FAQ

Sistemas Gráficos

Grado en Ingeniería Informática

Curso 2019/2020

Preguntas

¿Cómo puedo tener un modelo cargado de un archivo y que detecte colisiones mediante física?

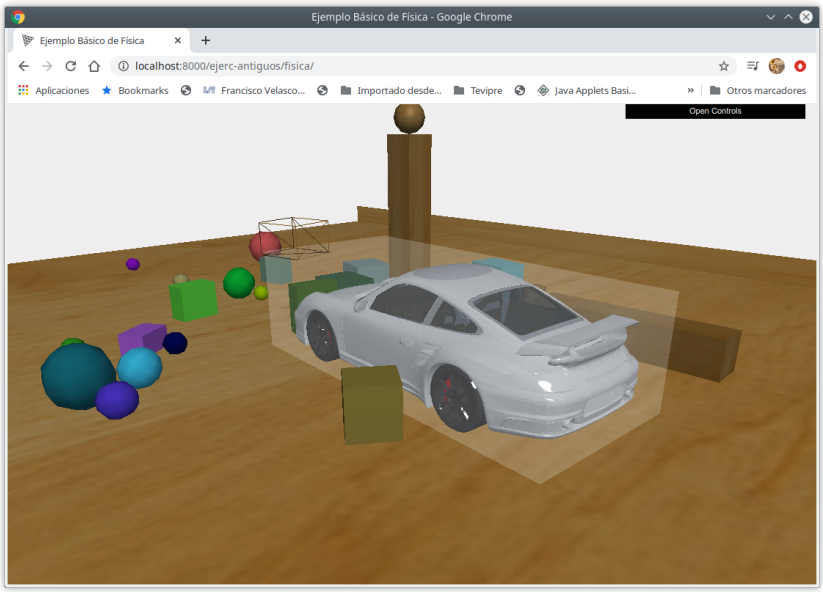
¿Cómo puedo tener un modelo jerárquico, por ejemplo, un personaje que mueve las piernas como si caminara, y que detecte colisiones mediante física?

1. Respuesta

La solución en ambos casos es tener una caja física invisible que colisiona y le afecta la física lo mismo que a cualquier objeto físico, y "dentro" contiene el modelo que no es físico, como el personaje que mueve las piernas, que se manipula normalmente.

Vamos, lo que viene siendo un *collider* en un motor de juegos.

La siguiente imagen muestra un ejemplo con un modelo cargado desde disco.



La caja que engloba al coche es física. La he dejado un poco visible en el ejemplo para que se vea mejor lo que estoy explicando. Pero habría que dejarla invisible.

Ojo. Cuando digo invisible, NO me refiero a poner el atributo `visible = false;`. Recordad que con eso estamos diciendo que NO SE TENGA EN CUENTA PARA NADA, y como efecto secundario no se ve, pero no es eso lo que queremos.

Para que la caja sea tenida en cuenta, por ejemplo, con la física y al mismo tiempo que no se vea, en el material de la caja hay que ponerle los atributos `transparent = true`, y `opacity = 0.0`.

En código, ¿cómo sería?

```
// Hacemos la geometría del collider, una caja con las dimensiones correctas,
//   lo más ajustadas posible a lo que va a contener

var geometriaCollider = new THREE.BoxGeometry (dimensionX, dimensionY, dimensionZ);

// Creamos el material invisible

var matInvisible = new THREE.MeshBasicMaterial ({transparent:true, opacity:0});

// Creamos el material físico, con propiedades de rozamiento y rebote

var matFisico = Physijs.createMaterial (matFisico, unRozamiento, unRebote);

// Creamos el collider físico dándole una masa

var collider = new Physijs.BoxMesh (geometriaCollider, matFisico, unaMasa);

// Tenemos nuestro modelo, cargado de disco o hecho por nosotros

var modelo = ... // el que sea
```

```
// Ya solo nos queda colgar el modelo de su collider físico

collider.add (modelo);

// Y colgar el collider de la raíz de la escena

raizEscena.add (collider);
```

Cuando yo quiera desplazar el personaje, lo que hago es desplazar el collider, bien modificando la .position y poniendo el __dirtyPosition = true; o bien aplicándole impulsos (página 9 de los apuntes de física.)

Al collider le afecta la física, pero como es invisible, parece que la física le está afectando a lo de dentro.

En el ejemplo del coche, si hago avanzar la caja que lo contiene, el coche que siempre está dentro de la caja se verá avanzar y parecerá que es el coche el que empuja a las cajas.

Y si mientras el collider del coche avanza, yo pudiera hacer que las ruedas girasen con alguna animación, parece que el coche avanza porque las ruedas giran, aunque sea todo simulado.

De igual modo, que si el collider de un personaje se desplaza y al mismo tiempo animo las piernas como si caminara, como el collider no se ve, parece que el personaje se está desplazando porque está moviendo las piernas.

Un último comentario

Dejar el collider un poco visible mientras ajustáis tamaños y posiciones. Ponerlo invisible solo cuando ya hayáis comprobado que las dimensiones de la caja y la posición del modelo con respecto a la caja son correctas.

El último fragmento de código sirve para calcular las dimensiones correctas de la caja automáticamente.

```
var modelo =  ... // el que sea

// Obtenemos una caja englobante del modelo

var bounding = new THREE.BoxHelper(modelo);


// Calculamos sus dimensiones y las usamos para hacer la geometría del collider

bounding.geometry.computeBoundingBox();
var bb = bounding.geometry.boundingBox;
var geometriaCollider =
  new THREE.BoxGeometry(bb.max.x-bb.min.x, bb.max.y-bb.min.y, bb.max.z-bb.min.z);

// Lo demás ya sería igual que el ejemplo de antes.
```