** Este archivo lo he generado convirtiendo un archivo .md a pdf. En caso de que se quiera ver de manera correcta, verlo desde el link del repositorio.** https://github.com/JorgeBarcena3/Box2D-Animation

# Box 2D - Animation - Jorge Bárcena Lumbreras

Este proyecto consistía en realizar una animación propuesta, utilizando la librería para simulación de físicas **Box2D**. Para llevar acabo este proyecto he utilizado **C++** y **SFML** para gestionar la creación de ventanas.

Se especificaran algunos detalles del proyecto, sobre como funcionan ciertos objetos y notas sobre su arquitectura.

Cambien podemos ver algunos ejemplos de la demo en la carpeta de "documents" (en formato .mp4 y .gif).

Todo el código esta documentado para que doxygen genere los documentos pertinentes (Se debe tener activada la opción de JAVA_DOC_BRIEF).

### Índice



## Librerias externas

Para la realización de este proyecto he utilizado las siguientes versiones de librerías externas:

- Box2D: Simulación de un mundo de físicas.
- SFML 2.5.1: Creación de la aplicación de ventana.
- Box2D SFML2 Debug Draw: Dibujar todas las fixtures de Box2D en modo debug.

## Utilizacion de la Demo

Para ver la demo en funcionamiento, podemos ejecutar y compilar el proyecto en Visual Studio 2019 o ejecutar el archivo llamado "JorgeBarcena-Box2D.exe" situado en la carpeta de "binaries".

### Controles

- **A / Flecha Izquierda:** Acelerar hacia la izquierda
- **D / Flecha Derecha**: Acelerar hacia la derecha
- **N**: Comenzar los sistemas de partículas del coche
- **M**: Parar el sistema de partículas
- **R:** Rotar el coche a la izquierda
- **P:** Que el ascensor aparezca para que el coche pueda subirse en el.

### Instrucciones para la demo

1. Se debe acelerar hacia la derecha todo el rato, hasta llegar a la plataforma de la derecha verde.
2. Una vez ahí un elevador aparecerá en el centro de la pantalla, y las pelotas que están en el molino caerán.
3. Una vez con las pelotas en la caja, bajaremos hasta estar situados en el elevador, tras 5 segundos después de haber tocado la plataforma del ascensor, esta ascenderá.
4. Una vez arriba debemos ir hacia la izquierda. Cuando estemos preparados y en posición, presionar el botón **R** varias veces, hasta que nuestro coche esté completamente rotado.
5. Cuando alguna pelota caiga en el recipiente con forma de cono invertido, la demo habrá acabado y los sistemas de partículas se activaran, impidiendo el movimiento del coche.

## Modo debug

El modo debug nos permitirá, ver todas las fixtures de Box2D que hay en nuestro mundo, y se están renderizando en la pantalla. También podremos ver la configuración de los joints y los objetos que se encuentran en movimiento. Para activar/desactivar esta funcionalidad debemos, incluir una clase que se encargue de realizar las llamadas del **b2World** por nuestra pantalla, esta clase debe heredar e implementar todos los métodos virtuales puros de **"b2Draw"**; también debemos setear la variable **"debugMode"** de la escena a **"true"**.

Este es un ejemplo de como se podría hacer:

```
Scene myScene;
myScene.setDrawTool(debugDraw);

myScene.debugMode = false;
```

## Capturador de colisiones

Todas las colisiones que se generan entre dos cuerpos son capturadas por la clase **"ContactHandler"**. Esta clase hereda de **"b2ContactListener"**, y solamente implementa el método **"BeginContact(b2Contact)"** Una vez ahí, en función de los objetos que han colisionado se envía añaden a la cola de la escena las acciones a realizar. A continuación se muestra como se implementa esta clase:

```
class ContactHandler : public b2ContactListener
{
    virtual void BeginContact(b2Contact* contact) override;

};
```

## Pool de acciones de la escena

La escena tiene un pool de acciones que realizará en cada ciclo de update. La clase **"Scene"** implementa un método estático que permite añadir acciones a la lista de la escena activa. Para añadir una acción a la lista de acciones a realizar se debe hacer de esta manera:

```
Scene::addActionToPool("Start scene");
```

Esta funcion añanade la accion *"Start scene"* al pool de acciones a realizar. Y se realizará en el siguiente ciclo de update de la escena.

## Sistemas de partículas

El sistema de partículas que he desarrollado, tienes posibles modos de configuración, para que actúen de la manera que nosotros deseamos. Las características de este sistema vienen dadas en la siguiente enumeración.

```
/*
 * Definicion del sistema de particulas
 */
struct ParticleSystemDef
{
    float coneAngle = 1; ///< Cono de apertura para las particulas

    int particlesNumber = 20; ///< Cantidad de particulas

    PARTICLE_TYPE type = PARTICLE_TYPE::CIRCLE; ///< Tipo de particulas

    sf::Vector2f direction = { 0,1 }; ///< Direccion de las particuals

    sf::Color color = sf::Color::Red; ///< Color base

    bool randomColor = false; ///< Randomiza el color

    sf::Vector2f baseSize = {5,5}; ///< Tamaño de la particula

    sf::Vector2i speedRange = {1, 5}; ///< Rango de velocidad

};
```