

Projet Logiciel Transversal

Advance Wars

par REN Xiaoyu et WONGWANIT Nicolas



Figure 1 : Un tour de jeu - Advance Wars

Table des matières

1. Présentation Générale	3
1.1. Archétype	3
1.2. Règles du jeu	3
1.3. Ressources	4
2. Description et conception des états	5
2.1. Description des états	5
2.1.1. Etat des cellules (Cell)	5
2.1.2. Etat des bâtiment (Building)	5
2.1.3. Etat des troupes (Unit)	5
2.1.4. Etat des joueurs (Player)	5
2.1.5. Etat général (State)	5
2.2. Conception logiciel	5

1. Présentation Générale

1.1. Archétype

Le projet proposé est un jeu suivant l'archétype du jeu de stratégie en tour par tour "Advance Wars".



Figure 2 : Exemple d'un tour de jeu (Joueur 1 en rouge)

1.2. Règles du jeu

- Le but du jeu est de vaincre l'adversaire, soit en détruisant la totalité de son armée, soit en capturant le Q.G.. Dans ce but, le joueur dispose de nombreuses unités de combat, avec des capacités variées (infanteries, tanks, artilleries, ...). Note : dans certains modes de jeu, il n'y a pas de Q.G., et/ou les joueurs commencent la partie sans troupe.
- Le jeu se déroule sur un plateau à maille carré. Il existe des cases ayant des propriétés diverses (plaines, montagnes, villes, usines, ...). Les joueurs jouent à tour de rôle. Lors du tour d'un joueur, il est possible de déplacer chaque unités du joueur une seule fois, et ensuite d'attaquer une unité ennemie adjacente. Il n'est pas possible d'attaquer PUIS de se déplacer. Il existe des exceptions, qui sont les unités d'attaque à distance et les unités de transport : unités d'attaque à distance ne peuvent que se déplacer ou bien attaquer, et les unités de transport ne peuvent tout simplement pas attaquer.
- Le joueur peut également produire de nouvelles unités, tant qu'il possède le capital nécessaire et des usines libres. Chaque usine ne peut produire qu'une unique unité par tour de jeu. De plus, l'unité produite ne peut pas agir durant le tour de production.

- Il est à noter que seul les infanteries peuvent capturer les villes, usines et Q.G. Capturer des villes et usines permet d'acquérir des fonds et de produire plus d'unités.
- Chacune des unités possèdent bien sûr leurs forces et leurs faiblesses. Il est donc nécessaire de déployer ses unités avec soin, en tenant compte de ces avantages et du terrain, afin de triompher de son adversaire.

1.3. Ressources

Des sprites seront utilisés pour réaliser le rendu graphique du jeu. Chaque sprite aura la taille d'un carré 16x16 pixels, représentant une case du terrain, une unité, ou un bâtiment.



Figure 3 : Sprites des unités utilisés dans le jeu



Figure 4 : sprites des bâtiments



Figure 5 : sprites des cellules terrain.

2. Description et conception des états

2.1. Description des états

Un état du jeu est formé par une matrice de cellules (Cell) 2D représentant le plateau de jeu. Sur ce plateau sont placés des bâtiments (Building), immobiles, et des troupes pouvant appartenir à différents joueurs, qui sont mobiles.

Les bâtiments et les troupes possèdent tous comme propriété leur coordonnées sur le terrain avec les attributs (x,y).

2.1.1. Etat des cellules (Cell)

Chaque cellule possède un attribut énuméré CellType qui décrit un type de terrain, qui offre divers avantages aux troupes sur ce terrain.

Les cellules peuvent, de plus, être associées à un objet Unit et/ou Building.

2.1.2. Etat des bâtiment (Building)

Les bâtiments possèdent les attributs énumérés BuildingType et BuildingTeam.

