Université Gustave Eiffel

LEBLANC Travis
BLONDEL Lucas

Rapport Projet de Programmation C

Sommaire

I. Le projet

Présentation

L'objectif du projet était de développer une application graphique reprenant le concept du jeu Gemcraft : un tower defense.
Le joueur doit alors protéger son camp des monstres qui arrivent du nid à l'aide de tours qu'il pose sur le terrain et de gemmes qui permettent de tirer depuis les tours. Il a a sa disposition une quantité de mana qui permet de réaliser toutes les actions du jeu mais qui sert aussi de barre de vie : si le joueur n'a plus de mana c'est la fin de la partie !

La difficulté du projet est alors de développer cette application en utilisant la bibliothèque MLV et aussi en produisant du code propre et modularisé.

Compilation et lancement

On compile le projet de manière classique en utilisant la commande : *make* pour produire l'exécutable.

Puis on lance le programme avec : ./td [OPTIONS]

Manuel utilisateur

Ligne de commande et options

Comme précisé ci dessus, le projet se lance avec ./td [OPTIONS], on peut alors très bien lancer le projet dans options avec ./td mais il est aussi possible d'utiliser les options -h ou –help qui affichent sur le terminal une courte description du programme suivie d'une aide pour lancer le jeu.

Si jamais la commande est mal écrite, un message s'affiche avec un exemple de bon usage de la commande.

Après avoir exécuté la commande, la fenêtre graphique avec le jeu s'ouvre.

Le terrain du jeu est généré aléatoirement à chaque nouvelle partie.

