Rapport
de Programmation Objet Avancée

Yuru HE et Yuetian XU

Mai 2019

Enseignant : M. Patrick Amar

1. Introduction

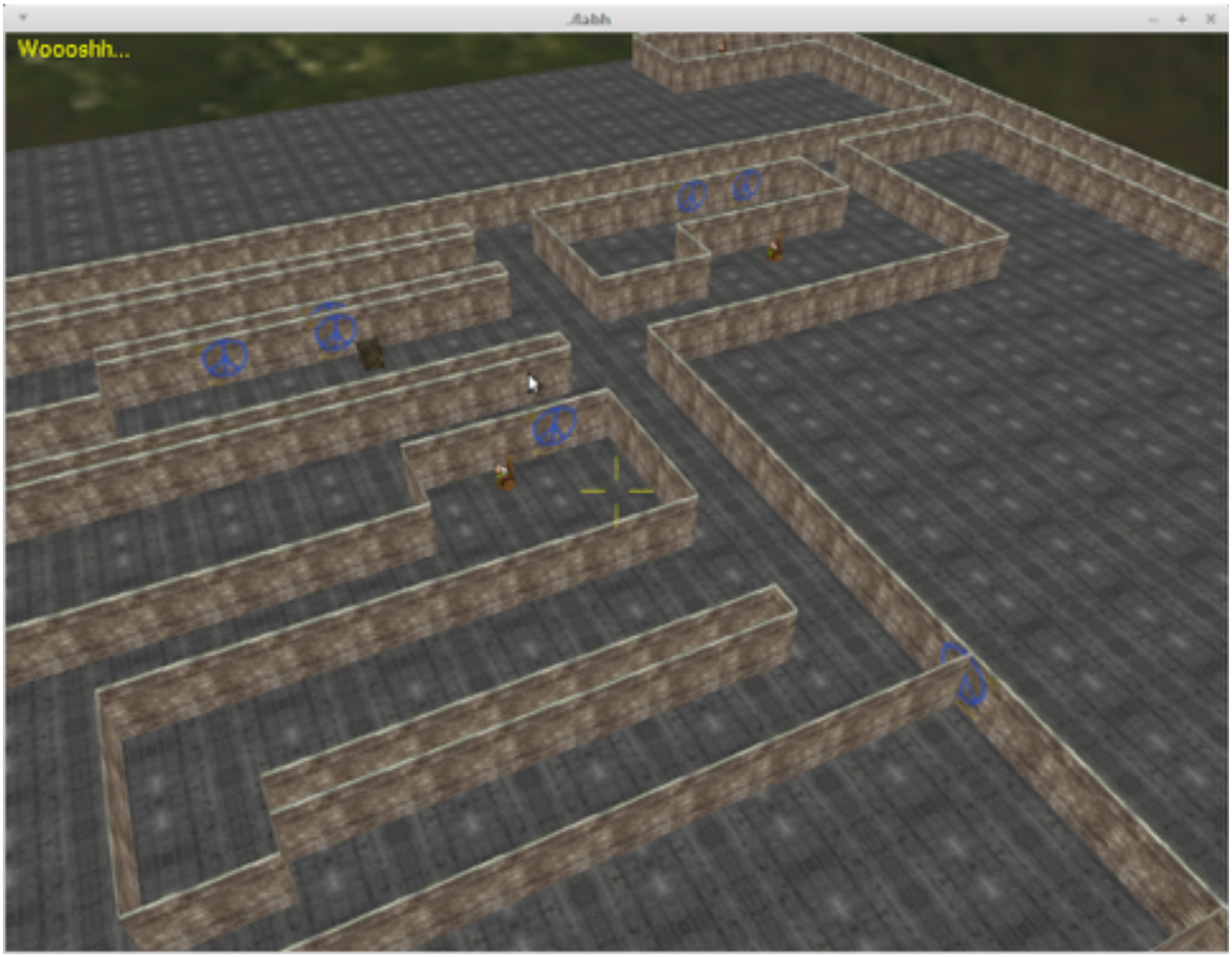
Notre projet est un jeu s’appel Labyrinthe Hanté de type Kill'emAll avec un affichage en 3D.
Notre but de ce jeu est de diriger le chasseur dans la labyrinthe pour rechercher le trésor en évitant et attaquant des gardiens.
Il y deux différentes rôles dans ce jeu :

- Chasseur (joueur) : Pour obtenir le trésor.
- Gardien (machine) : Il faut protéger le trésor caché dans le labyrinthe.

