特别说明

.doc in alpha 豆丁

Bullet 物理引擎中文文档

英文文档正在更新中

详情查看 Wiki 和论坛（http://bulletphysics.com）

翻译：五行谦饼
Email：cqw1022@gmail.com

目录

1. 简介

1.1. 类库描述

Bullet 物理引擎是开源的，专业的集刚体、软体和碰撞检测于一身的动力学类库。在 Zlib 授权下用户可以免费用于商业开发。

1.2. 2.74 版新添加的元素

1.3. 计划

1.4. 主要特性

1） 在 Zlib 授权中使用开源的 C++代码，可免费用于包括 PLAYSTATION 3, XBox 360, Wii, PC, Linux, Mac OSX and iPhone 平台的商业开发。

2） 包括射线和凸扫测试在内的离散和连续碰撞检测，可检测的碰撞物体形状包括凹凸网格和所有的基本形状。

3） 快速和稳定的刚体动力约束和求解、动态车辆、人物控制和滑动器、铰链、普通的 6 自由度和针对碎布木偶的圆锥和扭曲约束。

4） 软体动力学方面可用于布料、绳子和双向变容的刚体，包括约束支持。

5） Maya 动态插件，集成 Blender 渲染器，支持 COLLADA 格式导入导出。

1.5. 联系和支持

1） 公开的帮助和回馈论坛 http://bulletphysics.com

2） PS3 合法的开发用户可以从 https://ps3.scedev.net/projects/spubullet 下载到用于 Cell SPU 的优化版本

2. 快速入门

1） 下载

Windows 平台的开发者可以从 http://bulletphysics.com 下载到 BulletZIPPED 格式的源代码。

MacOSX、Linux 和其他平台的开发者可以下载到 GZIPPED TAR 格式的压缩包。

2） 生成

Bullet 能在所有的平台编译，并且包含了所有的依赖项。

Windows Visual Studio 平台的所有项目文件都在 Bullet/msvc 文件夹中，主要的解决方案在 Bullet/msvc/8/wksbullet.sln 中。

3） 测试示例

可以尝试着以运行体验 Demos/AllBulletDemos 来作为学习使用 Bullet 的起点。Bullet 可以用于一下几个方面，完整刚体模拟，碰撞检测类库和（low level snippets such as GJK closest point calculation.）。这方面的帮助支持可以在 Directories 下的 doxygen documentation 中找到。

4） 集成到应用程序

仔细研究 CcdPhysicsDemo 项目中是如何创建一个 btDiscreteDynamicsWorld，btCollisionShape,btMotionState 和 btRigidBody。在动态世界中每帧调用 stepSimulation，同时动态世界能转换你的图形对象。前提条件：

#include “btBulletDynamicsCommon.h “

添加头文件路径 Bullet/Src

添加类库 libbulletdynamics, libbulletcollision, libbulletmath

5） 只整合碰撞检测类库

Bullet 的碰撞检测也可以脱离 Dynamics/Extras 而单独使用，仔细研究起步的碰撞

接口示例，尤其是 CollisionWorld 这个课程。前提条件：

#include “btBulletCollisionCommon.h”

添加路径 Bullet/src

添加类库 libbulletcollision, libbulletmath

6） 只用切片？

3. 类库概述

3.1. 简介

物理引擎的主要任务是碰撞检测，解决碰撞和其他约束，给虚拟世界中的所有物体提供世界转换的更新。本章将会大体上介绍一下所有部分中共享的刚体动力学管道、基本数据类型和数学类序

