Subject: .dlm and c++ object
Posted by Rick Towler on Mon, 25 Feb 2002 18:06:27 GMT
View Forum Message <> Reply to Message

In my never ending quest to build efficient and intuitive user interfaces for my IDL apps I have embarked on my second round of .dlm programming adventure. In this round, I would like to use the window's directInput routines to get keyboard, mouse(with z roller and extra buttons) and joystick input. If successful this will be released into the wild so hopefully this will motivate some.

In very general terms, using directInput involves creating an instance of a DI object, setting props, acquiring it, polling it, and then destroying it. With my very limited knowledge of c++ and .dlm programming I envision writing an IDL based object that encapsulates the c++ routines. A gross example using the mouse is below.

Aside from everything I may be missing (which I know is a lot), will my c++ object continue to exist outside the scope of the called .dlm routine (I would assume so). If so, how do I keep track of it? Is it as simple as returning the object reference back to idl? If so, what data type are c++ object references? If I am on the wrong track, any pointers on what would be the right track?

:/

Thanks

-Rick

```
function mouse::init

    ;create direct input object.  On the c++ side the DI object
    ;is created and the object reference is returned to IDL

    self.cppobj = di_mouse_init()

end


function mouse::poll

    ;pass the c++ object reference to the DI routine to poll the
    ;mouse.  Return a struct containing the mouse state

    mouse_state = di_mouse_poll(self.cppobj)
```

```
    return, mouse_state

end


pro mouse::cleanup

    ;call some routine that cleans up our mouse object on the c++
    ;side

     di_mouse_cleanup

end


pro mouse__define

    struct={mouse, $
            cppobj : ?? }

end
```
