Bounding Volume Hierarchy BVH in OpenCASCADE

eryar@163.com

Abstract. Bounding Volume Hierarchy(BVH) organizes geometric objects in the tree based on spatial relationships. Each node in the tree contains an axis-aligned bounding box of all the objects below it. Bounding volume hierarchies are used in many algorithms to support efficient operations on the sets of geometric objects, such as collision detection, ray-tracing, searching of nearest objects, and view frustum culling. The paper focus on the usage of BVH on TopoDS_Shape, and its application.

Key Words. BVH, Bounding Volume Hierarchy, Collision Detection

1.Introduction

层次包围盒(Bounding Volume Hierarchy, BVH)是一种基于物体的场景管理技术，广泛用于碰撞检测，光线追踪，视锥体物体剔除等场合。对于三维场景的实时渲染来说，BVH 是最常用的数据结构。场景以层次树形结构进行组织，包含一个根节点、内部节点、叶子节点。树中的每个节点，包括叶子节点都有一个包围体，可以将其子树的所有几何体包围起来，这就是 BVH 名字的由来。如下图所示：

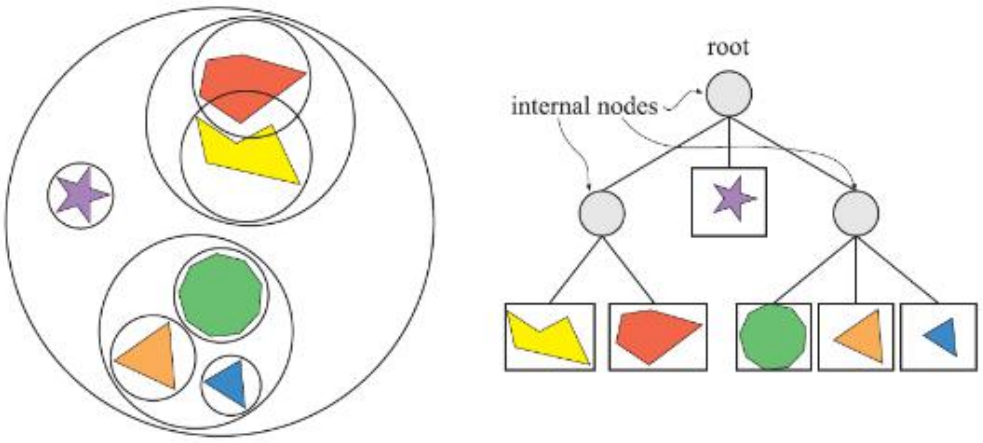


Figure 1. Simple Scene(left), BVH Scene Graph(right)

本文主要介绍 OpenCASCADE 中的 BVH 包及其应用，如碰撞检测。

2.Build BVH

在 BVH 包中的头文件里面找到 BVH 的作者：Denis BOGOLEPOV, Danila ULYANOV，在网上搜到他们的一篇论文：Real-Time SAH BVH Construction for Ray Tracing Dynamic Scenes. 因此，可知 BVH 中是应用 SAH（Surface Area Heuristic）策略来构造 BVH 树的。根据 BVH_BinnedBuilder 头文件中的注释可知：

```
//! Performs construction of BVH tree using binned SAH algorithm. Number
//! of bins controls BVH quality in cost of construction time (greater
//! better). For optimal results, use 32 - 48 bins. However, reasonable
//! performance is provided even for 4 - 8 bins (it is only 10-20% lower
//! in comparison with optimal settings). Note that multiple threads can
//! be used only with thread safe BVH primitive sets.
```

OpenCASCADE 中也提供了其他的构造方法，如类 BVH_LinearBuilder 中使用了 Spatial Morton codes 方法：

```
//! Performs fast BVH construction using LBVH building approach.
//! Algorithm uses spatial Morton codes to reduce the BVH construction
//! problem to a sorting problem (radix sort -- O(N) complexity). This
//! Linear Bounding Volume Hierarchy (LBVH) builder produces BVH trees
```

```
//! of lower quality compared to SAH-based BVH builders but it is over
//! an order of magnitude faster (up to 3M triangles per second).
//!
//! For more details see:
//! C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha.
//! Fast BVH construction on GPUs. Eurographics, 2009.
```

