Rapport projet de C/openGL

Création d'un Imac Tower Defense

Roxane Vallée
Sarah Veysset

IMAC 1 - 2018/2019

Introduction

Dans le cadre de nos études en **école d'ingénieur IMAC** (Image, Multimédia, Audiovisuel, Communication), nous avons été amenées à **réaliser un 'mini-jeu'** appelé Imac Tower Defense.

Le but est d'**apprendre à gérer un projet en équipe**, de la reformulation du cahier des charges jusqu'à sa réalisation complète en **respectant les exigences** du commanditaire. Vous verrez dans ce projet une application directe des compétences acquises au cours de notre cursus scolaire notamment en ce qui concerne les **langages C, OpenGL** ainsi que l'utilisation du logiciel **Illustrator.**

Le projet s'est déroulé en trois étapes :

- La **compréhension du sujet** qui consiste à étudier le cahier des charges fonctionnel et à le reformuler.

- La **conception et la réalisation** du projet : nous évoquerons dans cette partie le **choix des outils et des langages utilisés** pour la réalisation du site, du développement proprement dit (le codage) et enfin du Game Design. Nous parlerons aussi des **résultats obtenus, des problèmes rencontrés** lors de la conception et enfin nous proposerons quelques idées **d'amélioration du logiciel.**

- Le **planning** : celui-ci nous a permis de mesurer l'avancement de nos travaux et l'atteinte de nos objectifs.

Présentation de l'application

Brief du projet

Employé au sein d'une **entreprise de jeu vidéo**, notre groupe a été chargé de **créer un prototype de logiciel** permettant, d'une part, le **chargement d'une carte**, et d'autre part, la construction sur cette carte de différents **bâtiments** pour empêcher des vagues de monstres tentant de trouver la sortie d'atteindre cette dernière. Ce logiciel se nomme **'Imac Tower Defense'.**

Synopsis

À partir de ce brief nous avons imaginé le **mini-jeu 'CrabAttack' :**

«**Empêcher les crabes maléfiques** d'atteindre la plage et de pincer les vacanciers. En tout, vous devez résister à **50 vagues** de crabes tueurs. Pour vous défendre, vous devrez **construire des palmiers** qui lanceront des noix de coco.»

Travail accompli

	Fait	Pas fait	Fait mais ne marche pas
Réalisation d'une structure ordonnée (partie algorithmique)	X		
Makefile (partie algorithmique)	X		
Chargement d'une carte (partie infographie)	X		
Création du graphe des chemins (partie algorithmique)	X		
Vérification de la jouabilité de la carte (partie algorithmique)	X		
Sprites de bâtiments, de tours, de monstres (partie infographie)	X		
Création des structures de bâtiments, de tours, de monstres (partie algorithmique)	X		
Ajout, suppression et édition de bâtiments et de tours (partie infographie)	X		
Déplacement des monstres (partie algorithmique et infographie)	X		
Gestion des actions des tours et bâtiments (partie infographie)	X		
Gestion du temps (partie algorithmique)	X		

Travail en plus

	Fait	Fait mais ne marche pas
Barre de vie des monstres	**X**	
Projectile	**X** (fait mais c'est un laser)	
Supprimer une tour	**X**	
Musique	**X**	

Fonctionnement du jeu

Description détaillée des fonctionnalités

Démarrage du jeu

Pour démarrer le jeu, le joueur doit **ouvrir un terminal** depuis le dossier «Imac_Tower_Defense». Il entre ensuite la **commande «make»** puis le chemin : **«./bin/itd».**

