Proyecto final: Simulación física para videojuegos

Paula Morillas Alonso

Temática

En este entorno sencillo, el protagonista controla a una mosca que vive en la habitación de un niño y aprovecha los momentos del día en los que el niño no está para poder interactuar con las cosas que el chico deja en su habitación.

Objetivo

La intención es que el jugador pueda moverse por el entorno donde se encontrarán algunos objetos con los que podrá interactuar ya sea activando y desactivando generadores que provoquen cambios visibles en éstos o directamente aplicando fuerzas sobre ellos.

Especificaciones

En el entorno interactivo se incluyen:

1. Bola de cristal: Se quiere simular lo que sería agitar una bola de cristal llena de nieve. Para ello cuando el jugador interactúe con ella, se activarán fuerzas de explosión que moverán las partículas de su interior.

2.Vela: Se quiere simular una vela que muestre la generación y destrucción de partículas en forma de fuego.

3.Pecera: Se quiere simular una pecera donde se muestran dos peces con distintas masas que se acercan y se alejan mediante un muelle doble.

4.Bloques de construcción: Se quiere representar lo que serían unos juguetes de construcción. Éstos se simularán mediante sólidos-rígidos de distintos tamaños, formas, masas y tensor de inercia, y se les podrá aplicar una fuerza de explosión pulsando una tecla.

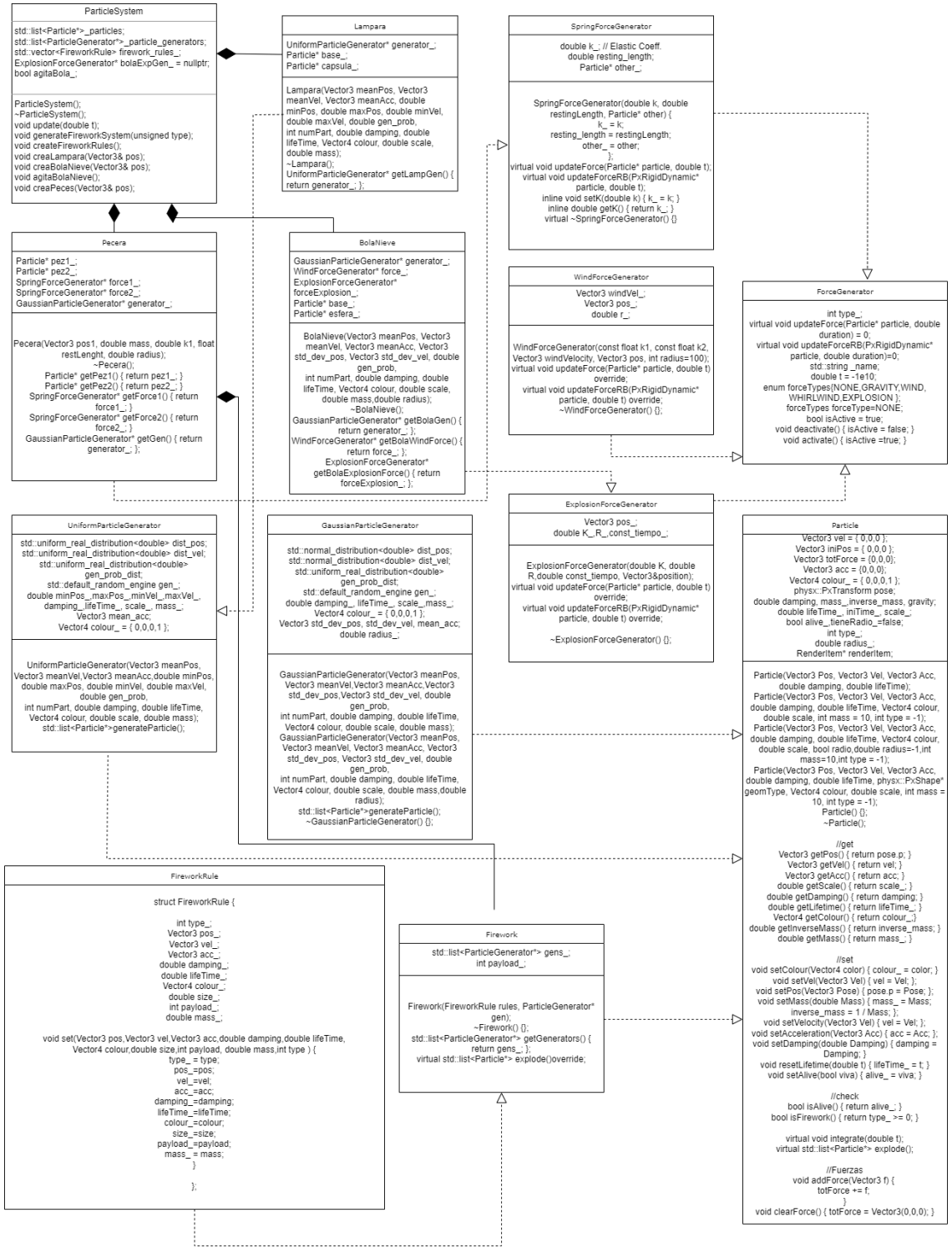
5. Fuegos artificiales: Se trata de dos tipos de fuegos artificiales, unos en forma de esfera y otros en forma de círculo.

