Polybasite

Projet POO : Jeu de la vie

STAMEGNA Clément / MICHON Guillaume

10/02/2018

Rapport Final pour le Projet de POO

Sommaire

1. Introduction

Ce rapport Final a pour but de présenter le projet de Programmation orientée Objet finalisé. Ce projet a été réalisé en Binômes par STAMEGNA Clément et MICHON Guillaume.

Le projet a commencé le 22 Novembre 2017 jusqu'au 15 Février 2018 soit sur une durée de deux mois et demi.

Le projet est produit dans le cadre du cursus Informatique de 4eme Année de Polytech Marseille.

Nous avons choisi le sujet du jeu vidéo.

2. Présentation du Sujet

2.1.L'Objectif

L'objectif principal du projet est d'utiliser les concepts "objet", les mécanismes induits et de multiplier le plus possible les interactions entre objets. Le sujet est totalement libre dans la mesure où il permet d'atteindre l'objectif.

Les années précédentes, les élèves ont majoritairement choisi d'implémenter des "moteurs de jeu", essentiellement des "jeux de la vie" détournés. L'idée directrice étant de faire évoluer des populations communicantes (communication objet) formés par divers d'individus (polymorphisme). L'évolution de ces populations étant dictée par des règles simples : se nourrir, se reproduire, vieillir, se déplacer, communiquer …

L'évaluation du projet se fait sur la base des notions objets utilisés. L'interface graphique associée au moteur de jeu est facultative et laissée à votre discrétion. Elle peut permettre de mieux comprendre les mécanismes mis en œuvre et d'avoir un retour "visuel" mais apporte peu au projet concernant l'aspect objet.

2.2.La Problématique

Quelle Jeu peut-on faire pour nous rapprocher de l'objectif ?

- Utilise les concepts et Mécanismes des langages orientés objets.

- Utilise des objets.

- Fait interagir des objets entre eux.

- Faire de l'héritage entre certains objets.

- Utilise des notions de jeux de la vie.

- Peu d'interactions avec l'utilisateur.

- L'évolution de Population Régies par des règles ou des ordres.

De plus il est légitime de se demander pourquoi faire un jeu pour un projet en langage objet ?

- Le Langage Objet est parfaitement adapté pour La création de jeu vidéo.

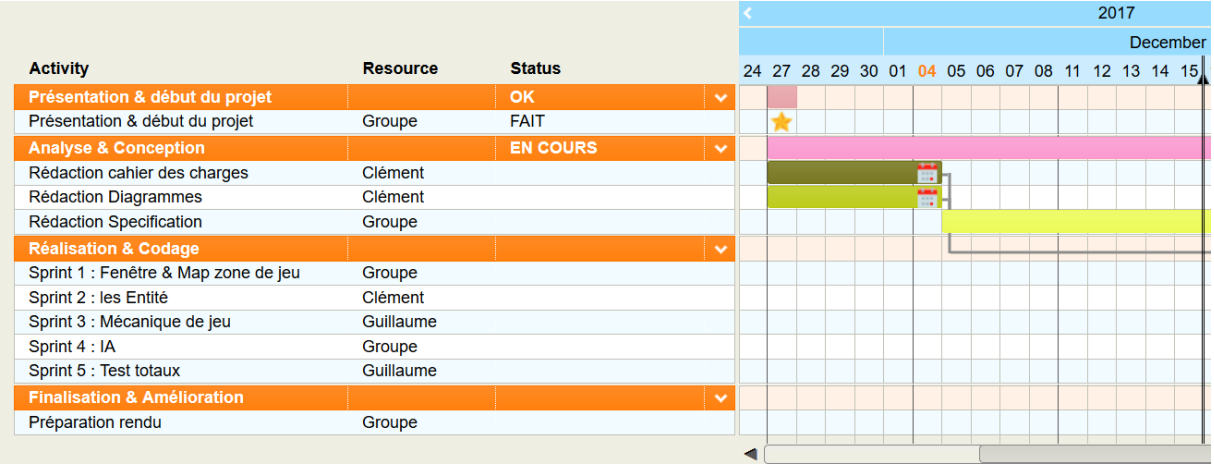
En effet les Concept objets tels que l'héritage permettent de structurer et de factoriser du code sur des entités mères et d'utiliser du code spécifique pour les entités filles.

