Subject: Re: Multi Planar Viewer
Posted by David Fanning on Wed, 04 Oct 2006 23:54:28 GMT
View Forum Message <> Reply to Message

Robbie writes:

> My understanding is that this is too complicated to do in iTools and
> that it is far easier just to code a MPV from scratch. Would I be
> correct in this assumption?

Have I mentioned lately how much I love this newsgroup! :-)

Cheers,

David

P.S. I've worked on an application that does everything
you asked for, but using direct graphics objects, and
it was slower than I expected it to be, too. (I haven't
seen the iTools application.) Lots of windows, lots
of images, lots of communication, huge amounts of data.
I think the definitive medical imaging application is
awaiting the next big leap in hardware capabilities.

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Multi Planar Viewer
Posted by Robbie on Thu, 05 Oct 2006 01:14:33 GMT
View Forum Message <> Reply to Message

I've got some more really annoying questions if you want them...

> of images, lots of communication

Speaking of messaging. Which would you think is faster?

IDLitIMessaging::DoOnNotify

or

IDLitData::NotifyDataChange

For heavily nested data, I would exepect there to be an overhead of

calling IDLitData::NotifyDataChange. The change would be propogated to every parent IDLitDataContainer and Observer.

IDLitIMessaging is more particular and makes a single method call to IDLitTool, but iTools must search for the matching identifier to call ::OnNotify. There are a lot of string comparisons in IDLitIMessaging which makes me feel very uncomfortable. I also don't know the etiqutte of defining new IDLitIMessaging "Message Strings". IDLitIMessaging feels like spaghetti but I guess it gets the job done.

Robbie

Subject: Re: Multi Planar Viewer
Posted by David Fanning on Thu, 05 Oct 2006 01:40:35 GMT
View Forum Message <> Reply to Message

Robbie writes:

> Speaking of messaging. Which would you think is faster?
>
> IDLitIMessaging::DoOnNotify
>
> or
>
> IDLitData::NotifyDataChange

I haven't the faintest idea. I didn't even know
they existed. Too old to learn about them, probably. :-)

> IDLitIMessaging is more particular and makes a single method call to
> IDLitTool, but iTools must search for the matching identifier to call
> ::OnNotify. There are a lot of string comparisons in IDLitIMessaging
> which makes me feel very uncomfortable. I also don't know the etiqutte
> of defining new IDLitIMessaging "Message Strings". IDLitIMessaging
> feels like spaghetti but I guess it gets the job done.

I implemented messaging the way JD recommended (I think it
was him, there were a lot of ideas swirling around at the
time). Every object has the ability to "register" interest
in a message with every other object. When an object does
something that could potentially justify a message, it
looks in its "objects to notify" bin and notifies those
objects who have registered for a particular "message".

For example, a colorbar object sends a "COLORS_CHANGED"
message whenever it modifies the color vectors. Images
that are interested in knowing about such a change (so

they can re-draw themselves with the new colors, for example)
register this interest with the colorbar object. Perhaps
four images are interested, then the colorbar object sends
this "COLORS_CHANGED" message to four image objects, and
four draw widgets are updated when the images draw themselves.

The message sending is very fast. The image re-draw is
relatively slow. If you have 50 mri images on the display
at once, it can take a second or so to see all the
images update themselves. This is primarily because we
have to resize them to display them, since we don't
know ahead of time what size image we are going to need.
This depends on what the user wants to see in each window, etc.

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Multi Planar Viewer
Posted by Robbie on Thu, 05 Oct 2006 03:17:42 GMT
View Forum Message <> Reply to Message

> The message sending is very fast. The image re-draw is
> relatively slow

Indeed, it seems that too many calls to IDLgrWindow::draw is the
bottleneck today.

One thing that I've found really handy about OO is that my rendering
classes report the redraw time. For this particular application I have
a redraw time of around 200ms. I guess this makes messaging semantics
somewhat irrelevant.

My speed problem is definately too many redraws. I just implemented a
"redraw flag". A call to ::redaw just flips a flag. My event processing
routine checks for flipped flags at each iteration of the event loop,
which guaruntees that there will be only one redraw per widget_draw per
event.

I consider this solution a bit of a hack, because I should be able to
program visualisations such that ::redraw is only called when it is
absolutely necessary.

I'm still squeamish about messaging between visualisations. I've got 14 visualisations with various interdependancies. A single message could trigger 2^13 messages if every dependancy was registered. I guess that's a *good* reason not to use iTools :-)