



Práctica 5: Sólidos Rígidos

En esta práctica vamos a usar *PhysX* para realizar la simulación y ver cómo se comporta el sistema profesional de física de *Nvidia*. Con su uso veréis que tiene una estructura muy parecida a la que hemos implementado en nuestros sistemas anteriores.

Para empezar, tenemos dos tipos de sólidos rígidos, los dinámicos y los estáticos. Los estáticos están pensados para no tener velocidad/aceleración (equivalente a un sólido de masa infinita). Los dinámicos son aquellos que sí que podrán tener velocidad y aceleración. Las clases para gestionarlos son: *PxRigidStatic* y *PxRigidDynamic*.

Para crear uno estático usaremos: *gPhysics->createRigidStatic()* con un *PxTransform* como parámetro.

Para crear uno dinámico llamaremos a: *gPhysics->createRigidDynamic()* también con un objeto *PxTransform* como parámetro.

En cualquier de los dos casos llamaremos a la función miembro *attachShape()* para unir el sólido rígido con un objeto *PxShape*, que podemos crear como hacíamos hasta ahora en nuestras prácticas. Tened en cuenta que, a parte de la forma, el objeto *PxShape* tiene información del material del que está hecho el sólido rígido, dicho material nos ajusta las propiedades de fricción entre superficies y el coeficiente de restitución cuando se produce un choque. Al inicio del archivo *main.cpp* se genera un material a modo de ejemplo, lo cual se hace con el siguiente método de la clase *PxPhysics*:

```
PxPhysics::createMaterial(staticFriction, dynamicFriction, restitution);
```

Tendremos que ir definiendo sus parámetros dinámicos (masa y tensor de inercia). Para ello haremos uso del método: *PxRigidBodyExt::updateMassAndInertia*. Con esta función especificamos su densidad, que junto con el volumen definido mediante las dimensiones del *PxShape*, permite calcular la masa y el tensor de inercia de forma automática. También se pueden generar de manera manual usando el método *new_solid->setMassSpaceInertiaTensor({lxx,lyy,lzz})*. En el Anexo de este documento se adjunta una tabla para el cálculo rápido de momentos de inercia para formas sencillas.

Una vez creados los sólidos rígidos hay que añadirlos a la escena mediante este método: *gScene->addActor()*. Pasando como parámetro el sólido rígido creado. Esto es necesario para que puedan interactuar entre sí.

Para que se puedan visualizar será necesario crear un *RenderItem*, igual que en prácticas anteriores, solo que en lugar de un *Transform* le pasaremos la instancia de sólido rígido que hemos creado.

En *PhysX* la gravedad se trata como algo separado. Se puede configurar de forma global para toda la escena poniendo un valor de *Vector3* en esta variable: *sceneDesc.gravity*. Esto debe hacerse en *initPhysics* antes de la llamada a *gScene = gPhysics->createScene(sceneDesc)*; De lo contrario no se tendrá en cuenta.

Actividad 1: Clase Sólido rígido

El primer paso es generar una nueva clase Sólido rígido. La constructora de esta clase se encargará del renderizado y el almacenamiento de atributos que sean relevantes para la gestión de sólidos rígidos (ejemplo: Tiempo de vida). Para los sólidos estáticos no hará falta crear una clase, podremos crearlos directamente en el main al inicio de nuestra ejecución.

Tarea

Genera, al menos, un sólido rígido con unos momentos de inercia calculados a mano siguiendo la tabla del Anexo. **(Opcional):** Aproxima los momentos de inercia de un sólido rígido con una forma distinta a las presentes en el Anexo con las fórmulas correspondientes a la forma presente en el Anexo que más se parezca. Luego observa como se comporta con respecto a un homólogo cuyos momentos de inercial calculas con el método *updateMassAndInertia*.

Actividad 2: Sistema de sólidos con gravedad

Una vez generada la clase sólido rígido, adaptaremos los generados de partículas que hicimos para que sean capaces de generar sólidos rígidos gestionados por *PhysX*. Ten en cuenta que todas las funcionalidades y clases que hicimos para partículas tienen que seguir funcionando. Crea nuevas clases si es necesario. Pondremos un máximo de elementos generados para no saturar el sistema. Haremos además que estos sólidos se vean afectados por la gravedad.

Tarea

Genera un sistema de sólidos rígidos que se vean afectados por la gravedad de la escena. Añade también, al menos, un suelo y algunos obstáculos estáticos como sólidos estáticos. Configura tu generador de sólidos de manera que varíen los momentos de inercia de los diferentes sólidos. Configura también las propiedades del material para variar la elasticidad de los choques y el coeficiente de rozamiento.

