



Rapport de projet labyrinthe

He Hendrick , Mangione Hugo

Récapitulatif des fonctionnalités attendues

- Le plateau de jeu est à réaliser sur la console ;
- Le labyrinthe est un plateau de jeu à 2 dimensions ;
- Il doit contenir au moins une entrée (pouvant servir de sortie) et éventuellement d'autres sorties ;
- Le labyrinthe contient des murs infranchissables ;
- Le labyrinthe contient des passages secrets : en les empruntant, Thésée se retrouve ailleurs dans le labyrinthe (n'importe où au choix du développeur).


Objectifs

1. L'objectif de ce projet est de recréer le labyrinthe et de permettre à Thésée d'atteindre le Minotaure

Présentation

LABY-ESCAPE :

Le jeu consiste à naviguer dans un labyrinthe et à vaincre un Minotaure pour trouver la sortie. Le labyrinthe est représenté comme une grille de cellules, et les murs sont marqués avec des symboles tels que "□" pour les murs indestructibles et "#" pour les murs destructibles (varie selon le niveau de difficulté). D'autres éléments dans le labyrinthe comprennent le Minotaure ("M"), les portails de téléportation ("O" et "o"), le personnage joueur Thésée ("T"), la sortie ("S") et l'entrée ("E").



Le code génère un labyrinthe aléatoire pour chaque session de jeu en définissant les dimensions du labyrinthe, en plaçant des murs autour des bords et en positionnant aléatoirement le Minotaure, les portails de téléportation, la sortie et l'entrée. Le joueur est invité à entrer son nom, choisir sa difficulté, et le jeu commence.

Architecture du projet

Importation des bibliothèques nécessaires :

random : pour la génération de nombres aléatoires

keyboard : pour la gestion des entrées clavier

colorama : pour la gestion des couleurs dans la console

numpy : pour la manipulation de matrices

pickle : pour la sérialisation des objets Python

mysql.connector: interagir avec la base de données

json: convertir la matrice en chaîne de caractères stockable dans la BDD et inversement

Définition de fonctions pour les couleurs :

blanc(text) : retourne le texte en couleur blanche

blue(text) : retourne le texte en couleur bleue

red(text) : retourne le texte en couleur rouge

yellow(text) : retourne le texte en couleur jaune

magenta(text) : retourne le texte en couleur violet

green(text) : retourne le texte en couleur vert

Demande du nom du joueur :



Le nom du joueur est demandé à l'utilisateur avec des contraintes sur la longueur du nom.

Affichage de l'histoire du jeu :

L'histoire du jeu est affichée avec des explications sur les symboles utilisés dans le labyrinthe.

Définition des dimensions du labyrinthe :

Les variables "hauteur" et "largeur" définissent les dimensions du labyrinthe.

Création de la matrice du labyrinthe :

Une matrice de cases est créée avec des murs représentés par des caractères spéciaux.

Positionnement aléatoire du monstre, des téléporteurs et de la sortie :

Les coordonnées des différents éléments (monstre, téléporteurs, sortie) sont déterminées de manière aléatoire.

Affichage du labyrinthe et des informations du jeu.

Boucle principale du jeu :

Cette boucle permet de relancer une partie à la fin du jeu.

Choix technique réalisés


-Afin de garantir le bon fonctionnement du jeu, une solution a été mise en place pour éviter que l'entrée ou la sortie du labyrinthe soient piégées, rendant ainsi impossible la progression du joueur. Cela a été réalisé grâce à la génération aléatoire des murs intérieurs du labyrinthe.

Une vérification de la faisabilité du labyrinthe a été intégrée au processus de génération. Cette vérification consiste à s'assurer que Thésée peut accéder au minotaure et ensuite atteindre la sortie du labyrinthe. Si ces conditions ne sont pas remplies, c'est-à-dire si le chemin entre Thésée, le minotaure et la sortie est bloqué, un nouveau labyrinthe est généré.

Cette vérification est effectuée automatiquement lors du lancement du code du jeu. Ainsi, chaque fois que le joueur commence une nouvelle partie, il est assuré d'avoir un labyrinthe réalisable, sans obstacles infranchissables pour atteindre le minotaure et la sortie. Cela garantit une expérience de jeu fluide et stimulante pour le joueur.

-Afin d'améliorer l'efficacité lors de l'application des couleurs aux murs, nous avons introduit des fonctions de couleurs spécifiques. Ces fonctions ont été conçues pour simplifier la procédure de liaison entre les couleurs et les éléments du labyrinthe, réduisant ainsi la duplication de code.

Grâce à ces nouvelles fonctions, il devient plus rapide et plus pratique d'appliquer différentes couleurs aux différents éléments du labyrinthe. Au lieu d'écrire à plusieurs reprises le code nécessaire pour chaque couleur et chaque élément, il suffit d'utiliser ces fonctions de couleurs, ce qui permet un gain de temps considérable et évite la répétition du code.



