Práctica –Simulación de Telas 3D

Asignación 23 de Abril; Entrega 9 de Mayo a las 23:59

¿Cómo entregar la práctica?

Se habilitará una entrega por Aula Virtual. Se deberán subir el código, una pequeña memoria explicativa (en pdf), y un vídeo.

- El código ha de incluir los fuentes y los assets para ejecutar la simulación.
- La memoria ha de incluir instrucciones para la ejecución de la simulación y la selección de parámetros. Asimismo, ha de incluir una breve relación de los componentes implementados. No se ha de elaborar un documento extenso; como máximo una página con las instrucciones y una página de relación de componentes.
- El vídeo ha de mostrar la simulación en acción y el resultado de los componentes implementados. En algunos casos será conveniente comparar ejecuciones con/sin alguna funcionalidad.

Especificaciones Generales:

Se va a simular la deformación de una tela o superficie en 3D. Para ello, se cargará un objeto GameObject con una malla triangular asociada, y se dotará a ese objeto de un comportamiento de simulación de tela deformable. La práctica trata de replicar el trabajo de desarrollo de herramientas en un estudio de videojuegos. Un artista debería poder cargar un objeto de manera sencilla, dotarlo del comportamiento de superficie 3D deformable, especificar condiciones de contorno, etc.

La práctica tiene unos requisitos principales. Completar con calidad estos requisitos garantizará una nota alta en la práctica. Además, se proporciona un listado de funcionalidades adicionales que se podrían implementar.

Requisito 1: Componente MassSpringCloth para simulación de superficie elástica.

Se puede tomar como punto de partida el proyecto de Unity Malla3D, en el que se programa un componente PhysicsManager para simulación de mallas 3D de MassSpringCloth. PhysicsManager se puede renombrar como MassSpringCloth. Hay que tener en cuenta que, en PhysicsManager, los nodos y muelles se inicializan a partir de esferas y cilindros, lo cual se ha de sustituir.

El componente MassSpringCloth se ha de poder añadir a un objeto GameObject que contiene un componente Mesh, y dotará a ese Mesh de la simulación de superficie deformable. Se han de inicializar los nodos del MassSpringCloth utilizando los vértices de Mesh, y los muelles del MassSpringCloth utilizando las aristas de Mesh. Para el Requisito 1, es aceptable que las aristas duplicadas generen muelles duplicados.

El objeto MassSpringCloth se deberá poder integrar mediante los métodos de Euler explícito y Euler simpléctico.

Las propiedades de masa y rigidez se han de definir a nivel de objeto completo MassSpringCloth, y han de poder pasarse a los objetos nodo y muelle subyacentes en todo momento de la simulación.

En este Requisito 1, la fijación de nodos se puede implementar de manera arbitraria. Es decir, se pueden seleccionar como fijos unos ciertos nodos para facilitar la implementación.

La implementación más básica del Requisito 1 puede utilizar el GameObject con su transformación por defecto. Sin embargo, se valorará positivamente que el GameObject se pueda trasladar/rotar/escalar de forma arbitraria. Para lidiar con estas transformaciones, es importante tener en cuenta que la malla del GameObject viene definida en coordenadas locales y luego se aplica la transformación a las coordenadas del mundo. Sin embargo, la simulación de físicas se ha de ejecutar en coordenadas globales. Para que las transformaciones se apliquen correctamente, será necesario transformar entre coordenadas locales y globales en la inicialización, y entre globales y locales en cada update.

Para desarrollar el Requisito 1, se aconseja comenzar con un GameObjet de tipo Quad (solo tiene 2 triángulos), y después pasar al GameObject de tipo Plane.

Requisito 2: Componente Fixer para fijar nodos.

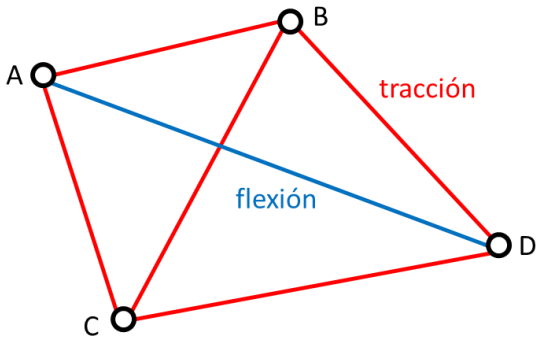
Se ha de crear un componente que permite fijar nodos del objeto MassSpringCloth. La implementación concreta de este requisito se deja a la elección del alumno, pero se ha de pensar en una funcionalidad sencilla para un artista.

