**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN**

Hưng Yên, ngày … tháng 11 năm 2011

GIẢNG VIÊN HƯỚNG DẪN

**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN**

Hưng Yên, ngày … tháng 11 năm 2011

GIẢNG VIÊN HƯỚNG DẪN

**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN**

Hưng Yên, ngày … tháng 11 năm 2011

GIẢNG VIÊN HƯỚNG DẪN

**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN**

Hưng Yên, ngày … tháng 11 năm 2011

GIẢNG VIÊN HƯỚNG DẪN

# MỤC LỤC

[**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN** 1](#_Toc312840712)

[**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN** 2](#_Toc312840713)

[**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN** 3](#_Toc312840714)

[**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN** 4](#_Toc312840715)

[MỤC LỤC 5](#_Toc312840716)

[**LỜI CẢM ƠN** 6](#_Toc312840717)

[PHẦN I: MỞ ĐẦU 7](#_Toc312840718)

[1 Lý do chọn đề tài 7](#_Toc312840719)

[2. Đối tượng nghiên cứu 8](#_Toc312840720)

[3. Giới hạn và phạm vi nghiên cứu 8](#_Toc312840721)

[4. Mục đích nghiên cứu 8](#_Toc312840722)

[5. Nhiệm vụ nghiên cứu 8](#_Toc312840723)

[6. Phương pháp nghiên cứu 8](#_Toc312840724)

[7. Ý nghĩa lý luận và thực tiễn của đề tài 8](#_Toc312840725)

[PHẦN II: NỘI DUNG 10](#_Toc312840726)

[I. Tìm hiểu Silverlight 10](#_Toc312840727)

[Chương 1: Tổng quan về Silverlight 10](#_Toc312840728)

[Chương 2: Layout trong silverlight 14](#_Toc312840729)

[Chương 3: Các control cơ bản 17](#_Toc312840730)

[Chương 4: Xử lý đồ họa trên Silverlight 33](#_Toc312840731)

[Chương 5: Animation và Media 48](#_Toc312840732)

[Chương 6: Expression Blend 54](#_Toc312840733)

[II. Lập trình game cờ caro 58](#_Toc312840734)

[Chương I: Giới thiệu trò chơi caro 58](#_Toc312840735)

[Chương II: Giải thuật của chương trình 59](#_Toc312840736)

[Chương III: Thiết kế chương trình và các hàm chính của chương trình 68](#_Toc312840737)

[PHẦN III: KẾT LUẬN 81](#_Toc312840738)

[1. Kết quả đạt được 81](#_Toc312840739)

[2. Hạn chế của đề tài 81](#_Toc312840740)

[3. Hướng phát triển của đề tài 81](#_Toc312840741)

[4. Đề nghị ý kiến. 81](#_Toc312840742)

[PHẦN IV: TÀI LIỆU THAM KHẢO 82](#_Toc312840743)

**LỜI CẢM ƠN**

Sau thời gian thực hiện làm đề tài đến nay nhóm đã hoàn thành đề tài của mình. Để có được kết quả như hôm nay, nhóm thực hiện đề tài đã nhận được rất nhiều sự quan tâm, giúp đỡ và chỉ đạo nhiệt tình của các thầy, cô trong khoa công nghệ thông tin.

Đặc biệt nhóm thực hiện đề tài xin gửi lời cảm ơn chân thành nhất tới thầy **“Hoàng Quốc Việt”.** Thầy là người đã tận tình chỉ bảo, chia sẻ cho nhóm những kiến thức bổ ích hướng dẫn động viên trong suốt thời gian qua, tạo mọi điều kiện tốt nhất cho nhóm trong quá trình thực hiện.

Mặc dù nhóm thực hiện đề tài đã rất cố gắng trong quá trình làm đề tài song đồ án của nhóm không thể tránh khỏi những thiếu sót, hạn chế. Nhóm thực hiện đề tài rất mong nhận được sự đóng góp chân thành từ các thầy, cô và các bạn để đề tài của nhóm được hoàn thiện hơn.

*Nhóm thực hiện đề tài xin chân thành cảm ơn!*

Hưng Yên, ngày 11tháng 11 năm 2011

**Sinh viên thực hiện**

***Nguyễn Thành Trường***

***Bùi Thị Thanh***

# PHẦN I: MỞ ĐẦU

## 1 Lý do chọn đề tài

Tìm hiểu công nghệ là hành trang tốt nhất để chuẩn bị kiến thức cho chúng em sau khi ra trường. Silverlight là một công nghệ mới kết hợp nhiều công nghệ vào một nền tảng phát triển, nó cho phép lựa chọn nhiều công cụ và ngôn ngữ lập trình thích hợp để giải quyết bài toán. Silverlight ra đời đã giải quyết vấn đề thường gặp của các ứng dụng web về khả năng tương thích với trình duyệt và hệ điều hành. Silverlight cung cấp một mô hình lập trình lập trình mềm dẻo và đồng nhất. Ứng dụng nó tạo ra giao diện thân thiện, hiệu ứng đẹp cho các phần mềm các trang web. Hiện nay cờ caro là một trò chơi rất phổ biến trong giới học đường. Từ nhu cầu giải trí đó, đã có rất nhiều hãng đã sản xuất trò chơi caro trên máy tính, giúp cho việc chơi cờ caro được dễ dàng hơn, và bạn có thể đánh cờ với máy mà không cần một người bạn thực sự.

Trên cùng mục đích muốn đem trò chơi cờ caro đến với mọi người, làm cho việc chơi cờ trở nên dễ dàng hơn, có thể hỗ trợ cho bạn chơi với bạn bè, nhóm em đã chọn đề tài là: “**Tìm hiểu silverlight và lập trình game cờ caro**”.

Qua quá trình tìm hiểu chúng em đã hoàn thành bản báo cáo với nội dung sau:

Phần I: Mở đầu

Phần II: Nội dung

I. Tìm hiểu công nghệ

* Chương 1: Tổng quan về Silverlight
* Chương 2: Layout trong silverlight
* Chương 3: Các control cơ bản
* Chương 4: Xử lý đồ họa trên Silverlight
* Chương 5: Animation và Media
* Chương 6: Expression Blend

II. Lập trình game caro

* Chương 1: Giới thiệu trò chơi caro
* Chương 2: Giải thuật của chương trình
* Chương 3: Thiết kế chương trình và các hàm chính của chương trình

Phần III: Kết luận

Phần IV: Tài liệu tham khảo

## 2. Đối tượng nghiên cứu

* Công nghệ Silverlight, xaml
* Chương trình chơi game caro
* Công cụ xây dựng phần mềm: Visual Studio 2010, Microsoft Expression Blend.

## 3. Giới hạn và phạm vi nghiên cứu

Tìm hiểu công nghệ Silverlight và lập trình game caro

## 4. Mục đích nghiên cứu

* Xây dựng được tài liệu giúp tìm hiểu về công nghệ Silverlight
* Xây dựng được chương trình chơi game caro có sử dụng công nghệ Silverlight.

## 5. Nhiệm vụ nghiên cứu

* Tìm hiểu công nghệ Silverlight và ngôn ngữ XAML.
* Tìm hiều các kiến thức của công nghệ Silverlight để áp dụng vào xây dựng chương trình game caro.

## 6. Phương pháp nghiên cứu

* Tìm hiểu hết tổng hợp kiến thức chung của Silverlight, sau đó đi sâu vào những kiến thức cụ thể để xây dựng giao diện ứng dụng cờ caro
* Nghiên cứu công cụ Microsoft Expression Blend để dựa vào đó thiết kế giao diện cho trang web của ứng dụng .

## 7. Ý nghĩa lý luận và thực tiễn của đề tài

* Qua việc tìm hiểu đã giúp em biết thêm kiến thức một công nghệ mới với nhiều tính năng nổi trội so với các công nghệ hiện có của Microsoft.
* Công nghệ Silverlight cho ta một cái nhìn tổng quan về thiết kế giao diện đẹp và bóng bảy mà không đòi hỏi phải sử dụng quá nhiều công nghệ cùng lúc mới tạo được, chính điều đó tạo nên sức mạnh của Silverlight.

# PHẦN II: NỘI DUNG

## Tìm hiểu Silverlight

### ****Chương 1: Tổng quan về Silverlight****

#### Silverlight là gì?

1. Lý do ra đời Silverlight

* Các ứng dụng viết cho web thường gặp vấn đề về khả năng tương thích với trình duyệt và hệ điều hành.
* Các công nghệ plug-in trên web trước đây không cho phép truyền tải những dữ liệu hình ảnh chất lượng cao như 720p HDTV.

1. Silverlight là gì?

* Silverlight là một dạng plug-in dựa trên công nghệ của Microsoft .Net
* Nó độc lập với đa nền tảng và đa trình duyệt
* Nó cho phép phát triển các ứng dụng đa phương tiện đặc biệt là các ứng dụng trên web.
* Silverlight cung cấp một mô hình lập trình lập trình mềm dẻo và đồng nhất, nó hỗ trợ Ajax, Python, Ruby và các ngôn ngữ lập trình .Net như Visual basic, C#.

#### Đặc điểm

Silverlight kết hợp nhiều công nghệ vào một nền tảng phát triển, nó cho phép bạn được lựa chọn nhiều công cụ và ngôn ngữ lập trình thích hợp để giải quyết bài toán của bạn. Silverlight cung cấp các tính năng sau:

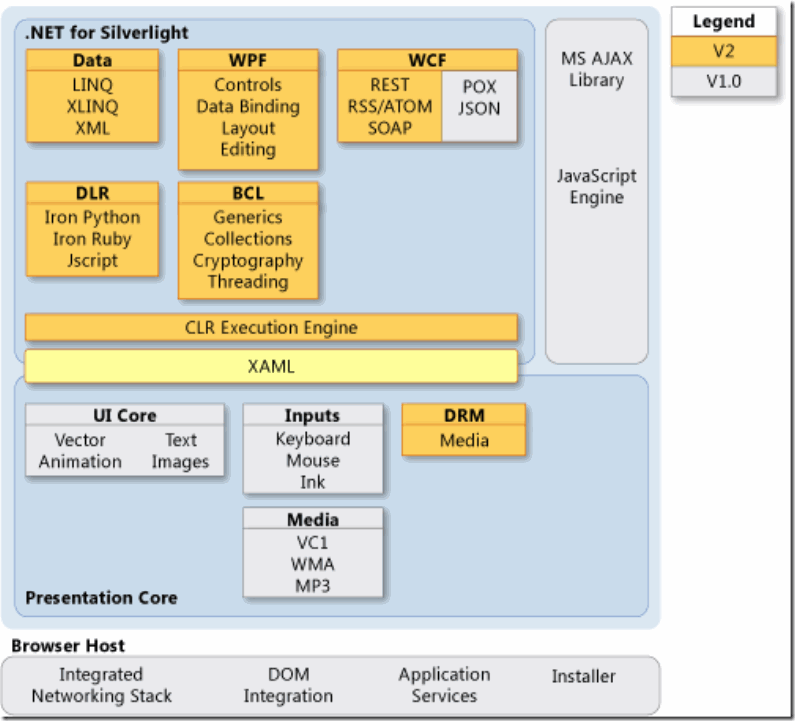
* 1. Sự kết hợp của WPF và XAML
  2. Mở rộng cho ngôn ngữ kịch bản
  3. Sự tích hợp với các ứng dụng đã có
  4. Sử dụng mô hình ngôn ngữ lập trình trên nền tảng .Net Framework và các công cụ để kết hợp.
  5. Hỗ trợ mạng
  6. Hỗ trợ ngôn ngữ tích hợp truy vấn (LINQ)

#### 1.3. Kiến trúc và các thành phần

Về cơ bản Silverlight là một nền tảng thống nhất của nhiều thành phần khác nhau. Dưới đây là bảng các thành phần chính của silverlight.

|  |  |
| --- | --- |
| **Thành phần** | **Diển tả** |
| Nền tảng trình bày cơ sở | Các thành phần và dịch vụ hướng tới giao diện người dùng và tương tác người dùng, bao gồm các control cho dữ liệu cho người dùng nhập, thiết bị đa phương tiện, quản lý phân quyền số, trình bày dữ liệu, đồ họa vector, chữ, hình ảnh động. cũng bao gồm XAML để đặc tả việc bố trí giao diện |
| .Net Framework cho silverlight | Là một gói nhỏ trong .Net Framework, bao gồm các thành phần và cá thư viện, kể cả tương việc tương tác dữ liệu, khả năng mở rộng các control, mạng, garbage collection, và CLR |
| Cài đặt và cập nhật | Là thành phần để sử lý các tiến trình cài đặt làm sao để đơn giản hóa cho lần cài đặt đầu tiên, tiếp sau đó chỉ cung cấp cơ chế tự động cập nhật và tương tác ở mức thấp. |

Dưới đây là hình ảnh mô tả những thành phần trong kiến trúc của Silverlight cùng với các thành phần và dịch vụ liên quan khác.

[](http://msdnvietnam.net/cfs-file.ashx/__key/CommunityServer.Blogs.Components.WeblogFiles/cuong/clip_5F00_image003_5F00_04AFB10A.gif)

**Core Presentation Components: (các thành phần chính)**

|  |  |
| --- | --- |
| **Tính năng** | **Mô tả** |
| Dữ liệu vào (input) | Xử lý dữ liệu đầu vào từ các thiết bị phần cứng như bàn phím, chuột, bảng vẽ hoặc các thiết bị đầu vào khác |
| Trình bày giao diện người dùng (UI Rendering) | Trình bày vector và các đồ hoạ ảnh bitmap, ảnh động, và văn bản |
| Thiết bị nghe nhìn (Media) | Các tính năng phát và quản lý một vài thể loại file âm thanh và hình ảnh như .WMP và .MP3 |
| Controls | Hỗ trợ mở rộng cho các control để có khả năng tùy chỉnh về kiểu dáng và khuôn mẫu |
| Xếp đặt Layout | Cho phép khả năng xếp đặt vị trí động các thành phần giao diện người dùng |
| Trình bày dữ liệu  (Data Binding) | Cho phép việc kết nối dữ liệu của các đối tượng và các thành phần giao diện người dùng |
| DRM | Khả năng Quản lý phân quyền số |
| XAML | Cung cấp trình phân tách cho XAML |

Các lập trình viên có thể tương tác với thành phần nền tảng trình bày cơ sở trên đây bằng cách sử dụng XAML để đặc tả. XAML là một yếu tố quan trọng nhất trong việc tương tác giữa .Net Framwork và các kiểu trình bày Layout, ngoài ra các lập trình viên cũng có thể sử dụng cơ chế quản lý code bên trong để thao tác với lớp trình bày

**.Net Frame work for Silverlight:**

|  |  |
| --- | --- |
| **Tính năng** | **Mô tả** |
| Data | Hỗ trợ ngôn ngữ truy vấn tích hợp (LINQ) và LINQ với đặc tả XML, dễ dang xử lý việc tích hợp và làm việc với dữ liệu từ nhiều nguồn khác nhau. Hỗ trợ việc sử dụng XML và các lớp biên đổi hóa (serialization) để xử lý dữ liệu |
| Base class library | Thuộc thư viện của .Net Framework, nó cung cấp các chức năng lập trình chủ yếu như việc xử lý chuỗi, biểu thức chính quy, đầu vào và đầu ra, ánh xạ, tập hợp và toàn cục hóa. |
| Window Communication Foundation  (WCF) | Cung cấp các tính năng để đơn giản hóa việc truy cập dữ liệu từ xa. Cơ chế này bao gồm một đối tượng trình duyệt, HTTP request và HTTP Response, RSS, JSON, POX, và các SOAP |
| Common language runtime (CLR) | Cung cấp việc quản lý bộ nhớ, dọn dẹp bộ nhớ thừa, xử lý ngoại lệ… |
| Windows Presentation Foundation controls  (WPF) | Cung cấp các control giầu tính năng như Button, Calendar, CheckBox, DataGrid, DatePicker, HyperlinkButton, ListBox, RadioButton, and ScrollViewer. |
| Dynamic language runtime  (DLR) | Hỗ trợ việc biên dịch và thi hành với tính năng động của các ngôn ngữ kịch bản như Javascript và IronPython cho các chương trình trên nền tảng Silverlight. |

#### 1.4. Mô hình lập trình

* + - Javascript API
    - Managed API

#### 1.5. Công nghệ và công cụ liên quan của Silverlight

**Microsoft Expression Blend:**Sử dụng công cụ này bạn bạn có thể tạo và thay đổi cách xắp xếp trình bày Layer của ứng dụng bằng cách thao tác đến canvas và control trong XAML, làm việc với các chức năng đồ họa, Lập trình với ngôn ngữ Javascript.

