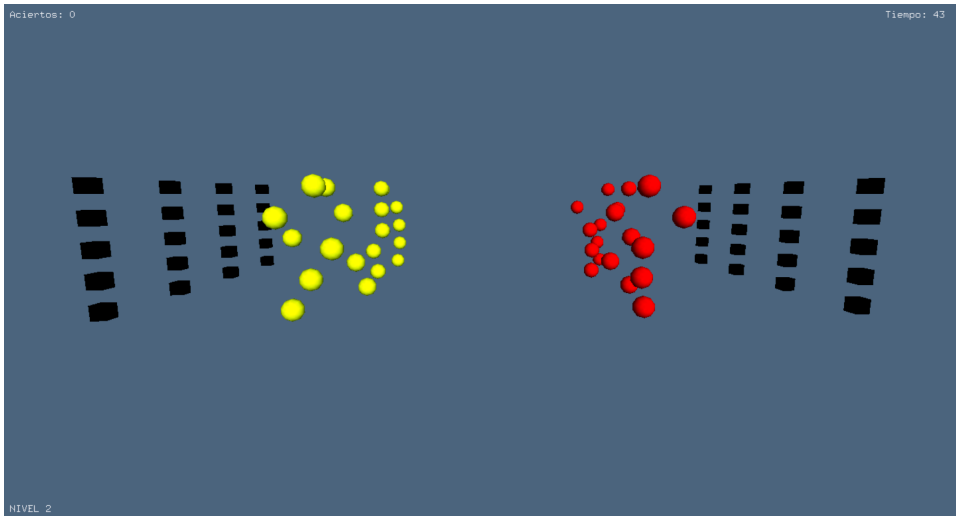
AIM DOWN - Memoria



Temática - ¿A qué se juega?

Aim down es un juego del tipo *shooter* en primera persona, en el cual el jugador debe eliminar el mayor número de objetivos posibles en un límite de tiempo, apelando a las atracciones del estilo *tiro al plato.* Los controles son muy básicos: *SPACE* para disparar y *click izquierdo* para rotar la cámara(aprovechando así el input ya presente en el proyecto base de *physx,* aunque suprimiendo la posibilidad de movernos por la escena con las teclas *WASD*).

Se estructura en 3 niveles distintos. En cada uno de ellos los objetivos a derribar del jugador se organizan de manera distinta, y el tiempo para derribar el máximo número de ellos posible cambia. De esta manera, variando cómo se colocan los objetivos, se han podido implementar diferentes generadores de fuerzas y de partículas/sólidos.

El primer nivel organiza los objetivos en dos columnas que se desplazan en direcciones opuestas (de derecha a izquierda y viceversa), movidas por generadores de viento. Los objetivos se generan aleatoriamente con distintas alturas en el eje Y. Al no existir colisiones entre objetivos las partículas se mueven hasta llegar al límite opuesto.

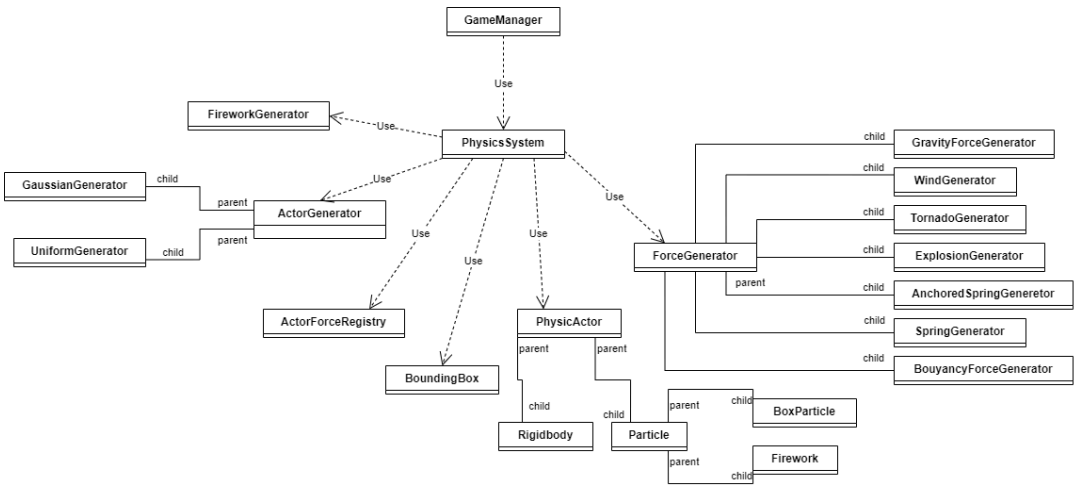
El segundo nivel se conforma de 4 filas de objetos con 4 columnas por cada una de ellas, donde cada uno está anclado a un muelle, para no complicar demasiado al jugador las partículas no sufren el efecto de la gravedad.

