实体模型高效碰撞检测算法大作业汇报展示

汇 报 人：刘毅 郭雪卿
指导教师：胡振中
汇报时间：2021-12-24

CONTENTS

01 选题背景

02 技术路线

03 成果展示
- 空间三角形相交判断模块
- 相交快速排除模块
- 多个空间体相交检测模块
- 成果可视化展示模块
- 多种算法对比验证模块

04 总结展望


清華大學 深圳国际研究生院
Tsinghua Shenzhen International Graduate School

Part 1

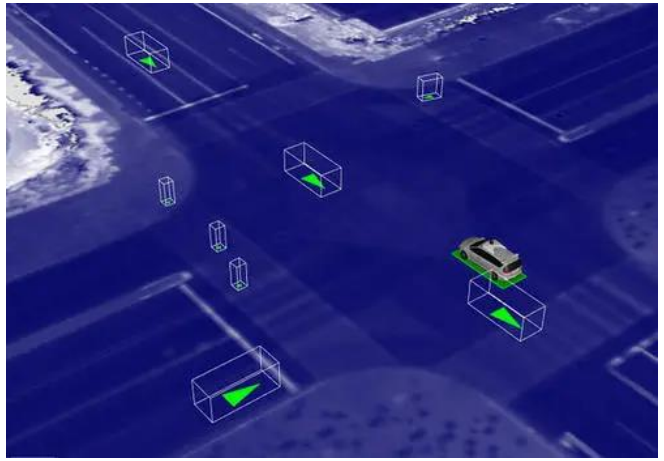
选题背景

选题背景

- **题目**

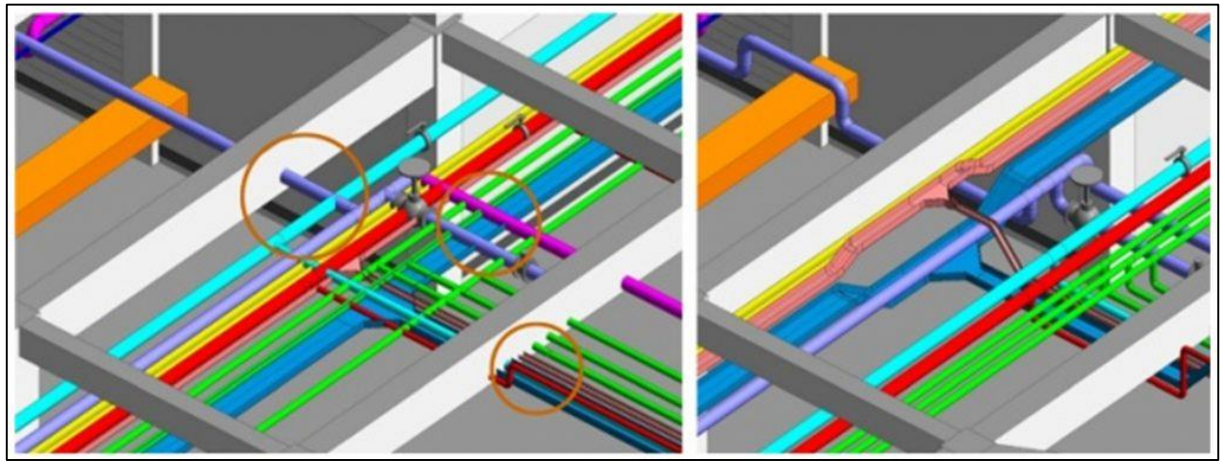
  实体模型高效碰撞检测算法

- **背景**
  - ◆ 碰撞检测：检查物体在空间上是否有重合的部分
  - ◆ 在众多领域有着十分广泛的应用，例如三维仿真、虚拟现实、自动驾驶、智能机器人、3D游戏等等
  - ◆ 碰撞检测是**BIM应用**中最实用的功能之一，能够在设计阶段发现大量隐藏的问题，达到减少返工、缩短工期的目的，特别对于多方合作具有重要的意义



自动驾驶碰撞检测

Unity3D碰撞检测



Navisworks中的碰撞检测和优化

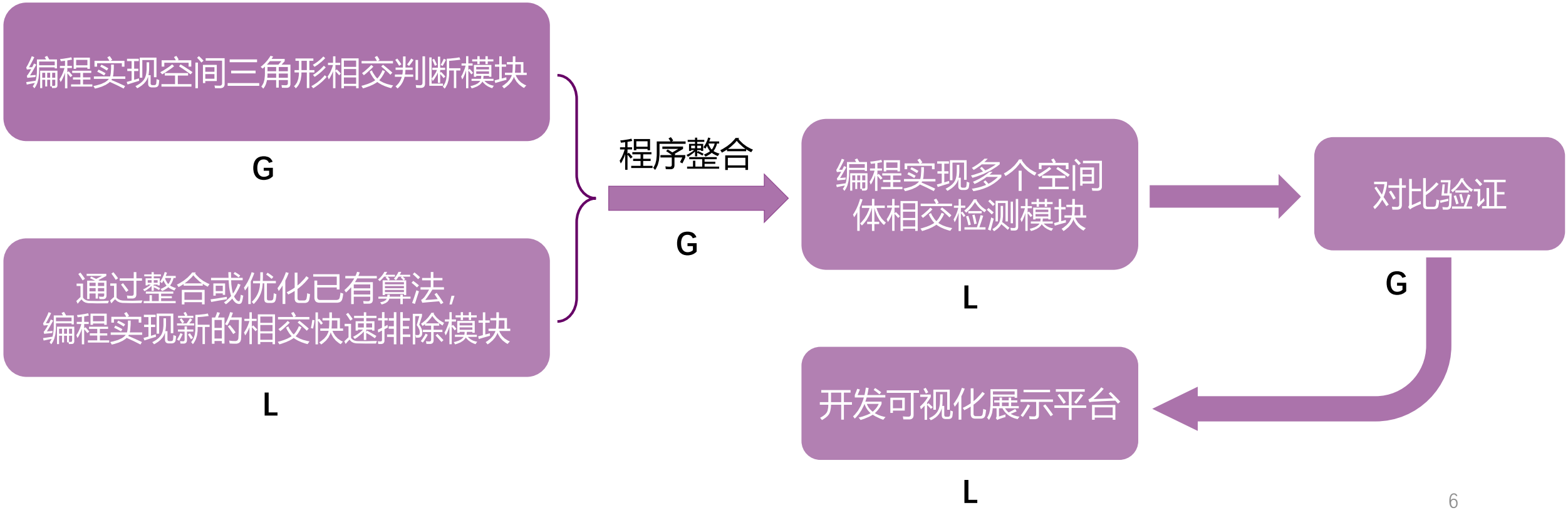

清華大學 深圳国际研究生院
Tsinghua Shenzhen International Graduate School

Part 2

技术路线

研究内容

- **研究目标**
  - ◆ 针对基于三角形网格描述的实体模型，提出一个**高效**的碰撞检测算法
  - ◆ 实现多个空间体（每个空间体含有多个三角面片）之间的相交检测
- **技术路线**




