Le Mans Université

Institut d'Informatique Claude Chappe
Le Mans Université

Le Mans Université

Licence Informatique *2ème* année

Module 174UP02 Rapport de Projet

MétaTravers

Bossard Guilian
Mezrhab Bilal
Perron Nathan

24 avril 2024

Lien vers notre GitHub

Table des matières

1 Introduction

Ce rapport présente le travail accompli dans le cadre de notre projet de fin d'année en deuxième année d'informatique. Notre objectif principal était de concevoir et de développer un jeu, exploitant les compétences acquises au cours de notre formation.

Notre jeu est inspiré de l'univers de « Mario Bros », où chaque niveau offre un mini-jeu unique et palpitant. Vous incarnez un personnage intrépide qui traverse ces niveaux, affrontant des pannes système, des virus informatiques et d'autres obstacles numériques. L'objectif est de résoudre les problèmes et de restaurer l'intégrité des systèmes informatiques, tout en explorant des paysages virtuels colorés et imaginatifs. Avec des mécaniques de jeu simples et intuitives, cette aventure promet une expérience accessible et amusante pour les joueurs de tous âges et de tous niveaux.

Chaque défi offre une expérience unique et mémorable, des environnements de bureau traditionnels aux mondes numériques fantaisistes, en passant par des défis de réflexion, d'adresse et de vitesse. Avec une grande diversité de niveaux et de mini-jeux, cette aventure propose une expérience sans cesse renouvelée, où chaque étape apporte son lot de surprises et de découvertes. Mêlant action et résolution de problèmes, notre projet offre une expérience de jeu riche et engageante qui saura captiver les joueurs du début à la fin.

1.1 Description des mini-jeux

— Niveau 1 - Découverte du jeu :

Ce premier niveau offre une introduction simple des mécaniques de déplacements de « MétaTravers ». Les joueurs seront plongés dans un environnement virtuel sobre mais engageant, où ils auront l'occasion de se familiariser avec les commandes de base du jeu. Des obstacles simples mais significatifs seront présents pour guider les joueurs à travers cette première étape de leur aventure.

— Niveau 2 - Problème de refroidissement :

Dans ce niveau, les utilisateurs seront confrontés à un défi technique crucial : un problème de refroidissement menace la stabilité

de la machine principale. Inspiré du concept du jeu « Pipe », ce niveau mettra à l'épreuve les compétences des joueurs pour reconfigurer efficacement le réseau de conduits de refroidissement. Des obstacles supplémentaires et des contraintes de temps seront ajoutés pour augmenter la tension et l'urgence de la situation.

— Niveau 3 - Invasion de virus :

Le système est submergé par une vague de virus informatiques. Les utilisateurs devront faire preuve de rapidité et de précision pour éliminer les virus tout en protégeant les données vitales du système. Des mécaniques de combat simples vues au premier niveau seront de retour pour permettre aux joueurs de repousser cette menace numérique.

— Niveau 4 - Reconstruction critique :

Ce niveau mettra à l'épreuve la capacité des utilisateurs à résoudre des puzzles et à utiliser leur créativité pour reconstruire une pièce essentielle du système. Les joueurs devront collecter et assembler des composants dispersés dans l'environnement pour restaurer la fonctionnalité du système. Un défi unique pour ajouter de la profondeur et de l'intérêt à cette tâche.

— Niveau 5 - Déplacement de données :

Bienvenue dans un labyrinthe numérique complexe où les utilisateurs doivent naviguer à travers une multitude de passages étroits et de chambres pour déplacer des données cruciales d'un point à un autre. Des énigmes de logique et des obstacles variés seront présents pour défier les joueurs tout au long de leur mission. La planification stratégique sera essentielle pour atteindre la destination finale.

— Niveau 6 - Évasion du monde virtuel :

Vous avez atteint le dernier niveau de « MétaTravers » ! Dans ce défi final, les utilisateurs devront utiliser toutes les compétences et les connaissances acquises au cours de leur aventure pour trouver un moyen de sortir de ce monde virtuel et de revenir à la réalité. Les obstacles seront plus complexes et les enjeux plus élevés que jamais. Seuls les joueurs les plus déterminés et les plus habiles réussiront à

surmonter ce dernier défi et à échapper à ce monde numérique.

2 Organisation du travail de chacun

La création du groupe s'est déroulée de manière naturelle, basée sur les affinités de chacun et sur la motivation commune envers le projet. Chaque membre était conjointement motivé et prêt à s'investir dans le travail nécessaire à fournir.

2.1 Émergence du projet

Le processus d'émergence du projet a été marqué par des discussions approfondies concernant le type de jeu à développer ainsi que son déroulement. Initialement, l'idée de concevoir un jeu dans le style de « Mario Bros », avec des plateformes et des ennemis, était prédominante. Cependant cette approche a rapidement été remise en question, car elle ne permettait pas d'incorporer suffisamment d'éléments personnalisés. C'est ainsi qu'est née l'idée novatrice de créer une série de mini-jeux tout en préservant l'essence du concept initial, offrant ainsi une expérience plus riche et variée aux joueurs.

