

Projet Logiciel Transversal

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

Table des matières

| | | |
|----------|--|----------|
| 1 | Présentation Générale | 1 |
| 1.1 | Archétype | 1 |
| 1.2 | Règles du jeu | 1 |
| 1.2.1 | Stat des personnages | 1 |
| 1.2.2 | Classe des personnage | 1 |
| 1.3 | Ressources | 1 |
| 2 | Description et conception des états | 2 |
| 2.1 | Description des états | 2 |
| 2.1.1 | L'état de la carte du jeu | 2 |
| 2.1.2 | L'état du joueur | 2 |
| 2.1.3 | L'état des personnages | 2 |
| 2.1.4 | L'état général du jeu | 2 |
| 2.2 | Conception Logiciel | 3 |
| 3 | Rendu : Stratégie et Conception | 5 |
| 3.1 | Stratégie de rendu d'un état | 5 |
| 3.2 | Conception logiciel | 5 |
| 4 | Règles de changement d'états et moteur de jeu | 6 |
| 4.1 | Règles | 6 |
| 4.1.1 | Actualisation | 6 |
| 4.1.2 | Règles extérieurs | 6 |
| 4.1.3 | Règles automatiques | 6 |
| 4.2 | Conception logiciel | 8 |
| 5 | Intelligence Artificielle | 8 |
| 5.1 | Stratégies | 8 |
| 5.2 | Conception logiciel | 9 |
| 6 | Modularisation | 9 |
| 6.1 | Organisation des modules | 9 |
| 6.2 | Conception logiciel | 10 |

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 permettent d'augmenter les stats des alliées et de les soignées.
- un archer qui fait des dégats : peu de défense, vitesse moyenne, beaucoup 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 blanche on a les énumérations, 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.

