Introducción: Selección

- Selección
  - Orientación de un triángulo y punto interior a un triángulo
  - Transformación de coordenadas

Ana Gil Luezas
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

Selección de un triángulo con el ratón

Queremos seleccionar un triángulo con el ratón para desplazarlo con el evento de movimiento del ratón. Establecemos el callback con

glutMotionFunc(motion); y definimos la función

```
void motion(int px, int py); // el ratón está dentro del triángulo?
```

Las coordenadas del ratón (px, py) están dadas en el sistema de ventanas.

Las coordenadas de los vértices del triángulo están en el sistema de coordenadas local que es transformado por la matriz de modelado (con los comandos glTranslated(), glRotated())

Necesitamos las coordenadas del ratón y las de los vértices del triángulo en el mismo sistema de coordenadas

Necesitamos un algoritmo para determinar si un punto está dentro de un triángulo

Orientación de un triángulo

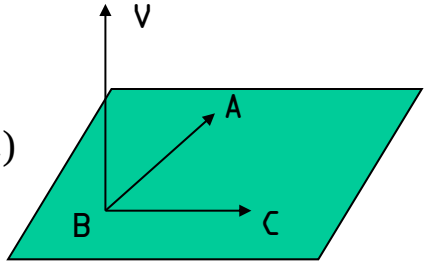
- Dado los tres vértices A, B y C de un triángulo, para saber si están orientados en sentido CCW (antihorario) o CW

Obtenemos el producto vectorial V = (C– B) x (A– B)

Si Vz > 0 -> los vértices están orientados en sentido CCW

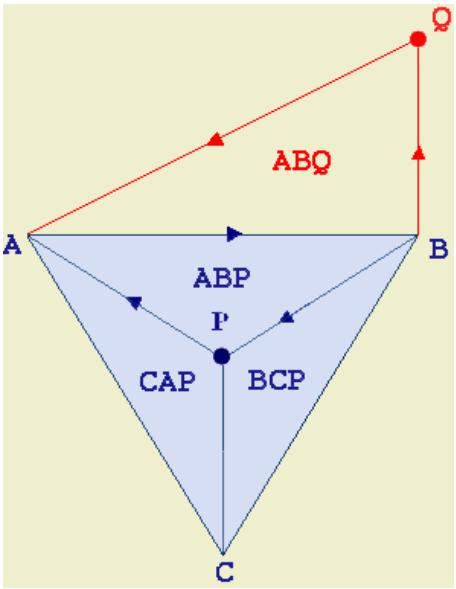
Si Vz < 0 -> los vértices están orientados en sentido CW

$$Vz = ((C-B)_x \cdot (A-B)_y) - ((C-B)_y \cdot (A-B)_x)$$



Punto en el interior de un triángulo

Dado los tres vértices A, B y C de un triángulo, para saber si un punto está en el interior del triángulo comprobamos si los tres triángulos ABP, BCP y CAP están orientados de la misma forma que el triángulo ABC.



El punto Q está fuera:

ABQ es CCW y ABC es CW

El punto P está dentro:

ABP, BCP y CAP son CW

igual que ABC

Punto en el interior de un triángulo

Dado los tres vértices A, B y C de un triángulo en CCW, para saber si un punto P está en el interior del triángulo comprobamos si los triángulos ABP, BCP y CAP también son CCW

