**PhysX SDK物理引擎开发包使用及获取c++源码教程**

官方网站：

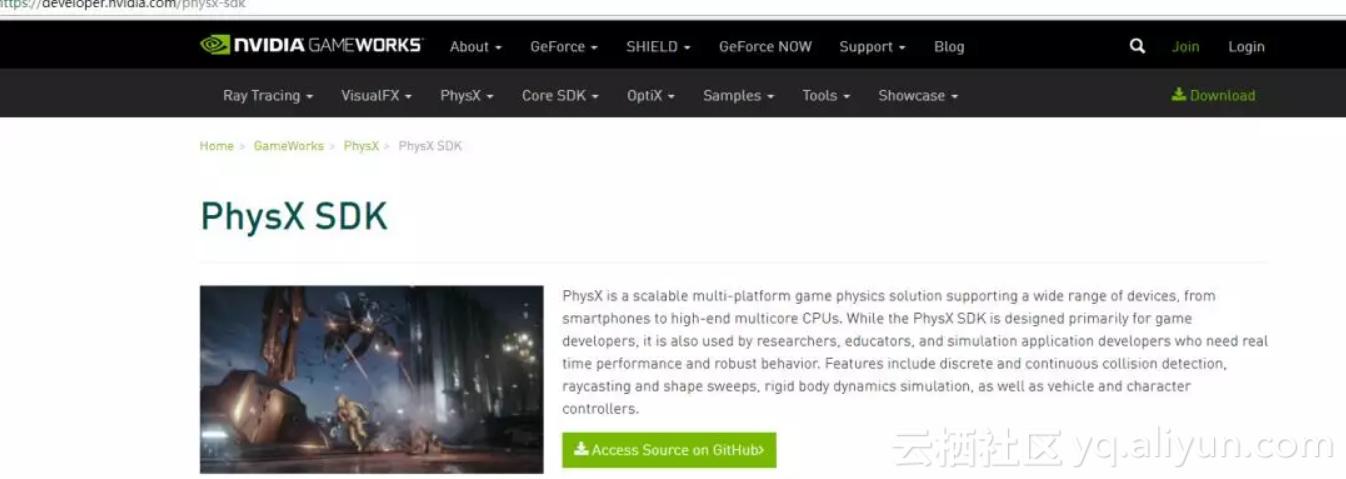
www.nvidia.cn



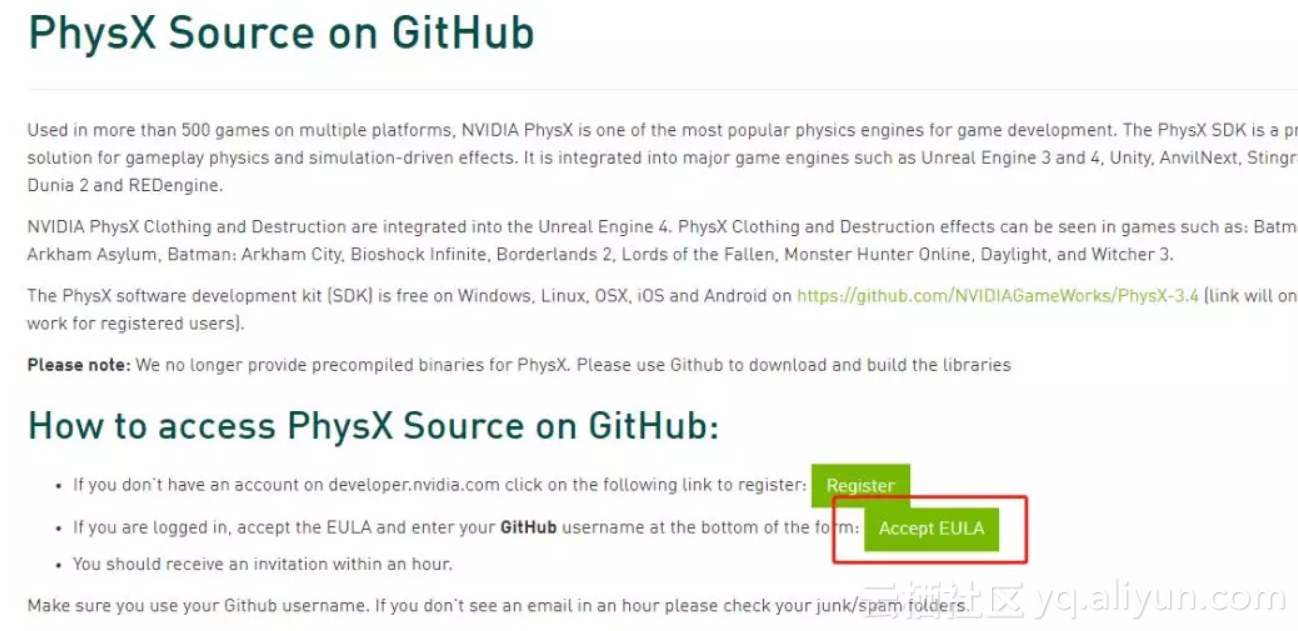
github获取c++版本physx代码流程

第一步 注册账号：

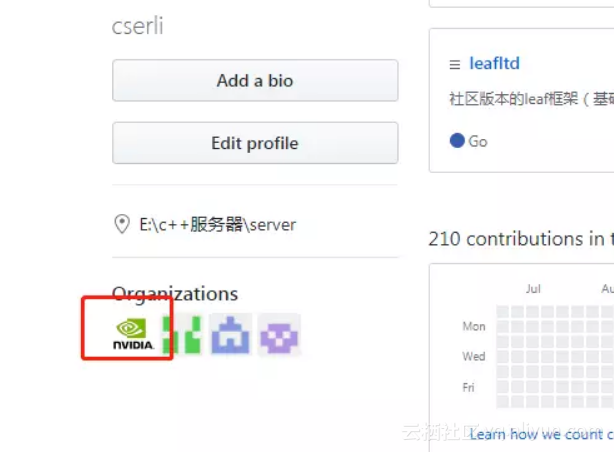
https://developer.nvidia.com/physx-sdk



第二步：



等待官方审核，正常会在第二天就可以再官网添加的GitHub账号下有



****社区/PhysX-3.3版本：****

https://github.com/Golangltd/PhysX-3.3

****社区/PhysX-3.4版本：****

https://github.com/Golangltd/PhysX-3.4

AGEIA的PhysX处理器是世界上首款物理模拟处理器 (PPU), 该处理器将解除中央处理器进行物理模拟的负担。PhysX PPU 的设计构架基于顶点的多线程操作，允许游戏开发人员进行精确、流畅和动画创作和运动模拟，例如毛发、布料、液体、流体等。本文介绍了如何利用PhysX SDK物理引擎开发包来实现我们仿真的效果。

AGEIA的PhysX处理器是世界上首款物理模拟处理器 (PPU), 该处理器将解除中央处理器进行物理模拟的负担。PhysX PPU 的设计构架基于顶点的多线程操作，允许游戏开发人员进行精确、流畅和动画创作和运动模拟，例如毛发、布料、液体、流体等。目前 AGEIA 的PhysX处理器是世界上第一款也是唯一一款专注于物理算法处理器的产品.

利用PhysX SDK物理引擎开发包来实现我们仿真的效果时，一般需要以下几个步骤：

（1） PrintControls();

（2） InitGlut(argc, argv);

（3） InitNx();

（4） glutMainLoop();

（5） ReleaseNx();

其中最为主要的函数是InitNx（），也既是初始化PhysX，创建一个PhysX SDK实例以及建立我们的场景。下面具体分析各个函数的作用。

一．PrintControls();

显而易见，利用该函数的目的是在告诉玩家该如何进行操作。操作的按键可根据自己的喜好进行设置。

二．InitGlut(argc, argv);

