TOWER DEFENSE

RAPPORT PROJET

MARGAUX VAILLANT

ANDREA GUILLOT

1. INTRODUCTION

Bonjour. Votre mission, si toutefois vous l'acceptez, consiste à prendre part au combat des *Slip Fighters* en exterminant leurs détracteurs. Les *Slip Fighters* ont été créés à l'occasion d'un WEI étudiant et comptent actuellement les meilleurs éléments de la promotion IMAC 2021. À travers ce tower defense, vous pourrez contrôler et diriger leurs effectifs afin d'empêcher ceux qui médisent d'eux de s'enfuir. Ce rapport ne s'autodétruira pas dans 5 secondes.

2. CAHIER DES CHARGES

	FAIT	MAIS NE FONCTIONNE PAS	PAS FAIT
STRUCTURE DU PROJET ▪ Réalisation d'une structure ordonnée ▪ Système de compilation	✔ ✔		
INITIALISATION D'UNE CARTE ▪ Chargement d'une carte ▪ Création du graphe des chemins ▪ Vérification de la jouabilité de la carte	✔ ✔ ✔		
CRÉATION DES DIFFÉRENTS ÉLÉMENTS ▪ Sprites tours, monstres et bâtiments ▪ Création des structures de tours, monstres et bâtiments	✔ ✔		
ITD ▪ Ajout, suppression et édition de bâtiments et de tours ▪ Déplacement des monstres ▪ Gestion des actions des tours et bâtiments ▪ Gestion du temps	✔ ✔ ✔ ✔		
FICHIER DE DESCRIPTION ITD ▪ Lignes rédigées comme demandé	✔		
VALIDATION D'UNE CARTE ▪ Fichier .itd correct ▪ Vérification des 6 paramètres existants ▪ Vérification du code des couleurs ▪ Vérification du nombre de noeud ▪ Vérification d'au moins une zone d'entrée et d'une zone de sortie	✔ ✔ ✔ ✔ ✔		
LES MONSTRES ▪ Disposent de points de vie ▪ Disposent d'une résistance propre à chaque tour ▪ Arrivent par vague de 10 ▪ Les points de vie augmentent avec le niveau ▪ Apportent de l'argent au joueur ▪ Implémentation de plus de 2 types de monstres ▪ Graphismes	✔ ✔ ✔ ✔ ✔ ✔ ✔		
LES TOURS ▪ Ont trois caractéristiques : puissance, portée, cadence ▪ Caractéristiques selon le type de tour ▪ Ont un coût d'achat ▪ Construction sur zones constructibles ▪ Empêcher construction sur tour ▪ Empêcher construction sur installation	✔ ✔ ✔ ✔ ✔	?	

LES BÂTIMENTS ▪ Ont une portée ▪ Caractéristiques selon le type d'installation ▪ Ont un coût d'achat ▪ Construction sur zones constructibles ▪ Empêcher construction sur tour ▪ Empêcher construction sur installation	✔ ✔ ✔ ✔ ✔	?	
INTERFACE ▪ Voir la carte ▪ Voir les monstres ▪ Voir les tours et installations ▪ Indiquer l'argent disponible ▪ Sélectionner une tour ou une installation ▪ Placer une tour ou une installation ▪ Déplacer les monstres ▪ Afficher propriétés des bâtiments et de tours	✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔		
BONUS ▪ Afficher propriétés et barre de vie des monstres ▪ Les vagues de monstres mettent des monstres de plus en plus vite ▪ Effets sonores ▪ Zones de type non constructibles ▪ Visualisation des tirs des tours sur les monstres ▪ Changement du bouton play/pause selon si le jeu est lancé ou non ▪ Lorsqu'on est dans l'aide, le jeu est en pause ▪ Commandes clavier ▪ Bouton pour sortir du jeu	✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔		

3. DESCRIPTION GLOBALE DE L'ARCHITECTURE

bin	include	obj	src	data	images	sounds	doc	makefile

▶️ BIN

Contient l'exécutable : *itd*.

▶️ INCLUDE

Contient les prototypes des fonctions et les structures.

element	file	geometry	ihm	sdl_tools.h

> element

Regroupe les fichiers liés aux éléments : *monster.h*, *tower.h*, *installation.h* et *shot.h*.
Nous avons choisi de tous les séparer afin de pouvoir les gérer un à un et résoudre plus facilement les erreurs. En effet, nous avons commencé par coder les monstres et les tours. Ensuite, nous avons pu développer les tirs des tours qui prennent en compte les propriétés des monstres et des tours. Enfin, lorsque les tours ont pu supprimer les monstres, nous avons codé les installations qui améliorent les propriétés des tours.

