ENSEA

École nationale supérieure de l'électronique et de ses applications

RAPPORT DE PROJET LOGICIEL TRANSVERSAL

QUENTIN AMIEL
TINGYUE TENG
ASLAN CHAPPE
HUGO THIERRY

Christophe BARES Professeur référent

Table des matières

Table des figures

Section 1

Présentation

1.1 Archétype

FIGURE 1.1 – Endless Space 2

L'objectif de ce projet est de réaliser un jeu de type Endless Space 2. A l'origine, Endless Space 2 est un jeu vidéo de stratégie au tour par tour développé par Amplitude Studios et édité par Sega, dans lequel les joueurs incarnent une civilisation qui devra étendre son influence sur une carte, une galaxie comprenant des systèmes solaires eux-mêmes composés d'une à 4 planètes, générée aléatoirement. Les systèmes solaires font office de « villes » (si on le compare à Civilisation), ils sont reliés entre eux par des voies stellaire, qui eux servent de route. Le joueur peut se déplacer sur la galaxie à l'aide de vaisseaux spatiales militaire ou colonisateur.

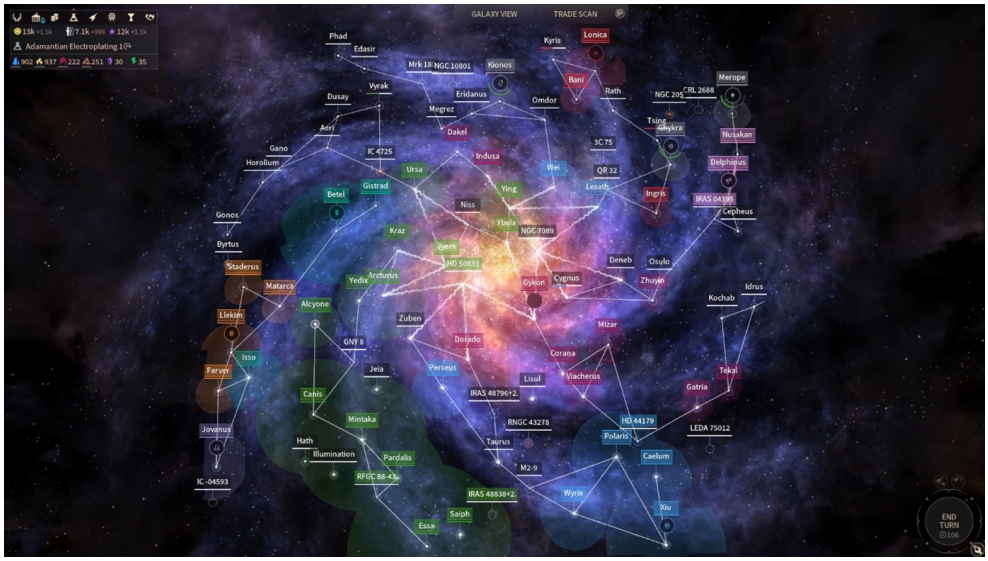

FIGURE 1.2 – Exemple de galaxie

1.2 Règles du jeu

En début de partie, vous ne commencez qu'avec une planète colonisée dans l'un des nombreux systèmes stellaire du jeu. Chaque planète produira en quantité plus ou moins importante quatre ressources différentes :

— L'industrie qui servira à la construction de bâtiments et de vaisseaux,
— La science qui permettra la rechercher de nouvelles technologies et améliorations,
— La brume comme monnaie unique du jeu pour acheter des bâtiments ou des vaisseaux directement,
— La nourriture pour augmenter le niveau des planètes.

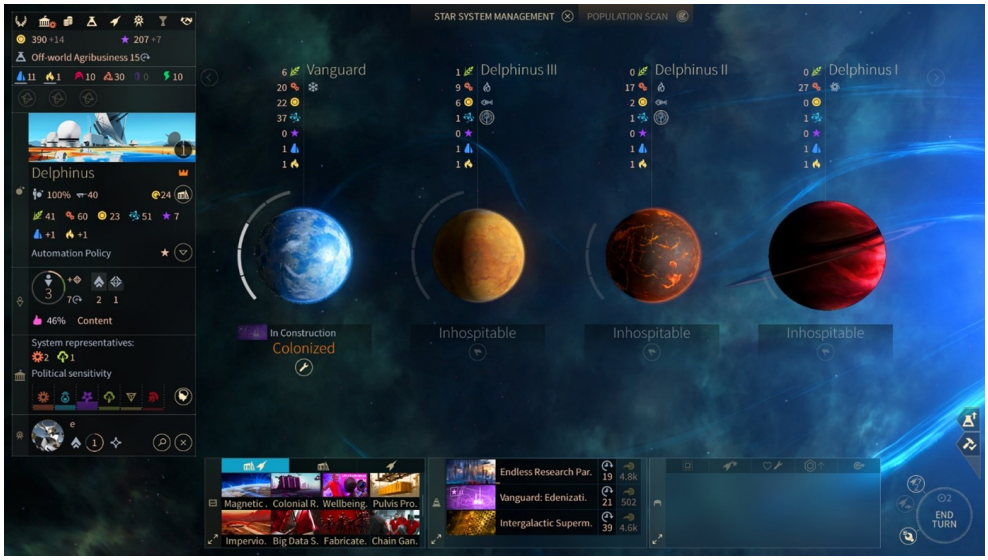

FIGURE 1.3 – Exemple de système stellaire colonisé par une civilisation

Le but du jeu est d'étendre sa civilisation à l'ensemble de la galaxie et de détruire les autres peuples. Afin de pouvoir rivaliser avec les autres peuples, il vous faudra deux choses : maîtriser l'extension de vos voisins et vous développer plus rapidement qu'eux. Ceci impose de coloniser aussi bien les planètes présentes sur votre système de départ que sur les autres systèmes présents dans la galaxie.

FIGURE 1.4 – Exemple de vaisseau

Pour gagner une partie d'Endless Space 2, plusieurs solutions s'offrent à vous, allant de celui qui a les plus grosses statistiques à la fin d'un nombre de tour définit, l'éradication pure et simple des autres factions ou arriver à la fin de l'arbre technologique. Côté combat, il n'est possible de se battre qu'uniquement que lorsque votre flotte est en orbite sur un système stellaire, sachant que dès qu'une route est empruntée, il est impossible de faire demi-tour tant que votre flotte n'est pas arrivée à la fin de la voie stellaire. Une fois le combat engagé, soit contre une flotte adverse, soit pour envahir une planète, ceux-ci se dérouleront de manière autonome.

