|  |  |  |
| --- | --- | --- |
|  | **Rochester Institute of Technology**  **Golisano College of Computing and Information Sciences**  **School of Interactive Games and Media**  **2145 Golisano Hall – (585) 475-7680** |  |

**Data Structures & Algorithms for Games & Simulation II**

**IGME 309**

**Midterm – Practical (version g)**

I’ve included a solution under the \_Binary folder. Please take a look at that before continuing reading this document.



You are working on a loading screen for a company, unfortunately the art director only gave you his artistic idea for what a loading screen for the game should look like:

“An increasing number of diamond shapes per second on an orbit”. His artistic view is to generate an steady increase of diamonds (one per second) which sides are 1 larger than the previous. The smallest diamond will have 3 triangles on the top of the diamond and 3 triangles on the bottom part of it. The next diamond will have 4 triangles in the top and 4 in the bottom and so forth. They will be evenly distributed on an orbit which will be increasing radius uniformly (1 unit per shape 0 index based, so 1 shape will be at the center 2 shapes will have radius one, 3 shapes radius 2, etc) and the orbit will be traveling in the background 1.5 units per shape.

What you are expected to do is replicate the demo to the best of your abilities.

You will start with a cube at the center of the world, if you middle click you can rotate it around its center. This rotation behavior is to be removed.

From the starter code there are some things you need to do:

1. Create a MyMesh::GenerateDiamond method under MyMesh.cpp in order to generate a Diamond. The center of the diamond needs to be located at (0,0,0) and the tips of at (0,-a\_fHeight / 2, 0) and (0, a\_fHeight / 2, 0). Diamonds will have also a thickness to them and a number of divisions. So a signature like this is encouraged: void MyMesh::GenerateDiamond(uint a\_uSides, float a\_fHeight, float a\_fWidth, vector3 a\_v3Color);
2. Once you have the GenerateDiamond working you should generate a couple of them and place them in an orbit.
3. Once you have that part you should generate a timer in such a way that one second has passed you add a new diamond and calculate their positions in the orbit / space
4. Once you have an arbitrary number of diamonds working make them revolve around its Y axis individually and around its Z axis all together.

Your grade will happen as follows:

30% (up to) if you generate a Diamond based on the arguments

15% (up to) if you can generate any arbitrary number of Diamond with different sides

15% (up to) if you locate them in a circular orbit

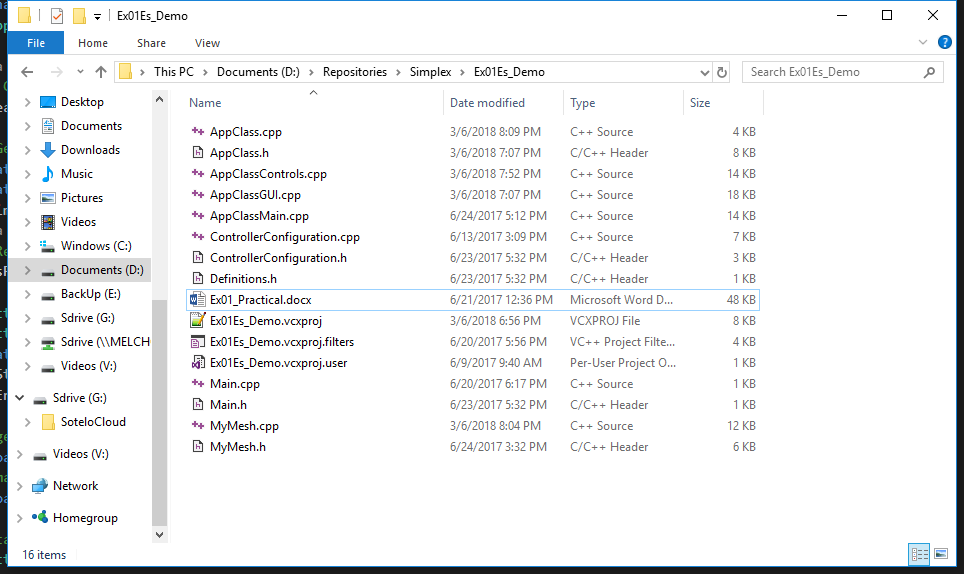
20% (up to) if you generate a new Diamond per second with the actual increment in the sides

10% (up to) if you make the mass of Diamonds revolve on the X/Y plane (Z axis)

10% (up to) if you make each diamond revolve around its Y axis individually

***Submit to the dropbox labeled Ex01 –Practical***

As usual the required submission asks only for the project folder, not the whole solution, it should be no larger than 200kb if you are using the class framework. The content of your .zip file in your submission should look like this (names might be different):



After you submit your file it is your responsibility to download your submission and make it is what you worked on and not the starter code, it has happened to other student before, do not let it happen to you.