브러시

* **계층구조**
* **Object**
* **DispatcherObject (abstract)**
* **DependencyObject**
* **Freezable (abstract)**
* **Animatable (abstract)**
* **Brush (abstract)**
* **GradientBrush (abstract)**
* **LinearGradientBrush**
* **RadialGradientBrush**
* **SolidColorBrush**
* **TileBrush (abstract)**
* **DrawingBrush**
* **ImageBrush**
* **VisualBrush**
* **WPF의 Color 설정.**
* WPF에서 Color설정 하는 총 3가지 방법으로 구성된다.

1. **Color구조체를 이용한 색상 설정**

WPF는 색상을 다루기 위해 **System.Windows.Media** 네임스페이스에 정의된 **Color 구조체**를 사용한다. Color구조체는 색상을 표현하기 위한 R, G, B채널과 투명도를 위한 Alpha채널 프로퍼티가 존재한다. Alpha채널은 색상의 투명도를 결정하며0(투명) ~ 255(불투명) 값을 가진다.

- Color 구조체를 RGB로 초기화 시키는 방법은 세가지

|  |  |
| --- | --- |
| 1. 객체 생성 뒤 초기화 | Ex) Color clr = new Color();  clr.A = 255; //알파값 설정  clr.R = 212; R값 설정  clr.G = 12; G값 설정  clr.B = 120; B값 설정 |
| 1. FromRgb 메소드를 이용 | Color clr = Color.FromRgb(r,g,b);   * R,g,b는 byte형.. * 이때 A값은 자동으로 255(불투명)으로 설정됨 |
| 1. FromArgb 메소드를 이용 | Color clr = Color.FromArgb(a,r,g,b); |

1. **scRGB를 이용한 색상 설정**

**- sRGB & scRGB**

**sRGB**란 R,G,B(빨강,녹색,파란색)으로 결정되는 RGB의 색공간을 뜻한다.

일반적인 비디오, 디스플레이 보드(디지털 카메라&스캐너 등…)로부터 모니터로 출력하는데 일반적으로 쓰는 색 공간.

**scRGB**란 컬러를 표현하는 또 다른 방법중 하나이다.

Black보다 더 진한 Black, White보다 더 하얀 White를 표현할 수 있다..

이러한 특징은 프린터처럼 모니터보다 훨씬 큰 범위의 색을 표현하는 기계에 유용하다.

scRGB도 Color구조체 내에 존재하며 **Float값으로 색의 성분이 저장된다.**

scRGB는 색 공간을 표현해주기 위해 **ScA, ScR, ScG, ScB라는 네가지 프로퍼티가** 존재한다

scRGB는 RGB와 비슷하게 사용된다.

RGB의 0 = scRGB의 0과 같고

RGB의 255 = scRGB의 1과 같다고 생각하면 된다.

(0보다 작거나 1보다 큰 경우 sRGB보다 더욱더 강하거나 약한색을 표현이 가능하다)

|  |  |
| --- | --- |
| 1. 객체 생성 뒤 초기화 | Ex) Color Clr = new Color();  Clr.ScA = 0.1f;  Clr.ScB = 1.0f; Clr.ScG = 1.5f; Clr.ScR = 1.1f; |
| 1. FromScRgb 메소드 이용 | Ex) Color clr = Color.FromScRgb(a, r, g, b);   * a,r,g,b는 float형태..(0.0f~1.0f) * 0.0f~1.0f의 범위에서는 일반적인 sRGB와 동등한 색을 표현하나, 0보다 작거나 1보다 큰 경우 sRGB보다 더욱더 강하거나 약한 색을 표현이 가능하다. |

sRGB 표현법에 맞는 ScR, ScG, ScB, ScA 구하기 (단, 지수 2.2는 SRGB표준에서 가정한 감마 값)

ScR ≒ , ScG ≒ , ScB ≒ , ScA ≒

1. **System.Windows.Medi Color 클래스를 이용한 색상 설정.**

Colors 클래스에는 알파벳 순으로 141개의 읽기전용 색상 프로퍼티를 재공한다.

|  |
| --- |
| Ex) Color clr = Colors.PapayaWhip; |

* **WPF의 브러시**

-브러시와 관련된 모든 클래스는 **System.Windows.Media** 네임스페이스 내부에 존재한다.

대표적인 브러시로 단색을 표현하는 **SolidColorBrush와 GradientBrush**가 있다.

