Zhao Yuge
Mailly Charles

Sujet n° 5 : Labyrinthe

https://github.com/Cmdu76/ProjetEsigelec/

SOMMAIRE

Introduction :

Notre projet est un programme en langage C qui permet de laisser l'ordinateur de résoudre lui-même un problème de labyrinthe et nous avons réalisé une version graphique avec la bibliothèque SDL2.

Notre projet consiste en 3 parties principales (dans le menu), soit le Jeu, l'algorithme 1 et l'algorithme 2.

Dans le 'Jeu', l'ordinateur va générer un labyrinthe et va l'afficher, puis la souris dans labyrinthe cherche automatiquement le fromage. En cliquant sur une case vide dans le labyrinthe, l'utilisateur peut y ajouter un fromage (droite), une souris (gauche) et un chat (milieu). Le but des souris est de trouver le fromage et le manger, et le but des chats est de suivre les souris afin de les manger. De plus, s'il n'y a pas de fromage ni de chat dans le labyrinthe, les souris ne bougeront plus.

Les algorithmes 1 et 2 sont des supports pour expliquer les méthodes générales des résolutions de ce problème de labyrinthe.

Dans l'algorithme 1, en appuyant sur le 'Enter' l'ordinateur affichera pas par pas le mouvement de la souris, pour expliquer la méthode qu'utilise la souris pour trouver les fromages.

Dans l'algorithme 2, l'ordinateur effectuera la génération des labyrinthes pas par pas.

Par ailleurs, le programme possède une petite musique d'ambiance.

I – Spécifications et conception détaillée

1- Spécifications :

Fonctionnalités

- Menu permettant de naviguer entre les différentes parties du programme
- Partie Algorithme n°1, utilisée comme support pour l'algorithme de l'intelligence des souris
- Partie Algorithme n°2, utilisée comme support pour l'algorithme de génération des labyrinthes
- Partie Jeu :
  i) Ajout de souris
  ii) Ajout de fromages
  iii) Ajout de chats

IHM

- Depuis le menu
  - Clic sur les différents boutons pour naviguer
- Depuis le jeu
  - Echap pour retourner au menu
  - Clic gauche pour ajouter une souris
  - Clic droit pour ajouter un fromage
  - Clic du milieu pour ajouter un chat
- Depuis l'algorithme 1
  - Echap pour retourner au menu
  - Entrée pour faire avancer la souris d'un pas
- Depuis l'algorithme 2
  - Echap pour retourner au menu
  - Clic gauche pour faire avancer d'une étape la génération
  - Clic gauche, si la génération est en cours, pour la mettre en pause

Globalement, les fonctionnalités planifiées ont été respectées. Hormis, le fait de demander à l'utilisateur sa préférence entre un labyrinthe généré ou chargé. Nous avons trouvé que la génération aléatoire était plus drôle pour le joueur. Et nous utilisons quand même la partie chargement depuis un fichier dans la partie Algorithme n°1, et aussi pour les tests dans les débuts du projet, avant la réalisation de la génération automatique.

2- Conception détaillée :

Structures employées

- Node : Nœud permettant à la souris de se repérer dans son exploration
- Sprite : Image simplifiant l'utilisation de la SDL pour l'affichage
- Context : Elements de la SDL (Fenêtre, Renderer, Icône, Curseur, Musique)
- Maze : Tableaux des cases, souris, fromages, chats.
- Mouse : Sprite et intelligence pour une souris
- Cat : Sprite pour un chat
- Cheese : Sprite pour un fromage

