Subject: 3D point cloud visualization
Posted by Nuno Ferreira on Tue, 28 Apr 2015 15:31:18 GMT
View Forum Message <> Reply to Message

Hi,

I am trying to visualize a 3D point cloud that was captured with a Kinect camera. I have "v", a (3,N) array of the xyz coordinates of N vertices and "p", a (4,M) array of M polygons (all triangles). Each polygon contains the number of vertices (always 3) and indices to 3 vertices in array "v". For each vertex I also have the color, given as RGB components and alpha.

I have two questions:

1 - I would like to use POLYSHADE with the SHADES keyword and the RGB information, but the manual says "When using the SHADES keyword on TrueColor devices, we recommend that decomposed color support be turned off by setting DECOMPOSED=0". However, using decomposed=0 limits the number of simultaneous colors to only 256... Isn't it possible to use the SHADES keyword with true color (RGB components)?

2 - I am also using PLOTS to draw all the vertices in 3D as dots, with a command such as:

plots, v[0,*], v[1,*], v[2,*], color=quantized_colors_256, /T3D, psym=3

How can I increase the size of the dots with PLOTS? Apparently keyword SYMSIZE does not serve this purpose...

Any help would be greatly appreciated.
Thanks,

Nuno

Subject: Re: 3D point cloud visualization
Posted by Jeremy Bailin on Tue, 28 Apr 2015 16:39:58 GMT
View Forum Message <> Reply to Message

On Tuesday, April 28, 2015 at 10:31:19 AM UTC-5, Nuno Ferreira wrote:
> Hi,
>
> I am trying to visualize a 3D point cloud that was captured with a Kinect camera. I have "v", a (3,N) array of the xyz coordinates of N vertices and "p", a (4,M) array of M polygons (all triangles). Each polygon contains the number of vertices (always 3) and indices to 3 vertices in array "v". For each vertex I also have the color, given as RGB components and alpha.
>
>
> I have two questions:

>
> 1 - I would like to use POLYSHADE with the SHADES keyword and the RGB information, but the manual says "When using the SHADES keyword on TrueColor devices, we recommend that decomposed color support be turned off by setting DECOMPOSED=0". However, using decomposed=0 limits the number of simultaneous colors to only 256... Isn't it possible to use the SHADES keyword with true color (RGB components)?
>
>
> 2 - I am also using PLOTS to draw all the vertices in 3D as dots, with a command such as:
>
> plots, v[0,*], v[1,*], v[2,*], color=quantized_colors_256, /T3D, psym=3
>
> How can I increase the size of the dots with PLOTS? Apparently keyword SYMSIZE does not serve this purpose...
>
> Any help would be greatly appreciated.
> Thanks,
>
> Nuno

I can only help with the second one. A PSYM=3 dot is always exactly one pixel. If you want a filled circle, you should use something like cgSymCat(16).

-Jeremy.

Subject: Re: 3D point cloud visualization
Posted by Dick Jackson on Wed, 29 Apr 2015 02:55:54 GMT
View Forum Message <> Reply to Message

Hi Nuno,

Sounds fascinating! I have time for only a quick reply now: If you're looking to quickly visualize this, Object Graphics is pretty handy, and you have all the data in exactly the right form!:

; Assumes 'rgba' is (4,N) byte array of RGBA values (0-255 for alpha as well)

o = Obj_New('IDLgrPolygon', v, Polygons=p, Vert_Colors=rgba, Style=2)
XObjView, o

You may want to look at other IDLgrPolygon Properties in Online Help, and options to XObjView.

The IDL 8.0 "function graphics" routine POLYGON() is another possibility you might try. Write back if you'd like to know more.

--

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada
www.d-jackson.com

Nuno Ferreira wrote on 2015-04-28 8:31am:
> Hi,
>
> I am trying to visualize a 3D point cloud that was captured with a Kinect
> camera. I have "v", a (3,N) array of the xyz coordinates of N vertices and
> "p", a (4,M) array of M polygons (all triangles). Each polygon contains the
> number of vertices (always 3) and indices to 3 vertices in array "v". For
> each vertex I also have the color, given as RGB components and alpha.
>
>
> I have two questions:
>
> 1 - I would like to use POLYSHADE with the SHADES keyword and the RGB
> information, but the manual says "When using the SHADES keyword on TrueColor
> devices, we recommend that decomposed color support be turned off by setting
> DECOMPOSED=0". However, using decomposed=0 limits the number of simultaneous
> colors to only 256... Isn't it possible to use the SHADES keyword with true
> color (RGB components)?
>
> 2 - I am also using PLOTS to draw all the vertices in 3D as dots, with a
> command such as:
>
> plots, v[0,*], v[1,*], v[2,*], color=quantized_colors_256, /T3D, psym=3
>
> How can I increase the size of the dots with PLOTS? Apparently keyword
> SYMSIZE does not serve this purpose...
>
> Any help would be greatly appreciated. Thanks,
>
> Nuno
>

