Synthèse d’image

RAPPORT PROJET : MODELISATION TORTUE

Mohamed BOUCHENGUOUR
Mehdi ASNI

TP6

ANNÉE UNIVERSITAIRE 2022/2023

Table des matières

Introduction

La synthèse d'image permet de modéliser des objets en trois dimensions à partir de formes telles que des cubes, des sphères, des cylindres, etc. Ces formes peuvent être habillées avec des textures et des couleurs. Il est également possible d'éclairer et d'animer ces objets. Ceci permet de réaliser des jeux vidéo, des films, des dessins animés, etc.

Objectifs

Pour ce module, nous avions la tâche de réaliser une tortue en 3D seulement à l'aide de primitive (sphère, cylindre, cube, etc.), de l'habiller avec des textures, de l'animer et de l'éclairer en utilisant le langage c++, la librairie openGl ainsi que l'environnement de développement code blocks.

Hiérarchie

Pour modéliser une tortue, étudier les caractéristiques morphologiques d'une tortue est essentiel afin d'avoir une modélisation la plus réaliste possible. Après quelques recherches sur Internet, nous sommes parvenus à ce résultat :

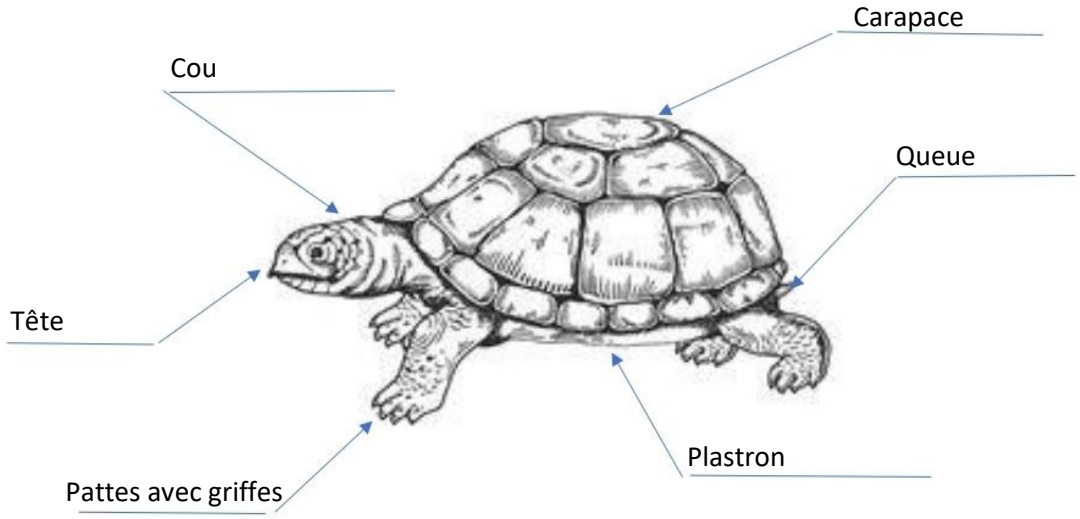


Figure 1 *- Les différentes parties du corps d'une tortue*

D’après ce schéma, nous avons donc choisi d’opter pour une hiérarchie comportant : carapace, plastron, queue, tête, cou, yeux, pattes et les griffes. Cette hiérarchie est décrite dans le schéma suivant :