3.2. 软件设计

Bullet 是定制式和模块化开发的。开发者可以按如下方式自由使用

1） 只使用碰撞检测组件

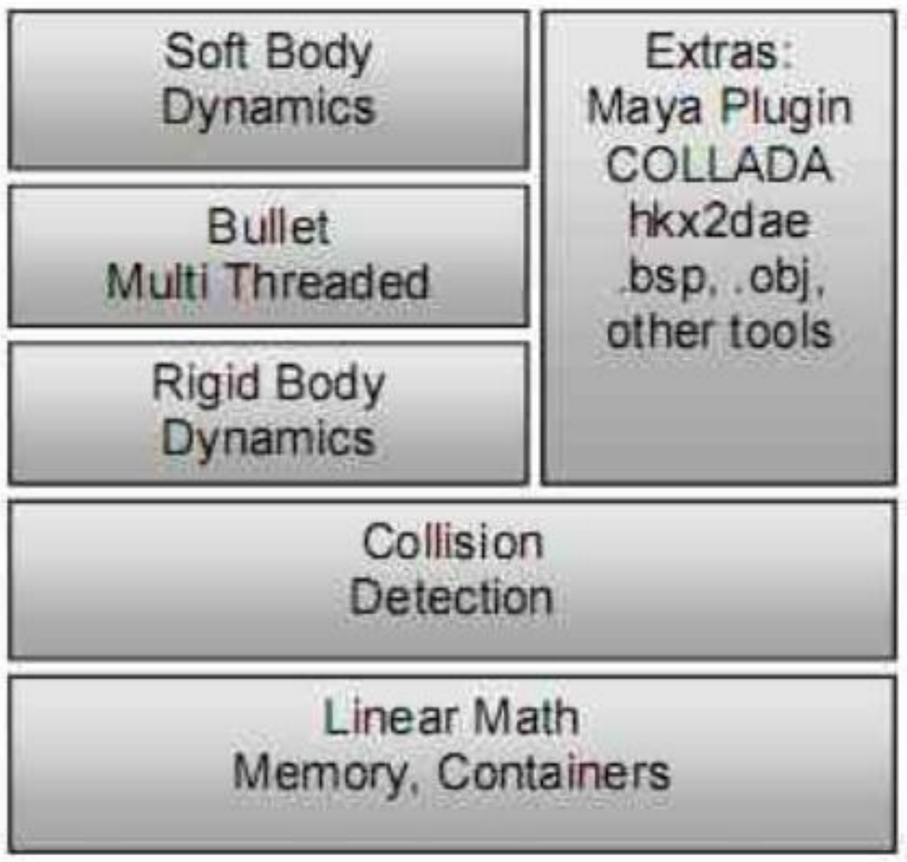
2） 使用刚体动力学组件而不使用软体动力学组件

3） 以众多方式使用类库的一部分和扩展类库

4） 选个单精度或者双精度版本的类库

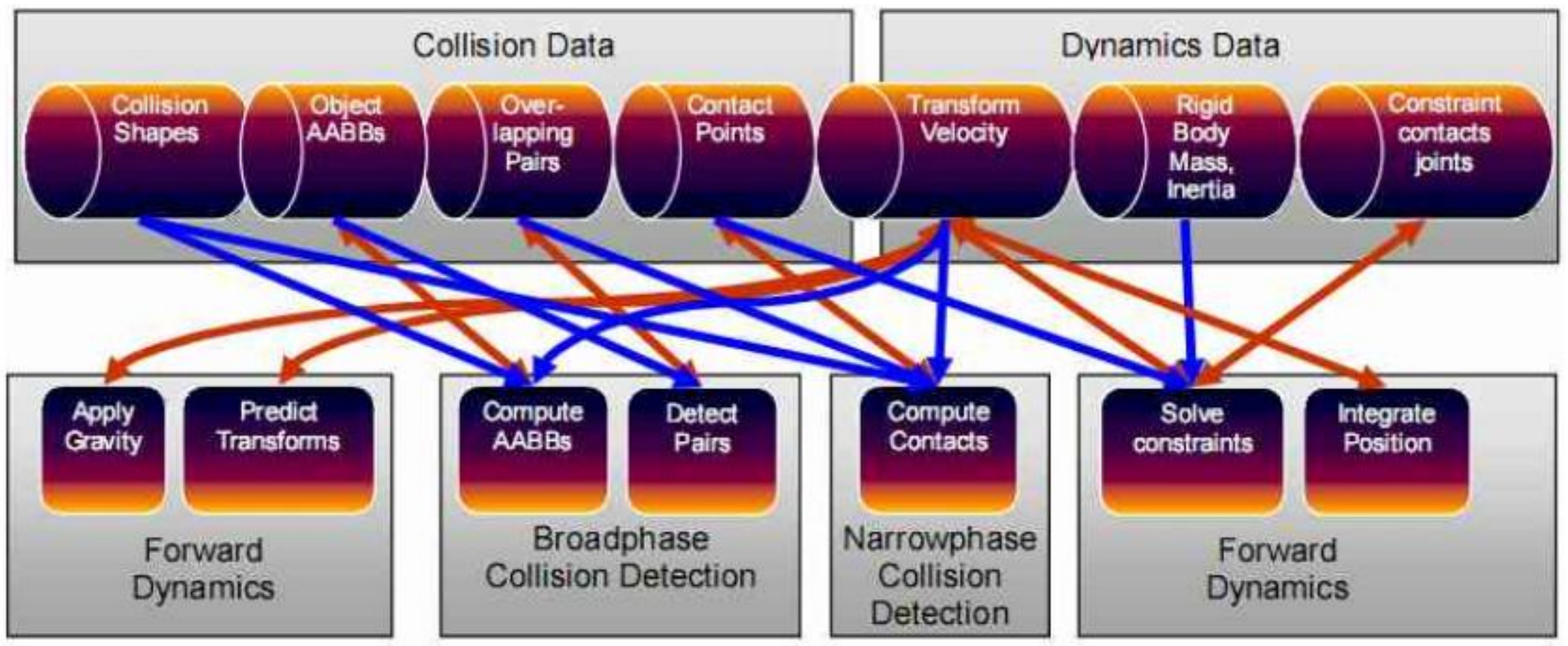
5） use a custom memory allocator, hook up own performance profiler or debug drawer

主要组件的机构如下：



3.3. 刚体物理管线

在介绍细节前，下图展示了在 Bullet 引擎动态管线中的主要的数据结构和计算阶段。管线从左到右执行，以接受重力作用开始，位置整合结束，然后更新世界转换。



整个物理管线的计算过程和它的数据结构在 Bullet 的动态世界中完全展现出来。当在动态世界中执行 stepSimulation 时，所有上面的阶段都会执行。默认的动态世界实现是 btDiscreteDynamicsWorld。

Bullet 让开发者明确地选择动态世界的几个部分，如初测阶段（broadphase）碰撞检测，细测阶段（narrowphase）碰撞检测（分配器）和约束解决程序。

3.4. 整体概貌

如果你想在你自己的 3D 程序中使用 Bullet，你最好按照 Bullet/Demos/HelloWorld 文件夹中 HelloWorld 中的步骤来。简单地说：

创建一个 btDiscreteDynamicsWorld 或者 btSoftRigidDynamicsWorld

这写来自 btDynamicsWorld 的类提供了一个高水平管理动态物体和约束的接口，也实现了每帧更新所有物体状态的功能。

创建 btRigidBody 然后添加到 btDynamicsWorld

要创建 btRigidBody 或者 btCollisionObject，你还要提供：

1） 质量，动态移动的物体质量为正数，静态物体则为 0；

2） 碰撞形状，想盒子、球体、圆锥体、三角网格、凸壳

3） 材料的属性如摩擦系数和 restitution

每帧更新仿真情况

stepSimulation

在动态世界中调用 stepSimulation，btDiscreteDynamicsWorld 会自动通过插值取代小仿真 stepSimulation 来计算 timestep 变量，btDiscreteDynamicsWorld 使用内置的大小有 60Hertz 的 timestep。stepSimulation 会进行碰撞检测和物理仿真，通过调用 btMotionState 的 setWorldTransform 来更新主动物体的世界转换。