**Visual Studio 2008:**Visual Studio cung cấp các công cụ hiệu quả cho việc phát triển các ứng dụng có hỗ trợ thao tác code bên tron. Tất cả các phiên bản đã có của Visual Studio đều có khả năng hỗ trợ Silverlight,Tuy nhiên ở phiên bản mới này nó còn hỗ trợ các tính năng đặc biệt hơn như bao gồm khả năng IntelliSense, debugging và các template cho việc tạo mới một ứng dụng Silverlight.

**ASP.NET AJAX:**Bao gồm tập các Control, Service, và các thư viện cần thiết cho việc tạo và tương tác với nền ứng dụng web

**Microsoft ASP.NET 3.5 Extensions Preview:**Công nghệ này cung cấp chức năng thêm để việc tăng cường các ứng dụng ASP.NET AJAX. Nó bao gồm 2 control sử dụng hữu ích cho việc xây dựng ứng dụng nền tảng silverlight cũng như là một phần của ứng dụng ASP.NET:

- ASP.NET MediaPlayer Server Control

- ASP.NET Silverlight Server Control

**Internet servers**: Bao gồm IIS (Microsoft Internet Information Services), và Apache Web server

**Microsoft Windows Communication Foundation (WCF**) services**.**

### ****Chương 2: Layout trong silverlight****

Khi xây dựng ứng dụng Silverlight, một trong những điều cần quan tâm là việc bố trí giao diện đồ họa như thế nào để một developer có thể tận dụng những sức mạnh về đồ họa của Silverlight, trước hết chúng ta cần phải biết được những đối tượng cơ bản dành cho việc sắp xếp, thiết kế các thành phần UI. Silverlight cung cấp cho ta 3 kiểu bố trí khác nhau đó là: Canvas, StackPanel và Grid

#### 2.1.Canvas

Định nghĩa một khu vực mà trong đó bạn có thể chỉ ra vị trí của từng đối tượng thành phần bằng cách sử dụng tọa độ tham chiếu. Bạn có thể sử dụng lồng các Canvas với nhau và những thành phần bên trong của Canvas phải là một UIElement. Trong nhiều trường hợp thì Cancas chỉ đóng vài trò như một đối tượng để chứa đựng những đối tượng khác và không có một thuộc tính hiển thị nào.

Canvas là đối tượng layout duy nhất cho phép định vị đối tượng dựa vào giá trị tọa độ tuyệt đối, thông qua 2 thuộc tính được sử dụng dưới dạng Attached Property là Canvas.Top và Canvas.Left

Trong quá trình gõ đoạn code trên trong Visual Studio để thử nghiệm, khi bạn gõ đến Canvas., công nghệ IntelliSense sẽ hiện ra một menu pop-up cho thấy các thuộc tính dạng AttachedProperty của Canvas. Khi đó, bạn có thể thấy thuộc tính ZIndex. Tính chất của thuộc tính này là giải quyết vấn đề khi các đối tượng trên Canvas che phủ lên nhau (overlap). Nếu như 2 đối tượng có ZIndex như nhau hoặc cùng không định nghĩa ZIndex thì đối tượng nào được khai báo sau sẽ đè lên trên. Tuy nhiên, nếu 2 đối tượng có giá trị thuộc tính ZIndex khác nhau, thì ZIndex của đối tượng nào lớn hơn sẽ giúp đối tượng được ưu tiên xuất hiện phía trên.

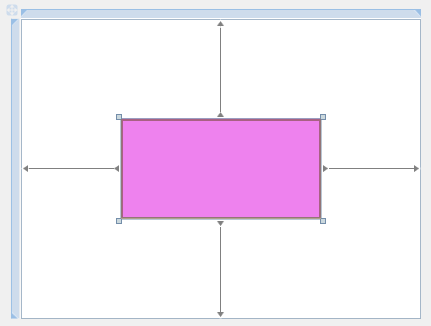
Ví dụ minh họa: chỉ ra hình chữ nhật cách trái 20pixel và cách trên 53pixel

<Grid x:Name="LayoutRoot" Background="White">

<Rectangle Width="200" Height="100" Canvas.Left="20" Canvas.Top="53" Fill="Violet">

</Rectangle >

</Grid>



Hình 2.1: sử dụng thẻ canvas

#### 2.2. StackPanel

- Sắp xếp những thành phần bên trong nó thành một dòng và có thể hiển thị theo hai kiểu ngang hoặc dọc.

- StackPanel hay được dùng khi muốn sắp xếp chỉ trong một phần của toàn bộ trang.

- Gía trị mặc định được gán cho thuộc tính *Orientation* là chiều dọc(*Vertical)* và giá trị mặc định được xét cho hai thuộc tính *HorizontalAlignment* và *VerticalAlignment* là *Stretch*.

- Ví dụ: StackPanel sắp xếp các nút theo chiều ngang

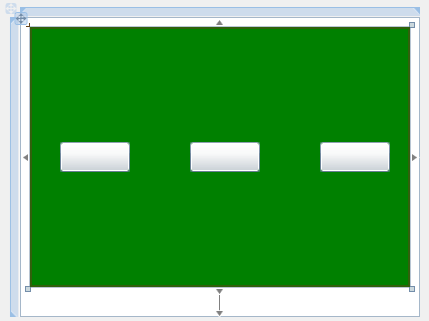
<StackPanel Orientation="Horizontal" Background="Green" Margin="10,10,10,30">

<Button Width="70" Height="30" Margin="30"></Button>

<Button Width="70" Height="30" Margin="30"></Button>

<Button Width="70" Height="30" Margin="30"></Button>

</StackPanel>



Hình 2.2: StackPanel

#### 2.3. Grid

- Grid là layout panel mềm dẻo nhất, nó cho phép sắp xếp các control theo dạnh bảng bao gồm nhiều dòng và nhiều cột. Về tính năng nó tương tự như thẻ Table trong HTML.

- Mặc định Grid chưa một cột và một dòng.

- Để định nghĩa nhiều cột hoặc nhiều dòng ta dùng ColumnDefinitions và RowDefinitions. Mỗi ColumnDefiniton và RowDefinitons trong ColumnDefinitions và RowDefinitions xác định một dòng hoặc cột.

- Ví dụ sử dụng Grid

<Grid x:Name="LayoutRoot" Background="White" ShowGridLines="True" Margin="30,30,10,10">

<Grid.RowDefinitions>

<RowDefinition ></RowDefinition>

<RowDefinition ></RowDefinition>

<RowDefinition ></RowDefinition>

<RowDefinition ></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="80"/>

<ColumnDefinition Width="90"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button Content="button1" Grid.Row="0" Grid.Column="0" Width="70" Height="30"/>

<Button Content="button2" Grid.Row="1" Grid.Column="1" Width="70" Height="30"/>

<Button Content="button3" Grid.Row="2" Grid.Column="2" Width="70" Height="30"/>

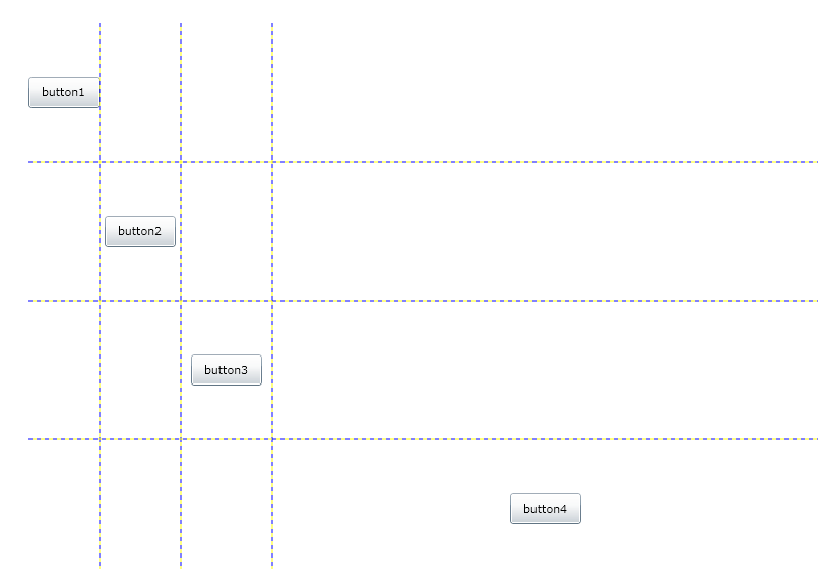
<Button Content="button4" Grid.Row="3" Grid.Column="3" Width="70" Height="30"/>

</Grid>

*- Width="Auto":* Đặt kích thước tự động theo kích thước của nội dung nó chứ bên trong

- Width="90": Đặt kích thước 1 cách tuyệt đối

- Width="\*": Cột này sẽ chiếm toàn bộ phần còn lại



Hình 2.3: Grid

### Chương 3: Các control cơ bản

#### 3.1. Border

- Border cung cấp một background, khu vực cho phép bạn nhúng vào đó một thành phần duy nhất nhằm tạo ra hiệu ứng về đồ họa

- Ví dụ:

<Border Background="Yellow" Width="350" Padding="40" CornerRadius="30">

<TextBlock FontSize="20">Dùng border để bo góc </TextBlock>

</Border>



Hình 3.1: Border

#### 3.2. Button

- Button kiểm soát các phản ứng của người dùng qua các thiết bị đầu vào, bao gồm các sự kiện, style.

- Ví dụ:

<Button Content="Click here" Height="30" Width="100" Background="Violet" Click="Button\_Click" />



Hình 3.2: Button

#### 3.3. Calendar

- Calendar cho phép người sử dụng chọn ngày tháng năm. Ta có thể sử dụng các style khác nhau thông qua Mode.

- Ví dụ:

<Grid x:Name="LayoutRoot" Background="White">

<sdk:Calendar Height="169" Name="calendar1" Width="230" />

</Grid>



Hình 3.3: Calendar

#### 3.4. CheckBox

- Cho phép người dùng lựa chọn 3 trạng thái đánh dấu, bỏ đánh dấu và trạng thái trung gian

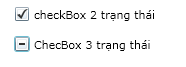
- Ví dụ:

<Grid x:Name="LayoutRoot" Background="White">

<CheckBox Name="checkbox1" ClickMode="Release" Content="checkBox 2 trạng thái" IsThreeState="False" Margin="80,20,0,0 "> </CheckBox>

<CheckBox Name="checkbox2" ClickMode="Release" IsThreeState="True" Content="ChecBox 3 trạng thái" Margin="80,50,0,0"></CheckBox>

</Grid>



Hình 3.4: CheckBox

#### 3.5. ComboBox

- Cho phép lựa chọn một Item từ một danh sách Item từ ComboBox

- Ví dụ:

<Grid x:Name="LayoutRoot" Background="White">

<ComboBox Name="ComboBox" BorderThickness="3" IsDropDownOpen="True" SelectedIndex="0" Width="90" Height="30">

<ComboBoxItem x:Name="ComboBox1" Content="Item1"></ComboBoxItem>

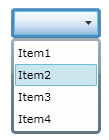
<ComboBoxItem x:Name="ComboBox2" Content="Item2"></ComboBoxItem>

<ComboBoxItem x:Name="ComboBox3" Content="Item3"></ComboBoxItem>

<ComboBoxItem x:Name="ComboBox4" Content="Item4"></ComboBoxItem>

</ComboBox>

</Grid>



Hình 3.5: ComboBox

#### 3.6. ContentControl

- ContentControl có các dẫn xuất như Button, panel. Ta có thể tùy chỉnh ContentControl thông qua template.

- Ví dụ: Cài đặt một số control được thừa kế từ ContentControl

<Grid x:Name="LayoutRoot" Background="White">

<StackPanel Name="root" Width="300"

HorizontalAlignment="Center"VerticalAlignment="Center">

<!-- Tạo một nút với nội dung gán từ content-->

<Button Margin="10" Content="Đây là nội dung của 1 nút" />

<!--Tạo một nút với UIElement gán từ content-->

<Button Margin="10">

<Rectangle Height="40" Width="40" Fill="Violet"/>

</Button>

<!--Tạo một button với một panel chưa nhiều đối tượng gán từ content-->

<CheckBox Margin="10">

<StackPanel Margin="3,0,0,0" Orientation="Horizontal">

<Ellipse Height="10" Width="10" Fill="Violet"/>

<TextBlock TextAlignment="Center" Text="A string of text">

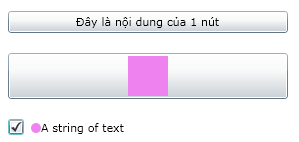
</TextBlock>

</StackPanel>

</CheckBox>

</StackPanel>

</Grid>



Hình 3.6: ContentControl

#### 3.7. DatePicker

- Cho phép người sử dụng lựa chọn ngày hoặc gõ trực tiếp vào texbox hoặc sử dụng lịch thả xuống

DataPicker cung cấp một loạt các sự kiện hỗ trợ việc kiểm soát lịch thả xuống

- Ví dụ:

<StackPanel>

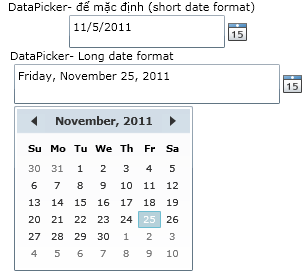
<TextBlock Width="303" >DataPicker- để mặc định (short date format)</TextBlock>

<sdk:DatePicker Name="dp1" Width="180" Height="33" />

<TextBlock Width="298">DataPicker- Long date format</TextBlock>

<sdk:DatePicker Name="dp2" SelectedDateFormat="Long" Width="290" Height="40" />

</StackPanel>



Hình 3.7: DatePicker

#### 3.8. GridSpliter

- GridSplitter cho phép người dùng phân phối lại không gian giữa các hàng hoặc cột. Nó là yếu tố đồ họa tượng trưng cho một thành phần gắn vào nó mà thành phần đó có thể tương tác.

- Ví dụ:

<Grid ShowGridLines="True" Canvas.Top="60" Canvas.Left="25" Width="390" Height="390">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Text="1" Grid.Row="0" Grid.Column="0"/>

<TextBlock Text="2" Grid.Row="1" Grid.Column="1"/>

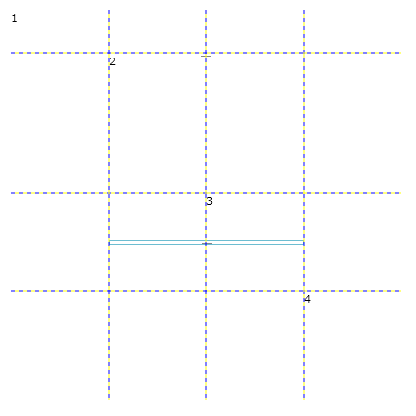
<TextBlock Text="3" Grid.Row="2" Grid.Column="2"/>

<TextBlock Text="4" Grid.Row="3" Grid.Column="3"/>

<sdk:GridSplitter Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="4" Height="5" HorizontalAlignment="Stretch" VerticalAlignment="Top" ShowsPreview="True"/>

<sdk:GridSplitter Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="2" Height="5" HorizontalAlignment="Stretch" ShowsPreview="True"/>

</Grid>



Hình 3.8: GridSpliter

#### 3.9. HyperlinkButton

- HyperlinkButton hiển thị một siêu liên kết. Khi bấm vào HyperlinkButton sẽ cho phép người dùng truy cập vào một trang web, địa chỉ URL chỉ được xác định với các NavigateUrl. Bạn có thể chỉ định cửa sổ hoặc một khung trong những điểm đến trang web bằng cách sử dụng TargetName.

- Ví dụ:

<Grid x:Name="LayoutRoot" Background="White">

<TextBlock Text="HyperlinkButton"

Margin="134,51,71,221" Width="200" Height="35"

FontFamily="Times New Roman" FontSize="15" FontWeight="Bold"

Foreground="Blue"/>

<HyperlinkButton Content=">> vào trang google"

NavigateUri="http://www.google.com.vn/"

TargetName="\_Blank" Margin="84,100,84,169"

FontSize="18" Width="200" Height="35"/>

</Grid>



Hình 3.9: HyperlinkButton

#### 3.10. Image

- Hiển thị hình ảnh trong định dạng jpeg hoặc png.

- Ví dụ:

<Grid x:Name="LayoutRoot" Background="White">

<Image Source="Sunset.jpg"></Image>

</Grid>



Hình 3.10: Image

#### 3.11. InkPresenter

- InkPresenter cung cấp các bản vẽ mặt để hỗ trợ tính năng TabletPC. InkPresenter được dẫn xuất từ canvas nên có thể hiển thị một hoặc nhiều đối tượng.

