

## SilverLight 3D Cube Component Reference Guide

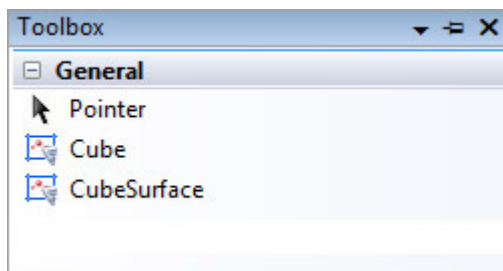
### 1. Using the Component :

There are 3 different ways to use the 3D Cube component.

#### 1. Adding the component to the toolbox :

You will have to right click with the mouse somewhere on the toolbox panel (where there's an empty space) and click on the "Choose Items" button. In the new window that will open you will click on the "Browse" and then select the "Element.dll" file that is located on the Bin directory inside the project.

You will get 2 new components in the ToolBox panel :



The CubeSurface component is a rotating cube shape without the option to Present images on the faces of the cube.

The Cube component is the full version that allow you to present images on The faces of the cube.

After you add it to the ToolBox, all you need to do it to drag it to the Xaml Editor and use it as a regular component.

#### 2. Adding the component to the Xaml editor :

You will have to add a reference to the "Elements.dll" file to the project references. After that you'll be able to use the component in the Xaml editor as follows :

```
<my:Cube x:Name="My_Cube_Name" ></my:Cube>
```

Of course you'll be able to set most of the properties from the Xaml editor.

#### 3. Using the component in code :

You will have to add a reference to the "Elements.dll" file to the project and add the "using Elements" statement in the top of the page. Then you'll be able to use it as a regular object :

```
Cube myCube = new Cube();
```

You'll be able to set the cube with its properties and methods. When you Finish, don't forget to add it the relevant canvas / grid.

## **2. Component Properties :**

<b>Property Name</b>	<b>Description</b>
AngleX	Gets / Sets the X rotation angle of the cube
AngleY	Gets / Sets the Y rotation angle of the cube
AngleZ	Gets / Sets the Z rotation angle of the cube
Width	Gets / Sets the Width of the cube as it appears on the screen
Height	Gets / Sets the Height of the cube as it appears on the screen
Depth	Gets / Sets the Depth of the cube as it appears on the screen
ActualWidth	Gets / Sets the Actual Width according to the input images
ActualHeight	Gets / Sets the Actual Height according to the input images
ActualDepth	Gets / Sets the Actual Depth according to the input images
CanvasWidth	Gets /Sets the Width of the canvas that wrapping the cube
CanvasHeight	Gets /Sets the Height of the canvas that wrapping the cube
CanvasBackground	Gets /Sets the Background of the canvas that wrapping the cube Any brush like SolidColor, Gradient, Image, Movie can be uses for painting the canvas background
CanvasOpacity	Gets /Sets the Opacity of the canvas that wrapping the cube
FrontImage	Gets / Sets the URL of the Front image of the cube
BackImage	Gets / Sets the URL of the Back image of the cube
LeftImage	Gets / Sets the URL of the Left image of the cube
RightImage	Gets / Sets the URL of the Right image of the cube
TopImage	Gets / Sets the URL of the Top image of the cube
BottomImage	Gets / Sets the URL of the Bottom image of the cube

## **3. Component Methods :**

<b>Methods Name</b>	<b>Description</b>
RotateCube	Rotates the cube to a specific angle. Int x – the x coordination Int y – the y coordination Int z – the z coordination

Methods Name	Description
SetFacesBackColor	<p>Sets the back color of the faces of the cube.</p> <p><u>Version 1</u> :</p> <p>Color default_color – same color to all the faces</p> <p><u>Version 2</u> :</p> <p>Color front_color – the color of the front face</p> <p>Color back_color – the color of the back face</p> <p>Color left_color – the color of the left face</p> <p>Color right_color – the color of the right face</p> <p>Color bottom_color – the color of the bottom face</p> <p>Color top_color – the color of the top face</p>
SetCubeDimensions	<p>Set the dimensions of the cube.</p> <p>_width – the width of the cube</p> <p>_height – the height of the cube</p> <p>_depth – the depth of the cube</p>
SetFacesImages	<p>Set the images urls to the faces of the cube.</p> <p>String frontImg – url of the front face image</p> <p>String backImg – url of the back face image</p> <p>String leftImg – url of the left face image</p> <p>String rightimg – url of the right face image</p> <p>String bottomImg – url of the bottom face image</p> <p>String topImg – url of the top face image</p> <p>Int actualWidth – the actual cube width according to the images</p> <p>Int actualHeight – the actual cube height according to the images</p> <p>Int actualDepth – the actual cube depth according to the images</p>

## **2. Preparing images for the cube :**

It is worth to take the time and prepare the images correctly in order to make sure that they will appear perfect on the cube faces.

The main issue with the cube algorithm is that the input images must be with a square shape. We would like to have the tile operation to take place automatically when needed, but unfortunately Silverlight doesn't support any image processing tools.

This means that when is necessary, you will have to prepare the images yourself (Or you can send Microsoft angry email and demand them to add image processing abilities to Silverlight).

If your cube is actually a cube (means all faces are in the same equal size) you won't have to do nothing with images, just make them in square shapes in the desired size.

If your cube is more like a chest, you have to do some tiling job. Just take all the 6 images of the chest and tile them to a square shape according to the bigger side of the image.

For example :

This image is in size of 200px width and 95px height.



You will have to tile it to size of 200px width and 200px height.



Do the same for all 6 images of the cube, each image according to the bigger size.

Note : when you set the ActualWidth / ActualHeight / ActualDepth properties in code, make sure you enter the original sizes of the cube and the sizes after you process them.