Dans notre quête d'insérer une touche personnelle distinctive dans le jeu, nous avons entrepris une reflexion approfondie sur la trame scénaristique et les personnages. Diverses idées ont été explorées, telles que la réinvention de l'histoire de « Mario Bros » ou la création d'un univers fantaisiste. Cependant, nous avons opté pour une approche innovante en choisissant un concept lié à l'informatique, où le personnage principal voyagerait à travers les systèmes pour les réparer. L'idée d'un agent spécial revêtant un costume et des lunettes de soleil est apparue comme une évidence, symbolisant son engagement dans des missions complexes et risquées, à l'instar de « Neo » et de « Trinity » dans « Matrix ».

La phase de conception du jeu a également nécessité la recherche d'un nom évocateur. Après un début animé et la proposition de nombreuses idées, le nom « MétaTravers » a été retenu. Cette appelation incarne parfaitement notre concept, symbolisant le voyage à travers les différentes dimensions numériques pour atteindre nos objectifs.

2.2 Répartition des tâches

Concernant la répartition des tâches, nous avons opté pour une approche méthodique et équilibrée. À cette fin, nous avons utilisé un diagramme de Gantt pour planifier et attribuer les différentes responsabilités en fonction des compétences et des disponibilités de chaque membre de l'équipe.
Parrallèlement, l'utilisation de GitHub comme plateforme de collaboration a grandement facilité l'échange des données et la gestion du travail en permettant à chacun de contribuer de manière transparente et organisée.
Malgré quelques écarts par rapport au planning initial, notre approche flexible, basée sur des réunions régulières et une communication ouverte, nous a permis de maintenir un niveau élevé d'efficacité et de cohésion au sein de l'équipe.

Dans un souci d'équité et d'efficacité, chaque membre s'est vu attribuer des tâches en fonction de ses compétences, de ses envies, et de la dureté des tâches. Cette approche a permis de garantir que chacun puisse contribuer de manière significative au projet, tout en assurant une répartition équilibrée de la charge de travail. De plus, la mise en place d'un système de suivi des tâches et de leur avancement a permis à chacun de rester informé et engagé dans le processus, renforçant ainsi notre collaboration et notre cohésion en tant qu'équipe.

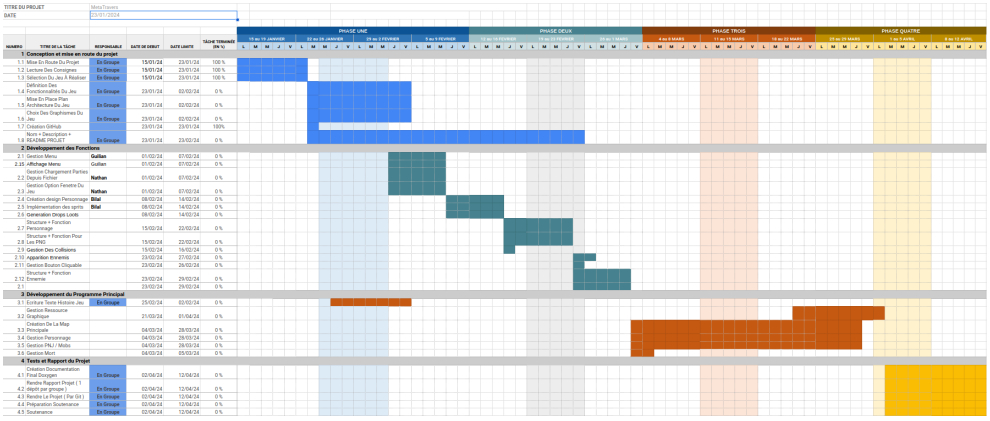


FIGURE 1 – Diagramme de Gantt

3 Analyse et conception du projet

Dans cette seconde partie, nous allons présenter ce que nous avons fait au début du projet juste après avoir décidé du style de jeu que nous allions faire. Nous avons analysé le problème que l'on devait traiter, c'est à dire réfléchir aux différentes règles du jeu et aux fonctionnalités disponibles pour le joueur. Par la suite, nous avons réfléchi à la conception des différents onglets présents dans le jeu. On peut notamment y retrouver le menu principal, les options, etc.

3.1 Analyse du projet

Nous avons donc décidé de créer un projet dans le même style que les jeux « Mario Bros » qui possèdent des règles de jeu relativement simples. Le but de notre jeu est d'arriver à la fin des quatre niveaux différents en essayant de récupérer un maximum d'objets et de mourir le moins de fois possibles. Chaque niveau possède trois objets différents, parfois très bien cachés ou très difficiles à récupérer ce qui pousse le joueur à bien explorer tous les recoins du jeu et à prendre des risques.

