# Lab Week 1.1 : Geometry

## Requirements for Drawing Primitives

1. Choose the suitable Vertex Type
   1. Color
   2. Texture
   3. ColorTexture
   4. TextureNormal
2. Create an array to store the vertices
3. Declare the vertices in local space and add them to the array in a clockwise order
4. Create an effect (BasicEffect) to draw the vertices
5. Create a View and Projection Matrix for use with the effect to transform the vertices when

Drawn

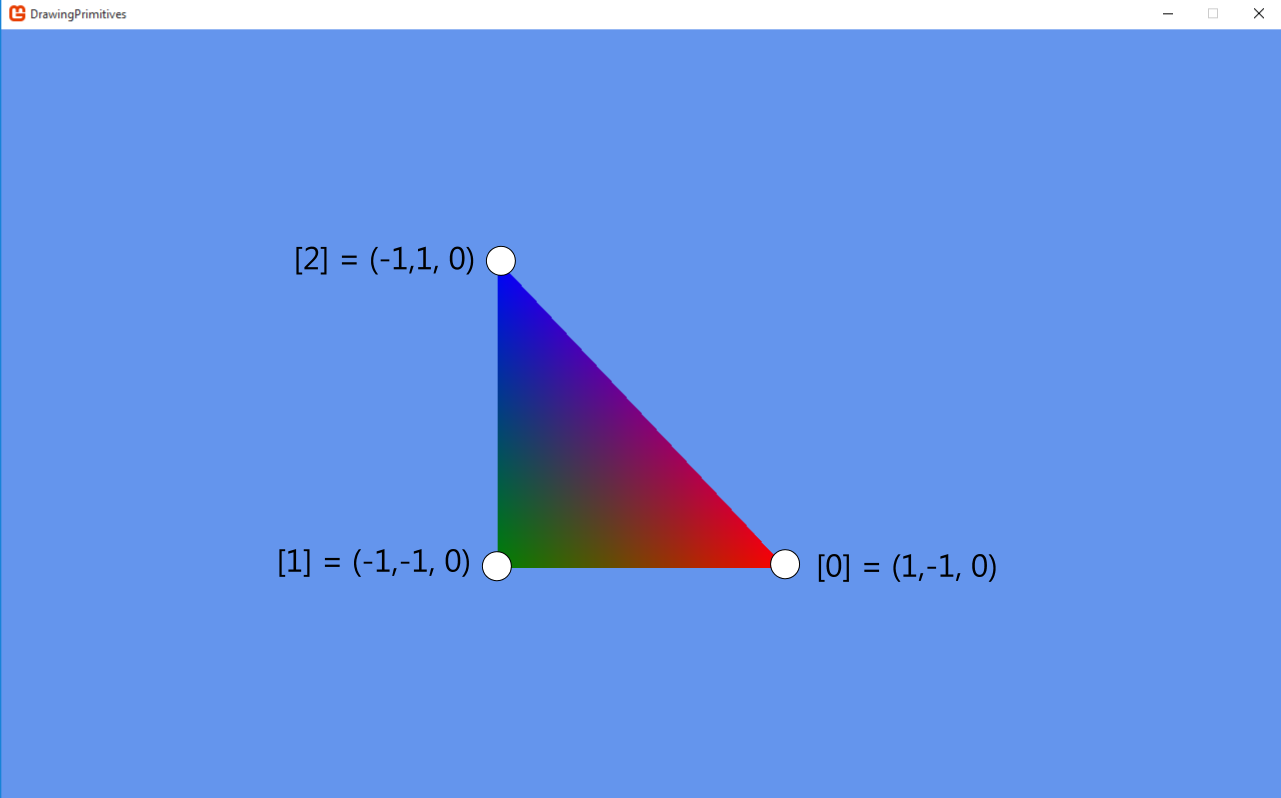
## Setup

1. Create a new MonoGame Windows Project called Week1ManualGeometry

# Follow Along : Vertex Position Color

* Follow along in class to see how to
  + Create and fill an array of VertexPositionColor
  + Create an instance of BasicEffect
  + Setup the View and Projection matrices for our camera
  + Draw our vertex array using GraphicsDevice.DrawUserPrimitives

