OpenCASCADE PCurve of Topological Face

eryar@163.com

Abstract. OpenCASCADE provides a class BRepBuilderAPI_MakeFace to build topological faces. A face maybe built from a surface, elementary surface from gp package, surface from Geom, from a wire and find the surface automatically if possible, etc. If a face is built, how to check it for visualization? What does PCurve means? The paper will answer those question.

Key Words. OpenCASCADE, Topological Face, PCurve, Holes

1. Introduction

OpenCASCADE 中边界表示法 BRep 的拓朴面 Topological Face 包含了完整的几何信息，即给定一个 TopoDS_Face，其中包含了边和顶点，而边和顶点包含了几何曲线及三维点。对边和顶点的可视化很好理解：显示顶点就是在场景中绘制一个三维点；显示边最简单的算法可以将边中的曲线在参数空间中等分采样，再将参数对应到曲线上的点连接起来就可以简单显示边了；而面的可视化如何实现呢？

几何造型内核中都有个参数曲线的概念，即 PCurve(Parametric Curve)，它是实现面可视化的一个很关键的数据。PCurve 的定义是参数表示的曲面上的曲线在二维(u, v)参数空间中的二维样条曲线，也就是曲面上的曲线（Curve on Surface）。

在理解 PCurve 定义的基础上才好对其做进一步的研究，好回答“从哪儿来到哪儿去的问题”，即如何产生 PCurve，如何使用生成的 PCurve 数据。本文主要介绍如何将 OpenCASCADE 拓朴面中的 PCurve 可视化，从而方便对拓朴面的检查，也可以看出 PCurve 在曲可视化方面的应用。PCurve 的产生及其他应用有待进一步挖掘。若您对 PCurve 有何看法，欢迎不吝赐教。

2. PCurve of a Face

由 OpenCASCADE 中对拓朴形状可视化的算法[2]可知，对面的网格化需要先对面中环，环中的边进行离散化，最后都统一到参数空间，即只需要一个二维网格化算法来对二维的参数空间进行网格化，最后将参数空间网格化的点映射回面中的几何曲面上，即得到曲面的空间网格剖分。

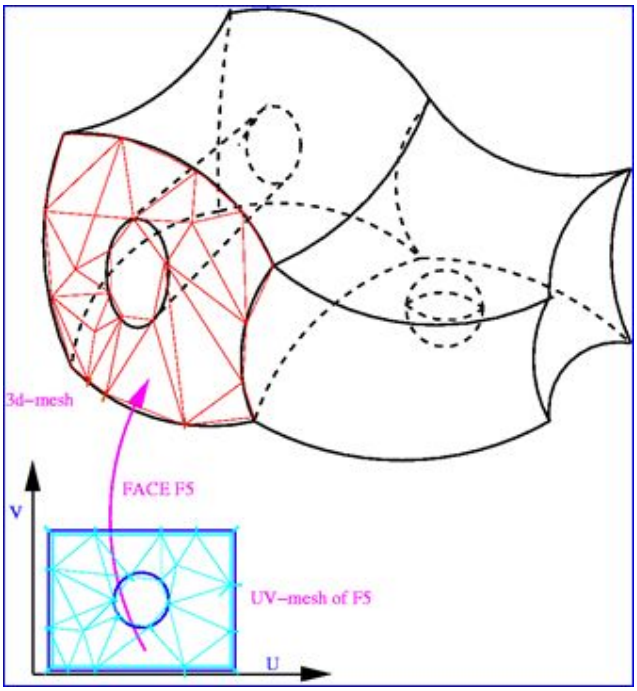


Figure 2.1 Mesh UV domain of a Surface

如上图 2.1 所示，UV 参数空间中的边界线就是拓朴面的环中的边的 PCurve，即拓朴面中的环对应了参数空间中的边界，而这些边界的表示就是使用了 PCurve。对参数空间进行网格化最常见的算法就是 Delaunay 三角剖分算法[3]，早期 OpenCASCADE 的版本中使用了一个开源 Delaunay 库 Triangle[4]。通过将曲面在参数空间的结果映射回曲面的三维空间即可将曲面可视化了。如何控制曲面离散精度还有待进一步学习。

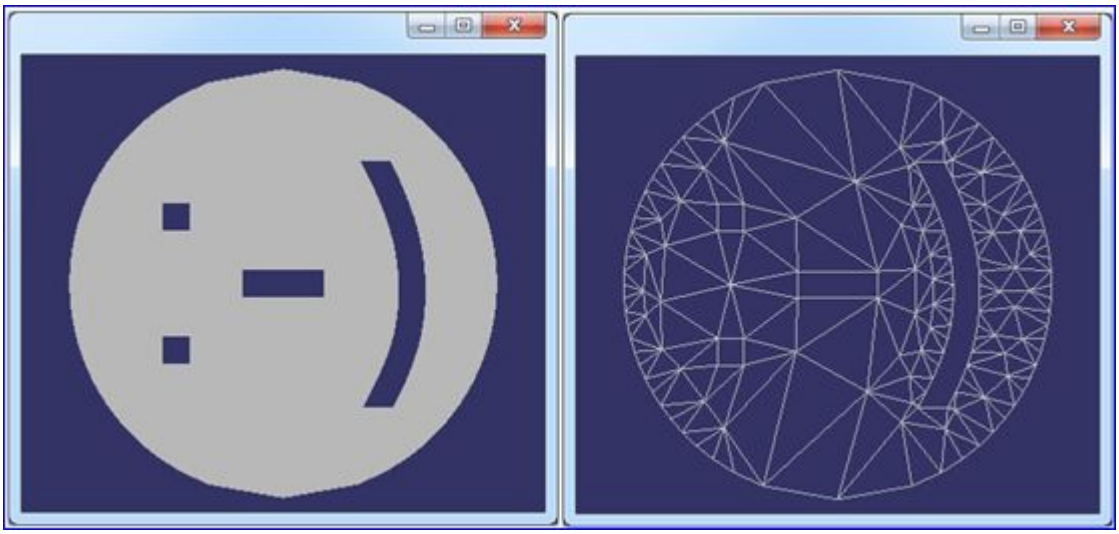


Figure 2.2 A Smiley Face Meshed by Triangle

上图 2.2 所示为二维三角剖分库 Triangle 对一个笑脸进行剖分的结果。

在 Draw Test Harness 中，OpenCASCADE 提供了对面的 PCurve 可视化的命令 pcurve。使用 pcurve 命令可以将一个面中所有的 pcurve 根据朝向 orientation 以不同的颜色进行显示。这个命令对检查面中边的朝向的正确性非常有用。下面使用 Tcl 命令在 Draw Test Harness 中对基本曲面的 PCurve 进行显示。

2.1 Plane PCurve

OpenCASCADE 中的平面的参数方程为：

$$S(u,v) = P + u \cdot D_u + v \cdot D_v,\ (u,\ v) \in (-\infty,\ \infty) \times (-\infty,\ \infty).$$

由上述参数表示的平面方程可知，平面的定义域是无穷的，所以为了生成一个有环的拓朴面，需要对平面设置边界来对无限的平面进行裁剪。相应的 Tcl 脚本如下所示：

```
# 1. view the pcurves of a plane face
plane p

# trim the plane to (u,v)->[-1, 1][-1, 1]
trim p p -1 1 -1 1

# make the topo face
mkface p p

# extract the 2d curve of an edge on a face
pcurve p

# display pcurve in 2d viewer
av2d
fit
2dfit

# display face in 3d viewer
vdisplay p
```

