LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

Ce document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors documentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations complémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET : *Le Labyrinthe des Ombres*

Bonnichon Axel
Zhang Elise
Million Jonas

I- Présentation du projet

Notre projet porte le nom de ‘‘Labyrinthe des Ombres’’, il s’agit de la génération aléatoire d’un labyrinthe dont le but est de s’échapper. On est donc plongé dans l’obscurité d’un mystérieux labyrinthe, où notre jumeau a décidé de s’y aventurer, et qu’on va devoir retrouver à la fin des différents labyrinthes, après avoir esquivé les différentes et mystérieuses créatures qui peuplent et hantent ces lieux sordides. Le but premier du projet était en la génération aléatoire de labyrinthes, qu’on a décidé d’améliorer en le transformant en un jeu-vidéo, du style horreur et pixel art, réalisé en totalité par nous, du design des personnages à la bande-son, en passant par le programme. Le jeu se comporte de la façon suivante : on débarque dans un labyrinthe dont on doit trouver la sortie tout en esquivant les monstres. Mais ce n’est pas un, ni deux, mais bien trois labyrinthes qu’il faut surmonter pour pouvoir finir le jeu. Trois labyrinthes de trois tailles différentes, et des monstres qui augmentent en nombre en même temps que le labyrinthe. La difficulté est à son paroxysme, dans un labyrinthe gigantisme, dont la légende raconte qu’il est presque impossible d’y survivre. A vous de surmonter ce défi impossible.

II- Organisation du travail

Pour réaliser notre jeu, nous nous sommes partagés les différents rôles et tâches pour mener à bien celui-ci. Notre équipe est composé de trois membres : Axel, Elise et Jonas. Jonas a la charge de la réalisation du programme et du code, visant à faire fonctionner notre jeu. Il s’est occupé du développement du labyrinthe du début jusqu’à la fin, mettant en œuvre nos différentes idées, que nous avions pensées et imaginées. Elise était de son côté chargée de la direction artistique, notamment dans la réalisation des sprites et des frames des différents personnages, monstres, décors et objets. Elle s’est aussi concentrée sur l’affiche de présentation du projet. Enfin il y a Axel, qui s’occupe du côté mathématique dans la compréhension du labyrinthe et des différents calculs liés aux caractéristiques des éléments du labyrinthe, comme le nombre de monstres, les dimensions, les vitesses… Il a aussi créé la bande-son originale, en concordance avec les idées d’Elise. Il a suivi toute la réalisation du projet, s’occupant de prendre en notes les différentes opérations et en rendre compte. C’est donc pendant près de deux mois, que nous nous sommes attelés à la réalisation du projet, réfléchissant en classe pour pouvoir mettre nos idées en commun, pour ensuite nous pencher dessus chez nous en travail personnel, tout en communiquant nos avancés et difficultés via un groupe sur Discord, une plateforme de communication très pratique pour organiser nos différentes discussions sur nos idées.

III- Déroulé des étapes du projet

1. Réalisation des sprites des personnages et autres éléments

Pour récapituler tous les différents éléments et personnages qui ont été réalisés en pixel art, on dénombre : deux personnages principaux, quatre créatures monstrueuses, quatre objets d’aide et le décor principal du labyrinthe.

A) Les personnages principaux

Les deux personnages principaux sont deux jumeaux : un frère et une sœur. Ce fut le travail le plus long et le plus complexe, nécessitant le plus de temps pour leur réalisation. Ils sont réalisés en pixel-art en 64x64 pixels. Le labyrinthe étant fait en perspective, le personnage se déplace sur un plan en 2D perspective, dont il peut s’orienter dans quatre directions différentes : haut, bas, gauche et droite. Ils ont d’abord été réalisés sous la forme de sprite puis sous un enchaînement de frame, pour faciliter la création de déplacement et minimiser le nombre de frame d’un même personnage. Ils possèdent deux boucles : une première dite idle, où ils sont à l’arrêt ; et la seconde en mouvement. Pour plus de réalisme, des mouvements de corps et de vêtements ont été réalisés.

