Points inside Mesh Solution Design

Yufan Lu

I. Architecture

1. *Basic Architecture Template – NeHe's OpenGL Tutorial*

   NeHe's OpenGL Framework

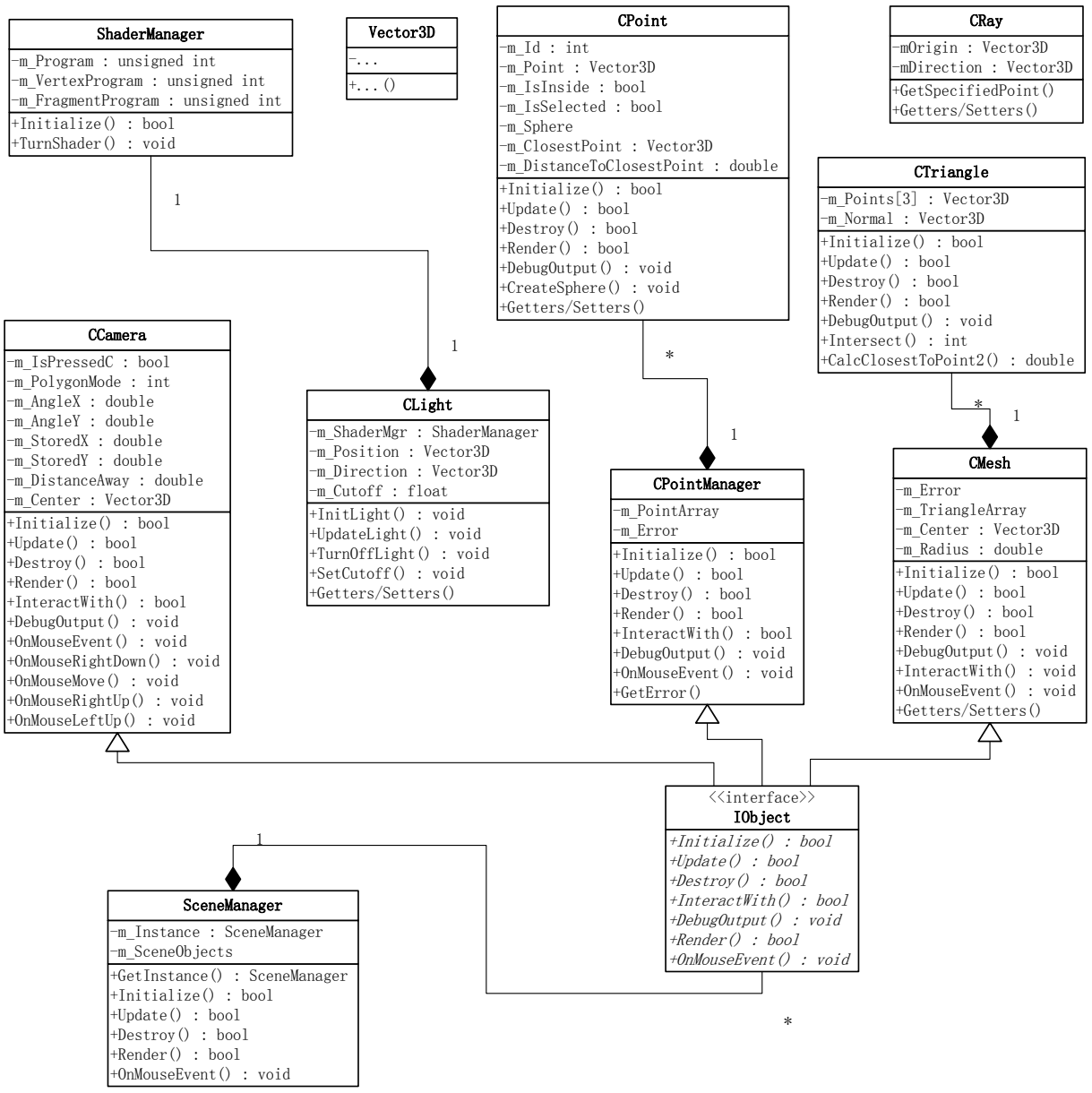
   Source: http://nehe.gamedev.net/tutorial/creating_an_opengl_window_(win32)/13001/

   Multiple key-functions exposed:

   - *bool Initialize();*
   - *bool DrawGLScene();*
   - *void ReSizeGlScene(GLsizei width, GLsizei height);*
   - *void KillGLWindow();*
   - *LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam);*

   These functions are used to connect the Win32 application and OpenGL.

