XAML Do-It-Yourself シリーズ

第 12 回 3D グラフィックス

Microsoft®

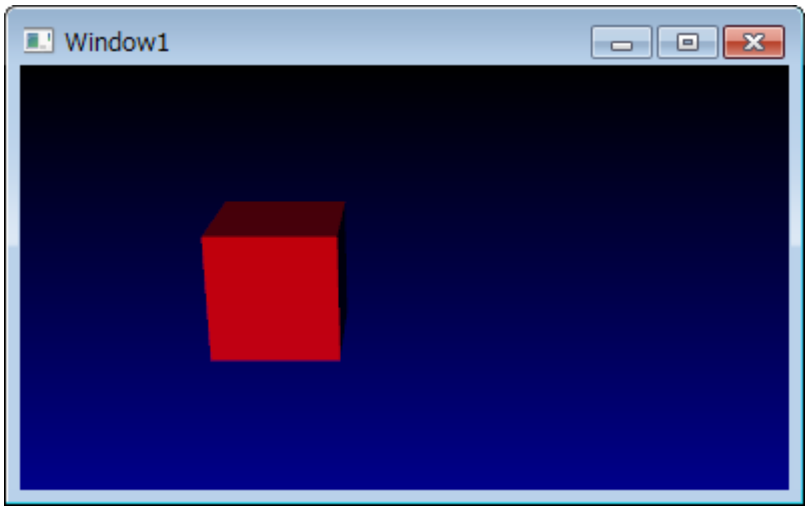
XAML Do-It-Yourself
第 12 回 3D グラフィックス

XAML Do-It-Yourself 第 12 回は、3D グラフィックスについて学習します。これまでアプリケーションで 3D グラフィックスを扱うには、DirectX のコンポーネントを使用する必要がありましたが、WPF (XAML) では 3D グラフィックスを標準でサポートしています。

今回は、次の 2 つを中心に学習します。

・3D グラフィックスの基礎
・3D グラフィックスのアニメーション

3D グラフィックスを利用して作成したアプリケーションの実行例を示します。このアプリケーションでは、赤い立方体が画面内を回転します。



■3D グラフィックスの基本
XAML および WPF アプリケーションでは 3D グラフィックスを扱うことが可能ですが、それにはいくつかの 3D グラフィックス特有の知識が必要になります。しかし以降ではそれらにあまり深入りせず、XAML で 3D グラフィックスを作成するために最低限必要な要素を解説していきます。

●WPF の 3D グラフィックスは右手座標系
XAML で記述する 3D グラフィックスの座標系は右手座標系 (親指が X 軸、人差し指が Y 軸、中指が Z 軸)と呼ばれるもので、X-Y 座標平面を基準にすると、Z 軸は X-Y 平面から手前方向に向かって正の値をとります。

●ビューポート (Viewport3D クラス)
3D グラフィックスでは、3D 空間に物体を配置し、それを 2 次元の平面であるスクリーンに投影した内

容を描画します。このスクリーンに相当するのが Viewport3D です。Viewport3D には、カメラ (Camera) や光源 (Light)、空間に配置する 3D モデル (MeshGeometry など) が含まれます。Viewport3D は、これまで紹介した ListBox や TextBox といったコントロールと同様に記述できるので、簡単に 3D グラフィックスを WPF アプリケーションに組み込むことができます。

次の XAML コードでは、Grid 上に 3D グラフィックスを表示する場合の記述例です。パネル上に配置した <Viewport3D> 要素が 3D グラフィックスの描画領域となります。

```
<Window x:Class="WpfApplication12.MainWindow"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="MainWindow" Height="350" Width="450">
  <Grid>
    <Viewport3D>
      <!-- 3D グラフィックスの内容 -->
    </Viewport3D>
  </Grid>
</Window>
```

●カメラ (Camera)

Camera は 3D 空間に配置した物体をスクリーンに映し出す働きをします。Camera 自身も 3D 空間内に位置し、カメラの位置や視野角などが設定できます。

次の例では、 (x,y,z) = (0,0,10) の位置から、水平視野 (FieldOfView) が 30 度のカメラを設定しています。

```
<Viewport3D.Camera>
  <PerspectiveCamera Position="0, 0, 10" FieldOfView="30" />
</Viewport3D.Camera>
```

