UBND TỈNH BÀ RỊA – VŨNG TÀU

TRƯỜNG CAO ĐẲNG KỸ THUẬT CÔNG NGHỆ

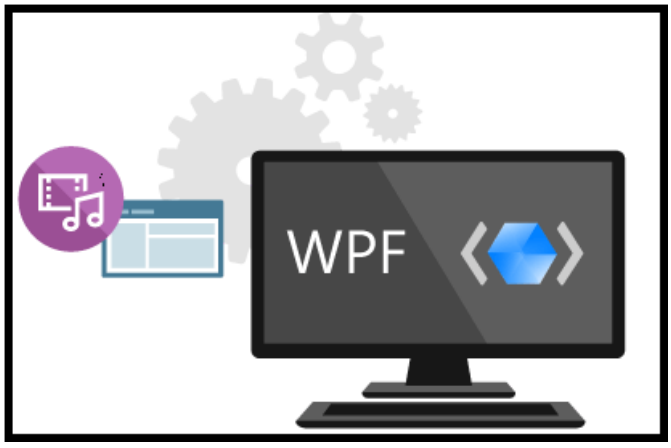
GIÁO TRÌNH

MÔ ĐUN: LẬP TRÌNH ỨNG DỤNG WPF

NGHỀ: CÔNG NGHỆ THÔNG TIN

TRÌNH ĐỘ: TRUNG CẤP

(Ban hành kèm theo Quyết định số: /QĐ-CĐKTCN ngày…….tháng….năm ..... của Hiệu trưởng Trường Cao đẳng Kỹ thuật Công nghệ BR – VT)



BÀ RỊA-VŨNG TÀU, NĂM 2020.

TUYÊN BỐ BẢN QUYỀN

Nhằm đáp ứng nhu cầu học tập và nghiên cứu cho giảng viên và sinh viên nghề Công nghệ Thông tin trong trường Cao đẳng Kỹ thuật Công nghệ Bà Rịa – Vũng Tàu, chúng tôi đã thực hiện biên soạn tài liệu Lập trình ứng dụng WPF này.

Tài liệu được biên soạn thuộc loại giáo trình phục vụ giảng dạy và học tập, lưu hành nội bộ trong Nhà trường nên các nguồn thông tin có thể được phép dùng nguyên bản hoặc trích dùng cho các mục đích về đào tạo và tham khảo.

Mọi mục đích khác mang tính lệch lạc hoặc sử dụng với mục đích kinh doanh thiếu lành mạnh sẽ bị nghiêm cấm.

LỜI GIỚI THIỆU

Giáo trình Lập trình ứng dụng WPF được biên soạn dựa trên khung chương trình đào tạo Cao đẳng nghề Công nghệ Thông tin đã được Trường Cao đẳng Kỹ thuật Công nghệ Bà Rịa – Vũng Tàu phê duyệt.

Bên cạnh đó nhằm giúp cho người học các kiến thức và kỹ năng cần thiết để thiết kế và lập trình được các ứng dụng nhỏ trong cuộc sống hằng ngày .

Tác giả đã nghiên cứu một số tài liệu, trang web liên quan kết hợp với kinh nghiệm làm việc, giảng dạy thực tế để biên soạn giáo trình này.

Trong giáo trình này bao gồm các bài sau:

Bài 1: Tổng quan về WPF

Bài 2: Bố trí giao diện

Bài 3: Sử dụng các điều khiển cơ bản

Bài 4: Tạo hộp chọn font chữ

Bài 5: Tạo hộp chọn hình ảnh

Bài 6: Sử dụng Expander

Bài 7: Tạo hộp soạn thảo văn bản

Bài 8: Tạo menu

Bài 9: Tạo ToolBar

Bài 10: Tạo ContextMenu

Bài 11: Tạo StatusBar

Bài 12: Sử dụng Style

Bài 13: Sử dụng Template

Bài 14: Truy xuất cơ sở dữ liệu

Với tiêu chí trình bày cô động, dễ hiểu áp dụng thực tế, kèm theo chi tiết các bước hướng dẫn thực hành cụ thể giúp cho người học dễ tiếp thu các kiến thức cần thiết và hình thành được kỹ năng thực hành và rèn luyện thao tác.

Trong quá trình biên soạn, chắc chắn giáo trình còn nhiều thiếu sót. Tác giả rất mong nhận được ý kiến đóng góp của quý thầy/cô và các em sinh viên để tiếp tục hoàn thiện hơn.

Xin chân thành cảm ơn quý đồng nghiệp, bạn bè đã có những ý kiến đóng góp trong quá trình xây dựng giáo trình này.

