Computación Gráfica Marc-Antoine Le Guen	TP 5 - Modelo digital del terreno

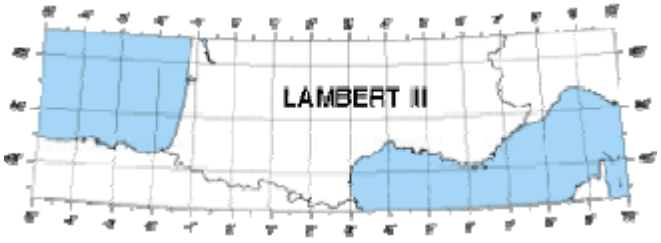
Se usara la cámara ya implementada en la práctica anterior y el sistema de iluminación.

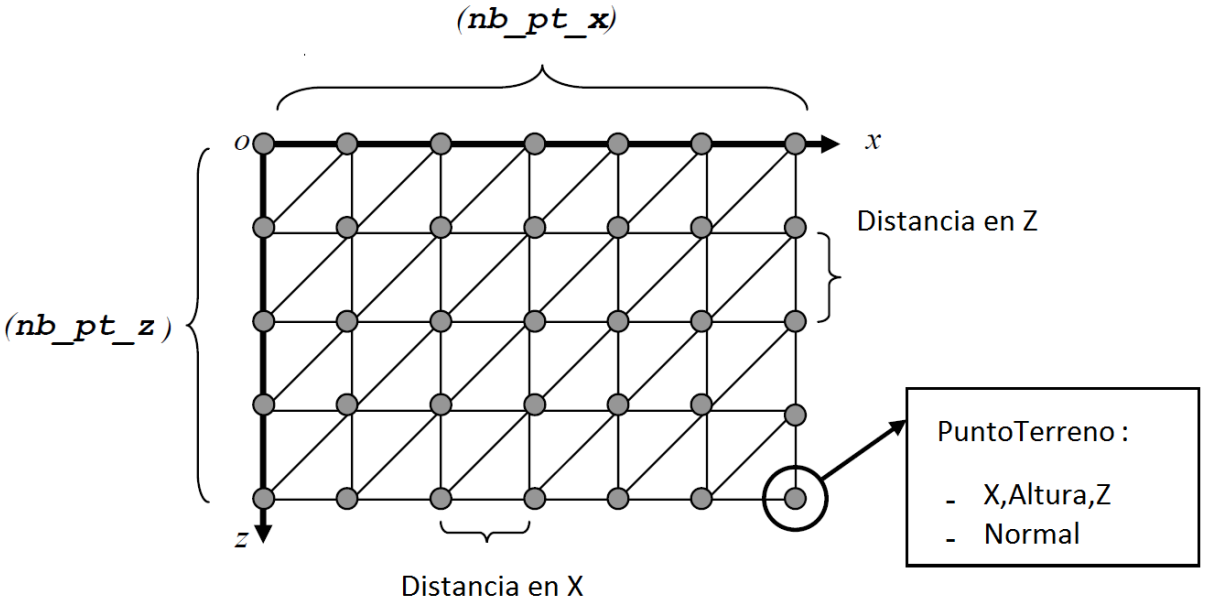
Escribirán una clase Terreno que permite la creación y la visualización de un terreno en 3D constituido de una malla regular de triángulos. Las declaraciones de esta clase están dados en el Anexo. Construcción del terreno

1. Creación
   Escribir el método load(char* filename) de la clase Terreno, asignaran y llenaran la tabla lista_puntos leyendo cada altitud de cada vértice de los triángulos (coordenada en y) en el archivo de texto al formato MNT. En este archivo el terreno esta discretizado de manera uniforme con un punto cada 50 metros. En el formato del archivo cada línea equivale a una línea del terreno. Pueden abrir este archivo con un editor de texto para ver como los datos están almacenados. Algunas informaciones les interesaran en la primera línea:

   MNT 1.0 dep_13 metre 0.01 790000.00(1) 165000.00(2) 50.00(3) -50.00(4) 101(5) 101(6) metre 1