生成结果如下图所示：

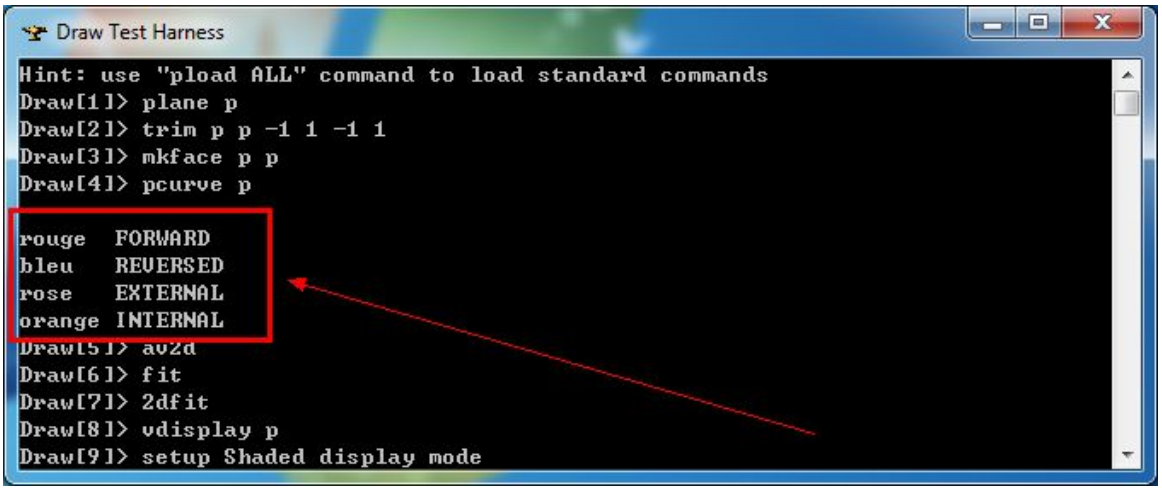


Figure 2.3 Color for PCurves

由图 2.3 可知，pcurve 有四种颜色：

- rouge: FORWARD 胭脂红表示正向；
- bleu: REVERSED 蓝色表示反向；（不知道是法语写法还是个错别字，蓝色英语应该为 blue）
- rose: EXTERNAL 玫瑰红表示向外；
- orange: INTERNAL 橙黄色表示向内；

根据上述的颜色规则来看图 2.4，可以看出平面边界的 pcurves 是逆时针闭合的。

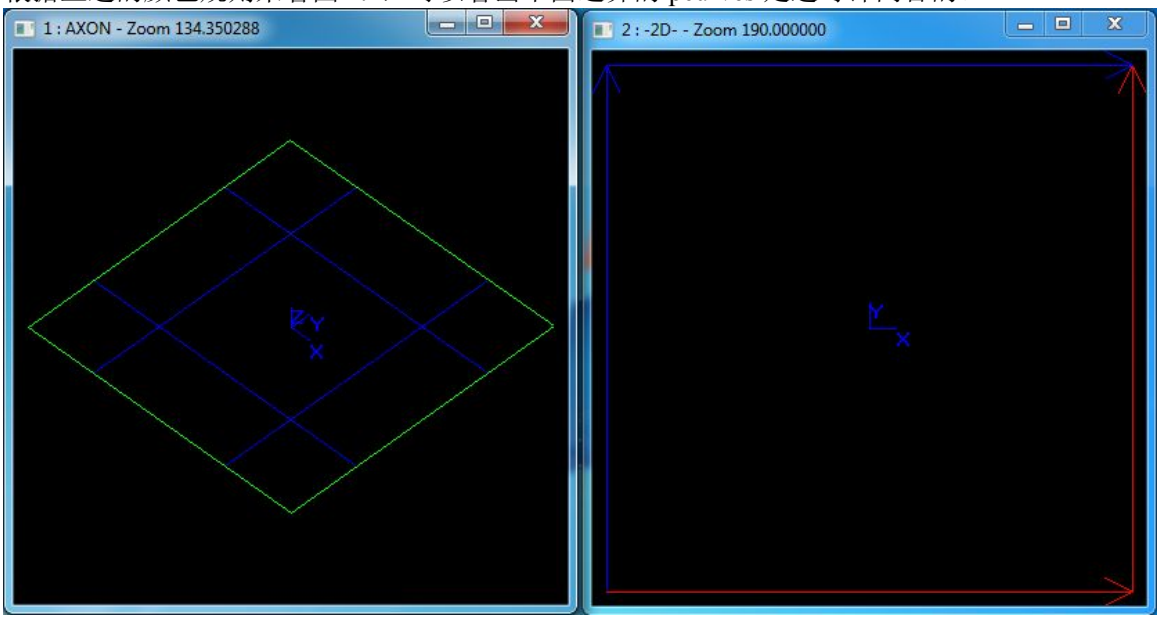


Figure 2.4 Plane PCurves

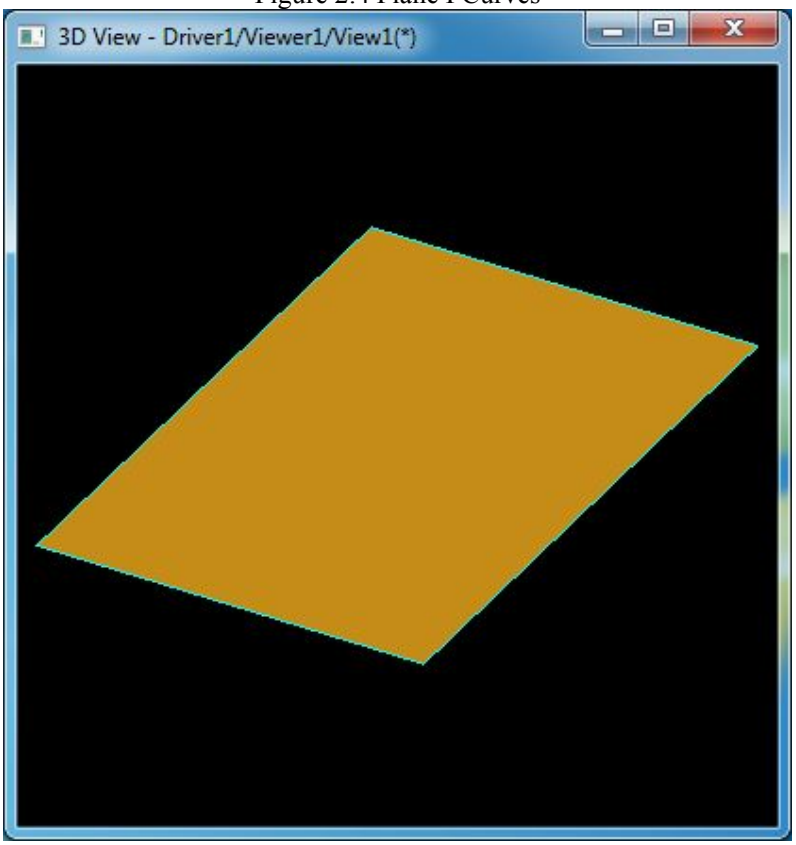


Figure 2.5 Plane Face in 3D Viewer

2.2 Cylinder PCurve

OpenCASCADE 中圆柱面的参数方程为：

$$S(u,v) = P + r \cdot (cos(u) \cdot D_x + sin(u) \cdot D_y) + v \cdot D_v,\ (u,v) \in [0,\ 2 \cdot \pi) \times (-\infty,\ \infty).$$