●光源 (Light)

3D 空間に物体を配置しても、光源 (Light) がなければ何もスクリーンに投影されません。3D 空間には光源が必須です。カメラと同様に、光源も 3D 空間内に配置します。

次の例では、DirectionalLight という種類の光源を用い、光源の色 (Color) を "White" に、光源が照らす方向 (Direction 属性) を (0,0,-1) の方向に設定しています。ただし DirectionalLight は太陽光のような平行な光を表しますので、光源の位置を指定する必要はありません。

```
<ModelVisual3D>
  <ModelVisual3D.Content>
    <DirectionalLight Color="White" Direction="0,0,-1" />
  </ModelVisual3D.Content>
</ModelVisual3D>
```

光源は、DirectionalLight のほかにも、空間全体に照らす AmbientLight、特定の領域を照らす、いわゆ

るスポットライトの効果を持つ SpotLight などが利用できます。

●3D モデル

WPF による 3D グラフィックスでは、すべての物体は三角形を組み合わせて定義する必要があります。球や円錐といった一般的な形状は用意されていません。

例えば X-Y 平面上にある厚さのない (平面の) 正方形の物体を定義するにしても、2 つの三角形を組み合わせる必要があります。これは、<MeshGeometry3D> 要素を使って次のように記述します。

```
<GeometryModel3D.Geometry>
  <MeshGeometry3D
    Positions="-1 1 0, -1 -1 0, 1 -1 0, 1 1 0"
    TriangleIndices="0 1 2, 0 2 3" />
</GeometryModel3D.Geometry>
```

Positions 属性では、物体として定義する図形の頂点の座標を並べて記述します。さらに TriangleIndices 属性を使って、Position 属性で用意した頂点をどのように結んで三角形を形成するかを示します。TriangleIndices 属性では、頂点を指定するために Positions 属性で指定した座標のインデックス (0 始まり) を指定します。

上記のリストでは、TriangleIndices 属性で記述している “0 2 1“ が 1 つの三角形を示し、その頂点の座標は、(-1,1,0)、(1,-1,0)、(-1,-1,0) です。このような三角形を多数記述することにより、3D 空間に物体を定義していきます。

また、通常は物体の表面には色が付いていますが、これを指定するには<Geometry3D.Material> 要素を使用します。

```
<GeometryModel3D.Material>
  <DiffuseMaterial Brush="Blue" />
</GeometryModel3D.Material>
```

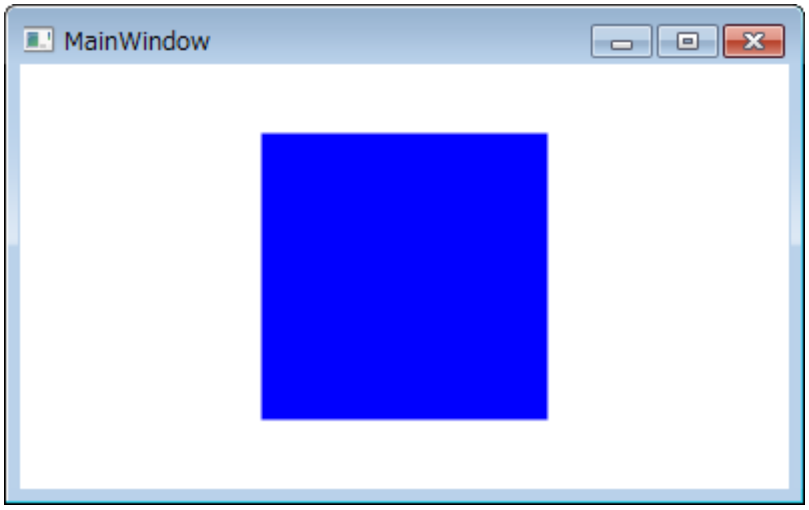
<DiffuseMaterial> 要素は、(鏡面とは逆の) つや消しのような、すべての方向に均一に光が拡散する表面を表します。表面の色には "Blue" のブラシ (Brush) を使用します。ここでは行いませんが、画像を使った表面の修飾も可能です。