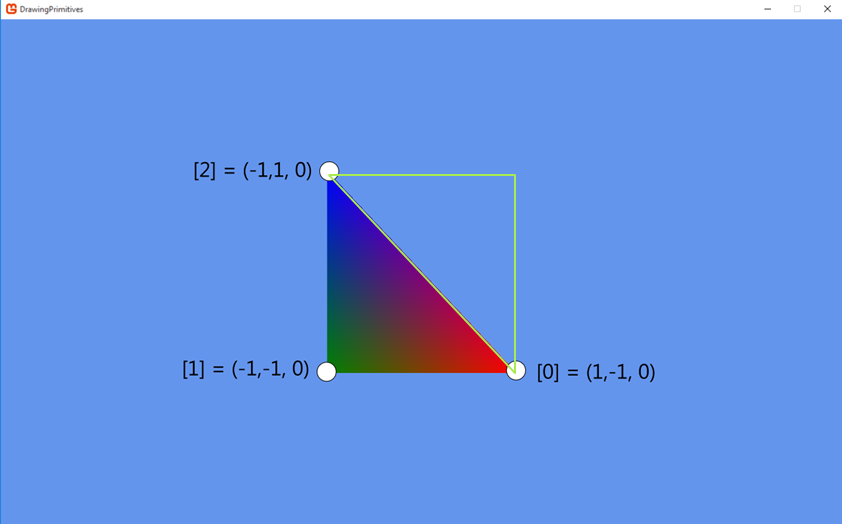
## Expected Result



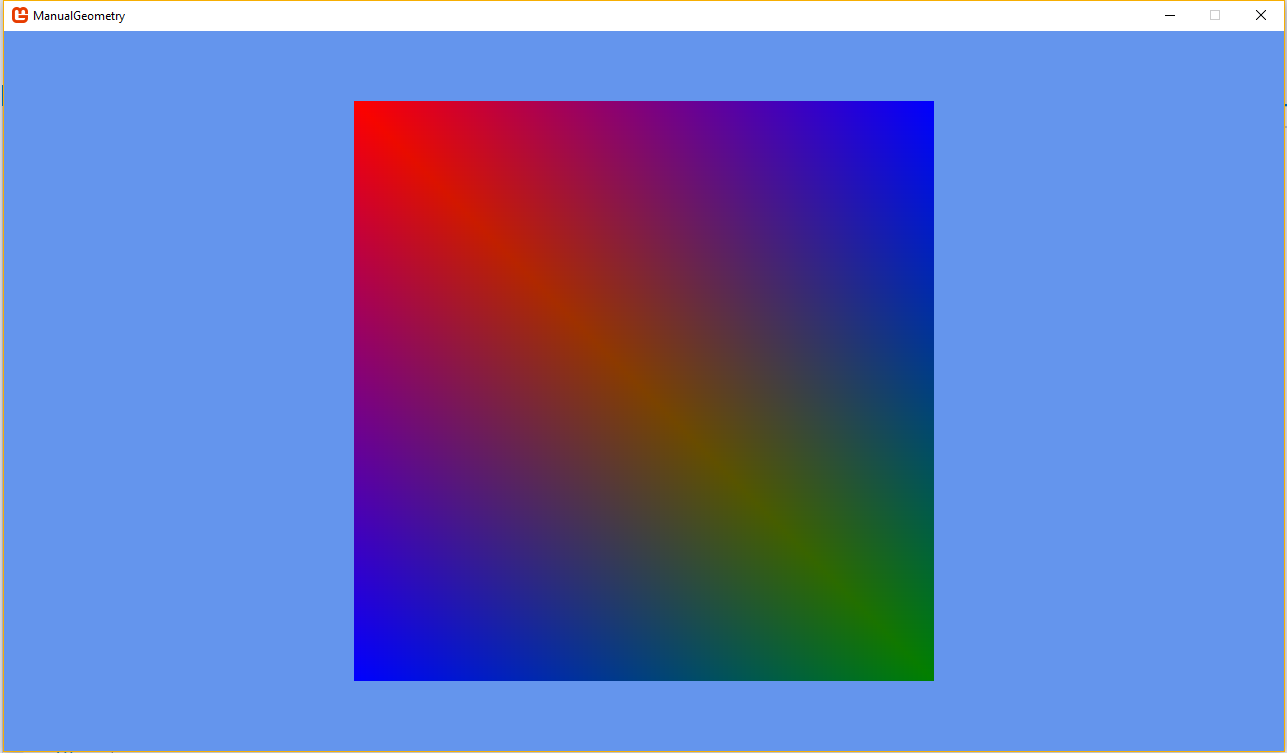
# Exercise 1 : Drawing a Square

Building on the previous follow along:

* Increase the size of the vertex array to 6
* Add three new vertices to the array to create a second triangle



## Expected Result



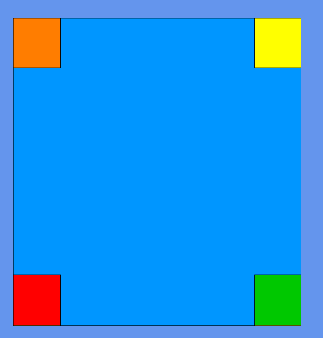
# Exercise 2 : Vertex Position Texture

* Textured vertices are only slightly differnet to color ones
  + We now use the VertexPositionTexture vertex type (stored in an array)
  + Need a texture to be applied to our vertices
  + BasicEffect with TextureEnabled set to true
  + When you access the properties of the vertex you will see we still have a Position property but the color one has been replaced with a Vector2 TextureCoordinate



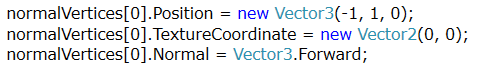
* Create a new method Called CreateTexturedVertices
* Like the previous exercise we want two triangles to create a square
* Create a new method called DrawTexturedVertices
  + Update GraphicsDevice.DrawUserPrimitives to now use VertexPositionTexture and the array textured vertices

## Expected Results



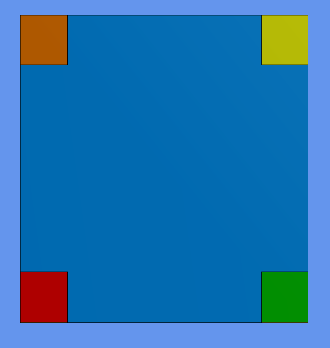
# Exercise 3 : Vertex Position Normal Texture

* For the final Vertex type we are now adding a Normal to the vertex
* A normal defines the direction in which the vertex is pointing
  + Later used in lighting calculations to determine the amount of light the vertex will receive and how that light should reflected etc.
* Texture is still required for this vertex type
  + Reuse the one from the previous exercise



* Create a new method Called CreateNormalVertices
* Like the previous exercise we want two triangles to create a square
* Create a new method called DrawTexturedVertices
  + Update GraphicsDevice.DrawUserPrimitives to now use VertexPositionTexture and the array textured vertices

## Expected Results



# Follow Along : World Transformations

* So far all out vertices have been declared in local space
* If we want to move the object somewhere else in the game world we need to transform there vertices by a matrix
  + Known as a world matrix
* Lets create a world matrix and use it to:
  + Move our vertices to another position
  + Rotate our vertices on the Y axis
  + Scale our vertices to bigger or smaller