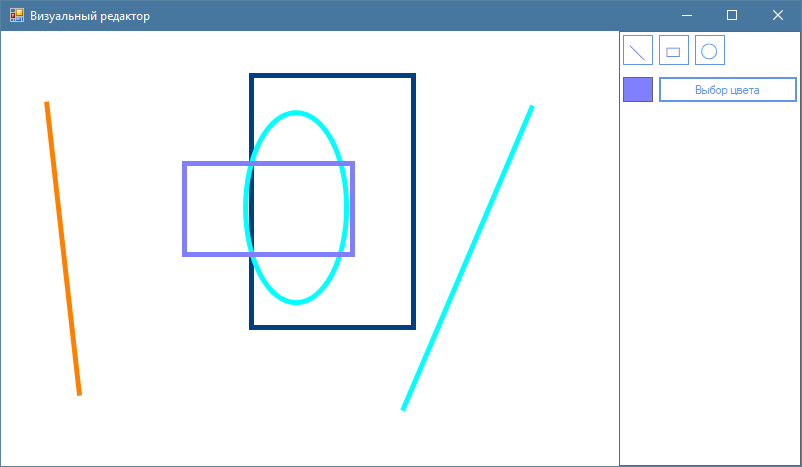


Рисунок 11 - Пример работы программы

Приложение C. Исходный код программы.

1. **Файл Drawings.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Drawing;

using System.Text;

using System.Threading.Tasks;

namespace Лабораторная\_работа\_\_6

{

public abstract class GraphicObject

{

public int x;

public int y;

public int width;

public int height;

public bool isSelected = false;

public Color color;

public int thickness = 5;

public GraphicObject(int x, int y, int width, int height, Color color)

{

this.x = x;

this.y = y;

this.width = width;

this.height = height;

this.color = color;

}

public abstract string className();

public abstract bool belongsTo(int x, int y);

}

public class Line : GraphicObject // В Line width и height становятся x2 и y2

{

public Line(int x, int y, int width, int height, Color color)

: base(x, y, width, height, color) { }

public override string className()

{

return "Line";

}

public override bool belongsTo(int x, int y)

{

double tgA = (double)(height) / (width) ;

int minX = Math.Min(this.x, this.x + width);

int maxX = Math.Max(this.x, this.x + width);

int minY = Math.Min(this.y, this.y + height);

int maxY = Math.Max(this.y, this.y + height);

double d = Math.Abs(tgA \* (this.x - x) + y - this.y) / Math.Sqrt(tgA \* tgA + 1);

if (d <= thickness && minX <= x && x <= maxX && minY <= y && y <= maxY)

return true;

return false;

}

}

public class Rectangle : GraphicObject

{

public Rectangle(int x, int y, int width, int height, Color color)

: base(x, y, width, height, color) { }

public override string className()

{

return "Rectangle";

}

public override bool belongsTo(int x, int y)

{

if (this.x - thickness <= x && x <= this.x + width + thickness)

if (this.y - thickness <= y && y <= this.y + height + thickness)

{

int result = 0;

if (Math.Abs(x - this.x) <= thickness)

result++;

if (Math.Abs(x - this.x - width) <= thickness)

result++;

if (Math.Abs(y - this.y) <= thickness)

result++;

if (Math.Abs(y - this.y - height) <= thickness)

result++;

if (result == 1)

return true;

}

return false;

}

}

public class Ellipse : GraphicObject

{

public Ellipse(int x, int y, int width, int height, Color color)

: base(x, y, width, height, color) { }

public override string className()

{

return "Ellipse";

}

public override bool belongsTo(int x, int y)

{

int a = width / 2;

int b = height / 2;

int centerX = this.x + a;

int centerY = this.y + b;

int lowR = (a - 3) \* (b - 3);

int highR = (a + 3) \* (b + 3);

int temp = b \* b \* (x - centerX) \* (x - centerX) + a \* a \* (y - centerY) \* (y - centerY);

return lowR \* lowR <= temp && temp <= highR \* highR;