ここまでに説明した、Viewport3D、Camera、Light、3D モデルをまとめると、次のような XAML コードになります。

```
<Window x:Class="WpfApplication12.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="250" Width="400">
  <Grid>
    <Viewport3D>
```

```
    <!-- カメラ -->
    <Viewport3D.Camera>
      <PerspectiveCamera Position="0, 0, 10"  FieldOfView="30" />
    </Viewport3D.Camera>
    <!-- 光源 -->
    <ModelVisual3D>
      <ModelVisual3D.Content>
        <DirectionalLight Color="White" Direction="0,0,-1" />
      </ModelVisual3D.Content>
    </ModelVisual3D>
    <ModelVisual3D>
      <!-- 3D モデル -->
      <ModelVisual3D.Content>
        <GeometryModel3D>
          <!-- 正方形 -->
          <GeometryModel3D.Geometry>
            <MeshGeometry3D
              Positions="-1 1 0, -1 -1 0, 1 -1 0, 1 1 0"
              TriangleIndices="0 1 2, 0 2 3"/>
          </GeometryModel3D.Geometry>
          <!-- 表面の色 -->
          <GeometryModel3D.Material>
            <DiffuseMaterial Brush="Blue" />
          </GeometryModel3D.Material>
        </GeometryModel3D>
      </ModelVisual3D.Content>
    </ModelVisual3D>
  </Viewport3D>
 </Grid>
</Window>
```

これを実行すると、次のようなウィンドウが表示されます。これだけでは Rectangle を使って 2D グラフィックスの四角形を描画しているのと見た目は変わりませんが、物体の位置やカメラの設定を変更することで、描画内容が変化するのを確認できます。



●立方体を表示する

これまでの説明で、3D グラフィックスを作成するために必要な最小限の手順を学習できました。次に、より 3D らしく見えるオブジェクトとして、立方体を表示してみます。

正方形を定義するには基本図形である三角形が 2 個必要でしたので、立方体を作成するには 12 個の三角形が必要になります。立方体を定義する <MeshGeomety3D> 要素の記述は次のようになります。6 個の正方形を定義する Positions 属性と、12 個の三角形を定義する TriangleIndices 属性を記述しています。

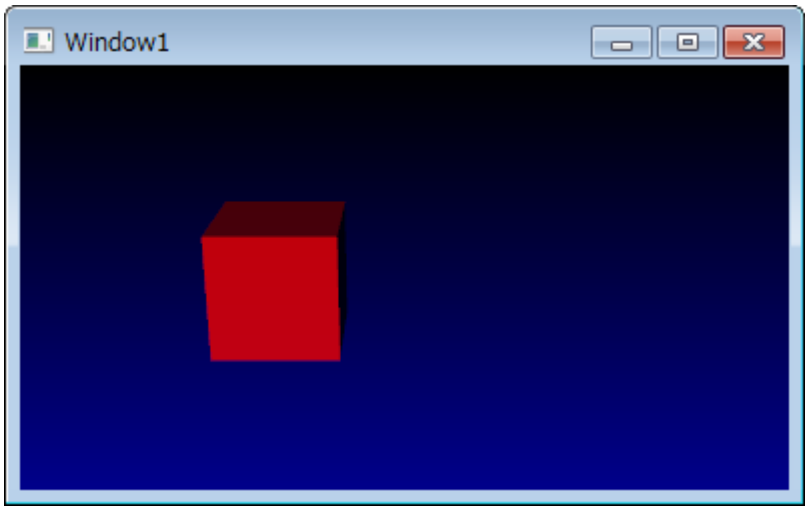
```
<MeshGeometry3D
  Positions="1 1 -1, 1 -1 -1, -1 -1 -1, -1 1 -1,
                1 1 1, -1 1 1, -1 -1 1, 1 -1 1,
                1 1 -1, 1 1 1, 1 -1 1, 1 -1 -1,
                1 -1 -1, 1 -1 1, -1 -1 1, -1 -1 -1,
                -1 -1 -1, -1 -1 1, -1 1 1, -1 1 -1,
                1 1 1, 1 1 -1, -1 1 -1, -1 1 1"
  TriangleIndices="0 1 2, 0 2 3, 4 5 6, 4 6 7,
    8 9 10, 8 10 11, 12 13 14, 12 14 15,
    16 17 18, 16 18 19, 20 21 22, 20 22 23"
/>
```

