Práctica 1 – Parte b: Simulación de Telas 3D

(Asignación 16/02; Entrega 28/02 a las 23:59; Test presencial 02/03 en clase)

¿Cómo entregar la práctica?

Se habilitará una entrega por Aula Virtual. Se deberán subir los ficheros PhysicsManager.cs, MassSpring.cs, Node.cs y Spring.cs, los cuatro comprimidos en un único fichero.

No se han de enviar más ficheros que los cuatro listados. La práctica está preparada para que no necesite librerías adicionales. Todo el código adicional que se precise ha de estar incluido en los ficheros a entregar. La práctica se testeará intercambiando dichos ficheros, compilando y ejecutando, con lo cual no funcionará si se precisan otros ficheros.

Especificaciones Generales:

Se proporciona la escena de Unity Exercise1.unity, con un asset visual al que se la ha añadido un comportamiento masa-muelle. El código incluye un motor de simulación, un interfaz para un objeto masa-muelle, e implementaciones de nodos y muelles. La fijación de nodos se realiza mediante objetos de tipo Fixer. En la inicialización, se han de fijar todos los nodos incluidos dentro de un Fixer.

Se ha de programar el comportamiento del sistema masa-muelle siguiendo distintos métodos de integración. Concretamente, se han de programar el cálculo de fuerzas y jacobianas, y las rutinas que ejecutan los pasos de integración. En esta práctica, se han de programar fuerzas de amortiguamiento.

También se ha de programar la inicialización del modelo masa-muelle a partir de la malla de triángulos del asset visual asociado. Del mismo modo, se ha de programar la actualización del asset visual a partir del estado de la simulación.

Por último, los alumnos con la asignatura completa deberán implementar de manera separada muelles de tracción y muelles de flexión. Eso requiere identificar la topología de la malla de triángulos para inicializar muelles de flexión, así como programar propiedades de rigidez distintas para los dos tipos de muelles.

Programación del comportamiento masa-muelle y los métodos de integración

Esta parte de la práctica se puede heredar casi en su totalidad de la práctica 1a. La diferencia principal es la necesidad de incorporar fuerzas de amortiguamiento, siguiendo el modelo de Rayleigh para masa-muelle (amortiguamiento en los nodos y en los muelles).

Se han de programar los siguientes métodos:

```
private void PhysicsManager::stepSymplectic()
private void PhysicsManager::stepImplicit()
public void Node::GetForce(VectorXD force)
public void Node::GetForceJacobian(MatrixXD dFdx, MatrixXD dFdv)
public void Spring::GetForce(VectorXD force)
public void Spring::GetForceJacobian(MatrixXD dFdx, MatrixXD dFdv)
```

Se aconseja comprobar el comportamiento dinámico de la tela con distintos valores de los parámetros, con distintas inicializaciones de la posición y rotación inicial, con distintos fijadores…

Creación, inicialización, actualización y visualización

La creación del modelo masa-muelle comporta la programación del siguiente método:

```
public void MassSpring::Awake()
```

Para crear el modelo masa-muelle se debe acceder a la malla de triángulos del asset visual, concretamente a los vértices y aristas, para instanciar nodos y muelles asociados. Se proporciona código de soporte para el acceso a estas estructuras de datos, mediante:

```
Vector3[] vertices = GetComponent<MeshFilter>().mesh.vertices
int[] triangles = GetComponent<MeshFilter>().mesh.triangles
```

La inicialización requiere la asignación inicial de parámetros a los distintos elementos del modelo masa-muelle, así como la fijación de nodos. Se han de programar los siguientes métodos:

```
public void MassSpring::Initialize(int ind, PhysicsManager m, List<Fixer> fixers)
public void Node::Initialize(int ind, float mass, float damping, PhysicsManager m)
public void Spring::Initialize(float stiffness, float damping, PhysicsManager m)
```

El objeto masa-muelle tiene unos parámetros de masas y rigidez. Se asume que el parámetro de masa es la masa total del objeto (a repartir entre los nodos), mientras que los parámetros de rigidez son directamente los valores de rigidez de los muelles.

La actualización y visualización comporta dos tareas: el paso de parámetros a los elementos del modelo masa-muelle cuando estos cambian durante la ejecución, y la aplicación del estado de simulación al asset visual. Se han de programar los siguientes métodos:

```
public void MassSpring::Update()
public void MassSpring::FixedUpdate()
```

Programación de muelles de flexión

Esta funcionalidad se ha de programar en la creación del modelo masa-muelle. También implica adaptar la inicialización de propiedades de los muelles, para considerar los dos tipos de muelles (de tracción y de flexión).

Cada triángulo de la malla tiene tres aristas. Entonces, si una arista está compartida por dos triángulos, aparecerá dos veces al recorrer todas las aristas. En este caso, se introduce un muelle de flexión que une los dos vértices opuestos a dicha arista.

Para identificar si hay aristas iguales, conviene programar objetos comparadores, y se pueden utilizar mapas o algoritmos de ordenación.

Extensiones (no evaluables)

Programación del solver mediante matrices dispersas. Para ello, se puede utilizar la clase SparseMatrix de la librería Math.Net.

https://numerics.mathdotnet.com/api/MathNet.Numerics.LinearAlgebra.Double/SparseMatrix.htm