Informática Gráfica
Selección

Juan Carlos Torres
Grupos C y D

Dpt. Lenguajes y Sistemas Informáticos
ETSI Informática y de Telecomunicación
Universidad de Granada

—

Curso 2024-25

Sistemas interactivos

Un sistema gráfico interactivo es un conjunto de hardware y software que permite a los usuarios crear, manipular y visualizar modelos, respondiendo de manera **fluida** a las entradas del usuario.

El sistema funciona según el siguiente ciclo:

- El usuario realiza una acción (*Evento*).
- El sistema llama a una función del programa, diseñada para responder al evento. Esta función (*CallBack*) recibe la información asociada a la acción y responde a la acción.
- Si es necesario se redibuja el modelo

Por ejemplo, para borrar un componente de la escena:

- El usuario pulsa un botón del ratón con el cursor sobre un modelo.
- El callback asociado con el ratón recibe el evento y responde a la acción borrando el componente.
- Se redibuja el modelo.

Interactividad

La incorporación de interactividad permite realizar aplicaciones que respondan a las acciones de los usuarios.

La mayor parte de los sistemas gráficos son interactivos.

La interactividad es esencial en

- Aplicaciones de diseño
- Juegos
- Modeladores
- Simuladores

Respuesta fluida

La interactividad requiere que el retardo (**latencia**) entre la acción del usuario y la respuesta del sistema sea suficientemente pequeño para que el usuario perciba una relación de causa efecto entre su acción y la respuesta del sistema.

Las restricciones de tiempo son menos estrictas que las de un sistema de tiempo real. En un sistema de tiempo real el retardo debe ser menor que un valor prefijado.

La latencia es el tiempo que transcurre entre la acción del usuario y la respuesta (normalmente la respuesta es la actualización de la imagen).

Realimentación

La realimentación es el mecanismo mediante el cual el sistema da información útil al usuario para determinar la siguiente acción a realizar. La información de realimentación que el sistema genera en cada momento depende lógicamente del estado del sistema y de la información previamente entrada por el usuario.

Se puede usar con diferentes fines:

- Mostrar el estado del sistema.
- Como parte de una función de entrada.
- Para reducir la incertidumbre del usuario.

Por ejemplo, al dibujar líneas, una vez introducido el primer punto, se puede mostrar la línea que se crearía en la posición actual del cursor mientras éste se mueve.

Técnicas

Con frecuencia, el usuario debe introducir una posición que satisfaga una determinada condición, por ejemplo tener la misma coordenada X que la posición previa.

En estas situaciones, la aplicación puede ayudar al usuario a realizar su tarea proporcionándole técnicas de interacción especificas que garanticen que se cumplen las restricciones de entrada.

En el caso anterior, puede hacer que el cursor solo se mueva en horizontal.

Por ejemplo, al crear elementos en un escenario el sistema puede hacer que se ajuste su altitud para que apoye en el elemento que este en esa posición.

Funciones

Un sistema gráfico interactivo necesita normalmente funciones de entrada para leer texto y valores numéricos como cualquier sistema informático, pero necesita además poder:

- Leer posiciones
- Seleccionar componentes del modelo geométrico
- Introducir transformaciones geométricas

La lectura de posiciones (posicionamiento) permite generar en coordenadas en el sistema de coordenadas de la escena.

La selección permite identificar elementos del modelo geométrico.

