Université Paris Scalay

Faculté des Sciences

PROJET IGSD

momie & pyramide

Présenté par :

Wang Marc — Pourchot Alistair

Groupe : LDDIM2

Introduction

Le but de ce projet est de réaliser deux modèles 3D : une pyramide ainsi qu'une momie dans une scène. Ce projet a été réalisé en l'espace de deux mois en codant sur Processing. Nous retrouverons dans ce rapport les différentes étapes de réalisation du projet, des images du rendu final ainsi que les différentes difficultées rencontrées.

Contents

1 L'intérieur

1.1 Le labyrinthe

Ce projet commence à partir d'un simple labyrinthe 2d, créé aléatoirement suivant un algorithme de backtracking (fourni) nous donnant un labyrinthe avec une entrée à la position (1,0) et une sortie à la position (n, n-1) où n est la taille du labyrinthe. On représente dans cette première modélisation un mur par un # (voir figure 1) et c'est à partir de cette forme que l'on va construire notre labyrinthe en 3d. Simplement, on va parcourir le labyrinthe 2d et mettre un bloc 3d dans notre scène à chaque fois que l'on tombe sur un #.

```
# ###################
#    #    #      #
# # ### # ####### # #
#   #   #        # #
# ######### #########
# # #    #       # #
# # # # # # ####### #
# # # # # #         #
# # ### ### ##### ###
#    # #   # #   # #
# ##### # ### ### # #
#   #    #         #
# ##### ### ##### # #
#      #   # # # # #
### ######### # ### #
#        # #       #
### ##### # # ##### #
# # # #    #     # #
# ### ##### ##### # #
#          #    #
#####################
```

Figure 1: Labyrinthe 2d

Après avoir obtenu notre labyrinthe en 3d on y rajoute un sol et un plafond, et voilà, notre labyrinthe est prêt (figure 2).

Figure 2: Labyrinthe 3d

A présent, il nous reste plus qu'à rajouter un peu de texture et de couleur pour rendre le labyrinthe dans le thème et plus joli (figure 3).

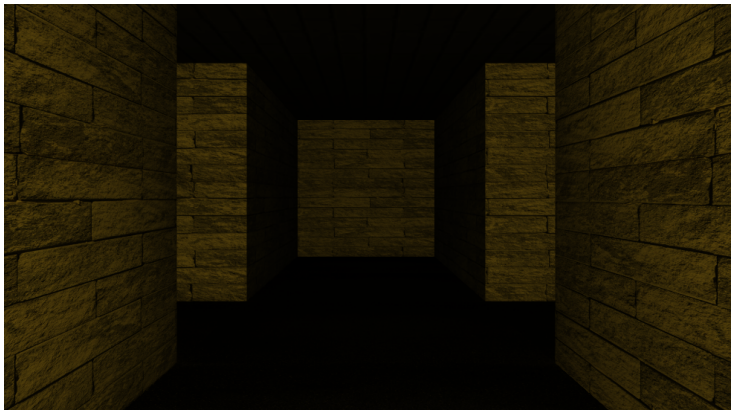

Figure 3: Labyrinthe avec textures

Ah, il nous manque un dernier détail. Pour rendre la circulation plus facile dans ce labyrinthe, on rajoute une mini carte en haut à gauche de l'écran, dans laquelle on peut voir la position du joueur, la direction dans laquelle il regarde, ainsi que les endroits déjà explorés. (figure 4)

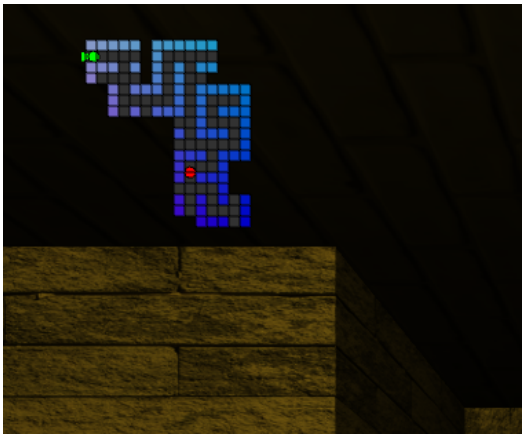

Figure 4: Map et posJoueur

Enfin, une dernière touche pour notre labyrinthe : on rajoute une boussole.

