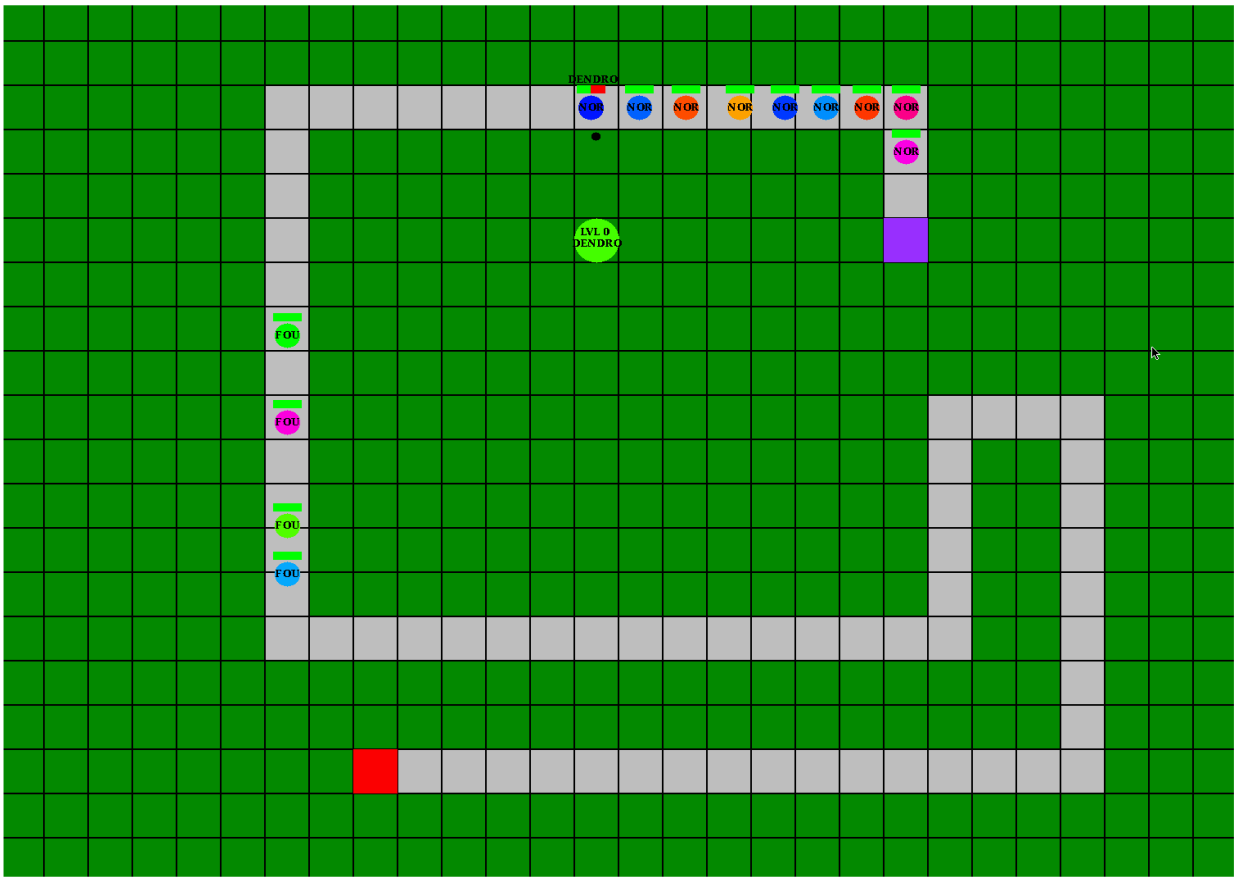
PROJET TOWER DEFENSE

SUTRA Jérémy, SY Thierno



Introduction:

Dans le cadre du projet S5 en Programmation C, nous avons du réaliser un jeu de type Tower Defense. Il consiste a fabriquer des tours, y placer des gemmes afin de tuer les ennemis tout en respectant une certaine quantité de mana.