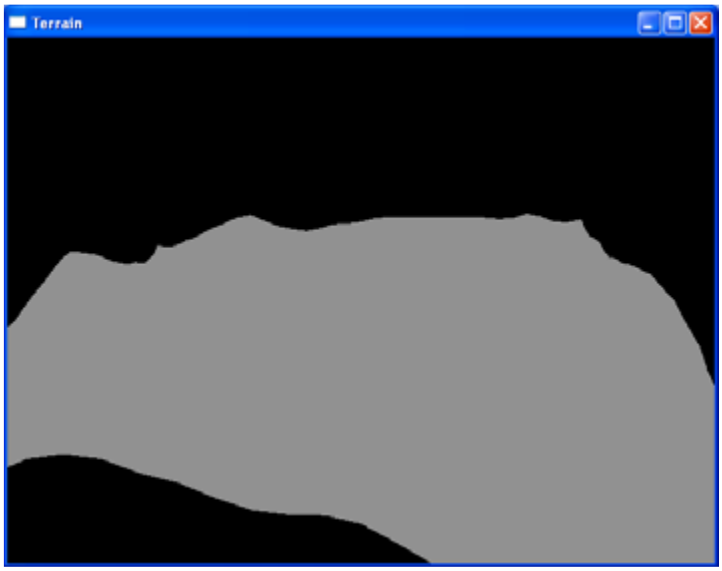
   (1) coordenada x en Lambert III de la esquina noroeste
   (2) coordenada y en Lambert III de la esquina noroeste
   (3) distancia entre dos puntos segun X
   (4) distancia entre dos puntos segun Z
   (5) Cantidad de puntos en x
   (6) Cantidad de puntos en z





2. Visualización

Implementaran la función display(). Visualizaran el terreno gracias a un VertexArray de GL_TRIANGLES, ver en Anexo.



3. Textura

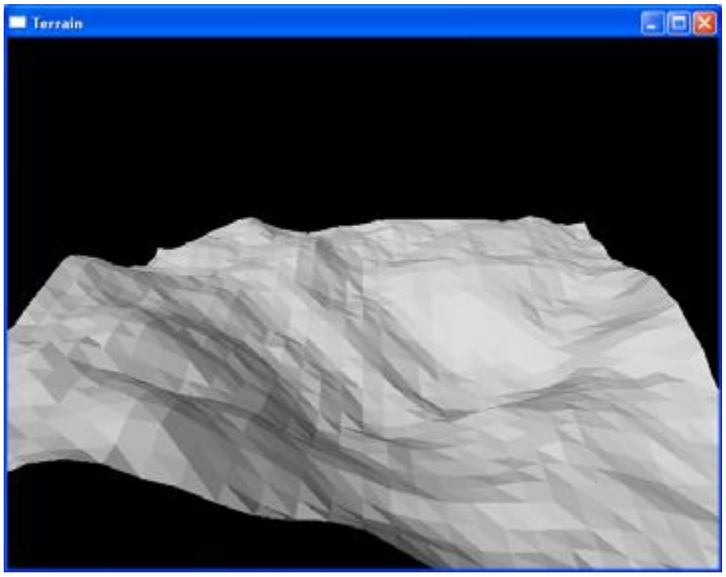
Usaran la clase TextureManager y usaran la textura fontvieille.tga.

Implementaran una función computeTexCoord() que calcula las coordenadas de textura S y T por cada vértice del terreno. Queremos que la textura sea estirada sobre toda la superficie del terreno.

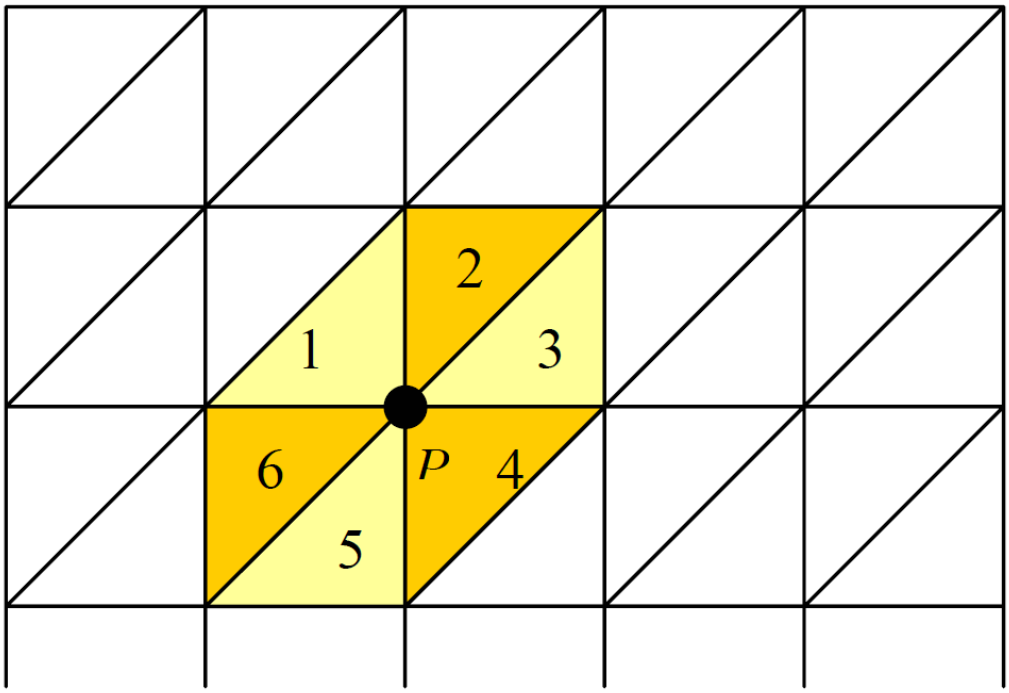
Modificaran la función display() para que tome en cuenta la textura.

