Subject: Re: Texture Map problems when using IDLgrPolygon
Posted by Karl[1] on Mon, 04 Jun 2012 14:18:59 GMT
View Forum Message <> Reply to Message

On Sunday, June 3, 2012 9:08:46 PM UTC-6, mikrin wrote:
> Hi All, I've been trying to develop an IDL object graphics approach
> of putting image swaths onto a 2D Orthographic
> projection of the globe. It seems I've got most of it in place put
> the image that is currently texture mapped does
> not look correct. My images are bytarr's of 1354x2030. My approach
> was to texture map the image onto a IDLgrPolygon
> object. I use the known latitude and longitude grid (1354x2030) to
> convert to an xy grid. These xy grid points
> are the polygon vertices. I then normalize these to (0.0 -> 1.0) and
> use them as the texture_coords.
> The code snippet below shows the relevant part of the process,
>
> ---------- code snippet below ----------
>
> ; Scale the image to a byte
> scaledImage = BytScl(image, Top=254) + 1B
> sz = size(scaledImage,/dimensions)
>
> ; Define a texture image byte array (next power of 2 larger)
> ; to hold scaledImage. This is preferable because of a vaguery
> in how
> ; a texture map is warped onto an image. If not a power of 2
> then texture
> ; mapping introduces sampling artifacts into the image
> pwr2Sz = findNextPwr2(sz)
> textureImg = bytarr(pwr2Sz[0],pwr2Sz[1])
> textureImg[0:sz[0]-1, 0:sz[1]-1] = scaledImage
>
> ;
> ; Make a pallete object for the image
> oPal = OBJ_NEW('IDLgrPalette')
> oPal->LOADCT, 33
>
> ;
> ; Use the 1354x2030 bytarr the image object. Attempt 2
> described below
> ;oImg = obj_new('IDLgrImage', scaledImage, PALETTE=oPal,
> ORDER=1)
>
> ; Or
>
> ; Use a 2048x2048 expanded bytarr the image object. Attempt 3
> described below.

```
>       oImg = obj_new('IDLgrImage', textureImg, PALETTE=oPal, ORDER=1)
>
>       ;
>       ; Create the polygon vertices and texture coordinates from the
>  lats lons arrays.  Uses
>       ; map_proj_forward to convert lat lons to x,y
>       makePolyVerts, lats, lons, MAP=map, vertexes, textureCoords
>
>       ;
>       ; Create a polygon object to hold the texture mapped image to.
>       oPoly = obj_new('IDLgrPolygon', DATA=vertexes, TEXTURE_MAP=oImg,
>  $
>             Texture_Coord=textureCoords, COLOR=[255,255,255])
>
>       ;
>       ;Add the image to the model
>       oModel->add, oPoly
>
>       ; Draw image
>       window->draw, oView
>
>  ---------------------- code snippet above --------------------
>
>
>  The code snippet above where I use 'scaledImage' (a 1354x2030 bytarr)
>  results in an image that is not correct.  Is this due to the known
>  texture mapping sampling problem when using non power of 2 arrays?
>
>
>  The code snippet above where I use 'textureImg' (a 2048x2048 bytarr)
>  is my attempt to follow the suggestion that texture maps work best
>  when the array is a power 2 multiple.  The results here actually look
>  worse.
>
>  Can anyone suggest how to fix this problem?  Is my implementation  of
>  the larger 2^n array wrong?
>
>  Thanks for any help,  Mike
```

It is hard to tell because not all of the code is here, but...

I think you still have to go with the power of two size for the textures.

When you go this route, you have to normalize the texture coordinates differently and not to the [0.0, 1.0] range. It is not clear if your makePolyVerts procedure handles this.

In your case, you'll want to normalize to [0, 1354.0/2048] in x and [0, 2030.0/2048] in y. The idea is that you want to limit your texture coordinates to the "active" area of the texture.

