ETGG3801	Lab2: **Linking / Building**	Points: **50**
Assigned: **9/5/2017**	Due: **9/21/2017 [Thursday] before Jason leaves (~5:00pm)**	

Pre-Lab Discussion

- Discussion of how this fits into our larger engine
- std::vectors and std::map

Tasks:

1. Start from a working version of Lab1. I made a few small changes from my lab1 solution (mainly changing the name from lab1_soln to ssuge).
2. Download the built Ogre 1.10 library from ssugames (the full version has docs, samples; minimal only has files necessary in this lab).
3. (**10 points**) Change project settings to incorporate the new dependency.
    - (Remember to do most of these for All Configurations)
    - Properties => C/C++ => Additional Include Directories. Add the Ogre include folder.
    - Properties => Linker => General => Additional Library Directories. Add the ogre lib folder *and* use the $(Configuration) variable to set the debug/release sub-folders.
    - (This one will need to be done individually for debug and release)
        i. In Debug build, add "OgreMain_d.lib" to Properties => Linker => Input => Additional Dependencies
        ii. Similarly for release, but use "OgreMain.lib"
    - Edit stdafx.h and add an include for <ogre.h>
4. (**30 points**) Create a very simple Application class in a namespace called ssuge. This will be the (almost entirely) platform-agnostic hub of all ssuge-programs.
    - Split the class into declaration (.h) and definition (.cpp)
    - Should have <u>only</u> these attributes (protected)
        i. An Ogre::Root *
        ii. An Ogre::RenderSystem *
        iii. An Ogre::RenderWindow *
    - <u>Should have these methods:</u>
        i. A constructor – just initialize the variables to NULL (or nullptr)
        ii. A destuctor – make sure to clean up any memory <u>you</u> allocate!
        iii. **initialize_win32** – pass in the HWND from the main program from Lab1. This should call initialize_common to do any platform-idependent ogre-setup. Don't let the compiler "see" this function if it's not a windows application (use the _WIN32 macro)
        iv. **initialize_common** – the vast majority of ogre setup code should go here. I passed it a Ogre::NameValuePairList (by reference).
        v. **update** – pass in a delta-time (float, in seconds) from the lab1 main program. It'll be empty in this lab. I would suggest calling this from winMain as an else to the if (TranslateMessage) block [in other words, we update / render our ogre application if there's no win32 event].
        vi. **render** – just call this immediately after update from lab1. It'll have a call to root->renderOneFrame. Call this one in winMain after update and also in response to the WM_PAINT message in WndProc.
    - I want the Ogre application to be 1024 x 768 (we'll change the main program to reflect this), not full-screen. Create a viewport and change the background color to something easily identifiable (i.e. not white or gray). I don't want you to use the ogre configuration dialog box for changing settings (I want you to do it manually). I also want the ogre window to render into our existing win32 window from lab1.
    - Use an ST_GENERIC scene manager (octree)
    - You can use my Ogre quick-start guide or the internet – your choice. IF using my guide, you probably want section 3, except for most of 3.6 – 3.8.

5. (**10 points**) Modify the main program to create a minimal Ogre application which renders to our existing window.
    - This should be pretty simple – just create an Application instance, call initialize_win32 and make sure it's destroyed when the program shuts down.
    - Change the window we created in lab1 to be 1024 x 768 and don't allow the user to re-size it (do you see why?)
6. I'll grade this one as I did Lab1 (you show it to me by the end of class Thursday and I'll check-mark it). I'll post your grades on blackboard, but you don't need to submit anything on blackboard.