**Physika核心内容文档**

全柯

**零、概述**

1、Physika总共包含6个项目，另外还有一个用于测试调试的项目。我做的工作几乎全部集中在Physika\_Dynamics中，这其中又几乎全部集中在Collidable\_Objects和Rigid\_Body中。其中Collidable\_Object包含碰撞检测所有内容。Rigid\_Body包含刚体驱动器和碰撞反应。除此之外Physika\_Geometry/Bounding\_Volume中有相关的包围盒代码。

2、系统虽然可以扩展支持二维和三维仿真，但是当前很多算法不能应用于二维仿真，并且没有提供需要实现的接口，所以很多cpp文档末尾的显示定义删除了二维模板类的定义。

3、系统对于不同场景的鲁棒性很差。调实验时主要工作都在针对刚体嵌入等问题修改仿真算法，而不仅是在调参数。

**一、关键类**

1、RigidBodyDriver：刚体驱动器，其中存储了场景中所有刚体，定义了碰撞检测和反应的方法，且负责仿真流程。

2、CollisionDetectionMethodDTBVH：离散碰撞检测（使用双包围盒）。

3、CollisionDetectionMethodCCD：连续碰撞检测。

4、CollisionPairManager：存储当前帧所有碰撞对信息（刚体和面）。

5、ContactPointManager：存储当前帧所有碰撞点信息（点和法向）。

6、RigidResponseMethodBLCP：碰撞反应，解双线性正交方程。

7、RigidBodyDriverUtility：用于计算所需的各个矩阵（质量、雅克比、摩擦雅克比）。

8、RigidDriverPluginMotion：用于设定刚体的固定运动、固定加速、定时运动等动作。

9、RigidDriverPluginRender：用于场景的实时渲染显示，可以设置渲染方式（实心、线框、显示碰撞点、显示碰撞法向）。

**二、仿真流程**

主体内容在RigidBodyDriver::advanceStep()中。具体见下表：

|  |  |
| --- | --- |
| 执行各个插件的onBeginRigidStep() | |
| 检查当前时刻的定时添加刚体 | |
| 施加重力performGravity() | |
| 碰撞检测collisionDetection() | |
| 离散碰撞检测：DTBVH | 连续碰撞检测：CCD |
| 根据物体位置构建包围盒 | 根据物体上一帧和这一帧的位置构建包围盒 |
| 场景包围盒自交 | 场景包围盒自交 |
| 物体包围盒相交测试 | 物体包围盒相交测试 |
| 两物理的三角形相交测试，得到collision\_pairs\_ | 两物体点面测试和边边测试，得到collision\_pairs\_，并将接触点和法向存储 |
| 通过getMeshContactPoint()从collision\_pais\_计算得到contact\_points\_ | 直接从collision\_pairs\_提供的接口得到contact\_points\_ |
| —— | 更新上一帧位置previous\_position\_ |
| 碰撞反应collisionResponse() | |
| 解双线性方程BLCP | 二次模型QCE |
| 通过RigidBodyDriverUtility类中的函数计算得到所需要的矩阵：质量、雅克比、摩擦雅克比、相对速度、摩擦系数、恢复系数 | —— |
| 通过RigidBodyDriverUtility::solveBLCPWithPGS()函数计算各接触点的法向和切向冲量 | —— |
| 将计算出的冲量标记在各刚体中 | —— |
| 更新刚体updateRigidBody()，将已标记的冲量加在刚体上 | |

**三、新开发内容**

1、**连续碰撞检测**

包含点面测试（点P和面ABC）和边边测试（边AB和边CD）。两个测试可以归结为同一个形式的矢量方程，并进一步化简为一个三次方程。

使用方法：

RigidBodyDriver<double, 3> driver;

driver.setCollisionDetectionMethod(new CollisionDetectionMethodCCD<double, 3>()); // 设置碰撞检测方法

2、**定时添加刚体**：例如让10个刚体在仿真开始后4.5秒时添加到系统中。

类TimerRigidBody的成员中有一个刚体指针rigid\_body\_和一个添加时刻add\_time\_。类RigidBodyDriver的成员中有一个优先队列timerRigidBodies，用于存储系统中所有定时添加的刚体。每个仿真步开始时会检查当前时间是否到了优先队列中处于队首的刚体的添加时刻，将所有到了的刚体添加到系统中。

使用方法：

RigidBodyDriver<double, 3> driver;

RigidBody<double, 3> \*object = ...;

driver.addRigidBody(object); // 添加刚体object

driver.addTimerRigidBody(object, 4.5); // 在仿真第4.5秒时添加刚体object

3、**恒定动作和定时动作**：例如让小车恒定速度移动，然后在仿真开始后第3秒后停止。

类RigidPluginMotionCustomization的成员变量有一个刚体指针和各个运动方式的参数（恒定平移运动、恒定旋转运动、周期平移运动、周期旋转运动、恒定加速平移运动、恒定加速旋转运动），成员函数update()负责直接根据当前时刻来更新刚体。类RigidBodyTimerTask的成员变量记录了刚体指针、动作类型、动作发生时间和优先级。类RigidDriverPluginMotion为负责恒定动作和定时动作的插件，其成员变量有一个映射表customized\_motions\_用于存储所有特殊运动方式的刚体，还有一个优先队列tasks\_用于存储所有定时任务。每次更新时检查当前时间和task\_中处于队首的任务的动作发生时间。

使用方法：

RigidBodyDriver<double, 3> driver;

RigidBody<double, 3> \*object = ...;

RigidDriverPluginMotion<double> \*pm = new RigidDriverPluginMotion<double>();

pm->setConstantTranslation(object, Vector<double, 3>(1.0, 0.0, 0.0) // 设置object以恒定速度(1.0, 0.0, 0.0)平移运动

pm->setConstantRotationAcceleration(object, Vector<double, 3>(0.0, 1.0, 0.0)); // 设置object以恒定加速度(0.0, 1.0, 0.0)旋转运动。

pm->addTimerTask(object, 3, RigidBodyTimerTask<double>::ADD\_CONSTANT\_TRANSLATION, Vector<double, 3>(0.0, 0.0, 0.0)); // 设置object从第3秒开始停止平移

driver.addPlugin(pm);