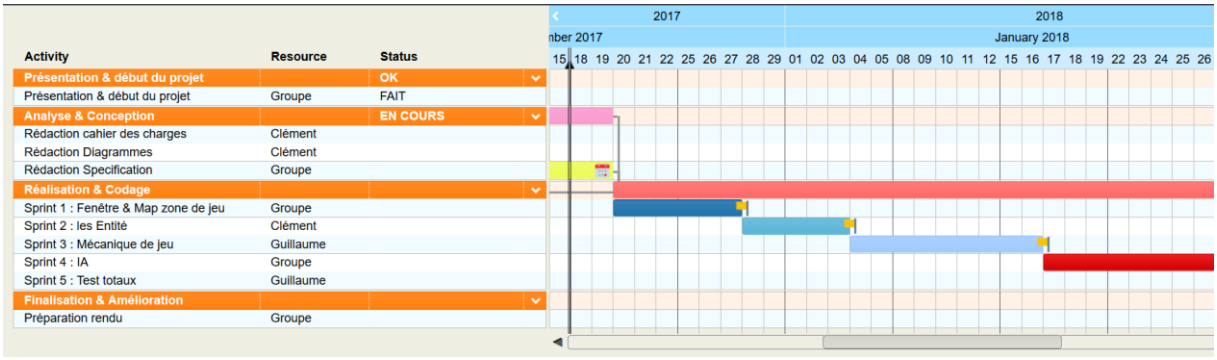
De plus on peut se représentez facilement les entités que possède un jeu par des objets, cela aide pour la conception et la spécification.

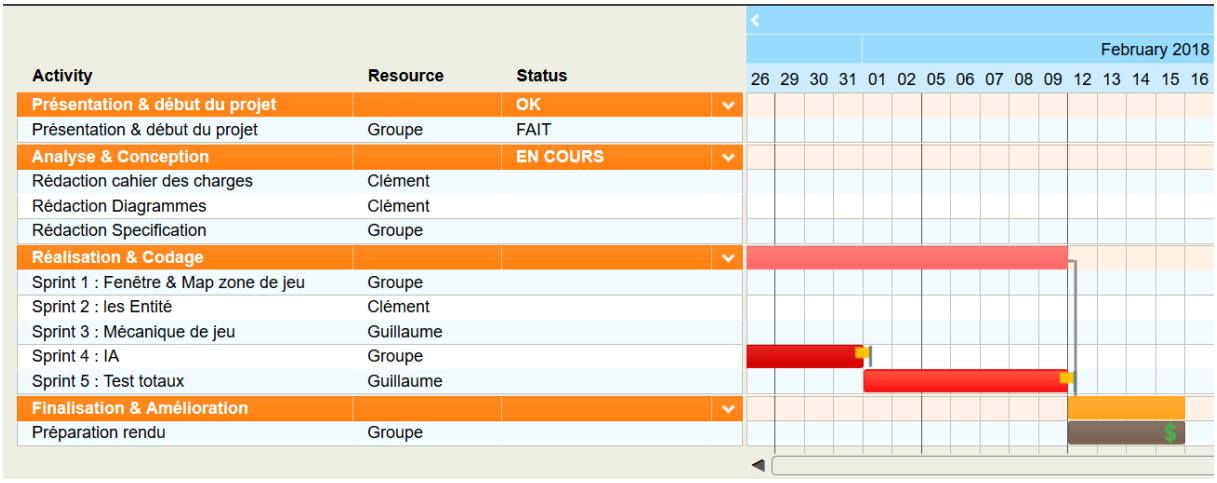
3. Organisation du Travail

3.1.Le planning

Nous avons commencé par produire un planning prévisionnel.







Ce planning nous a servi de fil conducteur tout au long du projet. Nous l'avons respecté lors de chaque tâche. Il y a cependant quelques erreur telle que les rédactions de documents,

puisque le projet est un projet agile nous avons pu continuer la rédaction de nos documents à chaque sprint.

3.2.Découpage des Taches

Nous avons découpé notre projet en deux parties distinctes, La rédaction et la conception des différents documents et la programmation du jeu. Cette répartition a été très efficace pour ce projet, en effet grâce à une communication efficace nous avons bien définie ensemble qu'elle était le but lors de chaque sprint ainsi que les règles que nous voulions spécifier.

3.3.Les Sprints

Notre projet est un projet Agile permettant de faire évoluer notre projet constamment. Nous avons perçu les atouts de l'Agilité lors des divers questions que nous nous sommes posé tous le long du projet.

Cela nous a permis de bien prévisualiser chaque réponse avant d'en choisir une et de bien la spécifier.

Voici les différents Sprints que nous avons réalisés :

