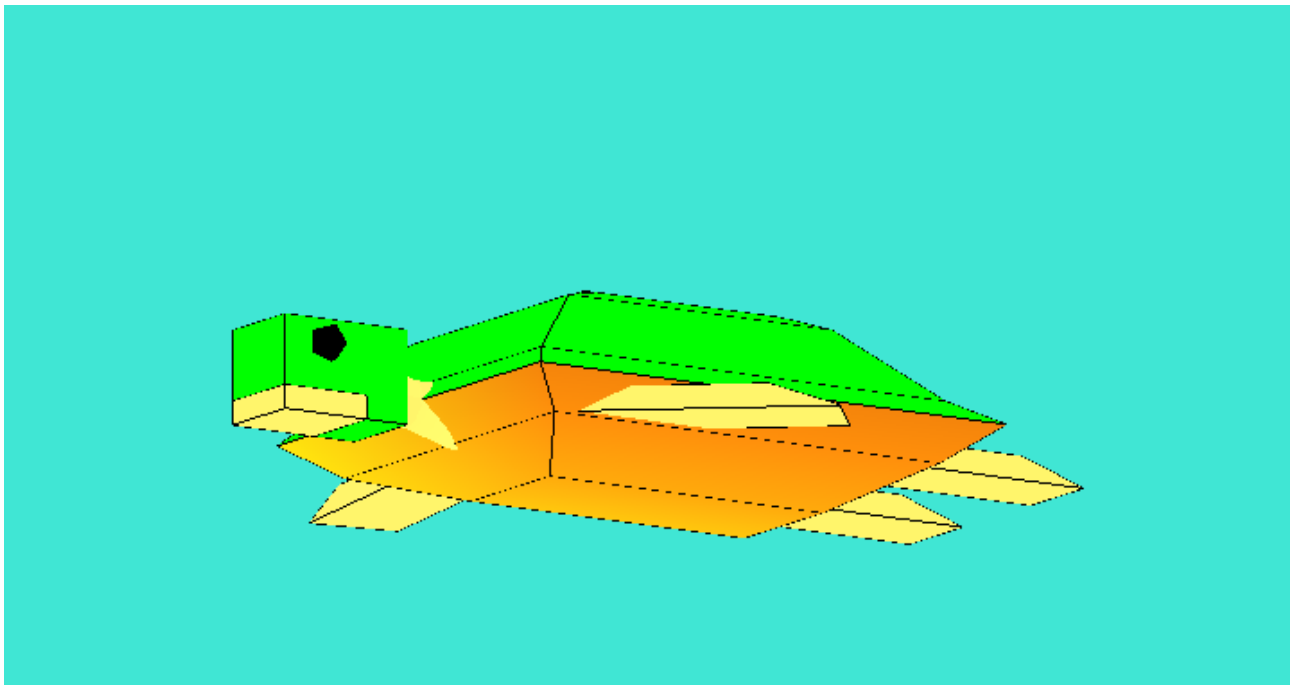
Sid Ali 2022-2023

Projet Synthèse d’image

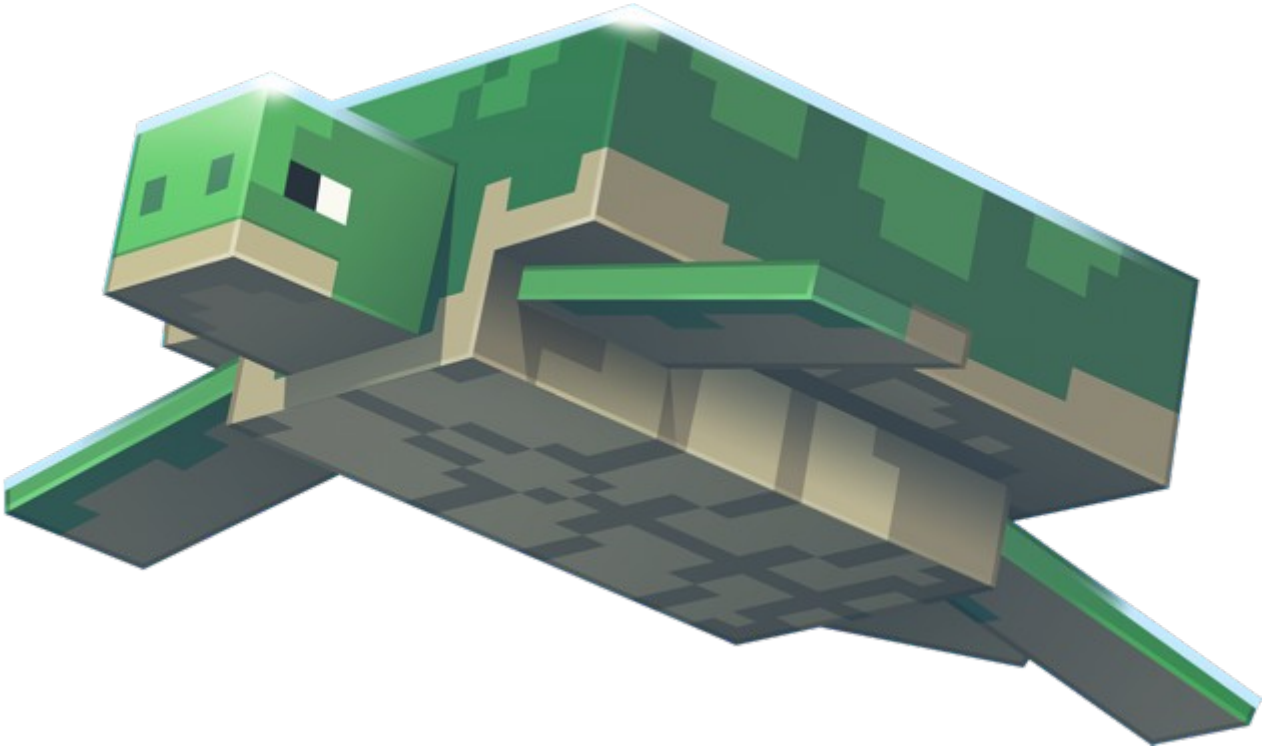
Réalisation d’une tortue en 3D



Sommaire

CHOIX ET IDÉES POUR LA TORTUE

Initialement j’ai décider de représenter la tortue inspirée du jeu-vidéo Minecraft.
La tortue dans Minecraft est une tortue de mer au aspects cubiques tout comme le principe du jeu.



Tortue issue du jeu Minecraft

La tortue ici est composée de deux pattes à l’arrière , deux pattes sur le coté, une énorme carapace qui forme un « T » vu de devant, une tête mais pas de cou.
Je me suis donc inspiré de ce modèle pour mon projet.

Ma tortue sera donc une tortue de mer composée de :

-Une carapace

La carapace dans mon modèle concerne seulement la partie verte qu’on peut voir sur les images si dessus.

-Un « corp »

Le corp ici est la partie inférieur de la carapace, c’est à dire la partie beige.

-Quatre pattes

Quatre pattes , dont deux sur le coté et deux derrière la tortue .

