GammaLib - Feature #119

Perform automatic dynamic typecasting in Python interface

03/09/2012 10:28 PM - Knödlseder Jürgen

Status:	Closed	**Start date:**	03/09/2012
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	100%
Category:		**Estimated time:**	36.00 hours
Target version:	00-09-00		

Description

The SWIG Python interface always returns a base class pointer even if we deal with a derived class. This seems to be a difficult problem to solve in SWIG, but there seem to be possibilities around, such as using the %feature directive.

To solve this problem we should:

- investigate possible solutions by writing sample code
- implement best solution

Subtasks:

Action # 288: Implement solution for derived classes of GObservation	**Closed**
Action # 287: Investigate possible solutions for type casting in SWIG	**Closed**

History

#1 - 03/09/2012 10:48 PM - Knödlseder Jürgen

I checked the %factory directive, but this requires that all derived types are known beforehand. This is not practical for our case as we want the library to be extendable, and we don't want to hard code derived class types into the base class.

Note that in typemaps/swigtype.swg there exists the following typemap which may be useful:

```
/* DYNAMIC typemap */
%typemap(out,noblock=1) SWIGTYPE *DYNAMIC, SWIGTYPE &DYNAMIC {
%set_output(SWIG_NewPointerObj(%as_voidptr($1), SWIG_TypeDynamicCast($descriptor, %as_voidptrptr(&$1)), $owner | %newpointer_flags));
}
```

Here a couple of links that may be useful:
http://comments.gmane.org/gmane.comp.programming.swig.devel/21148
http://osdir.com/ml/programming.swig/2003-09/msg00066.html
http://old.nabble.com/Status-of-SWIG_TypeDynamicCast--td24364473.html
http://stackoverflow.com/questions/3921294/how-do-i-down-cast-a-c-object-from-a-python-swig-wrapper
http://altdevblogaday.com/2011/08/10/swig-casting-revisited/
http://osdir.com/ml/programming.swig/2004-08/msg00057.html

An interesting option could be the use of:

```
obj = SWIG_NewPointerObj(widget, SWIG_TypeQuery(myTypeString), 0)
```

where myTypeString is a string representation of the type. We could add a generic type method that returns the class type as string, which we then could use for type casting. Maybe this can be done at SWIG interface level only, avoiding to pollute the C++ classes?

Maybe we can add a method to each derived class (here GCTAObservation) of the type:

```
%{
swig_type_info* swig_type(void) {
return SWIGTYPE_p_GCTAObservation;
}
%}
```

and then define this as a pure virtual method of the base class. We may then use

```
obj = SWIG_NewPointerObj(base, base->swig_type(), 0)
```

so that the object base gets downcasted.

Here a similar code fragment from http://www.dabeaz.com/cgi-bin/wiki.pl?SwigFaqCPlusPlusFactoryIntrospection:

```
%typemap(out) Base* Base::create {
swig_type_info* info = objectIDToSWIGInfo($1->objectID());
if (!info)
SWIG_exception_fail(SWIG_RuntimeError, "Fatal error: objectIDToSWIGInfo returned null.");
$result = SWIG_NewPointerObj(SWIG_as_voidptr($1), info, SWIG_POINTER_OWN);
};
```

Here another note on using SWIG_TypeQuery:

Yes that should work. One caveat: SWIG_TypeQuery() is pretty slow. To speed up conversions, it may be better to call it once. For example:

```
static swig_type_info *mytype_descriptor = 0;
if (!mytype_descriptor) {
mytype_descriptor = SWIG_TypeQuery("mytype *");
}
obj = SWIG_NewPointObj(widget, mytype_descriptor, 0);
```

#2 - 07/08/2012 10:12 PM - Knödlseder Jürgen

- Target version set to Stage Jean-Baptiste Cayrou

#3 - 07/28/2012 12:53 AM - Knödlseder Jürgen

- Target version deleted (Stage Jean-Baptiste Cayrou)

#4 - 09/10/2014 12:15 PM - Knödlseder Jürgen

- Status changed from New to Feedback

- Target version set to 00-09-00

Following #287, full automatic type casting has now been implemented.

#5 - 10/30/2014 12:10 PM - Knödlseder Jürgen

- Status changed from Feedback to Closed