- Ví dụ:

*XAML*

<Grid x:Name="LayoutRoot" Background="White">

<Rectangle Height="400" Width="400" Margin=" 50,40,0,0" Stroke="Black"/>

<InkPresenter x:Name="Caro" Height="400" Width="400"

Margin="50,50,0,0" MouseLeftButtonDown="Caro\_MouseLeftButtonDown"

LostMouseCapture="Caro\_LostMouseCapture"

MouseMove="Caro\_MouseMove"

Background="Transparent" Opacity="1"/>

</Grid>

*Mã C#:*

using System.Windows.Ink;

Stroke newStroke;

private void Caro\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

Caro.CaptureMouse();

StylusPointCollection MySPC = new StylusPointCollection();

MySPC.Add(e.StylusDevice.GetStylusPoints(Caro));

newStroke = new Stroke(MySPC);

Caro.Strokes.Add(newStroke);

}

private void Caro\_LostMouseCapture(object sender, MouseEventArgs e)

{

//nếu để thì kẻ được đường thẳng

//if (newStroke != null)

//{

// newStroke.StylusPoints.Add(e.StylusDevice.GetStylusPoints(Caro));

//}

newStroke = null;

}

private void Caro\_MouseMove(object sender, MouseEventArgs e)

{

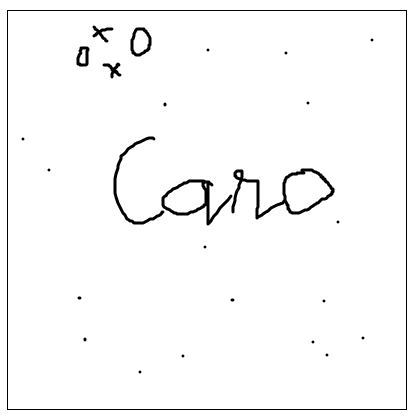
if (newStroke != null)

{

newStroke.StylusPoints.Add(e.StylusDevice.GetStylusPoints(Caro));

}

}



Hình 3.11: InkPresenter

#### 3.12. ListBox

- ListBox chứa một danh sách các item, item có thể là một đoạn văn bản hay một control.

- Ví dụ:

<TextBlock Text="ListBox chứa các item" Margin="102,90,0,0"/>

<ListBox Height="131" HorizontalAlignment="Left"

Margin="102,111,0,0" Name="listBox1"

VerticalAlignment="Top" Width="207" >

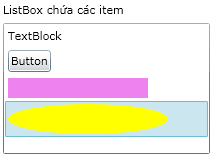
<TextBlock>TextBlock</TextBlock>

<Button Content="Button"/>

<Rectangle Fill="Violet" Height="20" Width="140"></Rectangle>

<Ellipse Fill="Yellow" Height=" 30" Width="160"/>

</ListBox>



Hình 3.12: LisBox

#### 3.13. MediaElement

- Các MediaElement kiểm soát nội dung Audio hoặc Video. MediaElement cung cấp một vùng chữ nhật có thể hiển thị video hoặc audio trên đó.

- Ví dụ:

<MediaElement x:Name="media" Source="Be what you wanna be - Darin.mp3"

CurrentStateChanged="media\_CurrentStateChanged"

Width="300" Height="300"/>

#### 3.14. MultiSaleImage

- MultiScalelmage cho phép người dùng mở một hình ảnh cho phép phóng to thu nhỏ thông qua công nghệ DeepZoom.

- Ví dụ:

XAML:

<Grid x:Name="LayoutRoot" Background="White">

<MultiScaleImage x:Name="caleimage" Source="source/item.bin" MouseEnter="caleimage\_MouseEnter"/>

</Grid>

C#:

private void caleimage\_MouseEnter(object sender, MouseEventArgs e)

{

if (caleimage.UseSprings==false)

{

caleimage.UseSprings = true;

}

this.caleimage.ZoomAboutLogicalPoint(3, 0.5, 0.5);

}

#### 3.15. PasswordBox

- PasswordBox được sử dụng để nhập vào thông tin mật khẩu, ký tự hiển thị chỉ là ký tự đại diện

- Ví dụ:

<StackPanel>

<TextBlock Text="Mật khẩu" Width="116" Margin="10,20,0,0" />

<PasswordBox x:Name="Pass" PasswordChanged="Pass\_PasswordChanged" Width="300" Height="24" />

</StackPanel>



Hình 3.13: PasswordBox

#### 3.16. Popup

- Popup luôn luôn xuất hiện lên trên cùng cửa sổ ứng dụng hữu ích cho việc hiển thị thông báo cho một nhiệm vụ.

- Ví dụ:

XAML:

<StackPanel x:Name="LayoutRoot" Background="White">

<Button Width="100" Height="50" x:Name="showPopup"

Click="showPopup\_Click" Content="Show Popup" />

</StackPanel>

C#:

Popup p = new Popup();

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

{

Border border = new Border();

border.BorderBrush = new SolidColorBrush(Colors.Blue);

border.BorderThickness = new Thickness(5.0);

}

#### 3.17. ProgressBar

- ProgressBar hiển thị cho biết tiến trình đang hoạt động bằng việc thay đổi giá trị thay đôi.

- Ví dụ:

<Grid Height="250" Width="300">

<Border BorderThickness="5" BorderBrush="Black" >

<StackPanel Background="Violet">

<TextBlock HorizontalAlignment="Center" Margin="10" Text="value \_based progress bar"/>

<ProgressBar x:Name="pgb1" Value="100" Margin="10" Maximum="200"

Height="15" IsIndeterminate="False"/>

<TextBlock HorizontalAlignment="Center"

Margin="10" Text="Indeterminate progress bar(VÔ HẠN)"/>

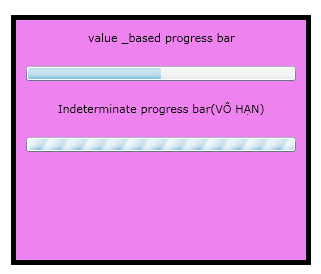
<ProgressBar x:Name="pgb2" Value="100" Margin="10" Maximum="200"

Height="15" IsIndeterminate="True"/>

</StackPanel>

</Border>

</Grid>



Hình 3.14: ProgressBar

#### 3.18. RadioButton

- Cho phép người dùng chọn một tùy chọn từ một nhóm các tùy chọn. Các RadioButton tổ chức trong cùng một nhóm bằng cách đặt các GroupName chung cho mỗi button. Các RadioButton là một ContentControl.

- Ví dụ:

<StackPanel Height="200" Width="250">

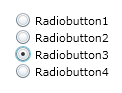
<RadioButton Width="200" Content="Radiobutton1" />

<RadioButton Width="200" Content="Radiobutton2"/>

<RadioButton Width="200" Content="Radiobutton3"/>

<RadioButton Width="200" Content="Radiobutton4"/>

</StackPanel>



Hình 3.15: RadioButton

#### 3.19. RepeatButton

- RepeatButton là một nút kiểm soát các sự kiện tác động vào nút như Mouse Click và thời gian giữa 2 lần click liên tiếp.

- Ví dụ:

XAML:

<StackPanel Height="200" Width="250">

<RepeatButton x:Name="RepeatButton"

Content="Bấm và giữ cho nhiều sự kiện click"

Click="RepeatButton\_Click"/>

<TextBlock x:Name="ClickTextBlox" Text="Số lần clicks: "></TextBlock>

</StackPanel>

C#:

static int clicks = 0;

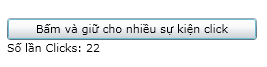
private void RepeatButton\_Click(object sender, RoutedEventArgs e)

{

clicks += 1;

ClickTextBlox.Text = "Số lần Clicks: " + clicks;

}



Hình 3.16: RepeatButton

#### 3.20. ScrollBar

- ScrollBar cung cấp một thanh trượt có khả năng di chuyển định hướng theo chiều ngang hoặc theo chiều dọc.

#### 3.21. Scrollviewer

- ScrollViewer kiểm soát một mảng content, cung cấp 2 ScrollBar để kiểm soát việc di chuyển mảng content.

ScrollViewer cho phép bạn thiết lập kích cỡ của khoảng View liên quan đến Content theo chiều ngang và chiều dọc.

- Ví dụ:

<StackPanel Height="200" Width="250">

<ScrollViewer Height="200" HorizontalScrollBarVisibility="Auto" >

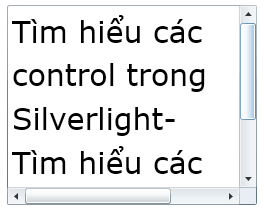
<TextBlock Width="300" TextWrapping="Wrap" FontSize="30"

Text="Tìm hiểu các control trong Silverlight-

Tìm hiểu các control trong Silverlight"/>

</ScrollViewer>

</StackPanel>



Hình 3.17: Scrollviewer

#### 3.22. Slider

- Slider cho phép người dùng lựa chọn từ một loạt các giá trị bằng cahs di chuyển dọc theo một thanh ngang.

- Ví dụ:

<StackPanel Height="200" Width="250">

<TextBlock> Basic Slider</TextBlock>

<Slider x:Name="slider1" Minimum="0" Maximum="10"

ValueChanged="slider1\_ValueChanged"/>

</StackPanel>



Hình 3.18: Slider

#### 3.23. TabControl

- TabControl cung cấp một giao diện hiển thị cho các thành phần. Các thành phần con của TabControl là các TabItem.

- Ví dụ:

<Grid >

<StackPanel Height="300" Width="400">

<TextBlock>TabControl</TextBlock>

<sdk:TabControl x:Name="tab1" Canvas.Top="20" Canvas.Left="20"

Height="200" Width="300">

<sdk:TabControl.Background>

<SolidColorBrush Color="Green" Opacity="0.4"/>

</sdk:TabControl.Background>

<sdk:TabItem Header="Hình tròn">

<StackPanel>

<Ellipse Fill="BurlyWood"

Width="100" Height="100"

Stroke="Blue" StrokeThickness="5"/>

</StackPanel>

</sdk:TabItem>

<sdk:TabItem Header="Hình vuông">

<Rectangle Fill="BurlyWood" Width="100" Height="100"

Stroke="Khaki" StrokeThickness="5"/>

</sdk:TabItem>

<sdk:TabItem Header="Đa giác">

<Polygon Points="50,100,100,50,60,150" Fill="Violet"

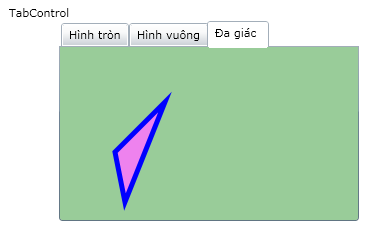
Stroke="Blue" StrokeThickness="5"/>

</sdk:TabItem>

</sdk:TabControl>

</StackPanel>

</Grid>



Hình 3.19: TabControl

#### 3.24. TextBox

- Được sử dụng nhập dữ liệu dạng text từ người dùng hoặc hiển thị văn bản.

TextBox thường được dùng để chỉnh sửa văn bản đôi khi cũng được dùng ở chế độ chỉ đọc, TextBox có thể hiển thị nhiều dòng văn bản.

- Ví dụ:

<StackPanel Height="300" Width="400">

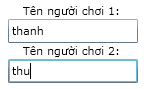
<TextBlock x:Name="ng1" Width="100"> Tên người chơi 1:</TextBlock>

<TextBox x:Name="Nguoichoi1" Width="130"/>

<TextBlock x:Name="ng2" Width="100"> Tên người chơi 2:</TextBlock>

<TextBox x:Name="Nguoichoi2" Width="130"/>

</StackPanel>



Hình 3.20: TextBox

#### 3.25. TextBlock

- TexBlock hiển thị nội dung văn bản, bạn có thể gán các nội dung của một TexBlock bằng cách sử dụng nguồn text. Ngoài ra có thể gán TextBlock bằng một tập hợp các đối tượng text.

- Ví dụ:

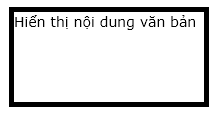
<Border BorderThickness="5" BorderBrush="Black" Width="200" Height="100">

<TextBlock Canvas.Top="20" Canvas.Left="20" FontSize="14" >

Hiển thị nội dung văn bản

</TextBlock>

</Border>



Hình 3.21: TextBlock

#### 3.26. ToolTip

- ToolTip cung cấp thông tin về một đối tượng đồ họa bằng cách sử dụng một của sổ pupup.

- Ví dụ:

<StackPanel Height="300" Width="400">

<Image Source="source/Leaves1.png" Width="200" Height="150">

<ToolTipService.ToolTip>

<ToolTip Content="Leaves"/>

</ToolTipService.ToolTip>

</Image>

</StackPanel>



Hình 3.22: ToolTip

#### 3.27. DataGrid

- DataGrid cung cấp một cách linh hoạt để hiển thị một tập hợp các dữ liệu trong các hàng và cột.

DataGrid hỗ trợ nhiều Style hiển thị dữ liệu trong các cột hoặc hàng.

DataGrid hỗ trợ các bảng tùy chọn định dạng và khả năng hiển thị hoặc ẩn các tiêu đề, các thanh trượt di chuyển.

### Chương 4: Xử lý đồ họa trên Silverlight

#### 4.1. Giới thiệu

- Silverlight cung cấp nhiều lựa chọn cho việc thêm các tính năng trực quan thú vị cho ứng dụng. Ta có thể sử dụng vẽ, Shape, Path và những hình học phức tạp. Những khu vực được xác định bởi dạng hình thì có thể tô hiệu ứng, như là ảnh, dải màu hoặc là đoạn video.

- Dưới đây là cá đối tượng đồ họa như Ellipse, Line, Path, Geometries…

#### 4.2. Shapes và Drawing

- Trong silverlight, Shapes là kiểu UIElement nên ta có thể dễ dàng hiển thị một đối tượng dạng Shape lên màn hình. Những Shape đều có chung những đặc tính dưới đây:

* Stroke: dùng để vẽ viền ngoài của Shape
* StrokeThickness: độ dày của viền ngoài của Shape.
* Fill: Mô tả cách phía bên trong của Shape được vẽ.
* Data: là một đặc tính chỉ rõ tọa độ và các đỉnh được định nghĩa tùy theo thông tin đầu vào.

##### 4.2.1. Ellipse

- Bạn có thể tạo một Ellipse bằng cách xác định hai thuộc tính cơ bản là rộng và cao như ví dụ sau:

<StackPanel Height="300" Width="400">

<Ellipse Fill="Violet"

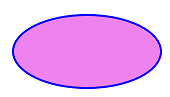
Height="150" Width="300"

Stroke="Blue"

StrokeThickness="4">

</Ellipse>

</StackPanel>



##### 4.2.2. Line

- Cho phép bạn vẽ một đoạn thẳng giữa hai điểm. Ví dụ sau chỉ ra một số cách để bạn xác định tọa độ của đoạn thẳng và các đặc tính Stroke của nó.

- Ví dụ:

<Canvas Height="300" Width="400">

<!--Vẽ một đường thẳng nằm ngang từ tọa độ (0,100) đến (200,100)-->

<Line X1="0" Y1="100" X2="200" Y2="100" Stroke="Blue" StrokeThickness="2"/>

<!--Vẽ một đường thẳng từ tọa độ (0,0) đến (200,200)-->

<Line X1="0" Y1="0" X2="200" Y2="200" Stroke="Blue" StrokeThickness="2"/>

<!--Vẽ một đường thẳng từ tọa độ (0,0) đến (200,200)

và dịch chuyển nó tới gốc tọa độ mới thêm 100px về phía bên phải-->

<Line X1="0" Y1="0" X2="200" Y2="200" StrokeThickness="4" Canvas.Left="100">

<Line.Stroke>

<RadialGradientBrush GradientOrigin="0.5,.5"

Center=".5,.5" RadiusX="0.5"

RadiusY="1.5">

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

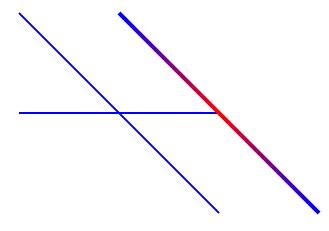
<GradientStop Color="Blue" Offset="1"/>

</RadialGradientBrush>

</Line.Stroke>

</Line>

</Canvas>



##### 4.2.3. Path

- Lớp Path cho phép bạn vẽ những hình cong và những hình phức tạp. Những hình cong và dạng hình (Shape) được diễn tả thông qua việc sử dụng đối tượng Geometry. Để sử dụng Path, bạn tạo một Geometry và dùng nó để xét cho đặc tính Data của đối tượng Path. Có nhiều loại đối tượng Geometry khác nhau để bạn lựa chọn: LineGeometry, RectangleGeometry… Để tạo những dạng hình phức tạp hoặc hình tròn ta sử dụng PathGeometry.

Thẻ <Path /> được sử dụng để tạo đường cong Bezier.