この立方体を表示する XAML の記述例を示します。ここでは背景やカメラの方向なども設定しています。

```
<Window x:Class="WpfApplication12.Window1"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="Window1" Height="250" Width="400">
  <Grid>
    <!-- 背景 -->
    <Grid.Background>
      <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">
        <LinearGradientBrush.GradientStops>
          <GradientStop Color="Black" Offset="0"/>
          <GradientStop Color="DarkBlue" Offset="1"/>
        </LinearGradientBrush.GradientStops>
      </LinearGradientBrush>
    </Grid.Background>
    <Viewport3D>
      <Viewport3D.Camera>
        <PerspectiveCamera Position="0,0,-15" UpDirection="0,1,0"
            LookDirection="0,0,1" FieldOfView="45" NearPlaneDistance="0.125"/>
      </Viewport3D.Camera>
      <Viewport3D.Children>
        <!-- 光源 -->
        <ModelVisual3D>
          <ModelVisual3D.Content>
            <DirectionalLight Color="White" Direction="0,0,1" />
          </ModelVisual3D.Content>
        </ModelVisual3D>
        <!-- 3D モデル -->
```

```
      <ModelVisual3D>
        <!-- アニメーション用の座標変換 -->
        <ModelVisual3D.Transform>
          <RotateTransform3D>
            <RotateTransform3D.Rotation>
              <AxisAngleRotation3D x:Name="rotation" Angle="0" Axis="0,1,0" />
            </RotateTransform3D.Rotation>
          </RotateTransform3D>
        </ModelVisual3D.Transform>
        <!-- 立方体 -->
        <ModelVisual3D.Content>
          <GeometryModel3D>
            <GeometryModel3D.Geometry>
              <MeshGeometry3D
                 Positions="1 1 -1, 1 -1 -1, -1 -1 -1, -1 1 -1,
                   1 1 1, -1 1 1, -1 -1 1, 1 -1 1,
                   1 1 -1, 1 1 1, 1 -1 1, 1 -1 -1,
                   1 -1 -1, 1 -1 1, -1 -1 1, -1 -1 -1,
                   -1 -1 -1, -1 -1 1, -1 1 1, -1 1 -1,
                   1 1 1, 1 1 -1, -1 1 -1, -1 1 1"
                 TriangleIndices="0 1 2, 0 2 3, 4 5 6, 4 6 7,
                   8 9 10, 8 10 11, 12 13 14, 12 14 15,
                   16 17 18, 16 18 19, 20 21 22, 20 22 23"
                />
            </GeometryModel3D.Geometry>
            <GeometryModel3D.Material>
              <DiffuseMaterial>
                <DiffuseMaterial.Brush>
                  <SolidColorBrush Color="Red" Opacity="0.8"/>
                </DiffuseMaterial.Brush>
              </DiffuseMaterial>
            </GeometryModel3D.Material>
            <!-- Translate the plane. -->
            <GeometryModel3D.Transform>
              <Transform3DGroup>
                <Transform3DGroup.Children>
                  <TranslateTransform3D OffsetX="2" OffsetY="0" OffsetZ="0"   >
                  </TranslateTransform3D>
                  <RotateTransform3D>
                    <RotateTransform3D.Rotation>
                      <AxisAngleRotation3D  Angle="-20" Axis="1,0,0" />
                    </RotateTransform3D.Rotation>
                  </RotateTransform3D>
                </Transform3DGroup.Children>
              </Transform3DGroup>
            </GeometryModel3D.Transform>
          </GeometryModel3D>
        </ModelVisual3D.Content>
      </ModelVisual3D>
    </Viewport3D.Children>
  </Viewport3D>
 </Grid>
</Window>
```

