Práctica 2 – Sistemas de Partículas

Duración: 1 Sesión

Objetivos:

- Definir e implementar un sistema de partículas y simular las interacciones de las mismas con fuerzas externas como el viento y la gravedad.
- Dotar a las partículas de ciertas propiedades como color, forma y tiempo de vida.

Entrega: 1 semana después de finalizar la práctica. Se debe entregar el código y una memoria/documento donde se expliquen los procedimientos realizados (los tipos de palmeras diseñados, cómo se han implementado éstos, y especialmente cuál es el modelo de viento que se ha utilizado), los resultados obtenidos y se analicen e interpreten estos resultados.

Evaluación: a través de la entrega del código y del documento de análisis.

La *Nit del Foc* Bidimensional

Se pretende simular un castillo de fuegos artificiales para recrear (¡parcialmente!, y sólo en 2 dimensiones) la *Nit del Foc* de las Fallas de Valencia. Para ello, se proporcionan, como material adicional de la práctica, cuatro ficheros con código de Processing. **Es importante entender el código y leer los comentarios insertados en él antes de comenzar**. Estos ficheros implementan un programa principal y tres conceptos que vemos a continuación:

Clase *Particle*
Conceptualiza una partícula.

Hay dos tipos de partículas: partículas de gran tamaño, que sirven para simular el ascenso de la carcasa sin explotar, y partículas de pequeño tamaño, que sirven para simular un punto de luz de la palmera, una vez la carcasa ya ha explotado en el aire. Ambos tipos están sometidos a la fuerza de la gravedad y a la del viento. La fuerza del viento debe ser configurable en intensidad y en dirección.

Cada partícula tiene su propio integrador (en esta práctica sólo usaremos Euler semi-implícito) que va calculando la velocidad y la posición de la partícula en todo instante.

Clase *Rocket*
Conceptualiza un cohete.

Un cohete contiene dos conjuntos de partículas:
· La carcasa, que es una única partícula que parte con una velocidad inicial principalmente vertical.
· El sistema de partículas, almacenado en un vector, que formará (cuando estalle la carcasa en el aire) la palmera.

La clase cohete permitirá implementar diferentes tipos de palmeras cuando se produzca la explosión. Según la dirección y el módulo de la velocidad de cada partícula del sistema, la palmera tendrá una forma u otra. También es posible asignar diferentes colores a las partículas, creando palmeras con diferentes tonos de color.

La dinámica del cohete presenta dos partes diferenciadas. En la primera parte se simula el ascenso y en la segunda parte, la explosión. Esto significa que, desde su lanzamiento, durante un tiempo se simulará el ascenso de la carcasa a través de una única partícula que parte con una gran velocidad inicial vertical. En un momento dado, la carcasa explotará. En ese momento la partícula única desaparece y se empieza a utilizar el vector de partículas, añadiéndole a la simulación un gran número de ellas, dotándolas de diferentes velocidades y direcciones. Aquí debéis jugar con vuestra imaginación para encontrar procedimientos que generen diferentes tipos de palmeras.

Clase *FireWorks*
Conceptualiza un castillo de fuegos artificiales. El castillo es un vector de cohetes. Esta es básicamente una clase contenedora que permite añadir cohetes a la simulación y actualizar cada uno de ellos.

Programa principal

Con él se manejará la interacción con el usuario. Por ejemplo, usando el ratón, lanzaremos los cohetes que formarán el castillo con una velocidad y dirección que dependerá de donde se pinche el ratón. Igualmente, debería existir una pequeña interfaz para definir la fuerza y la dirección del viento. La función *draw*() se debe encargar de visualizar el castillo y de escribir información adicional en la pantalla.

Realiza las siguientes tareas:

1- Completa el código proporcionado para implementar las siguientes funcionalidades:

1.1- Diseñar adecuadamente la física del sistema para que se simule la ascensión de la carcasa (en presencia de viento) y la explosión y caída de las partículas que constituyen cada palmera. La simulación del viento debe tener en cuenta que el viento es una fuerza, pero esta fuerza **debe depender de la velocidad relativa entre la partícula y la velocidad del viento**. **No se debe confundir nunca la velocidad del viento con la fuerza con la que ese viento empuja a una partícula**. Por ejemplo, una partícula que se mueve con la misma velocidad y sentido que el viento, no será empujada por este, mientras que una partícula que se mueva en sentido contrario al viento experimentará una fuerza de frenado que será proporcional a la suma de las magnitudes de las velocidades del viento y de la partícula. En la memoria se debe explicar y justificar el modelo de viento utilizado, que en cualquier caso será **lineal** (la dependencia entre la fuerza y la velocidad del viento deberá ser lineal).