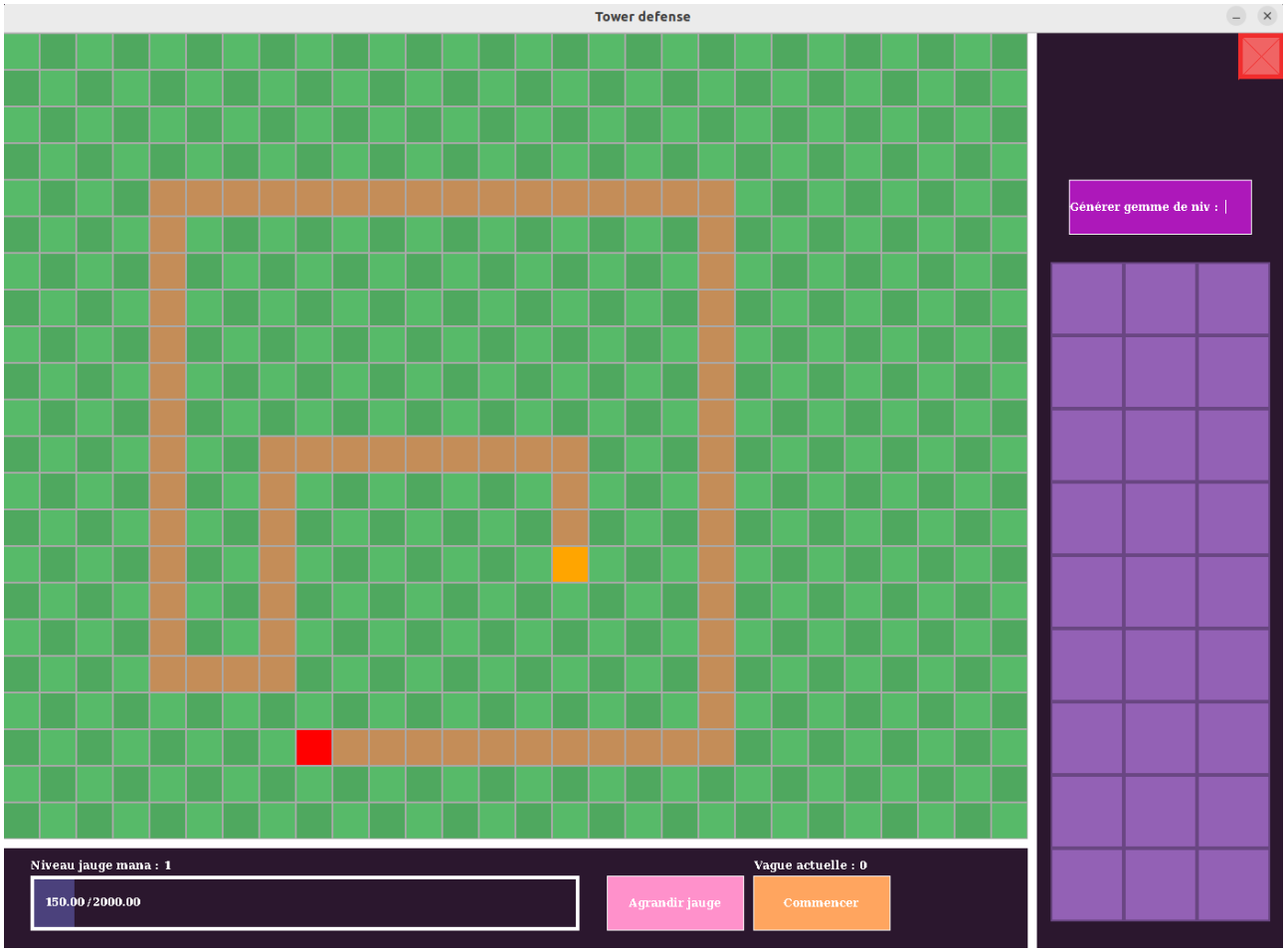

Figure 1: La fenêtre de jeu après exécution de la commande ./td

Actions du joueur

Sur la fenêtre on distingue deux zones : le terrain et le « menu » du joueur.
Depuis ce menu, le joueur peut effectuer différentes actions avant en pendant la partie :

- **Générer gemme de niv** : qui une fois cliqué permet d'écrire le niveau de la gemme à générer et qui après un appui sur Entrer, génère la gemme pur de niveau souhaité.
A noter qu'au début de la partie le joueur ne peut générer qu'une gemme de niveau 0 car le coût de génération d'une gemme est de 100x2^n et qu'il ne commence qu'avec 150 de mana. Cependant, plus aucune gemme ne sera générée si l'inventaire est plein et on ne génère que des gemmes entre le niveau 0 et 25.

- **Inventaire** : c'est là où sont toutes les gemmes qui ne sont pas dans des tours. Le joueur peut faire glisser une gemme sur une autre pour les fusionner (si elles sont de même niveau) ou encore glisser les gemmes dans les tours pour les poser ou les échanger avec d'autres gemmes qui y étaient déjà. Il peut aussi organiser les gemmes dans l'inventaire.

- **Agrandir jauge** : permet d'agrandir la jauge de mana pour en augmenter la capacité (attention très coûteux !). De même nous avons limité le niveau maximum de la jauge au niveau 200 pour ne pas casser le jeu (dans le sens facilité).

- **Commencer** / **Passer à la vague suivante** : permet de soit commencer la partie pour lancer la 1ère vague de monstre soit passer directement à la vague suivante pour gagner un certaine quantité de mana basée sur le temps restant avant la vague suivante.
A noter que les vagues apparaissent automatiquement toutes les 35 secondes.

- **Croix** (en haut à droite) : permet de quitter le jeu proprement.

Le joueur peut également réaliser des actions sur le terrain :

- **Poser une tour** : en cliquant sur une case vide du terrain, une tour vide est posée de plus les 3 premières tours sont gratuites puis le coût démarre à 100 de mana et double à chaque tour.

- **Sélectionner une tour** : le joueur peut cliquer sur une tour avec une gemme pour la sélectionner et ainsi voir la portée de tir.

- **Poser** / **échanger les gemmes** : comme dit dans l'inventaire, le joueur peut faire glisser les gemmes dans les tours pour les poser. Attention, à chaque échange de gemmes ou pose de gemme, cette dernière va mettre 2 secondes avant de tirer.

- **Retirer une gemme** : pour retirer une gemme, le joueur doit sélectionner la tour puis faire glisser la gemme en dehors de la tour pour la poser dans l'inventaire. A noter qu' il suffit simplement de faire glisser la gemme en dehors de la tour et relâcher le clic pour qu'elle se range dans l'inventaire.

