**Лабораторная работа № 9: Создание приложения-презентации Windows Foundation Presentation**

**Содержание**

1. [**Вводная часть**](#B01)
2. [**Создание приложения Windows Foundation Presentation**](#B02)
3. [**Модификация приложения Windows Foundation Presentation**](#B03)
4. [**Модификация приложения Windows Foundation Presentation: работа с решением в Expression Blend**](#B04)
5. [**Завершающая часть**](#B05)
6. [**О приложении к Лабораторной работе № 9**](#B06)

**1. Вводная часть**

В предыдущей лабораторной работе данного практикума были рассмотрены основные направления работы с Windows Foundation Presentation и языка разметки XAML.

Несомненно, WPF очень мощный инструмент по созданию «презентационных» приложений. Приложений графически очень «привлекательных». При это создание такого приложения может занять гораздо больше времени на чисто «шарпе» (к примеру **Windows Forms**). И разумеется создать такое приложение на WPF только в среде разработки Visual Studio и при этом применить уникальный стиль можно, но для этого нужно хорошо знать XAML в его первозданном виде. Целью этого практикума не ставится ознакомить читателя с XAML на уровне кода, потому можно пойти другими путями к конечной цели.

Стандартные инструменты редактирования элементов управления и формирования разметки для среды разработки хороши, но не более. Специально для иных, более расширенных оформлений и дизайнов создан другой мощнейший инструмент: **Microsoft Expression Blend** (текущая версия 4).

Целью данной работы станет разработка приложения-презентации, оперирующего страницами XAML как «слайдами» (привет **PowerPoint**!). Для разработки такого приложения используем класс презентации (самописный) и организуем стандартный набор функциональности приложения. Слайды делать можно (и я считаю нужно) не в среде разработки, а непосредственно в **Expression Studio 4 Ultimate**.



Рис. 1. 1. Рис. 1. 1. Получение Epression Studio 4 Ultimate ([DreamSpark.com](http://dreamspark.com/))

Как показано на рисунке выше, пакет Expresson Studio доступен бесплатно. В пакет включены четыре отдельных приложения (Web, blend, Desing, Encoder), а также набор документации. Другие компоненты кроме Blend на не интересуют (опять же цель данного практикума не ознакомить читателя с работой иных сторонних продуктов, но основы работы в Expression Blend будут изложены в кратком виде).

**2. Создание приложения Windows Foundation Presentation**

Запускаем Visual Studio 2010, откроется **Начальная страница**:

Для начала, надо создать проект, для этого выполним последовательно: **Файл** -> **Создать** -> **Проект**… (также можно просто нажать сочетание клавиш **Ctrl+Shift+N** или пункт «**Создать проект**…» на *Начальной странице*):

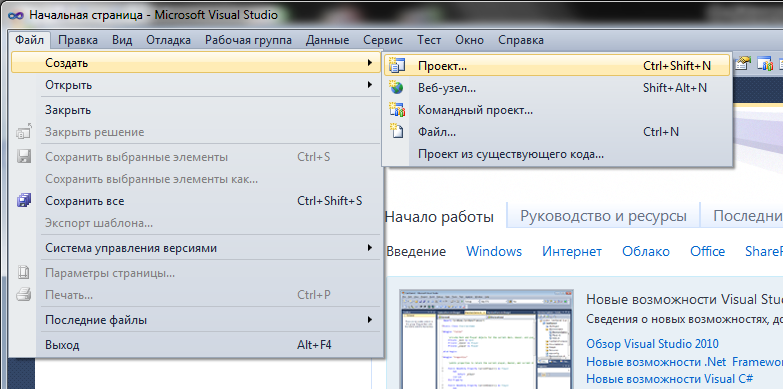


Рис. 2. 1. Создание нового проекта

Выберем слева в пункте **Установленные шаблоны** язык **Visual C#**, далее найдём в списке **Приложение WPF**. Также здесь можно выбрать какой использовать «фреймворк» (набора компонентов для написания программ, конечную платформу). В нашем случае выберем **.NET Framework 4**.

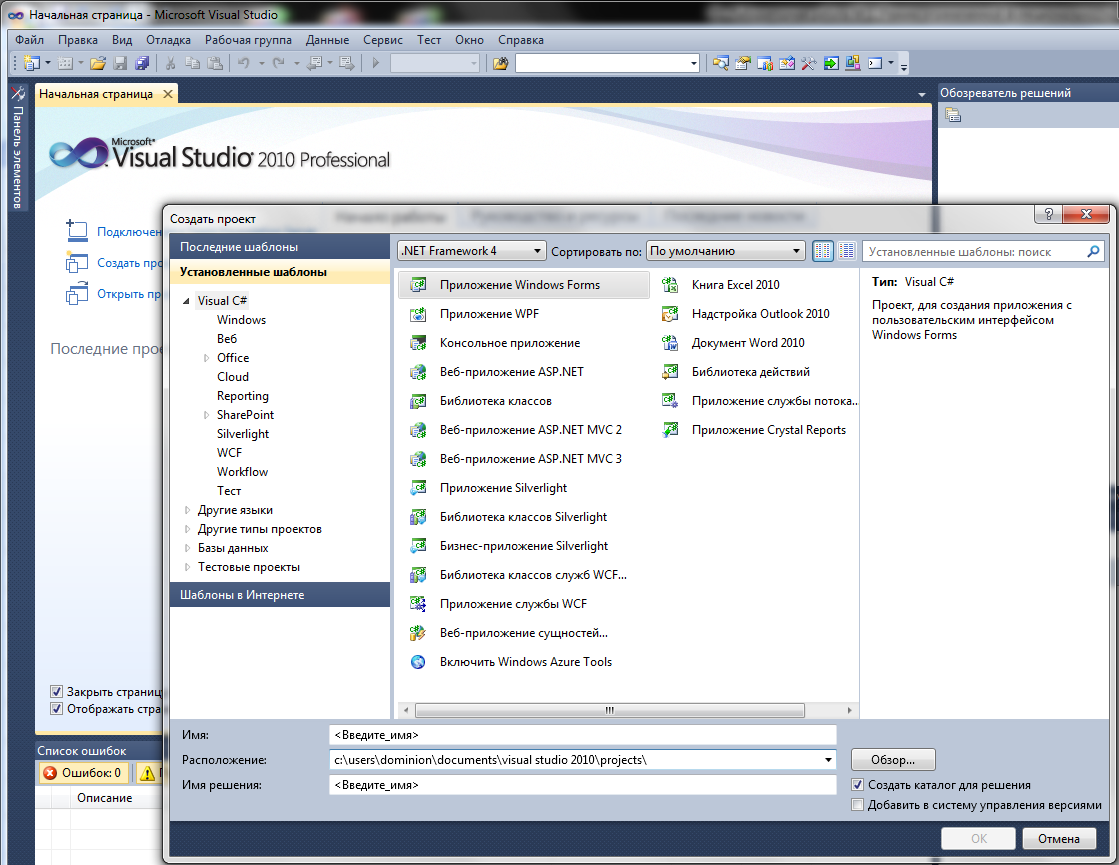


Рис. 2. 2. Окно создания нового проекта

В поле **Имя** вводим **LWP09WPF02** **—** это название программы (выбрано по названию лабораторного практикума, номеру и названию работы). В поле **Расположение** указана конечная директория где будет находится весь проект. Выберем расположение удобное для быстрого поиска. В поле **Имя решения** вводится либо название программы «по умолчанию» из поля *Имя* автоматически, либо можно ввести своё собственное. Под этим именем будет создана конечная папка проекта (если *Имя* и *Имя решения* разные).

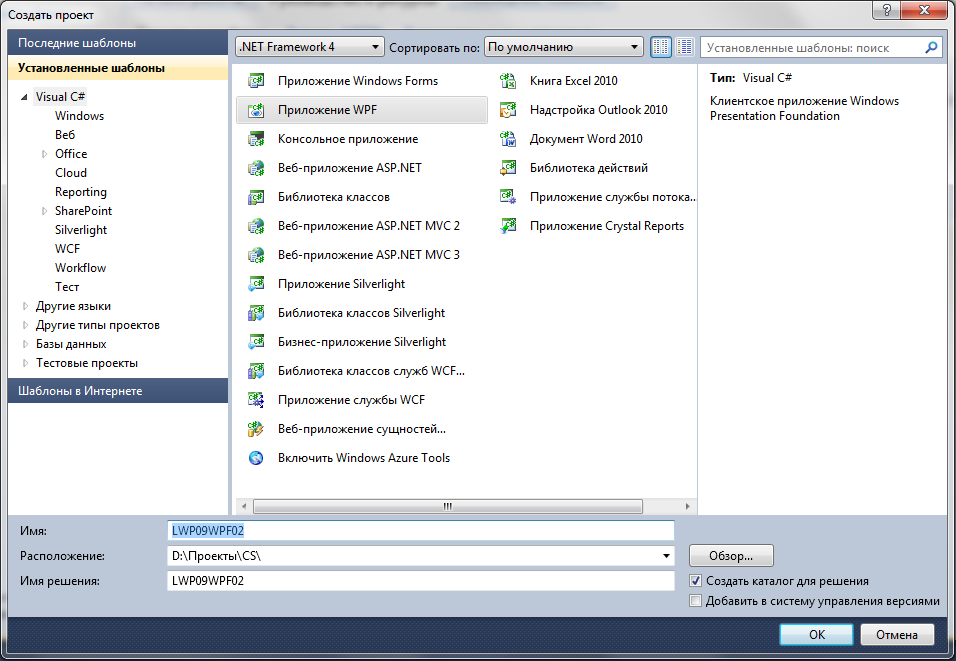


Рис. 2. 3. Вводим данные нового проекта приложения *WPF*

После нажатия клавиши **ОК** мы увидим сформированный проект и исходный код приложения WPF (не пустого изначально). Среда разработки сформировала исходный код в двух файлах (не считая стандартного **AssemblyInfo.cs**). Один из файлов отвечает за программу в целом (это файл **App.xaml.cs**), а второй за формирование окнаприложения **—** инициализацию конструктора формы и всех её элементов (это файл **MainWindow.xaml.cs**). Как видим, исходный код, например файла *App.xaml.cs* не отличается от шаблонного:

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Windows;

namespace LWP09WPF02

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

}

}

XAML-код **App.xaml** выглядит так:

<Application x:Class="LWP09WPF01.App"

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

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

StartupUri="MainWindow.xaml">

<Application.Resources>

</Application.Resources>

</Application>

Файл *MainWindows.xaml.cs* содержит следующий исходный код:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace LWP09WPF01

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

XAML-код **MainWindows.xaml** выглядит так:

<Window x:Class="LWP09WPF01.MainWindow"

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

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

Title="MainWindow" Height="350" Width="525">

<Grid>

</Grid>

</Window>

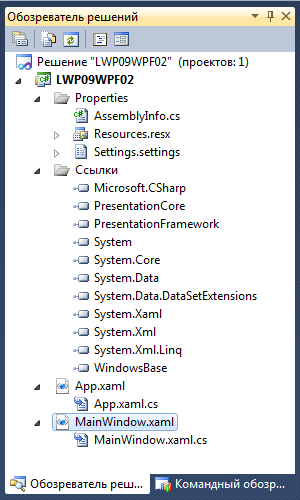


Рис. 2. 4. **Обозреватель решений**: состав проекта приложения WPFсформированного средой разработки

Теперь посмотрим текущий макет нашей главной формы, которая должна запуститься в качестве главного окна:

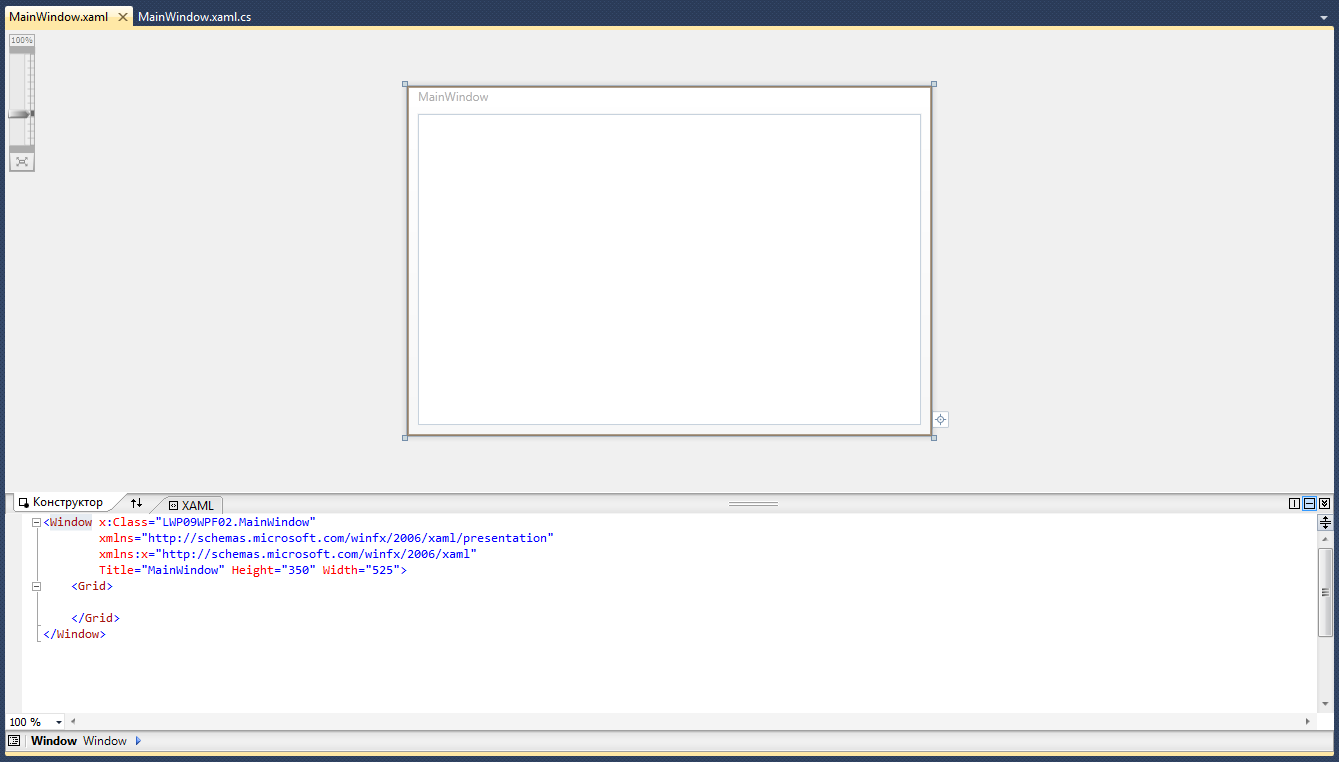


Рис. 2. 5. Макет формы *MainWindows.xaml*: отображение конструктора формы (сверху) и представление этой формы в качестве рамзетки XAML (внизу)

Теперь, можно откомпилировать созданную программу, нажав клавишу **F5** ( **Отладка** -> **Начать отладку** или нажав на иконку. Тем самым мы запускаем приложение в режиме отладки (и производим компиляцию debug-версии программы) (**Debug** выбрано изначально).

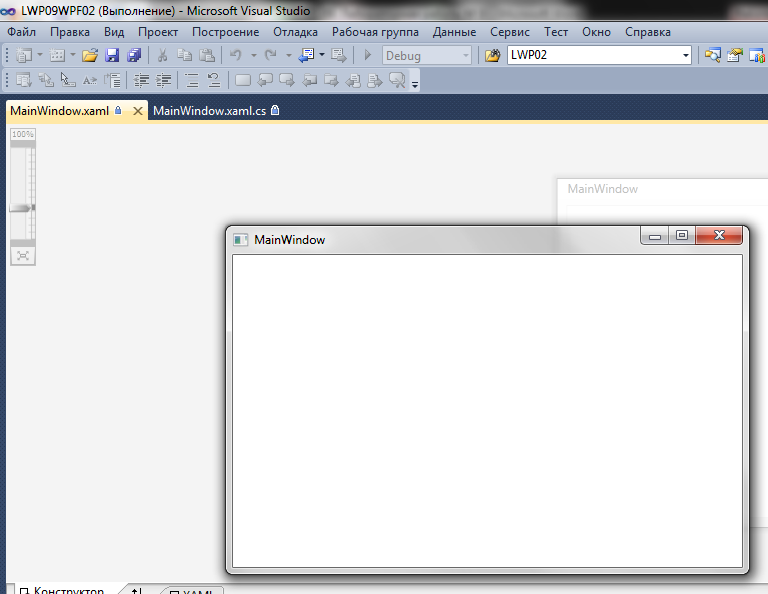


Рис. 2. 6. Запуск приложения WPF по конфигурации *Debug*

**3. Модификация приложения Windows Foundation Presentation**

Начнём сразу с нового класса. Добавим новый класс в проект. Правая кнопка мыши по названию проекта (), затем **Добавить** -> **Создать элемент…** (**Ctrl**+**Shift**+**N**). Выбираем (**Установленные шаблоны**: **Visual C#**): **Клас**с, вводим **Имя**: **Presenation.cs** (пространство имён будет выбрано как у приложения, имя класса будет выбрано по названию файла):



Рис. 3. 1. **Добавление нового элемента – LWP09WPF02**

Сформируется новый файл в обозревателе решений (), его код такой:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace LWP09WPF02

{

class Presentation

{

}

}

Модифицируем код следующим образом:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.ComponentModel;

namespace LWP09WPF02 // Пространство имён приложения (должно быть такое же как и у главного окна)

{

public class Presentation : INotifyPropertyChanged

{

private string[] slides = { "01Title.xaml", "02Slide.xaml", "03Slide.xaml" }; // Список слайдов (последовательность определяет вызов)

private int currentIndex = 0;

private bool currentTimer = false;

public string[] Slides // Свойство для получения списка всех слайдов

{

get { return this.slides; }

}

public string CurrentSlide // Свойство для получения текущего выбранного слайда

{

get { return this.slides[this.currentIndex]; }

}

public int CurrentIndex // Свойство для получения текущего выбранного индекса для слайда или установки нового индекса и генерации PropertyChanged

{

get { return this.currentIndex; }

set

{

if (this.currentIndex != value)

{

this.currentIndex = value;

this.OnPropertyChanged("CurrentSlide");

this.OnPropertyChanged("CanGoBack");

this.OnPropertyChanged("CanGoNext");

}

}

}

public bool CanGoBack // Логическая переменая (свойство): проверка, есть ли ещё страницы для перехода назад

{

get { return this.currentIndex > 0; }

}

public bool CanGoNext // Логическая переменая (свойство): проверка, есть ли ещё страницы для перехода вперёд

{

get { return this.currentIndex < this.slides.Length - 1; }

}

public void GoBack() // Метод Назад

{

if (this.CanGoBack)

{

this.CurrentIndex--;

}

}

public void GoNext() // Метод Вперёд

{

if (this.CanGoNext)

{

this.CurrentIndex++;

}

}

public event PropertyChangedEventHandler PropertyChanged; // Событие PropertyChanged

private void OnPropertyChanged(string propertyName) // Метод получения параметра для события PropertyChanged (при изменении свойств компонента), компонент для события: propertyName

{

if (this.PropertyChanged != null)

{

this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

public void GoHome() // Метод Домой

{

this.CurrentIndex = 0;

}

public bool GoAuto // Метод Авто

{

get { return this.currentTimer; }

set

{

if (this.currentTimer == true)

{

this.currentTimer = false;

}

else

{

this.currentTimer = true;

}

}

}

}

}

Класс есть, теперь нужны слайды с содержимым, события и макет для того чтобы эти слайды выводить.

Все слайды, добавляемые в студии автоматически или в ручную должны быть типа **Page** (страница), а не **Window**. *Page*, как уже было показано в предыдущей лабораторной работе прекрасно подходит для веб-браузера и отображает всё содержимое.

Добавим три слайда (потом ещё один, четвёртый в Expression Blend). Для этого нам нужно выбрать в окне добавления нового элемента (*Установленные шаблоны* -> **WPF**) выберем элемент **Страница (WPF)**. Для первой страницы *Имя* укажем как **01Title.xaml**:



Рис. 3. 2. Добавление новой страницы: *Страница (WPF)*, тип *Page*

XAML-код для *01Title.xaml*:

<Page x:Class="LWP09WPF02.\_01Title"

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

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

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

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

mc:Ignorable="d"

d:DesignHeight="300" d:DesignWidth="300"

Title="\_01Title">

<Grid>

</Grid>

</Page>

Добавим ещё две такие же страницы под именами **02Slide.xaml** и **03Slide.xaml**.

Код добавленных страниц трогать мы не будем (это просто не нужно). Всё будет выполнять главное окно и наш класс. Поэтом добавление новых страниц слайда максимально упрощено и превращается в работу по заполнению их содержимым (как в PowerPoint). Естественно, чтобы добавить для страницы уникальную функциональность, код придётся редактировать. Но лучше всю функциональность организовывать в уже готовом классе, тогда будет максимально просто добавлять новые функции для новых слайдов не заботясь о правильном перенесении кода. Для слайдов же останется лишь добавлять обработчики новых методов и расставлять новые элементы с привязкой.

