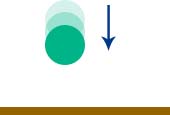
基于物理的动画仿真是计算机图形学下的一个重要的分支，为了让动画看起来更加真实，减轻动画师K帧的工作量。程序员们尝试将真实的物理规律引入计算机中，时至今日，已经发展成了一个十分活跃的技术领域。从最开始简单的弹簧质点模型，粒子模型，到现在复杂的布料、流体、烟雾等等。基于物理的动画仿真技术已经在影视动画行业大放异彩。与此同时，随着计算机CPU和GPU的不断更迭，计算性能得到了飞跃的进步，电脑游戏也进入了次世代，越来越多的在影视动画中应用的离线技术被引入可以实时交互的游戏中。比如这些年大热的基于物理的渲染以及光线追踪等技术，让实时渲染得到了蓬勃的发展，实时物理仿真也不甘落后，PBD以及其延伸技术正不断被越来越多的游戏引擎使用，比如这几年刚出的unreal engine的chaos、Nvidia的Flex，游戏行业真正进入了一个黄金时代。

说到游戏物理引擎，目前市面上比较流行的是2D的box2d，3D的PhysX、Bullet，这些物理引擎想必大部分游戏程序员都有所接触，估计也有不少人像我一样，被这些物理引擎实现的惊艳效果所吸引，觊觎其内部的实现原理。可惜，网络上关于这方面的资料非常少，加上很多学术文章一上来就推导出一纸的数学物理公式，很多介绍相关技术的文章也是零零散散，知其然不知所以然的样子。让gameplay工程师们望而却步，停下了探索的步伐，也失去了一个可以让游戏更有趣的机会。

其实，游戏物理引擎并没有想象中的那么复杂，如果只是游戏中最常用的刚体物理碰撞的话，其核心框架代码量估计也就不足千行。我认为从刚体角度来研究物理引擎是一个很好的切入点。其他物理效果，比如布料，柔体，流体等都可以看作刚体的扩展和延伸。本文尝试从简单的刚体物理出发，由浅入深的介绍物理引擎中的涉及的关键步骤和内部原理。由于时间关系，我并没有实现一个简单的物理引擎框架，这里将拿unreal engine的chaos作为参考范例，该引擎比较新而且是开源的，很有研究价值。

在研究一个技术之前，我们先思考一下这个技术解决了什么问题，实现了什么效果。举个简单的例子，我们让一个球只受重力自由下落



根据初中物理的知识，设小球位置用字母表示，注意它是一个向量，表示三维空间，后面所有的向量或者矩阵我们都用粗体表示，速度是字母，力是字母，质量是，质量的逆是，每次迭代的间隔时间是，则用计算机来模拟小球下落的算法公式是：

这里的下标n表示当前迭代的次数。上面的两行式子叫做欧拉积分。在游戏刚体引擎中，因为对模拟的精确性和稳定性要求不高，所以欧拉积分已经足够使用，在仿真和机器人等领域这方面会比较复杂，显式积分、隐式积分等一系列积分法，大有文章。这里将不会展开，后面有时间再一一细说。

上面两个式子虽然简单，但确实是物理引擎中让物体动起来的关键算法。除了下落，给定一个初速度，一个恒力，我们还可以模拟很多场景，比如按抛物线飞行的炮弹和篮球，爆炸四散的烟花等等。

我们再来看下面一个例子，在天宫空间站无重力环境中有一个方盒



我们按图示中手指的方向推它一下，可以预想到，盒子被推远的同时，还会发生旋转。这里为了计算旋转我们需要引入一系列跟旋转有关的物理量。为了方便记忆我们可以跟位置作类比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 位置 | 速度 | 力 | 质量 | 质量的逆 |
| 朝向 | 角速度 | 力矩 | 惯性张量 | 惯性张量的逆 |

朝向一般用四元数表示，用来表示力作用点到盒子重心的向量（如图中所示），则模拟小球旋转的迭代公式是：

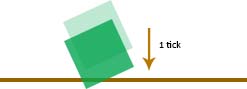
局部惯性张量要转换成世界惯性张量，四元数Q变换为矩阵：

有了位置和朝向的迭代算法，我们就可以通过计算机让物体运动起来，但当世界中有多个物体时，可能会发生碰撞。

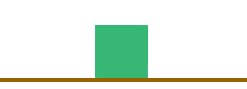
我们设想这样一个场景，一个方盒从空中自由下落，根据前面提到的迭代公式，我们可以很容易的模拟这个盒子下落的过程



上图中绿色盒子的每一个重影都是某一帧的位置，当到了盒子接触地面那一帧时，如下图所示



盒子的最终位置会陷入地下，此时，根据牛顿第三定律，盒子会受到地面的反弹力，如果盒子有水平上的速度的话，还可能会受到摩擦力，盒子在这些力的作用下从地面弹起，直到动能耗散，静止在地面上。



上面这个例子就涵盖了整个刚体物理模拟的过程，业内对这些步骤都有统一的叫法。

按照顺序来说的话就是以下几步：

1. **积分Integrate**（盒子下落位置和朝向的迭代）
2. **碰撞检测 Collision Detect**（检测盒子跟地面是否接触以及相关碰撞信息）
   1. **粗略碰撞检测BroadPhase**（粗略判断盒子跟地面是否接触）
   2. **精确碰撞检测NarrowPhase**（精确计算盒子跟地面的接触点，接触法向量和穿透距离，为后续碰撞处理做准备）
3. **碰撞约束Collision** **Constraint** (用约束求解的方法处理碰撞问题)
   1. **碰撞处理Collision Handling**（模拟反弹力和摩擦力以及小球碰撞后的运动）
   2. **静止接触 Resting Contact**（让盒子静止在桌面上，避免穿透）