Karl

Subject: Re: Texture Map problems when using IDLgrPolygon
Posted by mikrin on Mon, 04 Jun 2012 21:00:07 GMT
View Forum Message <> Reply to Message

On Jun 4, 7:18 am, Karl <Karl.W.Schu...@gmail.com> wrote:
> On Sunday, June 3, 2012 9:08:46 PM UTC-6, mikrin wrote:
>> Hi All, I've been trying to develop an IDL object graphics approach
>> of putting image swaths onto a 2D Orthographic
>> projection of the globe. It seems I've got most of it in place put
>> the image that is currently texture mapped does
>> not look correct. My images are bytarr's of 1354x2030. My approach
>> was to texture map the image onto a IDLgrPolygon
>> object. I use the known latitude and longitude grid (1354x2030) to
>> convert to an xy grid. These xy grid points
>> are the polygon vertices. I then normalize these to (0.0 -> 1.0) and
>> use them as the texture_coords.
>> The code snippet below shows the relevant part of the process,
>
>> ------------------ code snippet below ----------------------
>
>> ; Scale the image to a byte
>> scaledImage = BytScl(image, Top=254) + 1B
>> sz = size(scaledImage,/dimensions)
>
>> ; Define a texture image byte array (next power of 2 larger)
>> ; to hold scaledImage. This is preferable because of a vaguery
>> in how
>> ; a texture map is warped onto an image. If not a power of 2
>> then texture
>> ; mapping introduces sampling artifacts into the image
>> pwr2Sz = findNextPwr2(sz)
>> textureImg = bytarr(pwr2Sz[0],pwr2Sz[1])
>> textureImg[0:sz[0]-1, 0:sz[1]-1] = scaledImage
>
>> ;
>> ; Make a pallete object for the image
>> oPal = OBJ_NEW('IDLgrPalette')
>> oPal->LOADCT, 33
>
>> ;
>> ; Use the 1354x2030 bytarr the image object. Attempt 2
>> described below
>> ;oImg = obj_new('IDLgrImage', scaledImage, PALETTE=oPal,
>> ORDER=1)

>
>>   ; Or
>
>>   ; Use a 2048x2048 expanded bytarr the image object. Attempt 3
>> described below.
>>   oImg = obj_new('IDLgrImage', textureImg, PALETTE=oPal, ORDER=1)
>
>>   ;
>>   ; Create the polygon vertices and texture coordinates from the
>> lats lons arrays. Uses
>>   ; map_proj_forward to convert lat lons to x,y
>>   makePolyVerts, lats, lons, MAP=map, vertexes, textureCoords
>
>>   ;
>>   ; Create a polygon object to hold the texture mapped image to.
>>   oPoly = obj_new('IDLgrPolygon', DATA=vertexes, TEXTURE_MAP=oImg,
>> $
>>   Texture_Coord=textureCoords, COLOR=[255,255,255])
>
>>   ;
>>   ;Add the image to the model
>>   oModel->add, oPoly
>
>>   ; Draw image
>>   window->draw, oView
>
>> ---------- code snippet above ----------
>
>> The code snippet above where I use 'scaledImage' (a 1354x2030 bytarr)
>> results in an image that is not correct. Is this due to the known
>> texture mapping sampling problem when using non power of 2 arrays?
>
>> The code snippet above where I use 'textureImg' (a 2048x2048 bytarr)
>> is my attempt to follow the suggestion that texture maps work best
>> when the array is a power 2 multiple. The results here actually look
>> worse.
>
>> Can anyone suggest how to fix this problem? Is my implementation of
>> the larger 2^n array wrong?
>
>> Thanks for any help, Mike
>
> It is hard to tell because not all of the code is here, but...
>
> I think you still have to go with the power of two size for the textures.
>
> When you go this route, you have to normalize the texture coordinates differently and not to the [0.0, 1.0] range. It is not clear if your makePolyVerts procedure handles this.

>
> In your case, you'll want to normalize to [0, 1354.0/2048] in x and [0, 2030.0/2048] in y. The idea is that you want to limit your texture coordinates to the "active" area of the texture.
>
> Karl

Thanks for the pointer Karl, I've added the makePolyVerts procedure to elaborate on what I'm doing,

I tried to make the texture coordinates without success, but maybe you can see what I'm doing wrong by looking at my short makePolyVerts routine. Recall that the original image is 1354x2030 (as are the lats and lons arrays that correspond to the image). The pwr2Sz[2] array sizes corresponding to the larger array are both 2048,2048.