Bà Rịa – Vũng Tàu, ngày …… tháng …… năm 2020

Tham gia biên soạn

1. Trần Thị Thanh Hương – Chủ biên

MỤC LỤC

GIÁO TRÌNH MÔ ĐUN

Tên mô đun: Lập trình ứng dụng WPF

Mã mô đun: MĐ 24

Vị trí, tính chất, ý nghĩa và vai trò của mô đun:

- Vị trí của mô đun: được bố trí sau khi học xong các môn cơ sở và Lập trình Windows.

- Tính chất của mô đun: là mô đun nghề tự chọn của trình độ Cao đẳng nghề Công nghệ Thông tin, cung cấp kiến thức và kỹ năng tạo ứng dụng WPF với giao diện chuyên nghiệp.

- Ý nghĩa và vai trò của môn học/mô đun: Môn học giúp làm quen với ứng dụng WPF, thực hành được các giao diện lập trình ứng dụng nhỏ như: quản lý quán café, quản lý khách sạn, quản lý nhân viên, quản lý sinh viên….

Mục tiêu của mô đun:

- Về kiến thức:
  + Biết ưu điểm của XAML, WPF trong phát triển ứng dụng Windows
  + Biết quy trình tạo project WPF
  + Biết công dụng của các loại panel, control trong WPF
  + Trình bày công dụng của Menu, Toolbar, ContextMenu, StatusBar
  + Trình bày công dụng và quy trình sử dụng Style, Template
  + Biết quy truy xuất CSDL trong WPF
- Về kỹ năng:
  + Tạo project WPF
  + Sử dụng hợp lý các loại panel, control trong thiết kế giao diện cho ứng dụng WPF
  + Tạo Menu, Toolbar, ContextMenu, StatusBar cho ứng dụng
  + Sử dụng Style và Control Template cho giao diện
  + Sử dụng Data Template hiển thị dữ liệu
  + Truy xuất CSDL trong WPF

- Về năng lực tự chủ và trách nhiệm:

+ Lưu solution, project đúng đường dẫn

+ Tự phát triển ứng dụng với giao diện chuyên nghiệp

+ Tham gia phát triển ứng dụng quản lý cho các tổ chức, doanh nghiệp, công ty phần mềm

+ Reèn luyện va`nâng cao ky~năng lập trình theo công nghệ hiện đại

+ Vận dụng ngôn ngữ XAML trong việc tự học thiết kế giao diện cho các ứng dụng di động

+ Thực hiện các bài thực hành đảm bảo đúng trình tự, an toàn.

Nội dung của mô đun:

BÀI 1: GIỚI THIỆU TỔNG QUAN

Mã bài: 20.1

Giới thiệu:

Windows Presentation Foundation hay gọi tắt là WPF – là một nền tảng cho phép developer có thể tạo ra các ứng dụng trên nền .NET framework cho Windows nói chung. Cũng có thể gọi WPF là một GUI framework. Nó ra đời sau Winform và tốt hơn người tiền nhiệm về mọi mặt..

Mục tiêu:

- Trình bày được khái và mục tiêu cơ bản của WPF
- Trình bày được khái niệm và vai trò của XAML trong việc tạo môi trường làm việc chung giữa người thiết kế giao diện và người lập trình
- Tạo được ứng dụng WPF đơn giản
- Đảm bảo quy tắc thẻ mở, đóng
- Lưu project, solution đúng đường dẫn

Nội dung chính:

1. Khái niệm WPF

WPF, viết tắt của Windows Presentation Foundation, là hệ thống API mới hỗ trợ việc xây dựng giao diện đồ hoạ trên nền Windows. Được xem như thế hệ kế tiếp của WinForms, WPF tăng cường khả năng lập trình giao diện của lập trình viên bằng cách cung cấp các API cho phép tận dụng những lợi thế về đa phương tiện hiện đại. Là một bộ phận của .NET Framework 3.0, WPF sẵn có trong Windows Vista và Windows Server 2008. Đồng thời, WPF cũng có thể hoạt động trên nền Windows XP Service Pack 2 hoặc mới hơn, và cả Windows Server 2003.

WPF được xây dựng nhằm vào ba mục tiêu cơ bản:

Cung cấp một nền tảng thống nhất để xây dựng giao diện người dùng;

Cho phép người lập trình và người thiết kế giao diện làm việc cùng nhau một cách dễ dàng;

Cung cấp một công nghệ chung để xây dựng giao diện người dùng trên cả Windows và trình duyệt Web.

