

# Informática Gráfica

## Lectura de posiciones

Juan Carlos Torres  
**Grupos C y D**

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

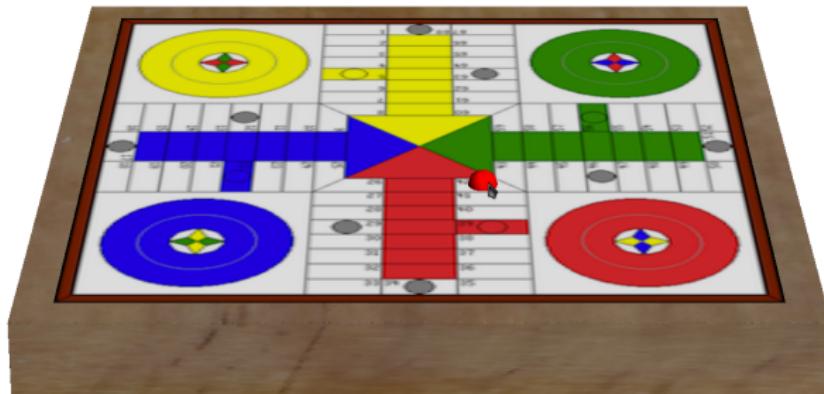
---

Curso 2024-25

# Leer posiciones

Se leen posiciones para:

- Colocar componentes en la escena.
- Dar parámetros de operaciones de edición.
- Especificar transformaciones geométricas.



# Dispositivos de entrada

## Desplazamientos 2D

Dispositivos que devuelven el desplazamiento realizado sobre una superficie, como el ratón. El sistema de entrada puede generar una posición 2D en pantalla integrando estos desplazamiento y usando como información de realimentación un cursor.



# Dispositivos de entrada

## Desplazamientos 3D

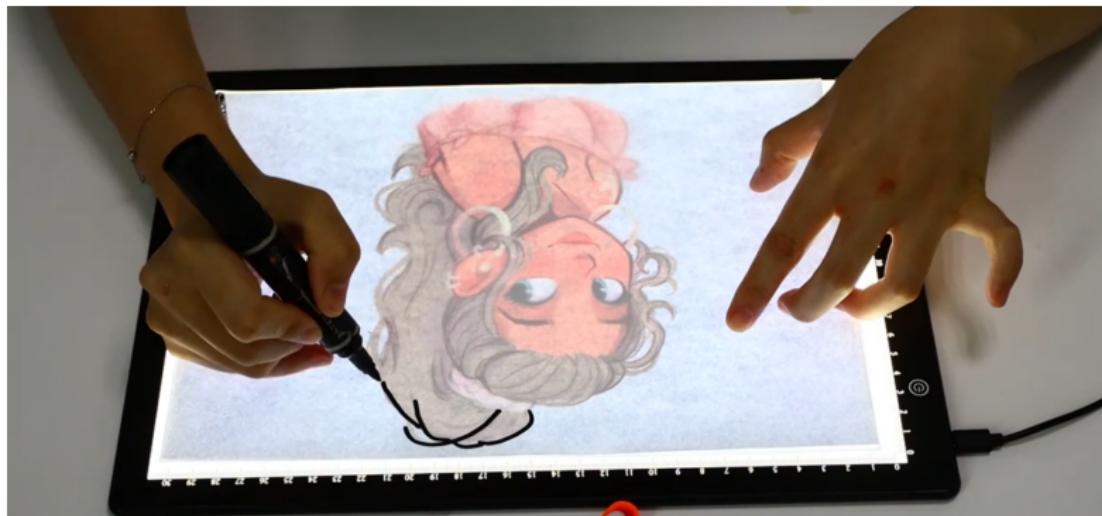
Dispositivos que devuelven desplazamientos en 3 dimensiones como el spaceMouse.



# Dispositivos de entrada

## Posiciones 2D

Devuelve una posición en un plano o en pantalla. Ejemplos de este grupo son las pantallas táctiles, los lápiz ópticos y las tabletas digitalizadoras.



# Dispositivos de entrada

## Posiciones 3D

Devuelve una posición en el espacio. Este es el caso los brazos de medición, los dispositivos de seguimiento (trackers) y los dispositivos hápticos.