-Un cou

J’ai décidé d’ajouter un cou qui sera composé de deux cylindres

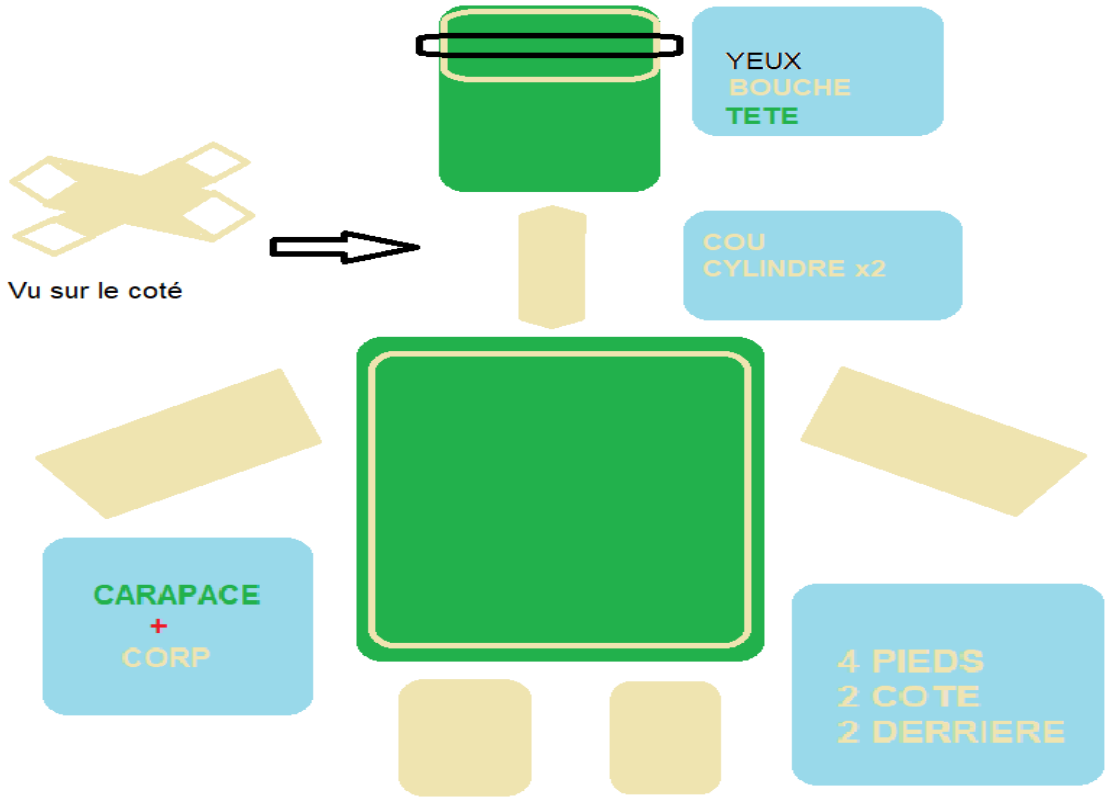
-Une Tête

la tête est composée d’un crâne en vert, d’une bouche en beige et les yeux sont fait à l’aide d’un cylindre qui traverse le crane des deux cotés
(préférable que de créer deux cylindres des deux cotés)

J’ai personnellement décidé d’ajouter un petit cou à la tortue pour lui donner du réalisme malgré que le cou des tortues de mer sont cour et font le même diamètre que leurs têtes, on a donc l’impression qu’elle n’en n’ont pas.
La carapace est divisée en deux parties c’est à dire la partie supérieure que j’appelle « carapace » et la partie supérieure que j’appelle « corp », ce choix est dû à une idée d’animation que j’ai finalement décidé d’abandonner mais j’ai garder les deux parties comme elles le sont.
Chaque arêtes des faces de chaque éléments sont aussi affichées en noir excepté pour les cylindres car une grande subdivision rend le cylindre totalement noir, je trouve que ça donne du contraste à la tortue.

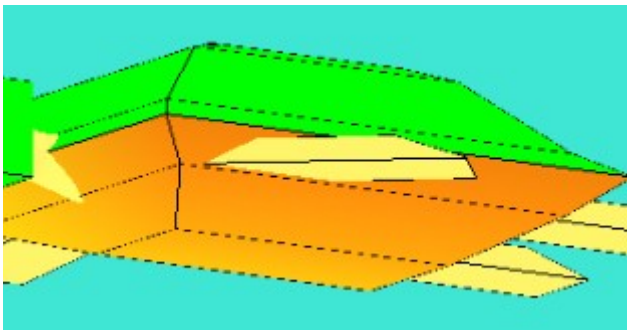
Pour les animations j’ai choisis un mouvement des pattes avant pour l’animation auto et pour l’animation par touche j’ai choisis de faire rentrer la tête et le cou dans la carapace
Il n’y a pas de partie texturée car je n’ai pas réussi à les appliqués.

