02장 – 브러시

1)WPF의 Color 설정

1-1) Color 구조체를 이용한 색상 설정

WPF에서는 색상을 다루기 위해 System.Windows.Media 네임스페이스에 정의된 Color구조체 사용한다

Color 구조체 안에는 색을 나타내는 R,G,B 외에도 알파 채널(Alpha Channel)을 의미하는 A라는 이름의 프로퍼티를 가지고 있다(0~255 : 투명~불투명)

Color 구조체를 RGB로 초기화 시키는 방법은 세가지가 있다.

|  |  |
| --- | --- |
| 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); |

표 1-1 RGB색상 설정하기

1-2) scRGB를 이용한 색상 설정

※ sRGB(standard RGB)와 scRGB란?

sRGB란 R(빨강), G(녹색), B(파란색)으로 결정되는 RGB의 색 공간을 뜻한다.

일반적으로 비디오 디스플레이 보드(디지털 카메라나 스캐너 등등)로부터 모니터로 출력하는데 일반적으로 쓰는 색 공간이 sRGB이나, RGB가 표현하는 색 보다 더욱더 강한 색을 출력하는 장치를(예를 들어 프린터) 모니터에서 표현해 주기 위해 scRGB라는 방법이 지원됨

scRGB도 Color 구조체 내에 존재하며 float 값으로 색의 성분이 저장이 되게 된다.

Color 구조체에서 scRGB 색 공간을 표현해 주기 위해서 ScA(Alpha Channel), ScR(Red), ScG(Green) Scg(Blue)라는 네가지 프로퍼티가 존재한다.

sRGB의 프로퍼티와 scRGB 프로퍼티는 별개로 존재하는 값이 아니다. 예를 들어서 G 프로퍼티를 바꾸게 되면 ScG프로퍼티도 바뀌게 된다.

scRGB도 sRGB와 초기화 하는 방식이 비슷하다.

|  |  |
| --- | --- |
| 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. |

표 1-2 scRGB색상 설정하기

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

ScR ≒ , ScG ≒ , ScB ≒ , ScA ≒

* 1. System.Windows.Media Colors 클래스를 이용하기

Colors 클래스에는 알파벳 순으로 141개의 읽기전용 색상 프로퍼티를 재공한다.

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

표 1-3 Colors구조체를 이용한 색상 설정하기

2) WPF의 브러시

브러시와 관련된 모든 클래스는 System.Windows.Media 네임스페이스 내부에 존재한다. 대표적인 브러쉬로 단색을 표현하는 SolidColorBrush와 GradientBrush가 있다.

1. **SolidColorBrush 사용하기**

|  |
| --- |
| 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. |

표 1-4 SolidColorBrush 색상 설정하기

예제 1 - 마우스가 움직일 때마다 배경색을 바꿔주는 프로그램

|  |
| --- |
| 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;  } |

* **Freezable 클래스**

예제 1은 마우스가 움직일 때마다 호출되며, 배경색을 바꿔주게 된다. 그런데 클라이언트 영역을 다시 그리는 코드( Invalidate 같은..)가 없어도 해당 영역이 다시 그려진다. 이는 Brush가 Freezable 클래스를 상속 받았기 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.

Freeze() 메소드로 Frozen(고정 상태)가 된 개체는 수정 되거나 다시 UnFrozen 상태로 만들 수 없다. 또한 Frozen 상태가 되면, 더 이상 개체에 Notification에 대한 처리를 더 이상 할 필요가 없다. 따라서 성능을 높일 수 있다.

Freezable을 통해 고정된 객체는 스레드간 에도 공유가 가능하며, 고정된 객체의 복사본을 만들어서 사용 할 수 있다. 예를들면 SolidColorBrush brush = Brushes.Black; 같은 경우는 에러를 발생 시키나, SolidColorBrush brush = Brushes.Black.Clone();을 이용하면 고정된 Brushes.Black 객체의 복사본을 만들어서 brush에 복사하여 사용 할 수 있다.

Freezable 클래스의 Changed 이벤트는 객체의 값이 바뀌었을 때 발생한다. 예제 1에서는 brush.Color = clr; 이라는 코드를 통해서 brush라는 객체의 상태가 변화가 발생 됨에 따라서 브러시에 변화가 생길 때마다 배경을 다시 그릴 수 있게 된다.

1. **그라디언트(Gradient) 브러시 사용하기**

그라디언트란, 한 점의 색상이 다른 점의 색상으로 점진적으로 섞어서 두 개의 색상을 표현하는 방법이다.

그라디언트 브러쉬에는 두가지 종류가 있는데 첫 번째로 선의 형태로 표현되는 그라디언트 브러시인 LinearGradientBrush와 원형으로 뿌려지게 되는 RadialGradientBrush가 있다.