B) Les créatures monstrueuses

Parmi les 4 monstres qui peuplent le labyrinthe, on compte : un chien-loup aveugle, tâché de sang ; un petit slime ; une souris-araignée et un lapin qui crache des larves. Tous sont réalisés sous le format pixel-art en 64x64 pixels et réalisés dans seulement deux directions possibles (droite et gauche mais ils peuvent aller en haut et en bas) sauf pour le slime et suivent la même logique que les personnages principaux pour les déplacements,

réalisés par une boucle de frame. Le fait qu'ils aient été réalisés seulement dans deux directions pour certains relève d'un problème de complexité quant à la réalisation de ceux-ci. Le chien-loup possède une spécificité : si on ne bouge pas, il ne vient pas vers nous. La souris-araignée est similaire dans le fonctionnement et la réalisation du chien-loup. De plus elles peuvent tisser des grandes toiles, qui ralentiront le joueur. Le slime est le seul monstre qui possède une animation peu importe la direction dans laquelle il se déplace. Le lapin cracheur de larves est un lapin qui reste immobile tant qu'on reste éloigné de son champ de vision, puis dès qu'il nous voit, il se transforme et des larves lui sortent de la bouche, avant de se mettre à nous poursuivre. Il est donc animé par deux boucles : au début quand il est immobile puis quand il est en mouvement. Entre ces deux phases, une animation de transformation est réalisée.

D'autres monstres devaient être réalisés pour rajouter de la diversité dans notre florilège de créature : comme un homme nous poursuivant avec une lanterne ou bien une créature qui au lieu de nous poursuivre, apparaîtrait instantané à un endroit au hasard du labyrinthe. Ils n'ont pas été faits par manque de temps, parce que nous avons préféré faire des sprites de qualité fait par les soins de notre directeur artistique, plutôt qu'ils soient bâclés ou pris sur internet. Par simplicité et pour rendre un travail complet, pour la plupart des monstres, seuls les directions latérales ont été réalisées, ce qui est préférable puisque nous avons réalisé toutes les animations nous-mêmes.

C) *Les objets d'aide*

Pour nous aider à échapper ou survivre aux monstres, quelques objets se trouvent aléatoirement sur la carte, afin que nous puissions nous y référer pour nous faciliter l'extraction du labyrinthe. Ces quatre objets se trouvent dans un ou plusieurs coffres selon la taille du labyrinthe, qui est accompagné d'une torche permettant de pouvoir le repérer très clairement. Dans celui-ci, on y trouve aléatoirement un des quatre objets : un morceau de carte, qui visualise une partie de la carte, un gel d'ennemi, une seconde vie et un ciseau, permettant de couper les murs. On avait l'idée de réaliser du soin ainsi qu'une arme, le premier étant dans le cas d'une barre de vie du joueur, le second nous permettant de nous défendre, en tuant l'ennemi.

D) *Le décor du labyrinthe*

Les décors du labyrinthe concernent les chemins et les murs de celui-ci. On s'est inspiré des labyrinthes constitués d'arbustes comme murs. On a donc réalisé des murs y ressemblant car nous pensions que ce serait le meilleur moyen d'apporter une touche d'horreur même sur le décor. On avait aussi pensé à faire que certains murs comportent des marques de sang, pour ajouter un côté glauque. Le sol ressemble à de l'herbe par soucis de réalisme, de telle sorte à avoir une cohérence dans l'élaboration du labyrinthe.

2. *Réalisation de la direction artistique*

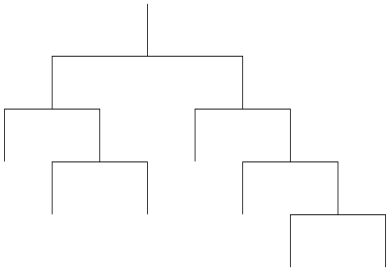
A) *Les écrans de jeu d'introduction en jeu et de fin*

Les écrans de jeu sont très importants dans le déroulement du jeu, parce que ceux sont eux qui mettent le joueur en contexte en début de partie et qui concluent le jeu. L'écran de lancement nous met directement dans le bain, quant au thème du jeu : une télé, vieille, avec une mystérieuse tache de sang (celle de notre jumeau ?). Il y a deux alternatives pour la fin : ou vous gagnez, ou vous perdez. A vous de découvrir votre fin, car les deux réservent bien des surprises.

