Projet
Algorithmique et openGL :
IMAC Tower Defense

Par Maéva ROSENBERG et Zoé DURAND

Semestre 2, IMAC 1, juin 2019

SOMMAIRE

Introduction :

Sans que personne ne sache comment ni pourquoi, des virus ont réussi à se matérialiser sous forme physique, êtres de métal dont le seul but est de nuire à la race humaine.

Vous jouez un citoyen du pays IMAC, pays membre de l'Union Internationale Esipienne dont les autres pays membres ont déjà perdu contre les virus. Vous êtes chargés de protéger votre pays, en utilisant une forme de technologie particulière, insensible aux virus et se rapprochant de la "magie" selon certains. Sous la forme de tours et de bâtiments, ces installations de défense peuvent être construits au bord des chemins sur lesquels les virus se propagent en cherchant à atteindre le quartier général de la défense humaine. Mais les virus sont malins : ils prendront le chemin le moins dangereux pour eux. A vous d'organiser vos défenses de manière efficace...

Tel est le pitch du jeu Union Internationale Esipienne VS Virus, un jeu de type Tower Defense développé dans le cadre des cours d'algorithmique et de synthèse d'image en IMAC1.

Sur une carte représentant la zone à défendre, plusieurs chemins sont reliés entre eux par des noeuds. Les virus, nommés monstres dans nos fichiers, se déplacent sur le chemin qui semble être le plus sûr pour eux au moment de leur apparition, leur destination finale étant la zone de sortie correspondante au quartier général humain.

Les zones entre ces chemins sont constructibles, c'est-à-dire qu'elles peuvent accueillir des tours qui attaqueront tous les monstres à leur portée et les bâtiments qui peuvent améliorer une caractéristique des tours à leur portée (cadence, puissance ou portée).

Chaque tour et chaque bâtiment peuvent être améliorés ou supprimés. Bien évidemment, construire et améliorer une installation a un coût : le joueur peut renflouer ses caisses en tuant les monstres ou en supprimant une installation.

Le jeu se termine si un monstre atteint la sortie, ce qui signe la défaite du joueur et de l'Union Internationale Esipienne, ou si le joueur a réussi à tuer tous les monstres qui ont osé s'attaquer à son pays, marquant la victoire de l'humanité.

Présentation de l’application

"Bonjour et bienvenue dans cette édition spéciale défense antivirus. Cela fait maintenant 47 jours que des virus intelligents s'attaquent à notre civilisation en détruisant tout sur leur passage. Face à la panique de la population, les gouvernements de nos voisins de l'Union Internationale Esipienne tombent tous les uns après les autres, et le peuple de l'IMAC s'inquiète de plus en plus. Cependant nous avons également de bonnes nouvelles puisque notre suprême leader Vencesclas Biri a annoncé ce matin qu'une équipe de chercheurs vient de découvrir une technologie très ancienne, je cite "similaire à de la magie et immunisée contre les virus".

Tout de suite, notre repor... Ah ? Attendez, on vient de m'informer que des virus ont été repérés à la frontière nord de l'IMAC... Sud ? à la frontière sud pardon... Comment ? Partout sur les frontières ? Ah... oui -*se racle la gorge*- Mesdames et Messieurs, le gouvernement vient de décréter l'état d'urgence suite à l'entrée simultanée de nombreux virus à plusieurs frontières de l'IMAC. Nous vous demandons de rester chez vous et d'attendre de prochaines instructions. -biiiiiiiip-".

Cet enregistrement, accompagné d’une musique originale, était censé marquer le début du jeu Union Internationale Esipienne VS Virus, plaçant le joueur dans le contexte et l’ambiance du jeu dès le départ. Citoyen de l’IMAC, le joueur est chargé d’utiliser une forme de magie pour repousser les virus qui s’attaquent à son pays, avant que ceux-ci n’atteignent le quartier général humain. (fichier du son dans le git : “”)

Apparence

Le jeu se présente sous la forme d’une fenêtre de taille 1800 * 1012 pixels, les monstres apparaissant dans la zone d’apparition en bas à droit et doivent rejoindre la sortie en haut à gauche. L’interface du jeu se présente en deux parties : en bas se trouve le menu où toutes les installations sont affichées, permettant au joueur de cliquer sur celle qui l’intéresse pour ensuite la poser sur la carte et en haut à droite se trouve une zone d’affichage permettant de donner des informations sur les installations sur lesquels le joueur clique.

Afin de rendre le jeu plus vivant et d’en améliorer la dynamique, les installations et les monstres sont animés : une classe sprites affiche le sprite voulu sur une image au format png comportant tous les sprites d’un élément, en changeant régulièrement afin de donner une impression de mouvement (par exemple, les monstres peuvent avoir un morceau de leur corps qui clignote).

