# 1、安装与编译

## 1.1 软件下载与安装

（1）从Github上下载 PhysX 3.4（这个版本是一个比较经典的版本）或PhysX4.1的源码。

（2）根据官方文档readme\_windows提出的编译要求，PhysX的编译需要安装CMake 3.12 或以上版本和Python 2.7.6版本

（3）安装DirectX SDK（若只需要使用PhysX的核心库，无需运行PhysX提供的官方示例，可不安装DirectX SDK）。

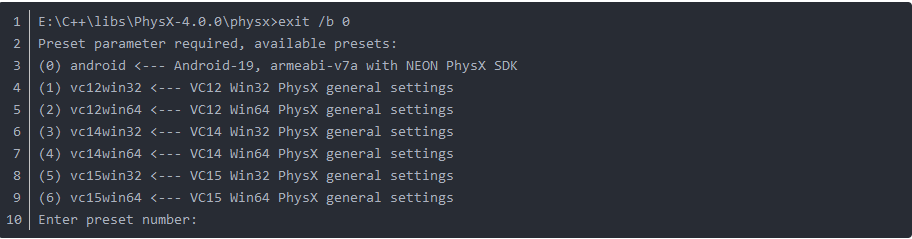
如果遇到了Error Code:s1023的错误，则需要从控制面板中找到并卸载 Microsoft Visual C++ 2010 x86 Redistributable-1010.0.40219，包括x64和x86两个版本。



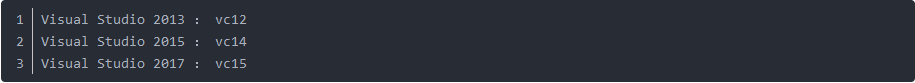
出现这个问题的原因在于电脑中可能安装了Virtual Studio这样的软件，已经安装了C++ Redistributable这个SDK，因此在安装DirectX SDK会重复。

## 1.2 PhysX编译（PhysX3.4无需编译，下载的源码包中自带编译好的版本）

（1）进入工程的PhysX目录，运行 generate\_projects.bat 来生成 VS 解决方案，可以进入编译选择界面：



除了第一个android版本外，其余的都是 windows 平台下使用的，VC版本对应 VS 编译器版本大致关系如下：



选择对应的VS版本后，会在PhysX目录下的compiler文件夹下生成对应版本的解决方案。

（2）使用VS打开。点击“生成”→“批生成”，选择生成debug与release版本的PhysX；之后点击“生成”→“批生成”，选择“INSTALL”debug与release版本，便会自动在PhysX目录下创建install文件夹，里面包含了使用PhysX所需要导入的头文件、lib库。

编译过程中可能会报错误：fatal error C1083: 无法打开包括文件: “d3dx9.h”: No such file or directory，这是由于没有导入DirectX SDK的头文件目录和lib库目录。在解决方案资源管理器中分别选择报错的三个工程：SamplePlatform 、SampleRenderer和Samples，右键→属性，设置DirectX SDK安装目录中的include目录以及lib目录。

## 1.3 工程创建

在VS中创建一个空的win32控制台程序，然后设置相应的附加包含目录、附加库目录以及附加依赖项即可。注意：在使用PhysX中的类时，需要加上命名空间physx，如physx::PxPhysics。

# 2、基本使用

## 2.1 PhysX的基本程序结构

（1）PxScene：物理场景的定义。可以看作是一个虚拟的物理世界，PhysX的所有物理运动都在这个Scene中进行，可以定义场景的重力、阻尼以及恢复系数等。

（2）PxActor：角色节点基类，最常用的例如PxRigidActor，即刚体角色。在场景中，所有参与运算的实体都是一个角色Actor。一个角色Actor包含定义形状的shape和

