Introduction à la modélisation 3D

Travaux pratiques

Au début de ce TP/TD, vous recevrez une archive zip contenant une base de code. Ce code permet d'afficher un *maillage triangulaire* à l'aide d'openGL.

1. Nous commencerons par l'analyser ensemble pour vous familiariser avec.
2. Vous devez faire évoluer ce code au fur et à mesure du TP, pour répondre aux questions.

1 Base de code

Téléchargez l'archive.
Nous allons le compiler et l'analyser ensemble dans un premier temps.

2 Rendu de maillages

Le fichier tp.cpp contient une méthode `drawTriangleMesh()`, qui contient du code openGL basique permettant d'afficher un maillage triangulaire. Ce code est obsolète (non compatible avec les versions récentes d'openGL).

```
void drawTriangleMesh( Mesh const & i_mesh ) {
    // This code is deprecated.
    glBegin(GL_TRIANGLES);
    for(unsigned int tIt = 0 ; tIt < i_mesh.triangles.size(); ++tIt) {
        Vec3 p0 = i_mesh.vertices[i_mesh.triangles[tIt][0]];
        Vec3 n0 = i_mesh.normals[i_mesh.triangles[tIt][0]];

        Vec3 p1 = i_mesh.vertices[i_mesh.triangles[tIt][1]];
        Vec3 n1 = i_mesh.normals[i_mesh.triangles[tIt][1]];

        Vec3 p2 = i_mesh.vertices[i_mesh.triangles[tIt][2]];
        Vec3 n2 = i_mesh.normals[i_mesh.triangles[tIt][2]];

        glNormal3f( n0[0] , n0[1] , n0[2] );
        glVertex3f( p0[0] , p0[1] , p0[2] );
        glNormal3f( n1[0] , n1[1] , n1[2] );
        glVertex3f( p1[0] , p1[1] , p1[2] );
        glNormal3f( n2[0] , n2[1] , n2[2] );
        SglVertex3f( p2[0] , p2[1] , p2[2] );
}
glEnd();
}
```

3 Exercice 1 : création d'un maillage triangulaire de sphère 3D

Dans le fichier tp.cpp, compléter la fonction `void setUnitSphere(Mesh & o_mesh( int nX, int nY )`, qui créera un maillage triangulaire de sphère 3D.

1. Indications : un point 3D sur la sphère peut être obtenue à l'aide de la paramétrisation sphérique fonction de deux angles $(\theta, \phi) \in [0, 2\pi] \times [-\pi/2, \pi/2]$:
   — $x = cos(\theta) * cos(\phi)$
   — $y = sin(\theta) * cos(\phi)$
   — $z = sin(\phi)$
2. Ajouter la fonctionnalité suivante : l'appui de la touche "-" augmente le nombre de méridiens et celui de parallèles de 1. De la même manière l'appuie sur la touche "-" diminue de 1 le nombre de méridiens et celui de parallèle.

4 Exercice 2 : génération de terrain

Dans le fichier tp.cpp, compléter la fonction `void setTesselatedSquare(Mesh & o_mesh( int nX, int nY )`, qui créera un maillage triangulaire d'un plan horizontal subdivisé en nX × nY points.

1. Indication : passez d'un rendu surface à un rendu filaire à l'appui sur "w".
2. Utiliser la fonction `float GetNoise(float x, float y)` de l'objet "noise" pour modifier la position z de chaque sommet avec une valeur aléatoire comprise entre -1 et 1.
3. Calculer les normales pour chaque sommet. Vous pouvez calculer la normal par sommet ou par triangle.
4. Ajoutez de la couleur en fonction de la direction de la normale : si la normale est verticale, colorez le sommet en vert, sinon en brun.
5. Ajouter la fonctionalité suivante : l'appui sur les touches "a" et "d" ajoutent un décalage dans le calcul du bruit dans la direction X. Les touches "z" et "s" ajoutent un décalage en Y.