B) *Bande-son originale*

La bande-son est une création originale. Elle a été réalisée via le logiciel FL Studio, un logiciel de production musicale, basé sur la base de paterns, par des synthétiseurs et des instruments virtuels. Les instruments virtuels suivants utilisés sont les suivants : Carabeic piano, Dark, Grand piano, Music box, orchestral, Silver raw et violon. La musique a été longue à créer et a nécessité plusieurs versions tests. Le tout sur une composition de six temps selon un temps de 20. Pour coller avec l'ambiance horreur, si ce n'est que presque mortifère, on a décidé d'être dans les aigus entre C7 et C8, selon les cordes d'un piano. En plus de cela, pour avoir une atmosphère très sinistre et glauque, les instruments utilisés sont assez stridant et peu agréable à écouter. Il n'y a pas eu de réel problème de réalisation, il fallait juste prendre le temps de trouver les bons instruments et les bonnes notes utilisés.

3. Compréhension mathématique



Le but de la compréhension mathématique est de comprendre comment fonctionne le labyrinthe et voir comment est-ce qu'on peut améliorer sa structure et son fonctionnement. Lors des premiers codes de la génération du labyrinthe, on utilisait une surface quadrillée par des cases d'une certaine taille. On avait instancié un point de départ en haut à gauche, et ensuite un chemin se générait automatiquement suivant un certain chemin principal avec des culs-de-sac. Mais en étudiant la complexité des chemins par un arbre des possibilités (voir exemple ci-contre), on arrivait à la conclusion suivante, qu'il y avait beaucoup trop de culs-de-sac qui piègeraient le joueur beaucoup trop facilement. Entre autres, seul un chemin permettait d'accéder à la sortie, ce qui faisait qu'il était pratiquement impossible de l'atteindre sans tomber sur un monstre, dont on ne pourrait s'échapper. C'est pour résoudre ce problème, qu'on a décidé d'ajouter une fonction aléatoire lors de la génération du labyrinthe, qui permet de casser certains murs, pour pouvoir ainsi avoir un labyrinthe un peu plus ouvert, qui permet au joueur d'user de stratégie, pour échapper aux monstres. D'autres problèmes ont pu être ainsi résolus, comme le fait de déterminer la sortie la plus éloignée possible. Puisqu'il n'y a plus de chemin principal, et que toutes les zones sont facilement accessibles. On a donc programmé pour que la sortie soit située au point le plus éloigné du joueur. Il reste cependant une infime possibilité que la sortie soit juste à côté du joueur mais séparés par un mur, sauf si un coffre se génère à ce même endroit et casse le mur.

4. Organisation du labyrinthe

L'organisation du labyrinthe et des éléments présents est très importantes, puisqu'elle va déterminer la faisabilité du jeu ainsi que la difficulté qui y est associée. Pour cela plusieurs tests ont été réalisés, en modifiant à chaque fois les différents paramètres comme la taille totale du labyrinthe, l'épaisseurs des murs, le nombre de monstres…

A) Le labyrinthe

Pour former trois niveaux de difficultés, trois tailles de labyrinthe ont été mis en place : 3200x2000px, 6400x4000px et 9600x6000px. La taille des murs est de 120px de côté. Pour les zones autour du coffre, les murs sont effacés pour permettre d'avoir une sorte de petite salle, facilement percevable.

