**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**



**ЗВІТ**

До лабораторної роботи №6

**На тему:** *«Створення мультимедіа засобами WPF»*

**З дисципліни:** *«Програмування мультимедійних систем»*

**Виконав:**

студ. гр. ПІ-31

*Портах Б.В.*

**Прийняв:**

Гасько Р.Т.

Львів – 2017

**Тема роботи:** Створення мультимедіа засобами WPF.

**Мета роботи:** Отримати практичні навики роботи з мультимедійними даними та анімованими об’єктами з використанням засобів WPF.

**Теоретичні відомості**

WPF є високорівневим об’єктно-орієнтованим функціональним шаром (framework), що дає змогу створювати двовимірні та тривимірні інтерфейси. За допомогою WPF можна створювати широкий спектр як автономних, так і веб-орієнтованих застосунків, що виконуються у браузері.

В основі WPF лежить векторна система візуалізації, що не залежить від роздільності пристрою виведення і створена з урахуванням можливостей сучасного графічного устаткування. WPF надає засоби для створення візуального інтерфейсу, включаючи мову XAML, елементи керування, макети, стилі, шаблони, двомірну і тривимірну графіку, анімацію, мультимедійні засоби.

XAML (Extensible Application Markup Language – розширювана мова розмітки застосунків) є мовою розмітки, яку використовують для створення екземплярів об’єктів .NET. Хоча мова XAML – це технологія, що може бути застосована до багатьох різних предметних областей, її головне призначення – конструювання інтерфейсів користувачів WPF. Документи XAML визначають розташування панелей, кнопок та інших елементів керування у вікнах застосунків WPF. Користувачеві не доводиться писати код XAML вручну – він просто використовує інструмент, що генерує необхідний код XAML.

**Завдання до лабораторної роботи**

*Варіант 7:*

Розробити форму з анімованими елементами керування (розкривними списками (ComboBox) і кнопками (Button)), яка видає у вікно перегляду анімовані текстові повідомлення про вибраний варіант. Анімація повинна виконуватись циклічно.

**Код програми**

Назва файлу: **MainWindow.xaml.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Media;

using System.Windows.Media.Animation;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Multimedia\_Lab6

{

public partial class MainWindow : Window

{

private readonly List<DoubleAnimation> \_transparencyAnimation = new List<DoubleAnimation>();

private readonly List<ColorAnimation> \_colorAnimation = new List<ColorAnimation>();

private readonly List<DoubleAnimation> \_sizeAnimation = new List<DoubleAnimation>();

private readonly List<DoubleAnimation> \_rotateAnimation = new List<DoubleAnimation>();

public MainWindow()

{

InitializeComponent();

}

private void ShowAnimation\_Button(object sender, RoutedEventArgs e)

{

DoubleAnimation tAnimation = null, sAnimation = null, rAnimation = null;

ColorAnimation cAnimation = null;

if (ComboBox1.SelectedIndex > -1 && ComboBox1.SelectedIndex < \_transparencyAnimation.Count)

tAnimation = \_transparencyAnimation[ComboBox1.SelectedIndex];

if (ComboBox2.SelectedIndex > -1 && ComboBox2.SelectedIndex < this.\_colorAnimation.Count)

cAnimation = \_colorAnimation[ComboBox2.SelectedIndex];

if (ComboBox3.SelectedIndex > -1 && ComboBox3.SelectedIndex < this.\_sizeAnimation.Count)

sAnimation = \_sizeAnimation[ComboBox3.SelectedIndex];

if (ComboBox4.SelectedIndex > -1 && ComboBox4.SelectedIndex < this.\_rotateAnimation.Count)

rAnimation = \_rotateAnimation[ComboBox4.SelectedIndex];

var AnimatedMessage = new AnimatedMessage(tAnimation, cAnimation, sAnimation, rAnimation);

AnimatedMessage.ShowDialog();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

