LES TROPHÉES NSI

Édition 2022

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 Sorcier de la Montagne de Feu

> PRÉSENTATION GÉNÉRALE :

• *Idée et objectifs*
• *Origines et intérêts du projet*
• *(...)*

Notre projet est une adaptation en jeu 2D du livre dont vous êtes le héros Le Sorcier de la Montagne de Feu de Steve Jackson et Ian Livingstone. Il est programmé en python et utilise la bibliothèque pygame.

Dans ce livre, le lecteur doit explorer les galeries souterraines d'une montagne peuplée de monstres qu'il lui faudra combattre à la recherche du trésor du sorcier. Le lecteur doit faire des choix et lancer des dés pour déterminer l'histoire, rendant ainsi chaque lecture unique.

Le principal problème des livres dont vous êtes le héros est qu'il faut sans cesse rechercher la page qui correspond au choix que l'on fait : cela rend la lecture peu fluide et il faut faire attention à ne pas se tromper de numéro, sinon il faut recommencer depuis le début ! Avec notre jeu, le joueur n'a pas besoin de se préoccuper des numéros qui correspondent aux textes puisqu'ils sont gérés par l'ordinateur.
Un autre défaut du Sorcier de la Montagne de Feu est que les combats contre les monstres, lors desquels il faut lancer des dés jusqu'à ce que le monstre ou le héro perde tous ses points de vie, sont longs et répétitifs. Dans notre jeu, nous avons donc ajouté la possibilité de passer les combats pour avoir directement le résultat.
Enfin, l'interface graphique offre un aspect plus ludique et permet de se plonger plus facilement dans l'univers du livre. Le seul bémol de notre adaptation, c'est qu'avec un ordinateur, on ne peut pas tricher !

> ORGANISATION DU TRAVAIL :

• *Présentation de l'équipe (prénom de chaque membre et rôle dans le projet)*
• *Répartition des tâches*
• *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.)*

Membres de l'équipe :

- Alice : structure générale du programme, implémentation des modules, etc…
- Thomas : création des images, création des cartes, etc…

Organisation du travail :

- travail au lycée et chez soi.
- partage du code et des images via la plateforme google drive

LES ÉTAPES DU PROJET :

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

- travail au brouillon : carte mentale, schémas, etc…
  -> volonté de faire une carte sur laquelle se déplace le joueur et dont le chemin se dévoile au fur et à mesure de son avancée, avec un texte qui s'affiche lorsque le joueur pénètre dans une pièce.

- création de la classe Vous, qui permet d'enregistrer les caractéristiques du joueur et de les modifier avec des méthodes spécifiques (augmenter ou diminuer les points du joueur, manger pour récupérer de l'endurance, boire une potion, gérer l'inventaire, etc…)
- création et implémentation du système de cartes : une carte est représentée par un tableau dont les éléments sont des objets représentant des types de sol.
- création d'une fenêtre qui affiche pour chaque sol de la grille une image spécifique de 50 pixels de côté.
- programmation du code permettant d'afficher l'avatar du joueur sur la carte. Lorsque l'utilisateur appuie sur les flèches du clavier, l'image se déplace d'une case.
- programmation du code permettant de donner l'illusion que le héros marche à l'aide d'un compteur et d'un timer (module time)
- amélioration de la carte : les sols ne sont affichés que s'ils sont révélés. Lorsque le joueur change de case, il la révèle si elle n'est pas encore révélée. Lorsqu'il rentre dans une pièce, toutes les cases de la pièce se révèlent. Lorsque le joueur arrive à une extrémité de la carte, on affiche une autre carte. Création de la fonction ajouter_bordure() qui change automatiquement les images des carrés d'herbe à proximité de l'eau. Création d'animations lorsque le joueur tombe dans l'eau ou dans le vide.
- création du système de page : du texte s'affiche lorsque le joueur rentre dans une pièce, une nouvelle page peut s'afficher et des méthodes sur l'objet de la classe Vous peuvent être appelées en fonction de la réponse que choisit l'utilisateur.
- implémentation de classes d'objets apparaissant sur la carte (coffres, portes, etc…), qui peuvent faire apparaître des pages lorsque le joueur s'en approche.
- création du système de combat.
- création de l'ensemble des cartes, de toutes les images, ajout d'objets et de monstres sur les cartes, écriture de toutes les pages.

