Subject: FG question: retrieve points within polygon
Posted by Helder Marchetto on Thu, 04 Dec 2014 09:50:38 GMT
View Forum Message <> Reply to Message

Hi,
I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
So here is a basic example that states what I want to do:

```
;first generate the graphics
img = dist(600)
w = window(dimensions=[500,500])
im = image(img, current=w)
pl =  polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
;make some changes to the polygon
pl.rotate, 12

;now extract the mean value of the points of the image that are inside the polygon

pl->getData, xx, yy
o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
mask = o->ComputeMask(dimensions=[600,600])
obj_destroy, o
pts = where(mask, cnt)
meanVal = mean(img[pts])
print, 'the mean value inside the polygon is ', meanVal
```

So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.

Any better way to do this? I couldn't find any FG method to get such info.

Thanks,
Helder

Subject: Re: FG question: retrieve points within polygon
Posted by lecacheux.alain on Thu, 04 Dec 2014 13:51:27 GMT
View Forum Message <> Reply to Message

On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
> Hi,
> I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.

> So here is a basic example that states what I want to do:
>
> ;first generate the graphics
> img = dist(600)
> w = window(dimensions=[500,500])
> im = image(img, current=w)
> pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
> ;make some changes to the polygon
> pl.rotate, 12
>
> ;now extract the mean value of the points of the image that are inside the polygon
>
> pl->getData, xx, yy
> o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
> mask = o->ComputeMask(dimensions=[600,600])
> obj_destroy, o
> pts = where(mask, cnt)
> meanVal = mean(img[pts])
> print, 'the mean value inside the polygon is ', meanVal
>
>
>
> So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
> Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>
> Any better way to do this? I couldn't find any FG method to get such info.
>
> Thanks,
> Helder

If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.
alx.

Subject: Re: FG question: retrieve points within polygon
Posted by Helder Marchetto on Thu, 04 Dec 2014 14:32:24 GMT
View Forum Message <> Reply to Message

On Thursday, December 4, 2014 2:51:28 PM UTC+1, alx wrote:
> On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
>> Hi,
>> I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
>> So here is a basic example that states what I want to do:

>>
>> ;first generate the graphics
>> img = dist(600)
>> w = window(dimensions=[500,500])
>> im = image(img, current=w)
>> pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
>> ;make some changes to the polygon
>> pl.rotate, 12
>>
>> ;now extract the mean value of the points of the image that are inside the polygon
>>
>> pl->getData, xx, yy
>> o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
>> mask = o->ComputeMask(dimensions=[600,600])
>> obj_destroy, o
>> pts = where(mask, cnt)
>> meanVal = mean(img[pts])
>> print, 'the mean value inside the polygon is ', meanVal
>>
>>
>>
>> So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
>> Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>>
>> Any better way to do this? I couldn't find any FG method to get such info.
>>
>> Thanks,
>> Helder
>
> If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
> Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.
> alx.

Hi Alx,
I wanted to avoid doing myself the calculation, but even trying I found that it is not that possible. It seems like the ellipse() function simply generates a polygon() function. Once created, I could not retrieve the center or radius (major or minor) and cannot therefore compute using the ellipse equation. What I can do is use the undocumented getData method as I would for a polygon and then proceed as if it were a polygon.

Still, a mask method would be a nice add to the FG.

Cheers,
Helder

Subject: Re: FG question: retrieve points within polygon
Posted by lecacheux.alain on Thu, 04 Dec 2014 15:24:08 GMT
View Forum Message <> Reply to Message