清華大學 深圳国际研究生院
Tsinghua Shenzhen International Graduate School

Part 3

成果展示

空间三角形相交判断模块

成果展示

空间三角形相交判断模块

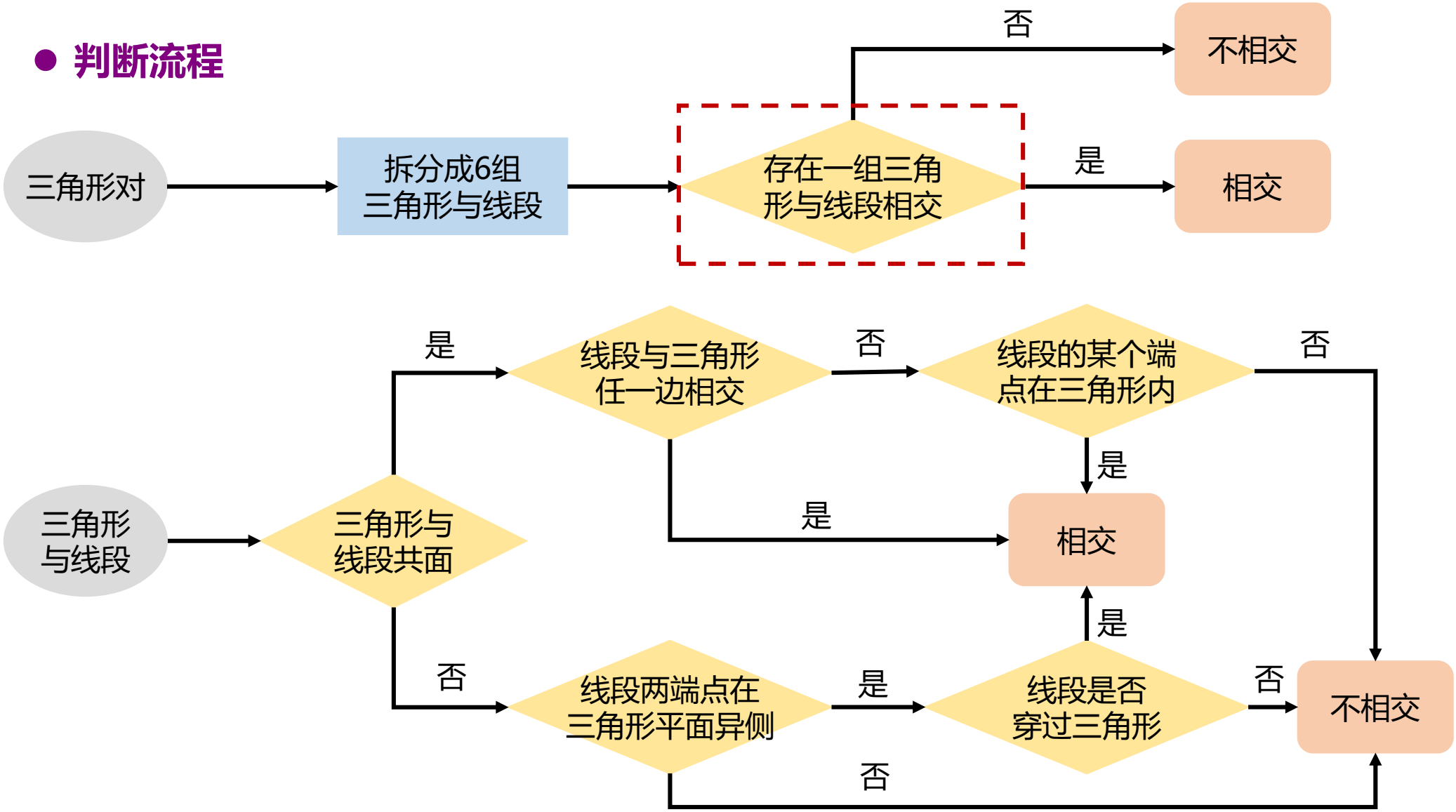
相交快速排除模块

多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

● 判断流程



空间三角形相交判断模块

成果展示

空间三角形相交判断模块

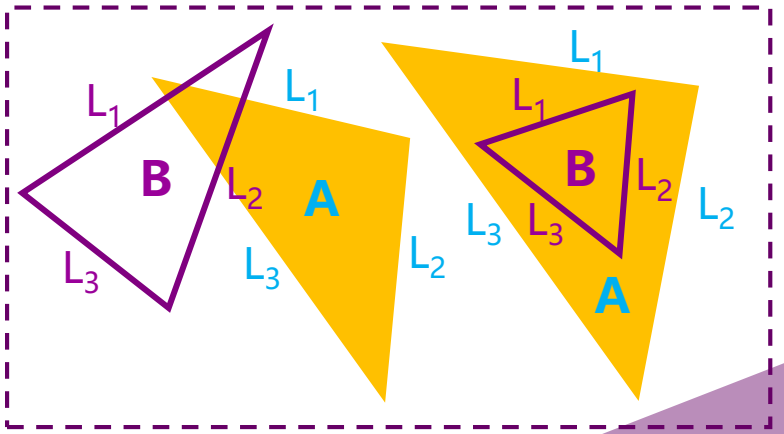
相交快速排除模块

多个空间体相交检测模块

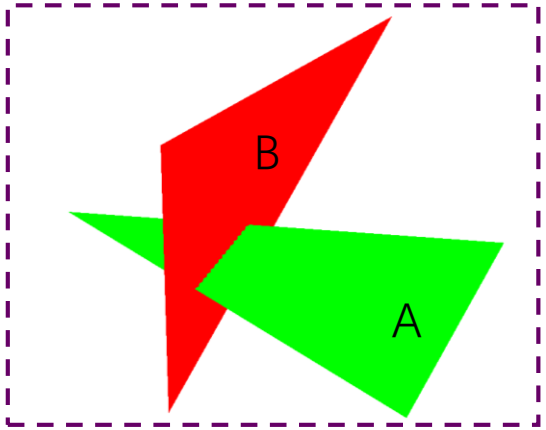
成果可视化展示模块

多种算法对比验证模块

- 两三角面片相交问题 ➡ 三角形与线段相交问题



共面



异面

是

三角形A与B相交

ΔA与线段B_L_1相交 || ΔA与线段B_L_2相交 || ΔA与线段B_L_3相交 || ΔB与线段A_L_1相交 || ΔB与线段A_L_2相交 || ΔB与线段A_L_3相交