De plus, notre jeu possède différentes fonctionnalités pour que tous les joueurs puissent jouer dans les meilleures conditions possibles et avoir un confort de jeu optimal. Notre projet possède une navigation simple qui permet aux utilisateurs même débutants de ne pas se perdre dans les menus ou dans le jeu. Notre jeu est composé de différentes pages qui représentent des onglets différents. On peut y retrouver les onglets du menu principal, de la création d'une nouvelle partie, des options, de la carte où l'utilisateur va pouvoir jouer au niveau qu'il souhaite et bien sûr les onglets des quatre niveaux, sans compter l'introduction lorsque l'on commence une nouvelle partie.

3.2 Conception du projet

Nous avons donc divisé notre jeu en plusieurs onglets qui possèdent tous différentes fonctionnalités pour que les joueurs puissent y naviguer facilement.

— Le menu principal :

Cet onglet est le tout premier qui s'affiche quand l'utilisateur lance le jeu, il est composé d'un fond d'écran qui représente le thème

de notre projet, son titre et de plusieurs boutons cliquables. Ces derniers permettent de continuer la partie s'il en existe déjà une, de créer une nouvelle partie si le joueur veut recommencer le jeu ou bien d'accéder aux différentes options mises à disposition.

FIGURE 2 – Menu principal du jeu

— La nouvelle partie :

En cliquant sur le bouton « Nouvelle partie », le joueur accède à une nouvelle page où il pourra recommencer le jeu du début. Il aura la possibilité de choisir différents paramètres comme son pseudonyme, son personnage entre un homme et une femme et le mode de jeu. Il existe le mode normal qui est accessible par tous les joueurs quel que soit leur niveau et un mode difficile qui est réservé aux personnes qui connaissent bien le jeu et qui cherchent un défi supplémentaire.

FIGURE 3 – Création d'une nouvelle partie

— Les options :

En cliquant sur le bouton « Options », le joueur accède à une nouvelle page où il pourra changer différents paramètres contenus dans deux onglets différents. On peut y retrouver l'onglet « Son » où la modification du son de la musique et celle des effets sonores est possible. De plus, on peut accéder à l'onglet « Touches » où le joueur peut modifier les touches pour aller à droite, pour aller à gauche, pour sauter ou monter, pour descendre et celle pour intéragir.

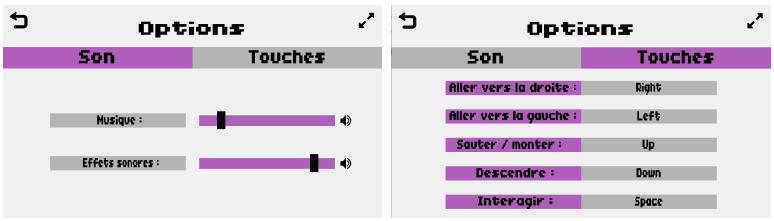


FIGURE 4 – Options du jeu

— La carte :

Puis en cliquant sur le bouton « Continuer », le joueur accède à l'onglet central du jeu où il peut se déplacer sur la carte du monde pour choisir le niveau auquel il veut jouer. Il est possible d'observer l'image de fond qui représente la carte et qui possède quatre niveaux. Au début d'une partie, le joueur est au premier niveau mais n'a pas

accès aux niveaux suivants. Ils se débloquent au fur et à mesure que l'utilisateur termine les niveaux.

FIGURE 5 – Carte du jeu

La navigation dans notre projet est toujours la même et permet de d'accéder directement à la plupart des onglets rapidement et sans difficulté.