- BuildingTeam indique quel joueur a le contrôle du bâtiment. BuildingType peut avoir les valeurs NEUTRAL, PLAYER1 ou PLAYER2.
- BuildingType indique le type de bâtiment. Parmi ces différents types, on trouve le type BASE, correspondant au Q.G. (quartier général) du joueur, la ville CITY, ainsi que les bâtiments de productions FACTORY, AIRPORT et SEAPORT.
- L'attribut *capturePoint* donne le temps avant capture du bâtiment par un joueur différent du propriétaire.

2.1.3. Etat des troupes (Unit)

Les unités possèdent deux attributs énumérés UnitTeam et UnitType, ainsi que quatre autres attributs de type entier.

- UnitTeam : correspond à l'affiliation des troupes, PLAYER1 ou PLAYER2. Il n'y a pas de troupes neutres.
- UnitType : le type d'unité. Il existe 18 types d'unités (voir figure 4).
- *health* : indique la vitalité de l'unité. Lorsque la vitalité atteint zéro, l'unité est détruite.
- *ammo* : la quantité de munition possédée par la troupe. Lorsque cette valeur atteint zéro, l'unité ne peut plus attaquer.
- *fuel* : représente l'énergie de l'unité. Si cette valeur est nulle, l'unité ne peut plus se déplacer.
- *vision* : le champ de vision de l'unité en mode de jeu avec brouillard de guerre.

2.1.4. Etat des joueurs (Player)

Les joueurs possèdent une liste de leurs unités et bâtiment contrôlés à travers les conteneurs *unitList* et *buildingList* de pointeurs uniques.

Un attribut *PlayerID* entier est aussi associé à chaque joueurs.

2.1.5. Etat général (State)

L'ensemble des éléments de jeu sont réunis dans une classe état (State) à travers un vector de pointeurs vers Unit et Building, et une matrice de Cell.

2.2. Conception logiciel

Le diagramme des classes d'état est présenté ci-dessous (figure X). Nous pouvons remarquer que les classes Player et State contiennent les classes Unit et Building à travers des conteneurs de pointeurs uniques.

