Programación con java LibGDX. BOX2D

Box2D es un motor físico de código abierto para simular cuerpos en 2D. Desarrollado por Erin Catto.

Un motor físico es un software capaz de realizar simulaciones de ciertos sistemas físicos como la dinámica del cuerpo rígido, el movimiento de un fluido y la elasticidad. Son utilizados ampliamente en los videojuegos.

Box2D está escrito en C++, pero a sido portado a diferentes lenguajes como Java, C#, JavaScript, Python, Flash.

**Mundo:**

Cada aplicación con Box2D comienza con la creación de un objeto del tipo World. La clase World se encarga de manejar la memoria, objetos y la simulación. Para inicializar Box2D, lo primero que hay que hacer es llamar

***Box2D.init();***

Crear un mundo por primera vez tiene el mismo efecto, pero de preferencia utilizar la clase Box2D.

Al configurar Box2D lo primero que necesitamos es un mundo. El objeto mundo es básicamente lo que tiene todos sus objetos físicos y simula las reacciones entre ellos. Sin embargo, los objetos no son renderizados automáticamente. Aun así, libgdx viene con un renderizado debug para Box2D.

Para crear el mundo se utiliza el siguiente código:

***World mundo = new World(new Vector2(0, -9.8f), true);***

Donde el primer valor (vector2) es la gravedad y el segundo valor es si está permitido que dos objetos entren en reposo. La gravedad afecta a cada cuerpo dinámico que se encuentra en nuestro mundo en este caso x=0 lo que significa que no hay gravedad horizontal e y=-9.8f que es casi la misma gravedad que la Tierra. Cuando un cuerpo entra en reposo el mundo deja de hacer simulaciones en el lo que incrementa el rendimiento de nuestra aplicación.

Si quisiéramos un mundo sin gravedad nuestro código se vería así:

***World mundo = new World(new Vector2(0, 0), true);***

Corriendo las simulaciones: Box2D utiliza un algoritmo computacional llamado "integrator" que simula la física del mundo en pasos discretos de tiempo. Para simular tenemos que escoger un paso de tiempo ( time step). Normalmente los juegos corren a 60fps y tienen un "time step" de 1/60 segundos.

Para nuestros ejemplos utilizaremos el tiempo transcurrido desde el frame pasado y el actual, delta.

Pero no solo necesitamos un time step. En cada paso de la simulación los cuerpos son actualizados dependiendo de las fuerzas que actúan en el. El algoritmo encargado de esto se llama "constraint solver".

Existen 2 fases en la simulación, la fase de velocidad y la fase de posición. En la fase de velocidad las cuerpos se mueven de acuerdo a sus impulsos y en la fase de posición se ajustan las posiciones para prevenir que se traslapen los cuerpos.

Por lo que entre más iteraciones más precisa será nuestra simulación pero menor rendimiento. Lo más usual es utilizar 8 iteraciones de velocidad y 6 iteraciones de posición. por lo que nuestro código se verá así:

Este código es importante añadirlo al final del método render() del ciclo de libgdx.

Si el juego funciona a 60 cuadros por segundo:

***mundo.step(1/60f, 8, 6);***

O si queremos que funcione con el mismo delta:

***mundo.step(delta, 8, 6);***

Antes de agregar cuerpos a nuestro mundo, es importante tener en cuenta que box2D trabaja con unidades MKS (metros, kilogramos, segundos), donde 1 metro del mundo es igual a 1 pixel de la pantalla. Es importante realizar una conversión adecuada de unidades para nuestro juego para poder programar de una forma cómoda. Esto es tedioso pero es importante.

Creamos la siguiente variable:

***public static float units = 100f;***

Siempre que creamos o manejamos un objeto del mundo, cada valor de posición se debe dividir por el valor de units. Por otro lado, si un objeto que no es del mundo queremos posicionarlo respecto al mundo, hay que multiplicar sus valores por units. Con esto nos aseguramos que un metro del mundo, sean representados con 100 pixeles de la pantalla, lo cual es más lógico.

**Debug Renderer:**

Lo siguiente que vamos a hacer es configurar el renderizado de Box2D. En general, esto no se usa en el juego, pero para propósitos de prueba es necesario poder visualizar los cuerpos que creamos con box2D:

***Box2DDebugRenderer debugRenderer = new Box2DDebugRenderer();***

Ahora creamos una cámara que se encargará de renderizar nuestro mundo:

***//crear la camara del mundo box2d***

***camaramundo = new OrthographicCamera();***

***camaramundo.setToOrtho(false, SCREEN\_WIDTH/units, SCREEN\_HEIGHT/units);***

Y en el método render():

***debugRenderer.render(mundo, camaramundo.combined);***

Pasos a seguir para agregar objetos a nuestro mundo:

-Definir el cuerpo (BodyDef)

-Crear el cuerpo (body)

-Definir el fixture

-Crear el fixture

**Cuerpos (Body)**

En Box2D nuestros objetos se denominan cuerpos (body), y cada cuerpo se compone de uno o más fixtures, que tienen una posición fija y orientación dentro del cuerpo. Nuestros fixtures pueden ser de cualquier forma que puedas imaginar o puede combinar una variedad de diferentes fixtures para lograr obtener la forma que desees.