由上述参数方程可知，u 的取值范围是有界的[2, 2pi)，v 的取值是无限的。所以为了得到有界的拓朴面，至少需要对其 v 方向进行裁剪。相应的 Tcl 脚本如下所示：

```
# 2. view the pcurves of a cylinder face
cylinder c 1

# trim the cylinder to (u,v)->[0, 2pi][0, 1]
trim c c 0 2*pi 0 1

# make the topo face
mkface c c

# extract the 2d curve of an edge on a face
pcurve c

# display pcurves in 2d viewer
av2d
2dfit
fit

# display face in 3d viewer
vdisplay c
```

生成结果如下图所示：

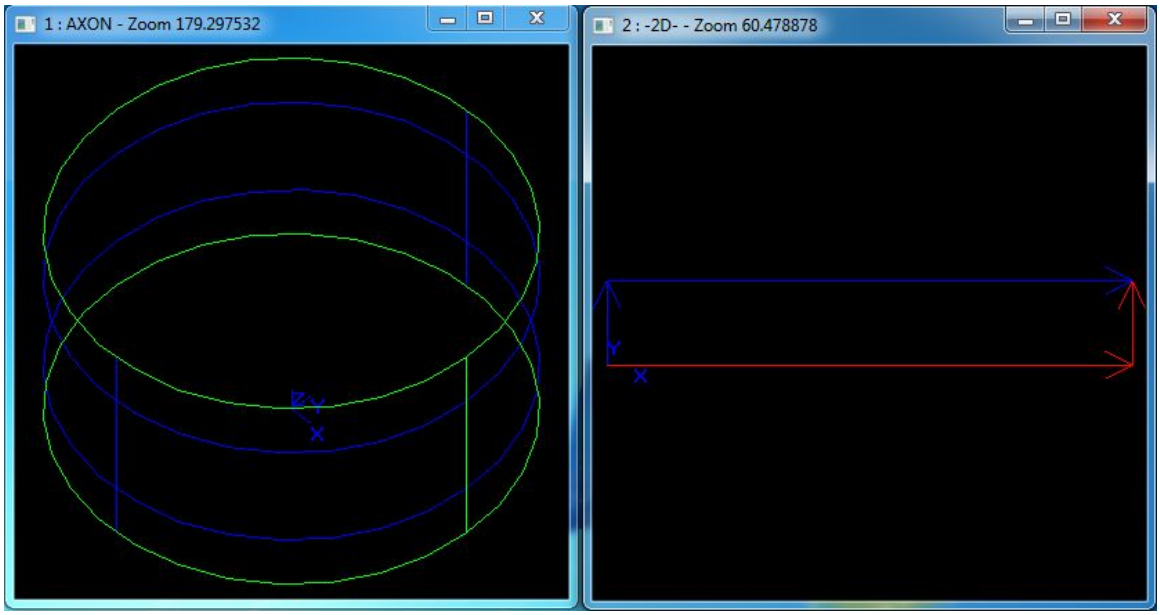


Figure 2.6 Cylinder PCurves

由上图根据 pcurve 的着色规则可知，圆柱面的 pcurves 也是按逆时针顺序闭合的。其在三维中的显示结果如下图所示：

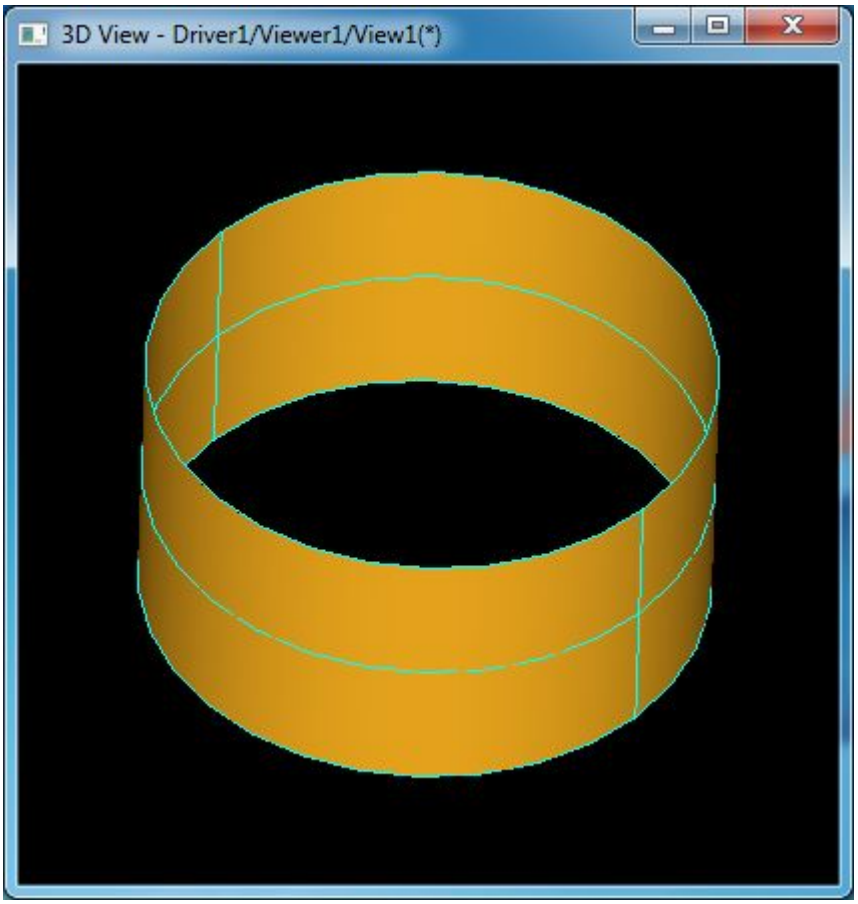


Figure 2.7 Cylinder Face in 3d Viewer

3.3 Cone PCurve

OpenCASCADE 中圆锥面的参数方程为：

$$S(u,v) = P + (r + v \cdot sin(\varphi)) \cdot (cos(u) \cdot D_X + sin(u) \cdot D_Y) + v \cdot cos(\varphi) \cdot D_Z, (u,v) \in [0,\ 2 \cdot \pi) \times (-\infty,\ \infty).$$

可知圆锥曲面与圆柱曲面一样，都是在 u 方向有界，在 v 方向无界。对其进行裁剪生成拓朴面并显示 pcurve 的 Tcl 脚本如下所示：

```
# 3. view the pcurves of a cone face
cone co 30 0

# trim the cone to (u,v)->[0, 2pi][0, 1]
trim co co 0 2*pi 0 1

# make the topo face
mkface co co

# extract pcurves
pcurve co

# display pcurves in 2d viewer
```

```
av2d
2dfit
fit

# display face in 3d viewer
vdisplay co
```