Fonctionnement du jeu:

Tout d'abord, le programme se compile à l'aide de la commande *make* puis il faut exécuter la commande *./jeu* afin de lancer l'exécutable. Une fenêtre graphique s'ouvre ensuite et affiche le plateau de jeu ainsi que les boutons et les informations sur la partie. Tout d'abord, vous pouvez commencer par créer des gemmes ou créer une tour. La première vague ne se lance uniquement par choix du joueur. Pour créer une tour, comme indique il suffit d'appuyer sur la touche *t*. De plus, si vous maintenez *t* vous pourrez voir la zone d'action des tours. Vous devez ensuite cliquer sur une case de la grille disponible (pas une case du chemin) sinon la tour ne ne s'implémentera pas. Les trois premières tours sont offertes et ensuite le prix des autres tours évolue au fur et a mesure. Cela est indiqué sur la fenêtre graphique lorsque vous maintenez la touche. Pour pouvoir ensuite faire tirer les tours sur les ennemis, vous devez ajouter des gemmes dans les tours. Pour cela, vous devez créer des gemmes. Vous pouvez choisir le niveau de la gemme en cliquant sur les boutons UP LVL ou DOWN LVL. Le prix de la création évoluera en fonction du niveau. Une fois le bouton cliqué, une gemme apparaitra dans la liste de gemmes. En dessous est indique son type parmi, mixte, dendro, hydro, pyro. Chacun de ses types, ont un effet different. Vous pouvez ajouter la gemme dans la tour en cliquant sur la gemme. Une information verte apparaitra sur la fenêtre vous indiquant quelle gemme a été choisie ainsi que vos possibilités d'action. Vous pouvez sélectionné une tour ou bien une autre gemme de la liste. Veillez a ce que les gemmes sont de même niveau sinon la fusion est impossible. Si vous fusionnez des gemmes de type different, le nouveau type crée sera un type mixte sinon le type ne sera pas modifié. Une fois prêt vous pouvez lancer la première vague en cliquant sur le bouton LANCER VAGUE. Une fois la première vague lancée, un chrono de 35 secondes est lancée. A la fin de celui ci, une autre vague se lancera. Le temps restant est indique en bas a gauche ainsi le numéro de la vague actuelle. Vous pouvez a tout moment lancer la vague suivante en cliquant sur lancer la vague. Vous pourrez récupérer une certaine quantité de mana en fonction du temps restant avant la prochaine vague. Vous pouvez ensuite augmenter le niveau de réserve de la mana en cliquant sur le bouton AUGMENTER LA RESERVE. Cela vous permettra de stocker plus de mana. Les monstres apparaissent chacun leur tour pendant une vague. Le type des monstres de la vague est choisi aléatoirement. Les boss n'arrivent qu'après la cinquième vague. Lorsque que vous frapper un monstre, selon le type de

votre gemme, cela va lui affecter l'effet. Si un effet est déjà en action, ils peuvent s'additionner pour des créer des effets spéciaux. Une fois le monstre tue, vous pourrez récupérer une certaine quantité de mana selon le niveau de la réserve ainsi que les points de vie du monstre. Si vous ne parvenez pas a tuer un monstre avant qu'il arrive a votre camp signale par une case rouge, le monstre revient au nid et une certaine quantité de mana est perdue. Si un monstre arrive au camp du joueur et que vous n'avez plus assez de mana pour courir son bannissement alors la partie est perdue. Si vous avez atteint le nombre maximum de vagues, la partie est aussi terminée. Bonne chance !!!

Modularisation:

Nous avons décidés de découper le projet en plusieurs parties inspiré par ce qui avait été indique dans le sujet du projet. Nous avions donc créé à la base du projet des modules qui gèrent les monstres, les tours, les gemmes, la mana, une qui génère le terrain et une qui gère la partie graphique. Nous avons ensuite pris la décision de rajouter deux autres modules c'est à dire, un pour les gestion des tirs et un qui gère le moteur du jeu. Cela permet d'utiliser une structure qui retourne tout les autres modules du jeu. Cette modularisation nous semblait la plus logique afin de bien découper le projet.

Choix d'implémentation :

Le projet contient une structure principale GAME regroupant toute les autres structure du jeu.

Concernant les choix d'implémentation, pour le plateau de jeu, on a décidé d'utiliser une structure Terrain comportant tableau dynamique car on ne connait pas la taille exacte de la génération du chemin. Au début, nous allouons la place pour un chemin de 85 cases car cela évite de réallouer de place pour un autre chemin. S'il est plus long que la taille minimale, on realloue une place nécessaire pour ajouter une case. La structure comporte aussi la la longueur du chemin, le nombre de virage et la case du nid. Concernant la mana, on a une structure assez simple constituée du niveau de la mana, son nombre et le niveaux maximum possible au niveau actuel.

