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 : Super Pablo Maker

> PRÉSENTATION GÉNÉRALE :

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

Le projet de création d'un mini-jeu vidéo de type Platformer 2D avec un bac à sable pour la création de niveaux personnalisés est né de notre désir de créer une expérience de jeu unique et amusante pour les joueurs. Notre objectif principal est de fournir un environnement de jeu engageant qui permette aux joueurs de libérer leur créativité et de concevoir des niveaux uniques et innovants, afin de pouvoir ensuite y jouer. En outre, nous souhaitons offrir une expérience de jeu agréable et stimulante pour tous les joueurs, en mettant l'accent sur les mécaniques, la création et le défi.

L'origine du projet remonte à nos nombreuses discussions et réflexions sur la manière de créer un jeu vidéo qui se démarquerait des autres et qui offrirait une expérience de jeu unique. Nous avons rapidement réalisé que la création de niveaux personnalisés était une fonctionnalité que les joueurs apprécieraient beaucoup, car cela leur donnerait la possibilité de jouer à un jeu qui change constamment et qui n'est jamais ennuyeux. En conséquence, nous avons décidé de créer un jeu de type platformer 2D, car c'est un type de jeu classique et indémodable qui est aimé par des millions de joueurs dans le monde entier.

Le projet présente un intérêt particulier pour les joueurs qui cherchent à exercer leur créativité et leur imagination, en concevant des niveaux uniques et innovants. De plus, le jeu offre une grande variété de défis et d'obstacles qui stimuleront la réflexion et l'analyse des joueurs. En outre, le jeu encourage l'exploration et la découverte, en offrant un environnement de jeu ouvert où les joueurs peuvent s’amuser sur une grille de 32 par 32 blocs.

En conclusion, notre projet de création d'un mini-jeu vidéo de type Mario avec un bac à sable pour la création de niveaux personnalisés est un projet ambitieux qui offre de nombreux avantages et intérêts pour les joueurs. Nous sommes convaincus que ce jeu sera très apprécié par les joueurs et nous sommes impatients de le présenter à ce concours pour qu'il soit évalué par les juges.

> 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.)*

Dans notre équipe, Aldwin Ragot est responsable de la programmation et Ethan Pobrezo s'occupe de la documentation du projet. Nous avons réparti les tâches de manière équitable afin de maximiser notre efficacité et d'assurer une collaboration fluide entre les membres de l'équipe. Nous avons également organisé notre travail en petits groupes, ce qui nous permet de nous concentrer sur des tâches spécifiques et de nous aider mutuellement en cas de besoin.

Nous nous réunissons régulièrement pour discuter de l'avancement du projet, échanger des idées et des solutions aux problèmes rencontrés. Ces réunions nous permettent également de prendre des décisions importantes concernant le design, les fonctionnalités et l'expérience utilisateur. Nous travaillons également en dehors de

l'établissement scolaire, afin de maximiser notre temps de travail et de respecter les délais de livraison du projet.
Pour faciliter notre collaboration, nous utilisons des outils de communication et de partage de code tels que Google docs, drive et Instagram. Ces outils nous permettent de communiquer efficacement et de travailler en temps réel sur le code et la documentation, même si nous ne sommes pas au même endroit physiquement. Nous sommes déterminés à réussir ce projet et à créer un jeu amusant et innovant pour les joueurs.

LES ÉTAPES DU PROJET :

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

1- Whiteboard

Nous avons observés les lauréats des années précédentes et remarqué que personne n'avait crée de créateur de niveaux jouables, encore moins en 1ère. Nous sommes donc partis de cette idée. Pour créer ce jeu, nous aurions besoin de Pygame.

2- Création du mode de jeu « créer »

Nous avons directement commencé le codage de ce mode de jeu, avec l'utilisation de classes, et le code marchait très bien : en 1 semaine, nous avions les bases qui étaient fonctionnelles (2 types de blocs : terre et herbe, placement, destruction).

3- Contourner les classes via des fonctions

Cependant, le concours ne permettant que l'accès aux connaissances de niveau première, il a fallu remodeler tout le programme afin qu'il fonctionne sans les classes, ce qui fut un point crucial dans la création de ce jeu. Après de multiples tentatives, le code ne fonctionnait plus qu'en fonctions.