否

三角形A与B不相交

空间三角形相交判断模块

成果展示

空间三角形相交判断模块

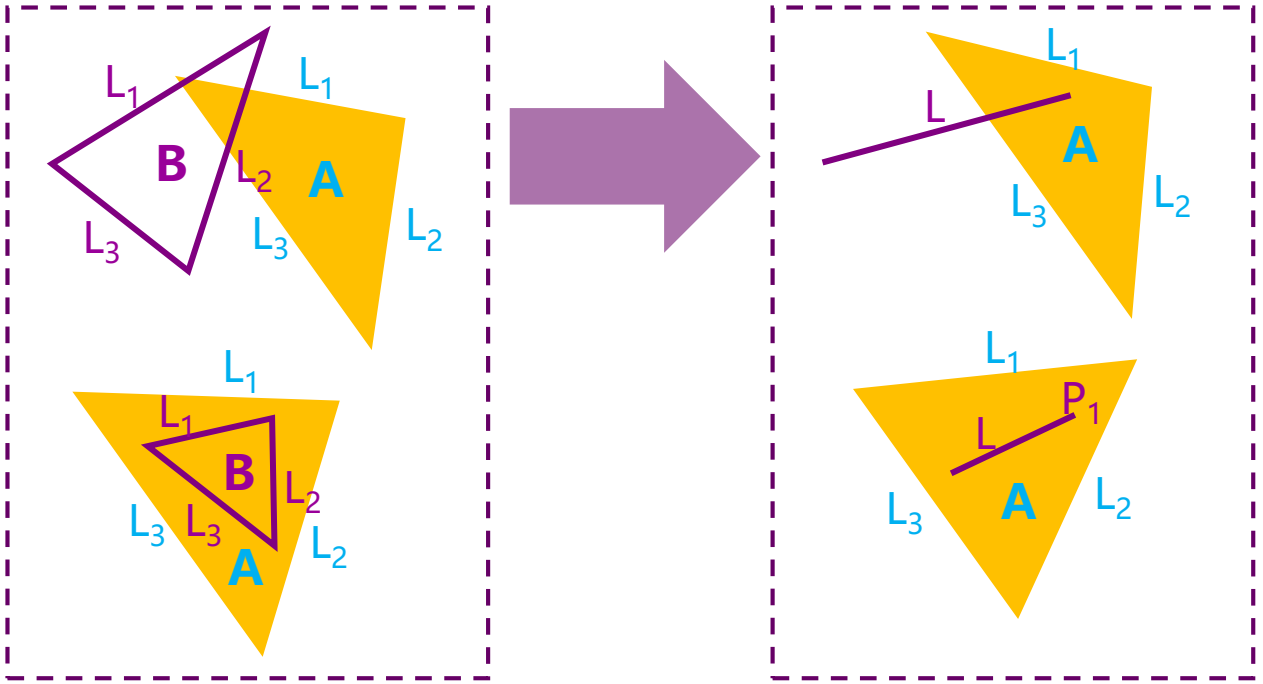
相交快速排除模块

多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

- 三角形与线段相交问题 → 共面情况/异面情况



线段L与线段A_L_1相交
|| 线段L与线段A_L_2相交
|| 线段L与线段A_L_3相交
|| 点L_P_1在三角形A内部

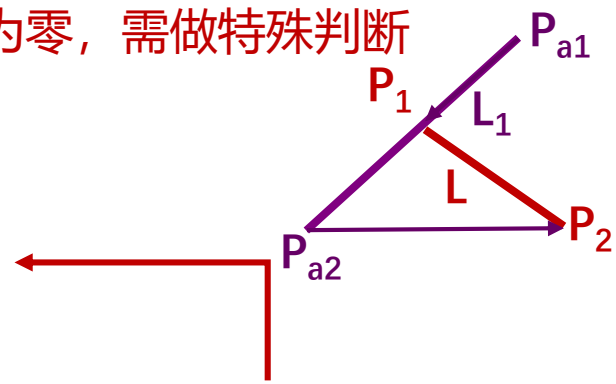
```
/// <summary> 是否与共平面的线段相交
5 个引用
public bool DoCollideInPlane(Line line)
{
    return line.DoCollideInPlane(new Line(P1, P2)) || line.DoCollideInPlane(new Line(P2, P3))
        || line.DoCollideInPlane(new Line(P3, P1)) || HasPoint(line.P1);
}
```

```
/// <summary> 三角形是否包含点p
3 个引用
public bool HasPoint(Point p)
{
    if (_normalIsZero)
    {
        return _abnormalLine.HasPoint(p);
    }
    var v1 = p - P1;
    if (_normal.DotProduct(v1) != 0)
    {
        // p不与三角形共面
        return false;
    }
    var v2 = p - P2;
    var v3 = p - P3;
    var c12 = v1.CrossProduct(v2);
    var c23 = v2.CrossProduct(v3);
    if (c12.DotProduct(c23) < 0)
    {
        return false;
    }
    var c31 = v3.CrossProduct(v1);
    if (c23.DotProduct(c31) < 0)
    {
        return false;
    }
    if (c31.DotProduct(c12) < 0)
    {
        return false;
    }
    return true;
```

空间三角形相交判断模块

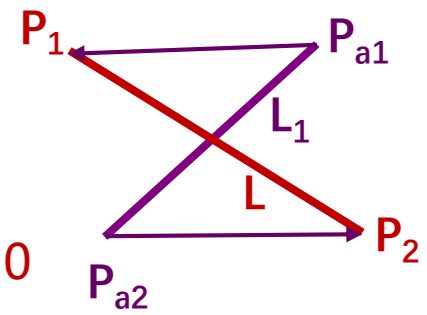
● 三角形与线段相交问题 ➡ **共面情况/异面情况**

➢ 当线段端点在另一线段上时，点积为零，需做特殊判断



□ 方法：跨立实验

➢ 两线段相交 ⇔ 任一线段的两端点在另一线段所在直线的两侧



➢ $(P_{a1}\rightarrow P_1)\times(P_1\rightarrow P_2)\cdot(P_{a2}\rightarrow P_2)\times(P_1\rightarrow P_2)<0$

&& $(P_{a1}\rightarrow P_1)\times(P_{a1}\rightarrow P_{a2})\cdot(P_{a2}\rightarrow P_2)\times(P_{a1}\rightarrow P_{a2})<0$

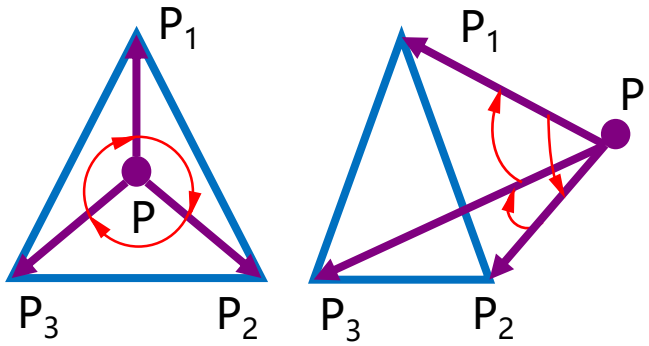
线段L与线段A_L_1相交

|| 线段L与线段A_L_2相交

|| 线段L与线段A_L_3相交

➢ 点 P 在三角形A内 ⇔ $P\rightarrow A\_P_1$、$P\rightarrow A\_P_2$、$P\rightarrow A_P_3$旋转方向相同

➢ $(\overrightarrow{PP_1}\times\overrightarrow{PP_2})\&\&(\overrightarrow{PP_2}\times\overrightarrow{PP_3})\&\&(\overrightarrow{PP_3}\times\overrightarrow{PP_1})$ 方向相同