生成结果如下图所示：

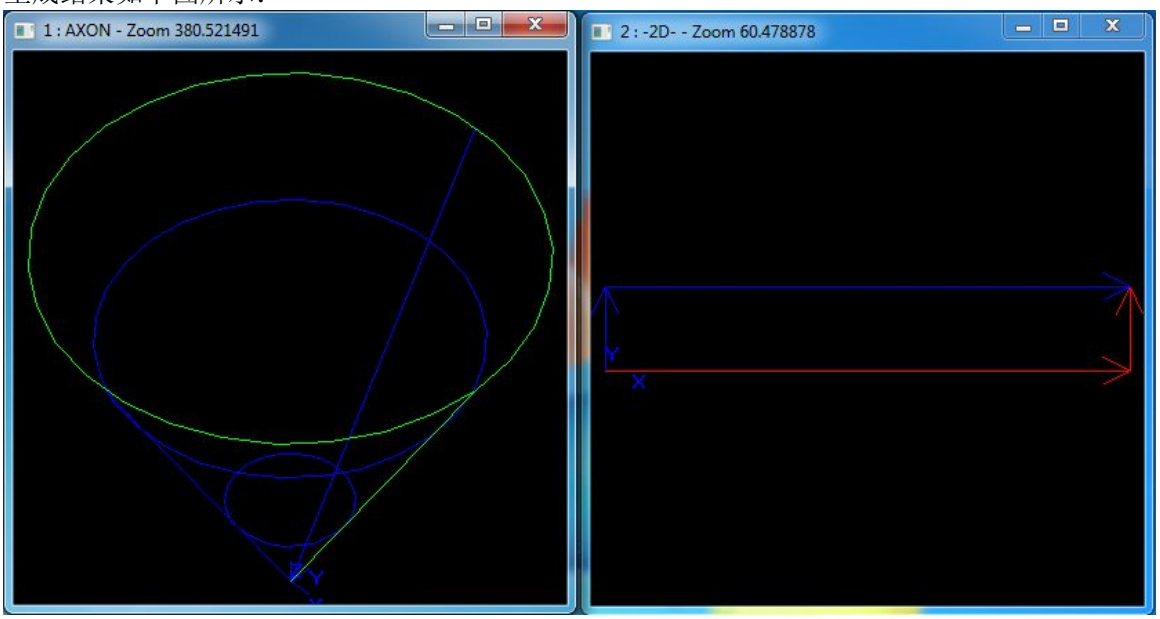


Figure 2.8 Cone PCurves

由上图根据 pcurve 的着色规则可知，圆锥面的 pcurves 也是按逆时针顺序闭合的。其在三维中的显示结果如下图所示：

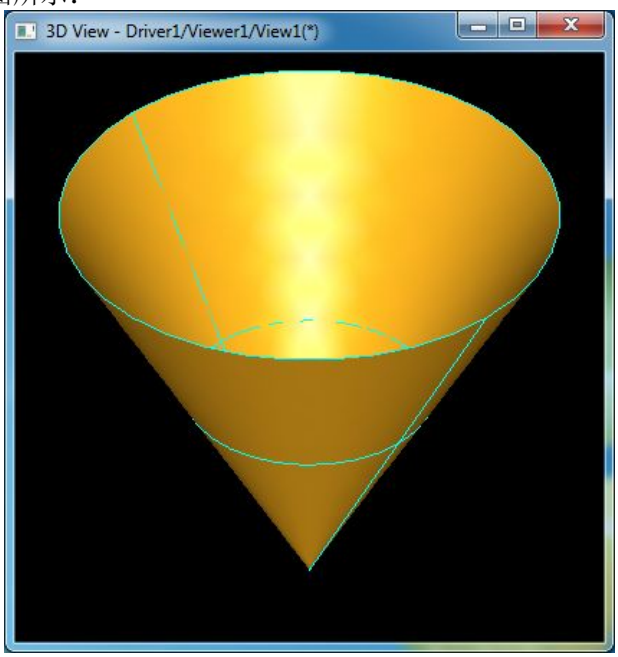


Figure 2.9 Cone Face in 3D viewer

3.4 Sphere PCurve

OpenCASCADE 中球面的参数方程为：

$$S(u,v) = P + r \cdot cos(v) \cdot (cos(u) \cdot D_x + sin(u) \cdot D_y) + r \cdot sin(v) \cdot D_Z,\ (u,v) \in [0,\ 2 \cdot \pi) \times [-\pi/2,\ \pi/2].$$

由上述参数方程可知，球面在 u 和 v 方向均为有界的，所以可不用对其进行裁剪就可生成拓朴面，当然也可对其裁剪得到球面的部分。显示球面 pcurves 的 Tcl 脚本如下所示：

```
# 4. view the pcurves of a sphere face
sphere s 1

# make the topo face
mkface s s

# extract pcurves
pcurve s

# display pcurves in 2d viewer
av2d
2dfit
fit

# display sphere face in 3d viewer
vdisplay s
```

