



Interacción Detección de Colisiones

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática
Curso 2021-2022

Contenidos

1 Introducción

2 Entrada/Salida de información por parte del usuario

- Mensajes en pantalla
- Órdenes mediante teclado
- Interacción con el ratón en la escena

3 Selección de objetos (Picking)

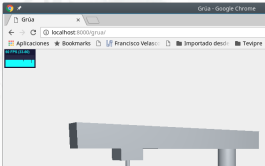
4 Detección de colisiones

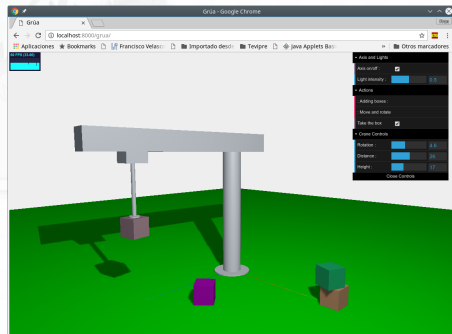
- Indexación espacial de la escena

Objetivos

- Conocer las técnicas para que el usuario interactúe con la escena
 - ▶ Mediante el ratón
 - ▶ Mediante el teclado
- Saber seleccionar objetos de la escena con el ratón: Picking
- Conocer técnicas básicas de detección de colisiones

Introducción

- El usuario puede actuar con la escena:
 - ▶ Seleccionando objetos o posiciones (**Picking**)
 - ▶ Modificando objetos (su posición, orientación, forma)
 - ▶ Modificando la cámara (es un objeto más)
 - Los objetos pueden interactuar entre ellos (**Colisiones**)
 - Implica realizar **búsquedas** en la escena
 - ▶ Más rápidas si se tiene la **escena indexada**
- 
- A screenshot of a web browser window titled 'Grua - Google Chrome'. The address bar shows 'localhost:8000/guara'. The browser's developer tools are open, showing the 'Elements' panel with a selected blue cube element. The main content area displays a 3D scene with a blue cube and a grey rectangular prism on a light grey surface.



Mensajes en pantalla

Salida HTML

- Se puede tener en el html una zona para escribir mensajes

HTML: Zona para mostrar mensajes

```
<div style="position:absolute; left:100px; top:10px" id="Messages">  
</div>
```

- Y enviarle texto (formateado) desde Javascript

Javascript: Método para escribir en la zona anterior

```
setMessage (str) {  
    document.getElementById ("Messages").innerHTML = "<h2>"+str+"</h2>";  
}
```

Ventanas pop-up → detienen ejecución del programa

- Se muestran con `window.alert ("Texto")`

Mensajes: Ventana pop-up

```
window.alert ("Hola Mundo!\nPulsa Aceptar para continuar.");
```

¡Hola Mundo!
Pulsa Aceptar para continuar.

Aceptar

Procesamiento de pulsaciones de teclado

- Se definen **métodos** asociados a **eventos** del teclado
 - ▶ **keydown**: Se pulsa una tecla
 - ▶ **keyup**: Se suelta
 - ▶ **keypress**: Pulsación y suelta. *no sirve cuando queremos que se pulsen a la vez*
Este evento no lo generan las teclas modificadoras.
↳ ctrl, shift, ...
- En el *main* se añaden los *listener* y se indican los métodos que se ejecutarán cuando se produzca cada evento

Listener: Ejemplo

```
window.addEventListener ("keydown", (event) => scene.onKeyDown(event));  
                        evento                      método
```

Procesamiento de pulsaciones de teclado

- En dichos métodos, mediante del parámetro recibido (event), se puede consultar la tecla concreta que produjo el evento a través de sus atributos which o key → *depende del navegador*

- ▶ `var x = event.which || event.key`

- ★ Así, la lectura del código asociado a dicha tecla funciona en todos los navegadores

- ▶ Mediante `import * as KeyCode from 'keycode.esm.js'`

- ★ Se puede saber si se ha pulsado una tecla no imprimible
- ★ Ejemplo: `if (x == KeyCode.KEY_CONTROL)`

- ▶ Para saber si se ha pulsado un carácter imprimible

- ★ Ejemplo: `if (String.fromCharCode (x) == 'A')`

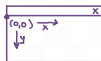
Interacción con el ratón en la escena

- Se definen **métodos** asociados a determinados **eventos** del ratón
- Se definen **estados** que indican **qué se está haciendo** con la aplicación en cada momento
- Cada método que procese un evento del ratón
 - ▶ Debe consultar el estado actual de la aplicación
 - ▶ O si hay alguna tecla pulsada a la vez (ejemplo Ctrl + clic)
 - ▶ Realizar el procesamiento correcto

