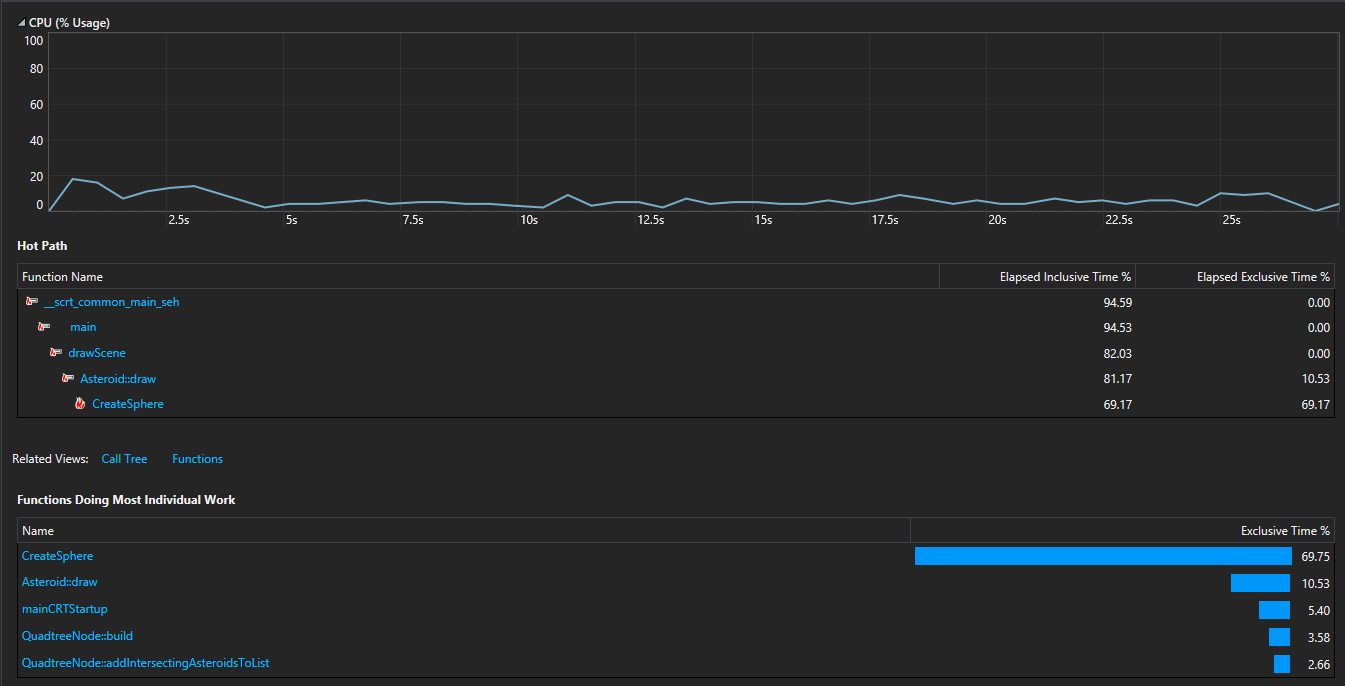
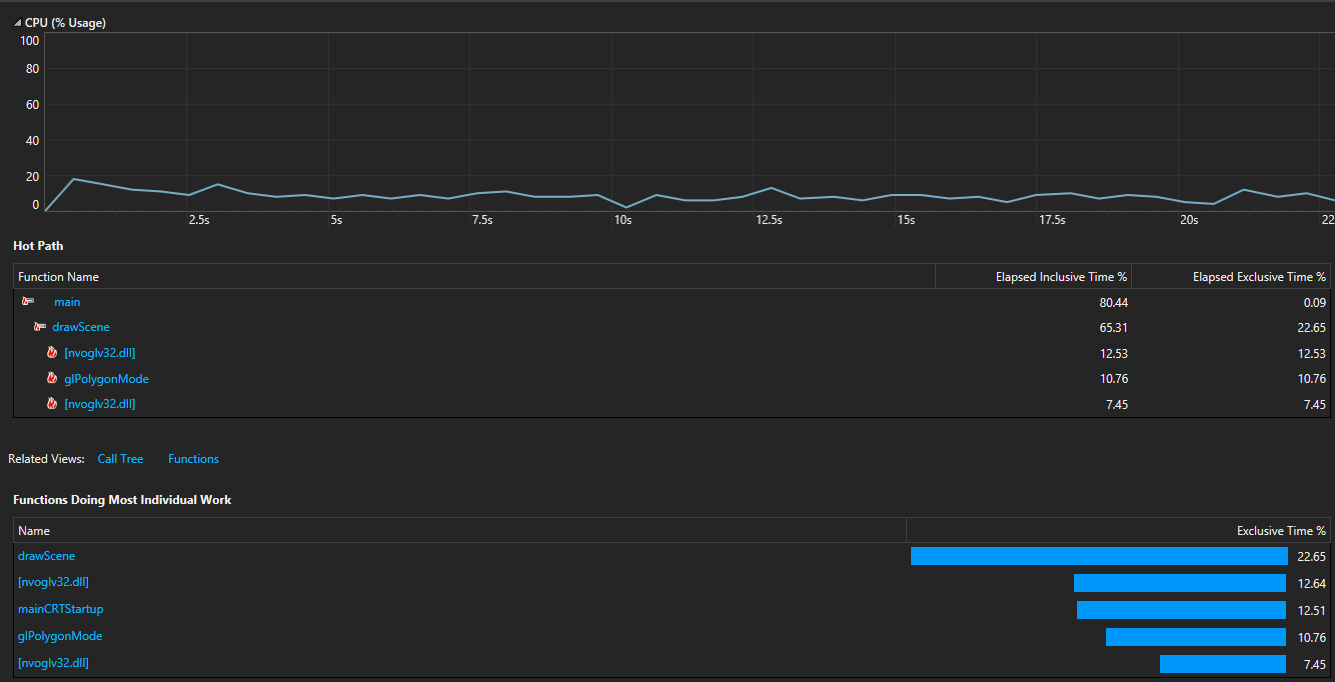
