1 LIBGDX Colisions3D

UNIDADE 4: Colisións en 3D

1.1 Sumario

1.2 Introdución

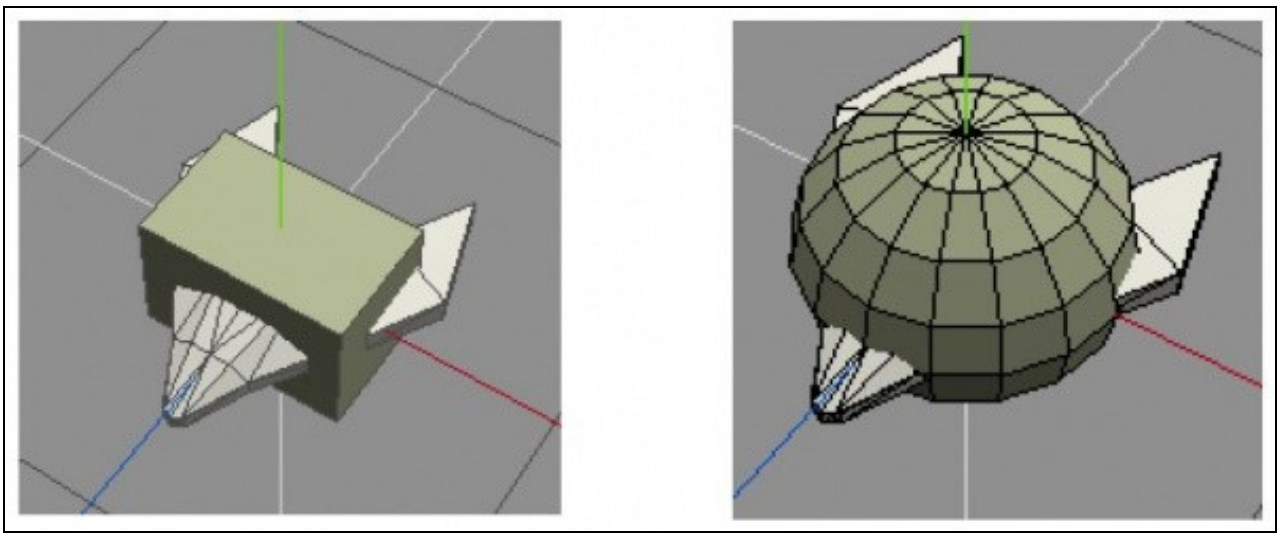
Clases utilizadas:

- Intersector
- Esfera
- BoundingBox.

Información na wiki: https://github.com/libgdx/libgdx/wiki/Circles%2C-planes%2C-rays%2C-etc.

En 3D temos varias formas de controlar cando unha figura 'choca' con outra.

- Envolver a figura cun prisma-cubo e comprobar se choca con outra figura.
- Envolver a figura cunha esfera e comprobar se esta choca con outra figura.
- Usar un raio: partir dun punto e cunha dirección ata o infinito, comprobar se dito raio atravesa (choca) con outra figura.



- O cubo-prisma denomínase BoundingBox e pódese obter a partires dun obxecto da clase Mesh ou ben crear nos un obxecto da clase BoundingBox e definir o seu tamaño e posición (con dous vectores (min-max) que conforman o volume)
- A esfera defínese cun centro (nunha posición no espazo 3D) e un radio.
- O uso de raios (parten dun punto e teñen unha dirección). Por exemplo, se prememos na pantalla podemos querer saber se estamos a tocar un obxecto que estaría en liña recta dende o punto ata o obxecto.

As colisións as temos que controlar usando a clase Intersector (no caso dos bounding box - rays), a propia clase BoundingBox, para controlar cando un BoundingBox choca con outro,chamando ó método intersects ou a clase Sphere que ten un método que indica cando se 'choca' con outra esfera

(método overLaps).

1.3 Clase BoundingBox

O proceso é o seguinte:

- Primeiro debemos de obter o BoundingBox do obxecto Mesh chamando ó método calculateBoundingBox.

```
Mesh cubo;
......

BoundingBox boundingBox = cubo.calculateBoundingBox()
```

- Unha vez o temos debemos de aplicarlle as mesmas transformacións que sufra o Mesh, chamando ó método mul.

```
..........
boundingBox.mul(matriz);
```

Sendo matriz un obxecto da clase Matrix4 que foi aplicada ó obxecto Mesh. Lembrar que xa vimos o uso de matrices neste punto do curso.

- Agora chamaremos ó método intersects enviando como parámetro o BoundingBox doutro obxecto a controlar:

```
..........
if (boundingBox1.intersects(boundingBox2)){
   // CHOCA
}
```

Nota: O cálculo para obter o BoundingBox asociado o Mesh (chamado ao método calculateBoundingBox()) é bastante custoso.

Se queremos aumentar o rendemento o lóxico é que gardemos unha copia do BoundingBox orixinal e outra copia para aplicar a matriz coas transformacións.

Así por cada Mesh do noso xogo crearíamos unha propiedade BoundingBox para gardar o BoundingBox orixinal (estaría situado na posición 0,0,0 sen transformación ningunha). Podería ser de clase (Static). Dependerá se dende a clase que representa o modelo no noso personaxe ten acceso a dita propiedade.