3.3.1. Sprint 1 : Fenêtre & Map/Zone de jeu :

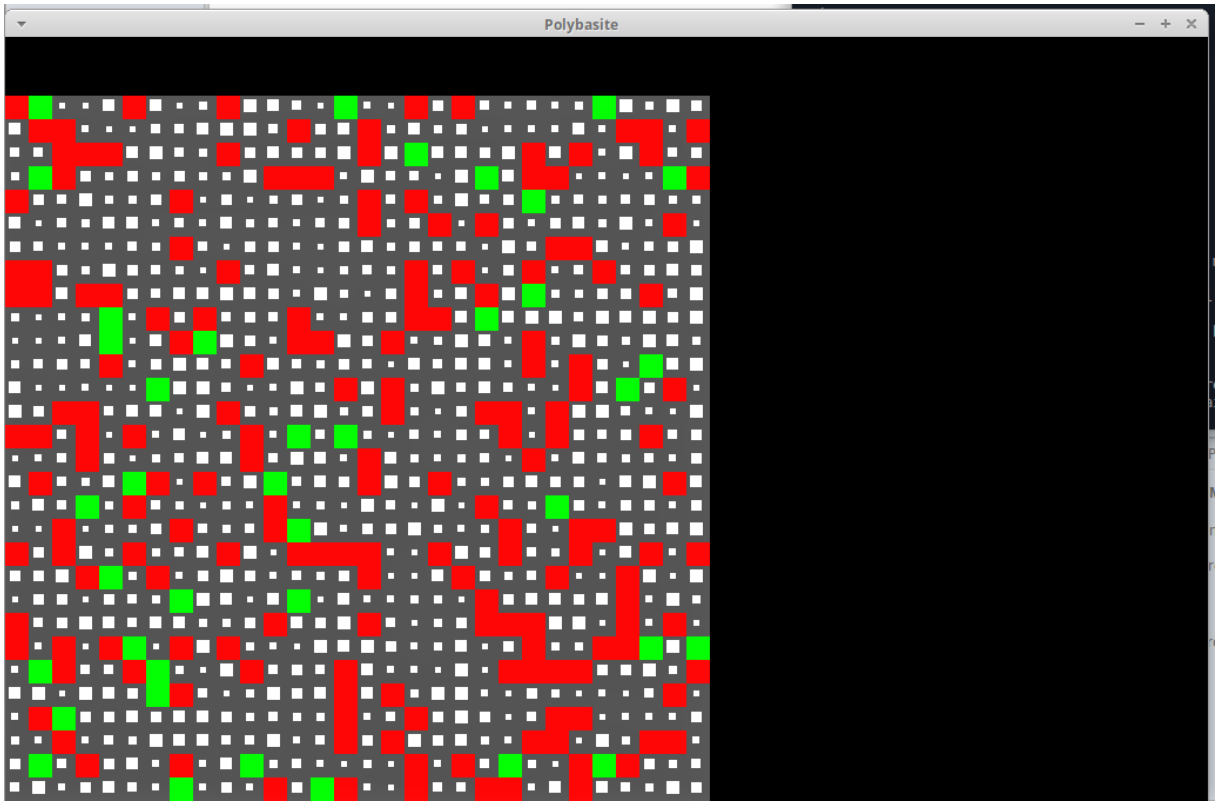
L'objectif :

Réaliser la création de la fenêtre ainsi que la création d'une zone de jeu où les entités pourront être placées et interagir.

Résultats :

La Map est fonctionnelle, c'est une grille d'entités de taille 30 par 30. Nous l'avons pensé pour pouvoir contenir un assez grand nombre d'entités, permettant d'avoir de nombreuses interactions en même temps.

Les entités étaient juste des petits carrés dessinés au hasard.

