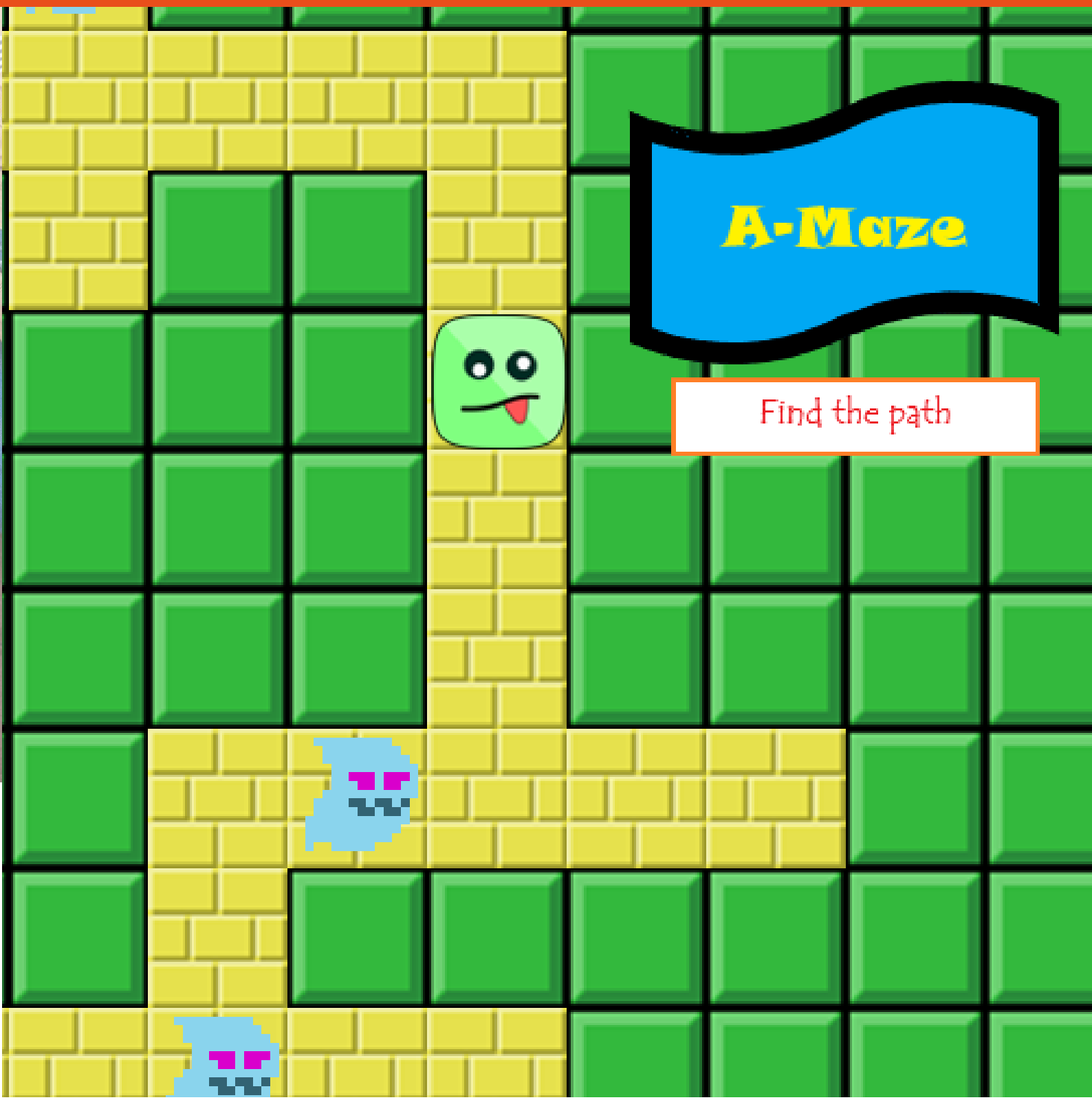
LES TROPHÉES NSI

Édition 2022

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET



Le projet A-MAZE

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 : A-MAZE

> PRÉSENTATION GÉNÉRALE :

• Idée et objectifs

Nous avons eu l'idée de programmer un jeu de labyrinthe, auquel on ajouterai des graphismes, des obstacles et même des clés. L'objectif du jeu est d'arriver à la porte. L'idée de base était de construire un seul niveau qui contiendrait une surprise à la fin, en s'inspirant d'un « prank » tiré d'internet.

