Práctica 2 – Sólidos Articulados

(Asignación 2 de Marzo; Entrega 23 de Marzo a las 23:59)

¿Cómo entregar la práctica?

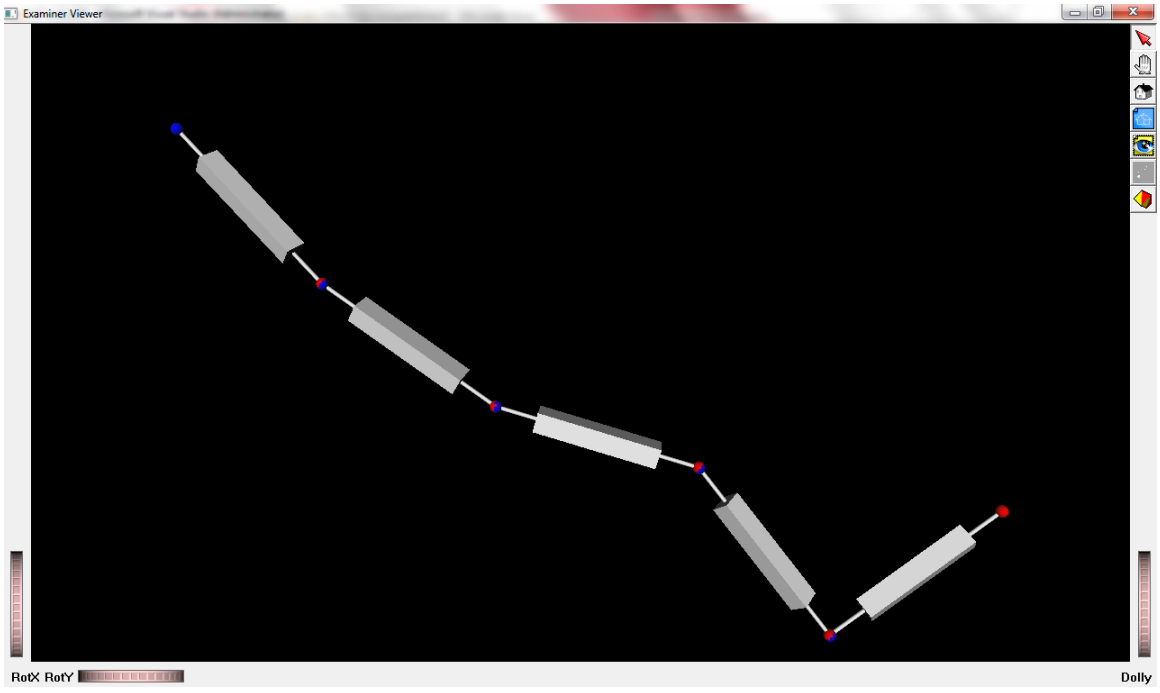
Enviar una copia del fichero Exercise.cpp por email a miguel.otaduy@urjc.es, antes del 23 de Marzo a las 23:59.

Especificaciones Generales:

> ATENCIÓN: La práctica tiene partes obligatorias y partes sólo para los alumnos con la asignatura completa.

Se proporciona un proyecto sobre MS VisualStudio 2010 que se encarga de inicializar y configurar la simulación, ejecutar el bucle de simulación, y visualizar los resultados utilizando Coin3D. Previamente, se han de instalar Coin3D (versión 3) y SoWin. La práctica también se puede realizar en otras plataformas (Linux, Mac…), ya que el fichero Exercise.cpp no hace referencia a SoWin.

El ejercicio consiste en el cálculo de un paso de simulación de varios sólidos rígidos fijados en una cadena. La rutina para la ejecución del paso de simulación está incluida en el fichero Exercise.cpp. Todo nuevo código ha de ser añadido en el fichero Exercise.cpp. No se han de enviar más ficheros, **sólo** Exercise.cpp. La práctica está preparada para que no necesite librerías adicionales. Todo el código adicional que se precise ha de estar incluido en el fichero Exercise.cpp. La práctica se evaluará intercambiando el fichero Exercise.cpp, compilando y ejecutando, con lo cual no compilará si se precisan otros ficheros.



