Programación 2D

Práctica 1: Dibujo de primitivas y cálculos trigonométricos

En la primera práctica de la asignatura, vamos a trabajar sobre una ventana de GLFW y a dibujar una serie de primitivas gráficas utilizando LiteGFX. También aplicaremos algunos de los conceptos de trigonometría vistos.

Clase Vec2

Como en esta asignatura vamos a trabajar sobre el plano euclídeo, gran parte de las tareas que vamos a realizar requieren describir puntos en el plano euclídeo. Es por ello por lo que conviene crear una clase **`Vec2`** que maneje vectores de 2 coordenadas. Debería tener componentes **`x`** e **`y`**, de tipo **`float`**, y los constructores, operadores y métodos que consideremos necesarios.
Algunas tareas comunes para realizar con vectores son:

- Suma, resta, producto y división (tanto con otro vector como con un escalar).
- Valor absoluto.
- Longitud.
- Normal.
- Producto escalar.

Implementar la clase teniendo en cuenta estas tareas. Además, es necesario implementar estos dos métodos:

```
float angle(const Vec2& other) const;
float distance(const Vec2& other) const;
```

Bucle principal

Vamos a dibujar los siguientes elementos:

- Un cuadrado en el centro de la pantalla.
- Otro cuadrado en las coordenadas del ratón.

- Un círculo que rota alrededor del ratón 32 grados por segundo, a una distancia del mismo definida por el alumno.
- Añadir tus iniciales hechas con líneas (Ejemplo: ALX).

En el título de la ventana, vamos a escribir la siguiente información:

- La distancia entre el puntero del ratón y el centro de la pantalla
- El ángulo entre el puntero del ratón y el círculo que orbita a su alrededor.
- Vuestras iniciales.

Ambos valores se deben calcular, respectivamente, con los métodos `angle` y `distance` de `Vec2`.

El ángulo y la distancia son valores numéricos. Para imprimirlos en la ventana, debemos convertirlos en textos. Podemos hacer una función que haga esta tarea apoyándonos en la librería estándar:

```
#include <iostream>

template <typename T>
std::string stringFromNumber(T val) {
	std::ostringstream stream;
	stream << std::fixed << val;
	return stream.str();
}
```