|| 点L_P_1在三角形A内部

空间三角形相交判断模块

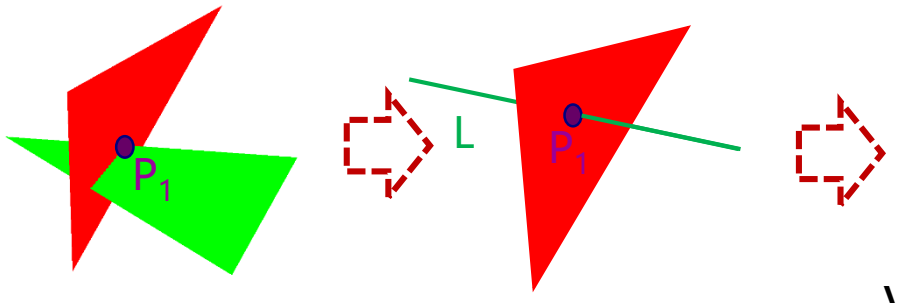
- 三角形与线段相交问题 ➡ 共面情况/异面情况

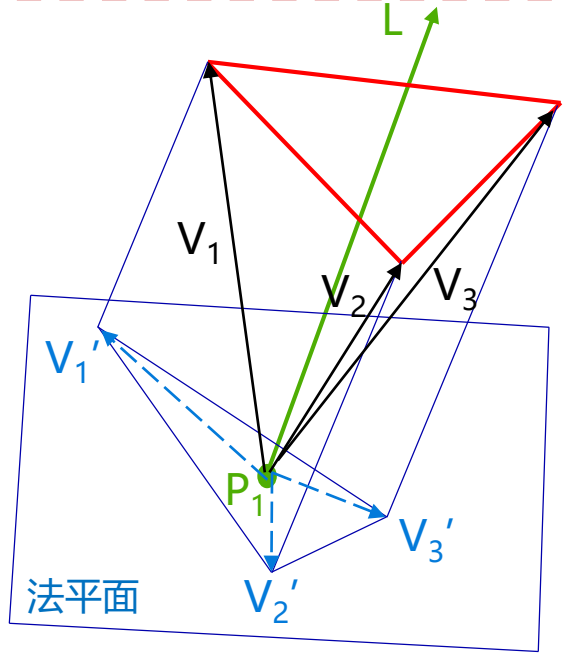
➢ 判断线段是否穿过三角形平面
- 不穿过（线段两端点在三角形平面同侧）➡
- 穿过➡判断交点是否在三角形内/上

```
/// <summary> 判断是否与空间线段相交
8 个引用
public bool DoCollide(Line line)
{
    if (line.P1.Equals(line.P2))
    {
        // line的两点重合时
        return HasPoint(line.P1);
    }

    if (_normalIsZero)
    {
        // 三角形不正常时（三顶点共线）
        return _abnormalLine.DoCollide(line);
    }

    var s1 = GetSide(line.P1);
    var s2 = GetSide(line.P2);
    if (s1 * s2 > 0)
    {
        // 两点位于三角形同侧
        return false;
    }
    if (s1 == 0 && s2 == 0)
    {
        // 线段与三角形共面
        return DoCollideInPlane(line);
    }

    // 沿line方向进行投影，判断点是否在投影三角形内
    var lineDir = line.P2 - line.P1;
    var v1 = P1 - line.P1;
    var v2 = P2 - line.P1;
    var v3 = P3 - line.P1;
    var dc12 = v1.CrossProduct(v2).DotProduct(lineDir);
    var dc23 = v2.CrossProduct(v3).DotProduct(lineDir);
    if (dc12 * dc23 < 0)...
    var dc31 = v3.CrossProduct(v1).DotProduct(lineDir);
    if (dc23 * dc31 < 0)...
    if (dc31 * dc12 < 0)...
    return true;
}
```

