ETGG3802	Lab2: **cmake + bullet**	Points: **100**[1]
Assigned: **1/19/2017**	Due: **1/31/2017 (before class)**	

Overview / Goals

- Get some exposure to cmake and building a dependency from source
- Integrate physics into our engine
- Test our callback mechanisms

Tasks:

1. Get a working copy of Lab1 (if yours wasn't finished). Glance at the solution even if you *think* your solution is 100% correct.
2. Build bullet from source (I'm assuming your downloaded folder is in c:\temp\bullet3-2.85.1)
   a. Make these configuration changes: enable USE_MSVN_RUNTIME_LIBRARY_DLL
   b. Build the ALL_BUILD (Debug and Release) *then* the INSTALL (Debug and Release) projects.
3. Move to our ssuge/dependencies folder (in a bullet_2_85_1 sub-folder)
   a. Copy contents of c:\temp\bullet3-2.85.1\src to <ssuge_home>\dependencies\bullet_2_81_1
   b. Copy contents of c:\temp\bullet3-2.81.1\build\lib to <ssuge_home>\dependencies\bullet_2_81_1\lib. You only need:
      i. BulletCollision[_debug].lib
      ii. BulletDynamics[_debug].lib
      iii. LinearMath[_debug].lib
   c. Do the normal setup to include this new dependency (no dll for this one).
4. Additions to ssuge
   a. [**20 points**] **PhysicsManager** manager class
      i. A singleton
      ii. Use https://github.com/bulletphysics/bullet3/blob/master/examples/HelloWorld/HelloWorld.cpp as a guide.
         1. Add the initialization "stuff" to the constructor[2].
      iii. Method to set gravity
      iv. Update method to step the simulation
      v. Have *functions* to convert between Ogre and Bullet Vector3 and Quaternions
   b. [**30 points**] **PhysicsComponent** class
      i. Derived from Component, as our other components
      ii. enum class indicating which type of collider to use (for now, just BOX and SPHERE)
      iii. Have a means to specify whether this object is under physics or Ogre control. You only need to implement the former – the update method should make the ogre object move to the bullet object's position.
   c. [**5 points**] **PhysicsComponentFactory** manager (use MeshComponentFactory as a guide)
   d. [**5 points**] (Modifications to) **script_game_object** "class":
      i. Have a method to create a physics component (see init.lua for an example)
   e. [**5 points**] (Modifications to) script **top-level functions**

[1] I'll cap you at 140 points…

[2] After we get the basics working, we'll likely tweak this initial setup.

            i. Add a **setGravity**(x, y, z) lua function
    f. [**15 points**] (Modifications to) Application class:
        i. Create the PhysicsManager and PhysicComponentFactory singletons and destroy them at the end.
        ii. Attach (in C++) a PhysicsComponent to the ground plane (I'd just use a box)
        iii. Update the PhysicsManager before updating game objects in the run method.

5. Test it! Use something similar to the init.lua file on ssugames.
    a. Feel free to edit, but make sure you support everything in the init script
    b. (although, I implemented Bonus 6.b – if you aren't doing this, the creation of colliders will be a bit different)
6. Bonus
    a. (**25 points**) When two objects collide (I plan to use http://www.bulletphysics.org/mediawiki-1.5.8/index.php?title=Collision_Callbacks_and_Triggers [the first method]), call a onCollision script method in the two game objects, if they are script-aware. We'll do this in the "group" ssuge projects…
    b. (**10 points**) Add a means to *infer* the bounding area for the collider, rather than having the user call this.
        i. All Ogre::MoveableObject's have a method to get their AABB
        ii. You'll need to account for the scale of the object.
    c. (**10 points**) If the game object is scaled at run-time, rebuild the rigid body. Use the nodeUpdated method for full points.
    d. (**20 points**) Support ogre-controlled objects (e.g. the player) in addition to bullet-controlled objects.
7. (**10 points**) Documentation and style on new files.
8. (**10 points**) Properly Submit (you may want to make a copy first)
    a. **Make sure all .h and all .cpp files are in /include and /src, repspectively (remember: they default to the build folder)!!**
    b. Make sure you have a Debug and Release build working
    c. Delete the contents of your tmp and dependencies folder
    d. Delete the build\ssuge.sdf file and build\ipch folder.
    e. Change the title bar or your application to "lab2_xy" (xy is your initials)
    f. For reference: your file should be about:
        i. 53 MB if using .7z format
        ii. 89 MB if using .zip format