Componentes de un sistema gráfico interactivo

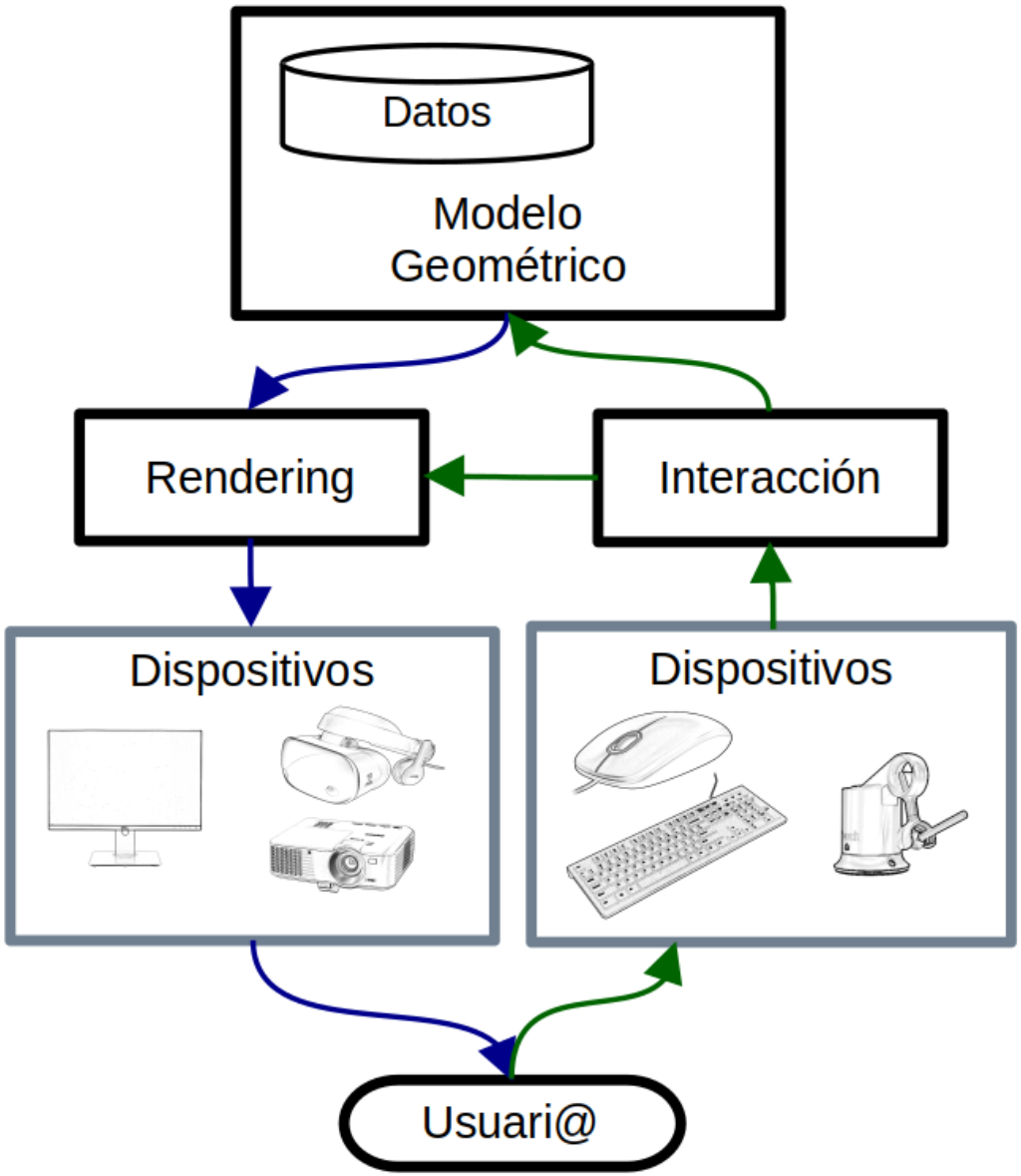
Hardware

- Dispositivos de visualización: Se utilizan para mostrar la escena. Suelen ser monitores, pero también pueden ser otros equipos, como sistemas de proyección o gafas de realidad virtual.
- Dispositivos de entrada: Permiten la interacción del usuario y pueden incluir dispositivos más allá del ratón y teclado, como controladores táctiles, sensores de posición, dispositivos hápticos u otros dispositivos.

Software

- Modelo geométrico: Contiene la representación de la escena, que podrá ser modificada según las operaciones de entrada recibidas.
- Motor de renderizado: Genera las imágenes a partir del modelo geométrico.
- Subsistema de interacción: Se encarga de interpretar y procesar las acciones realizadas por el usuario.

Sistema de gestión de entradas



Dispositivos lógicos

La dependencia de la aplicación con el hardware de entrada se resuelve, en parte, trabajando con dispositivos abstractos, denominados dispositivos lógicos. Cada dispositivo lógico representa un dispositivo físico ideal, que genera un único tipo de información de entrada. Cada dispositivo físico puede ser tratado como una realización de uno, o varios, dispositivos lógicos.

Dispositivos lógicos:

- **Locator:** Lee una posición.
- **pick:** Lee el identificador de un objeto.
- **stroke:** Lee una lista de posiciones.
- **string:** Lee una cadena de caracteres.
- **choice:** Lee uno de entre n items.
- **valuator:** Lee un número.

Modos de entrada

Otro de los problemas relacionados con las operaciones de entrada es el de sincronizar las acciones del usuario con la aplicación. La sincronización debe realizarse de forma distinta dependiendo de la función a realizar.

Hay tres modos de funcionamiento:

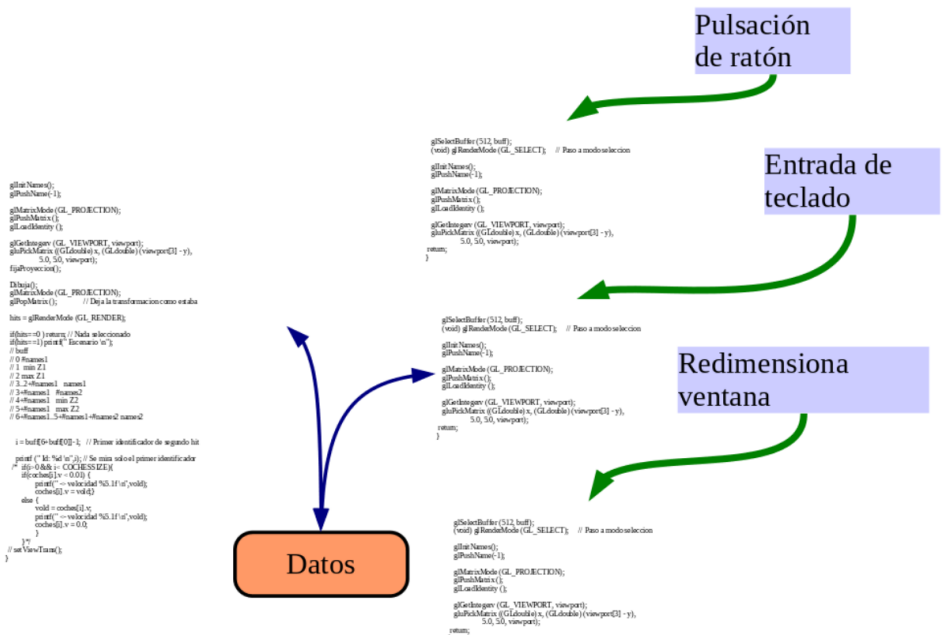
- **Petición:** El programa se detiene hasta que se completa la lectura.
- **Evento:** Cada acción genera un evento que son atendidos por funciones del programa (callbacks).
- **Muestreo:** El dispositivo tiene asociado un valor que el programa puede consultar.

Interacción en modo evento

En modo evento la aplicación tiene funciones especiales (callback) que se ejecutan cuando se produce un evento. Cada tipo de evento tiene un callback asociado.

No hay operaciones de lectura. Los eventos sin callback se ignoran. Los distintos callbacks se ejecutan de forma concurrente.

Si se produce un evento mientras se está gestionando un evento anterior el nuevo evento se encola en una cola FIFO y se procesa cuando el callback se quede libre.

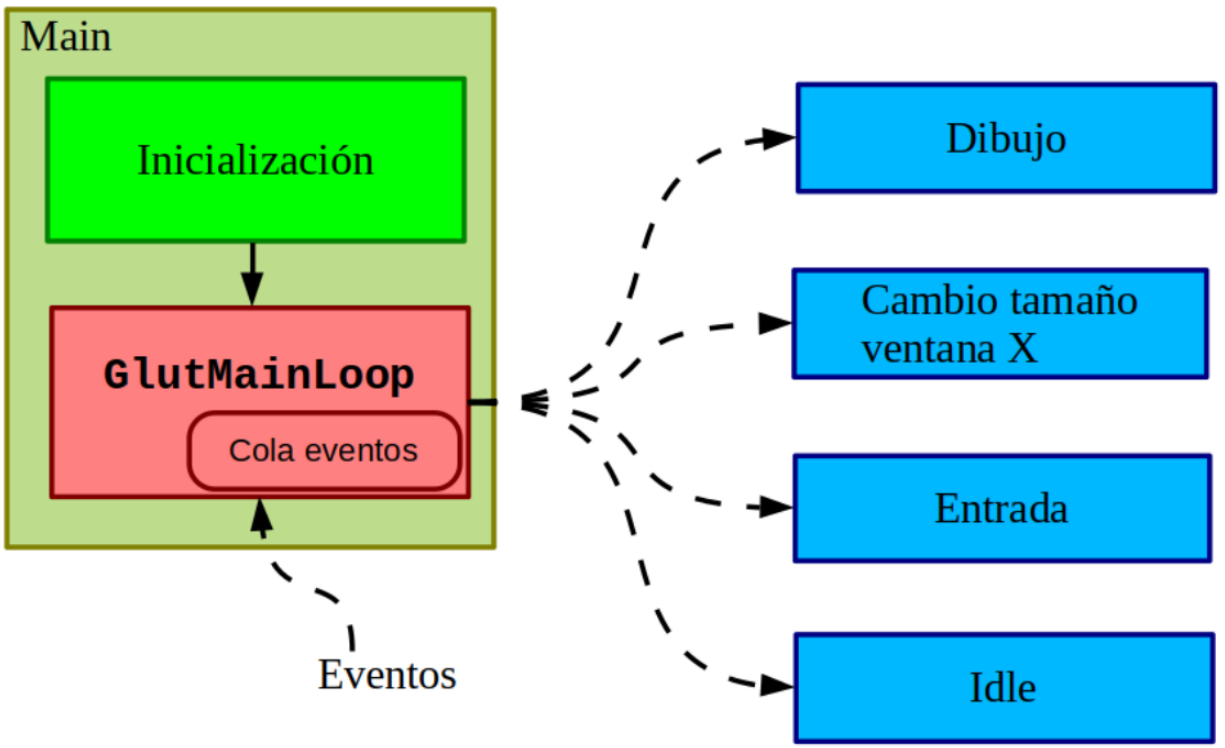


Eventos en GLUT

GLUT (OpenGL Utility Toolkit) es una biblioteca de utilidades para OpenGL que gestiona funciones de entrada/salida, además de gestión de ventanas y dibujo de primitivas.

