7장 캔버스

- Canvas 패널은 전통적 그래픽 환경에 가장 가까운 레이아웃이다.

여기서 레이아웃이란 컨트롤과 엘리먼트들을 패널에 붙이는 작업을 일컫는 말이다.

엘리먼트가 어디에 위치할지를 좌표로 지정하는 방식이다.

Canvas 패널은 UI요소들을 절대적인 위치로 배치시킴으로써 용자가 Window의 크기를 변경한다 하더라도 Canvas 안의 UI는 움직이지 않고 고정되어 있다. 핵심 속성은 Canvas.Left, Canvas.Right, Canvas.Top, Canvas.Buttom 이다.

- 컨트롤과 같은 여러 엘리먼트들은 Canvas 위에서 적잘하게 크기가 조절된다.

그러나 일부 엘리먼트(Rectangle 이나 Ellipse 클래스)는 이렇게 되지 않으므로

명시적으로 Width와 Height에 값을 대비해야한다.

Canvas 패널 자체의 Width와 Height 프로퍼티에도 값을 지정하는 것이 일반적이다.

- Canvas 패널에 엘리먼트를 겹치는 것도 가능하며, 때로는 그편이 바람직하다. 앞서 보았 던 것처럼 여러 엘리먼트를 Grid의 한 셀에 넣을 수 있었지만 그 결과를 통제하기 어려

웠다. Canvas에서는 통제와 예측의 측면에서 볼 때 엘리먼트의 배치가 상당히 용이하다.

Children 컬렉션에 먼저 추가한 엘리먼트는 나중에 추가한 엘리먼트에 가려진다.

Canvas - 콘텐츠를 위치와 크기를 이용하여 정렬. Canvas의 크기가 재조정되어도

자동으로 재정렬 되지 않음.

UniformGrid - 필요한 만큼의 행과 열을 동일한 크기로 자동생성하고, 그에 따라 콘텐츠 정렬.

Grid - 개발자가 지정한 행과 열에 콘텐츠를 정렬.

UniformGrid의 경우에는 생성되는 컨트롤의 갯수만큼 자동으로 행과 열을 생성하여 배치하게된다. 대신 4\*2표는 안되고 2\*2 / 3\*3 / 4\*4 등처럼 행과 열의 갯수가 같이 설정을 해야된다.

-일반적으로 Canvas 내부 요소의 Z 순서는 XAML 순서에 의해 지정된다. 하지만 Z-Order를 사용하여 명시적으로 정의를 하면 순서가 바뀐다.

ex)<Canvas >

<Ellipse Fill = "Green" Width = "60" Height = "60" Canvas. Left = "30" Canvas. Top = "20" Canvas. ZIndex = "1" />

<Ellipse Fill = "Blue" Width = "60" Height = "60" Canvas. Left = "60" Canvas. Top = "40" />

</Canvas >

<ZIndex사용하기 전> <ZIndex사용한 후>



07\_Canvas 예제PaintTheButton

<Window x:Class="PaintTheButton.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="PaintTheButton" Height="350" Width="525">

