Subject: C++ templated class referencing each other
Posted by Xemuth on Sat, 10 Oct 2020 22:45:36 GMT
View Forum Message <> Reply to Message

Hello,

I have 2 class :

Object.h

```
#include "ComponentManager.h"
class Object{
    public:
      Object() : componentManager(*this){}
     /*
         Many public things
     */
    private:
       ComponentManager componentManager;
};
```

ComponentManager.h

```
class Object;

class ComponentManager{
    public:
     ComponentManager(Object& _object) : object(_object){}

     template<class T> T& function1(...)
     /*
        Many templated function that force me to write declaration in this .h file
     */
    private:
      Object& object;
}
```

My both class are made to work together, ComponentManager make no sens without an object etc...
All is fine and work well. Untill, in one of my templated function in ComponentManager, I need to use my Object reference to call a function of object :

```
template <class T> T& CreateComponent(bool active, int position){
/*
   Working code
```

```
*/
Object& obj = object.GetScene().GetObjectManager().GetObjects()[i];  //This line is problematic
/*
   Working code
*/
}
```

since I need to use some function of Object, compiler need Object declaration. But no way I can give to ComponentManager the implementation of object because Object need Compoment implementation. I can't split my CompomentManager into .h and .cpp so I'm kind of blocked.

Someone have an idea of how I could trick the compiler ? do I will need to change the way my architecture work (only for a function :().

here is the complete problematic chunck of code :

```
for(int i = 0; i <  object.GetScene().GetObjectManager().GetObjectsCount(); i++){
 Object& obj = object.GetScene().GetObjectManager().GetObjects()[i];
 if(&obj != &object){
  if(obj.GetComponentManager().HasComponent<T>()){
   throw Exc("...");
  }
 }
}
```

I have thinked about externalise in a .cpp a function (not templated) which do the job and then I could have include in the .cpp the Object defintion but since I need to call some Function with my templated class argument, I dont think I can work at all.

Subject: Re: C++ templated class referencing each other
Posted by Lance on Sun, 11 Oct 2020 02:25:45 GMT
View Forum Message <> Reply to Message

This hack seems to compiles in CLANG. I am sure there are smarter and more standard ways to serve your purpose.

Object.h

```
#ifndef _TestTemplate_Object_h_
#define _TestTemplate_Object_h_

#include "CompManager.h"
class Object{
    public:
```

```
      Object() : componentManager(*this){
      }

      void SayHello(){ componentManager.CreateComponent<int>(false,100); }
      Object& GetObject(){ return *this;}
     /*
        Many public things
     */
    private:
       ComponentManager componentManager;
};
#include "CompManagerEx.h"
#endif
```

CompManager.h

```
class Object;

class ComponentManager{
    public:
     ComponentManager(Object& _object) : object(_object){}

     template <class T> T& CreateComponent(bool active, int position);

     /*
        Many templated function that force me to write declaration in this .h file
     */
    private:
      Object& object;
};
```

CompManagerEx.h

```
template <class T> T&
ComponentManager::CreateComponent(bool active, int position)
{
 static T t;
/*
   Working code
*/
 Object& obj=object.GetObject();  //This line is problematic
 LOG("Hello inside ComponentManager::CreateComponent(bool active, int position)");
/*
   Working code
*/
```

```
 return t;
}
```

test.cpp

```
#include <Core/Core.h>

#include "Object.h"

using namespace Upp;

CONSOLE_APP_MAIN
{
 Object obj;
 obj.SayHello();
}
```

Subject: Re: C++ templated class referencing each other
Posted by Lance on Sun, 11 Oct 2020 02:36:00 GMT
View Forum Message <> Reply to Message

Notice that I essentially put everything in the same header file (equivalently).

Also, if your can guarantee that ComponentManager Will be the first member variable (because why-not and you should put a static_assert to avoid accidently add some other member variable in front it subsequently), you can probably save the Object& obj;
member variable in the ComponentManager class. Simply define it like:

```
class Object;

class ComponentManager{
    public:
     ComponentManager(){}

     template <class T> T& CreateComponent(bool active, int position);

     /*
        Many templated function that force me to write declaration in this .h file
     */
    private:
      Object& object(){
          return *reinterpret_cast<Object*>(this);
      }

};
```

And change other part of your code accordingly.

PS: If your Object class is virtual, some compiler might put vtable at this. In this case, above hack will be problematic. I will consider to let Object to privately inherit from ComponentManager.

Subject: Re: C++ templated class referencing each other
Posted by Xemuth on Sun, 11 Oct 2020 14:38:54 GMT
View Forum Message <> Reply to Message

Hello Lance,

Thanks for the time you take.

CompManagerEx.h

Did not worked for me. I had compilation error saying it was impossible to define/redefine CreateComponent because it was out of Upp (even if it was in Upp namespace)

Quote:Notice that I essentially put everything in the same header file (equivalently).
It dont work either, it fix the circulare dependencies between my problematic code and Object definition, however it also need the Scene object definition. If I include this definition in object.h file (since ComponentManager is in object.h file) it will lead to the same problem, circular dependencies include (because Scene include ObjectManager which include Object which will include Scene...)

Quote:Also, if your can guarantee that ComponentManager Will be the first member variable (because why-not and you should put a static_assert to avoid accidently add some other member variable in front it subsequently), you can probably save the
Object& obj;

member variable in the ComponentManager class. Simply define it like:
class Object;

```
class ComponentManager{
   public:
    ComponentManager(){}

    template <class T> T& CreateComponent(bool active, int position);

    /*
      Many templated function that force me to write declaration in this .h file
    */
   private:
```

```
    Object& object(){
        return *reinterpret_cast<Object*>(this);
    }

};
```

And change other part of your code accordingly.

I dont really understand this solution, are you meaning inherriting privatly/publicly Object from componentManager ? if yes I'm afraid the problem will remain since circulare reference between Scene and Object.h will appear. Maybe my architecture is bad and should be change to something like Opaque pointer

---

Subject: Re: C++ templated class referencing each other
Posted by Xemuth on Sun, 11 Oct 2020 14:46:21 GMT
View Forum Message <> Reply to Message

OK I retried use of ComponentManagerEx.h and it worked fine, thanks Lance you fixed my probleme.
