Práctica: Simulación de Sólidos Deformables

Asignación 8 de Mayo; Entrega 3 de Junio a las 23:59

¿Cómo entregar la práctica?

Se habilitará una entrega por Aula Virtual, y todo el contenido se entregará mediante un único fichero, que deberá incluir la siguiente documentación:

- El código del proyecto: ha de incluir los fuentes y los assets para ejecutar la simulación. Por favor, limpiad el resto de ficheros, por tamaño y para facilitar la compatibilidad entre versiones de Unity.
- Memoria del proyecto (máximo 2 páginas, en pdf), indicando 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.
- Vídeo de demostración de 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 dotará a la escena de animación de objetos volumétricos 3D con deformación elástica, ElasticSolid. Estos objetos producirán efectos secundarios en la escena de animación. Para el artista será sencillo seleccionar un GameObject y asociarle la capacidad de deformación 3D.

La deformación dinámica se implementará mediante el método masa-muelle. A cada GameObject deformable se le dotará de un componente ElasticSolid, que ejecutará la simulación dinámica y actualizará las posiciones de vértices del GameObject en cuestión.

El modelo masa-muelle se implementará sobre una malla de tetraedros. El objeto GameObject estará embebido en esta malla de tetraedros, y recibirá las posiciones 3D de vértices de él.

Requisito 1. Simulación y visualización de una malla de tetraedros (2 puntos)

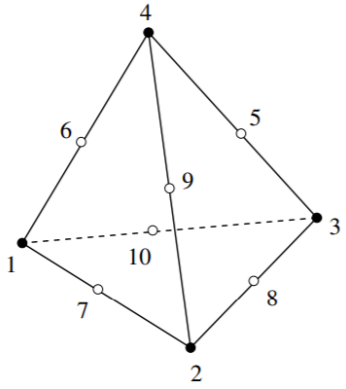
Se adaptará el componente de masa-muelle programado en prácticas anteriores para simular la deformación de una malla de tetraedros. Se ha de modelar un nodo por cada vértice de la malla de tetraedros, y un muelle por cada arista. En este primer requisito no es necesario eliminar muelles duplicados. En comparación con la práctica anterior solo son necesarios los muelles de tracción, no los de flexión.

Se aconseja comenzar la práctica con uno o dos tetraedros definidos “a fuego” en el código.

<u>- Ordenación de los nodos de un tetraedro:</u>

Para tareas posteriores de la práctica va a ser interesante poder calcular normales de las caras de un tetraedro, conociendo si las normales apuntan hacia fuera o hacia dentro. Para ello es importante asumir un convenio en la ordenación de los nodos de un tetraedro.

Se aconseja utilizar un ordenamiento como el de la figura (es el ordenamiento de la herramienta TetGen que se menciona en el punto 2 de la práctica, Figura 20 de la sección 5.2.4 del manual de TetGen). Los 4 nodos del tetraedro se ordenan de manera que el producto vectorial $(p_2 - p_1) \times (p_3 - p_1)$ apunta en la dirección del nodo p_4.

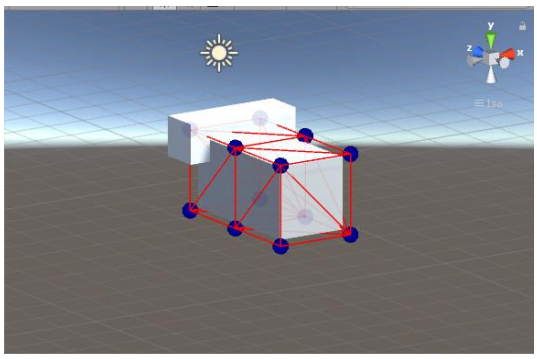


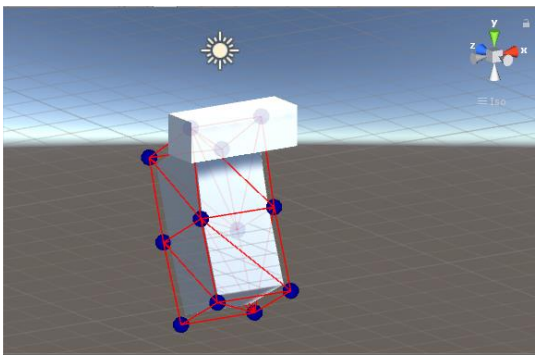
- Depuración visual:

La malla de tetraedros no tiene assets visuales asociados. Entonces, se han de crear estos assets para poder depurar el comportamiento de la simulación de manera visual. Existen varias opciones:

a) Instanciar desde código esferas y cilindros para representar los nodos y muelles.
b) Crear una malla de triángulos que represente la superficie de la malla de tetraedros. Pista: los triángulos de la superficie son aquellos que solo pertenecen a un tetraedro y no están duplicados.
c) (Opción recomendada) Utilizar Gizmos de Unity. Los Gizmos son elementos visuales que se pueden pintar directamente con una orden de código. Se pueden activar tanto en la ventana Scene como en la ventana Game de Unity.

En las siguientes imágenes, los nodos azules y los muelles rojos se han pintado utilizando la funcionalidad de Gizmos.





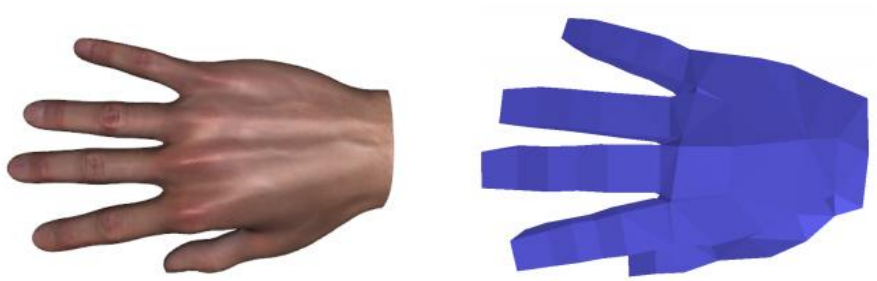
- Fijación de la malla de tetraedros:

Se ha de programar la funcionalidad de fixers de la práctica anterior.

Requisito 2. Carga de una malla de tetraedros por fichero (2 puntos)

La malla de tetraedros se va asociar a un asset visual de la escena de Unity, de manera que la malla de tetraedros rodee completamente al asset visual.

Para dotar de mayor calidad a la simulación, se va a diseñar una malla de tetraedros similar a la forma del asset visual. La malla de tetraedros ha de englobar completamente a los vértices de la malla de triángulos del asset visual, como se muestra por ejemplo en la siguiente figura.



- Generación de una malla superficial de baja poligonización.

Se va a desacoplar la complejidad del asset visual de la complejidad de la malla de tetraedros. Para ello, se ha de comenzar creando una malla de triángulos de baja poligonización que englobe al asset. Se recomienda cargar el .obj del asset visual en 3ds Max (u otra herramienta de edición), y allí se edita a mano una superficie de triángulos que englobe al asset visual. Se puede empezar con una superficie de triángulos de muy baja resolución, y posteriormente se puede editar dicha superficie si se necesita más detalle. Pero es más que aceptable que la superficie de triángulos sea de muy baja resolución.

La superficie de triángulos ha de cumplir varios requisitos.

1) Ha de ser cerrada y no puede tener intersecciones, para que se pueda generar una malla de tetraedros de manera correcta.
2) Ha de englobar completamente al asset visual, para que pueda funcionar el procedimiento de malla embebida descrito en el Requisito 3.

Se recomienda salvar la superficie de triángulos en formato .stl, ya que es un formato soportado por la herramienta TetGen.

Nota: Los sistemas de referencia de 3ds Max y Unity son diferentes. Para los ficheros .obj, 3ds Max permite salvar con la opción “Flip YZ axis”, y así el objeto se cargará en Unity exactamente como se ve en 3ds Max. Sin embargo, esta opción no está disponible para los ficheros .stl. Hay que tener cuidado que los ejes de coordenadas de las dos mallas de triángulos, la del asset visual y la de baja poligonización, sean los mismos.

