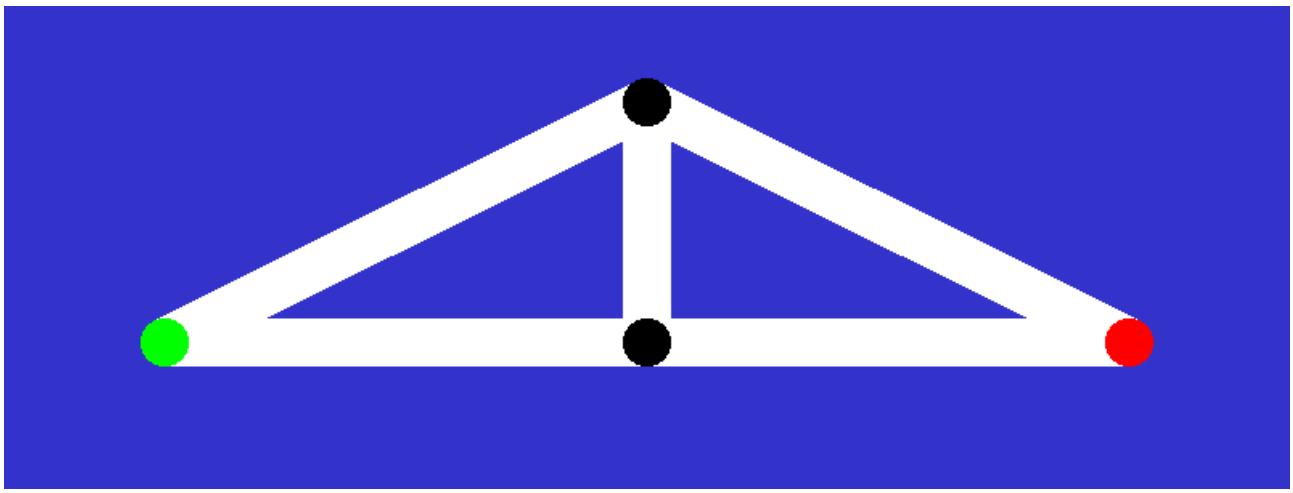
Imac Tower Defense

Ruben BRAMI , Pierre LABENDZKI
IMAC1



Sommaire :

Introduction

Nous étions très contents à l'idée de pouvoir créer, par nos propres moyens et selon nos propres choix, un jeu vidéo. Initialement les propositions fusaient dans nos têtes et les idées de création venaient naturellement, mais peu à peu la difficulté de les implémenter nous a poussé à revoir nos ambitions créatives à la baisse et à réaliser un jeu qui soit fonctionnel.

Nous avons cependant fait du mieux possible pour offrir une expérience de jeu originale malgré un visuel plutôt sobre. En effet le choix de la création de la carte est laissé à l'utilisateur, ce qui offre une personnalisation du jeu. Cette liberté dans le choix du parcours entraînait donc un design simple et efficace.

Nous avons choisi de programmer notre projet ITD en langage C, car malgré les difficultés qu'il peut présenter pour un projet de cette taille, nous nous sentions plus à l'aise avec celui-ci. Nous avons fait le choix de nous séparer le travail en deux parties : infographie et algorithmique, puis de discuter régulièrement des actions effectuées par chacun, ce qui nous a permis une grande souplesse dans l'implémentation, notamment dans le choix des règles et des options laissées à l'utilisateur.

C'est pour toutes ces raisons que nous sommes fiers de présenter notre jeu Imac Tower Defense.

Présentation de l'application

Le jeu est disponible depuis n'importe quel ordinateur sous Ubuntu, il se présente sur la forme d'une fenêtre unique, il est cependant conseillé d'avoir un terminal sous les yeux pour lire certaines informations. La fenêtre de jeu (de 800px sur 600px) présente la carte, les tours et leur rayon d'attaque, le monstres qui se déplace sur les chemins. Le terminal affiche les points de vie de l'unique monstre, l'argent disponible et le résultat de la partie : gagnée ou perdue.