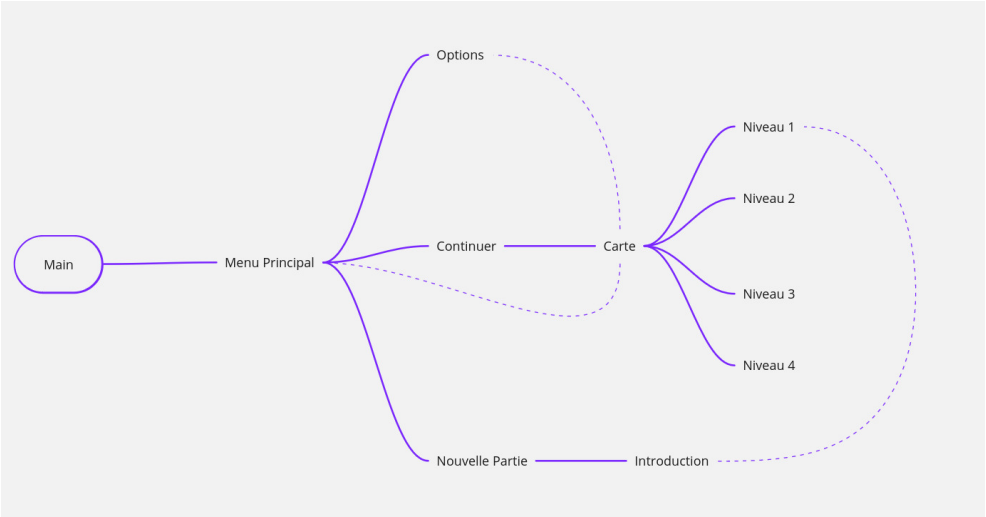


FIGURE 6 – Schéma de la navigation dans le jeu

Pour plus d'informations, voir le schéma détaillé dans l'annexe : 6.1

De plus, les quatre niveaux auxquels le joueur peut jouer demandent des mécaniques souvent différentes pour pas que l'utilisateur ne perde l'envie

de jouer au bout d'un certain temps. Dans tous les niveaux et en particulier dans le premier et le dernier, le joueur doit utiliser les touches pour aller vers la droite, vers la gauche et pour sauter afin de pouvoir déplacer le personnage. Cependant, dans les mini-jeux des deuxième et troisième niveaux, ces touches ne correspondent pas exactement au déplacement du personnage comme on pouvait le voir jusqu'à présent. Dans le premier mini-jeu du second niveau, l'objectif et de relier des tuyaux entre eux pour faire un chemin. Le joueur utilisera les touches principales pour se déplacer de tuyau en tuyau et la touche d'interaction pour tourner ces derniers. Nous pouvons retrouver cette mécanique dans le second mini-jeu du troisième niveau, qui va juste demander en plus au joueur de rester appuyé sur la touche d'interaction, pour pouvoir bouger un bloc du départ à l'arrivée et tout ceci dans un labyrinthe. Pour finir, dans le premier mini-jeu du troisième niveau, le joueur reconstitura un composant qui est divisé en plusieurs parties. Il devra seulement utiliser la souris pour y parvenir.

4 Outils et méthodes utilisés pour le codage

Dans cette partie, nous allons traiter l'onglet centrale de notre projet, qui regroupe les outils utilisés et l'écriture du code.

4.1 Les différents outils exploités

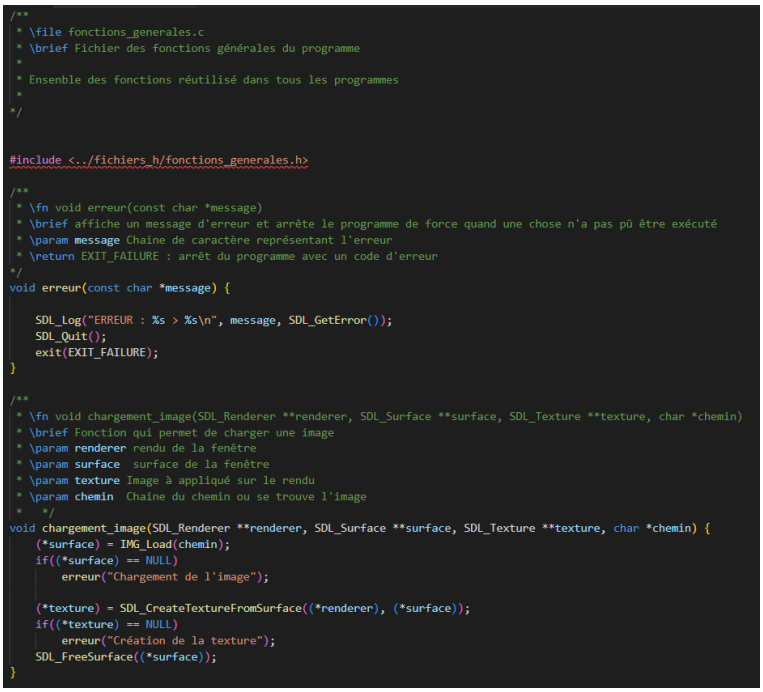
Avant de commencer le codage, nous avons pris connaissance de différents outils que nous allons vous présenter tout en détaillant leur fonctionnement :

— Visual Studio Code 1.81.1
— GitHub 2.25.1
— GCC 9.4
— Valgrind 3.15
— Débogueur GDB 9.2
— LaTeX 3.14
— Google Drive

4.1.1 Visual Studio Code

Visual Studio Code est l'outil de développement principal que nous avons utilisé, notamment pour ses diverses extensions et son intégrité qui nous a permis de mieux exploiter certains outils supplémentaires :

