Subject: Ongoing Object Graphics Quest
Posted by David Fanning on Mon, 19 Nov 2001 01:23:42 GMT
View Forum Message <> Reply to Message

Folks,

I've been on a bit of a quest lately to learn more about object graphics. I continue to be amazed at how hard it is to figure everything out. Partly this is because object graphics are so haughty. When you make an absolutely asinine mistake, they just stare at you with this incredibly disgusted look on their face. They don't say *anything*, unlike direct graphics, which sometimes deign to send you a cryptic error message that can occasionally get you back on track.

We spend an inordinate amount of time just staring at each other, although I usually have more of a blank look on my face than an intimidating one. :-(

Anyway, to solve a problem for a client, and to give myself a challenging programming exercise I decided to work on an image processing application that would allow me to interactively set the contrast/brightness (also called the window/level) of an image. Several weeks ago I reported on a similar program I had written in direct graphics. (That direct graphics program, WindowImage, has been upgraded, incidentally, as a result of lessons learned in the past couple of days.)

Because I has already *mostly* solved the contrast/ brightness problem and I (naively) believed that converting that to object graphics wouldn't be much of a challenge, I decided to make the problem more difficult. I also wanted to be able to zoom the image "in place", and in a way that preserved the aspect ratio of the zoomed image subset. This was a bigger challenge because laying things out in object graphics windows (at least according to all the examples RSI provides and my own experience) is one gigantic pain in the ol' wazoo. I wanted to develop a rational way of doing this that I could explain to someone.

Finally, I wanted to know how to have several

"views" of data in one graphics window, and how
to interact with those views independently. (Had
I thought about this for longer than five minutes I would
certainly have given the whole project up as hopeless
before I went to all this trouble.)

The result is a new program on my web page, named
ContrastZoom.

http://www.dfanning.com/programs/contrastzoom.pro

There are three "views" in the window. The view on
the left is the zoom window. You can draw a rubberband
box about a portion of the image that you want to
see closer up. Although I don't zoom into the image,
I take that portion and display it in the same location
in the graphics window in a way that preserves its
aspect ratio. I call this "zoom in place", because
the effect is to zoom into a particular location. You
can go back to the entire image by just clicking and
releasing the cursor in that window.

The center "view" or image is the image that you
use to adjust the contrast/brightness of the
image. Dragging the cursor horizontally sets the
brightness or level. Dragging the cursor vertically
sets the contrast or window. Clicking and releasing
will set the original values of 25% contrast and
75% brightness.

What I particularly like about this program is that
the colorbar in the third "view" on the right
reflects the current window and level. I know
I will offend you medical guys with some color,
but the rest of us can see this better by running
the program with a red-temperature color table like this:

IDL>ContrastZoom, Colortable=3

I understand that this is not the best program I've
even written. But I spent most of 10 days writing it,
and I think even as it is, it might save someone else
a heck of a lot of time. I already have ideas for how
the program can be improved if I decide to put the
lessons learned here in a book.

(By the way, I didn't get anyone responding with
suggestions for improving my previous contrast/brightness

algorithm, so I had to do it myself. I'm still not
totally in love with it, but it's getting better
every time I work on it. At least with the color bar
feedback, I can tell it works now the way I expect
it to work. That's something, anyway.)

As always, I appreciate the feedback.

If you haven't visited my web page in a while,
you can find several new programs at the usual place:

   http://www.dfanning.com/documents/programs.html

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Ongoing Object Graphics Quest
Posted by Martin Downing on Mon, 19 Nov 2001 10:15:11 GMT
View Forum Message <> Reply to Message

Hi David,

Nice program, however I noticed two things you might want to change if your
not tired of the project yet :)
1) The rubber band currently includes the next pixel above and to the right
of the defined area.
2) The extent of contrast and brightness stretching appears to be restricted
when the left image has been zoomed

cheers

Martin

"David Fanning" <david@dfanning.com> wrote in message
news:MPG.16620d0932648f9498977f@news.frii.com...
> Folks,
>
> I've been on a bit of a quest lately to learn
> more about object graphics. I continue to be