B) Les monstres

De nombreux tests et projections ont été faites pour déterminer au mieux le nombre d'ennemis et leurs capacités. Le nombre est déterminé selon la formule : $\frac{(largeur\ labyrinthe * hauteur\ labyrinthe * proportion\ monstres)}{1000\text{^}2}$

Eléments		(Vitesse, distance vision, nombre, distance d'apparition)	Nombre			
			1000x1000	3200x2000	6400x4000	9600x6000
Monstres	Perso	(1, x, 1, x)	1	1	1	1
	Blobs	(0.4, 200, x, proche)	0.5	3	12	28
	Chiens	(1, 600, x, loin)	0.125	0	3	7
	Souris-araignée	(0.8, 300, x, moyen)	0.375	2	9	21
	Lapin	(0.9, 450, x, proche)	0.2	1	5	11

C) Les objets

Eléments		Nombre		
		3200x2000	6400x4000	9600x6000
Objets	Torches	1	3	5
	Coffre (les objets étant présents à l'intérieur sont uniques)	1	3	5

5. *Réalisation du code et du programme du jeu*

A) Développement du code principal du jeu

Le code a été réalisé de A à Z par notre programmeur informatique, en charge du développement du jeu, en accord avec la personne étudiant la complexité mathématique. Tout d'abord, la première étape a été de créer tout simplement un labyrinthe qui se génère automatiquement en 2D, et de se mettre d'accord, de vérifier, si le programme de base était fonctionnel dans sa globalité. Cela passe par l'optimisation d'un menu de debugging, dans la programmation des collisions et des hitbox du personnage avec les murs du labyrinthe. Après avoir été sûr de la bonne tenue du fondement de notre code, nous avons implanté des ennemis, sous la forme de monstres. Ces ennemis sont des IA (cf. III.1.B), qui déambulent dans le labyrinthe mais sont, à partir d'une certaine distance du joueur, attirés par celui-ci, et donc pour pouvoir réaliser cela, nous nous sommes basés sur l'algorithme A*, qui a pour rôle de guider les IA dans le labyrinthe. Les divers ennemis et leur sprite ont été implantés tout au long du développement de ceux-ci. Puisque notre labyrinthe a pour vocation d'être dans une ambiance horreur, il fallait bien le plonger dans l'obscurité la plus totale. Néanmoins pour guider le joueur, il fallait l'éclairer d'une source de lumière, qui était sa lanterne. Pour se faire, nous avons utilisé la bibliothèque PyGame_Light, qui permet ainsi au joueur de pouvoir voir son chemin soit voir au travers des murs.

B) Développement des éléments mineures, de l'optimisation de l'expérience de jeu

Le reste du développement a été dans l'optimisation des divers éléments, comme le pathfinding des IA, de la génération du labyrinthe comme vu dans la partie de la compréhension mathématique, du rendu de l'image/son… Néanmoins plusieurs fonctions majeures à la bonne expérience de jeu ont été faite : l'ajout d'un deltaTime, de la bande-son ou l'ajout d'un menu, qui permet au joueur de modifier le rendu gamma, le volume voire l'assignation des touches. La correction des bugs a aussi été un point critique de la finalisation du projet, pour avoir un jeu qui fonctionne le mieux possible.

C) Résolution des bugs

Il y a eu plusieurs bugs dans la conception du code, notamment beaucoup concernant les collisions et les déplacements, mais aussi sur l'éclairage avec la bibliothèque PyGame_Lights, ou encore des bugs mineurs, facilement réglés. Tous ces problèmes ont été en premier repérés à la suite de nombreux tests effectués par l'équipe de développement mais aussi des personnes tierces, ce qui nous a permis de rapidement résoudre, avec du temps et du labeur, les différents bugs et problèmes qui en résultaient.

IV- Conclusion du projet dans sa globalité

Le projet a été mené tel que l'on le voulait, dans sa globalité. Notre idée de base, de créer un jeu d'horreur mêlant ainsi la génération automatique et aléatoire du labyrinthe, s'est très bien réalisée. Bien évidemment, ayant prévu beaucoup de choses et voyant certaines beaucoup trop en grand, certaines idées de monstres, d'objets, de structures de jeu et de développement du labyrinthe n'ont pas pu être réalisées. On aurait sûrement pu aller plus loin sur la conception de certains éléments du jeu, comme un sprint, qui permet au joueur de fuir plus facilement les ennemis. Au cours de la réalisation du projet, à chaque nouvel ajout ou modification, des tests étaient directement menés pour ainsi le bon fonctionnement continuel. Le fait d'avoir organisé les différentes parties du code en plusieurs fichiers et ensuite au moyen de classes séparées nous a justement permis de nous assurer qu'il y aurait le moins de bugs possibles et s'il y avait de pouvoir les localiser assez facilement.