Nous précisons bien qu'il faut d'abord sélectionner la tour PUIS faire glisser la gemme en dehors de la tour.

Nous ajoutons aussi que pour faire glisser une gemme, il faut maintenir le clic gauche sur la gemme et que toutes les actions se font avec le clic gauche.

Les coûts de mana pour toutes les actions (sauf la génération de gemme) sont affiché sous le pointeur de la souris.

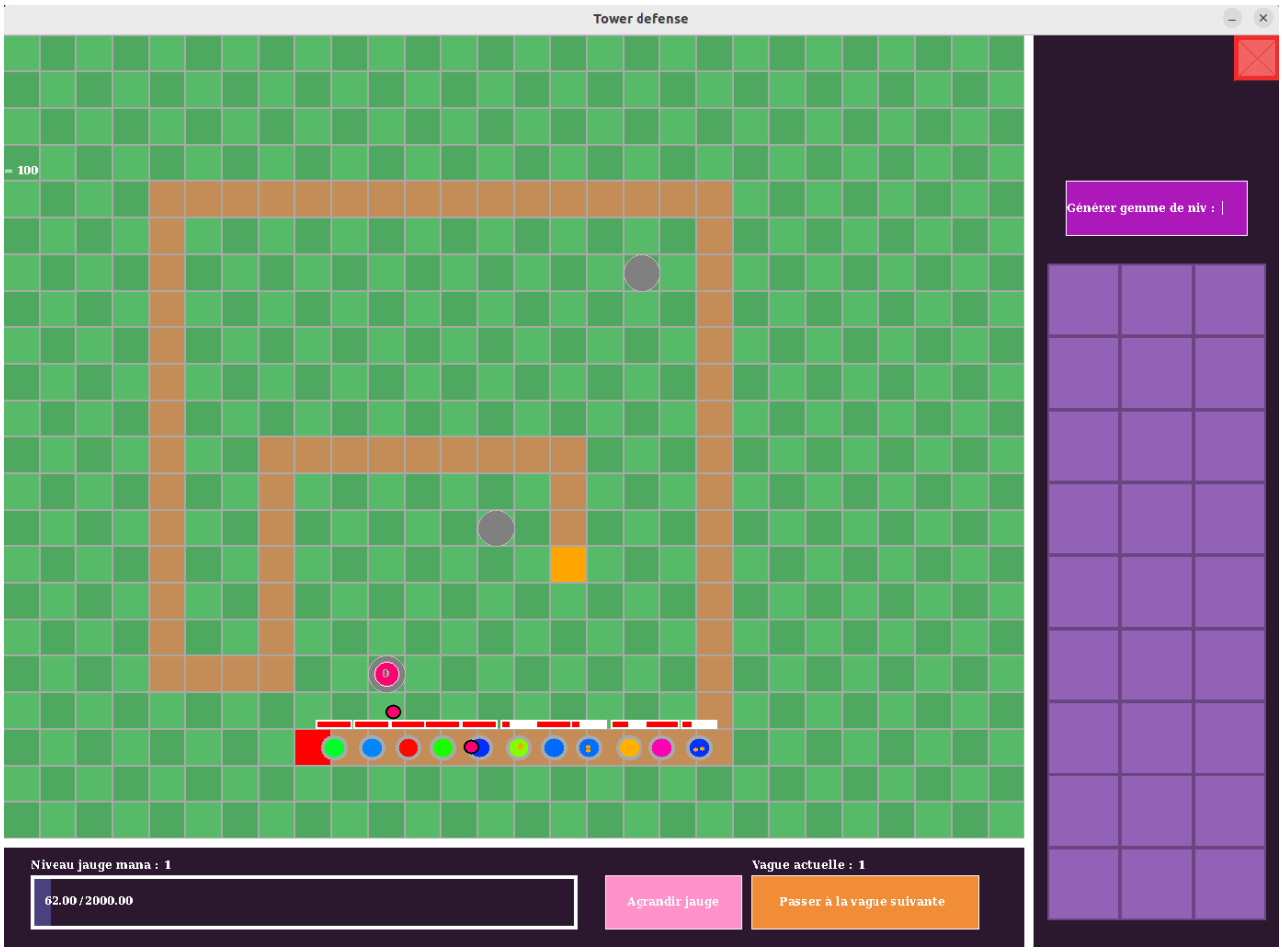

Figure 2: Une partie après avoir posé des gemmes et lancé la 1ère vague

Mécanismes du jeu

Le principe du jeu est de tuer les monstres du **nid** pour défendre le **camp** grâce aux gemmes.

