### 1. ¿Qué es un Emisor de Partículas?

Un **emisor de partículas** es básicamente un "generador" de objetos simples que salen volando (partículas) en un videojuego. Imagina humo, fuego, nieve, o hasta enemigos que aparecen uno tras otro. Este emisor permite que esas partículas se creen automáticamente con ciertas propiedades como **posición, velocidad, color y tamaño**, y que respondan a cosas como **gravedad o colisiones**. Al usarlo, puedes simular todo esto sin tener que poner cada partícula manualmente.

**2. ¿Qué Debes Crear? 🛠️**

Para esta práctica, necesitas dos clases principales:

1. **Clase EmitterConfiguration**: Esta clase define la configuración del emisor, es decir, los ajustes que queremos usar.
2. **Clase Emitter**: Esta es la clase que genera las partículas según la configuración que le demos.

**3. ¿Cómo se Configura el Emisor? 🎛️**

La clase EmitterConfiguration tiene algunos datos que debemos definir:

* **Número de partículas**: cuántas partículas lanzará el emisor.
* **Periodo de emisión**: el intervalo de tiempo entre cada partícula (en milisegundos).
* **Tipo de partícula**: qué tipo de partícula queremos (puede ser un cubo, esfera, etc.), que puede ser de cualquier clase que herede de una clase base llamada Solid.

Este emisor, cuando esté listo, debería poder crear diferentes efectos visuales en el juego dependiendo de cómo lo configures.

**4. Implementando el Emisor 🌌**

La clase Emitter hereda de Solid (por lo que también tiene una posición, orientación, etc.), y tiene algunos métodos clave que necesitas entender:

* **Método Render**: este método dibuja todas las partículas en la pantalla. Lo que hace es básicamente "mandar a pintar" cada partícula generada.
* **Método Update**: aquí es donde se genera la magia. Este método:
  + Verifica si es el momento de crear una nueva partícula (usando el tiempo transcurrido).
  + Genera una partícula nueva y le cambia ciertas propiedades para que no todas sean iguales.
  + Agrega esta partícula al vector de partículas para que sea visible en el siguiente renderizado.
  + Repite este proceso hasta alcanzar el número máximo de partículas.

**5. ¿Cómo Saber Cuándo Crear una Nueva Partícula? ⏳**

Para eso usamos el tiempo, y aquí la cosa se pone un poquito técnica, pero básicamente es así:

* Usamos variables de tiempo, como initialMilliseconds (tiempo cuando arranca el emisor) y lastUpdateTime (última vez que se generó una partícula).
* En Update, verificamos si ha pasado el tiempo suficiente desde la última partícula generada para crear la siguiente.
* Todo este control de tiempo se hace con la librería **chrono** de C++.

**6. Clonar Partículas con el Patrón Prototype 🧬**

El **patrón Prototype** es un método para hacer copias de objetos sin tener que crearlos desde cero cada vez. En este caso:

* En la configuración (EmitterConfiguration) tenemos una referencia a la partícula base.
* Cuando queremos crear una nueva partícula, usamos el método Clone en esta partícula base. Así obtenemos una copia que luego podemos modificar.
* Para que esto funcione, cada clase de partícula (como Cube, Sphere, etc.) necesita tener un método Clone propio.

**7. Usando Números Aleatorios 🎲**

Podemos darle un toque random a las partículas para hacerlas más interesantes. Esto puede incluir cosas como:

* Cambiar el color de cada partícula.
* Variar su velocidad.
* Añadir rotación.

Para generar números aleatorios en C++, usamos rand y srand, que permiten generar diferentes valores cada vez que corres el programa.

**8. Entregables 📄**

Para entregar esta práctica debes tener:

* **Una memoria (documento PDF)**: donde explicas cómo implementaste la práctica (no todo el código, solo las partes clave).
* **Código fuente en un archivo comprimido** (ZIP o 7z).
* **Video de la ejecución del sistema de partículas** mostrando el resultado.