Una opción es aprovechar los componentes Collider asociados a GameObject, p.ej., mediante la bounding box Bounds.

Al añadir objetos Fixer en la escena y asociarlos con el MassSpringCloth, la fijación de nodos ha de ser completamente automática y se ha de resolver por código.

Como parte opcional de este Requisito 2, se considera la posibilidad de mover el Fixer durante la simulación. Si se mueve, eso afectará a todos los nodos fijados, que se deberán mover con el propio Fixer, y arrastrar así al resto de la tela.

Requisito 3: Muelles de tracción y muelles de flexión.



Dada una malla con triángulos (en rojo) como en la figura, se ha de crear un muelle de tracción por cada arista natural de la malla (también en rojo), y un muelle de flexión (en

azul) entre los dos vértices opuestos de una arista compartida por dos triángulos. En contraposición a lo solicitado en el Requisito 1, solo puede haber un muelle de tracción por arista; no pueden estar duplicados.

Los muelles de tracción tendrán una rigidez notablemente mayor que los muelles de flexión, y esas rigideces se podrán establecer desde la escena, como parámetros editables del objeto MassSpringCloth.

Para resolver este Requisito 3, se ha de ejecutar una búsqueda de aristas duplicadas. Esto se puede hacer, por fuerza bruta, con un doble bucle que recorra todos los pares de triángulos. Si se encuentra una arista duplicada (p.ej., B-C en la figura), se ha de crear un único muelle de tracción. Del mismo modo, si se encuentra una arista duplicada, se ha de crear un muelle de flexión para los vértices opuestos A-D. Se admite como solución la resolución por fuerza bruta, mediante el doble bucle de coste cuadrático.

Se valorará la resolución de este problema mediante un algoritmo más eficiente. La clave es definir una estructura de datos Edge (VertexA, VertexB, VertexOther), donde VertexA y VertexB siempre están ordenados. VertexA es menor que VertexB según algún criterio (p.ej. índice en la malla). En el ejemplo de la figura, se crean 6 datos de tipo Edge: (A, B, C), (B, C, A), (A, C, B), (A, C, D), (A, D, C), (C, D, A). Estos datos se pueden ordenar, de nuevo de acuerdo a VertexA y VertexB. En ese caso, los datos (A, C, D) y (A, C, B) aparecerán de manera consecutiva, lo cual permite detectar aristas duplicadas simplemente recorriendo la lista ordenada. Este algoritmo tiene coste O(n log n), y es habitual en la creación de la estructura de datos HalfEdge o DCEL para mallas de triángulos.

Para desarrollar el Requisito 3, de nuevo se aconseja comenzar con un GameObjet de tipo Quad. Debería dar lugar un MassSpringCloth con 4 muelles de tracción y solo uno de flexión. Después, en el paso al GameObject Plane, se puede dar el siguiente paso intermedio: seleccionar solo los triángulos en las posiciones 0, 1, 2, 3, 39, 40, 41 y 42, lo cual da lugar a una malla de 2x2 quads (8 triángulos, 16 muelles de tracción, 8 muelles de flexión). Por último, se puede dar el salto al Plane completo (11x11 vértices, 2x10x10 triángulos, 320 muelles de tracción, 280 muelles de flexión).

Funcionalidades Adicionales:

Las funcionalidades adicionales se tendrán en cuenta a la hora de evaluar la práctica. Estas funcionalidades son necesarias para pasar de un notable a un sobresaliente.

- Simulación con substeps.
- Amortiguamiento. Ver tema 2.3 de Física de 1º. Se pueden implementar los modelos de amortiguamiento en nodos y muelles. La constante de amortiguamiento se puede hacer proporcional, respectivamente, a la masa y la rigidez.
- Fuerza de viento.
- Colisiones con objetos simples (p.ej. una esfera o un plano). Ver Fuerza de Penalty en el tema 2.5 de Física de 1º.
- Aspectos visuales y/o de interacción.
- Implementación de Prefabs para copiar un objeto MassSpringCloth y los Fixer asociados.