**Design Description:**

My program was designed to simulate the journey through my idea of what a science fiction wormhole looks like. The code is written in C# and uses the OpenTK framework.

To implement this, I create a queue (of fixed size) of torus segments, which I generate and stitch together procedurally (I was especially happy with this) each frame. Once I populate the queue with stitched torus segments, each time a new segment is placed I remove the first segment in the queue. This means there is not a risk of memory overflow within the program, as each frame the GPU is doing the same amount of work and the memory usage stays the same.

To allow the view matrix to display the current progress through the tunnel, I choose two coordinates: the midpoint of an early torus segment, and the midpoint of a torus segment from later on. I then use the LookAt function from OpenGL to create the view matrix with these two arguments.

**Problems:**

The main problem that I had was that it was difficult to debug this program, as some of the matrix multiplication was non-trivial and required some thought. The algorithm I wrote to attach one torus segment to another moves the new torus segment in terms of the old torus segment’s local space, so it required quite a bit of trial and error to properly implement. I think I overcame this quite well by getting out a pencil and paper and working it out using a diagram. Using libraries with lots of deprecated code floating about on the web (such as ImageSharp) was a hard thing to overcome too- but made me more comfortable overall with inventing my own solutions or combining a few which already exist.

**Extra Accomplishments:**

The main things that I accomplished which were not specifically covered in the lectures were: Fresnel shading, using a cube-map for a skybox and tunnel generation using torus segments.

I used Fresnel shading to calculate and model more effectively the amount of light transmitted through my transparent wormhole to the part behind it, as the wormhole is transparent. I used the equation from Nvidia’s Cg tutorial and played around with the variables to accomplish this task *(Equation 7-3 An Approximation of the Fresnel Equation):*

 <http://developer.download.nvidia.com/CgTutorial/cg_tutorial_chapter07.html>

Using a cube-map for a skybox was difficult as I had to use a C# OpenTK specific image loader (ImageSharp) whose documentation left a lot to be desired. I was able to overcome this however with some fiddling around and loaded in a skybox I created using the 3rd party software Spacescape (creating royalty-free skyboxes). I also used an extra shader to accommodate this.