生成结果如下图所示：

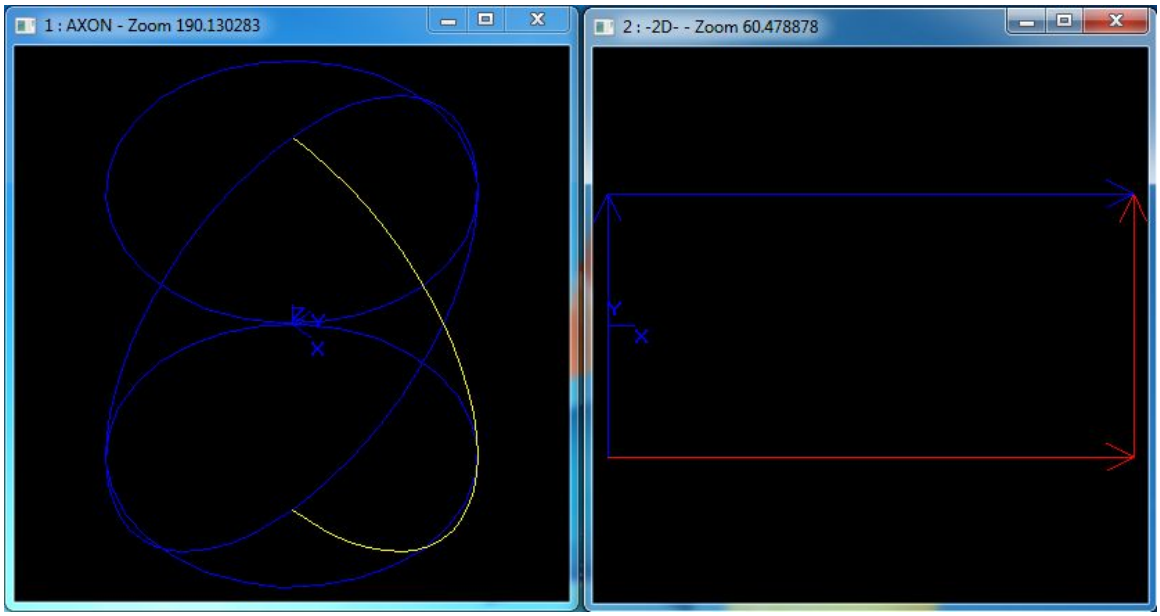


Figure 2.10 Sphere PCurves

由上图根据 pcurve 的着色规则可知，球面的 pcurves 也是按逆时针顺序闭合的。其在三维中的显示结果如下图所示：

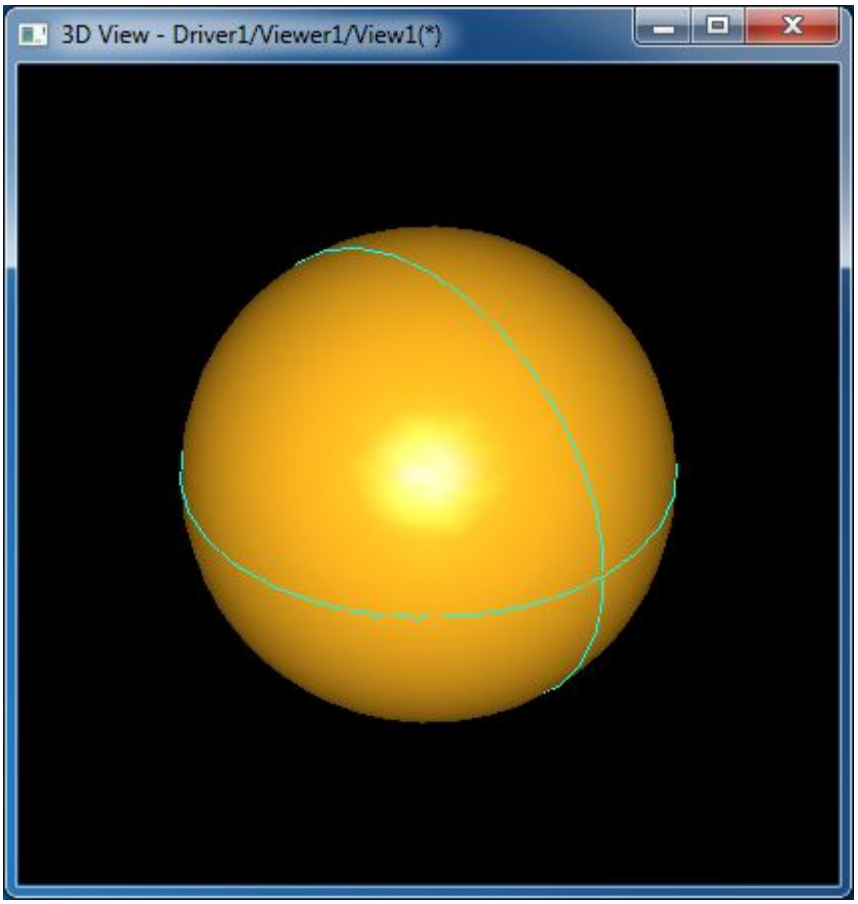


Figure 2.11 Sphere in 3d viewer

3.5 Torus PCurve

OpenCASCADE 中圆环面的参数方程为：

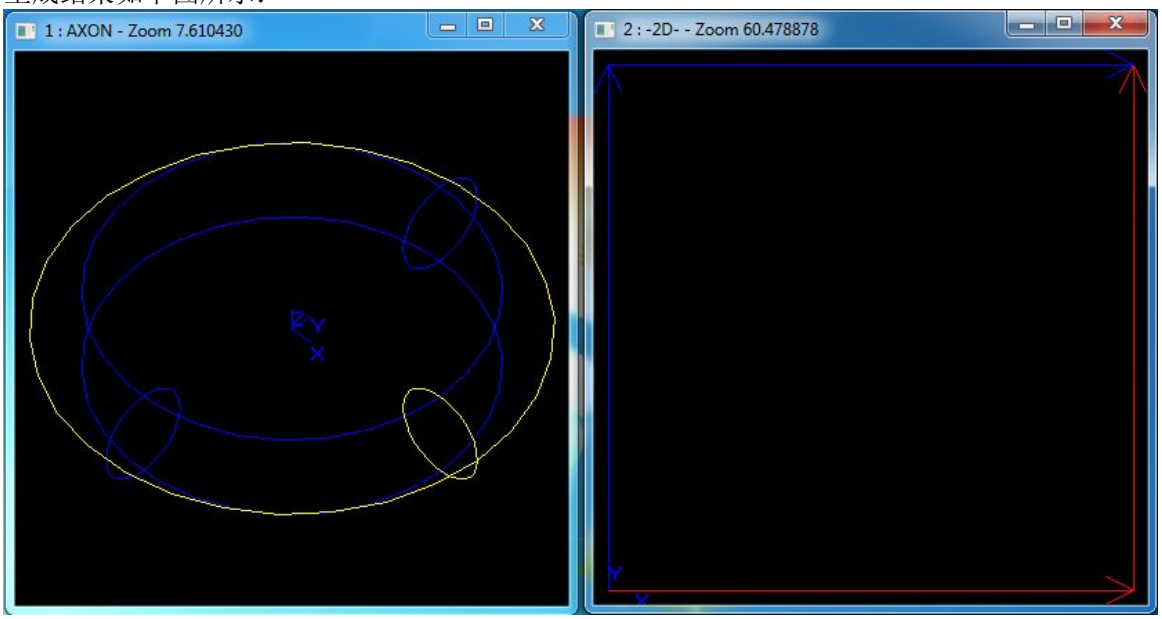
$$S(u,v) = P + (r_1 + r_2 \cdot cos(v)) \cdot (cos(u) \cdot D_x + sin(u) \cdot D_y) + r_2 \cdot sin(v) \cdot D_Z,\ (u,v) \in [0,\ 2 \cdot \pi) \times [0,\ 2 \cdot \pi).$$

由圆环面的参数方程可知，在参数区间上 u 和 v 都是有界的，所以可不用对其进行裁剪就可生成拓朴面，当然也可对其裁剪得到圆环面的部分。显示圆环面 pcurves 的 Tcl 脚本如下所示：

```
# 5. view the pcurves of a torus face
torus t 20 5

# make the topo face
mkface t t

# extract pcurves
pcurve t

# display pcurves in 2d viewer
av2d
2dfit
fit
# display torus in 3d viewer
```

```
vdisplay t
```

生成结果如下图所示：



Figuer 2.12 Torus PCurves

由上图根据 pcurve 的着色规则可知，圆环面的 pcurves 也是按逆时针顺序闭合的。其在三维中的显示结果如下图所示：

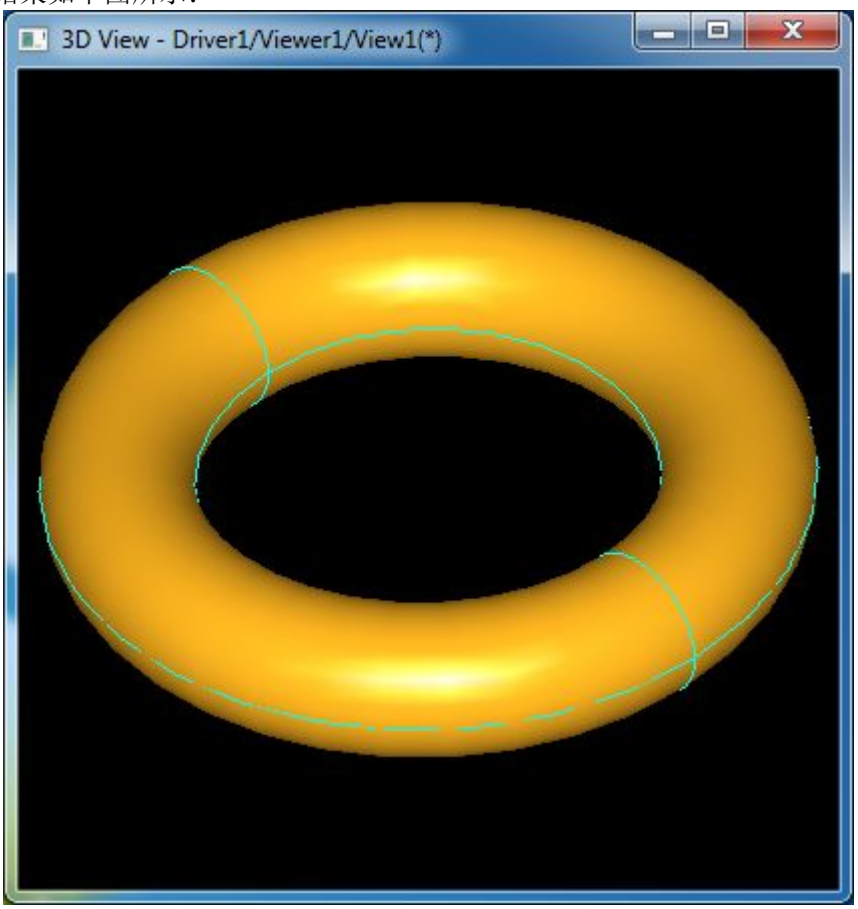


Figure 2.13 Torus in 3d viewer

3. PCurve of a Face With Holes

由 OpenGL 编程指南[10]可知，要对一个 NURBS 曲面进行裁剪，可以创建 gluPwlCurve 和 gluNurbsCurve 来在参数空间形成闭合区域。其中 gluPwlCurve 创建是多段直线，而 gluNurbsCurve 生成的是在单位参数空间的 NURBS 曲线。创建裁剪曲线时需要考虑曲线的朝向（orientation），即曲线是顺时针的还是逆时针的。曲线裁剪曲面的方式很简单，想像你沿着曲线走，左手边的将会被保留，右手边的将会被去除。