Il existe 2 types de gemmes : les purs (cercles) qui ont des effet élémentaires et les impurs (carrés) qui font plus de dégâts. De plus chaque gemme possède une couleur représentant son élément et un niveau écrit au centre de la gemme.

Chaque monstre et gemmes sont d'éléments différents (pyro, dendro, hydro) et ainsi certaines gemmes feront plus de dégâts sur un monstre ayant une couleur opposée à celle de la gemme.

De plus les gemmes purs peuvent faire des dégâts élémentaires ou laisser des résidus élémentaires sur les monstres qui une fois combinés peuvent faire d'autres effets.

Le effets possibles sont :

- Pyro : génère un dégât de 15% du dégât de base aux monstres dans un rayon de 2 cases du point d'atterrissage d'un tir, en prenant la teinte en compte.

- Dendro : génère un dégât de 2.5% du dégât de base tous les 0.5 secondes du monstre pendant 10 secondes.

- Hydro : divise la vitesse par 1.5 pendant 10 secondes.

Et ceux avec combinaison de résidu :

- Vaporisation (pyro + hydro) : génère un dégât de 5% aux monstres dans un rayon de 3.5 cases du point d'atterrissage, et diviser aussi leur vitesse par 1.25 pendant 5 secondes.

- Combustion (pyro + dendro) : triple le dégât du tir courant.

- Enracinement (dendro + hydro) : immobilise le monstre pendant 3 secondes.

Chacun de ces effets sont visibles durant la partie sur les monstres touchés par les projectiles.

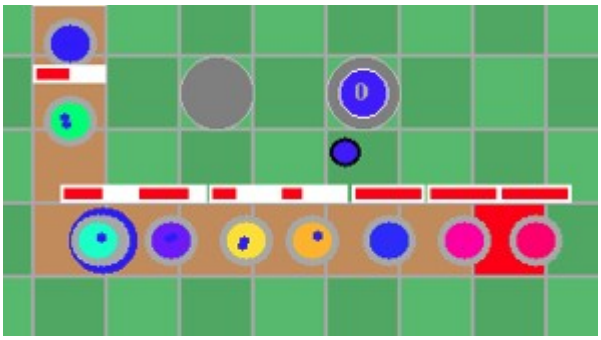

Figure 3: En bas : gemmes avec résidus hydro/ Bas gauche : gemme avec effet hydro

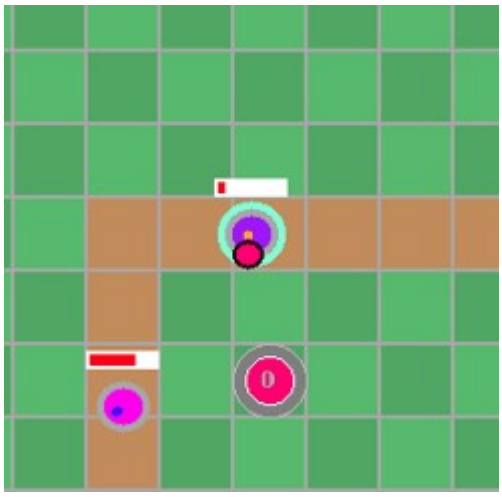

Figure 6: En bas : gemme avec résidus hydro / En haut : gemme avec effet vaporisation activé par le projectile de la gemme de type pyro

Figure 4: Une gemme impur de niveau 1 et une gemme pur de niveau 0 dans l'inventaire

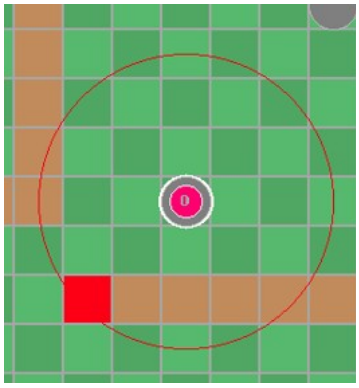

Figure 5: Une tour à gemme sélectionnée

Les monstres quant à eux peuvent apparaître sous forme de vagues de 4 types différents :