1. **SolidColorBrush**

|  |
| --- |
| Color clr = Color.FromRgb(0, 255, 255); //Color 설정  SolidColorBrush brush = new SolidColorBrush(clr); //Brush Color 설정  Background = brush;  // Background = new SolidColorBrush(Color.FromRgb(0,255,255); 이나,  //SolidColorBrush brush = new SolidColorBrush();  //brush.Color = Color.FromRgb(255,255,255);  //와 같이 표현이 가능함.   * Background는 윈도우의 배경색을 나타내는 프로퍼티이다. |

1. **GradientBrush**

-Gradient란 한 점의 색상이 다른 점의 색상으로 점진적으로 섞어서 두 개의 색상을 표현하는 방법이다.

-GradientBrush는 두 가지의 종류가 있다.

선의 형태로 표현되는 **LinearGradientBrush**와 원형으로 뿌려지는 **RadialGradientBrush**

**2.1) LinearGradientBrush**

LinearGradientBrush는 두개의 Color 객체와 Point 객체로 초기화 하거나, 표현할 각도를 이용하여 초기화 할 수도 있다.

|  |
| --- |
| LinearGradientBrush Gbrush = LinearGradientBrush(clr1, clr2, pt1, pt2);  //또는  LinearGradientBrush Gbrush = LinearGradientBrush(clr1, clr2, angle); |

< LinearGradientBrush 초기화 >

WPF에서는 클라이언트 영역의 좌 상단 끝을 (0,0) / 우 하단 끝을 (1,1)로 설정한다.

클라이언트 영역의 크기가 변하더라도 계속 일정한 Gradient를 표현해 줄 수 있다.

|  |
| --- |
| 2개의 Color객체와 2개의 Point객체로 LinearGradientBrush를 사용하는 방법<cs> |
| public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  LinearGradientBrush brush;  brush = new LinearGradientBrush();  brush.StartPoint = new Point(0, 0);  brush.EndPoint = new Point(0, 1);  brush.GradientStops.Add(new GradientStop(Colors.OrangeRed, 0));  brush.GradientStops.Add(new GradientStop(Colors.LightYellow, 1));  brush = new LinearGradientBrush(Colors.OrangeRed, Colors.DarkRed, 90);  brush = new  LinearGradientBrush(Colors.OrangeRed, Colors.DarkRed, new Point(0, 0), new Point(0, 1));  GradientStopCollection stopCollection = new GradientStopCollection();  stopCollection.Add(new GradientStop(Colors.OrangeRed, 0));  stopCollection.Add(new GradientStop(Colors.LightYellow, 1));  brush = new LinearGradientBrush(stopCollection);  brush.StartPoint = new Point(0, 0);  brush.EndPoint = new Point(0, 1);  brush = new LinearGradientBrush(stopCollection, 90);  brush = new LinearGradientBrush(stopCollection, new Point(0, 0), new Point(0, 1));  this.Background = brush;  }  } |

|  |
| --- |
| xaml |
| <Window x:Class="\_1110\_test.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  <Window.Background>  <LinearGradientBrush>  <LinearGradientBrush.StartPoint>  <Point X="0" Y="0" />  </LinearGradientBrush.StartPoint>  <LinearGradientBrush.EndPoint>  <Point X="0" Y="1" />  </LinearGradientBrush.EndPoint>  <LinearGradientBrush.GradientStops>  <GradientStopCollection>  <GradientStop Color="OrangeRed" Offset="0" />  <GradientStop Color="LightYellow" Offset="1"/>  </GradientStopCollection>  </LinearGradientBrush.GradientStops>  </LinearGradientBrush>  </Window.Background>  </Window> |

실행결과.



두 번째는 각도를 이용하여 초기화 할 수 있는데, 이때 double 형으로 각도를 표현할수 있다.

또한, LinearGradientBrush는 여러 영역에서의 각각 다른 Griaiente도 구현 할 수 있는데,

이 때 GradientStop 프로퍼티가 사용 된다.

|  |
| --- |
| New GradientStop(clr, offset); //offset은 0~1까지의 값 |

GradientStop의 첫번째 인자를 활용하여 Gradient의 색을 입히고, 두 번째 offset값을

활용하여 클라이언트에 그릴 곳의 상대적인 거리를 나타내 준다.

**2.2) RadialGradientBrush**

타원형 Gradient를 표현할 수 있는 RadialGradientBrush는 네 가지 프로퍼티를 가진다.