On Thursday, December 4, 2014 3:32:26 PM UTC+1, Helder wrote:
> On Thursday, December 4, 2014 2:51:28 PM UTC+1, alx wrote:
>> On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
>>> Hi,
>>> I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
>>> So here is a basic example that states what I want to do:
>>>
>>> ;first generate the graphics
>>> img = dist(600)
>>> w = window(dimensions=[500,500])
>>> im = image(img, current=w)
>>> pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
>>> ;make some changes to the polygon
>>> pl.rotate, 12
>>>
>>> ;now extract the mean value of the points of the image that are inside the polygon
>>>
>>> pl->getData, xx, yy
>>> o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
>>> mask = o->ComputeMask(dimensions=[600,600])
>>> obj_destroy, o
>>> pts = where(mask, cnt)
>>> meanVal = mean(img[pts])
>>> print, 'the mean value inside the polygon is ', meanVal
>>>
>>>
>>>
>>> So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
>>> Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>>>
>>> Any better way to do this? I couldn't find any FG method to get such info.
>>>
>>> Thanks,
>>> Helder
>>
>> If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
>> Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.
>> alx.
>
> Hi Alx,

> I wanted to avoid doing myself the calculation, but even trying I found that it is not that possible. It seems like the ellipse() function simply generates a polygon() function. Once created, I could not retrieve the center or radius (major or minor) and cannot therefore compute using the ellipse equation. What I can do is use the undocumented getData method as I would for a polygon and then proceed as if it were a polygon.
>
> Still, a mask method would be a nice add to the FG.
>
> Cheers,
> Helder

Hi Helder,
If you draw the ellipse by calling the ELLIPSE function, you should know everything. If you draw it by hand, you can get the rectangle containing the ellipse by doing (after selecting it):
 pos = GetWindows(/CURRENT).GetSelect().Position
then, center and axis lengthes.
alx.

Subject: Re: FG question: retrieve points within polygon
Posted by Helder Marchetto on Thu, 04 Dec 2014 15:41:24 GMT
View Forum Message <> Reply to Message

On Thursday, December 4, 2014 4:24:11 PM UTC+1, alx wrote:
> On Thursday, December 4, 2014 3:32:26 PM UTC+1, Helder wrote:
>> On Thursday, December 4, 2014 2:51:28 PM UTC+1, alx wrote:
>>> On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
>>>> Hi,
>>>> I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
>>>> So here is a basic example that states what I want to do:
>>>>
>>>> ;first generate the graphics
>>>> img = dist(600)
>>>> w = window(dimensions=[500,500])
>>>> im = image(img, current=w)
>>>> pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
>>>> ;make some changes to the polygon
>>>> pl.rotate, 12
>>>>
>>>> ;now extract the mean value of the points of the image that are inside the polygon
>>>>
>>>> pl->getData, xx, yy
>>>> o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
>>>> mask = o->ComputeMask(dimensions=[600,600])
>>>> obj_destroy, o
>>>> pts = where(mask, cnt)
>>>> meanVal = mean(img[pts])

>>>> print, 'the mean value inside the polygon is ', meanVal
>>>>
>>>>
>>>>
>>>> So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
>>>> Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>>>>
>>>> Any better way to do this? I couldn't find any FG method to get such info.
>>>>
>>>> Thanks,
>>>> Helder
>>>
>>> If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
>>> Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.
>>> alx.
>>
>> Hi Alx,
>> I wanted to avoid doing myself the calculation, but even trying I found that it is not that possible. It seems like the ellipse() function simply generates a polygon() function. Once created, I could not retrieve the center or radius (major or minor) and cannot therefore compute using the ellipse equation. What I can do is use the undocumented getData method as I would for a polygon and then proceed as if it were a polygon.
>>
>> Still, a mask method would be a nice add to the FG.
>>
>> Cheers,
>> Helder
>
> Hi Helder,
> If you draw the ellipse by calling the ELLIPSE function, you should know everything. If you draw it by hand, you can get the rectangle containing the ellipse by doing (after selecting it):
> pos = GetWindows(/CURRENT).GetSelect().Position
> then, center and axis lengthes.
> alx.

Hi Alx,
thanks.
The idea is that I have a widget window where I move and change my ellipse as I wish.
Say I start with this:

```
img = dist(600)
w = window(dimensions=[500,500])
im = image(img, current=w)
el = ellipse(300,300,/data, minor=100, major=150, target=im)
```

and then I modify the ellipse with the mouse (make it bigger, rotate, stretch,...).
The changes that will be made I don't know "a priori", so by stretching and rotating I end up with a completely different ellipse.