- normale : 12 monstres, vitesse d'une case par seconde
- foule : 24 monstres, vitesse d'une case par seconde
- agile : 12 monstres, vitesse de deux cases par seconde
- boss : 2 monstres, vitesse d'une case par seconde

Fin de partie

La partie se termine lorsque la réserve de mana du joueur est vide, de là un écran de fin apparaît pendant quelques secondes puis le jeu se ferme.

Figure 7: Ecran de fin de partie avec le score

II. Organisation du travail

Répartition et communication

Dans les grandes lignes, Lucas s'est chargé des modules graphique, action, monstre, terrain, element et options tandis que Travis a fait les modules animation, bittab, constantes, gemme, jeu, joueur, projectile et tour.

Il faut aussi noter que Travis a amélioré l'algo de génération de terrain et certaines actions du joueur comme glisser la gemme pour la poser / fusionner ou encore l'ajout d'animation pour certains effets comme pyro et les animations des résidu élémentaires sur les monstres. Il a aussi ajouté un module generictab car les tableaux alloués dans le projet (pour les monstres, projectiles et animations) étaient gérés de la même manière.

Pour communiquer sur l'avancement du projet et pour travailler ensemble, nous avons utilisé discord et bien sûr github.

Difficultés rencontrées

Nos plus grosses difficultés ont été d'abord de décider des structures de donnée du projet et de comment le modulariser. Durant ces 3 mois de travail il a beaucoup changé et certaines structures ont drastiquement changé.

Par exemple, nous nous sommes rendus compte de l'utilisation de liste chaînée était totalement inutile et que les tableaux étaient plus judicieux pour le stockage des données.

Le module monstre a totalement changé, à la base nous utilisions des listes chaînées pour les vagues qui elles mêmes contenaient des listes doublement chaînées pour les monstres de la vague. C'est après un cours d'amphi que nous nous sommes rendu compte de notre erreur et avons utilisé à la place un seul tableau dynamique pour tous les monstres sur le terrain.

De même nous avons condensé les informations du terrain (les tours présentes, le chemin etc..) dans un tableau à 2 dimensions dans la structure terrain plutôt que séparément ans plusieurs tableaux.

En général nous avons évité au plus d'utiliser l'allocation dynamique pour les ensembles de taille connue et fixe.

III. Conclusion

Pistes d'améliorations et bug connus

Pour ce qui est des bugs, nous avons corrigés les derniers bug début janvier et n'en n'avons pas trouvés d'autres. Par exemple nous avons limité le niveau des gemmes pour ne pas casser le jeu (notamment à cause du bouton passer à la vague suivante).

Nous ne pouvons pas affirmer cependant s'il n'y a pas d'autres bugs mais nous pensons avoir éliminés les plus important et visibles.

Plusieurs pistes d'améliorations sont possibles :

- quelques détails graphiques comme le points d'apparition des monstres, ajouter des décorations et habiller l'interface des actions etc.

- améliorer le mouvement des monstres pour le rendre plus naturel

- ajouter un tableau des scores et la sauvegarde dans un fichier

- ajouter pourquoi pas d'autres types élémentaires pour les monstres et les gemme et aussi leurs effets.

Apports du projet

Ce projet nous a permis de sortir de notre zone de confort en nous obligeant à faire un code très propre et modularisé pour un sujet qui peut être vague par endroit et qui est plutôt lourd et conséquent.

De plus il nous a permis de nous remettre en question sur plusieurs mauvaises habitudes acquises les années précédentes comme notamment la sur utilisation des listes chaînées plutôt que les tableaux et aussi d'utiliser un gestionnaire de versions comme git pour avoir une meilleur organisation.

Ainsi, c'est un bon projet avec un sujet intéressant pour mobilier toutes nos connaissances acquises et c'est une bonne conclusion de notre apprentissage en C qui s'étend depuis la L2.