4- Amélioration

Le problème des fonctions, c'est qu'elles ne permettent pas d'appliquer des paramètres à plusieurs itérations de différents objets. Ainsi, le nombre de fonction était très grand. Nous les avons donc optimisées afin d'augmenter la vitesse du programme et de diminuer le code. Aussi, 3 nouveaux types de blocs ont étés ajoutés : le bloc de spawn, la lave et l'eau (ce qui correspond à l'actuelle gelée).

5- Correction de bugs

Le passage aux fonctions à apporté un lot de bugs notamment sur le placement et la destruction de blocs. Après quelques modifications de checks et de variables, les bugs n'existaient plus.

6- Création du système de sauvegarde et de chargement

Probablement l'une des fonctionnalité les plus dures à ajouter. Pour cela, nous avons utilisé la libraire OS, qui permet la lecture de fichiers. Nous avons créer un message console permettant de taper le chemin d'accès au fichier. L'idée est que nous stockons dans chaque ligne un tuple contenant coordonnées relatives x, y et la couleur du bloc.

Quand nous voulons charger un niveau, nous récupérons pour chaque ligne le tuple qui est en string, et nous passons cette string dans une fonction utilisant la librairie ast, qui permet de passer une string comme tuple si elle est valide, sinon la fonction ne retourne rien. Bien sûr, il a fallu beaucoup modifier ce système afin qu'il fonctionne correctement. De plus, le chargement contenait un bug qui faisait apparaître des blocs aux mauvais endroits, ne les supprimait pas, etc. Ces bugs sont bien sûr maintenant à 100 % réglés.

7- Création du mode de jeu "jouer"

Puisque le chargement et la sauvegarde fonctionnent, nous sommes passés au mode de jeu suivant et nous sommes partis sur la documentation pygame, afin de connaître les différentes choses qui nous serviraient. Après un certain temps, les bases sont crées, sauf que le jeu permet au joueur de passer à travers les murs

8- Collisions infâmes

A l'étape 7 nous étions en début Mars ; A l'étape 8 nous étions en début Avril. La raison ? Les collisions qui ne veulent pas fonctionner correctement. Après s'être renseignés via des forums, nous en avons conclut que des collisions parfaites, ce n'est pas possible. Ainsi, l'idée m'est venue de jouer avec la vitesse de déplacement et vérifier la couleur des blocs à côté du joueur. Au fil et a mesure que le temps passait, les checks devenaient de plus en plus nombreux, et enfin ils marchaient correctement. Après simplification de la fonction, les collisions marchaient parfaitement.

C'est pendant cette période là que le reste des blocs ont étés ajoutés.

9- Amélioration

Un système d'arrière-plan a été rajouté, le programme a été raccourci, … .
Cela concerne aussi certains bugs.

Concernant la sauvegarde/ le chargement, tkinter propose une librairie permettant de sélectionner un fichier avec l'interface homme-machine de windows (explorer).

10- Création d'un launcher

Il nous a ensuite fallu créer un launcher : un programme permettant de lancer les modes de jeux. Ça a été très basique à créer.

11- Ajouts divers

Nous avons rajouté de la musique libre de droit 8bit afin de l'accorder mieux actuellement ; mais nous avons aussi optimisé la boule de feu (aussi nommé Jean Pascal dans l'équipe).

12- Documenter le projet, soit ce que l'on fait actuellement.

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

• *Avancement du projet (ce qui est terminé, en cours de réalisation, reste à faire)*

Le projet à un système de jeu complet, avec 1 type d'ennemi (dont le nombre pourrait être augmenté), 7 types de blocs (dont un bloc d'apparition pour le joueur (spawn), un bloc d'arrivée (finish) et un bloc d'apparition d'ennemis type boule de feu). Le jeu dispose d'un mode création, permettant la conception de niveau jouables (à condition que le niveau contienne un spawn et un finish), et la sauvegarde de ceux-ci. Il y a un système de 8 arrière-plans changeables à souhait. On pourrait ainsi rajouter uniquement plus d'ennemis, d'arrière plan, et des collectables, ainsi que des types de blocs avec des fonctionnalités spéciales (portes verrouillées, clés, ...). Il y a un système de vie et des déplacements assez fluides considérant la limitation imposée (pas de class, limitation de pygame et python,..) ; Ainsi qu'un mode multijoueur et un mode d'accès aux niveaux en ligne.