- 方法：将三角形沿线段L任一垂平面方向进行投影，判断线段L与垂平面交点 P_1 是否在三角形投影 $\Delta V_1' V_2' V_3'$ 内

⇨ 平面内点在三角形内问题





$$V_1' \times V_2' \cdot L = V_1 \times V_2 \cdot L$$

空间三角形相交判断模块

成果展示

空间三角形相交判断模块

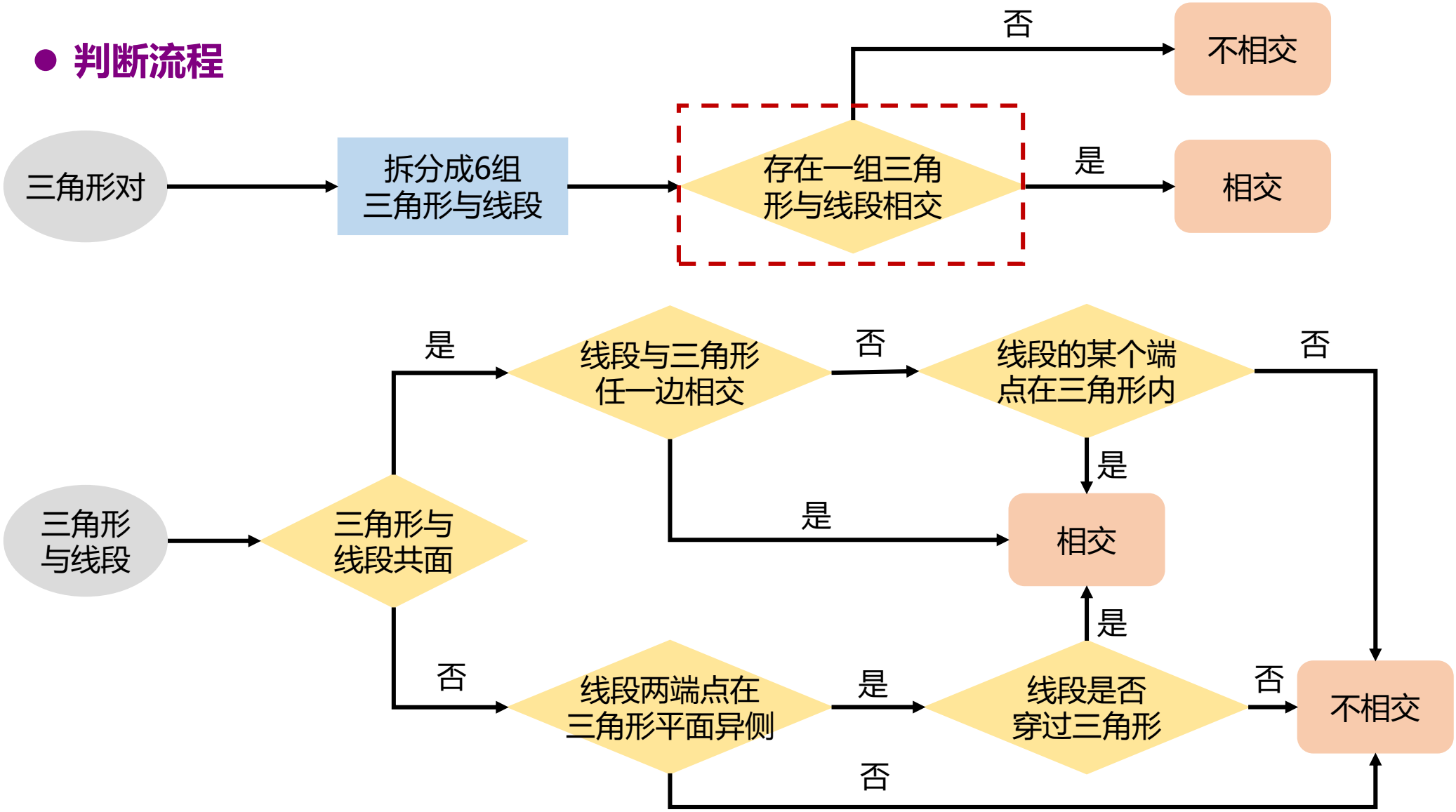
相交快速排除模块

多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

判断流程



相交快速排除模块

成果展示

空间三角形相交判断模块

相交快速排除模块

多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

● 加速算法简介

输入 → 逐对检测 → 输出

$\frac{n(n-1)}{2} \rightarrow O(n^2)$

输入 → 预处理 → 过滤后的物体对 → 逐对检测 → 输出

◆ 加速思路

□ 通过**预处理**步骤提前排除**绝大部分**不相交的空间体对

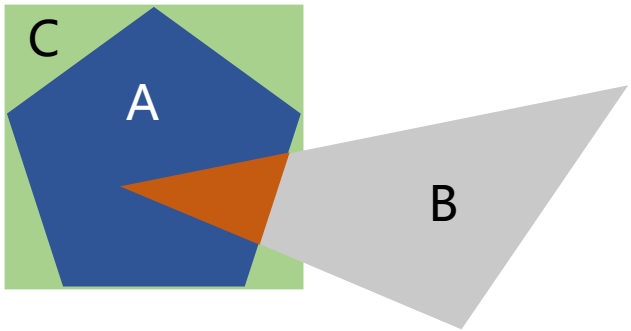
◆ 前提假设

□ 空间体的碰撞是**非频繁**的（或者说碰撞发生几率较低）

◆ 基本原理

□ 如果两个空间体具有公共部分，那么他们的**包围盒**也一定具有公共部分

⇐ 若 A ⊆ C 且 A ∩ B ≠ ∅，则 A ∩ B ⊆ A ⊆ C



◆ 包围盒的选择

✓ 能够比较**紧密**的包裹空间体

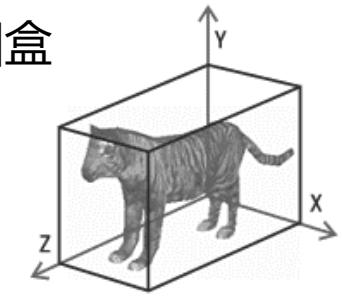
✓ 复杂空间体的包围盒也能**容易求出**

✓ 包围盒之间的碰撞检测**容易实现**（速度比较快）

AABB包围盒

Axis aligned bounding box

轴对齐包围盒



相交快速排除模块

成果展示

空间三角形相交判断模块

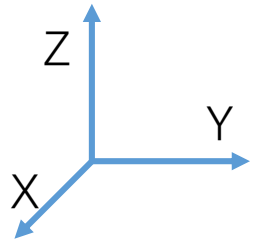
相交快速排除模块

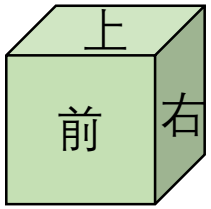
多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

● AABB包围盒






```
public interface IBoxLine
{
    /// <summary> 包围盒，P1为小点，P2为大点
    67 个引用
    Line BoxLine { get; }
}
```

◆ 表示方法

- 上平面+下平面+左平面+右平面+前平面+后平面（**6个坐标**）
- 以**线段**的形式表示（P1为较小的XYZ组成的点，P2为剩下的XYZ组成的点）

◆ 求物体的包围盒

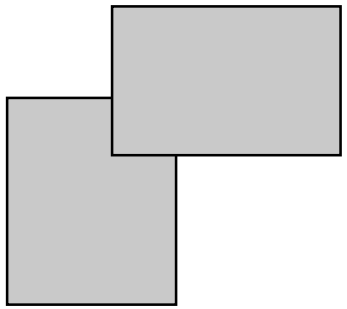
- **遍历**物体的所有顶点，分别获取最小和最大的X、Y、Z坐标

```
for (int i = 0; i < _elements.Length; i++)
{
    infos.Add(i);
    var element = _elements[i];
    if (element.BoxLine.X1 < boxLine.X1)
        boxLine.X1 = element.BoxLine.X1;
    if (element.BoxLine.Y1 < boxLine.Y1)
        boxLine.Y1 = element.BoxLine.Y1;
    if (element.BoxLine.Z1 < boxLine.Z1)
        boxLine.Z1 = element.BoxLine.Z1;
    if (element.BoxLine.X2 > boxLine.X2)
        boxLine.X2 = element.BoxLine.X2;
    if (element.BoxLine.Y2 > boxLine.Y2)
        boxLine.Y2 = element.BoxLine.Y2;
    if (element.BoxLine.Z2 > boxLine.Z2)
        boxLine.Z2 = element.BoxLine.Z2;
}
```

◆ 包围盒碰撞检测

- X方向**区间相交**且Y方向区间相交且Z方向区间相交

⇒ A_X1 ≤ B_X2 且 B_X1 ≤ A_X2 （X1为较小坐标）



```
/// <summary> 判断两个物体的包围盒是否碰撞
2 个引用
public static bool DoCollideBox(this IBoxLine box1, IBoxLine box2)
{
    return box1.BoxLine.X1 <= box2.BoxLine.X2 && box2.BoxLine.X1 <= box1.BoxLine.X2 &&
        box1.BoxLine.Y1 <= box2.BoxLine.Y2 && box2.BoxLine.Y1 <= box1.BoxLine.Y2 &&
        box1.BoxLine.Z1 <= box2.BoxLine.Z2 && box2.BoxLine.Z1 <= box1.BoxLine.Z2;
}
```

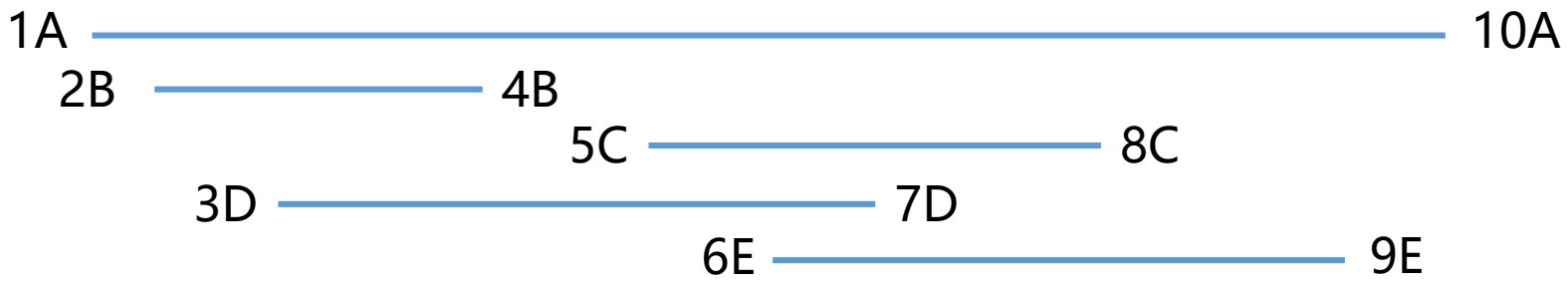
相交快速排除模块

BoxLines → 端点排序 → 遍历序列 → 包围盒相交对

Line Sort算法

◆ 基于**排序**的**多区间相交**检测算法，是自命名的**创新**算法

□ 假设有5条线段A、B、C、D、E，将它们的10个端点按照X坐标进行排序



→ **ABDBCEDCEA**

- ABDBCEDCEA → S = [A]
- ABDBCEDCEA → S = [A B]
- ABDBCEDCEA → S = [A B D]
- ABDBCEDCEA → S = [A D]
- ABDBCEDCEA → S = [A D C]
- ABDBCEDCEA → S = [A D C E]
- ABDBCEDCEA → S = [A C E]
- ABDBCEDCEA → S = [A E]
- ABDBCEDCEA → S = [A]
- ABDBCEDCEA → S = [ ]

