**Physics Module Requirements Specification**

**Important notes:**

* I’ve uploaded a suggested design document. It’s not necessarily going to be the best way to do it since I put it together in 30 minutes – but it should cover must of the structure of the module. Use it if you think it will be helpful.
* You will need to use a **namespace** to prevent the ‘Mesh’ class from conflicting with the other ‘Mesh’ class from the Acting Module.
* This module is mostly related to optimisation. Some physics related optimisation strategies include:
  + Broad phase and narrow phase collision detection
  + Using axis aligned meshes before calculating orientated meshes

|  |  |
| --- | --- |
| **1.** | **To be able to create a PhysicsNode as a shared pointer** |
|  | The module should be able to produce physics nodes as shared pointers so that they may be attached to entities (See “Entity.cpp”) |

|  |  |
| --- | --- |
| **2.** | **To be able to set the type of collision mesh** |
|  | Nodes must be able to be configured to use spherical, axis aligned cuboid or oriented cuboid based collision meshes. |

|  |  |
| --- | --- |
| **3.** | **To be able to set the size of the collision mesh** |
|  | The physics node must be able to set the size of its collision mesh. Note that the parameters of the size function may change with the current type of mesh |

|  |  |
| --- | --- |
| **4.** | **To be able to enable and disable collisions on one particular mesh** |
|  | Each node should have a method of enabling or disabling its own collisions. This is preferable to destroying and recreating the collision data each time it is toggled |

|  |  |
| --- | --- |
| **5.** | **To be able to enable and disable collisions as a whole** |
|  | The physics module should provide a means of disabling and enabling all collision detection. This is useful for saving processing power while gameplay is not occurring (on the splash-screen, for instance) |

|  |  |
| --- | --- |
| **6.** | **To be able to detect and resolve collisions of the following type:**  **Spherical vs Spherical**  **Spherical vs AAC**  **Spherical vs OC**  **AAC vs AAC**  **AAC vs OC**  **OC vc OC** |
|  | The physics module should have a method of detecting and resolving collisions amongst two collision meshes |

|  |  |
| --- | --- |
| **7.** | **To be able to register a collision ‘listener’ with a physics node to receive notification of when it is hit** |
|  | Each physics node will need a way to add a ‘listener’ to it. Listeners will then be able to receive various notifications relating to the physics node, such as “onCollision()” and “onPositionChanged()”. This should use the observer design pattern. You can find this in many places throughout the engine, such as the TimingModule |

|  |  |
| --- | --- |
| **8.** | **To be able to make a physics node immovable** |
|  | Physics nodes must be able to be immovable. This will be used to stop the player from shoving terrain and decorative objects out of the way |

|  |  |
| --- | --- |
| **9.** | **To provide a function that, when called, will perform a collision check between the calling node and all other relevant nodes** |
|  | A physics nodes collision detection and resolution is not performed in the update function of the module. That function is for physics module updates (sorting lists etc). The node should provide its own method of checking collisions that can be called via the entity update function. |