Por último, el tercero consta de un generador de objetos sólidos que rebotan con una cama elástica, y sí se ejerce sobre ellos la fuerza de gravedad. Los sólidos son generados a la misma altura pero con distinto tamaño y peso para que su comportamiento al rebotar con la cama elástica sea distinto en cada uno de ellos.

Para acertar a los objetivos y sumar un punto se disparan proyectiles blancos cuyo origen y dirección coinciden con los de la cámara. Como dificultad añadida, sobre estos se aplica una fuerza de gravedad negativa (de ahí el título del juego: apunta hacia abajo en inglés).

El objetivo del juego es derribar el mayor número de objetos por cada nivel en el tiempo otorgado. Para lograr las puntuaciones más altas el jugador deberá aprender a tener en cuenta la elevación de sus proyectiles tras ser disparados y calcular las futuras posiciones de sus objetivos. Al acabar se verán fuegos artificiales felicitando la partida al jugador.

Diagrama de clases



Ecuaciones físicas

A continuación se explicarán las distintas ecuaciones físicas presentes en el proyecto y se detallarán los distintos valores escogidos para ellas.

1. Gravedad

$$\overrightarrow{F_g} = \overrightarrow{g} \cdot m$$

La fuerza de la gravedad presente en el proyecto sigue la fórmula de arriba, donde se iguala al producto del vector aceleración de la gravedad por la masa del objeto.

De manera predeterminada, el motor *physx* permite asignar un vector aceleración gravitacional a la escena al crearla y se encarga automáticamente de aplicar la fuerza gravitacional sobre los sólidos rígidos que sean dinámicos (como los que se encuentran el el nivel 3).

```
sceneDesc.gravity = PxVec3(0.0f, -9.8f, 0.0f);
```

Para los objetos del juego que no son sólidos sino partículas (los objetivos y nuestros proyectiles) se cuenta con la implementación del generador de fuerza de gravedad realizado en la práctica 2. Mi implementación de este recibe también un vector aceleración gravitacional (además de origen y límites de acción del generador en sí). Aquí abajo encontramos los generadores creados para los proyectiles y los fuegos artificiales respectivamente (nótese que el de los proyectiles tiene aceleración positiva en Y).

```
GravityGenerator(Vector3(0, 7.5, 0), Vector3(0), Vector3(200));
```

```
GravityGenerator(Vector3(0, -9.8, 0), CENTER_POSITION, Vector3(200));
```

2. Viento

$$\overrightarrow{F_v} = k_1(\overrightarrow{v_v} - \vec{v}) + k_2|\overrightarrow{v_v} - \vec{v}|(\overrightarrow{v_v} - \vec{v})$$

Para el viento contamos con esta fórmula, donde k_1 y k_2 son constantes de rozamiento del viento (la primera estática y la segunda dinámica), *v* el vector velocidad del objeto sobre el que se ejerce la fuerza y v_v el vector velocidad del viento.

Es el nivel 1 el único que cuenta con generadores de viento y en él encontramos 2 de estos. Ambos son prácticamente idénticos (mismas constantes y dimensiones), salvo por sus vectores velocidad, que son opuestos (se pueden observar como el segundo atributo de la constructora).

```
WindGenerator(1, 0.1, Vector3(-25,0,0), Vector3(45, 25, 80), Vector3(20, 100, 20));
```

```
WindGenerator(1, 0.1, Vector3(25, 0, 0), Vector3(-45, 25, 80), Vector3(20, 100, 20));
```

3. Muelles

$$\vec{F} = -k(|\vec{d}| - l_0) \cdot \left(\frac{\vec{d}}{|\vec{d}|}\right)$$

En tercer lugar llegamos a los generadores de fuerza elástica. Dicha fuerza se describe según la ecuación superior, donde *k* es la constante de elasticidad del muelle, *d* el vector distancia entre ambos extremos del muelle y l_0 la longitud inicial del muelle.