3.Traversal BVH

由类 BVH_TreeBase 可知，当得到 BVH 树后，可以取出节点索引的数组。下面给出将 TopoDS_Shape 的 BVH 树显示出来的示例代码：

```
/*
Copyright(C) 2017 Shing Liu(eryar@163.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files(the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and / or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions :

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
*/

// Visual Studio 2013 & OpenCASCADE7.1.0
// 2017-04-18  20:52

#include <BRepExtrema_TriangleSet.hxx>

#include <BRepTools.hxx>
#include <BRep_Builder.hxx>

#include <TopoDS.hxx>

#include <TopExp_Explorer.hxx>

#include <BRepPrimAPI_MakeBox.hxx>

#include <BRepMesh_IncrementalMesh.hxx>

#pragma comment(lib, "TKernel.lib")
#pragma comment(lib, "TKMath.lib")

#pragma comment(lib, "TKG2d.lib")
```

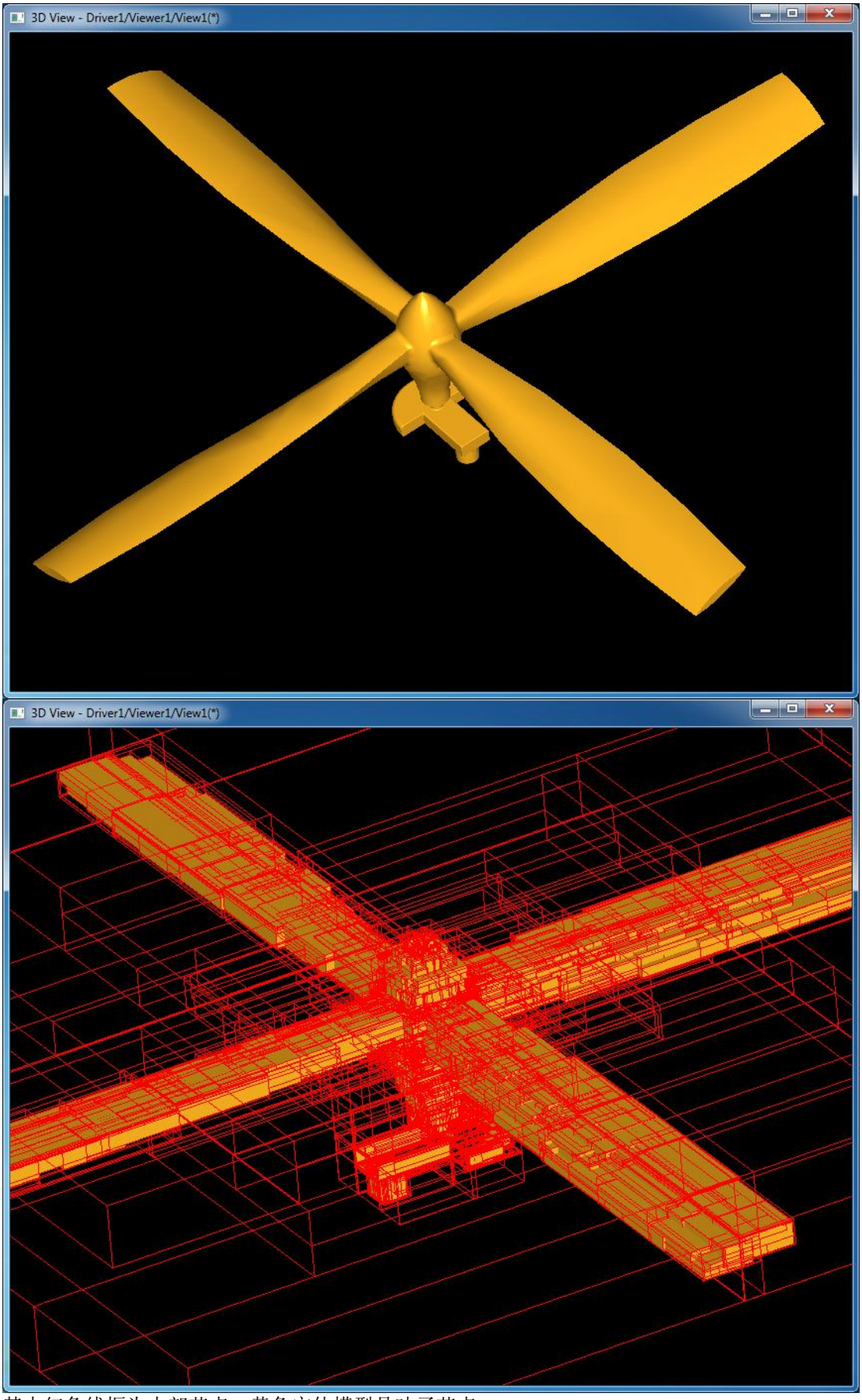
```
#pragma comment(lib, "TKG3d.lib")
#pragma comment(lib, "TKGeomBase.lib")
#pragma comment(lib, "TKGeomAlgo.lib")

#pragma comment(lib, "TKBRep.lib")
#pragma comment(lib, "TKPrim.lib")
#pragma comment(lib, "TKMesh.lib")
#pragma comment(lib, "TKTopAlgo.lib")


void testBvh(const std::string& theFileName)
{
    BRep_Builder aBuilder;
    TopoDS_Compound aCompound;

    TopoDS_Shape aShape;
    BRepTools::Read(aShape, theFileName.c_str(), aBuilder);

    BRepExtrema_ShapeList aFaceList;
    for (TopExp_Explorer i(aShape, TopAbs_FACE); i.More(); i.Next())
    {
        aFaceList.Append(TopoDS::Face(i.Current()));
    }

    aBuilder.MakeCompound(aCompound);

    BRepMesh_IncrementalMesh aMesher(aShape, 1.0);

    BRepExtrema_TriangleSet aTriangleSet(aFaceList);
    const NCollection_Handle<BVH_Tree<Standard_Real, 3> >& aBvh = aTriangleSet.BVH();
    for (int i = 0; i < aBvh->Length(); i++)
    {
        const BVH_Vec4i& aNodeData = aBvh->NodeInfoBuffer()[i];

        if (aNodeData.x() == 0)
        {