Nous souhaitons mettre en valeur l'enseignement de spécialité NSI, qui de notre avis, pourrait encore être développé (NSI expertes en plus des maths expertes peut-être?)

• Origines et intérêts du projet

Nous étions parti sur un projet d'exploration de donjon vu de haut. Mais le projet semblait difficile et il aurait prit beaucoup de temps à être développé. Je dis « nous », cependant, c'est un bien grand mot. Moi, Pottier Victor dirige ce projet et j'ai été mis au courant assez tard de l'existence des trophées NSI (le 30/04!).

Depuis ce jour, j'ai travaillé H24 sur le projet. Le problème réside dans le fait que j'ai programmé l'intégralité du jeu tout seul et j'ai appris seulement le 04/05/2022 qu'il fallait constituer une équipe. J'ai recruté Noah seulement le 5 mai (donc au dernier moment) et nous collaborerons avec Kenan (qui ne souhaite pas participer aux trophées NSI). Personnellement, je tiens énormément à participer à ce concours car l'informatique est ma passion.

> ORGANISATION DU TRAVAIL :

• Présentation de l'équipe (prénom de chaque membre et rôle dans le projet)

Victor Pottier => chef de projet, programmeur principal, level designer

Noah Vitry => level designer, responsable de la créativité ainsi que des aspects visuels et sonores du jeu

Kenan Dalleau => travailleur de l'ombre

• Répartition des tâches

Victor Pottier => je programme le prototype du projet (la version du projet jointe avec ce dossier) et je traduis toutes les idées et solutions apportées par l'équipe en langage Python. J'aide Noah dans la réalisation des niveaux.

Noah Vitry => je crée les niveaux du jeu et propose de nouvelles idées à l'équipe. Je suis chargé d'améliorer l'aspect visuel et sonore du jeu. Je cherche à apporter une meilleure expérience utilisateur.

Kenan Dalleau => je trouve et propose des idées originales afin d'améliorer le jeu. Je m'investis dans le projet mais ne souhaite pas participer aux trophées NSI ni apparaitre dans la vidéo. Toutefois, je suis responsable de la réalisation du montage vidéo associé au projet.

• Organisation du travail (répartition par petits groupes, fréquence de réunions, travail en dehors de l'établissement scolaire, outils/logiciels utilisés pour la communication et le partage du code, etc.)

- Je (Victor) suis l'auteur principal du projet. C'est moi qui ait réalisé l'intégralité du prototype et remplit tous les pré-requis nécessaires afin de participer (monter l'équipe, remplir le dossier…).J'ai monté l'équipe assez tardivement mais je promets cependant de mobiliser l'équipe monté en urgence (je rappelle que je fus prévenu le 30/04) pour améliorer le projet au maximum. Il est possible que nous utiliserons l'environnement replit et que de nouveaux membres soient recrutés lors du développement du projet.

LES ÉTAPES DU PROJET :

• *Présenter les différentes étapes du projet (de l'idée jusqu'à la finalisation du projet)*

1. Mise en place nécessaire à pygame (configuration de l'écran, boucle du jeu, création de l'objet Clock)

2. Division du projet en plusieurs fichiers (main : pour le programme principal, settings : les constantes du jeu, level : la configuration du niveau, game_data : stocker les données qu'exploite la configuration du niveau, game_sprites : classes définissant les sprites du jeu, player : classes définissant le personnage que contrôle le joueur)

3. Structure pour créer un niveau à partir de la classe Level et déterminer la position de chaque éléments. On utilise des groupes de sprites pour différencier chaque sprite et les manipuler efficacement.

4. Le joueur : ses déplacements, les collisions avec le mur

5. Gestion des portes verrouillés, animation de la clé et du verrou lorsqu'il se débloque.

6. Création de la classe Ghost afin de définir les fantômes du jeu. Configuration de leur déplacement et collision avec le joueur (le joueur est téléporté à un checkpoint défini par l'attribut initial_position de la classe PlayerConstructor)

7. Création de différents personnages pour le joueur avec des propriétés différentes. Notamment le FrowningPlayer qui possède un déplacement récursif (tant qu'il n'est pas bloqué, il se déplace en fonction du vecteur défini par l'attribut direction.)

