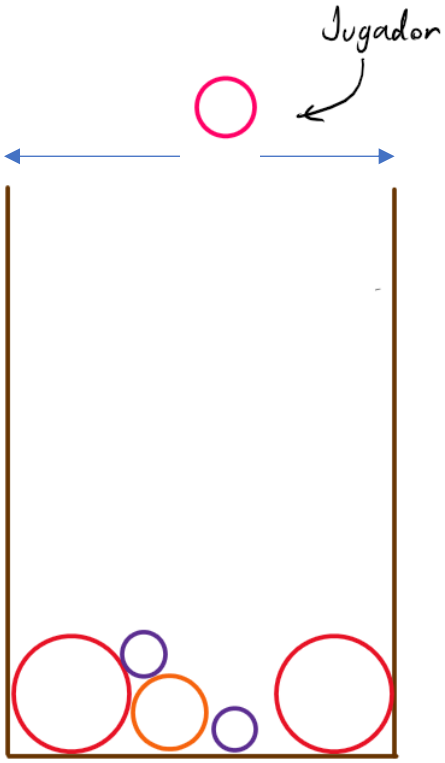
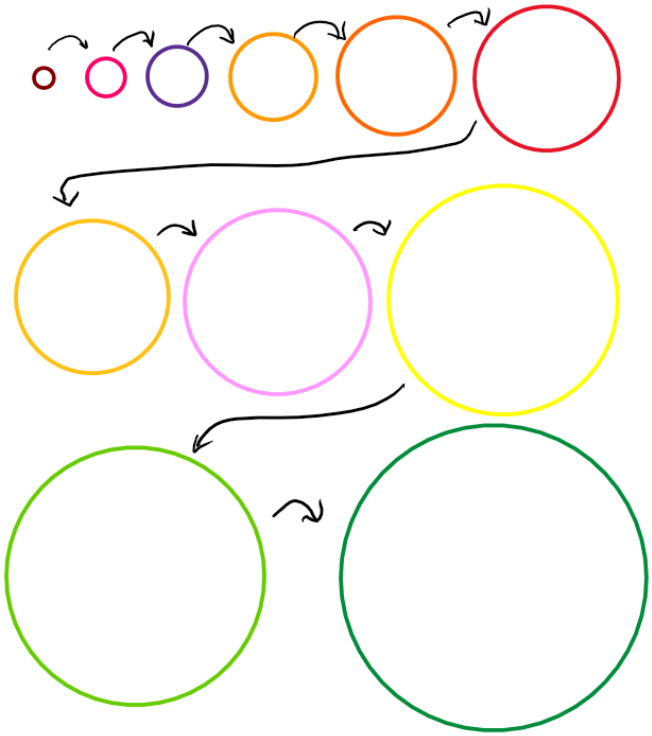
MEMORIA

TEMÁTICA

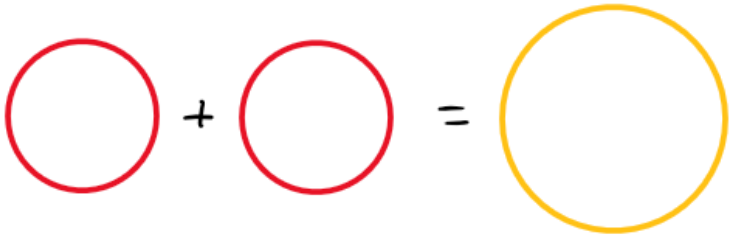
El juego es una copia del juego “Suika Game”, un juego de puzles desarrollado por “Aladdin X” para la Nintendo Switch. El jugador es capaz de soltar una fruta en un bol, cuando dos frutas iguales se tocan, se combinan en una fruta más grande y se consiguen puntos. La posición en la que se suelta la fruta es controlada mediante el movimiento del ratón El objetivo del juego es intentar conseguir la mayor puntuación posible. El jugador pierde si alguna fruta sale del bol, es decir como en el “Tetris”, una fruta no cabe en el bol y se pasa de altura