— Doxygen : mettre en forme les commentaires sous forme de Doxygen



```c
/**
 * \file fonctions_generales.c
 * \brief Fichier des fonctions générales du programme
 *
 * Ensemble des fonctions réutilisé dans tous les programmes
 *
*/


#include <../fichiers_h/fonctions_generales.h>

/**
 * \fn void erreur(const char *message)
 * \brief affiche un message d'erreur et arrête le programme de force quand une chose n'a pas pû être exécuté
 * \param message Chaine de caractère représentant l'erreur
 * \return EXIT_FAILURE : arrêt du programme avec un code d'erreur
*/
void erreur(const char *message) {

    SDL_Log("ERREUR : %s > %s\n", message, SDL_GetError());
    SDL_Quit();
    exit(EXIT_FAILURE);
}

/**
 * \fn void chargement_image(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, char *chemin)
 * \brief Fonction qui permet de charger une image
 * \param renderer rendu de la fenêtre
 * \param surface  surface de la fenêtre
 * \param texture Image à appliqué sur le rendu
 * \param chemin  Chaine du chemin ou se trouve l'image
 *   */
void chargement_image(SDL_Renderer **renderer, SDL_Surface **surface, SDL_Texture **texture, char *chemin) {
    (*surface) = IMG_Load(chemin);
    if((*surface) == NULL)
        erreur("Chargement de l'image");

    (*texture) = SDL_CreateTextureFromSurface((*renderer), (*surface));
    if((*texture) == NULL)
        erreur("Création de la texture");
    SDL_FreeSurface((*surface));
}
```

FIGURE 7 – Exemple du Doxygen dans un programme

Pour plus d'informations, voir le Doxygen dans l'annexe : 6.2.

— GitHub Pull Request : permet d'appliquer les commandes GitHub à partir de Visual Studio Code de manière rapide

4.1.2 GitHub

GitHub est le premier outil que nous avons appris à manipuler et à utiliser pour mettre notre code en commun. Cet outil fournit beaucoup de possibilités d'organisation, tel que la création de différentes branches et la faculté de fusion. Des complications dûes à notre manque de connaissance sur cet outil et différents problèmes rencontrés nous ont fait perdre du temps. Nous avons donc décidé de limiter l'usage de GitHub pour tirer et pousser nos travaux dans la branche principal, laissant une sauvegarde de côté en cas de soucis sur le code principal.

4.1.3 GCC

GCC a été choisi comme compilateur dû à sa capacité à créer des versions « .o » de nos programmes, permettant une compilation plus rapide des modules.

4.1.4 Valgrind

L'usage de Valgrind nous a permis la détection des différentes fuites de mémoire et du bon usage des pointeurs (hors usage de la bibliothèque SDL qui en produit naturellement). Cet outil nous a fourni une bonne détection des erreurs d'usage des pointeurs, notamment les erreurs de segmentation et des fuites de mémoire liées à l'oubli de la commande « free » après une allocation dynamique.

4.1.5 GDB

GDB est un outil de débogage que nous avons utilisé pour identifier des erreurs lors de nos tests sur plusieurs entités, surtout liées à l'utilisation de la bibliothèque SDL. Vous pouvez voir ci-dessous une utilisation de cet outil permettant de comprendre où se situait l'erreur de segmentation lors de l'appel d'une fonction.

FIGURE 8 – Exemple de débogage

Dans le cas que l'on peut observer au-dessus, nous avons obtenu une erreur de segmentation lorsque l'on voulait mettre à jour les textures des tuiles pour la *tilemap*. En utilisant l'outil GDB, nous avons pû repérer facilement l'erreur et avons découvert qu'il manquait un *return* à la fin de la fonction.

4.1.6 Google Drive

Nous avons utilisé cet outil pour la mise en commun de nos idées, pour l'utilisation du diagramme de Gantt et pour le cahier des charges.

4.2 La méthode *tilemap*

À travers les nombreuses mécaniques employées dans notre code, l'utilisation du *tilemap* est l'une des parties les plus cruciales. La méthode *tilemap*, appelée aussi « la carte par tuile » est une méthode largement utilisée dans le domaine du jeu vidéo et dans la simulation graphique grâce à ses avantages sur la mémoire (petit coût par rapport au chargement d'image), mais aussi sur son utilisation (facilement utilisable et exploitable). Chaque *tilemap* utilise une matrice qui est remplie de valeurs numériques et dont la taille varie (en fonction de son utilité). Ces valeurs possèdent chacune une texture unique qui leur est assignée. Lors de l'affichage de la partie graphique, les données de chaque tuile ne sont pas sauvegardées mais actualisées en fonction de leur position dans la matrice. La *tilemap* est le centre d'interaction entre les informations reçues par le joueur dans les niveaux et le bon déroulement du programme.

Malgré les avantages de la méthode *tilemap*, nous pouvons constater que cette technique dispose de quelques limites. La *tilemap*, ne pouvant pas laisser place à un système de génération automatique, est créée manuellement nécessitant une rigueur dans la construction de la matrice. Cette dernière ne procure qu'un seul moyen pour afficher les éléments au travers de la SDL et en aucun cas ne peut s'occuper d'un autre système, tel que celui de la collision ou de la reconnaissance de texture.