> **file**

Regroupe les fichiers liés au jeu : *map.h*, *image.h*, *texture.h, fileTower.h*, *fileInstallation.h* et *sound.h*.
Première étape, avec le fichier *map*, nous réalisons des actions sur notre map au format ppm pour ensuite recréer, à partir des données obtenues, l'image du plateau de jeu avec le fichier *image*. Seconde étape, avec le fichier *texture*, nous avons pu tout d'abord générer les textures de la map. Par la suite, ce fichier nous a permis de générer celle de tous les autres éléments du jeu. Troisième étape, nous avons codé *FileTower* et *fileInstallation* qui sont les fichiers nous permettant de gérer les propriétés propres à chaque construction et leur ajout ou suppression sur le plateau de jeu (à travers les listes de tours). Et, finalement, quand tout le reste fonctionnait, nous avons ajouté des effets sonores pendant la partie (musique de fond, son quand un monstre meurt).

> **geometry**

Regroupe les fichiers d'éléments géométriques : *color3f*, *intersection*, *point2D*, *vector2D*.
Nous avons individualisé ces fichiers pour pouvoir utiliser ailleurs seulement ceux qui nous intéressent. *Color3f* prend en charge tout ce qui est relatif aux couleurs (ajouter couleur par exemple). *Intersection* gère toutes les intersections qui existent dans notre jeu (intersection entre segments par exemple). *Point2D* gère les points à deux coordonnées et *vector2D* gère toutes actions sur vecteur à deux coordonnées également.

> **ihm (interface-homme-machine)**

Regroupe les fichiers liés à l'affichage et aux interactions : *draw*, *interface*, *menu* et *node*.
Draw est le fichier qui nous permet de dessiner absolument tous les éléments du jeux (la carte, les monstres/tours/installations mais aussi les menus, etc.). C'est avec le fichier *menu* qu'on va pouvoir gérer les interactions entre le joueur et l'application lorsque l'on clique sur les différents éléments proposés (menu d'accueil, bouton aide, bouton exit, choix des tours, etc.). *Interface* va quant à lui nous permettre d'afficher le niveau du joueur (soit le nombre de vague passée) et son argent. Ce fichier nous permet également de tenir à jour ces informations tout au long de la partie. Enfin, le fichier *node* est celui qui nous permet de faire bouger les monstre sur le chemin.

> *sdl_tools*

Ce fichier nous permet d'ajouter les pixels à la surface SDL.

OBJ

Dossier de destination des fichier obj.

SRC

Contient les fonctions.
Le découpage est le même que dans le dossier include.
On y trouve en plus le fichier *main* qui appelle les fonctions créées.

DATA

Contient la map au format idt.

IMAGES

Contient la map au format ppm et toutes les images du jeu au format png.

SOUNDS

Contient les effets sonores du jeu au format wav (format plus léger que le mp3).

DOC

Contient le sujet de ce projet et ce même rapport.

MAKEFILE

Permet la compilation des fichiers.

4. DESCRIPTION DES STRUCTURES DE DONNÉES

Main, Tower, Monster, Installation, Shot, Map, Image, Draw, Menu, Interface, Node

Main

Nous créons dans un premier temps la fenêtre de l'application. Nous initialisons ensuite les éléments du jeu (texture, propriétés, etc.). Nous avons également plusieurs variables qui nous permettront de savoir notamment si on a une tour ou un bâtiment "en main" (c'est-à-dire qu'on a cliqué dessus mais qu'on ne l'a pas encore construit.e), si nous sommes en zone constructible ou non, le menu dans lequel nous nous trouvons (menu principal, menu d'aide, la fenêtre de jeu, l'écran final) et enfin un booléen permettant de savoir si nous sommes entrain de jouer ou non. On a aussi une variable "propriété" qui nous permet de savoir si nous souhaitons visualiser les propriétés d'un tour, d'un monstre ou d'un bâtiment. Nous créons ensuite un nouveau joueur, avec une liste de monstres, de tours et d'installations, qui vont s'instancier petit à petit durant la partie. La carte est ensuite chargée, et si on veut commencer à jouer, on dessine tout notre jeu.

Tower

Une tour possède plusieurs attributs :

- **Une position** définie grâce à deux flottants : x et y
- **Un type** de tour défini par un char
- **Une cadence**, **une portée**, **une puissance** et **un coût** définis par des int
- **Des booléens** nommées *affectedByRadar* / *Usine* / et *Stock* (qui nous permettent de savoir si la tour est affectée par une installation)
- **Deux pointeurs** : sur la tour précédente et la suivante

On a ensuite une structure de liste de tours qui a pour attributs :

- **Une taille**
- **Deux pointeurs** : sur la tête et la queue de la liste

Avec ces structures viennent plusieurs fonctions permettant la création de chacune des tours, la vérification de la constructibilité des tours, et enfin leur délétion (autant graphiquement que algorithmiquement).