SI (Ax - Px) * (By - Py) – (Ay - Py) * (Bx - Px) < 0

-> P está fuera

SI (Bx - Px) * (Cy - Py) – (By - Py) * (Cx - Px) < 0

-> P está fuera

SI (Cx - Px) * (Ay - Py) – (Cy - Py) * (Ax - Px) < 0

-> P está fuera

En otro caso -> P está dentro

Selección de un triángulo con el ratón

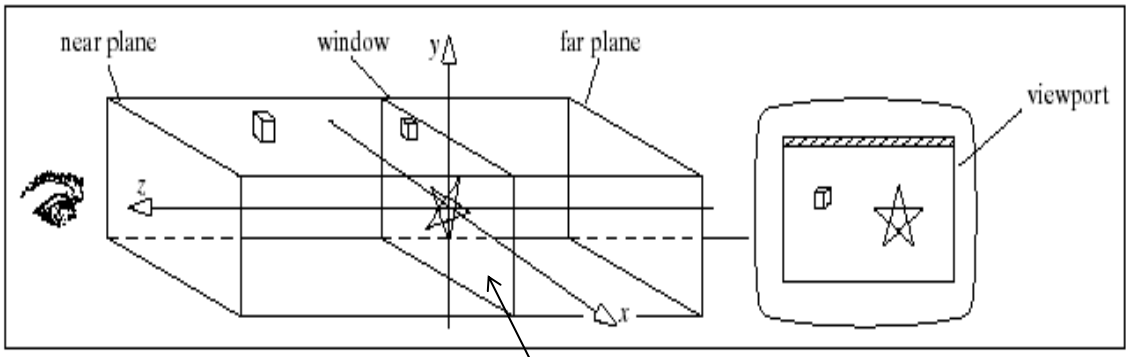
En el proceso anterior hemos supuesto que las coordenadas de los vértices y del punto están establecidas en el mismo sistema de coordenadas.

Al seleccionar un triángulo con el ratón, las coordenadas del ratón están dadas en el sistema de ventanas.

Necesitamos las coordenadas del ratón en el sistema de coordenadas de la Ventana de Vista

Selección: Transformación de coordenadas

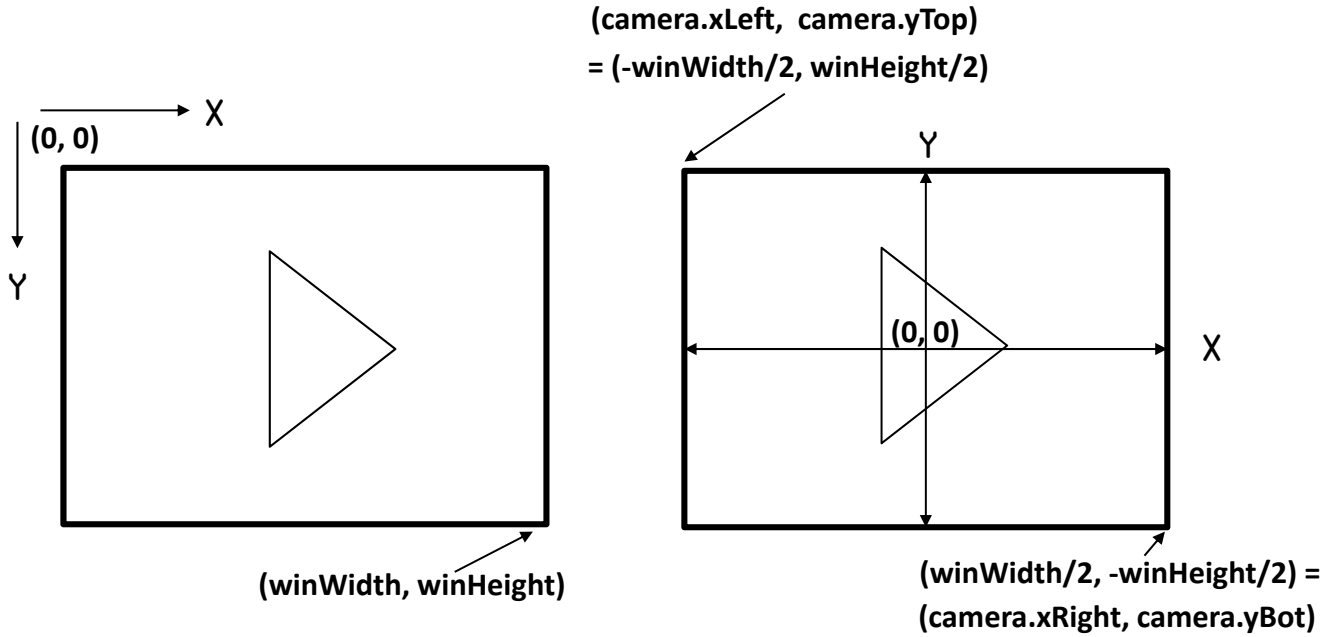
Del sistema de ventanas al plano de proyección ortogonal



