Subject: Re: IdlGrPolygon - Intersection with planes and lines
Posted by Rick Towler on Tue, 19 Nov 2002 23:54:06 GMT
View Forum Message <> Reply to Message

"Martin Downing" <martin.downing@ntlworld.com> wrote

> Hi all,
>
> Has anyone written code they wish to share which will give the
intersection
> of a polyon surface* with a 3D plane, and also the intersection of a
polygon
> surface with a line (3d ray)?
> One object graphics way to do the plane intersect could be to transform
the
> object to map the plane of intersection onto the z = 0, then use a very
> small zclip range and the result would be given in image form, however I
> would very much like the "exact" geometric solution, i.e the intersect as
a
> polyline for the plane and point(s) for the line (kind of graphics gems
type
> stuff)

Hi Martin,

Could you double mesh_clip the surface? The resultant mesh would still be a polygon though. Or, could you mesh_clip, where() the verts that lie on the clipping plane, and somehow reconstruct the polyline connectivity? That last bit might be sticky.

I've been playing with some polyhedron/box intersection code. Not what you are looking for, but in my search I have found a few resources which might be helpful if you can't find an all IDL solution.

There is the magic software archive:

http://www.magic-software.com/Intersection3D.html

which is a large C++ archive of many 3d graphics routines. I know he has a plane/ plane intersection test which returns the line of intersection which you could use if your surface was meshed with quads. I played around with some of the magic code in a dlm. You have to work a bit to get the data into a form that works with his classes and buy the book if you want any decent docs, there are few comments in the source.

Realtime Rendering:

http://www.realtimerendering.com/#isect

A section on intersection testing. Has some of the links I have provided. Also links to this on the same site:

http://www.realtimerendering.com/int/

This web page has a table of static object intersection testing methods. It lists a few sources for triangle/triangle tests which you could use if your surface was meshed with tris.

There is also the Graphics Gems archive:

http://www.acm.org/tog/GraphicsGems/

This page might be helpful in tracking down which gem you'll need to get from your local library. I can tell you that there is little on this in Gems V. I also can tell you the Gems source code is where you'll probably want to start. It is implemented in C and tends to be very straightforward. Easier for us IDL'ers to chew on.

Hopefully this will get you started.

-Rick

Subject: Re: IdlGrPolygon - Intersection with planes and lines
Posted by Dick Jackson on Wed, 20 Nov 2002 00:36:29 GMT
View Forum Message <> Reply to Message

"Martin Downing" <martin.downing@ntlworld.com> wrote in message news:6svC9.1136$Bc.182603@newsfep2-win.server.ntli.net...
> Hi all,
>
> Has anyone written code they wish to share which will give the intersection
> of a polyon surface* with a 3D plane, and also the intersection of a polygon
> surface with a line (3d ray)?
> One object graphics way to do the plane intersect could be to transform the
> object to map the plane of intersection onto the z = 0, then use a very
> small zclip range and the result would be given in image form, however I
> would very much like the "exact" geometric solution, i.e the intersect as a

> polyline for the plane and point(s) for the line (kind of graphics gems type
> stuff)

Hi Martin,

For the plane vs. mesh, I think you'll find what you want in the Mesh_Clip routine and its Cut_Verts output keyword. IDL 5.6 Help has a nice example program, I hope RSI doesn't mind me posting it here:

```
PRO ClippingAMesh

; Create a mesh of an octahedron.
vertices = [[0, -1, 0], [1, 0, 0], [0, 1, 0], $
  [-1, 0, 0], [0, 0, 1], [0, 0, -1]]
connectivity = [4, 0, 1, 2, 3, 3, 0, 1, 4, 3, 1, 2, 4, $
  3, 2, 3, 4, 3, 3, 0, 4, 3, 1, 0, 5, 3, 2, 1, 5, $
  3, 3, 2, 5, 3, 0, 3, 5]

; Initialize model for display.
oModel = OBJ_NEW('IDLgrModel')

; Initialize polygon and polyline outline to contain
; the mesh of the octahedron.
oPolygon = OBJ_NEW('IDLgrPolygon', vertices, $
  POLYGONS = connectivity, SHADING = 1, $
  COLOR = [0, 255, 0])
oPolyline = OBJ_NEW('IDLgrPolyline', vertices, $
  POLYLINES = connectivity, COLOR = [0, 0, 0])

; Add the polygon and the polyline to the model.
oModel -> Add, oPolygon
oModel -> Add, oPolyline

; Rotate model for better initial perspective.
oModel -> Rotate, [-1, 0, 1], 22.5

; Display model.
XOBJVIEW, oModel, /BLOCK, SCALE = 1, $
  TITLE = 'Original Octahedron Mesh'

; Clip mesh.
clip = MESH_CLIP([1., 1., 1., 0.], vertices, connectivity, $
  clippedVertices, clippedConnectivity, $
  CUT_VERTS = cutVerticesIndex)

; Update polygon with the resulting clipped mesh.
oPolygon -> SetProperty, DATA = clippedVertices, $
```

```
   POLYGONS = clippedConnectivity

; Display the updated model.
XOBJVIEW, oModel, /BLOCK, SCALE = 1, $
   TITLE = 'Clipped Octahedron Mesh'

; Determine the vertices of the clipped plane.
cutVertices = clippedVertices[*, cutVerticesIndex]

; Derive the x and y components of the clipped plane's
; vertices.
x = cutVertices[0, *]
y = cutVertices[1, *]

; Triangulate the connectivity of the clipped plane.
TRIANGULATE, x, y, triangles

; Derive the connectivity of the clipped plane from the
; results of the triangulation.
arraySize = SIZE(triangles, /DIMENSIONS)
array = FLTARR(4, arraySize[1])
array[0, *] = 3
array[1, 0] = triangles
cutConnectivity = REFORM(array, N_ELEMENTS(array))

; Initialize the clipped plane's polygon and polyline.
oCutPolygon = OBJ_NEW('IDLgrPolygon', cutVertices, $
   POLYGONS = cutConnectivity, SHADING = 1, $
   COLOR = [0, 0, 255])
oCutPolyline = OBJ_NEW('IDLgrPolyline', cutVertices, $
   POLYLINES = cutConnectivity, COLOR = [255, 0, 0], $
   THICK = 3.)

; Add polyline and polygon to model.
oModel -> Add, oCutPolyline
oModel -> Add, oCutPolygon

; Display updated model.
XOBJVIEW, oModel, /BLOCK, SCALE = 1, $
   TITLE = 'Clipped Octahedron Mesh with Clipping Plane'

; Clean-up object references.
OBJ_DESTROY, [oModel]

END
```

I might also mention that IDLDE 5.6 has the excellent
multi-line-command-input-pasting feature, so I was able to copy all the

lines of this routine, paste them straight into the command input and it worked a charm! (that's one more feature caught up to idlwave-shell... about 300 to go! ;-)... I still want a command-line IDL for Windows, any new insight, anyone?)

For the line vs. mesh, I once found some notes once that may apply, but I've wondered if you couldn't do it by defining your line as the intersection of two planes, use Mesh_Clip twice in succession and get your result from that somehow. These are the notes I found from a helful source at Boston University:

http://cas00.bu.edu/course/cs580/spring2001/mishka/p2/

Hope this is of some help.

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392