Pour les gemmes, il y a d'abord un attribut niveau, un attribut teinte et un attribut type qui est une structure enum composée des différents types de gemme: PYRO, HYDRO, DENDRO, MIXTE, PYRO_HYDRO, PYRO_DENDRO, DENDRO_HYDRO et VIDE. A la création ou la fusion de la gemme, le type peut uniquement être parmi les 4 premiers cités. Les autres types serviront pour les affecter aux monstres. Il y a aussi une structure Liste_gemme qui correspond a la liste de gemme

que l'on peut créer. Cette structure comprend donc un tableau de 10 gemmes ainsi que le nombre de gemme actuellement présent dans la liste. Comme la taille de la liste de gemmes est fixe, lorsque l'on n'utilise plus une gemme on lui attribue une teinte a -1 afin de ne pas l'afficher.

On a créé une structure Tir représentant les caractéristiques d'un tir. Cette structure contient ses coordonnées x et y, le nombre de tirs actuel(dans les 2 dernières secondes) et deux attributs indice vague et indice monstre qui nous permettent d'avoir les indices du monstres ciblé. Si il n'ya pas de monstre ciblé, alors les indices valent -1.

Pour les tours, on a implémenté une structure Tour composée d'un attribut case composé d'une position x et y, une gemme 4 attributs timespec: *debut_tir, fin_tir, debut_gem, fin_gem*. Ces attributs servent a gérer les temps de remplacement des gemmes et du nombre de tir par seconde. Il y a aussi une structure Tir incluse qui reprensente le tir de la tour. Cette structure tour est incluse dans une structure Lst_Tour qui est un tableau de taille 541 c'est a dire le nombre de cases possibles en enlevant au minimum 75 cases. Il y a aussi un attribut pour connaitre le nombre de tour actuel.

Concernant les monstres, on a créé plusieurs structures. Tout d'abord, on a une structure Monstre comprenant une vitesse(float), la vie initiale du monstre, la vie restante, ses coordonnées x et y l'indice de la case dans laquelle ill se trouve, un effet qui peut être compose a partir des types d'effet de la structure Type_Gemme et trois indicateurs de temps *debut_effet ,fin_effet, dendro* ainsi que la valeur du dernier dégât fait par une gemme. Cela permet respectivement de calculer le temps de l'effet et de le retirer une fois la durée terminée et de calculer la valeur des prochains dégâts pour l'effet dendro. Les effets sont affiché au dessus des monstres. La structure monstres est inclu dans la structure vague qui représente une vague d'un certains nombre de monstre. Selon son type , qui est un enum représentant tout les types de monstre: Agile, Foule, Normale, Boss; le nombre de monstres sont différents. Le nombre maximum de monstre est de 24 donc pour éviter tout type d'allocation de mémoire, on a créer un tableau de taille 24. Un entier est aussi inclu dans la structure représentant le nombre de monstres présent dans le tableau et un attribut vitesse qui représente la vitesse initiale des monstres de la vague est aussi présent. Ces vagues sont donc incluses dans une autre structure appelée Lst_Vague qui prend toute les types de Vague. On a décide de faire un tableau de taille 1000 car cela permet de stocker un grand nombre de vague. On a préféré faire un tableau de taille finie car cela évite d'allouer de l'espace et d'utiliser beaucoup de mémoire lorsque l'on devra reallouer de la place pour des vagues. On a aussi un entier indiquant le nombre de vagues écoulées et un entier indiquant le nombre total de dégâts infliges aux monstres pour l'afficher a la fin de la partie.

Toute ces implémentations nous permettent seulement d'utiliser une seule structure ou l'on va allouer dynamiquement, le tableau représentant le chemin.

Pour les valeurs des constantes D pour le tir et H pour les monstres, nous avons décidés d'attribuer les valeurs respectives 8 et 10. Cela permet de donner assez de mana au joueur en début de partie sans que cela soit trop facile.

Problèmes rencontrés:

Nous avons rencontres tout de même quelques bugs durant la création du projet qui nous a forcé à modifier certaines choses. Tout d'abord, concernas le tableau de vagues de taille 1000, lorsque l'on arrive à 5 vagues de la fin et que l'on veut lancer la suivante, cela va créer une erreur de segmentation. Nous avons donc essayé de comprendre avec gdb d'ou venait cette erreur et cela a indique qu'il n'y avait plus assez de places pour y ajouter une structure vague dans la liste. Nous avons du donc limiter le nombre de vagues a 995 pour éviter cette erreur. De plus, le fait d'utiliser un *MLV_wait_millisecond* dans la boucle while du jeu fais ralentir le fait de créer une tour ou d'ajouter des gemmes. Sans cela, ces implémentations ne sont pas ralenties.

Conclusion:

Pour conclure, nous n'avons pas eu de difficultés particulière par rapport a ce sujet. Nous avons apprécié le réaliser et ensuite de pouvoir y jouer afin de faire notre record personnel. Nous n'avons pas fait d'options supplémentaires.