Trước khi WPF ra đời, việc tạo giao diện người dùng theo những yêu cầu mô tả ở ví dụ trên đòi hỏi sử dụng rất nhiều công nghệ khác nhau (xem Bảng 2.1). Để tạo form, các control và các tính năng kinh điển khác của một giao diện đồ họa Windows, thông thường lập trình viên sẽ chọn Windows Forms, một phần của .NET Framework. Nếu cần hiển thị văn bản, Windows Forms có một số tính năng hỗ trợ văn bản trực tiếp hoặc có thể sử dụng Adobe's PDF để hiển thị văn bản có khuôn dạng cố định. Đối với hình ảnh và đồ họa 2 chiều, lập trình viên sẽ dùng GDI+, một mô hình lập trình riêng biệt có thể truy nhập qua Windows Forms. Để hiển thị video hay phát âm thanh, lập trình viên lại phải sử dụng Windows Media Player, và với đồ họa 3 chiều, anh ta lại phải dùng Direct3D, một thành phần chuẩn khác của Windows. Tóm lại, quá trình phát triển giao diện người dùng theo yêu cầu trở nên phức tạp, đòi hỏi lập trình viên quá nhiều kỹ năng công nghệ.

	Windows Forms	**Windows Forms/ GDI+**	**WPF**
Giao diện đồ họa (form và các control)	X		X
On-screen văn bản	X		X
Fixed-format văn bản			X
Hình ảnh		X	X
Video và âm thanh			X

Đồ họa 2 chiều		X	X
Đồ họa 3 chiều			X

Bảng 1.1. Thành phần giao diện theo yêu cầu và những công nghệ chuyên biệt cần thiết để tạo chúng.

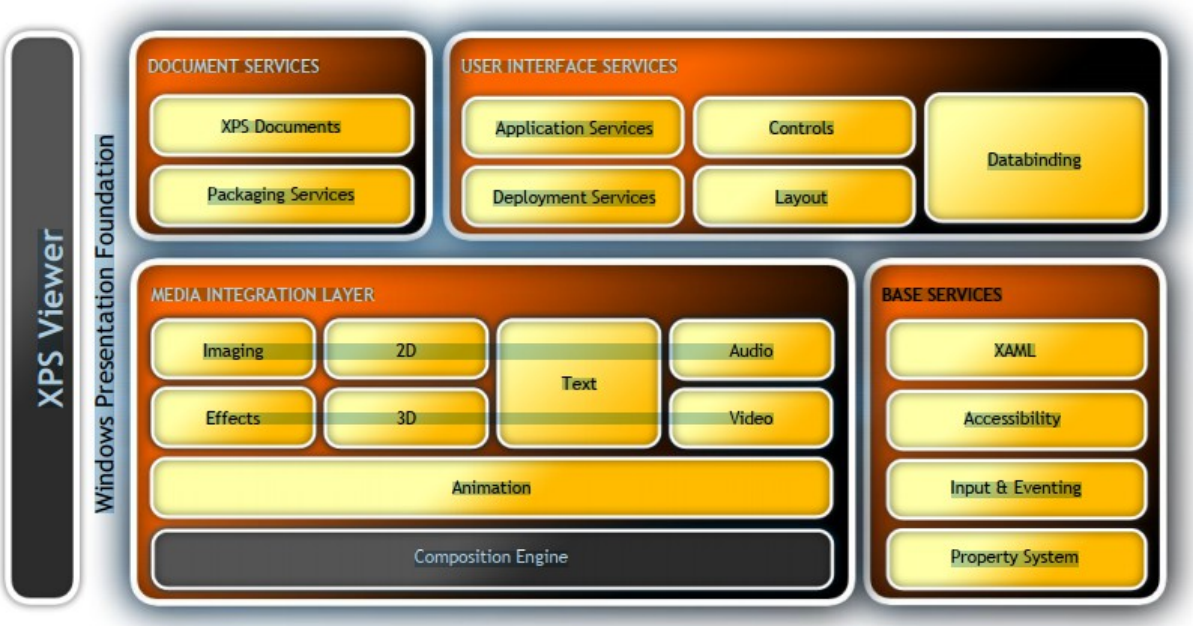
2. Các thành phần của WPF

Giống như các thành phần khác của .NET Framework, WPF tổ chức các chức năng theo một nhóm *namespace* cùng trực thuộc namespace **System.Windows**. Bất kể chức năng nào được sử dụng, cấu trúc cơ bản của mọi ứng dụng WPF đều gần như nhau. Là ứng dụng Windows độc lập hay là một XBAP, một ứng dụng WPF điển hình bao giờ cũng gồm một tập các trang XAML và phần code tương ứng được viết bằng C# hoặc Visual Basic, còn gọi là các file *code-behind*. Tất cả các ứng dụng đều kế thừa từ lớp chuẩn **Application** của WPF. Lớp này cung cấp những dịch vụ chung cho mọi ứng dụng, chẳng hạn như các biến lưu trữ trạng thái của ứng dụng, các phương thức chuẩn để kích hoạt hay kết thúc ứng dụng.