Subject: Re: 3D point cloud visualization
Posted by Jim Pendleton on Wed, 29 Apr 2015 04:13:53 GMT
View Forum Message <> Reply to Message

On Tuesday, April 28, 2015 at 9:31:19 AM UTC-6, Nuno Ferreira wrote:
> Hi,

>
> I am trying to visualize a 3D point cloud that was captured with a Kinect camera. I have "v", a (3,N) array of the xyz coordinates of N vertices and "p", a (4,M) array of M polygons (all triangles). Each polygon contains the number of vertices (always 3) and indices to 3 vertices in array "v". For each vertex I also have the color, given as RGB components and alpha.
>
>
> I have two questions:
>
> 1 - I would like to use POLYSHADE with the SHADES keyword and the RGB information, but the manual says "When using the SHADES keyword on TrueColor devices, we recommend that decomposed color support be turned off by setting DECOMPOSED=0". However, using decomposed=0 limits the number of simultaneous colors to only 256... Isn't it possible to use the SHADES keyword with true color (RGB components)?
>
>
> 2 - I am also using PLOTS to draw all the vertices in 3D as dots, with a command such as:
>
> plots, v[0,*], v[1,*], v[2,*], color=quantized_colors_256, /T3D, psym=3
>
> How can I increase the size of the dots with PLOTS? Apparently keyword SYMSIZE does not serve this purpose...
>
> Any help would be greatly appreciated.
> Thanks,
>
> Nuno

Dick is right... This is a job for either "function graphics" or good old Object Graphics. Take a look at the POLYGON function as he suggests and possibly an overplot with a SCATTERPLOT3D to show the vertices, for example

```
IDL> symbol = orb(density = 1, radius = 1., color = [255, 0, 0])
IDL> x = randomu(seed, 100)
IDL> y = randomu(seed, 100)
IDL> z = randomu(seed, 100)
IDL> s = scatterplot3d(x, y, z, sym_object = symbol)
```

Jim P.

Subject: Re: 3D point cloud visualization
Posted by Nuno Ferreira on Wed, 29 Apr 2015 16:18:03 GMT
View Forum Message <> Reply to Message

Thank you very much for your fast replies, they were most helpful! Since I now have to change my widget application that presents several views of the 3D point cloud, I have a few more questions... (since they relate to the same topic, I keep them in the same thread).

1 - I could use the POLYGONS function ("function graphics") with my widget application by adding a WIDGET_WINDOW. However, I could not do the same with the "object graphics" routine XObjView: it keeps showing the point cloud in a separate window instead of inside the main application window. How can I make XObjView to draw the point cloud inside e.g. a draw widget? (BTW, I read this group's thread "Object Graphics and Widgets" that explains the basics)

2 - With object graphics (XObjView and IDLgrPolygon) I can easily rotate the point cloud in 3D by using the mouse, which is great. However, with "function graphics" POLYGON() I only could move and resize the data, not rotate it. Is there a way to interactively rotate in 3D for this case? (I could not find this in the documentation, sorry)

3 -Finally, if you have suggestions on having multiple 3D views (3, for instance) of the same data inside a widget application, I'm all ears...

Thanks in advance,
Nuno

Subject: Re: 3D point cloud visualization
Posted by Jim Pendleton on Wed, 29 Apr 2015 19:14:34 GMT
View Forum Message <> Reply to Message

On Wednesday, April 29, 2015 at 10:18:04 AM UTC-6, Nuno Ferreira wrote:
> Thank you very much for your fast replies, they were most helpful! Since I now have to change my widget application that presents several views of the 3D point cloud, I have a few more questions... (since they relate to the same topic, I keep them in the same thread).
>
>
> 1 - I could use the POLYGONS function ("function graphics") with my widget application by adding a WIDGET_WINDOW. However, I could not do the same with the "object graphics" routine XObjView: it keeps showing the point cloud in a separate window instead of inside the main application window. How can I make XObjView to draw the point cloud inside e.g. a draw widget? (BTW, I read this group's thread "Object Graphics and Widgets" that explains the basics)
>
>
> 2 - With object graphics (XObjView and IDLgrPolygon) I can easily rotate the point cloud in 3D by using the mouse, which is great. However, with "function graphics" POLYGON() I only could move and resize the data, not rotate it. Is there a way to interactively rotate in 3D for this case? (I could not find this in the documentation, sorry)
>
>
> 3 -Finally, if you have suggestions on having multiple 3D views (3, for instance) of the same data inside a widget application, I'm all ears...
>

> Thanks in advance,
> Nuno

I've looked at the Kinect API but have yet to write anything with it, so I'd like to see what you come up with as well. Please share some screen shots if you get a chance and you're allowed.