1.2- Diseñar distintos tipos de palmeras. Es decir, distintos tipos de formas de palmeras (circulares, en racimo, con formas pintorescas como corazones, elipses...). Esto se logra configurando adecuadamente las velocidades iniciales del vector que forma el sistema de partículas. En la memoria se debe explicar qué tipos de palmeras se han diseñado y sus ecuaciones.

1.3- Regular el efecto del viento mediante una interfaz (visual o a través de teclado) para configurar la velocidad y la dirección del viento. Las teclas o acciones de ratón usadas deben ser explicadas con textos en la pantalla.

1.4- (Opcional) Mejorar la visualización utilizando otros sistemas de partículas para representar la traza de la subida de la carcasa, la traza de las partículas explotadas, la presencia de humo, etc. Esto permite visualizar palmeras que de otra forma no podrían verse correctamente.

2- Realiza un estudio del rendimiento del sistema haciendo una gráfica donde se

represente el tiempo medio que cuesta simular cada paso de simulación (T_s) frente al número de partículas (n) presentes en el sistema. Para hacerlo bien se deben tomar varias medidas y promediar, porque los tiempos pueden fluctuar significativamente de un instante a otro. Se debe analizar esta gráfica y explicar las conclusiones que se obtienen de ella.

3- Repite el estudio anterior, pero comparando esta vez el *frame rate* real de la aplicación frente al número de partículas (n) presentes en cada paso de dibujado. Para hacerlo bien se deben tomar varias medidas y promediar. ¿Existe alguna diferencia reseñable con la gráfica del apartado anterior? ¿Por qué?

Nota 1: es importante especificar el computador en el que se hacen las pruebas, ya que esta aplicación no está programada para funcionar en tiempo real, por lo que el rendimiento dependerá de la capacidad de cálculo del ordenador.

Nota 2: para todas las teclas usadas (o acciones de ratón), debe indicarse en la aplicación cuáles son estas teclas (y qué hacen) con un texto descriptivo en pantalla.

Nota 3: se valorará especialmente la claridad de la entrega y la correcta justificación de las respuestas.

Trabajo adicional (pero muy recomendable): añadir la posibilidad de realizar la integración numérica con el método de Euler implícito (también llamado Euler hacia atrás, Euler al revés o Euler inverso).

El método de Euler implícito se basa en evaluar la derivada al final del paso de simulación (en lugar de al principio del paso como hace Euler explícito).

La expresión general del método de Euler implícito para solucionar la ecuación diferencial

$$\frac{dy(x)}{dx} = f(x, y(x))$$

es:

$$y_{i+1} = y_i + f(x_{i+1}, y_{i+1})h$$

donde podemos ver que para calcular y_{i+1} hemos de evaluar la derivada (f) en el punto x_{i+1}, y_{i+1}. Se resalta en color rojo, porque realmente es la única diferencia con Euler explícito. Pero es una diferencia importante porque ahora ¡ y_{i+1} depende del propio y_{i+1} !

En el caso de las ecuaciones de Newton, la expresión sería:

$$\vec{v}_{i+1} = \vec{v}_i + \vec{a}(\vec{s}_{i+1}, \vec{v}_{i+1}, t_{i+1})h$$
$$\vec{s}_{i+1} = \vec{s}_i + \vec{v}_{i+1}h$$

En el caso del problema de esta práctica, la aceleración $\vec{a}$ de una partícula no depende directamente del tiempo, ni depende de la posición, por lo que la expresión se puede simplificar a:

$$\vec{v}_{i+1} = \vec{v}_i + \vec{a}(\vec{v}_{i+1})h$$
$$\vec{s}_{i+1} = \vec{s}_i + \vec{v}_{i+1}h$$

Como $\vec{a}$ depende de $\vec{v}_{i+1}$ y lo que buscamos precisamente es calcular $\vec{v}_{i+1}$, no podemos aplicar este método sin antes plantear **el sistema de ecuaciones correspondientes y resolverlo** para hallar una expresión para $\vec{v}_{i+1}$ que no dependa de la propia $\vec{v}_{i+1}$. En el caso de las ecuaciones de esta práctica, éstas son fáciles de resolver, aunque no siempre es así.