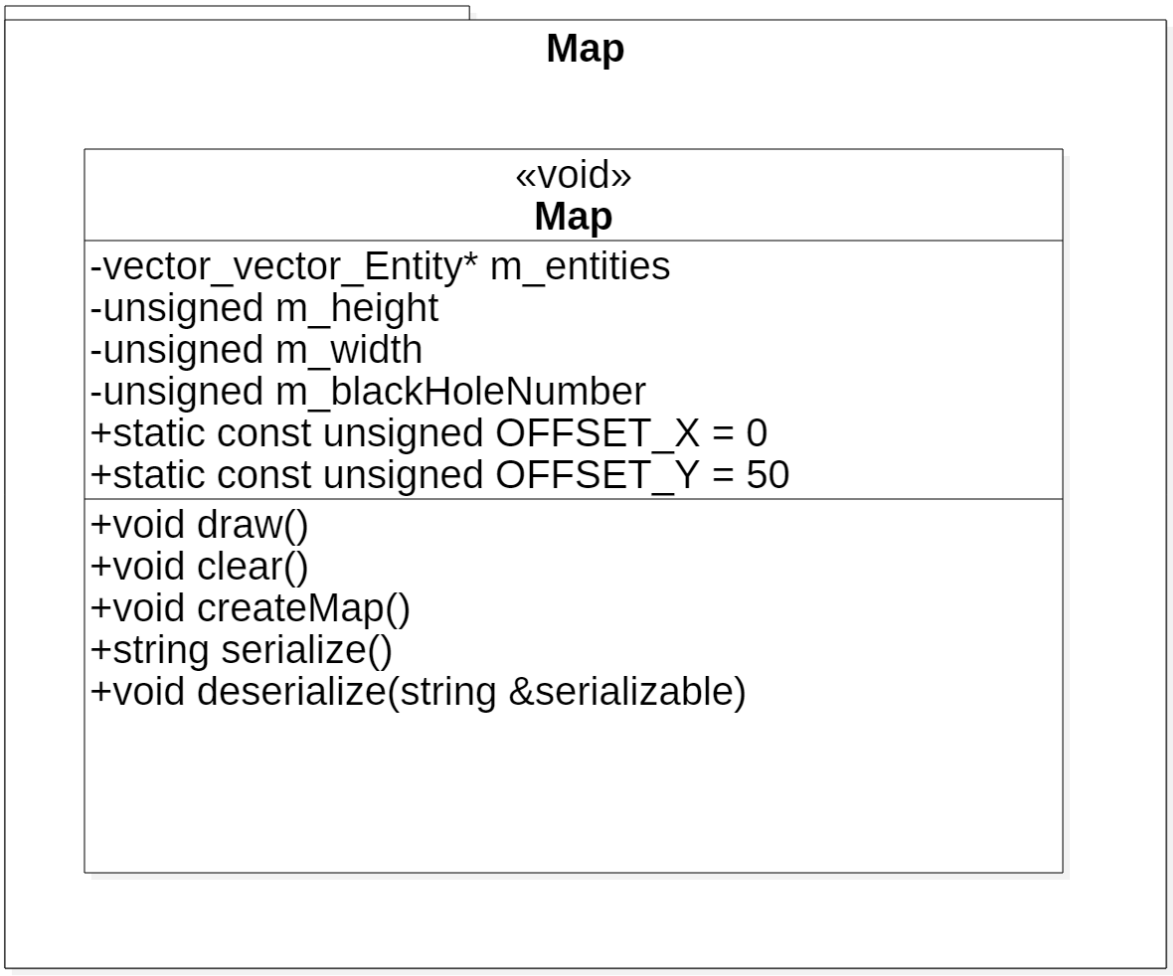


Nous avons rapidement trouvé la méthode pour dessiner notre Map :

```
Draw :
|       Boucle [pour chaque entité dans la Map]
|       |       entité.Draw()
|       |
|       Fin Boucle
```

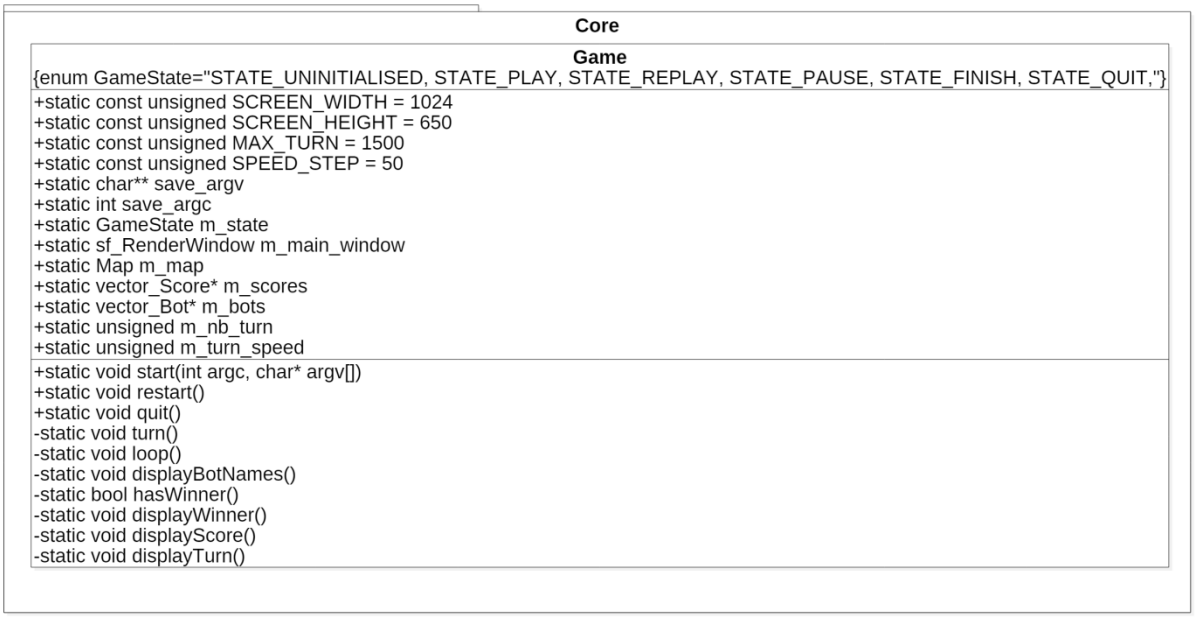
Nous sommes ensuite passés aux entités.

Voici la représentation de notre package Map en fin de projet.



Pour la Fenêtre Nous avons choisi arbitrairement une résolution de 1024 par 650 Car cela permet un bon compromis entre une fenêtre à peu près moyenne et une vision assez claire de l'action, de la zone de jeu et des entités.