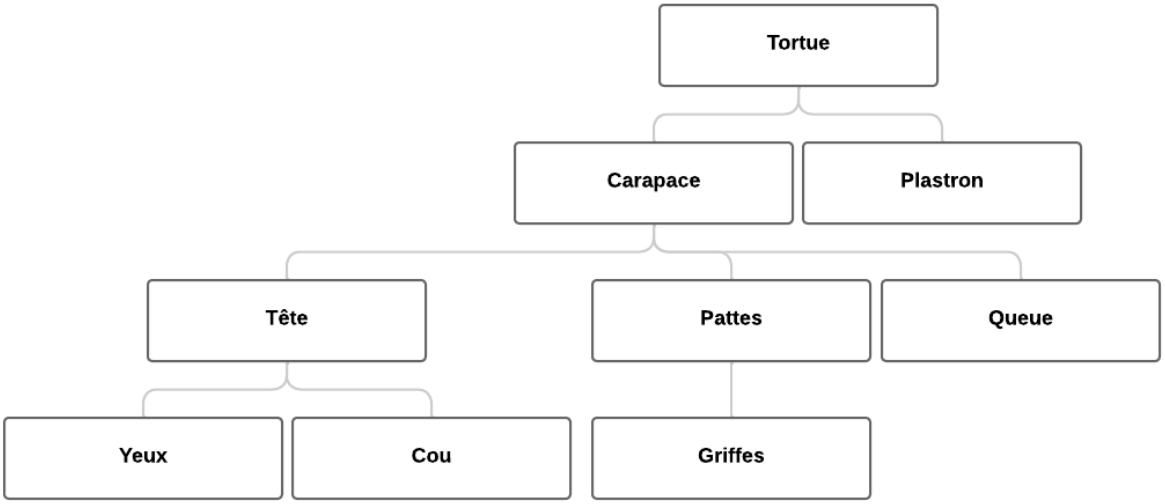


Figure 2 *- Arbre hiérarchique représentant les composants d’une tortue*

Modélisation

Nous allons maintenant présenter les primitives utilisées pour modéliser chaque partie de notre tortue. Toutes les primitives ont été mises dans une seule matrice qui représentera la tortue. Ceci nous sera utile pour notre animation.

La carapace

La carapace de la tortue est représentée par une demi-sphère à partir de la représentation paramétrique d’une sphère dans la méthode "void sphereCarapace(double r, int NM, int NP)" qui modélise une demi-sphère avec un rayon, un nombre de méridiens et de parallèles spécifiés dans les paramètres (respectivement r, NM et NP). Pour modéliser la demi-sphère, nous avons simplement modifié la boucle qui permet de tracer les polygones en remplaçant NM par NM/2. Notre demi-sphère a un rayon de 0.5 dans le but d’avoir une carapace de bonne taille. La carapace d'une tortue n'est pas complètement sphérique. Nous avons donc mis en paramètre des petites valeurs pour NM et NP (respectivement 8 et 7). Notre demi-sphère a été mise à l’échelle sur y et z afin d’avoir une forme ovale. L'application d'une texture sur cette demi-sphère se fera sur toute la primitive (enroulée).

Le plastron

Pour le plastron, nous avons procédé de la même façon que pour la carapace. Nous avons créé une deuxième fonction qui se nomme "void spherePlastron(double r, int NM, int NP)". Nous avons décidé d'implémenter une deuxième fonction demi-sphère car dans cette primitive, la texture sera appliquée sur chaque face. Notre demi-sphère a été également mise à l’échelle sur y et z avec une valeur plus faible sur y afin d’avoir une forme plus aplatie.

Le cou

Le cou de la tortue est représenté par un cylindre à partir de sa représentation paramétrique dans la méthode "void cylindre (double r, int n, double h)" qui modélise un cylindre avec un rayon, un nombre de face et une hauteur spécifiée dans les paramètres (respectivement r, n et h). Pour avoir un cylindre un peu moins rond, nous avons appliqué une mise à l'échelle sur x.

La tête

Notre tête est représentée par une sphère que nous avons créée à l'aide sa représentation paramétrique dans la méthode "void sphere(double r, int NM, int NP)" qui modélise une sphère avec un rayon, un nombre de méridiens et de parallèles spécifiés dans les paramètres (respectivement r, NM et NP). L'application d'une texture sera enroulée sur cette primitive. Pour éviter d'avoir une tête trop ronde, nous avons appliqué une mise à l'échelle sur x pour avoir une sphère un peu plus allongée.

Les yeux

Nous avons modélisé 2 sphères pour avoir 2 yeux avec la fonction prédéfinie glutSolidSphere.

Les pattes

Chaque patte a été représentée en deux parties : une première partie sous forme d'un cylindre, et une deuxième partie sous la forme d'une sphère avec une forte mise à échelle sur x et z afin d'avoir une sphère aplatie. Ces deux parties ont été respectivement modélisées à partir des méthodes "void cylindre (double r, int n, double h)" et "void sphere(double r, int NM, int NP)" (défini précédemment) pour les 4 pattes de notre tortue. Chaque patte aura cinq griffes. Le cylindre, la sphère ainsi que les cônes seront mises dans une matrice ce qui facilitera notre animation.

Les griffes

Chaque griffe a été représentée par un cône à partir de la méthode prédéfinie glutSolidCone.

La queue

Cette partie du corps a été représentée par un cône généré à partir de la méthode prédéfinie glutSolidCone.