DoubleAnimation da;

da = new DoubleAnimation(0.1, 1, new Duration(TimeSpan.FromSeconds(1)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_transparencyAnimation.Add(da);

da = new DoubleAnimation(0.1, 1, new Duration(TimeSpan.FromSeconds(2)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_transparencyAnimation.Add(da);

da = new DoubleAnimation(0.1, 1, new Duration(TimeSpan.FromSeconds(3)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_transparencyAnimation.Add(da);

da = new DoubleAnimation(0.1, 1, new Duration(TimeSpan.FromSeconds(4)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_transparencyAnimation.Add(da);

da = new DoubleAnimation(0.1, 1, new Duration(TimeSpan.FromSeconds(5)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_transparencyAnimation.Add(da);

ColorAnimation ca;

ca = new ColorAnimation(Colors.Black, Colors.DeepPink, new Duration(TimeSpan.FromSeconds(2)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_colorAnimation.Add(ca);

ca = new ColorAnimation(Colors.Black, Colors.Red, new Duration(TimeSpan.FromSeconds(2)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_colorAnimation.Add(ca);

ca = new ColorAnimation(Colors.Black, Colors.Green, new Duration(TimeSpan.FromSeconds(2)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_colorAnimation.Add(ca);

ca = new ColorAnimation(Colors.Black, Colors.Orange, new Duration(TimeSpan.FromSeconds(2)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_colorAnimation.Add(ca);

ca = new ColorAnimation(Colors.Black, Colors.Blue, new Duration(TimeSpan.FromSeconds(2)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_colorAnimation.Add(ca);

da = new DoubleAnimation(8, 16, new Duration(TimeSpan.FromSeconds(1)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_sizeAnimation.Add(da);

da = new DoubleAnimation(70, 120, new Duration(TimeSpan.FromSeconds(1)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_sizeAnimation.Add(da);

da = new DoubleAnimation(8, 200, new Duration(TimeSpan.FromSeconds(1)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_sizeAnimation.Add(da);

da = new DoubleAnimation(0, 30, new Duration(TimeSpan.FromSeconds(9)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_rotateAnimation.Add(da);

da = new DoubleAnimation(0, 180, new Duration(TimeSpan.FromSeconds(9)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_rotateAnimation.Add(da);

da = new DoubleAnimation(0, 360, new Duration(TimeSpan.FromSeconds(9)))

{

AutoReverse = true,

RepeatBehavior = RepeatBehavior.Forever

};

\_rotateAnimation.Add(da);

foreach (DoubleAnimation doubleAnimation in \_transparencyAnimation)

{

var tb = new TextBox { Text = "Opacity", IsReadOnly = true };

tb.BeginAnimation(OpacityProperty, doubleAnimation);

ComboBox1.Items.Add(tb);

}

ComboBox1.Items.Add(new TextBox { Text = " Nothing ", IsReadOnly = true });

foreach (ColorAnimation colorAnimation in \_colorAnimation)

{

var scb = new SolidColorBrush();

var tb = new TextBox { Text = "Color", IsReadOnly = true, Foreground = scb };

scb.BeginAnimation(SolidColorBrush.ColorProperty, colorAnimation);

ComboBox2.Items.Add(tb);

}

ComboBox2.Items.Add(new TextBox { Text = " Nothing ", IsReadOnly = true });

foreach (DoubleAnimation doubleAnimation in \_sizeAnimation)

{

var tb = new TextBox { Text = "Size", IsReadOnly = true };

tb.BeginAnimation(FontSizeProperty, doubleAnimation);

ComboBox3.Items.Add(tb);

}

ComboBox3.Items.Add(new TextBox { Text = " Nothing ", IsReadOnly = true });

foreach (DoubleAnimation doubleAnimation in \_rotateAnimation)

{

var rt = new RotateTransform();

var tb = new TextBox { Text = " Rotation ", IsReadOnly = true, RenderTransform = rt };

rt.BeginAnimation(RotateTransform.AngleProperty, doubleAnimation);