（3）PxShape：角色的形状定义基类，即角色的物理碰撞体（立方体，球，胶囊体，凸包，三角网格等）。

（4）PxMaterial：角色形状的物理材质，例如它的动、静摩擦力和恢复系数等属性设置。

PxRigidActor的派生类：

PxRigidStatic：静态刚体

PxRigidBody：动力学刚体

PxRigidDynamic

2.2 编程实现

 PhysX编程流程为：首先，定义各种不同的角色（actor）；然后指定每个角色的形状（shape）属性和刚体（body）属性；最后是把这些角色都加入到场景（scene）空间中去，这样就可以构造出一个完整的物理世界。

（1）创建scene

首先我们要创建一个场景的描述(Descriptor)。PhysX就利用这个场景描述结构来创建生成一个场景实例。描述其实就是一个数据结构，主要是用来保存各种在创建实体时所需要的相关信息。可以调整描述体中各种参数来达到不同的效果，如果不作任何修改，这样的话实体在创建时会使用描述体的默认值。

在本例子中，创建一个指定了重力加速以及碰撞检测算法的场景实例。PhysX中提拱了三种碰撞检测算法提拱给大家选择。默认选用的“broad phase-coheret collison detoction”。

|  |
| --- |
|  |

（2）给场景（scene）增加物理材质（Materials）

 物理材质指的是某一具体物体的表面属性和碰撞属性。这些属性可以确定一个物体和另一个物体发生碰撞时，是如何在该的物体上反弹、滑动或者滚动。你可以给场景中的所有物体指定一个相同的默认物理材质。

|  |
| --- |
|  |

以上材质的系数最小值都是0，最大值是1。如果要实现一个物体落在地上会自动弹跳，那就得把还原系数设得大一点。

（3）创建角色对象

 在本程序例子中，只有两个角色实体：地面和方块。我们首先来看如何创建地面。

|  |
| --- |
|  |

创建一个地面角色，只使用了四行代码。首先分别创建一个平面形状描述和角色描述。两个描述都不作任何修改，使用默认值。平面的中心位于世界坐标原点（0,0,0）处，而法线则是指向Z轴的正方向。

第二步,把平面描述添加到角色描述中的形状列表中去，从这里我们也可以看到，一个角色是可以包含多个形状物体的。

第三步,就是把角色加到场景（scene）中去。实体必须包括形状描述和刚体描述两个部分，这里没有为地面角色刚体描述，这是因为在角色创建时会自动生成一个默认的刚体描述。一个刚体的默认值是这样的：它不会移动但是会把与它发生碰撞的物体反弹回去。因为它的质量是无限大的。

 前面介绍了如何创建一个地面，这是场景中最简单的一个角色了。下面我们将要创建一个稍为复杂一点的角色，一个盒子。

|  |
| --- |
|  |

这里我们创建了一个叫"Box"的物理角色。我们可以看到，盒子角色完整地包含了形状和刚体两大部份。和创建平面角色不同的是盒子角色描述中多了"desity"和"globalpose"，这两个参数分别指的是角色的密度和初始位置。PhysX会根据密度和体积来自动计算角色的质量。

"globalpose"指的是在世界位标中的位姿矩阵。值得注意的是：PhysX中与坐标尺寸相关的数值，其单位都是“米”。

（4）开始物理仿真循环

每执行一次物理仿真都需要使用以下接口：

|  |
| --- |
|  |

之后球体就会产生自由落体运动。其中中的参数为仿真步长。通常情况下是使用上一渲染的时间作为仿真步长，这样会让物体运动更加逼真。

（5）对象销毁

仿真结束后需要销毁掉创建的PhysX对象，以释放掉内存。对象销毁所需调用的接口如下：

|  |
| --- |
|  |

# 3、physX与osg结合使用

（1）将physX的物理角色类与osg的Transform类合并，产生一个新的PhysicsNode类；

（2）根据它的osg的子节点几何体信息构建对应的物理碰撞体，注册到物理场景中；

（3）对osg中的根节点和各个子节点分别建立更新回调UpdateCallback。其中，在根节点的更新回调中进行物理仿真；在子节点的更新回调中负责将对应的物理角色的仿真结果传递给osg节点；

physX对象负责物理仿真，osg对象负责结果显示，两者泾渭分明。

结合使用流程图