Изменим код главного окна следующим образом:

<Window x:Class="LWP09WPF02.MainWindow"

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

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

xmlns:LWP09WPF02="clr-namespace:LWP09WPF02"

Title="Презентация WPF (C#)" Height="600" Width="800" ResizeMode="NoResize" WindowState="Maximized" WindowStyle="None" Name="Window" KeyDown="Window\_KeyDown" PreviewMouseMove="Window\_PreviewMouseMove" PreviewMouseLeftButtonDown="Window\_PreviewMouseLeftButtonDown">

<Window.Resources>

<LWP09WPF02:Presentation x:Key="presentation"/>

</Window.Resources>

<Grid>

<Viewbox Margin="10,20,10,40" Stretch="Uniform">

<StackPanel>

<!-- Подставляем всегда: заголовок страницы -->

<TextBlock VerticalAlignment="Top" Height="84" FontFamily="Calibri" FontSize="65" FontWeight="Bold" Text="{Binding Content.Title, ElementName=Frame, Mode=Default}" TextAlignment="Center" TextWrapping="Wrap">

</TextBlock>

<!-- Подставляем всегда: содержимое слайда -->

<Frame Width="1000" Height="600" Source="{Binding CurrentSlide, Source={StaticResource presentation}}" x:Name="Frame" NavigationUIVisibility="Hidden" Background="{x:Null}" Focusable="False"/>

</StackPanel>

</Viewbox>

<Button Width="63.684" Height="36.709" Content="Назад" Click="Back\_Click" RenderTransformOrigin="0.5,0.5" HorizontalAlignment="Left" Margin="10,0,0,10" VerticalAlignment="Bottom" IsEnabled="{Binding CanGoBack, Source={StaticResource presentation}}" />

<Button HorizontalAlignment="Right" Margin="0,0,10,10" VerticalAlignment="Bottom" Width="63.684" Height="36.709" Content="Вперёд" Click="Next\_Click" IsEnabled="{Binding CanGoNext, Source={StaticResource presentation}}" />

<Button Content="Домой" Height="36.709" Margin="80,0,0,10" VerticalAlignment="Bottom" Click="Home\_Click" HorizontalAlignment="Left" Width="64" />

<Button Content="Авто" Height="36.709" HorizontalAlignment="Right" Margin="0,0,80,10" VerticalAlignment="Bottom" Width="63.684" Click="Auto\_Click" />

<Ellipse Height="23" HorizontalAlignment="Right" Margin="0,0,150,12" Name="EAuto" Stroke="Black" VerticalAlignment="Bottom" Width="23" Visibility="Hidden" />

<Label Content="А" Height="26" HorizontalAlignment="Right" Margin="0,0,150,12" Name="LAuto" VerticalAlignment="Bottom" Width="20" Visibility="Hidden" />

</Grid>

</Window>

На самом деле это было сконструрировано в местном визуальном редакторе. Сначала были изменены свойства главного окна:

|  |  |
| --- | --- |
| **Tilte**: | Презентация WPF (C#) |
| **Height**: | 600 |
| **Width**: | 800 |
| **ResizeMode**: | NoResize |
| **WindowState**: | Maximized |
| **WindowStyle**: | None |

Затем добавили ресуры окна:

<LWP09WPF02:Presentation x:Key="presentation"/>

Добавили элемент **ViewBox** (), растянув его по сетке и закрепив, дав свойство:

|  |  |
| --- | --- |
| **Stretch**: | Uniform |

Затем внутрь *ViewBox* запихнули **StackPanel** с двумя элементами: **TextBox** и **Frame** (). Ключевые привязки для отображения подгружаемого содержимого для этих элементов такие:

*TextBox*:

Text="{Binding Content.Title, ElementName=Frame, Mode=Default}"

*Frame*:

Source="{Binding CurrentSlide, Source={StaticResource presentation}}"

Поставили кнопки *Button* (4 штуки), а также нарисовали маленький эллипс и внутрь поставили Label с надписью «А». Кнопки, эллипс и надпись были поставлены вне элемента **ViewBox**, так как они должны отображаться на каждом слайде.

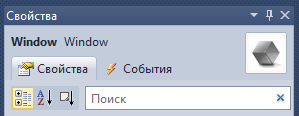
Ключевые привязки (если включена) на события: для кнопки **Назад**:

IsEnabled="{Binding CanGoBack, Source={StaticResource presentation}}"

Кнопка **Вперёд**:

IsEnabled="{Binding CanGoNext, Source={StaticResource presentation}}"

Модифицируем код самого главного (*MainWindow.xaml*) окна следующим образом: для начала задаём окну имя **Window**:



Добавим новые пространства имён в начало файла *MainWindow.xaml.cs*:

using System.IO; // Для работы с классом File

using System.Windows.Threading; // Для создания таймеров

Объявим новые переменные после объявления классаокна:

public partial class MainWindow : Window

{

private Presentation NewPresentation; // Создаём экземпляр нашего класса Presenation

private DispatcherTimer Timer1;

private DispatcherTimer Timer2;

private bool AutoClick;

Изменим код метода **MainWindow()**:

public MainWindow()

{

InitializeComponent();

NewPresentation = (Presentation)this.FindResource("presentation");

AutoClick = false;

// После перезапуска приложения, стартует слайд, дата изменения которого наиболее поздняя (дата изменения страницы XAML)

int IndexLastWritten = 0;

DateTime latestDateTimeWritten = DateTime.MinValue;

for (int i = 0; i < NewPresentation.Slides.Length; i++) // Проходим по общему числу слайдов (через наш собственный класс и метод)

{

String Slide = NewPresentation.Slides[i];

DateTime DateLastWritten = File.GetLastWriteTime(@"..\..\" + Slide); // Узнаём дату из каталога проекта по файлам слайдов

if (DateLastWritten.CompareTo(latestDateTimeWritten) > 0)

{

latestDateTimeWritten = DateLastWritten;

IndexLastWritten = i;

}

}

NewPresentation.CurrentIndex = IndexLastWritten;

//presentation.CurrentIndex = 0; // Если нам нужно, можем всегда стартовать с первой страницы

// Создаём таймеры, которые обрабатывают исчезновение мышки после 5 секунд и перелистывание слайда через 15 секунд

Timer1 = new DispatcherTimer();

Timer1.Interval = TimeSpan.FromSeconds(5);

Timer1.Tick += new EventHandler(Timer1\_Tick);

Timer1.Start();

Timer2 = new DispatcherTimer();

Timer2.Interval = TimeSpan.FromSeconds(15);

Timer2.Tick += new EventHandler(Timer2\_Tick);

}

Впишем события для таймеров. Первый таймер **Timer1** скрывает курсор по событию **Tick**, а второй **Timer2** вызывает из класса *Presentation* метод **GoNext()**:

private void Timer1\_Tick(object sender, EventArgs e)

{

this.Cursor = Cursors.None; // Убираем курсор мышки

}

private void Timer2\_Tick(object sender, EventArgs e)

{

NewPresentation.GoNext();

}

Метод для показа курсора мышки **ShowCursor()** после того, как её скрыли (вызов метода находится в ином месте: в методе нажатия левой клавиши мышки). После вызова, снова запускает *Timer1* отвечающий за событие скрытия курсора.

private void ShowCursor()

{

Timer1.Start();

this.ClearValue(FrameworkElement.CursorProperty);

}

Теперь определим события окна **Window**. Первым станет событие *PrieviewMouseLeftButtonDown* отвечающее за нажатие левой кнопки мышки в окне формы:

private void Window\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

ShowCursor();

Point CurrentPoint = e.GetPosition(this);

TranslateTransform Transform = new TranslateTransform(CurrentPoint.X, CurrentPoint.Y);

//this.ClickCanvas.RenderTransform = Transform;

}

Закомментированная строчка относится к ещё не определённой нами функциональности (а именно к анимации «нажатия», о которой ниже). Пока что поясним, почему выбрана группа методов *PreviewMouse…* вместо обычных методов *Mouse…*. Разница в уровне восприятия события. В первом события **MouseLeftButtonDown** отслеживаются нажатия только в самой форме, если же нажатие было совершено в каком-либо элемента, событие не возникнет.

Событие **PreviewMouseMove** формы:

private void Window\_PreviewMouseMove(object sender, MouseEventArgs e)

{

ShowCursor();

}

События *Click* всех кнопок:

private void Back\_Click(object sender, RoutedEventArgs e)

{

NewPresentation.GoBack();

}

private void Next\_Click(object sender, RoutedEventArgs e)

{

NewPresentation.GoNext();

}

private void Home\_Click(object sender, RoutedEventArgs e)

{

NewPresentation.GoHome();

}

private void Auto\_Click(object sender, RoutedEventArgs e)

{

if (AutoClick == false)

{

AutoClick = true;

NewPresentation.GoAuto = true;

Timer2.Start();

}

else

{

AutoClick = false;

NewPresentation.GoAuto = false;

Timer2.Stop();

}

if (!NewPresentation.GoAuto)

{

LAuto.Visibility = Visibility.Hidden;

EAuto.Visibility = Visibility.Hidden;

}

else

{

LAuto.Visibility = Visibility.Visible;

EAuto.Visibility = Visibility.Visible;

}

}

Кнопки *Назад*, *Домой* и *Вперёд* вызывают методы класса *Presentation* и работают просто. Если есть слайд до текущего слайда, *Назад* отображается активной и её можно нажать, будет выбран предыдущий слайд из списка, определяющего порядок показа. Если нажата кнопка *Домой*, происходит вызов самого первого слайда. Нажатие кнопки Вперёд (если активна), переводит отображение на следующий слайд.

Работа кнопки *Авто* заключается в следующем: проверяем состояние (была нажата или нет). Если нажата, передаём параметр нажатия в класс *Presentation*, активируем таймер или останавливаем, и затем показываем или скрываем элементы на главной формы, отображающие состояние нажатия кнопки (эллипс и текст «А»). Передачу параметра в свойство можно и не делать. Но лучше всё-таки передавать ключевые параметры формы чтобы можно было получить их в любом месте приложения.

Событие **KeyDown** (нажатие клавиши) окна формы. Ловим нажатия клваш стрелок (лево, право) и **Esc**:

private void Window\_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)

{

if (e.Key == Key.Left)

{

NewPresentation.GoBack();

e.Handled = true;

}

else if (e.Key == Key.Right)

{

NewPresentation.GoNext();

e.Handled = true;

}

else if (e.Key == Key.Escape)

{

Application.Current.Shutdown();

e.Handled = true;

}

}

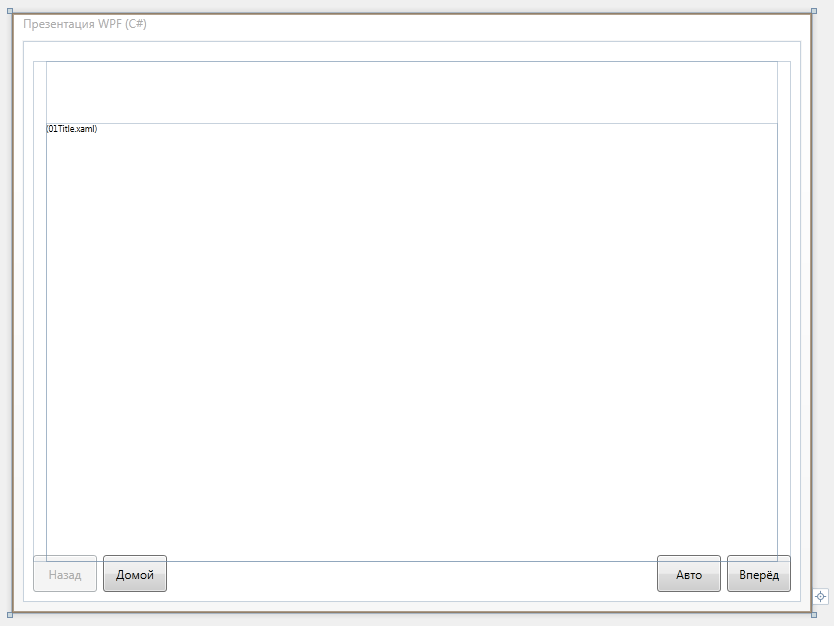


Рис. 3. 3. Модифицированная форма приложения

Компилируем, проверяем работоспособность. Попробуем изменить какой-либо слайд и добавить на него элементов, а также изменить свойство *Title* для слайда (не забываем что открывается тот слайд, который мы редактировали последним, полезно когда в данный момент нам нужно посмотреть как выглядит слайд «в живую»; если не хотим такую функциональность, выключаем как написано в комментариях к коду)

**4. Модификация приложения Windows Foundation Presentation: работа с решением в Expression Blend**

Всё. Код теперь можно не менять, у нас есть готовый каркас для презентации. Нам осталось только облагородить кнопки и добавить немного «красоты» в наш каркас. Когда будет готово и это, можно приступать к заполнению приложения слайдами. Процесс заполнения слайдов практически напоминает аналогичную работу в Microsoft Office PowerPoint. Если создавать слайды в Visual Studio то проблем с заполнением не возникнет. Элементов достаточно, возможностей много. Для добавления слайдов используем добавление новых страниц (*Page*), а занесения их в порядок вывода, добавляем полное имя XAML-файла в переменную **slides** класса *Presentation*.