Thuộc tính Data="M 10,100 C 100,0 200,200 300,100 z" khai báo các thông số tạo nên một Path. Trong đó M 10,100 nghĩa là đường cong bắt đầu từ điểm có tọa độ (10,100) tính theo hệ tọa độ của Canvas chứa Path này. Các thông số của thuộc tính Data có phân biệt chữ hoa, chữ thường. Nếu là chữ hoa thì tọa độ điểm được tính theo vị trí tuyệt đối, chữ thường thì tọa độ được tính theo vị trí tương đối. Ví dụ, M khai báo tọa độ điểm bắt đầu của Path tính theo vị trí tuyệt đối, còn nếu thay bằng m thì sẽ hiểu là vị trí tương đối. Ký tự C dùng để khai báo hai điểm điều khiển (Control Point) của đường cong. Ví dụ với C 100,0 200,200 thì hai điểm điều khiển sẽ có tọa độ là (100,0) và (200,200). Sau hai điểm điều khiển là điểm kết thúc của đường cong, trong ví dụ trên, tọa độ điểm kết thúc là (300,100).

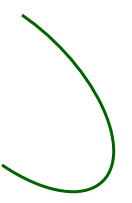
- Ví dụ sau sử dụng cú pháp rút gọn để biểu diễn một dạng hình phức tạp

<Canvas Height="300" Width="400">

<Path Stroke="DarkGreen" StrokeThickness="3"

Data="M 80,200 A 100,50 45 1 0 100,50" />

</Canvas>



* M: Chữ cái M đại diện cho một hành động di chuyển và di chuyển đến điểm nhất định từ điểm hiện tại.
* Cú pháp M  *điểm bắt đầu*
* Ví dụ: M 80,200 lệnh di chuyển từ điểm hiện tại đến điểm (80,200).
* L: Vẽ một đường thẳng giữa điểm hiện tại và điểm kết thúc quy định.
* Cú pháp: L  *điểm cuối*
* Ví dụ: L 100,200
* H: tạo ra một đường thẳng nằm ngang giữa điểm hiện tại và điểm chỉ định về phía trục x.
* Cú pháp : H  *x*

x: Tọa độ của điểm cuối của đường kẻ

* Ví dụ: H 90
* V: tạo ra một đường thẳng đứng giữa điểm hiện tại và điểm chỉ định về phía trục y.
* Cú pháp: V  *y*

y: Tọa độ của điểm cuối của đường kẻ

* Ví dụ: V 30
* C: điểm hút, tạo ra một đường cong giữa 2 điểm quy định.
* Cú pháp: C  *ĐiểmĐầuTiên* *Điểm2* *ĐiểmĐượcTạoRa*

*ĐiểmĐầuTiên: quyết định tiếp tuyến bắt đầu của đường cong.*

*Điểm2: quyết định tiếp tuyến kết thúc của đường cong.*

* *Ví dụ: C 100,200 200,400 300,200*
* T: vẽ một đường cong từ điểm đầu đến điểm thứ 2 và điểm đầu tiên được sử dụng như các điểm kiểm soát
* Cú pháp: T  *ĐiểmKiểmSoát* *ĐiểmĐượcTạoRa*

*ĐiểmKiểmSoát: quyết định tiếp tuyến của đường cong.*

* Ví dụ: T 50 20
* Z: đóng đường hiện tại bằng cách vẽ một đường thẳng từ điểm hiện tại tới điểm khởi đầu.
* Cú pháp: Z

***Vẽ bàn cờ caro***

*Xaml:*

<Canvas Canvas.Left="20" Width="490" Height="490" Canvas.Top="30">

<!--Vẽ một đường thẳng từ tọa độ (4,180) đến (490,180)

và dịch chuyển nó tới gốc tọa độ mới thêm 30px về phía dưới cứ như vậy ta sẽ vẽ được

các đoạn thẳng tiếp theo. Tương tự cho các đường dọc-->

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="30" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="60" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="90" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="120" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="150" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="180" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="210" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="240" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="270" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="300" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="330" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="360" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="420" Height="1" Canvas.Left="4" Canvas.Top="390" Stretch="Fill" Stroke="#FF000000" Data="M 4,180L 490,180"/>

<Path Width="1" Height="420" Canvas.Left="30" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="60" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="90" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="120" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="150" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="180" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="210" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="240" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="270" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

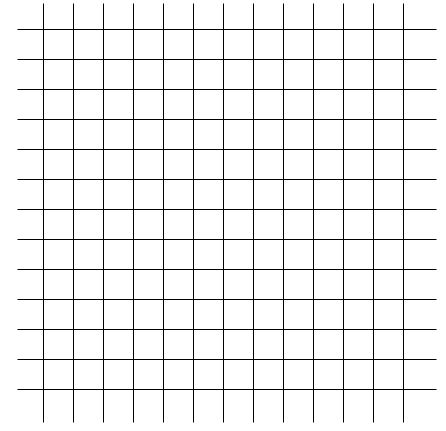
<Path Width="1" Height="420" Canvas.Left="300" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="330" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="360" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

<Path Width="1" Height="420" Canvas.Left="390" Canvas.Top="4" Stretch="Fill" Stroke="#FF000000" Data=" M 40,4L 40,490" />

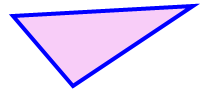
</Canvas>

Bàn cờ caro

##### 4.2.4. Polygon

- Để xác định một đa giác (Polygon) bạn cần cung cấp danh sách các điểm (Point) và tô màu cho nó qua đặc tính Fill.

- Ví dụ sau vẽ một đa giác với màu viền xanh và bên trong được tô màu tím.



Hay vẽ một hình vuông:

<Canvas Height="300" Width="400">

<Polygon Points="20,20 300,20 300,300 20,300" Stroke="Black"

StrokeThickness="2">

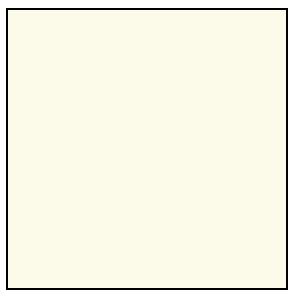
<Polygon.Fill>

<SolidColorBrush Color="Khaki" Opacity="0.2"/>

</Polygon.Fill>

</Polygon>

</Canvas>



Phân biệt 2 thuộc tính của FillRule là EvenOdd và Nonzero

* FillRule=Nonzero(Ký hiệu trong Data là F1)

<StackPanel Height="300" Width="400">

<!--<Canvas Height="300" Width="400">-->

<Polygon

Points="10,100 50,0 90,100 0,35 100,35"

Fill="Gold"

Stroke="Black"

StrokeThickness="3"

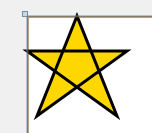
FillRule="Nonzero"

Canvas.Top="20">

</Polygon>

<!--</Canvas>-->

</StackPanel>



* FillRule= EvenOdd (Ký hiệu trong Data là F0)

<StackPanel Height="300" Width="400">

<Polygon

Points="10,100 50,0 90,100 0,35 100,35"

Fill="Gold"

Stroke="Black"

StrokeThickness="3"

FillRule="EvenOdd"

Canvas.Left="150"

Canvas.Top="20">

</Polygon>

</StackPanel>



##### 4.2.5. Polyline

- Tương tự như đối tượng Polygon ngoại trừ việc không nhất thiết Polyline phải khép kín. Bạn cũng có thể nhận được kết quả tương tự như Polygon khi vẽ chồng lên nhau nhiều Line. Chú ý: Polyline với một điểm sẽ không được hiển thị lên.

##### 4.2.6. Rectangle

- Một hình chữ nhật (Rectangle) được xác định bởi hai đặc tính rộng và cao. Để xác định vị trí của Rectangle trong container chứa chúng ta sử dụng thuộc tính Margin hoặc Canvas.Left, Canvas.Top để tùy chỉnh. Rectangle không hỗ trợ đối tượng con bên trong bởi vậy nếu muôn khu vực Rectangle chứa những đối tượng khác bạn có thể sử dụng Canvas. Trong trường hợp này thì nên sử dụng RectangleGeometry.

- Ví dụ:

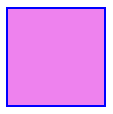
<Canvas Height="300" Width="400">

<Rectangle Width="100" Height="100"

Fill="Violet" Stroke="Blue"

StrokeThickness="2"/>

</Canvas>



#### 4.3. Geometries(Hình học)

Cung cấp các lớp cơ sở cho những đối tượng để xác định dạng hình học cho đối tượng. Đối tượng Geometry có thể sử dụng để làm vùng giao (clipping regions) và cũng như định nghĩa dữ liệu cho đối tượng Path.

##### 4.3.1. EllipseGeometry

- Miêu tả hình dạng của đường tròn hoặc ellipse. FillRule là đặc tính kế thừa từ lớp cơ sở Geometry nhưng tùy chọn này không có tác dụng đối với EllipGeometry.

- EllipGeometry chỉ đơn giản là định nghĩa hình dạng cho ellipse và không thể tự nó hiển thị được.

- Ví dụ: vẽ một path sử dụng dữ liệu dạng EllipseGeometry.

<Canvas Height="300" Width="400">

<Path Fill="Gold" Stroke="Orange" StrokeThickness="3">

<Path.Data>

<EllipseGeometry Center="100,100" RadiusX="50" RadiusY="50"/>

</Path.Data>

</Path>

</Canvas>



##### 4.3.2. PathGeometry

- Miêu tả dạng hình phức tạp có thể gồm hình cung, đường cong, ellipse, đoạn thẳng, hình chữ nhật. Mỗi PathGeometry xác định bởi một danh sách đối tượng pathFigure.

- Mỗi đối tượng PathFigure bao gồm một hoặc nhiều PathSegment.

- Để phủ kín diện tích của PathGeometry thì mọi đối tượng PathFigure bạn phải xét thuộc tính IsFilled là True và FillRule để xác định khu vực được fill.

- Ví dụ:

<StackPanel Height="150" Width="200">

<Path Stroke="Black" StrokeThickness="2">

<Path.Data>

<PathGeometry>

<PathFigure StartPoint="50,20">

<PathFigure.Segments>

<LineSegment Point="50,100" />

</PathFigure.Segments>

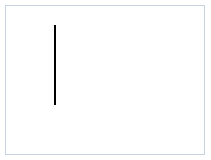
</PathFigure>

</PathGeometry>

</Path.Data>

</Path>

</StackPanel>



##### 4.3.3. GeometryGroup

- Miêu tả một hỗn họp hình dạng, được tạo thành từ nhiều đối tượng Geometry khác. Sử dụng FillRule cho GeometryGroup là hợp lý bởi sự kết hợp các hình dạng có thể có khả năng kết hợp các đoạn nơi mà có sự áp dụng của quy luật fill có tác dụng.

<StackPanel Height="300" Width="400">

<Path Stroke="Black" StrokeThickness="2" Fill="Violet" >

<Path.Data>

<GeometryGroup FillRule="EvenOdd">

<LineGeometry StartPoint="10, 10" EndPoint="150,150"/>

<EllipseGeometry Center="150,150" RadiusX="40" RadiusY="40"/>

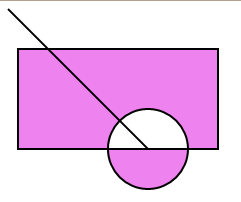
<RectangleGeometry Rect="20, 50 200 100"/>

</GeometryGroup>

</Path.Data>

</Path>

</StackPanel>



#### 4.4. Bushes

- Bạn có thể tô màu các đối tượng trong Silverlight với màu đơn, dải nghiêng, bán kính nghiêng và ảnh.

##### 4.4.1. Solid Color (Tô màu đồng nhất)

- Để tô màu một khu vực với một màu đơn sắc Silverlight cung cấp lớp SolidColorBrush. Những cách khác nhau để tô màu với SolicolorBrush:

* SolicolorBrush được định nghĩa trước theo tên
* Chọn từ bảng màu 32bit
* Sử dụng thuộc tính kiểu phần tử để mô tả SolicolorBrush.

##### 4.4.2. Gradient

- Gradient brush vẽ một khu vực với nhiều màu sắc pha trộn vào nhau dọc theo một trục.

Sử dụng chúng để tạo ra hiệu ứng ánh sáng và bóng, mang lại cho thành phần đồ họa của bạn cảm giác 3 chiều. Cũng có thể sử dụng chúng để mô phỏng kính, chrome, nước và các bề mặt mịn. Silverlight cung cấp 2 loại Gradient brush là: tô kiểu đổ màu theo tuyến tính (Linear Gradient), tô đổ màu dọc theo bán kính hình tròn (Radial Gradient)

* LinearGradientBrush
* RadialGradientBrush
  1. *LinearGradientBrush (Kỹ thuật đổ màu tuyến tính)*

Đối tượng LinearGradientBrush dùng để tô một vùng theo kỹ thuật tản màu dựa trên kỹ thuật nội suy các màu nằm giữa các cặp màu chỉ định dọc theo một trục (Gradientaxis) dạng đường thẳng. Hướng của đường thẳng chính là hướng đổ màu, ta phải chỉ định những màu cần xuất hiện tại các điểm nằm trên đường thẳng này nhờ đối tượng GradientStop. Ta có thể chỉ định hướng đổ màu nằm ngang từ trái qua phải, nằm dọc từ trên xuống, hay theo đường chéo bất kỳ. Mặc định thì hướng đổ màu sẽ theo đường chéo từ góc trên bên trái tới góc dưới bên phải.

* LinearGradientBrush dùng để vẽ một khu vực với một dải nghiêng.
* Thông qua đối tượng GradientStop để chỉ ra màu sắc cho dải nghiêng và vị trí dọc theo trục nghiêng.
* Thuộc tính StartPoint và EndPoint của LinearGradientBrush để xác định điểm đầu và điểm cuối của trục tô mỗi thuộc tính này được xác định bởi một cặp giá trị “x,y”. Trong đó, x và y là các số thực (double) có giá trị từ 0 đến 1, tương ứng với 0% và 100% tính tương đối so với đỉnh trên trái của hình chữ nhật cần vẽ (hay đỉnh của hình chữ nhật bao đối với các hình khác như Ellipse, Polygon,..).
* Thẻ <GradientStop Color="White" Offset="0.04"/> để chỉ định các điểm chốt nằm dọc theo trục tô. Thuộc tính Color xác định màu cần tô tại điểm chốt. Thuộc tính Offset nhằm xác định vị trí của điểm chốt, giá trị này là một số thực nằm trong khoảng từ 0 đến 1, giá trị càng gần 0 thì càng gần với điểm xuất phát StartPoint và, giá trị gần với 1 thì càng gần với điểm kết thúc EndPoint của trục tô. Hệ thống sẽ tự động nội suy các màu nằm giữa các cặp điểm chốt.
* Ví dụ1: sử dụng dải nghiêng với 4 màu và sử dụng nó để vẽ con cờ hình ellipse

<StackPanel Height="300" Width="400">

<Ellipse Width="30" Height="30" Stroke="Black" Opacity=".25" >

<Ellipse.Fill>

<LinearGradientBrush StartPoint=".3,.3" EndPoint="1,1">

<GradientStop Color="#FFFFFFFF" Offset="0.118644"/>

<GradientStop Color="#FFCACACA" Offset="0.402542"/>

<GradientStop Color="#FF000000" Offset="1"/>

<GradientStop Color="#FF656565" Offset="0.834746"/>

</LinearGradientBrush>

</Ellipse.Fill>

</Ellipse>

</StackPanel>



* Ví dụ 2: sử dụng dải nghiêng với 4 màu và sử dụng nó để vẽ hình chữ nhật

XAML:

<StackPanel Width="400" Height="300">

<Rectangle Width=" 300" Height="200">

<Rectangle.Fill>

<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">

<GradientStop Color="White" Offset="0.04"/>

<GradientStop Color="Blue" Offset="0.25"/>

<GradientStop Color="Red" Offset="0.65"/>

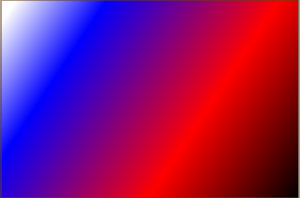
<GradientStop Color="Black" Offset="1.0"/>

</LinearGradientBrush>

</Rectangle.Fill>

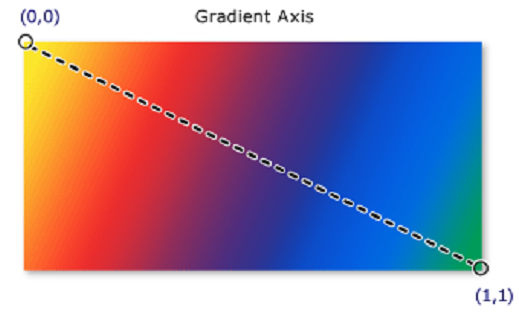
</Rectangle>

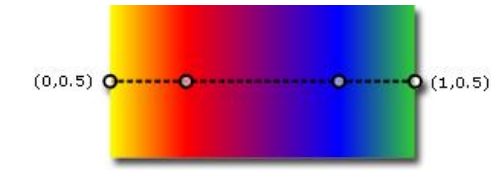
</StackPanel>



Vị trí màu khi sử dụng chổi LinearGradientBrush

Tọa độ các điểm StartPoint và EndPoint của trục tô ảnh hưởng đến hướng tô màu. Mặc định thì giá trị của StartPoint là (0,0) và EndPoint là (1,1), nghĩa là trục tô là đường chéo của hình chữ nhật bao.