I can get the position (as you said) with

pos = el.position

and that gives me always the center as [(pos[0]+pos[2])/2.0,(pos[1]+pos[3])/2.0]. But I cannot retrieve the rotation or the axis and, as far as I understand, there are infinite ellipses that can fit inside the rectangle defined by the position property that have a different set of rotation/axis.

I think that I will have to stick with the polygon type solution using IDLanROI.

Thanks,
Helder

Subject: Re: FG question: retrieve points within polygon
Posted by Jim Pendleton on Thu, 04 Dec 2014 23:58:45 GMT

View Forum Message <> Reply to Message

On Thursday, December 4, 2014 8:41:26 AM UTC-7, Helder wrote:
> On Thursday, December 4, 2014 4:24:11 PM UTC+1, alx wrote:
>> On Thursday, December 4, 2014 3:32:26 PM UTC+1, Helder wrote:
>>> On Thursday, December 4, 2014 2:51:28 PM UTC+1, alx wrote:
>>>> On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
>>>> > Hi,
>>>> > I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
>>>> > So here is a basic example that states what I want to do:
>>>> >
>>>> > ;first generate the graphics
>>>> > img = dist(600)
>>>> > w = window(dimensions=[500,500])
>>>> > im = image(img, current=w)
>>>> > pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
>>>> > ;make some changes to the polygon
>>>> > pl.rotate, 12
>>>> >
>>>> > ;now extract the mean value of the points of the image that are inside the polygon
>>>> >
>>>> > pl->getData, xx, yy
>>>> > o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
>>>> > mask = o->ComputeMask(dimensions=[600,600])
>>>> > obj_destroy, o
>>>> > pts = where(mask, cnt)

>>>> > meanVal = mean(img[pts])
>>>> > print, 'the mean value inside the polygon is ', meanVal
>>>> >
>>>> >
>>>> >
>>>> > So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
>>>> > Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>>>> >
>>>> > Any better way to do this? I couldn't find any FG method to get such info.
>>>> >
>>>> > Thanks,
>>>> > Helder
>>>>
>>>> If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
>>>> Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.
>>>> alx.
>>>
>>> Hi Alx,
>>> I wanted to avoid doing myself the calculation, but even trying I found that it is not that possible. It seems like the ellipse() function simply generates a polygon() function. Once created, I could not retrieve the center or radius (major or minor) and cannot therefore compute using the ellipse equation. What I can do is use the undocumented getData method as I would for a polygon and then proceed as if it were a polygon.
>>>
>>> Still, a mask method would be a nice add to the FG.
>>>
>>> Cheers,
>>> Helder
>>
>> Hi Helder,
>> If you draw the ellipse by calling the ELLIPSE function, you should know everything. If you draw it by hand, you can get the rectangle containing the ellipse by doing (after selecting it):
>>   pos = GetWindows(/CURRENT).GetSelect().Position
>> then, center and axis lengthes.
>> alx.
>
> Hi Alx,
> thanks.
> The idea is that I have a widget window where I move and change my ellipse as I wish.
> Say I start with this:
>
> img = dist(600)
> w = window(dimensions=[500,500])
> im = image(img, current=w)

> el = ellipse(300,300,/data, minor=100, major=150, target=im)
>
> and then I modify the ellipse with the mouse (make it bigger, rotate, stretch,...).
> The changes that will be made I don't know "a priori", so by stretching and rotating I end up with a completely different ellipse.
>
> I can get the position (as you said) with
>
> pos = el.position
>
> and that gives me always the center as [(pos[0]+pos[2])/2.0,(pos[1]+pos[3])/2.0]. But I cannot retrieve the rotation or the axis and, as far as I understand, there are infinite ellipses that can fit inside the rectangle defined by the position property that have a different set of rotation/axis.
>
> I think that I will have to stick with the polygon type solution using IDLanROI.
>
> Thanks,
> Helder

How adventurous do you want to be? The transformation information is stored in the container of the polygon. The coordinates of the raw data don't actually change after the polygon is created, I would guess.