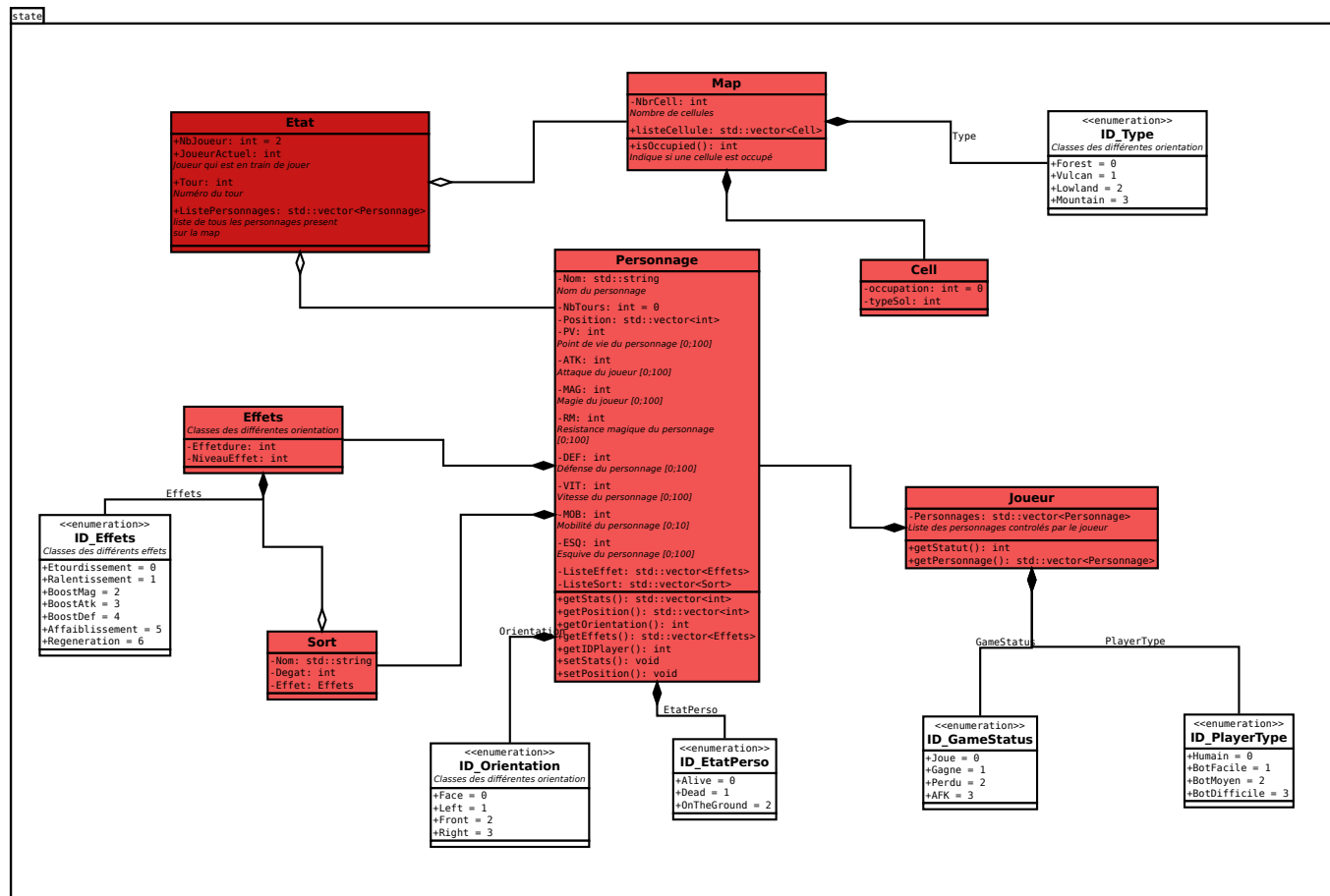


FIGURE 1 – Diagramme des classes d'état.

3 Rendu : Stratégie et Conception

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

4 Règles de changement d'états et moteur de jeu

4.1 Règles

4.1.1 Actualisation

L'engine est complètement indépendant du render, le moteur de jeu ne sert qu'à mettre à jour l'état du jeu, en fonction des commandes extérieures (Clavier, souris ou serveur), et des règles automatiques, qui sont vérifiées à chaque changement d'état. Le moteur de jeu effectue une mise à jour de l'état à chaque commande extérieure, les règles automatiques n'ont besoin de s'activer qu'uniquement dans le cas d'un changement d'état (jeu tour par tour). Il existe aussi une règle autonome qui doit s'incrémenter entre les tours, afin de déterminer l'ordre de passage des tours des différents personnages. Cette commande sera appelée par une horloge.

4.1.2 Règles extérieures

Les règles extérieures sont provoquées par les clics de souris à différents endroits, ou les ordres reçus par le réseau.

- Sélectionner un personnage, afin d'afficher les statistiques du personnage
- Déplacer les personnages
- Attaquer un ennemi
- Lancer un sort

4.1.3 Règles automatiques

Les règles automatiques sont les checks qui sont lancés à chaque fois que l'on veut faire une action, afin de savoir si l'action est valide, elles sont exécutées en fonction de la commande extérieure qui a été exécutée auparavant. Un autre type de règle automatique est la barre d'action, qui détermine l'ordre des tours, cette barre d'actions est une commande qui va s'exécuter à intervalles réguliers.

Les différentes règles suivant les actions sont, pour :

- Incrémenter la barre d'action :
 - Tester si un personnage est arrivé au maximum de la barre
 - Si un personnage est arrivé au maximum : passer en mode tour par tour, lui permettre de jouer et lui donner son mana.
- Sélectionner un personnage :
 - Chercher les informations sur le personnage sélectionné
 - Afficher les informations sur le personnage sélectionné
- Déplacer un personnage :
 - Vérifier la disponibilité des cases
 - Calculer le pathfinding
 - Effectuer le déplacement
- Attaquer un personnage
 - Tester la possibilité de l'attaque (portée, ligne de vue)
 - Calculer les positions relatives, pour les dégâts directionnels
 - Calculer les dégâts infligés (défense, crit et autre)
 - Appliquer les dégâts au personnage
 - Tester si le personnage est mort
- Lancer un sort
 - Obtenir les informations sur le type de sort
 - Obtenir les informations sur la cible

— Réaliser l'action en fonction du sort (si c'est une attaque, on attaque, un soin, on soigne...)

4.2 Conception logiciel

Le diagramme de classe de moteur de jeu est représenté sur la figure Le jeu repose sur un patron de conception de type Command.

La classe commande est une classe abstraite qui parente toutes les autres classe commande

Les classes commandes, ce sont les classes qui représentent les différentes commandes.

5 Intelligence Artificielle

5.1 Stratégies

5.2 Conception logiciel

6 Modularisation

6.1 Organisation des modules

6.2 Conception logiciel