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 April 2024

Table des matières

Table des figures

1 Introduction

Dans le cadre de notre seconde 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 SDL 2 en 3 mois, en utilisant nos connaissances dans le language C et en apprenant a utiliser la SDL 2, nous avons choisi de coder un jeu de type tower defense, que nous avons nommé Diffenders. Notre but envers la création de ce jeu soit de faire que les ennemis suivent bel et bien un chemin prédéfini, mais que les tours, appelées héros dans notre jeu, soient posé partout sur la carte autre que le chemin et la base à défendre. Nous avons aussi opté pour un système de défense au nombre de vague, ou il faudrait défendre un certain nombre de vague d'ennemis pour gagner. La carte serait générée aléatoirement, pour permettre un semblant de rejouabilité, et sera composée de tableaux de trois types de case, chemin, base et plaine, avec la plaine étant l'endroit où les héros pourront être posés.

L'inspiration de ce projet vient de notre intérêt pour ce genre particulier et unique, qui a été le plus souvent un dérivé d'un jeu, à ces débuts, utilisant ses ressources pour le transformer en ce genre. Avant même de présenter le

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

travail réalisé pour chaque partie du projet, il faut d'abord définir les termes et présenter les partis elle-même.

2 Analyse

2.1 Definition

Le terme tower defense vient de l'anglais, signifiant "défense de tour", bien que la traduction littérale ne soit pas la plus correcte, "défense avec la tour(les tours) " est bien plus correspondant au type de jeux en lui-même. Un tower defense est un genre de jeu vidéo ou l'objectif principal est de défendre un point, une base, un personnage ou un objectif. Selon les jeux, il faut le défendre soit un nombre de tours ou vague, soit un certain temps, soit en accomplissant des objectifs secondaires ou même tenir le plus longtemps possible sans chance de réussite. Dans la plupart des cas, les ennemis se déplacent sur un chemin prédéfini, bien que certains jeux optent pour un chemin personnalisable par le joueur. Pour réussir la mission de le protéger, les joueurs pourront mettre en place des tours qui ont chacune leur spécialité : ralentir les ennemis, leur infliger des dommages, leur mettre des "statuts" comme le poison qui inflige des dégâts sur le temps... Ainsi, le jeu demande au joueur de réfléchir à la meilleure stratégie pour cet défense grâce au tour, d'où le nom tower défense.

Les ennemis dans un tower défense sont souvent très typés, par exemple, il y aura toujours l'ennemi de base, sans effet qui avance vers la tour a une vitesse moyenne. Tout ennemi différent dans le jeu sera une modification légère ou énorme à cet ennemi de base : plus de points de vie (déterminant le nombre de dommage nécessaire avant de tuer l'ennemi), plus de vitesse, de capacité de soin pour les autres ennemis, dans les jeux en 3D l'amélioration peut même être le fait de voler, incitant le joueur a diversifier sa manière de jouer. L'élimination des ennemis rapporte de l'argent, permettant de soit construire de nouvelles tours soit d'améliorer les tours déjà existants.

Une vague d'ennemis est un terme utilisé pour décrire un groupe d'ennemis qui, lorsque tous les ennemis de cette vague sont morts, déclenche une vague suivante. Elles sont progressives en difficulté, allant jusqu'à avoir un boss, un ennemi qui a de base énormément plus de points de vie que tout autre ennemi, et qui a la plupart du temps un effet pouvant gêné le joueur ou améliorer les ennemis.

FIGURE 2 – Rampart l'un des premiers towers defense

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

Voici deux exemples de tower défense.

3 Conception

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

3.1 Règles et fonctionnalités

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 commencé avant (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, avec la possibilité d'accéder à l'option, quitter ou de lancer une partie ;
- Les options comporte la possibilité de passer en plein écran, de revenir au menu ou de revenir en écran fenêtré ;
- La création d'une map aléatoire, généré à chaque nouvelle parti, et les ennemis, quelqu'ils soient, ne doivent pas dépasser du chemin prédéfini ;
- les ennemis se génèrent au niveau des points d'entrée, se dirige vers la base en suivant le chemin ;
- les héros infligent des dégâts au ennemi ;
- une bar de sélection, permettant de choisir quel héros poser ou de détruire un héros ;
- un système d'argent, ou chaque monstre tué rapporte de l'argent ;
- les ennemis deviennent plus nombreux ou plus fort à chaque vague.

Ceux-ci sont les fonctionnalités de bases pour que le jeu soit considéré comme opérationnel.