これを実行すると、次のように立方体が表示されます。



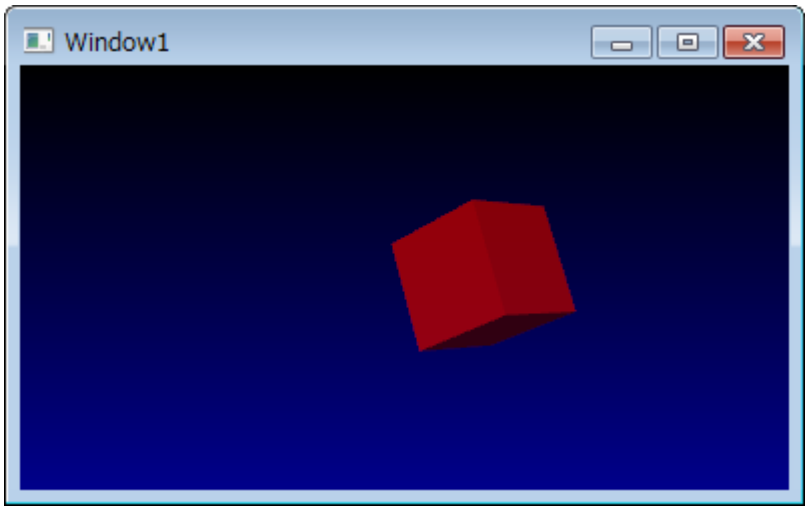
■アニメーション

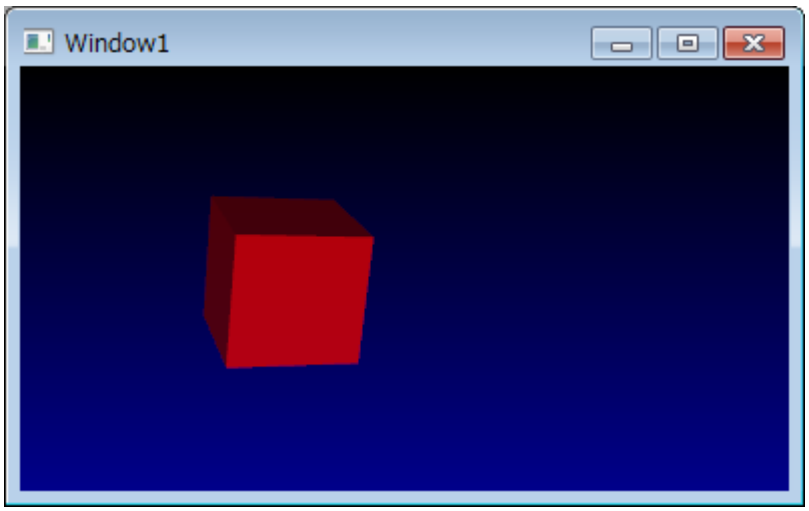
さらにアニメーションを行うことで、より 3D グラフィックスらしく見えるようにします。3D グラフィックスのアニメーションを実現するには、座標変換の機能を利用し、座標変換時に指定するパラメータをアニメーションにより連続的に変化させるのが簡単です。

以下では、アプリケーション ウィンドウがロードされた際のイベントをトリガーにして、立方体が Y 軸を基準に 5 秒間で 360 度回転するアニメーションの設定を記述しています。

```
<!-- 立方体を Y 軸を基準に回転 -->
<Window.Triggers>
  <EventTrigger RoutedEvent="Window.Loaded" >
    <BeginStoryboard>
      <Storyboard Name="myStoryBoard">
        <DoubleAnimation
          Storyboard.TargetName="rotation"
          Storyboard.TargetProperty="Angle"
          From="0" To="360" Duration="0:0:5" RepeatBehavior="Forever"/>
      </Storyboard>
    </BeginStoryboard>
  </EventTrigger>
</Window.Triggers>
```

スクリーンショットでは分かりにくいですが、アプリケーションを実行すると 3D 空間に浮かんだ立方体が飛び回ります。





■まとめ

今回は XAML における 3D グラフィックスの基礎について学習しました。WPF では 3D グラフィックスは特殊なものではありません。このため Image や TextBox といったほかの要素と組み合わせることもでき、これまでなかったような GUI の構築も可能です。