Projet Logiciel Transversal

Patrick Contin, William Duval Bourligueux, Bastien Guillard, Abdel-Oihed Houta

Table des matières

1 Présentation Générale

1.1 Archétype

Le jeu est un tactical RPG, basé sur des jeux tels que Final Fantasy tactics, Fae tactics ou encore la série des Fire Emblem.

1.2 Règles du jeu

Le jeu se déroule sur une map la forme d'une grille, celle-ci voit s'affronter 2 équipes de plusieurs personnages. Chaque personnage peut se déplacer et interagir (attaquer, soigner, etc...) avec les autres sur la map. La victoire est déclarée quand l'ensemble des membres d'une équipes sont KO (PV = 0).
Les personnages commencent avec 0 de mana et en gagnent un peu en début de chaque tour, les sorts ont différents couts de mana en fonction de leur puissance.

1.2.1 Stat des personnages

Nom en jeu	Effet
PV(Point de Vie)	Point de vie du personnage, il est KO s'ils tombent a 0.
PM(Point de Mana)	Point de Mana, consommés par les capacités.
Attaque	Attaque physique d'un personnage.
Armure	Défense physique d'un personnage. Résiste aux dégats physique.
Magie	Puissance d'effets des capacités qui consomme du mana.
Ténacité	Défense magique d'un personnage. Résiste au dégats magique
Vitesse	Permet de déterminer l'ordre des tours
Mobilité	Nombre de case pouvant être parcouru en 1 seul tour.
Esquive	Chance d'esquive du personnage.

1.2.2 Classe des personnage

Les classes sont reparties en plusieurs categories selon leur utilités (degat, tank, support) et leur portées (mêlé et porté). De base il y aurait 3 classes :

- un guerrier tank : peu de mobilité, vitesse, beaucoup de défense et de vie. Son attaque de base est un coup d'épée au corps a corps, avec un sort de protection, et un sort qui force un ennemi a l'attaquer.
- un mage support : peu de mobilité, peu de défense, peu de dégat, moyenne portée, ses sorts permêttent d'augmenter les stats des alliées et de les soignées.
- un archer qui fait des dégats : peu de défense, vitesse moyenne, beaucopu de portée et de dégats, un sort qui fait beaucoup de dégat sur une seule cible, et un sort qui fait des dégat de zone.

1.3 Ressources

Pour les ressources de ce projet, nous avons réaliser la carte du jeu avec le logiciel Tiled (Voir dossier res).

2 Description et conception des états

2.1 Description des états

Un état de jeu a besoin de 3 informations, la carte du actuelle du jeu, les personnages et leur état, ainsi que les joueurs et leur état.

2.1.1 L'état de la carte du jeu

La carte du jeu est composé d'une liste de cellule, qui compose la carte. Chaque cellule peut être, soit :
— Être vide, juste le sol de la case, avec rien dessus
— Avoir un personnage dessus, peut importe son état
— Avoir un obstacle (arbre, rocher, ou autre)
Il y a plusieurs carte de jeu prédéfini.

2.1.2 L'état du joueur

Le joueur possède une liste de personnage qu'il possède, avec lesquels il peut jouer. Il possède aussi un état, qui indique si :
— il est encore en jeu
— il a gagné sa partie
— il a perdu sa partie
— il ne joue plus au jeu (un timer compte, si le joueur prend trop de temps à jouer)

2.1.3 L'état des personnages

Les personnages de chaque joueur possède un ensemble de statistiques :
— Leur point de vie (PV), permet de déterminer combien de coup le personnage peut prendre avant de mourrir
— Leur attaque (ATK), permet de déterminer combien de dégats le personnage va faire avec son attaque de base
— Leur attaque magique (MAG), permet de déterminer combien de dégats le personnage va faire avec ses sorts
— Leur défense magique (RM), permet de réduire les dégats pris par les sorts
— Leur défense physique (DEF), permet de réduire les dégats pris par les attaques de base
— Leur vitesse (VIT), permet de déterminer l'ordre des personnages
— Leur mobilité (MOB), permet de déterminer la quantité de case que le personnage peut se déplacer
— Leur esquive (ESQ), permet d'éviter les attaques et les sorts

Chaque personnage possède également un nom, un compteurs de tours, ainsi qu'une liste de sorts qu'il peut lancer. Il possède aussi une liste d'effets qui va être remplie aux cours de la partie, au fur et a mesure que des effets lui sont appliqué.