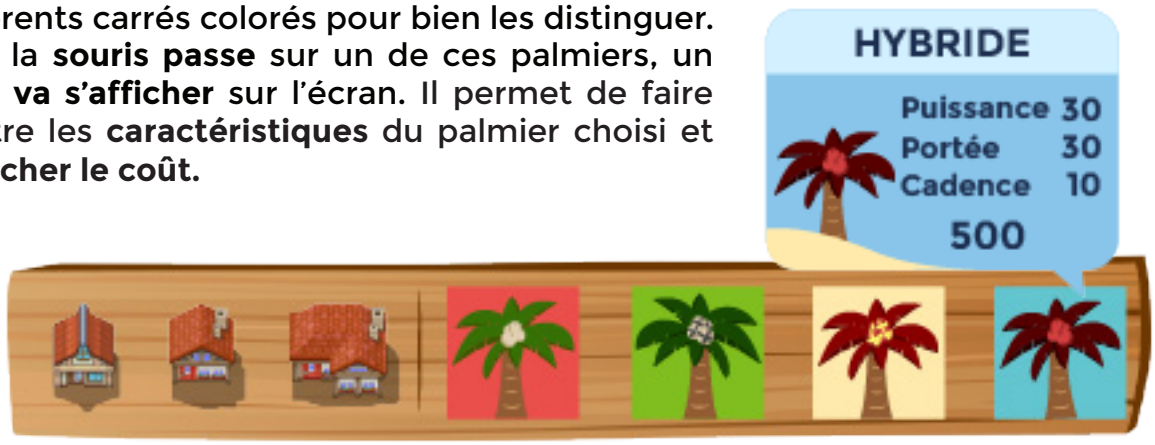
Menu principal

Lorsque le jeu se lance, la première image qui apparaît est le **menu.** Il permet soit **d'accéder directement à la partie** en tapant sur le **bouton 's' (pour start)** soit d'accéder au guide du jeu en tapant sur le **bouton 'a'** (pour aide). Nous avons choisi des **clics au clavier** pour ne pas entraver la jouabilité du jeu. Les zones cliquables pour nos boutons sont en effet présentes dans l'ensemble du code, elles auraient donc **posées des problèmes** lors du placement des tours ou des bâtiments. Pour revenir au menu, l'utilisateur devra taper 'm' (pour menu).

Dans une partie

Le joueur débute la partie avec **1000 noix de coco** (représentant son argent). Cette donnée s'affiche **en bas à gauche de l'écran** du joueur. Une fois que le joueur est prêt il pose une première tour pour **démarrer la partie.**

Il va devoir faire face à une **première vague de monstres** qui entreront **en haut à gauche** de l'écran. Pour y faire face, le joueur va devoir **placer des palmiers** (les tours). **En bas à droite** de l'écran se trouve le menu permettant **la sélection des palmiers**. Ils sont placés sur différents carrés colorés pour bien les distinguer. Lorsque la **souris passe** sur un de ces palmiers, un **encadré va s'afficher** sur l'écran. Il permet de faire apparaître les **caractéristiques** du palmier choisi et d'en **afficher le coût.**



Fonctionnement du jeu

Pour **créer un palmier,** il faut **cliquer sur l'un d'entre eux** et ensuite **cliquer sur la carte.** Si l'utilisateur ne possède **pas assez de noix de coco** ou si la zone où le joueur clique n'est **pas constructible** alors le **palmier ne s'affiche pas** à l'écran.

De la même manière, le joueur va pouvoir **poser des bâtiments.** Ils permettent **d'augmenter certaines caractéristiques des palmiers** à sa portée mais coûtent plus chers. Une fois les tours ou les bâtiments posés, le joueur peut **consulter leurs caractéristiques** en passant la souris sur ceux-ci.

À tout moment du jeu, le joueur pourra **consulter le guide du jeu en passant sa souris** sur le point d'interrogation (en haut à droite de l'écran). L'utilisateur peut **mettre en pause** volontairement le jeu en cliquant sur la flèche ou **quitter le jeu** en cliquant sur la croix.

Le joueur peut aussi **supprimer une tour** pour **regagner de l'argent.** Il lui suffit de **cliquer sur le bouton 'supprimer tour'** puis de **cliquer sur la tour** qu'il souhaite enlever.