Для получения по настоящему красивых вещей лучше использовать предназначенные для этого инструменты. При создании полноценной презентации необходимо знание английского хотя бы на базовом уровне для работы в Expression Blend 4 (эта среда пока что не доступна на русском языке). Небольшой опыт работы в этой мощнейшей дизайнерской среде конечно тоже необходим.

Важно отметить, что Expression Blend 4 поддерживает редактирование кода С# и добавление новых страниц. Фактически это замена среды разработки Visual Studio (в ней даже можно компилировать приложение).

Этого нам будет достаточно.

Откроем наше решение в Microsoft Expression Blend 4 (). Для этого запустим приложение, откроется начальное пустое окно с запросом действия:

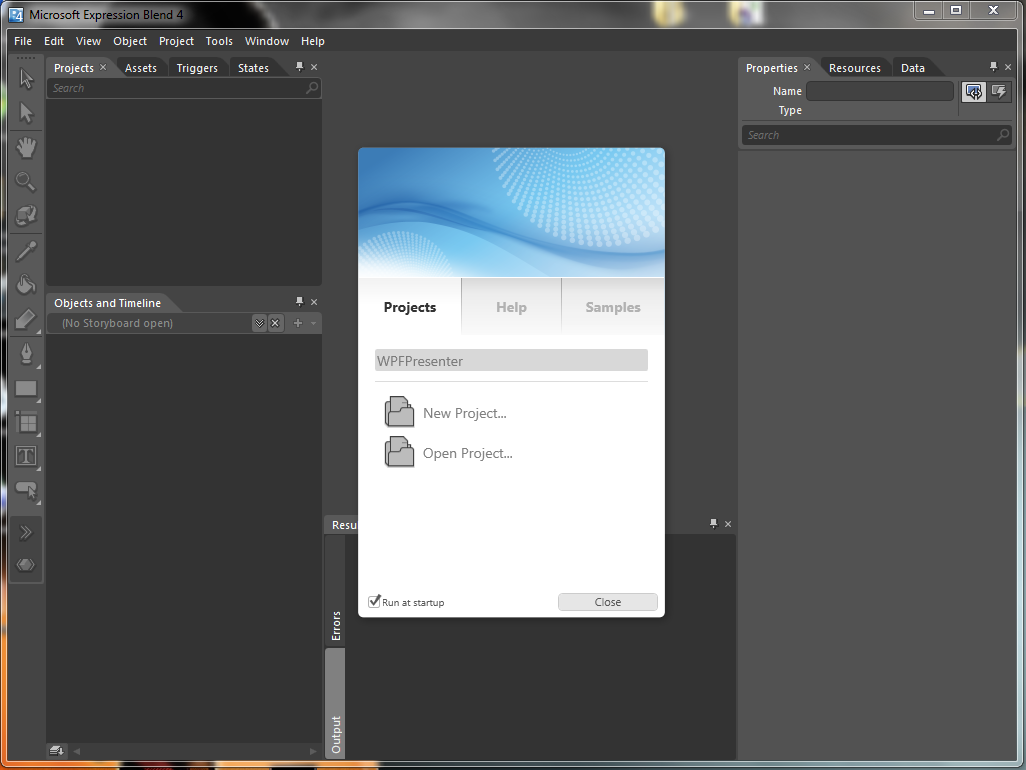


Рис. 4. 1. Окно Microsoft Expression Blend 4 с запросом действия

Выбираем **Open Project…**: выберем наше решение (**LWP09WPF02.csproj**), после чего оно будет загружено:

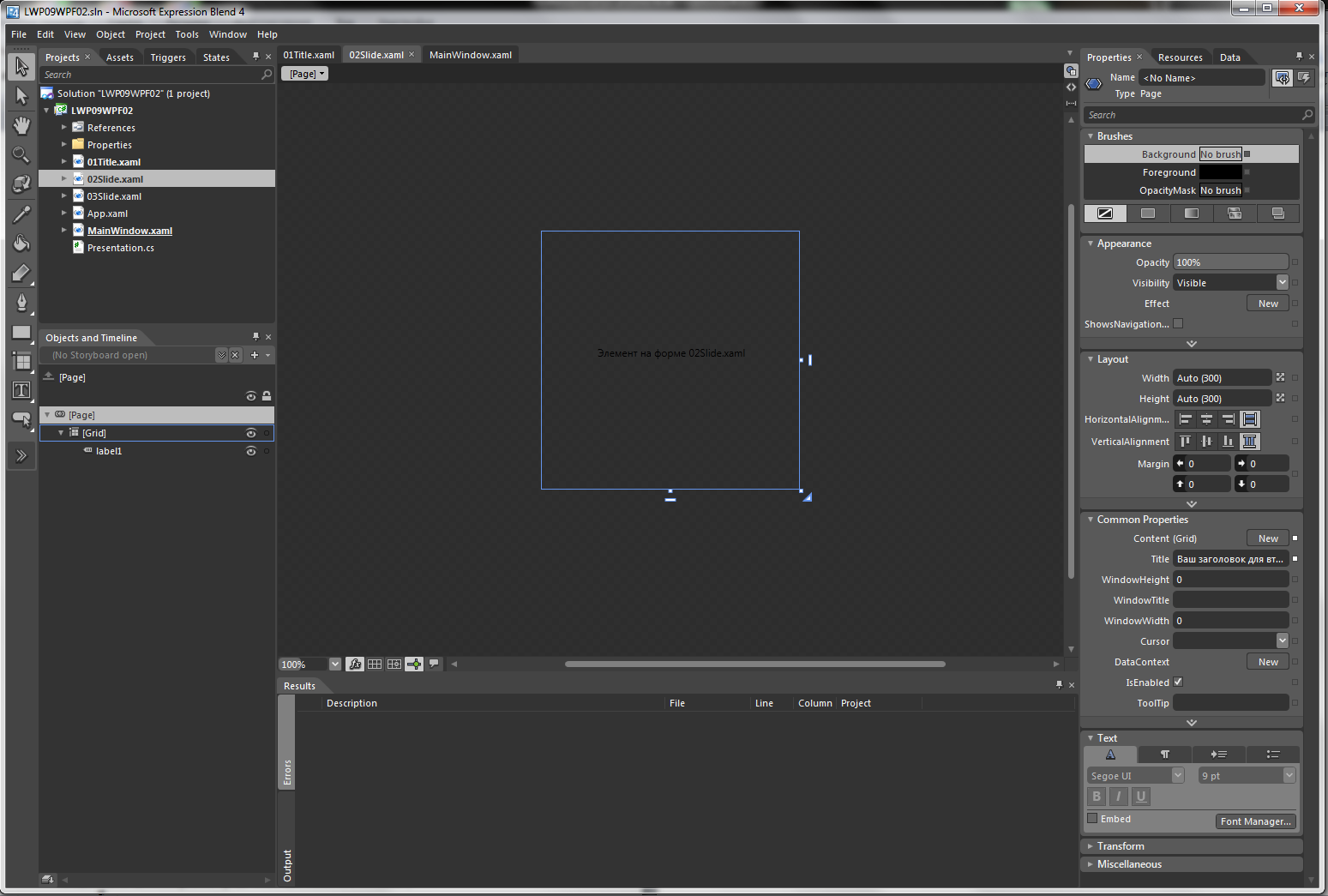


Рис. 4. 2. Загруженное решение презентации (вид на файл *02Slide.xaml*)

Как видим, всё напоминает среды разработки Visual Studio. Аналог обозревателя решений находится слева вверху, справа всю область занял местный аналог свойств выбранного элемента. Слева внизу окно доступных в файле XAML объектов и «линия времени» для анимации. В центре внизу, привычное окно вывода (со вкладками **Errors** и **Output**). Вверху над окном представления (конструктора по сути) находятся вкладки для переключения между выбранным файлов XAML.

Нажмём **F5** (**Project** -> **Run Project**), увидим сообщения выводы в нижнем окне И запустится наше приложение (если не будет ошибок).

Как добавить новый слайд (страницу)? Как и в среде разработки, парой действий. **Project** -> **Add New Item**… (**Ctrl**+**N**).

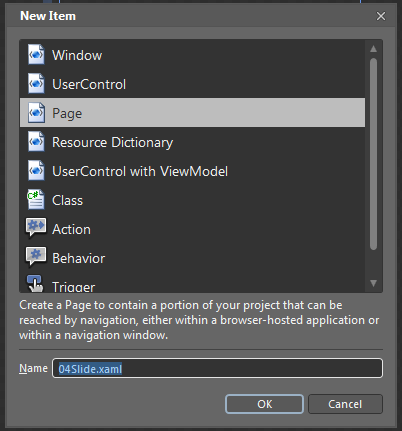


Рис. 4. 3. Окно **New Item**

Добавим новую страницу (*Page*) с именем **Name**: **04Slide.xaml**. И внесём ей в код:

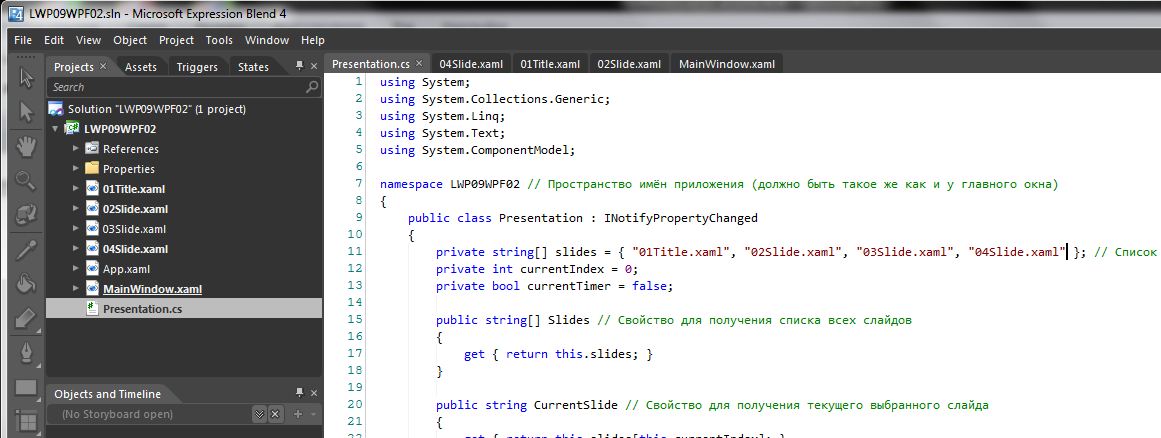


Рис. 4. 4. Редактируем код в Expression Blend

Расставим элементы в главном на слайде *01Title.xaml*:

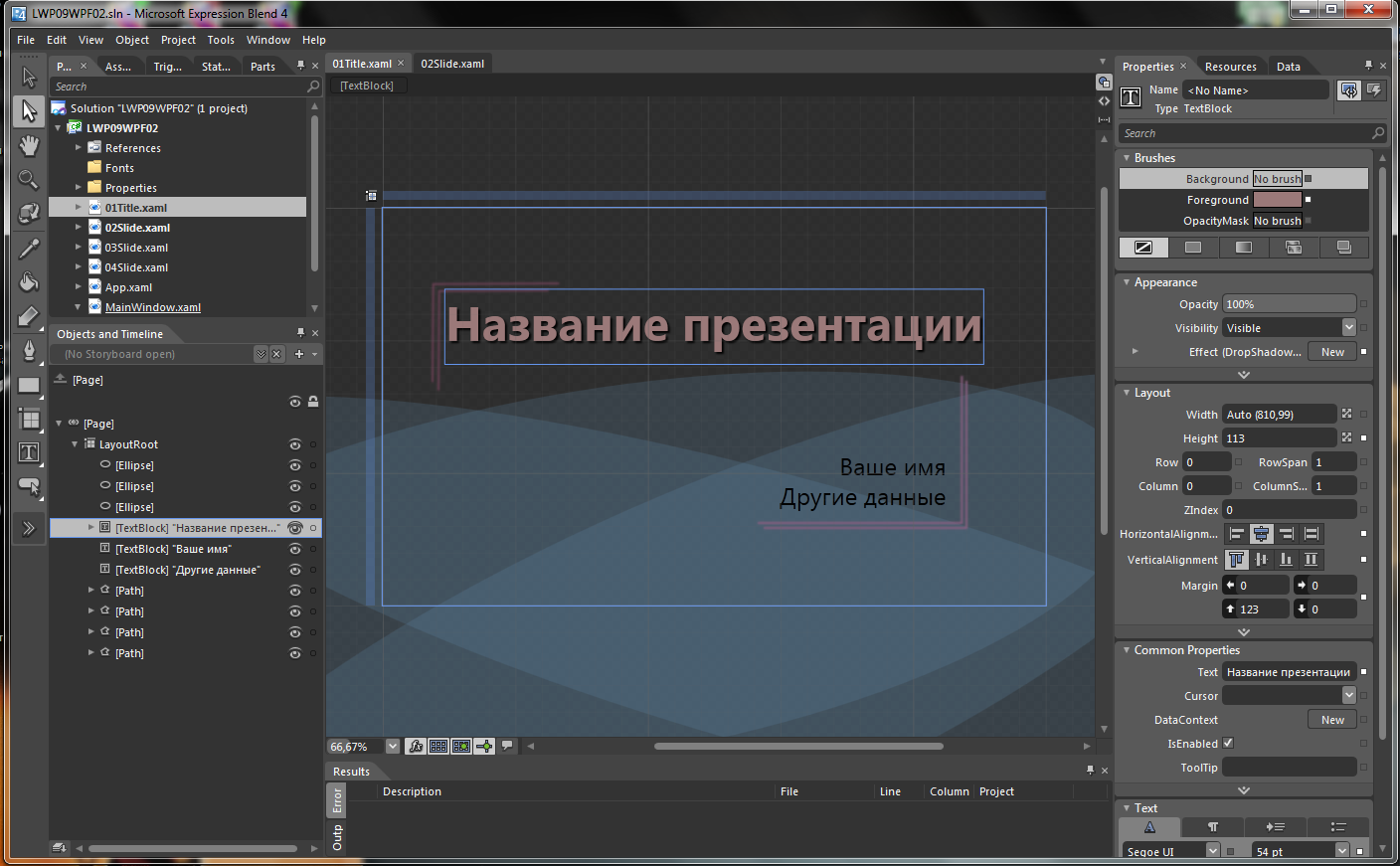
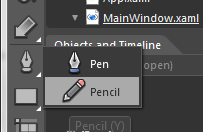


Рис. 4. 5. Расстановка элементов на первом слайде

«Рисовать» можно при помощи боковой панели слева. Иноки понятные. Если нужно добавить элемент, которого нет изначально на боковой панели, можно поискать в контекстном меню кнопки (нужно правой кнопкой мышки нажать на кнопку боковой панели). Например, выбор инструмента рисования **Pencil** (**Y**):



Если элемента нет в контекстном меню кнопки, можно поискать его из библиотеки доступных элементов. Нажимаем на боковой панели на символ «кавычек вправо»:

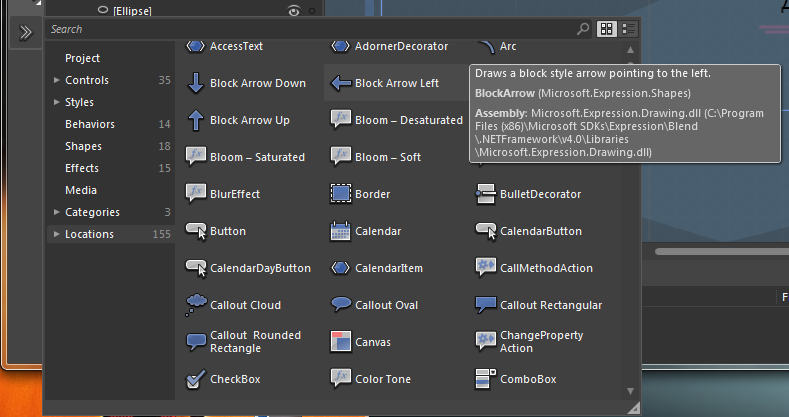


Рис. 4. 6. Выбор элемента из библиотеки элементов

В данной работы для кнопок были «собраны» новые стили. Для создания стиля на подобие элемента можно нажать правой кнопкой мыши объект формы в **Objects and Timeline** -> **Edit Template** -> **Create Empty…**. Если нужно скопировать конкретный стиль на основе выбранного элемента (например *Button*), жмём на **Create Copy…**.

Окно, всплывающие при создании на основе кнопки *Button*:

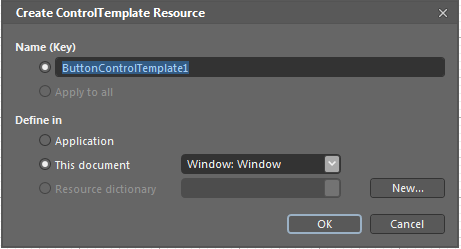
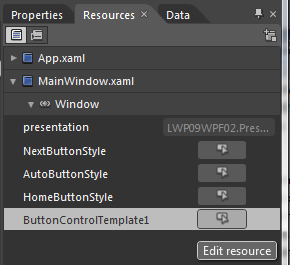


Рис. 4. 7. Создание чистого стиля на основе кнопки

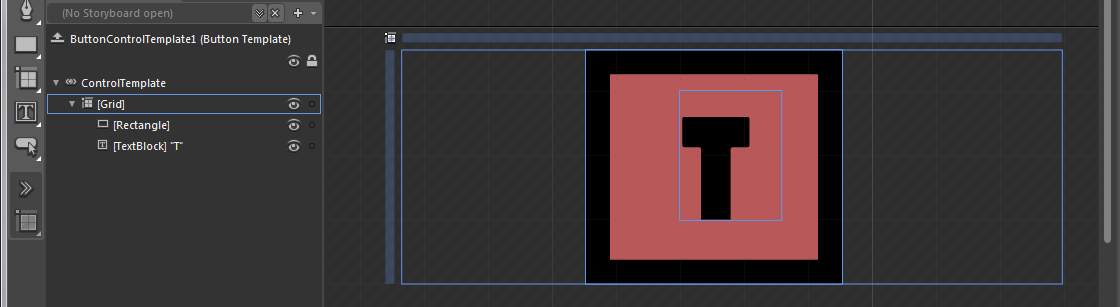
Далее даём стилю имя, после чего новый стиль должен появиться в окне ресурсов справа (**Resources**). Выбираем тот файл, куда добавили ресурс, далее новый стиль и жмём на **Edit resource**.



Далее размещаем в месте под элемент (чёрная граница) сетку Grid:



И после этого внутри этой сетки рисуем что угодно. Например такой шаблон (**Rectangle** и **Label**):



Теперь, сохраним решение. Далее можно собрать события наведения курсора на кнопку и событие нажатия. Для этого выделяем нужный элемент кнопки, для которой хотим менять значения при «свершении» события и переходим на вкладку **Triggres** и жмём на **+Property**:

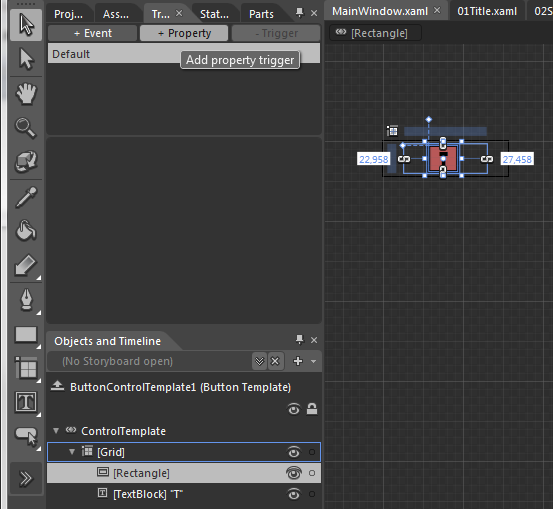


Рис. 4. 8. Добавление *Property* для стиля кнопки, элемента *Rectangle*

Появилось новое свойство **Activated when**. Оно отвечает на вопрос: «куда, когда и как» активировать событие. Выберем в Activated when: target-element (можно выбрать и rectangle), после «точки» выберем **IsMouseOver**, а после равно впишем **True**. В поле конструктора элемента мы видим надпись: «Событие = Параметр запись триггера включена».

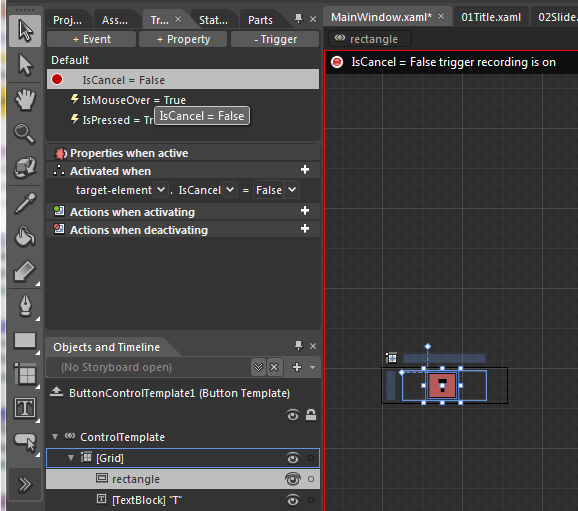


Рис. 4. 9. Запись триггера

В этом режиме меняем например цвет заполнения прямоугольника:

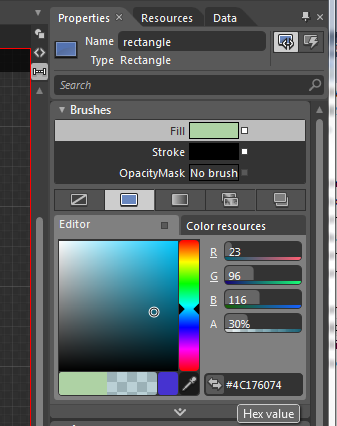


Рис. 4. 10. Записываем изменение цвета заполнения прямоугольника для события наведения мышки на элемент

Добавим ещё одно *Property* и на этот раз запишем изменение цвета прямоугольника для события **IsPressed** = **True**:

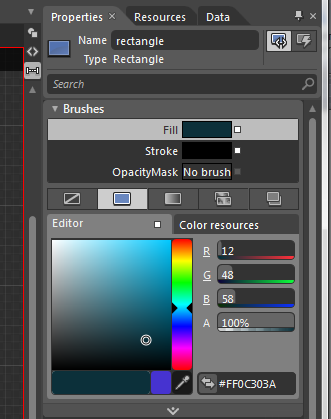
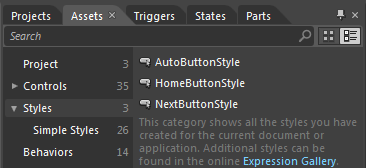
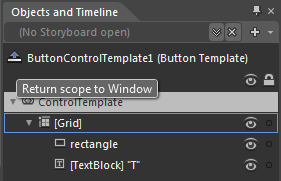