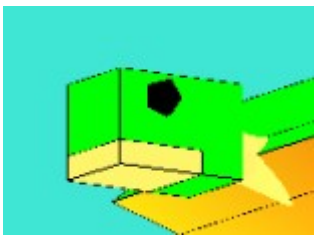
STRUCTURE DE LA TORTUE



Voici un diagramme de ma tortue vu du dessus, les formes non pleines représentent une vu « x-ray »car elles se trouvent en dessous d'autre formes, par exemple la bouche se trouve en dessous des yeux mais du coup aussi en dessous de la tête, du « crane », les yeux se trouvent au dessus de la bouche mais dans la tête, le corp lui est en dessous de la carapace.
Chaque partie sont séparées pour mieux les distingués.

le corps et la carapace se relient pour ne former qu'un, tout comme la bouche et la tête, c'est à dire que je n'ai pas eu besoin de translate pour que les parties se rejoignent.





Pour chaque parties excepté les cylindres qui sont les primitives créées à partir de leurs représentation paramétriques, j'ai remplis les tableaux point par point et ensuite générer à l'aide d'une boucle comme dans le tp1

Le cou est composé de deux cylindres placés comme un X, c'est un choix personnel je trouve que ça rend mieux que de faire un cylindre à gros rayon.

le cou et les pattes sont toutes translates de sorte à se relier au corps de la tortue
la tête est elle translate vers l'autre bout du cou.

DIFFÉRENTES FONCTIONS ET LEURS FONCTIONNEMENT

repere() ;

la fonction repere() ; affiche juste des lignes des axes X,Y, Z.

carapace() ;

la fonction carapace comme beaucoup d'autre vont faire appel à un tableau de points « pCarapace » que j'ai définis un par un et un autre tableau de faces « fCarapace », à l'aide de deux boucles et des deux tableaux les faces et leurs arrêtes seront générés, une boucle pour les faces et une autre pour la carapace.
J'ai ajouté un translate pour recentrer au point (0,0,0) et une couleur verte uni.

corps() ;

la fonction corps faire appel à un tableau de points « pCorp » que j'ai définis un par un et un autre tableau de faces « fCorp », à l'aide de deux boucles et des deux tableaux les faces et leurs arrêtes seront générés, une boucle pour les faces et une autre pour le corp.
J'ai ajouté un translate pour recentrer au point (0,0,0) et une couleur marron uni.

cylindre(float r, float hauteur, int subdivision) ;

la fonction est faite en quatre parties, la première partie qui défini le principe de subdivision, une autre qui va créer 4 faces initiales, plus il y a de subdivision plus on augmente le nombre de face ce qui vas arrondir la surface du cylindre, la troisième partie vas générer toute ces faces et la 4 eme partie vas générer les deux faces aux extrémités du cylindre en haut et en bas.

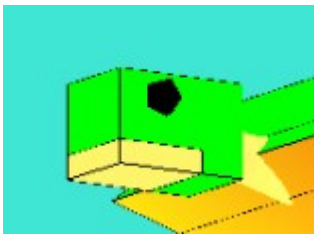
La variable r vas permettre de choisir le rayon du cylindre, la variable hauteur permet de choisir la hauteur du cylindre et la variable subdivision permet de choisir le nombre de subdivisions du cylindre , plus il y a de subdivisions plus le cylindre paraît rond sur les cotés et lisse car le nombre subdivisions est proportionnel au nombre de faces.