Mặc dù WPF cung cấp một nền tảng thống nhất để tạo giao diện người dùng, những công nghệ mà WPF chứa đựng có thể phân chia thành những thành phần độc lập. Nhân của WPF là cơ chế tạo sinh đồ họa dựa trên vector và độc lập với độ phân giải nhằm tận dụng những lợi thế của phần cứng đồ họa hiện đại. WPF được mở rộng với các tập tính năng phát triển ứng dụng bao gồm XAML, các control, cơ chế móc nối dữ liệu, layout, đồ họa 2 chiều, ba chiều, hoạt họa, style, khuôn dạng mẫu, văn bản, media, text và in ấn. WPF nằm trong .NET Framework, nên ngoài ra, ứng dụng WPF có thể kết hợp các thành

Mặc dù WPF cung cấp một nền tảng thống nhất để tạo giao diện người dùng, những công nghệ mà WPF chứa đựng có thể phân chia thành những thành phần độc lập. Nhân của WPF là cơ chế tạo sinh đồ họa dựa trên vector và độc lập với độ phân giải nhằm tận dụng những lợi thế của phần

cứng đồ họa hiện đại. WPF được mở rộng với các tập tính năng phát triển ứng dụng bao gồm XAML, các control, cơ chế móc nối dữ liệu, layout, đồ họa 2 chiều, ba chiều, hoạt họa, style, khuôn dạng mẫu, văn bản, media, text và in ấn. WPF nằm trong .NET Framework, nên ngoài ra, ứng dụng WPF có thể kết hợp các thành phần khác có trong thư viện lớp của .NET Framework.



Hình 1.1. Các thành phần cơ bản của WPF

3.1 Layout và Control

Để sắp đặt các thành phần khác nhau trên giao diện, ứng dụng WPF sử dụng *panel*. Mỗi panel có thể chứa các thành phần con, bao gồm các control như nút bấm hay hộp thoại, hay bản than những panel khác. Những loại panel khác nhau cho phép sắp xếp thành phần con theo những cách khác nhau. Ví dụ, **DockPanel** cho phép các thành phần con có thể được đặt dọc theo cạnh của panel đó, trong khi **Grid** cho phép sắp đặt các thành phần con của nó trên một lưới tọa độ.

Giống như bất kỳ một công nghệ giao diện người dùng nào, WPF cung cấp một số lượng lớn các control. Ngoài ra, người dùng có thể tùy ý định nghĩa các control theo ý mình. Các control chuẩn gồm **Button**, **Label**, **TextBox**, **ListBox**, **Menu**, **Slider,** hay phức tạp hơn có **SpellCheck,**

PasswordBox… Các sự kiện do người dùng tạo ra, như di chuyển chuột hay ấn phím, có thể được các control nắm bắt và xử lý. Trong khi các control và các thành phần giao diện khác có thể được đặc tả đầy đủ bằng XAML, các sự kiện bắt buộc phải được xử lý bằng mã trình.

3.2 Style và Template

Giống như sử dụng Cascading Style Sheets (CSS) đối với HTML, việc định ra thuộc tính đồ họa cho các đối tượng giao diện một lần, rồi sau đó áp dụng lại cho các đối tượng khác cùng loại thường rất tiện lợi. WPF cũng cung cấp tính năng tương tự bằng việc sử dụng thành phần **Style** của XAML. Ví dụ, kiểu ButtonStyle có thể được định nghĩa như sau:

```
<Style x:Key="ButtonStyle">
  <Setter Property="Control.Background" Value="Red"/>
  <Setter Property="Control.FontSize" Value="16"/>
</Style>
```

Bất kỳ nút bấm nào sử dụng kiểu này sẽ có nền màu đỏ và sử dụng font chữ kích thước 16. Ví dụ:

```
<Button Style="{StaticResource ButtonStyle}">
  Click Here
</Button>
```

Một Style có thể được dẫn xuất từ một Style khác, thừa kế hoặc chồng lên những thuộc tính đã thiết lập. Mỗi style có thể định nghĩa các *trigger* cho phép tạo ra những hiệu ứng tương tác đặc biệt, chẳng hạn như khi lướt chuột qua nút bấm, nút bấm chuyển thành màu vàng.