- Generación de malla de tetraedros mediante TetGen.

Para generar una malla de tetraedros a partir de una superficie de triángulos, se va a utilizar el software TetGen. Se proporciona el código junto a la práctica, y también se puede descargar aquí: http://wias-berlin.de/software/index.jsp?id=TetGen&lang=1

TetGen implementa un método de generación de malla por tetraedralización de Delaunay, donde la superficie de triángulos aportada define el contorno de la tetraedralización.

TetGen no es una aplicación, sino que es un código C++ que se ha de compilar. La ejecución se hace por comandos, pasando los argumentos como texto.

Documentación TetGen http://wias-berlin.de/software/tetgen/1.5/doc/manual/manual.pdf

Dentro de la documentación, hay un tutorial básico en la sección 3.2.

- Importación de la malla de tetraedros en Unity.

TetGen genera como salida varios ficheros, entre ellos un fichero .node con la relación de nodos de la malla de tetraedros, y un fichero .ele con la relación de tetraedros. Estos ficheros se han de cargar en Unity. Como los ficheros no son estándar, la manera de cargar los datos es leyendo los datos de los ficheros mediante un parser.

Para acceder a un fichero de datos desde un script, se puede declarar un objeto de tipo TextAsset. Esto crea un cuadro en el Inspector de Unity, donde se puede arrastrar el nombre del fichero en cuestión. Esto sólo funciona con ficheros con extensión .txt, por lo que es necesario renombrar los ficheros de salida de TetGen. TextAsset.txt da acceso directamente a un string que contiene todo el texto del fichero.

De cara a diseñar el parser, hay que tener en cuenta el formato de los ficheros de Unity. Esto viene especificado en las secciones 5.2.1 y 5.2.4 de la documentación.

Nota: La numeración de nodos en el fichero .node empieza en 1.

Requisito 3. Malla de triángulos embebida (2 puntos)

Cada vértice del asset visual se va a mover siguiendo el movimiento de la malla de tetraedros. Este método de animación es equivalente a los métodos de skinning o blend shapes vistos en clase. La animación se define mediante un estado (en este caso, las físicas de la malla de tetraedros), y este estado se usa para gobernar las posiciones de los vértices de los assets visuales.

- Asociación de datos y comportamientos a un GameObject.

Se recomienda definir el asset visual a deformar como un único fichero .obj. Entonces, se importará este fichero .obj en la escena de Unity, y al GameObject correspondiente se le asocia el componente de masa-muelle volumétrico.

Cuando se importa un .obj en la escena de Unity, la malla de triángulos aparece como un elemento hijo del GameObject principal. Entonces, en el código no se puede acceder a la malla de triángulos simplemente como `this.GetComponent<MeshFilter>().mesh`. En su lugar, se puede acceder mediante `this.GetComponentInChildren<MeshFilter>().mesh`.

En el asset correspondiente al fichero .obj, es importante activar la opción “Read/Write Enabled”, o no se podrá acceder al Mesh en el código.

- Cálculo de posición por interpolación baricéntrica.

Dado un tetraedro con 4 nodos, cada uno de ellos con posición p_i, la posición de un punto interno p se puede calcular como una interpolación lineal de las posiciones de los nodos:

$$p = \sum_i w_i \, p_i$$

Los pesos w_i son las coordenadas baricéntricas del punto p dentro del tetraedro. Dichas coordenadas baricéntricas se pueden calcular una vez, en el momento de cargar las mallas. Para un punto p interno a un tetraedro, la coordenada baricéntrica w_i correpondiente a uno de los nodos p_i del tetraedro se calcula como:

$$w_i = \frac{V_i}{V}$$

Donde V es el volumen total del tetraedro, y V_i es el volumen del tetraedro formado por el punto p y los otros 3 nodos del tetraedro, excluyendo a p_i.

- Identificación del tetraedro contenedor.

En la inicialización, se ha de detectar qué tetraedro contiene a cada uno de los vértices del asset visual, para determinar los pesos de interpolación definidos arriba. Esta detección se realiza comparando la posición de cada vértice respecto a la orientación de las caras de cada tetraedro.