Pour la fenêtre nous utilisons la classe principale Game :



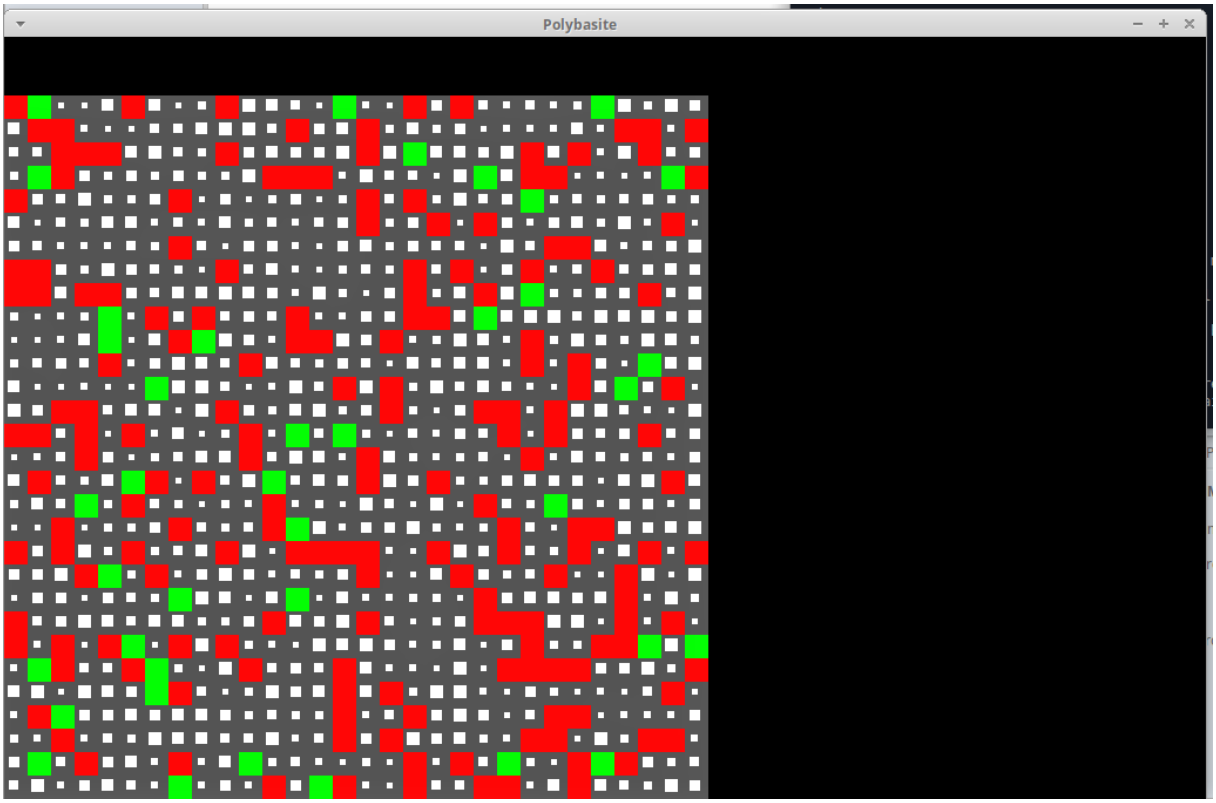
3.3.2. Sprint 2 : Les Entités :

L'objectif :

Créer les différentes entités de notre jeu notamment, les trous noirs, les miner et les basites

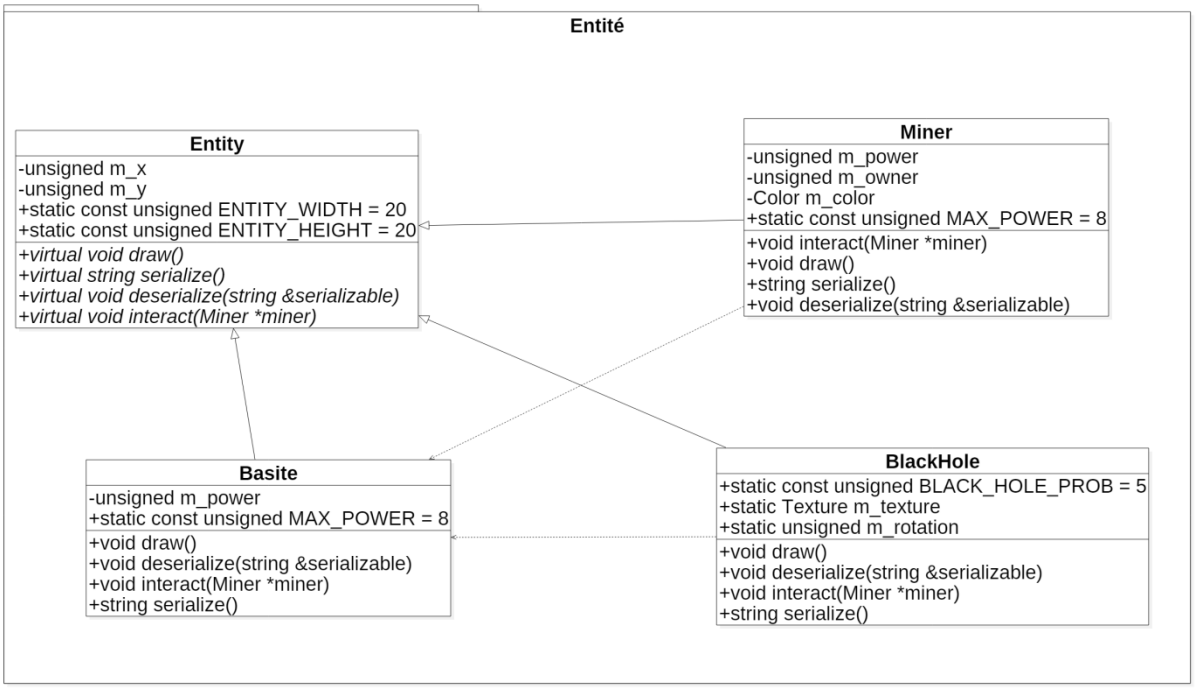
Résultats :

