
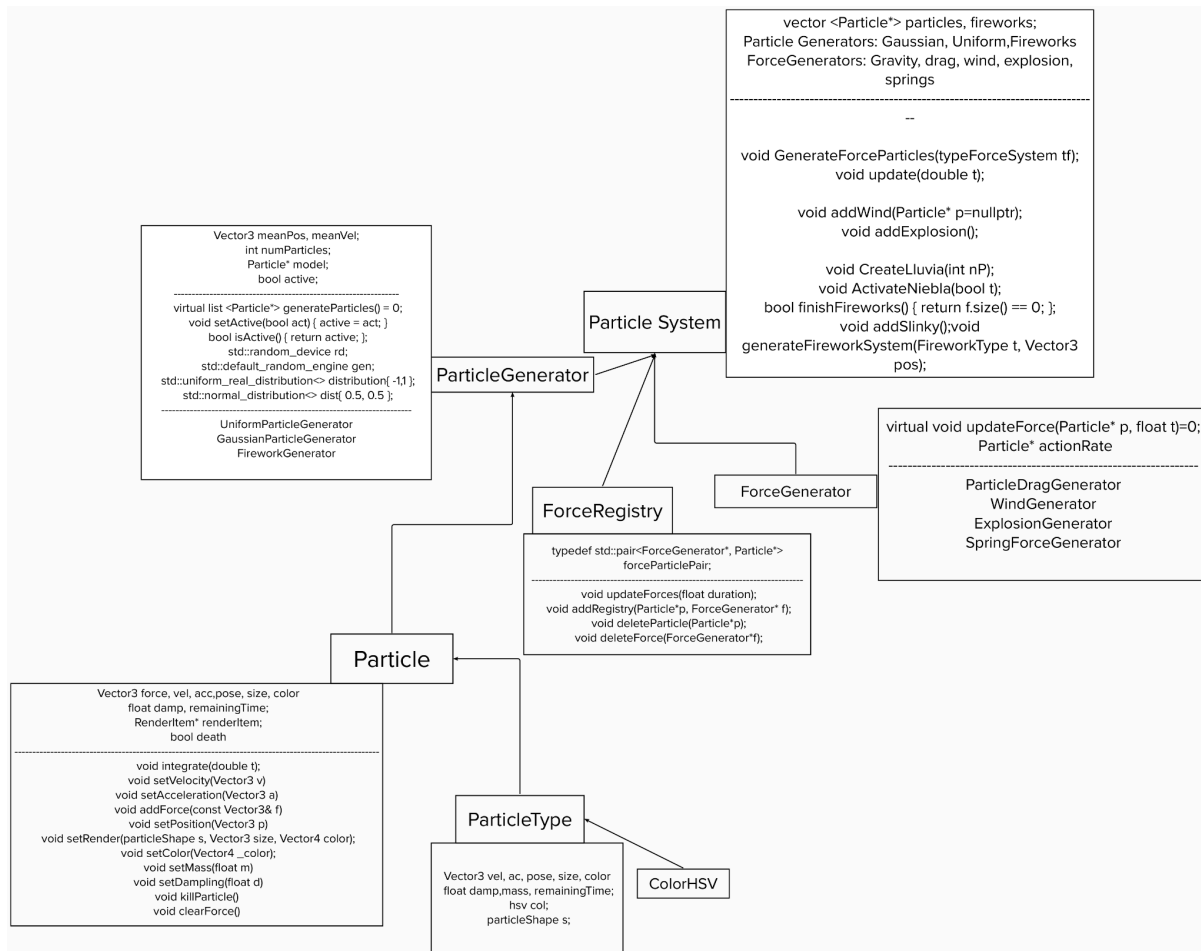


SIMULACIÓN FÍSICA PARA VIDEOJUEGOS: “POWER WASH SIMULATOR”