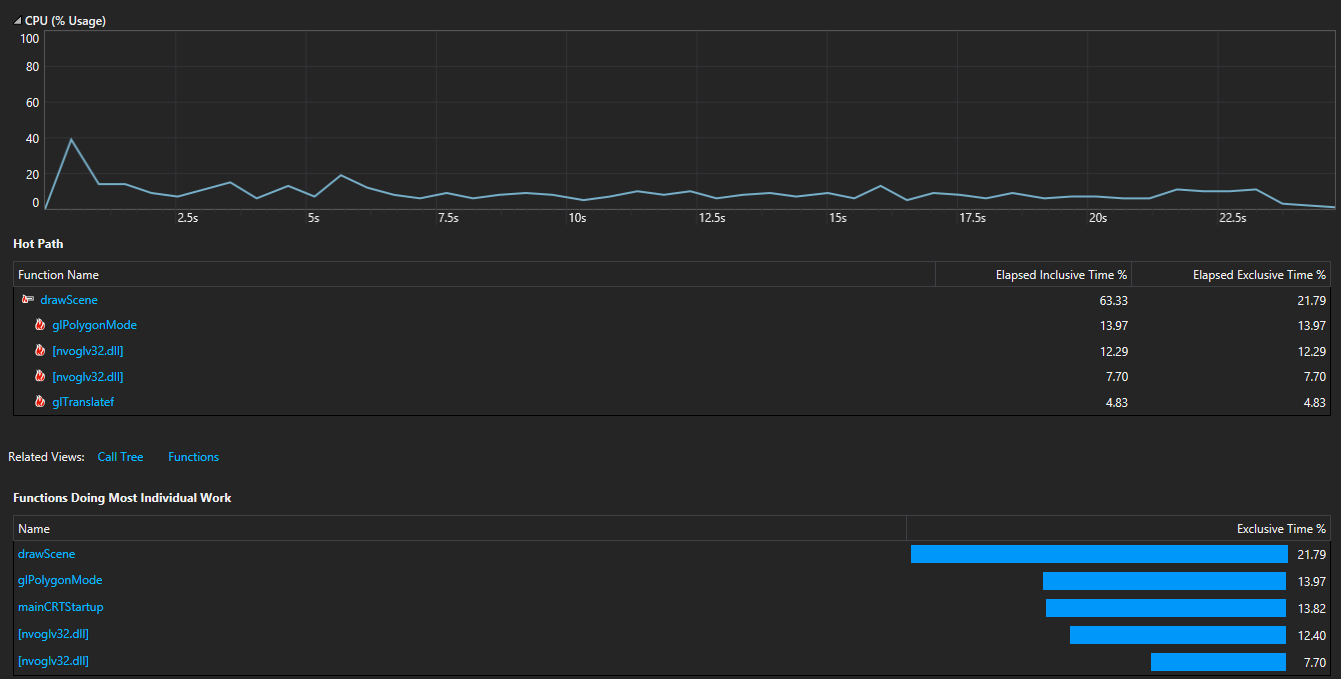