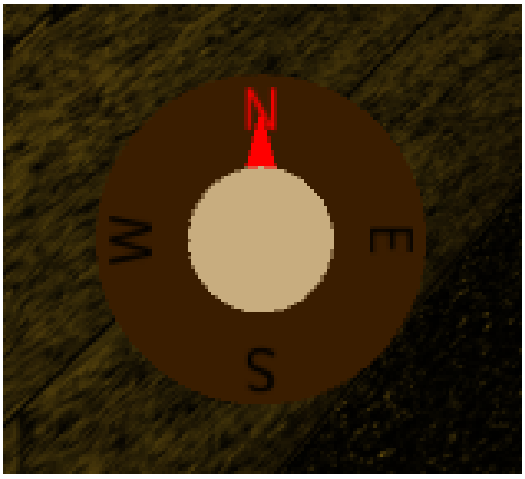

Figure 5: Boussole

Voilà tout ce dont on a besoin pour notre labyrinthe.

1.2 Les étages

Pour faire notre pyramide, on a dû empiler plusieurs de ses labyrinthes. Alors notre fonction de base qui était dans notre setup et qui nous préparait un labyrinthe ne fonctionnait plus vraiment, puisqu'elle marchait à l'aide de variables globales que l'on instanciait. On a alors mis en place une classe LabyrintheRes qui contient le labyrinthe 2d, la taille, et aussi des variables cases et sides qui sont utilisées lors de la création du labyrinthe mais qui ne vont pas nous interesser ici. On peut alors maintenant utiliser la fonction fournie de base pour nous créer un labyrinthe de type LabyrintheRes, ce qui nous permet maintenant d'avoir plusieurs labyrinthes de différentes tailles. On finit par les empiler et ça y est, le tour est joué.

Cependant, il faut pouvoir passer d'un labyrinthe à un autre lorsque l'on arrive à la fin du premier. On a alors modélisé un escalier.

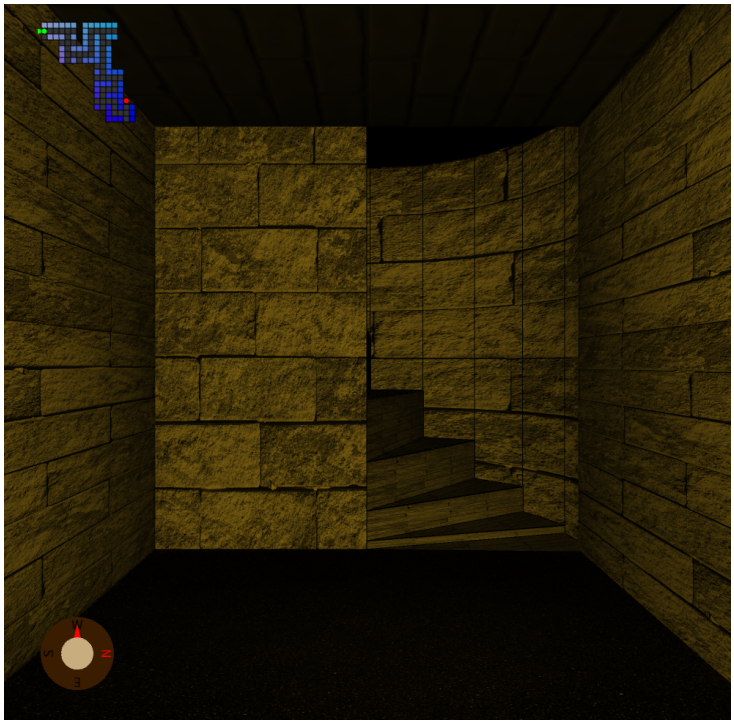

Figure 6: Escalier

L'escalier est situé à la sortie et il nous suffit d'appuyer sur la touche 'e' du clavier une fois que l'on se trouve devant pour le prendre et se rendre à l'entrée de l'étage suivant.