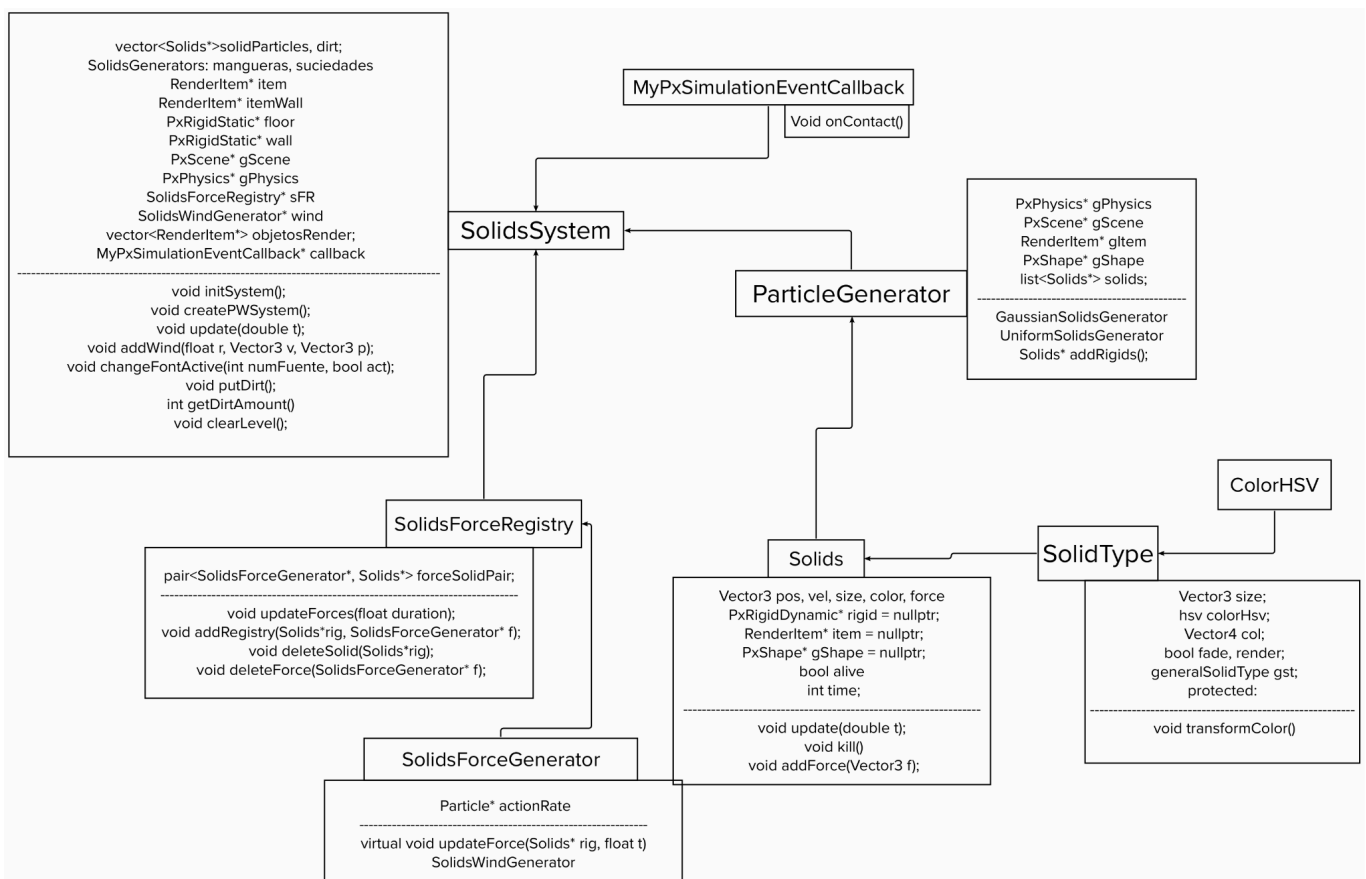
- **Temática:**

- Proyecto inspirado en el juego Power Wash simulator:
 [PowerWash Simulator Launch Trailer](#)
- El jugador tiene varias mangueras que salen desde la cámara, y con estas debe recorrer el escenario limpiando todas las partículas de barro que están pegadas a las diferentes estructuras. Cada manguera está diseñada para limpiar un tipo de barro. Una vez se ha limpiado toda la basura salen fuegos artificiales a modo de victoria de nivel y lobby entre niveles.
- Las mangueras:
 - La primera manguera, tiene velocidad y tamaño medio, limpia las partículas de barro más pequeñas.
 - La segunda manguera está formada por partículas más pequeñas que van a gran velocidad, y sirven para eliminar las partículas más grandes de barro.
 - La tercera manguera tiene partículas de forma más alargada y una velocidad más baja que las otras dos, sirve para eliminar tanto las partículas medianas como las más pequeñas.
- El juego consta de Tres niveles:
 - El primer nivel es sencillo, con una plataforma simple para limpiar.
 - El segundo nivel tiene un viento, que afectará al movimiento del agua de las mangueras, dificultando la limpieza de las plataformas.
 - El tercer nivel es un escenario en el que hay lluvia, que puede dispersarse generando una explosión que tiene su centro en la cámara.
- Al acabar cada nivel aparecen fuegos artificiales en el centro del escenario, y al acabar el juego aparece un silky como victoria final que se queda de forma permanente hasta que se cierra la aplicación.

