Object Life Cycles and Garbage Collection

One of the more difficult issues to deal with was understanding the "life cycle" of FOX objects (that is, the actual C++ objects) and their relationship to the associated Ruby instances. Understanding this relationship is critical when dealing with Ruby's garbage collector, to ensure that objects are disposed of at the right time.

For our purposes, we can divide the set of all objects in an FXRuby program into two groups, those objects that Ruby "owns" and those that it doesn't. The first group (the "owned" objects) includes those that you create explicitly from Ruby code. This is usually done by calling a class' `new` method, e.g.

```
myIcon = FXPNGIcon.new(myApp, File.open("icon.png", "rb").read)
myButton = FXButton.new(parentWin, "Hello, World!", myIcon)
```

It's important to keep in mind that when you create an object like this you're not only creating the Ruby instance part (i.e. whatever overhead is usually associated with a Ruby instance) but a C++ FOX object as well. Because we created these objects, we would reasonably expect them to be destroyed when they are garbage-collected so that our programs don't leak memory.

The other group of objects (those not owned by Ruby) are those returned from most class instance methods; they are references to already- existing objects. For example, `FXStatusBar#statusline` returns a reference to the status bar's enclosed status line instance.

GL Objects

A C++ `FXGLGroup` object owns all of the `FXGLObject` objects it "contains". In other words, when that `FXGLGroup` object is destroyed, it will also destroy all of the `FXGLObject` objects for which it holds pointers.

In order to keep track of *which* GL objects have been added to an `FXGLGroup`, all of the FXRuby C++ classes derived from `FXGLObject` have a boolean member variable *`owned`* that indicates whether this object is "owned" or not. Until an `FXGLObject` object is added to a group, this member variable should stay false.