下一章将提到更多关于碰撞检测和刚体运动学的内容。许许多多的细节在 Bullet/Demos 都展示出来。如果你不能找到某个函数，请去 http://bulletphysics.com 论坛查询。

3.5. 基本数据类型和数学类库

基本数据类型，内存管理和内存容器都保存在 Bullet/src/LinearMath 文件中。

btScalar

btScalar 是 Bullet 中用来表示浮点数类型的词。为了能用当精度浮点数和双精度浮点数

编译类库，Bullet 在类库中使用 btScalar 数据类型代替浮点数类型。默认的 btScalar 是定义为 float，用户可以在自己创建的系统中或者在 Bullet/src/LinearMath/btScalar.h 文件头部定义 BT_USE_DOUBLE_PRECISION 来更改 btScalar 表示的类型。

btVector3

3D 的位置和向量都可以用 btVector3 来表示。btVector3 有 3 个 btScalar 类型的数据 x,y,z 变量组成。实际上它也有第四个因为定向和 SIMD 兼容的原因没有使用的变量 w。很多操作都很用在 btVector3 上，比如加法、减法和或获取向量长度。

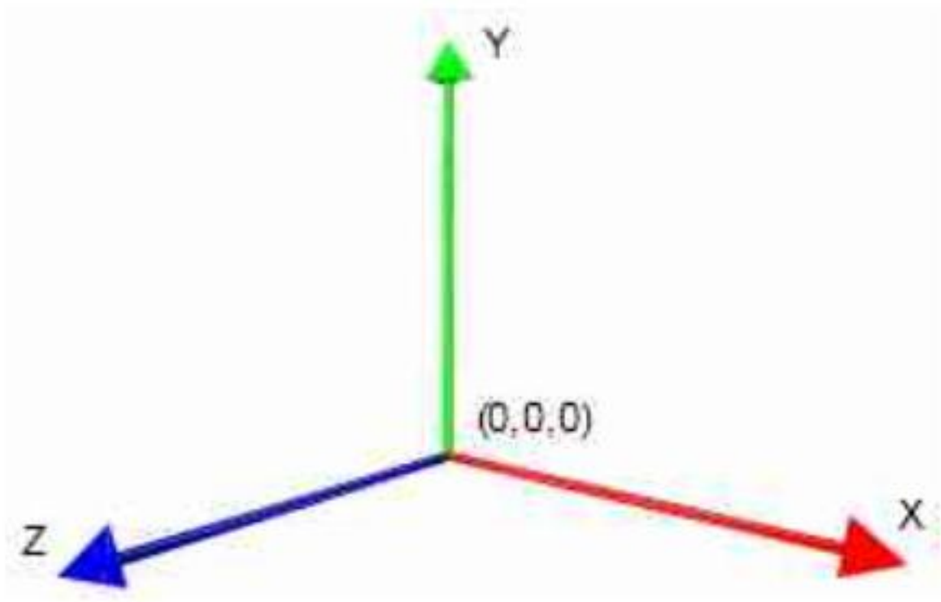
btQuaternion 和 btMatrix3x3

btQuaternion 和 btMatrix3x3 都可以用来 3D 定向和旋转。

btTransform

btTransform 是位置和方向的组合，点和方向可以用它进行坐标系转换。不允许进行裁剪和放大缩小。

Bullet 采用的是右手系坐标轴，如图：



btTransformUtil 和 btAabbUtil 提供常用的函数来进行转换和 AABBs

3.6. 内存管理、分配和容器

很多情况下数据是按 16 位分配的，比如在 CellSPU 中使用用 SIMD 或者 DMA 传输的时候。Bullet 提供默认的内存分配器来处理内存分配，用户也可以自由使用其他的内存分配器。Bullet 中使用以下方法来分配内存：