1.3 Ressource

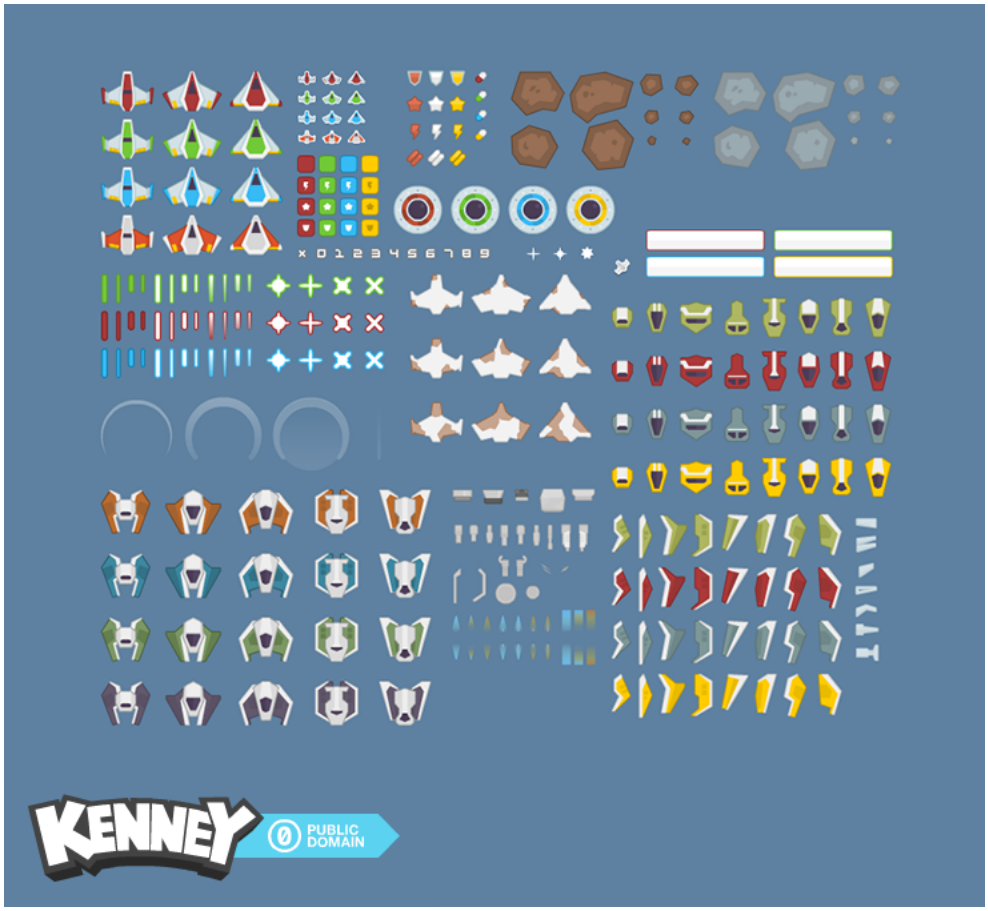

FIGURE 1.5 – Texture pour les vaisseaux

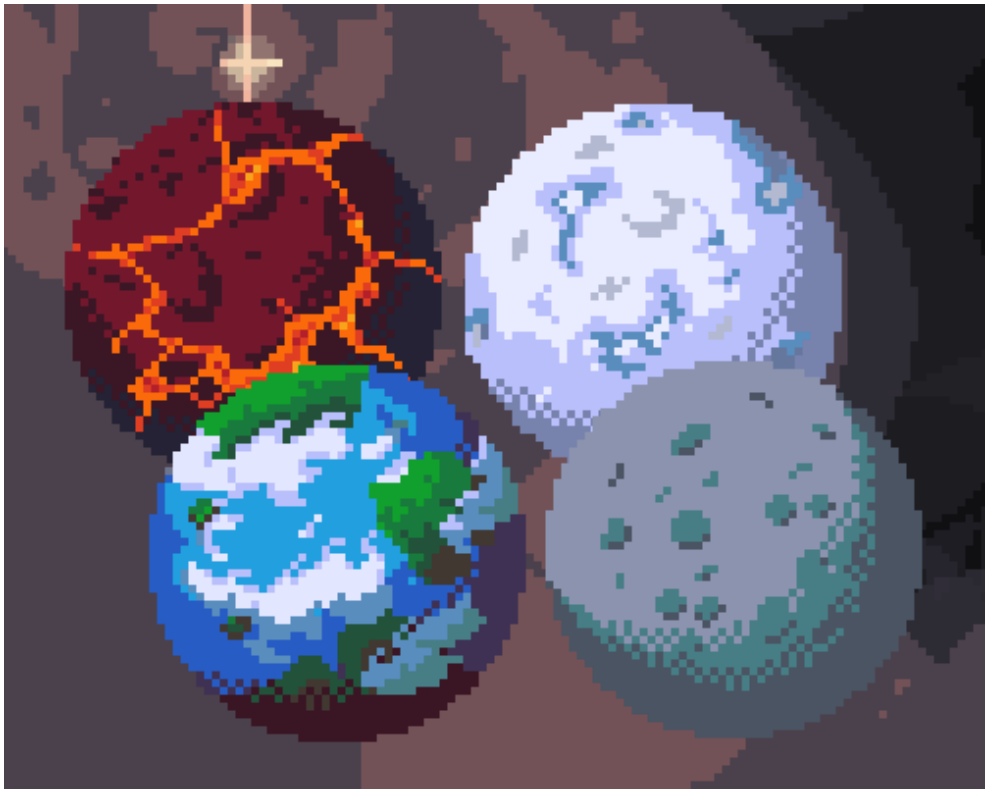

FIGURE 1.6 – Texture pour les planètes

Section 2

Description et Conception des états

2.1 Description des états

Un état du jeu est formé par un ensemble d'éléments fixes sur le terrain et un ensemble d'éléments mobiles ainsi que l'état du joueur.

— Nom,
— Position

2.1.1 États éléments fixes

La map est formée par une grille d'éléments nommé « SpaceCell ». La taille de cette grille est fixée. Les éléments fixes sur cette grille sont :

— L'élément "StellarSystem". Les systèmes stellaires sont des éléments fixes que le joueur pourra coloniser. Ils sont composés de 1 à 4 planètes, il existe 3 types de planètes :

  — les planètes "Neutral" qui ont une répartition des ressources équilibré
  — les planètes "Hot" qui favorisent la ressource de production
  — les planètes "Cold" qui favorisent la ressource de science

  De plus lorsqu'un joueur possède une système stellaire, il a la possibilité de construire des bâtiments qui seront liés au système. Ces derniers produiront une quantité différente des 4 ressources

— L'élément "StellarWay". Ce sont des routes de l'espace qui relient deux systèmes stellaires entre eux. Sa longueur influera sur le temps qu'un vaisseau mettra à la parcourir entièrement.