# Dispositivos de entrada

## Gestos

Devuelven los gestos realizados. Estos sistemas reconocen todo o parte del cuerpo y permiten entregar información de los gestos realizados. Por ejemplo la Kinect reconoce todo el cuerpo y los dispositivos Leap Motion identifican gestos realizados con la mano.



# Lectura de posiciones con ratón

Cuando se leen posiciones se obtienen coordenadas 2D o 3D en el sistema de coordenadas del dispositivo.

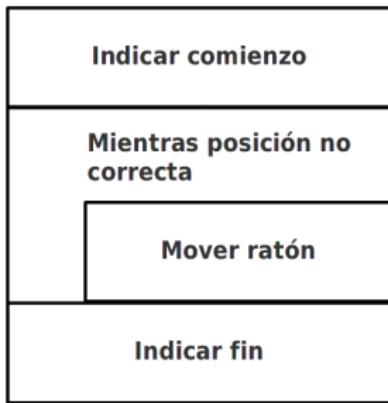
Por tanto, debemos resolver normalmente uno de estos dos problemas:

- Transformar las coordenadas del sistema de coordenadas del dispositivo al de la escena (*Sistema de coordenadas del mundo WC*), cuando los dos espacios tienen la misma dimensión.
- Generar coordenadas 3D a partir de las coordenadas 2D devueltas por el dispositivo, en caso contrario.

Por ejemplo, si estamos introduciendo posiciones para colocar elementos en un escenario 3D usando un ratón, las coordenadas entregadas por el dispositivo serán coordenadas 2D de pantalla, y necesitaremos coordenadas 3D en nuestro escenario.

# Acciones del usuario

Desde el punto de vista del usuario, que utiliza un dispositivo que controla un cursor, el proceso de entrada es el que se muestra en la figura



# Acciones de la aplicación

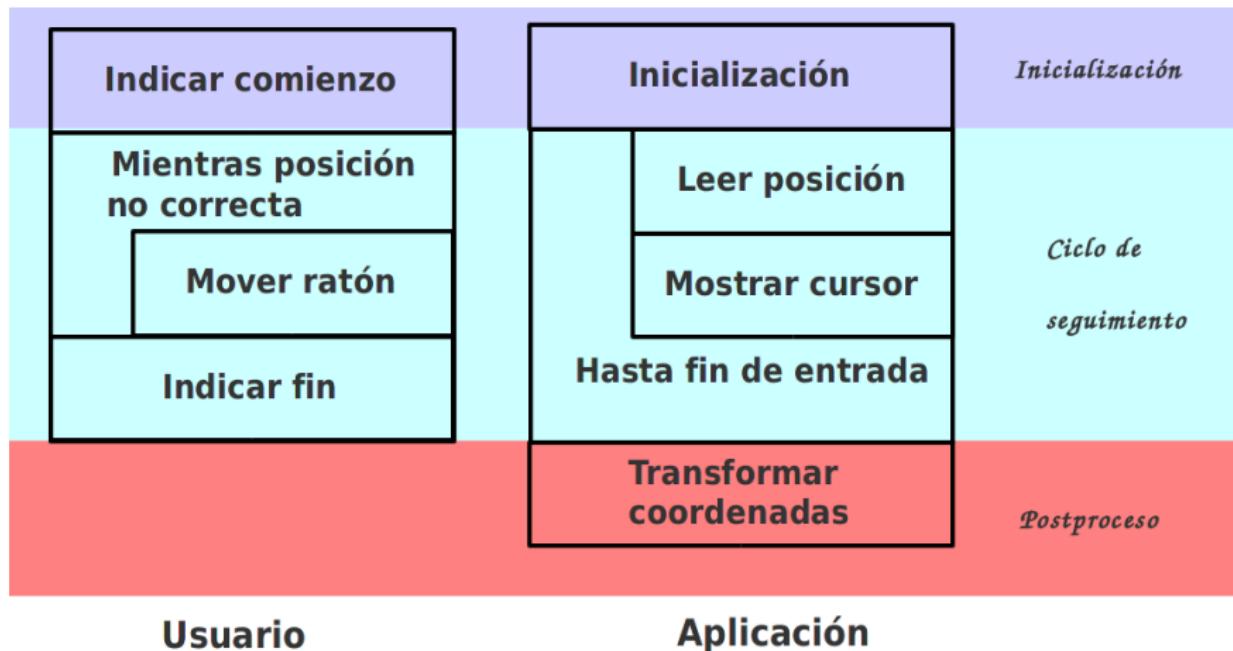
El proceso en el software de tratamiento se realiza en una estructura cíclica, que se corresponde con el ciclo de búsqueda que realiza el usuario. En este proceso podemos distinguir tres etapas:

- **inicialización**
- **ciclo de seguimiento**
- **postproceso**



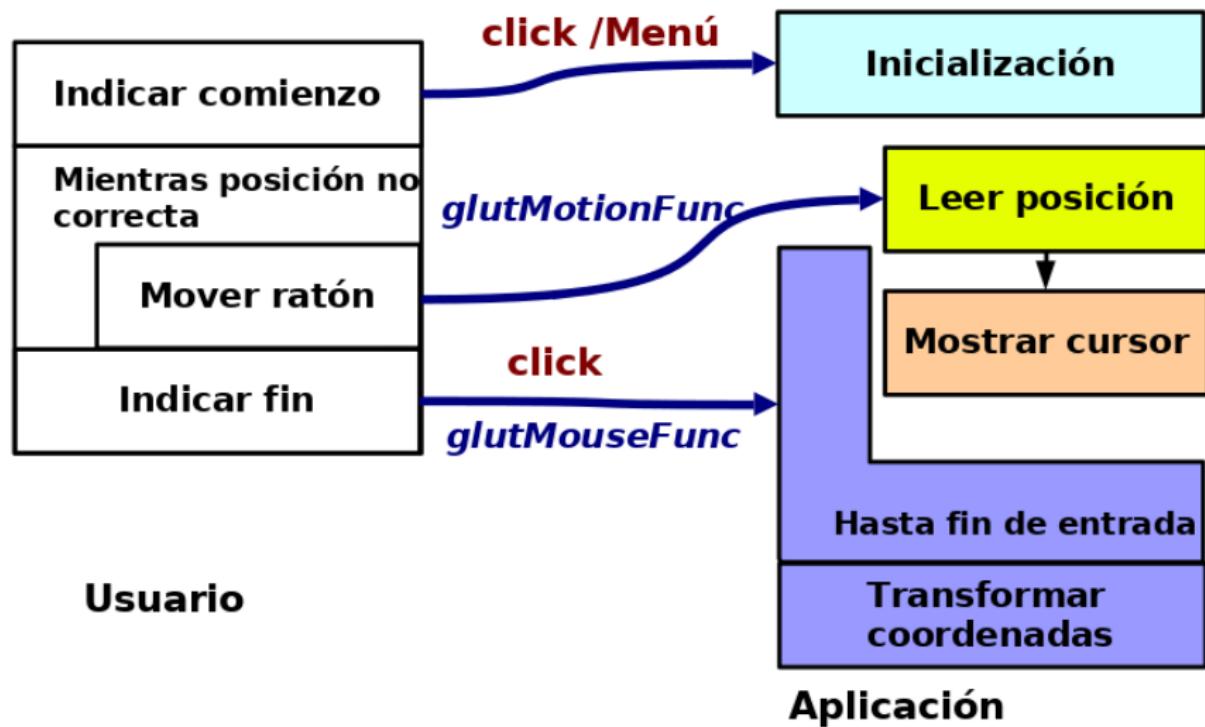
# Correspondencia de operaciones

Correspondencia entre operaciones del usuario y del sistema



# Procesamiento en OpenGL con glut

Correspondencia entre operaciones del usuario y del sistema



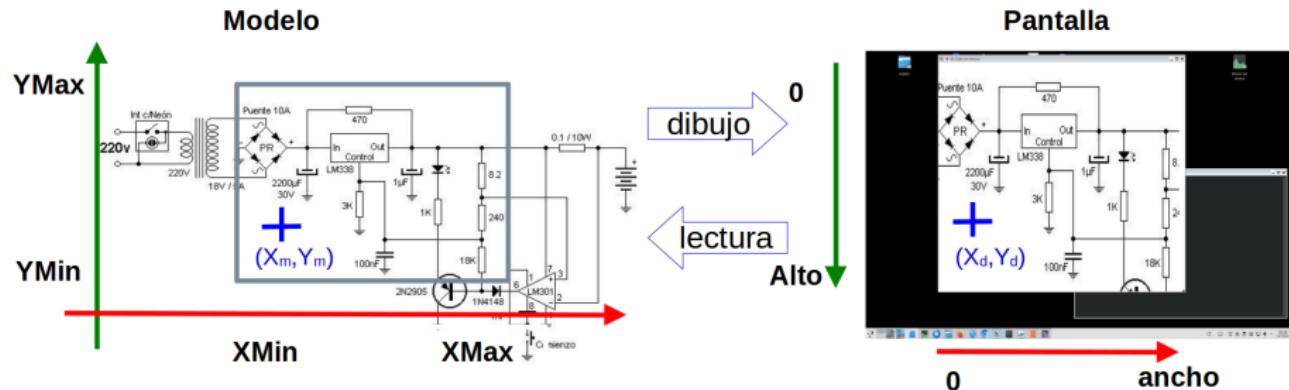
## Ejemplo: entrada de una línea

```
int dibujaLinea=0;
int linea[4];
void clickRaton( int boton, int estado, int x, int y ) {
    if(boton==GLUT_LEFT_BUTTON && estado==GLUT_DOWN) {
        linea[0]=x; linea[1]=y;
        linea[2]=x; linea[3]=y;
        dibujaLinea=1;
    } else if(boton==GLUT_LEFT_BUTTON && estado==GLUT_UP) {
        dibujaLinea=0;
        addLine(); // Crea la linea en el modelo.
    }
    glutPostRedisplay();
}

void RatonMovido( int x, int y ) {
    if(dibujaLinea) {
        linea[2]=x; linea[3]=x;
    }
    glutPostRedisplay();
}
void Dibuja() {
    ...
    if(dibujaLinea) {
        glBegin (GL_LINES);
        glColor3f (1, 1, 0);
        glVertex2f (linea[0],linea[1]); glVertex2f (linea[2],linea[3]);
        glEnd();
    }
}
```

# Lectura de posiciones en modelos 2D

Cuando la escena es 2D, la transformación a aplicar a las posiciones de pantalla es la inversa de la transformación de visualización.

