Práctica 1: ¡Nuestra primera partícula!

Crear una partícula

En esta primera parte de la práctica diseñaréis vuestra primera clase partícula en C++ y la dotaréis de lo necesario para ser representada (un objeto *RenderItem)*. El aspecto que tendrá la partícula es algo que podéis decidir vosotros. Dentro del esqueleto, como se ha comentado en clase, tenéis varias opciones para pintar objetos en pantalla. Cada uno de ellos tiene parámetros para poder personalizarlo como queráis.

Actividad 1a: Crear una partícula con velocidad constante

En esta actividad probaréis vuestra partícula. Para ello, haréis que actualice su posición implementando el método de integración *"integrate" usando el método de integración de Euler*. Haréis que dicha partícula tenga velocidad constante, definida como un Vector3.

Aquí tenéis un ejemplo de definición de partícula con velocidad. Como último apunte, recordad que tendréis que *deregistrar* el objeto *RenderItem* de la escena en el destructor de la partícula.

```
class Particle
{
public:
    Particle(Vector3 Pos, Vector3 Vel);
    ~Particle();

    void integrate(double t);

private:
    Vector3 vel;
    physx:: PxTransform pose; // A render item le pasaremos la direccion de este pose, para que se actualice automaticamente
    RenderItem* renderItem;
};
```

Una vez realizada dicha partícula, la instanciareis en el archivo principal (main.cpp), llamaréis al método *integrate* en la función de actualización y tendréis cuidado de liberar la memoria reservada a dicha partícula en la función *"cleanup"* si habéis hecho una reserva dinámica con el operador "new".

Al realizar esta práctica, se debe mostrar por pantalla una partícula moviéndose a velocidad constante por la pantalla.

Las leyes de Newton

Basándonos en el esqueleto proporcionado de la clase Partícula de la primera parte de la práctica, vamos a realizar el primer paso hacia tener una partícula que se mueva bajo la primera ley de Newton (ley de inercia). Para la implementación de los siguientes apartados seguid con la misma clase partícula generada en el apartado anterior pero añadiendo las funcionalidades siguientes:

Actividad 2: Partícula con aceleración

Para ello extenderemos la partícula de manera que se mueva con velocidad constante pero teniendo en cuenta que la única manera de cambiar la velocidad de la partícula es cambiando la aceleración.

Prueba tu partícula con distintos valores de la aceleración.

Actividad 3: Añadiendo damping

En este caso vamos a extender el comportamiento de la partícula para que tenga un movimiento más realista. Para ello incluiremos el *damping* en este movimiento. El *damping* es una herramienta matemática que se usa para corregir posibles errores numéricos introducidos durante la integración. Estos errores se van acumulando en el tiempo y a la larga pueden producir incrementos importantes de velocidad. El *damping* contrarresta esta aceleración artificial.

De momento la aceleración aplicada será constante y la especificaremos al construir un objeto de la clase Partícula. El *damping* será especificado también en dicho momento.

Tendremos, además, que modificar el método *integrate* de la partícula para que se actualicen tanto la velocidad como la posición de la partícula adecuadamente conforme a la aceleración y el valor de *damping*.

Actividad 4 (opcional) ¿Qué tal si te animas a implementar algún integrador más elaborado que Euler?

Prueba con Euler semi-implícito y/o con el integrador de Verlet que se emplea mucho en el entorno de los motores de físicas y de las simulaciones de movimiento.