Ejercicio discretización

Sara Martín Aramburu

June 7, 2022

Abstract

Documentación de una aplicación implementada en C y documentada mediante Latex.

1 Objetivos de nuestra aplicación

Esta maravillosa aplicación tiene como objetivo insertar una foto dentro de un triángulo dado, y deformarla según este. Para ello hemos partido de un código en lenguaje de programación C dado por el profesorado.
Otro punto importante es el aprendizaje a la hora de desarrollar un programa a partir de unas pauta específicas dadas. En este caso ha sido el profesorado quien nos ha guiado para que pudieramos lograr la ejecución de este.

2 Sobre este documento

Se trata de un documento para comentar el trabajo realizado en este trabajo y especificar todo su contenido.Además, también sirve para mostrar los nuevos conocimientos adquiridos y futuros objetivos.

3 Interacción con la usuaria

Las opciones que proporciona nuestra aplicación son:

1. Tecla Enter

Una de las pocas cosas que permite hacer nuestro programa es pulsar dicha tecla. La usamos para movernos entre los triángulos. Cada vez que la pulsemos, aparecerá un nuevo triángulo en nuestra ventana.4

2. Tecla Esc

   Al pulsar la tecla ESC, el programa se cierra.

Ambas funcionalidades están implementadas en la funcion "teklatua", dentro del fichero "dibujar-puntos.c". La función es una de las que se nos dieron inicialmente, pero hemos tenido que modificar el caso en el que se pulsa la tecla "Enter" para que pase al siguiente triángulo de la lista.

```
static void teklatua(unsigned char key, int x, int y){
switch(key){
    case 13:
        printf("Hay que dibujar el siguiente triangulo.\n);
        if (indice!=num_triangles-1){
            indice++;
        }
        else{
            indice=0;
        }
        break;
        }
    }
```

Como podemos ver, controlamos el índice de la lista de triangulos. Cada vez que pulsamos enter, el índice aumenta, por lo que pasamos al siguiente triángulo. Si la lista llega al final, volvemos a empezar desde el principio, asignando el número 0 al índice.

4 Ficheros de objetos tridimensionales

En esta aplicación utilizamos un fichero, llamado "triangles.txt", en el que escribimos los datos y coordenadas de cada triángulo. También requerimos una foto en formato .ppm para deformar dentro del polígono.

```
t 1 1 1 0.1 0.1 200 400 0 0.4 0.8 450 10 0 0.9 0.1
```

Como podemos observar, escribimos una "t" al principio para indicar que empieza un nuevo triangulo. Los primeros 5 números son las coordenadas x,y,z,u,v del primero punto, y asi sucesivamente con todos los puntos.

5 Algoritmo

Lo primero que hacemos es ordenar los puntos en función de su coordenada Y: el superior, el inferior y el del medio.
A continuación, miramos si es un caso especial o no.
Después, dividimos el triángulo en dos, para pintar primero una sección y después otra. En cada altura, calculamos ambas intersecciones entre las dos aristas y el punto, y dibujamos el segmento entre las dos intersecciones. Realizamos lo mismo con la otra sección del triángulo.

6 Código C

Aparte de las funciones dadas como base de la aplicación, hemos tenido que implementar alguna más.

1. Color textura
   Esta función sirve para saber el color de cada píxel y poder usarlo más adelante para insertar la foto.
   ```
   unsigned char * color_textura(float u, float v)
   ```

2. Calcular intersección
   Lo que se hace en esta función es calcular la intersección entre dos puntos pasados como parámetros. Las coordenadas de la intersección se guardan en la variable "intersección", y j es la altura a la que queremos calcular el punto de corte. Es una función muy importante, ya que si funciona mal, no se pintarán correctamente los triángulos.
   ```
   void calcularinterseccion(punto *sup_ptr, punto *inf_ptr,
   punto *interseccion, float j)
   ```

3. Intercambiar
   Sirve para intercambiar dos puntos después de compararlos. Por ejemplo, si el punto "superior" es el inferior y demás casos. Los parámetros necesarios son los dos puntos que hay que cambiar.
   ```
   void intercambiar(punto *p1,punto *p2)
   ```

4. Dibujar segmento
   El pilar del programa. Es necesario ir pintando poco a poco el triángulo, por lo que esta función es necesaria para pintar sus segmentos. Son necesarias dos intersecciones para poder pintar.

```
void dibujar_segmento( punto *interseccion1, punto *interseccion2,
punto *sup, punto *mid, punto *inf)
```

5. Dibujar triángulo
   En esta función se combinan "Dibujar segmento" y "Calcular intersección" para formar el triángulo. Es importante comparar los puntos para saber cual es el superior, inferior y el del medio. Se pasa como parámetro un puntero que apunta a un triángulo (que es el que tenemos que dibujar).
   ```
   static void dibujar_triangulo( hiruki *triangulosptr)
   ```

6. Calcular coordenadas baricéntricas
   Consiste en realizar un cálculo para calcular las coordenadas baricéntricas y poder coger el color de la textura.
   ```
   void calcularbaricentricas( punto *interseccion1, punto *interseccion2,
   punto *sup, punto *mid, punto *inf)
   ```

7. Calcular color de textura
   Teniendo en cuenta las coordenadas de la textura, hallamos su color.
   ```
   void calcularuv( punto *p1, punto *p2, punto *p3, float *u, float *v)
   ```

7 Dificultades y obstáculos encontrados

Principalmente se me hizo complicado empezar a programar, ya que tenía la idea del algoritmo pero no sabía como pasarlo a código, por el hecho de que apenas sabía programar en C (ya que no lo hemos utilizado demasiado).
Cuando conseguí programarlo, tuve que arreglar muchísimo errores y modificar muchas cosas. Me costó que apareciera el triángulo en la ventana. Además, no se me rellenaba, y estuve mucho tiempo con eso.
Otro problema muy grande fue la programación de casos especiales del triángulo. Hay muchos, y aunque escribía diferentes soluciones, siempre había alguno que seguía sin dibujarse correctamente.
El tema de insertar la foto ha sido una completa odisea. Lo más complicado fue insertar la foto y deformarla, pero después de estar bastante tiempo trabajando en ello, conseguí que funcionara correctamente.

8 Trabajo futuro

Los objetivos que propuse la última vez han sido alcanzados, por lo que no hay nada más en lo que trabajar. El funcionamiento del programa es correcto.

References

[1] Transparencias de Egela, *Transparencias-discretización*, 2023