cou() ;

la fonction cou possède deux cylindres qui sont générés tout deux par la fonction cylindre, les cylindres sont déplacés au bout du corps par des translate, les translate ont une particularité, à l'axe x il est ajouté la variable animationTouche la variable vas permettre de réaliser l'animation qui se produit lorsqu'on appuis sur une touche et vas donc changer de valeur tout le temps de l'animation.
Il est appliqué un rotate aux deux cylindre pour former le X comme expliqué précédemment et je leurs ai donné une couleur beige uni.

tete() ;

la fonction tete est particulière car dans un premier lieu elle sera générée de la même manière que le corp et la carapace avec des tableaux de point et de faces mais pour ça il en faut deux de chaque, c'est à dire qu'il y a deux tableaux points et deux autre faces.
Il y des tableaux de points « pTete » et de faces « fTete » pour les faces à quatre point et il y a deux autres tableaux « pTeteCote » et « fTeteCote » pour les deux faces à six points.



les deux faces à 6 points sont les faces ou se trouvent les yeux comme on peut le voir sur les images si dessus.
Du coup il y aura 2 boucles pour les faces à 4 points pour afficher chaque face et chaque arête
mais il y a aussi 2 boucles pour les faces à 6 points pour afficher chaque face et chaque arête, soit au total 4 boucles.

La fonction à un translate qui permet de placer la tête au bout du cou, mais tout comme la fonction cou l'axe x du translate à pour ajout animationTouche pour l'animation faite par touche.
Le rotate permet de mettre la tête dans le bon sens car j'ai écris les points dans l'autre sens, et la tête a une couleur vert uni comme la carapace.

La fonction tete appel la fonction() ; bouche et yeux() ; cela vas faciliter le déplacement de la tête tout entière c'est à dire le crane la bouche et les yeux lors de l'animation, et aussi la bouche n'a pas besoin de translate car j'ai réalisé la bouche en fonction des points de base de la tête.

yeux() ;

la fonction yeux appel juste la fonction cylindre , le cylindre va traverser le crane pour ressortir des deux cotés, ainsi c'est plus rapide de placer les deux yeux d'un seul coup plutôt que de mettre deux objets qu'il faudra aligner.
Un translate et utiliser pour le centrer en fonction du crane et le placer la ou yeux devraient être, les yeux resteront noir.

bouche() ;

la fonction bouche à le même procédé que la carapace , deux tableaux, « pBouche » et « fBouche » pour générer les faces et les arêtes de la bouche, les points sont écris en fonction de la tête donc pas besoin de translate vu que la tête appelle la fonction bouche.

pied() ;

la fonction pied vas permettre de generer un seul pied de la tortue de la même manière que la carapace donc 2 tableaux « pPied » et « fPied » qui vont permettre de générer les faces et arêtes du pied avec les 2 boucles.
La fonction pied sera appelée 4 fois dans la fonction allpied() ; afin de tous les générer et les placer à leurs bonne place, les pieds ont une couleur beige uni.

allpied() ;

la fonction vas appeler quatre fois la fonction pied pour generer quatre pieds, soit les deux à l'avant sur le coté et les deux autres à l'arrière, les pieds sont collés au corps.
Chaque pied à un translate qui lui est propre pour les placer au bon endroit,
les deux pieds à l'avant auront un rotate de sorte à ce qu'ils soient en diagonales par rapport au corps, mais en plus de ça il est ajouté une variable animationAuto à leurs axe x de leurs rotate, la variable change de valeur en permanence afin de réaliser l'animation auto.

