Projet Logiciel Transversal

Gaétan RAYNAUD – Mathis OUDIN

Table des matières

1 Objectif

1.1 Présentation générale

Ce projet à pour but de la réalisation d'un jeu de type Battle for Wesnoth, avec des règles simplifiées.

1.2 Règles du jeu

Le jeu voit s'affronter deux équipes. Le jeu se déroule sur une carte découpée en cases, les cases sont réparties selon plusieurs types, ayant chacun leurs caractéristiques propres. Le jeu est découpé en tour, les joueurs jouent chacun leur tour.

Chaque équipe dispose d'une unité principale, le général, qui est l'unité principale de l'équipe. Si le général meurt, l'équipe perd la partie.

Chaque équipe commence avec une quantité d'or et peut en gagner à chaque début de tour en revendiquant une case de type maison, en plaçant une unité dessus. Si le général est situé sur une case de type château, il peut recruter d'autres unités de type épéiste, sur les cases de type caserne adjacentes.

A chaque tour, toutes les unités disposent d'une distance maximale de déplacement qui leur sont propre. Si une unité est sur une case adjacente à une unité ennemie, elle peut l'attaquer. Si une unité n'a plus de vie, elle meurt et disparaît du plateau.

1.3 Ressources

Le projet utilise plusieurs ressources pour l'affichage. Chaque texture est découpée en image de 72x72pixels.

Il y a six textures pour les différents types de cases :

Figure 1: Texture des cases

Il y a six types d’unités, avec plusieurs animation pour les différentes actions :

Figure 2: Texture du général humain

Figure 3: Texture de l'épéiste humain

Figure 4: Texture de l’archer humain

Figure 5: Texture du général orc

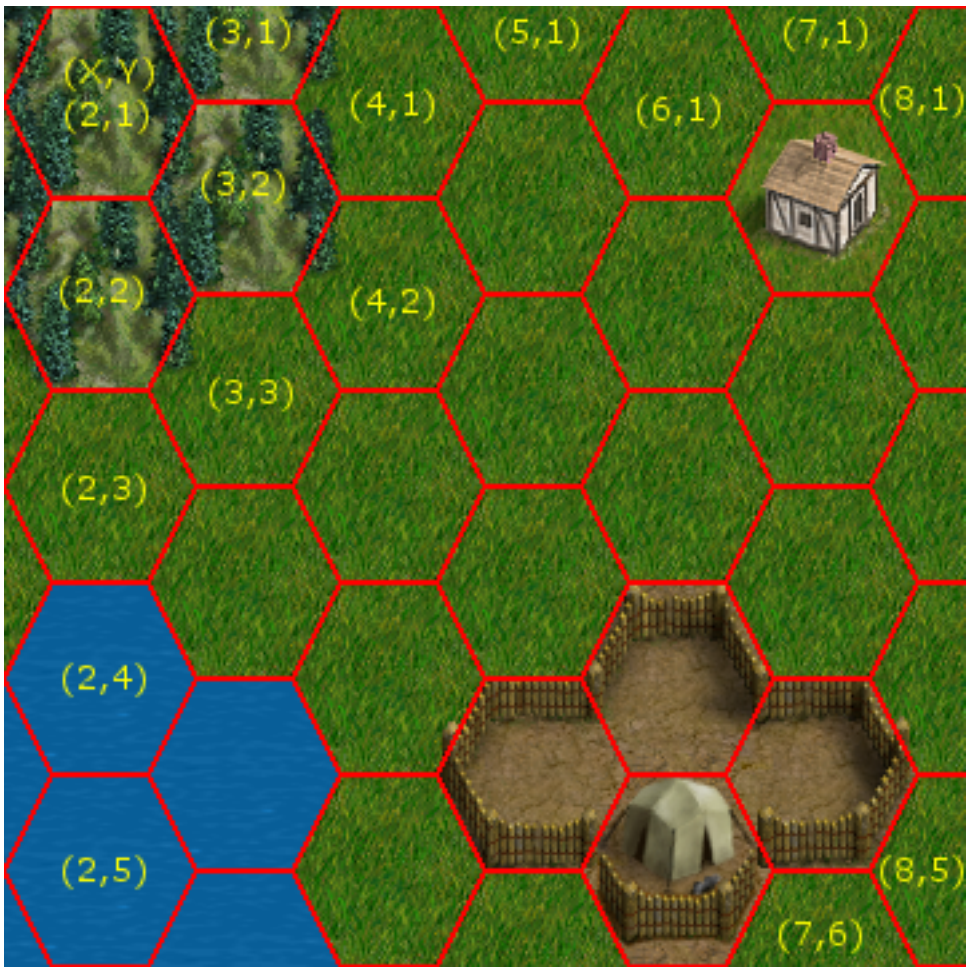
Figure 6: Texture de l'épéiste orc

Figure 7: Texture de l'archer orc

2 Description et conception des états

2.1 Description des états

Le jeu est formé d'une grille divisée en cases hexagonales avec différentes textures pour chaque case de la grille.