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

• Avancement du projet (ce qui est terminé, en cours de réalisation, reste à faire)
• Approches mises en œuvre pour vérifier l'absence de bugs et s'assurer de la facilité d'utilisation du projet
• Difficultés rencontrées et solutions apportées

Les principales difficultés rencontrées sont liées à la classe Page :
Exemple : lorsque la classe Game appelle la méthodes get_page_suivante() de la classe Page pour pouvoir afficher la page suivante correspondant au choix de l'utilisateur, les méthodes renseignées sur la Page à appeler sur les autres classes (diminuer l'endurance, ouvrir un coffre…) sont appelées avant de renvoyer la page suivante. La classe Game remplace ensuite son attribut self.__page qui contient la page actuelle à afficher par la nouvelle page renvoyée par get_page_suivante() ou par None s'il n'y pas de nouvelle page. Mais parfois les méthodes appelées par la page doivent créer elles-mêmes la page à afficher en fonction de certaines variables et la classe Game écrasait ces pages avec le None renvoyé par la page initiale après l'exécution des méthodes. Il y avait deux solutions : mettre en argument la page initiale dans les méthodes appelées pour qu'elles ajoutent la page suivante à la page elle-même au lieu de changer directement la page à afficher de la classe Game, ou créer une variable globale contenant un booléen (la variable self.__page_a_change de la classe Game) égale à True dans les cas où la classe Game ne

doit pas remplacer self.__page par la page renvoyée par get_page_suivante(). Nous avons choisi la deuxième solution qui nous semblait plus simple à court terme.

Autres bugs fréquemment rencontrés : suite à des fautes de frappes, il arrivait que le programme plante car il ne trouvait pas l'image à afficher. Nous avons donc fait en sorte que le programme affiche une image vide s'il ne trouvait pas l'image recherchée.

Enfin, sur certains ordinateurs, la partie du code permettant d'afficher des rectangles autour des textes lève une erreur. Cet élément étant facultatif, nous vous conseillons de commenter cette partie du code si cela vous arrive (voir la docstring du fichier main.py)

> OUVERTURE :

• *Idées d'améliorations (nouvelles fonctionnalités)*
• *Stratégie de diffusion pour toucher un large public (faites preuve d'originalité !)*
• *Analyse critique du résultat (si c'était à refaire, que changeriez-vous dans votre organisation, les fonctionnalités du projet et les choix techniques ?)*

Idées d'améliorations :

- adapter le livre en entier (le jeu s'arrête à la première partie du livre)
- ajouter des bruitages et une musique de fond
- améliorer les graphismes

DOCUMENTATION

• *Spécifications fonctionnelles (guide d'utilisation, déroulé des étapes d'exécution, description des fonctionnalités et des paramètres)*
• *Spécifications techniques (architecture, langages et bibliothèques utilisés, matériel, choix techniques, format de stockage des données, etc)*
• *Illustrations, captures d'écran, etc*

Pour lancer le jeu, exécutez le script main.py : la fenêtre du jeu apparaîtra.
Pour faire des choix ou se déplacer sur la carte, utilisez les flèches directionnelles du clavier.
Pour valider un choix, appuyez sur entrée.

Langage utilisé : python

Bibliothèques utilisées : pygame, time, random

Images modifiées avec : Gimp, Piskel
Monstres créés avec l' application Gacha Life

Source des images :
Sols : https://stealthix.itch.io/rpg-nature-tileset
Serpent : https://elthen.itch.io/2d-pixel-art-snake-sprites
Objets : https://pipoya.itch.io/pipoya-rpg-tileset-32x32