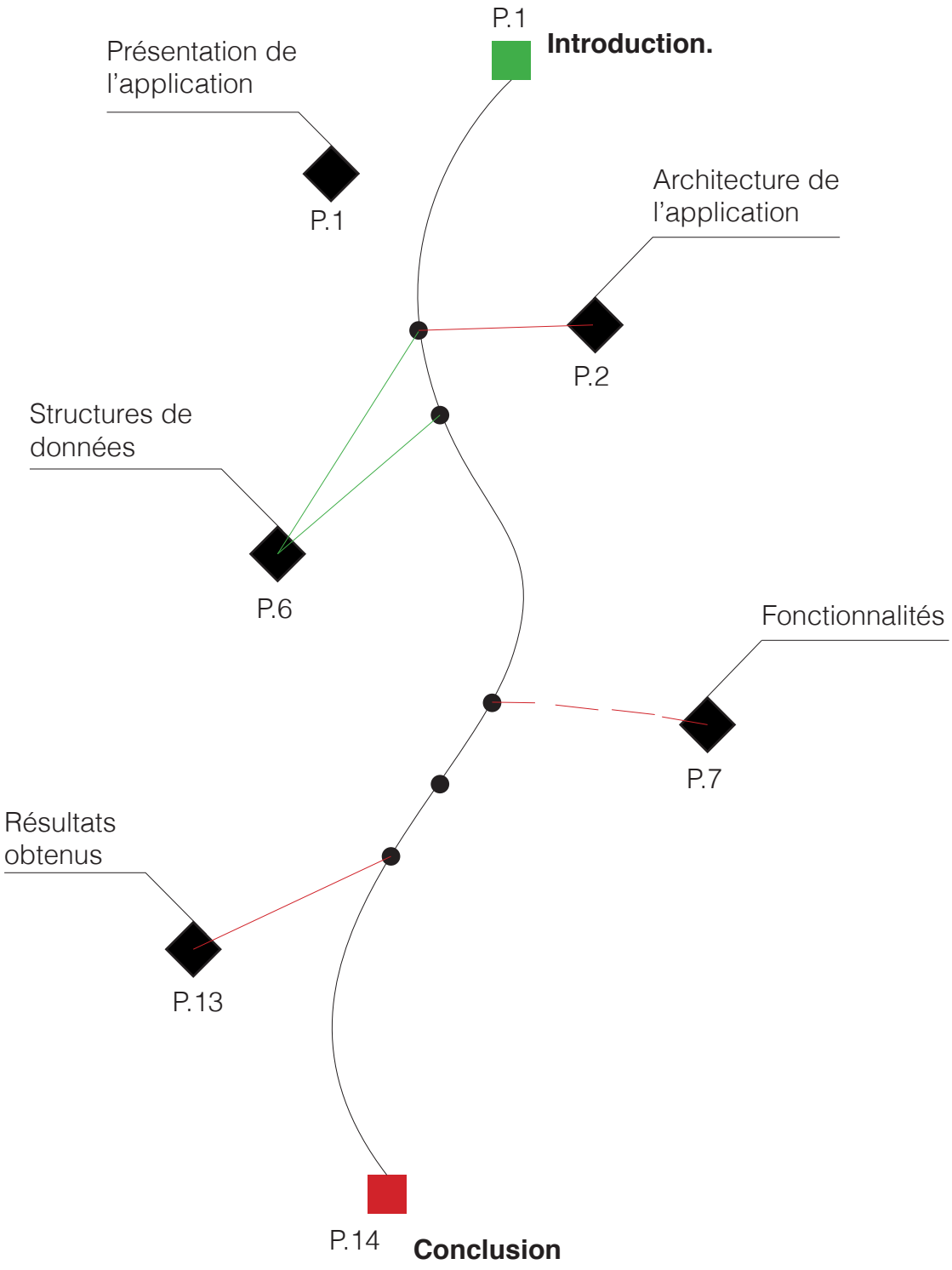
Projet semestre 2 - imac'15

Pokimac Tower Defense

VILLEDIEU MORGAN & BLONDEAU MAXIME

sommaire.itd



1 Introduction

Le projet de second semestre de première année d'école IMAC en cours de cours de synthèse d'image consiste à la réalisation d'un jeu de type "Tower Defence".
Le Tower Defense est un style de jeu de stratégie, ou le but est d'empêcher le passage de créatures d'une zone de départ vers une zone d'arrivée, en plaçant des tours de défense qui éliminent les créatures au cours de leur progression avant qu'elles n'arrivent dans ladite zone. Les vagues sont généralement de plus en plus difficiles à contenir ce qui pousse le joueur à établir une stratégie.

