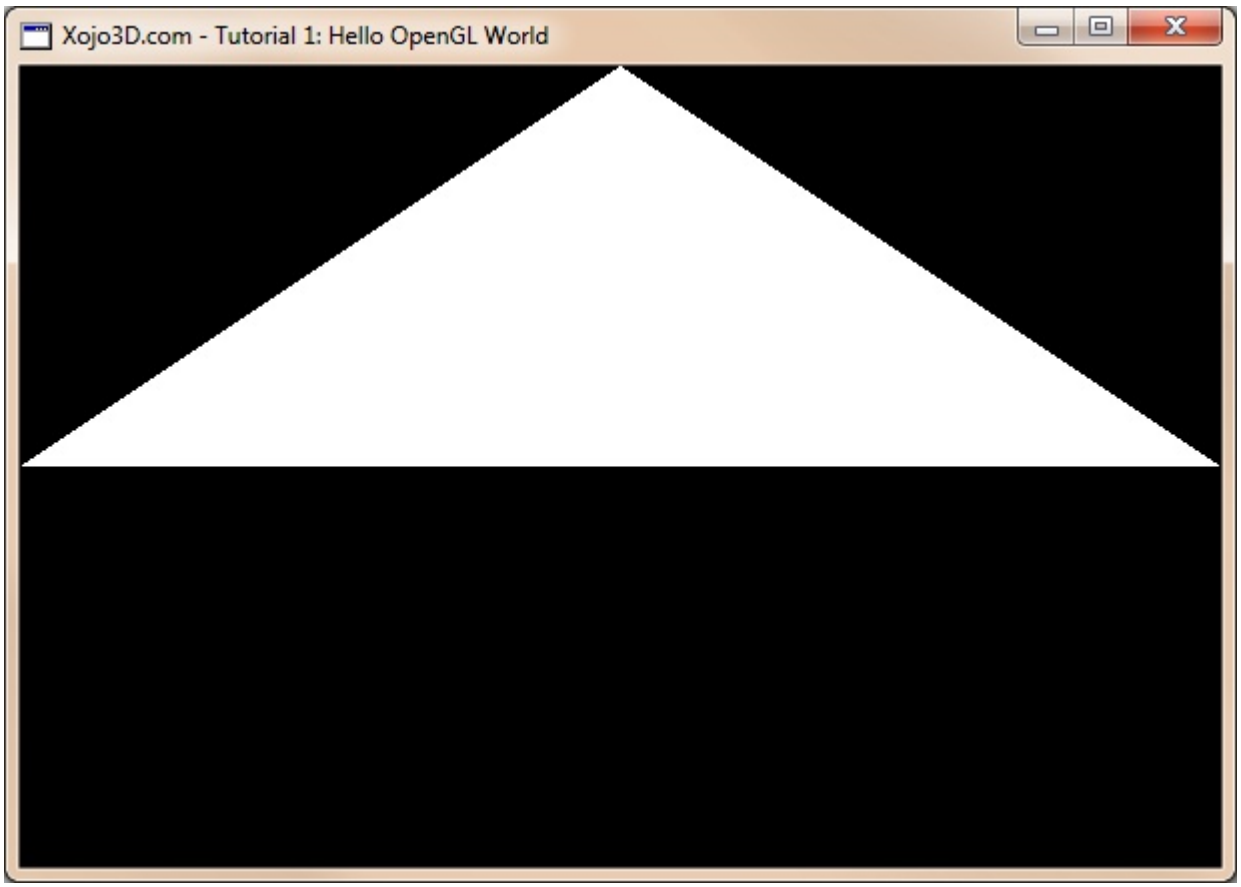
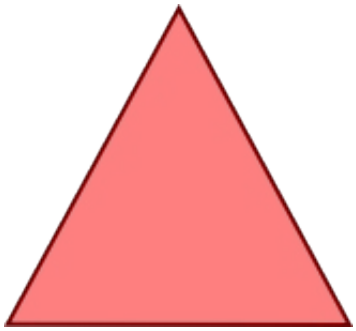
Tutorial 1: Hello OpenGL world

Write your first 3D program in Xojo. This tutorial introduces you to polygons and shows you how to draw them with OpenGL.

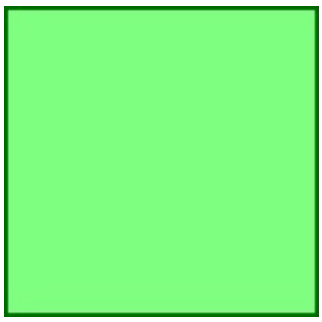


Theory

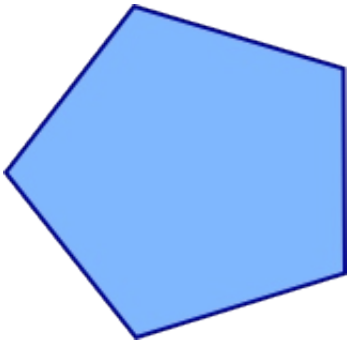
The main building blocks of 3D models are **polygons**. A polygon is a collection of three or more edges, connected in such a way to form a closed path.