Les règles du jeu sont :
- le joueur doit survivre à un nombre de vague 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 lorsque poser, les ennemis rapportent de l'argent lorsqu'ils sont tués ;
- le joueur est limité à un certain nombre de tours, ici 10, principalement pour le bon fonctionnement du jeu ;
- lorsqu'un ennemi touche la base elle perd des points de vie ;
- lorsque la base atteint zéro point de vie, le joueur perd.

4 Développement

4.1 Menus de démarrage

Au démarrage du jeu, il y a un menu qui s'ouvre qui contient trois boutons ayant chacun une fonction qui lui est propre, pour les faire et les afficher nous avons créé une fonction qui génère une image dans une texture que l'on a nommé loadImage, cette fonction nous a été utile pour toutes les fonctions. Une fois, les images mise dans des textures, il ne restait plus qu'à les afficher à l'écran à des coordonnées spécifique et mettre à jour le rendu via un SDL_RenderPresent.

Une fois les bouttons afficher nous avons essayer de les rendres un peut plus interactif et pour cela il fallait faire une séri de condition en vérifiant constament les coordonnées de la souris, on à alors eu recours a la fonction SDL_PollEvent qui verifie si il y a un evennement sur la fenêtre ouverte et si c'est le cas il fait une action, ici on vérifie juste les coordonnées de la souris et si elle se trouve sur la position de la texture du boutton alors on change l'image du boutton en une image plus grise pour donner l'impression qu'on passe réellement sur le bouton.

Maintenant il ne fallait plus que faire une action lorsque l'on clique sur cette même position pour que le bouton soit interactif et que le joueur puisse aller dans soit les paramètres, soit quitter le jeu, ou alors lancer une partie. Pour lancer une partie il ne fallait qu'appeler la fonction de jeu fait par Alexis, pour quitter la partie, il fallait sortir de la boucle et fermer la fenêtre, et pour les paramètres, il fallait faire une autre fonction qui devait être utilisable à la fois dans le menu de démarrage, mais aussi dans le jeu pour pouvoir le quitter.

Dans les paramètres, nous avons décidé de ne pas mettre de sauvegarde vu que ça ne se prêtait pas à ce style de jeu, a la place dans les paramètres, nous avons mis comme fonctionnalités un plein écran, un mode fenêtré, un retour au menu ou au jeu et un moyen de quitter le jeu ou la partie en cours. Pour passer en plein écran, il fallait juste modifier les paramètres de les passer en SDL_WINDOW_FULLSCREEN et pour revenir en fenêtrer, il faut mettre ce même paramètre a zéro, une fois cela fait, il faut récupérer les dimensions de la fenêtre pour que la page ne se décentre pas et qu'on puisse passer du plein écran au fenêtré sans que ça ne fasse bizarre.

4.2 Phases de jeu

Le fichier jeu.c comporte donc toutes les fonctions et le code servant au bon déroulement d'une partie de tower defense. C'est ici que sont utilisé les fonctions de créations de chemins aléatoires avec la répartition des points d'apparitions des adversaire et le placement de la base à défendre. Mais avant d'utiliser cela, il se trouve une étape cruciale du jeu. L'affichage du terrain. C'est ici qu'est chargé en mémoire les images nécessaires aux différents éléments tels que le terrain, les tours de défenses, représenter ici par des "héros", ainsi que les ennemies. Une fois ces images chargées, la deuxième mission de jeu. C'est de permettre le dépôt de tour sur l'écran avec une simple clique. Les images sont donc placées par-dessus le fond d'écrans déjà affiché.

FIGURE 4 – Tour de défense imagé par des héros

FIGURE 5 – Carte brute

Ensuite, c'est au tour des chemins et points importants, les points importants sont la base à défendre, le point d'apparitions des ennemies et les chemins générer aléatoirement qui sont donc affiché à l'écran. C'est ensuite au tour des ennemies qui vienne s'afficher par-dessus les chemins, les points d'apparitions et la base. Ennemies qui se déplacent en suivant la position des chemins, chaque case du chemin possédant un emplacement dans une matrice caché qui permet de calculer ses coordonnées en pixel à l'écran. C'est avec les coordonnées calculées des cases des chemins que les ennemies avances. Ils avancent en suivant les coordonnées menant aux prochains points importants du chemin (points d'apparitions, les virages, la base). C'est donc ainsi que les ennemies avance précisément dans la bonne direction menant à la base. Par ailleurs, les ennemies ont été faites de sorte qu'il ne puisse 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 ennemies atteigne la base ? Eh bien, il disparaissent, les ennemies 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 coûts 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 ennemies qui sont détruites sur la route ainsi que les tours de défense qui sont supprimées offre une valeur >. Il y a donc la possibilité de supprimé des tour ce qui rend une partie de la monnaie dépenser pour les poser au joueur.