El ciclo de gestión de eventos se activa con la llamada a **glutMainLoop()**

Todo el código (salvo la inicialización) se ejecuta en callbacks.



Eventos en GLUT

Cada calback tiene asociado una función de registro. GLUT incluye, entre otros, funciones para gestionar eventos de:

- Cambio de tamaño de ventana.
```
void glutReshapeFunc(void (*func)(int width, int height));
```
- Pulsación de letras
```
void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));
```
- Entrada de caracteres especiales
```
void glutSpecialFunc(void (*func)(int key, int x, int y));
```
- Pulsación de ratón
```
void glutMouseFunc(void (*func)(int button, int state, int x, int y));
```
- Movimiento de ratón
```
void glutMotionFunc(void (*func)(int x, int y));
void glutPassiveMotionFunc(void (*func)(int x, int y));
```
- Ausencia de eventos pendientes de procesar (idle)
```
void glutIdleFunc(void (*func)(void));
```
- Eventos cronometrados
```
void glutTimerFunc(unsigned int msecs, void (*func)(int value), value);
```
- Redibujado
```
void glutDisplayFunc(void (*func)(void));
```

☞ https://www.opengl.org/resources/libraries/glut/spec3/node45.html

GLUT: redibujado

El dibujo de la visualización se realiza como respuesta a un evento interno (lanzado por el propio programa llamando a la función *glutPostRedisplay*())

```
int main( int argc, char *argv[] ){
  ....
  glutDisplayFunc( dibujar );
  glutMouseFunc( clickRaton );
  ...
 }

void clickRaton( int boton, int estado, int x, int y ) {
  ...
  glutPostRedisplay();
  }

void dibujar() {
  ....
  glutSwapBuffers(); // Intercambia framebuffers
}
```

GLUT: ratón

Hay varios callbacks asociados a los eventos de ratón: pulsación de botones, movimiento con botón pulsado y movimiento pasivo. El callback de detección pulsación de botones se realiza con la función:

```
void glutMouseFunc(void (*func)(int button, int state, int x, int y));
```

en el que *button* es el botón pulsado, que puede ser:

- **GLUT_LEFT_BUTTON**
- **GLUT_MIDDLE_BUTTON**
- **GLUT_RIGHT_BUTTON**

El segundo argumento *state* indica el estado del botón

- **GLUT_UP**
- **GLUT_DOWN**

La posición del cursor en coordenadas de pantalla es (x,y).

GLUT: ratón

El call *MotionFunc* se llama cuando se mueve el ratón mientras tiene algún botón pulsado y el *PassiveMotionFunc* cuando se mueve sin pulsar botones:

```
void glutMotionFunc(void (*func)(int x, int y));

void glutPassiveMotionFunc(void (*func)(int x, int y));
```

Ambas funciones reciben solamente la posición del cursor. Si necesitamos saber que botón está pulsado para procesar *MotinFunc* es necesario almacenar el estado del ratón en variables del programa.

GLUT: ratón

```
int main( int argc, char *argv[] ){
   ....
   glutMouseFunc( clickRaton );
   glutMotionFunc( RatonMovido );
   ...
}

void clickRaton( int boton, int estado, int x, int y ){
   if(boton== GLUT_LEFT_BUTTON && estado == GLUT_DOWN) {
   Xref=x;
   Yref=y;
   ...
   }
}

void RatonMovido( int x, int y ) {
   ...
   vofy -= (Yref -y) / 100.0;
   vofx += (Xref -x) / 100.0;
   ...
}
```

GLUT: menús

Glut permite definir menús emergentes.La funcón *glutCreateMenu* crea un nuevo menú, le asocia un callbacl y devuelve el id asociado al menú.

```
int glutCreateMenu(void (*func)(int value));
```

La función *glutAddMenuEntry* añade entradas al menú. Recibe el texto de la opción del menú y el valor que se devolverá cuando se seleccione.

```
void glutAddMenuEntry(char *name, int value);
```

La función glutAttachMenu asocia el menú emergente a uno de los botones del ratón.

```
void glutAttachMenu(int button);
```

☞ https://www.opengl.org/resources/libraries/glut/spec3/node35.html

GLUT: menús

```
int     menu;
typedef enum{ START,SALIR,...,PAUSE} opciones_menu;

void Init(){
  menu = glutCreateMenu( seleccionMenu );
  glutAddMenuEntry( "Start", START );
  ...
  glutAddMenuEntry( "Exit", SALIR );
  glutAttachMenu( GLUT_RIGHT_BUTTON );
}

void seleccionMenu( int opcion ) {
  switch ( opcion ) {
    case START:
    ...
    break;
    case PAUSE:
    ...
    case SALIR:
    exit(0);
    }
  glutPostRedisplay();
}
```

Interacción con Unity 3D

Unity permite consultar el estado de los dispositivos de entrada en cada frame. Los scripts asociados a los objetos pueden tener un métodos *update* que se ejecuta en cada frame y puede consultarlos mediante métodos de la clase *Input*:

Input.GetMouseButtonDown(int button): Devuelve true en el frame en el que se presiona un botón del ratón.