btAlignedAlloc，允许指定分配内存的大小并且和分配内存

btAlignedFree，释放由 btAlignedAlloc 分配的内存

用户可以选择下面的两个方法之一来覆盖默认的内存分配器：

btAlignedAllocSetCustom 当用户的分配器不支持内存分配的时候调用

btAligneAllocSetCustomAligned 可以用来设置用户自己的内存分配器

用户可以使用下面的宏来确保结构和类对象内存的自动分配：

ATTRIBUTE_ALIGNED16(type) variablename 创建一个 16 位的变量

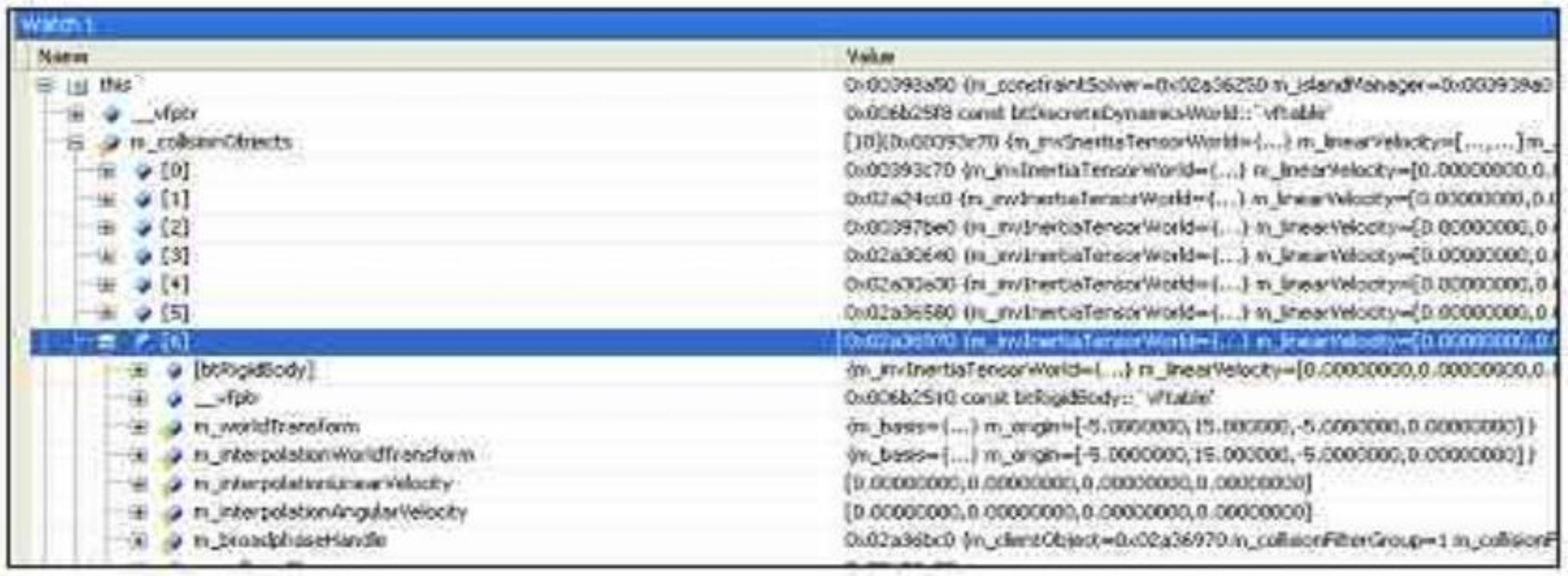
很多情况下我们都需要维护数组对象，开始的时候 Bullet 类库是用 STL std::vector 数据

结构给数组，但是为了可移植性和兼容性作者已经更换为使用自己的数组类了

btAlignedObjectArray 和 std::vector 非常相似，它使用对齐分配器来保证分配，它还提供数组排序、快速排序、堆排序的接口。

To enable Microsoft Visual Studio Debugger to visualize `btAlignedObjectArray` and `btVector3`, follow the instructions in Bullet/msvc/autoexp_ext.txt