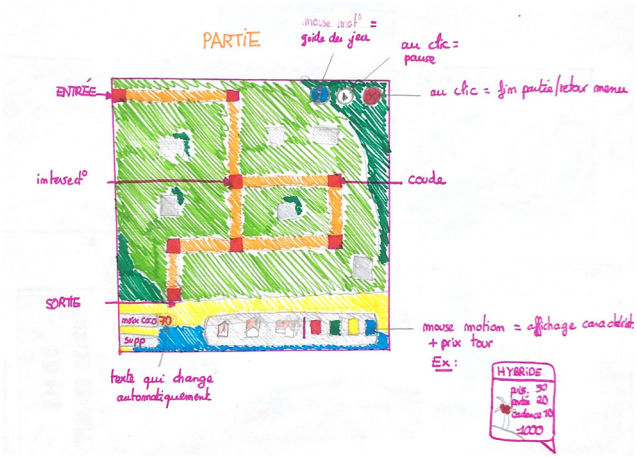
Lorsque le joueur **gagne,** un **panneau 'WIN' s'affiche.** Lorsqu'il **perd le panneau 'GAME OVER' s'affiche.** La **croix rouge** permet au joueur de **revenir au menu principal.**

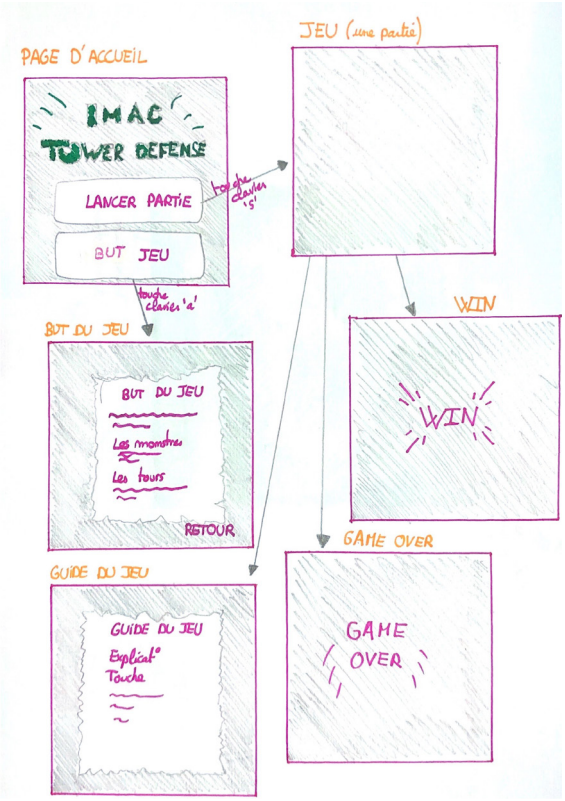
Game Design

De l'imagination à la réalisation

Pour nous, le **design du jeu** est tout aussi **important** que son fonctionnement. C'est pourquoi nous avons mis un **point d'honneur** à développer cette partie. L'ensemble des infographies a été réalisé sur **Illustrator.**

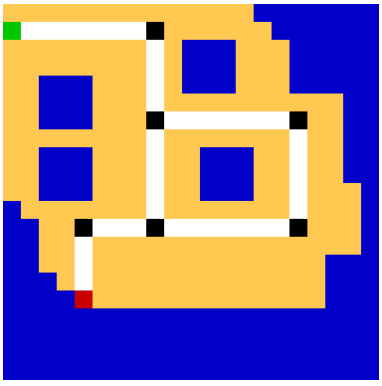
La création du design nous a également **aidé à structurer notre jeu :** « qu'affiche t'on en premier ? », « qu'est ce que l'utilisateur va pouvoir effectuer comme action sur tel ou tel menu ? », « sur quoi va-t-il devoir cliquer ? », ... Ainsi nous avons conçu **l'arborescence** (ébauche) de notre jeu, autrement dit **le scénario.**