Input.GetMouseButton(int button): Devuelve true mientras el botón del ratón está pulsado.

Input.GetMouseButtonUp(int button): Devuelve true en el frame en el que se libera el botón del ratón.

Input.mousePosition: Devuelve la posición del cursor del ratón en la pantalla, en coordenadas de ventana.

Input.getAxis: Dispositivo lógico que devuelve el desplazamiento en un eje.

Los mismos eventos se pueden consultar desde diferentes objetos.

Interacción con Unity 3D

Además, los script asociados a los objetos pueden también consultar la situación del ratón respecto al objeto:

OnMouseDown: se ha presionado el botón del ratón mientras el puntero está sobre el objeto.

OnMouseUp: se ha liberado el botón del ratón cuando el puntero está sobre el objeto.

OnMouseEnter: El puntero del ratón ha entrado en el área del objeto.

OnMouseExit: Se llama cuando el puntero del ratón sale del área del objeto.

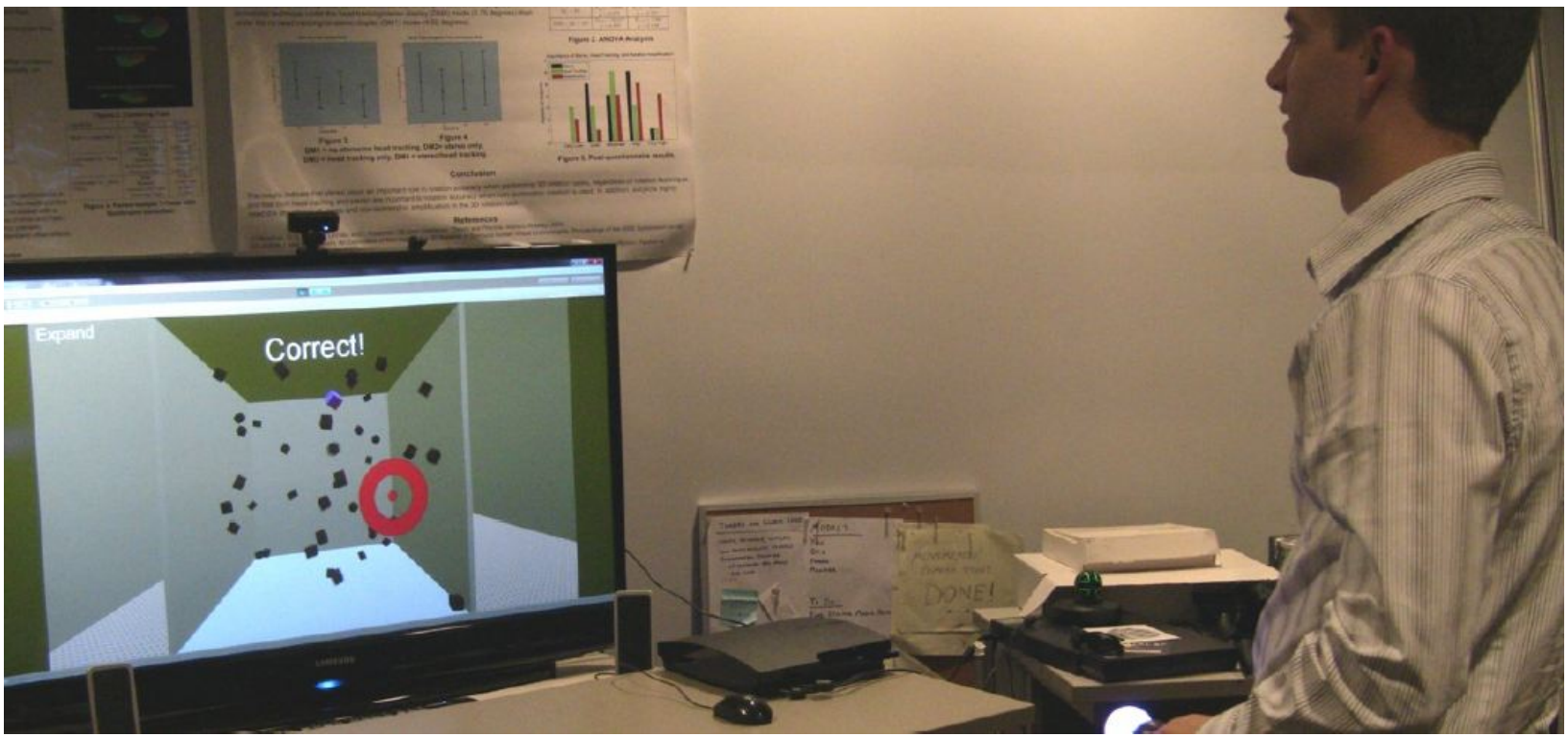
OnMouseOver: Se llama cada frame mientras el puntero del ratón está sobre el objeto.

