Colokao

Tabla de contenido

Temática y objetivo del juego

Es un juego para 2 jugadores que se divide en 3 fases:

- ***Construcción***: Los jugadores deberán (por turnos) construir una pequeña fortaleza encima de una plataforma designada con una lista aleatoria de bloques que el juego generará para ellos. Estos bloques pueden colocarse en la plataforma y encima de ella, a cualquiera altura. Son bloques a los que no le influye la física durante la colocación, por lo tanto, se puede colocar un bloque en el aire y este quedará ahí hasta el final de esta fase. La fase termina al una vez pase el tiempo máximo o se coloquen todas las piezas disponibles, haciendo que las piezas colocadas comiencen a ser afectadas por físicas, haciendo que caigan y choque entre ellas para formar una supuesta estructura. Se dejan unos segundos finales para observar el comportamiento de las piezas colocadas, y entonces comenzará el turno del siguiente jugador, o, en su defecto, la siguiente fase.
- ***Destrucción***: En esta fase los jugadores, también por turnos, deberán disparar con un cañón a la fortaleza del contrincante inmediatamente en frente de sí. Una vez se hayan disparado todos los cañones, se calculan cuántas piezas tiene cada jugador encima de su plataforma, ganando así los jugadores que hayan mantenido más piezas.
- ***Puntuaciones***: Fase inmediatamente posterior a acabar de disparar el último cañón, donde se muestra un texto en pantalla con los ganadores del juego, y la cámara hace un *panneo* entre las diferentes estructuras. Al pasar un par de segundos, el juego muestra un texto mostrando acerca de cómo volver a jugar.

Manual de Usuario

Controles básicos:

- ***Construcción:***
  - ***WASD:*** Mover el bloque por el volumen designado.
  - ***Q/E:*** Rotar el bloque hacia derecha o izquierda.
  - ***R:*** Cambiar el eje de rotación.
  - ***ESPACIO:*** Colocar el bloque en la posición elegida.
  - ***Z/X:*** Subir/bajar el bloque.
  - ***B:*** Deshacer movimiento. Reduce el número de bloques totales que podemos colocar en 1.
  - ***1234:*** Cambian el punto de vista de la cámara.
  - ***Mantener Click Derecho:*** Cambiar la dirección de vista de la cámara.
- ***Destrucción:***
  - ***ESPACIO:*** Disparar proyectil.
  - ***Mantener Click Derecho:*** Cambiar la dirección de vista de la cámara.
- ***Puntuaciones:***

- ***ESPACIO:*** Reiniciar el juego al aparecer el texto.

Efectos incorporados

Todos los elementos del juego son *Sólidos Rígidos*, exceptuando diversos efectos de partículas y generadores de fuerza.

Los diferentes objetos implementados mediante el uso de *Sólidos Rígidos* son los siguientes:

- ***Suelo y Paredes***: Tal y como el nombre indica, se han creado paredes y suelo con colisiones y con un *body estático*, haciendo que estos sean objetos inamovibles con los poder chocar, intentando marcar un límite de la zona de juego.
- ***Plataforma***: Se trata de la plataforma sobre la que los jugadores han de colocar sus piezas. Se basa en los mismos principios que el suelo y las parades.
- ***Bloques de Construcción***: Estos bloques usan dos representaciones: la *estática* y la *dinámica*. Durante la fase de *Construcción* anteriormente descrita, en su primera parte, los *sólidos* se comportarán de forma estática; pero al acabar la construcción, se sustituirán dichos sólidos por unos con un *body dinámico*, para que tenga comportamiento físico, le afecta la gravedad, y pueda colisionar con los demás bloques de la estructura y del escenario. Es importante notar que estos bloques pueden colocarse uno dentro de otro en modo estático, causando que, al acabar su sustitución por bloques dinámicos, el sistema de colisiones haga que los bloques se muevan de forma inesperada. Esto se ha dejado así por cuestión de diseño: El jugador debería no cometer errores a la hora de construir su estructura.
- ***Bolas de Cañón***: Usan un comportamiento *dinámico*. Al ser instanciadas en la escena se les añade una fuerza determinada en la dirección en la que apunta la cámara. Este objeto produce, en su colisión con bloques de la estructura rival, una explosión de fuerza igual a la mitad de la magnitud de su velocidad, empujando los bloques a su alrededor. Este comportamiento también genera una explosión de partículas que se ven igualmente empujadas por la explosión. Las bolas de cañón desaparecen al cabo de 3.0 segundos.