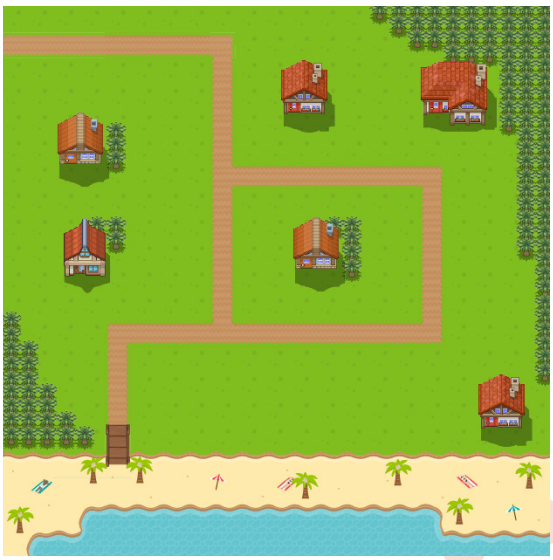


Une fois nos ébauches terminées nous avons pu passer à la création. Nous avons choisi d'utiliser le logiciel **Illustrator** qui permet de créer des **images vectorielles.**

Création de la carte



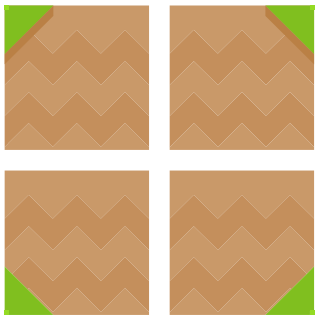
Premièrement nous avons **construit notre fichier.itd,** avec le chemin et les différents noeuds. La couleur bleue a été rajoutée sur le fichier ITD pour **avoir des zones non-constructibles,** puis nous avons créé notre carte.



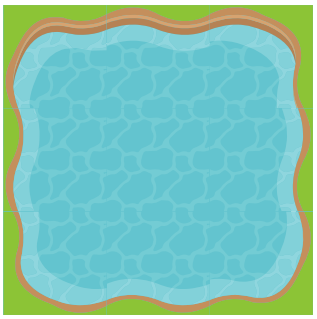
Game Design

Chaque élément de la carte est **une tuile** que nous avons créé de **dimension 30x30 pixels**. Nous avons choisi des **couleurs vives** afin de respecter le **thème** que nous nous sommes imposés **(plage).** De plus, nous voulions apporter un **côté ludique et estival** à notre jeu.

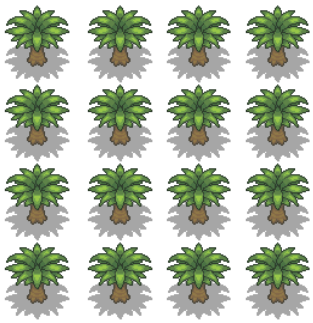
Pour les chemins



Pour la mer

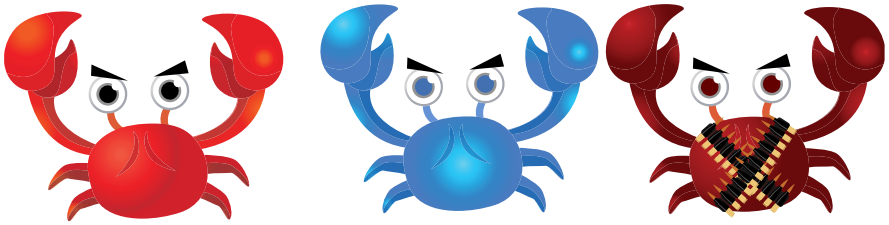


Pour les arbres



Création des sprites

Puis nous avons créé les **différents sprites** qui composent le jeu tels que **les palmiers, les bâtiments et les monstres :**