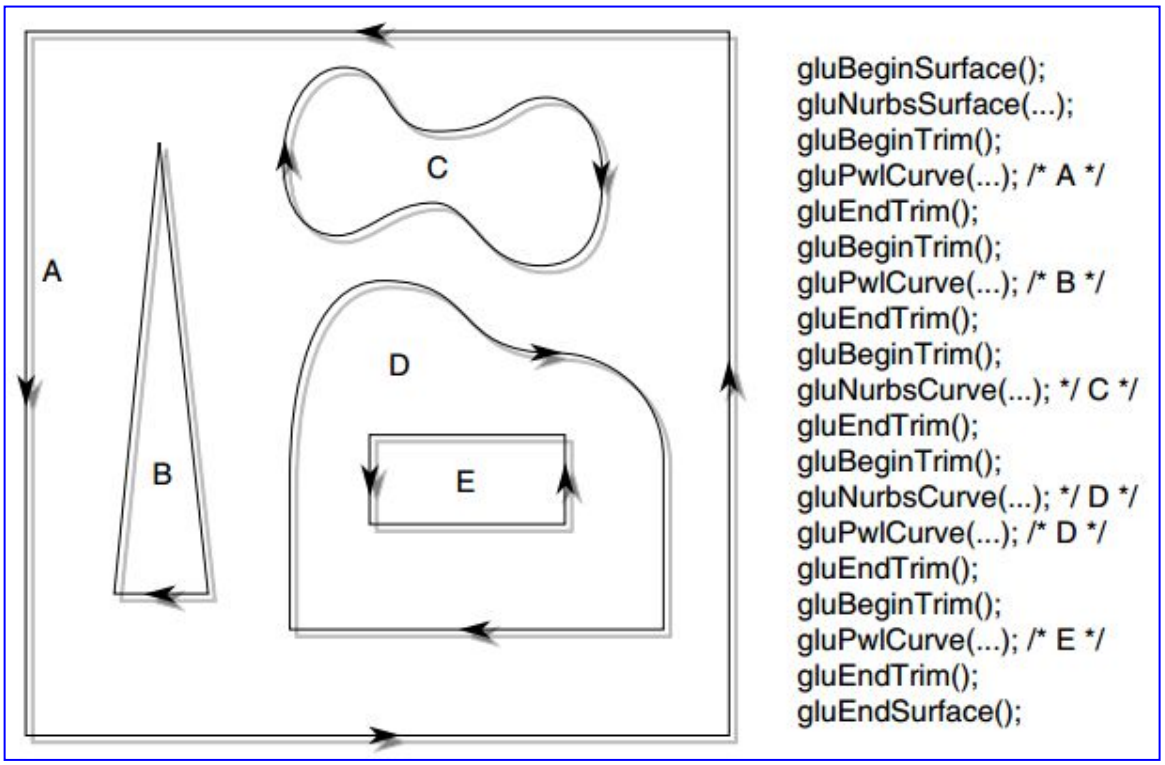


Figure 3.1 Parametric Trimming Curves

裁剪曲线还必须闭合且不能自交（Trimming curves must be closed and nonintersecting）。这里的裁剪曲线与 OpenCASCADE 中的参数曲线 pcurve 的概念相同。OpenCASCADE 中的面也是采用的相同的规则。下面通过 Tcl 脚本来测试 OpenCASCADE 中带有开孔的面的 pcuve 是否满足 OpenGL 中裁剪曲面的规则。

```
# test face with one hole
plane p
trim p p -10 10 -10 10
mkface p p
pcylinder c 1 2

bop p c
bopcut s

vdisplay s

explode s F
pcurve s_1

av2d
```

```
2dfit
fit
```