4.3 Les différentes structures et fonctions générales

Pour lier nos travaux et avoir une bonne organisation du code, nous avons utilisé diverses structures et énumérations. Ainsi, nous utilisons les mêmes noms de ces dernières pour avoir une meilleure fusion des modules du programme. Voici un échantillon des différentes structures et énumérations que nous avons utilisées :

— barreDeSon : gère l'aspect du son
— page_t : permet de savoir sur quelle page on se trouve
— position_t : donne la position du personnage sur la carte
— direction_t : donne la direction du personnage sur la carte

Nous avons aussi implémenté des fonctions communes tel que :

— erreur : fonction qui arrête le programme en cas d'erreur et retourne un message correspondant

— chargement_image : charge une image dans une texture

4.4 La gestion des niveaux et des mini-jeux

Les niveaux sont stockés dans des structures, qui sont elles-mêmes représentées par des pointeurs et des variables, servant à la représentation des différents éléments du niveau.

Parmi les obstacles notoires que nous avons rencontré dans le développement de nos mini-jeux, l'implémentation de ces derniers dans notre programme principal et l'empêchement des conflits de comportement ont été les plus difficiles et les plus longs à résoudre. Nous pouvons prendre l'exemple que lorsque le mini-jeu était isolé à son propre programme, celui-ci se comportait correctement tandis que lors de son implémentation au programme principal, il présentait des comportements anormaux (mauvais déplacement, erreurs de segmentation, bugs, etc). Malgré les outils mis à notre disposition (Valgrind, GDB), certains problèmes étaient difficilement détectables par ces derniers. Cela était majoritairement dû à des erreurs de signe ou d'assignation de variables subtiles. Nous pouvons prendre l'exemple du mini-jeu de la tuyauterie, lors de la vérification du trajet où il y avait des erreurs sur les modifications de coordonnées. Par conséquent, la complexité de certains niveaux a été remise en question soit à cause d'un nombre de calcul trop grand, tel que la tuyauterie qui faisait trop d'appels récursifs pour tester toutes les combinaisons possibles, soit à cause de certains paternes qui faisaient des boucles infinies pendant la vérification du chemin.

4.5 Les tests unitaires

L'objectif des tests unitaires est de vérifier l'intégralité et le bon déroulement d'une fonction ou d'une structure de donnée sans être biaisée par d'autres (c'est à dire prendre en compte les erreurs potentielles induites par des fonctions extérieures à une autre ou à une structure de donnée initiale).

4.5.1 Test unitaire du pseudo

Le test unitaire le plus important qui nous a permis de trouver le plus grand nombre d'erreurs dans notre code a été celui de la saisie du pseudo dans l'onglet « Nouvelle Partie ». Il possède un grand nombre de conditions qui permet au pseudo d'être valide.
Pour plus d'informations, voir le test unitaire test_pseudo.c sur notre GitHub et le tableau des différents cas possibles dans l'annexe : 6.3.

4.5.2 Création des mini-jeux

Afin de tester l'agencement de nos mini-jeux, nous avons créé un algorithme permettant d'avoir un suivi sur leurs comportements. La création de ces derniers à partir d'un fichier externe ont permis de tester différents paternes de jeu afin de prévoir des comportements qui pourraient poser plusieurs problèmes et la destruction de ces derniers (mise à « NULL »). En parallèle, nous avons fait appel à des personnes extérieures au projet, avec aucune connaissance du jeu et des règles, afin de tester les limites de nos mini-jeux. Ceci nous a permis de détecter plusieurs erreurs graphiques et pratiques. Par la même occasion, cela nous a permis de s'assurer qu'aucune fuite de mémoire était présente.

5 Résultat et conclusion

Dans cette conclusion, nous présentons le résultat obtenu ainsi qu'un bilan de notre expérience dans le développement de notre jeu.

5.1 Résultat du projet

Notre jeu, fruit de nombreuses heures de travail et de dévouement, fonctionne avec succès. Nous avons réussi à créer un environnement de jeu immersif et engageant, offrant aux joueurs une expérience divertissante et captivante. Cependant, malgré notre satisfaction initiale, nous avons identifié plusieurs aspects à améliorer pour enrichir davantage l'expérience des utilisateurs. Parmi ces améliorations figurent l'ajout de nouveaux niveaux pour étendre la durée de jeu, la possibilité de jouer à la manette pour offrir une alternative de contrôle plus intuitive, ainsi que l'introduction d'un système de vie pour augmenter le défi et la tension du jeu.

En outre, notre planification initiale a été perturbée par nos obligations personnelles et scolaires, notamment nos cours et nos partiels. Le manque de temps disponible a entravé notre capacité à respecter pleinement notre emploi du temps, retardant ainsi certaines tâches et fonctionnalités prévues. De plus, la découverte et la prise en main de la bibliothèque SDL ont pris plus de temps que prévu, retardant ainsi le développement global du jeu. Enfin, la répartition des rôles s'est faite au fur et à mesure de nos réunions, ce qui a parfois entraîné des retards et des conflits de vision.

