ETGG3801 Final Review Topics Final is 12/7/2017

Major topics, broken down into (mostly) arbitrary categories

- C++ specific
  - C++ pointers (the same as C pointers)
    - connection to arrays
    - memory allocation (new, delete)
    - getting address of
    - connection to structures / classes
    - de-referencing
  - C++ OOP
    - Access modifiers
    - constructor / destructor
    - static members (including how to initialize a static member)
    - methods
    - argument passing [these apply to stand-alone functions as well]
      - pass-by-pointer
      - pass-by-value
      - pass-by-reference
    - splitting into .h / .cpp (plus when to do things inline)
      - how circular dependencies occur and how to fix them.
    - inheritance
      - polymorphism and
      - pure virtual methods and abstract classes
    - alternatives to OOP (composition, especially)
  - namespaces (using and creating)
  - pre-processor macros (and include directives)
    - what #pragma once does (how its implemented)
  - template functions and template classes
    - template specialization
  - Pre-processor => Compiler => Linker => Runtime connections (and major settings in VS2017)
    - The process of incorporating an external package (e.g. Ogre, Irrklang, Python) into a project.
    - Role of pre-compiled headers
  - Major STL containers: maps, vectors, lists
    - iterating through
    - adding things
    - getting things
    - finding things
  - some kind of File I/O (C++ fstreams or C FILE objects)
- Ogre-specific
  - The role of the main objects we've encountered so far:
    - SceneManager
    - Root
    - Entities
    - SceneNodes

        - Cameras
        - Viewports
    - Scene management (parent / child relations, how it changes transforms)
- Python / C
    - high-level view (i.e. short answer question) on how we create a python extension in C
    - Some of the common operations we used in the lab:
        - Checking for correctness of parameters
        - Getting / Setting items in a tuple, creating a tuple
        - basics of incref and decref.
        - The layout of a typlical function (PyObject* func(PyObject * self, PyObject * args)
        -
- Design Patterns (main idea, implementation in C++, etc.)
    - Singleton
    - Factories (like GOM)
- Our interesting design decisions in ssuge so far:
    - the game-object / component connection
        - one vs. many of the same type in the same game object
    - the role of the GOM

Exam Format:

- About 4 - 7 pages.
- Will take the entire class period
- Pencil-and-paper, closed-notes, closed-computer
- You may, if you wish, bring a one page (8.5" x 11", one-sided) "cheat sheet"
- Questions will be a combination of:
    - Short answer
    - Code Analysis (I give you a code snippet, you tell me what it does)
    - Code Writing (I give you a small-ish goal, you write a concise program)
        - Generally, if you're close on function names, I give you full points
        - I'm focusing on whether you understand the concept. Sometimes this means syntax, sometimes not.

What will we do during our "regular" final exam time (12/14/2017 @ 8am)?

- 8am – 9am:
    - If you're finished merging and choose to participate (it's OK if you don't), we'll discuss ways to make ETGG3802 (next semester) better than ETGG3801. Be careful: if we come to a concensus, you'll be bound by the decisions we make (in next semster's syllabus).
    - If you're not finished merging this would be a good time to finish. PLEASE don't wait until today to start merging.
- 9:15am SHARP: start the group demos (10 minutes each group, with 2-3 minutes tear-down, set-up). Attendance is required.