——
(A,B)
(A,D) (B,D)
——
(A,C) (D,C)
(A,E) (D,E) (C,E)
——
——
——
——

- 若S中无元素X则加入S，否则从S中删除
- 元素X加入S时，与S中所有元素相交

→

(A,B) (A,C) (A,D) (A,E)
(B,D)
(C,D) (C,E)
(D,E)

相交快速排除模块

成果展示

空间三角形相交判断模块

相交快速排除模块

多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

- **Line Sort算法**

输入 → 求包围盒 → LineSort包围盒相交检测 → 逐对检测 → 输出

  - 三维情况
    - 对包围盒（BoxLine）进行**3次**区间相交检测（X向、Y向、Z向），求出相交对的**交集**

      例如：3次检测结果分别为(AB, AD, BC, **CD**)、(AB, AD, BD, **CD**)、(AC, BC, BD, **CD**)
  - 复杂度分析
    - 排序 $O(nlogn)$ + 遍历 $O(n)$ ⇒ $O(nlogn)$
  - 一些细节
    - 节点**基于坐标值排序**，但需附带所属**包围盒的编号信息**
    - 临时集合S需要进行频繁的查增删，采用**HashSet结构**
    - 检测结果采用**Dictionary+HashSet结构**，方便求交集
    - 需要考虑节点**坐标值重合**的特殊情况

```
protected Dictionary<int, HashSet<int>> Filter(ValueNode<int>[] valueNodes, bool[] meet,
    Dictionary<int, HashSet<int>> possible)
{
    Dictionary<int, HashSet<int>> newPossible = new Dictionary<int, HashSet<int>>();
    Array.Sort(valueNodes);
```

```
public class ValueNode<T> : IComparable
{
    /// <summary> 用于排序的值
    7 个引用
    public double Value { get; set; }

    /// <summary> Value的附加信息，如所属元素编号
    6 个引用
    public T Info { get; set; }

    2 个引用
    public ValueNode(double value, T info)...

    1 个引用
    public int CompareTo(object obj)
    {
        double d = Value - (obj as ValueNode<T>).Value;
        if (d < 0)
            return -1;
        else if (d > 0)
            return 1;
        else
            return 0;
    }
}
```

泛型

多个空间体相交检测模块

成果展示

空间三角形相交判断模块

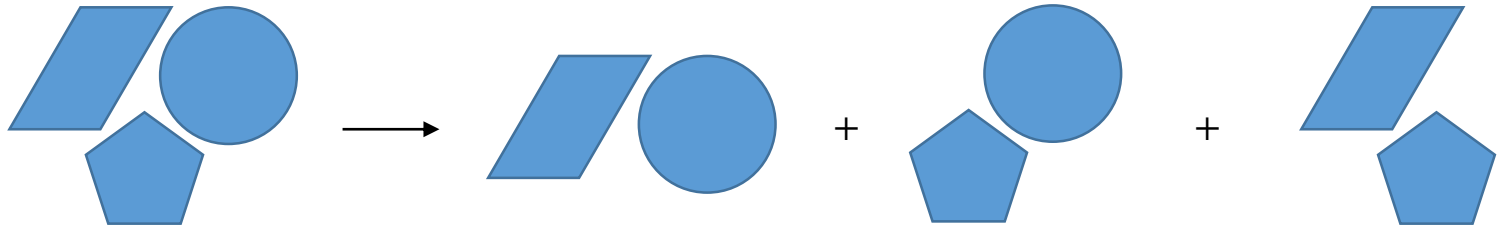
相交快速排除模块

多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

● 多空间体相交检测



◆ 实现思路

- 将空间体视为整体，先进行空间体的包围盒快速检测，再对得到的结果进行两两检测

  两两检测时，将空间体拆分成三角面片，然后进行快速碰撞检测
- 直接将空间体拆分为不同的三角面片，以三角面片为基本元素进行快速碰撞检测 √

◆ 包围盒碰撞检测器（BoxChecker）

- 检测基本单元Element，具有**包围盒**和**物体编号**
- 具有相同物体编号的Element不会被检测，物体编号为-1时表示不属于任何物体

Element<Triangle>

Content	BoxLine	BodyId
triangle1	box1	-1
triangle2	box2	0
triangle3	box3	0

```
public interface IBoxCheck
{
    /// <summary> 要比较的所有元素
    8 个引用
    IElement[] Elements { get; set; }

    /// <summary> 测试得到所有包围盒相交的编号对
    5 个引用
    Dictionary<int, HashSet<int>> BoxCheck();
}
```

```
public class Element<T> : IElement
    where T : IBoxLine
{
    /// <summary> Element的内容
    8 个引用
    public T Content { get; set; }

    15 个引用
    public int BodyId { get; set; }

    /// <summary> Element的包围盒
    34 个引用
    public Line BoxLine => Content.BoxLine;

    /// <summary> 体编号为-1时表示不属于任何物体
    1 个引用
    public Element(T content, int bodyId = -1)
    {
        Content = content;
        BodyId = bodyId;
    }
}
```

成果可视化展示模块

成果展示

空间三角形
相交判断模块

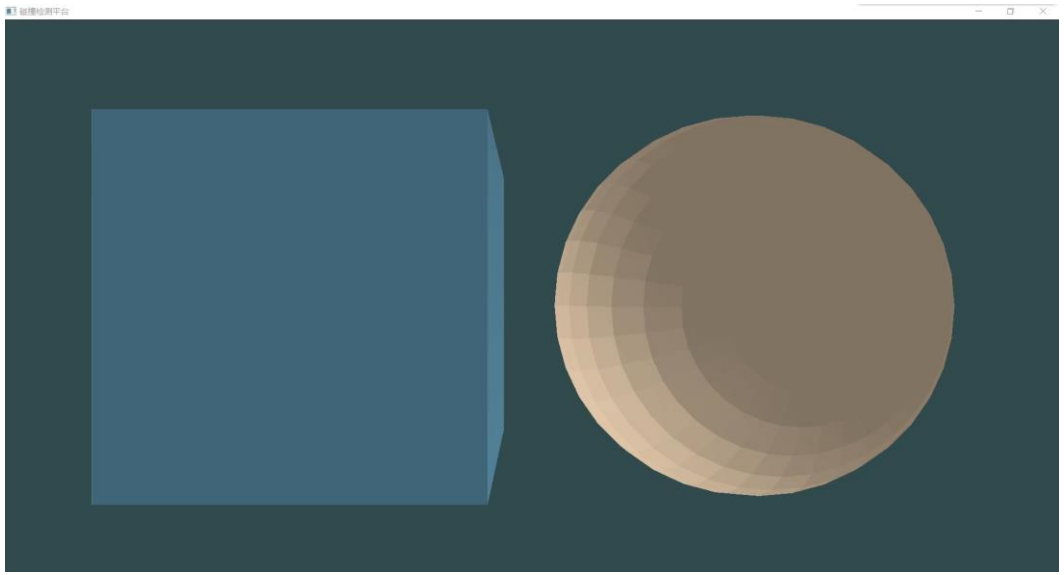
相交快速
排除模块

多个空间体
相交检测模块

成果可视化
展示模块

多种算法
对比验证模块

- **可视化展示模块**
  - 模块简介
    - 基于OpenTK开发的**可视化平台**
      支持三维模型的**动态浏览与交互**
    - 空间体对的实时动态检测
    - 多空间体的碰撞检测与展示