Le jeu ne comporte aucun bug, tourne de manière assez fluide même sur des ordinateurs qui ne sont pas très puissant. On est fier d'avoir réalisé notre projet de nos propres mains du début jusqu'à la fin, surtout sur le code et la direction artistique. Ne reste plus qu'à y jouer et à en profiter, qu'on soit français, anglais, voire allemand, puisque le jeu est disponible dans ces trois langues.

Documentation technique du projet

I- *Spécifications fonctionnelles*

Pour lancer le jeu :

1) Ouvrir un cmd et mettre la commande "cd chemin/du/fichier" puis "pip install -r requirements.txt"
2) Lancer le programme mazeGame et apprécier le jeu ;)

Etapes d'exécution :

- Création du labyrinthe
- Création des ennemis, du joueur et des salles de coffre

Fonctionnalités :

- Utilisation de l'algorithme A* pour le déplacement des ennemis ainsi que de l'algorithme DFS pour générer le labyrinthe
- 3 langues disponibles (français, anglais et allemand)
- Possibilité de choisir la difficulté entre facile, moyen et difficile
- Nombreux paramètres comme la luminosité, le volume ou encore la possibilité d'activer ou désactiver certains effets.

I) *L'algorithme A**

L'algorithme A* fonctionne de la manière suivante :

1) Initialisation de la case de départ et de la case d'arrivée
2) Initialisation de la liste ouverte avec la case de départ
3) Initialisation de la liste fermée comme vide
4) Tant que la liste ouverte n'est pas vide :
    a. Récupérer la case avec le coût f le plus bas de la liste ouverte
    b. Ajouter la case actuelle à la liste fermée
    c. Pour chaque voisin de la case actuelle
        i. Si le voisin est la case d'arrivée, nous avons trouvé le chemin. Le retourner
        ii. Si le voisin n'est pas franchissable ou est dans la liste fermée, le sauter
        iii. Si le voisin n'est pas dans la liste ouverte, l'ajouter à la liste ouverte
        iv. Si le voisin est déjà dans la liste ouverte, mettre à jour son coût f si le chemin actuel est meilleur

5) Si la case d'arrivée n'est pas trouvée, il n'y a pas de chemin de la case de départ à la case d'arrivée.

Définition des termes techniques :

- Coût h (coût heuristique) : une estimation de la distance de la case actuelle à la case objectif. Cela est généralement calculé à l'aide d'une fonction heuristique, qui peut ne pas être exacte mais ne doit jamais surestimer la distance réelle.

- Coût g (coût de mouvement) : le coût de déplacement de la case de départ à la case actuelle. Cela est généralement calculé par la somme des coûts de déplacement de toutes les cases visitées le long du chemin de la case de départ à la case actuelle.

- Coût f (coût total) : la somme du coût g et du coût h pour une case particulière. Cela représente le coût total estimé du chemin le moins cher de la case de départ) la case d'arrivée qui passe par la case actuelle.

II) Algorithme de cherche en profondeur

L'algorithme de cherche en profondeur (DFS) fonctionne de la manière suivante :

1. Commencez par une case départ er marquez-la comme visité
2. Explorez une case adjacente non visitée au hasard, marquez-la comme visitée et répétez l'étape 2 jusqu'à qu'il n'y ait plus de cases adjacentes non visitées
3. Faites marche arrière jusqu'à la case précédente non visitée et répétez l'étape 2 jusqu'à ce qu'il n'y ait plus de cases à explorer
4. SI toutes les cases du graphique ont été visitées, l'algorithme s'arrête. Sinon revenez à la première case non visitée et répétez les étapes 2 à 4

II- Spécifications techniques

Langage : Python exclusivement

Bibliothèques utilisées : pygame, moderngl, numpy, screeninfo, Pygame-Lights

Choix techniques : Nous avons utilisé python comme langage de programmation car il est beaucoup plus simple d'un son fonctionnement surtout concernant la librairie pygame qui est spécialisée dans la création de jeu et qui ainsi nous a permis de développer notre jeu.