}

}

public class Painter

{

private Graphics g;

private Color selectedColor;

public Painter(Graphics g, Color selectedColor)

{

this.g = g;

this.selectedColor = selectedColor;

}

public void drawObject(GraphicObject obj)

{

Pen pen;

if (obj.isSelected)

{

pen = new Pen(selectedColor, obj.thickness);

pen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;

}

else

pen = new Pen(obj.color, obj.thickness);

string className = obj.className();

switch(className)

{

case "Line":

g.DrawLine(pen, obj.x, obj.y, obj.x + obj.width, obj.y + obj.height);

break;

case "Rectangle":

{

int x1 = obj.width + obj.x;

int y1 = obj.height + obj.y;

System.Drawing.Rectangle rect = new System.Drawing.Rectangle(

Math.Min(obj.x, x1),

Math.Min(obj.y, y1),

Math.Abs(obj.width),

Math.Abs(obj.height));

g.DrawRectangle(pen, rect);

}

break;

case "Ellipse":

g.DrawEllipse(pen, obj.x, obj.y, obj.width, obj.height);

break;

}

}

}

}

1. **Файл Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Лабораторная\_работа\_\_6

{

public partial class MainForm : Form

{

Model model;

public MainForm()

{

InitializeComponent();

this.DoubleBuffered = true;

SetStyle(ControlStyles.AllPaintingInWmPaint | ControlStyles.UserPaint, true);

SetStyle(ControlStyles.OptimizedDoubleBuffer, true);

model = new Model();

}

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

{

if (e.Button == MouseButtons.Left)

{

model.mouseIsPressed = true;

model.doTheRightThing(e.X, e.Y);

pictureBox.Image = model.image;

}

}

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

{

if (e.Button == MouseButtons.Left)

if (model.creatingObject)

{

model.drawOnline(e.X, e.Y);

pictureBox.Image = model.image;

}

}

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

{

if (e.Button == MouseButtons.Left)

if (model.creatingObject == true)

{

model.mouseIsPressed = false;

model.creatingObject = false;

model.add();

pictureBox.Image = model.image;

}

}

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

{

model.setMode(0);

}

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

{

model.setMode(1);

}

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

{

model.setMode(2);

}

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

{

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

{

colorLabel.BackColor = colorDialog.Color;

model.setColor(colorDialog.Color);

}

}

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

{

switch(e.KeyCode)

{

case Keys.ControlKey:

model.ctrlIsPressed = true;

break;

case Keys.ShiftKey:

model.altIsPressed = true;

break;

case Keys.W:

model.wIsPressed = true;

model.move();

break;

case Keys.A:

model.aIsPressed = true;

model.move();

break;

case Keys.S:

model.sIsPressed = true;

model.move();

break;

case Keys.D:

model.dIsPressed = true;

model.move();

break;

case Keys.Delete:

model.delete();

break;

}

pictureBox.Image = model.image;

}

private void MainForm\_KeyUp(object sender, KeyEventArgs e)

{

switch (e.KeyCode)

{

case Keys.ControlKey:

model.ctrlIsPressed = false;

break;

case Keys.ShiftKey:

model.altIsPressed = false;

model.correct();

break;

case Keys.W:

model.wIsPressed = false;

break;

case Keys.A:

model.aIsPressed = false;

break;

case Keys.S:

model.sIsPressed = false;

break;

case Keys.D:

model.dIsPressed = false;

break;

}

model.velocity = 5;

}

}

}

1. **Файл Model.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Drawing;

using System.Text;

using System.Threading.Tasks;

namespace Лабораторная\_работа\_\_6

{

public class Model

{

public Graphics g;

public Bitmap image;

private Painter painter;

private Storage<GraphicObject> storage;

private Storage<GraphicObject> selectedStorage;

private Storage<Color> originColorStorage;

private int creationMode;

private Color color = Color.Black;

public bool ctrlIsPressed;

public bool altIsPressed;

public bool wIsPressed;

public bool aIsPressed;

public bool sIsPressed;

public bool dIsPressed;

public bool mouseIsPressed;

public bool creatingObject;

public int velocity;

public Model()