---------- makePolyverts ----------

```
pro makePolyVerts, lats, lons, MAP=map, vertexes, pwr2Sz, texCrds

    sz = size(lats,/dimensions)       ; size of original image
    vertexes = dblarr(2,sz[0],sz[1])
    texCrds = dblarr(2,sz[0],sz[1])
    ; Find the x,y values from the lat lons
    for i=0,sz[0]-1 do begin
       tmp = map_proj_forward(lons[i,*], lats[i,*] ,
map_structure=map)
       vertexes[0,i,*] = tmp[0,*]
       vertexes[1,i,*] = tmp[1,*]
    endfor


  ; Make Texture coordinates
  ; Make x max out at 1354/2048
  ; Make y max out at 2030/2048
  txmax = float(sz[0])/pwr2Sz[0]
  tymax = float(sz[1])/pwr2Sz[1]
  tDelx = txmax/sz[0]
  tDely = tymax/sz[1]
  ; Now make the texture coords so that they are 0->txmax and 0->tymax
  for i=0,sz[0]-1 do begin
     for j=0,sz[1]-1 do begin
        texCrds[0,i,j] = i*tDelx
        texCrds[1,i,j] = j*tDely
     endfor
  endfor

end
```

---------- -----

This still results in an image that seems cutoff and does not include
all the data in the original image.

Thanks, Mike

Subject: Re: Texture Map problems when using IDLgrPolygon
Posted by David Fanning on Mon, 04 Jun 2012 21:45:20 GMT
View Forum Message <> Reply to Message

mikrin writes:

> I tried to make the texture coordinates without success, but maybe you
> can see what I'm doing wrong by looking at my short makePolyVerts
> routine.

I think the explanation of the Texture Coordinates in
the documentation can lead you astray. See this article,
for example:

http://www.idlcoyote.com/ographics_tips/imgtex.html

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Texture Map problems when using IDLgrPolygon
Posted by Karl[1] on Tue, 05 Jun 2012 00:55:24 GMT
View Forum Message <> Reply to Message

On Monday, June 4, 2012 3:45:20 PM UTC-6, David Fanning wrote:
> mikrin writes:
>
>> I tried to make the texture coordinates without success, but maybe you
>> can see what I'm doing wrong by looking at my short makePolyVerts
>> routine.

> I think the explanation of the Texture Coordinates in
> the documentation can lead you astray. See this article,
> for example:
>
> http://www.idlcoyote.com/ographics_tips/imgtex.html
>
> Cheers,
>
> David
>
>
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

makePolyVerts looks reasonable.

David's referenced article talks about mapping a texture onto a subset of a surface, which is sort of the opposite of what you are trying to do. Still good reading.

I think that the number one problem here is that I don't see a POLYGONS keyword specified in your creation of the IDLgrPolygon object. You are just supplying a 2D array of 2D points, but IDLgrPolygon just interprets it as a list of 2D points and does not infer any connectivity information from the shape of the array (unless I'm missing something).

As it stands, I'm sort of surprised that anything resembling the intended output came out.

I think that you'll have to generate a polygon list. In this case, it is pretty easy because you have a regular 2D array of the points. The first polygon would be described as:

4 0 1 1355 1354

I can't think of any utilities or anything else sitting around that would make it easier. But you'll have to generate a sequence like this for every quad in your grid, by following the above pattern.

If that does not help, make sure that you are using the textureImg image for your texture. Look at it with xobjview or something to make sure that it looks right - it should be a square image with your data in the lower left, filling only part of the image.

If still fail, try converting the image from a pallete image to a rgb (3-channel) image and use that as the texture. Sometimes palette textures can be problematic.

Karl

Subject: Re: Texture Map problems when using IDLgrPolygon
Posted by Karl[1] on Tue, 05 Jun 2012 01:48:20 GMT
View Forum Message <> Reply to Message

Another thing: you have a lot of points here. Unless you zoom in super far, the quads are going to be sub-pixel in size. I would resample your vertex data after map projection and keep something like 1 out of 10 of them. Leave the image data alone and you should be ok.