Selección

La selección permite al usuario referenciar componentes de un modelo geométrico.

Es esencial en cualquier aplicación gráfica que requiera la edición de un modelo.

El proceso de selección suele realizarse como un posicionamiento y una búsqueda en el modelo geométrico.



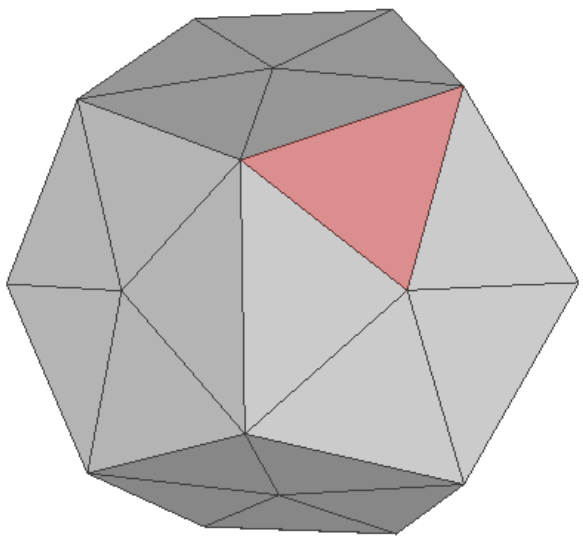
Identificadores de objetos

Para poder realizar la selección los componentes de la escena deben tener asociados **identificadores**.

Los identificadores se pueden asociar a distinto nivel, permitiendo controlar el **ámbito** de la selección: vértice, primitiva, objeto, etc.

Se pueden asignar **identificadores compuestos** para seleccionar elementos dentro de un componente.

El ámbito de la selección dependerá de la acción.



Métodos de selección

El proceso que se realiza para seleccionar implica:

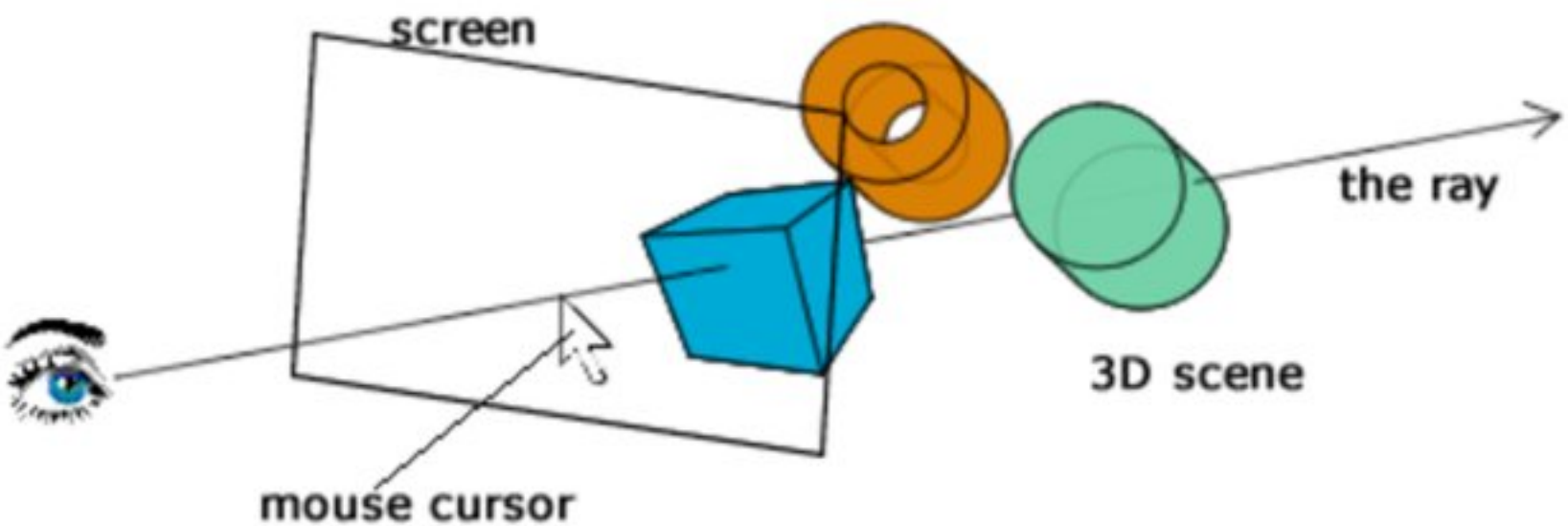
1. El usuario introduce una posición en pantalla
2. Se busca el componente (en el ámbito indicado) que se *corresponde* con esa posición de pantalla

Esta búsqueda se puede realizar de varios modos:

- Usando el buffer de selección (OpenGL hasta versión 3).
- Calculando la intersección de un rayo con la escena.
- Codificando el id de objeto como color y leyendo el frame buffer.
- Recortado (clipping).