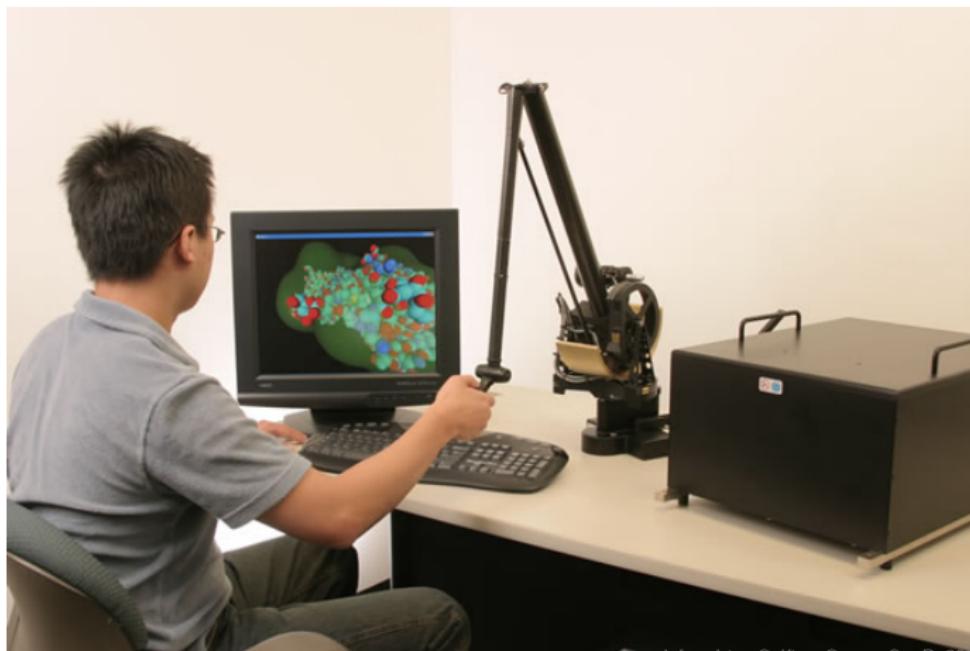


$$X_m = X_{min} + X_d * (X_{max} - X_{min}) / ancho$$

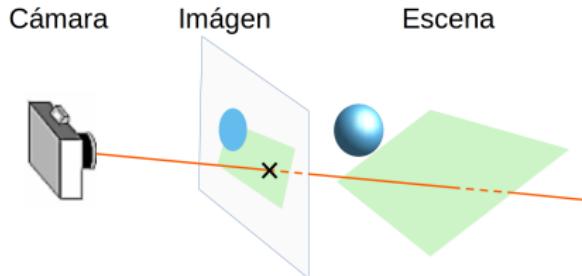
$$Y_m = Y_{max} - Y_d * (Y_{max} - Y_{min}) / alto$$

# Posiciones 3D: Dispositivos de entrada 3D

Para introducir posiciones 3D se pueden usar dispositivos de entrada que generen posiciones 3D (p.e. un palpador). El dispositivo entrega una posición en el espacio en su propio sistema de coordenadas. Esta posición se debe transformar al sistema de coordenadas de la escena.



# Posiciones 3D con dispositivos 2D



- Utilizar tres vistas con proyecciones paralelas ortogonales, con un cursor 2D en cada una.
- Utilizar un cursor virtual 3D que se manipula actuando en cada momento sobre una o dos coordenadas.
- Restricción a una superficie: el punto 3D se obtiene intersectando un rayo con la superficie o por selección en la superficie.

En los dos primeros casos el usuario realiza varias operaciones de entrada para generar la posición 3D, introduciendo en cada una de ellas dos coordenadas (X-Y, Y-Z o X-Z).

En el último caso se debe seleccionar en primer lugar el plano sobre el que se dará la posición.

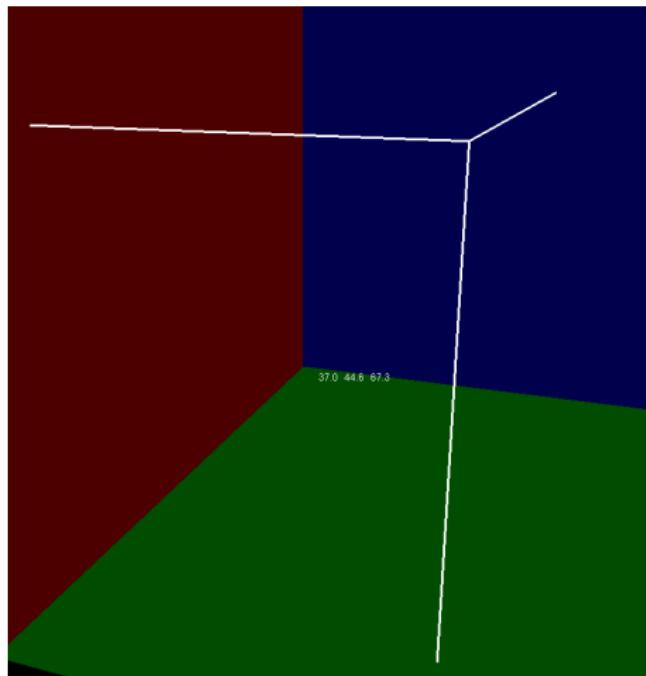
# Vista con tres proyecciones paralelas

En cada vista se fijan dos coordenadas. Se muestra como realimentación un cursor 2D en las tres vistas.