FIGURE 7 – Prix des tours de défenses

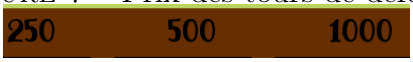
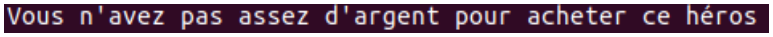


FIGURE 8 – Argent stocké

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



Pour finir, la partie se termine une fois quinze vagues d'ennemie vaincue ou si les points de vie de la base descendent en dessous à zéro et en dessous.

4.3 Génération de la carte

La génération de carte est pseudo-aléatoire, dut à la génération basée sur le temps. La génération peut se décomposer en plusieurs parties.

1. La base à défendre est d'abord posée aléatoirement parmi toutes les cases n'étant pas sur la bordure ;
2. le point d'entrée ou les points d'entrées des ennemis sont créer à l'opposé de la base, pour cela, nous divisons la carte en 4 parties ;
   — en haut à gauche ;
   — en haut à droite ;
   — en bas à gauche ;
   — en bas à droite.
   Par exemple, si la base s'est créée en haut à droite, les points d'entrées seront en bas à gauche ;
3. De ces entrées, nous allons générer un point de rencontre ou tous les chemins d'entrée se rejoindront, partant du coin non-bord d'où sont générés les points d'entrées, et prennent l'abscisse de l'entrée la plus proche du milieu et l'ordonnée de l'entrée le plus proche du milieu de la carte ;
4. De ce point de rencontre, on va ajouter le chemin vers la base, en ajoutant la possibilité de généré une bifurcation pour permettre de voir d'autre forme de chemin.

Tout en générant la carte, on retient toutes les coordonnées des chemins dans un tableau de chaque case que suit le chemin, qui sera transformé en un tableau n'ayant que les points importants ; c'est-à-dire les points d'entrées, les points de rencontre de plusieurs parcours et la base. Ce tableau sera retourné vers le déplacement des ennemis afin de permettre un bon suivi graphique 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 rapidement débuter le projet sous les meilleurs hospices. Premièrement, nous avons eu la chance dans le cadre du projet d'utilisé des outils que nous n'avions pas l'habitude de manipuler. À commencer par 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 mit en place il est donc grand temps de réellement organisé le travail entre tout les membres du groupe. Lors des différentes discussions autour de ce que le projet pourrait prendre comme directions, nous avons rapidement conclu qu'il faut au moins avoir un menu de démarrages fonctionnel qui permet de lancer le jeu. De cette manière, la tache de concevoir le menu de démarrage et toutes ses options a été attribuée à Gabriel. Il est donc la personne s'étant occupé des tous premiers aspects qu'un utilisateur verrons lors du lancement du jeu.

Évidement, il devait aussi avoir une phase de jeu qui vous a déjà été décrit dans le rapport de projet. Les ennemies apparaissent et avancent jusqu'à un objectif que nous joueurs défendons à l'air de tour de défense. Eh bien, cette phase de jeu est fractionnée en deux parties dans l'organisation des tâches.

La première partie est de s'occuper des premières fonctions et structures au bon fonctionnement du jeu. Ces fonctions et structures "primitives"

servent dans un premier temps pour avoir un avant-goût du fonctionnement du jeu sans interface graphique. Ici la création de héros, d'ennemie, du joueur, etc. Ceci formera le cœur de cette partie, car même après révisions des besoins et amélioration des structure et des fonctions leur utilisation n'en reste pas moins indispensable, cette partie est donc gérée par Alexis.

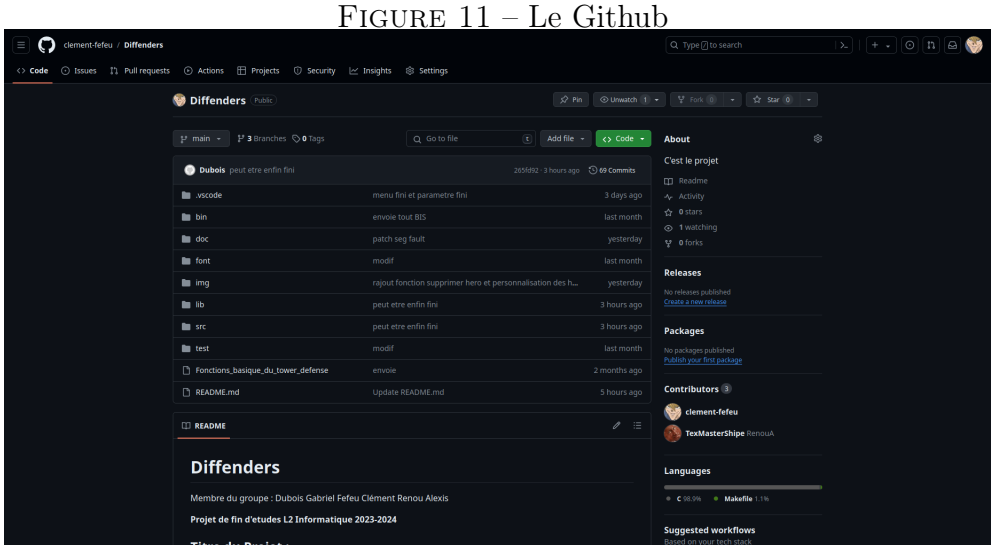
La seconde partie se penche sur la création de la carte, ici plusieurs idées sont parvenus dans le groupe. Faire une carte avec un chemin prédéfinis ? Faire plusieurs cartes avec des chemins différents et un système de rotation de carte ? Où encore faire une carte avec un chemin aléatoire. C'est la dernière option qui est mise en place ici, cette partie assez technique est proposé par Clément qui a décidé de la mettre en place.

5.2 Les Outils

A aussi été fait un Gantt dans un but prévisionnel , celui ci ayant été d'une grande aide pour la détermination des rôles au seins du groupes et la prévisions en temps de chaque tache. Du Gantt, nous avons pu, sur l'intervalle de temps donné par le jour du compte-rendu du projet qui doit être donné le 19 avril 2024, que nous avions à partir du 9 janvier pour faire le projet nous lassant trois mois et demi pour prendre en main Github et la SDL, l'utiliser pour créer notre projet, faire un compte-rendu et préparer un oral et un diaporama.

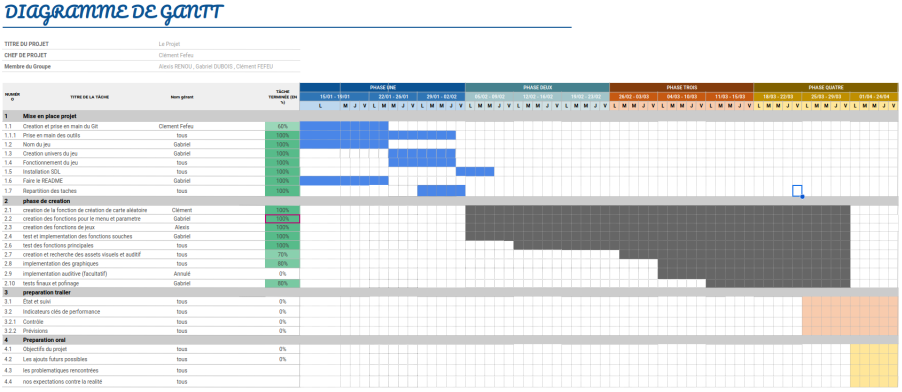
La prise en mains de Github pour le partage des programmes, images, sources et informations a été très vite prise en main, bien que peu utilisé. Cependant, nous avons fait le nécessaire pour l'utiliser lorsque cela est nécessaire.

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 consistent, nous a donner polus de fils à retordre. Bien que nous ayons fini par comprendre l'utiliser, nous avons pris plus de temps que prévu et ainsi avons pris du retard que nous avons eu du mal à rattraper.

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



6 Conclusion

Il est maintenant temps de passer à la conclusion. Est détaillé dans cette partie le résultat du projet qui a été mené, ainsi qu'un bilan de celui-ci et pour finir quelque ligne concernant l'avis des membres du groupe , un avis commun dans un premier temps suivi d'un avis personnel de chacun.

6.1 Résultats

À la fin de ce projet, les résultats observés sont donc la possibilité de lancé le jeu. Le lancement du jeu débouche directement sur un menu de démarrage qui permet alors de choisir entre lancer une partie, changer les paramètres et quitter le jeu. Les paramètres disponibles sont une mise en plein écran , en fenêtrer ou encore quitter le jeu.

FIGURE 13 – Paramètres

Lorsque la souris clique sur le bouton jouer du menu de démarrage une partie commence alors. Sans plus attendre les ennemies apparaît à leurs points d'apparitions générer aléatoirement et suivent le chemin qui mène à la base que le joueur doit défendre. Lorsque la partie démarre le joueur se voit attribué 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 différent. Il y a en tout trois héros disponibles. À chaque fin de manche cinq cents pièce de monnaie sont offerte aux joueurs. Une manche se termine lorsque tous les ennemis sont détruits. Si la base n'a plus de point de vie, le joueur perd et si le joueur termine la vague quinze d'ennemi, il gagne. Par ailleurs, chaque ennemi inflige ses points de vie en dégâts aux joueurs.

FIGURE 14 – Déroulement normal d'une partie de jeu

FIGURE 15 – Point de vie du joueur

6.2 Bilan

Tour au long de ce projet qui avait pour but 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 de la compréhension envers l'informatique.

La progression du début, où nous ne connaissions rien au développement d'un jeu, à la fin, où nous avons compris les bases à celle si, est la chose la plus remarquable, dans cette période de temps que nous avons passé à la création de Diffenders.

Ainsi, nous avons développé notre autonomie et notre capacité à nous adapter. Nous ne sommes certes pas complètement satisfait du final, mais somme content 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 génération des chemins pour plus de variété ;
- Les ennemis : ajoutés 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 : ajoutés différentes tours avec des effets, des dégâts, vitesse de tirs se distinguant les unes des autres, et l'ajout de la possibilité d'améliorer les tours pour plus de dégâts, tirer plus rapidement, ainsi que de tours à projectiles.
- Les menus : ajout d'un menu déroulant au lieu d'un menu fixe pour pouvoir poser ou enlever les tours ;
- Les graphismes : ajout de plus d'images et de l'animation, améliorer la beauté des images déjà présente ;
- Optimisation de la gestion de tours et ennemis.

6.4 Nos avis

Notre avis commun est que nous a permit de revenir à la réalité, car nous n'avons pas pu mettre tout ce que nous voulions dans notre jeu, et que nous serons plus réalistes envers les futurs projets qui se dresseront devant nous.

6.4.1 Gabriel

Personnellement, ce projet m'a permis de découvrir comment le travail de groupe en informatique fonctionne, j'ai aussi pu remarquer que le temps est extrêmement important et que commencer le plus tôt possible n'est pas qu'un luxe. Par contre j'ai beaucoup aimé coder un jeu même si ce n'est pas le plus grand jeu de tous les temps, c'est le premier et je compte bien en créer plusieurs dans le futur.

6.4.2 Alexis

Mon avis, concernant le projet et ma participation, a celui-ci est qu'il m'a permis de découvrir des choses que je n'aurais pas pensées possible du moins faisable pour ma part. J'ai vraiment bien aimé travaillé sur ce projet aussi intéressant qu'amusant malgré une organisation du temps de travail assez disperser.

6.4.3 Clément

Mon avis sur ce projet est qu'il m'a permis d'exploiter mes connaissances et d'élargir celle-ci, avec l'ajout de la SDL. Il a été intéressant de réaliser un jeu en commun avec mes comparses, nos avis divergents sur certains points furent une ouverture vers un nouveau point de vue.

7 Annnexes

Le projet

Il a notamment été utilisé lors de la conception de ce projet l'outil de débogage GDB. Dans l'exemple ci-dessous, il est utilisé pour parcourir l'exécutable test jusqu'au premier breakpoint se trouvant en ligne 6 et étant 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 a aussi été utilisé valgrind pour prévenir et guérir les fuites de mémoire dû au code des écrits 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 aussi utilisé d'innombrables jeux de test qui ont réussi à déceler des problèmes plus ou moins cachés au sein du programme. Les jeux de

test qui sont le plus utilisés dans le dérouler de ce projet sont :

— Ajouter des héros sur des héros pour voir si ceux-ci s'empilent.
— Ajouter puis vendre des héros de manière aléatoire pour comprendre si tous se vende bien
— Augmenter ou diminuer le nombre d'ennemies par vague pour comprendre s'il y a erreur lors de leurs créations ou suppression.
— Vérification des coordonnées des points important d'un chemin (base, virage, point d'apparition) et de leurs bonnes créations