Les cartes

Deux cartes sont jouables à l'heure actuelle : carte1.itd et carte2.itd. La première est très simple puisqu'elle ne comprend que 4 noeuds et donc un seul chemin possible : c'est celle qui nous a servi de support pour implémenter la majorité des fonctions de base, notamment celles concernant la récupération des noeuds et le calcul des chemins, avant de les tester sur une carte bien plus complexe.

La carte 2 a beaucoup plus de chemins et de noeuds (14 noeuds), ce qui permet d'observer la capacité des monstres à adapter leur chemin en fonction des tours existantes et des risques encourus sur un chemin.

Jouabilité

Lorsque le jeu se lance, les monstres commencent très vite à apparaître. Il faut donc dès le départ aller chercher une tour dans la liste d'installation en bas de la fenêtre, puis cliquer pour poser la tour où l'on souhaite. Celle-ci apparaît avec sa portée qui s'affiche sous la forme d'un cercle.

Après avoir posé la tour, elle se chargera elle-même de tirer sur les monstres qui se situent à sa portée. Deux tours dont les portées se croisent tirent ensemble sur les monstres mais à leur cadence respective.

En haut à droite apparaît l'argent disponible pour le joueur, le niveau de la dernière installation sur laquelle il a cliqué et les boutons augmenter et supprimer.

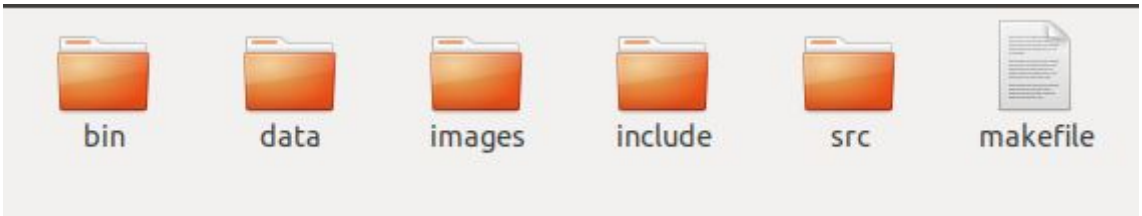
Le joueur peut également mettre le jeu sur pause à tout moment, en appuyant sur la touche *p*. Mettre le jeu en pause empêche également le joueur de poser une tour, mais il peut s'en servir pour réfléchir à une stratégie.

Le jeu s'arrête automatiquement si le joueur a gagné ou perdu, il peut alors relancer le jeu pour y rejouer.

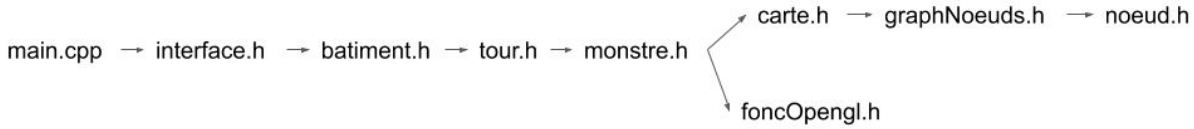
Architecture

Organisation des fichiers

L'architecture de l'application correspond à celle demandée par le sujet du projet : les dossiers bin, data, image, include et src, séparés contiennent respectivement : l'exécutable, les fichiers .itd, les fichiers .ppm, les fichiers .h, les fichier .cpp. Enfin un makefile permet de compiler le tout en respectant cette hiérarchie.



Et plus précisément, voici l'ordre d'inclusion des fichiers :



Une fois compilé, le jeu se lance par la commande *./bin/itd [fichier itd]*, le fichier itd étant par exemple carte1.itd ou carte2.itd. Les fonctions de carte.cpp s'occupent d'aller chercher le fichier itd dans le bon dossier (/data), puis de le lire pour en extraire toutes les informations nécessaires au déroulement du jeu :

- le nom de l'image ppm, dans le dossier /images, qui sera elle-même enregistrée sous la forme d'un vector pour permettre à différents fichiers d'accéder aux pixels de la carte
- les couleurs associées aux différents éléments de la carte (entrée, sortie, noeud, chemin, zone constructible)
- tous les noeuds, qui sont récupérés pour construire automatiquement les graphes servant aux monstres à choisir leur chemin

Structures de données utilisées

Nous avons décidé de programmer notre ITD dans le langage C++ car la programmation orientée objet nous paraissait toute indiquée pour la gestion des différentes entités.

La première classe appelée par le jeu est la classe Carte, qui en récupérant l'argument passé dans la ligne de lancement du jeu va vérifier si le fichier .itd est correct et en même temps récupérer toutes les informations à l'intérieur du fichier, informations essentielles pour le bon déroulement du jeu. Cette partie est un peu longue, car la totalité de l'image ppm correspondante à la carte est enregistrée dans un <vector>, ce qui fait gagner beaucoup de temps pendant le déroulement du jeu, notamment pour vérifier si une zone est constructible avant même de poser une installation.