On a décidé de manière arbitraire (car en réalité on pourrait en mettre autant que l'on veut) d'en mettre seulement trois, et une fois que le troisième est terminé, on a fini et on obtient le trésor de la pyramide.

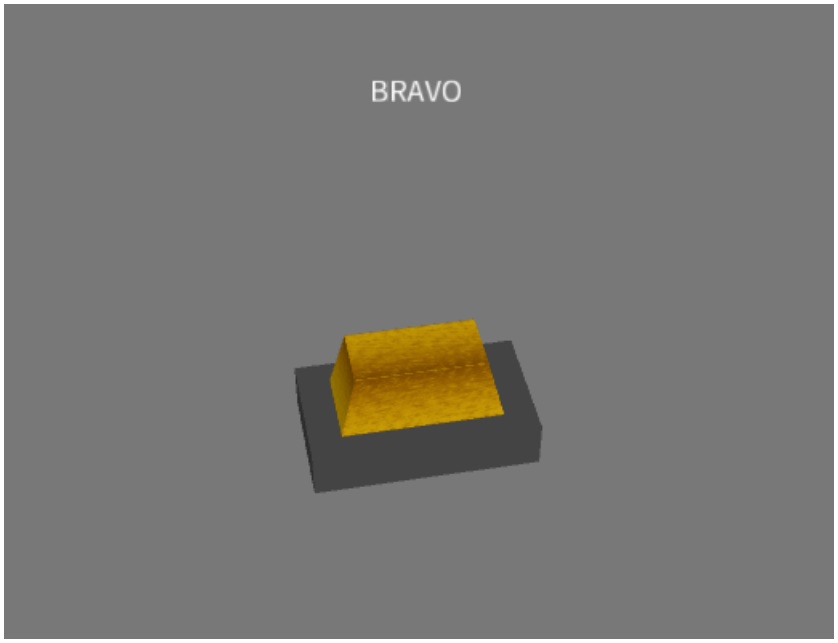

Figure 7: Trésor

Félicitations ! Vous venez de terminer le labyrinthe.

2 La momie

2.1 Design

Vous l'aurez sans doute remarqué, on observe un point rouge en plus du point vert du joueur sur la mini map en haut à gauche (figure 4, 6) Ce point rouge correspond à la position de notre momie. Mais tout d'abord, voyons à quoi elle ressemble.

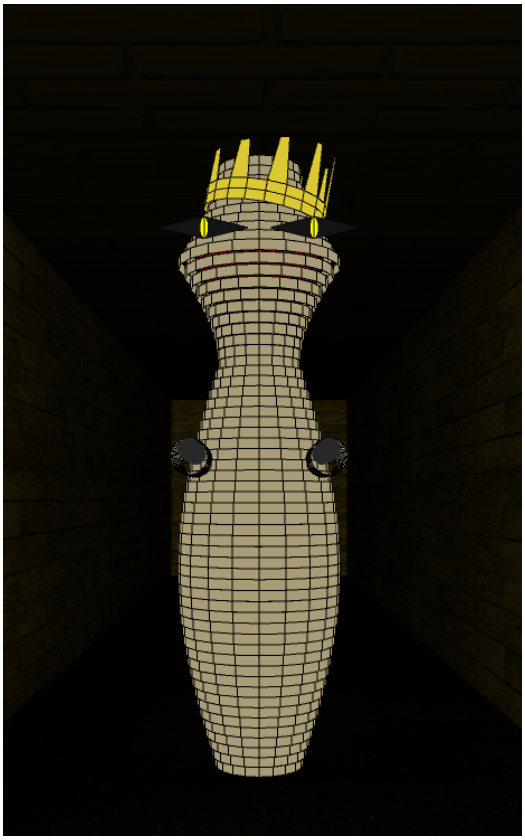

Figure 8: Momie

La voilà. Nous avons opté pour un design assez imposant, avec des yeux plutôt effrayants. Si on se rapproche assez d'elle, on peut même voir son cerveau à travers quelques bandelettes.