On voit sommairement les différentes entités telles que les Basites (carrés Blancs), les Mineurs (carrés Rouges) et les Trous Noirs (carrés Verts)



Nous avons décidé à ce moment de mettre une statistique de puissance pour les mineurs et les basites, cette statistique sera à la base des interactions entre les entités (Sprint 3 les mécaniques de jeux).

Voici Le Package représentant les entités :



La classe mère Entity nous a permis de factoriser des attributs communs aux entités filles notamment les Coordonnés X et Y. ainsi que du code commun pour les méthodes.

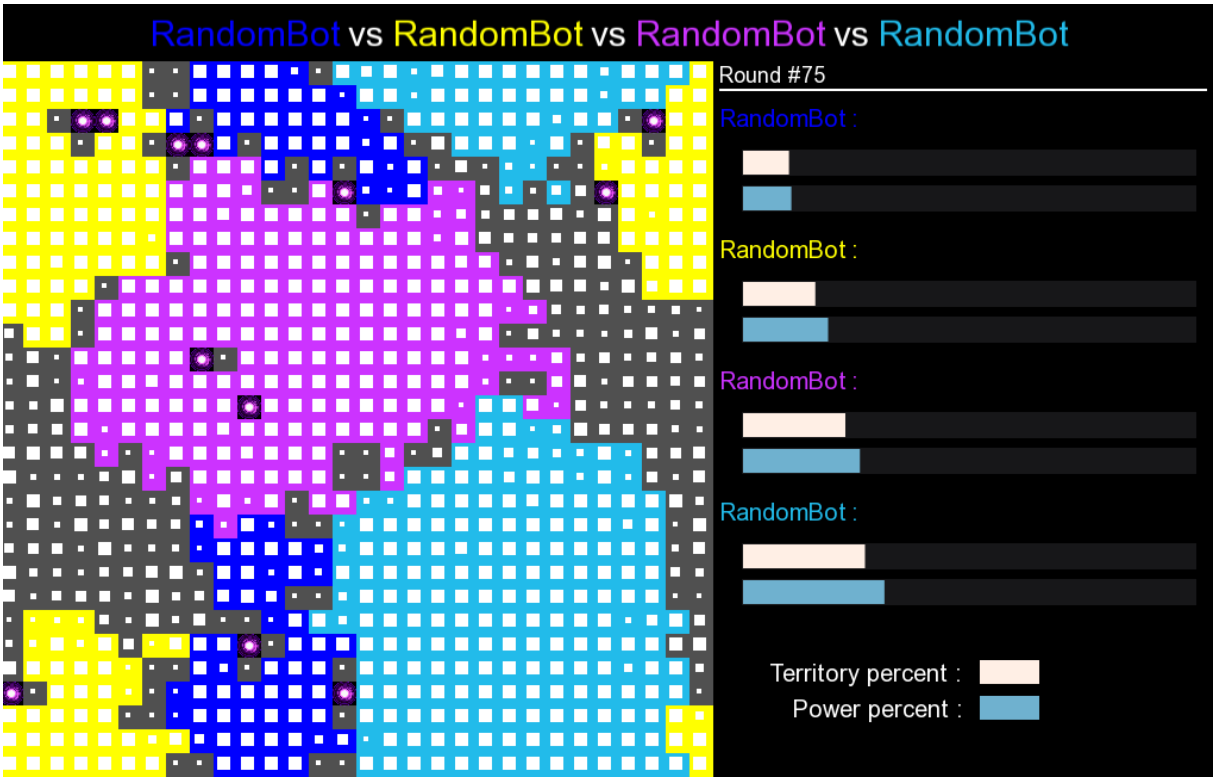
3.3.3. Sprint 3 : Les Mécaniques de Jeux :

L'objectif :

Intégrer les différentes Mécaniques d'interactions entre les différentes entités.