WPF cũng hỗ trợ sử dụng *template*. Mỗi template tương tự như một style, và ở hai dạng:

- *Template cho dữ liệu*: sử dụng thành phần **DataTemplate** của XAML để thiết lập một nhóm thuộc tính hiển thị của dữ liệu như màu sắc, phương thức căn lề...

- *Template cho control*: sử dụng thành phần **ControlTemplate** của XAML để định ra diện mạo của một control.

3.3. Text

Giao diện người dùng ít nhiều đều hiển thị chữ hay text. Đối với phần lớn mọi người, đọc text trên màn hình thường khó hơn đọc trên giấy in. Đó là do chất lượng hiển thị text trên màn hình kém hơn so với khi in ra giấy. WPF tập trung giải quyết vấn đề này, làm chất lượng text hiển thị trên màn hình tương đương trên giấy in. Cụ thể, WPF hỗ trợ các font chữ OpenType chuẩn, cho phép sử dụng các thư viện font đã có. WPF cũng hỗ trợ công nghệ font chữ mới ClearType, cho phép hiển thị các ký tự mịn hơn đối với mắt người, đặc biệt là trên màn hình tinh thể lỏng (LCD).

Để nâng cao hơn nữa chất lượng hiển thị text, WPF cho phép một số công nghệ khác như chữ ghép, theo đó một nhóm ký tự được thay thế bằng một ảnh đơn nhất, tạo tâm lý thoải mái hơn khi đọc đối với người dùng.

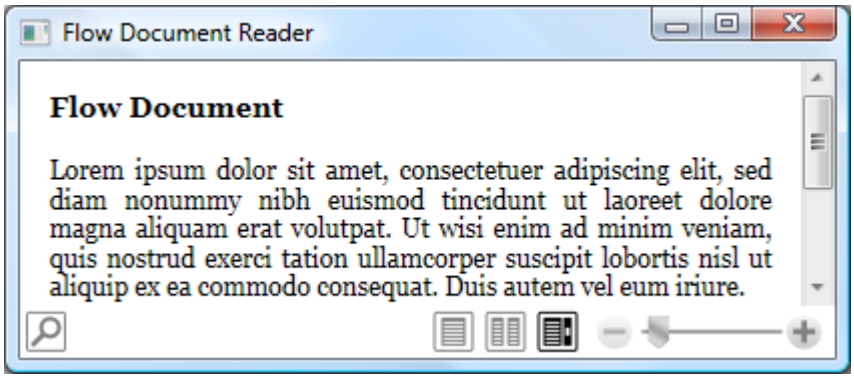
3.4. Văn bản

WPF hỗ trợ ba dạng văn bản: văn bản cố định (fixed), văn bản thích nghi (flow/adaptive) và văn bản XPS (XML Paper Specification). Kèm theo đó, WPF cũng cung cấp các dịch vụ để tạo, xem, quản lý, ghi chú, đóng gói và in ấn văn bản.

Văn bản cố định trông không đổi bất kể chúng được hiển thị trên màn hình hay in ra máy in. Trong WPF, những văn bản dạng này được định nghĩa bằng phần tử **FixedDocument** trong XAML và được hiển thị bằng control **DocumentViewer**.

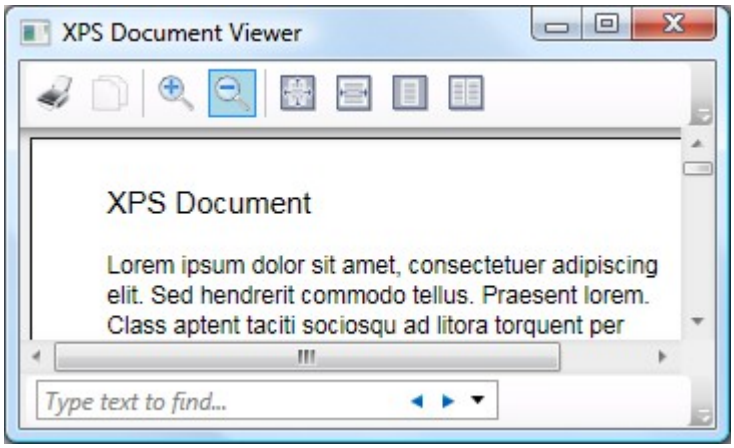
Trong khi đó, văn bản thích nghi thường chỉ dùng để đọc trên màn hình, và có khả năng tự động thay đổi các thuộc tính hiển thị ảnh và text cho phù hợp với kích thước cửa sổ hay các yếu tố môi trường khác nhằm nâng cao chất lượng đọc cho người dùng. Văn bản thích nghi được định nghĩa bằng phần tử **FlowDocument.** Để hiển thị văn bản thích nghi, WPF sử dụng

một số control khác nhau, chẳng hạn như **FlowDocumentPageViewer, FlowDocumentScrollViewer, FlowDocumentReader…**



Hình 1.2. Minh họa về văn bản thích nghi trong WPF.