2.2 Déplacement

Notre momie se déplace aléatoirement dans le labyrinthe. Pour faire ça, on a mis en place un algorithme qui permet de : tout d'abord, regarder chaque directions possibles (donc telle que la position de la momie + un pas vers la direction ne soit pas un mur). Une fois que l'on a une liste des directions possibles, on en prends une au hasard à l'aide de random. Pour éviter que la momie ne fasse demi-tour trop souvent, on a fait en sorte que si la direction que l'on rajoute dans la liste correspond au demi tour, on ne l'ajoute qu'une seule fois, sinon on la rajoute 20 fois. Cela évite à notre momie d'avoir une chance sur deux de faire demi-tour en plein milieu d'un couloir, par exemple.

3 L'extérieur

3.1 La pyramide

Maintenant que l'on a vu le contenu de notre labyrinthe, il ne nous reste plus qu'à voir l'extérieur.

Pour l'affichage de la pyramide à l'exterieur, c'est simple : on a empilé nos labyrinthes en les décalant d'un bloc et en réduisant leur taille de deux afin de pouvoir avoir un joli escalier de labyrinthes dehors. Tout en haut on retrouve une partie lisse de la pyramide, correspondant donc au toit.

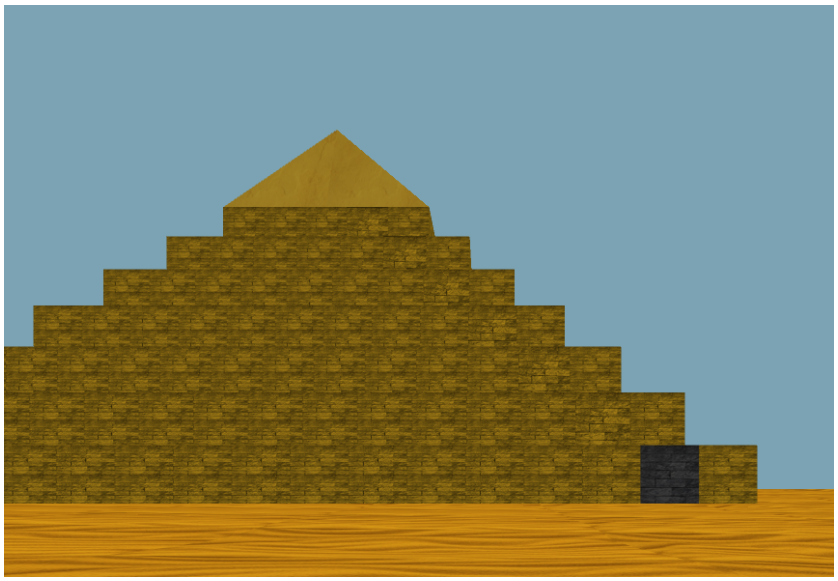

Figure 9: Pyramide

Pour rendre la scène plus jolie, on a ensuite rajouté une lune et plusieurs éclairages, le rendu final étant celui-ci :