Résultats :

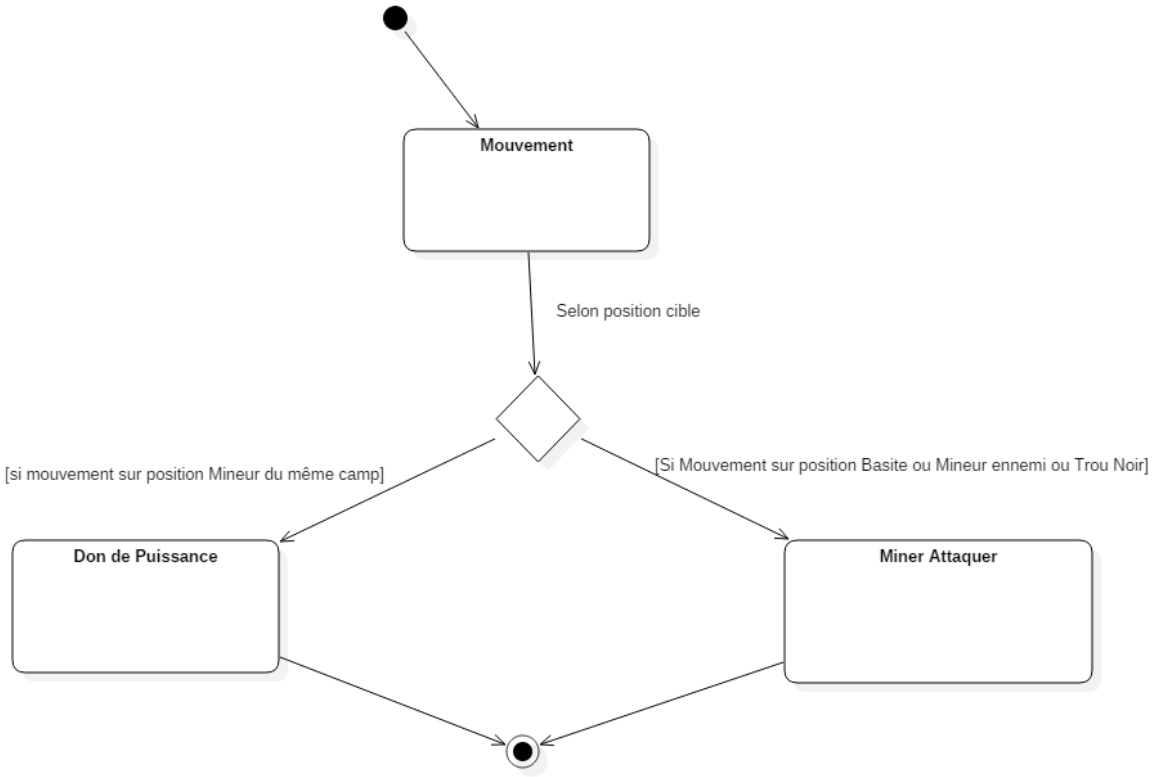
Nos entités peuvent maintenant interagir entre elles.



Chaque entité possède une fonction « interact » qui lui permet de définir les interactions possibles avec un miner qui interagit avec cette entité.

De plus nous avons ajouté un le passage des tours de jeux au niveau de la classe Game et un début pour la gestion des scores. Chaque tour on exécute une seule action pour chaque Mineur de la partie.

Voici un schéma résumant les différentes interactions.



Plus en détails.

Pour les Basite :

- lorsqu'un miner attaque une basite nous arrivons à deux cas soit la puissance du basite est supérieur dans ce cas le mineur meurt mais le basite perd la force correspondante à celle du mineur. (ref : PRJ Polybasite : UC : 03) dans l'autre cas c'est la basite qui meurt et le camp du mineur conquiert la case.

Pour les Mineurs :

- Lorsqu'un mineur attaque un autre mineur, si le mineur est un ennemi il agit comme s'il était une basite, sinon si l'autre mineur est un allié le premier lui donne sa puissance.

Pour les Trous Noir :

- si un mineur attaque un trou noir il ne peut que perdre peut importe sa puissance.