• Diagrama de clases: Particle System



• Diagrama de clases: Solids System



• Ecuaciones utilizadas:

- Viento:

```
float k = 0.8;

if (checkDistance(rig))
    rig->addForce(k * (vel - rig->getVel()));
```

- Movimiento de las partículas: (Método de integración Euler semiimplícito)

```
Vector3 totalAcceleration = ac;

vel += totalAcceleration * t;

pose = physx::PxTransform(pose.p.x + vel.x * t, pose.p.y + vel.y * t, pose.p.z + vel.z * t);
//si hay fuerza actualiza la aceleración
if (force.magnitude() > 0)
    ac = force * inverse_mass;
vel *= powf(damp, t);
```

- Explosión: (Cuanto más cerca está la partícula del centro de la explosión, se le da más fuerza)

```
float k = 2000;
float R = 3000000 * t;
Vector3 particlePos = p->getPos();
float r = sqrt(powf(particlePos.x - meanPose.x, 2) + powf(particlePos.y - meanPose.y, 2) + powf(particlePos.z - meanPose.z, 2));

float explosionMultiply = exp(-(t / 2));
float x = k / radius * (p->getPos().x - meanPose.x) * explosionMultiply;
float y = k / radius * (p->getPos().y - meanPose.y) * explosionMultiply;
float z = k / radius * (p->getPos().z - meanPose.z) * explosionMultiply;
```

- Muelles: (fórmula de la fuerza del muelle k = constante elástica del muelle)

```
Vector3 f = particle->getPos() - p->getPos();

const float l = f.normalize();
const float delta_x = l - restingLength;

f *= delta_x * k;
```

- Drag: (para frenar la aceleración de las partículas)

```
Vector3 v = p->getVel();
float drag_coef = v.normalize();
Vector3 dragF;
drag_coef = k1 * drag_coef + k2 * powf(drag_coef, 2);
dragF = -v * drag_coef;
```

- Gravedad: (utiliza la masa del objeto porque la gravedad afecta igual para todas las partículas)

```
p->addForce(gravity * p->getMass());
```

- **Efectos incorporados:**

- Escena PWSimulator que contiene:
 - Un sistema de sólidos que controla todos los sólidos rígidos, genera las suciedades, el viento y coloca los sólidos por el escenario
 - 3 generadores uniformes de sólidos rígidos (mangueras) cuyos sólidos van perdiendo saturación a medida que se alejan del origen
 - Sistema de eliminación y actualización de los sólidos
 - Métodos para crear y destruir los niveles
 - Método para eliminar todas las instancias cuando se cierra el programa
- Un sistema de partículas para controlar los fuegos artificiales, el slinky del final del juego, la lluvia y la explosión que dispersa la lluvia
- En el update se controla el cambio de niveles, para eliminar los elementos del anterior nivel y crea el siguiente, además de activar y colocar los fuegos artificiales cuando es necesario

- **Manual de usuario:**

- w,a,s,d-> mover la cámara
- z,x,c-> activar/desactivar las distintas mangueras
- q-> generar explosiones para dispersar la lluvia(3º nivel)
- p-> eliminar todas las partículas de suciedad para comprobar los cambios de escenas

- **Extras:**

- He añadido a las prácticas el conversor de color rgb a hsv(ya incluido en las prácticas anteriores)
- Manejo de las colisiones entre sólidos, convirtiéndolo en kinematic para que no se muevan, y metiendo el método MyPxSimulationEventCallback, que controla qué pasa cuando chocan los sólidos y que sólidos reaccionan entre ellos.