Es necesario definir la orientación de las caras de un tetraedro de manera consistente, todas hacia fuera o todas hacia dentro. Entonces, un punto p estará dentro del tetraedro si está dentro respecto a todas sus caras. Para poder calcular de manera correcta las orientaciones de las caras de los tetraedros, se ha de tener en cuenta el convenio de ordenación de los nodos mencionado en el Requisito 1 de la práctica.

El algoritmo de fuerza bruta de identificación de tetraedros contenedores recorre, por cada vértice, todos los tetraedros hasta encontrar al tetraedro contenedor. Este algoritmo tiene coste cuadrático. En una implementación eficiente, se puede evitar este coste cuadrático utilizando una estructura de aceleración como las utilizadas en detección de colisiones (p.ej., jerarquía de volúmenes contenedores o BVH).

Requisito 4. Parametrización de densidad y rigidez (2 puntos)

Hasta ahora, la masa y rigidez del modelo masa-muelle se han definido mediante valores por nodo o por muelle introducidos como parámetros desde la ventana de Unity. Esta estrategia hace que el comportamiento del objeto sea dependiente del tamaño y el mallado del objeto. Una alternativa que permite una edición más sencilla es definir los parámetros de masa y rigidez mediante densidades.

- Eliminación de aristas duplicadas:

Cada arista de una malla de tetraedros puede ser compartida por un número arbitrario de tetraedros. Para una correcta simulación, se creará sólo un muelle por cada arista. Se puede adaptar el algoritmo diseñado para eliminar duplicados en una superficie de triángulos (práctica de simulación de tela), aunque en aquel caso cada arista solo aparece dos veces, y en la malla de tetraedros puede aparecer múltiples veces.

- Densidad de masa:

Se tomará como dato de entrada la densidad de masa ρ del objeto. Dada esta densidad, se calculará la masa de cada tetraedro, y esta masa se repartirá a los nodos de dicho tetraedro, Así se consigue definir la masa de cada nodo.

Dados los 4 nodos de un tetraedro, $\{p_0, p_1, p_2, p_3\}$, el volumen del tetraedro se puede calcular como:

$$V = \frac{|(p_1 - p_0) \cdot (p_2 - p_0) \times (p_3 - p_0)|}{6}$$

Dada una densidad de masa ρ y el volumen del tetraedro V, la masa del tetraedro se calcula como:

$$m = \rho \cdot V$$

Esta masa se reparte entre los 4 nodos del tetraedro. Lo más sencillo es asignar simplemente ¼ de la masa total del tetraedro a cada nodo.

- Densidad de rigidez:

Para definir la rigidez mediante una densidad k, es necesario modificar la fórmula de la fuerza elástica de un muelle. En lugar de la fórmula habitual, se utilizará la siguiente fórmula, que precisa de un volumen V asociado al muelle:

$$F_a = -\frac{V}{L_0^2} \cdot k \cdot (L - L_0) \cdot \frac{x_a - x_b}{L}$$

Esta fórmula se basa en definir una deformación relativa, calcular una densidad de fuerza a partir de dicha deformación relativa, y multiplicar la densidad de fuerza por el volumen asociado al muelle. Para calcular el volumen asociado al muelle, se puede tomar el volumen de cada tetraedro y asignar $^1/_6$ a cada una de sus aristas.

- Extensión opcional: Parámetros variables por fichero.

Funcionalidades adicionales (2 puntos)

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, solo sobre los triángulos de la superficie de la malla de tetraedros.
- 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 ElasticSolid y los Fixer asociados.
- Asignación de parámetros de masa y rigidez heterogéneos, p.ej. por fichero. Si se quieren asignar parámetros variables en el volumen de la malla de tetraedros, una opción es que el artista asigne propiedades a cada nodo o tetraedro. Para ello, se puede cargar una nube de puntos en una herramienta de edición, p.ej. 3ds Max, y allí "pintar" las propiedades de cada punto. Esta información se guarda en un fichero que se ha de cargar en Unity.