3-sided polygon

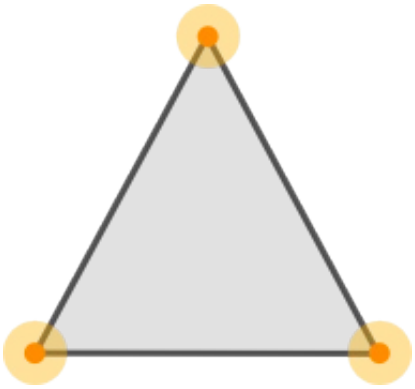


4-sided polygon

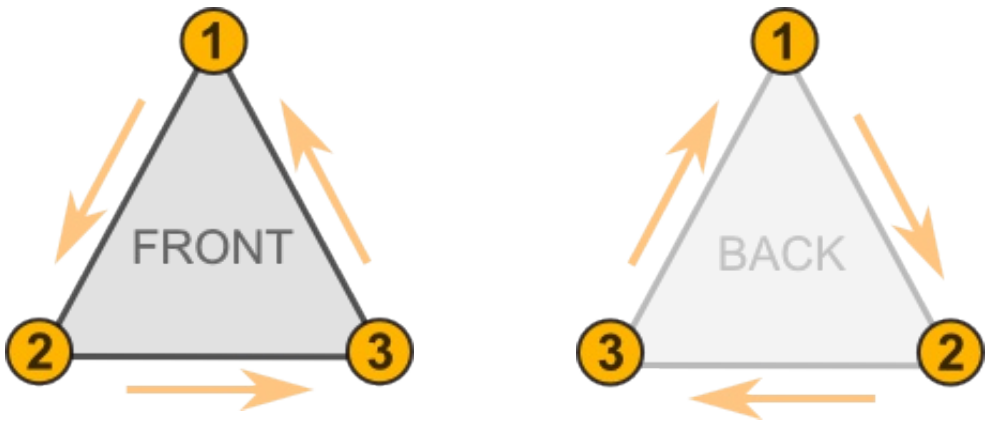


5-sided polygon

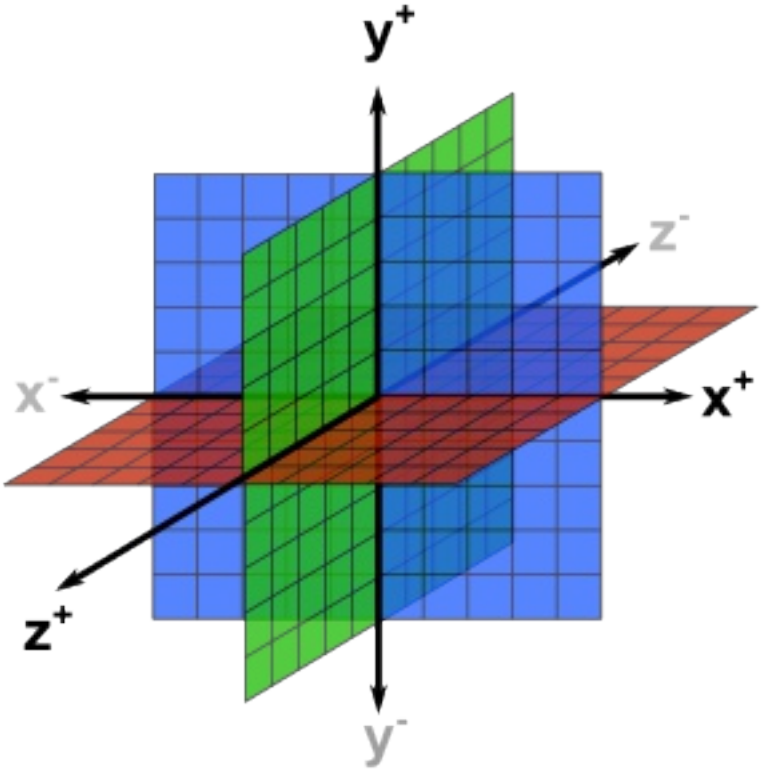
The points where the edges of a polygon meet are known as **vertices**. The image below illustrates the vertices of a triangular polygon.



Polygons rotate freely in 3D space (e.g., they can face towards you or face away from you). We therefore need to define the vertices in such a way that the computer knows which way the polygon is facing. The vertices of a polygon are always defined in an **anti-clockwise** order.