I think you're requiring enough custom code that I would steer you toward the lower level Object Graphics rather than function graphics. The trackball object will give you the 3D functionality you need. You can add your own models and event handling for translation and scaling. It would be a fair amount of work to try to put XOBJVIEW functionality directly into your existing widget. It was originally designed as a demo and wasn't designed to be embedded, which is unfortunate since it's one of the more useful tools for quickly viewing 3D model trees.

Below is some stripped down example code that could help you initially. Also see the IDL Data Point blog article I wrote back in September 2014 that includes a discussion of aliasing model trees: http://tinyurl.com/ohe6377

```
pro tball_event, e
widget_control, e.top, get_uvalue = tb
if (tb.update(e, transform = txf)) then begin
   d = widget_info(e.top, /child)
   widget_control, d, get_value = ow
   ow.getproperty, graphics_tree = ov
   om = ov.get()
   om.getproperty, transform = mxf
   om.setproperty, transform = mxf # txf
   ow.draw
endif
; You could add other event handling for zoom and translate here
; since trackball only manages rotation.
; Consider adding nested models to handle rotation, translation
; and scaling separately.
end


pro tb
; Make a cylinder
t = findgen(30)*12*!dtor
x = cos(t)
y = sin(t)
z = replicate(-2.5, t.length)
s = 512
mesh_obj, 5, v, c, transpose([[x],[y],[z]]),p1 = 2, p2 = [0, 0, 5], /closed
; Create a model and add orb objects at each vertex, each with its own color
om = idlgrmodel()
vc = bytarr(3, v.length/3)
```

```
for i = 0l, v.length/3 - 1 do begin
   vc[*, i] = [i *3 mod 256, i * 5 mod 256, 255 - i*7 mod 256]
   om.add, orb(pos = v[*, i], density = 1, radius = .1, color = vc[*, i])
endfor
; Add the cylinder itself to the model: optional if you have connectivity
;p = idlgrpolygon(v, poly = c, vert_colors = vc, style = 2, shading = 1)
;om.add, p
; Create a view space to hold the cylinder
ov = idlgrview(viewplane_rect = [-5., -5., 10, 10], zclip = [5, -5], eye = 6.)
ov.add, om
; Consider adding ALIAS models to contain alternative views of the same
; source model, so you don't need to maintain multiple copies.
; Simply modify the transform of the model containing the alias.
; Create a widget draw in a base
w = widget_base()
d = widget_draw(w, graphics_level = 2, xsize = s, ysize = s, $
   /button_events, /motion_events)
; Create a trackball to handle left-button drag events for rotation
tb = trackball([s/2, s/2], s/2)
widget_control, w, set_uvalue = tb
widget_control, w, /realize
widget_control, d, get_value = ow
ow.setproperty, graphics_tree = ov
ow.draw
xmanager, 'tball', w, /no_block
end
```

Subject: Re: 3D point cloud visualization
Posted by Nuno Ferreira on Thu, 30 Apr 2015 22:29:40 GMT
View Forum Message <> Reply to Message

Many thanks for the trackball code, Jim, it is very useful! I also took a look at your blog and learned several nice IDL features (I really need to update my knowledge in IDL - I tend to use the same old features from several years ago...).

As for the Kinect API, I also need to learn how to use it... (I have just been reading and processing the data created by the example tools, in the '.ply' format). When I have some nice images I will show some screenshots.

Nuno

Subject: Re: 3D point cloud visualization
Posted by Nuno Ferreira on Tue, 26 May 2015 08:55:20 GMT
View Forum Message <> Reply to Message

OK, trying to keep a promise, here are some screenshots:

- https://drive.google.com/file/d/0B6Ti5FMqve-dLU1KcVc4OWhaWkk /view?usp=sharing

- (close up, using a different dataset):
https://drive.google.com/open?id=0B6Ti5FMqve-da0RXNllHUFJfSl k&authuser=0

I still have plenty to do, but it's starting to show up...

Nuno