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

• *Avancement du projet (ce qui est terminé, en cours de réalisation, reste à faire)*
Fait :
- Les personnages du jeu et leur caractéristiques
- Leur déplacement
- Les collisions avec les murs
- La structure d'un niveau
- Les ennemis basiques (fantômes)
- Système de clé
- Version piégée du jeu (this one is supposed to a-maze you!)

En cours de réalisation :
- Les différents niveaux / puzzles du jeu (dans le fichier game_data.py)

Reste à faire :
- Nouveaux ennemis, avec des caractéristiques différentes de celles des fantômes
- Améliorer l'esthétique du jeu (les blocs et les chemins pourraient changer de couleur entre chaque niveau car pour l'instant, ils sont respectivement vert et jaune)
- Des objets à collecter (des étoiles par exemple)
- Gestion des cas où il y aura deux personnages à contrôler en même temps
- Ecran de démarrage, de fin et de pause
- Sauvegarde de la progression

• *Approches mises en œuvre pour vérifier l'absence de bugs et s'assurer de la facilité d'utilisation du projet*

- Utilisation de constantes
- Plusieurs tests ont été effectués
- Un fichier .exe pourrait être créé pour faciliter l'accès au projet

• Difficultés rencontrées et solutions apportées
- Gestion des images => les images téléchargées n'avaient pas toujours les dimensions adéquates. L'utilisation du module PIL a permis de redimensionner facilement plusieurs images à la fois.

- Transition pour rendre le déplacement plus agréable visuellement => le déplacement est coupé en 4. Au lieu de se déplacer directement de TILE_SIZE, le joueur se déplace 4 fois de TILE_SIZE / 4.

- Déplacement du FrowningPlayer => déplacement récursif qui a pris du temps à être résolu sans rentrer en conflit avec la transition mentionné çi-dessus

- Déplacement des fantômes => le déplacement du fantôme est défini dans le fichier game_data, H pour horizontal, V pour vertical. Création d'un délai après lequel un fantôme se déplace.

- Trouver des images/sons pour le jeu => les images viennent du site opengameart.org
Sources :
- Les personnages du joueur
- Les blocs, les chemins, la porte, le verrou
- La clé
- Les fantômes
- La musique du jeu
- Les éclaboussures

> OUVERTURE :

• Idées d'améliorations (nouvelles fonctionnalités) => voir fonctionnement et opérationnalité – reste à faire

• Stratégie de diffusion pour toucher un large public (faites preuve d'originalité !)

Version piégée du jeu pour faire peur à des amis (non destiné aux jeunes enfants car trop impressionables). Graphismes attrayants. Page web pour le projet avec notice d'utilisation du jeu. Fichier .exe du jeu pour ceux qui ne seraient pas familiers avec les fichiers .py ou qui n'auraient tout simplement pas d'interpréteurs Python.
Un chronomètre pour mesurer le temps de complétion du jeu. Cela encouragerait les joueurs à la compétition, finir le jeu plus rapidement que l'autre, faire des « speedrun »...

Easter Egg => Dans chaque niveau, il serait bien d'avoir une lettre cachée à découvrir. En assemblant ces lettres à la fin du jeu, on trouve une phrase secrète.(très bonne idée venant de Noah)

• Analyse critique du résultat (si c'était à refaire, que changeriez-vous dans votre organisation, les fonctionnalités du projet et les choix techniques ?)

Premièrement j'aurai monté une équipe plus tôt car il est tout aussi difficile qu'excitant de tout réaliser par soi-même. Deuxièmement, j'aurai essayé d'écrire des fonctions et méthodes plus courtes (la méthode simple_map_setup fait 26 lignes).

DOCUMENTATION

• Spécifications fonctionnelles (guide d'utilisation, déroulé des étapes d'exécution, description des fonctionnalités et des paramètres)

Pour lancer le jeu, il suffit d'exécuter le fichier main.py, je recommande cependant de lancer main(piege).py si vous souhaitez être a-mazed. Pour se déplacer, il suffit d'utiliser les flèches directionelles.

Evitez les ennemis si vous ne voulez pas être réduit en bouillie ! Et récupérez la clé afin d'ouvrir la porte qui mènera au niveau suivant.

Mais attention ! Certains personnages que vous contrôlerez auront des comportements différents. A vous de vous adapter !

Les différents fichiers du jeu :

Main : programme principal. Contient la boucle du jeu dans laquelle nous parcourons les niveaux du jeu (game_levels).