Las partículas son gestionadas por un *Sistema de Partículas*, que usa dos tipos de generadores de fuerzas, y 2 de partículas:

- ***Fuerzas – Gravedad:*** Es un generador de fuerza realmente simple, su única función es aplicar una fuerza constante (*Vector3(0,-9.8,0)*) a todas las partículas generadas, haciendo que cada vez caigan a mayor velocidad.
  Fórmula utilizada (siendo p una instancia de la clase partícula, y _gravity el vector anteriormente mencionado):
  `p->addForce(_gravity * p->getMass());`
- ***Fuerzas – Explosión***: Este generador simula una explosión desde un punto del espacio que se propaga circularmente desde 0 a X de radio durante Y tiempo (X e Y son parámetros configurables para cada explosión). Las partículas se ven empujadas en dirección contraria al vector que les une con el centro de la explosión con una fuerza cada vez menor, ya que la distancia aumenta.
  Fórmula utilizada (siendo time el tiempo que lleva activa la explosión, distDiff el vector que une la partícula con el centro de la explosión, r el radio actual, explosionForce una constante definida al construir la explosión, y timeConst una constante de tiempo):

```
time += t;
Vector3 force = Vector3(0, 0, 0);
float distDiff = (pPos - pos).magnitude();
```

```
const float timeConst = 20.0f;
if (time < 4 * timeConst) {
      r = velocity.magnitude() * time;

      force = (explosionForce / (distDiff * distDiff)) * (pPos - pos)
      * exp(-time / timeConst);
      if (distDiff < r) p->addForce(force);
}
```

- ***Partículas - Explosión***: Un generador de partículas que genera N partículas en el punto Y (siendo N e Y parámetros configurables para cada explosión). Estas partículas se generan en una disposición circular aleatoria, dejando un ligero espacio con el centro de la explosión, para poder tener su dirección definida con la fórmula anterior. Este generador también sirve para generar *Fireworks* con una generación K, haciendo que, al explotar, pueden generar un hijo de generación K-1, hasta que K = 0, cuando ya no se generan más hijos. Para simular una explosión, se usa un generador de generación 0 desde el principio.
- ***Partículas – Fireworks:*** Se usan dos generadores, el arriba mencionado para generar la explosión como tal, y las explosiones consecutivas de cada partícula generada; y el generador que genera las partículas que comenzarán la reacción en cadena. Este generador lanza un único *firework* cada 1.5 segundos, con generación K, que a su vez añadirá un generador de explosión a la escena, siguiendo el comportamiento descrito en *Partículas – Explosión.*

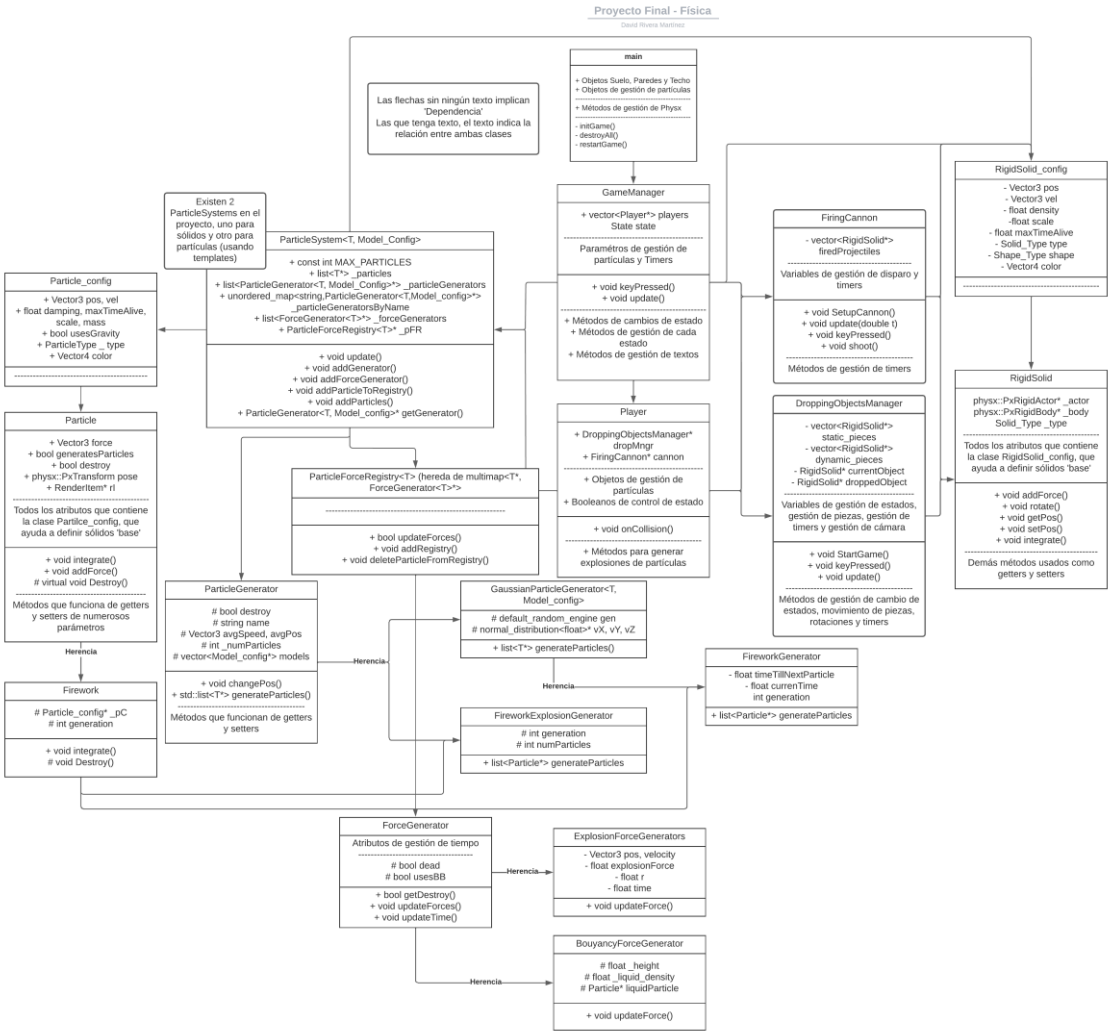
Se aplican 3 fuerzas a los *sólidos rígidos*, gestionados en otro sistema de sólidos, que tiene el mismo funcionamiento que el sistema de partículas, pero definido para la clase RigidSolid (la cual representa los objetos sólidos en el proyecto):