2.1.2 États éléments mobiles

Les seuls éléments mobiles du jeu sont les vaisseaux.

L'élément mobile "Ship" est dirigé par le "Player", il est construit ou acheter dans les systèmes stellaires. Chaque « Ship » possède des statistiques propres à sa classe. On lui associe ainsi des points de vie “health”, des

dégâts d'attaque "attack-point", une défense "defense-point" et des points de déplacement. Chaque vaisseaux possède aussi un avantage parmi ses quatre :

— Dégât augmenté contre les vaisseaux,
— Dégât augmenté contre les bâtiments,
— Vitesse de déplacement augmenté,
— Possibilité de coloniser.

Les vaisseaux possèdent un niveau qui augmente leurs statistiques, ses niveaux sont gagnés lors de victoires contre un vaisseau ennemie.

2.1.3 État du joueur

Le joueur possède un ensemble d'élément fixe, les systèmes stellaires, et mobile, les vaisseaux, ainsi que des ressource qu'il gagne à chaque tours de jeu.

2.2 Conception logiciel

L'architecture du diagramme de classe est fondée sur le Polymorphisme par sous-typage dont la classe "Objet" est la classe mère. Toute la hiérarchie des classes filles "Objet" permettent de représenter les différentes catégories et types d'élément.

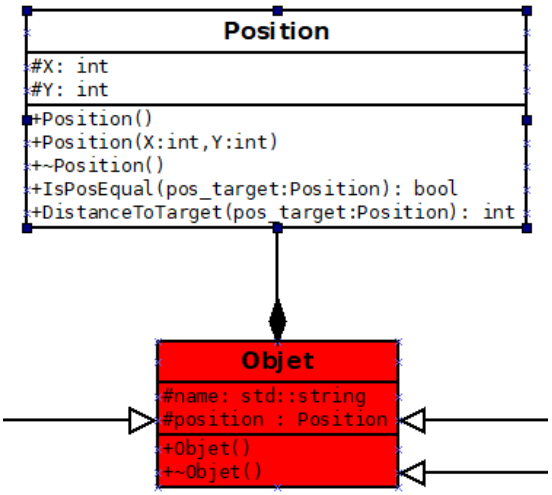

FIGURE 2.1 – Bloc "Objet"

On peut distinguer les classes filles qui héritent directement de la classe "Objet" :

— La classe "Ship" est la classe qui contient toutes les informations des vaisseaux. Chaque vaisseau est associé des statistiques. On associe également par relation de composition une structure "ShipStats" décrivant le type de vaisseau, ainsi qu'une énumération "Ship-TypeID" exposant sa classe de vaisseau.

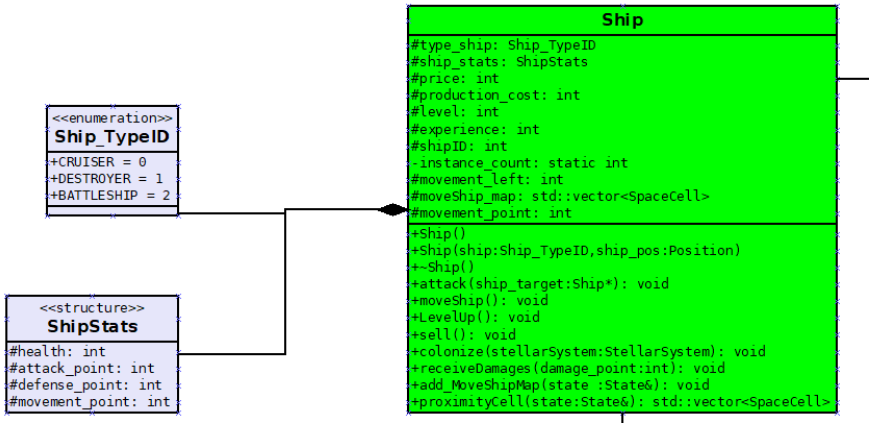

FIGURE 2.2 – Bloc "Ship"

— La classe "StellarSytem" est la classe qui contient toutes les informations sur les systèmes stellaires. On retrouve par exemple la liste des buildings, le type d'environnement du StellarSystem, son statut de colonisation ainsi que sa taille.

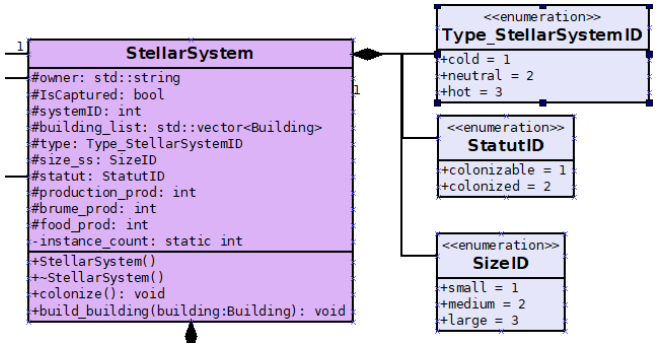

FIGURE 2.3 – Bloc "StellarSytem"

Elle est associée à deux sous classes "Building" et "Planet".

— La classe "Building" est la classe qui contient toutes les informations sur les Buildings. Chaque "Building" est associée à une "Ressources" en particulier, celle que le bâtiment va produire. De plus dans la classe on retrouve le prix dans chaque ressource permettant la construction du building ainsi que sa production par tour. La méthode "upgradeBuilding" permet elle d'augmenter la production du building.

— La classe "Planet" a elle été mise de côté pour simplifier le nomnbre d'objets dans notre jeu, nous avons donc transférer certains de ses attributs et méthodes dans la classe "StellarSystem".