▶️ **Monster** :

Un monstre possède comme attributs :

- **Une position** en x, y
- **Un type** de monstre désigné par une lettre
- **Un sens** (utile pour le déplacement, et de base, pour faire des sprites)
- **Le noeud précédent**
- **Le noeud suivant** qu'il va rencontrer
- **Ses points de vie** ainsi que **le maximum** qu'il a (pour la barre de vie)
- **Des résistances** propres à chaque tour
- **Un gain** (ce qu'on gagne quand il meurt)
- **Une vitesse**
- **Deux pointeurs** : vers le monstre précédent et suivant

La liste de monstres suit sur le même principe que la liste de tours. En plus des fonctions présentes avec les structures de tour (appliquées ici aux monstres), on a également une fonction permettant le déplacement.

▶️ **Installation** :

Les installations ressemblent beaucoup aux tours sauf qu'elles ont moins d'attributs étant donnée qu'elles n'ont qu'**une portée**. On retrouve dans cette structure les deux fonctions *moveInstallation* et *moveTower* qui permettent aux bâtiments de suivre la souris quand on les installent Elles sont dans le même fichier afin d'éviter la boucle infinie de la double inclusion (*tower.h* dans *installation.h* et inversement).

▶️ **Shot** :

Un shot est défini par :

- **Une position**
- **Un monstre ciblé**
- **Une tour d'origine**
- **Une puissance**
- **Le type de tour** depuis laquelle il est lancé
- **Le shot d'avant** et **le shot d'après**

Les fonctions associées à cette structure permettent la création d'un missile, son déplacement mais aussi de vérifier la collision avec un monstre, et le cas échéant, sa suppression.

▶️ **Map** :

La map est définie par :

- **Un pointeur** sur image
- **Une énergie** définie par un int
- **Les couleurs** du chemin, des noeuds, des zones constructibles, de la zone d'entrée et de sortie
- **La liste des pixels des zones constructibles**
- **Le nombre de noeud**
- **La liste des noeuds**

Les fonctions associées à cette structure permettent notamment d'initialiser notre map, d'effectuer toutes les vérifications nécessaires (format correct, validité des noeuds / chemins / entrée et sortie), d'envoyer les valeurs des couleurs vers Image et finalement de libérer la mémoire prise par la map à la fin.

▶️ **Image** :

L'image est définie par :

- **Un pointeur** sur le chemin de l'image
- **Le numéro magique** (qui nous permet d'identifier le type de fichier)
- **Deux dimensions** : hauteur et largeur
- **Une valeur maximale de résolution**

Les fonctions associées permettent simplement d'ouvrir une image, de la stocker, puis de la supprimer le moment venu.

▶️ **Draw** :

Ce fichier contient toutes les fonctions relatives au dessin. Elles nous permettent ainsi de dessiner :

- **Les éléments** : monstres et leurs pv, tours, installations, shots
- **Les différents menus** : accueil, aide, barres du haut et de gauche pendant le jeu
- **Les boutons**
- **La carte**
- Mais aussi du **texte** : propriétés des éléments, informations sur la partie (nombre de vagues, argent)

Menu :

Ce fichier va alors quant à lui faire le lien entre le dessin et les actions de l'utilisateur :

- Si on clique sur un bouton (jouer, aide, pause, quitter, supprimer bâtiments)
- Si on clique sur une tour ou une installation pour l'acheter
- Si on clique sur une tour, une installation ou un monstre pour voir ses propriétés

Interface :

Ce sont les fonctions qui permettent d'initialiser l'interface du joueur et de la tenir à jour. L'interface du joueur désigne la partie du code qui lui permet de voir son niveau (équivalent au nombre de vagues passées) et l'argent qu'il possède.

Node :

Un noeud est défini par :

- **Deux coordonnées** : x et y
- **Le noeud suivant**

On a ensuite une structure de liste de noeuds qui contient comme attributs :

- **Une taille**
- **Deux pointeurs** : vers le premier élément et vers le dernier

Les fonctions associées à ces structures permettent notamment l'initialisation des noeuds et l'ajout ou la suppression de noeuds.