Support : quelques fonctions utiles qui furent reprises d'un ancien projet, elles ne sont donc pas exclusives à notre projet. Elles peuvent être reprises encore et encore pour de futurs projets.

> *=> folder_import : renvoie la liste de toutes les images (converties en surfaces dans un format adapté pour que pygame puisse travailler dessus) à partir d'un répertoire qui contient uniquement des images.*

Level : définition de la classe Level, contenant tous les éléments constituant un niveau. La méthode run permet de dessiner tous les éléments du niveau associé à l'écran ainsi que d'appeler la méthode update de certains sprites (comme le personnage du joueur par exemple)

Settings : fichier contenant les constantes du jeu (la taille d'un bloc défini par TILE_SIZE par exemple)

Game_data : définit la structure des niveaux (où se place chaque élément des niveaux)

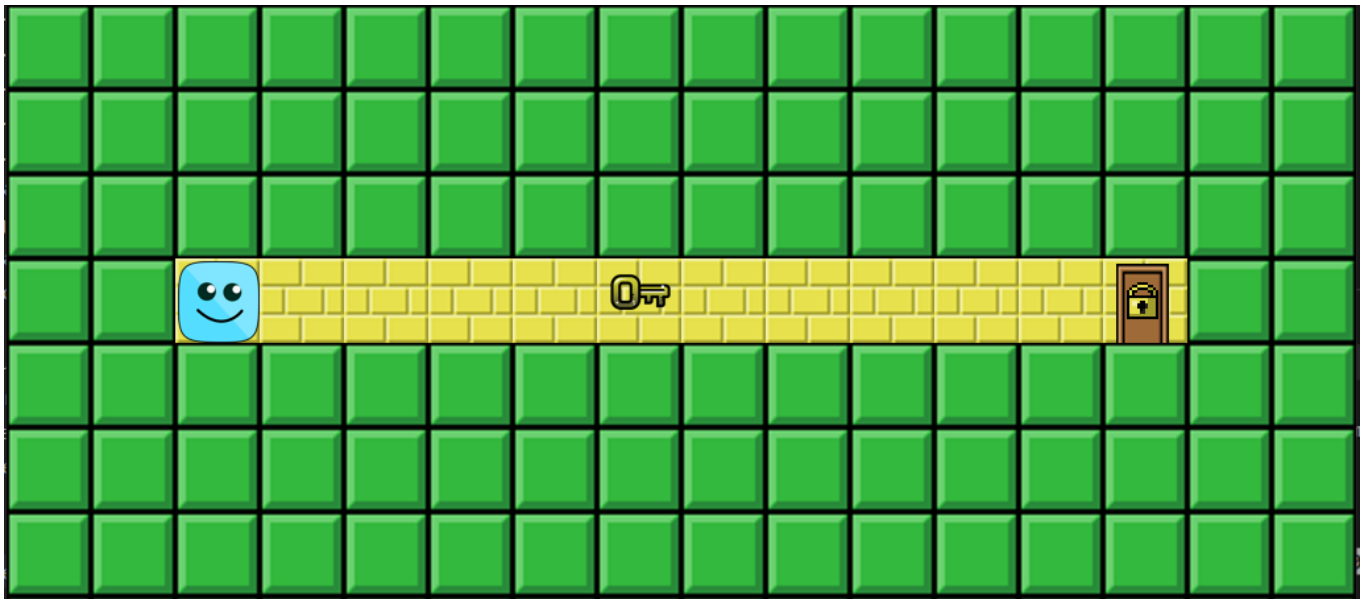
Player : fichier contenant des classes associées au personnage du joueur

Gamesprites : fichier contenant les classes sur lesquelles sont construits les différents sprites du jeu.

• Spécifications techniques (architecture, langages et bibliothèques utilisés, matériel, choix techniques, format de stockage des données, etc)

Le code est découpé en plusieurs fichiers. L'équipe A-MAZE a utilisé le langage Python ainsi que la librairie Pygame afin de réaliser le projet. L'équipe aurait pu utiliser l'application Tiled afin de créer les niveaux du jeu. Cependant, nous avons fait le choix de ne pas l'utiliser car nous craignons que Tiled ne s'inscrive pas dans le cadre du programme d'enseignement de spécialité NSI au lycée.Nous avons utilisé nos ordinateurs POP.

• Illustrations, captures d'écran, etc



Le niveau 1 de A-MAZE