2.1.4 L'état général du jeu

L'état général du jeu permet de compter le nombre de tours passé, de joueurs encore en jeu. Ainsi que l'index du joueur qui est en train de jouer, et une liste de tout les personnages encore en jeu.

2.2 Conception Logiciel

Le diagramme des classes pour les états est présenté en Figure 1. En rouge foncé on distingue la classe "état" principal, alors qu'en rouge plus clair on voit les classes lié aux personnages et la carte du jeu. En blance on a les énumerations, qui sont utilisé pour ne pas utiliser de "magic numbers", et de permettre une transparence sur l'utilisation des variables et de leur différents états.

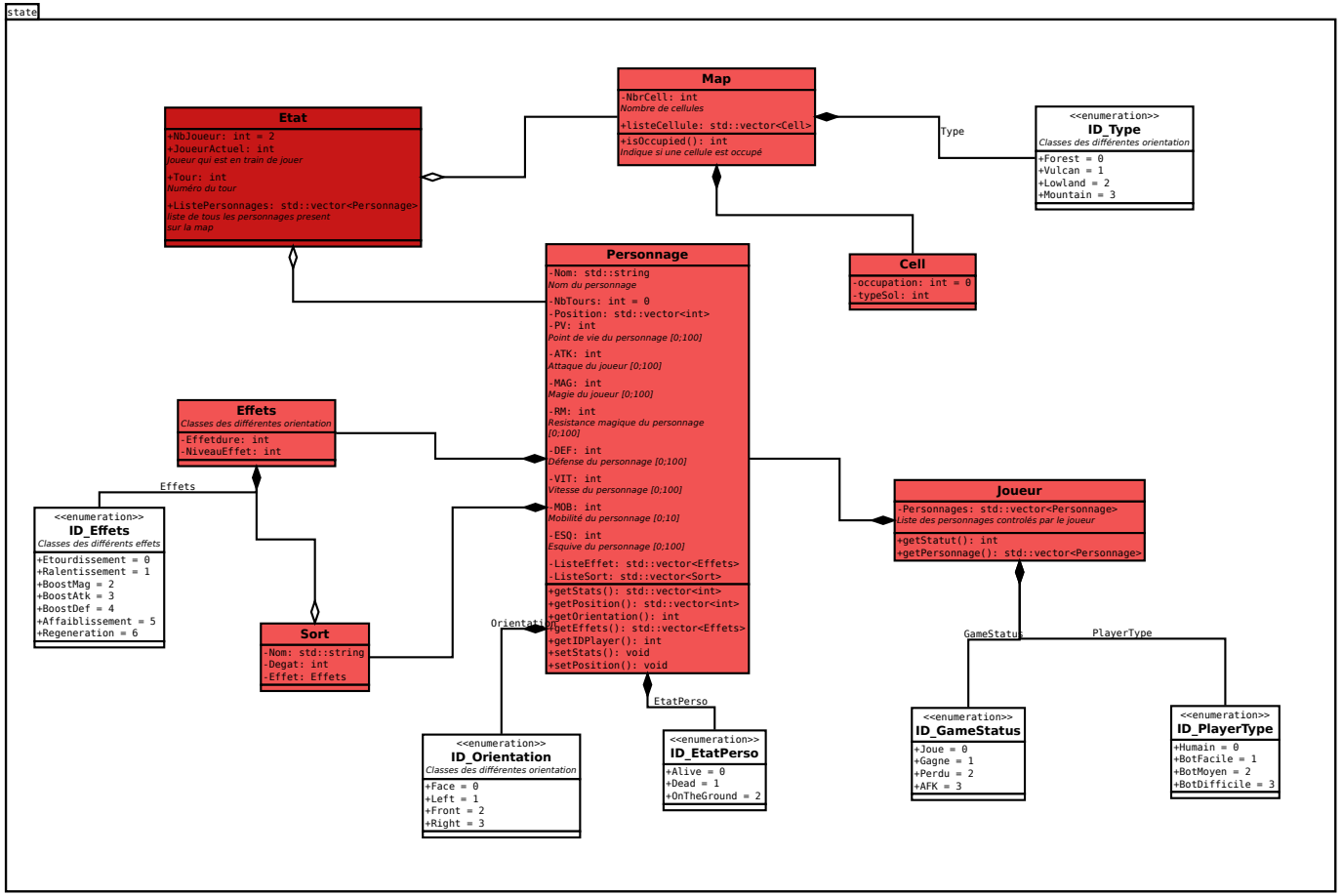

FIGURE 1 – Diagramme des classes d'état.

3 Rendu : Stratégie et Conception

3.1 Stratégie de rendu d'un état

La stratégie pour laquelle nous avons choisis d'opter est celle utilisée dans l'exemple de M.Gosselin. Cette stratégie simple est basée sur un développement de bas niveau avec des élément simples comme des textures, couches, etc..
En effet, nous avons choisi de représenté un état suivant 3 étapes :

— la map
— les personnages
— les objets

Map La première étape concerne la map, nous avons utilisé le tutoriel dans la documentation SFML. Pour éviter de recharger la map à chaque coup d'horloge, nous décidons de l'afficher sur une couche (layer). La map sera quadrillé sous forme de matrice sur laquelle on vient attribuer des textures grâce à des tuiles. Ainsi les textures seront fixées et la map ne sera plus rechargée, ce qui permettra d'alléger la charge du CPU et de la carte graphique.

Personnages Les personnages sont crées grâce à une liste d'objet instanciés avec la classe personnage du state. Ces personnages seront placés sur la map selon leurs positions définies par les attributs de la classe Position du state. Concernant les textures elles seront chargées grâce à des sprites définis dans la bibliothèques SFML. Cela demande plus de ressources que la méthode pour afficher la map mais comme nous avons peu de personnage, ce n'est pas un problème. Le rendu s'actualise en permanence suivant une fréquence que l'on définira par la suite sachant que 50/60 Hz est la fréquence de rafraîchissement habituelle.

Objets Les objets seront traités uniquement si la partie map et personnages sont fonctionnelles. Les objets pourront affecter le rendu des personnages (un personnage peu changer de couleur s'il obtient un boost d'attaque) ce qui peut compliqué le code. Ainsi les objets seront traités dernier suivant la même méthode que les personnages (avec des sprites).

3.2 Conception logiciel

Le diagramme du rendu est disponible ici : 2.

La classe **StateLayer** va nous permettre de créer les différentes couches pour les différents niveaux. Elle permet aussi d'avoir un affichage graphiques avec à l'instanciation d'une fenêtre graphique de la bibliothèque SFML. Elle récupère des informations de toutes les autres classes et gère l'ensemble du rendu.

La classe **LoadLayer** va nous permettre de texturer nos couches avec l'utilisation de quads, ensemble de 4 coins formant un rectangle, qui permet de cadrier la map pour y associer des textures.

La classe **Tiles** permet de gérer les tuiles. Mais surtout, elle permet de récupérer les ressources sous forme d'images que nous associons aux tiles. Sans ceci la texture est impossible.