Por ejemplo, hacer un clic y arrastrar el ratón puede ser:

 - ★ Realizar un movimiento de cámara
 - ★ Añadir un objeto a la escena en una posición
 - ★ Seleccionar y mover un objeto existente
 - ★ *Cualquier otra cosa . . .*

Eventos del ratón que pueden escucharse



- Entre otros, se pueden escuchar los siguientes eventos
 - ▶ mousedown mouseup mousemove wheel
- En el *main* se añaden los *listener* y se indican los métodos que se ejecutarán cuando se produzca cada evento

Listener: Ejemplo

```
window.addEventListener ("mousemove", (event) => scene.onMouseMove(event));  
                           evento                método
```

- Valores asociados al evento que se pueden consultar
 - ▶ clientX: La coordenada X del ratón (relativa a la ventana)
 - ▶ clientY: La coordenada Y
 - ▶ which: El botón concreto que se ha pulsado
 - ★ 0 (ninguno), 1 (izquierdo), 2 (central), 3 (derecho)

Ratón junto a una tecla modificadora

- Los eventos del ratón pueden realizar distintas acciones si se producen estando pulsada una o varias teclas modificadoras
- En la función que procesa un evento del ratón se puede consultar el estado de dichas teclas para realizar un procesamiento u otro
- Teclas modificadoras que pueden consultarse

ctrlKey altKey shiftKey

Ejemplo: Diferente funcionalidad dependiendo de Ctrl

```
function onMouseDown (event) {  
  if (event.ctrlKey) {  
    // Se realizan unas acciones ...  
  } else {  
    // ... u otras  
  }  
}
```

Estados de la aplicación

- Un atributo en la escena indica qué se está haciendo
- Se puede establecer al elegir una opción del menú
- Se consulta desde los métodos que procesan los eventos del ratón para determinar el procesamiento a realizar

Ejemplo: Definición y usos de estados de aplicación

```
// Se definen (normalmente) como constantes numéricas
MyScene.NO_ACTION = 0;
MyScene.ADDING_BOXES = 1;
MyScene.MOVING_BOXES = 2;

// Se usan para darle valor al atributo de estado de la aplicación
this.applicationMode = MyScene.NO_ACTION;

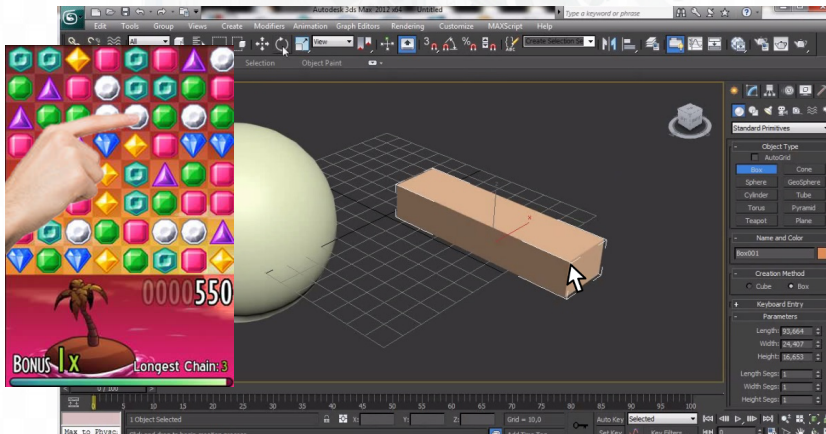
// Se consulta al procesar un evento del ratón
onMouseDown (event) {
    switch (this.applicationMode) {
        case MyScene.ADDING_BOXES :
            // procesamiento para mouseDown y ADDING_BOXES
```

Selección de objetos

● Picking

Seleccionar un elemento de la escena con un puntero

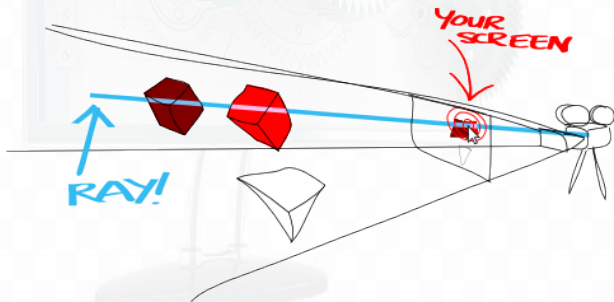
- Suele ser una operación habitual en muchas aplicaciones gráficas



Selección de objetos (Picking)