Tọa độ màu của Offset khi sử dụng chổi LinearGradientBrush

* 1. *Radial Gradient (Tô màu theo bán kính hình tròn)*
* Kỹ thuật tô đổ màu theo bán kính hình tròn tương tự như kỹ thuật đổ màu tuyến tính, nhưng điểm xuất phát bắt đầu từ tâm đường tròn và màu được lan dần ra ngoài cho tới biên của đường tròn, sử dụng đối tượng tên là RadialGradientBrush.
* Cách vẽ của RadialGradientBrush dựa theo một trục được xác định bởi đường tròn; màu sắc của nó tia ra ngoài từ phía gốc của nó.
* Các thông số:
* Center- Tọa độ tâm đường tròn. Ví dụ: Center=0.5,0.5 thì nó tỏa đều sang mọi phía
* RadiusX- Bán kính ngang
* RadiusY- Bán kính dọc được nhập các giá trí số thực trong khoảng từ 0 đến 1 tương ứng với tỷ lệ khoảng cách tới đỉnh trên trái của hình chữ nhật.
* Thông số GradientStop là tọa độ của điểm xuất phát. Các điểm chốt sẽ chạy từ điểm GradientStop này, dọc theo đường kính hình tròn.
* GradientOrigin : Xác định tâm của điểm sáng (nếu GradientOrigin =0.5,0.5 thì tâm của điểm sáng nằm giữa hình)
* Ví dụ: vẽ một hình ellipse sử dụng RadialGradientBrush để tô màu bên trong

*Tạo 1 con cờ màu đen*

<Ellipse Width="30" Height="30" >

<Ellipse.Fill>

<RadialGradientBrush RadiusX="1.4" RadiusY="1.4" Center="0.3,0.3"

GradientOrigin="0.3,0.3">

<RadialGradientBrush.GradientStops>

<GradientStop Color="#FFFFFFFF" Offset="0"/>

<GradientStop Color="#FF616161" Offset="0.09"/>

<GradientStop Color="#FF1C1C1C" Offset="0.4"/>

<GradientStop Color="#FF000000" Offset="1"/>

</RadialGradientBrush.GradientStops>

</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>



*Tạo một con cờ màu trắng*

<StackPanel Height="300" Width="400">

<Ellipse Width="30" Height="30">

<Ellipse.Fill>

<RadialGradientBrush RadiusX="1.3" RadiusY="1.3" Center="0.3,0.3"

GradientOrigin="0.3,0.3">

<RadialGradientBrush.GradientStops>

<GradientStop Color="#FFFFFFFF" Offset="0.118644"/>

<GradientStop Color="#FFCACACA" Offset="0.402542"/>

<GradientStop Color="#FF000000" Offset="1"/>

<GradientStop Color="#FF656565" Offset="0.834746"/>

</RadialGradientBrush.GradientStops>

</RadialGradientBrush>

</Ellipse.Fill>

</Ellipse>

</StackPanel>



##### 4.4.3. Images

- Lớp Image cho phép bạn sử dụng ảnh để fill, để xét làm backgroud, và viền ngoài.

- Mặc định ImageBrush sẽ giãn ảnh ra hết toàn bộ khu vực được vẽ. Có thể thay đổi trạng thái này bằng việc thay đổi thuộc tính Stretch mặc định của nó tới một trong những giá trị sau: None, Uniform, UnìormToFill.

* Thuộc tính ImageSource của ImageBrush để chỉ định đường dẫn đến tệp hình ảnh.
* Thuộc tính Stretch để chỉ định các co dãn hình khi tô vùng, giá trị mặc định của Stretch là Fill. Thuộc tính này có các giá trị:
* None: Chổi tô không tự động co dãn hình.
* Uniform: Chổi tô co dãn hình trùng khít với một chiều của vùng tô nhưng giữ nguyên tỷ lệ của ảnh gốc.
* UniformToFill: Chổi tô co dãn hình phủ kín vùng tô nhưng giữ nguyên tỷ lệ của ảnh gốc.
* Fill: Chổi tô co dãn hình phủ kín vùng tô, không giữ tỷ lệ ảnh. Nếu tỷ lệ hai chiều của vùng tô khác với tỷ lệ hai chiều của ảnh thỉ ảnh tô sẽ bị méo.

- Ví dụ:

XAML:

<Canvas Width="400" Height="300" Canvas.Top="20">

<Rectangle Canvas.Left="180" Canvas.Top="70" Height="150" Width="150"

Stroke="Black" >

<Rectangle.Fill>

<ImageBrush ImageSource="anh.jpg" Stretch="UniformToFill"/>

</Rectangle.Fill>

</Rectangle>

</Canvas>



##### 4.4.4. Video

- VideoBrush cho phép bạn vẽ một khu vực bằng video.

- Ví dụ sau sử dụng VideoBrush để vẽ màu chữ ( Foreground) của TextBlock

<StackPanel Width="300" Height="300" Background="BurlyWood" Opacity=".1">

<MediaElement x:Name="ab"

Source="if we hold on together.mpg" IsMuted="True"

Opacity="0.0" IsHitTestVisible="False"/>

<TextBlock Canvas.Left="5" Canvas.Top="30"

FontFamily="Verdana" FontSize="70"

FontWeight="Bold"

TextWrapping="Wrap" Text="VIDEO">

<TextBlock.Foreground>

<VideoBrush SourceName="ab"/>

</TextBlock.Foreground>

</TextBlock>

</StackPanel>

### Chương 5: Animation và Media

#### 5.1. Animation

- Animation là một ảo giác được tạo ra bằng cách thay đổi một loạt các hình ảnh. Não nhận các hình ảnh như là một thay đổi cảnh.

- Một animation là tập hợp các frames, để chạy animation, các frames sẽ được show lần lượt nhau, giống như một đoạn video được stop + play. Animation chủ yếu dựa vào việc thay đổi các thuộc tính (dependency property) theo 1 giá trị thời gian. Chẳng hạn, làm button lớn dần, ta có thể thay đổi thuộc tính Width của nó trong animation.

- Animations thực thi dựa trên việc ***thay đổi các thuộc tính***.

- Mỗi kiểu dữ liệu yêu cầu 1 class animation khác nhau. Chẳng hạn, thuộc tính Width của button dùng kiểu double data, dùng class DoubleAnimation, muốn thay đổi màu sắc của 1 Canvas có thể dùng class ColorAnimation.

- Có 2 kiểu Animation:

* **Linear interpolation**: Giá trị các thuộc tính biến đổi mềm mại, liên tục trong suốt khoảng thời gian Duration được thiết lập. Gồm có 3 class:
* Double Animation (thường được sử dụng)
* PointAnimation
* ColorAnimation.
* **Key-frame animation**: Giá trị các thuộc tính biến đổi theo từng frame, (nhanh chậm phụ thuộc thời gian Duration được thiết lập cho từng frame. Gồm 4 class:
* ColorAnimationUsingKeyFrames
* DoubleAnimationUsingPath
* PointAnimationUsingKeyFrames
* ObjectAnimationUsingKeyFrames.

##### 5.1.1. Storyboard

- Kiểm soát các animation với một timeline và cung cấp các đối tượng và tài nguyên nhắm mục đích cung cấp thông tin cho các animation con.

- Lớp Storyboard cung cấp cho ta 2 thuộc tính cơ bản là:

Storyboard.TargetProperty và Storyboard.TargetName.

Bạn cần phải thiết đặt các thuộc tính của Animation để xác định đối tượng được sở hữu.

- Lớp storyboard được sử dụng để quản lý timeline dành cho animation của bạn. Storyboard xác định giá trị thuộc tính nào áp dụng cho element nào thông qua 2 thuộc tính: TargetProperty và TargetName ( Có thể nhóm các animation áp dụng cho cùng 1 đối tượng thay đổi các thuộc tính khác nhau, điều khiển để lặp, dừng, thay đổi vị trí …các animation.). Các giá trị From, To, Duration, By…nhằm thiết lập giá trị đầu và cuối của thuộc tính trong khoảng thời gian duration…, thuộc tính RepeatBehavior (2x,3x/0:0:13/Forever) giúp bạn play lại các animation theo ý muốn.

- Ví dụ:

XAML:

<Grid x:Name="LayoutRoot" Background="White">

<Canvas>

<Canvas.Resources>

<Storyboard x:Name="Storyboard">

<PointAnimation Storyboard.TargetProperty="center"

Storyboard.TargetName="ellipse"

Duration="0:0:5"

From="20,200"

RepeatBehavior="forever"/>

</Storyboard>

</Canvas.Resources>

<Path Fill="Black">

<Path.Data>

<EllipseGeometry x:Name="ellipse" Center="20,20" RadiusX="15" RadiusY="15"/>

</Path.Data>

</Path>

<StackPanel Canvas.Left="10" Canvas.Top="265" Orientation="Horizontal">

<Button Click="Button\_Click" Width="65" Height="30" Margin="2" Content="begin"/>

<Button x:Name="Pause" Width="65" Height="30" Margin="2" Click="Pause\_Click" Content="Pause"/>

<Button x:Name="Resume" Width="65" Height="30" Margin="2" Click="Resume\_Click" Content="Resume"/>

<Button x:Name="Stop" Width="65" Height="30" Margin="2" Click="Stop\_Click" Content="Stop"/>

</StackPanel>

</Canvas>

</Grid>

</UserControl>

C#:

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

{

Storyboard.Begin();

}

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

{

Storyboard.Pause();

}

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

{

Storyboard.Resume();

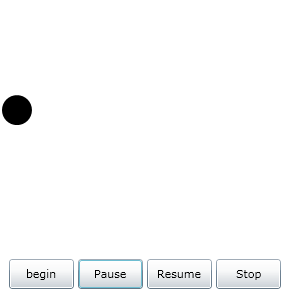
}

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

{

Storyboard.Stop();

}



##### 5.1.2. Key-Frame Animations

- Key- Frame animation cho phép bạn tương tác vào animation nhiều hơn hai đối tượng và kiểm soát một animation bằng phương pháp nội suy.

- Key- frame không có các thuộc tính như From/To. Bạn tạo ra key- frame, các đối tượng animation đưuọc key-frame kiểm soát. Khi hình ảnh động chạy nó sẽ chuyển giữa các frame mà bạn xác định tạo ra một chuỗi chuyển động liền mạch.

##### 5.1.3. Double Animation

- Double animation biến đổi giá trị giữa hai biến của đối tượng Animation bằng cách sử dụng nội suy tuyến tính trên một khoảng thời gian xác định.

- Ví dụ: Khi đánh cờ thắng sẽ thay đổi màu sắc của 5 con cờ. Có 2 cách thay đổi màu của con cờ là: sử dụng thuộc tính EasingColorKeyFrame hoặc SplineDoubleKeyFrame.

XAML:

<UserControl.Resources>

<Storyboard x:Name="Highlight">

<ColorAnimationUsingKeyFrames

BeginTime="00:00:00"

Storyboard.TargetName="Shape"

Storyboard.TargetProperty="(Shape.Stroke).(SolidColorBrush.Color)">

<SplineColorKeyFrame

KeyTime="00:00:01" Value="#FFFF0000"

KeySpline="0.704999983310699,0,1,1"/>

<SplineColorKeyFrame

KeyTime="00:00:02" Value="#FF000000"

KeySpline="0,0,1,1"/>

</ColorAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames

BeginTime="00:00:00"

Storyboard.TargetName="Shape"

Storyboard.TargetProperty="(Shape.StrokeThickness)">

<SplineDoubleKeyFrame

KeyTime="00:00:01" Value="1"

KeySpline="0.704999983310699,0,1,1"/>

<SplineDoubleKeyFrame

KeyTime="00:00:02" Value="1"

KeySpline="0,0,1,1"/>

</DoubleAnimationUsingKeyFrames>

<ColorAnimationUsingKeyFrames

BeginTime="00:00:00"

Storyboard.TargetName="Shape"

Storyboard.TargetProperty="(Shape.Fill).(GradientBrush.GradientStops)[2].(GradientStop.Color)">

<SplineColorKeyFrame

KeyTime="00:00:01" Value="#FF600101"

KeySpline="0.704999983310699,0,1,1"/>

<SplineColorKeyFrame

KeyTime="00:00:02" Value="#FF1C1C1C" KeySpline="0,0,1,1"/>

</ColorAnimationUsingKeyFrames>

<ColorAnimationUsingKeyFrames

BeginTime="00:00:00"

Storyboard.TargetName="Shape"

Storyboard.TargetProperty="(Shape.Fill).(GradientBrush.GradientStops)[1].(GradientStop.Color)">

<SplineColorKeyFrame

KeyTime="00:00:01"

Value="#FFFF0000"

KeySpline="0.704999983310699,0,1,1"/>

<SplineColorKeyFrame

KeyTime="00:00:02" Value="#FF616161"

KeySpline="0,0,1,1"/>

</ColorAnimationUsingKeyFrames>

</Storyboard>

</UserControl.Resources>

##### 5.1.4. Color Animation

- Color Animation biến đổi màu giữa hai biến của đối tượng Animation bằng cách sử dụng một đối tượng nội suy tuyến tính trên một khoảng thời gian xác định.

##### 5.1.5. Point Animation

- Point Animation biến đổi giá trị tọa độ của một điểm giữa hai biến của đối tượng Animation bằng cách sử dụng nội suy tuyến tính trên một khoảng thời gian xác định.

#### 5.2. Media

##### 5.2.1. MediaElement Object

- Để thêm media( Video, Audio) vào trang web cảu bạn ta sử dụng MediaElement. MediaElemnt sẽ cung cấp cho bạn một Fỏm để hiển thị Media, bạn chỉ cần cung cấp nguồn tài nguyên media cho URL, như là một file videp chẳng hạn.

##### 5.2.2. Controlling Media Playback Interactively

- Bạn có thể tương tác điều khiển media bằng cách sử dụng Play, Pause, Stop và các thuộc tính của một đối tượng MediaElement.

### Chương 6: Expression Blend

#### 6.1. Ví dụ tạo hiệu ứng di chuyển giữa các ảnh

Thiết kế bằng Expression Blend tạo hiệu ứng di chuyển giữa các ảnh

XAML:

<UserControl.Resources>

<Storyboard x:Name="thanh">

<DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(UIElement.Opacity)" Storyboard.TargetName="anh1">

<EasingDoubleKeyFrame KeyTime="0:0:1" Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:2" Value="0"/>

<EasingDoubleKeyFrame KeyTime="0:0:4" Value="1"/>

</DoubleAnimationUsingKeyFrames>

<DoubleAnimationUsingKeyFrames Storyboard.TargetProperty="(UIElement.Opacity)" Storyboard.TargetName="anh2">

<EasingDoubleKeyFrame KeyTime="0:0:2" Value="1"/>

<EasingDoubleKeyFrame KeyTime="0:0:4" Value="0"/>

</DoubleAnimationUsingKeyFrames>

</Storyboard>

</UserControl.Resources>

<Grid x:Name="LayoutRoot" Background="White">

<Image x:Name="anh1" Source="anh1.jpg"/>

<Image x:Name="anh2" Source="anh2.jpg"/>

</Grid>

</UserControl>



Ảnh 1



Thời khắc chuyển giao giữa 2 ảnh



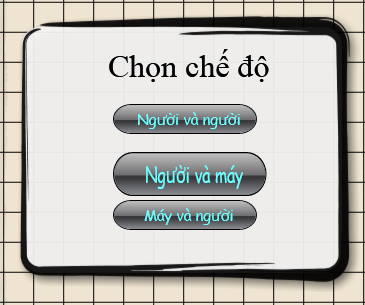
Ảnh 2

#### 6.2. Hiệu ứng các Button trong chương trình



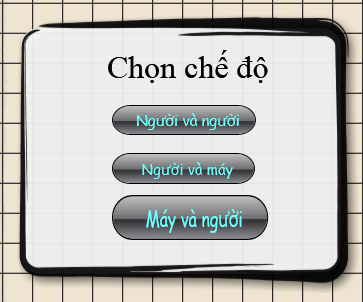
1

1: Khi người chơi di chuột vào nút “Người và Người”



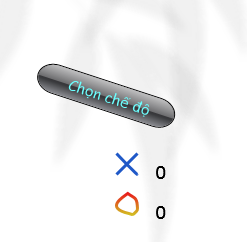
2

2: Khi người chơi di chuột vào button “Người và Máy”



3

3: Khi người chơi di chuột vào button “Máy và Người”

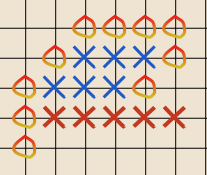
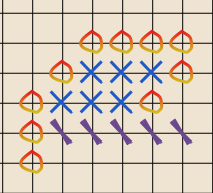
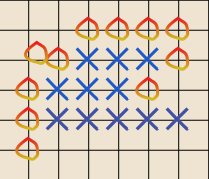
5

4

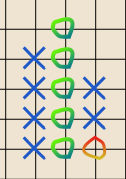
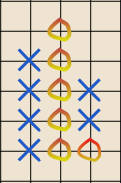
4,5: Trước và sau khi di chuột vào button “Chọn chế độ”

#### 6.3. Hiệu ứng các con cờ khi chiến thắng

* Khi con x thắng thì nó sẽ nháy và chuyển màu từ xanh sang màu đỏ



Khi con cờ O thắng thì nó sẽ chuyển từ màu đỏ sang màu xanh và quay vòng tròn



## II. Lập trình game cờ caro

### Chương I: Giới thiệu trò chơi caro

#### Giới thiệu sơ lược về trò chơi caro

Cờ caro là một loại trò chơi khá phổ biến vì luật chơi đơn giản và không đòi hỏi nhiều về công cụ chơi, bạn chỉ cần một tờ giấy có kẻ ô và bút là đủ.Chính vì vậy trò chơi này lại càng phổ biến trong giới học sinh sinh viên.