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

Il y a eu plusieurs tests de jeux rigoureux mettant en scènes des situations parfois allant jusqu'à l'improbable, ainsi qu'un niveau « test » (inclut dans les sauvegardes) afin de s'assurer que le jeu fonctionne correctement. Le fonctionnement entier du programme à déjà été 3 fois retracé afin de corriger les éventuels problèmes. En ce qui concerne la résolution elle même des problèmes, les documentations de python et de pygame / autres librairies ont étés utilisées afin de vérifier de potentielles erreurs, ainsi que des vérifications de variables (via print).

• *Pendant la réalisation de se projet nous avons rencontrés différentes difficultés telles que:*

1- Le système de placement/destruction de blocs
2- Le système de sauvegarde
3- Le système de chargement
4- Le système de changement d'arrière-plan
5- La gelée
6- Le système de collision du joueur
7- Le système de collision des ennemis

> OUVERTURE :

• *Idées d'améliorations (nouvelles fonctionnalités)*

On peut améliorer le jeu en rajoutant des blocs, un sélecteur de musique, un système de monde avec des vies, plus d'ennemis, des améliorations à ramasser pour le personnage, ... ; la liste est infinie.

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

Via Reddit serait une possibilité puisque l'on peut créer une communauté autour d'un projet avec ce site. L'accent serait alors mis sur le côté open source et communautaire, à la manière de RPG Maker.

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

Avec les connaissances d'aujourd'hui, j'aurais moins a consulter la documentation. Je réglerais des bugs bien plus vite et je rajouterais alors plus de contenu.

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*

0] Préambule

Afin d'utiliser ce logiciel il faut les librairies suivantes:

¤Pygame 2.1.2
¤ast
¤PySimpleGUI
¤keyboard
¤tkinter

Il est préférable d'avoir Python 3.7.9 car c'est la version utilisée lors du développement.

Le programme est composé de deux dossiers : Music (pour les fichiers de musique) et Saves (pour les fichiers de sauvegarde)
Le programme est situé dans le dossier du projet lui même.

/!\ Il est important de ne pas mettre les fichiers du jeu (*.py) dans un dossier dans le projet, car cela pourrait causer des bugs concernant la musique ou autre.

1] Lancement du programme

Afin de lancer le programme, il suffit simplement de lancer depuis l'interpréteur python le fichier « launcher.py ». Une fois lancé, le jeu vous accueillera sur le menu principal : C'est d'ici que l'on peut choisir de créer ou jouer à un niveau. Le programme en lui même ne sert juste qu'à afficher une interface, des boutons, et permettre au joueur de choisir le menu souhaité.

2] Le mode « Créer »

Bienvenue dans le mode « créer » ! Vous disposez ici d'une interface permettant de placer ou de casser des blocs, ainsi que de personnaliser un peu plus le niveau (arrière-plan, ennemis).

Voici les commandes de ce mode de jeu :
G : Grid (afficher/masquer la grille de placement des blocs)
I : Inventory (permet l'accès à l'inventaire permettant de placer les blocs et les ennemis
B : Background (permet de modifier l'arrière plan du niveau)
Clic Gauche : permet de placer des blocs
Clic Droit : permet de casser des blocs

Ici vous avez accès à une grille de 32*32 blocs, soit 1024 blocs au total !

Il n'y a pas de véritable bug connus, mais par précaution je conseillerais de ne pas ouvrir l'inventaire plusieurs fois d'affilées (résultat inconnu).

Ce créateur de niveau au démarrage propose de charger une sauvegarde, ce que le joueur va bien sûr refuser la première fois, puisqu'il n'aura pas encore crée de sauvegardes. On arrive sur un écran noir et c'est ici que tout se déroule. De base, le joueur est équipé d'un bloc d'herbe, qu'il peut placer sur l'écran. En sélectionnant d'autres blocs, on peut placer autre chose que de l'herbe, et en quantités désirées (sauf pour le spawn et le finish qui sont limités à 1 chacun).

Le programme en lui même se découpe en 6 fonctions et 1 programme principal.

A) Fonction drawgrid

Elle permet de dessiner une grille qui s'adapte à la taille de l'écran.