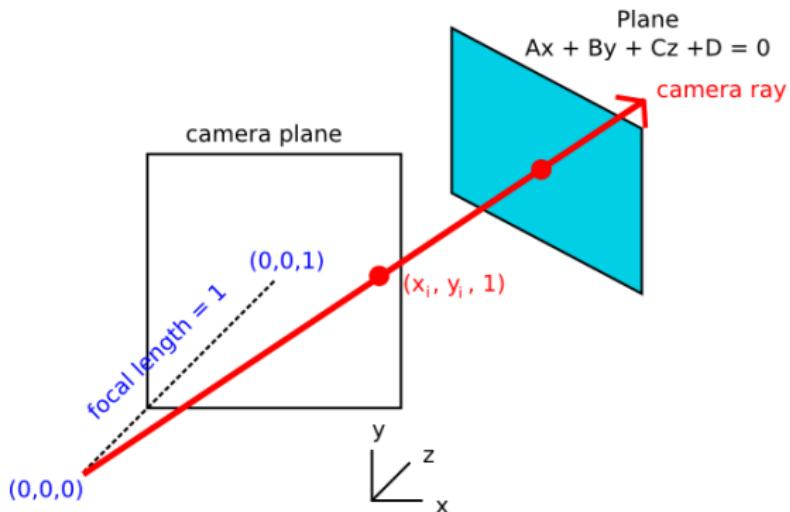
# Cursor 3D

Se muestra un cursor 3D, que se puede desplazar usando un dispositivo 2D alternando movimientos en el plano X-Y, X-Z o en el Y-Z en función de los botones del ratón que estén pulsados.



# Intersección rayo

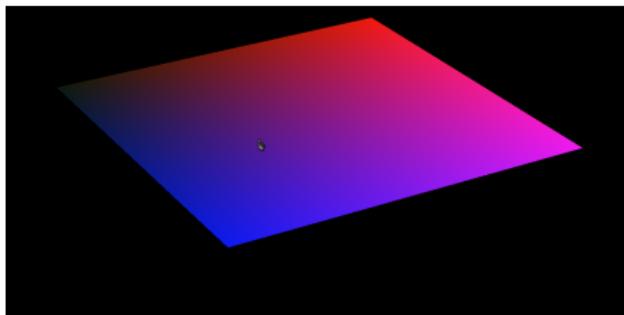
Si se restringe la posición a una superficie se puede obtener una posición 3D por intersección de la recta que pasa por el punto introducido (en el plano de proyección) y el centro de proyección con la superficie.



# Lectura posición por color

Proceso similar a la selección por color, utilizando como color la posición (pero colores float e interpolación de color).

$$\begin{aligned} R &= (x - X_{min}) / (X_{max} - X_{min}) \\ G &= (y - Y_{min}) / (Y_{max} - Y_{min}) \\ B &= (z - Z_{min}) / (Z_{max} - Z_{min}) \end{aligned} \quad (1)$$



# OpenGL: Varias vistas

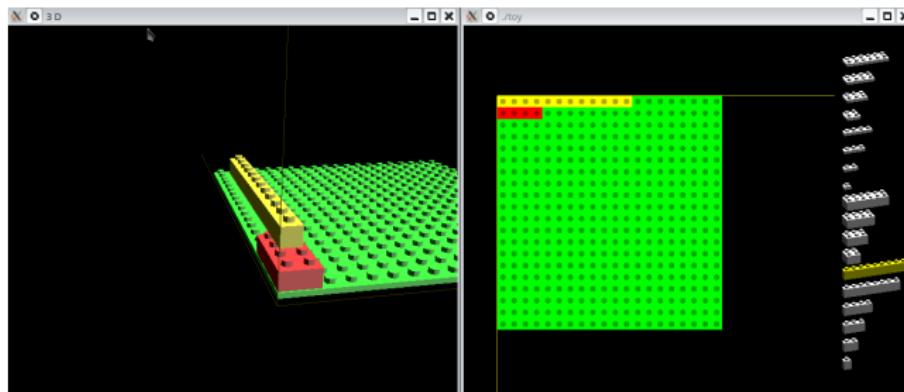
Creación de ventanas y subventanas de glut:

```
int ventana, subventana;  
ventana = glutCreateWindow ("titulo");  
subventana = glutCreateSubWindow (ventana,x1,y1,ancho,alto);
```

Cambio de la zona de dibujo:

```
glutSetWindow( identificadorDeVentana );
```

Los eventos y variables de estado son propios de cada subventana.



# OpenGL: Intersección rayo

La función *gluUnProject* calcula un punto en el espacio de la escena sobre el rayo que pasa por un pixel.

```
int gluUnProject(double X, double Y, double Z,  
    double *modelMatrix, double *projMatrix, int *viewport,  
    double *Xm, double *Ym, double *Zm);
```

$(X, Y, Z)$  es la coordenada del pixel. La profundidad de un pixel se puede leer con la función *glReadPixels*:

```
glReadPixels(X, Y, 1, 1, GL_DEPTH_COMPONENT, GL_FLOAT, &Z);
```