Explicación de la Demo:

La demo carga varios sólidos rígidos, unidos en una cadena. El primero de los sólidos tiene uno de sus extremos anclado a la posición (0,0,0), y el resto de los sólidos están unidos entre sí por articulaciones esféricas. Cada sólido tiene un extremo de color azul y un extremo de color rojo, y su extremo azul está unido al extremo rojo del sólido anterior. La gravedad ha de apuntar en la dirección –Y.

Los sólidos colisionan contra el suelo, que está situado en la posición en el eje Y dada por la variable `floor`.

Al iniciar la demo, se encuentra activa la opción de movimiento de cámara en la ventana principal de Coin (icono 'mano'). Para interactuar con la demo, se ha de seleccionar la opción de interacción de Coin (icono 'flecha'), y se ha de pinchar sobre la ventana. Entonces, la tecla 'c' permite modificar el método utilizado para unir los sólidos: modo 'muelle' o modo 'restricción'. Por defecto, la demo se inicia en modo 'restricción'. La tecla 'f' activa el flag 'collisions', que servirá para activar/desactivar la respuesta a colisiones. La tecla 's' permite pausar/continuar la simulación.

El número de sólidos rígidos en el sólido articulado es variable, y puede fijarse de dos maneras. Modificando la inicialización de `Scene::nbodies` en el constructor de `Scene`, o pasándolo como argumento al ejecutable.

Clase `RigidBody`:

Para facilitar el paso de información, las variables de los sólidos rígidos se almacenan en objetos de la clase `RigidBody`. Es una clase que sirve, básicamente, para almacenar información, no para hacer cálculos. Se proporcionan métodos para acceder a las variables internas del sólido (masa, inercia, inercia inversa, posición, rotación, velocidad lineal, velocidad angular), y métodos para fijar las variables dinámicas como resultado de la integración (posición, rotación, velocidad lineal, velocidad angular).

La masa e inercia del sólido rígido se fijan en la inicialización de la clase `Scene`. Para la aplicación de la fuerza de gravedad, en el fichero Exercise.cpp se ha definido la constante de aceleración gravitatoria `g`. Una vez aplicada la rotación a un sólido rígido, se pueden actualizar su tensor de inercia y su inversa simplemente llamando al método `RigidBody::UpdateInertia()`

El sólido rígido ha de recibir fuerza y par de amortiguamiento (damping). El valor de amortiguamiento `damping` está especificado en el fichero Exercise.cpp. Se ha de aplicar fuerza de amortiguamiento en la traslación, $\mathbf{F} = -\mathbf{d} * \mathbf{v}$, y par de amortiguamiento en la rotación, $\mathbf{T} = -\mathbf{d} * \boldsymbol{\omega}$. En ambos casos se utilizará la misma constante de amortiguamiento.

Los puntos azul y rojo de cada sólido rígido se encuentran en posiciones fijas en el sistema de referencia local del sólido. Concretamente, la posición del punto azul en el sistema de referencia local es (-0.5, 0, 0), y la posición del punto azul es (0.5, 0, 0). Estos valores son claves para la correcta aplicación de fuerzas y pares, así como para la aplicación de las restricciones.

La clase `RigidBody` también proporciona métodos útiles para transformar puntos y vectores de coordenadas locales a globales, y para obtener la velocidad de un punto expresado en coordenadas locales.

En el fichero Exercise.cpp se proporcionan ejemplos de utilización de la clase `RigidBody` y sus métodos.

Por último, los sólidos rígidos con los que se ha de operar se encuentran almacenados en un vector de la STL. En el fichero Exercise.cpp también se proporcionan ejemplos de utilización de dicho tipo de vector.

Recursos Adicionales:

Se proporcionan clases de C++ para vectores y matrices en 3D (`Vector3` y `Matrix33`), que deberían ser útiles para la resolución de la dinámica del sólido rígido. Estas clases implementan múltiples operaciones útiles como suma, resta, multiplicación, multiplicación por escalares, multiplicación entre matriz y vector, cálculo de transpuesta, cálculo de inversa, selección de filas y columnas de una matriz, producto escalar y vectorial, etc.

También se proporcionan clases de C++ para vectores y matrices densos de tamaño arbitrario (`Vector` y `MatrixMN`). Se proporcionan métodos de multiplicación matriz-vector, multiplicación matriz-matriz, y traspuesta. Igualmente, se proporciona un método para resolución de sistemas lineales densos de tamaño arbitrario por eliminación de Gauss, y otro método para resolución de LCPs por Gauss-Seidel proyectado.