<Button HorizontalAlignment="Center" VerticalAlignment="Center">

<Canvas Width="144" Height="144">

<Rectangle Width="144" Height="144" RadiusX="24" RadiusY="24" Fill="Blue">

</Rectangle>

<Polygon Fill="Yellow" Width="144" Height="144">

<Polygon.Points>

<Point X="72" Y="25"></Point>

<Point X="48" Y="105"></Point>

<Point X="115" Y="55"></Point>

<Point X="30" Y="55"> </Point>

<Point X="100" Y="105"></Point>

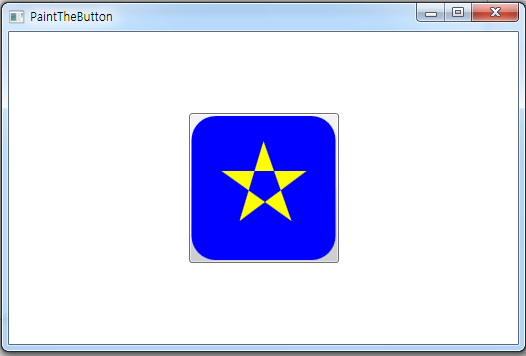
</Polygon.Points>

</Polygon>

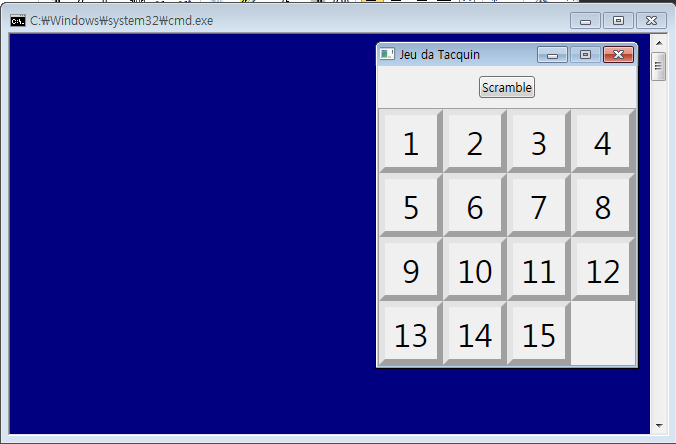
</Canvas>

</Button>

</Window>

자식 요소에 HorizontalAlignment, VerticalAlignment속성을 설정하면 부모 패널과의 남는 공간에 대한 배치를 다르게 할 수 있다. 사각형 모서리를 둥글게 하기위해 RadiusX, RadiusY를 사용하였다. Polygon.Points란 다각형의 꼭지점을 포함하는 컬렉션을 가져오거나 설정한다. Polygon.Points 안에서 Point를 사용하여 X좌표와 Y좌표를 선언해주고, 선언되어진 점들이 서로 연결되면서 아래와 같은 모양이 나오게 된다.

07\_Canvas 예제Tile



그림<1> 예제 소스 Tile 실행화면

그림<1>은 예제 소스를 실행한 화면이다.

Scramble를 누르면 자동적으로 퍼즐들이 섞이게 된다. 마우스로 누르거나 키보드를 입력하면 칸이 움직이게 설정되어있다.

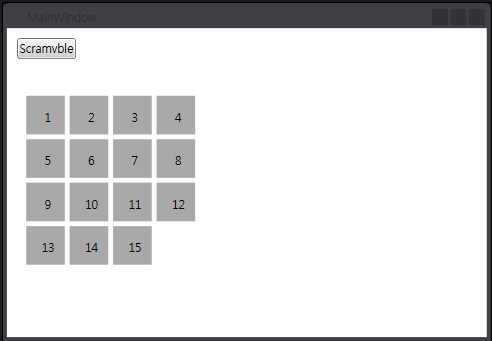
이 소스를 가지고 xaml 형태로 바꾸어 보았다.

|  |
| --- |
| <Window  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d" x:Class="WpfApplication1.MainWindow"  Title="MainWindow" Height="350" Width="505.325" KeyDown="Window\_KeyDown\_1">  <Canvas>  <Button Margin="10" HorizontalAlignment="Center" Content="\_Scramvble" Click="Button\_Click\_1"/>  <UniformGrid Rows="4" Columns="4" Canvas.Left="20" Canvas.Top="70" Height="180" Width="180" Name="unigrid" >  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="1" ></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="2" ></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="3"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="4"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="5"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="6"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="7"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="8"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="9"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="10"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="11"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="12"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="13"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0"/>  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="14"></TextBlock>  </Grid>  <Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">  <Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0" />  <TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="15"></TextBlock>  </Grid>  </UniformGrid>  </Canvas>  </Window> |

표1.Xaml 형태

표1 의 Xaml 형태는 외견을 잡아주는 것을 말해준다.

밑의 그림은 실행했을 때 나오는 화면이다.



그림<2> Xaml실행화면

버튼은 평소에 만들듯이

<Button Margin="10" HorizontalAlignment="Center" Content="\_Scramvble" Click="Button\_Click\_1"/>로 만들었다.

밑의 숫자가 들어있는 박스들은 UniformGrid 형태 안에다가 폴리곤과 그리드를 써서 만들었다.

UniformGrid는 행과 열을 정해주면 그거에 맞게 도형 등을 정열 시켜준다.

<UniformGrid Rows="4" Columns="4" Canvas.Left="20" Canvas.Top="70" Height="180" Width="180" Name="unigrid" >

이름을 unigrid로 정해주고 로우와 컬럼 을 4로주고 캔버스 안에서 사용하기에 캔버스 위치를 이용해 위치를 정해주었다.

폴리곤과 그리드를 같이 쓰는 이유는 폴리곤 자체는 도형을 그릴 수는 있지만 그 안에 텍스트를 넣기위해 그리드형태로 폴리곤을 감싸주었다.

<Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">

<Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0" />

<TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="15"></TextBlock>

</Grid>

기능 구현은 밑에 표2를 보면서 설명하겠다.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Navigation;  using System.Windows.Shapes;  using System.Windows.Controls.Primitives;  using System.Windows.Threading;  namespace WpfApplication1  {  /// <summary>  /// MainWindow.xaml에 대한 상호 작용 논리  /// </summary>  class Empty : System.Windows.FrameworkElement  {  }  public partial class MainWindow : Window  {  const int NumberRows = 4; //열  const int NumberCols = 4;//행    int xEmpty, yEmpty, iCounter;  Key[] keys = { Key.Left, Key.Right, Key.Up, Key.Down };  Random rand;  UIElement elEmptySpare = new Empty();  public MainWindow()  {  InitializeComponent();  unigrid.Children.Add(new Empty());//제일 마지막 부분에 하얀칸 삽입  xEmpty = NumberCols - 1;//마지막부분 위치 초기화부분  yEmpty = NumberRows - 1; //마지막부분 위치 초기화부분  }  void TimerOnTick(object sender, EventArgs args)  {  for (int i = 0; i < 5; i++)  {  MoveTile(xEmpty, yEmpty + rand.Next(3) - 1);  MoveTile(xEmpty + rand.Next(3) - 1, yEmpty);  }  if (0 == iCounter--)  (sender as DispatcherTimer).Stop();  }  void MoveTile(int xTile, int yTile)//빈구역으로 타일을 밀어주는 코드  {  if ((xTile == xEmpty && yTile == yEmpty) || xTile < 0 || xTile >= NumberCols || yTile < 0 || yTile >= NumberRows)  return;  int iTile = NumberCols \* yTile + xTile;  int iEmpty = NumberCols \* yEmpty + xEmpty;  UIElement elTile = unigrid.Children[iTile];  UIElement elEmpty = unigrid.Children[iEmpty];  unigrid.Children.RemoveAt(iTile);  unigrid.Children.Insert(iTile, elEmptySpare);  unigrid.Children.RemoveAt(iEmpty);  unigrid.Children.Insert(iEmpty, elTile);  xEmpty = xTile;  yEmpty = yTile;  elEmptySpare = elEmpty;  }  private void unigrid\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)//마우스 왼쪽 클릭시 정보를  { //받아와서 무브 코드로 보냄  Grid tile = sender as Grid;  int iMove = unigrid.Children.IndexOf(tile);  int xMove = iMove % NumberCols;  int yMove = iMove / NumberCols;  if (xMove == xEmpty)  while (yMove != yEmpty)  MoveTile(xMove, yEmpty + (yMove - yEmpty) / Math.Abs(yMove - yEmpty));  if (yMove == yEmpty)  while (xMove != xEmpty)  MoveTile(xEmpty + (xMove - xEmpty) / Math.Abs(xMove - xEmpty), yMove);  }  private void Button\_Click\_1(object sender, RoutedEventArgs e)  {  rand = new Random();  iCounter = 16 \* NumberCols \* NumberRows;  DispatcherTimer tmr = new DispatcherTimer();  tmr.Interval = TimeSpan.FromMilliseconds(10);  tmr.Tick += TimerOnTick;  tmr.Start();  }  private void Window\_KeyDown\_1(object sender, KeyEventArgs e)  {  base.OnKeyDown(e);  switch (e.Key)  {  case Key.Right: MoveTile(xEmpty - 1, yEmpty); break;  case Key.Left: MoveTile(xEmpty + 1, yEmpty); break;  case Key.Down: MoveTile(xEmpty, yEmpty - 1); break;  case Key.Up: MoveTile(xEmpty, yEmpty + 1); break;  }  }  }  } |

표2.Xaml.cs

표2는 Xaml.cs 로 이벤트를 처리하는 곳이다.

예제에 나와있는 소스를 가져와서 쓴 것이 많기에 바뀐 부분만 설명하겠다.

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

{

Grid tile = sender as Grid;

int iMove = unigrid.Children.IndexOf(tile);

int xMove = iMove % NumberCols;

int yMove = iMove / NumberCols;

if (xMove == xEmpty)

while (yMove != yEmpty)

MoveTile(xMove, yEmpty + (yMove - yEmpty) / Math.Abs(yMove - yEmpty));

if (yMove == yEmpty)

while (xMove != xEmpty)

MoveTile(xEmpty + (xMove - xEmpty) / Math.Abs(xMove - xEmpty), yMove);

}

이 소스는 처음에 Unigrid 형태에 맞춰서 클릭 되었을시 움직이도록 작동하는 소스이다.

하지만 polygon형태에는 택스트를 쓰기위해 grid형태로 바꾸었기에 타일을 그리드 형태로 해야만 타일 위치를 받아올 수 있다. 그리고 이 소스는 모든 타일 적용 시켜야 한다.

<Grid MouseLeftButtonDown="unigrid\_MouseLeftButtonDown">

<Polygon Points="0,0 0,40 40,40 40,0" Fill="DarkGray" Margin="0,0,0,0" />

<TextBlock VerticalAlignment="Center" HorizontalAlignment="Center" Text="15"></TextBlock>

</Grid>

그래서 Grid 부분에 버튼 다운을 적용 시켜야 한다.

그리고 키다운은 윈도우 창 속성에서 적용시켜줘야 적용이 된다.

아래 그림처럼 윈도우 제일 바깥 창에 만들어야 한다.