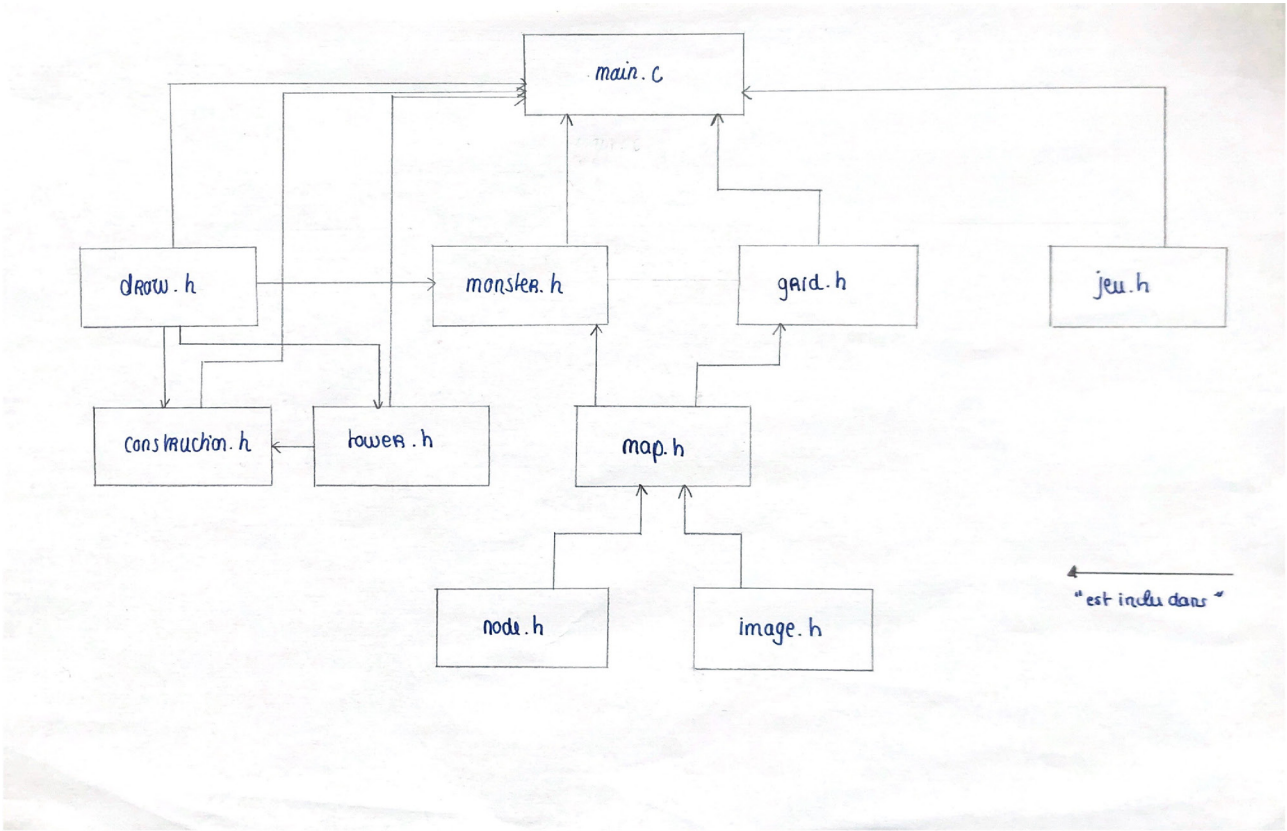
Architecture de l'application

Découpage de notre application

Pour **découper** au mieux notre application, nous avons décidé de répartir notre code en plusieurs fichiers .c et .h. **Chaque fichier** de notre code **constitue un élément du jeu** et **les méthodes** qui lui sont attribuées.

Par exemple, nous allons avoir **un fichier .c et .h** Tower qui **regroupe** la **création** des tours, la **sélection** des tours, **leurs affichages et leurs suppressions.** D'autres fichiers comme node.c, node.h et grid.c grid.h vont permettre de **gérer toute la structure** de notre jeu, à savoir créer **notre graphe de noeuds et décomposer notre carte en grille.** Les fichiers draw.c et draw.h **rassemblent plusieurs méthodes d'OpenGL** qui affichent à l'écran les différents éléments de notre jeu. Enfin notre fichier Map s'occupe de toute la **partie lecture de notre fichier itd** ainsi que de la **création de notre chemin.**

Voici un petit schéma pour mieux comprendre :



Les structures de données

Nous avons **séparé notre jeu** en différentes **structures de données :**

Certaines structures étaient utilisées pour la construction des **différentes entités du jeu :**
- structure monster
- structure tower
- structure construction

