Subject: Architectural and C++/Upp questions
Posted by Xemuth on Mon, 28 Sep 2020 13:44:11 GMT
View Forum Message <> Reply to Message

Hello, I'm currently working to improve the package I introduced here :
https://www.ultimatepp.org/forums/index.php?t=msg&th=111 48&start=0&

To improve it and add new functionnality / modularity I started to write bunch of new class:

One of them is named Context and hold a private Vector of Scene Object. Some of public methods allow user to Create/Remove/Get all the scene object depending on ID (int):

```
Scene& CreateScene(); //Return the fresh created scene
bool RemoveScene(int sceneId);
Scene& GetScene(int sceneId); //Return scene depending on ID
```

2 questions come in my head when reviewing this code
-First one : Is it good to return reference of Scene in CreateScene() ? the fact a Vector is holding Scene mean the reference can get dereferenced at next creation/ destruction, on other hand returning the ID of the scene in CreateScene() force user to call GetScene(int sceneId) (which could lead to the same behavior) May I should use Array instead of Vector and return Reference ? What you think ?
-Second one : Since ID is (in term of human view) less simple to remember, is it a good idea to swap int ID to String name, and use name to identify all Scene ?

The Context class hold one Array of Service.
Service class can be seen as a abstract class : class Service{

```
 public:
  Service();
  Service(Service&& service);
  Service(const Service& service);
  virtual ~Service();

  bool IsActive();

  virtual String GetName() = 0;

  virtual void OnCreation();
  virtual void OnDestruction();

  virtual void BeforeUpdate();
  virtual void AfterUpdate();

  virtual void BeforeActive();
  virtual void BeforeInactive();
```

```
  virtual Vector<Value> ReceiveMessage(const String& message,const Vector<Value>& args =
Vector<Value>());

 protected:
  friend class Context;
  Service& SetActive(bool b);
 private:
  bool active = true;
};
```

Later in my code, I want to inherite class of this Service base to integrate some new functionnality to my codebase. By example, my first test was to create an OpenGL service to render things on screen (it worked). The fact is, my OpenGL service hold mutch more functions than the abstract class Service which force me to do this (in order to call the child class functions):

```
Upp::RendererOpenGLService& service =
ufeContext.CreateService<Upp::RendererOpenGLService>();
/**
      I created the service and get a reference of RendererOpenGLService
      .....Later in the code :
**/
Upp::RendererOpenGLService& service =
static_cast<Upp::RendererOpenGLService>(ufeContext.GetService("RendererOpenGLService"));
```

I really don't like this static_cast, I feel like I could do the job I want a much better way. Can you guys confirm me this static_cast could be avoided in profit of better option ?

Find here a diagram of my service and my context : https://i.imgur.com/2BgUoRa.jpg
---

You may have noticed in service class I have this function :
`virtual Vector<Value> ReceiveMessage(const String& message,const Vector<Value>& args = Vector<Value>());`
Since my Context can hold multiple service, I have integrated a way to communicate with all services without knowing which service is who.

Here is how it work (taken from RendererOpenGLService) :

```
virtual Vector<Value> ReceiveMessage(const String& message,const Vector<Value>& args =
Vector<Value>()){
 if(message.IsEqual("AddQueue") && args.GetCount() >= 3){
  try{
     AddDrawToQueue(args[0].Get<String>(),args[1].Get<String>(), args[2].Get<String>());
     return Vector<Value>{true};
  }catch(Exc& exception){
     Cout() << exception << EOL;
```

```
 }
 }
	return Vector<Value>{};
}
```

It work but I feel like, using Vector<Value> and returning Vector<Value> is not the best way I could use to communicate between Services (and it do a tons of cast (I think it is really costly in term of performance)). Any advise to improve and/or a better way a communicating between abstract code ?

Don't see this post as a laziness from me (in term of architecture choice)but much more as a knowledge calling. I plan to spend a lot of time in this project to provide the most dynamic/modulable engine to work in 3D environment in Upp. That's why I want to go on a good choices codebase. If you think my project (the way I'm architecting it) is actually a Gaz factory model which will lead to too complex things, feel free to say.

Thanks for the time you will take to awnser this post.
Have good day

Xemuth