Le Mans Université

Institut d'Informatique Claude Chappe
Le Mans Université

Le Mans Université

License Informatique deuxieme année
Rapport de projet
Fefeu Clément|Dubois Gabriel|Renou Alexis
Diffenders

8 Avril 2024

Table des matières

Table des figures

1 Introduction

Dans le cadre de notre deuxième année de licence en informatique à l'Université du Mans, dans le module de projet, il nous est proposé de créer un jeu vidéo en utilisant la SDL2 en 3 mois, en mettant à profit nos connaissances dans le langage C tout en apprenant à utiliser la SDL2. Nous avons choisi de concevoir un jeu de type tower defense que nous avons nommé « Diffenders ».

Notre objectif en développant ce jeu est de garantir que les ennemis suivent un chemin prédéfini, tandis que les tours, que nous appelons héros dans notre jeu, peuvent être placées partout sur la carte, à l'exception du chemin et de la base à défendre. Nous avons également opté pour un système de défense basé sur le nombre de vagues, où il faut défendre un certain nombre de vagues d'ennemis pour remporter la partie. La carte sera générée aléatoirement pour offrir une certaine rejouabilité et sera composée de trois types de cases : chemin, base et plaine. La plaine représente l'endroit où les héros peuvent être placés.

L'inspiration de ce projet découle de notre intérêt pour ce genre particulier et unique, qui a souvent émergé comme un dérivé d'un jeu existant à ses débuts, en utilisant ses ressources pour le transformer en ce genre. Avant même

FIGURE 1 – Orc Must Die 3 : Le tower defense favori de deux membres du groupe

de présenter le travail réalisé pour chaque partie du projet, il est essentiel de définir les termes et de présenter les parties elles-mêmes.

2 Definition

Le terme « tower defense » vient de l'anglais et signifie littéralement « défense avec la tour ». Bien que la traduction littérale ne soit pas la plus précise, elle correspond mieux à la nature même du genre de jeux. Un tower defense est un type de jeu vidéo où l'objectif principal est de défendre un point, une base, un personnage ou un objectif. Selon les jeux, cela peut nécessiter de défendre un certain nombre de tours ou de vagues, de résister pendant un certain temps, d'accomplir des objectifs secondaires ou de tenir le plus longtemps possible sans chance de réussite. Dans la plupart des cas, les ennemis se déplacent le long d'un chemin prédéfini, bien que certains jeux permettent au joueur de personnaliser le chemin.

Pour réussir à protéger le point, les joueurs peuvent mettre en place des tours, chacune ayant ses propres spécialités : ralentir les ennemis, leur infliger des dommages, leur appliquer des « statuts » tels que le poison qui leur fait subir des dégâts au fil du temps, etc. Ainsi, le jeu demande au joueur de réfléchir à la meilleure stratégie de défense à adopter grâce aux tours, d'où le nom « tower defense ».

Les ennemis dans un tower defense sont souvent très variés. Par exemple, il y a généralement un ennemi de base qui avance vers la tour à une vitesse moyenne, sans effet particulier. Tous les autres ennemis du jeu sont des variations, légères ou importantes, de cet ennemi de base : plus de points de vie (déterminant le nombre de dommages nécessaires pour tuer l'ennemi), plus de vitesse, capacité à soigner les autres ennemis, etc. Dans les jeux en 3D, les améliorations peuvent même inclure la capacité de voler, incitant ainsi le joueur à diversifier ses tactiques. Éliminer les ennemis rapporte de l'argent, qui peut être utilisé pour construire de nouvelles tours ou améliorer celles qui existent déjà.

Une « vague d'ennemis » est un terme utilisé pour décrire un groupe d'ennemis. Lorsque tous les ennemis d'une vague sont éliminés, cela déclenche l'arrivée de la vague suivante. Les vagues sont progressives en difficulté, allant parfois jusqu'à inclure un boss, un ennemi ayant beaucoup plus de points de vie que les autres et souvent doté d'un effet spécial qui peut gêner le joueur ou renforcer les autres ennemis.

FIGURE 2 – Rampart l'un des premiers tower defense

FIGURE 3 – Bloons : le tower defense le plus joué actuellement

Voici deux exemples de tower défense.

3 Règles et fonctionnalités

Lors de la conception de notre jeu, il a fallu définir des règles et des fonctionnalités que nous voulons avoir dans le jeu.

Avant toute chose, notons qu'une fonction de sauvegarde n'est pas très utile dans un tower defense étant donné que reprendre une partie qu'on a déjà commencée (dans ce type de jeu) n'est pas le but ; quand on perd, on recommence depuis le début.

Les fonctionnalités du jeu sont donc :

- Un menu offrant la possibilité d'accéder aux options, de quitter ou de lancer une partie.
- Les options incluent la possibilité de passer en plein écran, de revenir au menu ou de revenir en mode fenêtré.
- La création d'une carte aléatoire générée à chaque nouvelle partie, où les ennemis, quels qu'ils soient, ne doivent pas dépasser du chemin prédéfini.
- Les ennemis se génèrent aux points d'entrée et se dirigent vers la base en suivant le chemin.
- Les héros infligent des dégâts aux ennemis.
- Une barre de sélection permettant de choisir quel héros poser ou de détruire un héros.
- Un système d'argent où chaque monstre tué rapporte de l'argent.
- Les ennemis deviennent plus nombreux ou plus forts à chaque vague

Celles-ci sont les fonctionnalités de base nécessaires pour que le jeu soit considéré comme opérationnel.

Les règles du jeu sont :

- Le joueur doit survivre à un nombre de vagues pour gagner, ici quinze.
- Pour cela, le joueur peut poser des tours qui tueront les ennemis.
- Lorsque tous les ennemis d'une vague sont tués, une nouvelle vague d'ennemis commence à arriver.
- Les tours coûtent de l'argent lorsqu'elles sont posées, chaque fin de vague rapporte de l'argent.
- Le joueur est limité à un certain nombre de tours, ici 10, principalement pour assurer le bon fonctionnement du jeu.
- Lorsqu'un ennemi touche la base, celle-ci perd des points de vie.
- Lorsque la base atteint zéro point de vie, le joueur perd.

4 Développement

4.1 Menus de démarrage

Lors du démarrage du jeu, la fonction de menu est la première à s'appliquer. Son objectif est d'offrir une interface de menu au joueur. Ce menu se compose d'un bouton permettant de lancer une partie, d'un bouton permettant d'accéder aux paramètres et d'un dernier bouton permettant de quitter le jeu. Tous les boutons fonctionnent de la même manière : leurs images sont affichées à l'écran, et lorsque le joueur place le curseur sur l'emplacement du bouton, celui-ci s'assombrit pour indiquer qu'un clic entraînera une action.

Parlons maintenant plus précisément des boutons disponibles. Le premier d'entre eux est le bouton « Jouer ». Comme son nom l'indique, ce bouton permet de lancer le jeu, qui démarre donc normalement depuis le début. En parlant de démarrage depuis le début, notons que l'utilisation de la sauvegarde de la progression n'est absolument pas adaptée à ce type de jeu, qui se prête très bien à lancer une partie, la compléter, puis recommencer depuis le début avec une nouvelle génération. C'est la raison pour laquelle l'option « Sauvegarde » n'est pas présente dans le menu ou dans les paramètres.

En parlant des paramètres, il s'agit du prochain bouton mis en place sur l'écran de menus. Celui-ci permet, après un clic, d'accéder à des options concernant la fenêtre de jeu, notamment les options « Plein écran », qui modifie la taille de la fenêtre de jeu pour la configurer en plein écran, et « Fenêtré », qui est l'action inverse permettant de configurer la fenêtre en mode fenêtré. Les deux dernières options présentes sont « Retour » et « Quitter », permettant de revenir au menu de démarrage et de quitter le jeu.

Le dernier bouton disponible dans le menu est le bouton « Quitter », qui permet de fermer le jeu et la fenêtre du menu.

4.2 Phases de jeu

Le fichier jeu.c contient toutes les fonctions et le code nécessaires au bon déroulement d'une partie de tower defense. C'est ici que sont utilisées les fonctions de création de chemins aléatoires, avec la répartition des points d'apparition des adversaires et le placement de la base à défendre. Mais avant d'en arriver là, une étape cruciale du jeu doit être franchie : l'affichage du terrain. C'est à ce stade que les images nécessaires aux différents éléments, tels que le terrain, les tours de défense représentées ici par des « héros », ainsi que les ennemis, sont chargées en mémoire. Une fois ces images chargées, la deuxième mission du jeu consiste à permettre le dépôt de tours sur l'écran avec un simple clic. Les images sont donc placées par-dessus le fond d'écran déjà affiché.

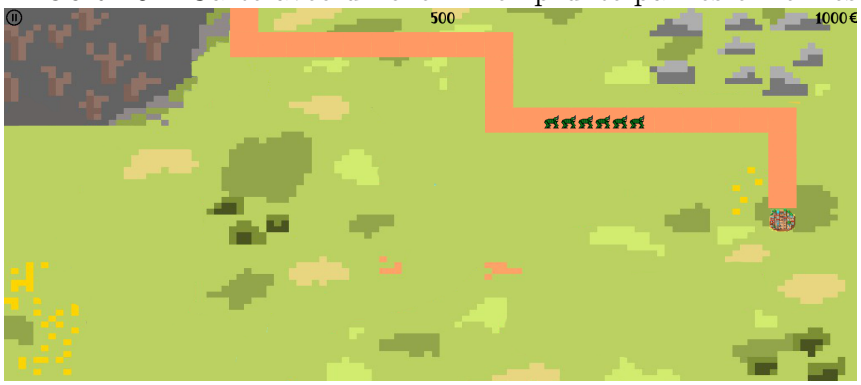
FIGURE 4 – Tour de défense imagé par des héros

FIGURE 5 – Carte brute

Ensuite, c'est au tour des chemins et des points importants, qui comprennent la base à défendre, le point d'apparition des ennemis et les chemins générés aléatoirement, qui sont donc affichés à l'écran. Ensuite, c'est au tour des ennemis, qui viennent s'afficher par-dessus les chemins, les points d'apparition et la base. Les ennemis se déplacent en suivant la position des chemins. Chaque case du chemin possède un emplacement dans une matrice cachée qui permet de calculer ses coordonnées en pixels à l'écran. C'est avec les coordonnées calculées des cases des chemins que les ennemis avancent. Ils se dirigent en suivant les coordonnées menant aux prochains points importants du chemin (points d'apparition, les virages, la base). C'est ainsi que les ennemis avancent précisément dans la bonne direction menant à la base. Par ailleurs, les ennemis ont été conçus de sorte qu'ils ne puissent normalement

pas se bloquer, se doubler ou se superposer.

FIGURE 6 – Carte avec un chemin emprunté par les ennemies



Mais que se passe-t-il si les ennemis atteignent la base ? Eh bien, ils disparaissent. Les ennemis touchant la base dans un ordre précis (j'entends ici que le premier est toujours le premier tout le long du chemin). Il suffit de dire que le premier prend comme valeur le deuxième et ainsi de suite jusqu'au dernier qui est finalement supprimé. C'est donc comme si le premier de la file disparaissait, ce qui est la situation optimale.

Pour finir, il y a une contrainte de coût concernant les tours de défense. Celles-ci ne doivent pas être posées sans que nous n'ayons en stock un nombre de points au moins égal à leurs coûts. De ce fait, les ennemis qui sont détruits sur la route ainsi que les tours de défense qui sont supprimées offrent une valeur monétaire. Il y a donc la possibilité de supprimer des tours, ce qui rend une partie de la monnaie dépensée pour les poser au joueur.

FIGURE 7 – Prix des tours de défenses

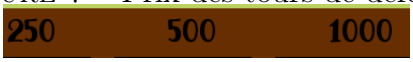
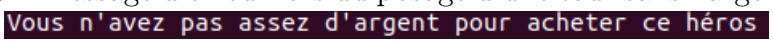


FIGURE 8 – Argent stocké

FIGURE 9 – Message d'erreur lors du posage d'une tour sans l'argent adéquat



Pour finir, la partie se termine une fois que quinze vagues d'ennemis ont été vaincues ou si les points de vie de la base descendent en dessous de zéro.

4.3 Génération de la carte

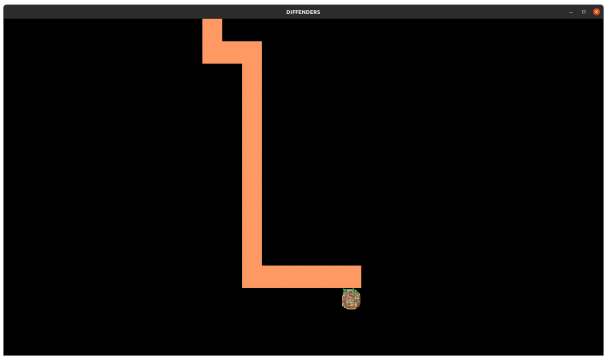
La génération de la carte est pseudo-aléatoire en raison de sa dépendance à la génération basée sur le temps. Elle peut se décomposer en plusieurs parties.

1. La base à défendre est d'abord placée aléatoirement parmi toutes les cases qui ne sont pas situées sur la bordure ;
2. Le point d'entrée ou les points d'entrée des ennemis sont créés à l'opposé de la base. Pour cela, nous divisons la carte en quatre parties.
   — en haut à gauche ;
   — en haut à droite ;
   — en bas à gauche ;
   — en bas à droite.

   Par exemple, si la base est créée en haut à droite, les points d'entrée seront en bas à gauche ;
3. De ces entrées, nous allons générer un point de rencontre où tous les chemins d'entrée se rejoindront. Ce point part du coin non-bord où sont générés les points d'entrée, et prend l'abscisse de l'entrée la plus proche du milieu ainsi que l'ordonnée de l'entrée la plus proche du milieu de la carte.
4. De ce point de rencontre, nous allons tracer le chemin vers la base, en ajoutant la possibilité de générer une bifurcation pour permettre d'explorer d'autres formes de chemins.

Tout en générant la carte, nous retenons toutes les coordonnées des chemins dans un tableau, chaque case suivie par le chemin étant enregistrée. Ensuite, ce tableau est transformé en un tableau ne contenant que les points importants : les points d'entrée, les points de rencontre de plusieurs parcours et la base. Ce tableau est ensuite renvoyé pour le déplacement des ennemis afin de permettre un suivi graphique précis du chemin.

FIGURE 10 – Exemples de génération des chemins



5 Gestion du projet

Il s'est avéré primordial lors de la mise en place du projet que la répartition des tâches se déroule rapidement et de manière optimale, dans le but que nous puissions débuter le projet sous les meilleurs auspices. Tout d'abord, nous avons eu la chance dans le cadre du projet d'utiliser des outils que nous n'avions pas l'habitude de manipuler, notamment un dépôt Git que nous avons mis en place pour les membres du groupe afin de pouvoir mettre à jour le code au fur et à mesure de son avancée.

5.1 L'Organisation des tâches

Une fois cela mis en place, il était grand temps d'organiser réellement le travail entre tous les membres du groupe. Lors des différentes discussions sur les orientations possibles du projet, nous avons rapidement conclu qu'il était nécessaire d'avoir au moins un menu de démarrage fonctionnel permettant de lancer le jeu. Ainsi, la tâche de concevoir le menu de démarrage et toutes ses options a été attribuée à Gabriel. Il est donc la personne qui s'est occupée des premiers aspects que l'utilisateur verra lors du lancement du jeu.

Bien entendu, il devait également y avoir une phase de jeu, qui vous a déjà été décrite dans le rapport de projet. Les ennemis apparaissent et avancent jusqu'à un objectif que nous, joueurs, défendons à l'aide de tours de défense. Eh bien, cette phase de jeu est fractionnée en deux parties dans l'organisation des tâches.

La première partie consiste à s'occuper des premières fonctions et structures nécessaires au bon fonctionnement du jeu. Ces fonctions et structures « primitives » sont utilisées initialement pour avoir un aperçu du fonction-

nement du jeu sans interface graphique. Il s'agit notamment de la création de héros, d'ennemis, du joueur, etc. Cette partie est gérée par Alexis, car même après des révisions des besoins et des améliorations des structures et des fonctions, leur utilisation demeure indispensable.

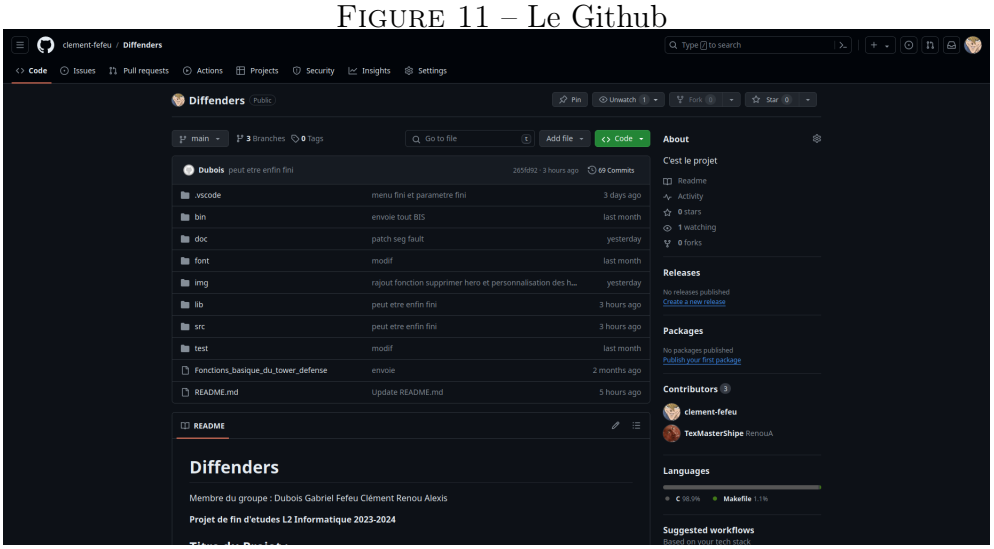
La seconde partie se concentre sur la création de la carte. Plusieurs idées ont émergé au sein du groupe. Devrions-nous créer une carte avec un chemin prédéfini ? Faire plusieurs cartes avec des chemins différents et un système de rotation de carte ? Ou encore créer une carte avec un chemin aléatoire ? C'est cette dernière option qui a été retenue ici. Cette partie, assez technique, a été proposée par Clément, qui a décidé de la mettre en place.

5.2 Les Outils

Il a également été réalisé un diagramme de Gantt dans un but prévisionnel, qui s'est avéré être d'une grande aide pour déterminer les rôles au sein du groupe et prévoir la durée de chaque tâche. Grâce au Gantt, nous avons pu constater, sur l'intervalle de temps donné par la date de remise du projet, prévue pour le 19 avril 2024, que nous disposions du 9 janvier pour commencer le projet, nous laissant trois mois et demi pour nous familiariser avec GitHub et la SDL, les utiliser pour créer notre projet, rédiger un compte-rendu, et préparer un exposé oral ainsi qu'un diaporama.

La prise en mains de Github pour le partage des programmes, images, sources et informations a été très vite prise en main, bien que peu utilisé. Cependant, nous avons fait le nécessaire pour l'utiliser lorsque cela est nécessaire.

FIGURE 11 – Le Github



La SDL (Simple DirectMedia Layer), étant la nouveauté qu'il fallait absolument apprendre à maîtriser pour avoir un projet fiable et cohérent, nous a donné plus de fil à retordre. Bien que nous ayons fini par comprendre comment l'utiliser, nous avons pris plus de temps que prévu, ce qui nous a fait prendre du retard que nous avons eu du mal à rattraper.

FIGURE 12 – Nous avons aussi utilisé le Gantt pour organiser notre travail.

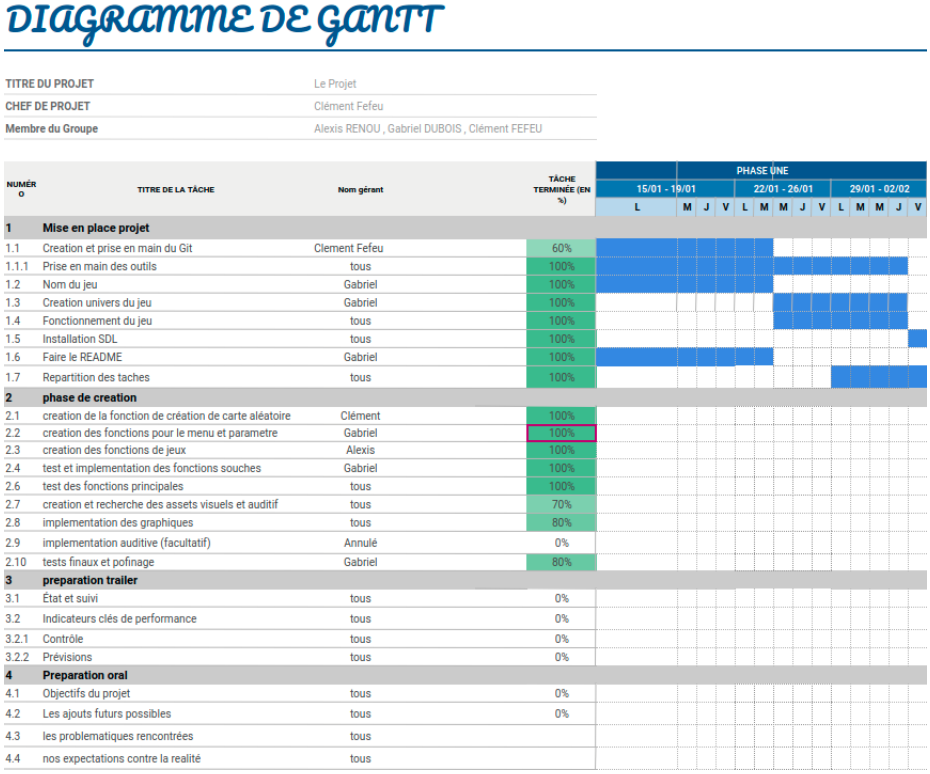


DIAGRAMME DE GANTT

TITRE DU PROJET	Le Projet
CHEF DE PROJET	Clément Fefeu
Membre du Groupe	Alexis RENOU , Gabriel DUBOIS , Clément FEFEU

NUMÉRO	TITRE DE LA TÂCHE	Nom gérant	TÂCHE TERMINÉE (EN %)
1	**Mise en place projet**		
1.1	Creation et prise en main du Git	Clement Fefeu	60%
1.1.1	Prise en main des outils	tous	100%
1.2	Nom du jeu	Gabriel	100%
1.3	Creation univers du jeu	Gabriel	100%
1.4	Fonctionnement du jeu	tous	100%
1.5	Installation SDL	tous	100%
1.6	Faire le README	Gabriel	100%
1.7	Repartition des taches	tous	100%
2	**phase de creation**		
2.1	creation de la fonction de création de carte aléatoire	Clément	100%
2.2	creation des fonctions pour le menu et parametre	Gabriel	100%
2.3	creation des fonctions de jeux	Alexis	100%
2.4	test et implementation des fonctions souches	Gabriel	100%
2.6	test des fonctions principales	tous	100%
2.7	creation et recherche des assets visuels et auditif	tous	70%
2.8	implementation des graphiques	tous	80%
2.9	implementation auditive (facultatif)	Annulé	0%
2.10	tests finaux et pofinage	Gabriel	80%
3	**preparation trailer**		
3.1	État et suivi	tous	0%
3.2	Indicateurs clés de performance	tous	0%
3.2.1	Contrôle	tous	0%
3.2.2	Prévisions	tous	0%
4	**Preparation oral**		
4.1	Objectifs du projet	tous	0%
4.2	Les ajouts futurs possibles	tous	0%
4.3	les problematiques rencontrées	tous	
4.4	nos expectations contre la realité	tous	

PHASE UNE

15/01 - 19/01					22/01 - 26/01					29/01 - 02/02				
L	M	J	V	L	M	M	J	V	L	M	M	J	V	

6 Conclusion

Il est maintenant temps de passer à la conclusion. Dans cette partie, nous détaillerons les résultats du projet qui a été mené, ainsi qu'un bilan de celui-ci. Ensuite, nous partagerons quelques lignes concernant l'avis des membres du groupe, avec un avis commun dans un premier temps, suivi d'un avis personnel de chacun.

6.1 Résultats

À la fin de ce projet, nous avons réussi à mettre en place la possibilité de lancer le jeu. Le lancement du jeu débouche directement sur un menu de démarrage qui permet de choisir entre lancer une partie, modifier les paramètres ou quitter le jeu. Les paramètres disponibles comprennent le passage en plein écran, le passage en mode fenêtré et la possibilité de quitter le jeu.

Lorsque la souris clique sur le bouton « jouer » du menu de démarrage, une partie commence immédiatement. Sans plus attendre, les ennemis appa-

FIGURE 13 – Paramètres

raissent à leurs points d'apparition générés aléatoirement et suivent le chemin menant à la base que le joueur doit défendre. Lorsque la partie démarre, le joueur se voit attribuer mille points de monnaie ainsi que cinq cents points de vie. La monnaie permet de poser des héros sur le terrain, chacun ayant un nom, un délai de tir et des dégâts différents. Il y a en tout trois héros disponibles. À chaque fin de manche, cinq cents pièces de monnaie sont offertes aux joueurs. Une manche se termine lorsque tous les ennemis sont détruits. Si la base n'a plus de points de vie, le joueur perd, et s'il termine la quinzième vague d'ennemis, il gagne. Par ailleurs, chaque ennemi inflige ses points de vie en dégâts au joueur.

FIGURE 14 – Déroulement normal d'une partie de jeu

FIGURE 15 – Point de vie du joueur

6.2 Bilan

Tout au long de ce projet, dont le but était d'implémenter la SDL et de la mettre en pratique lors de la création d'un jeu vidéo, nous avons dû chercher à dépasser nos connaissances afin de le réaliser. Durant tout ce projet, nous avons acquis de l'expérience et une meilleure compréhension de l'informatique.

La progression depuis le début, où nous ne connaissions rien au développement d'un jeu, jusqu'à la fin, où nous avons compris les bases, est la chose la plus remarquable de cette période que nous avons passée à la création de Diffenders.

Ainsi, nous avons développé notre autonomie et notre capacité à nous adapter. Nous ne sommes certes pas complètement satisfaits du résultat final, mais nous sommes contents que le projet soit assez abouti pour permettre de jouer.

6.3 Ameliorations possibles

Les améliorations possibles sont nombreuses, car un jeu peut toujours être amélioré :

— La carte : amélioration de l'aléatoire et de la génération des chemins pour plus de variété ;

— Les ennemis : ajouter plus de variété d'ennemis, avec des vitesses, des dégâts, des quantités de vies différentes et des capacités comme la régénération des ennemis proches ;

— Les tours : ajouter différentes tours avec des effets, des dégâts et des vitesses de tir se distinguant les unes des autres, ainsi que la possibilité d'améliorer les tours pour augmenter les dégâts, tirer plus rapidement, et ajouter des tours à projectiles.

— Les menus : ajoutd'un menu déroulant au lieu d'un menu fixe pour permettre de poser ou d'enlever les tours ;

— Les graphismes : ajout de plus d'images et d'animations, amélioration de la beauté des images déjà présentes ;

— Optimisation de la gestion des tours et des ennemis.

6.4 Nos avis

Notre avis commun est que ce projet nous a permis de rester pragmatiques, car nous n'avons pas pu intégrer toutes les fonctionnalités que nous souhaitions dans notre jeu. Nous serons donc plus réalistes face aux futurs projets qui se présenteront à nous.

6.4.1 Gabriel

Personnellement, ce projet m'a permis de découvrir le fonctionnement du travail de groupe en informatique. J'ai aussi réalisé l'importance cruciale du temps, et que commencer le plus tôt possible n'est pas seulement un avantage, mais une nécessité. Cependant, j'ai beaucoup aimé coder un jeu, même si ce n'est pas le plus grand jeu de tous les temps. C'est mon premier, et je compte bien en créer plusieurs dans le futur.

6.4.2 Alexis

Mon avis sur le projet et ma participation est qu'il m'a permis de découvrir des aspects que je ne pensais pas possibles ou du moins réalisables pour ma part. J'ai vraiment apprécié travailler sur ce projet, aussi intéressant qu'amusant, malgré une organisation du temps de travail assez dispersée.

6.4.3 Clément

Mon avis sur ce projet est qu'il m'a permis d'exploiter mes connaissances et de les élargir, notamment avec l'ajout de la SDL. Il a été intéressant de réaliser un jeu en collaboration avec mes coéquipiers ; nos points de vue divergents sur certains aspects ont été une ouverture vers de nouvelles perspectives.

7 Annnexes

Le projet

Lors de la conception de ce projet, nous avons notamment utilisé l'outil de débogage GDB. Dans l'exemple ci-dessous, il est utilisé pour parcourir l'exécutable de test jusqu'au premier point d'arrêt situé à la ligne 6, correspondant à la fonction int jeu.

FIGURE 16 – Utilisation de l'outil de débogage GDB

```
(gdb) run
Starting program: /info/etu/l2info/s2101025/LE projet/Diffenders/src/test
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7fffee6ce700 (LWP 15381)]
[New Thread 0x7fffede8d700 (LWP 15382)]
[Thread 0x7fffede8d700 (LWP 15382) exited]
[New Thread 0x7fffede8d700 (LWP 15383)]
INFO: 782 288

Thread 1 "test" hit Breakpoint 6, 0x000055555555be16 in jeu ()
```

Avec GDB, nous avons également utilisé Valgrind pour détecter et résoudre les fuites de mémoire causées par le code écrit par les membres du groupe.

FIGURE 17 – Utilisation de valgrind

```
==34316== HEAP SUMMARY:
==34316==     in use at exit: 868,092 bytes in 3,057 blocks
==34316==   total heap usage: 112,672 allocs, 109,615 frees, 310,150,269 bytes allocated
==34316==
==34316== LEAK SUMMARY:
==34316==    definitely lost: 264,321 bytes in 53 blocks
==34316==    indirectly lost: 11,262 bytes in 68 blocks
==34316==      possibly lost: 262,560 bytes in 2 blocks
==34316==    still reachable: 329,949 bytes in 2,934 blocks
==34316==         suppressed: 0 bytes in 0 blocks
==34316== Rerun with --leak-check=full to see details of leaked memory
==34316==
==34316== Use --track-origins=yes to see where uninitialised values come from
==34316== For lists of detected and suppressed errors, rerun with: -s
==34316== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 2 from 2)
```

Nous avons également utilisé d'innombrables jeux de tests qui ont permis de détecter des problèmes plus ou moins cachés au sein du programme. Les

jeux de tests les plus utilisés dans le déroulement de ce projet sont :

— Ajouter des héros sur d'autres héros pour voir s'ils s'empilent correctement.
— Ajouter puis vendre des héros de manière aléatoire pour comprendre si tous se vendent correctement.
— Augmenter ou diminuer le nombre d'ennemis par vague pour comprendre s'il y a des erreurs lors de leur création ou de leur suppression.
— Vérification des coordonnées des points importants d'un chemin (base, virage, point d'apparition) et de leur bonne création.