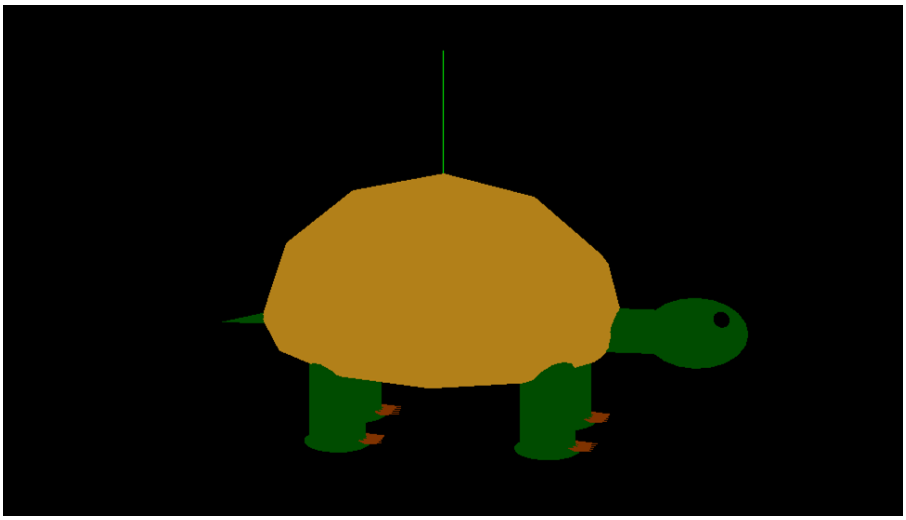


Figure 3 *- Le modèle obtenu après l'étape de la modélisation*

Habillage des primitives

Pour le chargement d'une texture à partir d'une image, nous avons utilisé la méthode loadJpegImage qui a pour paramètre le chemin de l'image et la texture (tableau) à charger. Cette fonction permet également de gérer les erreurs (mauvais format de l'image, mauvaise taille, etc.)

Pour chaque primitive, nous avons d'abord appliqué une couleur que nous trouvons convenable. Ceci aura son importance pour l'application des textures.

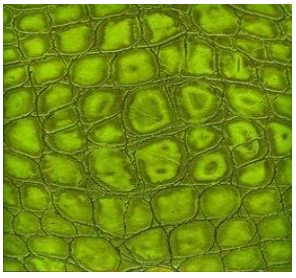
Pour notre tortue, nous avons utilisé au total 3 textures :

Première texture :

La première texture représente la carapace d'une tortue. Nous avons appliqué cette texture sur toute la demi-sphère représentant notre carapace. Cependant, cette texture est trop claire pour notre modèle. Définir une couleur marron foncé sur la demi-sphère avant d'appliquer une texture nous a permis de régler ce problème. En effet, en utilisant le mode GL_MODULATE, le texel et le fragment sont multipliés. Ainsi, notre carapace aura la texture avec une couleur qui sera plus sombre et cohérente avec notre modèle.

Figure 4 *- carapace.jpg*

Deuxième texture



La deuxième texture représente les écailles d'une tortue. Nous avons appliqué cette texture sur les sphères qui représentent la tête et le bas des pattes et sur les cylindres qui représentent le cou et l'autre partie de la patte. Cependant, cette texture est également trop claire pour notre modèle. Définir une couleur vert foncé sur les différentes primitives et utiliser le mode GL_MODULATE nous a également permis d'avoir un effet plus cohérent et de régler ce problème.

Figure 5 *- ecaille.jpg*

Troisième texture

La troisième texture représente les motifs du plastron d'une tortue. Nous avons appliqué cette texture sur chaque face de la demi-sphère qui représente le plastron de la carapace. Pour avoir un fond marron et des tâches plus foncés, nous avons également appliqué une couleur marron foncé sur notre primitive et utiliser le mode GL_MODULATE pour appliquer notre texture.

Figure 6 *- plastron.jpg*

À la fin de cette étape d'habillage, nous avons réussi à avoir le résultat suivant :

Figure 7 *- Le résultat obtenu après l'étape de l'habillage par des textures*

Lumières

Pour éclairer notre tortue, nous avons utilisé deux sources de lumière pour mettre de l'éclairage dans notre scène. Ces deux sources de lumière ont la même position (-2,2,3) et la même direction (0,0,0), là où se situe notre tortue.

Première source de lumière

Nous avons utilisé une lumière ambiente, diffuse et specular (phong). La lumière ambiante a comme paramètre ambient0[] = {0.1,0.1,0.1,1}. C'est-à-dire lorsque que la tortue ne recevra pas de lumière, au lieu de ne plus la voir, celle-ci sera visible avec des couleurs plus foncées que celle par défaut qui sont (0.2,0.2,0.2). La lumière diffuse a comme paramètre diffuse0[] = {0.7,0.7,0.2,1}. Notre tortue recevra donc de la lumière jaunâtre provenant de (-2,2,3) pour avoir un effet similaire au soleil. La lumière specular a comme paramètre specular0[] = {0.5,0.2,0.0,1}. Des reflets marrons seront ainsi visibles sur la tortue. GL_SHININESS a comme paramètre 125 pour avoir des plus petits reflets. Pour activer ou désactiver cette lumière, il faudra appuyer sur le bouton "l".