Let's say you've started a new IDL session and executed the above commands then you've selected and resized or repositioned it interactively. (I'm using IDL 8.4 so YMMV.)

Which object that was created is an IDLgrPolygon?

```
IDL> oall = obj_valid(/cast)
IDL> print, where(obj_isa(oall, 'idlgrpolygon'))
         262         550         568         586         604         621         640         657         676        1894
        2089
```

We have a bunch. One is likely the "main" polygon and the others the "selection" items for resizing and so forth.

Which one is the "main" polygon? Let's guess the first and change its color.

```
IDL> oall[262]->setproperty, color = [0, 255, 0] & el.refresh
```

Lucky guess.

Next, let's get its container, which is by definition a model.

```
IDL> oall[262]->getproperty, parent = omodel
```

Check out the transformation matrix.

```
IDL> omodel->getproperty, transform = t
IDL> print, t
     0.51712623     0.00000000     0.00000000      55.899910
     0.00000000      1.3790130     0.00000000     -120.23054
     0.00000000     0.00000000      1.0000000     0.00000000
     0.00000000     0.00000000     0.00000000      1.0000000
```

Of course, you'll have different values here.

Let's send the ellipse back "home":

```
IDL> t = identity(4)
IDL> omodel->setproperty, transform = t & el.refresh
```

Teasing out the rotation, translation, and scale from the quaternion is an exercise left for the reader.

In function graphics there's a whole lot of cruft, er, framework, that sits over the top of the most basic graphics atoms and models and knowing how the basic, documented, objects work will go a long way to helping you get closer to what you want.

If the new graphics API doesn't expose exactly the features you want, there are ways to get and modify the data and behaviors, with the possible side effect of producing inconsistent internal states, particularly during interactive use.

Run with scissors.

Jim P.
"I work for Exelis VIS, but these are my own observations"

Subject: Re: FG question: retrieve points within polygon
Posted by Helder Marchetto on Fri, 05 Dec 2014 09:16:12 GMT
View Forum Message <> Reply to Message

On Friday, December 5, 2014 12:58:48 AM UTC+1, Jim P wrote:
> On Thursday, December 4, 2014 8:41:26 AM UTC-7, Helder wrote:
>> On Thursday, December 4, 2014 4:24:11 PM UTC+1, alx wrote:
>>> On Thursday, December 4, 2014 3:32:26 PM UTC+1, Helder wrote:
>>>> On Thursday, December 4, 2014 2:51:28 PM UTC+1, alx wrote:
>>>> > On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
>>>> > > Hi,
>>>> > > I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
>>>> > > So here is a basic example that states what I want to do:
>>>> > >
>>>> > > ;first generate the graphics
>>>> > > img = dist(600)

>>>> > > w = window(dimensions=[500,500])
>>>> > > im = image(img, current=w)
>>>> > > pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
>>>> > > ;make some changes to the polygon
>>>> > > pl.rotate, 12
>>>> > >
>>>> > > ;now extract the mean value of the points of the image that are inside the polygon
>>>> > >
>>>> > > pl->getData, xx, yy
>>>> > > o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
>>>> > > mask = o->ComputeMask(dimensions=[600,600])
>>>> > > obj_destroy, o
>>>> > > pts = where(mask, cnt)
>>>> > > meanVal = mean(img[pts])
>>>> > > print, 'the mean value inside the polygon is ', meanVal
>>>> > >
>>>> > >
>>>> > >
>>>> > > So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
>>>> > > Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>>>> > >
>>>> > > Any better way to do this? I couldn't find any FG method to get such info.
>>>> > >
>>>> > > Thanks,
>>>> > > Helder
>>>> >
>>>> > If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
>>>> > Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.
>>>> > alx.
>>>>
>>>> Hi Alx,
>>>> I wanted to avoid doing myself the calculation, but even trying I found that it is not that possible. It seems like the ellipse() function simply generates a polygon() function. Once created, I could not retrieve the center or radius (major or minor) and cannot therefore compute using the ellipse equation. What I can do is use the undocumented getData method as I would for a polygon and then proceed as if it were a polygon.
>>>>
>>>> Still, a mask method would be a nice add to the FG.
>>>>
>>>> Cheers,
>>>> Helder
>>>
>>> Hi Helder,
>>> If you draw the ellipse by calling the ELLIPSE function, you should know everything. If you