Selección: Intersección rayo escena

1. Se calcula el rayo que pasa por el centro de proyección y la posición que corresponde al cursor en el plano de proyección.
2. Se calcula la intersección de este rayo con todos los objetos de la escena.
3. Seleccionamos el objeto intersectado por el rayo a menor distancia de la cámara.



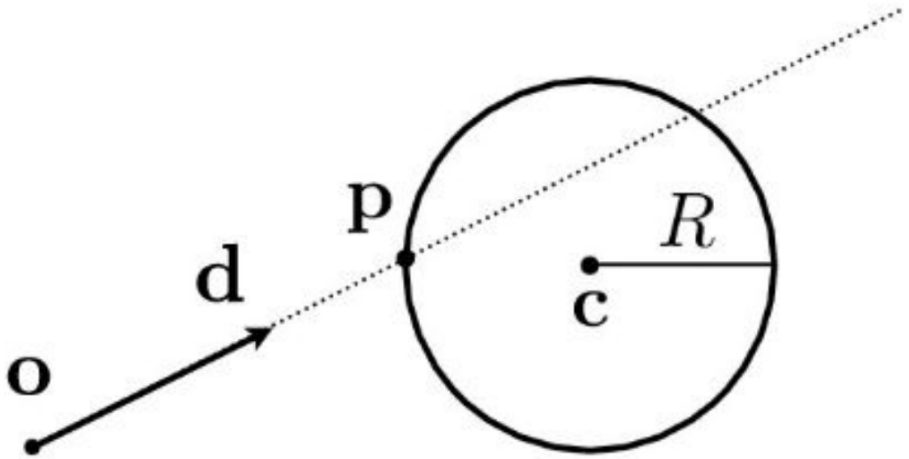
Modelo formado por esferas

$$\mathbf{r}(t) = \mathbf{O} + t \cdot \mathbf{d}$$

$$(\mathbf{p} - \mathbf{C})^2 - R^2 = 0$$

$$(\mathbf{O} + t \cdot \mathbf{d} - \mathbf{C})^2 - R^2 = 0$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

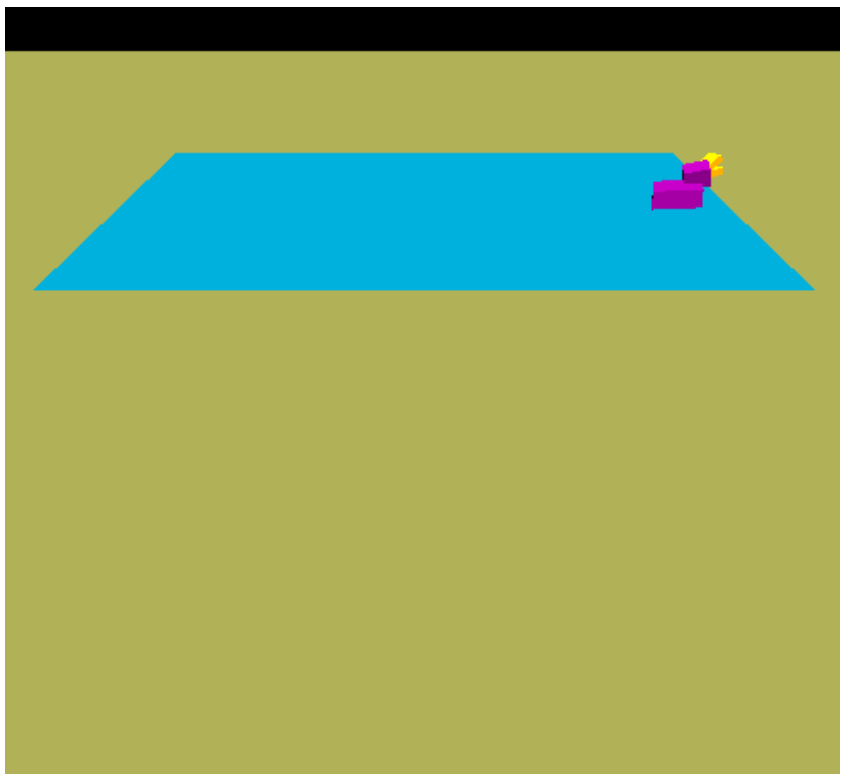


con $a = d \cdot d$, $b = 2(\mathbf{O} - \mathbf{C}) \cdot d$, $c = (\mathbf{O} - \mathbf{C}) \cdot (\mathbf{O} - \mathbf{C}) - R^2$

Habrá intersección si $(b^2 - 4ac) > 0$.

