Interacción

Boletín de la práctica 6

Objetivo

Esta práctica tiene como objetivo la generación de un terreno 3D y la implementación del movimiento de sobrevuelo controlado por las acciones del usuario.

El programador tiene total libertad de implementación siempre que se cumplan los requisitos indicados.

Fase 1. Terreno base

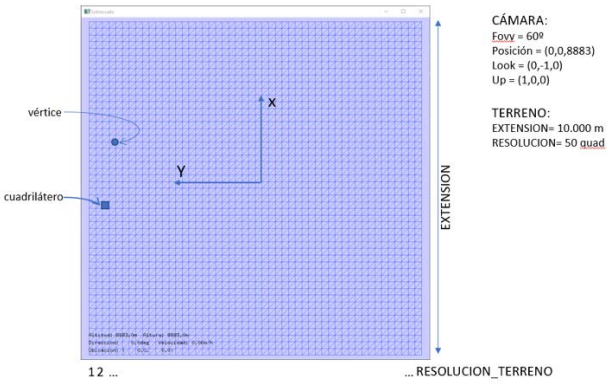
Se debe construir una malla regular de vértices en el plano *XY*, centrada en el origen, que va a ser la base de un terreno. La coordenada *z* se considera la altura, inicialmente a cero.

Para la construcción de la malla se empleará cualquier primitiva de OpenGL, (*GL_POLYGON, GL_TRIANGLE_STRIP, GL_QUAD_STRIP, ….)* o la función '*quad()*' del fichero '*Utilidades.h*', a decisión del programador. Las características de la malla deben ser:

- `const int EXTENSION` = … // Longitud del lado de la malla en metros
- `const int RESOLUCION_TERRENO` = … // Número de polígonos en cada lado de la malla

Se debe construir un programa que visualice la malla completa en alámbrico ajustando las características del tipo, posición y orientación de la cámara.

La figura muestra un ejemplo de lo que se pide:



Fase 2. Interacción

Se sitúa la cámara inicialmente en el origen a cierta altura del suelo (*z* cámara inicial) y mirando en la dirección del eje *X*. La cámara tiene una velocidad que se debe caracterizar por un ángulo que da la dirección del movimiento (ángulo respecto a X) y un módulo (m/sg o km/h) que indica la magnitud. El avance de la cámara es, en su forma básica, paralelo al plano *XY*.

Se quiere que la aplicación responda a las siguientes acciones del usuario al pulsar en el teclado:

- flecha arriba: `subir()` // Aumenta la z de la cámara
- flecha abajo: `bajar()` // Disminuye la z de la cámara
- flecha izquierda: `girarIzquierda()` // Aumenta el ángulo del vector velocidad
- flecha derecha: `girarDerecha()` // Disminuye el ángulo del vector velocidad
- 'a': `acelerar()` // Aumenta el módulo de la velocidad
- 'z': `frenar()` // Disminuye el módulo de la velocidad
- 'q': pilotoAuto() // Mantiene la distancia al suelo

Al arrastrar el ratón con el botón izquierdo pulsado (*dragging*) la dirección hacia donde mira la cámara debe cambiar así:

- arrastre horizontal: la cámara cambia la dirección en la que mira horizontalmente desde -90º a 90º siendo 0º la dirección en la que avanza el móvil/observador (movimiento de guiñada)
- arrastre vertical: la cámara cambia la dirección en la que mira verticalmente desde -89º a 45º siendo 0º la dirección paralela al plano XY (movimiento de cabeceo)

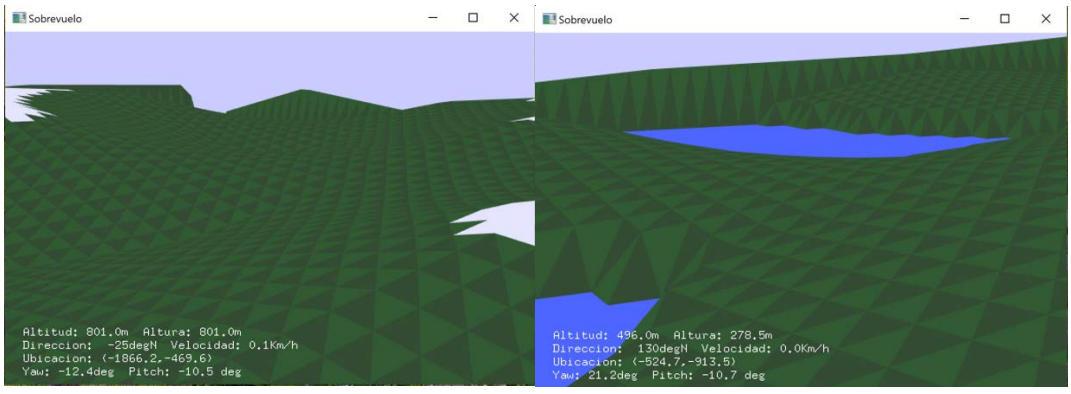
La velocidad se mantiene constante en módulo y dirección si no se pulsa ninguna tecla. La cámara siempre avanza en la dirección del movimiento (dirección velocidad) coherente con el tiempo transcurrido y su velocidad actual.

Fase 3. Elevación del terreno

Se debe implementar una función que, para todo *x*,*y*, devuelva una elevación *z* del terreno:

```
float elevacion(const float x, const float y);
```

Se dotará a la malla generada de elevación para cada uno de sus vértices con el objetivo de generar un terreno montañoso. Para la visualización del terreno se optará por polígonos rellenos de color a elegir que permita comprender las distintas elevaciones del terreno según se recorre con la interacción descrita en fase 2. No se pretende una visualización realista en esta fase sino simplemente comprender la forma del terreno. Las imágenes muestran un ejemplo de lo que se persigue:



En este vídeo se muestra la respuesta del movimiento a las acciones del usuario.

Notas adicionales

Se deben cumplir los requisitos de interacción, la coherencia temporal y el objetivo general de disponer de un terreno por el que moverse. Más allá de esto, el programador tomará las decisiones que crea oportunas para la implementación (dimensiones, colores, incrementos de ángulos y magnitudes, etc.)

Se debe valorar la resolución frente al rendimiento para una visualización fluida.

Los ejemplos son orientativos, no deben reproducirse.

La práctica constituye la base de las 3 siguientes. Es muy importante que funcione bien pues formará parte de la entrega del segundo trabajo puntuable (práctica 9).

EL TRABAJO ES INDIVIDUAL, Y AUNQUE SE FOMENTA EL INTERCAMBIO DE IDEAS, EL DE CÓDIGO NO ESTÁ PERMITIDO.