Área visible, ventana de vista, o plano de proyección

Selección: Transformación de coordenadas

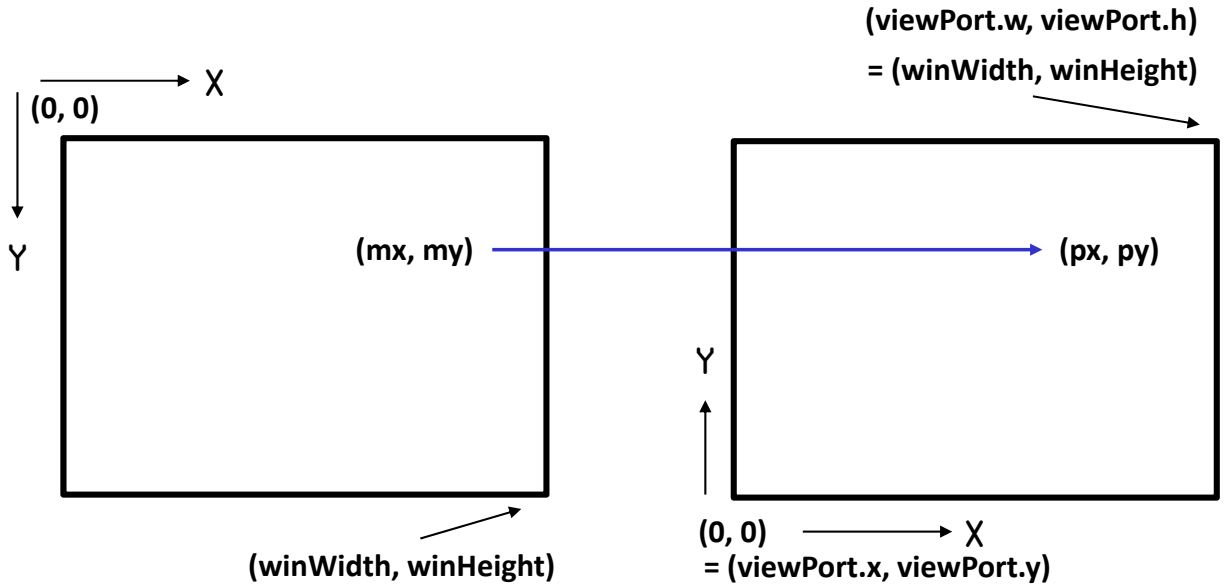
Hemos establecido el puerto de vista ocupando toda la ventana, y una proyección ortogonal con una ventana de vista centrada en la cámara, y con la misma escala que el puerto de vista (relación 1:1:1)