ComboBox4.Items.Add(tb);

}

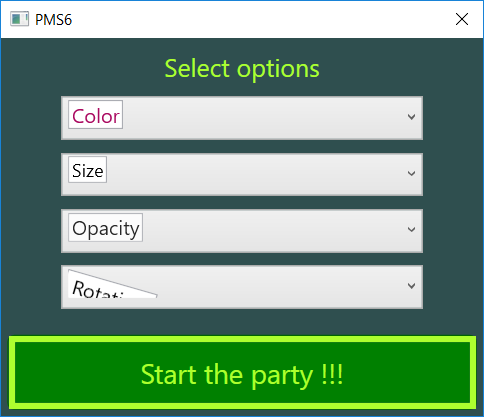
ComboBox4.Items.Add(new TextBox { Text = " Nothing ", IsReadOnly = true });

}

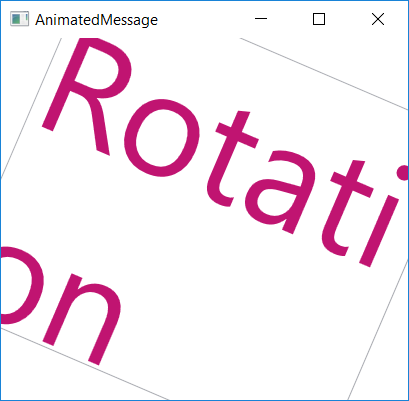
}

}

**Результати виконання програми**



*Рис.1 Головне вікно програми*



*Рис.2 Вікно з анімацією* **Контрольні питання**

**1. Означте поняття XML з погляду гіпермедіа.**

XML (Extensible Markup Language, розширювана мова розмітки) – рекомендований Консорціумом WWW текстовий формат для зберігання ієрархічно структурованих даних, стандарт для побудови спеціалізованих мов розмітки для обміну інформацією між різними застосунками.

**2. Які XML-базовані мови використовують у гіпермедіа?**

З DTD-шаблонів, що набули поширення, можна виділити SMIL (Synchronized Multimedia Integration Language – мова інтеграції синхронізованих засобів мультимедіа), SVG (Scalable Vector Graphics – масштабована векторна графіка), MathML (Math Markup Language – мова розмітки математики) і XFDL (eXtensible Forms Description Language – мова опису розширюваних форм). Визначенням мови HTML з використанням DTD-шаблона XML є XHTML.

**3. Як досягають синхронізації мультимедійних елементів в XML-базованих мовах?**

Реальним вмістом документа SMIL є його тіло (тег body). У тілі можна розміщувати так звані теги синхронізації, за допомогою яких визначають часові властивості презентації. Таких тегів є два: par (від parallel – паралельний) і seq (від sequence – послідовний). Кожен тег може містити елементи мультимедійних об’єктів (media object elements), які задають реальні зображення, відеокліпи, звук тощо, використані у презентації. Щоб описати складні взаємозв’язки синхронізовуваних об’єктів, теги синхронізації можна вкладати один в одний. Елементи, розміщені всередині par, можуть відтворюватися одночасно, тоді як елементи із seq відображаються послідовно.

**4. Поясніть суть анімування об’єктів в XML-базованих мовах.**

Основою механізму анімації у SMIL є тег animate, атрибути якого задають властивості елемента, який буде анімовано, і принципи зміни його параметрів у процесі анімації. За замовчуванням анімація застосовується до батьківського стосовно animate тегу. Наприклад, щоб анімувати зображення, animate потрібно розмістити всередині тегу img. Є й альтернативний варіант: присвоїти значення id анімованого елемента атрибуту targetElement тегу animate.

**5. Що таке WPF?**

WPF (Windows Presentation Foundation) – графічна підсистема у складі .NET Framework, призначена для створення клієнтських застосунків з візуально привабливими можливостями взаємодії з користувачем.WPF – це високорівневий об’єктно-орієнтований функціональний шар (framework), що дає змогу розробляти двовимірні та тривимірні програмні інтерфейси. За допомогою WPF можна створювати широкий спектр як автономних, так і веб-орієнтованих застосунків.