Le jeu crée ensuite une interface de la classe Interface. C'est cette classe qui va permettre de cliquer sur la tour ou le bâtiment que l'on veut poser, puis afficher les informations liées à une tour ou un bâtiment précis et augmenter ou supprimer ces derniers. Toutes ces fonctions sont implémentées et fonctionnelles, malheureusement l'interface n'est pas "jolie" car nous voulions la dessiner nous-même et nous avons été prises de court par le temps nécessaire à l'implémentation du jeu complet. Sur ce projet, l'accent a été mis sur la jouabilité.

Toutes les fonctions orientées affichage sont dans foncOpengl.cpp, qui contient notamment la classe Sprite. Cette classe permet de prendre un sprite et, en lui donnant la largeur et la hauteur de l'image et celles d'un seul sprite sur l'image, de se déplacer sur le sprite afin de donner une impression d'animation. Cette technique est appliquée sur trois de nos tours et les trois monstres, les autres sprites n'ayant pas encore été dessinés.

Nous avons donc créé les classes abstraites Monstre, Tour et Batiment possédant chacune la descendance adaptée (PetitMonstre, MoyenMonstre, GrosMonstre pour les Monstres par exemple). Ces classes sont définies dans les fichiers portant leur noms (un .cpp et un .h pour chacun) qui contiennent également les fonctions liées aux tableaux de monstres, tours et bâtiments.

Nous avons également créé une classe noeud pour gérer au mieux la création du graphe de noeuds, et ainsi le déplacement des monstres, un type carte pour permettre une analyse et un enregistrement plus simple des éléments du fichier .itd ainsi qu'une simplification de l'affichage de la carte. Les dernière classes que nous avons créées sont la classe interface permettant la gestion des interactions avec le joueur et la classe sprites permettant l'affichage des sprites animés de nos entités.

Fonctionnalités

Règles du jeu

Les règles du jeu sont relativement simples et ont déjà été évoquées plusieurs fois, mais en voici la liste exhaustive :

- Le joueur doit défendre une carte contre des monstres qui apparaissent par vagues
- Pour se défendre, il dispose de 4 types de tours possédant chacunes leur propres propriétés (puissance, portée, cadence, cot) :
    - Les tours jaunes alias Tour Dorée
    - Les tours bleues alias Tour Argent
    - Les tours vertes alias Tour Emeraude
    - Les tours rouges alias Tour Ruby
- Une installation ne peut être posée que sur une zone constructible, c'est-à-dire une zone qui n'est ni un chemin ou un noeud, ni déjà occupée par une autre installation
- Les caractéristiques d'une tour (sa cadence, sa puissance ou sa portée) peuvent être augmentées par la proximité d'un de ces bâtiments :
    - Les radars
    - Les usines d'armements
    - Les stocks de munitions
- Chaque installation (tour et bâtiment) peut être améliorée dans une limite de 5 niveaux, ou supprimée
- Le joueur a une somme d'argent au départ du jeu pour construire ses premières tours, puis il peut récupérer de l'argent en tuant les monstres ou en supprimant une installation. La suppression d'une installation rapporte moins que ce que sa construction a coûté.
- Les monstres sont dit "intelligents", c'est à dire qu'à leur apparition, ils déterminent le chemin le moins dangereux pour eux, c'est-à-dire celui où il y a le moins de tours.
- Le jeu est gagné lorsque tous les monstres sont morts avant d'avoir atteint leur objectif. Il est perdu lorsqu'un seul monstre arrive à la zone de sortie.
- Le jeu peut être mis en pause, ce qui immobilise les monstres mais empêche également le joueur de construire une installation

Les différentes cartes

La carte 1 (voir fig. 1) est une carte simple qui, comme expliqué plus haut, nous a permi d'implémenter la plupart de nos fonctions. Elle comport seulement quatre noeuds et un seul chemin possible pour les monstres (ici leur intelligence n'est donc pas importante).

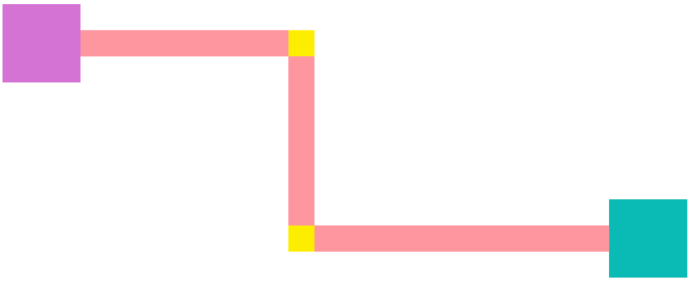


fig 1. carte1.ppm

La seconde carte est plus intéressante pour le joueur car les possibilités de chemins sont multiples et la prévision des déplacements des monstres est plus compliquée. Cette carte comporte 14 noeuds et 16 morceaux de chemins différents. Cette carte nous a permit de tester et corriger l'algorithme du meilleur chemin pour nos monstres.