Figure 8 *- Le résultat obtenu du coté où il y a la lumière*

Figure 9 *- Le résultat obtenu du coté où il n'y a pas la lumière*

Deuxième source de lumière

Nous avons utilisé une lumière specular avec comme paramètre specular1[]={1.0,1.0,1.0,1}. Des reflets blancs seront ainsi visibles sur la tortue ce qui donnera un effet "brillant". GL_SHININESS a comme paramètre 125 pour avoir des plus petits reflets. Pour activer ou désactiver cette lumière, il faudra appuyer sur le bouton “L”.

Figure 10 *- Le résultat obtenu du coté où il y a la lumière*

Animation

L'animation automatique

Notre animation automatique sera sur la queue. Celle-ci bougera de gauche à droite. Pour ceci, nous avons utilisé la fonction idle. Afin d'implémenter cette animation, nous avons créé un booléen rotationQueue initialisé à faux et une variable 'a' initialisée a 0. La variable 'a' représente le degré de rotation de la queue sur l'axe des y. À chaque appel de la fonction idle, a est incrémenté de 1.5 (car rotationQueue est initialisé à faux) ce qui fera pivoter la queue de 1.5 degré (à l'aide de la fonction glRotatef dans la matrice de la queue). Une fois que la queue a une rotation de 45 degrés, roationQueue passe à true et au prochain appel de la fonction, a sera décrémentée de 1.5 jusqu'à que la variable a soit égale à -45 ce qui fera passer rotationQueue à false. Nous aurons ainsi une queue qui pivote entre -45 et 45 degrés sur l'axe des x (après rotation sur y).

L'animation avec le clavier

Pour l'animation avec le clavier, nous avons décidé de faire avancer ou reculer notre tortue. Notre tortue avance de la façon suivante :

- La patte avant gauche est pivotée de 20 degrés
- La patte arrière droite est pivotée de 20 degrés
- On translate la tortue de 0.25 sur l'axe des x et nous pivotons de -20 degrés les deux pattes précédentes pour les avoir droites.
- La patte avant droit est pivotée de 20 degrés
- La patte arrière gauche est pivotée de 20 degrés
- On translate la tortue de 0.25 sur l'axe des x et nous pivotons de -20 degrés les deux pattes précédentes pour les avoir droites.

Pour reculer, le procédé est le même mais nous translatons la tortue de -0.25 sur l'axe des x et nous pivotons cette fois-ci de – 20 degrés les pattes dans l'ordre suivant : patte arrière droite, patte avant gauche, translations de -0.25, patte arrière gauche, patte avant droite, translation de -0.25 sur l'axe des x.

Le bouton "r" permet d'avancer la tortue et le bouton "R" de la reculer.

Zoom

Pour le zoom, nous avons placé la fonction glScalef(zoom, zoom, 1.0f) dans la fonction affichage. La variable zoom est une variable globale initialisée à 1. Puis lorsque nous appuyons sur 'z', zoom s'incrémente ce qui permet de zoomer sur la tortue. 'Z' permet de décrémenter la variable zoom donc de dézoomer.

Rotation

Pour la rotation autour de la tortue, nous avons placé les fonctions glRotatef(angley,1.0,0.0,0.0) et glRotatef(anglex,0.0,1.0,0.0) dans la fonction affichage. Ensuite, nous avons implémenté une fonction clavierSpecial(int key, int x, int y) qui est appelé lorsque nous appuyons sur un bouton « spécial » de notre clavier (échap, flèche, etc.). Ainsi, lorsque nous appuyons sur :

- la flèche de gauche, la caméra tourne autour de la tortue par la droite en décrémentant anglex
- la flèche de droite, la caméra tourne autour de la tortue par la gauche en incrémentant anglex
- la flèche du bas, la caméra tourne autour de la tortue par le haut en incrémentant angley
- la flèche du haut, la caméra tourne autour de la tortue par le bas en décrémentant angley

Conclusion

Dans ce projet, nous avons vu qu'il était possible de modéliser une tortue en trois dimensions simplement avec des formes basiques telles que des sphères, des polygones, des cônes et des cylindres tout en habillant ces formes et en éclairant la tortue de différentes manières. Nous avons également étudié des animations possibles comme faire avancer ou reculer la tortue. Cependant, nous pouvons remarquer que cette tortue n'est pas très réaliste. De plus, la création d'une primitive à partir de sa représentation paramétrique est compliquée et nécessite des formules mathématiques complexes. La modélisation d'une simple tortue avec deux animations et deux lumières exige de coder plus de 1000 lignes de codes. Néanmoins, il existe aujourd'hui d'autres logiciels plus récents qui permettent de faire de la modélisation en trois dimensions avec beaucoup d'outils plus simples et qui automatisent certaines tâches comme l'habillage, les animations et la génération de primitives par exemple.