

20. La clase Camera

- Añade a la [clase Camera](#) los métodos para desplazar la cámara en cada uno de sus ejes, sin cambiar la dirección de vista.

void [moveLR](#)(GLdouble cs); // Left / Right

void [moveFB](#)(GLdouble cs); // Forward / Backward

void [moveUD](#)(GLdouble cs); // Up / Down

Añade los atributos [mRight](#), [mUpward](#) y [mFront](#) para cada uno de los ejes; y el método protegido:

void [setAxes](#)(): Da valor a los tres ejes. Tendrás que incluir `<gtc/matrix_access.hpp>`

Todos los métodos que modifiquen algún elemento de la cámara tienen que actualizar todos los atributos necesarios para que sus valores sean coherentes.

Modifica el método

void [setViewMat](#)() { [viewMat](#) = [lookAt](#)(mEye, mLook, mUp); [setAxes](#)(); };

Quita (comenta) los métodos [pitch](#), [yaw](#) y [roll](#).

- Añade a la aplicación ([IG1App](#)) dos nuevos atributos: [dvec2 mMouseCoord](#) para guardar las coordenadas del ratón y, [int mMouseButton](#) para guardar el botón pulsado. Añade también los callbacks para los eventos del ratón:

- [glutMouseFunc](#)(s_mouse);

void [mouse](#)(int button, int state, int x, int y): captura, en [mMouseCoord](#), las coordenadas del ratón (x, y) y en [mMouseButton](#) el botón pulsado.

- [glutMotionFunc](#)(s_motion);

void [motion](#)(int x, int y): captura las coordenadas del ratón, obtiene el desplazamiento con respecto a las anteriores coordenadas y, si el botón pulsado es el derecho, desplaza la cámara en sus ejes [mRight](#) (horizontal) y [mUpward](#) (vertical) el correspondiente desplazamiento.

- [glutMouseWheelFunc](#)(s_mouseWheel);

void [mouseWheel](#)(int n, int d, int x, int y): Si no está pulsada ninguna tecla modificadora, desplaza la cámara en su dirección de vista (eje [mFront](#)).

- Añade a la [clase Camera](#) el método [orbit](#) para desplazar la cámara ([mEye](#)) siguiendo una circunferencia (en los ejes [mRight](#) y [mFront](#)) a una determinada altura (eje [mUpward](#)) alrededor de [mLook](#).

`void orbit(GLdouble incAng, GLdouble incY):` modifica la posición y la dirección de vista de la cámara

Añade los atributos para el ángulo y el radio (p. ej. 1000) de la circunferencia que recorrerá la cámara con este método. Tendrás que dar un valor adecuado al ángulo en los métodos [set2D](#), [set3D](#), ...

Modifica los métodos de la aplicación ([IG1App](#)):

[motion](#)(int x, int y): Además, si el botón pulsado es el izquierdo, actualiza la posición de la cámara en la circunferencia en función del desplazamiento del ratón.

[mouseWheel](#)(int n, int d, int x, int y): Si está pulsada la tecla CTRL escala la vista.

- Añade a la [clase Camera](#) un método [changePrj\(\)](#) para cambiar de proyección ortogonal a perspectiva. Modifica los métodos de la clase Camera afectados por el cambio de proyección.

Define la tecla [p](#) para cambiar entre proyección ortogonal y perspectiva

21. Añade a la aplicación la opción (tecla [k](#)) de visualizar dos vistas simultáneamente.

Añade a la aplicación un atributo [bool m2Vistas](#) para la opción, y utilízalo en el método [display](#) de la aplicación. Define el método [display2Vistas\(\)](#) para dividir la ventana en dos puertos de vista y visualizar en el lado izquierdo la vista actual, y en el lado derecho la vista cenital (utiliza una cámara auxiliar, la escala debe ser la misma). [display2Vistas\(\)](#) debe dejar la cámara (atributo [mCamera](#), incluido el puerto de vista) sin modificar. Puedes añadir a la clase [Camera](#) un método [setCenital](#).

```
void IG1App::display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    if (m2Vistas) display2Vistas();
    else { ... }
    glutSwapBuffers();
}
```