B) Fonction Load

Si le joueur veut charger une sauvegarde, alors le jeu propose à l'utilisateur de sélectionner son fichier afin que le jeu fasse appel à la fonction parse_tuple, afin de lire et placer les blocs à l'écran, et de faire en sorte que l'utilisateur puisse interagir avec eux.

C) Fonction parse_tuple

Fonction qui prend une string en entrée (exemple : ‘‘(a, b)’’) et qui la transforme en tuple (exemple : (a, b)). Ici, la sauvegarde sortira uniquement du texte. Il est donc nécessaire de le convertir en tuple.

D) Fonction update
Fonction qui permet d'obtenir des coordonnées de la souris alignées par rapport à la grille

E) Fonction draw

Elle permet de ‘‘placer’’ ou de ‘‘détruire’’ les blocs. Elle utilise la fonction update et ensuite traite des listes contenant les blocs, ainsi qu'effectuer des vérifications (vérifier que le bloc d'en dessous est supprimé si on y place un bloc par dessus, etc.).

F) Fonction drawontop

Cette fonction sert à dessiner les blocs par dessus l'arrière plan lorsqu'il est modifié

G) Programme principal

C'est lui qui va gérer : Les appuis de touche, le chargement, les interfaces, le choix du bloc / de l'arrière plan et la sauvegarde.

Le reste étant assez classique, attardons nous sur ce système de sauvegarde.

La sauvegarde s'effectue en ouvrant un fichier avec droit d'écriture et de lecture. Puis on le truncate (on le vide de tout son contenu), et pour chaque élément dans la liste de blocs, on l'ajoute sous forme de string avec un saut a la ligne. Quand tous les blocs sont passés, on ajoute la couleur d'arrière-plan, on écrit toutes ces données, et on ferme la sauvegarde.
Le fichier de sauvegarde contient des tuples (des objets pygame.Rect) sous cette forme :
(coord x, coord y, couleur)

2] Le mode « Jouer »

Une fois un niveau crée, sauvegardé, et valide (spawn + finish présent), on peut retourner au launcher afin de lancer le niveau. On le sélectionne et le jeu vérifie qu'il est valide. Si oui, le joueur peut y jouer, sinon, il est ramené au menu principal.
Le chargement du mode jouer est presque identique au chargement du mode créer, si ce n'est que l'on y sépare les ennemis des blocs, et que l'on vérifie la présence de certains blocs. Si il y a un problème avec la sauvegarde (niveau non valide, valeurs impossibles modifiées manuellement, etc, alors le jeu prévient de l'erreur trouvée et renvois au menu principal. De même, la fonction parse_tuple est encore présente. On notera enfin la présence de la fonction drawontop, qui est à nouveau présente

Si le niveau se charge, 4 autres fonctions vont gérer le jeu en lui-même :

1) La fonction check_coll, qui permet de vérifier les collisions entre les blocs individuels et le joueur. Il permet la chute du joueur lorsque le sol et absent, et permet (en majeure partie car pygame n'est pas parfait) d'éviter les passages a travers les blocs. Elle permet aussi de savoir si le joueur a touché un ennemi ou de la lave. Pour cela, le jeu récupère à des coordonnées précises les couleurs affichées et effectue certaine tâches en réponse aux résultats

2) La fonction deathp permet quant à elle, de tuer le joueur lorsqu'il tombe hors du niveau, et de le tuer quand il n'a plus de hp. Le jeu crée alors un arrière-plan rouge et permet uniquement au joueur de retourner au menu principal.

3) La fonction draw_enemy permet de créer a partir de la liste des ennemis, plusieurs même type d'ennemis avec un comportement prédéfini, qui est influencé par la fonction check_coll_enemy.

4) La fonction check_coll_enemy est assez générale puisqu'elle permet de donner des collisions a tout types d'ennemis en théorie. Elle se base sur la fonction check_coll.

Voici une Vidéo de démonstation.

(Rappel) Contrôles dans le jeu :


Déplacements gauche/droite (mode jeu)


Saut (mode jeu) ⇒ optionnel : Saut + déplacement : Saut en diagonale

Inventaire des blocs (mode créer)

Sélection de l'arrière plan

Afficher/Masquer la grille (mode créer)

Clic Gauche : Placer
Clic Droit : Détruire