Les pieds derrières sont collé au corps et suivent le corps dans sa longueur.

oeuf() ;

c'est une fonction abandonnée à la dernière minute car je comptais m'en servir pour y placer une texture mais je n'ai pas réussis à le faire, l'oeuf était un simple cylindre à coté de la tortue.

animeTouche() ;

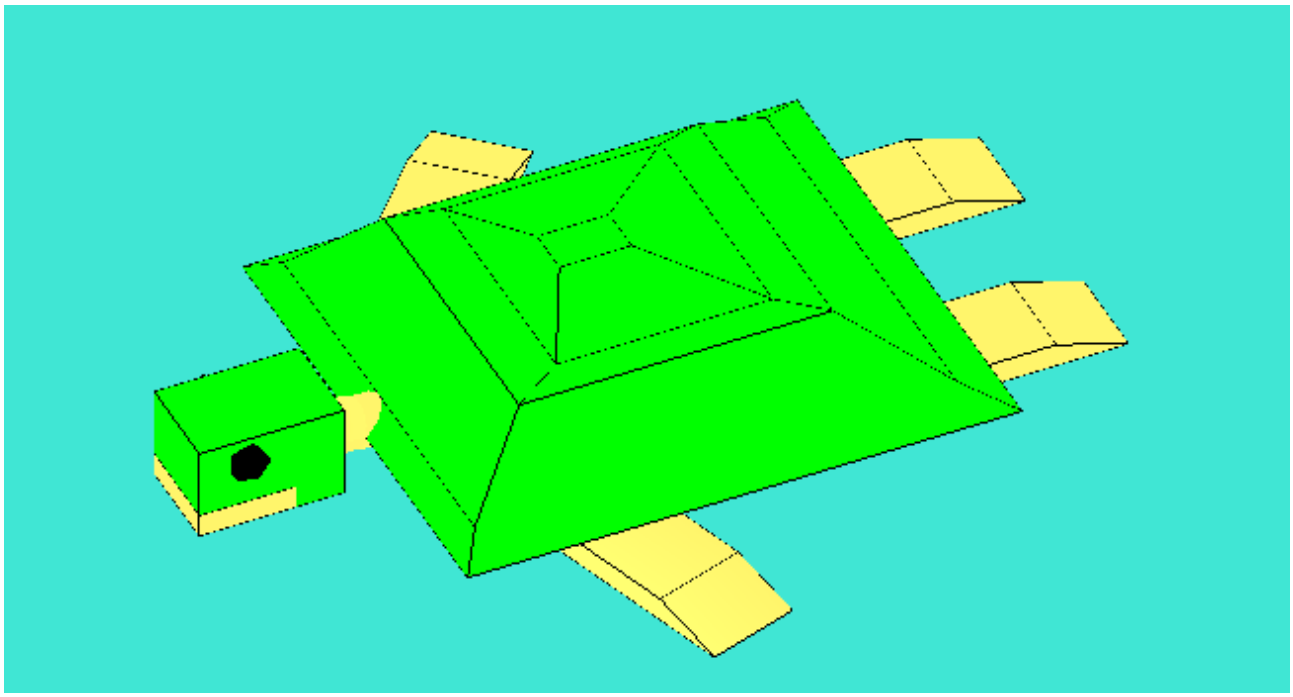
la fonction animeTouche vas permettre de réaliser l'animation qui se lance par touche, la touche est « y » et l'animation vas consister à déplacer la tête(comprends la bouche et les yeux) et le cou afin de les faire rentrer dans la carapace de la tortue, d'où la variable animationTouche dans les translate de ces deux objets.

Un booléen boolAnimTouche vas permettre de savoir si l'animation à déjà été lancée ou pas et sera une condition à deux animations en l'occurrence, lorsque la tortue rentre sa tête mais aussi lorsqu'elle la sort. AnimationTouche est aussi une condition pour s'arrêter à au bon endroit