L'utilisateur peut lui-même modifier le fichier `.itd` ,changer la couleur des nœuds, de l'entrée, de la sortie, des chemins, du fond, mais aussi placer les points sur toute la carte. Après une vérification que le

fichier `.itd` est bien valide, le jeu crée la carte demandée et lance une partie, où l'utilisateur doit (rapidement) placer des tours pour tuer le monstre. L'utilisateur peut aussi jouer avec une carte prédéfinie (mode par défaut).

Les tours se chargent d'attaquer le monstre si celui-ci est dans le rayon d'action de la tour.

Implémentation

Nous avons choisi d'aller au delà de ce que le sujet demandait. En effet grâce à la fonction creerCarte, il est possible (si la carte n'est pas conforme au fichier ITD) de générer une map .ppm qui soit conforme aux informations stockées dans le fichier ITD. Malheureusement nous n'avons pas réussi à créer une carte en temps réel après compilation globale (sans doute en raison d'un conflit entre scanCarte et creerCarte)

Fonctionnalités du jeu

Pour gagner la partie, l'utilisateur doit placer des tours sur le chemin du monstre pour le tuer avant qu'il n’atteigne le point d'arrivé. Pour poser des tours l'utilisateur doit :

1) Appuyer sur une touche du clavier :

j pour une tour jaune : rayon de 4m , attaque de 4pts , coût 4€
r pour une tour rouge : rayon de 10m , attaque de 10pts , coût 20€
b pour une tour bleue : rayon de 20m , attaque de 4pts , coût 8€
v pour une tour verte : rayon de 6m , attaque de 8pts , coût 10€

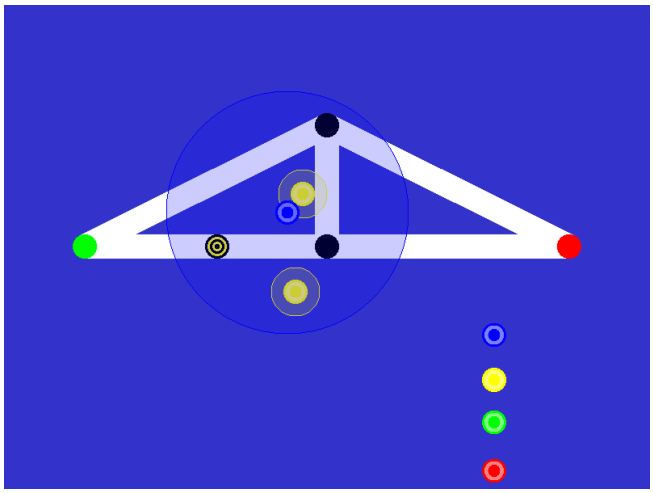
2)Cliquez sur la carte :

la tour sera de la couleur de la dernière touche du clavier enfoncée.
si le terrain n'est pas constructible, pas de construction possible,
si pas assez d'argent, pas de construction possible

L'utilisateur n'a pas à s'inquiéter si le terrain est constructible ou non, car le jeu gère lui même les terrains constructibles. Si le terrain n'est pas constructible, la tour n'est pas construite et l'argent n'est pas débité. On n'entend par terrain constructible toute zone de la carte qui n'est pas : un chemin, un nœud, la sortie ou l'entré. On veut aussi que toute la tour soit dans un terrain constructible et non pas seulement le centre de la tour.

Résultats

Le temps de chargement de la carte, c'est à dire de sa vérification et de son dessin (si l'image proposée n'est pas conforme à l'`itd`), est d'environ 10 secondes. Le reste du jeu est en temps réel.



Conclusion

Nous sommes fiers des implémentations que nous avons réalisées et du résultat final obtenu. La rapidité du jeu et la liberté des cartes engagent l'utilisateur à vouloir essayer plusieurs parties. De nombreux points sont encore à améliorer, comme l'affichage dans la fenêtre de jeu du score et de l'argent, le design des montres et des tours, les vagues de monstres ne sont pas possible et il n'y a pas de bâtiment et de gain en difficulté. Mais l'application est jouable et l'amusement a été présent de la création à la première partie jouée.

Nous avons chacun appris beaucoup sur la création d'un jeu, que ce soit la gestion de la SDL avec OpenGL, ou les algorithmes fondamentaux de la synthèse d'images.