먼저 LinearGradientBrush는 두 개의 Color 객체와 두 개의 Point 객체로 초기화 하거나, 표현할 각도를 이용하여 초기화를 할 수도 있다..

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

표 1-5 LinearGradientBrush 초기화

WPF 에서는 클라이언트 영역의 좌상단 끝을 (0,0), 우하단 끝을 (1,1)로 설정하여, 클라이언트영역의 크기가 변하더라도 계속 일정한 그라디언트를 표현해 줄 수 있다.

두번째는 각도를 이용하여 초기화를 할 수 있는데, 이때 double 형으로 각도를 표현 할 수 있다.

|  |
| --- |
| LinearGradientBrush brush;  public MainWindow()  {  Title = "Adjust the Gradient";  brush = new LinearGradientBrush(Colors.White, Colors.Black, 0);  brush.MappingMode = BrushMappingMode.Absolute;  Background = brush;  InitializeComponent();  }  private void Window\_SizeChanged(object sender, SizeChangedEventArgs e)  {  double Width = ActualWidth - 2 \* SystemParameters.ResizeFrameVerticalBorderWidth;  double Height = ActualHeight - 2 \* SystemParameters.ResizeFrameHorizontalBorderHeight - SystemParameters.CaptionHeight;  Point ptCenter = new Point(Width / 2, Height / 2); //중간 좌표를 구한다.  Vector vectDiag = new Vector(Width, -Height); //좌측 하단부터 우측 상단까지의 대각선  Vector vectPerp = new Vector(vectDiag.Y, -vectDiag.X); //vectDiag와 수직되고, ptCent를 지나는 는 대각선  vectPerp.Normalize(); //정규화 시키기((0,0)~ (1,1))  vectPerp \*= Width \* Height / vectDiag.Length;//Perp를 구하는 이유는 길이를 알아야 Gradient가 그릴 좌표를 알 수 있기 때문이다.  brush.StartPoint = ptCenter + vectPerp; //해당하는 클라이언트의 시작과 끝 좌표를 대입한다.  brush.EndPoint = ptCenter - vectPerp;  } |

예제 2 – 사이즈 변화에 따른 그라디언트 영역을 일정화 시키기.

또한 LinearGradientBrush는 여러 영역에서의 각각 다른 색깔의 그라디언트도 구현 할 수 있는데, 이 때 GradientStops 프로퍼티가 사용 된다.

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

GradientStops의 첫번재 인자를 활용하여 그라디언트의 색을 입히고, 두번째 offset 값을 활용하여 클라이언트에 그릴 곳의 상대적인 거리를 나타내 준다..

|  |
| --- |
| Title = "Follow the Rainbow";  LinearGradientBrush brush = new LinearGradientBrush();  brush.StartPoint = new Point(0, 0);  brush.EndPoint = new Point(1, 0);  brush.GradientStops.Add(new GradientStop(Colors.Red, 0));  brush.GradientStops.Add(new GradientStop(Colors.Orange, 0.17));  brush.GradientStops.Add(new GradientStop(Colors.Yellow, 0.33));  brush.GradientStops.Add(new GradientStop(Colors.Green, 0.5));  brush.GradientStops.Add(new GradientStop(Colors.Blue, 67));  brush.GradientStops.Add(new GradientStop(Colors.Indigo, 84));  brush.GradientStops.Add(new GradientStop(Colors.Violet, 1));  Background = brush;  InitializeComponent(); |

예제 3 – GradientStop을 활용한 무지개 그리기

타원형 그라디언트를 표현할 수 있는 RadialGraientBrush는 네가지 프로퍼티를 가진다. RadiusX와 RadiousY는 double 값을 가지며 타원의 수평(X)와 수직(Y)값을 나타낸다.

Center프로퍼티는 Point객체로써, 브러시가 미치는 중심이다. 마지막으로 GraidentOrigin은 그라디언트가 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;  }  }  }  } |

예제 4 - Center프로퍼티와 GradientOrigin프로퍼티의 차이

예제 4에서 왼쪽 마우스 버튼을 누르게 되면 누른 마우스 좌표를 센터로 지정하면서 그라디언트를 그리기 때문에 중심만 바뀌고, 오른쪽 마우스 버튼을 누르게 되면 선택한 좌표부터 그라디언트를 그리게 된다. 따라서 왼쪽 마우스 버튼을 누르게 되면 그라디언트의 사이 길이가 일정하게(Radius값을 지정 했기 때문에..) 그려지게 되고, 오른쪽 마우스를 센터 포인트 근처에서 누르면 원의 너비가 일정하지 않게 되고, 최초 그라이언트 영역 밖에서 누르게 되면 부채꼴 모양의 그라디언트가 그려지게 된다.

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.