上述 Tcl 脚本为将一个平面用圆柱去挖一个孔，如下图所示：

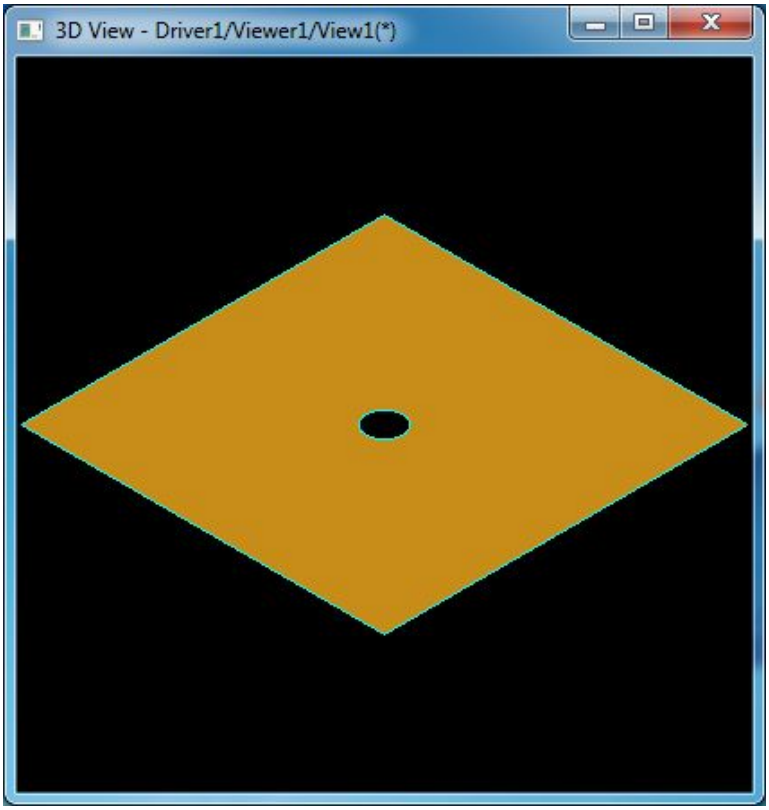


Figure 3.2 Face with a Hole

遍历被挖孔（boolean operation）得到的形状的面，得到一个面，显示这个面的 pcurve 如下图所示：

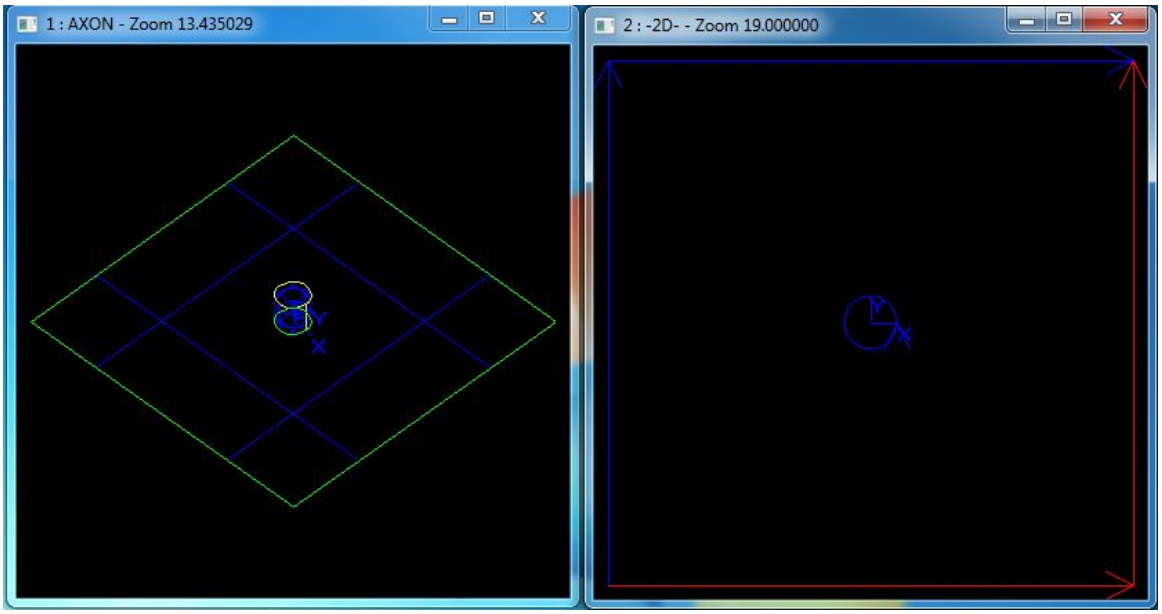


Figure 3.3 PCurve of a Face with one hole

由上图 3.3 可知，pcurve 的规则与 OpenGL 中的裁剪曲线一致。孔的 pcurve 为蓝色，即为逆时针的反向：顺时针。当一个面上生成多个孔时，是否仍然满足上述规则呢？下面使用 Tcl 来对验证一下：

```
# test pcurve of a face with multi-holes
sphere s 10
mkface s s

pcylinder c1 1 30
pcylinder c2 1 30

ttranslate c1 -5 -5 -15
ttranslate c2 5 5 -15

bop s c1
bopcut s

bop s c2
bopcut s

explode s F
pcurve s_1

av2d
2dfit
fit
```

用脚本在一个球面上生成四个孔，如下图所示：

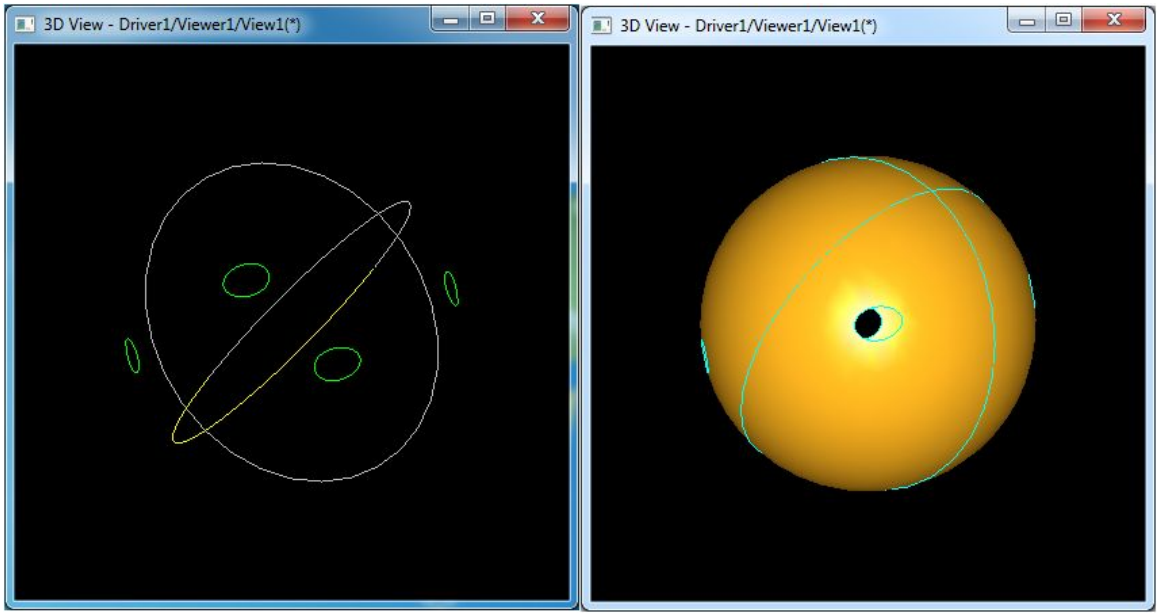


Figure 3.4 A sphere face with 4 holes

遍历 boolean operation 生成孔后的形状的面，可看到只生成了一个面。将这个面的 pcurve 显示出来如下图所示：

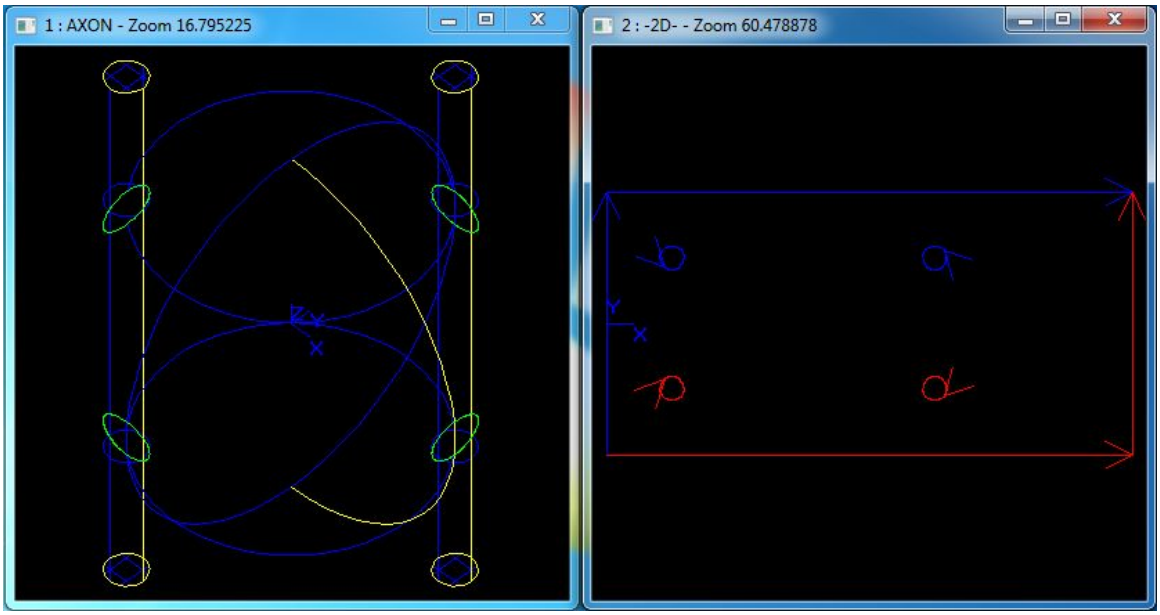


Figure 3.5 PCurves for a sphere with 4 holes

由上图 3.5 可知，pcurve 的朝向也与 OpenGL 中裁剪曲线的朝向一致。当一个面生成的开孔如图 3.1 中的 D 和 E 时，在 OpenCASCADE 中是如何表示的呢？下面也有 Tcl 脚本测试测试：

```
plane p
trim p p -10 10 -10 10
mkface p p

ptorus t 5 1

bop p t
bopcut s

vdisplay s

explode s F
pcurve s_1
pcurve s_2

av2d
2dfit
fit
```