{

storage = new Storage<GraphicObject>();

selectedStorage = new Storage<GraphicObject>();

originColorStorage = new Storage<Color>();

image = new Bitmap(1920, 1080);

g = Graphics.FromImage(image);

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

painter = new Painter(g, Color.Red);

velocity = 5;

}

public void setMode(int mode)

{

creationMode = mode;

}

public void setColor(Color color)

{

this.color = color;

if(selectedStorage.getSize()!=0)

{

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

selectedStorage.getCurrent().color = color;

}

}

public GraphicObject temp;

public void doTheRightThing(int x, int y)

{

if(check(x,y)) // Если попали по элементу, выбор

{

redrawAll();

}

else // Если попали по пустому месту, создание

{

deselectAll();

creatingObject = true;

if (creationMode == 0)

temp = new Line(x, y, 0, 0, color);

else if (creationMode == 1)

temp = new Rectangle(x, y, 0, 0, color);

else if (creationMode == 2)

temp = new Ellipse(x, y, 0, 0, color);

}

}

public void drawOnline(int x, int y)

{

if (x < 0)

x = 0;

if (y < 0)

y = 0;

g.Clear(Color.White);

temp.width = x - temp.x;

temp.height = y - temp.y;

redrawAll();

painter.drawObject(temp);

}

public void redrawAll()

{

g.Clear(Color.White);

storage.setFirst();

for (int i = 0; i < storage.getSize(); i++, storage.next())

painter.drawObject(storage.getCurrent());

}

private bool check(int x,int y)

{

bool found = false;

storage.setFirst();

for (int i = 0; i < storage.getSize(); i++, storage.next())

if (storage.getCurrent().belongsTo(x, y))

{

temp = storage.getCurrent();

if (ctrlIsPressed)

{

if (temp.isSelected)

{

temp.isSelected = false;

selectedStorage.checkAndSetCurrent(temp);

selectedStorage.del();

}

else

{

temp.isSelected = true;

selectedStorage.add(temp);

}

}

else

{

deselectAll();

if (temp.isSelected == false)

{

selectedStorage.add(storage.getCurrent());

temp.isSelected = true;

}

}

found = true;

}

return found;

}

private void deselectAll() // Убираем все элементы из списка выбранных

{

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

selectedStorage.getCurrent().isSelected = false;

selectedStorage = new Storage<GraphicObject>();

}

public void add()

{

if (Math.Abs(temp.width) < 10 && Math.Abs(temp.height) < 10)

{

redrawAll();

return;

}

if (temp.className() != "Line")

if (temp.width < 0 || temp.height < 0)

{

temp.x = Math.Min(temp.x, temp.x + temp.width);

temp.y = Math.Min(temp.y, temp.y + temp.height);

temp.width = Math.Abs(temp.width);

temp.height = Math.Abs(temp.height);

}

storage.add(temp);

}

public void correct()

{

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

{

temp = selectedStorage.getCurrent();

if (temp.className() != "Line")

if (temp.width < 0 || temp.height < 0)

{

temp.x = Math.Min(temp.x, temp.x + temp.width);

temp.y = Math.Min(temp.y, temp.y + temp.height);

temp.width = Math.Abs(temp.width);

temp.height = Math.Abs(temp.height);

}

}

}

public void move()

{

if (altIsPressed)

{

changeSize();

return ;

}

if (selectedStorage.getSize() == 0)

return ;

int x = 0;

int y = 0;

velocity++;

if (wIsPressed)

y--;

if (aIsPressed)

x--;

if (sIsPressed)

y++;

if (dIsPressed)

x++;

x \*= velocity / 5;

y \*= velocity / 5;

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

{

temp = selectedStorage.getCurrent();

if (temp.x + x >= 0 && temp.x + temp.width + x >= 0)

temp.x += x;

if (temp.y + y >= 0 && temp.y + temp.height + y >= 0)

temp.y += y;