            // inner node.
            const BVH_Vec3d& aP1 = aBvh->MinPoint(aNodeData.y());
            const BVH_Vec3d& aP2 = aBvh->MaxPoint(aNodeData.y());

            const BVH_Vec3d& aQ1 = aBvh->MinPoint(aNodeData.z());
            const BVH_Vec3d& aQ2 = aBvh->MaxPoint(aNodeData.z());

            try
            {
                BRepPrimAPI_MakeBox aBoxMaker(gp_Pnt(aP1.x(), aP1.y(), aP1.z()),
                    gp_Pnt(aP2.x(), aP2.y(), aP2.z()));
                for (TopExp_Explorer i(aBoxMaker.Shape(), TopAbs_EDGE); i.More();
i.Next())
                {
                    aBuilder.Add(aCompound, i.Current());
                }
            }
```

```
            catch (Standard_Failure f)
            {
            }

            try
            {
                BRepPrimAPI_MakeBox aBoxMaker(gp_Pnt(aQ1.x(), aQ1.y(), aQ1.z()),
                    gp_Pnt(aQ2.x(), aQ2.y(), aQ2.z()));
                for (TopExp_Explorer i(aBoxMaker.Shape(), TopAbs_EDGE); i.More();
i.Next())
                {
                    aBuilder.Add(aCompound, i.Current());
                }
            }
            catch (Standard_Failure f)
            {
            }
        }
        else
        {
            // leaves node.
            const BVH_Vec3d& aP1 = aBvh->MinPoint(i);
            const BVH_Vec3d& aP2 = aBvh->MaxPoint(i);

            try
            {
                BRepPrimAPI_MakeBox aBoxMaker(gp_Pnt(aP1.x(), aP1.y(), aP1.z()),
                    gp_Pnt(aP2.x(), aP2.y(), aP2.z()));

                aBuilder.Add(aCompound, aBoxMaker.Shape());
            }
            catch (Standard_Failure f)
            {
            }
        }
    }

    BRepTools::Write(aCompound, "d:/bvh.brep");
}

int main(int argc, char* argv[])
{
    if (argc > 1)
    {
        testBvh(argv[1]);
    }

    return 0;
}
```