**6. Назвіть ключові особливості WPF.**

В основі WPF лежить векторна система візуалізації, що не залежить від роздільності пристрою виведення і враховує можливості сучасного графічного устаткування. WPF надає засоби для створення візуального інтерфейсу, включаючи мову XAML, елементи керування, макети, стилі, шаблони, двомірну і тривимірну графіку, анімацію, мультимедійні засоби.

**7. Що таке XAML і як її використовують на практиці?**

XAML (Extensible Application Markup Language – розширювана мова розмітки застосунків) є мовою розмітки, яку використовують для створення екземплярів об’єктів .NET. Хоча мова XAML – це технологія, що може бути застосована до багатьох різних предметних областей, її головне призначення – конструювання інтерфейсів користувачів WPF.

**8. Опишіть загальну схему створення прозорого вікна засобами WPF.**

1. Встановити для властивості вікна AllowsTransparency значення True.

2. Задати для властивості вікна WindowStyle значення None, щоб сховати рядок заголовку.

3. Визначити для тла (властивість вікна Background) прозорий колір (значення Transparent).

**9. Перелічіть варіанти надання WPF-вікну нестандартної форми.**

• Використати фонову графіку такого формату, який підтримує прозорість (наприклад, PNG). Це простий прямолінійний підхід, небезпечний тим, що в разі зміни розмірів вікна чи параметрів роздільності монітора растрова фонова графіка може набути спотвореного вигляду.

• Застосувати доступні у WPF функції для малювання форм, щоб створити тло з векторним вмістом. Такий підхід виключає втрату якості, якими б не були розміри вікна.

• Використовувати стандартні WPF-елементи, що мають необхідну форму. Наприклад, вікно із чудовими округленими краями можна створити за допомогою елемента Border. Такий підхід дає змогу створювати вікна із сучасним зовнішнім виглядом без застосування яких-небудь дизайнерських навичок.

**10. Назвіть ключову особливість анімації, створеної засобами WPF.**

Щоб запустити її, не потрібно використовувати таймер і код обробки подій: замість цього анімацію створюють декларативно, аналогічно до засобів SMIL та SVG. Анімація інтегрована у звичайні вікна WPF: наприклад, якщо анімувати кнопку так, щоб вона дрейфувала вікном, вона при цьому залишається кнопкою – її можна натиснути, запустивши звичайний код обробника події.

**11. Опишіть необхідний мінімум параметрів для опису WPF-анімації.**

DoubleAnimation widthAnimation = new DoubleAnimation();

widthAnimation.From = <початкове значення>;

widthAnimation.To = <кінцеве значення>;

widthAnimation.Duration = TimeSpan.FromSeconds(<тривалість>);

cmdGrow.BeginAnimation(Button.WidthProperty, widthAnimation);

**12. Які особливості використання тривимірної графіки у WPF?**

Підтримка тривимірної графіки – одне з важливих нововведень платформи WPF. Попередні високорівневі набори інструментів розроблення, такі як Windows Forms, взагалі обходилися без підтримки 3D-засобів. WPF же надає користувачу надзвичайно прості у використанні й водночас ефективні інструменти для роботи з ними. З одного боку, можна створити складний код, що генерує і модифікує тривимірні сітки із залученням серйозного математичного апарату, водночас можна просто експортувати тривимірні моделі зі спеціалізованого програмного пакета і маніпулювати ними за допомогою простих трансформацій. Ключові засоби забезпечені високорівневими класами, що не потребують глибоких знань.

**Висновок**

У процесі виконання цієї лабораторної роботи я отримав практичні навики роботи з мультимедійними даними та анімованими об’єктами з використанням засобів WPF.

**Посилання на GitHub**

https://github.com/bportakh/PMS