# Cámara: Eventos de ratón

## Informática Gráfica I

Material de: **Ana Gil Luezas**
Adaptado por: **Elena Gómez** y **Rubén Rubio**
{mariaelena.gomez,rubenrub}@ucm.es

UNIVERSIDAD COMPLUTENSE MADRID

# Contenido

# Objetivo

## Objetivo

### Realizar movimientos de cámara con el ratón

Movimientos buscados:

- Con el **botón derecho**: trasladar la escena desde que se pulsa el botón hasta que se suelta, a izquierda/derecha y arriba/abajo
- Con el **botón izquierdo**: hacer girar la cámara alrededor de la escena. El efecto es que la escena rota, aunque lo que se mueve es la cámara
- Con la **rueda del ratón**:
  - Con la tecla de Control pulsada: escalar la escena
  - Sin la tecla de Control pulsada: aproximar o alejar la escena, según se gire en uno u otro sentido

# Atributos de la clase Camera

- Para implementar algunos movimientos se usan los siguientes atributos de la clase `Camera`:
  - `mRadio`: es el radio de una esfera imaginaria sobre cuya superficie se mueve la cámara
  - `mAng`: es la longitud (este-oeste, es decir, distancia angular) a la que se encuentra la cámara sobre esa esfera imaginaria, a contar desde el eje X positivo, que tiene longitud 0°, medida en sentido antihorario
- Para procurar que el enlace de unos movimientos de la cámara con otros sea coherente (no se produzcan saltos) es preciso mantener actualizados estos dos atributos, después de realizar cualquier movimiento que les afecte
- Estos atributos se suponen inicializados así:

```
GLdouble mAng = -45;       // Longitud 45 grados oeste
GLdouble mRadio = 1000.0; // Esfera virtual de radio 1000
```

# Movimientos de la clase Camera

- `void moveLR(GLdouble cs)`: movimiento de la cámara a izquierda/derecha, sobre el eje `U`, una distancia `cs`, sin cambiar la dirección de vista.
- `void moveFB(GLdouble cs)`, `void moveUD(GLdouble cs)`: movimientos análogos sobre los ejes `N` y `V`, respectivamente.
- `void lookLR(GLdouble cs)`, `void lookUD(GLdouble cs)`: movimientos análogos sobre los ejes `U` y `V`, respectivamente, pero sin cambiar `eye`, solo `look`.

# Movimientos de la clase Camera

- `void orbit(GLdouble ax)`: orbita la cámara alrededor de look, en sentido antihorario, haciendo que `eye` describa el paralelo de la esfera de radio `mRadio`, que se encuentra a altura `eye.y`, incrementando `mAng` en la cantidad `ax`.
- `void orbit(GLdouble ax, Gldouble ay)`: método similar al anterior, pero se deja que la cámara cambie su altura `eye.y`, una cantidad `ay`. Observa que `ax` es un `double` que incrementa un ángulo mientras que `ay` es un `double` que incrementa una distancia.

# Programación de los eventos del ratón

- Para programar los eventos de ratón sigue estos pasos:
  1. Registrar los respectivos *callbacks* en `IG1App`:
     - `glfwSetMouseButtonCallback (win, s_mouse)`: para implementar lo que ocurre cuando se presiona o se suelta un botón del ratón.
     - `glfwSetCursorPosCallback (win, s_motion)`: para implementar lo que ocurre cuando se mueve el ratón.
     - `getSetScrollCallback (win, s_mouseWheel)`: para implementar lo que ocurre cuando se gira la rueda del ratón o se hace el gesto equivalente con el *touchpad*.

# Programación de los eventos del ratón

- Para programar los eventos de ratón sigue estos pasos (continuación):
  2. Implementar los *callbacks* estáticos anteriores `s_` ...

```
static void s_mouse(GLFWwindow* win, int button,
                    int action, int mods);
static void s_motion(GLFWwindow* win,
                     double x, double y);
static void s_mouseWheel(GLFWwindow* win,
                         double dx, double dy);
```

  invocando sus respectivos métodos con una instancia estática `s_ig1app` de `IG1App`.

# Programación de los eventos del ratón

- Para programar los eventos de ratón sigue estos pasos (continuación):
  3. Añadir los siguientes *callbacks* a `IG1App`:
     - `void mouse(int button, int action, int mods);`
       Se genera cuando se presiona o se suelta (`action`) un botón del ratón (`button`) (con alguna tecla de modificación pulsada, `mods`). La posición se puede consultar con `glfwGetCursorPos`.
     - `void motion(double x, double y);`
       Se genera cuando el ratón se mueve y recibe su posición (x, y) en coordenadas de la ventana.
     - `void mouseWheel(double dx, double dy);`
       Se genera cuando se mueve la rueda del ratón o equivalente en `dx` unidades horizontales y `dy` verticales.

# Programación de los eventos del ratón

- Programar `mouse()` de forma que se registren los valores lanzados en los atributos mencionados de `IG1App`:

  ```
  void mouse(int button, int action, int mods) {
  ```

  1. Guarda en `mBot` el valor de `button`
  2. Guarda en `mCoord` la posición (x, y) del ratón.

  }

# Programación de los eventos del ratón

- Recuerda que la variable `y` se refiere a coordenadas de ventana y esta tiene su origen en la esquina superior izquierda, mientras que en el puerto de vista el origen está en la esquina inferior izquierda. Puede ser conveniente el paso de una a otra:

```
double height;
glfwGetWindowSize(mWindow, nullptr, &height);
y(viewport) = height - y;
```

# Programación de los eventos del ratón

- Para programar `motion()` recuerda que se quiere implementar el proceso de pulsar un botón del ratón, arrastrar el cursor y soltarlo

```
void motion(double x, double y) {
```

1. Guarda en una variable auxiliar `mp` la diferencia entre `mCoord` y (x, y)
2. Guarda en `mCoord` la posición (x, y) del ratón
3. Si `mBot` es el botón izquierdo, la cámara orbita `(mp.x * 0.05, mp.y)`
4. Si `mBot` es el botón derecho, la cámara se desplaza `moveUD()` y `moveLR()` según `mp`
5. `mNeedsRedisplay = true;`

```
}
```

- Recuerda que, cuando la cámara orbita, `mp.x` hace referencia a una distancia en radianes y, por tanto, es preciso reducirla.

# Evento de la rueda del ratón

- Programar `mouse()` teniendo en cuenta si se tiene pulsada la tecla de Control:

```
void mouseWheel(double dx, double dy) {
```

1. Averigua si algún modificador está pulsado con `glfwGetKey()`
2. Si no hay ninguna, la cámara se mueve con `moveFB()`, según el valor de `dy`
3. Si está pulsada la tecla Ctrl (`GLFW_MOD_CONTROL`), la cámara cambia la escala con `setScale()`, según el valor de `dy`
4. `mNeedsRedisplay = true;`

```
}
```