Todos los presentes en el proyecto son del tipo anclaje, por lo que se pueden entender como muelles estáticos, y se encuentran en el nivel 2. Cada uno de ellos está a la misma distancia de su respectiva partícula que sujetan y tienen la misma longitud. Su constante de elasticidad (o constante de *Hooke*) es aleatoria entre 1 y 10. En la imagen de abajo se aprecian a ver los valores de las distintas propiedades siguiendo el orden *k,* l_0 y x_0.

```
AnchoredSpringGenerator((rand() % 10) + 1, 20, posPartR + Vector3(10, 0, 0))
```

4. Flotación - Rebote

$$\vec{F_f} = V_s \cdot d_l \cdot |\vec{g}|$$

Por último, la flotación sigue la siguiente fórmula siendo V_s el volumen sumergido del objeto, d_l la densidad del líquido y *g* el vector aceleración gravitacional.

Antes de poder ejecutar esta fórmula debemos calcular el volumen sumergido del objeto, para ello tenemos en cuenta la altura del centro del mismo y la altura del líquido (se hacen los cálculos vistos en la imagen de abajo, que provienen de las presentaciones de clase).

```
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;
```

En el tercer nivel, donde se encuentra el generador de flotación, los sólidos generados tienen masa y tamaños aleatorios por lo que no a todos se les aplicará la misma fuerza. A continuación tenemos el generador de flotación: recibe la altura del líquido, su densidad, su posición y sus dimensiones de acción.

```
BouyancyForceGenerator(2, 1000, centroTrampolin, Vector3(30, 300, 30))
```

5. Integrate - General

```
acc = force * inv_mass;
vel += acc * t;

vel *= powf(damping, t);

pose.p += vel * t;
```

Para entender el funcionamiento general de los objetos físicos observemos su método *integrate* que actualiza sus principales parámetros relacionados con el movimiento.

Partiendo de la fórmula $\vec{F} = m \cdot \vec{a}$ (la segunda *Ley de Newton)* donde *F* es el vector fuerza, *m* la masa y *a* el vector aceleración, podemos deducir esta $\vec{a} = \frac{\vec{F}}{m}$, que nos permitirá actualizar la aceleración del objeto en cuestión según se modifique su vector *F*. Por ello los generadores de fuerzas que actúan sobre el objeto actualizan este vector, que a su vez modificará la aceleración. El resto de parámetros (*v y pose.p* = x_0) siguen las fórmulas del movimiento rectilíneo uniformemente acelerado, presentes a continuación.

$$v = v_0 \cdot at \qquad x = x_0 \cdot vt$$

Efectos implementados

- ***Nivel 1***

  Como ya se había mencionado anteriormente, el nivel uno cuenta con dos generadores de partículas y dos generadores de viento. Las partículas del oeste (situadas a la izquierda del jugador) son de color amarillo), y las de este (a la derecha del jugador) son rojas.

- ***Nivel 2***

  En este nivel encontramos la misma asignación de colores para las partículas, pero no hay generadores de ellas, se crean y colocan a mano (no se querían colocar de manera aleatoria sino estructuras en dos grupos de cuatro filas y columnas cada uno). Por cada partícula hay un generador de fuerza elástica (muelle) al que se anclan.

- ***Nivel 3***

  En el último nivel, encontramos un generador de sólidos rígidos de tamaños y masas aleatorios, pero todos de color rojo. La cama elástica sobre la que rebotan (generador de flotación) es de color amarillo.

- ***Jugador***

  El jugador dispara proyectiles de color blanco a los que les afecta un generador de gravedad negativa.

- ***Fin del juego***
  Por último, en la pantalla de final de juego, se ven fuegos artificiales en el fondo que se van acercando progresivamente, y sobre los que se ejerce una gravedad igual a la presente en la tierra. La existencia de fuegos artificiales indica también la de un generador de estos.

Enlaces de interés

- *UML* - https://drive.google.com/file/d/11iMWapygBNJ4NdC9cD1u6wbrUijTI897/view?usp=drive_link
- Repositorio - https://github.com/Rafaa0514/SimulacionFisicaVideojuegos