Cờ caro là trò chơi đối kháng 2 người. Gồm 2 người chơi, đối thủ này sẽ tìm cách dành chiến thắng trước đối thủ kia trong một số hữu hạn nước đi, mỗi nước đi đuợc tạo ra dựa từ 1 trạng thái bất kỳ của trận đấu. Nếu sau 1 số giới hạn nước đi, nếu chưa ai dành chiến thắng thì xem như hoà. Ngoài ra, thông tin về trận đấu là hoàn toàn biết đuợc đối với cả 2 đối thủ.

Trò chơi có luật chơi hết sức đơn giản như sau: mỗi đối thủ trong mỗi lượt đi của mình sẽ chọn 1 ô trống còn lại trên bàn cờ (kẻ sẵn các ô lưới ) sao cho tạo thành n con liên tiếp để chiến thắng ... Nếu n = 3 thì nó có 1 tên khác là Tic Tac Toe, nhưng ở đây trò chơi kết thúc khi có một người đạt được 5 con liền nhau cùng  trên một  hàng ngang, hàng dọc hoặc đường chéo. Nếu khi đã chơi kín bàn cờ nhưng không có ai đạt được 5 con liền nhau thì trò chơi kết thúc và hoà nhau.

Trò chơi cờ caro có nguồn gốc từ rất lâu, và là trò chơi mang tính quốc tế. Ở các nước khác nhau trò chơi này mang những cái tên khác nhau nhưng đều gắn với luật chơi đơn giản của nó. Ví dụ như: ở Nhật trò chơi có tên Gomoku, Gobang (Go trong tiếng Nhật có nghĩa là 5, Gomoku có nghĩa là 5 điểm), ở Anh trò chơi có tên là Connect 5.

#### Giới thiệu sơ lược về đề tài

Xây dưng một chương trình phục vụ chơi cờ caro với các chức năng cơ bản như sau:

* Chức năng Single Player: Chơi với máy tính, có phân các mức độ dễ và khó cho người chơi.
* Chức năng Multi Player: Chơi đa người dùng, chơi online: Chương trình hỗ trợ cho 2 người ở trên cùng mạng LAN để có thể chơi

### Chương II: Giải thuật của chương trình

Đây là một chương trình trò chơi mang tính trí tuệ nên phần giải thuật của chương trình là phần quan trọng nhất. Đánh giá một chương trình trò chơi cờ hay hay không là thông qua giải thuật của chương trình.Trong phần này, ta sẽ trình bày chi tiết về giải thuật của chương trình chơi cờ caro.

Giải thuật có 2 phần chính:

* Giải thuật xác định thắng thua cho một ván cờ, giải thuật này áp dụng cho cả chơi 2 người và chơi với máy nên có thể dùng chung.
* Giải thuật tìm nước đi của máy trong khi chơi với máy.

#### 2.1. Giải thuật xác định thắng thua cho một ván cờ

Giải thuật được dựa trên chương trình caro cổ điển với luật chơi là xác định một người thắng khi người đó có 5 con liên tiếp nằm trên cùng một hàng ngang, hàng dọc hoặc đường chéo.

#### 2.2. Giải thuật tìm nước đi cho máy tính

##### *2.2.1. Sơ lược về giải thuật*

Trong trò chơi Caro, cứ sau mỗi nước cờ, mỗi đối thủ sẽ chọn ra từ những ô trống để đi. Do đó, sau 1 mỗi nước đi thì số ô trống còn lại sẽ giảm. Như vậy, việc tìm nước đi tiếp theo cho trạng thái có sẵn chỉ là việc tìm kiếm những ô trống còn lại. Đồng thời, không gian tìm kiếm sẽ thu hẹp theo số nước đi đã tạo. Như vậy, để chọn 1 nước đi kế tiếp từ 1 trạng thái bàn cờ có sẵn, ta phải tìm kiếm nước đi. Không gian chọn nước đi từ mỗi trạng thái ban đầu là hữu hạn, không gian tìm kiếm 1 nước đi dẫn đến chiến thắng cũng là hữu hạn. Nhưng rõ ràng số lượng phần tử của hai không gian này đuợc so sánh giống như hạt cát và sa mạc. Do đó ta không thể vét sạch không gian tìm kiếm nước đi mà ta phải giới hạn không gian tìm kiếm. Một không gian tìm kiếm có thể hiện thực theo dạng 1 cái cây đa phân bình thường như trong Cấu trúc dữ liệu đã định nghĩa, lúc này nó đuợc gọi là cây tìm kiếm, cây trò chơi, mỗi nút cùng mức của cây này thể hiện một lựa chọn các nước đi có sẵn. Nếu số nút ở mỗi mức càng nhiều, tức là có nhiều khả năng chọn lựa 1 nước đi từ 1 trạng thái trước, do đó độ phân nhánhcủa cây này càng lớn. Dựa vào cây trò chơi đã định nghĩa ở trên, việc tìm kiếm nước đi là chọn 1 nút trên cây ( ở mức 1) sao cho nước đó là tốt, do mình đánh giá, theo thông thường khi chơi, một nước đi tốt hay không là phụ thuộc vào khả năng dành chiến thắng là cao hay thấp sau khi nước đi này được đi. Do đó, muốn chọn 1 nước đi tốt thì nếu chỉ dựa vào thế cờ hiện tại là chưa đủ, mà phải biết thông tin của những thế cờ sau khi chọn nước này để đi ... Ví dụ như khi chơi trò Carô, ta chọn một nước đi vào 1 ô nào đó để chận đuờng 3 hở hai đầu của đối thủ vì ta biết là nếu không đi nuớc này thì sẽ thua ở 2 nửa nước đi tiếp theo, tức là trạng thái thua còn chưa biết đuợc nếu ngay sau khi chọn đi 1 ô khác để đi xuất phát trạng thái này. Khái niệm độ sâu cung nảy sinh từ đây, đơn giản thì độ sâu là khả năng "nhìn thấy trước" 1 nước đi tốt sau một loạt nước đi xuất phát từ hiện tại , ví dụ như nếu từ trạng thái này, ta nhận biết đuợc là sau 6 con nữa là mình sẽ thắng (tức là mỗi bên đi 3 con), khi đó độ sâu tính toán của ta là >= **6**, Như vậy, Max depth thể hiện khả năng và trình độ của người chơi cờ, ta chơi càng hay thì giá trị này càng lớn .

Khi viết chương trình cho máy tính chơi cờ, tức là máy tính phải tự tìm nước đi khi mình đưa vào 1 trạng thái bàn cờ bất kì, do không gian tìm kiếm là quá lớn (coi như là vô hạn) nên mình chỉ giới hạn cho máy tính chỉ tìm kiếm đến 1 độ sâu nào đó mà thôi , đó là độ sâu tìm kiếm lớn nhất ... thể hiện khả năng của chương trình, chúng ta sẽ cố gắng nâng cao giá trị này bằng cách cài đặt thêm các tri thức cờ cho nó (Heuristic, Knowledge ...).

##### *2.2.2. Thuật toán tìm kiếm trong trò chơi*

Một chiến lược thường được cả người lẫn máy dùng là phân tích thế cờ chỉ sau một số nước đi nào đó của cả hai bên. Sau khi "nhìn xa" xem bàn cờ có những khả năng biến đổi như thế nào sau một số nước, ta sẽ đánh giá độ xấu tốt của các thế cờ nhận được. Tiếp theo, ta sẽ chọn nước đi sẽ dẫn tới một thế cờ tốt nhất trong số đó có cân nhắc đến cách đi của cả hai bên. Với máy, thế cờ này được đánh giá là tốt hơn thế cờ kia nhờ so sánh điểm của thế đó do bộ lượng giá trả lại. Chúng ta chỉ có khả năng xét trước một số hữu hạn các nước (ví dụ đại kiện tướng chơi cờ vua có thể xét trước 8-10 nước đi, người thường chỉ 2-4 nước đi). Rõ ràng là nếu xét càng sâu thì chơi càng giỏi. Nhưng không thể thực hiện điều này với độ sâu quá lớn được do số nút ở độ sâu đó có thể trở nên lớn khủng khiếp và không đủ thời gian để phân tích. Nếu dừng ở một độ sâu hợp lý thì bộ phân tích có thể hoàn thành việc tính toán trong một thời gian hạn định.

###### **Thủ tục minimax**

Giả sử chúng ta có một bộ phân tích thế cờ có thể áp dụng tất cả các luật, các phương pháp đánh cờ khác nhau vào từng thế cờ và chuyển đổi chúng thành một con số đại diện (cho điểm thế cờ). Mặt khác, ta giả sử con số đó là dương khi áp dụng cho thế cờ của một đấu thủ (được gọi là người chơi cực đại - maximizer), và là âm khi áp dụng cho đấu thủ bên kia (được gọi là người chơi cực tiểu - minimizer). Quá trình tính toán cho điểm thế cờ được gọi là lượng giá tĩnh (static evaluation). Hàm thực hiện việc tính toán được gọi là một bộ lượng giá tĩnh, và giá trị nhận được gọi là điểm lượng giá tĩnh. Cả hai đấu thủ đều cố gắng đi như thế nào đó để đạt được điểm tuyệt đối lớn nhất. Người chơi cực đại sẽ tìm những nước đi dẫn đến điểm của mình trở nên lớn hơn (hay cao nhất có thể được) hay điểm của đối thủ bớt âm hơn (nhỏ hơn về giá trị tuyệt đối). Còn đấu thủ của anh ta, người chơi cực tiểu, lại ra sức phản kháng lại, để dẫn tới điểm âm của anh ta âm hơn hay điểm dương của đối thủ nhỏ đi

Nói chung giải thuật minimax dành cho các game đối kháng(2 người chơi) thì bao gồm 3 phần:

* Hàm phát sinh nước đi
* Hàm đánh giá
* Hàm tìm kiếm
* ***Thuật toán Minimax***- Nếu như đạt đến giới hạn tìm kiếm (đến tầng dưới cùng của cây tìm kiếm), tính giá trị tĩnh của thế cờ hiện tại ứng với người chơi ở đó. Ghi nhớ kết quả   
  - Nếu như mức đang xét là của người chơi cực tiểu, áp dụng thủ tục Minimax này cho các con của nó. Ghi nhớ kết quả nhỏ nhất   
  - Nếu như mức đang xét là của người chơi cực đại, áp dụng thủ tục Minimax này cho các con của nó. - Ghi nhớ kết quả lớn nhất.
* ***Viết chương trình cho thuật toán Minimax***

Bây giờ, ta thử dựa vào phát biểu trên để viết chương trình cho thuật toán này bằng ngôn ngữ tựa Pascal. Đây là một hàm có tên là Minimax và sẽ là loại đệ qui. Trước hết, để hàm này biết đã đạt đến giới hạn tìm kiếm chưa, ta cần cung cấp cho nó một tham số về độ sâu tìm kiếm depth (để biết phải tìm đến đâu), đồng thời ta cũng phải cho biết thế cờ hiện tại pos để nó từ đó nó biết cách tính tiếp. Giá trị trả về của hàm chính là điểm của thế cờ (bàn cờ) pos.

Mỗi khi Minimax được gọi, nó sẽ càng gần đến giới hạn tìm kiếm, do đó ta sẽ gọi hàm này với độ sâu bằng độ sâu cũ trừ đi một. Đạt đến độ sâu giới hạn chính là khi depth = 0. Khi đạt độ sâu này ta sẽ gọi hàm lượng giá Eval để đánh giá chất lượng của thế cờ pos hiện tại (thực hiện điều một của thuật toán).

Ở trên, Minimax được gọi với độ sâu giảm đi một. Đó là độ sâu của các thế cờ là con. Các thế cờ con pos' đó là các thế cờ được tạo ra từ pos bằng cách đi một nước đi hợp lệ m nào đó. Do đó ta phải có các lệnh thực hiện đi quân để đến các thế cờ mới. Để biết từ thế cờ pos có thể đi được những nước nào, ta dùng một thủ tục Gen có tham số là thế cờ cha pos. Thủ tục này sẽ cất các thế cờ con pos' đó vào bộ nhớ (dạng danh sách). Việc tiếp theo là ta lấy từng thế cờ đó ra và áp dụng tiếp thủ tục Minimax cho nó để tính điểm value của nó.

Theo phát biểu của thuật toán, ta thấy các điều 2 và 3 chỉ khác nhau ở cách chọn kết quả tốt nhất best phụ thuộc vào người chơi đang là người chơi cực đại hay cực tiểu. Cuối cùng thuật toán sẽ trả về điểm tốt nhất đạt được.

Thông thường để cho tiện (và cũng rất gần sự thực) ta coi cả hai người chơi (hai bên) có cùng cách đánh giá về một thế cờ. Có điều thế cờ này là tốt với một người thì phải được đánh giá là tồi với người kia và ngược lại. Trong máy tính cách thể hiện tốt nhất là ta cho điểm một thế cờ có thêm dấu âm dương: dấu âm dành cho người chơi cực đại và dấu âm cho người chơi cực tiểu. Với người chơi cực đại sẽ mong muốn điểm này càng dương càng tốt, còn người chơi cực tiểu lại mong muốn điểm này càng âm càng tốt. Do đó để dễ xử lí ta sẽ tuỳ theo mức người chơi mà đổi dấu giá trị đánh giá thế cờ pos. Chú ý rằng, thay đổi độ sâu là chuyển sang đối phương nên phải đổi dấu.

Cũng do dùng cùng hàm lượng giá nên khi đến lượt người chơi cực đại và cực tiểu có cùng cái nhìn như nhau về một thế cờ. Điều này dẫn đến có thể dùng cùng cách chọn nước đi tốt nhất cho họ (gộp được điều 2 và 3 lại với nhau được). Giá trị best cần được khởi đầu rất nhỏ để đảm bảo không vượt mọi giá trị value, tốt nhất là giá trị -vô cùng:

Thông thường, bàn cờ được biểu diễn bằng các biến toàn cục. Do đó thay cho truyền tham số là một bàn cờ mới pos vào thủ thục Minimax thì người ta biến đổi luôn biến toàn cục này nhờ thực hiện nước đi "thử" (nước đi dẫn đến bàn cờ mới pos). Sau khi Minimax thực hiện việc tính toán dựa vào bàn cờ lưu ở biến toàn cục thì thuật toán sẽ dùng một số thủ tục để loại bỏ nước đi này. Như vậy Minimax bỏ các tham số pos như sau:

* **Đánh giá thuật toán Minimax**

Nếu hệ số nhánh trung bình của cây là b và ta thực hiện tìm kiếm đến độ sâu d thì số nút phải lượng giá ở đáy cây như ta đã biết là bd. Đây chính là số đo độ phức tạp của thuật toán. Nếu b = 40, d = 4 (các con số thường gặp trong trò chơi cờ) thì số nút phải lượng giá là 40^4 = 2560000 (trên 2 triệu rưỡi nút). Còn với b = 40, d = 5 thì số nút phải lượng giá sẽ tăng 40 lần nữa thành 40^5 = 102400000 (trên 102 triệu nút).  
Lưu ý: toàn bộ ý tưởng của thuật toán này là dựa trên việc chuyển đổi mỗi thế cờ thành một con số để đánh giá. Rất tiếc là các con số này thường không tốt và không đủ để đánh giá hết mọi điều. Mặt khác, thuật toán này có thể rất tốn kém (chạy chậm) do việc sinh các nước đi và lượng giá rất tốn thời gian tính toán, do vậy độ sâu của cây trò chơi cũng bị hạn chế nhiều. Việc tỉa nhánh của thuật toán AlphaBeta sẽ giúp để cải thiện tình hình.