> amazed at how hard it is to figure everything
> out. Partly this is because object graphics are
> so haughty. When you make an absolutely asinine
> mistake, they just stare at you with this incredibly
> disgusted look on their face. They don't say *anything*,
> unlike direct graphics, which sometimes deign to
> send you a cryptic error message that can occasionally
> get you back on track.
>
> We spend an inordinate amount of time just
> staring at each other, although I usually have
> more of a blank look on my face than an intimidating
> one. :-(
>
> Anyway, to solve a problem for a client, and to
> give myself a challenging programming exercise
> I decided to work on an image processing application
> that would allow me to interactively set the
> contrast/brightness (also called the window/level)
> of an image. Several weeks ago I reported on a
> similar program I had written in direct graphics.
> (That direct graphics program, WindowImage, has
> been upgraded, incidentally, as a result of lessons
> learned in the past couple of days.)
>
> Because I has already *mostly* solved the contrast/
> brightness problem and I (naively) believed that
> converting that to object graphics wouldn't be much
> of a challenge, I decided to make the problem more
> difficult. I also wanted to be able to zoom the image
> "in place", and in a way that preserved the aspect
> ratio of the zoomed image subset. This was a
> bigger challenge because laying things out in
> object graphics windows (at least according to all
> the examples RSI provides and my own experience)
> is one gigantic pain in the ol' wazoo. I wanted to
> develop a rational way of doing this that I could
> explain to someone.
>
> Finally, I wanted to know how to have several
> "views" of data in one graphics window, and how
> to interact with those views independently. (Had
> I thought about this for longer than five minutes I would
> certainly have given the whole project up as hopeless
> before I went to all this trouble.)
>
> The result is a new program on my web page, named
> ContrastZoom.

>
>   http://www.dfanning.com/programs/contrastzoom.pro
>
> There are three "views" in the window. The view on
> the left is the zoom window. You can draw a rubberband
> box about a portion of the image that you want to
> see closer up. Although I don't zoom into the image,
> I take that portion and display it in the same location
> in the graphics window in a way that preserves its
> aspect ratio. I call this "zoom in place", because
> the effect is to zoom into a particular location. You
> can go back to the entire image by just clicking and
> releasing the cursor in that window.
>
> The center "view" or image is the image that you
> use to adjust the contrast/brightness of the
> image. Dragging the cursor horizontally sets the
> brightness or level. Dragging the cursor vertically
> sets the contrast or window. Clicking and releasing
> will set the original values of 25% contrast and
> 75% brightness.
>
> What I particularly like about this program is that
> the colorbar in the third "view" on the right
> reflects the current window and level. I know
> I will offend you medical guys with some color,
> but the rest of us can see this better by running
> the program with a red-temperature color table like this:
>
>   IDL>ContrastZoom, Colortable=3
>
> I understand that this is not the best program I've
> even written. But I spent most of 10 days writing it,
> and I think even as it is, it might save someone else
> a heck of a lot of time. I already have ideas for how
> the program can be improved if I decide to put the
> lessons learned here in a book.
>
> (By the way, I didn't get anyone responding with
> suggestions for improving my previous contrast/brightness
> algorithm, so I had to do it myself. I'm still not
> totally in love with it, but it's getting better
> every time I work on it. At least with the color bar
> feedback, I can tell it works now the way I expect
> it to work. That's something, anyway.)
>
> As always, I appreciate the feedback.
>

> If you haven't visited my web page in a while,
> you can find several new programs at the usual place:
>
> http://www.dfanning.com/documents/programs.html
>
> Cheers,
>
> David
>
> --
> David W. Fanning, Ph.D.
> Fanning Software Consulting
> Phone: 970-221-0438, E-mail: david@dfanning.com
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Ongoing Object Graphics Quest
Posted by Martin Downing on Mon, 19 Nov 2001 12:12:08 GMT
View Forum Message <> Reply to Message

"Martin Downing" <martin.downing@ntlworld.com> wrote in message
news:BY4K7.10623$tm3.1406304@news11-gui.server.ntli.net...
> Hi David,
>
> Nice program, however I noticed two things you might want to change if
your
> not tired of the project yet :)
> 1) The rubber band currently includes the next pixel above and to the
right
> of the defined area.

