Programación 2D
Práctica 1 : Dibujo de primitivas gráficas. Cálculos trigonométricos

En este primer ejercicio, aprenderemos a activar los estados de OpenGL necesarios y configurar la matriz de proyección de OpenGL. Una vez hecho esto, trabajaremos sobre algunas de las cosas aprendidas en el tema 1 de la asignatura, básicamente el cálculo de ángulos y distancias entre dos puntos, y la realización de movimientos circulares.

La activación de estados, configuración de la matriz de proyección, y paso a la matriz de modelado, los realizaremos en la clase `Screen` del motor, dentro del método `Open`.

Se proporciona el código que abre la ventana de OpenGL ya implementado (utilizando la librería GLFW). Tras esto, los pasos que debemos implementar nosotros son:

- Inicializamos estados de OpenGL.
- Configuramos el viewport.
- Configuramos la matriz de proyección.
- Configuramos la matriz de modelado.

En la inicialización de estados, realizaremos las llamadas a `glEnable` y `glEnableClientState` necesarias para activar los siguientes estados:

- Lectura del array de vértices.
- Lectura del array de coordenadas de textura.
- Renderizado de texturas.
- Mezclado.

El ancho y alto del viewport de OpenGL se configurará para que ocupe toda la pantalla.

El paso de un modo de matriz a otro lo realizaremos con `glMatrixMode`. Cuando cambiamos de matriz, el primer paso a la hora de configurarla es cargar la identidad, de forma que partamos de una transformación neutral.

Para establecer una proyección 2D, debemos configurar la matriz correspondiente con la función `glOrtho`, que tiene los siguientes argumentos:

- Izquierda
- Derecha
- Abajo
- Arriba
- Profundidad inicio (poner el valor a 0)
- Profundidad fin (dar un valor alto, por ejemplo 1000).

OpenGL utiliza por defecto un sistema de coordenadas cartesianas estándar, con el eje de ordenadas apuntando hacia arriba. Es habitual que en programación 2D manejemos un sistema de coordenadas cuyo eje Y apunta hacia abajo, de forma que las coordenadas parten de la esquina superior izquierda de la pantalla. Esto lo podemos conseguir intercambiando los valores de los parámetros “Abajo” y “Arriba”, lo que nos creará un espejado vertical del sistema de coordenadas de OpenGL.

Para la matriz de modelado, no necesitamos realizar más transformaciones que la carga de la matriz identidad.

Para el use de las funciones trigonométricas, debemos implementar dos funciones del módulo `Math` del motor. Estas funciones son:

- `Angle`: Utilizaremos la función arcotangente (tenemos una versión en este módulo llamada `DegATan2` que utiliza grados sexagesimales en lugar de radianes) para devolver el ángulo entre los dos puntos pasados como parámetro. Debemos utilizar la función `WrapValue` de este mismo módulo para encapsular el resultado en un valor en el rango [0, 360) (esta función es similar al operador `%` (módulo) de C o a la función `modf`, pero utiliza valores reales, y asegura que el resultado devuelto sea siempre positivo).
- `Distance`: Se debe aplicar el Teorema de Pitágoras para, dados dos puntos, obtener la distancia entre ellos.

Una vez añadido esto al motor, se debe poner a prueba. Realizaremos un programa a partir del siguiente código:

```
int main(int argc, char* argv[]) {
	Screen::Instance().Open(800, 600, false);

	while ( Screen::Instance().IsOpened() /* TAREA: Salida con ESC */ ) {
		// TAREA: Calcular coordenadas del círculo

		// TAREA: Escribir título de la ventana
```

```
            // TAREA: Dibujar primitivas

            // Refrescamos la pantalla
            Screen::Instance().Refresh();
      }

      return 0;
}
```

En la condición del `while`, debemos incluir el código necesario para que la aplicación termine al pulsar la tecla Esc además de cuando se cierre la ventana.

Dentro del bucle while, debemos dibujar los siguientes elementos:

- Un cuadrado en el centro de la pantalla.
- Otro cuadrado en las coordenadas del puntero del ratón.
- Un círculo que dé vueltas alrededor del cuadrado que representa el puntero del ratón.