draw it by hand, you can get the rectangle containing the ellipse by doing (after selecting it):
>>>   pos = GetWindows(/CURRENT).GetSelect().Position
>>> then, center and axis lengthes.
>>> alx.
>>
>> Hi Alx,
>> thanks.
>> The idea is that I have a widget window where I move and change my ellipse as I wish.
>> Say I start with this:
>>
>> img = dist(600)
>> w = window(dimensions=[500,500])
>> im = image(img, current=w)
>> el = ellipse(300,300,/data, minor=100, major=150, target=im)
>>
>> and then I modify the ellipse with the mouse (make it bigger, rotate, stretch,...).
>> The changes that will be made I don't know "a priori", so by stretching and rotating I end up with a completely different ellipse.
>>
>> I can get the position (as you said) with
>>
>> pos = el.position
>>
>> and that gives me always the center as [(pos[0]+pos[2])/2.0,(pos[1]+pos[3])/2.0]. But I cannot retrieve the rotation or the axis and, as far as I understand, there are infinite ellipses that can fit inside the rectangle defined by the position property that have a different set of rotation/axis.
>>
>> I think that I will have to stick with the polygon type solution using IDLanROI.
>>
>> Thanks,
>> Helder
>
> How adventurous do you want to be? The transformation information is stored in the container of the polygon. The coordinates of the raw data don't actually change after the polygon is created, I would guess.
>
> Let's say you've started a new IDL session and executed the above commands then you've selected and resized or repositioned it interactively. (I'm using IDL 8.4 so YMMV.)
>
> Which object that was created is an IDLgrPolygon?
>
```
>  IDL> oall = obj_valid(/cast)
>  IDL> print, where(obj_isa(oall, 'idlgrpolygon'))
>          262        550        568        586        604        621        640        657        676       1894
      2089
```
>
> We have a bunch. One is likely the "main" polygon and the others the "selection" items for resizing and so forth.

>
> Which one is the "main" polygon? Let's guess the first and change its color.
>
> IDL> oall[262]->setproperty, color = [0, 255, 0] & el.refresh
>
> Lucky guess.
>
> Next, let's get its container, which is by definition a model.
>
> IDL> oall[262]->getproperty, parent = omodel
>
> Check out the transformation matrix.
>
```
> IDL> omodel->getproperty, transform = t
> IDL> print, t
>      0.51712623     0.00000000     0.00000000      55.899910
>      0.00000000      1.3790130     0.00000000     -120.23054
>      0.00000000     0.00000000      1.0000000     0.00000000
>      0.00000000     0.00000000     0.00000000      1.0000000
```
>
> Of course, you'll have different values here.
>
> Let's send the ellipse back "home":
>
> IDL> t = identity(4)
> IDL> omodel->setproperty, transform = t & el.refresh
>
> Teasing out the rotation, translation, and scale from the quaternion is an exercise left for the reader.
>
> In function graphics there's a whole lot of cruft, er, framework, that sits over the top of the most basic graphics atoms and models and knowing how the basic, documented, objects work will go a long way to helping you get closer to what you want.
>
> If the new graphics API doesn't expose exactly the features you want, there are ways to get and modify the data and behaviors, with the possible side effect of producing inconsistent internal states, particularly during interactive use.
>
> Run with scissors.
>
> Jim P.
> "I work for Exelis VIS, but these are my own observations"

Hi Jim,
thanks for your insights into FG. To me most (=all) of what's happening behind the scenes is a mystery, therefore such demos are enlightening.
With that said, my program is somewhat complicated and it might have to deal with more instances of ellipses and things might get nasty I'm not sure that going through all this trouble

would actually give me an advantage over getting the mask using the point coordinates used to draw the ellipse in the polygon object and feeding them in an IDLanROI object.