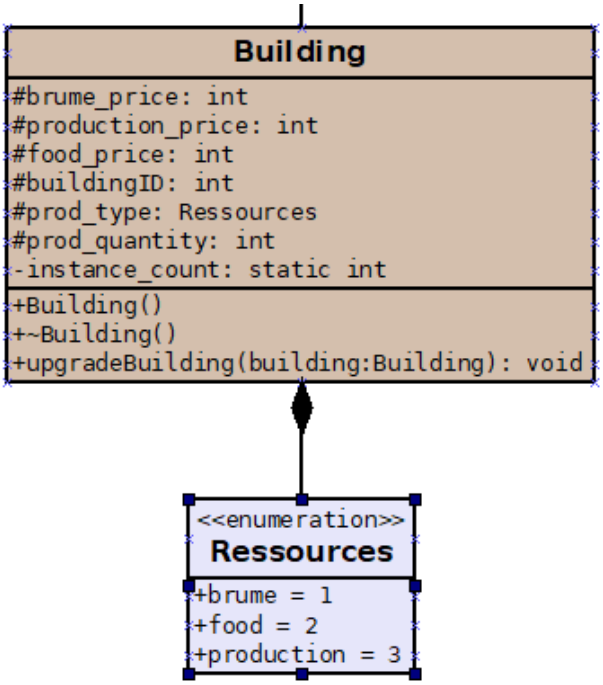

FIGURE 2.4 – Bloc "Building"

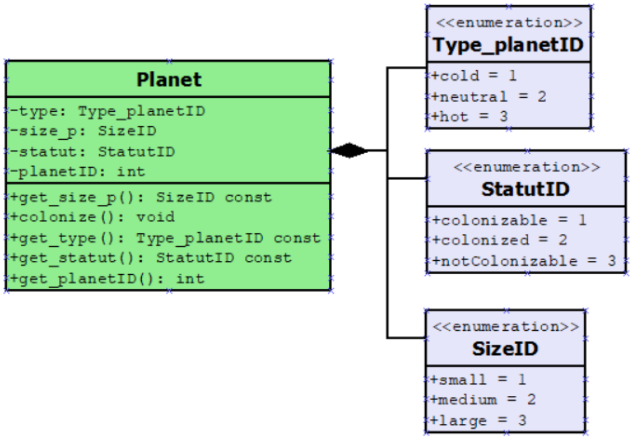

FIGURE 2.5 – Bloc "Planet"

— La classe "SpaceCell" défini chacune des cases de notre jeu, elle permet dans la classe "State" de définir une map.

— La classe " Player" est la classe qui va centraliser l'ensemble des informations concernant le joueur. Le joueur se voit associé des vaisseaux, des systèmes stellaires, des bâtiments,.. Il possède aussi diffèrent type de ressources, " Ressource".

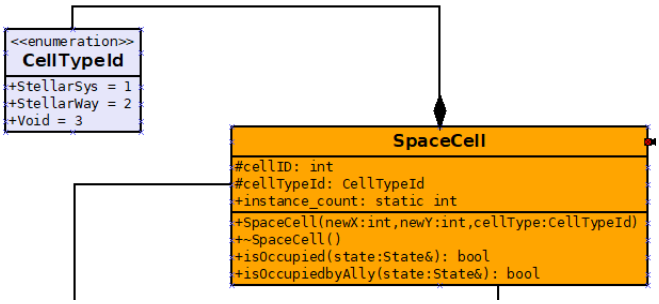

FIGURE 2.6 – Bloc "SpaceCell"

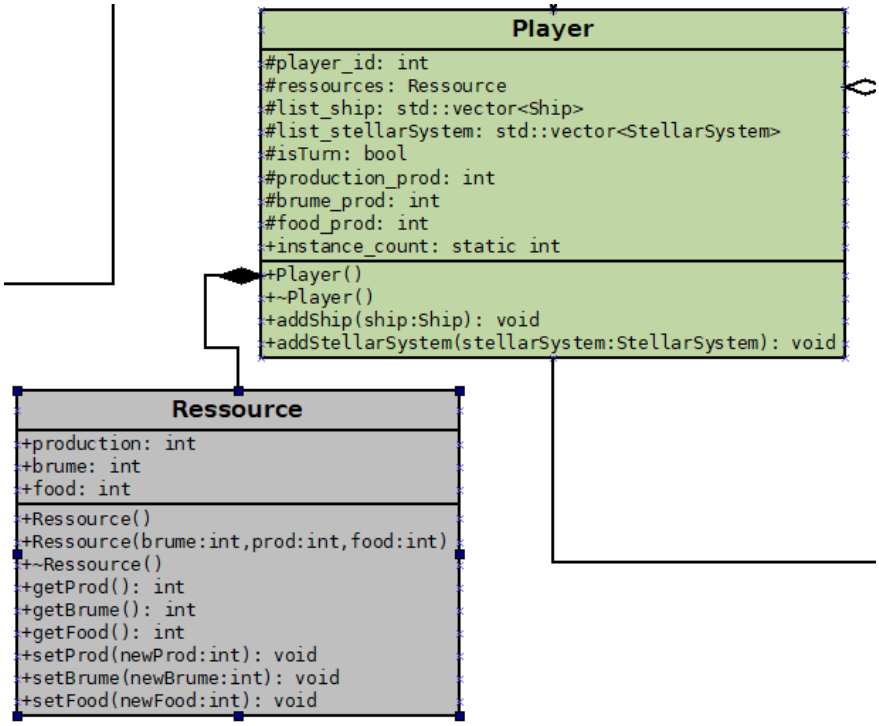

FIGURE 2.7 – Bloc "Player"

— La classe "State" contient les classes "Player" et "SpaceCell". Cette classe donne l'état complet du jeu, la plupart des informations sont dans la classe "Player" et donc dans "Player a" et "Player b".

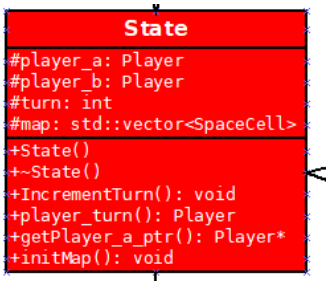

FIGURE 2.8 – Bloc "State"

— Finalement, on peut voir notre diagramme d'états entier sur la figure suivante :