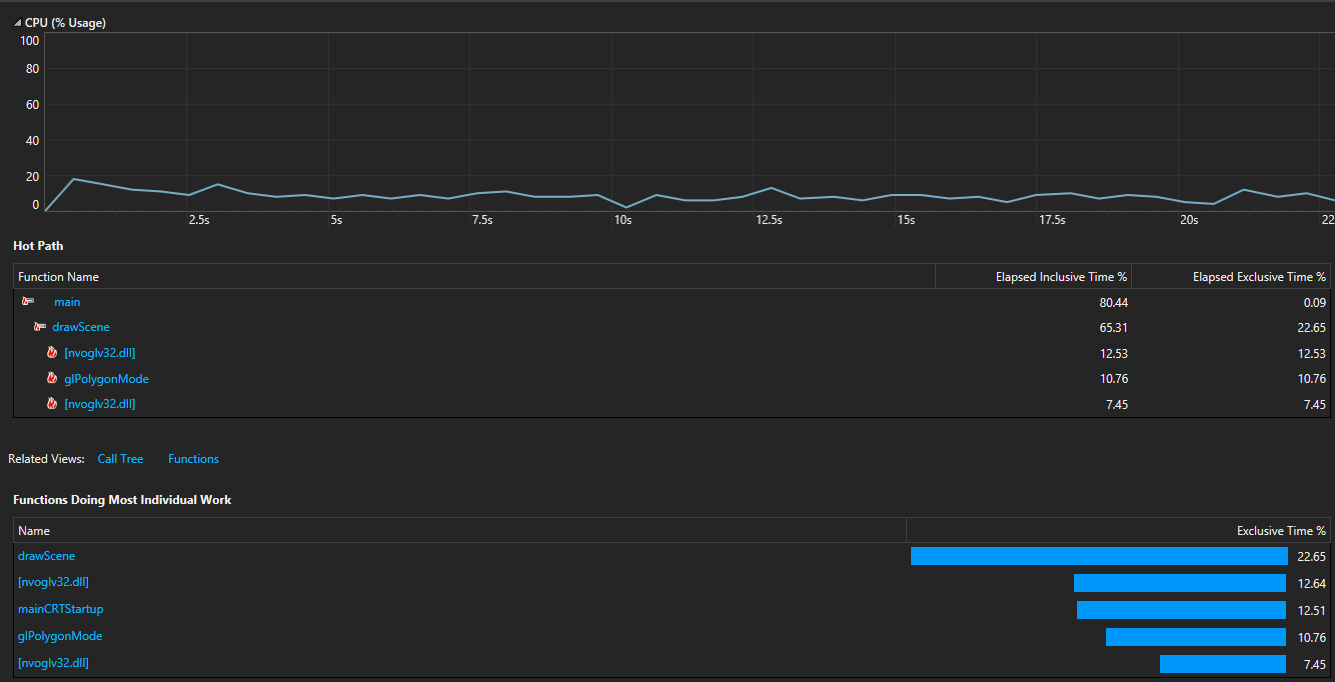
How I fixed the code:

B.

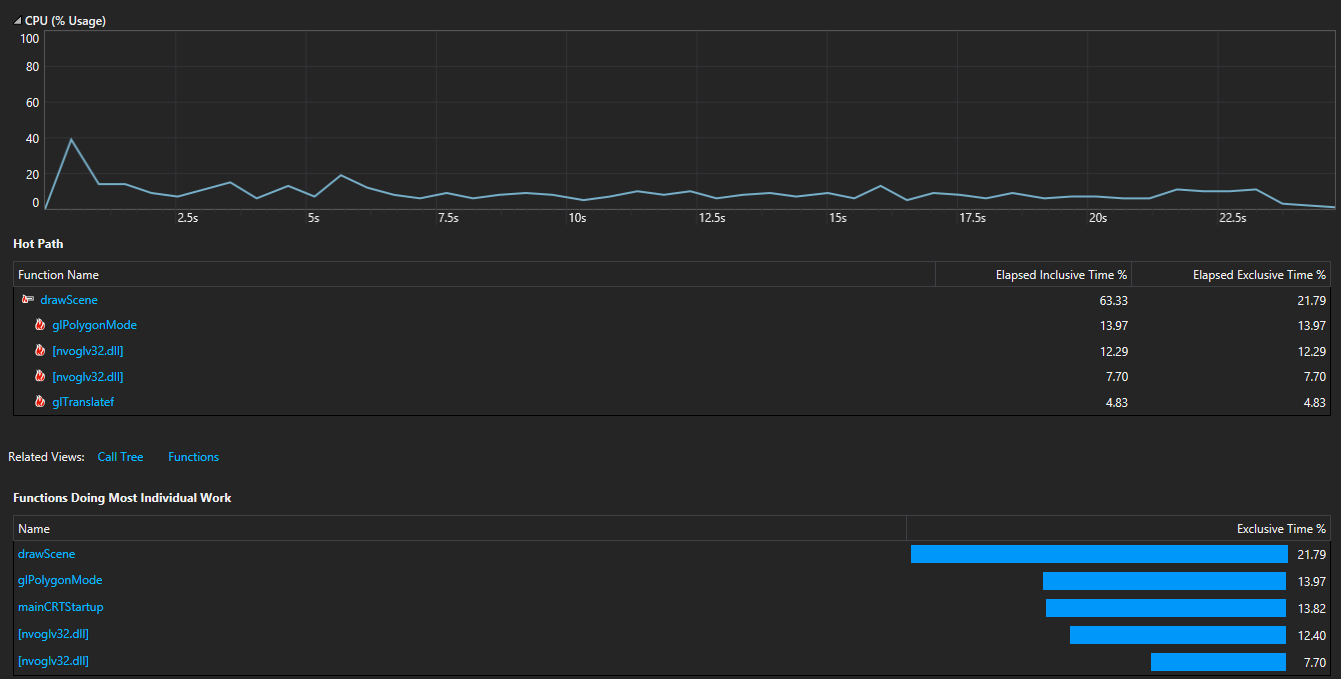
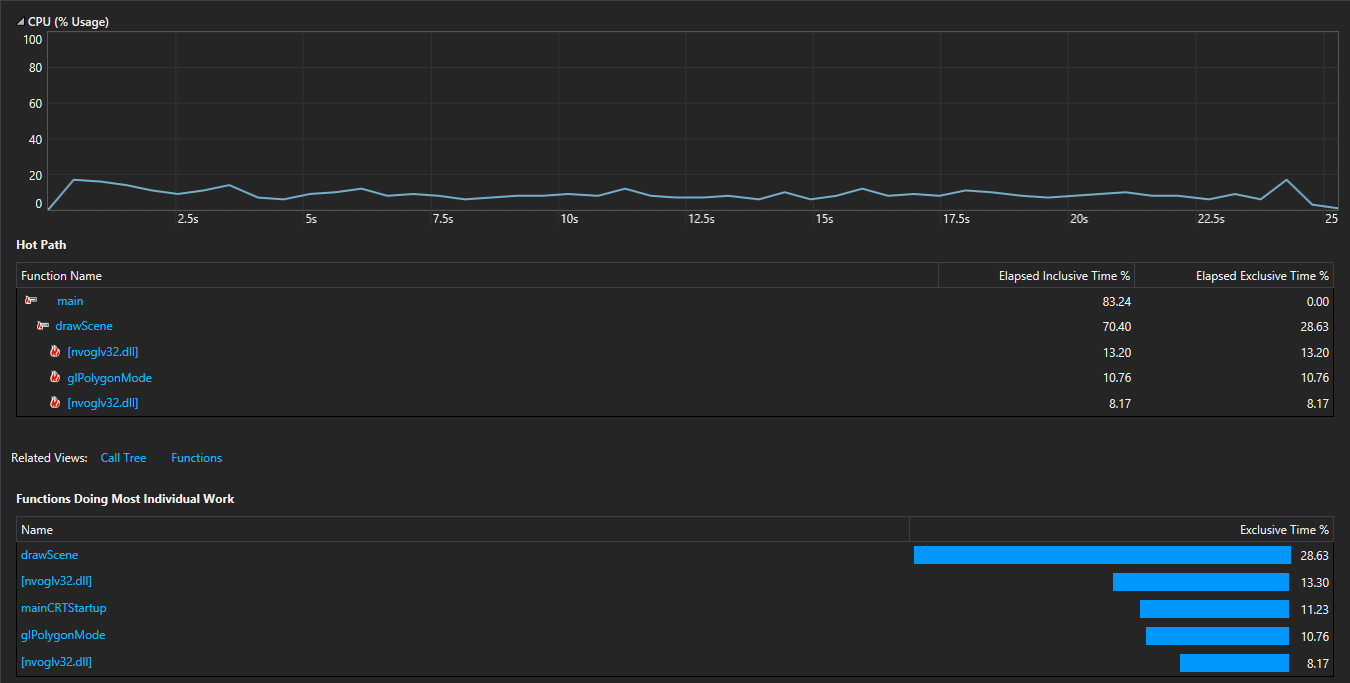
1. Remove the CreateSphere() call from Asteroid::draw()
   1. Since the spheres already existed, calling this every frame was costly for no reason.
   2. I removed the CreateSphere() call within Asteroid::Draw() entirely
   3. Considerably faster. Before:

After:

1. Change each instance of nested for loops from column first to row first
   1. Because each nested for loop was set to column first, it was skipping over the rest of a given row to get to the next due to how C/C++ arrays are organized
   2. I changed each nested loop to be row first
   3. Slightly faster. Before

After:

1. Memoized the CreateSphere() method
   1. sin(), cos() and multiplying/dividing by PI are all rather expensive, and they’re each called several times in a loop
   2. I attempted to memoize each one so they’re called as little as possible, though it only sort of worked. (Explained in section D at the bottom, and as a comment in the code). I also looped through creating the back of each asteroid, as the calculations are nearly identical besides the z coord. Moved it all to one loop as well.
   3. Slightly better performance. Before:

After: 

C. To make the quadtree more necessary, I would change the right camera to change position to show whatever quad the ship is in, and only draw whatever asteroids are within that quad, as well as only check collisions with them.

D. I had an issue within the CreateSphere() method where in my attempt to cache the sin() and cos() functions at the beginning of each loop rather than do it twice each, cutting each one in half effectively. However, this somehow changed the shape of the asteroids, and I couldn’t find out why. The math *should* be the same, and yet the points’ coordinates changed.