




9 DE MAYO DE 2021

# SIMULACIÓN DE TELAS 3D

## ANIMACIÓN 3D

PABLO DE LA HOZ MENÉNDEZ  
DOBLE GRADO EN DISEÑO Y DESARROLLO DE VIDEOJUEGOS E INGENIERÍA DE COMPUTADORES  
URJC



## Índice

Introducción .....	1
Instrucciones .....	2
Parámetros: .....	2
<i>Mass Spring</i> : .....	2
<i>Fixer</i> : .....	2
<i>Cloth Collider</i> : .....	2
Uso: .....	3
Relación de los componentes implementados.....	4
Extra.....	5
<i>Complex Cloth Collider</i> : .....	5

## Introducción

El desarrollo de esta práctica consiste en dotar a un objeto *GameObject* de un comportamiento de simulación de tela deformable. En la carpeta del proyecto que se encuentra junto a este documento se pueden distinguir varios componentes ordenados en subcarpetas. Tres de ellas se corresponden con los distintos requisitos de carácter obligatorio de la práctica, y la cuarta contiene todas esas funcionalidades junto con los requisitos opcionales de la misma.

A su vez, la escena presente en el proyecto hace uso de los componentes finales de la práctica, es decir, aquellos presentes en la carpeta de requisitos opcionales y que incluyen todas las funcionalidades.

## Instrucciones

### Parámetros:

#### *Mass Spring:*

- *Paused*: si está activo, detiene la simulación bloqueando la tela en la posición en la que se encuentre.
- *Time Step*: paso de tiempo sobre el que se calcula la nueva velocidad y la nueva posición cuando se llama a *FixedUpdate()*.
- *Gravity*: vector que almacena la aceleración de la gravedad, que se usa para calcular el peso de los nodos.
- *Integration Method*: método de integración numérica que se usa para simular el comportamiento de la tela. Puede ser Euler explícito o simpléctico.
- *Traction Stiffness*: rigidez de los muelles de tracción de la tela. Se utiliza en el cálculo de la fuerza que hacen sobre los nodos y el amortiguamiento de los muelles de tracción.
- *Flexion Stiffness*: rigidez de los muelles de flexión de la tela. Se utiliza en el cálculo de la fuerza que hacen sobre los nodos y el amortiguamiento de los muelles de flexión.
- *Mass*: masa total de la tela. Se divide en partes iguales entre todos los nodos.
- *Substeps*: número de veces que se ejecutará *FixedUpdate()* cada vez que se llama.
- *Air Friction*: fricción del aire sobre los nodos, que amortigua su movimiento.
- *Spring Damping*: constante de amortiguamiento de los muelles.
- *Air Behaviour*: marca si el viento tiene una velocidad y una dirección constante o si varía con el tiempo según una fórmula matemática.
- *Air Velocity*: velocidad del viento. Solo se puede definir si el comportamiento del viento es constante.
- *Cloth Friction*: fricción de la tela con el viento.
- *Obstacles*: lista de referencias a todos los objetos *Cloth Collider* que generan colisiones con la tela.

#### *Fixer:*

- *Cloth*: referencia al *GameObject* que representa la tela simulada y que se quiere fijar.

#### *Cloth Collider:*

- *Object Geometry*: se debe indicar al script el tipo de geometría del objeto, pudiendo ser este un plano o una esfera.
- *Rigidity*: rigidez que se usará para calcular la fuerza de *penalty* al colisionar con la tela.

### Uso:

Para hacer uso de los scripts presentes en la práctica, se hará una escena nueva. En esta se ha de crear un *GameObject Plane* y asignarle el script *MassSpring* (siendo *MassSpringOptional* la versión final y la más completa). Posteriormente, se les darán los valores deseados a los distintos parámetros especificados en el apartado “Parámetros” de este documento. En caso de no saber qué valores asignar, los que vienen dados por defecto son válidos para una buena simulación.

En caso de querer fijar la tela a algún objeto de la escena, se le dará a dicho objeto el script *Fixer* (si se está usando *MassSpringOptional*, se dará *FixerOptional*). En el parámetro *Cloth* del *Fixer* se asignará la tela.

Para terminar, se pueden crear planos y esferas que generen colisiones con la tela. Se añadirán los nuevos *GameObjects* a la escena y se les adjudicará el script *ClothCollider*. En el parámetro *Object Geometry* se debe indicar si el objeto es un plano o una esfera y en *Rigidity* se dará el valor de la rigidez. Al igual que ocurría con los parámetros de la tela, la rigidez que viene dada por defecto es válida para una buena simulación.

## Relación de los componentes implementados

En primer lugar, se modificaron los scripts *Node* y *Spring*, puesto que ya no dependían de *GameObjects* y no necesitaban heredar de la clase *MonoBehaviour*. Se creó un constructor en ambas y se añadieron métodos para poder actualizar sus parámetros desde *MassSpring*. Además, se modificó el método *ComputeForces* de ambas clases para que recibiera los datos necesarios.

Posteriormente, en el componente *MassSpring* se crearon listas para almacenar estos nodos y muelles. Cada nodo se corresponde con un vértice de la malla, y cada arista entre nodos con un muelle de tracción. Los muelles de flexión se crean entre nodos que no comparten aristas. *MassSpring* también guarda una lista de los objetos que generarán colisiones con cada tela. A la hora de calcular las fuerzas que actúan sobre los nodos, se comprobará si alguno de ellos se encuentra colisionando con uno de los objetos y, en caso afirmativo, se aplicará la fuerza de *penalty* correspondiente.

Para terminar, el componente *Fixer* guarda una referencia a la tela que debe fijar, comprobando en el primer *frame* de la simulación los nodos de esta que se encuentran en su interior, fijándolos y guardándolos en una lista. Cada vez que el *Fixer* se mueve durante la simulación, recorre dicha lista y aplica sobre los nodos la misma transformación que ha sufrido.

## Extra

### *Complex Cloth Collider:*

Además de las funcionalidades propuestas para esta práctica, se intentó hacer un script que permitiese detectar colisiones con mallas geométricas complejas. Para ello, se planteó un algoritmo que, calculando la normal de cada triángulo de la malla sobre un punto, proyectase el vector que une dicho punto con cada nodo sobre la normal. De esta forma, si el nodo está detrás del triángulo, se obtiene la distancia entre el nodo y este con signo negativo. En caso de que esté delante, se obtendrá con signo positivo.

Durante la simulación, se recorre la lista de nodos de la tela, calculando la distancia a cada triángulo de la malla y guardando tanto la mayor como la menor de estas, además de la normal del triángulo a menor distancia. Si ambas tienen signo negativo, quiere decir que el nodo está en el interior de la malla y se deberá aplicar la fuerza de *penalty* en la dirección de la normal antes guardada.

Sin embargo, este algoritmo es demasiado pesado para simulaciones en tiempo real, ya que por cada nodo se debe calcular la distancia a cada triángulo en todo momento, y no se consiguió hacer que funcionase correctamente.