Ladjerem Abderrahmane

Rapport

1. ***Labyrinthe***

Dans la classe labyrinthe on a :

- un constructeur Labyrinthe qui va prendre en argument un pointeur vers un fichier txt il va ouvrire ce fichier et il va le parcourir caractere par caractere (getc()) pour compter les lignes et les colonnes et enregistre les coordonnées des éléments suivants :
    - G : qui représente un gardien on va mettre les coordonnées(nbligne,nbligne) de tous les gardiens dans deux tableaux Xg, Yg
    - a : qui va représenter une affiche sur un mur (vert/hor) on va mettre les coordonnées des affiches dans un tableau Xa,Ya
    - b:qui va représenter une affiche sur un mur (vert/hor) on va mettre les coordonnées des affiches dans un tableau Xav,Yav
    - m: qui va représenter les marques par terre on va enregistrer les coordonnées(i,j) de toutes les marques dans deux tableau Xm,Ym
    - x: qui représente une box on va mettre tous les coordonnées (i,j) de tous les box dans deux tableau Xb,Yb
    - C:qui va représenter le joueur principale (le chasseur) donc comme on en a qu' un on va mettre ses coordonnées dans deux variables Xc,Yc
    - T: qui va représenter le Trésor, comme on en a qu'un on va mettre ses coordonnées dans deux variables Xt,Yt
    - +: qui va représenter le début d'un mur on va enregistrer tous les coordonnées(i,j) et les mettre dans un tableau

● ***Construction du labyrinthe :***

maintenant qu'on a les coordonnées de toutes les composantes du labyrinthe on va les construire mais avant ça on va définir quelques boolean :

- ***reliedLigne :***

  qui va prendre en argument deux coordonnées (X1,Y1) (X2,Y2) et qui va vérifier si les deux "+" dans la même lignes s'ils sont reliés par des "-" o u "a" ou "b" renvoie true ou false sinon

- ***reliedcol :***

  qui va prendre en argument deux coordonnées (X1,Y1) (X2,Y2) et qui va vérifier si les deux “+” dans la même colonne s’ils sont reliés par des “|” ou “b” ou un “a” renvoie true ou false sinon

ainsi on passe à l'étape de construction :

- ***Walls :***

  on va tester avec les deux boolean qu'on a définit si les débuts de murs sont reliés entre eux alors on construit des murs entre les coordonnées du premier point et du deuxième point

- ***Gardien:***

  comme on a 4 différents types de gardiens donc on va créer un gardien du type aléatoire entre les 4 et les mettre dans notre labyrinthe en utilisant les coordonnées enregistrées avant

- ***affiches :***

  - on deux types d’affiches :
  - sur un mur Horizontal : on vérifie si le prochain caractère est ‘-’ donc on la construit en prenant le Y2=Y1+2;
  - sur un mur Vertical : on vérifie si le prochain caractère est différent de ‘-’ donc on la construit en prenant le X2=X1+2;
  - Les affiches ‘a’ et ‘b’ n’ont pas la même texture.

- ***autres elément :***

  pour les autres élément on va mettre les objets en prenant les coordonnées quand on parcours les tableaux des coordonnées

- en dernier on va mettre data[x][y] ==1 ou x et y représente les coordonnées des objets créés comme ça on peux pas passer à travers, pour les murs on doit parcourire tous les x et y entre les coordonnées du premier et du deuxième mur

2. ***Chasseur :***

- ***Vie :***

  le chasseur a un certains point de vie défini par un int vie = 1000, a chaque fois que les gardien touche le chasseur avec la fireball les points de vie vont diminué

- ***force :***

  cette valeur représente de combien les points de vie du gardien touché par la boule de feu du chasseur vont diminuer
  la force de chasseur est de 50

- ***Boule de feu :***

  si la boule de feu tirée par le chasseur s'explose sur l'un des autres gardes les points de vie des gardien vont diminué:
  gardien.vie = gardien.vie -chasseur.force

- ***téléportation :***

  le chasseur peut se teleporter dans le labyrinthe si il se met sur la marque(marques[0]) par terre qui est representer par la fleche jaune et en appuyant sur le click droit de la souris le chasseur va se teleporter a la position de la marque (marques[1]) qui est representer par la fleche rouge, la teleportation peux pas se faire dans l'autre sens

3. ***Gardien :***

- ***type de gradient :***
  - on a 4 type de gradient :
    - Blade
    - Lézard
    - Serpent
    - Samourai
- ***points de vie :***
  - chaque type de gradient a des points de vie :

- Blade -> 300
- Lézard -> 500
- Serpent -> 200
- Samourai -> 1000

A chaque fois que le gardien est touché par la boule du chasseur les points de vie vont diminuer.

