**Game engine architecture document**

OGLEngine, Leo Asplund, 12 Dec 2024.

*What managers mentioned would be relevant to my game engine? (see lecture or powerpoint from lecture for inspiration)*

**Singleton managers:**

(order of priority):

+ Mesh manager,

*-for caching loaded meshes and using cached meshes instead of loading new ones every time.*

- Texture manager,

*-load textures,*

*-list of textures,*

- Camera manager,

*-multiple cameras, especially like in Unity where you can have both the Play-in-Editor window and regular editor window viewing at the same time.*

*-camera settings.*

- Graphics manager,

*-keep track of which entities are registered for rendering (e.g. a bool in the hierarchy window to enable or disable rendering of a specific object.)*

- Shader manager,

*-keep track of loaded shaders, assist in loading shaders, recompiling shaders if needed, OS/platform specific shader selection.*

- Maybe even a material manager / editor, so we can have specific materials for meshes whenever we implement PBR.

- Audio manager,

*-queue sounds,*

*-play sounds,*

*-queue music or ambience,*

*-play music or ambience,*

*-FMOD or OpenAL middleware communication and integration. Audio is important* *for games.*

- Animation manager,

*-load and manager 3d animations for models or particles (requires better asset importing than just .obj, right? Since .obj doesn't support animations).*

**Lower level singleton managers:**

- Input manager:

*-keep track of inputs,*

*-keep track of states of keys and buttons for both keyboard, mouse and gamepad.*

- Window manager:

*-Create and handle OS-windows, like in case I want to move an ImGui window to my second screen.*

*-Handle auxiliary windows (basically just an ImGui window manager for any tools I create for my editor in my case.).*

*-Process OS-level events (e.g. wndproc).*

- Application State manager:

*-handle changing states if my application has several modes (e.g. edit mode, play mode and whatnot).*

Threading?

How would I set up threads and messaging architecture? Or just do it on one thread and ensure all procedures are prompt and responsive? (re-write all my loading code to make sure it doesn't hang the program until it's done loading.)

Threading requires solid messaging architecture across threads, locking mutexes (?). Otherwise we risk writing to some data from two different location at the same time.

Messages:

How to handle messages between sub-systems in my game engine?