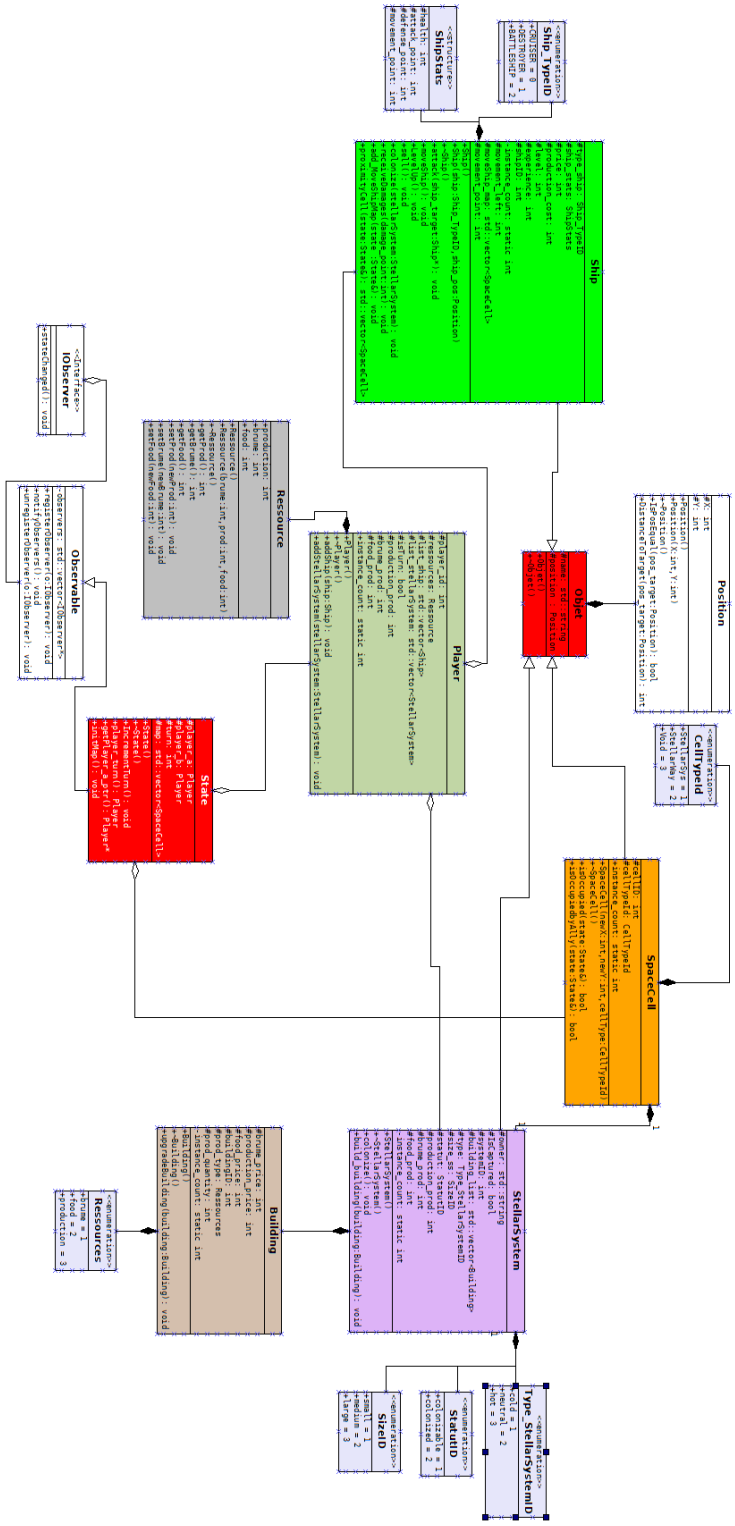

FIGURE 2.9 – State diagram

Section 3

ETATS

3.1 Description des états

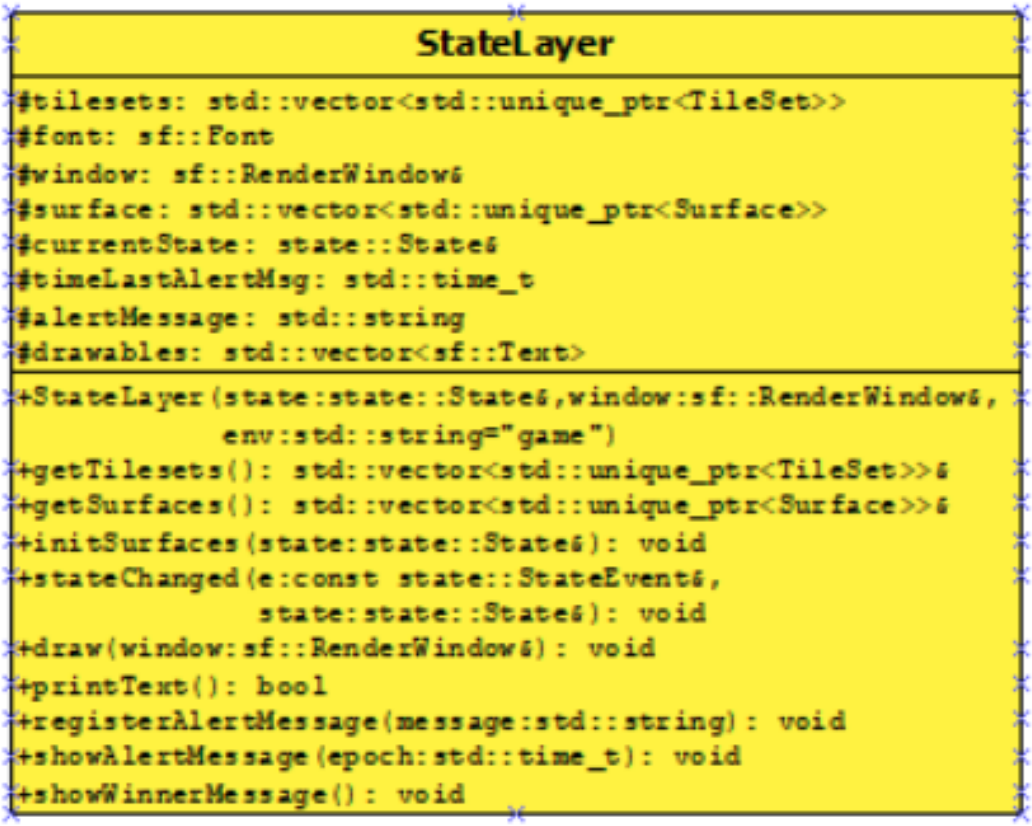

FIGURE 3.1 – Bloc "statelayer"

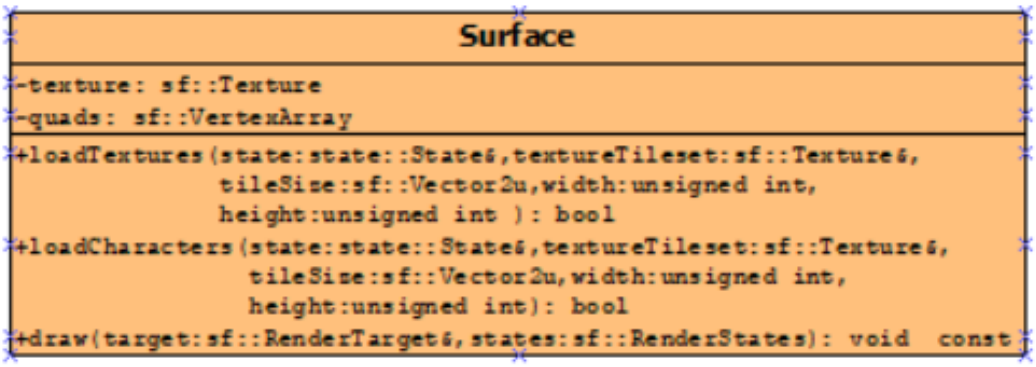

FIGURE 3.2 – Bloc "surface"