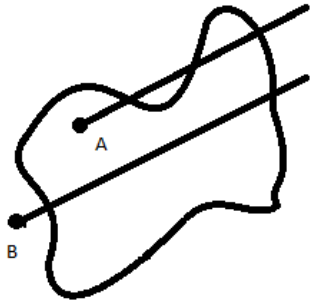
2. *UML Diagram*



II. "Inside or Outside" Algorithm Design

1. *Starting with 2D*

   Considering a closed curve in 2D space:



   And we have point A & B, and then we shoot a ray in a random direction.

   So we have this conclusion:

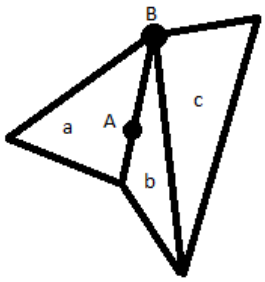
   - *If the point is inside curve, the ray should have the odd number of the intersections*
   - *Else the ray would have even number of intersections*

2. *Extending to 3D*

   Like the problem in 2D, this conclusion works in 3D space, so we only need to have a ray and calculate the number of intersections.

   a. *Problems*

      There's a problem when extending to 3D space, let's see a picture:



      i. *If the intersection is at point A, which is on the edge, then when we are testing the intersection with triangles a and b, the A point is counted twice.*

      ii. *If the intersection is at point B, which is on the vertex, then this point might be counted more than once.*

   b. *Solution A*

      Let's separate the situations into two: intersect at edge and at vertex

      i. *Intersect at edges*

         We count how many intersections at edges, and divide it by 2.

      ii. *Intersect at vertex*

         We count how many intersections happened at vertices, if it happened before, just don't count it.

      This solution will need the exact intersecting testing support, that is to say, find out if the intersection is at edge or at vertex, or inside the triangle.

   c. *Solution B*

      To save the intersecting point in a look-up table, if a new intersecting point is found in this look-up table,

we just leave it out.

3. *Overview*

Thus, we can have the algorithm as:

```
function IsPointInsideMesh: point, mesh
    let tbl be a LookUpTable
    count = 0
    for all triangle in mesh:
        if ray(from point with direction(1, 1, 1)) intersects at triangle
        let t be the intersecting point
        if tbl.notFind(t):
            tbl.add(t)
            count = count + 1
    if count % 2 == 1:
        return true
    else:
        return false
```

III. Spotlight Shader

Spotlight is a kind of special point light with the direction and angel.

In GLSL we can get the direction with gl_LightSource[x].spotDirection, which is the axis of the cylinder, and get the cos(angel) from gl_LightSource[x].spotCosCutoff.

So the algorithm we did is to get the ray from the source to the vertex, so the cosine of this angel must be greater than spotCosCutoff.

```
n = normalize(normal);
NdotL = max(dot(n,normalize(lightDir)),0.0);
if (NdotL > 0.0) {
    spotEffect =
        dot(normalize(gl_LightSource[0].spotDirection), normalize(-lightDir));
    if (spotEffect > gl_LightSource[0].spotCosCutoff) {
        /// this point is inside the lighting cylinder
        /// start calculating the light
        ……
     }
}
```

And in order to implement the effect that if the point is closer to the mesh the cylinder's angel is greater, I first calculate the approximate range of the distance to be (3, 60), and I want to limit it into (0, 60), thus I choose the function:

Cutoff = 150.0 / Distance

Effect: