OpenCASCADE Hidden Line Removal

eryar@163.com

Abstract. To provide the precision required in industrial design, drawings need to offer the possibility of removing lines, which are hidden in a given projection. OpenCASCADE provides two algorithms for this Hidden Line Removal component. The paper mainly translate the document of OpenCASCADE Modeling Algorithms, and give some applications in the plant design CAD software.

Key Words. OpenCASCADE, HLR, Hidden Line Removal

1. Introduction

用计算机生成三维物体的真实图形是计算机图形学研究的重要内容，在用显示设备描述物体时，必须把三维的信息经过某种投影变换，在二维的显示平面上绘制出来。由于投影变换失去了深度信息，往往导致图形的二义性。要消除这类二义性，就必须在绘制时消除被遮挡的不可见的线或面，习惯上称之为消除隐藏线（Hidden Line Removal）或消除隐藏面（Hidden Face Removal）。在工程应用中，需要根据三维模型自动生成二维的图纸，用于指导生产。其中二维图纸中主要包括三维模型的消隐图、尺寸标注及件号标注等内容。如图 1.1 所示为某 CAD 软件中自动根据三维模型生成二维图纸的效果图：

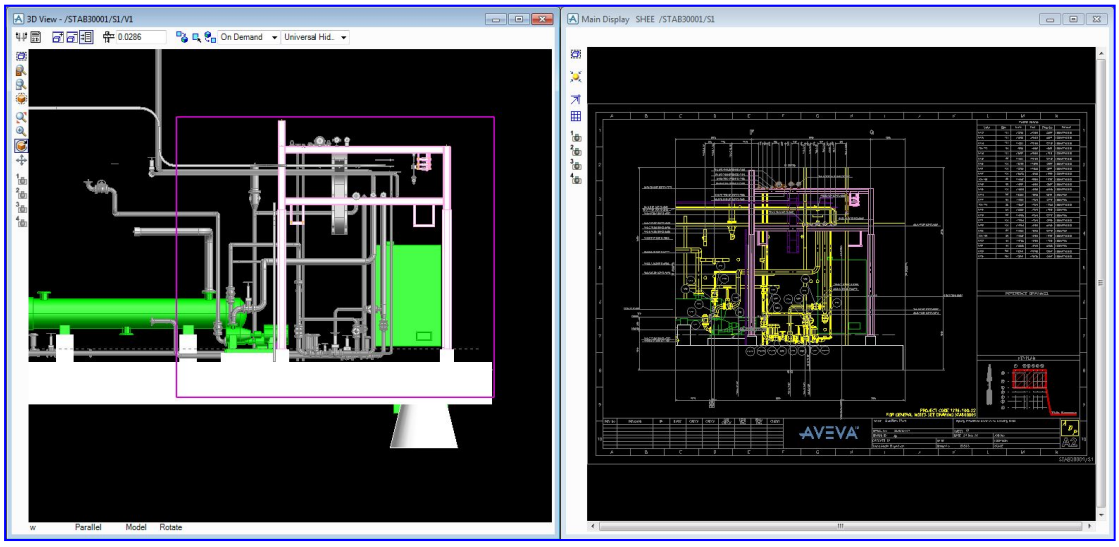

Figure 1.1 Drawing generated from 3D model by PDMS

上图 1.1 所示为 PDMS 软件中自动生成的图纸，图纸中的图形区的管道模型就是根据三维模型自动投影及消隐后生成的。还生成尺寸标注及管道名称，以及右上角所件号标示或材料表等相关信息。

尽管现在 3D PDF 格式很流行，但是二维的生产图纸在目前国内的设计及施工单位中还是不可或缺的。当模型量大时，消隐速度快及自动生成的标注文字排列整齐（或满足工程习惯）成了二维图纸自动生成的核心技术，也是程序处理中的难点。

消隐算法的原理其实很简单，只要满足两个条件：

- 物体 A 在物体 B 的后面；

❖ 物体 A 与物体 B 在投影平面上有重叠部分；

前一个条件实际上是广义的，既可以是物体，也可以是面或线等。命题物体 A 在物体 B 后面成立，消隐计算就变成一个二维问题：物体 A 与物体 B 在投影平面上的重叠部分就是 A 被消除的部分。经过投影变换后，物体在投影平面上所占据的区域称为物体的落影区，物体上任何一点的投影均落在此落影区内。显然，若空间有两个物体的落影区是重叠的，则位于后面的物体将被前面的物体遮挡，被遮挡的部分就是落影区重叠的部分。消隐过程就是求取两者的公共部分，且由第三维深度坐标来判断两者的前后的过程。因为是线输出，这个过程就是一条条线与每一物体（面）的比较过程，最后可见部分的交集即为此线的最终可见部分。

OpenCASCADE 提供了两种消隐算法：HLRBRep_Algo 和 HLRBRep_PolyAlgo。这些算法都是基于相同的原理：比较形状每条边相对每个面的可见性，并计算每条边的可见部分与消隐部分。算法通过计算在指定投影方向上的物体显示特性，去除或标记被面遮挡的边。这两个算法也与一些提取功能配合使用，如重构一个简化的模型等，简化后新的模型由边组成，就是在投影方向上的轮廓线。

HLRBRep_Algo 是根据模型来计算的一种高精度的算法，而 HLRBRep_PolyAlgo 是基于离散数据的算法。当使用 HLRBRep_Algo 时可以得到精确结果，而使用 HLRBRep_PolyAlgo 可以提高计算速度。他们两个算法都可以处理任意类型的模型，如组合体、面或线，但也有些约束，如下情况就未被处理：

❖ 点未被处理；

❖ Z 平面上没有被裁剪；

❖ 无限面或线没有处理；

如图 1.2 所示为 OpenCASCADE 中的一些边的定义：

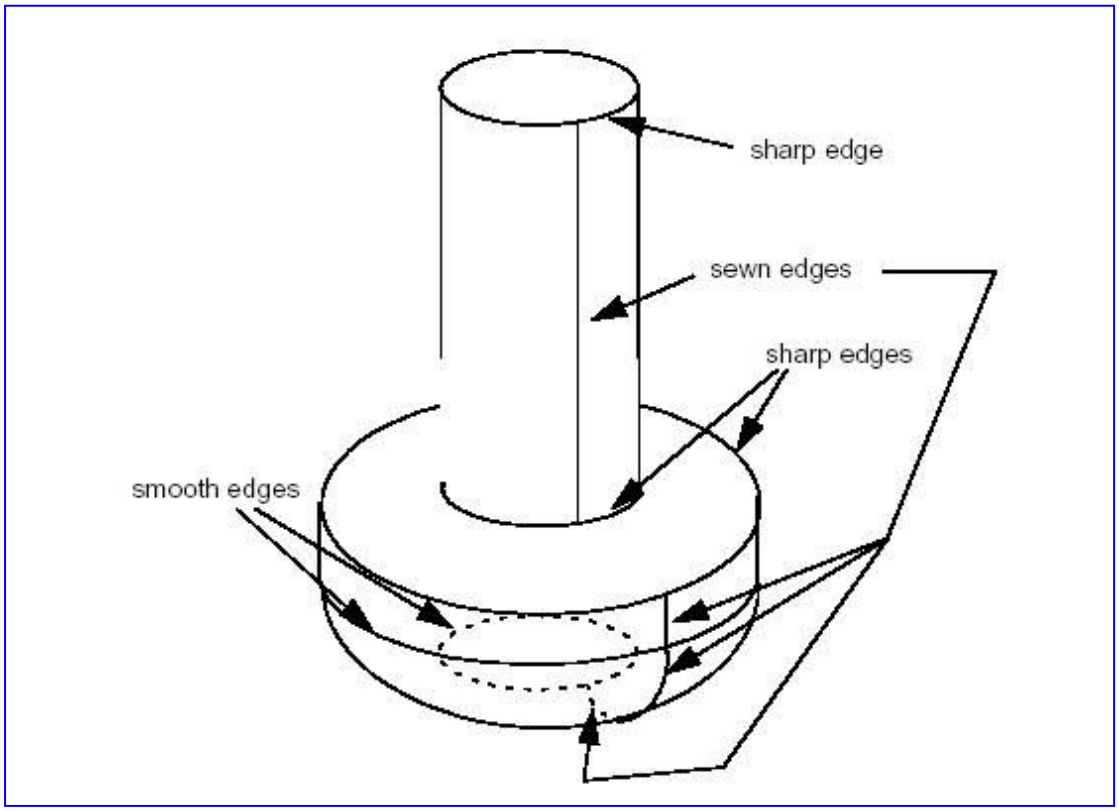

Figure 1.2 Sharp, smooth and sewn edges in a simple screw shape

图 1.3 中的实线为同相形状的外轮廓线，虚线部分为等分参数线。

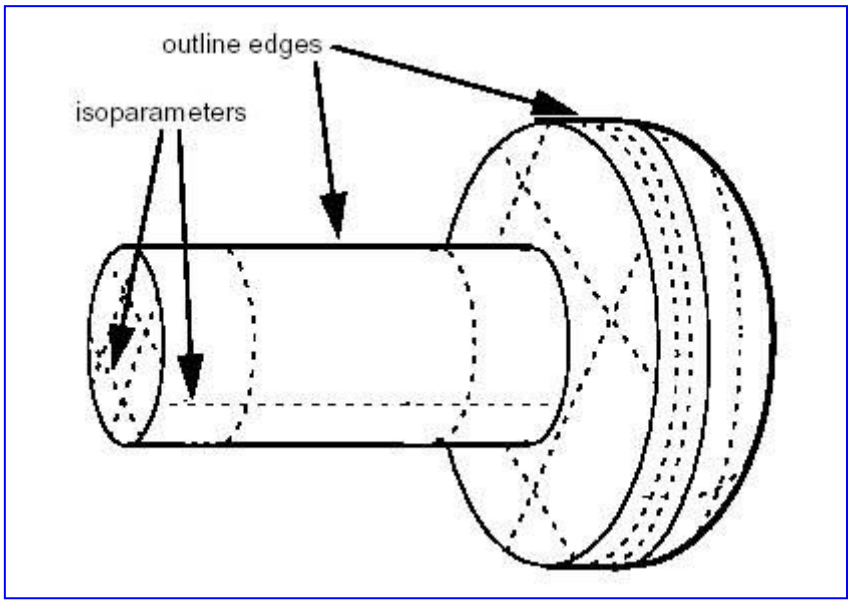

Figure 1.3 Outline edges and isoparameters in the same shape

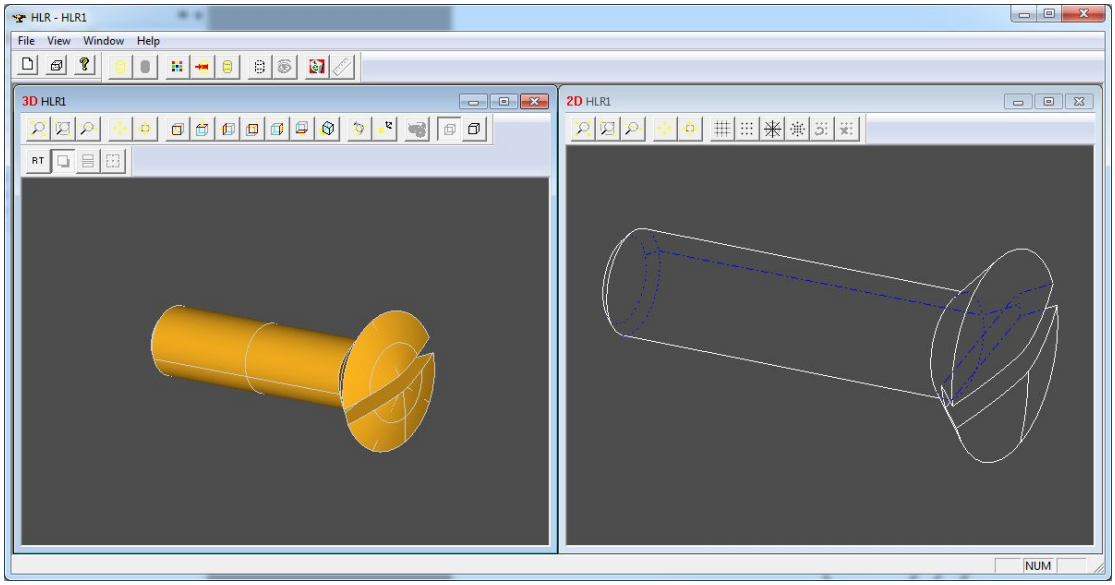

Figure 1.4 An extraction showing hidden sharp edges

如图 1.4 可以看出，蓝色虚线即为被遮挡的应该被去除的线。

2. HLR Usage

OpenCASCADE 隐藏线去除算法的使用涉及以下几个步骤：

2.1 Loading Shapes

通过使用 HLRBRep_Algo::Add()函数来将需要被消隐的形状加入到消隐算法中去。对于 HLRBRep_PolyAlgo 对象，使用 HLRBRep_PolyAlgo::Load()函数来添加一个或多个需要处理的形状。

2.2 Setting View Parameters

通过函数 HLRBRep_PolyAlgo::Projector() 来设置投影方向，其参数为一个 HLRAlgo_Projector 对象。一般会根据三维视图数据来得到这个投影数据，进而来设置需要消隐的投影参数。

2.3 Computing the Projections

通过类 HLRBRep_PolyAlgo 中的函数 HLRBRep_PolyAlgo::Update()来计算模型的外轮廓。当用类 HLRBRep_Algo 时，使用 HLRBRep_Algo::Update()这个算法时，必须调用方法 HLRBRep_Algo::Hide()来计算模型可见与隐藏线。使用类 HLRBRep_PolyAlgo 时，可见与隐藏线是通过 HLRBRep_PolyHLRToShape 来计算。

2.4 Extracting Edges

通过类 HLRBRep_HLRToShape 和 HLRBRep_PolyHLRToShape 来提取消隐后的模型数据，提取数据来源分别对应 HLRBRep_Algo 和 HLRBRep_PolyAlgo 对象。可提取的类型有：

- Visible/hidden sharp edges;
- Visible/hidden smooth edges;
- Visible/hidden sewn edges;
- Visible/hidden outline edges;

提取操作是由函数 HLRBRep_PolyHLRToShape::Update 来实现。

3. Examples

为了产生与 AVEVA PDMS 的 Draft 功能模块类似的功能，就需要隐藏线消除算法来自动根据模型生成二维图纸。如下代码为测试 HLR 算法的一个简单示例：

```
osg::Node* TestPolyHlr(void)
{
    osg::ref_ptr<osg::Geode> aGeode = new osg::Geode();
    osg::ref_ptr<osg::Geometry> aLineGeometry = new osg::Geometry();
    osg::ref_ptr<osg::Vec3Array> aVertices = new osg::Vec3Array();

    TopoDS_Shape aPipeModel;
    BRepTools::Read(aPipeModel, "d:/PipeModels/2007.brep", BRep_Builder());

    BRepMesh_IncrementalMesh aMesher(aPipeModel, 0.1);

    OSD_Timer aTimer;
    aTimer.Start();

    Handle_HLRBRep_PolyAlgo aHlrPolyAlgo = new HLRBRep_PolyAlgo();

    HLRAlgo_Projector aProjector;
    HLRBRep_PolyHLRToShape aHlr2Shape;

    aHlrPolyAlgo->Load(aPipeModel);

    aHlrPolyAlgo->Projector(aProjector);
    aHlrPolyAlgo->Update();

    aHlr2Shape.Update(aHlrPolyAlgo);

    aTimer.Stop();
    aTimer.Show(std::cout);

    for (TopExp_Explorer e(aHlr2Shape.VCompound(), TopAbs_EDGE); e.More();
e.Next())
    {
        TopoDS_Edge anEdge = TopoDS::Edge(e.Current());
        TopoDS_Vertex aFirstVertex = TopExp::FirstVertex(anEdge);
        TopoDS_Vertex aLastVertex = TopExp::LastVertex(anEdge);
        gp_Pnt aFirstPoint = BRep_Tool::Pnt(aFirstVertex);
        gp_Pnt aLastPoint = BRep_Tool::Pnt(aLastVertex);

        aVertices->push_back(osg::Vec3(aFirstPoint.X(),      aFirstPoint.Y(),
aFirstPoint.Z()));
```

```
        aVertices->push_back(osg::Vec3(aLastPoint.X(),        aLastPoint.Y(),
aLastPoint.Z()));
    }

    for   (TopExp_Explorer   e(aHlr2Shape.OutLineVCompound(),   TopAbs_EDGE);
e.More(); e.Next())
    {
        TopoDS_Edge anEdge = TopoDS::Edge(e.Current());
        TopoDS_Vertex aFirstVertex = TopExp::FirstVertex(anEdge);
        TopoDS_Vertex aLastVertex = TopExp::LastVertex(anEdge);
        gp_Pnt aFirstPoint = BRep_Tool::Pnt(aFirstVertex);
        gp_Pnt aLastPoint = BRep_Tool::Pnt(aLastVertex);

        aVertices->push_back(osg::Vec3(aFirstPoint.X(),      aFirstPoint.Y(),
aFirstPoint.Z()));
        aVertices->push_back(osg::Vec3(aLastPoint.X(),        aLastPoint.Y(),
aLastPoint.Z()));
    }

    aLineGeometry->setVertexArray(aVertices);
    aLineGeometry->addPrimitiveSet(new
osg::DrawArrays(osg::PrimitiveSet::LINES, 0, aVertices->size()));

    aGeode->addDrawable(aLineGeometry);

    return aGeode.release();
}
```

根据上述用法介绍一步一步来，就可以生成只包含线段数据的消隐后的结果，然后再在 OpenSceneGraph 中显示结果如下图所示：

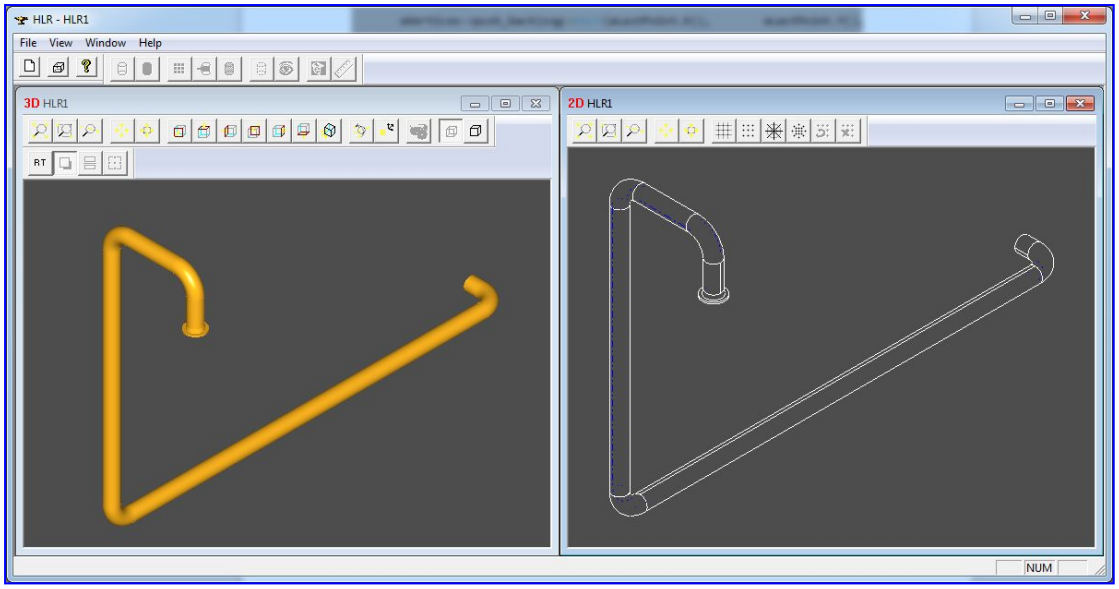

Figure 3.1 HLR for pipe model

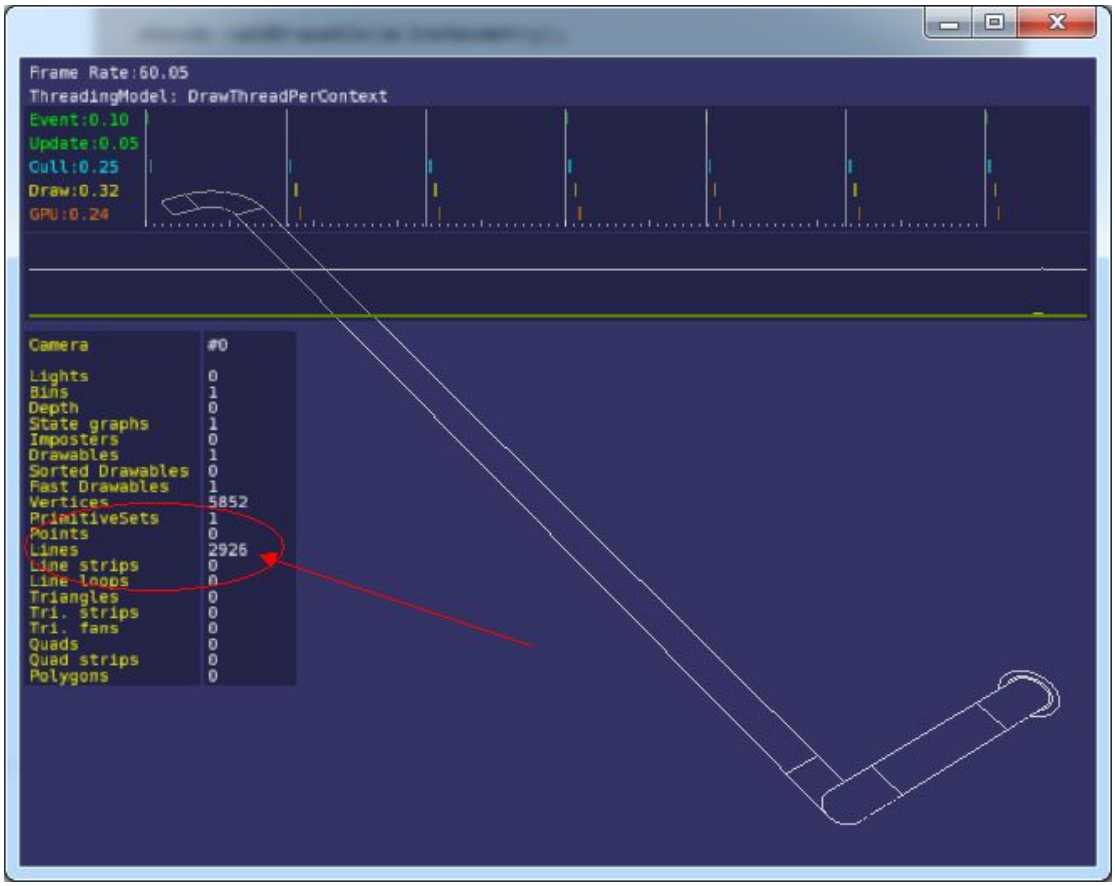

Figure 3.2 HLR pipe model in OpenSceneGraph

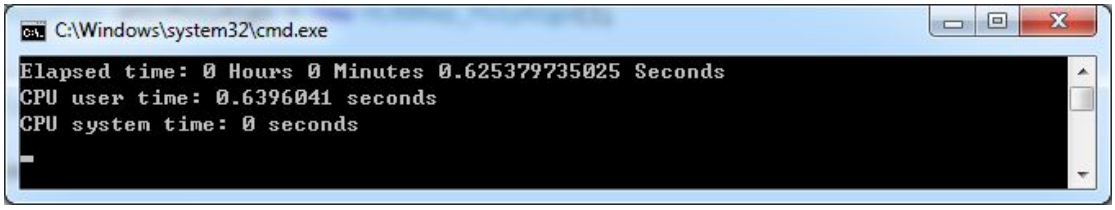

Figure 3.3 HLR time usage

由图 3.2 可知，一个简单的管道模型经过 HLRBRep_PolyAlgo 消隐后，产生很多线段数据，但是由图 3.3 可知，HLR 消隐速度还是比较快的。因为 HLRBRep_PolyAlgo 是基于离散网格及可视化数据的，所以当离散精度降低时，会产生较少数据。如下图为降低离散精度后，产生的线段数据明显减少。

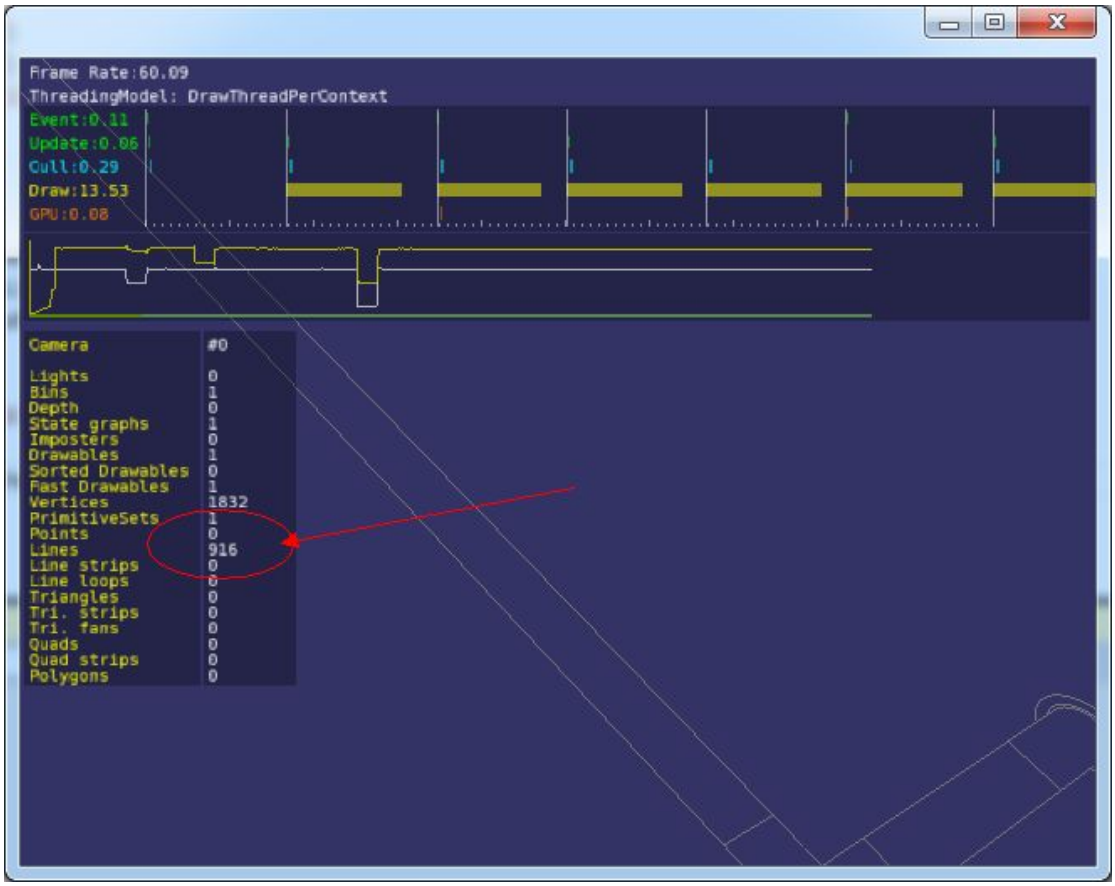

Figure 3.4 HLR pipe model in less tesslate precision

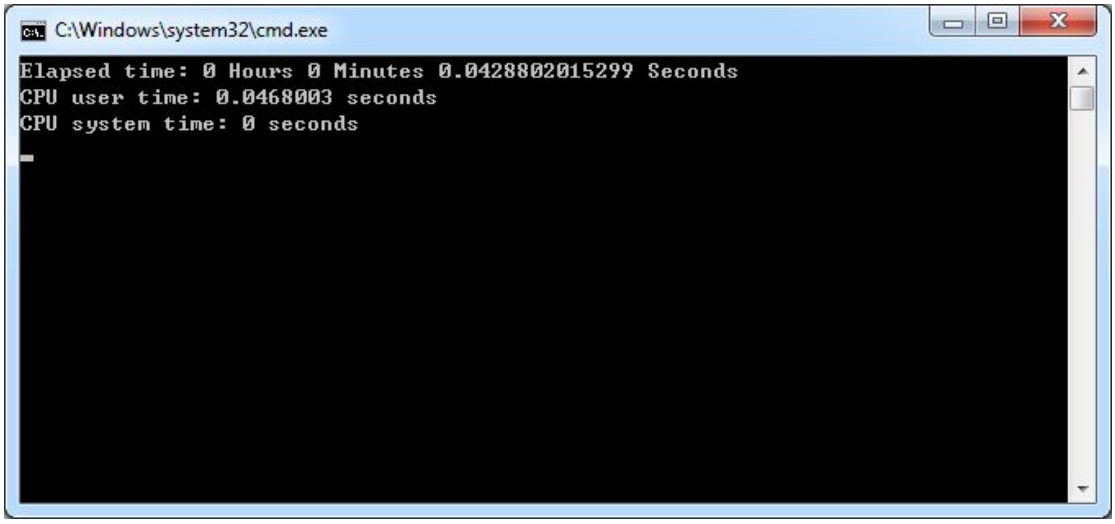

Figure 3.5 HLR pipe model in less tesslate precision time usage

由图 3.4 和图 3.5 可知，当降低模型的离散精度时，在不影响消隐后二维图形质量的情况下，消隐后产生的线段数据明显减少，且消隐算法的速度也明显要快很多。所以离散精度也是 HLR 消隐算法的一个关键因素，使用消隐 HLR 算法时需要选择合适的离散精度。

4. Conclusion

综上可知，OpenCASCADE 的隐藏线消除 HLR 算法使用起来还是比较简单的，不过彻底理解算法，还是需要静下心来，Debug 进代码，在理解大概原理的基础上，对其实现作进一步的理解。

要使用 OpenCASCADE 的 HLR 算法，只要指定好投影参数及加载好待消隐的模型，即可得到消隐后的模型的二维数据了。若想加快算法速度事减少模型的二维轮廓数据，则需要选择合适的网格离散精度。

5. References

1. OpenCASCADE Modeling Algorithms User Guide6.8.0 2014
2. 何援军. 计算机图形学. 机械工业出版社. 2010
3. 孙家广. 计算机图形学. 清华大学出版社. 2000