成果可视化展示模块

成果展示

空间三角形
相交判断模块

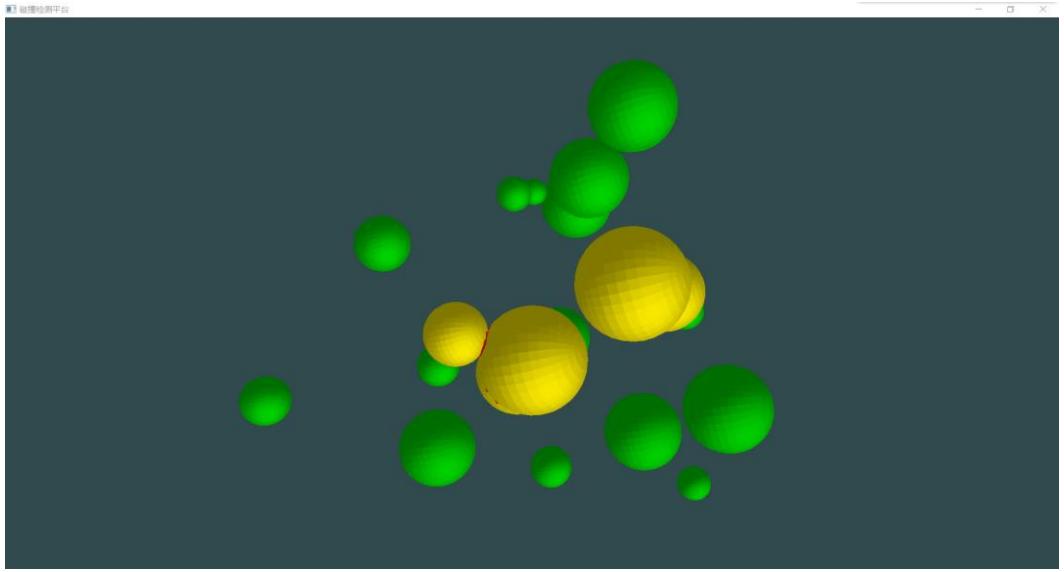
相交快速
排除模块

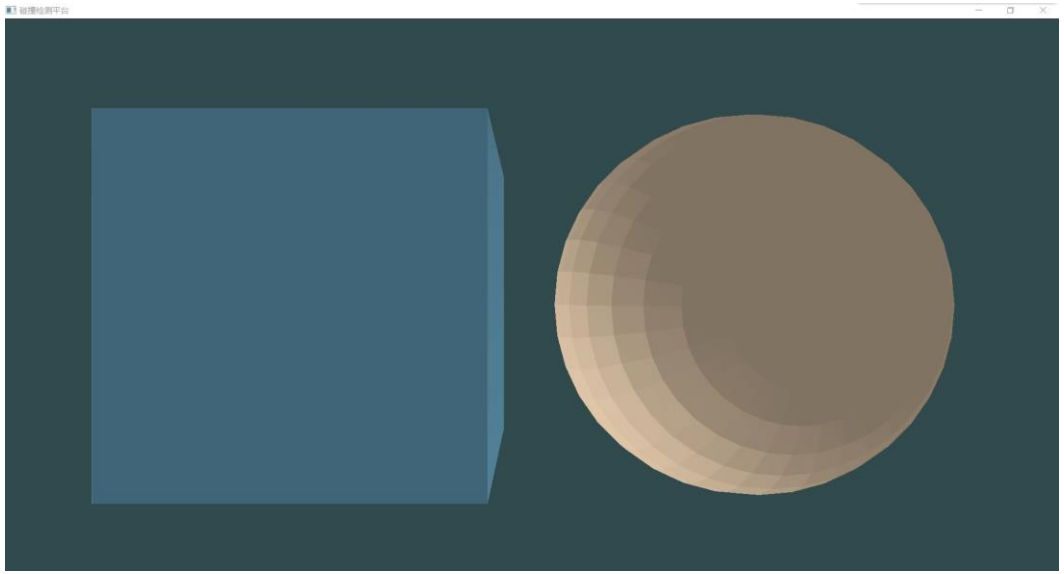
多个空间体
相交检测模块

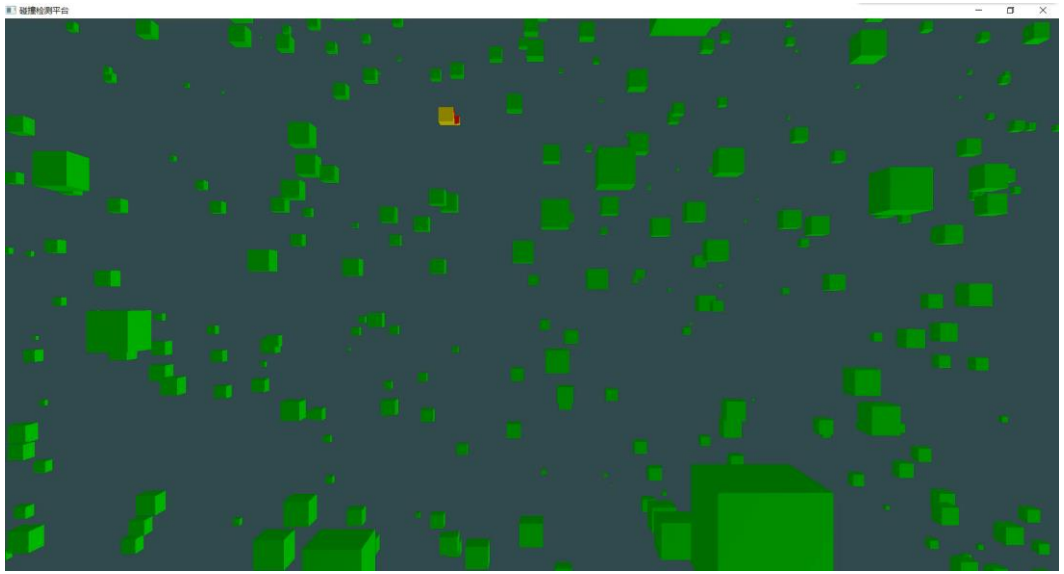
成果可视化
展示模块

多种算法
对比验证模块

- **可视化展示模块**
  - ◆ 模块简介
    - 基于OpenTK开发的**可视化平台**支持三维模型的**动态浏览与交互**
    - 空间体对的实时动态检测
    - 多空间体的碰撞检测与展示