Sistema de partículas

Para esta última práctica he utilizado el sistema de partículas que ya había implementado en las prácticas anteriores.

He utilizado cuatro generadores de partículas: uno para la pecera, otro para la bola de nieve, otro para el fuego de la vela y otro para los fireworks:

El diagrama UML tiene la siguiente forma:

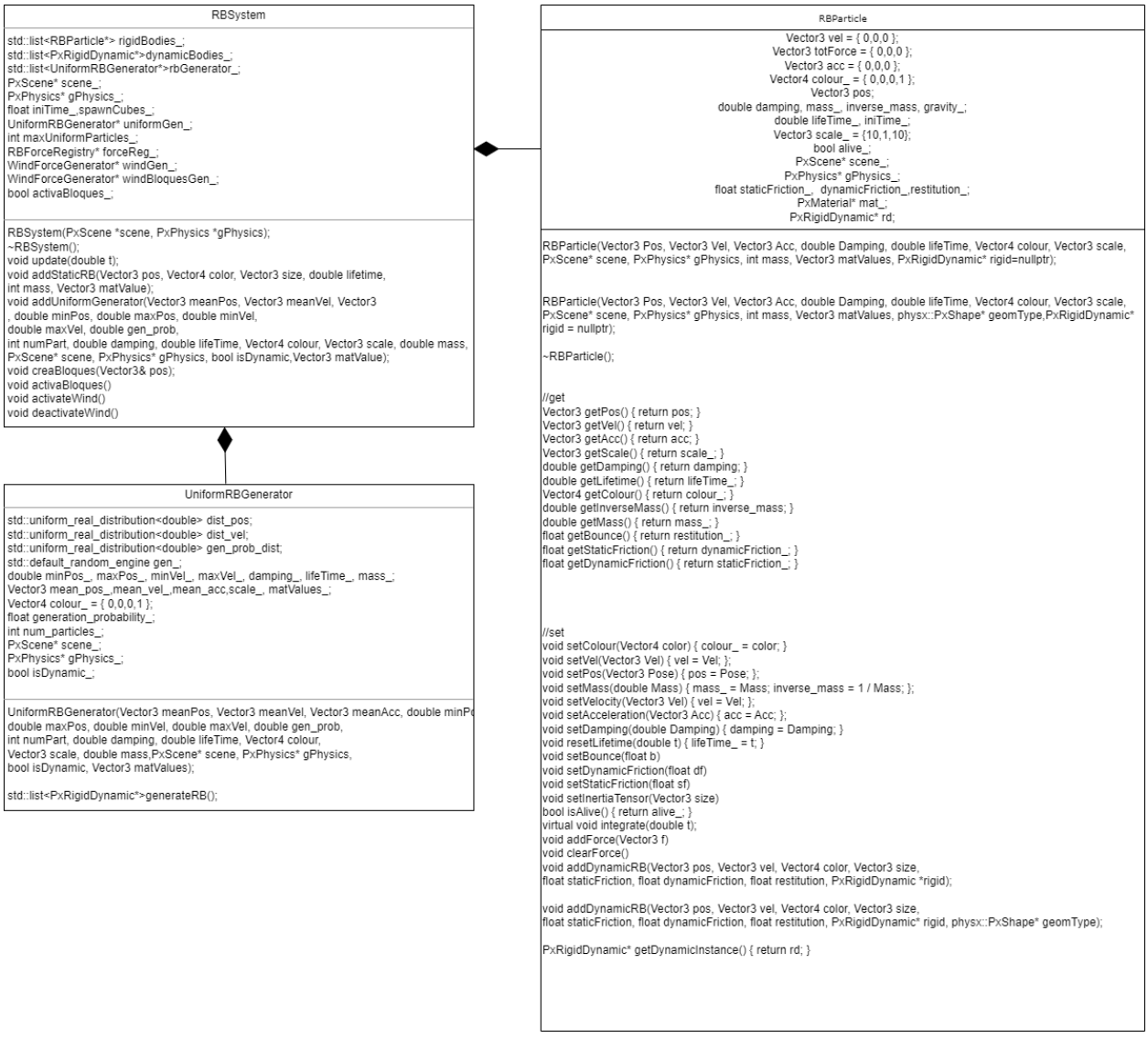


Sistema de sólido-rígido

Para esta última práctica he utilizado el sistema de sólidos-rígidos que ya había implementado en las prácticas anteriores.