Cela améliore la lisibilité du code, facilite sa maintenance et permet une plus grande flexibilité dans la gestion des couleurs. En utilisant ces fonctions, les développeurs peuvent se concentrer davantage sur la logique du jeu plutôt que sur la gestion détaillée des couleurs, ce qui contribue à une meilleure efficacité du processus de développement.

-Afin de prévenir tout dysfonctionnement potentiel, nous avons introduit une caractéristique spéciale pour le minotaure : il est capable de briser les murs intérieurs du labyrinthe dans la difficulté la plus complexe.

Cette fonctionnalité a été mise en place pour garantir une expérience de jeu fluide et éviter tout blocage ou bug qui pourrait survenir si le minotaure se retrouvait coincé derrière un mur intérieur. En permettant au minotaure de détruire les murs, nous assurons sa capacité à se déplacer librement dans le labyrinthe, poursuivant ainsi sa fuite du joueur.

Cette particularité apporte une dimension dynamique au jeu, rendant les interactions entre le joueur, le minotaure et l'environnement plus fluides et réalistes. De plus, elle ajoute une dose de suspense et de challenge, car le joueur doit prendre en compte la possibilité que les murs intérieurs puissent être détruits par le minotaure lors de la planification de sa stratégie de chasse.

En implémentant cette fonctionnalité, nous nous assurons que le jeu fonctionne de manière optimale, en évitant les problèmes potentiels liés aux collisions ou aux situations de blocage, et en offrant une expérience de jeu plus immersive et captivante pour les joueurs.


Spécificités de l'application

- utilisation de matrice pour stocker les données du labyrinthe et surtout l'affichage
- utilisation de variables pour gérer toutes les événements et boucle
- utilisation d'un algorithme de recherche en profondeur pour la vérification de faisabilité du labyrinthe
- utilisation de caractères spéciaux pour l'affichage
- génération de façon totalement aléatoire pour toutes les coordonnées des différents éléments
- déplacement à taper sur la console puis à entrée

Soucis rencontrés

Le jeu présente plusieurs problèmes, certains résolus et d'autres toujours pas.

Tout d'abord, le labyrinthe était interminable, ce qui pouvait rendre l'expérience de jeu moins intéressante. De plus, il y avait un problème d'intégration des murs aléatoires, ce qui pouvait entraîner des chemins impraticables ou des passages bloqués.



Un autre souci concerne la connexion entre les différents portails de téléportation. Il semble qu'il y avait un problème de liaison entre eux, ce qui peut rendre la progression dans le jeu difficile ou impossible.

De plus, le code de vérification de la faisabilité du parcours n'était pas optimisé et prenait beaucoup de temps pour effectuer les vérifications nécessaires. Cela peut entraîner des temps de chargement longs ou des retards dans le jeu.


L'affectation des touches de déplacement en utilisant la bibliothèque keyboard est également problématique. Il peut y avoir des conflits ou des difficultés pour les joueurs à utiliser les touches de manière intuitive.

Un autre point à souligner est lié à l'enregistrement de la partie. Les joueurs peuvent sauvegarder leur progression mais ne peuvent pas reprendre une partie ultérieurement.

De plus, il semble que les murs, les portails et le Minotaure soient tous générés aléatoirement sur une même case, ce qui crée une superposition d'éléments et rend le jeu confus ou impossible à jouer lors l'apparition.

L'usage de deux types de murs dans le labyrinthe avec l'un utilisant un caractère spécial, avait aussi posé problème, car soit le minotaure soit thésée pouvait les briser/ se déplacer dessus.

D'autre part, pendant un temps, il arrivait que le minotaure devienne invisible, continuait de bouger après sa mort, ou sortait carrément de la matrice lors de ses déplacements.



Enfin, l'apparition du fil d'Ariane a causé la suppression de l'entrée, portail de téléportation, sortie, minotaure, ce qui pouvait perturber la progression et la rendant difficile / impossible .

Axes d'amélioration futur

Pour améliorer le jeu, il serait judicieux d'optimiser le code en créant des classes d'objets pour une meilleure organisation et structure du programme.

En ce qui concerne les touches de déplacement, il serait préférable d'affecter ces actions en utilisant la bibliothèque keyboard. Cela facilitera la manipulation et le contrôle du personnage par les joueurs.

La possibilité de restaurer une partie sauvegarder dans la base de données, car la conversion effectuer pour la sauvegarder présente des problèmes lors de la récupération.

Une base de données connectée en ligne pour stocker les scores et sauvegarde, plutôt que de devoir les stocker localement sur chaque machine.