D'autres étaient utilisées pour la **construction du jeu :**
- structure jeu
- structure map
- structure node et ses successeurs
- structure caseTab

Les structures entités

La structure Tower

Notre première entité créée était **l'entité tour.** Afin de répondre à tous les besoins du jeu, nous avons dû rajouter plusieurs attributs lors de la conception du jeu. La structure Tower est composée de :
- Un index (utile pour retrouver notre tour une fois placée)
- Sa position X et Y en pixel
- Son type
- Sa puissance, sa portée, sa cadence et son coût en fonction de son type
- La tour suivante (pour créer une liste de tours)

Nous avons également créé une **structure pour le type Tower** afin de **retrouver** plus facilement **le type de notre tour.** Nous avons **4 types de tours** possibles : Rocket, Laser, Mitraillette, Hybride.

La structure Construction

Sa structure ainsi que ses méthodes **ressemblent beaucoup à la Tower**. Malheureusement nous ne sommes **pas en C++** du coup nous ne pouvions **pas faire d'héritage** avec une classe entité dont hériterait notre structure Tower et Construction. La structure Construction est composée de :
- Un index (utile pour retrouver notre tour une fois placée)
- Sa position X et Y en pixel
- Son type
- Sa portée et son coût en fonction de son type
- La construction suivante (pour créer une liste de constructions)

Nous avons une **structure pour le type Construction** et nous avons **3 types de constructions** possibles : Radar, Usine, Munitions.

Les structures de données

La structure Monster

Similaire à la structure de la Tower, elle possède **néanmoins d'autres attributs** qui permettent de faire **déplacer** le monstre, par exemple. La structre Monster est composée de :
- L'index du monstre ainsi que de l'index de sa position au début du jeu.
- Ses points de vie et ses points de vie maximum en fonction du type
- Son type
- Son gain, sa résistance, sa vitesse en fonction du type
- Sa position x et y
- Le noeud où il se dirige
- Le monstre suivant (pour créer une liste monstres)

Nous avons une structure pour le type Monster et nous avons **3 types** de monstres possibles : Rouge, Bleu, Massacreur.

Les structures pour la construction du jeu

La structure Jeu

Elle permet de **centraliser toutes les informations** liées à l'interface du jeu, au même endroit. La structure est composée de :
- des int pour le démarrage, mettre en pause, quitter, gagner ou perdre une partie
- d'un int pour afficher l'aide
Elle n'est pas reliée à un fichier.c et elle n'a **pas de méthodes** associées.

La structure Map

La structure Map permet de **récupérer toutes les données lues lors du fichier de l'ITD.** Elle est très importante pour la construction de la map et de la structure du jeu. La structure Map est composée de :
- du nom de la carte qu'on lit
- du nombre de noeuds
- de notre liste de noeuds
- de la couleur pour l'entrée, la sortie, le chemin, les noeuds et les zones constructibles

Elle utilise également la **structure Color** qui **permet de stocker une couleur** à partir des paramètres rouge, vert et bleu.

La structure Node et ses successeurs

La structure Node permet de **récupérer nos noeuds** lors de la lecture du fichier ITD. Elle est également **reliée à la structure AdjacentNode** qui permet de **créer une liste des successeurs de notre noeud.** La structure Node est composée de :
- un index
- le type de notre noeud
- la position x et y de notre noeud

- le noeud suivant dans notre liste de noeuds
- la liste des successeurs du noeud

La structure CaseTab

La structure CaseTab permet de **décomposer notre image en une grille de plusieurs case de type CaseTab.** La structure CaseTab est composée de :
- la position x et y de notre case dans notre tableau à dimension
- le type de notre case (si c'est un chemin, un noeud, une zone constructible...)
- videTower qui permet de vérifier si une tour est déjà posée sur notre case
- videBat qui permet de vérifier si un bâtiment est déjà posé sur notre case

« Résultats » obtenus

Dans une partie

Lors du lancement de notre jeu, **aucune latence** n'est apparente.