Actividad 2. Fuerzas y torques aplicados a sólidos.

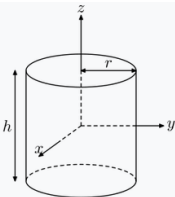
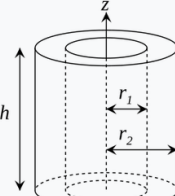
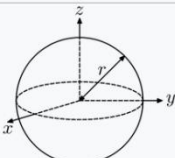
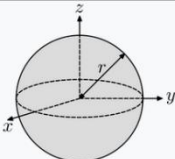
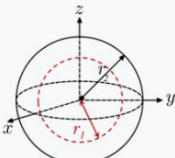
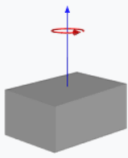
El siguiente paso es adaptar los generadores de fuerzas para que las fuerzas sean también aplicables sobre sólidos rígidos. Puedes crear nuevas clases si es necesario. Para aplicar una fuerza sobre un sólido rígido la clase *PxRigidDynamic* dispone del método *addForce()*, a la que se pasa un *Vector3* con la fuerza (dirección e intensidad). También podéis usar el método *addTorque()* para aplicar un par de fuerzas a un objeto, modificando el momento angular del sólido. Esto se deja como tarea optativa.

Tarea

Crea un generador de fuerzas que aplique fuerza a los sólidos de la misma manera que en el caso de las partículas. Tienes la documentación necesaria en este [enlace](#). Al menos una de las fuerzas debe de poder aplicarse en un punto diferente al centro de masas para la generación de pares.

Implementa una instancia de un generador de fuerzas concreto (por ejemplo, un viento o una explosión) de manera que puedas activarlo o desactivarlo según quieras.

Anexo: Lista de momentos de inercia directos

<p>Solid cylinder of radius r, height h and mass m.</p> <p>This is a special case of the thick-walled cylindrical tube, with $r_1 = 0$.</p>		$I_z = \frac{1}{2}mr^2$ [1] $I_x = I_y = \frac{1}{12}m(3r^2 + h^2)$
<p>Thick-walled cylindrical tube with open ends, of inner radius r_1, outer radius r_2, length h and mass m.</p>		$I_z = \frac{1}{2}m(r_2^2 + r_1^2) = mr_2^2 \left(1 - t + \frac{t^2}{2}\right)$ [1] [2] <p>where $t = (r_2 - r_1)/r_2$ is a normalized thickness ratio;</p> $I_x = I_y = \frac{1}{12}m(3(r_2^2 + r_1^2) + h^2)$ [citation needed] <p>The above formula is for the xy plane passing through the center of mass, which coincides with the geometric center of the cylinder. If the xy plane is at the base of the cylinder, i.e. offset by $d = \frac{h}{2}$, then by the parallel axis theorem the following formula applies:</p> $I_x = I_y = \frac{1}{12}m(3(r_2^2 + r_1^2) + 4h^2)$
<p>With a density of ρ and the same geometry</p>		$I_z = \frac{\pi\rho h}{2}(r_2^4 - r_1^4)$ $I_x = I_y = \frac{\pi\rho h}{12}(3(r_2^4 - r_1^4) + h^2(r_2^2 - r_1^2))$
<p>Hollow sphere of radius r and mass m.</p>		$I = \frac{2}{3}mr^2$ [1]
<p>Solid sphere (ball) of radius r and mass m.</p>		$I = \frac{2}{5}mr^2$ [1]
<p>Sphere (shell) of radius r_2 and mass m, with centered spherical cavity of radius r_1.</p> <p>When the cavity radius $r_1 = 0$, the object is a solid ball (above).</p> <p>When $r_1 = r_2$, $\frac{r_2^5 - r_1^5}{r_2^3 - r_1^3} = \frac{5}{3}r_2^2$, and the object is a hollow sphere.</p>		$I = \frac{2}{5}m \cdot \frac{r_2^5 - r_1^5}{r_2^3 - r_1^3}$ [1]
<p>Solid rectangular cuboid of height h, width w, and depth d, and mass m.</p> [7] <p>For a similarly oriented cube with sides of length s, $I_{CM} = \frac{1}{6}ms^2$</p>		$I_h = \frac{1}{12}m(w^2 + d^2)$ $I_w = \frac{1}{12}m(d^2 + h^2)$ $I_d = \frac{1}{12}m(w^2 + h^2)$

Fuente: https://en.wikipedia.org/wiki/List_of_moments_of_inertia