为了使得微软的 Visual Studio 的调式器能监视到 btAlignedObjectArray 和 btVector3，请按照 Bullet/msvc/autoexp_ext.txt 中的指示去做。



3.7. 时间和性能分析

为了找出性能瓶颈,Bullte 使用宏来进行分级性能测量

1）btClock 用微妙的精度来计算时间

2）BT_PROFILE(section_name)标识一个分析的开始

3）CProfileManager::dumpAll(); 将一个分级性能完全输出在控制台中，并调用后面的仿真步骤

4）CProfileIterator 遍历性能分析树的类

当然也可以通过在 Bullet/src/LinearMath/btQuickProf.h 中定义#define BT_NO_PROFILE 1 来取消性能分析

3.8. 调试画图

视觉调试仿真数据结构相当有用，例如它可以让你验证物理仿真数据和图形数据的吻合程度，也可以现实缩放比例问题，错误的约束帧和限制问题。

btIDebugDraw 是用于调试画图的接口，来源于用户实现的 drawLine 等其他方法。

4. Bullet 的碰撞检测

4.1. 碰撞检测

Bullet 的碰撞检测系统提供的算法和加速结构来进行最近点（距离和渗透）查询、射线检测和凸扫面检测，主要的数据结构如下：

btCollisionObject：拥有一个世界转换和碰撞图形的对象

btCollisionShape： 描述碰撞图形和碰撞对象，如盒子、球体、凸壳或者三角网格。一个简单的碰撞图形可以在很多的碰撞对象中共享使用

btGhostObject：一个特殊的 btCollisionObject，用来进行局部快速碰撞查询

btCollisionWorld：保存所有的 btCollisionObject 并且提供接口来进行查询碰撞图形

在碰撞检测的初测阶段，Bullte 提供了基于重叠包围盒的加速结构去快速排除对象对（pairs of object）。Bullte 有几个不同的初测阶段的加速结构可以好使用：