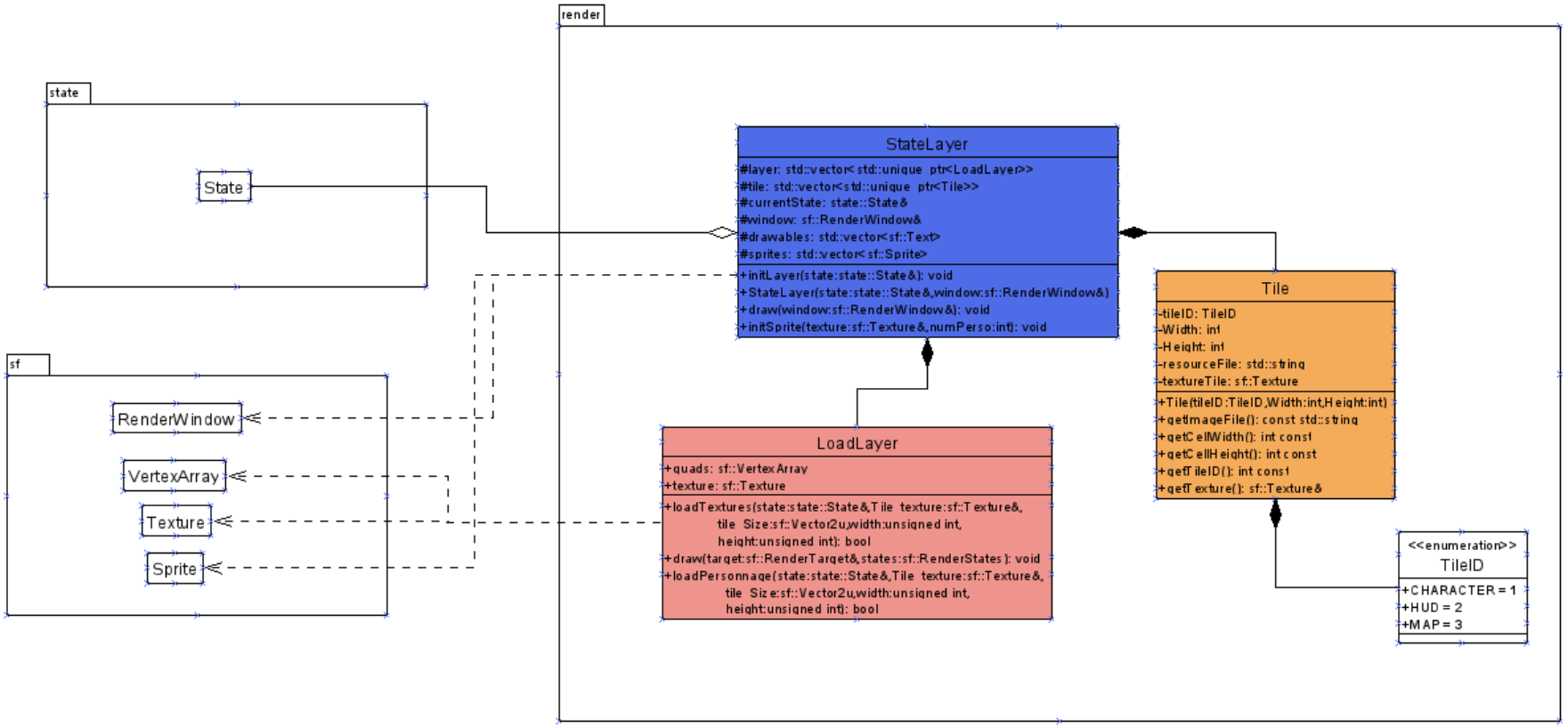

FIGURE 2 – Diagramme des classes de rendu.

4 Règles de changement d’états et moteur de jeu

4.1 Règles

4.2 Conception logiciel

5 Intelligence Artificielle

5.1 Stratégies

5.2 Conception logiciel

6 Modularisation

6.1 Organisation des modules

6.2 Conception logiciel