Subject: Visibility of objects.
Posted by 281264 on Wed, 15 Sep 2010 11:10:18 GMT
View Forum Message <> Reply to Message

Hi,

current arrangement of header and cpp files? For example, I can control the output in the arrays when a click is made in the GLCtrls; this is easy. But in the case of two user made classes, how can I achieve that?

Cheers,

Javier

File Attachments
1) prueba_OpenGL_DockWindow.7z, downloaded 229 times

Subject: Re: Visibility of objects.
Posted by 281264 on Wed, 15 Sep 2010 18:50:28 GMT
View Forum Message <> Reply to Message

corresponding forum) and not strictly a U++ related query.

Doubt solved, anyhow.

Cheers,

Javier.

Subject: Re: Visibility of objects.
Posted by Mindtraveller on Wed, 15 Sep 2010 20:15:39 GMT
View Forum Message <> Reply to Message

Please be more specific, take smallest possible piece of code and post here this code and your problem.

Subject: Re: Visibility of objects.
Posted by 281264 on Thu, 16 Sep 2010 14:48:44 GMT
View Forum Message <> Reply to Message

Actually it is a question related to C++: if you have different classes, each of them with their .h and

other. Instead of cross-include their .h files, I have included a forward class declaration; in their .cpp I have cross-included the .h file.

I have no idea whether there are better/more elegant/efficient ways to do it. Any thought?

Cheers,

Javier

Subject: Re: Visibility of objects.
Posted by mrjt on Thu, 16 Sep 2010 16:26:00 GMT
View Forum Message <> Reply to Message

The better way to do it is make it so they don't have to see each other at all. I'm guessing you probably want something like a MVC pattern, with the GLCtrls sending their events to a central class that modifies the data-model and then tells the GLCtrls (registered with in some fashion) to refresh themselves.

Subject: Re: Visibility of objects.
Posted by 281264 on Fri, 17 Sep 2010 08:14:51 GMT
View Forum Message <> Reply to Message

I think that what you are suggesting is more suitable for multi-user applications. In my case things are simpler. There are few messages that one GLctrl has to pass the other GLCtrl (views), being Refresh() the most important one. So far I do not see the actual benefit of a more complex configuration.

Accessibility to the geometrical model is simple for I have declared it global (so far the model consist on points and lines in 3D, so I am using simple structs with few variable and Vectors to storage and manipulation). This is fine, for the time being.

With a single central GLCtrl this works fine. But when I implement several views with Docking, things change:

1.- depiction of the geom. in different views as it is being created in the central view is ok and straight forward. It woks fine.

2.- rotation and pan of views is ok.

3.- Zoom with associated window: problems. This works fine for the central view. But when I try to make a zoom (this requires the dynamic drawing of a zoom-window) on a secondary view,

is floating, it works. I have spotted that, in some cases, Paint() function is not called correctly, for a reason that escapes to me. Paint() function is supposed to be called continuously, but when you have several views, how does it work?

4.- The auto-hide does not work. The auto-hide view destroys its hRC and then calls its Resize function, but with the central GLCtrl being the current hRC. Then glviewport function is directed toward the central GLCtrl, causing a deformation in the geometrical model. Ok, I shall not use it.

5.- Another alien thing is the behaviour of Key function for views. For example, when a view is floating (supposedly it is a new TopWindow, in addition to the main TopWindow), its Key method does not work, even if the view has the focus (i.e. it has been clicked). No idea why.

Although Docking is superb, it is evident that to work with Docking and several GLCtrls is not that simple. Also the lack of multithreading penalizes the combination of these features. Perhaps when Docking was not designed taking this scenario into account.

Please, if you have hints let me know them.

Many thanks.

Cheers,

Javier