I like anyway to fiddle around with these things, therefore I tried to get your code running and got stuck pretty soon. It seems like I can't change the color of the "main" polygon.
Here is the code I've tested and I've put a break in the for loop to check the color of the ellipse.

```
pro testEllipse
img = dist(600)
w = window(dimensions=[500,500])
im = image(img, current=w)
el = ellipse(300,300,'r3', /data, minor=100, major=150, target=im, fill_background=0)
el->rotate, 15
el->translate, 40,60
oall = obj_valid(/cast)
polys = where(obj_isa(oall, 'idlgrpolygon'))
for i=0,n_elements(polys)-1 do begin
   oall[polys[i]]->getproperty, color = c
   print, 'starting color = ', c
   oall[polys[i]]->setproperty, color = [0,255,0]
   el.refresh
   ;insert break in the following line to check if the ellipse has changed color
   print, 'color set'
   oall[polys[i]]->setproperty, color = c
   el.refresh
endfor
end
```

It so happens that the ellipse stays as it is. Interestingly, the colors I get from the objects I scan through are:
3 times [255,255,255]
8 times [30,144,255]
I get no red ([255,0,0]).

Since you have more insight than I do, I'll ask you the following:

What is the probability that in the near future (<1year) Exelis implements a mask for objects like Polygons and Ellipses?

I doesn't seem to me like a too complicated thing to implement if you have access (and understanding) to the lower layers of IDL FG and access to the mask method used for IDLanROI.

Thanks,
Helder

Subject: Re: FG question: retrieve points within polygon

Posted by Jim Pendleton on Fri, 05 Dec 2014 16:23:27 GMT
View Forum Message <> Reply to Message

On Friday, December 5, 2014 2:16:16 AM UTC-7, Helder wrote:
> On Friday, December 5, 2014 12:58:48 AM UTC+1, Jim P wrote:
>> On Thursday, December 4, 2014 8:41:26 AM UTC-7, Helder wrote:
>>> On Thursday, December 4, 2014 4:24:11 PM UTC+1, alx wrote:
>>>> On Thursday, December 4, 2014 3:32:26 PM UTC+1, Helder wrote:
>>>> > On Thursday, December 4, 2014 2:51:28 PM UTC+1, alx wrote:
>>>> > > On Thursday, December 4, 2014 10:50:42 AM UTC+1, Helder wrote:
>>>> > > > Hi,
>>>> > > > I'm looking for an easier way to get the indices inside a polygon or ellipse created in function graphics.
>>>> > > > So here is a basic example that states what I want to do:
>>>> > > >
>>>> > > > ;first generate the graphics
>>>> > > > img = dist(600)
>>>> > > > w = window(dimensions=[500,500])
>>>> > > > im = image(img, current=w)
>>>> > > > pl = polygon([0.25,0.75,0.75,0.25],[0.25,0.25,0.75,0.75],/norm,ta rget=im)
>>>> > > > ;make some changes to the polygon
>>>> > > > pl.rotate, 12
>>>> > > >
>>>> > > > ;now extract the mean value of the points of the image that are inside the polygon
>>>> > > >
>>>> > > > pl->getData, xx, yy
>>>> > > > o = obj_new('idlanroi', xx*600d, yy*600d, /double, type=2)
>>>> > > > mask = o->ComputeMask(dimensions=[600,600])
>>>> > > > obj_destroy, o
>>>> > > > pts = where(mask, cnt)
>>>> > > > meanVal = mean(img[pts])
>>>> > > > print, 'the mean value inside the polygon is ', meanVal
>>>> > > >
>>>> > > >
>>>> > > >
>>>> > > > So this method works fine. It's maybe not the most obvious, but works. Now the question is... How do I get the same result for an ellipse?
>>>> > > > Of course I could calculate the perimeter points of the ellipse and use the same method as above, but that would not really be... well ... cool.
>>>> > > >
>>>> > > > Any better way to do this? I couldn't find any FG method to get such info.
>>>> > > >
>>>> > > > Thanks,
>>>> > > > Helder
>>>> > >
>>>> > > If you could plot an ellipse with FG, you know its equation from the parameters (center, axes, orientation) you have given in the call. Let it be F(x,y)=0.
>>>> > > Then the indices of the (x,y) points inside the ellipse are those for which F(x,y) is strictly negative.

