Subject: How to prevent proliferation of leftover objects when NG plots are included in unit tests.
Posted by Paul Van Delst[1] on Tue, 14 Aug 2012 19:56:46 GMT
View Forum Message <> Reply to Message

Hello,

I'm using Michael Galloy's unit test framework for testing (it works great, btw!). Anyhoo, during the unit tests a
number of objects are created, e.g.

```
<Expression>    OBJREF    = <ObjHeapVar1(MGUTCOMPOUNDRUNNER)>  refcount=5
<Expression>    OBJREF    = <ObjHeapVar2(MGUTTESTSUITE)>  refcount=4
<Expression>    OBJREF    = <ObjHeapVar3(IDL_CONTAINER)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar4(MGUTCLIRUNNER)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar6(OSRF_UT)>  refcount=6
```

(the "osrf_ut" object is my unit test)

At the end of the tests all of the objects are destroyed:

```
IDL> mgunit, 'osrf_ut'
% Compiled module: MGUNIT.
% Compiled module: MGUTCOMPOUNDRUNNER__DEFINE.
% Compiled module: MGUTTESTRUNNER__DEFINE.
% Compiled module: MGUTTESTSUITE__DEFINE.
% Compiled module: MGUTCLIRUNNER__DEFINE.
"All tests" test suite starting (1 test suite/case, 17 tests)
  "osrf_ut" test case starting (17 tests)

...etc...

  Results: 17 / 17 tests passed, 0 skipped
Results: 17 / 17 tests passed, 0 skipped
IDL> help, obj_valid()
<Expression>    OBJREF    = <NullObject>
```

Lovely. I started with no objects and I ended the same way. Nice and clean.

However, when I include a PLOT() function call in a unit test (just to have a looksee at the pretend data I create), I
have thousands of leftover objects, all NG related:

```
IDL> help, obj_valid()
<Expression>    OBJREF    = Array[4796]
```

Printing out the first 11 objects:

```
IDL> for i=0,10 do help, (obj_valid())[i],/full
<Expression>    OBJREF    = <ObjHeapVar175(IDLITSYSTEM)>  refcount=154
<Expression>    OBJREF    = <ObjHeapVar176(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar177(IDL_CONTAINER)>  refcount=1
<Expression>    OBJREF    = <ObjHeapVar179(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar181(IDLITCONTAINER)>  refcount=4
<Expression>    OBJREF    = <ObjHeapVar182(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar183(IDL_CONTAINER)>  refcount=1
<Expression>    OBJREF    = <ObjHeapVar185(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar188(IDLITCONTAINER)>  refcount=3
<Expression>    OBJREF    = <ObjHeapVar189(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar190(IDL_CONTAINER)>  refcount=1
```

When I kill the plot window, I still have a great many leftovers:

```
IDL> help, obj_valid()
<Expression>    OBJREF    = Array[1395]
IDL> for i=0,10 do help, (obj_valid())[i],/full
<Expression>    OBJREF    = <ObjHeapVar175(IDLITSYSTEM)>  refcount=151
<Expression>    OBJREF    = <ObjHeapVar176(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar177(IDL_CONTAINER)>  refcount=1
<Expression>    OBJREF    = <ObjHeapVar179(IDLITPROPERTYDESCRIPTOR)>  refcount=2
<Expression>    OBJREF    = <ObjHeapVar181(IDLITCONTAINER)>  refcount=3
....etc....
```

My question is: How do I prevent this proliferation of NG plot objects? Or, at the very least, how do I safely delete
them? I can't do a

```
IDL> OBJ_DESTROY, OBJ_VALID()
IDL> help, obj_valid()
<Expression>    OBJREF    = <NullObject>
```

in case there are other objects that I do want to keep around.

Any info, tips, tricks, appreciated.

cheers,

paulv

p.s. What sort of software could create so many objects (for what is a relatively simple line plot) and then not clean
up after itself? Argh.

p.p.s. :

IDL> print, !version
{ x86 linux unix linux 8.1 Mar 9 2011   32   64}