多种算法对比验证模块

成果展示

空间三角形相交判断模块

相交快速排除模块

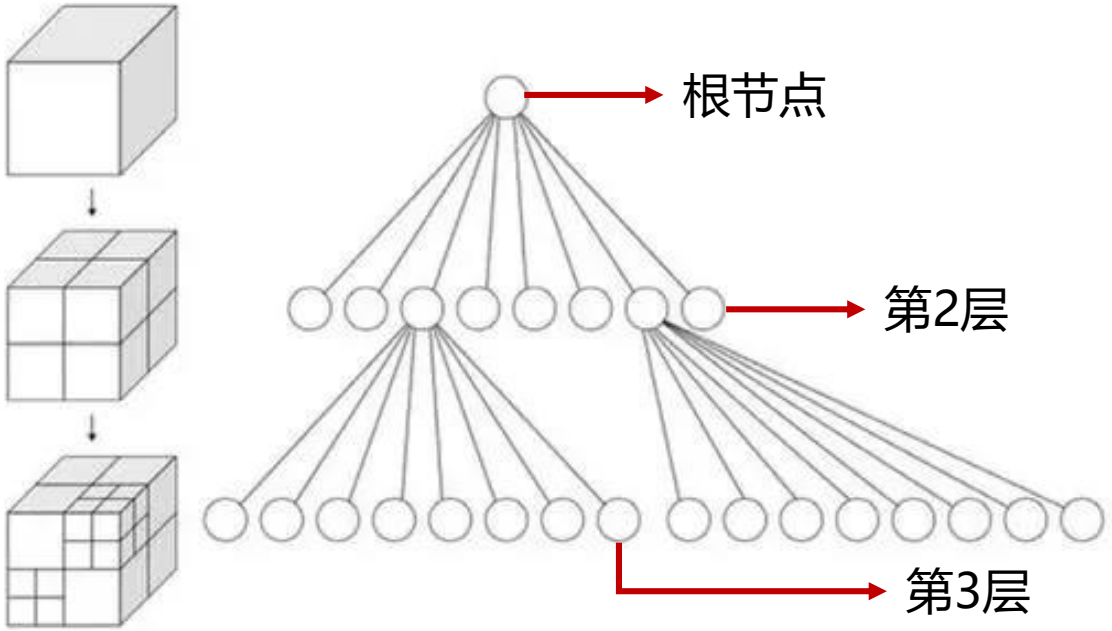
多个空间体相交检测模块

成果可视化展示模块

多种算法对比验证模块

八叉树算法简介

- 每个**子空间**看作一个包围盒，每次将**父空间**均匀划成8份
- 划分的同时父空间包含的物体将被**分配到各个子空间**中
  允许被分配到多个不同的子空间中
- 不同子空间内的物体**必定不相交**（排除了大量检测）
- 最多划分到**20层**，底层空间大小为碰撞检测空间的$1/8^{19}$




```
public class OctreeNode<T>
{
    /// <summary> 节点内容
    12 个引用
    public T Content { get; set; }

    /// <summary> 父节点
    2 个引用
    public OctreeNode<T> Parent { get; protected set; }

    /// <summary> 八个子节点
    10 个引用
    public OctreeNode<T>[] Children { get; protected set; }

    子树名称定义，按二进制规则，便于访问

    /// <summary> parent为null时说明为根节点
    3 个引用
    public OctreeNode(OctreeNode<T> parent)...
}
```

```
public class BoxContent<T> : IBoxLine
{
    /// <summary> 盒子的边界
    36 个引用
    public Line BoxLine { get; protected set; }

    /// <summary> 盒子中的信息 一般为所包含Element的索引
    10 个引用
    public List<T> Infos { get; protected set; }

    2 个引用
    public BoxContent(Line boxLine, List<T> infos = null)...
}
```

多种算法对比验证模块

算法对比结果

序号	总三角面片数	包围盒相交数	实际相交数	检测用时 /s		
				两两检测法	八叉树法	LineSort算法
1	10000	334	32	3.370	0.249	0.170
2	10000	422	43	2.699	0.374	0.232
3	20000	32	4	——	0.873	0.178
4	20000	366	19	——	0.957	0.508
5	20000	1210	67	——	1.155	0.997
6	40000	30	7	——	3.310	0.942
7	40000	344	18	——	2.678	2.798
8	40000	526	21	——	2.364	1.734
9	40000	790	50	——	3.196	1.692
10	80000	1668	78	——	5.228	2.287

LS算法略慢于八叉树算法


清華大學 深圳国际研究生院
Tsinghua Shenzhen International Graduate School

Part 4

总结展望

总结展望

研究目标

- ✓ 针对基于三角形网格描述的实体模型，提出一个**高效**的碰撞检测算法
- ✓ 实现多个空间体（每个空间体含有多个三角面片）之间的相交检测

研究展望

◆ 算法**占用内存较多**

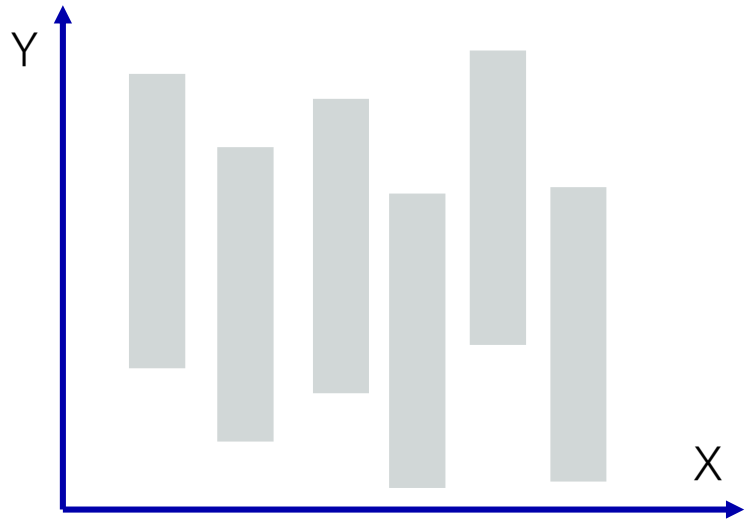
由于要临时存储区间相交检测的结果，然后再求交集，因此会占用更多的内存，特别是在一些不利的情况下会导致较低的效率

- ➢ 计算区间相交时同时考虑三个方向的信息（不用临时存储）
- ➢ 采用八叉树算法先对空间做粗略划分，再采用LineSort算法

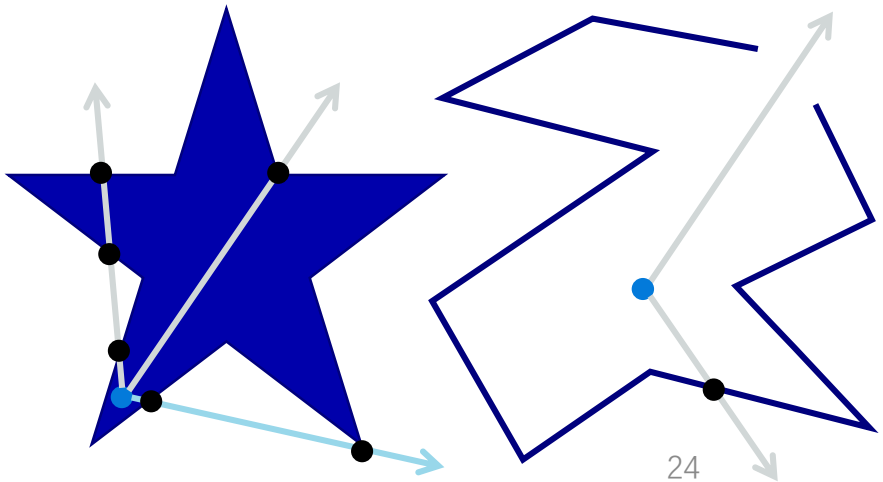
◆ 尚未实现**包含检测**

一方面，物体的包含关系要考虑的情况非常多而且复杂；另一方面，也很难确定物体是否是封闭的

- ➢ ……



物体的包围盒都不相交，但在Y方向全部区间相交



感 谢 聆 听 与 指 导

THANK YOU FOR LISTENING AND GUIDING