5.2 Bilan du projet

Malgré les problèmes rencontrés, cette expérience de développement de jeu a été incroyablement enrichissante. Nous avons acquis une compréhension plus approfondie du processus de développement de jeux vidéo, ainsi qu'une appréciation du travail d'équipe et de la résolution de problèmes. Nous aimerions continuer cette aventure et améliorer notre jeu avec une base solide dans le langage de programmation, ainsi qu'une planification plus réaliste et une gestion plus efficace du temps. Nous sommes reconnaissants pour cette opportunité et impatients de voir où nous mènera cette expérience dans le futur.

6 Annexe

6.1 Schéma détaillé de la navigation

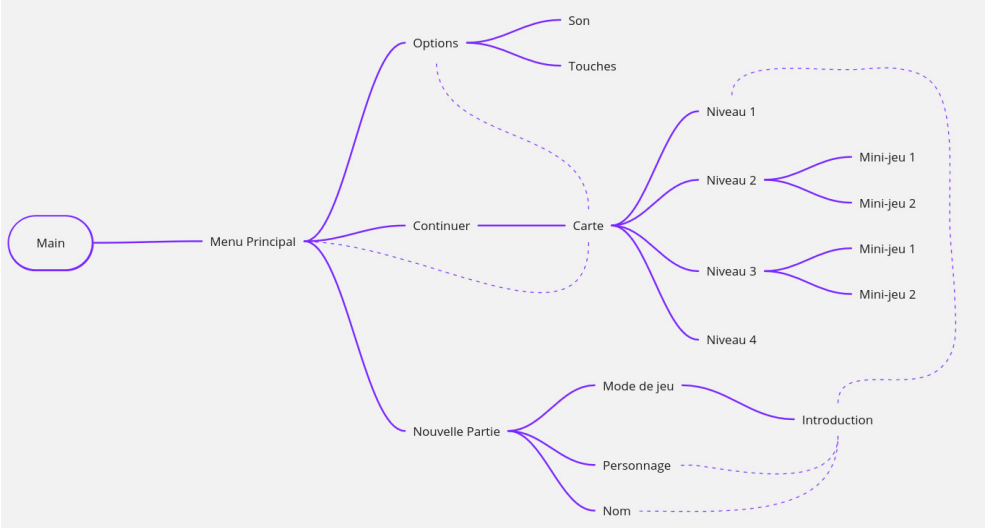
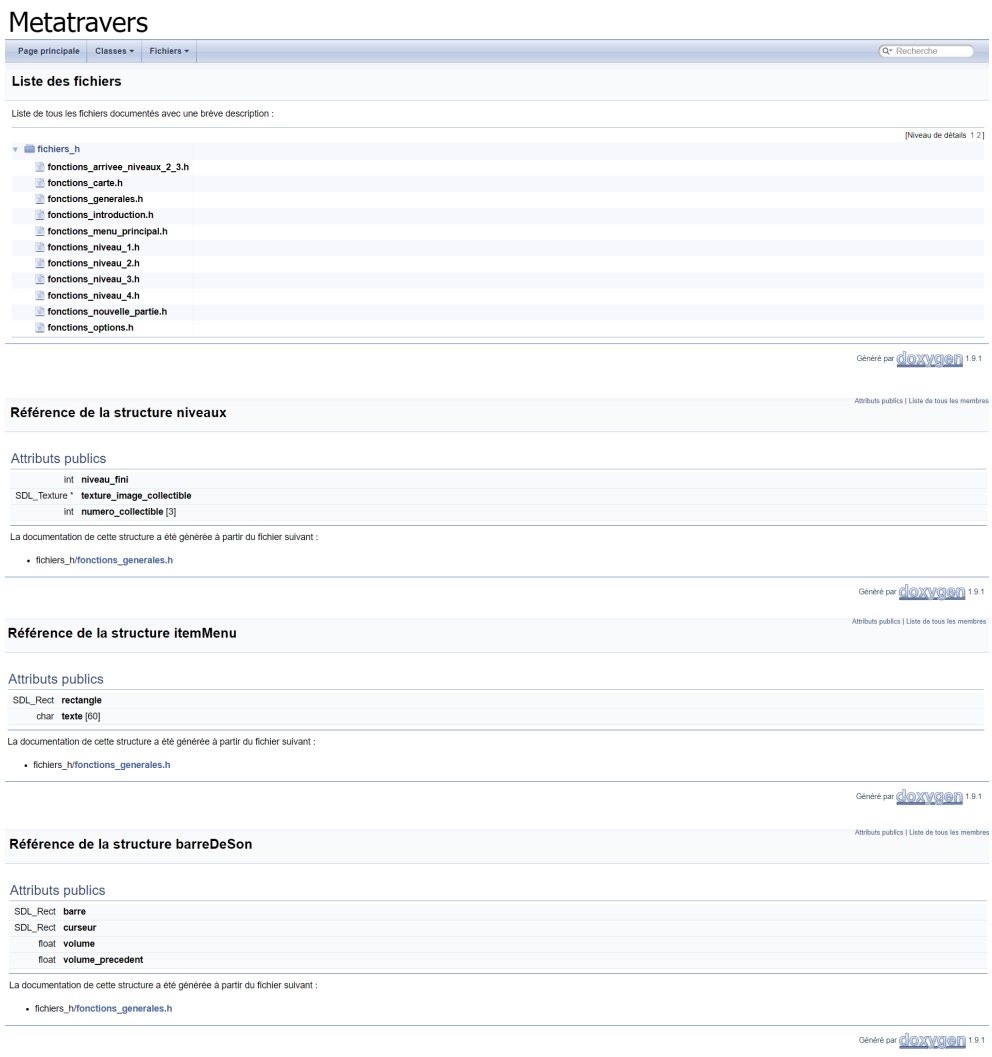