Рис. 4. 11. Записываем изменение цвета заполнения прямоугольника для события нажатия на элемент

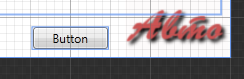
Всё. Наш элемент можно вставлять на форму. Для этого можно просто перетащить его как новую кнопку из окна ресурсов справа или если был создан стиль (Style), то из окна доступных стилей хранящихся в файлах решения (на рисунке ниже отображены все доступные стили):



Для добавления нашей кнопки из она ресурсов справа, сначало надо вернуть на форму. Для этого в окне Objects and Timiline ждмём на символ «стрелочки вверх»:



Далее размещаем наш новый элемент на форме перетаскиванием, либо можем применить стиль к свеже созданной или уже существующей кнопки. Размещаем новую кнопку (вкладка **Assets** -> **Controls** -> *Button*):



И применяем стиль к этой кнопке (выделяем кнопку в окне добавленных элементов формы, далее жмём правую кнопку мыши, ищем *Edit Template* -> **Apply Resource** -> **ButtonControlTemplate1**:

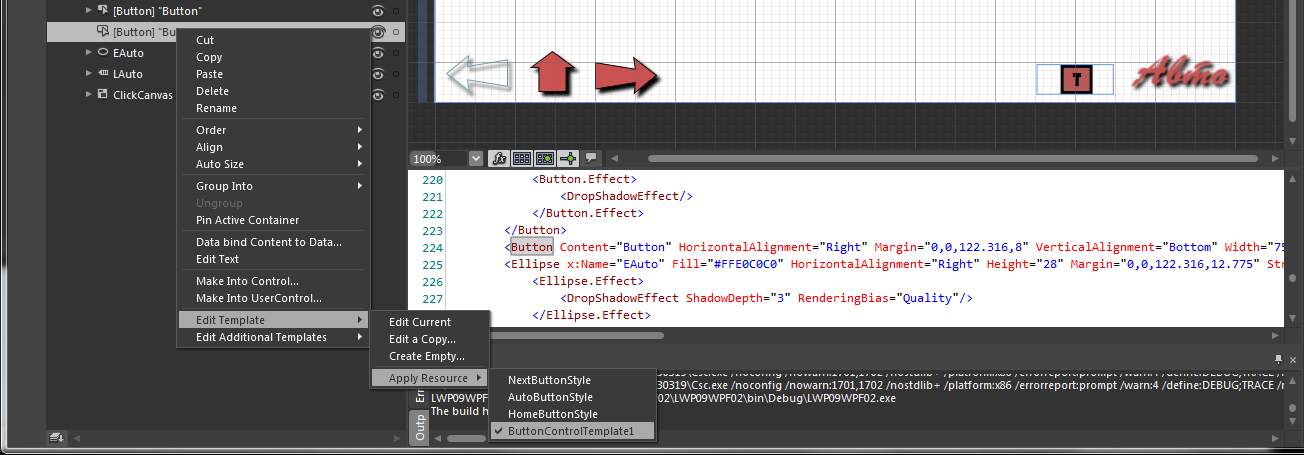


Рис. 4. 12. Применение стиля к новой кнопке

Если всё сделано правильно, кнопка будет корректно менять цвет при наведении мышки и нажатии на эту кнопку. Обработчик события добавляется либо в среде разработки (двойным нажатием), либо ручным редактированием кнопки в редакторе XAML. Если редактор не вызван, выбираем любой элемент на любой форме, далее нажимаем правую кнопку мышки на этом элементе и выбираем в раскрывающемся списке **View XAML**.

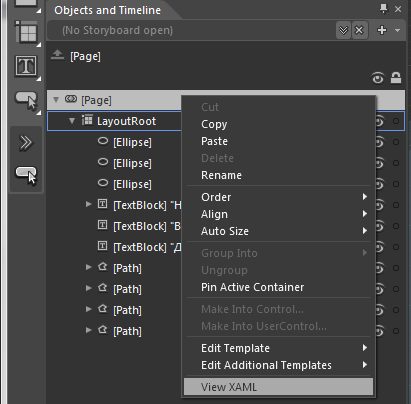


Рис. 4. 13. Открытие редактора XAML

Редактор откроется под окном конструктора формы.

Expression Blend также предоставляет возможности создания анимации. Как уже говорилось, в коде была добавлена строка для элемента **ClickCanvas** организующая анимацию «нажатия» левой кнопки мышки на любом месте. Добавление анимации выходит за рамки данной лабораторной работы (исходный код будет приведён), потому опишем как можно просмотреть анимационный элемент. Выберем *ClickCanvas* в списке доступных элементов формы *MainWindow.xaml*, далее в строке с текстом **(No Storyboard open)** нажмём на двойнуу стрелку вниз:

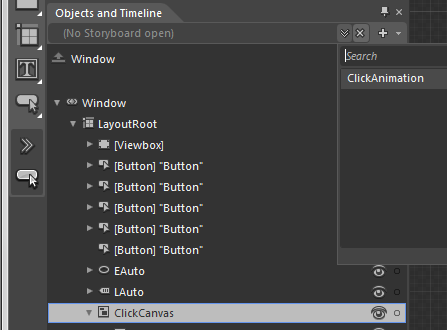


Рис. 4. 14. Доступная *Storyboard* для элемента *ClickCanvas*

Далее увидим открывшуюся полоску для управления анимацией и сам элемент в конструкторе формы. Анимацию (кадры) можно посмотреть в окне элементов формы):

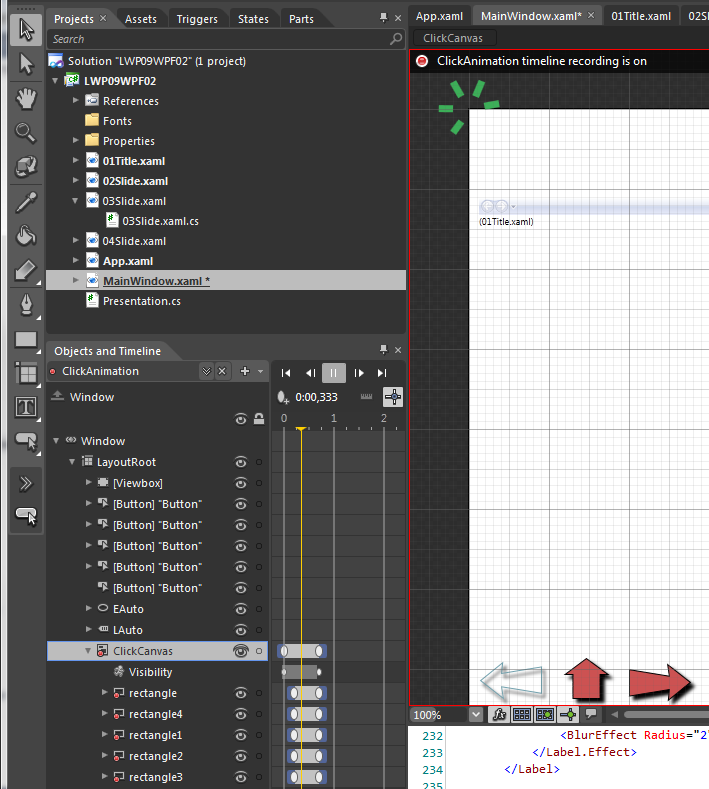
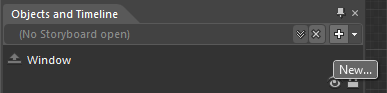


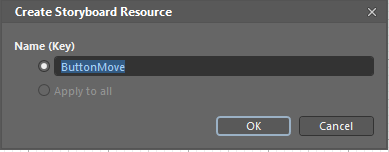
Рис. 4. 15. Выполнение анимационного элемента (текущий кадр указан жёлтой полосой)

Единственное что можно сказать про анимацию в целом. Запись действия происходит точно также как запись привязки *Property* (рисунок 4. 15, красная рамка).

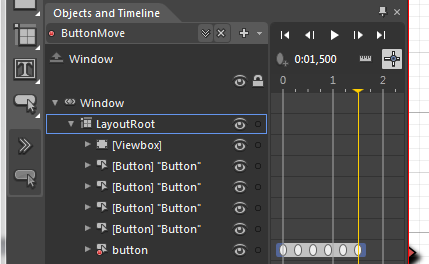
Можем проверить это на простом примере. Выделим любую кнопку, которую хотим «записать». Нажмём на «плюсик» правее от надписи *(No Strotyboar open)*:



Откроется окно добавления новой «истории» для записи. Впишем имя **ButtonMove**:



**ОК**. Перемещаем кнопку влево на несколько пикселей. Перемещаем жёлтую линию «временной зоны». Снова перемещаем кнопку влево на несколько пикселей… И так далее. У нас получится «дорожка» из положений для кнопки (можно и другие элементы двигать или менять, в режими записиь всё будет записано):



Всё. Теперь раскомментируем код для анимации в файле *MainWindows.xaml.cs* (в среде разработки Visual Studio или при помощи Expression Blend отредактируем файл кода):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO; // Для работы с классом File

using System.Windows.Threading; // Для создания таймеров

namespace LWP09WPF02

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private Presentation NewPresentation; // Создаём экземпляр нашего класса Presenation

private DispatcherTimer Timer1;

private DispatcherTimer Timer2;

private bool AutoClick;

public MainWindow()

{

InitializeComponent();

NewPresentation = (Presentation)this.FindResource("presentation");

AutoClick = false;

// После перезапуска приложения, встартует слайд, дата изменения которого наиболее поздняя (дата изменения страницы XAML)

int IndexLastWritten = 0;

DateTime latestDateTimeWritten = DateTime.MinValue;

for (int i = 0; i < NewPresentation.Slides.Length; i++) // Проходим по общему числу слайдов (через наш собственный класс и метод)

{

String Slide = NewPresentation.Slides[i];

DateTime DateLastWritten = File.GetLastWriteTime(@"..\..\" + Slide); // Узнаём дату из каталога проекта по файлам слайдов

if (DateLastWritten.CompareTo(latestDateTimeWritten) > 0)

{

latestDateTimeWritten = DateLastWritten;

IndexLastWritten = i;

}

}

NewPresentation.CurrentIndex = IndexLastWritten;

//presentation.CurrentIndex = 0; // Если нам нужно, можем всегда стартовать с первой страницы

// Создаём таймеры, которые обрабатывают исчезновение мышки после 5 секунд и перелистывание слайда через 15 секунд

Timer1 = new DispatcherTimer();

Timer1.Interval = TimeSpan.FromSeconds(5);

Timer1.Tick += new EventHandler(Timer1\_Tick);

Timer1.Start();

Timer2 = new DispatcherTimer();

Timer2.Interval = TimeSpan.FromSeconds(15);

Timer2.Tick += new EventHandler(Timer2\_Tick);

}

private void Timer1\_Tick(object sender, EventArgs e)

{

this.Cursor = Cursors.None; // Убираем курсор мышки

}

private void Timer2\_Tick(object sender, EventArgs e)

{

NewPresentation.GoNext();

}

private void Window\_KeyDown(object sender, System.Windows.Input.KeyEventArgs e)

{

if (e.Key == Key.Left)

{

NewPresentation.GoBack();

e.Handled = true;

}

else if (e.Key == Key.Right)

