

## **Práctica 1 – Parte 1: Masa-Muelle y ODEs**

(Asignación 20 de Septiembre; Entrega 6 de Octubre a las 23:59)

### **¿Cómo entregar la práctica?**

Enviar una copia del fichero Exercise.cpp por email a [miguel.otaduy@urjc.es](mailto:miguel.otaduy@urjc.es), antes del 6 de Octubre a las 23:59.

### **Especificaciones Generales:**

Se proporciona un proyecto sobre MS VisualStudio 2005 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 2) 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 en dos escenas diferentes, y con varios parámetros de entrada. Las rutinas para la ejecución de cada paso de simulación están incluidas 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.

Las escenas están simuladas en el plano XY, y la gravedad ha de aplicarse en la dirección -Y (hacia abajo). Las fuerzas de amortiguamiento se deberán programar como amortiguamiento en los nodos (proporcionales a su velocidad)

Los parámetros por defecto para la simulación son: masa de cada nodo  $m = 0.1$  Kg, rigidez de cada muelle  $k = 10.0$  N/m, amortiguamiento  $d = 0.01$  N\*s/m, paso de simulación  $dt = 0.003$  sec, método de integración 'Euler explícito'. Los parámetros se pueden modificar en el fichero Scene.cpp, o a través de la línea de comandos cuando se lanza el ejecutable. ¡Se recomienda probar las simulaciones con parámetros distintos para analizar el comportamiento!

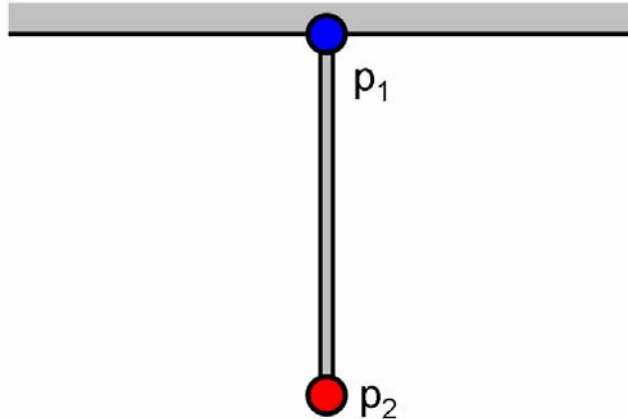
### **Recursos Adicionales:**

Se proporcionan clases de C++ para vectores y matrices en 2D, que deberían ser útiles para las escenas con simulación en 2D. Estas clases implementan operaciones básicas como suma, resta, multiplicación, producto escalar y vectorial, etc. Se encuentran en el fichero Vec2.h. Como se ha dicho antes, cualquier funcionalidad adicional ha de programarse en el fichero Exercise.cpp, ¡no se ha de modificar Vec2.h!

También se proporcionan clases (MatrixMN, Vector) y métodos (GaussElimination) para resolver sistemas de ecuaciones lineales por eliminación de Gauss. En el fichero Exercise.cpp se incluye un ejemplo de utilización de dicho código. Las clases y métodos se encuentran en el fichero LinSys.h. Esta funcionalidad sólo será necesaria para el problema 2.

**Problema 1: Nodo con masa colgando del techo.**

Este ejercicio consiste en la simulación de un muelle en 1 dimensión. Un extremo del muelle se encuentra fijado al techo, mientras que el otro cae por efecto de la gravedad. Los parámetros como masa ( $m$ ), rigidez ( $k$ ), amortiguamiento ( $d$ ), paso de simulación ( $dt$ ), y longitud inicial del muelle ( $L$ ) se pasan como argumentos.

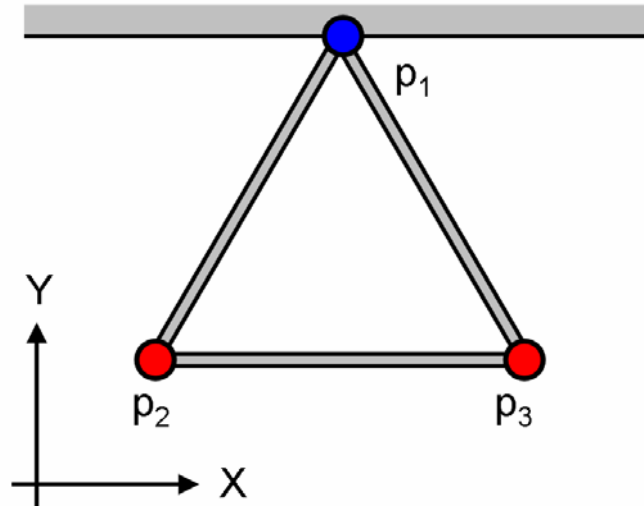
**INTERFAZ DE LA FUNCIÓN:**

```
void AdvanceTimeStep1(float k, float m, float d, float L, float dt,  
    int method, float p1, float v1, float& p2, float& v2);
```

**REQUISITOS:** Programar la actualización en cada paso de simulación de la posición  $p_2$  y la velocidad  $v_2$  del nodo inferior, para los siguientes métodos de integración: Euler Explícito ( $method = 1$ ), Euler Explícito Modificado ( $method = 2$ ), Punto Medio ( $method = 3$ ), Euler Semi-Implícito ( $method = 4$ ). Escribir la posición y velocidad actualizadas en  $p_2$  y  $v_2$ . Nota: la posición y velocidad anterior se pasan en los valores de entrada de  $p_2$  y  $v_2$ .

**Problema 2: Triángulo flexible colgando del techo.**

Este ejercicio consiste en simular un triángulo en 2D que está colgando del techo. Cada par de nodos está conectado por un muelle. Los parámetros de los nodos y los muelles (masa, etc.) son los mismos para todos, y se pasan como argumentos de forma similar al Problema 1.



INTERFAZ DE LA FUNCIÓN:

```
void AdvanceTimeStep2(float k, float m, float d, float L, float dt,  
    int method, const Vec2& p1, const Vec2& v1, Vec2& p2, Vec2& v2,  
    Vec2& p3, Vec2& v3);
```

REQUISITOS: Programar la actualización en cada paso de simulación de las posiciones y velocidades ( $p_2$ ,  $p_3$ ,  $v_2$ ,  $v_3$ ) de los dos nodos inferiores, para los métodos Euler Explícito Modificado ( $method = 2$ ) y Euler Semi-Implícito ( $method = 4$ ). Se proporciona código útil para programar el Euler Semi-Implícito (ver la sección de 'Recursos' en la primera página).