###### **Thuật toán AlphaBeta**

Thuật toán AlphaBeta nói chung giúp chúng ta tiết kiệm nhiều thời gian so với Minimax mà vẫn đảm bảo kết quả tìm kiếm chính xác. Tuy nhiên lượng tiết kiệm này không ổn định - phụ thuộc vào số nút mà nó cắt bỏ. Trong trường hợp xấu nhất thuật toán không cắt được một nhánh nào và phải xét số nút đúng bằng Minimax. Ta cần đẩy mạnh việc cắt bỏ nhờ đẩy nhanh sự thu hẹp của cửa sổ tìm kiếm alpha - beta. Cửa sổ này được thu hẹp một bước khi gặp một giá trị mới tốt hơn giá trị cũ. Khi gặp giá trị tốt nhất thì cửa sổ này thu hẹp nhất. Do đó nếu càng sớm gặp giá trị tốt nhất thì cửa sổ càng chóng thu hẹp. Như vậy phải làm sao cho các nút ở lá được sắp xếp theo trật tự từ cao xuống thấp. Trật tự này càng tốt bao nhiêu thì thuật toán chạy càng nhanh bấy nhiêu (các công thức về số nút phải lượng giá trong điều kiện lí tưởng ở trên tính được với trật tự là tốt nhất).

#### 2.3. Kỹ thuật sinh nước đi

- Sinh nước đi (generate moves) đây là 1 giai đoạn trong quá trình tìm kiếm nước đi, tức là từ trạng thái hiện tại, mình muốn tìm kiếm một nước đi hợp lệ cho lượt của người chơi hiện tại ==> khi đó phải "sinh" tất cả các nước đi có thể có, sau đó ta sẽ chọn một nước đi tốt nhất (phải đánh giá nó ...) trong số những nước đó.

- Thông tin cho quá trình tìm kiếm nước đi chỉ đơn giản chỉ là cái bảng với những ô trống hay ô đã bị chiếm (của cả 2 bên).  Nếu không có sẵn thông tin những nước đi hợp lệ thì việc tìm kiếm đơn giản chỉ là chọn 1 ô trống rồi đánh 1 con cờ , như vậy việc làm này sẽ dẫn đến việc tìm kiếm vô cùng phức tạp vì ta sẽ không biết 1 ô trống đã tìm qua hay chưa do đó có thể rơi vào 1 vòng lặp tìm kiếm vô hạn . Do vậy ta phải sinh nước đi để có thể duyệt chính xác và đầy đủ.

**Phân tích các kỹ thuật:**

##### ***2.3.1 Full generation ( sinh đầy đủ các nước đi)***

- Ý tưởng : Từ trạng thái hiện tại, kiểm tra xem còn bao nhiêu nước đi hợp lệ có thể có thì cất tất cả vào stack ...

- Phân tích :

+ Mã giả như sau :

for (i = 0; i < BOARD\_SIZE;i++)

if (IsLegalMove(i)) // kiểm tra xem (Board[i] == EMPTY)

if (IsNotInGenMove(i)) // kiểm tra xem đã có trong danh sách chưa...

StoreGenMove(i) // cất vào stack ...

##### ***2.3.2. Increasing generation ( Sinh dần các nước đi)***

- Ý tưởng : Từ trạng thái ban đầu, sinh 1 vài nước đi có triển vọng đưa đến chiến thắng), kiểm tra xem những nước đi này sau một vài độ sâu kế tiếp xem có thật sự là "cúm" chưa? Nếu chưa thì tiếp tục sinh những nước đi khác rồi kiểm tra tiếp ...

+ Kỹ thuật này khá hay, nhất là khi mình muốn tính sâu vào những nước "tốt" (theo quan điểm của mình) để mong chờ search-algorithm cắt nhánh sớm trước khi mình tiếp tục sinh thêm những nước lằng nhằng khác ... Tuy nhiên, nếu ở 1 trạng thái nào đó, nước đi tốt không nằm ở trong số những nước đi có triển vọng thì mình vẫn phải tiếp tục sinh ... cách cài đặt sẽ trở nên phức tạp. Rõ ràng cách này chỉ nên sử dụng để đối phó trong trường hợp bị giới hạn thời gian, tức là mình sẽ lựa 1 nước đi tốt nhất trong số đó "để dành", nếu thời gian không đủ để generate và search tiếp thì mình sẽ đem nó ra để đánh. Một nhược điểm khác đó là chưa chắc tìm thấy một nước đi tốt nhất như kỹ thuật đầu tiên (vì nó đã cắt nhánh ở 1 nước đi mà tưởng rằng là tốt nhất, hiện tượng này cũng có thể gọi là "cực đại cục bộ") ...

##### ***2.3.3. Decrescent generation (Sinh nước đi giảm dần)***

- Ý tưởng : Cách này đọc qua tưởng chừng như ngược lại với cách trên nhưng thực sự là có sự khác biệt rất lớn ! Nói chung là chọn một thang đo ban đầu để đánh giá tĩnh sự lợi hại của các nước đi hiện tại cần sinh ... Sau đó lần lượt giảm thang đo và sinh nước đi cho đến khi nào chứa ít nhất 1 nước đi trong stack thì break... nếu chưa thì sẽ tiếp tục sinh tiếp

+ Rõ ràng cách này luôn tìm ra nước đi để đưa vào move-stack từ đó thể hiện sự đúng đắn của nó ... Kỹ thuật này khá phổ biến trong các chương trình cờ bởi vì nó sẽ lần lượt chọn lựa những nước đi từ tốt đến xấu nhất dựa trên thang đo là giá trị của mark và nó bảo đảm sẽ sinh và chọn ra những nước tốt nhất có thể từ trạng thái hiện tại để đưa vào stack. Nói là vậy nhưng rõ ràng điều này có hiện thực được hay không là còn nhờ vào hàm ValueIt() nữa! Nếu hàm này làm nhanh và chính xác thì sẽ đem hiệu quả cao ... cách thiết kế hàm này cũng tương tự như hàm IsHasOutlook() ở trên nhưng cần rộng và chính xác hơn tức là phải tính thêm những nước được xem là "outlook" đối với đối thủ nữa, bởi vì những nước đó có thể thật sự có giá trị (do nếu ta đánh vào nước đó sẽ chiếm vị trí "chiến lược" của đối phương, tức là chặn trước những "đường mở" của đối thủ ...và đôi khi việc làm này là đáng giá hơn việc đánh 1 đường 5 (thuộc loại có triển vọng của ta) mà đối thủ dễ dàng chặn phá !!). Kỹ thuật này thật sự rất hay nếu nó khắc phục những nhược điểm sau, đó là việc thiết kế hàm ValueIt() thật sự là rất khó, bởi vì mình không thể lập trình chính xác cho máy biết được nước nào là có giá trị, đây chỉ là gần đúng mà thôi... với lại khi thiết kế xong rồi mà đem vào hàm search nước đi của mình thì sinh ra được 1 nước đi sẽ phải mất nhiều  micrô giây

##### ***2.3.4 Complete Generation (Sinh tất cả các nước đi)***

- Ý tưởng : Sử dụng stack để lưu tất cả nước đi của trạng thái *bắt đầu tìm kiếm*, sau đó việc tìm kiếm nước đi sẽ không cần sinh thêm lần nào nữa. Tức là chỉ cần sinh 1 lần và cất ngay vào move-stack, sau này khi lấy ra sẽ kiểm tra xem nó có còn là 1 legal move hay không (tức là ô này đã bị đánh chưa ...) rồi phân tích nó

- Phân tích :

+ Mã giả cho chương trình tương tự như kỹ thuật Full generation

+ Cách này sẽ không tốn thời gian sinh nước đi trong quá trình search, không tốn nhiều bộ nhớ để lưu các biến nước đi theo từng độ sâu nhưng tốn nhiều thời gian cho việc phân tích và chọn lựa nước đi. Có những đặc điểm trên thì dường như nó tiến bộ hơn Full generation nhiều ... tuy nhiên đó là trong trường hợp không kèm thêm việc lượng giá nước đi cùng với việc sinh nước đi ... bởi vì việc tìm kiếm chọn lọc nước đi là phải dựa trên chỉ số đánh giá nào, chỉ số đó được tính ra sao và khi nào, ở đâu ... Nói chung đây cũng là kỹ thuật chỉ ứng dụng cho từng trường hợp và từng trò chơi cụ thể.

##### *2.3.5 Selective generation (Sinh nước đi có chọn lọc)*

- Ý tưởng : Dùng một hàm chọn lựa để đánh giá những nước đi nào là đạt những tiêu chuẩn đặt ra thì mới đưa vào stack, rõ ràng hàm chọn lựa này là rất rộng, ví dụ : chỉ sinh những ô trong vòng bán kính 1,2 ô so với những ô đã đánh, chỉ sinh những ô mà có thể gây nên những đường đe doạ, chỉ sinh những ô nằm trong khu vực "nóng", chỉ sinh những ô mà có thể chận được đường hiện tại ... nói chung là rất nhiều tiêu chuẩn ... kỹ thuật này thì tổng quát hơn những kỹ thuật sinh có chọn lọc ở trên.

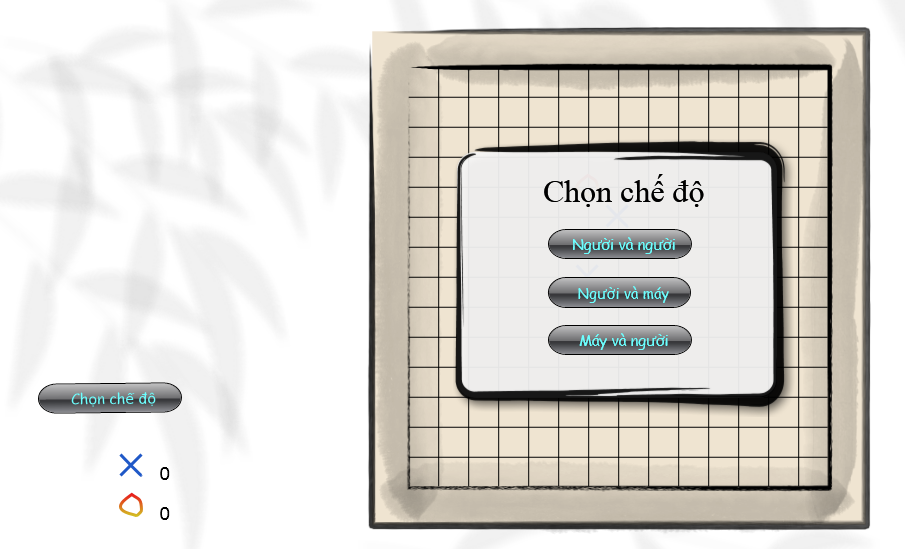
+ Thấy mã giả khá đơn giản nhưng thật ra là phức tạp vô cùng, ngay từ việc thiết kế những hàm đo tiêu chuẩn đã khó rồi, huống chi là tổ hợp của tất cả hàm đó lại ... tuy nhiên cách này sẽ mang lại hiệu quả khá cao dựa trên qui luật bù trừ (ví dụ nếu search nhanh thì mình sinh ra nhiều move-gen và đánh giá chúng sơ sơ, search chậm thì đưa ra ít move-gen nhưng mỗi nước đi đều đã qua bước chọn lọc kỹ càng, chỉ cần đem 1 nước đi ra là đã có ngay một nước đi khá tốt ). Ngoài ra nếu phối hợp cách này với những kỹ thuật ở trên thì càng tuyệt, ví dụ đơn giản là mình chỉ sinh những nước đi trong vòng bán kính 2 ô so với những ô đã đi và lựa những nước đi thoả mãn thang đo nào đó ( nếu đang ở mode offensive thì sẽ sinh những nước gây ra đường đe doạ còn ngược lại đang ở mode defensive thì tìm được những nước có triển vọng của đối phương mà đánh... ), khi đó, số lượng những nước đi thoả mãn các tiêu chuẩn trên sẽ không nhiều (tức là độ phân nhánh thấp) mà chất lượng của nước đi sẽ được nâng cao, khi đó ta còn có thể gia tăng độ sâu (cố định lẫn tĩnh) lên thêm vài bậc.

### Chương III: Thiết kế chương trình và các hàm chính của chương trình

#### 3.1. Thiết kế chương trình

##### *3.1.1. Giao diện chính*

- Website chơi cờ caro được xây dựng bằng công nghệ Silverlight được kết hợp bởi ngôn ngữ XAML và ngôn ngữ C# trong *code-behide.* Sau đây là phần minh họa giao diện chính của chương trình chơi cờ.



8

7

1

2

9

6

5

4

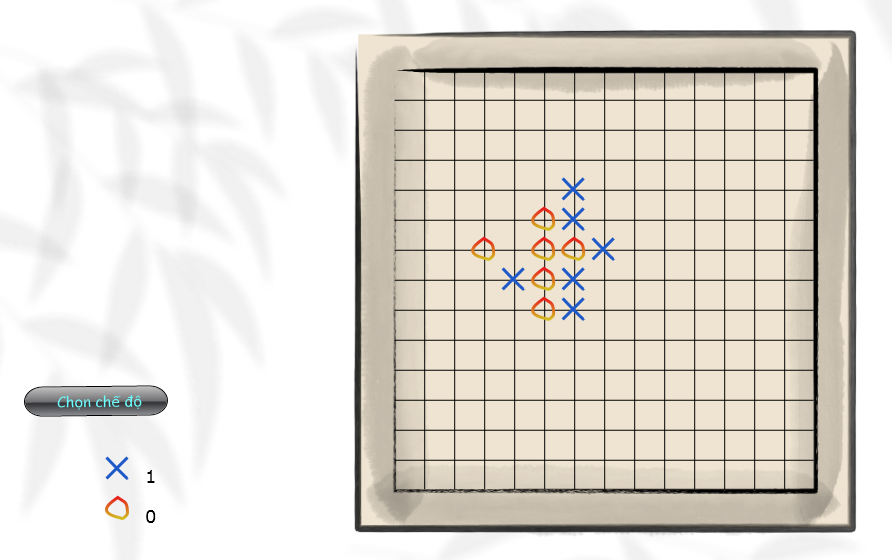
3

Hình 3.1: Giao diện chính

Trong đó:

1. Nút lựa chọn kiểu chơi. Khi click vào nút này thì bảng Select Model sẽ hiện ra cho phép người chơi chọn kiểu chơi.
2. Bàn cờ
3. Bảng lựa chọn kiểu chơi.
4. Nút Player vs Player. Khi click vào nút này thì có thể chơi cờ giữa 2 người với nhau.
5. Chọn người chơi với máy tính, người sẽ đi trước.
6. Máy tính chơi với người. Máy tính sẽ đi nước đầu tiên.
7. Lưu điểm của người chơi thứ nhất.
8. Lưu điểm của người chơi 2.
9. Là hình ảnh những lá tre chuyển động như là có gió thổi.