5. DIFFICULTÉS ET SOLUTIONS

5.1 Les vagues de monstres

Il a fallu réussir à créer 10 monstres d'affilé sans que ceux-ci s'affichent en même temps. On a donc créé une variable qui permet de créer un monstre toutes les secondes environ (variable j). Elle s'incrémente à chaque loop et c'est qu'à partir d'un certain nombre qu'on crée le monstre. Concernant la vitesse du monstre, on a une variable k qui s'instancie à chaque boucle et on fait un modulo avec la vitesse donnée afin de régler la vitesse du monstre sur la fréquence d'une boucle.

5.2 Les installations

Nous n'avions d'abord pas de booléen et elles augmentent donc les caractéristiques des tours infiniment (à chaque loop). Après la création du booléen, il fallait aussi prendre en compte que si une tour survolait l'installation avant d'être posée, elle ne devait pas être augmentée.

5.3 La constructibilité

quand on cliquait dans notre menu et qu'on ne bougeait pas encore la souris, la tour/installation partait du principe qu'elle était en zone constructible. Il a donc fallu instancier le booléen concernant la constructibilité lors des cliques sur les menus à 0. Aussi, les tours ne reconnaissent pas les installations et les installations les tours. Nous avons donc ajouté dans chacune la liste des tours/installations, et appliqué nos fonctions d'intersection. Cependant, elles ne semblent toujours pas se distinguer et sont donc constructibles les unes sur les autres.

5.4 Inclusion

Nous avons plusieurs fois eu le problème de la double inclusion, qui fit donc une boucle infinie entre deux fichiers .h. Il a donc fallu repenser les fonctions.

5.5 La musique

fut également compliquée à mettre. En effet, tout notre code se déroule dans la boucle. Si on mettait donc le code de la musique dedans, elle se répétait indéfiniment. Finalement on a donc choisi de la lancer avant la boucle. Les effets sonores sur la mort des monstres s'effectuent dans la fonction *removeMonster* et ne se passe donc que dès que le monstre meurt.

5.6 Pixel

Au début, se repérer en pixel pour écrire ou placer des objets et des zones a été compliqué. Finalement nous nous somme rabattu sur un dessin papier et avec calculé les zones que nous voulions ainsi que leurs coordonnées afin de savoir où tester nos x et y. Pour écrire du texte, nous avons souvent regardé approximativement où il arrivait.

5.7 GitKraken

Nous avons également eu plusieurs problèmes avec GitKraken. Parfois il ne reconnaissait plus le dossier où nous avions cloné notre projet. Il a donc fallu plusieurs fois le recloner dans un nouveau dossier.

6. DESCRIPTION DÉTAILLÉE DES FONCTIONNALITÉS

Instructions commandes :

1) Compiler le programme : $ make
2) Lancer le programme : $ ./bin/itd
3) Nettoyer les fichiers .o : $ make clean
4) Nettoyer en profondeur : $ make mrproper

Guide utilisateur :

Lorsque vous lancez l'application, vous arrivez sur la page d'accueil du jeu. Deux choix s'offrent alors à vous :

- JOUER : vous démarrez une nouvelle partie.
- AIDE : vous ouvrez la page d'aide qui explique le but du jeu et le rôle de chaque élément du jeu. Vous pouvez à tout moment sur la page précédente en cliquant sur la croix rouge située en haut à droite. Par ailleurs, pendant votre partie, vous pourrez la retrouver en cliquant sur le point d'interrogation bleu situé en haut à droite.

Dès que vous cliquez sur *jouer* la partie commence et la première vague de monstres apparaît. Il y a deux types de menu :

- **Menu de gauche** : il montre les tours et installations disponibles ainsi que leur prix. Quand on clique sur l'une d'elles, ses caractéristiques s'affichent.
- **Menu du haut** : à gauche, il affiche le numéro de la vague de monstre actuelle et l'argent disponible dans votre porte-monnaie. À droite, se trouve le bouton le bouton aide et le bouton croix pour quitter le jeu. Il y a également un bouton pause/play selon la situation du jeu.

Le reste de l'application constitue le plateau de jeu. Quand une tour ou une installation est sélectionnée et donc achetée, vous pouvez la placer dessus :

- Si la zone autour de l'élément est rouge c'est que l'emplacement n'est pas constructible.
- Au contraire, si la zone est verte, l'élément peut être construit à l'endroit souhaité.