**RadiusX**, **RadiousY**는 double 값을 가지며 타원의 수평(x)와 수직(y)값을 나타낸다.

**Center** 프로퍼티는 Point객체로써, 브러시가 미치는 중심이다.

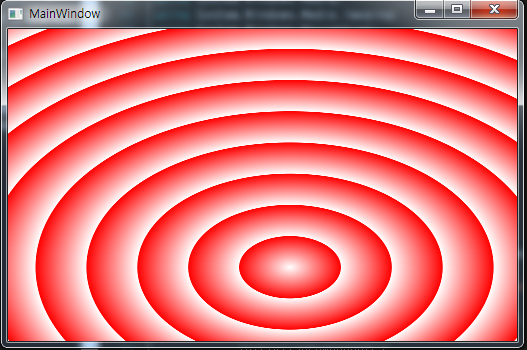
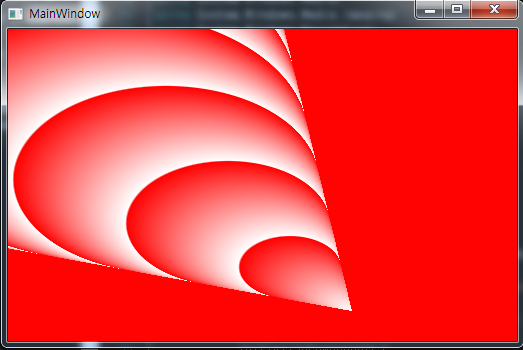
**GradientOrgin**은 Gradient가 시작되는 중심이다.

\*\*모든 프로퍼티의 기본값은 0.5이다.

|  |
| --- |
| namespace Rainbow  {  /// <summary>  /// MainWindow.xaml에 대한 상호 작용 논리  /// </summary>  public partial class MainWindow : Window  {  RadialGradientBrush brush;  public MainWindow()  {  Title = "Click the Gradient Center";  brush = new RadialGradientBrush(Colors.White, Colors.Red);  brush.RadiusX = brush.RadiusY = 0.10;  brush.SpreadMethod = GradientSpreadMethod.Repeat;  Background = brush;  InitializeComponent();  }  private void Window\_MouseDown(object sender, MouseButtonEventArgs e)  {  double width = ActualWidth - 2 \* SystemParameters.ResizeFrameVerticalBorderWidth;  double height = ActualHeight - 2 \* SystemParameters.ResizeFrameHorizontalBorderHeight - SystemParameters.CaptionHeight;  Point ptMouse = e.GetPosition(this);  ptMouse.X /= width;  ptMouse.Y /= height;  if (e.ChangedButton == MouseButton.Left)  {  brush.Center = ptMouse;  brush.GradientOrigin = ptMouse;  }  else if(e.ChangedButton == MouseButton.Right)  {  brush.GradientOrigin = ptMouse;  }  }  }  } |
| <Window x:Class="Rainbow.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="MainWindow" Height="350" Width="525" MouseDown="Window\_MouseDown">      </Window> |

Center프로퍼티와 GradientOrigin프로퍼티의 차이

실행결과.

1. **BorderBrush**

BorderBrush는 클라이언트 전체 영역에서의 경계선을 그리는데에 사용이 된다. 이 때 BorderThickness 프로퍼티와 ThickNess 구조체를 이용하여 초기화 시켜 줄 수 있다.

|  |
| --- |
| BorderBrush = Brushes.SaddleBrown;  BorderThickness = new Thickness(25, 50, 75, 100); |

또한 클라이언트와 마찬가지로 경계에도 그라디언트를 사용할 수 있다.

* **FreezaBle**

Freezable 클래스란 개체의 수정가능 상태와 고정 상태를 정의한다. Freezable 클래스를 상속받은 클래스는 개체 변경에 대한 알림을 제공 받을 수 있으며, 고정상태로 만들거나, 자신을 복제(clone) 할 수 있게 한다. 디폴트 값으로 UnFrozen 상태(수정 가능 상태)로 지정하나, Freeze() 메소드로 자신이 원하는 객체를 Frozen 상태로 만들 수 있다.

- 2가지의 상태를 가지고 있다 → Unfrozen/frozen State

- frozen 된 Freezable은 더 이상 변경할 수 없음.

- Freezable을 Freezing하면 변경 notification에 대한 처리를 더 이상 할 필요가 없으므로 성능을 높일 수 있음

- WPF의 대부분의 Freezable 객체는 그래픽 sub-system

- Freeze 메소드를 이용하여 객체를 Frozen 상태로 만듦.

if (myBrush.CanFreeze)

{

// Makes the brush unmodifiable.

myBrush.Freeze();

}

* **Using Freezable**

- unfrozen 상태의 객체는 다른 객체의 사용법과 비슷

Button myButton = new Button();

SolidColorBrush myBrush = new SolidColorBrush(Colors.Yellow);

myButton.Background = myBrush;

// Changes the button's background to red.

myBrush.Color = Colors.Red;

- Freeze 메소드를 사용하여 frozen 상태로 변경

- 다음의 경우는 frozen 상태로 만들 수 없다

. 애니메이션이 되고 있는 중이거나, data bound 속성을 가진 경우

. Dynamic 리소스로 설정된 속성을 가지고 있는 경우

. sub-object로 frozen 상태가 될 수 없는 객체를 가지고 있는 경우

- Freeze 메소드를 호출하면 더 이상 변경되거나, unfrozen 상태로 될 수 없다. 이를 위반할 시 예외가 발생한다.