##### *3.1.2. Giao diện bàn cờ khi đang chơi*



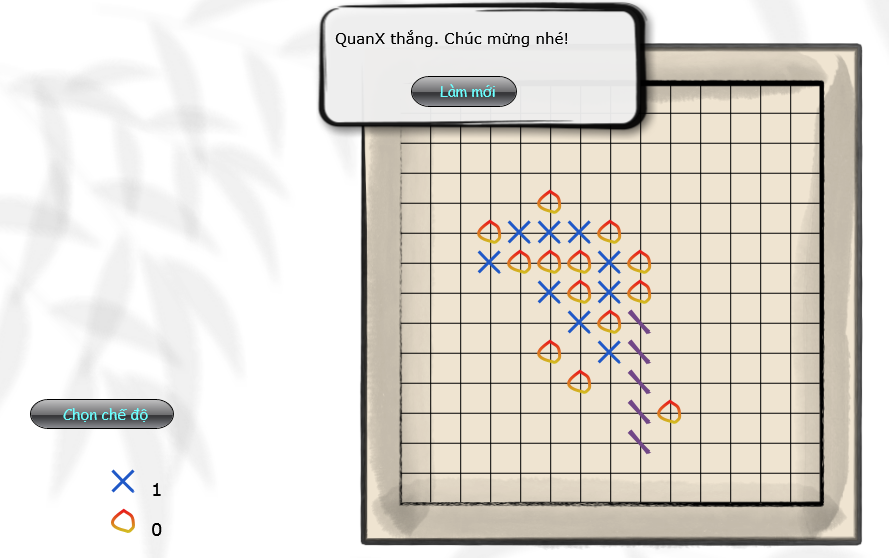
Hình 3.2: Khi đang chơi cờ

##### *3.1.3. Giao diện bàn cờ khi 1 người chơi thắng*

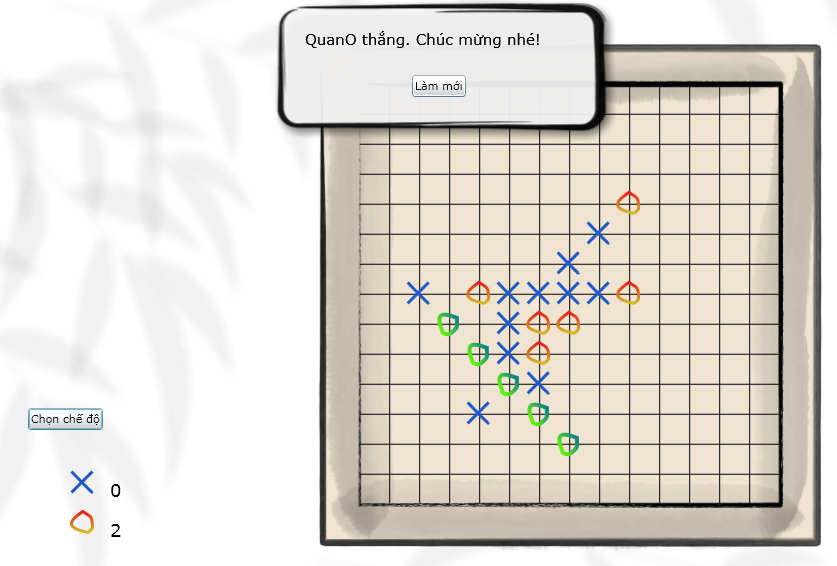
- Khi 1 người chơi thắng thì sẽ xuất hiện thông báo là người đó thắng và cộng điểm vào bảng ghi điểm của người đó.

- Đồng thời 5 con cờ thẳng hàng sẽ được đổi màu liên tục từ màu trắng sang màu đỏ và từ màu đỏ sang màu trắng.

- Khi người chơi Click vào nút Reset thì sẽ bắt đầu 1 ván mới.



Hình 3.3: Khi 1 người chơi thắng



* Khi người chơi đó tiếp tục thắng thì điểm sẽ được tăng lên. Trong ví dụ trên người chơi trắng đã thắng người chơi đen với tỉ số là 2:0.

#### 3.2. Các hàm của chương trình

##### *3.2.1. Mô hình PureMVC*

Kiến trúc Model-View-Controller(MVC) là gì?

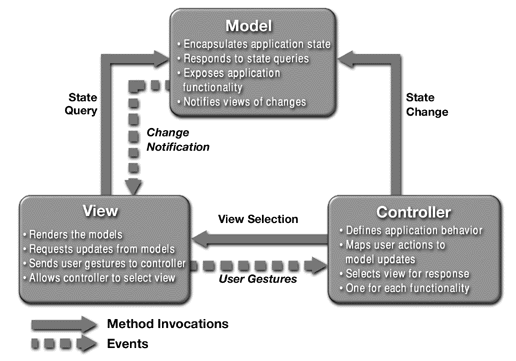
Kiến trúc MVC là việc chia tất cả mục của một ứng dụng ra làm ba thành phần (component) khác nhau Model, View và Controller. Các thành phần của kiến trúc MVC một trách nhiệm duy nhất và không phụ thuộc vào các thành phần khác. Những sự thay đổi trong một thành phần sẽ không có hoặc là có rất ít ảnh hưởng đến các thành phần khác. Các trách nhiệm của mỗi thành phần là:

- Model: Model được giao nhiệm vụ cung cấp dữ liệu cho cơ sở dữ liệu và lưu dữ liệu vào các kho chứa dữ liệu. Tất cả các nghiệp vụ logic được thực thi ở Model. Dữ liệu vào từ người dùng sẽ thông qua View được kiểm tra ở Model trước khi lưu vào cơ sở dữ liệu. Việc truy xuất, xác nhận, và lưu dữ liệu là một phần của Model.

- View: View hiển thị các thông tin cho người dùng của ứng dụng và được giao nhiệm vụ cho việc nhận các dữ liệu vào từ người dùng, gửi đi các yêu cầu đến bộ điều khiển, sau đó là nhận lại các phản hồi từ bộ điều khiển và hiển kết quả cho người dùng. Các trang HTML, JSP, các thư viện thể và các file nguồn là một phần của thành phần View.

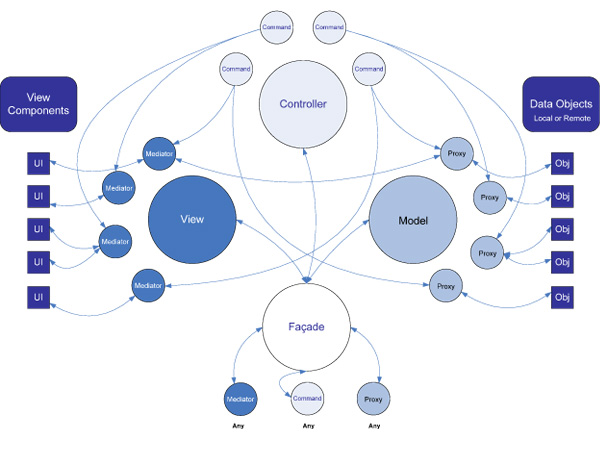
- Controller: Controller là tầng trung gian giữa Model và View. Controller được giao nhiệm vụ nhận các yêu cầu từ phía máy khách. Một yêu cầu được nhận từ máy khách được thực hiện bởi một chức năng logic thích hợp từ thành phần Model và sau đó sinh ra các kết quả cho người dùng và được thành phần View hiển thị.

Trong MVC nguyên tắc làm việc của các phần (Model, Controller và View) có mối liên hệ chặt chẽ như hình minh họa:



Cách này có vài nhược điểm các thành phần phải tạo liên kết tới nhau, thay đổi 1 thành phần sẽ có nguy cơ cả cái MVC bị lỗi.

PureMVC có thể khắc phục được vấn đề đó. Thực chất PureMVC chỉ tạo thêm một lớp trung gian giữa 3 cái **Model** **View** và **Controller**, để người dùng có thể thay đổi các Model, View và Controller dễ dàng hơn. Do đó trong PureMVC có thêm 3 phần nữa gồm có : **Proxy**, **Command**và **Mediator**. Trong framework, 3 phần chính này đã được thiết lập để cách liên lạc với nhau rồi, và 3 phần này sẽ phần trung gian của  **Model**, **View**, **Controler**. Proxy sẽ lo cho Model, Command sẽ lo cho Controller, và Mediator sẽ lo cho View. Sau đây là hình minh họa PureMVC:



Theo hình trên, ngoài ba phần Model, View và Control thì còn có một phần là **Facade**. Nhờ có phần này mà ba phần kia trong PureMVC làm việc được với nhau. **Facade** tạo ra ba phần kia làm trung gian giữa 3 phần này. Do đó framework PureMVC giúp cho người dùng có thể thay đổi các UI hay Data Object một cách dễ dàng mà không sợ làm ảnh hưởng tới các phần khác.

##### 3.2.2. Một số lớp của chương trình

1. *Hàm MainPage.cs:*

ApplicationFacade.Instance.Startup(this);

1. *ApplicationFacadate.cs:*

protected override void InitializeController()

{

base.InitializeController();

RegisterCommand(Thongbao.khoidong, typeof(Khoidong));

RegisterCommand(Thongbao.khoidonglai, typeof(Restart));

RegisterCommand(Thongbao.batdaugame, typeof(Batdaugame));

RegisterCommand(Thongbao.chonchedo, typeof(Chonchedo));

}

1. *Controller: Khoidong.cs:*

var applicationProxy = Facade.RetrieveProxy(Thaotacvoiform.ten) as Thaotacvoiform;

var pieceboardProxy = Facade.RetrieveProxy(Kiemtrabanco.ten) as Kiemtrabanco;

var pieceboardMediator = Facade.RetrieveMediator(PieceboardMediator.NAME) as PieceboardMediator;

var applicationMediator = Facade.RetrieveMediator(ApplicationMediator.NAME) as ApplicationMediator;

1. *Model (Thaotacvoiform.cs):*

public void Capnhattyso(Kieuquan kieu)

{

if (kieu == Kieuquan.QuanX)

thongke["QuanXthang"]++;

if (kieu == Kieuquan.QuanO)

thongke["QuanOthang"]++;

}

public void Lammoityso()

{

thongke["Xthang"] = 0;

thongke["QuanOthang"] = 0;

}

public int BlackWinsCount

{

get { return thongke["QuanXthang"]; }

}

public int WhiteWinsCount

{

get { return thongke["QuanOthang"]; }

}

1. Lớp AI.cs(Thuộc phần thao tác)

Lớp này tìm nước đi cho máy tính.

public class AI

{

private static AI \_instance;

public static AI GetInstance()

{

if (\_instance == null)

\_instance = new AI(true);

return \_instance;

}

private bool mifis;

public int X { get; set; }

public int Y { get; set; }

public AI(bool ifis)

{

mifis = ifis;

}

public void Danh(int[,] banco)

{

int[,] q = new int[Khaibaobien.dodai, Khaibaobien.dodai];

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 15; j++)

{

if (banco[i, j] != -1)

{

q[i, j] = -1;

}

else

{

q[i, j] = FindQz(i, j, banco);

}

}

}

Gangiatrilonnhat(q);

}

public void Gangiatrilonnhat(int[,] q)

{

int max = 0;

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 15; j++)

{

if (q[i, j] > max)

{

X = i;

Y = j;

max = q[i, j];

}

}

}

}

public int FindQz(int x, int y, int[,] board)

{

int qz = 0;

const int w1 = 10000000;

const int w2 = 50000;

const int w3 = 10000;

const int w4 = 5000;

const int w5 = 1000;

const int w6 = 500;

const int w7 = 100;

const int w8 = 50;

const int w9 = -100000000;

var move = new int[4];

board[x, y] = mifis ? 0 : 1;

move[0] = Luatchoi.X1(x, y, board);

move[1] = Luatchoi.X2(x, y, board);

move[2] = Luatchoi.X3(x, y, board);

move[3] = Luatchoi.X4(x, y, board);

if (x == 7 && y == 7)

{

qz += 1;

}

for (int i = 0; i < 4; i++)

{

if (Trituyetdoi(move[i]) == 5)

{

qz += w1;

}

else if (move[i] == 4)

{

qz += w3;

}

else if (move[i] == 3)

{

qz += w5;

}

else if (move[i] == 2)

{

qz += w7;

}

if (mifis)

{

if (Luatchoi.Fails(move, board[x, y]))

{

qz += w9;

}

}

}

board[x, y] = mifis ? 1 : 0;

move[0] = Luatchoi.X1(x, y, board);

move[1] = Luatchoi.X2(x, y, board);

move[2] = Luatchoi.X3(x, y, board);

move[3] = Luatchoi.X4(x, y, board);

for (int i = 0; i < 4; i++)

{

if (Trituyetdoi(move[i]) == 5)

{

qz += w2;

}

else if (move[i] == 4)

{

qz += w4;

}

else if (move[i] == 3)

{

qz += w6;

}

else if (move[i] == 2)

{

qz += w8;

}

}

board[x, y] = -1;

return qz;

}

public int Trituyetdoi(int x)

{

if (x < 0)

{

return -x;

}

return x;

}

}

1. *Kiemtrabanco.cs*

public class Kiemtrabanco:Proxy

//<const><kiểu dữ liệu><tên hằng> = <giá trị>;

{

public new const string ten = "PieceboardProxy";

public Quanco[,] quan;

private List<KQthang> dskq;

public Kiemtrabanco()

: base(ten)

{

quan = new Quanco[Khaibaobien.dodai, Khaibaobien.dodai];

dskq = new List<KQthang>();

}

public bool Kiemtraotrong(Quanco q)

{

if (quan[q.X, q.Y] == null)

return true;

SendNotification(Thongbao.vitrikhongdung);

return false;

}

public void Chenquanco(Quanco q)

{

if (Kiemtraotrong(q))

{

SendNotification(Thongbao.quancoduocchenvaobang, q);

}

}

public void Kiemtranguoithang(Quanco q)

{

dskq.Clear();

quan[q.X, q.Y] = q;

if (Kiemtrathuattoan(q.X - 4, q.Y - 4, 1, 1, 9, q.kieu)

|| Kiemtrathuattoan(q.X - 4, q.Y, 1, 0, 9, q.kieu)

|| Kiemtrathuattoan(q.X, q.Y - 4, 0, 1, 9, q.kieu)

|| Kiemtrathuattoan(q.X - 4, q.Y + 4, 1, -1, 9, q.kieu))

{

SendNotification(Thongbao.nguoithang, dskq);

}

else

{

SendNotification(Thongbao.tieptuc);

}

}

private bool Kiemtrathuattoan(int batdauX, int batdauY, int buocX, int buocY, int tongsobuocdi, Kieuquan kieu)

{

KQthang kq = new KQthang();

List<Quanco> ds = new List<Quanco>();

for (int i = 0; i < tongsobuocdi; i++)

{

int x = batdauX + buocX \* i;

int y = batdauY + buocY \* i;

if (x >= 0 && y >= 0 && x < Khaibaobien.dodai && y < Khaibaobien.dodai)

{

if (quan[batdauX + buocX \* i, batdauY + buocY \* i] != null &&

quan[batdauX + buocX \* i, batdauY + buocY \* i].kieu == kieu)

ds.Add(quan[batdauX + buocX \* i, batdauY + buocY \* i]);

else

ds.Clear();

if (ds.Count >= 5)

{

kq.ds = ds;

kq.kieuquan = kieu;

if (buocX == 1 && buocY == 1)

kq.Kieu = Kieuchienthang.Tutrenxuong;//cheo x

if (buocX == 1 && buocY == 0)

kq.Kieu = Kieuchienthang.Theochieungang;

if (buocX == 0 && buocY == 1)

kq.Kieu = Kieuchienthang.Theochieudoc;//doc xuong duoi

if (buocX == 1 && buocY == -1)

kq.Kieu = Kieuchienthang.Tuduoilen;

dskq.Add(kq);

return true;

}

}

}

return false;

}

public void Xoaquan()

{

quan = new Quanco[Khaibaobien.dodai, Khaibaobien.dodai];

}

}

# PHẦN III: KẾT LUẬN

## 1. Kết quả đạt được

Qua quá trình tìm hiểu nhóm chúng em đã nắm được một số kiến thức cơ bản của công nghệ Silverlight và đã áp dụng vào xây dựng chương trình chơi cờ caro với giao diện sinh động, đơn giản mà hấp dẫn.

Sau thời gian thực hiện đề tài, chương trình đã hoàn thành và đạt được một số kết quả sau:

* Website chơi cờ caro có giao diện dễ nhìn, dễ sử dụng.
* Trò chơi có thể chơi 2 người với nhau hoặc người chơi với máy

## 2. Hạn chế của đề tài

Website chưa xây dựng được chưa có nhiều các chức năng như có thể lưu lại ván cờ để lần sau có thể chơi tiếp, hay có thể vừa chơi vừa nghe nhạc trên web, thay đổi giao diện.

## 3. Hướng phát triển của đề tài

Với sự cố gắng, lỗ lực nhóm chúng em đã cố gắng hoàn thành yêu cầu đề tài. Do thời gian và năng lực có hạn nên website mới chỉ đáp ứng được chơi cờ caro bình thường mà chưa có các chức năng tiện ích khác.

Hướng phát triển Website trở thành một chương trình chơi cờ với đầy đủ các chức năng và có một giao diện bắt mắt, dễ sử dụng hơn nữa.

## 4. Đề nghị ý kiến.

Trong thời gian thực hiện đề tài này nhóm đề tài không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được ý kiến đóng góp từ phía quý Thầy Cô và các bạn để nhóm hoàn thiện hơn.

# PHẦN IV: TÀI LIỆU THAM KHẢO

[1]. Giáo trình silverlight

[2]. [**www.msdn.microsft.com**](http://www.msdn.microsft.com/)

[3]. [**www.msdnvietnam.net**](http://www.msdnvietnam.net/)

[4]. [**www.codeproject.com**](http://www.codeproject.com/)