Văn bản XPS xây dựng trên cơ sở văn bản bất động của WPF. XPS là một định dạng mở theo đặc tả XML, có khả năng sử dụng trên nhiều nền tảng khác nhau, được thiết kế nhằm tạo thuận lợi cho việc xây dựng, chia sẻ, in ấn và lưu trữ văn bản. Cũng như văn bản cố định, văn bản XPS được hiển thị bằng **DocumentViewer**.



Hình 1.3. Minh họa về văn bản XPS trong WPF.

3.5. Hình ảnh

Trong WPF, hình ảnh được hiển thị nhờ control **Image,** ví dụ:

```
      <Image
       Width="200"
       Source="C:\Documents      and      Settings\All      Users\Documents\My
Pictures\Ava.jpg" />
```

Control **Image** có thể hiển thị hình ảnh lưu trữ dưới nhiều khuôn dạng khác nhau, bao gồm JPEG, BMP, TIFF, GIF và PNG. Nó cũng có thể hiển thị hình ảnh dạng Windows Media Photo mới được sử dụng trong Windows Vista. Bất kể ở khuôn dạng nào, WPF sử dụng Windows Imaging Component (WIC) để tạo ra hình ảnh. Cùng với các *codec* dùng cho các khuôn dạng ảnh kể trên, WIC cũng cung cấp một nền tảng chung để bổ sung codec khác.

3.6. Video và âm thanh

Khi tốc độ của các bộ xử lý và truyền thông mạng ngày một nâng cao, video trở thành một phần tương tác lớn của người dùng với phần mềm. Người dùng cũng sử dụng nhiều thời gian để nghe nhạc và các dạng âm thanh khác trên máy tính. Do đó, WPF cung cấp tính năng hỗ trợ cả hai dạng media này thông qua phần tử **MediaElement**. Control này có thể chơi các định dạng video WMV, MPEG và AVI, và nhiều định dạng âm thanh khác nhau. Việc lập trình để chạy một đoạn video trở nên khá đơn giản, như trong ví dụ sau:

```
<MediaElement
 Source="C:\Documents and Settings\All Users\Documents\
  My Videos\Ruby.wmv" />
```

3.7. Đồ họa 2 chiều

Trong 20 năm gần đây, việc tạo ra đồ họa hai chiều trên Windows dựa trên Graphics Device Interface (GDI) và phiên bản sau của nó GDI+. Các ứng dụng Windows Forms phải sử dụng chức năng này thông qua một namespace khác hoàn toàn, bởi bản thân Windows Forms không tích hợp đồ họa 2 chiều. Đối với đồ họa 3 chiều thì càng tồi hơn, Windows Forms phải dựa trên công nghệ hoàn toàn biệt lập là Direct3D. Với WPF, vấn đề trở nên đơn giản hơn nhiều. Cả đồ họa 2 chiều và 3 chiều đều có thể được tạo ra trực tiếp trong XAML hoặc trong code sử dụng thư viện WPF tương ứng.

Đối với đồ họa 2 chiều, WPF định ra nhóm control của các khuôn hình (shapes) mà ứng dụng có thể sử dụng để tạo nên hình ảnh, gồm:

- **Line**: vẽ đường thẳng qua 2 điểm.
- **Elllipse**: vẽ ellipse.
- **Rectangle**: vẽ chữ nhật.
- **Polygon**: vẽ đa giác.
- **Polyline**: vẽ đa giác mở.
- **Path**: vẽ hình theo một đường bất kỳ.

Mỗi khuôn hình đều có các thuộc tính phong phú cho phép hiển thị với nhiều tính chất khác nhau: màu nền, màu biên… Một đặc điểm quan trọng trong WPF là: vì mọi thứ đều được xây dựng trên một nền chung, việc kết hợp các đặc tính và đối tượng khác nhau, chẳng hạn, lồng một ảnh vào một hình chữ nhật, trở nên đơn giản. Điểm thú vị nữa là các đối tượng hình học này còn có thể thu nhận các sự kiện từ phía người dùng như một control, chẳng hạn sự kiện nhắp chuột.