{

NewPresentation.GoNext();

e.Handled = true;

}

else if (e.Key == Key.Escape)

{

Application.Current.Shutdown();

e.Handled = true;

}

}

private void ShowCursor()

{

Timer1.Start();

this.ClearValue(FrameworkElement.CursorProperty);

}

private void Window\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

ShowCursor();

Point CurrentPoint = e.GetPosition(this);

TranslateTransform Transform = new TranslateTransform(CurrentPoint.X, CurrentPoint.Y);

this.ClickCanvas.RenderTransform = Transform;

}

private void Window\_PreviewMouseMove(object sender, MouseEventArgs e)

{

ShowCursor();

}

private void Back\_Click(object sender, RoutedEventArgs e)

{

NewPresentation.GoBack();

}

private void Next\_Click(object sender, RoutedEventArgs e)

{

NewPresentation.GoNext();

}

private void Home\_Click(object sender, RoutedEventArgs e)

{

NewPresentation.GoHome();

}

private void Auto\_Click(object sender, RoutedEventArgs e)

{

if (AutoClick == false)

{

AutoClick = true;

NewPresentation.GoAuto = true;

Timer2.Start();

}

else

{

AutoClick = false;

NewPresentation.GoAuto = false;

Timer2.Stop();

}

if (!NewPresentation.GoAuto)

{

LAuto.Visibility = Visibility.Hidden;

EAuto.Visibility = Visibility.Hidden;

}

else

{

LAuto.Visibility = Visibility.Visible;

EAuto.Visibility = Visibility.Visible;

}

}

}

}

Раскомментированная строка:

this.ClickCanvas.RenderTransform = Transform;

Обновим главным файл *App.xaml* для добавление ресурсов анимации (цветов отображения):

<Application x:Class="LWP09WPF02.App"

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

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

StartupUri="MainWindow.xaml">

<Application.Resources>

<SolidColorBrush x:Key="MouseClickEffectColor" Color="#FFFFFFFF"/>

<SolidColorBrush x:Key="OrangeClick" Color="#FF41DE67"/>

</Application.Resources>

</Application>

И обновим код MainWindow.xaml для добавления всей описанной функциональности, включая анимацию:

<Window

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

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

xmlns:LWP09WPF02="clr-namespace:LWP09WPF02"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d" x:Class="LWP09WPF02.MainWindow"

Title="Презентация WPF (C#)" Height="600" Width="800" ResizeMode="NoResize"

WindowState="Maximized" WindowStyle="None" x:Name="Window"

KeyDown="Window\_KeyDown" PreviewMouseMove="Window\_PreviewMouseMove"

PreviewMouseLeftButtonDown="Window\_PreviewMouseLeftButtonDown">

<Window.Resources>

<LWP09WPF02:Presentation x:Key="presentation"/>

<Style x:Key="NextButtonStyle" BasedOn="{x:Null}" TargetType="{x:Type Button}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Path Stretch="Fill" Stroke="Black" x:Name="path" Width="61.908" Height="36.709" Data="M576.19695,27.913945 L576.08822,68.249537 C576.08822,68.249537 623.05612,58.293283 623.05612,58.293283 L621.74723,74.696597 647.68857,47.340314 621,20 623.05612,37.870199 C623.05612,37.870199 576.19695,27.913945 576.19695,27.913945 z" Fill="#FFCC5252" OpacityMask="Black"/>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Fill" TargetName="path" Value="#4C176074"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Fill" TargetName="path" Value="#FF0C303A"/>

</Trigger>

<Trigger Property="IsEnabled" Value="False">

<Setter Property="Fill" TargetName="path" Value="{x:Null}"/>

<Setter Property="Stroke" TargetName="path" Value="#FF72A1AE"/>

<Setter Property="StrokeThickness" TargetName="path" Value="1"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="AutoButtonStyle" BasedOn="{x:Null}" TargetType="{x:Type Button}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Label x:Name="label" Content="Авто" Height="Auto" Width="Auto" BorderBrush="{x:Null}" Background="#02FF0000" Foreground="#FFCC5252" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" HorizontalAlignment="Center" VerticalAlignment="Stretch" FontWeight="Bold" FontFamily="UkrainianBrushScript" FontSize="42.667" ToolTip="Автоматический показ слайдов">

<Label.Effect>

<BlurEffect Radius="2"/>

</Label.Effect>

</Label>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Foreground" TargetName="label" Value="#4C176074"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Foreground" TargetName="label" Value="#FF0C303A"/>

</Trigger>

<Trigger Property="IsEnabled" Value="False">

<Setter Property="Foreground" TargetName="label" Value="#FF72A1AE"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="HomeButtonStyle" BasedOn="{x:Null}" TargetType="{x:Type Button}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Path Stretch="Fill" Stroke="Black" x:Name="path" Width="44.15" Height="41" Data="M589.13955,33.208404 L598.54392,33.208406 C598.54392,33.208406 598.54392,69.969662 598.54392,69.969662 L630.46071,69.969662 630.46071,33.208404 639.86497,33.208404 614.37289,8.7009021 C614.37289,8.7009021 589.13955,33.208404 589.13955,33.208404 z" Fill="#FFCC5252" OpacityMask="Black" VerticalAlignment="Stretch"/>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Fill" TargetName="path" Value="#4C176074"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Fill" TargetName="path" Value="#FF0C303A"/>

</Trigger>

<Trigger Property="IsEnabled" Value="False">

<Setter Property="Fill" TargetName="path" Value="{x:Null}"/>

<Setter Property="Stroke" TargetName="path" Value="#FF72A1AE"/>

<Setter Property="StrokeThickness" TargetName="path" Value="1"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Storyboard x:Key="ClickAnimation">

<ObjectAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="ClickCanvas" Storyboard.TargetProperty="(UIElement.Visibility)">

<DiscreteObjectKeyFrame KeyTime="00:00:00" Value="{x:Static Visibility.Visible}"/>

<DiscreteObjectKeyFrame KeyTime="00:00:00.7000000">

<DiscreteObjectKeyFrame.Value>

<Visibility>Collapsed</Visibility>

</DiscreteObjectKeyFrame.Value>

</DiscreteObjectKeyFrame>

</ObjectAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[0].(ScaleTransform.ScaleX)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="4.749"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-5.623"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-0.082"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle3" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[0].(ScaleTransform.ScaleX)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="4.996"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle3" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="5.328"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle3" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-2.748"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle" Storyboard.TargetProperty="(UIElement.Opacity)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="1"/>

<SplineDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle4" Storyboard.TargetProperty="(UIElement.Opacity)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="1"/>

<SplineDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle1" Storyboard.TargetProperty="(UIElement.Opacity)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="1"/>

<SplineDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle2" Storyboard.TargetProperty="(UIElement.Opacity)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="1"/>

<SplineDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle3" Storyboard.TargetProperty="(UIElement.Opacity)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="1"/>

<SplineDoubleKeyFrame KeyTime="00:00:00.7000000" Value="0"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle4" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[0].(ScaleTransform.ScaleX)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="4.685"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle4" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-3.261"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle4" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="4.463"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle1" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[0].(ScaleTransform.ScaleX)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="5.573"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle1" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-3.388"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle1" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-5.964"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle2" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[0].(ScaleTransform.ScaleX)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="5.555"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle2" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="2.621"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames BeginTime="00:00:00" Storyboard.TargetName="rectangle2" Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)">

<SplineDoubleKeyFrame KeyTime="00:00:00.2000000" Value="-6.309"/>

</DoubleAnimationUsingKeyFrames>

</Storyboard>

<ControlTemplate x:Key="ButtonControlTemplate1" TargetType="{x:Type Button}">

<Grid Height="29.188" Width="82.416">

<Rectangle x:Name="rectangle" Fill="#FFB85858" Margin="22.958,0,27.458,0" Stroke="Black" StrokeThickness="3"/>

<TextBlock Margin="34.708,5.094,35.042,8" TextWrapping="Wrap" Text="T" FontFamily="Poplar Std" FontSize="18.667"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter

Property="IsEnabled" TargetName="rectangle" Value="True"/>

<Setter

Property="Fill" TargetName="rectangle" Value="#4C176074"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Fill" TargetName="rectangle" Value="#FF0C303A"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

<Storyboard x:Key="ButtonMove">

<DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.Y)" Storyboard.TargetName="button">

<EasingDoubleKeyFrame KeyTime="0" Value="-4.413"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.3" Value="-3.638"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.6" Value="-2.863"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.9" Value="-2.088"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.2" Value="-1.313"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5" Value="-0.538"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(UIElement.RenderTransform).(TransformGroup.Children)[3].(TranslateTransform.X)" Storyboard.TargetName="button">

<EasingDoubleKeyFrame KeyTime="0" Value="5"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.3" Value="-35"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.6" Value="-75"/>

<EasingDoubleKeyFrame KeyTime="0:0:0.9" Value="-115"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.2" Value="-150"/>

<EasingDoubleKeyFrame KeyTime="0:0:1.5" Value="-198"/>

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</Window.Resources>

<Window.Triggers>

<!-- Подключаем RoutedEvent и вытаскиваем событие нажатия левой кнопки мышки -->

<EventTrigger RoutedEvent="UIElement.PreviewMouseLeftButtonDown">

<BeginStoryboard Storyboard="{StaticResource ClickAnimation}"/>

</EventTrigger>

<EventTrigger RoutedEvent="FrameworkElement.Loaded">

<BeginStoryboard Storyboard="{StaticResource ButtonMove}"/>

</EventTrigger>

</Window.Triggers>

<Grid x:Name="LayoutRoot">

<Viewbox Margin="10,20,10,40" Stretch="Uniform">

<StackPanel>

<!-- Подставляем всегда: заголовок страницы -->

<TextBlock VerticalAlignment="Top" Height="84" FontFamily="Calibri" FontSize="65" FontWeight="Bold" Text="{Binding Content.Title, ElementName=Frame, Mode=Default}" TextAlignment="Center" TextWrapping="Wrap">

<TextBlock.Effect>

<BlurEffect/>

</TextBlock.Effect>

</TextBlock>

<!-- Подставляем всегда: содержимое слайда -->

<Frame Width="1000" Height="600" Source="{Binding CurrentSlide, Source={StaticResource presentation}}" x:Name="Frame" NavigationUIVisibility="Hidden" Background="{x:Null}" Focusable="False"/>

</StackPanel>

</Viewbox>

<!-- Кнопка Назад (перевёрнутая кнопка Вперёд) -->

<Button Style="{DynamicResource NextButtonStyle}" Width="63.684" Height="36.709" Content="Button" Click="Back\_Click" RenderTransformOrigin="0.5,0.5" HorizontalAlignment="Left" Margin="10,0,0,10" VerticalAlignment="Bottom" IsEnabled="{Binding CanGoBack, Source={StaticResource presentation}}" Cursor="Cross" ToolTip="Назад">

<Button.Effect>

<DropShadowEffect Direction="490"/>

</Button.Effect>

<Button.RenderTransform>

<ScaleTransform ScaleX="-1" ScaleY="-1"/>

</Button.RenderTransform>

</Button>

<!-- Кнопка Вперёд -->

<Button HorizontalAlignment="Left" Margin="159,0,0,8" Style="{DynamicResource NextButtonStyle}" VerticalAlignment="Bottom" Width="63.684" Height="36.709" Content="Button" Click="Next\_Click" IsEnabled="{Binding CanGoNext, Source={StaticResource presentation}}" Cursor="Cross" ToolTip="Вперёд">