En este caso he creado un método en el propio sistema para generar los bloques que simulan ser juguetes.

El diagrama UML tiene la siguiente forma:



Ecuaciones físicas y valores de los parámetros

Para los generadores he adaptado los siguientes parámetros de la siguiente forma:

-Agua de la pecera: Para este generador gaussiano he utilizado valores muy bajos para la velocidad y para el cambio de posición ya que al ser el agua de una pecera tiende a ser más estática.

```
generator_ = new GaussianParticleGenerator({ pos1.x,pos1.y - 3,pos1.z+1 }, { 1,1,1 }, { 0,0,0 }, { 0.8,0.8,0.8 }, {0.5,0.5,0.5},
    1, 4, 0.99, 4000, {0,0,1,1}, 0.3, mass, radius - 1);
```

-Vela: Para el generador uniforme he utilizado valores muy bajos comparados a algunos utilizados en las prácticas anteriores, pues se trata de un objeto pequeño y sin mucha dispersión (ya que la llama sólo se dirige hacia arriba).

```
Lampara* lamp = new Lampara(pos, { 0,2,0 }, { 0,-3,0 }, 0.2, 0.8, 0, 1, 0.6, 1, 0.99, 1000, { 0.9,0.3,0,1 }, 0.08, 1);
_particle_generators.push_back(lamp->getLampGen());
```

-Bola de nieve: Para el generador gaussiano de nuevo he utilizado valores muy bajos ya que también se trata de un objeto pequeño. Además he cambiado la clase de Particle para que las partículas se eliminen al salir de un determinado radio (en este caso el radio de la esfera “de cristal”).

```
BolaNieve* bola = new BolaNieve(pos, { .2,.2,.2 }, { .1,.2,.1 }, { 0.6,0.6,0.6 }, { 0.3,.1,.3 }, 0.8, 1, 0.99, 4000, {0.8,0.8,0.8,1},0.08,1,3);
auto gen = bola->getBolaGen();
_particle_generators.push_back(gen);
```

Para las fuerzas he adaptado los parámetros de la siguiente forma:

-Peces de la pecera: Para el muelle doble he puesto una masa mayor en uno de los peces y he utilizado una longitud de reposo pequeña con el objetivo de que no se salieran del espacio que representa la pecera.

```
Pecera* pecera_ = new Pecera(pos,1,3,7,8);

_particle_generators.push_back(pecera_->getGen());
```

-Viento de los bloques: Para el viento que afecta los bloques he puesto un mayor impulso en el eje Y con el objetivo de que los bloques se elevasen y cayesen permitiendo ver el efecto de los sólidos-rígidos implementados.

```
windBloquesGen_ = new WindForceGenerator(1, 0, Vector3(40, 80, 30), {pos.x,pos.y,pos.z }, 20);
windBloquesGen_->deactivate();
```

-Explosion de la bola de nieve: Para la explosión en este caso he utilizado valores más reducidos en la k y la R ya que se trata también de un objeto de menor tamaño que los de prácticas anteriores.

```
//force_ = new WindForceGenerator(-1, 0, { -8.5,-8,8 }, { meanPos.x,meanPos.y,meanPos.z }, 1);
forceExplosion_ = new ExplosionForceGenerator(20, 10, 10000, Vector3(meanPos.x,meanPos.y,meanPos.z));
```

Manual de usuario

El movimiento de la cámara se controla con el ratón y el desplazamiento se realiza pulsando las teclas WASD que moverán la cámara en la posición hacia la que está mirando.

Por otro lado se utilizan las siguientes teclas independientes:

-Tecla '1': Sirve para activar y desactivar la fuerza de explosión situada en la bola de nieve.

-Tecla '2': Sirve para activar y desactivar la fuerza de viento situada en los juguetes.

-Tecla '3': Sirve para generar fireworks en una posición fija.

Extra

Todos los objetos extra de la habitación (cama, mesa, estanterías, televisión y cuadros) han sido generados utilizando la clase Particle.