2 Présentation de l'application

Notre projet est réalisé en langage C, ainsi que les librairies OpenGL GLut et SDL. Nous avons utilisé des librairies complémentaires afin d'étoffer notre projet avec SDL_Image (pour la gestion des images), SDL_ttf (pour la gestion du texte), et SDL_mixer (pour la gestion du son). Au démarrage l'application demande le nom du fichier itd décrivant la carte de jeu, permettant de faire appel à l'image de fond ("Carte en fond"), au chemin (point de liaisons des noeuds et couleurs), ainsi que la couleur de zone constructible.

3 Architecture de l'application

3.1 - Arborescence .c et .h

[-] **PokimacTowerDefense**
- [+] bin
- [+] data
- [+] doc
- [+] images
- [-] include
  - carte.h
  - chemin.h
  - main.h
  - map.h
  - menu.h
  - monster.h
  - node.h
  - sdl.h
  - texte.h
  - texture.h
  - tower.h
  - vague.h
  - main.h
  - map.h
- [+] lib
- Makefile
- [+] obj
- [+] son
- [-] src
  - carte.c
  - chemin.c
  - main.c
  - map.c
  - menu.c
  - monster.c
  - node.c
  - sdl.c
  - texte.c
  - texture.c
  - tower.c
  - vague.c
  - main.c
  - map.c

3.2 - Architecture

> L'application se déroule de la manière suivante

Début

Initialisation

- Initialisation de la SDL
- Initialisation de SDL_mixer (pour l'audio)
- initialisation de SDL_ttf (pour le texte)
- On déclare les tableaux qui seront utilisés pour chacun des paramètres (construction , noeud...)
- On demande le fichier itd
- On vérifie la validité du fichier puis on ajoute les paramètres de celui-ci.
- On déclare les listes utilisées
- On déclare les musiques utilisées
- Puis on déclare toutes les variables liées au jeu

Boucle d'affichage

On entre dans la boucle du jeu (le jeu se lance à proprement dit), cette boucle se répétera de manière continue tant que le joueur n'aura pas décidé de quitter le jeu.
Toutes les étapes suivantes seront effectuées entre chaque rafraîchissement d'image et donc totalement transparentes pour le joueur.

- Affichage de la carte en fond.
- Affichage du chemin.
- Si le jeu est en cours (ni perdu ni gagné)
  - *Affichage des tours.*
  - *Gestion des vagues*
- Gestion des monstres éliminés.
- Gestion des déplacements des monstres et des tirs des tours (cette étape ne s'opère que toutes les 0.1 seconde).
- Affichage des monstres.
- Si la partie est perdue ou gagnée (un monstre est parvenu au bout du chemin ou le joueur survit à 20 vagues).
  - *Suppression des monstres et des tours.*
  - *Affichage de victoire ou défaite*
- Affichage de la barre de menu.
- Affichage du curseur personnalisé.