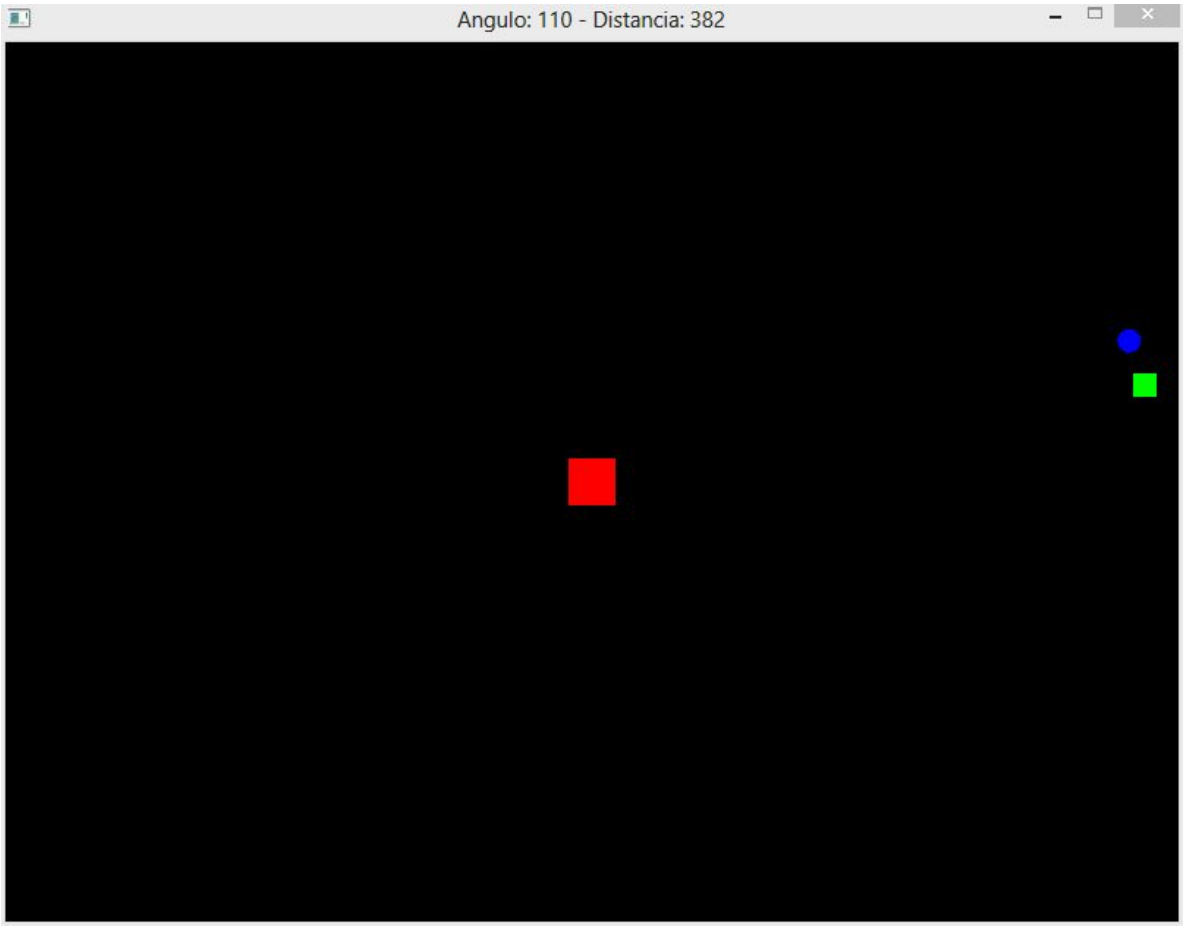
Para calcular las coordenadas del círculo, utilizaremos las funciones `DegSin` y `DegCos` del módulo `Math` para, dado un ángulo en grados sexagesimales, calcular su seno y su coseno.

Además, en el título de la ventana, utilizando la función `SetTitle` de la clase `Screen`, escribiremos la siguiente información:

- El ángulo que forma el círculo con respecto de las coordenadas del ratón (utilizando la función `Angle`).
- La distancia del puntero del ratón al centro de la pantalla (utilizando la función `Distance`).

Para convertir estos valores numéricos en objetos `String`, podemos utilizar el método estático `FromInt` de la clase `String`.

A continuación, mostramos una captura con la práctica resuelta:


Angulo: 110 - Distancia: 382