Fichiers et fonctions

- Structs.h : Utilisé pour stocker les structures énoncées plus haut
- Defines.h : Utilisé pour définir des constantes
  - Taille de la fenêtre et images
  - Enumérations des States, Direction, et des différentes textures
- Utils.h/c : Utilisé principalement pour simplifier l'usage de la SDL
  - Fonctions pour manipuler les Sprites
  - Fonctions pour manipuler le Context
  - Fonction pour obtenir la position de la souris
  - Fonction pour obtenir des nombres aléatoires
- Application.h/c : Utilisé pour gérer la boucle principale du programme
- States.h/c : Utilisé pour répartir le travail dans le State, et de pouvoir en changer
- Game.h/c & Menu.h/c & Algo1.h/c & Algo2.h/c : Fichiers gérant les fonctions relatives aux States, dépendant de l'écran dans lequel on est.
- Entities.h/c : Fichiers gérant les entités (chat, souris, fromage)
  - Fonctions gérant la gestion (création, destruction)
  - Fonctions gérant la position (lien entre position et position de la Sprite)
  - Fonctions gérant l'affichage
- Path.h/c : Utilisé pour l'intelligence des entités
- Maze.h/c : Utilisé pour la manipulation simple du labyrinthe
  - Création & Destruction
  - Affichage
  - Modification
  - Lecture / Sauvegarde depuis un fichier
- MazeGenerator.h/c : Utilisé pour générer un labyrinthe

Fichiers de données

Le seul type de fichier que nous utilisons est celui permettant de gérer la sauvegarde et la lecture d'un labyrinthe. Mais la fonctionnalité génération de labyrinthe rend son utilité secondaire.

II – Méthode générale de résolution

1- L'intelligence de la souris et du chat :

Pour faire bouger les souris vers le fromage dans le labyrinthe, nous calculons d'abord les positions suivantes pour chaque souris et chat, puis nous modifions les coordonnées des souris et des chats dans le labyrinthe et enfin nous renouvelons les images dans la fenêtre avec les nouvelles positions des événements afin de réaliser l'animation.

Le calcul des prochaines positions pour les souris se fait en 3 étapes. Nous construisons tout d'abord un labyrinthe dans la 'mémoire de la souris', c'est un tableau où est enregistré tous les pas (on l'appelle une 'node') possibles dans le labyrinthe actuel pour la souris à partir de sa position initiale. Chaque node pour le souris contient les informations suivantes : les coordonnées de la node actuel, les coordonnées de la node précédente (le parent, pour faire retour les souris dans le cas d'un chemin mort) et une variable 'test' indiquant si le chemin avait été testé par le souris ou pas. Ensuite nous cherchons dans le 'mémoire' de chaque souris la node correspondant à la position actuelle du souris. Enfin nous déterminons ses 'enfants' qui n'ont pas encore été testé par le souris, si les enfants existent, nous en prenons un comme le pas prochain, s'il n'en existe pas, c'est un chemin mort ou un chemin déjà testé, nous retournons la souris en prenant la node parent du node actuelle comme le prochain pas.

Pour le prochain mouvement du chat, nous déterminons d'abord s'il y a des souris à coté, puis s'il en a existe, nous enregistrons cette souris comme une variable dans la structure du chat et nous prenons la position de cette souris comme le pas prochain pour le chat.

2- La génération du labyrinthe :

La génération du labyrinthe s'est faite en 2 étapes.

La partie commune aux deux étapes est le début qui consiste à remplir les bords comme des murs. Puis remplir, ce que nous avons appelé les WallNodes, qui sont obligatoirement des murs.

Lors de la première approche, on prenait au hasard des WallNodes et on leur ajoutait un nombre aléatoire de côtés, tout en respectant le fait de ne pas séparer les labyrinthes en plusieurs parties. Le problème de cette méthode, c'était que le labyrinthe générait des « îlots », chose que nous voulions éviter pour réduire la complexité de l'intelligence des souris et aussi avoir un labyrinthe plus beau visuellement.

La seconde approche quant à elle prend 4 WallNodes en les connectant aux bords du labyrinthe, une au hasard sur chaque côté du labyrinthe. Ces Nodes servent en quelques sortent de racines afin de générer la suite. Toutes les racines peuvent se connecter à d'autres Nodes et les Nodes ainsi ajoutées peuvent elles aussi devenir de futures racines pour la suite de la génération. Cela assure que le labyrinthe sera rempli à la fin.

Lorsque le labyrinthe est terminé, nous le texturisons avec différentes textures pour les murs et les chemins. Les textures sont choisies aléatoirement parmi quelques-unes en fonction de leur type.

III – Détails éventuels des parties difficiles ou originales

La génération de labyrinthe était un défi qui a demandé de la réflexion afin de produire un résultat intéressant développable en un algorithme par la suite. Plusieurs essais ont dû être réalisés sur papier afin de réfléchir à la façon dont notre logique humaine le ferait, puis retranscrire cette logique en algorithme puis en langage C.

