|  |  |  |
| --- | --- | --- |
|  | **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, 2015 Spring**

**E12: Bounding Spheres**

As usual for this ICE you can use ReEngine or your own solution. There is starter code that will minimize the amount of work you have to do under the class repo. If you want to build it in your own solution it will require some code translation.

If you plan to use the provided start code you may use the E12\_BoundingSpheres project in the repository

Your goal is to check if there is a collision between their Bounding Spheres. For that purpose you will complete the BoundingSphere class that, at the very least, should have these following methods implemented:

BoundingSphere(std::vector<vector3> listOfVertex); //constructor, will take in a list of vector3 that will represent all of the vertices of your shape

bool IsColliding(BoundingSphere\* pOther); //will take another BoundingSphere and returns whether the current object its colliding with it or not.

void UpdatePosition(vector3 v3Input);// will update the position of this bounding sphere in global space

void SetModelMatrix(matrix4 a\_m4ToWorld); that will set the vounding sphere into world coordinates to be rendered

You can use any other member fields and functions that you feel are necessary for your BoundingSphere to work. Think about the example code we did in class to determine what would you need to store said information.

You will be given two models that are initialized for you and you will require the preprogrammed GetVertices() method in order to return the list of vertices from the objects (this is an std::vector<vector3>).

You will need to create BoundingSphere objects and display them around the 2 models no matter where they are being rendered in the world.

One of the shapes will be controllable with the keyboard arrows and its BoundingSphere object should move accordingly. To check that the objects have their associated BoundingSpheres moving along with them you can use the Mesh manager draw sphere method as we did in the example code. If they are not colliding draw the Bounding Sphere in WHITE, if they are colliding, draw it in RED.

Show this to the professor or TA and submit to the dropbox labeled E12 Bounding Spheres.

Remember to delete the Z\_Delete file, the .SDF folder and the projects you are not using for this solution, and rename your zip file with the naming convention E12\_LastName.zip