 **Pontificia Universidad Javeriana**

Facultad de ingeniería

Estructuras de datos

Alejandro Mayorga

Diego Guevara

**Documento entrega 2**

En el presente informe se realizara la descripción de la implementación de los comandos necesarios para la entrega 2 del proyecto.

Para esta entrega se realizaron 3 comandos básicos cuyo objetivo es el cálculo de distancias entre vértices con la finalidad de obtener mayor información de las mayas 3D.

Comandos:

**v\_cercano px py pz nombre\_objeto:**

Pretende calcular la distancia entre un punto dado con coordenadas XYZ y un vértice existente en un polígono cuya única restricción es que sea el más cercano al punto dado.

Representación en código:

float\* componente2Punto1(string nombrePoligono, list<poligono> poligonosMemoria, float x, float y, float z, bool on);

Entradas:

px, un numero racional positivo que denota la variable X (ancho) en un espacio de 3 dimensiones.

py, un numero racional positivo que denota la variable Y (largo) en un espacio de 3 dimensiones.

pz, un numero racional positivo que denota la variable Z (alto) en un espacio de 3 dimensiones.

nombre\_objeto: nombre que denota un polígono cargado previamente en memoria para ser buscado y procesado.

Salidas:

La distancia a la que se el punto dado y el vértice más cercano al polígono, sin embargo indirectamente también da información de cuáles son las coordenadas del vértice del polígono y su índice mediante impresiones por pantalla.

Paso a Paso:

1. Busca el nombre del polígono en la memoria de polígonos insertados.
2. Si no está, termina la ejecución.
3. Extrae del polígono encontrado su lista de vértices.
4. Recorre la lista de vértices.
5. Para cada elemento, extrae las coordenadas x, y yz y llama una función de cálculo de distancia euclidiana.
6. Ingresa la distancia resultante en un árbol
7. Si el parámetro ON de entrada se encuentra como true, llamara la función del árbol que retorna el valor más a la izquierda de este (ósea el mínimo) y retornara dicho valor
8. Si el parámetro On se encuentra como falso, volverá a recorrer la lista de vértices y volverá a hacer el cálculo de distancias por cada uno
9. Si la en dicho vértice la distancia euclidiana es igual, imprimirá los las coordenadas de el vértice, el nombre del objeto y acabara la ejecución de la función.

**v\_cercano px py pz:**

Busca entre todos los objetos cargados en memoria, el que tenga el vértice de menor distancia entre un punto dado e imprimirá su información.

Representación en código:

void componente2Punto2(list<poligono> poligonosMemoria, float x, float y, float z);

Entradas:

px, un numero racional positivo que denota la variable X (ancho) en un espacio de 3 dimensiones.

py, un numero racional positivo que denota la variable Y (largo) en un espacio de 3 dimensiones.

pz, un numero racional positivo que denota la variable Z (alto) en un espacio de 3 dimensiones.

Salida: la información impresa en pantalla sobre el nombre del objeto cuyo vértice esta a menor distancia al punto dado y, las coordenadas de dicho vértice.

Paso a paso:

1. Recorre la lista de polígonos en memoria.
2. Para cada polígono, obtiene su nombre y llama la función componente2Punto1 descrita anteriormente., insertando el valor que retorne (una distancia) en un árbol binario.
3. Una vez haya recorrido toda la lista la vuelve a recorrer.
4. Para cada elemento compara el valor mínimo del árbol binario (el que está más a la izquierda) con un nuevo llamado de la función componente2Punto1 sobre cada polígono.
5. Si las distancias son iguales, imprimirá toda la información de dicho polígono y terminara la ejecución.

**v\_cercanos\_caja nombre\_objeto:**

Dado el nombre de un objeto, se calculara la envolvente de dicho objeto y a partir de esta y sus 8 esquinas, se buscara los vértices del polígono que encapsula, que estén a menor distancia de las esquinas, así cada esquina, tendrá un vértice correspondiente más cercano.

Representación en código:

void componente2Punto3(string nombrePoligono,list<poligono> poligonosMemoria)

Entradas:

nombre\_objeto: nombre que denota un polígono cargado previamente en memoria para ser buscado y procesado.

Salida: la información impresa en pantalla sobre el los 8 vértices que estén más cerca a las 8 esquinas de la envolvente.

Paso a paso:

1. Buscar el nombre del polígono en la lista de polígonos cargados en memoria.
2. Si se encuentra, extraer de este la correspondiente lista de vértices, sino se termina la ejecución.
3. Recorrer la lista de vértices
4. Para cada elemento de la lista, llamar la función componente2Punto1 descrita previamente.
5. Una vez recorridos todos los elementos de la lista se finaliza la ejecución.

Funciones adicional:

calcularDistancia:

Dados dos pares de coordenadas XYZ calcular la distancia euclidiana entre estas.

Representación en código:

float\* calcularDistancia( float x, float y, float z, float x2, float y2, float z2)

Esta función solo aplica la fórmula de distancia euclidiana y la retorna.

Formula de distancia euclidiana para coordenadas en 3 dimensiones:

Distancia euclideana espacio.gif

Equivalente en código:

abs(sqrt( pow((x2 - x),2) + pow((y2 - y),2) + pow((z2 - z),2)));