Selección: codificación de identificador como color

1. Se dibuja la escena con iluminación desactivada usando como colores los identificadores de los objetos en un frame buffer no visible.
2. Se lee el pixel que corresponde a la posición del cursor.
3. Se decodifica el color para obtener el identificador.



Selección OpenGL

El dibujo en frame-buffer no visible se puede hacer de dos formas:

1. Creando un objeto OpenGL de tipo frame-buffer object (FBO), y haciendo rasterización con ese objeto como imagen de destino (rendering target).
2. Sin buffer adicional, usando el modo de doble buffer.
   En este caso, para evitar que la imagen generada para seleccionar sea visible se debe redibujar la escena después de seleccionar sin intercambiar los buffers (es decir sin llamar a $glutSwapBuffers()$).

Selección: codificación de identificador como color

```
void ColorSeleccion( int i, int Parte)
   {
   unsigned char r = (i 0xFF);
   unsigned char g = (Parte  0xFF);
   glColor3ub(r,g,0);
}

void dibujoEscena()
   {
   ....
   glMaterialfv( GL_FRONT, GL_AMBIENT_AND_DIFFUSE,naranja);
   ColorSeleccion( i, PatasId);
   ....
}

void dibujar()
{
   ...
   glEnable(GL_LIGHTING);
   dibujoEscena();
   glutSwapBuffers();  // Intercambia buffers: muestra buffer trasero
}
```

Selección OpenGL (identificador como color)

```
int pick(int x, int y,int * i, int * parte)
{
  GLint viewport[4];
  unsigned char data[4];

  glGetIntegerv (GL_VIEWPORT, viewport);
  glDisable(GL_DITHER);
  glDisable(GL_LIGHTING);
  dibujoEscena();
  glEnable(GL_LIGHTING);
  glEnable(GL_DITHER);
  glFlush();

  glReadPixels(x, viewport[3] - y,1,1, GL_RGBA, GL_UNSIGNED_BYTE, data);

  *i=data[0];
  *parte=data[1];

  dibujoEscena();
  return *i;
}
```

Selección OpenGL (identificador como color)

- Es necesario codificar los identificadores como colores (R,G,B), y usar esos colores en lugar de los materiales de los objetos.
- Los colores se fijan con la función *glColor3ub* a la que se le pasa los tres bytes del color como enteros sin signo (0 a 255).
- Es necesario desactivar iluminación y texturas y activar sombreado plano.
- Para obtener el identificador usamos la función *glReadPixels*.
- Los objetos no seleccionables no se deben dibujar.
- El modo de dibujo dependerá del ámbito de la selección.

```
void clickRaton( int boton, int estado, int x, int y ){
  int parte=-1, i=-1,k=0;
  if(boton== GLUT_LEFT_BUTTON && estado == GLUT_DOWN) {
    // Se ha actuado sobre el izquierdo que esta ahora pulsado
    k=pick(x,y,&i,&parte);
    if(k>-1) { // se ha seleccionado algo
      switch (parte) {
        case 0:
        ...
  }
}
```

Selección en Unity: Raycast

La selección en Unity se puede realizar calculando la intersección rayo-objeto con el método *Raycast* si el objeto tiene asociado un colisionador.

```
void Update()
{
  if (Input.GetMouseButtonDown(0)){
    // Devuelve verdadero si se ha pulsado el boton en este frame
    RaycastHit hitInfo = new RaycastHit();
    bool hit = Physics.Raycast(Camera.main.ScreenPointToRay
               (Input.mousePosition), out hitInfo);
    //Devuelve verdadero si el rayo intersecta el collider de un objeto
    if (hit) {
      Selected = hitInfo.transform.gameObject; // Objeto seleccionado
      Debug.Log("Hit " + Selected.name);
      // Muestra el nombre del objeto en consola
      ...
      }
  }
}
```

Selección en Unity: OnMouseDown

Con el método *OnMouseDown* se puede ocmprobar si el cursor está sobre el objeto. El siguiente código cambia el color del objeto cuando el cursor está sobre el objeto y lo hace girar.

```
public class encima : MonoBehaviour {
  public Material MaterialPick,MaterialBase;
  public float speed = 1.0f;
  private bool fuera = true;

void OnMouseExit(){
  GetComponent<MeshRenderer> ().material = MaterialBase;
  fuera=true;
}

void OnMouseOver(){
  if(fuera) GetComponent<MeshRenderer> ().material = MaterialPick;
    fuera = false;
    transform.Rotate(0, speed, 0, Space.Self);
  }
}
```