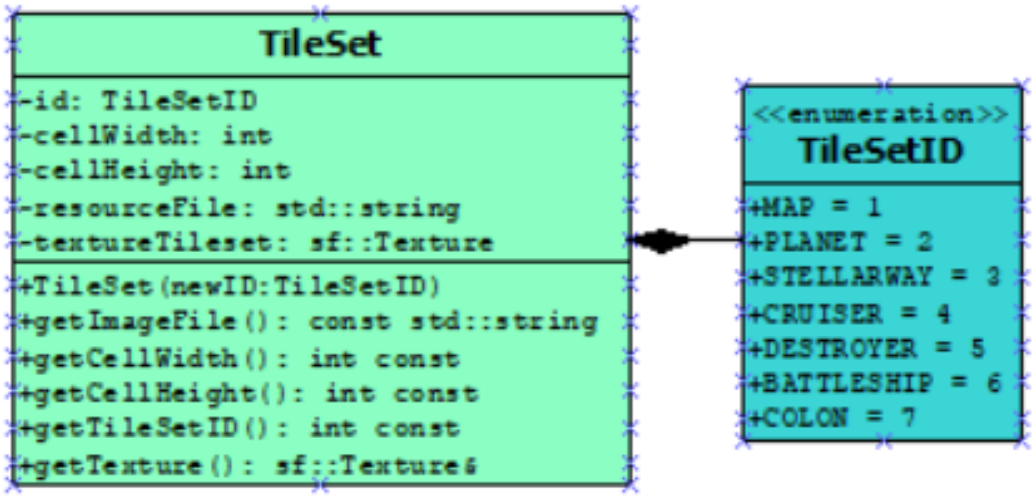

FIGURE 3.3 – Bloc "tileset"

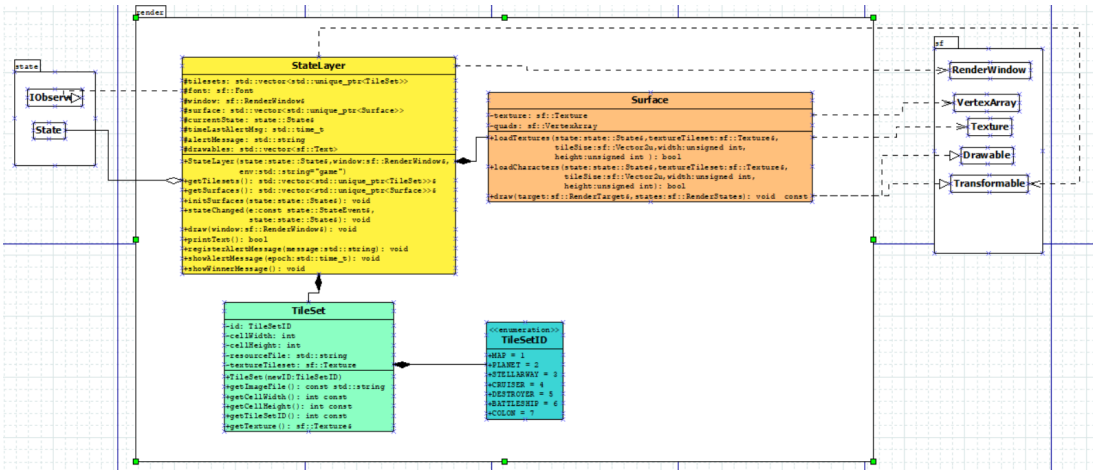

FIGURE 3.4 – Bloc "render"

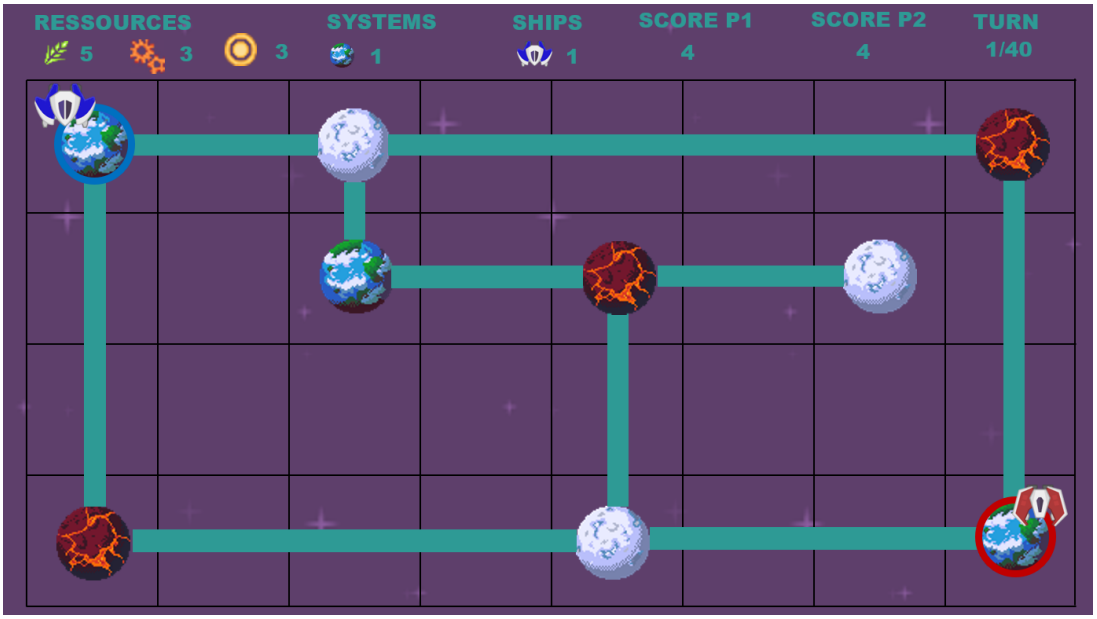

FIGURE 3.5 – Exemple d'affichage

Section 4

Moteur de Jeu

4.1 Actions utilisateur

Le joueur va devoir allier un grand nombre d'actions par tour. Pour cela il aura la possibilité de choisir parmi les commandes suivantes :

- Sélectionner un vaisseau : **SelectShipCommand**.
- Déplacer un vaisseau : **MoveCommand**.
- Attaquer un vaisseau ennemi : **AttackCommand**.
- Coloniser une planète à l'aide d'un vaisseau : **ColonizeCommand**.
- Construire un bâtiment ou un vaisseau : **BuildBuildingCommand** ou **BuildShipCommand**.
- Terminer son tour : **FinishTurnCommand**.