L'intelligence des souris a elle aussi été un casse-tête. Nous avions prévu de faire des souris qui se dirigeraient directement vers la bonne direction. A la place nous avons opté pour une intelligence « primaire », dans le sens où la souris aurait à explorer le labyrinthe. Cette amélioration était plus difficile à implémenter mais le résultat obtenu est plus sympathique.

IV- Problèmes rencontrés et solutions apportées

1- Les erreurs cachées :

Dans la partie de la génération du labyrinthe dans la 'mémoire de la souris', il y avait un crash de notre programme. Puisqu'il y a pas mal de boucle dans la fonction 'generateMouseTree' et quand il y a des petits erreurs dans les conditions de la sortie des boucles, le programme ne peut plus continuer, et de plus c'est difficile de trouver des erreurs, car on aura aucun affichage dans la fenêtre pour ces calculs.

Pour résoudre ce problème, on a utilisé le 'printf' pour afficher les résultats calculés par l'ordinateur. Et comme ça on peut trouver petit à petit les erreurs et compléter nos idées et nos fonctions.

2- L'ajout des chats :

Après on ajoute les chats dans le labyrinthe, nous devons faire beaucoup de modification pour des mouvements des souris pour qu'ils puissent évider de rencontre les chats et fuir des chats. Et de temps en temps des souris ou des chats bougent bizarrement.

Pour résoudre ce problème, nous avons testé dans plusieurs situations la rencontre des souris et des chats et nous avons ajouté petit à petit des conditions et des nouveaux calculs pour les mouvements des souris pour chaque type de rencontre : par exemple, nous 'ouvrirons' les chemins 'fermés' pour les souris puisent les reprendre pour fuir des chats.....

V- Les écarts par rapport au cahier des charges

Par rapport aux objectifs prévus, nous avons corrigés quelques choses.

Tout d'abord nos souris sont plus intelligentes dans le sens où elles ne sont pas « sur-intelligente ». Elles font des essais et retiennent leurs erreurs. Elles font aussi des tentatives d'esquives des chats qu'elles peuvent rencontrer.

La génération du labyrinthe, qui était vue comme une option, a été rajoutée, car elle rendait le jeu plus intéressant.

Le planning a été aussi un peu modifié. En effet, nous sommes allés vite dans les premières séances, car la SDL, le labyrinthe et les entités ne nous ont pas dérangées. Une fois que nous avons choisi d'améliorer l'intelligence des souris, nous avons perdu du temps à cause de la complexité de leur intelligence, cela nous a pris 3 séances pour arriver au résultat actuel. La génération nous a pris 2 séances, une pour chaque implémentation, comme expliqué plus haut.

Bilan :

Paragraphe personnel Charles :

Adepte du C++ depuis quelques années et de la programmation de jeux, ce projet avait beaucoup de points similaires avec d'autres projets que j'ai déjà pu réaliser, ce qui nous a aidés à planifier efficacement le cahier des charges dès la première séance. L'utilisation de la SDL était une première pour moi, mais ayant beaucoup de ressemblance avec la SFML, l'adaptation fut rapide. Travailler en équipe avec GitHub était un aspect qui nous permis de gagner du temps et de suivre le travail de l'autre plus simplement.

Paragraphe personnel Yuge :

Comme c'est la première fois que je réalise un projet informatique assez complexe, j'ai bien vu les différents étapes pour construire un programme, j'ai appris pas mal des nouveaux codes dans le langage C (par exemple les pointeurs, le 'enum'...) et je connais bien le fonctionnement de la bibliothèque SDL2 et je peux l'utiliser pour faire une version graphique.

Même si nous avons pris beaucoup de temps pour réaliser ce programme, nous avons pris plaisir à ce projet, en particulier quand on a résolu un problème difficile.

Conclusion :

Ce projet traitait des thématiques intéressantes au niveau algorithmique et c'est en partie ce qui a orienté nos choix vers ce sujet. En effet, la programmation d'intelligence artificielle et de génération sont complexes et demandait une réflexion avec des problèmes, comme les chats et les impasses pour les souris ou la génération aléatoire tout en respectant certaines règles. Le travail en équipe sur un projet concret était une expérience enrichissante pour notre futur et tout aussi passionnante.