>>>> > > alx.
>>>> >
>>>> > Hi Alx,
>>>> > I wanted to avoid doing myself the calculation, but even trying I found that it is not that possible. It seems like the ellipse() function simply generates a polygon() function. Once created, I could not retrieve the center or radius (major or minor) and cannot therefore compute using the ellipse equation. What I can do is use the undocumented getData method as I would for a polygon and then proceed as if it were a polygon.
>>>> >
>>>> > Still, a mask method would be a nice add to the FG.
>>>> >
>>>> > Cheers,
>>>> > Helder
>>>>
>>>> Hi Helder,
>>>> If you draw the ellipse by calling the ELLIPSE function, you should know everything. If you draw it by hand, you can get the rectangle containing the ellipse by doing (after selecting it):
>>>>   pos = GetWindows(/CURRENT).GetSelect().Position
>>>> then, center and axis lengthes.
>>>> alx.
>>>
>>> Hi Alx,
>>> thanks.
>>> The idea is that I have a widget window where I move and change my ellipse as I wish.
>>> Say I start with this:
>>>
>>> img = dist(600)
>>> w = window(dimensions=[500,500])
>>> im = image(img, current=w)
>>> el = ellipse(300,300,/data, minor=100, major=150, target=im)
>>>
>>> and then I modify the ellipse with the mouse (make it bigger, rotate, stretch,...).
>>> The changes that will be made I don't know "a priori", so by stretching and rotating I end up with a completely different ellipse.
>>>
>>> I can get the position (as you said) with
>>>
>>> pos = el.position
>>>
>>> and that gives me always the center as [(pos[0]+pos[2])/2.0,(pos[1]+pos[3])/2.0]. But I cannot retrieve the rotation or the axis and, as far as I understand, there are infinite ellipses that can fit inside the rectangle defined by the position property that have a different set of rotation/axis.
>>>
>>> I think that I will have to stick with the polygon type solution using IDLanROI.
>>>
>>> Thanks,
>>> Helder
>>

>> How adventurous do you want to be? The transformation information is stored in the container of the polygon. The coordinates of the raw data don't actually change after the polygon is created, I would guess.
>>
>> Let's say you've started a new IDL session and executed the above commands then you've selected and resized or repositioned it interactively. (I'm using IDL 8.4 so YMMV.)
>>
>> Which object that was created is an IDLgrPolygon?
>>

```
>>  IDL> oall = obj_valid(/cast)
>>  IDL> print, where(obj_isa(oall, 'idlgrpolygon'))
>>          262        550        568        586        604        621        640        657        676
1894       2089
```

>>
>> We have a bunch. One is likely the "main" polygon and the others the "selection" items for resizing and so forth.
>>
>> Which one is the "main" polygon? Let's guess the first and change its color.
>>

```
>>  IDL> oall[262]->setproperty, color = [0, 255, 0] & el.refresh
```

>>
>> Lucky guess.
>>
>> Next, let's get its container, which is by definition a model.
>>

```
>>  IDL> oall[262]->getproperty, parent = omodel
```

>>
>> Check out the transformation matrix.
>>

```
>>  IDL> omodel->getproperty, transform = t
>>  IDL> print, t
>>       0.51712623      0.00000000      0.00000000       55.899910
>>       0.00000000       1.3790130      0.00000000      -120.23054
>>       0.00000000      0.00000000       1.0000000      0.00000000
>>       0.00000000      0.00000000      0.00000000       1.0000000
```

>>
>> Of course, you'll have different values here.
>>
>> Let's send the ellipse back "home":
>>

```
>>  IDL> t = identity(4)
>>  IDL> omodel->setproperty, transform = t & el.refresh
```