Pour effectuer les commandes concernant un vaisseau, le joueur devra préalablement sélectionner le vaisseau avec lequel il veut effectuer une action.

4.2 Action automatique

A chaque fin de tour, le jeu va venir appeler la fonction "CheckWinnerCommand" afin de déterminer si oui ou non la partie doit s'arrêter.

4.3 Description logicielle

L'ensemble des commandes contiennent un constructeur permettant de leur attribuer leur commandID.
De plus, elles possèdent toutes la méthode "serialize" permettant de mettre sous format Json l'ensemble des informations utiles concernant la commande.

4.3.1 Command

Cette classe permet de décrire l'ensemble des commandes.
Elle contient 2 attributs, l'un pour préciser lequel des joueurs effectue les com-

mandes, l'autre pour préciser l'ID de la commande, c'est à dire la commande à effectuer. La méthode "execute" est elle définie par la méthode du même nom appartenant à la classe fille dont il est question.
L'ensemble des commandes héritent de cette classe.

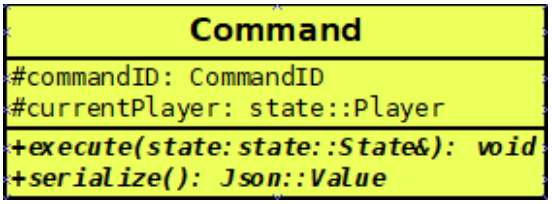

FIGURE 4.1 – Bloc "Command"

4.3.2 SelectShipCommand

Cette commande permet de sélectionner le vaisseau "target" sur lequel on veut effectuer des actions. La méthode execute vient vérifier si le vaisseau ciblé appartient bien au joueur effectuant la commande. Si c'est le cas elle permet alors l'affichage des différentes commandes permettant les actions du vaisseau.

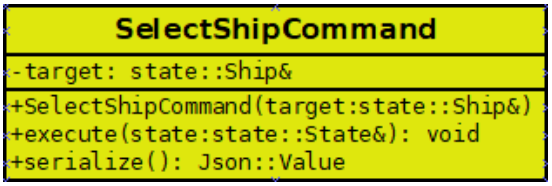

FIGURE 4.2 – Bloc "SelectShipCommand"

4.3.3 MoveCommand

Cette commande possède 2 attributs lui permettant de définir le vaisseau "ShipTarget" à déplacer et la position "PositionTarget" sur laquelle il souhaite le déplacer.
La méthode "execute" vient regarder si le déplacement est possible :
- aucun vaisseau allié est sur la position indiqué,
- la position ciblée est bien une SpaceCell de type "StellarSyst" ou "StellarWay",
- et enfin, si la position ciblée est bien dans la portée de déplacement du vaisseau.
Elle mets ensuite à jour la position du vaisseau ainsi que ses points de mouvement restants.

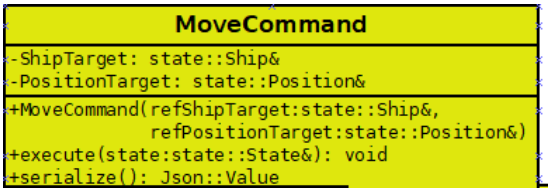

FIGURE 4.3 – Bloc "MoveCommand"

4.3.4 AttackCommand

Cette commande possède 2 attributs définissant le vaisseau "attacker" et le vaisseau "target".
La méthode "execute" vient vérifier que l'action d'attaque est bien possible c'est à dire que les 2 vaisseaux se trouvent bien sur la même position. Elle compare ensuite les types des vaisseaux pour actualiser l'état de chacun d'entre eux (destruction ou perte de point de vie).

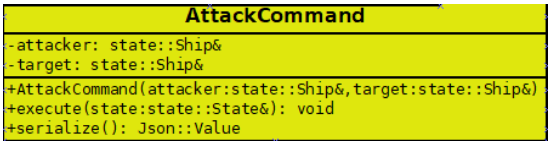

FIGURE 4.4 – Bloc "AttackCommand"

4.3.5 BuildBuildingCommand et BuildShipCommand

Ces 2 commandes ont un fonctionnement similaire. Elles permettent la construction d'un building ou d'un ship sur le système stellaire "StellarTarget".
La méthode "execute" vérifie si la construction est possible :
- StellarTarget est bien un système stellaire appartenant au joueur,
- et le joueur possède le nombre de ressources nécessaire à la construction.
La méthode ajoute donc l'objet construit à la liste de ship ou de building du joueur et actualise la production du système dans le cas de la construction du building.

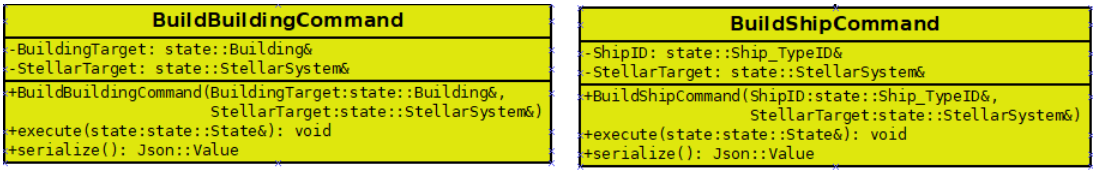

FIGURE 4.5 – Bloc "BuildBuildingCommand et BuildShipCommand"

4.3.6 FinishTurnCommand

Cette commande permet au joueur de mettre fin à son tour, d'incrémenter le nombre de tour dans la classe State pour permettre à l'autre joueur de jouer.
La méthode "execute" vérifie bien que c'est le joueur qui joue le tour qui effectue cette commande.

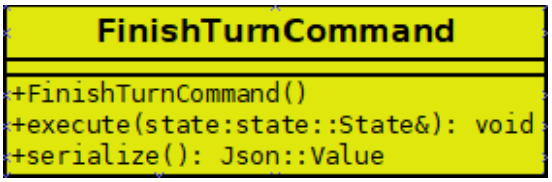

FIGURE 4.6 – Bloc "FinishTurnCommand"

4.3.7 CheckWinnerCommand

Cette commande permet à chaque fin de tour de vérifier si il y a un vainqueur.
La méthode "execute" vérifie que chacun des joueurs possèdent au moins 1 système stellaire ou 1 vaisseau. Dans le cas contraire il déclare vainqueur (victoire militaire) le joueur ayant toujours au moins un de ces objets. Si aucun joueur n'est gagnant à la fin d'un nombre de tour prédéfini (40 par exemple), alors il vient comparer les scores de chacun des joueurs pour déclarer gagnant le joueur ayant le plus de score.