- IsFrozen 메소드를 사용하여 현재 Freezable 객체가 frozen 상태인지 파악할 수 있음

- XAML에서는 PresentationOptions:Freeze=”True” 코드로 frozen 상태로 만들 수 있음  
xmlns:PresentationOptions="http://schemas.microsoft.com/winfx/2006/xaml/presentation/options"

네임스페이스 필요

- frozen된 객체의 변경 : Clone 메소드를 사용하여 변경할 수 있다.

|  |
| --- |
| Button myButton = new Button();  SolidColorBrush myBrush = new SolidColorBrush(Colors.Yellow);  if (myBrush.CanFreeze)  {  myBrush.Freeze();  }  myButton.Background = myBrush;  SolidColorBrush myBrushClone = myBrush.Clone();  myBrushClone.Color = Colors.Red;  myButton.Background = myBrushClone; |

* **Creating Your Own Freezable Class**

- Freezable을 상속함으로써 얻어지는 장점

. Special State : read-only, writable State

. Thread 안정성 : frozen Freezable은 쓰레드간 공유가 가능

. 손쉬운 복제 : 객체의 deep copy를 제공하는 여러 복제 메소드 제공

- Freezable을 상속받아 구현되는 모든 클래스에는 CreateInstanceCore 메소드 재정의 해야 함.

- 클래스에 non\_dependency property를 가지고 있다면, CloneCore, CloneCurrentValueCore,

GetAsFrozenCore, GetCurrentValueAsFrozenCore, FreezeCore 메소드를 재정의 해야 함

- not dependency 속성 data 멤버에 접근하기 위해서는 다음을 따라야 함

. 읽기 전에 ReadPreamble 메소드 호출

. 쓰지 전에 WritePreamble 메소드 호출

. 쓰는 메소드 종료 전에 WritePostscrip 메소드 호출

- Dependency Object 인 non-dependency 속성을 데이터 멤버로 가지고 있는 경우에는

OnFreezablePropertyChanged 메소드를 호출해줘야 함

|  |
| --- |
| private void Window\_MouseMove(object sender, MouseEventArgs e)  {  double width = ActualWidth - 2 \* SystemParameters.ResizeFrameHorizontalBorderHeight;  double height = ActualHeight - 2 \* SystemParameters.ResizeFrameHorizontalBorderHeight  - SystemParameters.CaptionHeight;  //마우스의 위치가 변한 좌표를 가져와서 중심점을 기준으로 동일한 색을 나타내게 설정함(원형)  Point ptMouse = e.GetPosition(this);  Point ptCenter = new Point(width / 2, height / 2);  Vector vectMouse = ptMouse - ptCenter;  double angle = Math.Atan2(vectMouse.Y, vectMouse.X);  Vector vectEllipse = new Vector(width / 2 \* Math.Cos(angle), height / 2 \* Math.Sin(angle));  Byte byLevel = (byte)(255 \* (1 - Math.Min(1, vectMouse.Length / vectEllipse.Length)));  //색깔 채우고 배경에 표현  Color clr = Color.FromRgb(byLevel, byLevel, byLevel);  brush.Color = clr;  Background = brush;  } |

예제 1)

위의 예제 1은 마우스가 움직일 때마다 호출되며, 배경색을 바꿔주게 된다.

그런데 클라이언트 영역을 다시그리는 코드(Invalidate)가 없어도 해당 영역이 다시 그려진다.

이는 Brush가 Freezable 클래스를 상속 받앗기 때문이다.

Freezable 클래스란 개체의 수정가능 상태와 고정 상태를 정의한다. Freezable 클래스를 상속받은 클래스는 개체 변경에 대한 알림을 제공 받을 수 있으며, 고정상태로 만들거나, 자신을 복제(clone) 할 수 있게 한다. 디폴트 값으로 UnFrozen 상태(수정 가능 상태)로 지정하나, Freeze() 메소드로 자신이 원하는 객체를 Frozen 상태로 만들 수 있다.