Boucle de gestion des événements souris

Boucle de gestion des événements clavier

FIN Boucle d'affichage

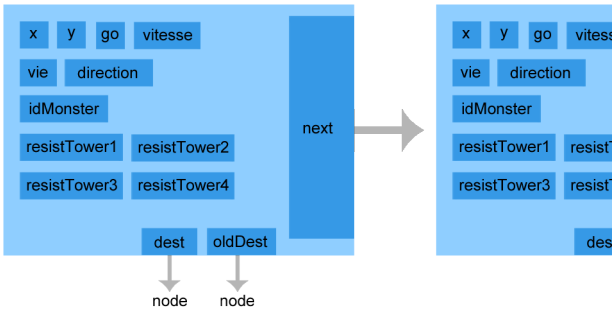
Libération des espaces mémoire

Fin

4 Structures de données

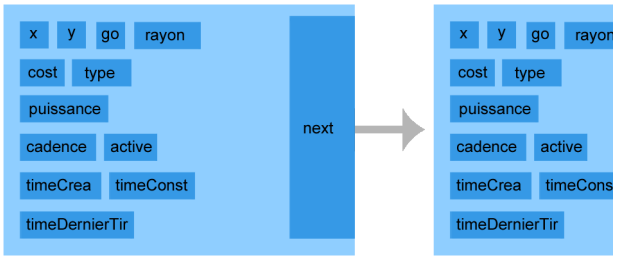
4.1 - Schéma des structures

Tower



Node

Monster

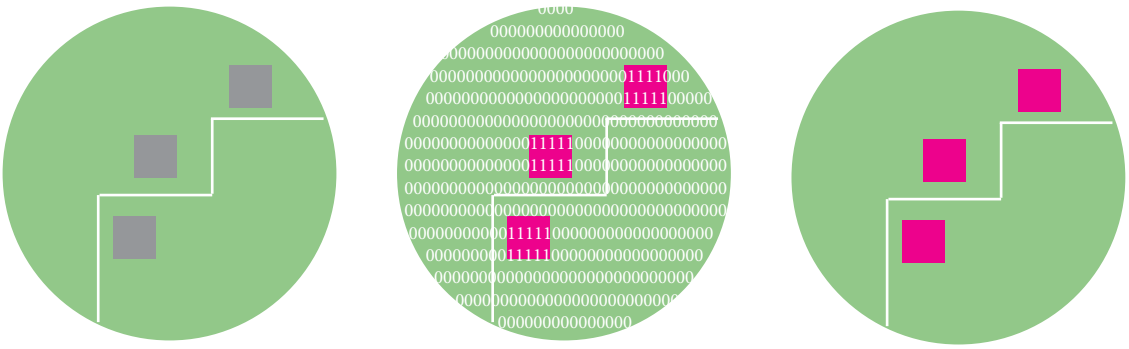


5 Description des fonctionnalités

5.1 - Re-mapping

La couleur de zone constructible indique une couleur présente dans l'image de la carte définissant les zones où l'on peut disposer des tours.
Le remmaping consiste à "colorier" ces zones d'une couleur plus agréable visuellement. Afin de garder en mémoire les zones constructibles ou non, nous créons un tableau de pixels à deux dimensions x et y, puis à l'aide d'une fonction nous parcourons la sdl_surface où est située l'image, si la couleur est magenta (couleur à re-maper, alors nous inscrivons la valeur un dans le tableau de pixels que nous avons créé précédemment, sinon nous inscrivons un 0). Ensuite il suffit de recolorer les zones constructibles dans la couleur souhaitée. Le tableau de pixels créer va ensuite nous permettre lors de l'ajout d'une tour de vérifier si l'endroit sélectionné est bien une zone constructible si à cet indice il y a un 1 dans le tableau, et inversement avec la valeur 0.

1. *On trouve les zones constructibles*
2. *On place les 1 et les 0 dans le tableau de pixel x,y*
3. *On re-map les zones constructibles*



*

(En réalité nos 1 et 0 sont définis pixel par pixel, donc la précision est parfaite contrairement à ce schéma.)

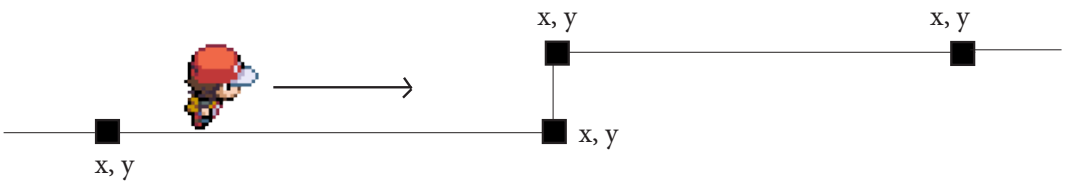
5.2 - Gestion des éléments du jeu

> *Le chemin*

Les monstres se déplacent selon un chemin défini par une suite de noeuds. Chaque noeud possède ses coordonnées (x et y) ainsi qu'un pointeur sur son prochain.
Les monstres possèdent chacun leurs coordonnées en x et en y, le noeud de provenance, le noeud de destination, leur vitesse... Chaque monstre possède un pointeur sur son prochain. Les déplacements d'un monstre consistent à le mener d'un point A (noeud d'origine) à un point B (noeud de destination). Lorsqu'il attend sa destination il prend pour noeud d'origine celui-ci et pour noeud de destination le prochain dans la liste de noeud (chemin). Le jeu recalcule les évènements de jeu à une fréquence de 1/10 ème de seconde.