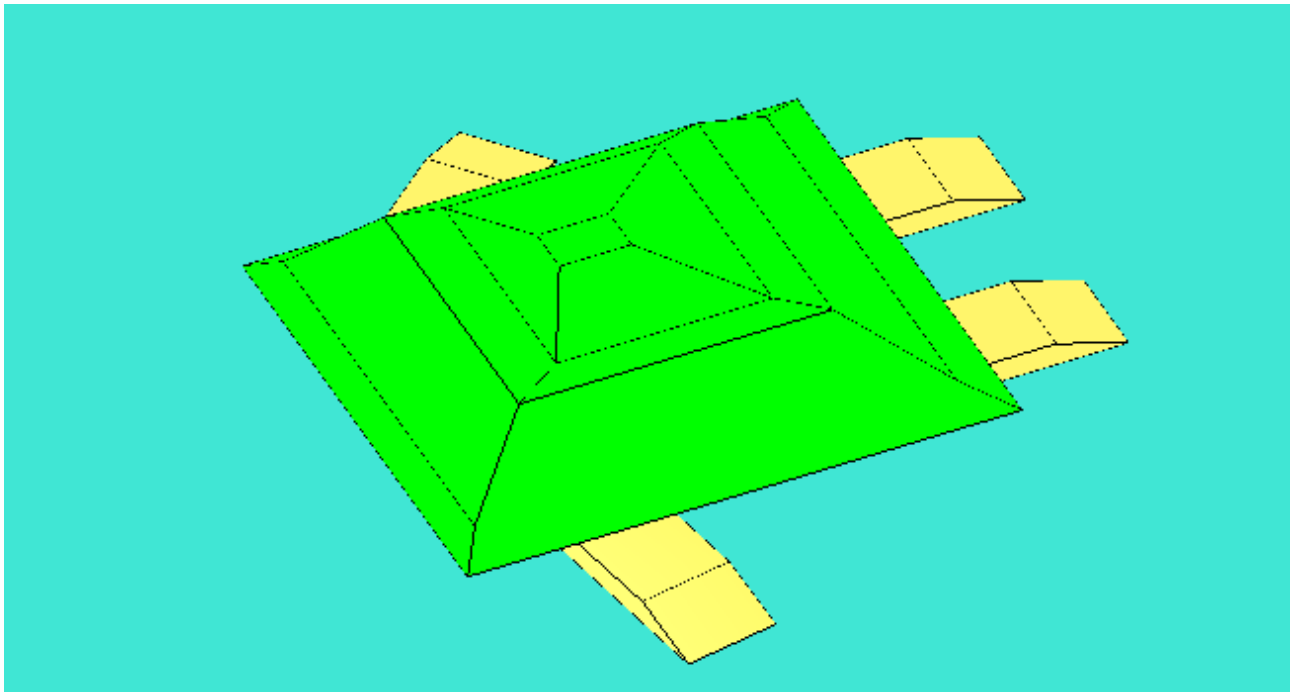
animationTouche est la variable qui change de valeur constamment en fonction d'une autre variable qui gère la vitesse animationToucheSpeed, lorsque la tête rentre animationTouche diminue et l'inverse lorsqu'elle sort.

Afin que l'animation marche sans avoir besoin de rien d'autre elle est appelé dans la fonction suivante car la fonction suivante marche sans arrêt mais aussi rafraîchit les images à chaque nouveau déplacement car la fonction suivante est appelé comme pointeur par la fonction glutIdleFunc.

tête sortie :



tête rentrée :



animationAuto() ;

cette fonction comme dit précédemment est appelé en tant que pointeur par la fonction glutIdleFunc, ce qui permet de rafraîchir les images à chaque mouvement, ce qui est important car l'animation ici commence dès le début et n'as pas de fin.

La fonction à un booléen boolAnimAuto qui change de valeur à la fin d'une des deux animation, car les pattes font des vas et viens. Lorsque le booléen est true la variable animationAuto vas prendre de la valeur en fonction de la variable animationAutoSpeed, une fois que la var animationAuto est supérieur ou égal a la variable Max , alors le booléen prend sa valeur contraire et vas exécuter l'animation inverse c'est à dire le retour du pied, donc animationAuto vas perdre de la valeur ensuite lorsqu'il sera inférieur ou égal à la valeur inférieur de Max alors le booléen prend de nouveau sa valeur inverse et ainsi de suite.