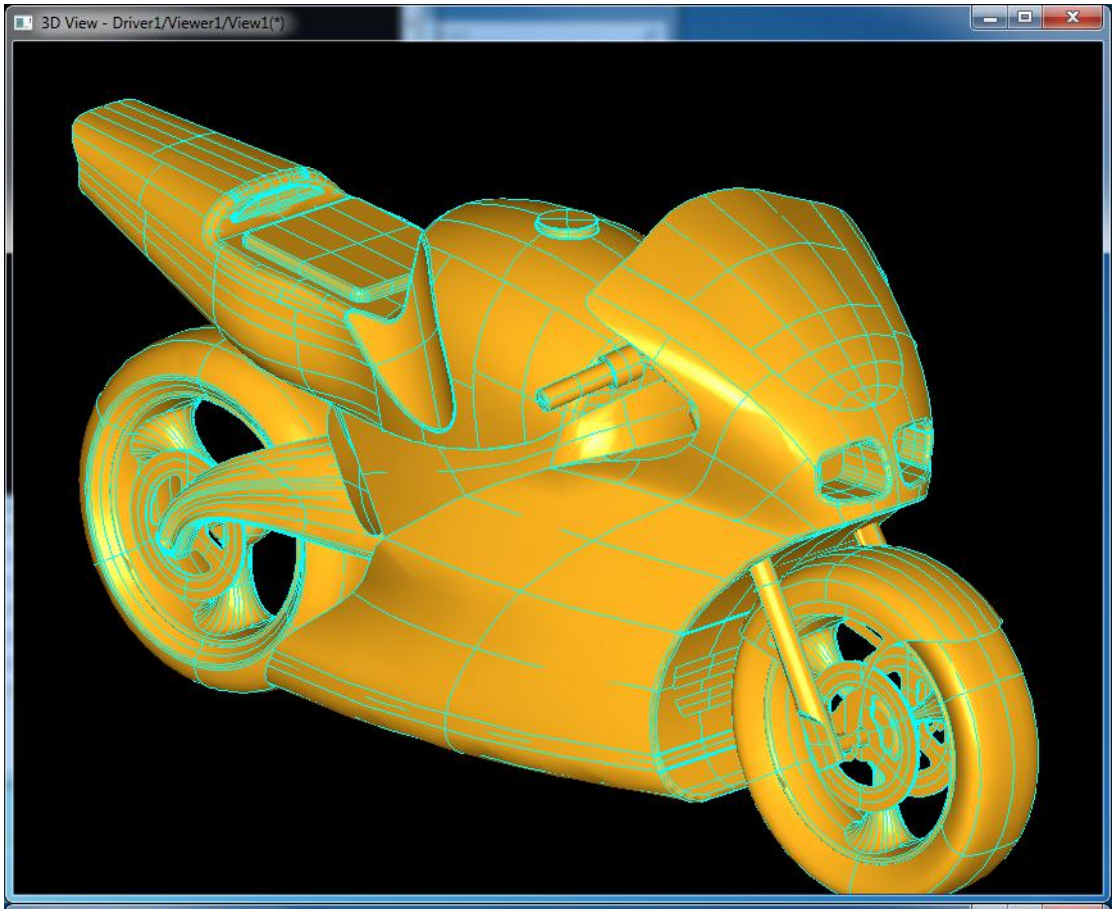
由上述代码可知，当得到节点信息

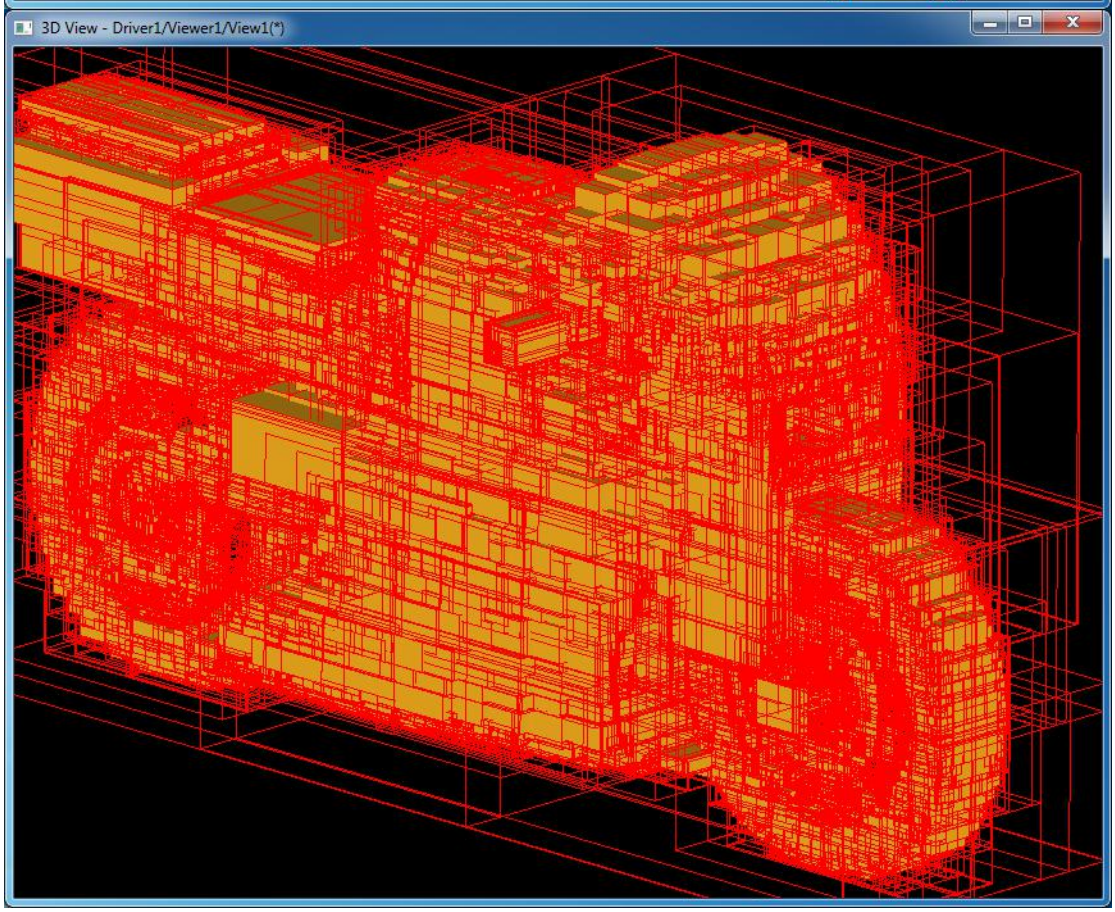
```
const BVH_Vec4i& aNodeData = aBvh->NodeInfoBuffer()[i];
```

后，通过判断 aNodeData.x()分量来确定此节点是内部节点还是叶子节点。显示 OCC 自带的螺旋桨模型的 BVH 如下图所示：



其中红色线框为内部节点，黄色实体模型是叶子节点。


3D View - Driver1/Viewer1/View1(*)


3D View - Driver1/Viewer1/View1(*)

4.BVH Application

BVH 在 OpenCASCADE 中也有广泛地应用，如开源版本中的模型快速碰撞检测，使用类 BRepExtrema_ShapeProximity. 模型选择操作，光线跟踪等算法中都有应用。

5.Conclusion

因为 OpenCASCADE 中构造 BVH 时依赖模型的网格三角形，所以 BVH 算法的应用的结果就依赖于网格化算法的质量了。如两个 Topo 形状碰撞检测时，网格化的质量越高，结果精度越高。如果用解析算法去对两个 Topo 形状做碰撞检测，也会涉及到解析算法的精度问题。

BVH 结构广泛应用于碰撞检测、光线跟踪等算法中，大大提高程序性能。本文只是抛砖引玉，对 BVH 感兴趣的童鞋可以参考相关资料及 OpenCASCADE 中的代码实现，去深入学习应用。

6.References

1. Dmitry Sopin, Denis Bogolepov, Danila Ulyanov. Real-Time SAH BVH Construction for Ray Tracing Dynamic Scenes. http://graphicon.ru/html/2011/conference/gc2011Sopin.pdf
2. BVH with SAH. http://www.cnblogs.com/lookof/p/3546320.html
3. 3D 游戏引擎中常见的三维场景管理方法.
   http://www.cnblogs.com/wangchengfeng/p/3495954.html