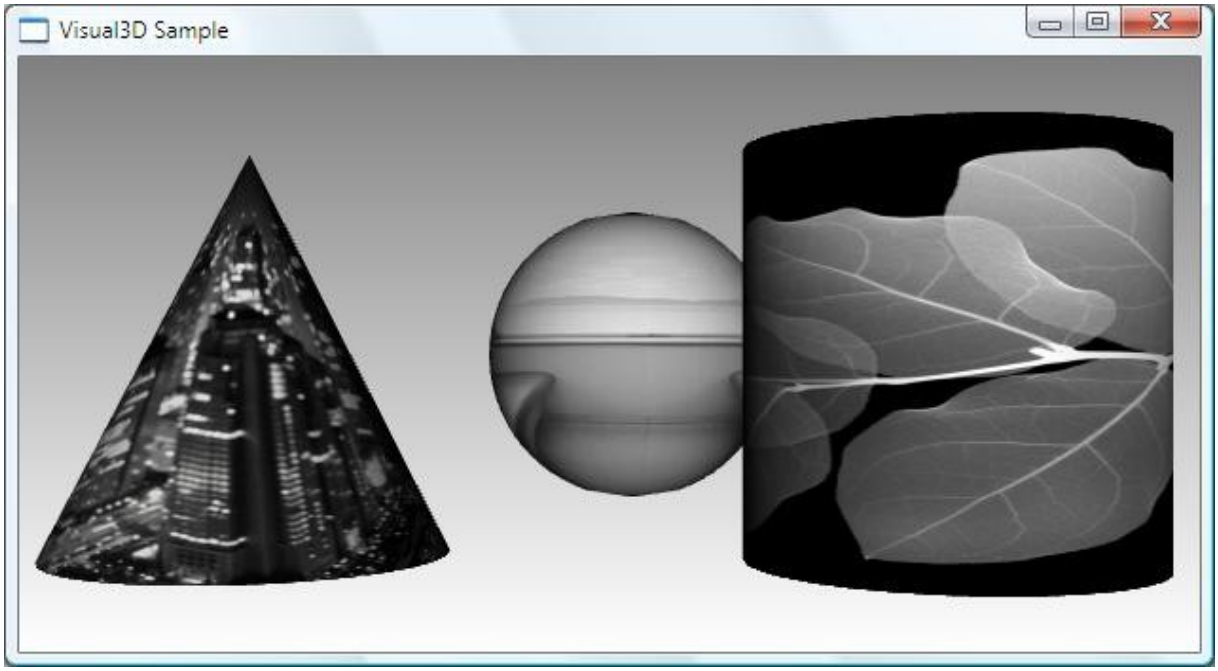
Ngoài ra, WPF cũng cung cấp một nhóm chức năng hình học khác, gọi là *geometries*, để làm việc với đồ họa hai chiều, như **LineGeometry**, **RectangleGeometry**, **EllipseGeometry**, và **PathGeometry.** Dạng hình học này có nhiều thuộc tính và chức năng tương tự như các khuôn hình đã nêu trên. Điểm khác biệt quan trọng nhất là các geometries không được dùng để hiển thị, chúng được dùng chủ yếu để tính toán hình học, ví dụ như để định ra các vùng miền, theo dõi vị trí bấm chuột...

Thêm vào đó, WPF cung cấp lớp **Transform** cho phép thực hiện các biến đổi hình học như xoay, dịch chuyển, co giãn đối tượng đồ họa; hoặc cho phép thực hiện các hiệu ứng hoạt họa theo thời gian thông qua các lớp **Animation** và **Timing.**

3.8. Đồ họa 3 chiều

WPF hỗ trợ đồ họa 3 chiều bằng việc gói các lời gọi API của Direct3D, và do vậy, việc sử dụng chúng trở nên thống nhất và đơn giản hơn đáng kể. Để hiển thị đồ họa ba chiều, ứng dụng WPF sử dụng control **Viewport3D**. Để tạo ra các cảnh ba chiều, lập trình viên mô tả một hay nhiều *mô hình*, sau đó, phân định cách thức các mô hình này được chiếu sáng hay hiển thị. Như thường lệ, điều này được thực hiện bằng XAML, bằng code hay trộn cả hai. Để mô tả mô hình, WPF cung cấp lớp **GeometryModel3D** để tạo ra hình dạng của mô hình. Khi mô hình đã được định hình, diện mạo bên ngoài của nó có thể được điều khiển bằng việc phủ lên các *vật liệu* (material). Chẳng hạn, lớp **SpecularMaterial** cho phép tạo bóng trên bề mặt mô hình.

