ЛАБОРАТОРНАЯ РАБОТА 13

Тема: «Разработка отладка и испытание программ с пользовательским интерфейсом WPF»

Цель: Сформировать умения и навыки разработки программ с пользовательским интерфейсом технологии Windows Presentation Foundation.

Время выполнения: 4 часа.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на **DirectX**. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Одной из важных особенностей является использование языка декларативной разметки интерфейса XAML, основанного на XML: вы можете создавать насыщенный графический интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C# и VB.NET, либо совмещать и то, и другое.

Преимущества WPF

Что вам, как разработчику, предлагает WPF?

Использование традиционных языков .NET-платформы - C# и VB.NET для создания логики приложения

Возможность **декларативного определения** графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET

Независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением.

Новые возможности, которых сложно было достичь в WinForms, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и др.

Хорошее **взаимодействие с WinForms**, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms.

Богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое

Аппаратное ускорение графики - вне зависимости от того, работаете ли вы с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной.

Создание приложений под множество ОС семейства Windows - от Windows XP до Windows 10

В тоже время WPF имеет определенные ограничения. Несмотря на поддержку трехмерной визуализации, для создания приложений с большим количеством трехмерных изображений, прежде всего игр, лучше использовать другие средства - DirectX или специальные фреймворки, такие как Monogame или Unity.

Также стоит учитывать, что по сравнению с приложениями на Windows Forms объем программ на WPF и потребление ими памяти в процессе работы в среднем несколько выше. Но это с лихвой компенсируется более широкими графическими возможностями и провышенной производительностью при отрисовке графики.

Архитектура WPF

Managed API (управляемый API-интерфейс) содержит код, исполняемый под управлением общеязыковой среды выполнения .NET - Common Language Runtime. Этот API описывает основной функционал платформы WPF и состоит из следующих компонентов:

PresentationFramework.dll: содержит все основные реализации компонентов и элементов управления, которые можно использовать при построении графического интерфейса

PresentationCore.dll: содержит все базовые типы для большинства классов из PresentationFramework.dll

WindowsBase.dll: содержит ряд вспомогательных классов, которые применяются в WPF, но могут также использоваться и вне данной платформы

Unmanaged API используется для интеграции вышележащего уровня с DirectX:

milcore.dll: собственно обеспечивает интеграцию компонентов WPF с DirectX. Данный компонент написан на неуправляемом коде (C/C++) для взаимодействия с DirectX.

WindowsCodecs.dll: библиотека, которая предоставляет низкоуровневую поддержку для изображений в WPF

Еще ниже собственно находятся компоненты операционной системы и DirectX, которые произвоят визуализацию компонентов приложения, либо

выполняют прочую низкоуровневую обработку. В частности, с помощью низкоуровневого интерфейса Direct3D, который входит в состав DirectX, происходит трансляция

Для элементов управления в WPF определено большое количество событий, которые условно можно разделить на несколько групп:

события клавиатуры;

события мыши;

события стилуса;

события сенсорного экрана/мультитач;

события жизненного цикла.

Подключение обработчиков событий

Подключить обработчики событий можно декларативно в файле xaml-кода, а можно стандартным способом в файле отделенного кода.

Декларативное подключение:

```
<Button x:Name="Button1" Content="Click"
Click="Button_Click" />
```

Для определения маршрутизированных событий в классе создвалось статическое поле по типу **RoutedEvent**:

```
public static RoutedEvent СобытиеEvent
```

Это поле, как правило, имеет суффикс *Event*. Затем это событие регистрируется в статическом конструкторе.

Пример обработки события по нажатию клавиши:

```
public abstract class ButtonBase : ContentControl, ...
{
    // определение событие
    public static readonly RoutedEvent ClickEvent;

    static ButtonBase()
    {
        // регистрация маршрутизированного события
                        ButtonBase.ClickEvent =
EventManager.RegisterRoutedEvent("Click",
                          RoutingStrategy.Bubble,
typeof(RoutedEventHandler), typeof(ButtonBase));
        //.............................
    }
    // обертка над событием
    public event RoutedEventHandler Click
    {
        add
        {
            // добавление обработчика
             base.AddHandler(ButtonBase.ClickEvent,
value);
```

```
            }
            remove
            {
                // удаление обработчика
                base.RemoveHandler(ButtonBase.ClickEven
t, value);
            }
        }
      // остальное содержимое класса
    }
```

Маршрутизированные события регистрируются с помощью метода EventManager.RegisterRoutedEvent(). В этот метод передаются последовательно имя события, тип события (поднимающееся, прямое, опускающееся), тип делегата, предназначенного для обработки события, и класс, который владеет этим событием.

Модель событий WPF отличается от событий WinForms не только декларативным подключением. События, возникнув на одном элементе, могут обрабатываться на другом. События могут подниматься и опускаться по дереву элементов.

Так, маршрутизируемые события делятся на три вида:

прямые (direct events) - они возникают и отрабатывают на одном элементе и никуда дальше не передаются. Действуют как обычные события.

поднимающиеся (bubbling events) - возникают на одном элементе, а потом передаются дальше к родителю - элементу-контейнеру и далее, пока не достигнет наивысшего родителя в дереве элементов.

опускающиеся, туннельные (tunneling events) - начинает отрабатывать в корневом элементе окна приложения и идет далее по вложенным элементам, пока не достигнет элемента, вызвавшего это событие.