- ***Gravedad:*** Esta fuerza no se encuentra implementada por nosotros, sino por el propio comportamiento de la librería de *Physx* con su sistema de escenas, actores y bodys. Tiene un comportamiento ciertamente similar al descrito en nuestro generador de gravedad anteriormente descrito.
- ***Explosión:*** Usa la misma implementación descrita en *Fuerzas – Explosión,* en el apartado de partículas, salvo porque está hecho para funcionar con *sólidos rígidos*, añadiendo la fuerza directamente a su *physx::actor*, que será gestionada por propia librería física.
- ***Flotación:*** El sistema de flotación se encuentra implementado como agua en nuestro proyecto, cubriendo toda la superficie del suelo de la escena. Afecta a los *sólidos rígidos* una vez se caigan de la plataforma.
  Fórmula utilizada (siendo particle el sólido/partícula actual, _liquid_particle la partícula que nos ayuda a ubicar el generador, _height la altura del sólido/partícula, immersed un float que describe cómo de inmerso está el sólido en el líquido, _liquid_density la densidad del fluido):

```
float h = particle->getPos().y;
float h0 = _liquid_particle->getPos().y;

_height = particle->getHeight();

Vector3 f(0, 0, 0);
float immersed = 0.0;
if (h - h0 > _height * 0.5)
     immersed = 0.0;
else if (h0 - h > _height * 0.5)
    immersed = 1.0;
else
```

```
        immersed = (h0 - h) / _height + 0.5;
    f.y = _liquid_density * particle->getVolume() * immersed * 9.8;

    particle->addForce(f);
```

Diagrama de Clases



El flujo del programa comienza, como no, en la función *main*, que inicializa todo el sistema de Physx, además de los objetos de la escena (paredes y suelo). Inicializa también los *managers* necesarios para el funcionamiento del juego: *GameManager, ParticleSystem<Particle>, ParticleSystem<RigidSolid>, ParticleForceRegistry<Particle>* y *ParticleForceRegistry<RigidSolid>. Main* se encargará de hacer las llamadas de *update* a todos los managers mencionados.

GameManager se encarga de gestionar todos los estados del juego. Crea a los jugadores, gestiona el movimiento de cámara y los tiempos de cada estado. Llama en cada *update* al jugador actual, para que realice su turno; y gestiona todo lo que tenga que ver con los textos en pantalla. Es realmente el que maneja todo el proyecto.

ParticleSystem gestiona todas las partículas en escena, gestionando su movimiento y añadiéndolas al registro de fuerzas (ParticleForceRegistry) para hacer que se le apliquen las fuerzas designadas. Existen dos instancias de ParticleSystem, uno que acepta Particles y otro que acepta RigidSolids.