On appel comme dit précédemment la fonction animeTouche comme ça il n'y a pas besoin de glutTimerFunc ou quoi que ce soit d'autre puisque l'animation se lancera seulement si le booléen boolAnimTouche est true ou false suivit de la condition de l'emplacement de la tête.

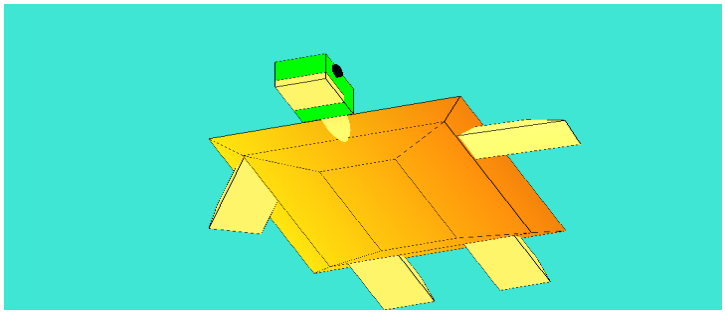
zoom() ;

zoom possède trois variables, camZoom qui va permettre de définir l'état du zoom , la var touche qui est en paramètre qui vas lire la touche sur laquelle on appuis donc soit z soit Z dans la fonction clavier et enfin zoomSpeed qui sert à définir la force du zoom.

camZoom vas être placé partout dans un glOrtho ce qui permettra d'effectuer le zoom lorsque camZoom change de valeur.
Si on appuis sur z camZoom prend la valeur touche (= -1) * zoomSpeed
et inversement avec la touche Z.

lumiere() ;

pour les lumières j'ai utilisé une source, et j'ai initialisé l'ambiant , le diffuse , le specular le shininess et une direction.
Malheureusement ont voir par réellement les effets que ça donne sur ma tortue car elle absorbe la lumière, ceci dit ont peut voir sur les pattes animés qu'elle se blanchissent lorsque qu'elle bougent et la partie pas de ma tortue l'effet de la lumière



clavier() ;

en plus des touches utiles qu'on retrouve dans le tp1 , j'ai ajouté donc la touche z et Z qui dezoom et zoom et rafraîchissent l'image à chaque pression, et ensuite la touche y qui du coup change le booléen boolAnimTouche en lui faisant prendre sa valeur inverse ce qui permet de changer d'animation.

clavierSpecial() ;

même principe que la fonction clavier mais cette fois-ci sert aux touches spéciales tel que les flèches car elle ne peuvent pas êtres utilisés dans la fonction clavier.
On appelle cette fonction en tant que pointeur par la fonction glutSpecialFunc ce qui nous permet d'utiliser les touches spéciales.

Les flèches vont donc servir au déplacement de la caméra en fonction du point d'observation, pour se faire je vais me servir des variables angley et anglex la fonction mousemotion() ; donnée dans le tp1 pour effectuer le déplacement de caméra.

J'initialise une variable moveSpeed qui permet de définir la vitesse de déplacement de la caméra et pour chaque flèche j'effectue addition d'un angle donné à une flèche ou bien une soustraction, ex lorsque j'appuie sur la flèche de gauche ça me déplace sur la droite et pour se faire la touche exécute le changement de variable suivant

anglex = anglex + movesSeed ;

ainsi chaque angle change de valeur en fonction de la touche et est rafraîchit pour montrer l'image sous un autre angle de caméra.

Avis sur le projet

J'ai assez bien aimer réaliser ce projet, la synthèse d'image pousse à la créativité et à toucher à un peu tout pour voir les résultats différents, malgré que certaines choses soient je trouve ou compliqués à faire , ça montre comment sont fait les procédés de la modélisation étape par étapes à la source plutôt que d'avoir un logiciel qui nous facilite la tache ou fait les calculs pour nous.