Neste exemplo estamos a supoñer que cargamos un Mesh en forma de Cubo dende a clase AssetsXogo:

```
public class AssetsXogo{
      .......
      public static BoundingBox BBoxCuboOrixinal;
      public staic void cargarMesh(){
      .........
      BBoxCuboOrixinal = meshCubo.calculateBoundingBox();
      }

      .....
}
```

Agora dende cada clase que faga uso deste Mesh crearemos un BoundingBox temporal ao que asinaremos o orixinal antes de aplicar as transformacións:

```
public class Inimigos {
      .........
      BoundingBox temporal;
      public Inimigos(){

         temporal = new BoundingBox();
      }
```

```
        public void update(float render){
           ..........
           temporal.set(AssetsXogo.BBoxCuboOrixinal);
           temporal.mul(matriz);
        }

    }
```

Sendo matriz un obxecto da clase Matrix4 que ten gardadas todas as transformacións a aplicar sobre o Inimigo (posición, escala e rotación).

1.4 Clase Intersector - Rays

O que facemos neste caso é calcular a dirección que ten un raio (vector unidade cunha dirección) e comprobamos se a proxección deste raio no espazo 3D choca cun BoundingBox.

O proceso é o seguinte:

- Primeiro temos que calcular o raio. Neste exemplo imos supoñer que o xogador preme na pantalla e que queremos controlar se premendo nese punto, a proxección dese punto cara 'dentro da pantalla' choca cun BoundingBox.

Para obter o raio temos que facer uso do método getPickRay da cámara en perspectiva. Normalmente dito control o faremos no método touchDown que se produce cando prememos na pantalla.

```
@Override
public boolean touchDown(int screenX, int screenY, int pointer, int button) {

    Ray ray = camara3d.getPickRay(screenX, screenY, 0, 0, Gdx.graphics.getWidth(),Gdx.graphics.getHeight());

   ......
}
```

O método está sobrecargado polo que temos varias opcións. Nesta enviamos as coordenadas en pixeles da pantalla e o tamaño do mesma estando a posición 0,0 na parte inferior esquerda da pantalla.

- Unha vez temos o raio podemos chamar ó método intersectRayBoundsFast da clase Intersector que nos indica se o raio choca cun BoundingBox.

```
if (Intersector.intersectRayBoundsFast(ray, boundingBox)){
                // CHOCA
        }
```

1.5 Clase Sphere

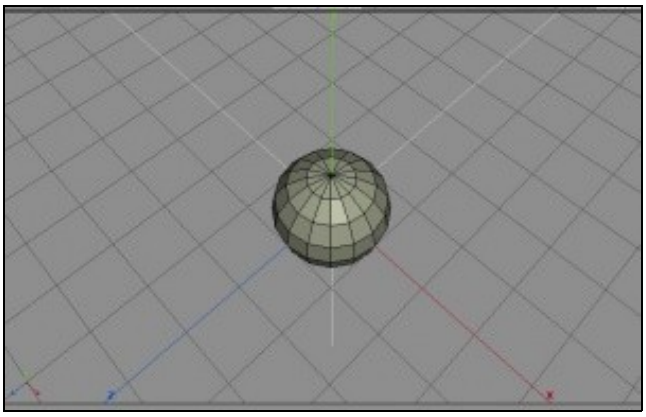
Neste caso temos que asociar a cada obxecto Mesh un obxecto da clase Sphere.

Unha esfera está definida pola posición e radio.

Normalmente nos xogos o tamaño da esfera asociada ó obxecto Mesh non vai variar e o que temos que facer é actualizar a posición en función da posición do obxecto Mesh.

Se estamos a desenvolver o xogo seguindo o patrón Model-View-Controller como xa vimos no xogo 2D teremos que definir a esfera como unha propiedade máis dentro da clase. Instanciala no constructor e movela no método update da clase.

- O tamaño: o tamaño ten que ven co tamaño que teña o obxecto no programa 3D de deseño. Así, no seguinte gráfico podemos ver como a esfera que representa a Terra ten un tamaño de dúas unidades (2x2x2)



Neste exemplo o radio da esfera é de 1 unidade e ten un diámetro de 2 unidades:

diámetro = radio x 2

Agora ben, cando eu defino unha figura a podo escalar. Entón a esfera teremos que aplicarlle a escala tamén.

Así, se instancio a terra cunha escala de 25 significa o seguinte:

diámetro orixinal sen escalar = 2 Unidades
diámetro escalado = 25 x 2 = 50 Unidades

Polo tanto eu terei que definir unha esfera de 50 unidades de **diámetro** e **polo tanto de 25 Unidade de Radio**.

Isto o teremos que facer unha vez (pode ser no constructor) se durante o xogo o tamaño da figura 3D non ten que variar.

O proceso polo tanto sería:

- Definimos a esfera:

```
public Sphere esfera;
```

- Instanciamos a esfera e lle damos unha posición igual á do obxecto a controlar e unha escala de acordo a escala do obxecto a controlar. No caso da Terra no noso exemplo:

```
esfera = new Sphere(pos, escala);
```

- Actualizamos a esfera coa posición no obxecto a controlar. Debemos chamar á propiedade center:

```
esfera.center.set(posicion);
```

- Agora queda controlar cando a esfera 'choca' con outra esfera. Para iso temos que chamar ó método overlaps da clase Sphere:

```
if (esfera.overlaps(outraesfera) {
                    // CHOCA
}
```

TAREFA 4.6 A FACER: Esta parte está asociada á realización dunha tarefa.

-- Ángel D. Fernández González -- (2014).