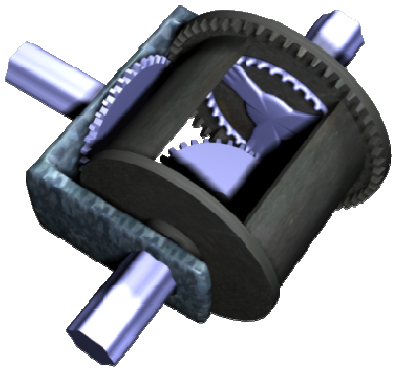
Master 1 informatique
Multimédia

I But

On désire réaliser une animation en images de synthèse stéréoscopiques pour montrer le fonctionnement d'un système physique : le différentiel. Cette pièce mécanique permet à un essieu moteur de véhicule de transmettre une vitesse différente aux roues intérieures et extérieures dans un virage.

Le but de ce TP est d'appliquer le modèle de synthèse d'images stéréoscopique en temps réel sur un modèle géométrique complexe défini sous 3DS puis converti en MD3.

II Le modèle géométrique sous 3DS



La scène ci-contre a été réalisée sous 3DS Max puis chaque élément a été isolé puis exporté au format MD3 (format géométrique simple défini pour le jeu vidéo Quake 3). Les éléments sont :

- Le chassis
- Le bloc différentiel, muni d'un engrenage de 106 dents, diamètre 90
- Les engrenages de type 1 de 72 dents recourbées, diamètre 60
- Les engrenages de type 2 de 72 dents droites, diamètre 60
- Les axes (diamètre 20, longueur 50).

III Sous OpenGL

Le programme initial, fourni sur le site http://www-igm.univ-mlv.fr/~piranda, permet de charger les fichiers MD3 en mémoire et de les visualiser. On utilisera pour cela la classe **`md3::Model`** et son constructeur **`Model(repertoire, fichier_md3)`** qui charge le fichier **`fichier_md3`**. La chaîne de caractères **`repertoire`** permet d'indiquer l'emplacement des fichiers de textures. La méthode **`glDraw()`** permet de visualiser l'objet sous OpenGL.

1. Visualisation des éléments

Compilez et exécutez le programme, vous pouvez observer un premier objet : le chassis.

Sous OpenGL, visualisez les différents éléments MD3 proposés. Identifiez leur repère propre et leurs dimensions.

2. Modèle complet

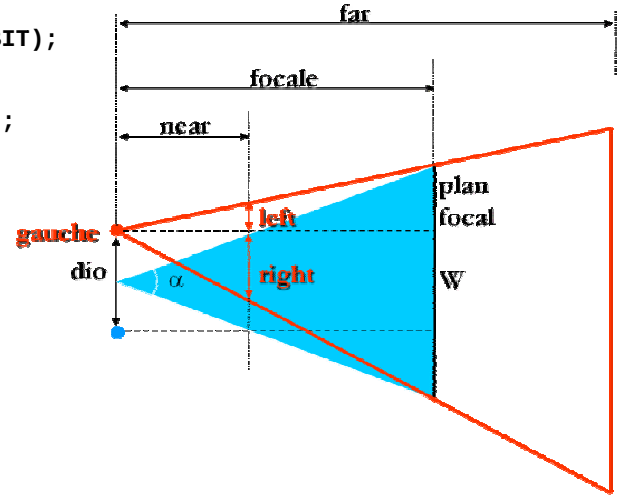
Reconstruisez le modèle complet puis animez ce modèle en ajoutant deux paramètres :

- alpha, l'angle de l'axe moteur
- alpha2, la différence angulaire entre les deux axes des roues.

3. Visualisation stéréoscopique

Modifiez le programme précédent pour visualisez les images de synthèse en stéréoscopie anaglyphe. Pour cela, vous utiliserez l'algorithme suivant :

```
// effacer l'écran et le Z-buffer
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
// OEIL GAUCHE
// activer le plan rouge uniquement
  glColorMask(GL_TRUE,GL_FALSE,GL_FALSE,GL_FALSE);
// activer l'écran décalé à gauche
  camera.activerCameraGauche();
// dessin de la scène
  drawScene();
// OEIL DROIT
// effacement du Z-buffer
  glClear(GL_DEPTH_BUFFER_BIT);
// activer les plans cyan uniquement
  glColorMask(GL_FALSE,GL_TRUE,GL_TRUE,GL_FALSE);
// activer l'écran décalé à droite
  camera.activerCameraDroite();
// dessin de la scène
  drawScene();
// réactiver tous les plans de couleurs RVB
  glColorMask(GL_TRUE,GL_TRUE,GL_TRUE,GL_FALSE);
```



On déduit de la figure précédente les relations suivantes :

$$left = near \times \tan\frac{\alpha}{2} - \frac{dio \times near}{2\,focal}$$

$$right = near \times \tan\frac{\alpha}{2} + \frac{dio \times near}{2\,focal}$$