Un fixture tiene una forma, densidad, la fricción y la restitución que se le atribuye. La forma es obvio. La densidad es la masa por metro cuadrado (una bola de boliche es muy denso, pero un globo no es muy densa en absoluto, ya que está lleno principalmente con el aire) . La fricción es la cantidad de fuerza de oposición cuando el objeto se desliza a lo largo de algo (un bloque de hielo tendrían una fricción muy baja, pero una pelota de goma o un cuerpo rugoso tendrían una alta fricción). La restitución es el rebote (una roca tendría una restitución muy baja, pero una pelota de baloncesto tendría bastante alto) Un cuerpo con una restitución de 0 llegará a su fin tan pronto como toque el suelo, mientras que un cuerpo con una restitución de 1 rebotaría a la misma altura de la cual se dejó caer.

Hay tres tipos diferentes de cuerpos: dinámicos (dynamic), estáticos (static) y cinemáticos (kinematic). Los cuerpos estáticos no reaccionan a ninguna colisión (pero si proporcionan una), fuerza o impulso por lo que no se mueven. Los cuerpos dinámicos si reaccionan a fuerzas, impulsos, colisiones y a otros eventos del mundo. Y los cuerpos cinemáticos son muy parecidos a los cuerpos estáticos ya que no reaccionan a las fuerzas por ejemplo en una colisión, pero a diferencia de los estáticos los cinemáticos si se pueden mover.

Antes de empezar, creamos las variables de nuestros cuerpos:

***private Body cuerpo;***

***private Body piso;***

***private Body caja;***

1- Para crear un cuerpo en nuestro mundo el primero paso es crear un BodyDef que es la definición del cuerpo y sirve para mantener toda la información necesaria para crear nuestro cuerpo. Teniendo creadas las variables con los datos del ancho y alto de nuestra pantalla, creamos un nuevo objeto dinámico en el centro de la pantalla y en la parte superior.

***BodyDef cuerpoPelota = new BodyDef();***

***cuerpoPelota.position.set(SCREEN\_WIDTH/2/units, SCREEN\_HEIGHT/units);***

***cuerpoPelota.type = BodyType.DynamicBody;***

2- Creamos la figura del objeto:

Nuestro cuerpo tendrá una figura circular por lo que tenemos que crear una figura circular (Shape) Una figura es un objeto geométrico, como un circulo, cuadrado, una línea o un polígono, en caso de que sea un polígono debe ser un polígono convexo ya que Box2D solo funciona con polígonos convexos (recordar SAT Separating Axis Theorem)

Para crear nuestra figura circular utilizamos el siguiente código:

***CircleShape circulo = new CircleShape();***

***circulo.setRadius(25/units); //un radio de 25 pixeles de la pantalla***

3- Creamos y definimos el fixture:

Como ya dije, sirven para dar tamaño, forma y otras propiedades como densidad, fricción y restitución. Las principales propiedades son:

Densidad - Que tan pesado es un cuerpo en relación con su área

Fricción - Que tan resbaloso es el cuerpo

Restitución - Que tanto rebote tiene el cuerpo

Figura (Shape)

Existen más propiedades que explicaré más adelante.

Creamos la definición del fixture y le asignamos el cuerpo circular que creamos anteriormente.

***FixtureDef fixDef=new FixtureDef();***

***fixDef.shape=circulo;***

***//Propiedades opcionales***

***fixDef.density = 2f;***

***fixDef.friction = 1f;***

***fixDef.restitution = 0.5f;***

***//propiedades mas opcionales***

***fixDef.filter.categoryBits = BIT\_PELOTA;***

***fixDef.filter.maskBits = BIT\_PISO | BIT\_CAJA;***

***fixDef.isSensor = false;***

Por último añadimos el objeto al mundo que hemos creado:

***cuerpo = mundo.createBody(cuerpoPelota);***

***cuerpo.createFixture(fixDef).setUserData("pelota"); // añadimos el fixture al objeto***

***//parámetros opcionales:***

***cuerpo.setGravityScale(1); // escala de la gravedad que lo afecta***

***cuerpo.isBullet(); //si es un cuerpo muy pequeño que viaja muy rápido***

Si hacemos funcionar lo que tenemos por el momento, deberíamos ver un circulo cayendo en nuestra pantalla.

