Rapport of Projet de Programmation Objet Avancée

Xinneng XU et Wei HUANG

May 5, 2019

Abstract

Le but de ce projet est pour réaliser un jeu de type *Kill'emAll* avec un affichage en 3D, pendant cette durée, nous allons apprendre à écrire un programme à partir de spécifications informelles (énoncé en français), familiariser avec le fait que vos programmes seront réutilisés par d'autres plus tard; donc à les écrire lisiblement en les commentant judicieusement, entrainer à encapsuler vos données et méthodes pour obtenir des programmes robustes et montrer l'intérêt de séparer la partie traitement d'un programme de sa partie interface utilisateur.

Keywords: A* Pathfinding; Kill'emAll

1 Introduction

C'est un jeu de type *Kill'emAll* et il y a deux rôles dan ce jeu : *Gardien* et *Chasseur*. le but pour *Chasseur(nous)* est de essayer prendre un trésor caché dans une pièce du labyrinthe, et pour *Gardien(la machine)* le but est de le protéger. Le chasseur et les gardiens peuvent tirer le pistolet et ils vont perdre le capital de survie une fois ils sont touché par le pistolet de l'autre. Pour les gardiens, nous avons implémenter l'algorithe de *A* Pathfinding* pour que les gardiens pouvent touver le plus court chemin quand il voulent aller à une endroit. Nous allon présenter ce jeu en 3 parties : *Labyrinthe*, *Gardien* et *Chasseur*.

2 Labyrinthe

La classe *Labyrinthe* est responsable de la construction de la carte et de l'instanciation des rôles.

Nous avons crée une classe *Map* qui responsable de lire le ficher de labyrinthe, et le transformer à une matrix de caractères associé à chaque position. 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 chaseur et

les gardiens pouvent traverser. Dans la classe *Map*, nous avons lu le ficher de labyrinthe deux fois. La prémière fois, nous avons lu des caractères un par un et calculé le longeur(lab_width) et largeur(lab_height) de labyrinthe, et après nous avons créé une tableau par rapport à le longeur et largeur. Pour la deuxième fois, nous vons rempli le tableau que nous avons créé par les caractères.

Dans la fonction de Constructeur de *Labyrinthe* qui prend une argement 'char* filename', nous pouvons obtenir le tableau en utilisant la classe *Map*, et ensuite nous utilisons 3 *std::vector* pour garder les pointeur de *Mover(Gardien, Chasseur)*, les *Box* et les *Wall* en appliquant la fonction *push_back*. Quand nous parcourons le tableau, si le caractère n'est pas espace, nous mettons le valeur de *_data(i,j)=1* dans la cas (i,j) pour que les objets ne pouvent pas marcher ici. Après le parcours, nous assignons les valeus aux variables *(_guards, _nguards, _walls etc)* par raport à les vectors.

3 Garidien

Dans ce partie nous allons présenter les ce aue les gardiens vont jouer.

3.1 Etat

Il y a 3 etats pour les gardiens : Normale, Danger, Dead .

Normale ou Défense. Le gradiden calcule son potentiel de protection (PP) 3.2, le gardien est dans ce etat si PP est supérieur à le seuil 3.2. Dans ce mode, le gardien a *une zone de gestion* d'un cercle de rayon 10*Environement::scale. Le gardien partrouille(en appelant la fonction Move) dans ce zone. Il a aussi *une distance visuelle* de 20*Environement::scale. Le gardien a *une gamme d'attaque*3.3. Une fois le chasseur entre à la zone, le gardien va attaquer3.4 le chasseur. Si le chasseur sort de la zone, le gardien va continuer à patrouiller dans la direction de la dernière patrouille.

Danger. Le gradiden calcule son potentiel de protection (PP), le gardien va passer a l'etat *Danger* si PP est inférieur à le seuil. Dans cet etat, si le chasseur est dans le gamme d'attaque3.3, le gardien va attaquer3.4, sinon, le gardien va calculer la direction de marcher du plus court chemin en utilisant **A* Pathfinding** pour trouver le chasseur pour protéger le trésor. Dans cet etat, le gardien marche 2 fois plus vite que dans l'etat Normal, et la distance de visuelle va 2 fois augementer qui peut augementer le gamme d'attaque 3.3.

Dead. Si le capital de survie est 0, le gardien va être Dead, il ne peut rien faire.

3.2 Calcule Potentiel de Protection

Le potentiel de protection est lié à 3 facteurs :

1. Le percentage de gardien vivant : P.
2. La distance entre le chasseur et le trésor : DCT.
3. La distance entre le gardien et le trésor : DGT.

Nous calculons la distance maximal entre le chasseur et le trésor : DCT_{max} et la distance entre le gardien et le trésor : DGT_{max}. Pour calculer Potentiel de Protection(PP), nous appliquons le formule

$$PP = DCT_{max} - DCT + DGT + P * DCT_{max}$$

. Et nous mettons **le seuil** = $(2 * DCT_{max} + DGT_{max})/2$ pour que quand les gardiens qui sont plus loin de le trésor que le chasseur, pouvent chercher le chasseur et essayer de le arrêter de trouver le trésor, les autres qui sont plus proche de trésor que le chasseur, restent à l'etat Normal et partrouiller.