En el fichero Exercise.cpp se incluyen ejemplos de utilización de las clases de vectores y matrices y de sus métodos.

Tarea: Cálculo de la dinámica del sólido articulado.

INTERFAZ DE LA FUNCIÓN:

```
void AdvanceTimeStep(vector<RigidBody>& bodies, float step, bool constraint,
                     bool collisions, float floor)
```

*** En todos los casos, se utilizará el método de integración Euler simpléctico.**

REQUISITO 1 (Todos): Un sólido rígido anclado mediante un muelle.

Programar la actualización en cada paso de simulación de la posición, rotación, velocidad lineal y velocidad angular de un único sólido rígido.

El punto azul del sólido estará anclado mediante un muelle de longitud de reposo 0 a la posición (0,0,0). En el fichero Exercise.cpp se indica el valor por defecto de la rigidez del muelle (`k`).

El sólido colisionará contra el suelo, y la colisión también se modelará como un muelle (fuerza de penalty). Se utilizará la misma rigidez.

Este caso se corresponde con una simulación con `Scene::nbodies=1` y `constraint=false`. El vector `bodies` contiene un único elemento.

REQUISITO 2 (Todos): Un sólido rígido anclado mediante una restricción.

Modificar el caso anterior, sustituyendo el muelle por una articulación esférica implementada mediante restricciones fuertes, y la fuerza de penalty por una restricción fuerte unilateral. En la práctica, para la articulación se utilizarán tres restricciones, que hagan que el punto azul del sólido coincida con la posición (0,0,0).

Se ha de utilizar el método de restricciones fuertes basado en integración de velocidad y linealización de restricciones.

Sugerencia: Crear una clase Articulación y una clase Contacto para gestionar de manera modular la definición de las restricciones y sus jacobianas.

REQUISITO 3 (Asignatura completa): Cadena de sólidos unidos mediante muelles.

Programar la actualización de todos los sólidos incluidos en el vector bodies. La articulación esférica que une cada par de sólidos ha de estar modelada mediante un muelle de longitud de reposo 0. Nuevamente, la rigidez de los muelles está indicada con un valor por defecto en Exercise.cpp (`k`).

Los sólidos colisionarán contra el suelo, y las colisiones también se modelarán como muelles (fuerza de penalty). Se utilizará la misma rigidez.

Nota: Este problema se traduce simplemente en resolver el requisito 1 para todos los sólidos rígidos. Como la integración es explícita, no hay que resolver un problema global.

REQUISITO 4 (Asignatura completa): Cadena de sólidos unidos mediante restricciones.

Programar la actualización de todos los sólidos incluidos en el vector bodies. La articulación esférica que une cada par de sólidos ha de estar modelada mediante una restricción fuerte (Realmente tres, una por eje).

Los sólidos colisionarán contra el suelo, y las colisiones se modelarán como restricciones fuertes unilaterales.

Nota: Se recomienda formular primero las velocidades sin restricción, luego el sistema que permite calcular los multiplicadores de Lagrange, y finalmente utilizar dichos multiplicadores para calcular la velocidad restringida. Cuando no hay colisiones, se puede formular como un sistema lineal e utilizar el método de eliminación de Gauss. Cuando hay colisiones, se ha de resolver un LCP usando Gauss-Seidel proyectado.

Sugerencia: Crear vectores de articulaciones y contactos antes de crear el sistema a resolver.

Sugerencia: Formular la matriz M inversa, la matriz J, y utilizar la clase MatrixMN para formular el sistema a resolver. Esto es muy ineficiente, porque se hacen productos de matrices densas, pero simplificará la programación.

COMENTARIOS:

Con los parámetros por defecto, la simulación es estable tanto con restricciones como con muelles. Debido al amortiguamiento, el sólido articulado va perdiendo energía cinética.

En la simulación con muelles, se aprecia una separación visible en las articulaciones. En la simulación con articulaciones, dicha separación no debería ser apreciable.

Mediante la tecla ‘c’ se puede modificar dinámicamente el método de simulación, pero es muy probable que en el paso del modo ‘muelles’ al modo ‘restricciones’ se inestabilice la simulación.