Esquema del tablero de juego

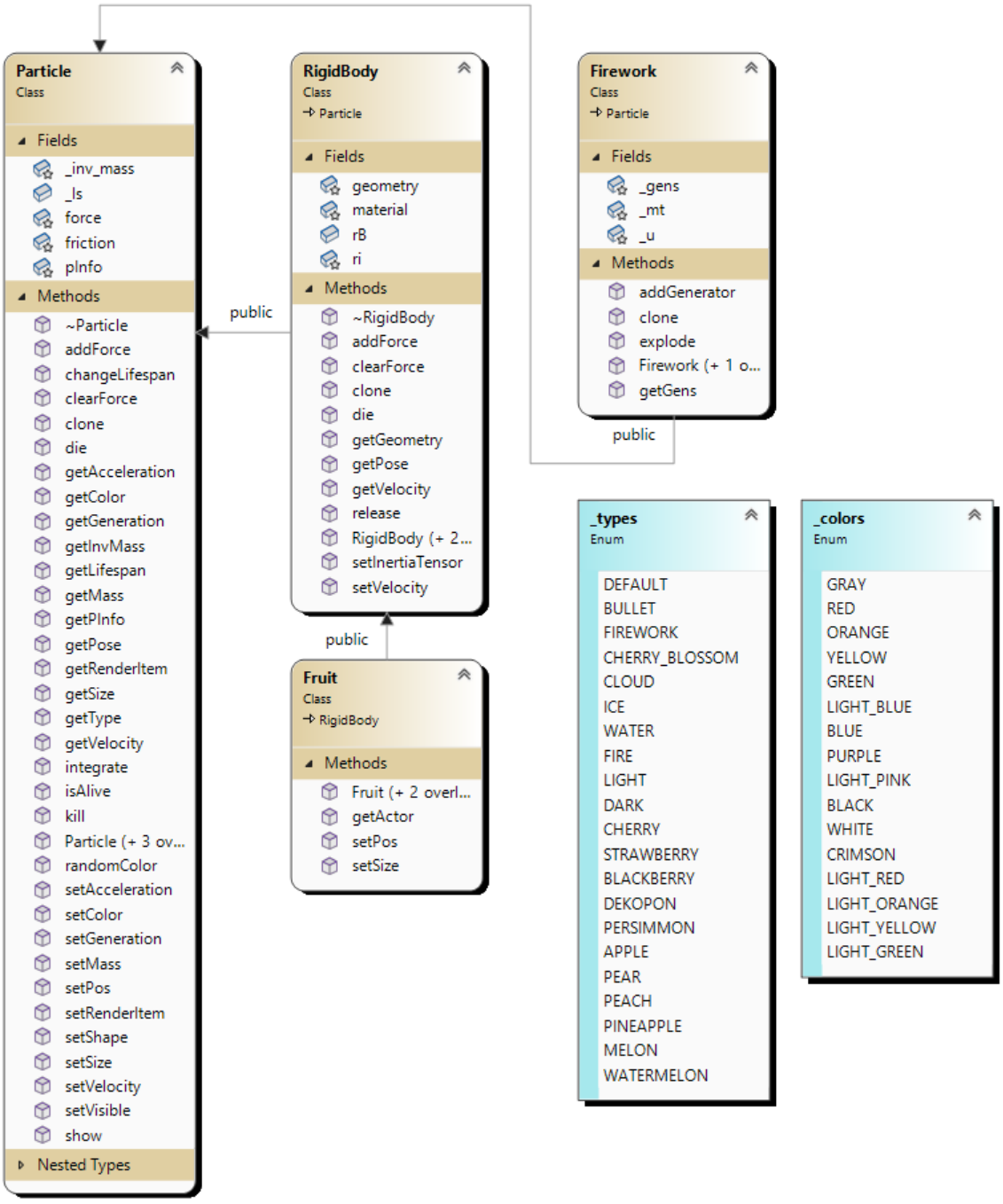


Progresión de frutas



Fusión de dos frutas

DIAGRAMA DE CLASES DEL SISTEMA DE PARTÍCULAS Y DE SÓLIDO RÍGIDO



ECUACIONES FÍSICAS UTILIZADAS

SQUISHEDTORNADOGENERATOR

Simula la parte circular de un tornado en los ejes “x” y “y”. Hereda de WindGenerator. No se aplican fuerzas en el eje z para simular el comportamiento de un juego 2D

```
const Vector3 aux = p->getPose().p;

windVel = K * Vector3(-(aux.z - origin.z) + 2.0, aux.x - origin.x, 0);

WindGenerator::updateForce(p, t);
```

Parámetros:

- k: 0.05 (fuerza del tornado)
- Origen: { -50,80,-45 }
- Área: -1 (infinito)
- Velocidad { 0.1,0.0,0.1 }

SPRINGFORCE

Simula un muelle. Se usa para que el indicador de la fruta siguiente tenga un poco de movimiento

```
Vector3 rel_pos = other->getPose().p - p->getPose().p;

const float length = rel_pos.normalize();

const float delta_x = length - restingLength;

Vector3 force = rel_pos * delta_x * K;

p->addForce(force);
```

Parámetros:

- k: 20 (fuerza del muelle)
- RestLen: 5 (longitud en descanso)

GRAVITYFORCEGENERATOR

Simula la gravedad

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

Parámetros:

- Gravedad : { 0.1,0.0,0.1 }

EXPLOSIVEFORCE

Simula una explosión. Se utiliza para conseguir un poco de espacio cuando se instancia una fruta y así evitar superposiciones entre rigidbodies.

```
Vector3 pos = p->getPose().p;

double r = (pos - origin).magnitude();

if (r >= (area / dur) * _t) return;

if (r < 1e-10) r = 0.1;

Vector3 expF = (K/ pow(r, 2)) * (pos - origin) * exp(-(_t / dur));

p->addForce(expF);
```

Parámetros:

- Origen: fruit2->getPose().p (posición de una de las frutas que se combinan)
- K: 1000
- Area: nextFruit->getSize() + nextFruit->getSize() - fruit1->getSize() * fruitScale (diferencia de tamaño entre la fruta que se combina y la fruta en la que se transforma)
- Duración: 0.1

EFECTOS INCORPORADOS

- Sistema de juego
  - Se encarga de gestionar todo lo relacionado al juego
  - Instancia y destrucción de "tablero" y frutas
  - Detección de ganar/perder
  - Procesar colisiones
  - Procesar input del jugador (ratón y teclado)
- Generadores de fuerzas
  - Tornado 2D
  - Gravedad
  - Muelle
  - Explosión
- Generador de partículas
  - Brillo para la fruta del jugador
  - Fuegos artificiales
- 11 tipos distintos de rigidbodies cada uno con su propia masa y tamaño

MANUAL DE USUARIO

- Movimiento de izquierda a derecha con el ratón
- Dejar caer fruta con el botón izquierdo del ratón
- Poder de tornado con la tecla "s"

EFECTOS NO INCLUÍDOS

- Fuerza centrípeta (Vector3 f = p->getMass() * (vel * vel) / r * K-----9
- Dynamical billiards
- Espiral de Arquímedes (x^2+y^2=k^2(arctan(yx)−θo)^2)