Ahora añadimos dos nuevos objetos a nuestro mundo, uno estático que funcionará como el suelo, y otro cinemático:

***//OBJETO PISO***

***BodyDef cuerpoPiso = new BodyDef();***

***cuerpoPiso.type = BodyType.StaticBody;***

***//creamos una línea que representa el suelo, del ancho de la pantalla, en la posición 20 en Y***

***EdgeShape linea = new EdgeShape();***

***linea.set(0, 20/units, SCREEN\_WIDTH/units, 20/units);***

***//creamos el fixture***

***FixtureDef fixDef=new FixtureDef();***

***fixDef.shape=linea;***

***fixDef.filter.categoryBits = BIT\_PISO;***

***fixDef.filter.maskBits = BIT\_PELOTA | BIT\_CAJA;***

***//lo añadimos al mundo***

***piso = mundo.createBody(cuerpoPiso);***

***piso.createFixture(fixDef).setUserData("piso"); // añadimos el fixture al objeto***

***//OBJETO CAJA***

***BodyDef cuerpoCaja = new BodyDef();***

***cuerpoCaja.type = BodyType.KinematicBody;***

***cuerpoCaja.position.set(SCREEN\_WIDTH/2/units, SCREEN\_HEIGHT/4/units);***

***//creamos un rectángulo que representa la caja giratoria***

***PolygonShape rectangulo = new PolygonShape();***

***rectangulo.setAsBox(30/units, 5/units);***

***//creamos el fixture***

***FixtureDef fixDef=new FixtureDef();***

***fixDef.shape=rectangulo;***

***fixDef.filter.categoryBits = BIT\_CAJA;***

***fixDef.filter.maskBits = BIT\_PELOTA;***

***//lo añadimos al mundo***

***caja = mundo.createBody(cuerpoCaja);***

***caja.createFixture(fixDef).setUserData("caja"); // añadimos el fixture al objeto***

Para concluir, le damos una velocidad de rotación a nuestra caja para aprovechando sus propiedades cinemáticas. En render() añadimos la siguiente linea:

***caja.setAngularVelocity((float) Math.toRadians(360));***

Lo haremos girar en sentido contrario a las agujas del reloj, es importante recordar que Box2D trabaja con radianes y no con grados.

Con eso concluye la base para el manejo de Box2D en libgdx.

Código completo de la clase:

**import** com.badlogic.gdx.ApplicationAdapter;

**import** com.badlogic.gdx.Gdx;

**import** com.badlogic.gdx.graphics.GL20;

**import** com.badlogic.gdx.graphics.OrthographicCamera;

**import** com.badlogic.gdx.graphics.g2d.SpriteBatch;

**import** com.badlogic.gdx.math.Vector2;

**import** com.badlogic.gdx.physics.box2d.Body;

**import** com.badlogic.gdx.physics.box2d.BodyDef;

**import** com.badlogic.gdx.physics.box2d.BodyDef.BodyType;

**import** com.badlogic.gdx.physics.box2d.Box2D;

**import** com.badlogic.gdx.physics.box2d.Box2DDebugRenderer;

**import** com.badlogic.gdx.physics.box2d.CircleShape;

**import** com.badlogic.gdx.physics.box2d.EdgeShape;

**import** com.badlogic.gdx.physics.box2d.FixtureDef;

**import** com.badlogic.gdx.physics.box2d.PolygonShape;

**import** com.badlogic.gdx.physics.box2d.World;

**public** **class** Clase **extends** ApplicationAdapter {

SpriteBatch batch;

**private** World mundo;

**private** Box2DDebugRenderer debugRenderer;

**private** OrthographicCamera camaramundo;

**public** **static** **final** **int** ***SCREEN\_WIDTH*** = 800;

**public** **static** **final** **int** ***SCREEN\_HEIGHT*** = 600;

**public** **static** **float** *units* = 100f;

**private** Body cuerpo;

**private** Body piso;

**private** Body caja;

**public** **static** **final** **short** ***BIT\_PELOTA*** = 2;

**public** **static** **final** **short** ***BIT\_CAJA*** = 4;

**public** **static** **final** **short** ***BIT\_PISO*** = 8;

@Override

**public** **void** create () {

batch = **new** SpriteBatch();

Box2D.*init*();

mundo = **new** World(**new** Vector2(0, -9.8f), **true**);

debugRenderer = **new** Box2DDebugRenderer();

//crear la camara del mundo box2d

camaramundo = **new** OrthographicCamera();

camaramundo.setToOrtho(**false**, ***SCREEN\_WIDTH***/*units*, ***SCREEN\_HEIGHT***/*units*);

//OBJETO PISO

BodyDef cuerpoPiso = **new** BodyDef();

cuerpoPiso.type = BodyType.***StaticBody***;

//creamos una línea que representa el suelo, del ancho de la pantalla

EdgeShape linea = **new** EdgeShape();

linea.set(0, 20/*units*, ***SCREEN\_WIDTH***/*units*, 20/*units*);