4. Calculo de las normales

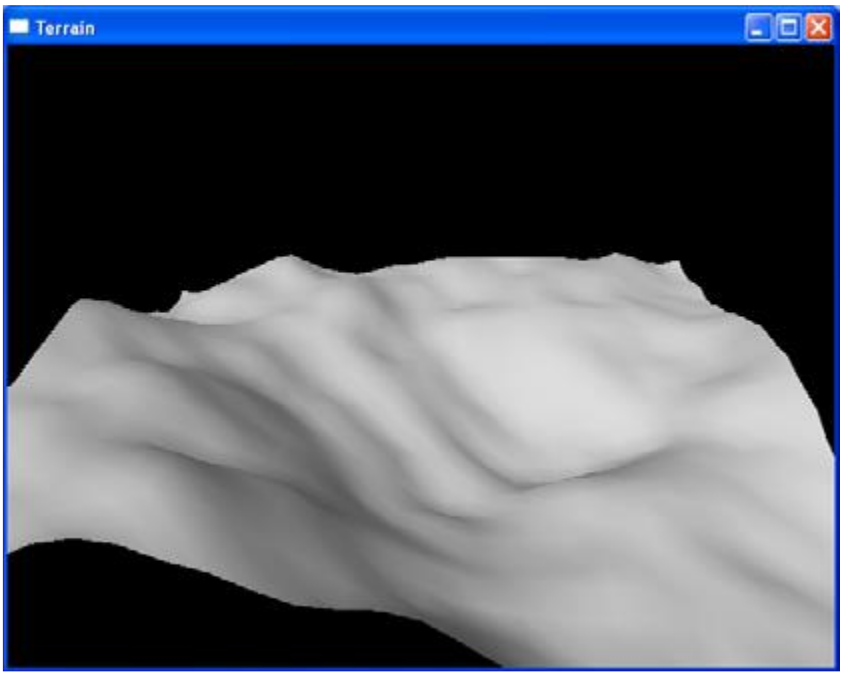
El terreno hasta ahora no está correctamente iluminado. Hay que precisar la normal de cada triangulo a visualizar pero si especiamos un normal por cada triangulo todos los pixeles del mismo triangulo tendrán el mismo color.



Para tener un aspecto liso de la superficie del terreno, la solución consiste en asociar una normal para cada vértice ya no por cada triangulo. Escribir una función computeNormals() que permite calcular una normal por cada punto de lista_puntos, la normal del punto P es el promedio de las normales de los triángulos del vecindario de P. Esta normal se guardara en nx,ny,nz de la estructura PuntoTerreno.



Sobre el ejemplo arriba el punto P tiene 6 triángulos en su vecindad, así su normal será el promedio las seis normales de estos triángulos. Obtendremos así una superficie más lisa.



Anexo

```
typedef struct // Definicion de un punto del terreo
{
    GLfloat s, t; // Coordenadas de texturas
    GLfloat nx, ny, nz; // Coordenadas de la normal
    GLfloat x, hauteur, z; // posición del vértice
} PuntoTerreno;

class Terreno
{
    public :
        Terreno();
        ~Terreno();
        bool load(char *filename) ; // carga de un archivo de modelo
digital de terreno
        void display(); // Visualizacion del terreno
        void computeNormals(); // calcula las normales de cada vertice
    private:
        int nb_pt_x, nb_pt_z; // quantidad de punto en X y Z
        float dist_x, dist_z; // distancia entre dos puntos segun X y
Z
        PuntoTerreno *lista_puntos; // Tabla que contiene los puntos
del terreno
        GLuint *lista_indices ; // Tabla que contiene los indicess
};

void Terreno::display()
{
    // glInterleavedArrays(GL_T2F_N3F_V3F, sizeof(PuntoTerreno),
&lista puntos[0].s);
    //glDrawElements(GL_TRIANGLES, count, GL_UNSIGNED_INT,
(void*)lista_indices);
}
```