<Button.Effect>

<DropShadowEffect/>

</Button.Effect>

</Button>

<Button Content="Button" Style="{DynamicResource HomeButtonStyle}" Width="75" HorizontalAlignment="Left" Margin="80,0,0,10" VerticalAlignment="Bottom" ToolTip="В начало" Cursor="Cross" Click="Home\_Click">

<Button.Effect>

<DropShadowEffect/>

</Button.Effect>

</Button>

<Button Content="Button" HorizontalAlignment="Right" Margin="0,0,0,-0.986" Style="{DynamicResource AutoButtonStyle}" VerticalAlignment="Bottom" Width="107.378" Cursor="Cross" Click="Auto\_Click">

<Button.Effect>

<DropShadowEffect/>

</Button.Effect>

</Button>

<Button x:Name="button" Content="Button" HorizontalAlignment="Right" Margin="0,0,141,11.587" VerticalAlignment="Bottom" Width="75" Template="{DynamicResource ButtonControlTemplate1}" RenderTransformOrigin="0.5,0.5">

<Button.RenderTransform>

<TransformGroup>

<ScaleTransform/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Button.RenderTransform>

</Button>

<Ellipse x:Name="EAuto" Fill="#FFE0C0C0" HorizontalAlignment="Right" Height="28" Margin="0,0,122.316,12.775" Stroke="Black" VerticalAlignment="Bottom" Width="28" Visibility="Hidden">

<Ellipse.Effect>

<DropShadowEffect ShadowDepth="3" RenderingBias="Quality"/>

</Ellipse.Effect>

</Ellipse>

<Label x:Name="LAuto" Content="A" HorizontalAlignment="Right" Height="30" Margin="0,0,107.378,14.709" VerticalAlignment="Bottom" Width="40" Background="#00E07878" FontFamily="Trajan Pro" FontSize="18.667" Foreground="#FFB82913" Visibility="Hidden">

<Label.Effect>

<BlurEffect Radius="2"/>

</Label.Effect>

</Label>

<Canvas x:Name="ClickCanvas" Visibility="Collapsed">

<Rectangle RenderTransformOrigin="0.5,0.5" x:Name="rectangle" Width="3" Height="7" Fill="{DynamicResource OrangeClick}" Stroke="#FF000000" StrokeThickness="0" Canvas.Left="-19" Canvas.Top="-3.75">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="0.836"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="3" Height="7" Fill="{DynamicResource OrangeClick}" Stroke="#FF000000" StrokeThickness="0" RenderTransformOrigin="0.5,0.5" x:Name="rectangle4" Canvas.Left="-10" Canvas.Top="10.125">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="-53.842"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="3" Height="7" Fill="{DynamicResource OrangeClick}" Stroke="#FF000000" StrokeThickness="0" RenderTransformOrigin="0.5,0.5" x:Name="rectangle1" Canvas.Left="-10" Canvas.Top="-17.375">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="-119.602"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="3" Height="7" Fill="{DynamicResource OrangeClick}" Stroke="#FF000000" StrokeThickness="0" RenderTransformOrigin="0.5,0.5" x:Name="rectangle2" Canvas.Left="5.625" Canvas.Top="-17.375">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="-67.444"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Rectangle Width="3" Height="7" Fill="{DynamicResource OrangeClick}" Stroke="#FF000000" StrokeThickness="0" RenderTransformOrigin="0.5,0.5" x:Name="rectangle3" Canvas.Left="15.875" Canvas.Top="-4.875">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="-10.339"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

</Canvas>

</Grid>

</Window>

Файл первого слайда *01Title.xaml* сформировали таким образом:

<Page x:Class="LWP09WPF02.\_01Title"

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

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

FlowDirection="LeftToRight"

Width="1000" Height="600">

<Grid x:Name="LayoutRoot">

<Ellipse RenderTransformOrigin="0.5,0.5" Fill="#33628FB2" StrokeThickness="3" Margin="-494,274.5,-148,-230.5">

<Ellipse.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="-9.455"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Ellipse.RenderTransform>

</Ellipse>

<Ellipse RenderTransformOrigin="0.5,0.5" Fill="#33628FB2" Stroke="{x:Null}" StrokeThickness="3" Margin="-446.5,0,-401.5,-345.5" VerticalAlignment="Bottom" Height="513.5">

<Ellipse.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="0"/>

<RotateTransform Angle="-23.748"/>

<TranslateTransform X="0" Y="0"/>

</TransformGroup>

</Ellipse.RenderTransform>

</Ellipse>

<Ellipse RenderTransformOrigin="0.5,0.5" Fill="#33628FB2" Stroke="{x:Null}" StrokeThickness="3" Margin="-342,0,-598,-136.563" VerticalAlignment="Bottom" Height="424">

<Ellipse.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="1" ScaleY="1"/>

<SkewTransform AngleX="0" AngleY="14.911"/>

<RotateTransform Angle="0"/>

<TranslateTransform X="0" Y="93.734"/>

</TransformGroup>

</Ellipse.RenderTransform>

</Ellipse>

<TextBlock HorizontalAlignment="Center" Margin="0,123,0,0" VerticalAlignment="Top" Height="113" FontSize="72" FontWeight="Bold" Text="Название презентации" TextAlignment="Center" TextWrapping="Wrap" Foreground="#FF9B7979">

<TextBlock.Effect>

<DropShadowEffect/>

</TextBlock.Effect>

</TextBlock>

<TextBlock Margin="499,0,150.213,170" VerticalAlignment="Bottom" Height="65" FontSize="36" Text="Ваше имя" TextAlignment="Right" TextWrapping="Wrap"/>

<TextBlock Margin="499,0,150.213,127" VerticalAlignment="Bottom" Height="63" FontSize="36" Text="Другие данные" TextAlignment="Right" TextWrapping="Wrap"/>

<Path Fill="{x:Null}" Stretch="Fill" Stroke="#FF895D80" StrokeThickness="3" HorizontalAlignment="Right" Margin="0,251.5,125.5,123.5" Width="311" Data="M875,253 L875,477 565,477">

<Path.Effect>

<BlurEffect/>

</Path.Effect>

</Path>

<Path Fill="{x:Null}" Stretch="Fill" Stroke="#FF895D80" StrokeThickness="3" Width="309" Data="M872.98701,253 L872.98701,477 565,477" HorizontalAlignment="Right" Margin="0,259.5,117.5,115.5">

<Path.Effect>

<BlurEffect/>

</Path.Effect>

</Path>

<Path Data="M80.5,237.5 L81.5,87.5 246.5,86.5" Fill="{x:Null}" HorizontalAlignment="Left" Height="154" Margin="82.5,123,0,0" Stretch="Fill" Stroke="#FF683F52" StrokeThickness="3" VerticalAlignment="Top" Width="169">

<Path.Effect>

<BlurEffect/>

</Path.Effect>

</Path>

<Path Data="M72.499038,227.50034 L73.653041,79.509324 263.49905,78.500323" Fill="{x:Null}" HorizontalAlignment="Left" Height="152" Margin="73.5,113,0,0" Stretch="Fill" Stroke="#FF683F52" StrokeThickness="3" VerticalAlignment="Top" Width="194">

<Path.Effect>

<BlurEffect/>

</Path.Effect>

</Path>

</Grid>

</Page>

Готово. Приложение-презентацию можно запускать. Учитывая возможности, которые открываются при применении WPF, альтернатива PowerPoint получилась достойная.

**5. Завершающая часть**

Компилируем приложение (**Release**) и запускаем. Результат работы показан ниже (Рис. 5. 1):



Рис. 5. 1. Модифицированное приложение *Windows Foundation Presentation* (Зелёным анимирован щелчок по движущейся кнопки, включена автопрокрутка, отображается первый слайд *01Title.xaml*)

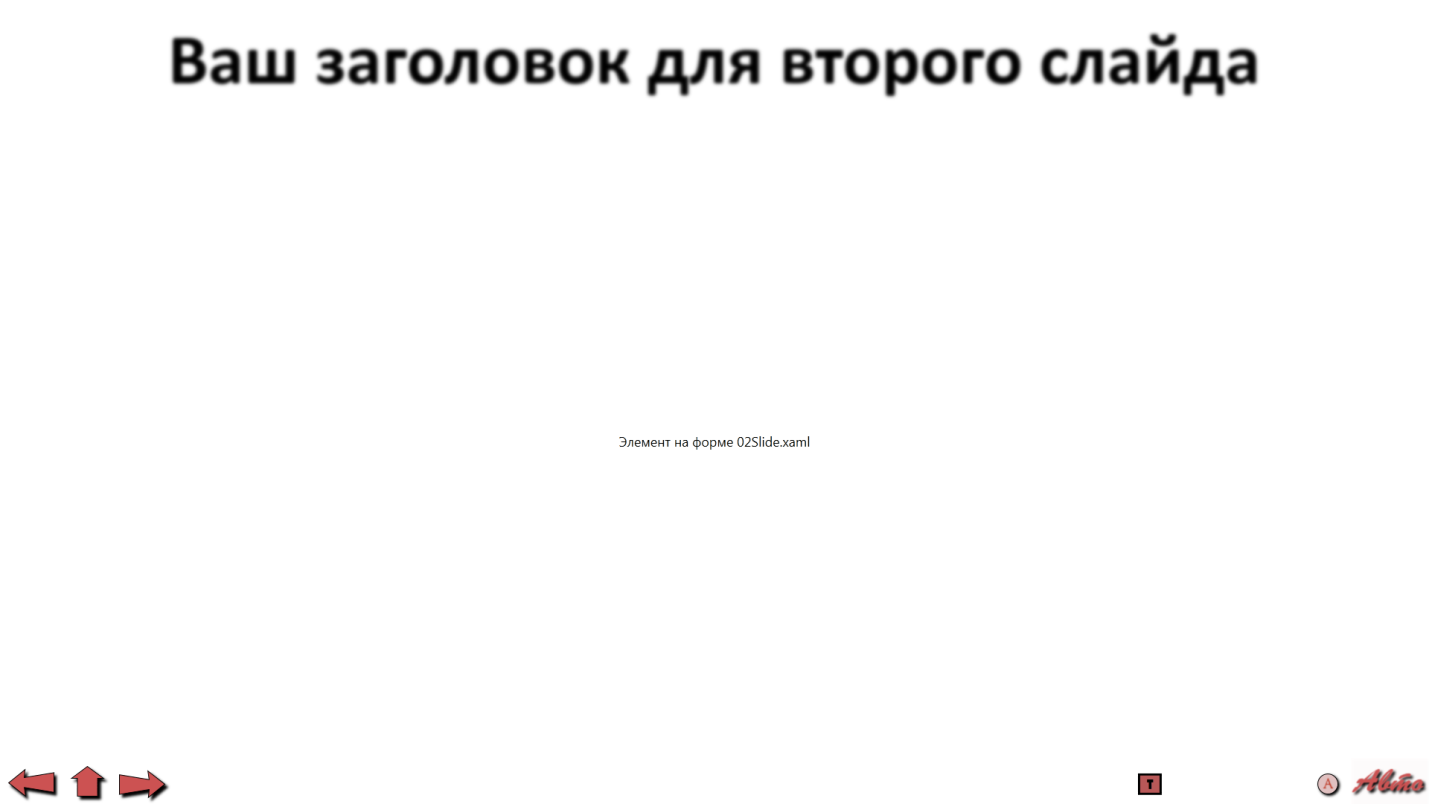


Рис. 5. 2. Модифицированное приложение *Windows Foundation Presentation* (слайд *02Slide.xaml*)