Voici les différentes captures d'écran :

On observe bien les différents monstres qui arrivent depuis l'entrée. Leur **barre de vie s'affiche correctement** et une **tour a été posée.**

« Résultats » obtenus

Sur cette deuxième capture d'écran, on observe bien le **changement de monstres** : ils sont devenus bleus et sont plus puissants. On peut également voir qu'un bâtiment de type USINE **a été posé** afin d'améliorer certaines caractéristiques de la tour.

On peut remarquer que le **nombre de vagues (en haut de l'écran) a augmenté.** De plus, le joueur ayant posé des tours, son **nombre de noix de coco a diminué.**

Finalement, sur la dernière capture on peut voir **l'affichage des informations d'une tour posée.**

Nos difficultés et solutions

1. Vérification du fichier ITD

Nous avons eu **quelques difficultés pour lire le fichier ITD**. Nous devions réfléchir à **comment stocker** toutes les informations dans des structures pour pouvoir les utiliser plus tard. De plus nous devions vérifier ligne par ligne le fichier et vérifier également la position des noeuds par rapport à l'image.

Solution : la **création de nos structures** a été très importante pour la lecture et la vérification du fichier ITD. Notre structure Map nous a permis de **regrouper toutes les données lues.** Notre structure Node nous a permis de créer notre graphe pour ensuite réaliser l'algorithme du plus court chemin.

2. L'algorithme du plus court chemin ou algorithme de Dijsktra

Il nous a été demandé de **chercher,** dans notre graphe, le **chemin le plus court.** Pour réaliser cette demande nous avons dû réfléchir à comment trouver le plus court chemin. Nous nous sommes intéressées alors à **l'algorithme de Dijsktra,** qui permettait de trouver le plus court chemin dans un graphe. Nous avons dû comprendre l'algorithme avant d'essayer de le coder. La compréhension de l'algorithme a été plus facile que la mise en code.

Solution : Après plusieurs réflexions, nous avons pu mieux comprendre l'algorithme et les arguments dont nous avions besoin dans notre code. Nous avons essayé de reproduire au maximum chaque étape de l'algorithme. Le **résultat marche mais n'est pas optimal** car nous n'augmentons pas la valeur des chemins en fonction du danger (présence de tours).

3. Problème pour le positionnement d'objet

Nous avions du **mal à positionner les différents éléments** à l'écran (notamment sur notre carte).

Solution : Pour éviter tout problème d'affichage et pour faciliter l'utilisation, nous avons décidé de **placer notre repère entre 0 et 1.** Notre carte faisant 630 pixels de hauteur par 630 pixels de large, cela donne 1x1 à l'échelle de notre carte.

Pour aller plus loin

D'une autre manière

Nous avons choisi de réaliser notre logiciel en **langage C** car nous avions encore du mal **avec le langage C++.** Le C++ nous aurait offert **plusieurs avantages** comme la création de vecteurs.

Du côté des améliorations, elles sont **quasiment infinies.** Nous pouvons **ajouter différents niveaux** (augmentant la difficulté), **ajouter différentes cartes, améliorer les projectiles, ajouter du son** sur chaque tir ou encore **créer des sprites de monstres et palmiers animés.** Nous pouvons également ajouter une fonctionnalité permettant au **monstre de changer de chemin lorsque le danger est trop important.**

Conclusion

Malgré les difficultés que nous avons rencontrées nous sommes **satisfaites du résultat.** Nous avons essayé de **remplir toutes les contraintes du projet** tout en allant plus loin en **proposant d'autres fonctionnalités** telles que l'affichage des barres de vie.

Ce projet nous a permis de **mieux comprendre le fonctionnement d'un jeu** entre la partie infographie (comprenant le game design) et la partie algorithmie, ainsi que de comprendre l'ampleur d'un logiciel tel qu'un Tower Defense.

Il nous a aussi permis de **mettre en application toutes les connaissances apprises** au cours de cette première année au sein de l'IMAC.

Remerciements

Nous tenons particulièrement à **remercier nos professeurs** qui nous ont permis de **mener à bien ce projet** de part **leur disponibilité** et **leurs précieux conseils.**