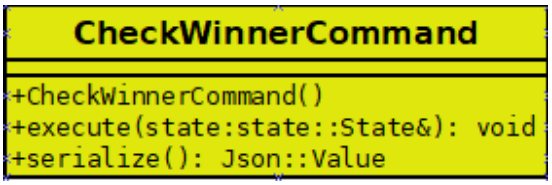

FIGURE 4.7 – Bloc "CheckWinnerCommand"

Section 5

Intelligence Artificielle

5.1 Intelligence Artificielle Aléatoire

Notre Intelligence Artificielle va devoir s'expandre et se développer, nous allons donc dans un premier temps effectuer aléatoirement les actions liées aux vaisseaux puis celles liées aux systèmes. Dans un premier temps, la stratégie pour les vaisseaux suit simplement le diagramme suivant :

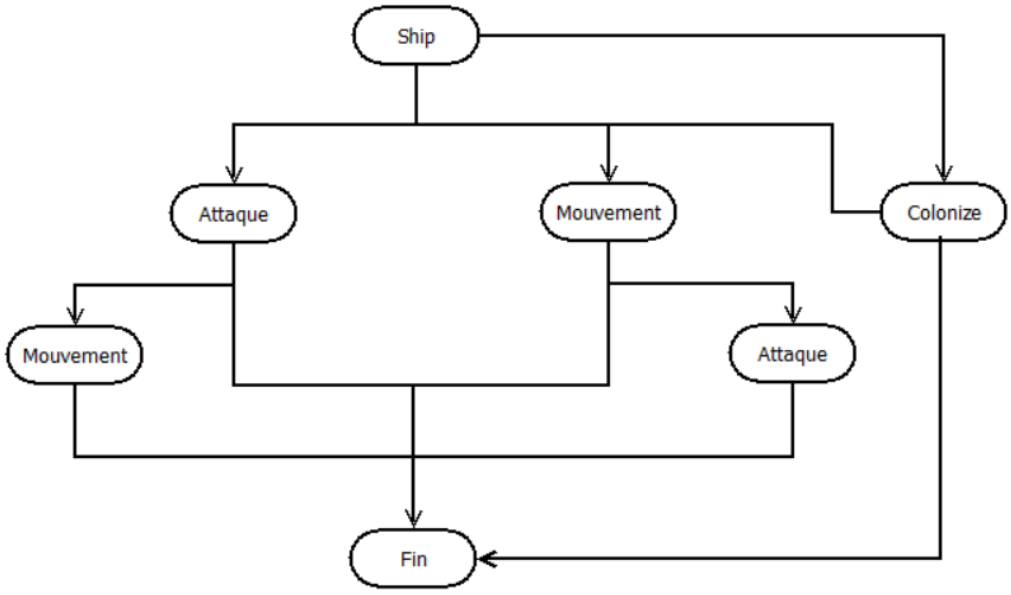

FIGURE 5.1 – Stratégie d'expansion

— L'IA va sélectionner un vaisseau,
— Elle va ensuite choisir aléatoirement une action parmi les 3 disponibles sachant que l'action de coloniser à une probabilité plus élevé que les deux autres,
— Selon l'action qu'elle a effectué elle pourra déplacer ou attaquer, si elle choisit la commande de colonisation et qu'elle est possible le vaisseau est alors détruit sinon on teste les commandes d'attaque et de déplacement,

— Finalement si il reste des vaisseaux disponible, on recommence le même paterne

Dans un second temps, la stratégie de développement des systèmes et la suivante :

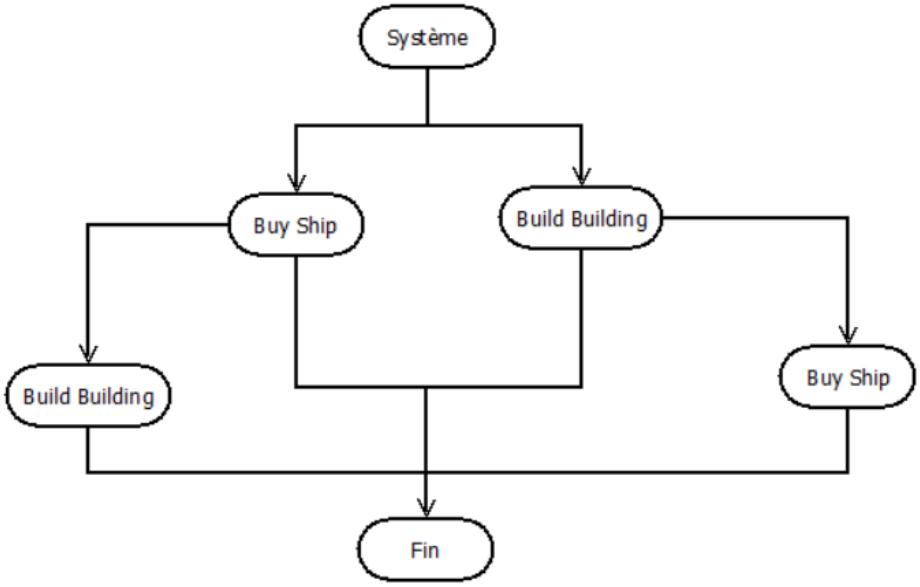

FIGURE 5.2 – Stratégie de développement

— L'IA va sélectionner un système,
— Elle va alors choisir aléatoirement entre construire un bâtiment ou acheter un vaisseau,
— Si une commande est effectuée, on passe à un autre système disponible sinon on test l'autre commande,
— Finalement on passe à un autre système disponible

5.2 Intelligence Artificielle Heuristique

Nous allons maintenant voir l'IA heuristique qui va prendre de meilleurs décisions que l'IA aléatoire. Nous avons implémenter des poids sur certaines commandes en fonction de sa situation actuelle et celle de l'adversaire. Pour la partie expansion. Un vaisseau n'attaquera plus les vaisseaux qui ont un avantages sur lui, il favorisera la colonisation de nouveaux systèmes Pour la partie développement, cela dépendra principalement de la flotte de l'adversaire :

— Si l'IA possède une flotte de vaisseaux qui ont l'avantage sur celle de l'adversaire, l'IA va alors préférer la construction de nouveau bâtiment pour augmenter sa production de ressources.
— Si la flotte de l'IA est plus faible que celle de l'adversaire, l'IA va alors favoriser l'achat de vaisseaux qui contre la flotte actuelle de l'adversaire.