Bất kể được làm từ vật liệu gì, một mô hình có thể được chiếu sáng theo nhiều cách. Lớp **DirectionalLight** cho phép ánh sáng tới từ một hướng xác định, trong khi lớp **AmbientLight** tạo ra ánh sáng đồng đều trên mọi vật trong cảnh. Cuối cùng, để định ra cách nhìn cảnh, lập trình viên phải định ra một *camera*. Ví dụ, **PerspectiveCamera** cho phép phân định khoảng cách từ vị trí nhìn tới vật thể và kiểu nhìn phối cảnh (tuân theo luật gần xa).



Hình 1. 4. Thiết lập đối tượng đồ họa ba chiều với WPF

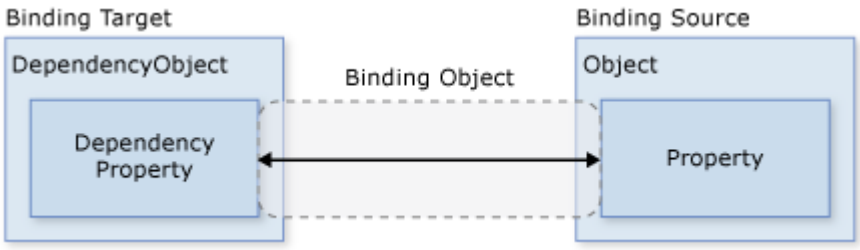
Xây dựng cảnh ba chiều trực tiếp bằng XAML hay mã trình đều không đơn giản. Do đó, chỉ nên dùng ứng dụng WPF để hiển thị cảnh ba chiều, việc xây dựng cảnh nên được thực hiện bằng những công cụ đồ họa chuyên biệt.

3.9. Kết nối dữ liệu

Phần lớn các ứng dụng được tạo ra đều cung cấp cho người dùng phương tiện để xem và sửa đổi dữ liệu. Trong các ứng dụng WPF, việc lưu trữ và truy xuất dữ liệu đã được thực hiện bởi các công nghệ như Microsoft SQL Server và ADO.NET. Sau khi dữ liệu được truy xuất và tải vào các đối tượng quản lý dữ liệu trên ứng dụng, phần việc khó khăn của ứng dụng WPF mới bắt đầu. Về cơ bản, có hai công việc phải thực hiện:

- Sao chép dữ liệu từ các đối tượng quản lý dữ liệu vào các control trên giao diện, qua đó, dữ liệu có thể được hiển thị hay sửa đổi.
- Đảm bảo rằng những thay đổi trên dữ liệu từ các control được cập nhật trở lại các đối tượng quản lý dữ liệu

Để đơn giản hóa quá trình phát triển ứng dụng, WPF cung cấp một cơ chế móc nối dữ liệu để thực hiện tự động những bước này. Phần nhân của cơ chế móc nối dữ liệu là lớp **Binding** mà nhiệm vụ của nó là liên kết control trên giao diện (đích) với đối tượng quản lý dữ liệu (nguồn). Mối quan hệ này được minh họa trong hình dưới đây:



Hình 1.5. Quan hệ giữa đối tượng dữ liệu và đối tượng phụ thuộc

Việc hỗ trợ móc nối dữ liệu được xây dựng ngay từ nhân của WPF. Tất cả các đối tượng đồ họa trong WPF đều kế thừa từ **DependencyObject**, chúng là các *đối tượng phụ thuộc*. Chức năng mà lớp cơ sở này hỗ trợ cho phép thực hiện hiệu ứng hoạt họa, tạo kiểu mẫu (styling) và móc nối dữ liệu. Các đối tượng này đều mang một thuộc tính đặc biệt gọi là

DependencyProperty, *thuộc tính phụ thuộc*. Phần lớn các thuộc tính hay dùng như **Text, Content, Width, Height**, vân vân đều là các thuộc tính phụ thuộc. Tất cả các thuộc tính phụ thuộc đều có thể tạo hiệu ứng hoạt họa, tạo kiểu và kết nối dữ liệu.

Cơ chế móc nối dữ liệu trong WPF còn cung cấp thêm những tính năng như xác thực tính hợp lệ, sắp xếp, lọc và phân nhóm dữ liệu. Thêm vào đó, tính năng móc nối dữ liệu cũng hỗ trợ sử dụng khuôn mẫu dữ liệu (data template) để tạo ra các đối tượng giao diện tùy biến có kết nối dữ liệu, khi các control chuẩn không phù hợp. Móc nối dữ liệu và khuôn dạng dữ liệu có thể được coi là tính năng mạnh nhất của WPF.

3. Tạo ứng dụng WPF đầu tiên