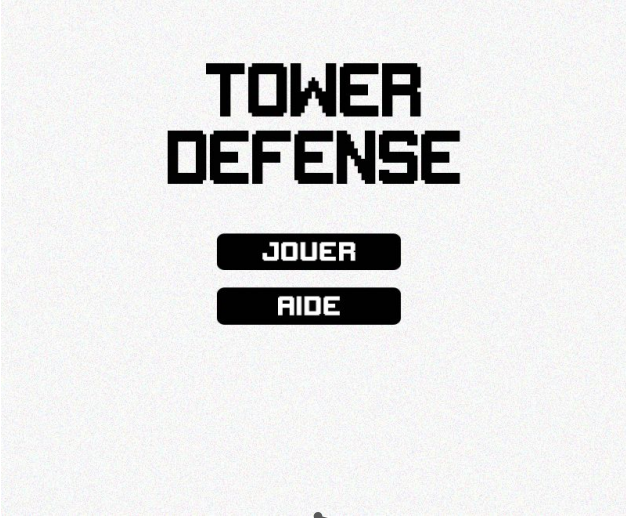
Vous pouvez supprimez une tour ou une installation à tout moment en cliquant dessus puis sur le bouton *supprimer* du menu gauche. Attention, vous ne récupérerez pas entièrement l'argent ayant servi à son achat, mais seulement 60% du prix initial (mais cela vous permet de libérer de la place sur le terrain).

Pour gagner la partie, il faut passer 50 vagues de monstres.
Si un monstre réussi à passer les deux champignons délimitant la fin du chemin, la partie est perdue. Vous pouvez ensuite recommencer une partie.

Commandes clavier : - On peut quitter la partie en appuyant sur q ou echap,
- On peut mettre le jeu sur pause en appuyant sur p ou sur la barre espace.

7. RÉSULTATS OBTENUS

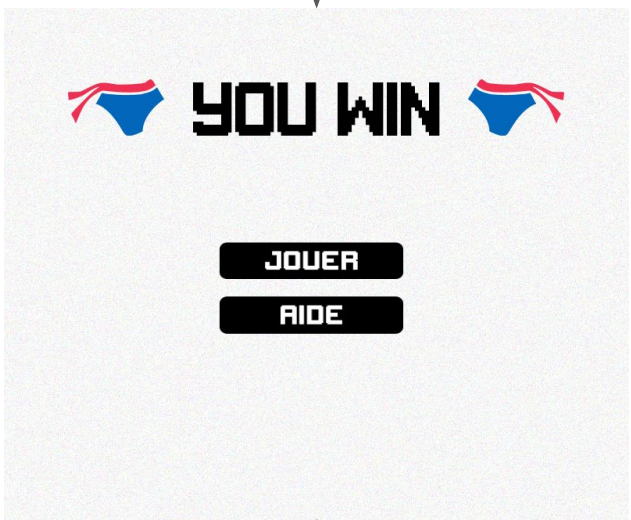
→ Exemple de chemin utilisateur

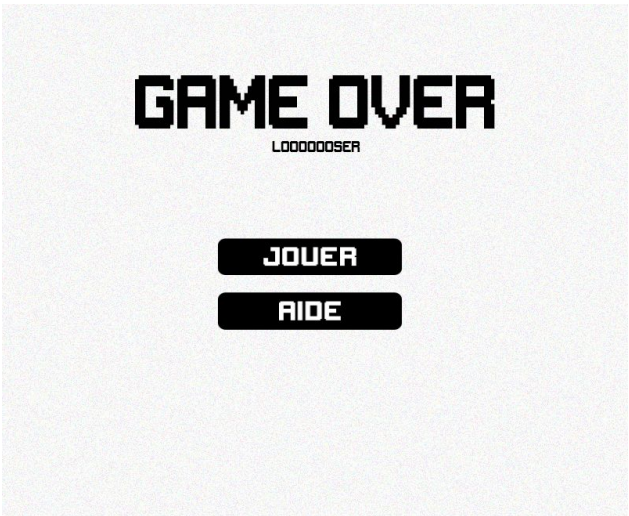


AIDE

JOUER

ATTEINDRE LE PALIER 50





TOUCHE Q : QUITTER LE JEU

8. CONCLUSION

Travailler sur ce projet nous parut compliqué au premier abord : choisir par où commencer, se répartir le travail, savoir gérer le temps. Avant de coder, nous avons donc pris le temps de bien comprendre le sujet pour déterminer quelle serait la structure du jeu. Petit à petit, en plus de nous avoir permis d'appliquer nos cours du second semestre et ainsi encore mieux les comprendre, cet exercice s'est révélé être amusant pour nous.

Le fait de créer un jeu, de pouvoir y jouer, nous a poussé à créer des fonctionnalités supplémentaires par rapport à celles attendues. Ce projet nous a donc également mis dans la position d'une entreprise qui crée un jeu et souhaite combler au maximum les attentes du joueur.

Finalement, nous avons atteint les objectifs que nous nous étions fixées et nous sommes fières du rendu.