> *Les déplacements*

À chaque calcule de la position on opère comme ceci : Le monstre ce voit afficher et sa nouvelle position recalculée si le monstre qui le suit dans la liste a 50 pixels d'avance sur lui (sauf pour le dernier). De cette manière c'est le dernier de la liste qui part en premier.
On calcule dans un premier temps la nouvelle position comme si le monstre se déplaçait seulement selon les x ou y, s'il y a un angle de déplacement on calcule la pente du segment [noeud d'origine ; noeud de destination] (équation de la droite), puis on y applique le déplacement en x ou y précèdent afin de connaître les coordonnées exactes sur le segment, que l'on donnera en suite au monstre pour qu'il puisse s'y placer. On passe alors au monstre suivant et on réitère ces étapes.



> *Les sprites*

Pour l'effet de mouvement du sprite, on utilise une texture sur laquelle est disposé un personnage sous plusieurs angles. À chaque appel de la fonction de déplacement, on incrémente une variable de de 1 à 4, cette variable nous permettra de nous déplacer dans la texture afin d'afficher les différents mouvements du personnage. De plus l'orientation du personnage est définie selon qu'il se déplace de haut en bas, de bas en haut...

> *Création et gestion des tours*

Pour la création d'une tour, on vérifie si elle se situe sur une zone constructible (terrain et superposition de tours) , ont créé une tour et on lui attribue ses caractéristiques (position, puissance...), puis on l'insère dans la liste de tour.
Tout les 0.1 second on calcule les tirs des tours, on procède de la manière suivante :
On calcule la différence du temps courant avec la date du dernier tir de la tour, si cette différence est supérieure ou égale à la cadence de la tour celle-ci peut tirer. On cherche le monstre le plus près de la tour dans son rayon d'action, puis on le verrouille pour lui ôter de la vie à hauteur de la puissance de la tour (en appliquant la résistance du monstre à ce type de tour). La tour de type mitraillette tant qu'elle tire et ôte de la vie à tous les monstres dans son rayon d'action. On réitère ces étapes pour les tours suivantes.

5.3 - Règles du jeu

L'objectif du joueur et d'empêcher les monstres d'atteindre l'extrémité du chemin. Les monstres apparaissent par groupe de 10 et en vagues successives. Si le joueur réussit sa mission au-delà de la 20e vague sans qu'aucun monstre ne parvienne à sortir de la carte il gagne. Si un monstre parvient à son but, le joueur perd.

Pour y parvenir, le joueur dispose d'une somme d'agent lui permettant d'acheter des tours afin de les disposer sur les zones constructibles de la carte. Vous pouvez augmenter la puissance de vos tours afin de pouvoir augmenter leurs capacités, ceci vous coutera un nombre déterminé de coins.
La portée de vos tours est représentée par le cercle blanc qui s'affiche lors du glissé déposé. d'une tour sélectionnée.

Attention, veillez à construire vos bâtiments à portée des chemins et sur des zones constructibles représentés en gris sur la carte.

Le joueur ne peut superposer plusieurs tours, chaque tour possède ses propres caractéristiques (rayon d'action, puissance, cadence, coût). Chaque monstre éliminé rapporte de l'argent au joueur afin de pouvoir acheter de nouvelles tours.

Les monstres voient leurs points de vie augmenter à chaque vague, augmentant la difficulté du jeu, de plus toutes les 5 vagues une vague "BOSS" vient rendre plus difficile la tache, car elle dispose de monstres ayant une résistance aux différentes tours.

> Les vagues de monstres

Chaque vague voit sa vie augmenter de 10 000, à chaque nouvelle vague.
La première vague avance à 5 pixels par seconde
Ensuite les autres avances à 10 pixels par seconde
(nous avons choisi de faire ceci, car la vitesse max imposée par l'énoncé est 10 pixels par seconde, donc il est préférable pour le déroulement du jeu qu'ils avancent à leurs vitesses max, car celle-ci n'est déjà pas bien élevée.)

Vague 5 : la résistance est 50% sur les tours de type rocket.
Vague 10 : la résistance est 50% sur les tours de type laser.
Vague 15 : la résistance est 70% sur les tours de type laser, et 50% pour les tours mitraillette.
Vague 20 : la résistance est 30% sur les tours de type laser, et 50% pour les mitraillettes, et 50% pour les hybrides.

