HATANBAATAR VAN ATARTUGS Erkhembileg

LAVOINE Paul-Louis

Pyramide :

Dans ma partie, j'étais chargé de créer des labyrinthes et de les empiler les uns sur les autres en partant du code du TP6, j'ai réussi à accomplir cette tâche en suivant ces étapes :

J'ai ajouté une dimension au tableau "labyrinthe", qui est devenu un tableau à 4 dimensions. La première dimension représente le labyrinthe d'un étage spécifique, et j'ai également modifié "sides" en conséquence.

J'ai créé un tableau "LAB_SIZE[4]" de quatre éléments entiers qui représentent, dans l'ordre, les dimensions d'un labyrinthe d'un étage spécifique (par exemple, LAB_SIZE[0] représente la dimension du labyrinthe[0]). De plus, j'ai créé une variable globale "étage" de type entier pour garder la trace de l'étage actuel.

Cela m'a permis de reprendre le code fourni en imbriquant les boucles existantes dans une troisième boucle qui itère sur le nombre d'étages (dans mon cas, 4, mais cela peut être modifié) dans ma fonction "setup". J'ai ensuite appliqué l'algorithme du professeur sur l'ensemble de mes labyrinthes.

Évidemment, le code du TP6 utilise la variable "PShape laby0" pour dessiner un seul labyrinthe. J'ai donc créé un tableau de "PShape" pour chaque labyrinthe. Ainsi, dans ma fonction "draw", j'ai ajouté une boucle pour dessiner tous mes "laby0" avec une translation verticale de la hauteur d'un mur (100) multipliée par l'étage, et une translation diagonale dont les composantes du vecteur sont (la largeur du mur multipliée par le nombre d'étages, la longueur du mur multipliée par le nombre d'étages).

Avec toutes ces translations, j'ai dû modifier les positions des caméras en ajoutant le vecteur de translation en fonction de l'étage. Dans ma fonction "keyPressed", j'ai fait en sorte que l'on puisse se téléporter vers l'étage suivant à chaque fois que l'on atteint la fin d'un labyrinthe, et également revenir en arrière, c'est-à-dire redescendre en conservant les positions, les directions et les orientations. Sauf si l'on se trouve au rez-de-chaussée, on peut alors sortir de la pyramide et se promener librement dans le désert, ce qui n'était pas permis à l'époque des pharaons !

Enfin, j'ai créé un désert en utilisant un PShape en mode QUADS et une texture, qui est une image que j'ai trouvée sur internet.

Éléments non réalisés :

La première difficulté que j'ai rencontrée était de ne pas avoir remarqué à quel point la fonction de translation facilite le travail. Cela m'a fait perdre beaucoup de temps précieux ! Malheureusement, je n'ai pas pu accomplir les tâches suivantes :

La boussole

Les escaliers

Un joli soleil tropical

Momie :

Nous allons présenter la partie Momie du projet, cela consistait à créer une momie en 3D réalisée en utilisant Processing. Le projet utilise des objets PShape pour représenter les différentes parties de la momie, comme la tête, le corps, les bras et les mains. Le code permet de générer et d'animer la momie en simulant un mouvement de rotation autour de l'axe X.

Le projet utilise des objets PShape pour représenter les différentes parties de la momie, et une texture est appliquée pour donner l'illusion de bandelettes enroulées autour de la momie.

Le projet est basé sur l'utilisation d'objets PShape pour créer la momie. La texture "bandage.jpg" est utilisée pour donner un aspect réaliste aux bandelettes des mains, le reste du corps est coloré avec noise. Les parties de la momie sont créées à l'aide de boucles for et de calculs mathématiques pour déterminer les coordonnées 3D des sommets des formes géométriques utilisées. Les parties sont ensuite ajoutées à un objet PShape "momie" pour former l'ensemble de la momie.

Le code est divisé en plusieurs parties principales :

Initialisation des variables et des objets PShape : création des différentes parties de la momie en utilisant des objets PShape, et chargement de la texture "bandage.jpg" pour la momie.

Fonction setup() : initialisation des objets PShape, création de chaque partie de la momie en détail, et ajout de chaque partie à l'objet PShape "momie".

Fonction draw() : animation et affichage de la momie en utilisant des transformations (translation et rotation) et la fonction shape() pour dessiner la momie et ses parties à l'écran.

Résultats et démonstration

Le code permet de générer une momie en 3D et de l'animer en simulant un mouvement de rotation autour de l'axe X. La démonstration montre que la momie est correctement créée et animée, avec des bandelettes réalistes et une apparence globale convaincante.

Conclusion

Nous avons réalisé un projet de création et d'animation d'une momie en 3D avec Processing. Le code permet de générer une momie en 3D avec des bandelettes réalistes et d'animer la momie en simulant un mouvement de rotation. Des améliorations pourraient être apportées, notamment en ajoutant plus de détails et de réalisme à la momie, ou en intégrant ce projet dans des applications plus larges, telles que des jeux, des animations, ou des projets artistiques.