PhysX是OpenGL上开发的，所以在初始化PhysX实例之前，必须建立一个OpenGL的框架。

①. glutInit(&argc, argv) 用来初始化GLUT，并且处理任意的命令行变量

②. glutInitWindowSize(int width, int size) 指定了窗口以像素为单位的尺寸

③. glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE | GLUT\_DEPTH) 建立一个带有双缓存、RGB颜色模型和很大缓存的窗口

④. glutCreateWindow（char\* string） 创建一个具有OpenGL创建的窗口，string为该窗口的窗口名

⑤. glutSetWindow（）

⑥. glutDisplayFunc(RenderCallback) 渲染

1ProcessCameraKeys();

2

3 SetupCamera();

4

5 if (gScene && !bPause)

6

7 {

8

9 GetPhysicsResults();

ProcessInputs();根据选择的对象，给该对象施加前后、上下、左右不同方向的力，然后调用对象的方法addForce，产生不同的物理效果

StartPhysics();

}

// Display scene

RenderActors(bShadows);

调用函数DrawActor(NxActor\* actor)将场景中的物体渲染出来，实在是在DrawActor(NxActor\* actor)函数中根据物体形状调用不同形状的绘画函数将物体渲染出来的。在渲染的过程中，利用显示列表绘制不同形状的物体。在PhysX中，物体形状分为以下几种：NX\_SHAPE\_PLANE(面板状), NX\_SHAPE\_BOX(盒子状), NX\_SHAPE\_ SPHERE(球形状), NX\_SHAPE\_CAPSULE(胶囊状), NX\_SHAPE\_CONVEX(凸多边形状), NX\_SHAPE\_MESH(网状状)。

当bShadows为true时，渲染物体的阴影；为false时就不绘制

DrawForce(box, gForceVec, NxVec3(1,1,0));

将物体受力的受力方向渲染出来

⑦. glutReshapeFunc(ReshapeCallback)

设置窗口

⑧. glutIdleFunc(IdleCallback);

⑨. glutKeyboardFunc(KeyboardCallback);

⑩. glutKeyboardUpFunc(KeyboardUpCallback);

⑪. glutSpecialFunc(SpecialCallback);

在此，调用ResetNx（），重新渲染

⑫. glutMouseFunc(MouseCallback);

⑬. glutMotionFunc(MotionCallback);

⑭. MotionCallback(0,0);

三．InitNx() 因为我们需要初始化PhysX SDK实例，并且建立我们需要的场景；所以我们需要设置以下几个变量，并且将它们设置为全局变量

1 NxPhysicsSDK\*gPhysicsSDK = NULL;//PhysX SDK实例对象

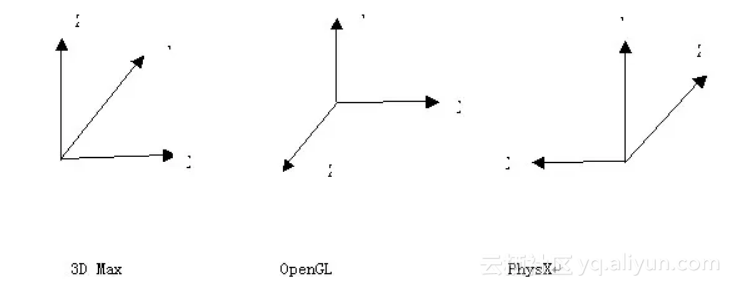
2

3NxScene\*gScene = NULL;//场景对象

4

5NxVec3 gDefaultGravity(0,-9.8,0);

\*\*\*注意：坐标系的方向指向，在PhysX、OpenGL以及3DMax都有一些不一样，当运行里面的demo的时候就可以体会到。它们的坐标系分别如下：



下面就在InitNx（）中开始初始化实例以及建立场景.

