**ИИТ БГУИР**

**Факультет повышения квалификации и переподготовки**   
  
**Дисциплина: Средства визуального программирования приложений**

**Лабораторная работа №5**

Меню, команды, диалоговые окна

Вариант 3

**Выполнил слушатель гр. 40322-1** Проверила

**Гончаров Максим Евгеньевич Желакович Ирина Миролюбовна**

Минск 2025

*Цели лабораторной работы*:

Получить навыки работы с многооконными приложениями. Научиться использовать меню, панели инструментов, статусную строку.

*Выполнение работы:*

Создан графический редактор: в окне документа, в точке, где пользователь щелкнув левой клавишей мыши, вызывает эллипс с радиальным градиентом. Анимация – смещение одной из стоп-точек градиента. В приложении предусмотрено диалоговое окно для выбора толщины линий фигуры, выбора цвета фона и цвета линий фигуры, диалог отображения сведений о программе. Отображение окна диалога реализован через меню.

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:xctk="http://schemas.xceed.com/wpf/xaml/toolkit"

xmlns:local="clr-namespace:LB5"

xmlns:Globalization="clr-namespace:System.Globalization;assembly=System.Runtime" x:Class="LB5.WindowShape"

mc:Ignorable="d"

Title="WindowShape" Height="450" Width="400" FontSize="24">

<Grid x:Name="grid">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label Content="Foreground" Margin="10 3"/>

<Label Content="Background" Grid.Row="1" Margin="10 3"/>

<Label Content="Thickness" Grid.Row="2" Margin="10 3"/>

<Label Content="Width" Grid.Row="3" Margin="10 3"/>

<Label Content="Height" Grid.Row="4" Margin="10 3"/>

<xctk:ColorPicker Grid.Column="1" Margin="15" FontSize="18" SelectedColor="{Binding Foreground}"/>

<xctk:ColorPicker Grid.Row="1" Grid.Column="1" Margin="15" FontSize="18" SelectedColor="{Binding Background}"/>

<xctk:IntegerUpDown Grid.Row="2" Grid.Column="1" Margin="15" Value="{Binding Thickness}" Minimum="0" Maximum="10"/>

<xctk:IntegerUpDown Grid.Row="3" Grid.Column="1" Margin="15" Value="{Binding Width}" Minimum="0" Maximum="300"/>

<xctk:IntegerUpDown Grid.Row="4" Grid.Column="1" Margin="15" Value="{Binding Height}" Minimum="0" Maximum="300">

</xctk:IntegerUpDown>

<Button Content="OK" Grid.Row="5" Margin="20 10" Click="ButtonOK\_Click"/>

<Button Content="Cancel" Grid.Column="1" Grid.Row="5" Margin="20 10" Click="ButtonCancel\_Click"/>

</Grid>

</Window>

Для выбора цвета и размера фигуры использовать элементы управления библиотеки DotNetProjects.Extended.Wpf.Toolkit:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным. Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

В приложение добавлено чтение и запись сформированного пользователем изображения в файл. Файл содержит размер фигуры, цвет линий и цвет фона, значение толщины линий. Сохранение файла доступно только при наличии фигур в документе. Формат файла – xml.

public void draw(Canvas canvas, System.Windows.Point point)

{

Ellipse ellipse = new Ellipse();

ellipse.Width = Width;

ellipse.Height = Height;

ellipse.Margin = new Thickness(point.X - Width / 2, point.Y - Height / 2, 0, 0);

// Создаем радиальный градиент

RadialGradientBrush gradient = new RadialGradientBrush();

gradient.GradientStops.Add(new GradientStop((Color)Background, 0));

gradient.GradientStops.Add(new GradientStop((Color)Foreground, 1));

gradient.RadiusX = 0.7;

gradient.RadiusY = 0.7;

gradient.Center = new System.Windows.Point(0.5, 0.5);

ellipse.Fill = gradient;

ellipse.Stroke = new SolidColorBrush((Color)Foreground);

ellipse.StrokeThickness = Thickness;

// Добавляем анимацию смещения стоп-точки

DoubleAnimation animation = new DoubleAnimation();

animation.To = 0.3;

animation.Duration = TimeSpan.FromSeconds(2);

animation.AutoReverse = true;

animation.RepeatBehavior = RepeatBehavior.Forever;

gradient.GradientStops[1].BeginAnimation(GradientStop.OffsetProperty, animation);

canvas.Children.Add(ellipse);

// Добавляем перечеркивающие диагональные линии (в пределах эллипса)

double centerX = point.X;

double centerY = point.Y;

double halfWidth = Width / 2;

double halfHeight = Height / 2;

Line diagonalLine1 = new Line();

diagonalLine1.X1 = centerX - halfWidth \* 0.7;

diagonalLine1.Y1 = centerY - halfHeight \* 0.7;

diagonalLine1.X2 = centerX + halfWidth \* 0.7;

diagonalLine1.Y2 = centerY + halfHeight \* 0.7;

diagonalLine1.Stroke = new SolidColorBrush((Color)Foreground);

diagonalLine1.StrokeThickness = Thickness;