3.3 Gamme d'Attaque

Il y a 3 facteur pour la gamme d'attaque.

1. Le chasseur faut être dans la fourchette de 180 en face du champ de vision de gardien (Le gardien ne peut pas tourner le dos au chasseur).

2. Il ne peut y avoir aucun obstacle sur le chemin de la garde et le chasseur.

3.La distance entre le chasseur et le gardien doit inférieur à la distance visuelle de gardien.

Donc, si le gardien est dans la gemme d'attaque de gardien, le gardien va se tourner vers le chasseur et attaquer le chasseur dans 1 seconde.

3.4 Attaque

Nous avons ecrit une fonction *attaque*, et nous avons mit une interval de attaque. Chaque fois le programme appelle ce fonction, si l'intervalle de la dernière attaque est inférieur à l'interval de attaque, le gardien fait rien, sinon le gardien va tirer le pistolet vers le chasseur.(Le chasseur peut aussi attaquer les gardiens et les gardiens vont subir de belssure3.6 si ils sont touché par une balle tiré par le chasseur.)

Probabilité de Manquer Cible. Le gardien a le capital de survie(currentBlood) 3.5, initialisé à 100. Si currentBlood est supérieur à 50, le gardien a un probabilité de 10% de manquer sa cible, et si currentBlood est inférieur à 50, ce probabilité va augementer à 60%.

Puissance du pistolet. Le puissance du pistolet maximal que nous avons mit est 40(Pour chasseur est 50). La puissance du pistolet diminue à mesure que la distance augement. *La distance maximal* du vol de balle d'un pistolet est 200*Environment::scale, ça veut dire le chasseur ne sera pas blessé si il est à *La distance maximal* de le gardien..

3.5 Rétablissement de Capital de Survie

Temps Intervalle Chaque gardien a le capital initilisé de survie de 100 (capital maximal de survie), une fois il est touché par une balle, le gardien va diminuer un certain nombre du capital par rappot à la puissance de pitolet et la distance de le chasseur tire. Le programme calcule l'intervalle du temps a partir du moment de la dernière blessure, si le gardien reste 8 secondes sans subir de blessure3.6,

ce capital va augementer par 5% du capital maximal(pour Chasseur est 2%). Ce intervalle peut être réinitialisé par blessure.

Réserves de Santé sous forme de Caisses de Survie Si le gardien tire le pistolet et le balle touche une caisse, son capital de survie va augementer à la maximun, et le caisse va supprimé dans le memoire.

3.6 Subir de Blessure

Nous avons ecrit une fonction (hited) pour gérer la réduction du capital de survie. Il prend 3 arguments : 'int power' : la puissance maximal du pistolet, 'float dist_max' : La distance la plus éloignée qu'un balle peut voler, et 'dist' : la distance parcourue par la balle. Le gardien a **l'armure** = 15. Quand le gardien est touché par une balle, cette fonction va être appelé et le capital de survie va diminué par

$$power * (dist/dist\_max) - armure$$

3.7 A* Pathfinding

Nous avons ecrit une fonction *find_direction* qui prend 2 argument: point de début et fin, et renvoie la direction(angle). Cette fonction utilise algorithme de *[A* Pathfinding]* pour trouver le plus court chemin du début à la fin. Et dans cette fonction, après trouver le chemin, Nous calculons le point le plus éloigné(PE) pour que il y a aucun obstacle(le mur ou caisse) sur le chemin de début à PE, et ensuite calculons la direction(angle) de début à PE et le renvoyons.

Afin d'équilibrer le coût du calcul et de s'adapter au mouvement du chasseur, Nous appelons cette fonction chaque seconde.

4 Chasseur

Le chasseur se déplace dans le labyrinthe dans une direction qui est indiquée par la souris. Et il tire le pistolet par *left_click* de la souris. Il a la même manière que les gardiens sur **Puissance du pistolet** 3.4 et **Rétablissement de Capital de Survie** 3.5.Il y a 2 modes pour le chasseur: *Normal* et *Runaway*.

Etat Normal. Normalement le chasseur est dans ce etat, le rétablissement de capital de survie par seconde est 2%, et la puissance de pistolet est 50 et l'armure est 10.

Si le jouer tape **shift** + **right_click**, le chasseur va passer à l'etat *Runaway*.

Etat Runaway. Le chqsseur dans cet etat va rester 5 seconde, pendant cette durée, le rétablissement de capital de survie va 500% augementer, la puissance de pistolet va 50% augementer et l'armure va 50% augementer.

Après 5 secondes, le chasseur va passer à l'etat Normal et il faut rester 30 secondes pour réutiliser **shift** + **right_click** pour passer à l'etat *Runaway* (30 secondes de refroidissement).

5 Conclusion

6 Reference

A* Pathfinding [https://www.gamedev.net/articles/programming/artificial-intelligence/a-pathfinding-for-beginners-r2003/]