Все маршрутизируемые события используют класс **RoutedEventArgs** (или его наследников), который представляет доступ к следующим свойствам:

source: элемент логического дерева, являющийся источником события.

originalsource: элемент визуального дерева, являющийся источником события. Обычно то же самое, что и Source

routedevent: представляет имя события

handled: если это свойство установлено в True, событие не будет подниматься и опускаться, а ограничится непосредственным источником.

К событиям клавиатуры можно отнести следующие события, представленные в таблице 1.

Таблица 1 – События клавиатуры в WPF

Событие	Тип события	Описание
KeyDown	Поднимающееся	Возникает при нажатии клавиши

PreviewKeyDown	Туннельное	Возникает при нажатии клавиши
KeyUp	Поднимающееся	Возникает при освобождении клавиши
PreviewKeyUp	Туннельное	Возникает при освобождении клавиши
TextInput	Поднимающееся	Возникает при получении элементом текстового ввода (генерируется не только клавиатурой, но и стилусом)
PreviewTextInput	Туннельное	Возникает при получении элементом текстового ввода

В WPF для мыши определены следующие события, представленные в таблице 2.

Таблица 2 – События мыши в WPF

Событие	Тип события	Описание
GotMouseCapture	Поднимающееся	Возникает при получении фокуса с помощью мыши
LostMouseCapture	Поднимающееся	Возникает при потере фокуса с помощью мыши
MouseEnter	Прямое	Возникает при вхождении указателя мыши в пределы элемента
MouseLeave	Прямое	Возникает, когда указатель мыши выходит за пределы элемента
MouseLeftButtonDown	Поднимающееся	Возникает при нажатии левой кнопки мыши
PreviewMouseLeftButtonDown	Туннельное	Возникает при нажатии левой кнопки мыши

MouseLeftButtonUp	Поднимающееся	Возникает при освобождении левой кнопки мыши
PreviewMouseLeftButtonUp	Туннельное	Возникает при освобождении левой кнопки мыши
MouseRightButtonDown	Поднимающееся	Возникает при нажатии правой кнопки мыши
PreviewMouseRightButtonDown	Туннельное	Возникает при нажатии правой кнопки мыши
MouseRightButtonUp	Поднимающееся	Возникает при освобождении правой кнопки мыши
PreviewMouseRightButtonUp	Туннельное	Возникает при освобождении правой кнопки мыши
MouseDown	Поднимающееся	Возникает при нажатии кнопки мыши
PreviewMouseDown	Туннельное	Возникает при нажатии кнопки мыши
MouseUp	Поднимающееся	Возникает при освобождении кнопки мыши
PreviewMouseUp	Туннельное	Возникает при освобождении кнопки мыши
MouseMove	Поднимающееся	Возникает при передвижении указателя мыши
PreviewMouseMove	Туннельное	Возникает при передвижении указателя мыши

MouseWheel	Поднимающееся	Возникает при передвижении колесика мыши
PreviewMouseWheel	Туннельное	Возникает при передвижении колесика мыши

Индивидуальные задания для лабораторной работы 4:

1. Создайте список, в котором элементы могут быть выделены, как в файловых менеджерах. При клике на элемент списка выделяется только этот элемент, отменяется выделение остальных элементов.
2. Создайте список, в котором элементы могут изменять начертание символов, например, с обычного начертания на курсивное. При клике на элемент списка выделяется только этот элемент, но не отменяется начертнаие остальных элементов.
3. Создайте список, в котором элементы могут изменять цвет символов, например, с черного на красный. При клике на элемент списка выделяется только этот элемент, но не отменяется начертнаие остальных элементов.
4. Создайте список, в котором элементы могут изменять начертание символов, например, с обычного начертания на курсивное. При клике правой кнопкой мыши на элемент списка происходит смена начертания на обычное.
5. Создайте приложение, в котором по щелчку правой кнопкой мыши на форму изменяется цвет формы на тот, который выберет пользователь.
6. Создайте приложение, в котором по щелчку левой кнопки мыши на форму происходит появление надписей с номером лабораторной работы и Вашей фамилией.
7. Создайте приложение, в котором по щелчку правой кнопки мыши по форме появляется диалоговое окно для выбора цвета формы.
8. Создайте приложение, в котором по нажатию клавиши Enter на клавиатуре будет происходить подтверждение ввода данных.
9. Создайте приложение, в котором по нажатию клавиши Esc будет происходить выход из приложения.
10. Создайте приложение, в котором по нажатию на сочетание клавиш будет происходить вызов диалогового окна для сохранения документа.
11. Создайте приложение, в котором по нажатию на сочетание клавиш будет происходить открытие диалоговго окна для открытия файла.

12. Создайте приложение, в котором по нажатию на сочетание клавиш будет происходить копирование и встака текста.

13. Создайте приложение, в котором по нажатию на сочетание клавиш будет происходить открытие диалогового окна для печати документов.

14. Создайте приложение, в котором по нажатию на сочетание клавиш будет происходить открыте диалогового окна для выбора цвета формы.

15. Создайте приложение, в котором по нажатию на сочетание клавиш будет происходить открытие диалогового окна для выбора шрифта текста.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть лабораторной работы.

2. Реализовать индивидуальное задание по вариантам, представленные в теоретических сведениях, сделать скриншоты работающих программ. Написать комментарии.

3. Написать отчет, содержащий:

1. Титульный лист, на котором указывается:

а) полное наименование министерства образование и название учебного заведения;

б) название дисциплины;

в) номер практического занятия;

г) фамилия преподавателя, ведущего занятие;

д) фамилия, имя и номер группы студента;

е) год выполнения лабораторной работы.