2. Labyrinthe

Le constructeur de la classe Labyrinthe servira à initialiser un labyrinthe selon la représentation interne du programme de jeu à partir d'une représentation textuelle du labyrinthe (à l'aide des caractères + - |).

Le labyrinthe est en 2D, c'est l'affichage graphique qui va le rendre en 3D, vos personnages évoluent dans un plan où les murs des pièces (ou des couloirs qui ne sont que des pièces particulières) sont des lignes droites horizontales ou verticales. Le labyrinthe est fermé, et possède deux pièces particulières : celle où le chasseur est placé initialement, celle où se trouve le trésor.



Le but est de construire un labyrinthe basé sur le fichier "Labyrinthe", et initialiser les éléments dans la classe de labyrinthe. Ensuite, il définit un algorithme pour calculer les distances. Les caractères contient ’+’ : intersection des murs, ’-’ : un morceau de mur horizontal, ’|’ : un morceau de mur vertical, ’C’ : chasseur, ’G’ : gardien, ’X’ : caisse, ’a’ et ’b’ : les morceau de murs ayant la texture et ’espace’: que le chasseur et les gardiens peuvent traverser.

Après avoir créé la carte du labyrinthe, nous devons utiliser BFS pour calcule les distance.

3. Action

La machine anime des personnages de type gardien dont le but est de protéger un trésor caché dans une pièce du labyrinthe, que vous, le chasseur, essayez de prendre.

Les gardiens peuvent attaquer le chasseur à distance en tirant dessus avec une arme qui tire en ligne droite, et dont le projectile est arreté par un obstacle quelconque (mur, objet, etc.), la cible du gardien doit donc être visible pour que le tir ait une chance d'être efficace.

Réciproquement, le chasseur peut attaquer les gardiens. Le coefficient n'est pas le même que pour un gardien car le tir du chasseur est dirigé par un humain.

Appeler cette fonction avec 'true' si le joueur a gagne et 'false' s'il a perdu. Dans les deux cas cela invalidera la possibilite de tir pour le chasseur et cela affichera le message 'You Win!' ou 'You Lose!!' en gros sur la fenetre.

Le but de ce fichier est de construire les gardiens, et definir 3 modes de gardiens et definir les actions sur les gardiens. Il y a 3 modes des les gardiens, “garde”, “défense” et “attaque”.

Quand on est à le mode “garde”, les gardiens se promènent dans la labyrinthe. Lorsque le gardien est à plus de 5 et moins de 50 de distance à le trésor, le gardien devient le mode de défense, il choisit un route qui est plus proche à le trésor, puis choisit aléatoirement une vitesse pour approcher le trésor. Il faut protéger le trésor. Lorsque le gardien constate qu'un chasseur approche, il choisira au hasard un chemin à partir du chemin le plus proche du chasseur, puis choisissez au hasard une vitesse pour aller près du chasseur. Lorsque la distance est inférieure à une certaine valeur, le garde tirera et passera en mode attaque.

4. Conclusion

Quand nous avons fait ce projet, nous avons rencontré quelques challenges et difficultés, parce que c'est la première fois que nous avons complètement fait un jeu, le langage C ++ est aussi plus difficile et nous en savons beaucoup sur les détails. Après cet projet, on apprend à écrire un programme à partir de spécifications informelles, nous avons examiné les connaissances théoriques et nous en avons discuté avec nos camarades, nous avons finalement terminé le projet.