btDbvtBroadphase：使用一个基于 AABB 树的动态层次包围体

btAxisSweep3 和 bt32BitAxisSweep3：实现增量三维扫描和裁剪

btCudaBroadphase：实现了使用 GPU 图形硬件的快速均匀网格

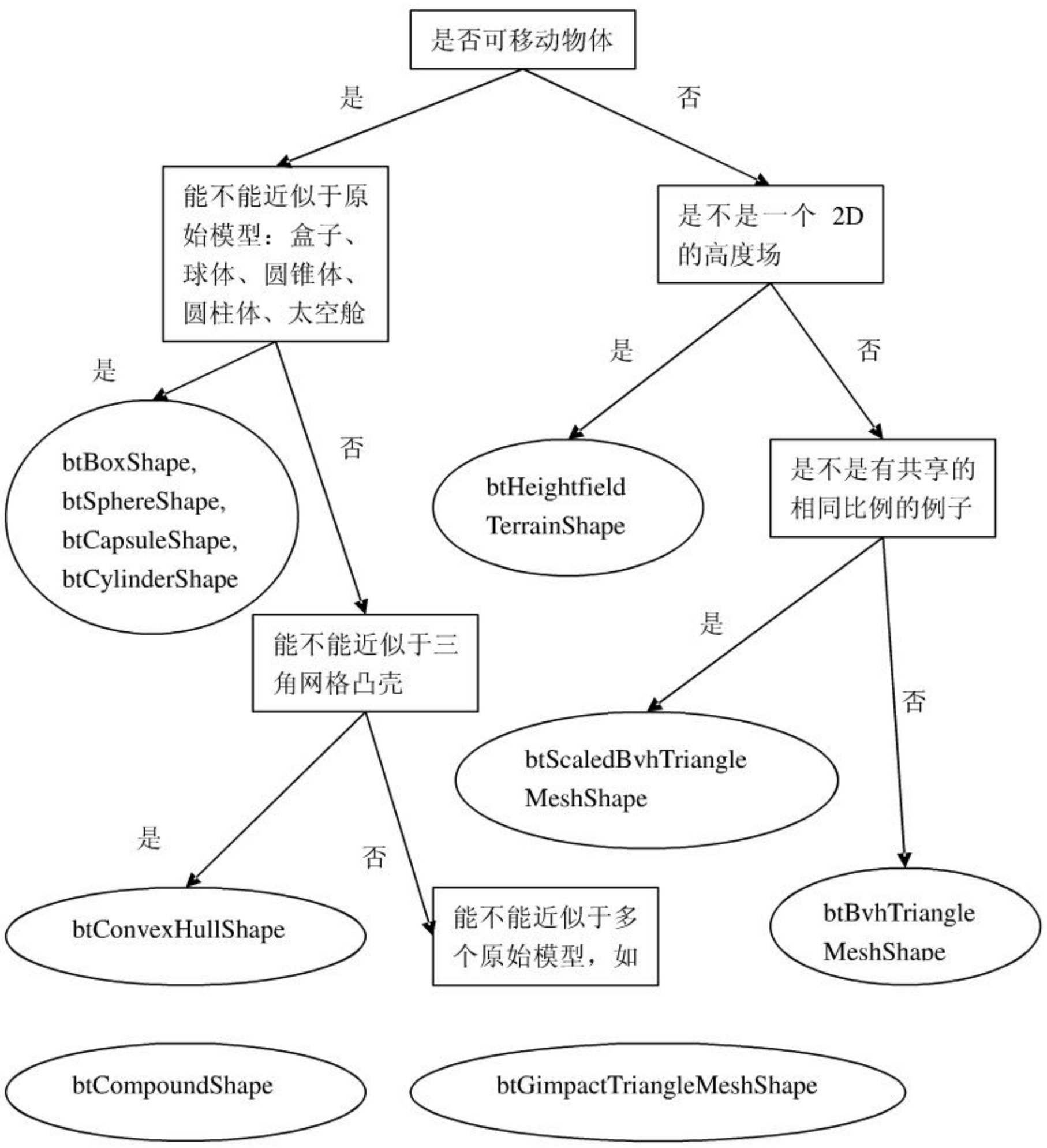
初测阶段在对象对缓存中添加和移除重叠对象对，用户可以选择对象对缓存的类型。

一个碰撞分配器迭代每个对象对，查找匹配基于所包含对象的类型的碰撞算法并且执行碰撞算法来计算接触点。

btPersistentManifold：一个用来保存给定的对象对的接触点的缓存

4.2. 碰撞图形

Bullet 支持很多不同种类的碰撞图形，也可以加入用户自己碰撞图形。为了让程序拥有更好的性能和质量，根据需要选择适当的碰撞图形很重要。下图能帮助你进行选择：



4.3. 凸原始图元

很多原始图元形状是围绕起源的局部坐标框定义的：

btBoxShape：由盒子边长的一半定义

btSphereShape：通过半径定义

btCapsuleShape:：绕着 Y 轴定义的胶囊，同理可定义绕着 X 和 Z 轴。

btCylinderShape：绕着 Y 轴定义的圆柱，同理也可定义绕着 X 和 Z 轴的

btConeShape：绕着 Y 轴的圆锥，同理也可定义绕着 X 和 Z 轴的圆锥

btMultiSphereShape：多个球体组成的凸壳，比如由两个球体组成的胶囊

4.4. 复合图形

多个凸图形使用 btCompoundShape 可以组合到一个合成图形或者复合图形，这是一个有凸图形的部件组成的凹图形，有叫子图形，每个 btCompoundShape 子图形相对于 btCompoundShape 都有自己的局部偏移转换方法。