①. 实例化 physics SDK

gPhysicsSDK = NxCreatePhysicsSDK(NX\_PHYSICS\_SDK\_VERSION);

初始化完Physics SDK后，只是简单的一个实例。可以通过设置实例的物理参数来充实我们的模拟效果.

gPhysicsSDK->setParameter(NX\_SKIN\_WIDTH, 0.01);

②. 创建场景

1 NxSceneDesc sceneDesc; //场景表述表对象

2

3 sceneDesc.gravity = gDefaultGravity;

4

5 sceneDesc.broadPhase = NX\_BROADPHASE\_COHERENT;

6

7 sceneDesc.collisionDetection = true;

8

9 gScene = gPhysicsSDK->createScene(sceneDesc);

在PhysX中，不管是创建场景还是创建各个物体角色时，都是先通过各自对应的描述器（翻译的不是很准确）设置场景和各个物体的物理参数，用来模拟真实的世界环境和物体。建立好表述器后，通过函数createScene（NxSceneDesc）函数就可以建立需要的场景对象。

一般情况下，场景描述器的参数就是设置重力加速度sceneDesc.gravity，是否进行碰撞检测collisionDetection, true为进行，在PhysX SDK中描述器被广泛的应用. 描述器包括所有你创建物体的信息broadphase-coherent是三种碰撞检测中的一种。

1gPhysicsSDK->setParameter(NX\_SKIN\_WIDTH, 0.01);

当相互碰撞的物体的材质都很软的时候，在现实中就会发现当发生碰撞的时候物体之间就会相互嵌入一部分，在这里我们就可以利用物理参数NX\_SKIN\_WIDTH，它的默认值为0.05m，该值越大，嵌入的就越多

同时，我们可以对场景中的所有物体创建材质。创建的材质定义了碰撞和物体材料的物理属性。比如反弹系数、静摩擦力、滑动摩擦力等。

1 // Create the default material通过材质索引创建一个材质的对象

2

3 NxMaterial\* defaultMaterial = gScene->getMaterialFromIndex(0);

4

5 defaultMaterial->setRestitution(0.5);

6

7 defaultMaterial->setStaticFriction(0.5);

8

9 defaultMaterial->setDynamicFriction(0.5);

10

11

12

13创建物体，以box为例

14

15NxActor\* box = CreateBox(NxVec3(5,1,0));

16

17NxActor\* CreateBox(const NxVec3& pos)

18

19{

20

21 // Add a single-shape actor to the scene

22

23 NxActorDesc actorDesc;

24

25 NxBodyDesc bodyDesc;

26

27

28

29 // The actor has one shape, a box

30

31 NxBoxShapeDesc boxDesc;

32

33 boxDesc.dimensions.set(0.5,1,0.5);

34

35 actorDesc.shapes.pushBack(&boxDesc);

36

37

38

39 actorDesc.body = &bodyDesc;

40

41 actorDesc.density = 10;

42

43 actorDesc.globalPose.t = pos;

44

45 return gScene->createActor(actorDesc);

46

47}

我们创建一个角色参与者box，它的类型为NxActor\*。建立该对象的时候需要设置它的描述器，然后利用函数createActor(NxActorDesc actorDesc)将该对象加入场景中。每一个对象又有和自己形状相对应的描述器。利用它设置对象的物理参数。boxDesc该描述器描述了该盒子的长、宽、高分别为0.5，初始化的位置以及该盒子的密度。

③. 创建完所有的物体对象时，调用UpdateTime()得到从上一帧渲染到现在经过的时间

④. 当创建的场景成功，利用函数StartPhysics()开始它的第一帧模拟。

1 void StartPhysics()

2

3{

4

5 // Update the time step

6

7 NxReal deltaTime = UpdateTime();

8

9

10

11 // Start collision and dynamics for delta time since the last frame

12

13 gScene->simulate(deltaTime);

14

15 gScene->flushStream();

16

17}

18

19simulate(deltaTime)是PhysX 解决物理学的关键

20

21 flushStream()对时间步进行仿真

四．glutMainLoop()

程序将一直停留在glutMainLoop()中，直到用户自己结束。当场景一旦被渲染后，在每次设置下一场景时，RenderCallback()回调函数将被调用

五．ReleaseNx()

删除场景中所有的物体对象以及场景本身