>>
>> Teasing out the rotation, translation, and scale from the quaternion is an exercise left for the reader.
>>
>> In function graphics there's a whole lot of cruft, er, framework, that sits over the top of the most basic graphics atoms and models and knowing how the basic, documented, objects work will

go a long way to helping you get closer to what you want.
>>
>> If the new graphics API doesn't expose exactly the features you want, there are ways to get and modify the data and behaviors, with the possible side effect of producing inconsistent internal states, particularly during interactive use.
>>
>> Run with scissors.
>>
>> Jim P.
>> "I work for Exelis VIS, but these are my own observations"
>
> Hi Jim,
> thanks for your insights into FG. To me most (=all) of what's happening behind the scenes is a mystery, therefore such demos are enlightening.
> With that said, my program is somewhat complicated and it might have to deal with more instances of ellipses and things might get nasty I'm not sure that going through all this trouble would actually give me an advantage over getting the mask using the point coordinates used to draw the ellipse in the polygon object and feeding them in an IDLanROI object.
>
> I like anyway to fiddle around with these things, therefore I tried to get your code running and got stuck pretty soon. It seems like I can't change the color of the "main" polygon.
> Here is the code I've tested and I've put a break in the for loop to check the color of the ellipse.
>

```
>  pro testEllipse
>  img = dist(600)
>  w = window(dimensions=[500,500])
>  im = image(img, current=w)
>  el = ellipse(300,300,'r3', /data, minor=100, major=150, target=im, fill_background=0)
>  el->rotate, 15
>  el->translate, 40,60
>  oall = obj_valid(/cast)
>  polys = where(obj_isa(oall, 'idlgrpolygon'))
>  for i=0,n_elements(polys)-1 do begin
>     oall[polys[i]]->getproperty, color = c
>     print, 'starting color = ', c
>     oall[polys[i]]->setproperty, color = [0,255,0]
>     el.refresh
>     ;insert break in the following line to check if the ellipse has changed color
>     print, 'color set'
>     oall[polys[i]]->setproperty, color = c
>     el.refresh
>  endfor
>  end
```

>
> It so happens that the ellipse stays as it is. Interestingly, the colors I get from the objects I scan through are:
> 3 times [255,255,255]
> 8 times [30,144,255]

> I get no red ([255,0,0]).
>
> Since you have more insight than I do, I'll ask you the following:
>
> What is the probability that in the near future (<1year) Exelis implements a mask for objects like Polygons and Ellipses?
>
> I doesn't seem to me like a too complicated thing to implement if you have access (and understanding) to the lower layers of IDL FG and access to the mask method used for IDLanROI.
>
> Thanks,
> Helder

Ah, I see what you did there. You changed the problem on me.

The color that I was setting was the fill color on the filled IDLgrPolygon from your first example. You're attempting to change the line color. The ellipse graphic you're creating is a composite of objects. One is the filled IDLgrPolygon. Another is the IDLgrPolyline that serves as the outline. Armed with that info, you should be able to make further progress. (A shortcut is to know that for our scene the IDLgrPolygon and its associated IDLgrPolyline likely both belong to the same parent, if there are lots of polylines in a scene.)

Here's the transform of the first point, showing how the effects of the rotation and translation operations:

```
IDL> oall[polys[0]]->getproperty, parent = p
IDL> p->getproperty, transform = t
IDL> oall[polys[0]]->getproperty, data = data
IDL> help, data
DATA            DOUBLE    = Array[3, 182]
IDL> print, data[*, 0]
      450.00000       300.00000      0.00000000
IDL> print, [data[*, 0], 1] # t
      498.22221
      418.82286
     0.00000000
      1.0000000
```

I will need to defer to the always-knowledgeable Chris T. with respect to what may or may not appear in the IDL language or graphics features in the future. My wishlist has been quite lengthy for decades, but I'm usually on the naughty-not-nice list. I got a sock full of gifts with 8.4 this year, but many years I've received lumps of coal. Running with scissors takes a toll on one's rep.

Jim P.