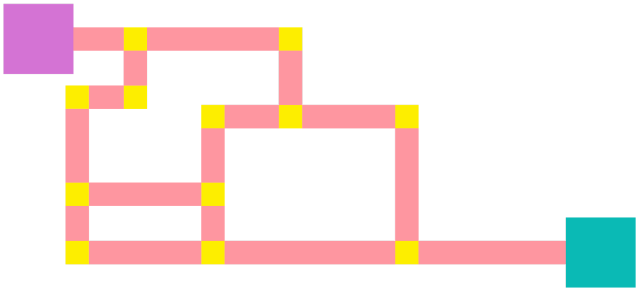
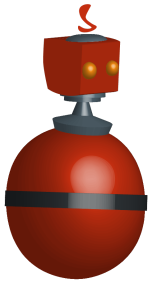


fig 2. carte2.ppm

Les éléments du jeu

Les Monstres :

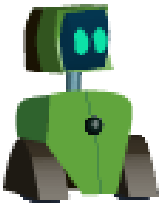


Les gros monstres : BOT2UL ou Chulmie ou Patapon

Oulah hup, boule qui pup

Vie : 500 **Vitesse :** 1/2

Tour recommandée : tour Rubis

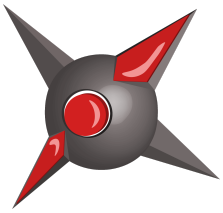


Les monstres moyens : E-Wall

A ne pas confondre avec l'éboueur de Pixar

Vie : 250 **Vitesse :** 1

Tour recommandée : tour Emeraude



Les petits monstres : Spikee ou SS4PIK

Je fais mal si on me marche dessus

Vie : 100 **Vitesse :** 2

Tour recommandée : tour or ou argent

Les Tours

La tour jaune : tour or

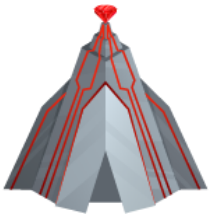
Puissance : 20 **Portée :** 110 px
Cadence : 1/150 **Coût :** 5

La tour bleue : tour argent

Puissance : 30 **Portée :** 150 px
Cadence : 1/75 **Coût :** 10

La tour verte : tour émeraude

Puissance : 50 **Portée :** 130 px
Cadence : 1/100 **Coût :** 15



La tour rouge : tour rubis

Puissance : 150 **Portée :** 170 px
Cadence : 1/500 **Coût :** 20

Les Bâtiments :

Le Radar :

Portée : 130 **Effet :** Augmente la portée des tours à proximité
Coût : 10

L'usine d'armement :

Portée : 130 **Effet :** Augmente la puissance des tours à proximité
Coût : 10

Le stock de munition :

Portée : 130 **Effet :** Augmente la cadence des tours à proximité
Coût : 10

CONCLUSION

Nous aimerions continuer à travailler sur le jeu car le fait de tout créer par nous-même est une expérience intéressante, bien que compliquée à gérer dans le temps qui nous était imparti. Bien que visuellement assez simple, le jeu est jouable et les fonctions demandées dans le sujet y sont.

Parmis les améliorations possible, il y a déjà le fait de finir tous les graphismes afin que l'ambiance du jeu corresponde au pitch d'origine, et également d'avoir une interface graphique. Il faudrait également tester différentes valeurs pour les caractéristiques des tours et des monstres, afin d'en faire un jeu agréable sans pour autant qu'il soit trop facile.

Nous aimerions également ajouter de la musique, celle-ci ayant déjà été composée pour le jeu.

Pleins d'autres améliorations sont imaginables, comme de nouveaux monstres, de nouvelles tours, la gestion de l'énergie et des centrales… On pourrait également avoir des ambiances et des monstres différents selon les cartes.

La principale difficulté de ce projet a été d'apprendre en même temps à utiliser GIT, que nous ne connaissions pas. La quantité de commits sur le projet GIT n'est pas du tout proportionnel au travail de chacune car, devant un conflit, on a souvent préféré s'échanger localement nos fichiers. Mais le fait d'avoir eu des soucis en utilisant GIT nous a forcé à apprendre à l'utiliser, ce qui nous sera probablement utile pour de futurs projets également.

Que ce soit sur la partie algorithmique ou synthèse d'image, les défis ont été divers et variés mais le fait d'avoir réussi à les surmonter est une satisfaction personnelle pour toutes les deux.