# OpenGL: Intersección rayo

La función *gluUnProject* calcula un punto en el espacio de la escena sobre el rayo que pasa por un pixel.

```
void mouseClicked(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        float X = x;
        float Y = windowHeight - y;
        float Z;
        GLdouble modelMatrix[16], projMatrix[16];
        GLint viewport[4];

        // Obtener las matrices actuales y el viewport
        glGetDoublev(GL_MODELVIEW_MATRIX, modelMatrix);
        glGetDoublev(GL_PROJECTION_MATRIX, projMatrix);
        glGetIntegerv(GL_VIEWPORT, viewport);

        glReadPixels(X, (int) Y, 1, 1, GL_DEPTH_COMPONENT, GL_FLOAT, &Z);

        // Desproyectar las coordenadas del ratón para Z
        gluUnProject(X, Y, Z, modelMatrix, projMatrix, viewport, &Xm, &Ym, &Zm);
        std::cout << "X = " << Xm << ", Y = " << Ym << ", Z = " << Zm << std::endl;
        glutPostRedisplay();
    }
}
```

# Lectura Unity con ScreenToWorldPoint

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CreaCajaCamara: MonoBehaviour
{
    public GameObject cajaPrefab; // Prefab del objeto a crear

    private void Update()
    {
        if (Input.GetMouseButtonDown(1))
        // Si se hizo clic con el botón derecho del ratón
        {
            Vector3 mousePos = Input.mousePosition; // Posición 2D del ratón
            mousePos.z = 15f; // Distancia de la cámara
            // Obtiene la posición del clic del ratón en el mundo
            mousePos = Camera.main.ScreenToWorldPoint(mousePos);
            // Crea una nueva instancia del objeto
            Instantiate(cajaPrefab, mousePos, Quaternion.identity);
        }
    }
}
```

# Posicionamiento en superficie con raycast

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class rayCreate : MonoBehaviour {
    public GameObject suelo; // Superficie sobre la que posicionar
    public GameObject cajaPrefab; // Prefab del objeto a crear

    private void Update()
    {
        if (Input.GetMouseButtonUp(2)) //Click en botón central
        {
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            if (Physics.Raycast(ray, out hit))
            {
                if (hit.collider != null) // Intersectó con un objeto?
                {
                    if(hit.collider.gameObject == suelo){// Es el suelo?
                        Vector3 posicionEnSuperficie = hit.point;
                        posicionEnSuperficie.y=0.5f;
                        Instantiate(cajaPrefab, posicionEnSuperficie, Quaternion.identity);
                    }
                }
            }
        }
    }
}
```

# Transformaciones Geométricas

Las transformaciones geométricas se pueden definir a partir de puntos.

- Una traslación se puede definir por el vector que va de la posición original a la posición nueva.
- Una rotación por el ángulo formado por el vector que va del punto de referencia a la posición actual con la horizontal.

La transformación se puede indicar haciendo dos posicionamientos o solo uno, si se asume una posición concreta para el punto de referencia (p.e. el centro del objeto).

En cualquier caso, la transformación se puede calcular en el ciclo de seguimiento, dando como información de realimentación el resultado de la transformación.