Sur cette grille des personnages mobiles se déplacent sur les cases à chaque tour. Les coordonnées (x, y) de chaque personnage correspond à sa position dans la grille.

2.1.1 Éléments statiques : Terrain

La grille est formée de cases hexagonales de taille fixe. La grille est construite à partir de différentes cases possédant chacune une texture. La grille est généré au démarrage du niveau. Chaque case possède des coordonnées (x, y) de sa position dans la grille notées comme l'illustration ci-dessus. Chaque case possède un coup de déplacement pour les unités mobiles.

Les différentes types de cases sont :

- **Les cases « Default » :** Les cases « Default » sont des éléments franchissables par les personnages et qui n'ont pas de propriétés particulières. Le choix du type change seulement le coût de déplacement du personnage ainsi que l'apparence de la case.
  Il existe plusieurs types de cases « Default » :
  - Les cases « Grass » classiques.
  - Les cases « Forest » avec une apparence de forêt.
  - Les cases « Water » qui représente l'eau.
- **Les cases « House » :** Les cases « House » sont des éléments franchissables par les personnages. Lorsqu'une personnage termine son tour sur une case « House », celle-ci est revendiquée par l'équipe de l'unité. Chaque case revendiquée rapporte de l'or à l'équipe qui la possède au début de son tour.
- **Les cases « Castle » :** Les cases « Castle » sont des éléments franchissables par les personnages. Si un personnage « Leader » se situe sur une case « Castle », il peut alors recruter d'autres personnages pour son équipe. Soit des « Sworman » soit des « Bowman », en contre parti d'un prix en or.
- **Les cases « Wall » :** Les cases « Wall » sont des éléments franchissables par les personnages. Les cases « Wall » sont associées à une case « Castle ». Lorsqu'une case « Wall » est collée à un case « Castle », le personnage « Leader » qui se troue sur la case « Castle », peut recruter un personnage pour son équipe sur les case « Wall » vides. Le « Leader » ne peut recruter de personnage que sur les cases « Wall » ce qui permet de limiter le nombre de personnages recruter à chaque tour.

2.1.2 Éléments mobiles : Unit

Les éléments mobiles possèdent des coordonnées qui correspondent à une position sur la grille. Chaque unité possède également des points de vies, une vitesse correspondant au nombre de cases qu'il reste à l'unité pour se déplacer, un identifiant d'équipe, et un tableau d'armes. Les armes disponibles sont une épée et un arc.

