Computer Graphics Assignment Expectations

1. The homework should be in a ZIP or TAR file that extracts the unpacks the files into the same directory as the ZIP file. Storing textures and similar files in subdirectories are acceptable, but the makefile and executable must be in the base directory.
    - Do **make clean** before creating the ZIP or TAR to remove object and executable code.
    - If you use Qt do a **make distclean**
    - Check that all necessary files are included, including the README and makefile.
    - The source code of the CSCIx229 library must be included (if used).
    - Clean out unrelated code and data files.
    - Build what you uploaded on a different machine to check your upload.
    - Watch out for file case - it matters on Linux.
    - OSX users make sure you ***do not*** include __MACOSX.
2. Document the code in a README
    - Call the file README.
    - Prove special instructions if building the program requires more than just simply doing **make** or **qmake**.
        - If using something like cmake that requires a multi step process to compile, provide a makefile that perform these steps.
    - Describe all key bindings required to run the program.
    - If you want me to take note of something, point it out in the README.
    - Include the time spent in the README.
3. Acknowledge code reuse
    - In the README provide general references.
    - When functions are reused, provide a reference in the code.
    - If you started with something that is substantially what the minimum requirements are, point out what you contributed.
4. The makefile must work on Linux, OSX and Windows.
    - The executable must be called **hw*x*** (e.g. for homework 3 it should be **hw3**). For the final project the executable should be called **final.**
    - If using Qt, call use **hw*x*.pro**. If you need additional libraries that are OS specific make it conditional (e.g. **win32 {LIBS += -lopengl32 -lglu32}**).
    - The homework should be the default target in the makefile.
    - The makefile must build any libraries that may be needed, but you can assume that I will have system versions of these commonly used libraries available.
        - libglut3-dev
        - libsdl1.2-dev
        - libsdl-mixer1.2-dev
        - libsdl-image1.2-dev
        - libsdl2-dev
        - libsdl2-mixer-dev
        - libsdl2-image-dev
        - libglfw3-dev
        - qt5-default
        - libjpeg-dev
        - libpng-dev
        - libtiff-dev
        - libglm-dev

      Note that the specific verion will be a stock Ubuntu LTS, so if you need a specific verion of a library your makefile will need to build it from source. Check with me if there are other libraries provided by Ubuntu that you want to use.

- Do **not** attempt to build anyting in /usr/local or similar system wide locations, as the requirements for your program may break others.
- Do **not** expect me to build complex dependencies by hand.

5. The code must be portable.
   - Use #ifdef to compensate for differences in header files, e.g. **GLUT/glut.h** on Apple.
   - If GLEW is required it must be made conditional with an #ifdef, since I do not need or provide it.
   - Make sure you have required header files like **string.h** and **math.h** if you use it.
   - I will use the current gcc/g++ on Ubuntu to compile your code, which for Ubuntu 18 uses the gnu11 dialect for C and gnu++14 dialect for C++ by default. If you need a different dialect you need to set your makefile to ask for it explicitly. The machine in CSEL is a good place to test what I will see.
6. The code should be clean and follow good software practice.
   - The code must compile cleanly without warnings or errors.
   - Use appropriate data structures, functions and loops.
   - If you choose to use C++, develop a proper set of classes, don't just use the C++ compiler but write C code.
     - Qt requires that you use C++, but provides elegant base classes.
   - Comments should be meaningful and sufficiently detailed that I can follow what you are doing.
   - If the code is sufficiently complex, break it into different files.
   - Remove dead code, do not simply comment it out.
   - Remove unused variables.
   - Use consistent formatting, indentation and style.
     - Do not use tabs - it does not give consistent results.
     - Refactor borrowed code to conform.
   - Compile with **-Wall** so the compiler helps you find problems.
7. Make the program work on startup.
   - The code should immediately show the best possible view of your scene.
   - No keystokes should be required to view the scene.
   - Enable lighting, textures and similar features by default.
   - Provide an overhead view mode so I can get an overview of everything.
   - Provide shortcuts to things you want me to see. Mention this prominently in the README.
8. The code must be correct.
   - The scene must not distort when the window is resized.
   - Objects should render correctly and hidden lines and surfaces must be removed.
   - Objects should (usually) be water tight.
   - Lighting must be correct, specifically surface normals.
   - Textures should be appropriate for the scale they are used.
   - Textures should be small. Textures larger than 256x256 are rarely justified.
   - Use textures sizes that are a power of 2 even it is not strictly required.
   - Add an **ErrCheck** at the bottom of the display function to catch OpenGL errors.
   - Calling **Fatal** is acceptable for trapping errors.
   - You own all mistakes in reused code, including mine.
   - Under no circumstances should your program segfault.
9. The code must be user fiendly.
   - The window title should be your name.
   - The use must be able to freely navigate, but reasonable limits such as not being able to go under the ground surface are appropriate.
   - Navigation should be natural and intuitive.
     - Use cursor keys or mouse to change view direction.
     - The keys or mouse should do what you expect them to do.
     - Use WASD and keypad as required.
   - Provide a way to exit gracefully (ESC or q).
   - Allow the window to be resized.
   - Do not capture the mouse.

    - Do not force full screen mode. Full screen may be appropriate for games, but makes grading difficult so make it optional.
10. The homework requirements are an absolute minimum.
    - Assignments are open ended so that you can be creative.
    - My examples are intended to illustrate specific points as simply as possible. Much more is expected for the homeworks and project.
    - The goal is to impress your friends. That takes effort.
    - If you do cool stuff, show up for class because I may show your homework.

Gotchas

1. When reusing my code, watch out for the **mode** variable. It means something different in every example.
2. Watch out for upper and lower case in file names. It matters on Linux, but not on Windows and OSX.
3. Watch out for the varargs in the Print() function. The compiler does not check the type and number of arguments.
4. My machine runs without VSYNC and at a high frame rate. Make sure that you animate at a predictable frame rate using a timer or reading the wall clock.