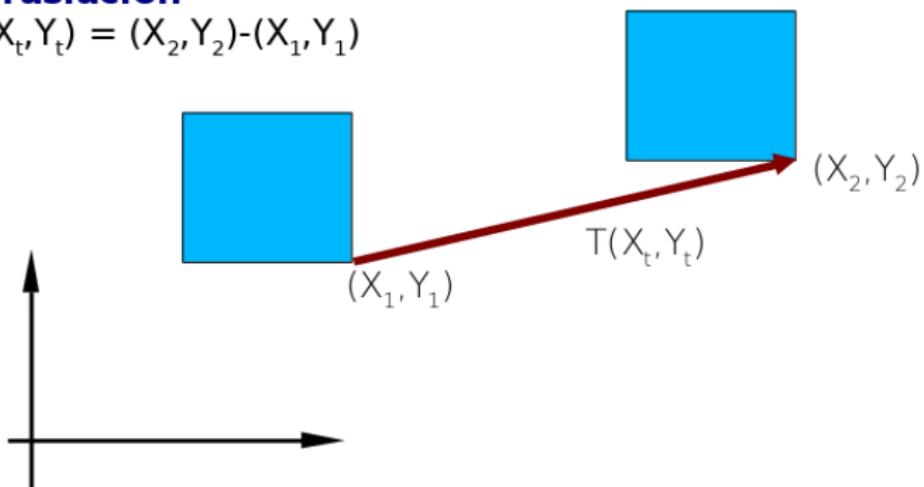
# Transformaciones: Traslación

**a) Interacción: Arrastre**

**b) Cálculo de parámetros**

## Traslación

$$(X_t, Y_t) = (X_2, Y_2) - (X_1, Y_1)$$

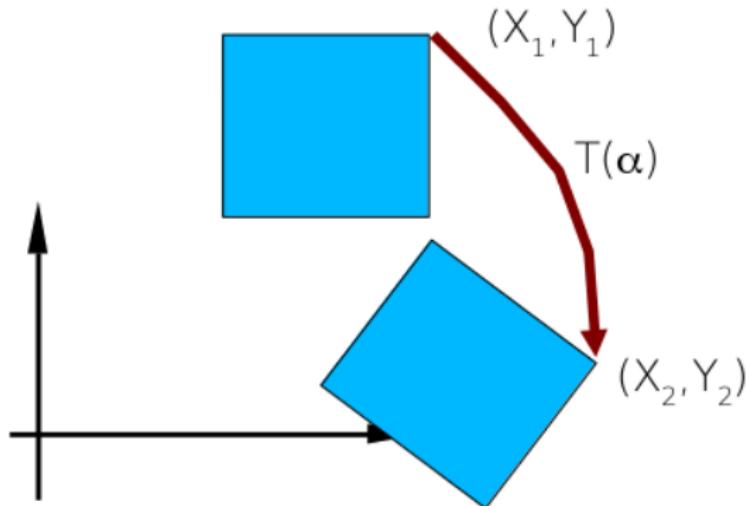


## a) Interacción: Arrastre

## b) Cálculo de parámetros

### Rotación

$$\alpha = \text{angulo}(X_2, Y_2) - \text{angulo}(X_1, Y_1)$$



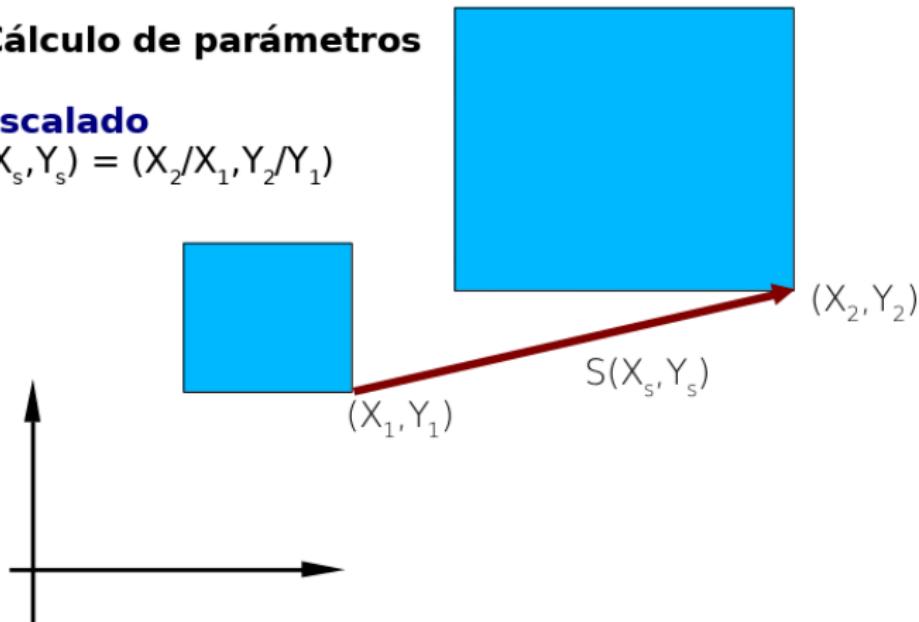
# Transformaciones: Escalado

a) Interacción: Arrastre

b) Cálculo de parámetros

## Escalado

$$(X_s, Y_s) = (X_2/X_1, Y_2/Y_1)$$



# Transformaciones: Manipuladores

Se usan para modificar interactivamente las transformaciones geométricas.

Para cada transformación se visualizan manipuladores específicos en los que se pueden seleccionar los ejes y modificar cada uno de las dimensiones.

