COM S 327, Spring 2017
Programming Project 1.06
"Fog of War" and interfacing C with C++

Working with C and C++ in one project can be tedious. If you've got a C library that you want to use in a C++ project, it's simply a matter of wrapping headers in an `extern "C"` block. And even if the headers themselves aren't suitably sanitized, you can usually get away with wrapping the `include` directive. Going the other direction—using a C++ library in C code—is usually more work. Here you've got to explicitly export a C interface from the C++ code, so if the library developer hasn't already done it, it will require source code, it could be a lot of work, and you may be better off choosing another library.

Working with C and C++ source files in a single executable; well that's just silly. I *have* seen examples of it, but I can't see any good reason for it, except maybe to teach, which is why we're going to do it. 1.07 will convert the whole project to C++, so we won't have to deal with this tedium again, but for now, I want you to get experience interfacing C and C++ code, so this week only we'll have them both in our project.

This description of how we'll divide code into C and C++ parts applies to those of you who are using my code drops or otherwise developing your own code based roughly on the design ideas I discuss in lecture. If your code differs in the implementation of characters significantly, we'll need to figure out another way to do this division for you. In that case, please arrange to discuss it with me.

My code uses an object oriented design of the `character_t`, with sub-types `pc_t` and `npc_t`. We will be converting all three of these to classes with C++ class inheritance. Any method that you want to call from the C side of your code will need to export a C interface.

To the PC class we're going to add yet another map of the dungeon, this one will contain terrain—like the first map we ever made—however, it will be the terrain that the PC can see or remembers having seen. The dungeon is now dark. In a later assignment, we'll implement objects, including lights, but for now we treat the PC as if it is carrying a light with a light radius of 5 dungeon cells. Everything within that radius is illuminated and thus visible. Dungeon terrain, once seen, is remembered.

Rendering is changed to render only visible monsters, and visible and remembered terrain. All unseen and unremembered terrain is rendered with spaces (like rock). Note that monsters can alter the dungeon. If the PC doesn't see this happening, what has been remembered doesn't change. For instance, dungeon cell (y, x) is rock and is in sight of the PC. The PC moves to a position where (y, x) is no longer visible, and a monster tunnels through that position, converting it to corridor. The PC hasn't witnessed that event and doesn't know about it. The rendered view still displays the cell as rock. The PC moves back so that (y, x) re-enters view and the remembered terrain is updated so that the cell is displayed as corridor.

All new code should be written in C++, except what is necessary to interface into the C side of the project. `character.c`, `pc.c`, and `npc.c` should be renamed `character.cpp`, `pc.cpp`, and `npc.cpp`, The three associated structs should become classes. If you're developing your own code, conform to the *spirit* of these ideas, and contact me if you have questions.