FIGURE 9 – Schéma détaillé de la navigation dans le jeu

6.2 Doxygen



Metatravers

Page principale | Classes | Fichiers | Recherche

Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

[Niveau de détails 1 2]

- fichiers_h
  - fonctions_arrivee_niveaux_2_3.h
  - fonctions_carte.h
  - fonctions_generales.h
  - fonctions_introduction.h
  - fonctions_menu_principal.h
  - fonctions_niveau_1.h
  - fonctions_niveau_2.h
  - fonctions_niveau_3.h
  - fonctions_niveau_4.h
  - fonctions_nouvelle_partie.h
  - fonctions_options.h

Généré par doxygen 1.9.1

Attributs publics | Liste de tous les membres

Référence de la structure niveaux

Attributs publics

int	**niveau_fini**
SDL_Texture *	**texture_image_collectible**
int	**numero_collectible** [3]

La documentation de cette structure a été générée à partir du fichier suivant :

- fichiers_h/fonctions_generales.h

Généré par doxygen 1.9.1

Attributs publics | Liste de tous les membres

Référence de la structure itemMenu

Attributs publics

SDL_Rect	**rectangle**
char	**texte** [60]

La documentation de cette structure a été générée à partir du fichier suivant :

- fichiers_h/fonctions_generales.h

Généré par doxygen 1.9.1

Attributs publics | Liste de tous les membres

Référence de la structure barreDeSon

Attributs publics

SDL_Rect	**barre**
SDL_Rect	**curseur**
float	**volume**
float	**volume_precedent**

La documentation de cette structure a été générée à partir du fichier suivant :

- fichiers_h/fonctions_generales.h

Généré par doxygen 1.9.1

FIGURE 10 – Doxygen

6.3 Test unitaire du pseudo

Cas valides	Données	Résultats obtenus
Uniquement des minuscules et au plus dix caractères	Pseudo = « test »	Données valides
Uniquement des majuscules et au plus dix caractères	Pseudo = « TEST »	Données valides
Uniquement des caractères spéciaux et au plus dix caractères	Pseudo = « &é »	Données valides
Minuscules, majuscules et au plus dix caractères	Pseudo = « Test »	Données valides
Minuscules, caractères spéciaux et au plus dix caractères	Pseudo = « test&é »	Données valides
Minuscules, espaces et au plus dix caractères	Pseudo = « te st »	Données valides
Majuscules, caractères spéciaux et au plus dix caractères	Pseudo = « TEST&é »	Données valides
Majuscules et espaces et au plus dix caractères	Pseudo = « TE ST »	Données valides
Caractères spéciaux, espaces et au plus dix caractères	Pseudo = « & é »	Données valides
Minuscules, majuscules, caractères spéciaux et au plus dix caractères	Pseudo = « Test&é »	Données valides
Minuscules, majuscules et espaces et au plus dix caractères	Pseudo = « Te st »	Données valides
Minuscules, caractères spéciaux, espaces et au plus dix caractères	Pseudo = « tes t&é »	Données valides
Majuscules, caractères spéciaux, espaces et au plus dix caractères	Pseudo = « TES T&é »	Données valides
Minuscules, majuscules, caractères spéciaux, espaces et au plus dix caractères	Pseudo = « Tes t&é »	Données valides

Cas invalides	Données	Résultats obtenus
Aucun caractère	Pseudo = « »	Données invalides
Uniquement des espaces et au plus dix caractères	Pseudo = « »	Données invalides
Uniquement des espaces et plus de dix caractères	Pseudo = « »	Données invalides
Plus de dix caractères	Pseudo = « TestPlusDixCara »	Données invalides