Line diagonalLine2 = new Line();

diagonalLine2.X1 = centerX + halfWidth \* 0.7;

diagonalLine2.Y1 = centerY - halfHeight \* 0.7;

diagonalLine2.X2 = centerX - halfWidth \* 0.7;

diagonalLine2.Y2 = centerY + halfHeight \* 0.7;

diagonalLine2.Stroke = new SolidColorBrush((Color)Foreground);

diagonalLine2.StrokeThickness = Thickness;

canvas.Children.Add(diagonalLine1);

canvas.Children.Add(diagonalLine2);

}

public void save()

{

SaveFileDialog fileDialog = new SaveFileDialog();

fileDialog.Filter = "Файлы xml|\*.xml|Все файлы|\*.\*";

if (fileDialog.ShowDialog() == false) return;

XmlSerializer serializer = new XmlSerializer(typeof(Shape));

using (FileStream file = new FileStream(fileDialog.FileName, FileMode.Create))

{

serializer.Serialize(file, this);

}

}

public static Shape load()

{

OpenFileDialog fileDialog = new OpenFileDialog();

fileDialog.Filter = "Файлы xml|\*.xml|Все файлы|\*.\*";

if (fileDialog.ShowDialog() == false) return null;

XmlSerializer serializer = new XmlSerializer(typeof(Shape));

Shape shape;

using (FileStream file = new FileStream(fileDialog.FileName, FileMode.Open))

{

shape = (Shape)serializer.Deserialize(file);

}

return shape;

}

Загрузка и сохранение файла, выход из приложения и вызов справки реализовано с помощью механизма команд WPF.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

В панели статуса добавьте вывод координат мыши.

Изображение выглядит как текст, снимок экрана, графический дизайн, Графика

Содержимое, созданное искусственным интеллектом, может быть неверным.

Продублировать все функции в панели инструментов.

<ToolBar DockPanel.Dock="Top" Height="40">

<Button x:Name="buttonShape" Content="◯" ToolTip="Shape" Click="MenuItemShape\_Click"/>

<Button x:Name="buttonSave" Content="💾" Command="Save"/>

<Button x:Name="buttonOpen" Content="📂" Command="Open"/>

<Button x:Name="buttonHelp" Content="ℹ️" Command="Help"/>

<Button x:Name="buttonExit" Content="❌" Command="Close"/>

</ToolBar>

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Добавьте анимированную фигуру согласно варианту. Залейте фигуру градиентом с несколькими стоп-точками. Линейная анимация должна применяться к указанному свойству с реверсом, вечно.

Изображение выглядит как текст, снимок экрана, мяч, круг

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как снимок экрана, круг, текст, мяч

Содержимое, созданное искусственным интеллектом, может быть неверным.

<Canvas x:Name="canvas" Background="Beige" MouseDown="canvas\_MouseDown"

MouseMove="canvas\_MouseMove">

<Ellipse Height="230" Width="200" Canvas.Left="550" Canvas.Top="50" Stroke="Black">

<Ellipse.Fill>

<RadialGradientBrush Center="0.5,0.5" RadiusX="0.5" RadiusY="0.5">

<GradientStop Color="Aquamarine" Offset="0"/>

<GradientStop Color="Gold" Offset="0.7"/>

<GradientStop Color="White" Offset="0.3"/>

<GradientStop Color="RosyBrown" Offset="1"/>

</RadialGradientBrush>

</Ellipse.Fill>

<Ellipse.Triggers>

<EventTrigger RoutedEvent="Ellipse.MouseDown">

<BeginStoryboard>

<Storyboard >

<DoubleAnimation Storyboard.TargetProperty="Fill.RadiusX"

To="0.8" Duration="0:0:2" AutoReverse="True" RepeatBehavior="Forever"/>

<DoubleAnimation Storyboard.TargetProperty="Fill.RadiusY"

To="0.8" Duration="0:0:2" AutoReverse="True" RepeatBehavior="Forever"/>

<PointAnimation Storyboard.TargetProperty="Fill.Center"

To="0.7,0.7" Duration="0:0:3" AutoReverse="True" RepeatBehavior="Forever"/>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

</Ellipse.Triggers>

</Ellipse>

</Canvas>

Код проектов всех лабораторных работ, задания и отчеты находятся в публичном доступе на сайте GitHub: <https://github.com/Maxevgen555/SVPP_LB>. Все проекты находятся в одном решении и выполнены в VS2022. Файл решения SVPP\_LB.sln находится в папке \ЛБ1\LB1\_Calc\.

*Вывод:*

Я получил навыки работы с многооконными приложениями. Научился использовать меню, панели инструментов, статусную строку, рисовать анимированные фигуры, сохранять их в файл и загружать.