2.2 Conception logiciel

2.3 Conception logiciel : extension pour le rendu

2.4 Conception logiciel : extension pour le moteur de jeu

2.5 Ressources

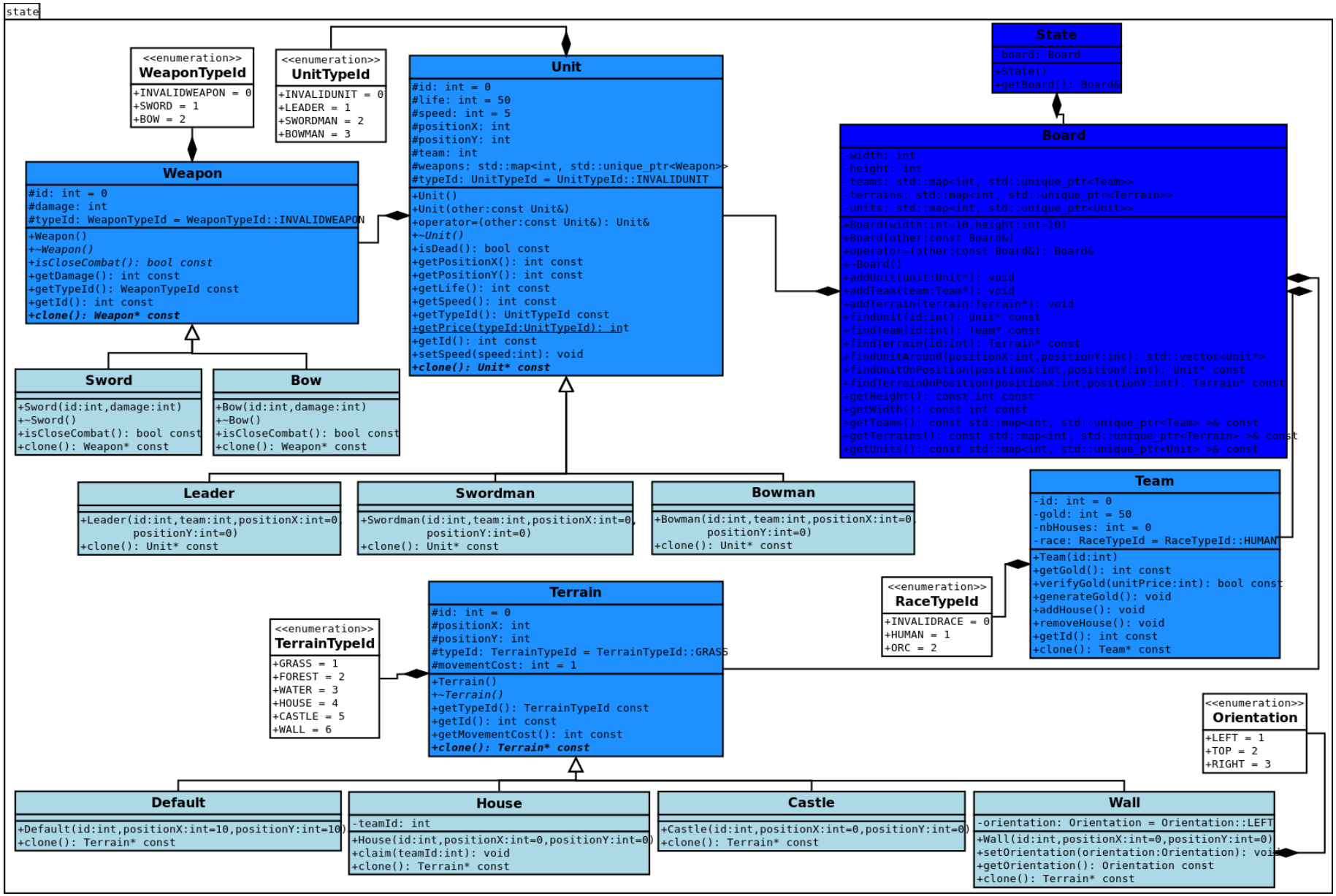


Figure 8: Diagramme des classes

3 Rendu : Stratégie et Conception

Présentez ici la stratégie générale que vous comptez suivre pour rendre un état. Cela doit tenir compte des problématiques de synchronisation entre les changements d'états et la vitesse d'affichage à l'écran. Puis, lorsque vous serez rendu à la partie client/serveur, expliquez comment vous aller gérer les problèmes liés à la latence. Après cette description, présentez la conception logicielle. Pour celle-ci, il est fortement recommandé de former une première partie indépendante de toute librairie graphique, puis de présenter d'autres parties qui l'implémente pour une librairie particulière. Enfin, toutes les classes de la première partie doivent avoir pour unique dépendance les classes d'état de la section précédente.

3.1 Stratégie de rendu d'un état

3.2 Conception logiciel

3.3 Conception logiciel : extension pour les animations

3.4 Ressources

3.5 Exemple de rendu

Illustration 1: Diagramme de classes pour le rendu

4 Règles de changement d'états et moteur de jeu

Dans cette section, il faut présenter les événements qui peuvent faire passer d'un état à un autre. Il faut également décrire les aspects lié au temps, comme la chronologie des événements et les aspects de synchronisation. Une fois ceci présenté, on propose une conception logiciel pour pouvoir mettre en œuvre ces règles, autrement dit le moteur de jeu.

4.1 Horloge globale

4.2 Changements extérieurs

4.3 Changements autonomes

4.4 Conception logiciel

4.5 Conception logiciel : extension pour l'IA

4.6 Conception logiciel : extension pour la parallélisation

Illustration 2: Diagrammes des classes pour le moteur de jeu

5 Intelligence Artificielle

Cette section est dédiée aux stratégies et outils développés pour créer un joueur artificiel. Ce robot doit utiliser les mêmes commandes qu'un joueur humain, ie utiliser les mêmes actions/ordres que ceux produit par le clavier ou la souris. Le robot ne doit pas avoir accès à plus information qu'un joueur humain. Comme pour les autres sections, commencez par présenter la stratégie, puis la conception logicielle.

5.1 Stratégies

5.1.1 Intelligence minimale

5.1.2 Intelligence basée sur des heuristiques

5.1.3 Intelligence basée sur les arbres de recherche

5.2 Conception logiciel

5.3 Conception logiciel : extension pour l'IA composée

5.4 Conception logiciel : extension pour IA avancée

5.5 Conception logiciel : extension pour la parallélisation

6 Modularisation

Cette section se concentre sur la répartition des différents modules du jeu dans différents processus. Deux niveaux doivent être considérés. Le premier est la répartition des modules sur différents threads. Notons bien que ce qui est attendu est un parallélisation maximale des traitements: il faut bien démontrer que l'intersection des processus communs ou bloquant est minimale. Le deuxième niveau est la répartition des modules sur différentes machines, via une interface réseau. Dans tous les cas, motivez vos choix, et indiquez également les latences qui en résulte.

6.1 Organisation des modules

6.1.1 Répartition sur différents threads

6.1.2 Répartition sur différentes machines

6.2 Conception logiciel

6.3 Conception logiciel : extension réseau

6.4 Conception logiciel : client Android

Illustration 3: Diagramme de classes pour la modularisation