- ***force :***

  cette valeur représente de combien les points de vie du chasseur touché par la boule de feu d'un des gardiens vont-ils diminuer chaque modèle de gardien a une force :

  - Blade -> 30
  - Lézard -> 40
  - Serpent -> 20
  - Samourai -> 50

  chasseur.vie =chasseur.vie -gardien.force

- ***Mort :***

  pour chaque gardien on une variable de type boolean qui va nous dire si le gardien est mort ou pas cette valeur est par défaut à false dès que les points de vie du gardien en question passe à 0 la valeur de mort = true, on va avoir besoin de ce boolean dans les fonction qui suivent, un gradient mort tombe et peut plus se déplacer ni tirer .

- ***Déplacement :***

les gardien font des déplacement dans le labyrinthe selon l'angle de ce dernier :

- ***Gardien::move_g(dx,dy) :***

cette fonction va nous renvoyer true si le déplacement est possible (si il y a rien au coordonnées (_x+dx,_y+dy) et elle effectue le déplacement, false sinon.

- ***Gardien::déplacement():***

selon l'angle du gardien en question si le déplacement est accepter et le gardien n'est pas mort donc on avance à une vitesse = 0.05 sinon les gardien vont se déplacer rapidement :
on a le tableau qui contient les quatres angles principales :
angles = {0,90,180,270};

- **0 <=angle<90 (y+0.05)**
  - if(move(0,0.05)) // si le déplacement est accepté on met a jour les coordonnées
- else angle=angles[u]; // on change de direction aléatoirement //(u = rand() % taille de tableau angles)

- **90 <=angle<180 (x-0.05)**
  - if(move(-0.05,0.0)) // si le déplacement est accepté on met à jour les coordonnées
  - else angle=angles[u]; // on change de direction aléatoirement //(u = rand() % taille de tableau angles)

- **180 <=angle<270 (y-0.05)**
  - if(move(0.0,-0.05)) // si le déplacement est accepté on met à jour les coordonnées
  - else angle=angles[u]; // on change de direction aléatoirement (u = rand() % taille de tableau angles)

- **angle=>270 (x+0.05)**
- if(move(+0.05,0.0)) // si le déplacement est accepté on met a jour l les coordonnées
- else angle=angles[u]; // on change de direction aléatoirement (u = rand() % taille de tableau angles)

- ***attack():***

Cette fonction va permettre aux gardiens d'attaquer le chasseur en vérifiant avant si le gardien en question n'est pas mort :

Dès que le chasseur rentre dans un rayon de 20.0 de distance entre lui et le gardien, le gardien va se retourner vers le chasseur et va lui tirer une boule de feu initialisé en utilisant la fonction Gardien::fire().

Puis on va voir si la boule de feu touche le chasseur alors les points de vie du chasseur vont diminuer.

- ***update():***

dans cette fonction on appel les deux fonctions de déplacement et attack leur appel dans cette fonction va nous permettre de les appeler chaque un slot de temps et ça va permettre d'avoir les gardien en déplacement tout le temps et puis si on se rapproche du rayon du gardien il va nous attaquer

4. Partie terminée :

la partie peut se terminer de deux façons ;

- ***Gagnée:***

  si on arrive à tirer 10 fois sur le trésor, si le trésor est touché on a un message qui nous affiche le nombre de fois qui nous reste à tirer avant d' ouvrir le trésor et si il est ouvert alors on a gagné

- ***Perdue:***

  si à force de se faire attaquer les points de vie du chasseur == 0 (chasseur mort) alors la partie est perdu

5. quelques modification:

- textures de jeu :
  . walls
  . Sol
  . affiches 'a'
- Mover.h :

    - int vie qui représente les points de vie pour chaque gardien y compris le chasseur
    - bool mort qui représente si le gardien ou le chasseur est mort ou pas
    - int force qui représente la force du chaque gardiens même le chasseur
- Fireball.h :
    - j’ai ajouter une fonction get qui renvoie l'état de la fireball (pour corriger un bug de tir, même si la fireball est explosé le chasseur recevais des dégâts, alors pour régler ça j’ai ajouté cette petite fonction)

6. Difficultés :

- mettre à jour le 1 dans data[gardien.x][gardien.y] donc pour toucher un gardien il faut faire exploser une boule de feu sur un mur pas loin du gardien.
- une première fois que je travaille sur un projet de C++ orienté objet