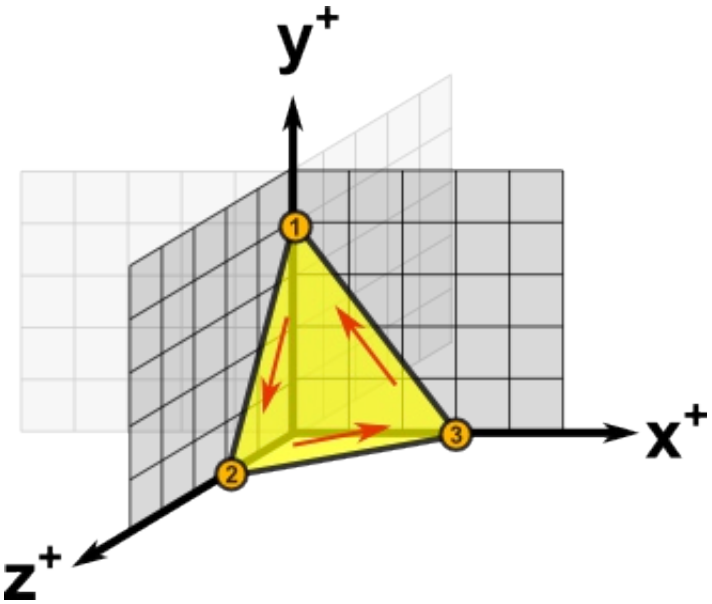
A vertex consists of three components known as **x**, **y** and **z**. These values indicate where in 3D-space the vertex is located. The following image gives a logical view of 3D-space.



Take note of the following properties of 3D-space:

- x increases towards the right.
- x decreases towards the left.
- y increases towards the top.
- y decreases towards the bottom.
- z increases towards you.
- z decreases away from you
- The XY-plane is shown in blue.
- The XZ-plane is shown in red.
- The YZ-plane is shown in green.

A polygon with the vertices (0,4,0), (0,0,2) and (3,0,0) is shown below.



Vertices are given as (x,y,z) coordinates in an anti-clockwise order.

- Vertex 1: (x=0, y=4, z=0)
- Vertex 2: (x=0, y=0, z=2)
- Vertex 3: (x=3, y=0, z=0)

Tutorial Steps

1. Open Xojo.
2. In the Project Chooser select Desktop.
3. Enter "Tutorial001" as the Application Name, and click OK.
4. Save your project.
5. Configure the following controls:

Control	Name	DoubleBuffer	Left	Top	Maximize Button
Window	SurfaceWindow	-	-	-	ON
OpenGLSurface	Surface	ON	0	0	-

6. Position and size *Surface* to fill the whole window, and set its locking to left, top, bottom and right.

7. Add the following code to the *SurfaceWindow.Open* event handler:

```
Self.MouseCursor = System.Cursors.StandardPointer
```

8. Add the following code to the *SurfaceWindow.Paint* event handler:

```
Surface.Render
```

9. Add the following code to the *Surface.Render* event handler:

```
OpenGL.glClearColor(0, 0, 0, 1)
OpenGL.glClear(OpenGL.GL_COLOR_BUFFER_BIT)

OpenGL.glBegin OpenGL.GL_TRIANGLES

OpenGL.glVertex3d 0, 1, 1
OpenGL.glVertex3d -1, 0, 1
OpenGL.glVertex3d 1, 0, 1

OpenGL.glEnd
```

10. Save and run your project.

Analysis

SurfaceWindow.Open:

```
Self.MouseCursor = System.Cursors.StandardPointer
```

While OpenGL is loading, the mouse cursor is shown as a busy icon. The instruction above ensures that the normal mouse cursor is displayed when OpenGL is done loading.

SurfaceWindow.Paint:

```
Surface.Render
```

Surface.Render instructs the OpenGL surface to render itself.

Surface.Render:

```
OpenGL.glClearColor(0, 0, 0, 1)
OpenGL.glClear(OpenGL.GL_COLOR_BUFFER_BIT)

OpenGL.glBegin OpenGL.GL_TRIANGLES

OpenGL.glVertex3d 0, 1, 1
OpenGL.glVertex3d -1, 0, 1
OpenGL.glVertex3d 1, 0, 1

OpenGL.glEnd
```

It is in the Surface.Render event handler where we do our drawing.

glClearColor sets the color to use for the background. The four parameters, red, green, blue and alpha, makes up an RGBA color value. These values are floating point values in the range of 0 to 1. All the values are mixed together to determine the final color.

The call to the glClear function clears the background with the selected color. The GL_COLOR_BUFFER_BIT constant indicates to OpenGL that we want to clear the background.

glBegin is the start of our drawing routine. We indicated that we will be drawing triangular polygons by passing the GL_TRIANGLES constant.

We supply OpenGL with the polygon data by passing the x, y and z coordinates of each vertex to glVertex3d. It is important to note that the vertices of a polygon are always given in an anti-clockwise direction. OpenGL uses the order of the vertices to determine if the polygon is facing towards us or away from us.

Finally we indicate to OpenGL that we have supplied all the vertex data of the polygons by calling the glEnd function.