Subject: Re: Architectural and C++/Upp questions
Posted by Xemuth on Mon, 28 Sep 2020 21:50:30 GMT
View Forum Message <> Reply to Message

Hello Klugier, Thanks for your awnser and your time !

1. ---------------------------------
Quote:1. You could create API similar to Unity. You will need two following classes:
- SceneManager (Context probably not very approrpiate name)
- Scene
Indeed the main idea behind Context was to create a Scene manager ! but yeah looking at SceneManager, it will be smarter to create a SceneManager class and give it to the context. Context must carry services and track time ellapsed.

Quote:you could use String as in Unity to identify specific scene Yeah I think it will be easier.
Quote:The next think I would change is the use of reference. I would opt for pointer, because right now you can not detect errors in your API My code is actually full of exception.

2. ---------------------------------
The main idea behind virtual Vector<Value> ReceiveMessage(const String& message,const Vector<Value>& args = Vector<Value>());
is the fact Service can be everything, by example my openGL Service looks like this : https://i.imgur.com/2BgUoRa.jpg
Many of functions here must be used by user, but in order to allow my gameobjects to be render they must call some function of this service, to do it. They send message the service.
The message must be "AddQueue" and it must have 3 args (name of MasterRenderer, Renderer, RawModel to use which are all String). Since service is all up to the developpers which developpe them, it can be anythings

Quote:Backing to static_cast proble. Why not use template method instead and do the cast here - it will be hidden for the user: Yeah it is the same but it will be the best solution I think

Quote:3. Vector<Value> - seems like task for optional not vector. In c++17 std::optional should do the job. In the world of U++ you could return One<Value>. For more information please read - Core tutorial (3.11 One). What if the receveid message must return multiple Value ?

After all, maybe it would be simpler to retrieve the service reference directly in game object and use it to communicate with the service