生成结果如下图所示：

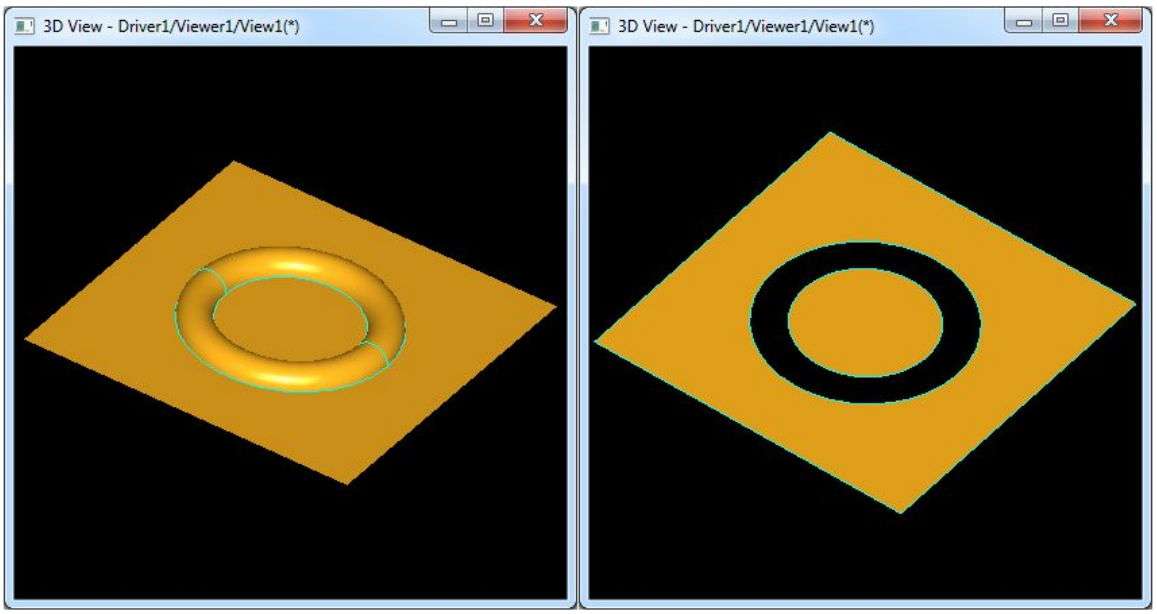


Figure 3.6 A Plane cut a Torus

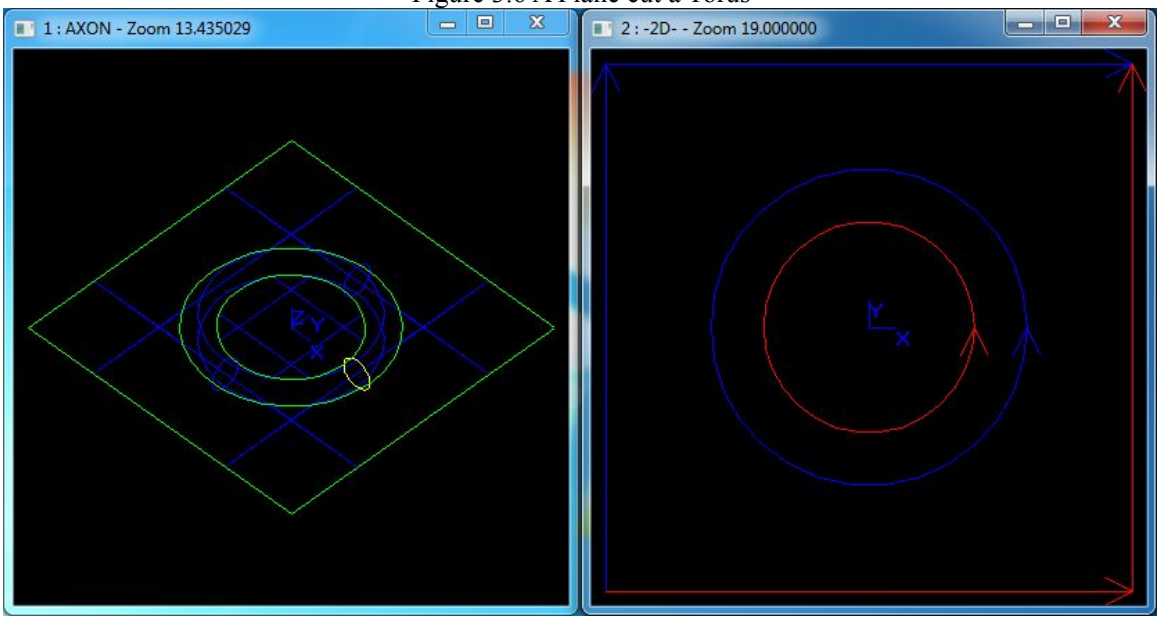


Figure 3.7 PCurves for the faces

由上图可知，当一个平面去掉一个圆环面后，生成了两个面，与有些几何内核的用链表来将结果表示成一个面不同，OpenCASCADE 中将结果生成了两个面。其 pcurves 的朝向也与 OpenGL 中的裁剪曲线的朝向一致。

4. Conclusion

综上所述，曲面上的曲线 pcurve 的概念是一个非常重要的概念，理解 pcurve 对造型及可视化有着重要意义。本文结合 OpenGL 中裁剪曲线的规则及 OpenCASCADE 的 Draw Test Harness 中显示拓朴面 pcurve 的命令，来对基本曲面及裁剪曲面的 pcurve 的朝向进行检验。掌握了这些规则，可方便对自己构造拓朴面的正确性进行检验。

5. Acknowledge

时光荏苒，从毕业到现在不经意间就到了而立之年。沧海桑田，岁月蹉跎，经历春夏秋冬的四季，品尝酸甜苦辣的人生。感谢一路走来，亲人、朋友、同事等对给予我的支持，信任和鼓励，让我可以做自己喜欢的事情，找到人生的方向。

寄蜉蝣与天地，渺沧海之一粟。准备回到离家近的武汉工作，切换到生活模式，多些时间陪伴父母亲人，以报养育之恩。

6. References

1. Shing Liu. PCurve - Curve on Surface.
   http://www.cnblogs.com/opencascade/p/3601859.html
2. Shing Liu. Topology and Geometry in OpenCASCADE-Edge.
   http://www.cnblogs.com/opencascade/p/3604052.html
3. Topology and Geometry in OpenCASCADE-Face.
   http://www.cnblogs.com/opencascade/p/3605729.html
4. Shing Liu. Mesh Algorithm in OpenCASCADE.
   http://www.cnblogs.com/opencascade/p/3648532.html
5. Shing Liu. Delaunay Triangulation in OpenCASCADE.
   http://www.cppblog.com/eryar/archive/2013/05/26/200605.aspx
6. Shing Liu. Triangle-Delaunay Triangulator.
   http://www.cnblogs.com/opencascade/p/3632705.html
7. OpenCASCADE, Draw Test Harness User Guide.
8. OpenCASCADE, BRep Format White Paper.
9. Richard S. Wright Jr., Benjamin Lipchak. OpenGL SuperBible. Sams Publishing. 2004
10. Dave Shreiner. OpenGL Programming Guide. Addison-Wesley. 2009