Selección: Transformación de coordenadas

Coordenadas del puerto de vista -> dar la vuelta a la coordenada Y

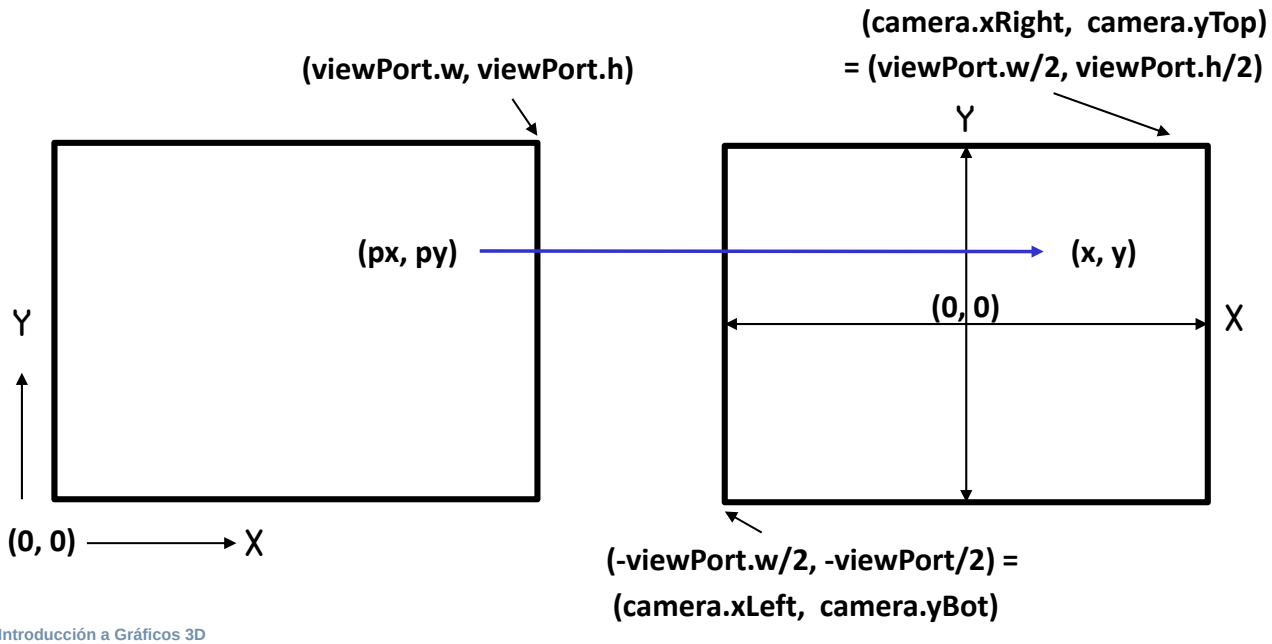
px = mx
py = winHeight – my



Selección: Transformación de coordenadas

Coordenadas del VV Ortogonal -> trasladar el origen del sistema de coordenadas

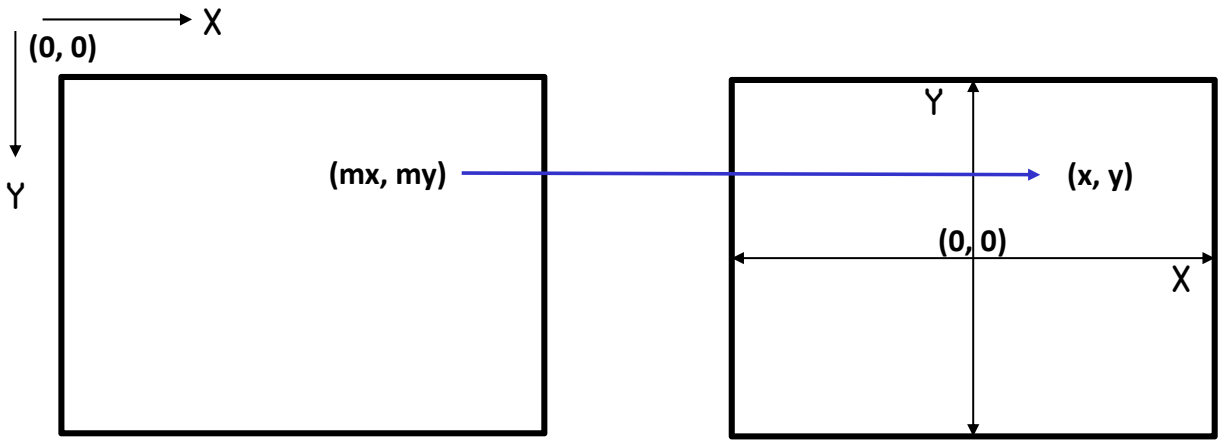
x = camera.xLeft + px
y = camera.yBot + py



Selección: Transformación de coordenadas

Dado un punto (mx, my) de coordenadas del ratón (en el sistema de ventanas) -> (x, y) en el sistema de coordenadas de la ventana de vista
Relación 1:1:1

```
GLdouble xWin2VV(int mx) { return mx - winWidth / 2.0; }
GLdouble yWin2VV(int my) { return (winHeight - my) – winHeight / 2.0; }
```



Selección de un triángulo con el ratón

En el proceso anterior hemos supuesto que las coordenadas de los vértices y del punto están establecidas en el mismo sistema de coordenadas.

Las coordenadas de los vértices del triángulo están en el sistema de coordenadas local que es transformado por la matriz de modelado (con los comandos glTranslated(), glRotated())

Tenemos que aplicar las transformaciones de modelado a los vértices del triángulo.

Selección: Transformación de coordenadas

Inicialmente el triángulo aparece sin transformaciones, y para colocarlo en la textura vamos a utilizar dos transformaciones

Posicionar(px, py)
RotarCentro()

Que se irán ejecutando a medida que se utilice el ratón y la tecla ‘r’.

Para aplicar estás transformaciones a los vértices del triángulo, utilizamos atributos para el centro del triángulo, el radio y el ángulo de rotación sobre su centro.

Al mover el ratón, se actualiza el centro y se recalculan los vértices.

Al pulsar la tecla ‘r’ se actualiza el ángulo y se recalculan los vértices.

Ecuación de la circunferencia con radio R y centro (Cx, Cy):

(x , y) = (Cx + R cos(ang), Cy + R sin(ang)) para 0 <= ang < 2.PI

Selección

Para seleccionar un objeto cualquiera de una escena 3D:

Podemos generar un rayo, que parta del ojo de la cámara y pase por el punto (x, y) del plano de vista, y obtener el objeto más cercano con el que colisiona (ray cast).