//creamos el fixture

FixtureDef fixDef=**new** FixtureDef();

fixDef.shape=linea;

fixDef.filter.categoryBits = ***BIT\_PISO***;

fixDef.filter.maskBits = ***BIT\_PELOTA*** | ***BIT\_CAJA***;

//lo añadimos al mundo

piso = mundo.createBody(cuerpoPiso);

piso.createFixture(fixDef).setUserData("piso"); // añadimos el fixture al objeto

//PAREDES

cuerpoPiso.type = BodyType.***StaticBody***;

//creamos una línea que representa la pared izquierda, del alto de la pantalla

linea.set(0, 0, 0 , ***SCREEN\_HEIGHT***\*2/*units*);

//creamos el fixture

fixDef.shape=linea;

fixDef.filter.categoryBits = ***BIT\_PISO***;

fixDef.filter.maskBits = ***BIT\_PELOTA*** | ***BIT\_CAJA***;

//lo añadimos al mundo

piso = mundo.createBody(cuerpoPiso);

piso.createFixture(fixDef).setUserData("pared"); // añadimos el fixture al objeto

cuerpoPiso.type = BodyType.***StaticBody***;

//creamos una línea que representa la pared derecha, del alto de la pantalla

linea.set(***SCREEN\_WIDTH***/*units*, 0, ***SCREEN\_WIDTH***/*units* , ***SCREEN\_HEIGHT***\*2/*units*);

//creamos el fixture

fixDef.shape=linea;

fixDef.filter.categoryBits = ***BIT\_PISO***;

fixDef.filter.maskBits = ***BIT\_PELOTA*** | ***BIT\_CAJA***;

//lo añadimos al mundo

piso = mundo.createBody(cuerpoPiso);

piso.createFixture(fixDef).setUserData("pared"); // añadimos el fixture al objeto

//OBJETO CAJA GIRATORIA

BodyDef cuerpoCaja = **new** BodyDef();

cuerpoCaja.type = BodyType.***KinematicBody***;

cuerpoCaja.position.set(***SCREEN\_WIDTH***/2/*units*, ***SCREEN\_HEIGHT***/4/*units*);

//creamos un rectángulo que representa la caja giratoria

PolygonShape rectangulo = **new** PolygonShape();

rectangulo.setAsBox(100/*units*, 10/*units*);

//creamos el fixture

fixDef.shape=rectangulo;

fixDef.filter.categoryBits = ***BIT\_CAJA***;

fixDef.filter.maskBits = ***BIT\_PELOTA***;

//lo añadimos al mundo

caja = mundo.createBody(cuerpoCaja);

caja.createFixture(fixDef).setUserData("caja"); // añadimos el fixture al objeto

//es importante hacer dispose de los shapes despues de usarlos

rectangulo.dispose();

linea.dispose();

}

**private** **void** crearPelota() {

//OBJETO PELOTA

BodyDef cuerpoPelota = **new** BodyDef();

cuerpoPelota.position.set(***SCREEN\_WIDTH***/2/*units*, ***SCREEN\_HEIGHT***/*units*);

cuerpoPelota.type = BodyType.***DynamicBody***;

CircleShape circulo = **new** CircleShape();

circulo.setRadius(25/*units*); //un radio de 25 pixeles de la pantalla

FixtureDef fixDef=**new** FixtureDef();

fixDef.shape=circulo;

//Propiedades opcionales

fixDef.density = 30f;

fixDef.friction = 5f;

fixDef.restitution = 0.5f;

//propiedades mas opcionales

fixDef.filter.categoryBits = ***BIT\_PELOTA***;

fixDef.filter.maskBits = ***BIT\_PISO*** | ***BIT\_CAJA*** | ***BIT\_PELOTA***;

fixDef.isSensor = **false**;

cuerpo = mundo.createBody(cuerpoPelota);

cuerpo.createFixture(fixDef).setUserData("pelota"); // añadimos el fixture al objeto

//parámetros opcionales:

//cuerpo.setGravityScale(1); // escala de la gravedad que lo afecta

//cuerpo.isBullet(); //si es un cuerpo muy pequeño que viaja muy rápido

//es importante hacer dispose de los shapes despues de usarlos

circulo.dispose();

}

@Override

**public** **void** render () {

Gdx.*gl*.glClearColor(0, 0, 0, 1);

Gdx.*gl*.glClear(GL20.***GL\_COLOR\_BUFFER\_BIT***);

debugRenderer.render(mundo, camaramundo.combined);

//Movimiento rotatorio de la caja

caja.setAngularVelocity((**float**) Math.*toRadians*(360));

//cada vez que hacemos click, se crea una nueva pelota:

**if**(Gdx.*input*.justTouched()){

crearPelota();

}

mundo.step(1/60f, 8, 6);

}

}