Proceso a realizar

- 1 Saber en qué píxel se ha hecho clic
- 2 Lanzar un rayo
 - ▶ Desde la cámara
 - ▶ Que pase por dicho píxel
- 3 Obtener los objetos alcanzados por ese rayo
Normalmente el seleccionado es el más cercano



Picking

Three.js

Ejemplo: Picking

```
onDocumentMouseDown (event) {  
    // Se obtiene la posición del clic  
    // en coordenadas de dispositivo normalizado  
    // - La esquina inferior izquierda tiene la coordenada (-1,-1)  
    // - La esquina superior derecha tiene la coordenada (1,1)  
    var mouse = new THREE.Vector2 ();  
    mouse.x = (event.clientX / window.innerWidth) * 2 - 1;  
    mouse.y = 1 - 2 * (event.clientY / window.innerHeight);  
  
    // Se construye un rayo que parte de la cámara (el ojo del usuario)  
    // y que pasa por la posición donde se ha hecho clic  
    var raycaster = new THREE.Raycaster ();  
    raycaster.setFromCamera (mouse, camera);  
  
    // Hay que buscar qué objetos intersecan con el rayo  
    // Es una operación costosa, solo se buscan intersecciones  
    // con los objetos que interesan en cada momento  
    // Las referencias de dichos objetos se guardan en un array
```

Picking

(continuación)

Ejemplo: Picking

```
// pickableObjects es el array de objetos donde se van a buscar intersecciones con el
// rayo
// Los objetos alcanzados por el rayo, entre los seleccionables,
// se devuelven en otro array
var pickedObjects = raycaster.intersectObjects
    (pickableObjects, true);

// pickedObjects es un vector ordenado desde el objeto más cercano
if (pickedObjects.length > 0) {
    // Se puede referenciar el Mesh clicado
    selectedObject = pickedObjects[0].object;
    // E incluso el punto concreto donde se ha hecho clic
    selectedPoint = new THREE.Vector3 (pickedObjects[0].point);
    ...
}
```


Selección del objeto global

Uso del atributo `userData`

- Pick devuelve el Mesh 'clicado'
- Se puede desear acceder a la raíz del árbol de la figura
- Se usa el atributo `userData` de Mesh
 - ▶ En cada Mesh se hace que `userData` apunte a la raíz
 - ▶ Tras el Pick, se accede a la raíz mediante `userData`



Selección del objeto global

Uso del atributo `userData`

Ejemplo: Uso del atributo `userData` al construir

```
class DarthVader extends THREE.Object3D {
  constructor() {
    . . .
    this.cabeza = new THREE.Mesh ( . . . );
    this.cabeza.userData = this;
    . . .
  }

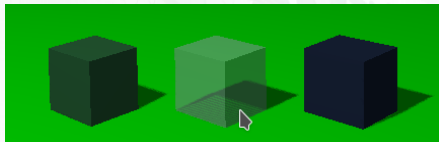
  caminar() {
    . . .
  }
  . . .
}
```

Ejemplo: Uso del atributo `userData` al hacer clic

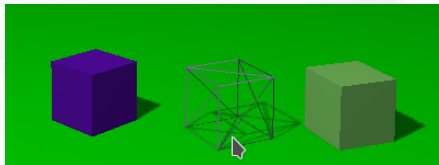
```
var meshClicado = pickedObjects[0].object;
meshClicado.userData.caminar();
```

Feedback

- El objeto concreto seleccionado debe indicarse al usuario
- Se realiza con un cambio en su aspecto
 - ▶ Transparencias, cambio de color, modo alambre, etc.



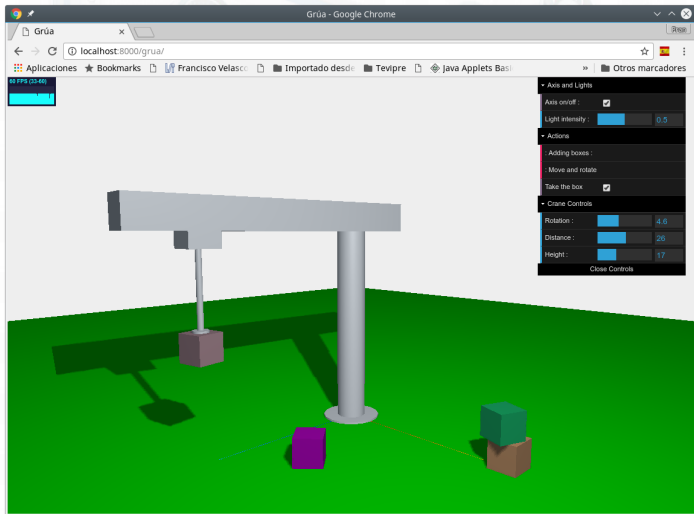
Atributo del material `opacity = 0.5` y `transparent = true`



