Reegan Mitchell

PhysX Documentation

The PhysXSystem Class:

Note: the variable name used to describe the use of the PhysXSystem class in this document will be “pxPhysics”.

Initialising:

To use the PhysXSystem class, you must have a persistent pointer in the game class (See TutorialGame.h) like this:

PhysXSystem* pxPhysics;

At some point, the PhysXSystem must be initialised, best place is in the class constructor (See TutorialGame.h) like this:

pxPhysics = new PhysXSystem();

In the game Update method, the PhysXSystem Update function must be called like this:

pxPhysics->Update(dt);

Closing/Deleting:

First, clear the PhysXSystem using pxPhysics->Clear().
This should free all memory used by the PhysXSystem.
Then, if needed use: delete pxPhysics;

Changing level/Resetting the scene:

To Reset the scene for changing level, first use pxPhysics->Clear().
Then create a new instance using: pxPhysics = new PhysXSystem();

How to create an object:

Use the Add___ToWorld function as before (or create your own), inside SetPhysicsObject() use one of the functions below.

Reegan Mitchell

Functions:

CreateCubeActor(const Maths::Vector3& position, const Maths::Vector3& size, float mass, float friction = 0.5f, float restitution = 0.1f);

Description: Creates a PhysX actor with the shape of a Cuboid, and adds it to the physics scene. Objects

Returns: PxActor*

How to use: Use as a parameter for SetPhysicsObject() when creating an object.

CreateCapsuleActor(const Maths::Vector3& position,float halfHeight,float radius, float mass, float friction = 0.5f, float restitution = 0.1f);

Description: Creates a PhysX actor with the shape of a Capsule, and adds it to the physics scene.

Returns: PxActor*

How to use: Use as a parameter for SetPhysicsObject() when creating an object.

CreateSphereActor(const Maths::Vector3& position, float radius, float mass, float friction = 0.5f, float restitution = 0.1f);

Description: Creates a PhysX actor with the shape of a Sphere, and adds it to the physics scene.

Returns: PxActor*

How to use: Use as a parameter for SetPhysicsObject() when creating an object.

CreateWall(const Maths::Vector3& position, const Maths::Vector3& size, float friction = 0.5f, float restitution = 0.1f);

Description: Creates a PhysX actor with the shape of a Cuboid, and adds it to the physics scene.

Returns: PxActor*

How to use: Use as a parameter for SetPhysicsObject() when creating an object.

CreateMeshActor(const Maths::Vector3& position, const Maths::Vector3& size, OGLMesh* mesh, float friction = 0.5f, float restitution = 0.1f);

Reegan Mitchell

Description: Creates a PhysX actor with the shape of whichever mesh is sent as a parameter, and adds it to the physics scene.

Returns: PxActor*

How to use: Use as a parameter for SetPhysicsObject() when creating an object.

AddActor(bool isDynamic, const Maths::Vector3& size, const Maths::Vector3& position, float mass, std::string shape, float friction = 0.5f, float restitution = 0.1f);

Description: Creates a PhysX actor of whichever shape is sent as a string parameter, and adds it to the physics scene. Can be static or dynamic, does not work for a triangle mesh.

Returns: PxActor*

How to use: Don't, use one of the above functions. Unless an unlikely combination of parameters such as a static capsule is required, but even then it may be easier to message me on discord so that I can write a better function for your use case.

AddActor(physx::PxActor* actor);

Description: Simply adds a specified PhysXActor to the physics scene

Returns: N/A

How to use: Can be used if you have manually created a custom PxActor and just want to add it to the scene. Unlikely to be needed in most cases.

Extra Notes:

For objects that are dynamic (take a mass value), setting the mass to 0 will have them fall through the world