4.5. 凸核图形

Bullet 有好几个方法支持表示凸三角网格，最早的方法是创建一个 btConverHullShape 并传给一个向量数组。大多数情况下，图形网络都包含很多的向量让 btConverHullShape 直接使用。

4.6. 凹三角网格

在静态环境中，有个非常高效的表示静态三角网格的方法是用 btBvhTriangleMeshShape。这个碰撞图形会从 btTriangleMesh 或者 btStridingMeshInterfase 中创建一个内部的加速器结构。除了在运行时创建树外，它还可以序列化为二进制树保存在硬盘当中。请看 Demos/ConcaveDemo 是怎么样保存和加载这些 btOptimizeBvh 树加速器结构的。当你有几个相同的三角网格而大小不同的示例时，你可以用 btScaledBvhTriangleMeshShape 多次实例化 btBvhTriangleMeshShape 实例。

4.7. 凸分解

理想的情况下，凹网格最好只用于做静态的品，不然的话它的凸壳应该在它的网格转换成 btConverhullshape 后使用，如果单个凸图形不够详尽，可以使用多个凸图形组合成复合物体（btCompoundShape）。凸分解是用来将凹网格分解成几个凸图形。请看 Demos/ConverDecompositionDemo 中的自动分解方法。

4.8. 高度场

Bullet 通过 btHeighfieldTerrainShape 来支持特殊情况下的平面二维凹地形，请在 Demos/TerrainDemo 查看它的用法。

4.9. Buttle 的静态平面（btStaticPlane）图形

顾名思义，btStaticPlaneShape 是表示无限的平面或者半空间，只能用于静态的不会移动的物体

4.10. 碰撞图形缩放

有些碰撞图形能有局部的缩放应用。使用 btCollisionShape::setScaling(vector3)。不同轴的缩放比例不一致的非均匀缩放值能在 btBoxShape,btMultiSphereShape,btConvexShape，btTriangleMeshShape 中能使用，不同轴使用一致的缩放比例的均匀的缩放值能在 btSphereShape 中使用。要说明的是非均匀缩放的球体能用 btScaledBvhTriangleMeshShape 来创建。前面提到国，btScaledBvhTriangleMeshShape 允许实例化一个在非均匀缩放因素下的 btBvhTriangleMeshShape 对象。btUniformScalingShape 允许实例化不同规模的凸图形来减少内存的使用总量。

4.11. 碰撞边框

Bullet 在碰撞图形中使用碰撞边框来提高碰撞检测的性能和准确度。最好不要更改默认

的碰撞边框，而且必须使用正值，边框为 0 可能会出错误。默认的碰撞边框是 0.04，也就是 4 厘米（如果你使用米为单位的话）。

4.12. 碰撞矩阵

Bullet 用分配器给每对碰撞图形分配一个特定的碰撞算法。如下图，分配器的矩阵中默认由下面的算法填充。要解释的是 GJK 算法，GJK 实际上是三个人名的缩写，分别是 Gilbert, Johnson 和 Keerthi，他们是开发凸距离算法的人，它结合了 EPA 的穿透深度的计算。EPA 全称是“Expanding Polythope Algorithm“。Bullet 有自己的免费的 GJK 和 EPA 的实现。

	box	sphere	convex,cylinder cone,capsule	compound	triangle mesh
box	boxbox	spherebox	gjk	compound	concaveconvex
sphere	spherebox	spheresphere	gjk	compound	concaveconvex
convex, cylinder, cone, capsule	gjk	gjk	gjk	compound	concaveconvex
compound	compound	compound	compound	compound	compound
triangle mesh	concaveconvex	concaveconvex	concaveconvex	compound	gimpact

4.13. 注册常用的碰撞图形和算法

用户能使用 btDispatcher::registerCollisionAlgorithm 来注册一个特定的碰撞检测算法，也能使用它来覆盖任意碰撞矩阵中的入口。Demos/UserCollisionAlgorithm 例子中注册了一个 SphereSphere 碰撞检测算法。

5. 碰撞过滤