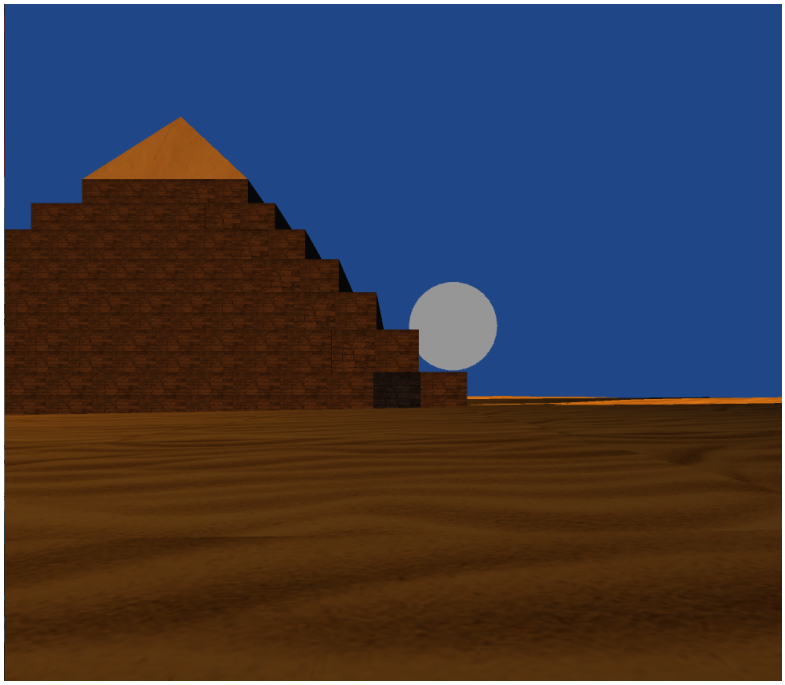

Figure 10: Rendu final

3.2 Le sol

Le sol a été une difficulté à faire. Etant donné que nous avons voulu reproduire un sol de désert, on ne pouvait pas se contenter de mettre un rectangle plat en dessous de nous et de rajouter une texture de sable dessus. Pour faire notre sable, on a du initialiser une double liste de flottants appelée desert comprenant des valeurs obtenues à l'aide de la fonction noise() puis en les répartissant de -20 à 20 à l'aide de map. En effet, puisque noise() renvoie des valeurs entre 0 et 1 et que l'on utilise cette fonction dans le but de créer un effet de légères 'dunes' de sable, il fallait un décalage plus grand que 1. Donc voilà. On a notre tableau, et lorsque l'on va faire nos Shapes pour notre sol, et bien la shape à l'indice (i, j) sera surélevée de la valeur obtenue à desert[i][j]. Ce n'est pas évident à expliquer, mais voici le résultat.

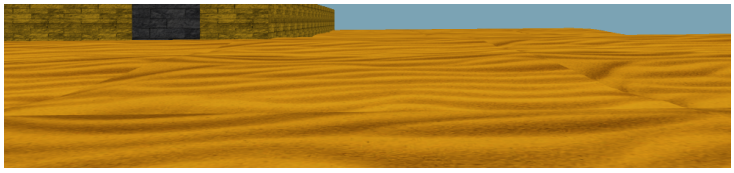

Figure 11: Sable

On obtiens de très légères déformations, mais c'est toujours plus joli que si le sol était plat. De même, on a appliqué une texture de sable pour rendre l'effet encore plus convainquant.

4 Structure du code

Afin de pouvoir mieux gérer notre projet, on a structuré notre code en plusieurs fichiers. Tout d'abord, le fichier principal dans lequel nous déclarons les éléments principaux au projet, c'est à dire la position du joueur, de la momie, les variables d'animation etc. C'est également dans ce fichier qu'on retrouvera le draw, dans lequel on gère les différents cas (être à l'intérieur où à l'exterieur par exemple). On a un fichier supplémentaire pour le labyrinthe, contenant donc notre classe labyrinthe ainsi que le code permettant de créer et afficher notre labyrinthe. On a un fichier pour notre momie, un pour la boussole et un pour l'affichage de fin, et enfin un fichier pour le déplacement, contenant les keyPressed() et keyReleased().

5 Conclusion

Au final, ce projet a été très intéressant à réaliser car il nous pousse à réflechir d'une manière totalement différente de celle avec laquelle on a l'habitude dans un projet informatique. Il fallait beaucoup réflechir à l'ordre dans lequel les choses doivent s'afficher, il faut réflechir à chaque conditions pour les déplacements, les différentes manières de faire les animations, etc.