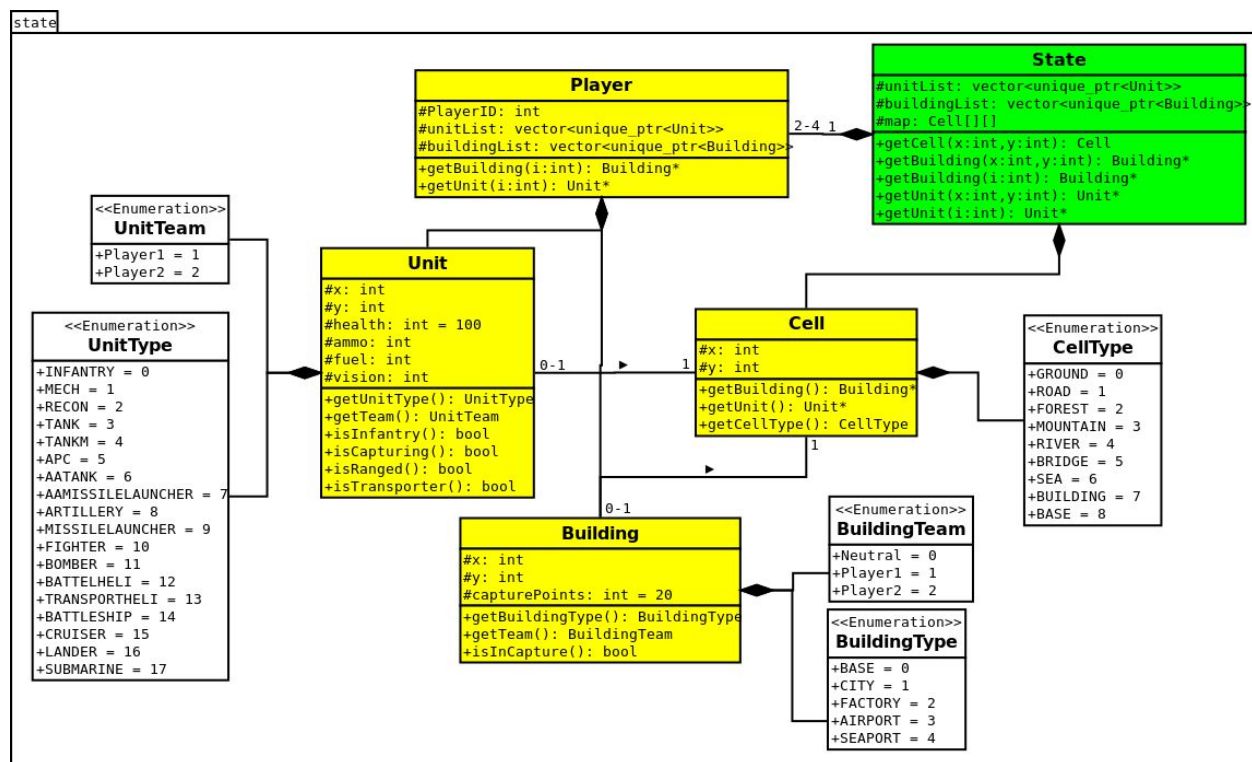


Figure 6 : Diagramme des classes d'état

3. Modèle du rendu

3.1. Description des classes

3.2. Conception logicielle

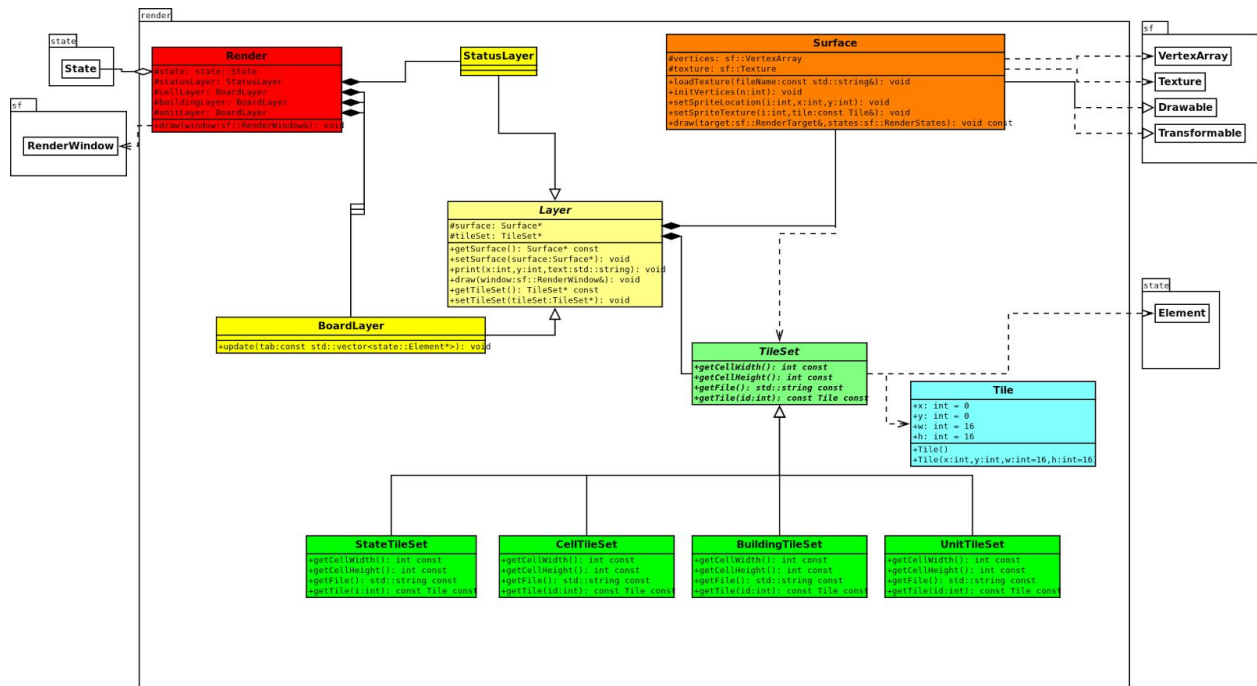