Atributo del material `wireframe = true`

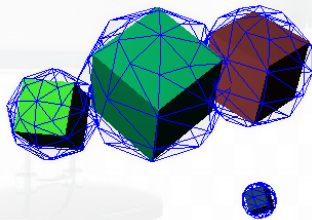
*children
↳ hijos
del nodo*

Ejemplo que incluye estas técnicas



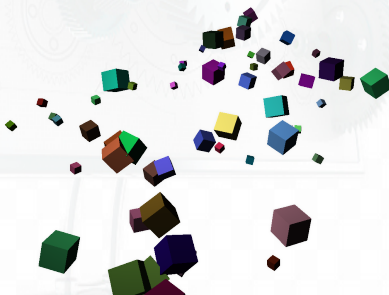
Detección de colisiones

- Suele ser necesario saber cuándo 2 objetos colisionan
- La detección de colisiones se realiza en dos fases
 - ▶ **Fase gruesa:**
Se descartan rápidamente los elementos que no colisionan
 - ▶ **Fase fina:**
Se determina con exactitud si 2 elementos están colisionando
- En la fase gruesa se usa:
 - ▶ **Indexación espacial**
 - ▶ **Cajas o esferas englobantes**



Indexación espacial de la escena

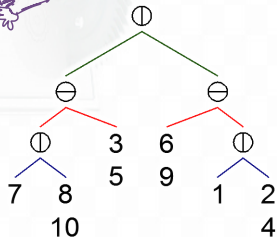
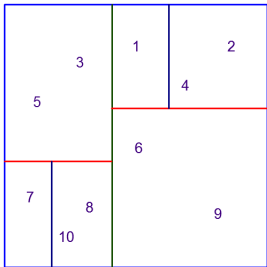
- Cuando se tienen muchos objetos en la escena es importante saber indentificarlos con rapidez
 - ▶ Cuando se lanzan rayos para Ray Tracing o Picking
 - ▶ Cuando se buscan colisiones entre objetos
- Para ello se usan estructuras de descomposición espacial



Estructuras para indexación espacial

KD-Trees

- El espacio se subdivide en 2 semiespacios por un plano
 - ▶ Solo si tiene más elementos que un límite
 - ▶ El plano está alineado con los ejes *L en ejemplo*
 - ▶ Situado de manera que quede un **árbol balanceado**
 - ▶ En cada nivel se cambia el eje de división, alternativamente $X \rightarrow Y \rightarrow Z \rightarrow X \dots$

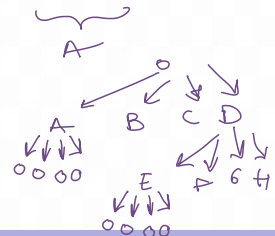
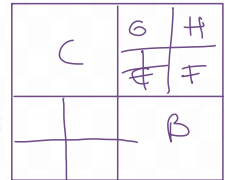
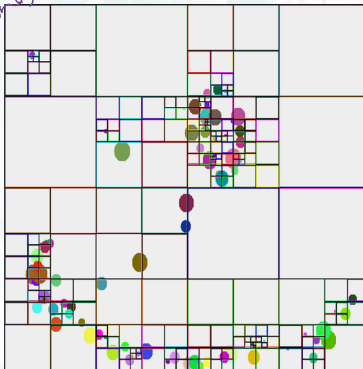


Estructuras para indexación espacial

Octrees

- El espacio se subdivide jerárquicamente en octantes
- Un octante solo se subdivide si contiene más objetos que el límite establecido

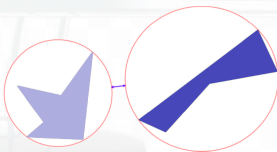
*se busca que
queden
equilibrados
divisiones*



Cajas y esferas englobantes

- Formas sencillas que engloban completamente al objeto
- Permiten saber rápidamente cuando 2 objetos no colisionan
- Según la precisión exigida, se usan para determinar la colisión

Ejercicio: Diseñar sendos algoritmos para calcular la caja y la esfera englobante de un objeto cualquiera a partir de su lista de vértices



si 2 cajas
englobantes
no intersecan, lo
de dentro no.
si intersecan
lo de dentro



Interacción Detección de Colisiones

Francisco Velasco Anguita

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Granada

Sistemas Gráficos

Grado en Ingeniería Informática
Curso 2021-2022