> *Les towers*

- La tour pokeball bleue ou de type « mitraillette » :

Puissance : 70
Portée : 100
Cadence : 0.2s
Cout : 30
temps de construction 1s

- La tour pokeball rouge ou de type « hybride » :

Puissance : 200
Portée : 200
Cadence : 0.5s
Cout : 10
temps de construction 2s

- La tour pokeball violette ou de type « rocket » :

Puissance : 700
Portée : 200px
Cadence : 0.9s
Cout : 15
temps construction : 2s

- La tour pokeball jaune ou de type « laser » :

Puissance : 100
Portée : 80
Cadence : 0.1s
Cout : 20
temps de construction 10s

> Construire une tour

Afin de déposer un bâtiment sur la carte, cliquer sur la tour que vous souhaitez ajouter dans le menu en haut à droite, la tour s'accroche ensuite à votre curseur. Enfin déposer la tour sur la carte pour construire la pokeball désirée.
(ou consultez les raccourcis clavier grâce à la touche H)

> Vendre une tour

On sélectionne une tour puis soit nous cliquons directement sur la touche «s», soit nous nous rendons en haut à droite et nous cliquons sur le bouton vendre qui apparait en dessous de la description.

> Visualiser éléments

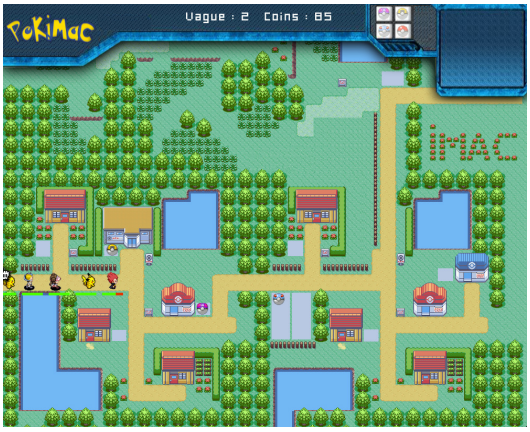
Afin de connaitre les caractéristiques liés à une une tour du jeu cliquez sur la tour la description s’affichera en haut à droite.

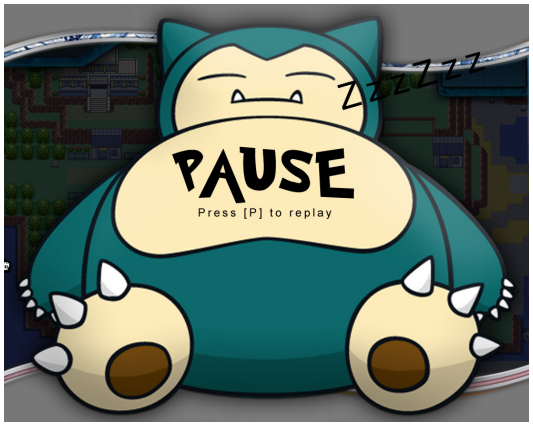
> Raccourcis clavier

Q / ESC : quitter
H : Afficher l’aide
P : Pause
À : Tour rocket
Z : Tour laser
E : Tour mitraillette
R : Tour hybride
T : Affiche le chemin, ainsi que le rayon des zones constructibles (cercle blanc) , et le rayon autour d’une tour où nous ne pouvons pas construire (noir), et les noeuds IN (vert) et OUT (rouge)
U : Évolution d’une tour
S : Vente d’une tour

6 Résultats obtenus

6.1 - Images





7 Conclusion

Le projet Imac Tower Defence avait pour objectif d'appliquer les notions vues en cours de synthèse d'image, mais les nombreuses étapes nous ont poussée à aller par delà les notions vues en cours. Ce projet nous a demandé beaucoup de travail et nous avons appris à nous organiser afin d'optimiser notre productivité.

Il s'agissait ici de notre premier jeu et cela fut très intéressant de créer une application jouable et une réelle satisfaction de pouvoir y jouer, c'est pourquoi nous avons prit du plaisir à ajouter de nombreuses fonctions au jeu, et à réaliser un visuel et des sonoritées attrayantes. La dimension visuelle nous a motivés dans le développement, car jusqu'à présent nous avions peu affiché d'image en programmation C. Cette application nous a permis d'approfondir nos connaissances dans le langage C et d'assimiler des automatismes. Nous avons de plus appris à nous servir de librairie comme OpenGL, SDL, SDL_ttf, SDL_image, et SDL_mixer.