Figure 7 : Diagramme des classes de rendu

4. Conception du moteur de jeu

4.1. Description des classes

4.2. Conception logicielle

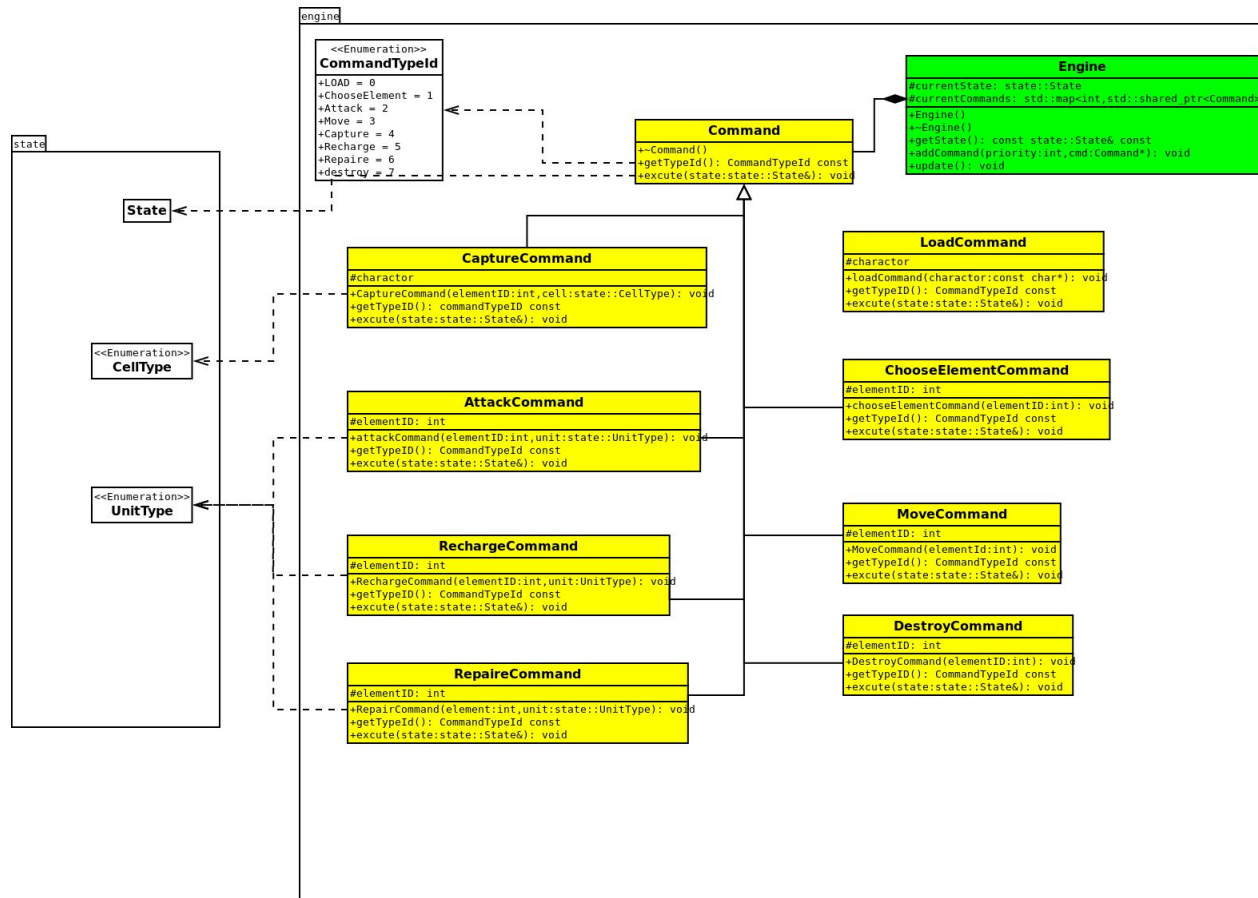


Figure 8 : Diagramme des classes du moteur de jeu