}

redrawAll();

}

public void changeSize()

{

if (selectedStorage.getSize() == 0)

return;

velocity++;

int currentVelocity = velocity / 5;

selectedStorage.setFirst();

for (int i = 0; i < selectedStorage.getSize(); i++, selectedStorage.next())

{

temp = selectedStorage.getCurrent();

if (wIsPressed)

if (temp.y - currentVelocity >= 0 && temp.y + temp.height - currentVelocity >= 0)

{

temp.height -= currentVelocity;

}

if (aIsPressed)

if (temp.x - currentVelocity >= 0 && temp.x + temp.width - currentVelocity >= 0)

{

temp.width -= currentVelocity;

}

if (sIsPressed)

if (temp.y + currentVelocity >= 0 && temp.y + temp.height + currentVelocity >= 0)

temp.height += currentVelocity;

if (dIsPressed)

if (temp.x + currentVelocity >= 0 && temp.x + temp.width + currentVelocity >= 0)

temp.width += currentVelocity;

}

redrawAll();

}

public void delete()

{

storage.setFirst();

for (int i = 0; i < storage.getSize(); i++)

if (storage.getCurrent().isSelected)

{

storage.del();

i--;

}

else

storage.next();

selectedStorage = new Storage<GraphicObject>();

redrawAll();

}

}

}

1. **Файл Storage.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лабораторная\_работа\_\_6

{

public class Storage<T>

{

private class Node

{

public T obj;

public Node previous;

public Node next;

}

private int size;

private Node first;

private Node last;

private Node current;

public Storage()

{

size = 0;

}

public void add(T obj) // Добавляет объект в хранилище в конец списка

{

Node temp = new Node();

temp.obj = obj;

size++;

if (first == null)

{

first = temp;

last = temp;

current = temp;

}

else

{

last.next = temp;

temp.previous = last;

last = temp;

}

}

public void del() // Удаляет текущий элемент

{

if (current != null)

{

// Переназначение "указателей" соседних элементов

if (current.previous != null)

current.previous.next = current.next;

if (current.next != null)

current.next.previous = current.previous;

// Перевод current на следующий или предыдущий элемент

Node oldCurrent = current;

if (current.next != null)

current = current.next;

else if (current.previous != null)

current = current.previous;

else

current = null;

// Смена "указателей" first и last, если current был им равен

if (oldCurrent == first)

first = current;

if (oldCurrent == last)

last = current;

// Коррекция размера списка

size--;

}

}

public void previous() // Переносит current на предыдущий элемент в списке, если предыдущий элемент существует

{

if (current != null)

if (current.previous != null)

current = current.previous;

}

public void next() // Переносит current на следующий элемент в списке, если следующий элемент существует

{

if (current != null)

if (current.next != null)

current = current.next;

}

public bool check(T obj) // Проверяет наличие объекта хранилище, не изменяя current

{

Node buffer = first;

for (int i = 0; i < size; i++, buffer = buffer.next)

if (buffer.obj.Equals(obj))

return true;

return false;

}

public bool checkAndSetCurrent(T obj) // Проверяет наличие объекта хранилище и устанавливает current на этот объект

{

Node buffer = first;

for (int i = 0; i < size; i++, buffer = buffer.next)

if (buffer.obj.Equals(obj))

{

current = buffer;

return true;

}

return false;

}

public int getSize()

{

return size;

}

public T getFirst() // Возвращает первый объект в списке

{

return first.obj;

}

public T getLast() // Возвращает последний объект в списке

{

return last.obj;

}

public ref T getCurrent() // Возвращает текущий объект

{

return ref current.obj;

}

public void setFirst() // Устанавливает current на начало списка

{

current = first;

}

public void setLast() // Устанавливает current на конец списка

{

current = last;

}

public bool eol() // End Of List

{

if (current == null)

return true;

if (current.next == null)

return true;

else

return false;

}

}

}