Oh I see - its actually the other way round. The drawn rubber band, defined
as "theBox", does not draw round the top-right corner of the top right
pixel.
You could round this up like for instance:

```
box_x0 = min([info.xs, info.xd])
box_x1 = max([info.xs, info.xd])+1
box_y0 = min([info.ys, info.yd])
box_y1 = max([info.ys, info.yd])+1

box[0,*] = [box_x0,box_x1,box_x1,box_x0,box_x0]
box[1,*] = [box_y0,box_y0,box_y1,box_y1,box_y0]

info.theBox->SetProperty, Data=box
```

Very helpful reading though, will definitely help if I go to full object

graphics for my 2D viewers

cheers

Martin

Subject: Re: Ongoing Object Graphics Quest
Posted by David Fanning on Mon, 19 Nov 2001 14:48:30 GMT
View Forum Message <> Reply to Message

Martin Downing (martin.downing@ntlworld.com) writes:

> Oh I see - its actually the other way round. The drawn rubber band, defined
> as "theBox", does not draw round the top-right corner of the top right
> pixel.
> You could round this up like for instance:
>
> box_x0 = min([info.xs, info.xd])
> box_x1 = max([info.xs, info.xd])+1
> box_y0 = min([info.ys, info.yd])
> box_y1 = max([info.ys, info.yd])+1
>
> box[0,*] = [box_x0,box_x1,box_x1,box_x0,box_x0]
> box[1,*] = [box_y0,box_y0,box_y1,box_y1,box_y0]
>
> info.theBox->SetProperty, Data=box

Actually, I think it is a little more pernicious
than this. :-)

The problem is that I get a location in the window,
but what I want is an image subscript. If I have two
locations in the window (e.g. a line), those endpoints
are one pixel longer than the subscripts that I need.

But here is the dilemma: which end of the line should
I subtract one from (or add one to)? It depends on
how you have drawn the line (e.g. right to left, or
left to right) and whether one end of the line is
on the image boundary. I think there must be at
least eight possibilities you have to check to be
able to get your box boundaries correct.

Rather than doing this, I elected to take the FLOOR
of all points, since this always keeps me inside the
image. The downside is that I often get one more
row and one more column of the image in the zoom

than I really wanted. For the images I work with,
this is really not a concern, since one extra row
or column in a 512x512 image is almost invisible.

But when you work with images with really big pixels
(e.g., image = Findgen(5, 6)), then the problem
becomes obvious. I'll probably have to fix it, but
it may have to wait a couple of days. I really have
to get some real work done today.

> 2) The extent of contrast and brightness stretching
> appears to be restricted when the left image has been zoomed

I don't think so, since there is no connection at all
between what is happening in the zoom window and the
contrast/brightness algorithm.

I appreciate the help.

Best Regards,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Ongoing Object Graphics Quest
Posted by Martin Downing on Mon, 19 Nov 2001 16:48:33 GMT

View Forum Message <> Reply to Message

"David Fanning" <david@dfanning.com> wrote in message
news:MPG.1662c9a6c36c44e6989782@news.frii.com...
> Martin Downing (martin.downing@ntlworld.com) writes:
>
>> Oh I see - its actually the other way round. The drawn rubber band,
defined
>> as "theBox", does not draw round the top-right corner of the top right
>> pixel.
>> You could round this up like for instance:
>>
>> box_x0 = min([info.xs, info.xd])
>> box_x1 = max([info.xs, info.xd])+1
>> box_y0 = min([info.ys, info.yd])

>> box_y1 = max([info.ys, info.yd])+1
>>
>> box[0,*] = [box_x0,box_x1,box_x1,box_x0,box_x0]
>> box[1,*] = [box_y0,box_y0,box_y1,box_y1,box_y0]
>>
>> info.theBox->SetProperty, Data=box
>
> Actually, I think it is a little more pernicious
> than this. :-)
>
> The problem is that I get a location in the window,
> but what I want is an image subscript. If I have two
> locations in the window (e.g. a line), those endpoints
> are one pixel longer than the subscripts that I need.
>
> But here is the dilemma: which end of the line should
> I subtract one from (or add one to)? It depends on
> how you have drawn the line (e.g. right to left, or
> left to right) and whether one end of the line is
> on the image boundary. I think there must be at
> least eight possibilities you have to check to be
> able to get your box boundaries correct.
>

Hi David,

The 8 possibilities was why I used max( ) and min( ). This modification was only intended for the bounding box polygon of the selected area, not for indexing the sub image.
I tried it in your code and the only problem was that sometimes the top/right side was just outside the drawable view, other times it worked fine.

Ok, I'll stop messing now!

Martin