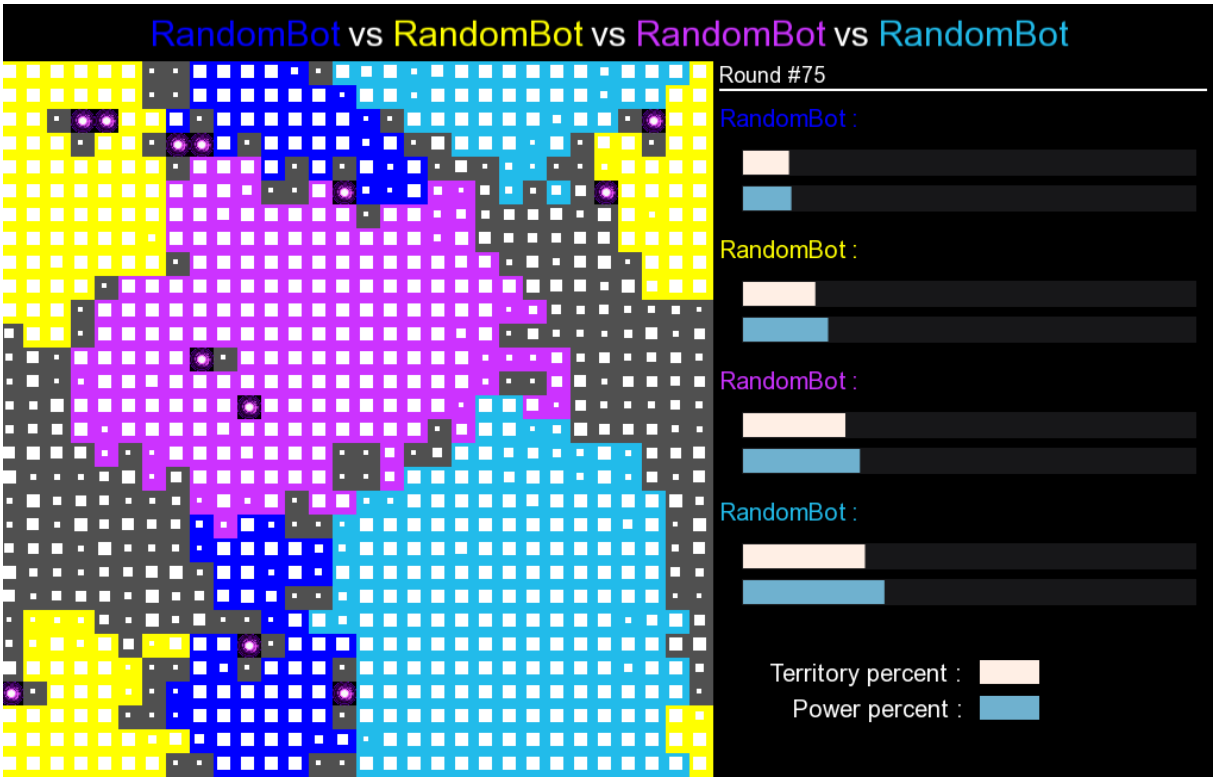
3.3.4. Sprint 4 : l'IA :

L'objectif :

Créer les IA qui commanderont les Miner via certaines commandes et permettre la connexion et le transfert d'informations entre les IA et le jeu.

Résultats :

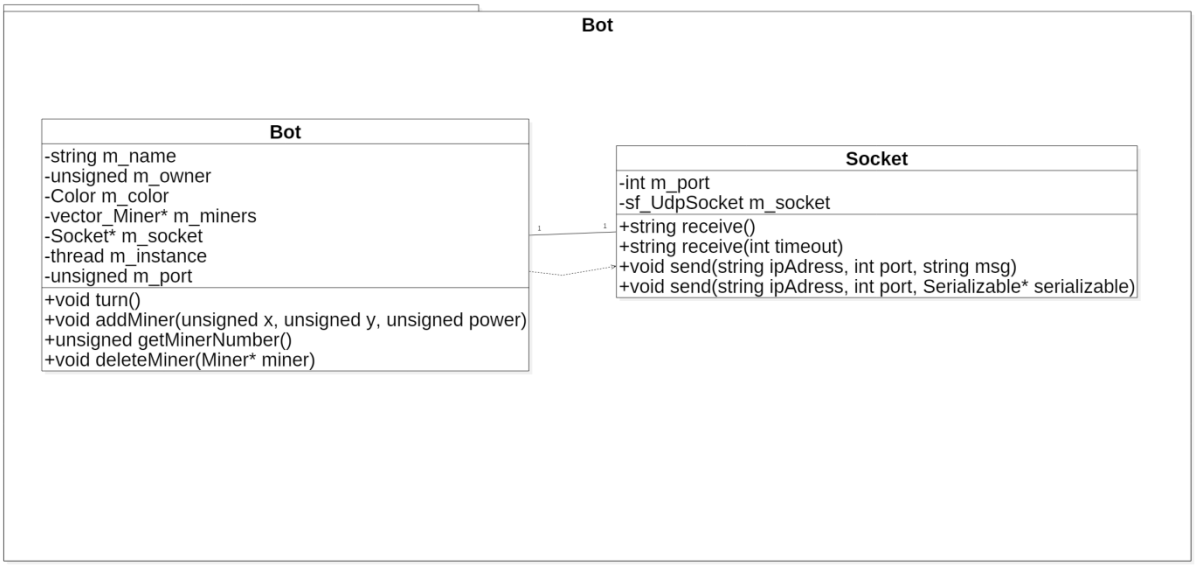
Cette fois les bots sont bien des IA que nous avons créées et que notre logiciel lance via des threads. De plus les IA forment maintenant des camps de mineurs et on peut voir en temps réel l'avancement de chaque IA (en pourcentage de territoires gagnés ainsi que leurs forces en pourcentage de pouvoir possédé) via les barres de progression sur le côté droit.

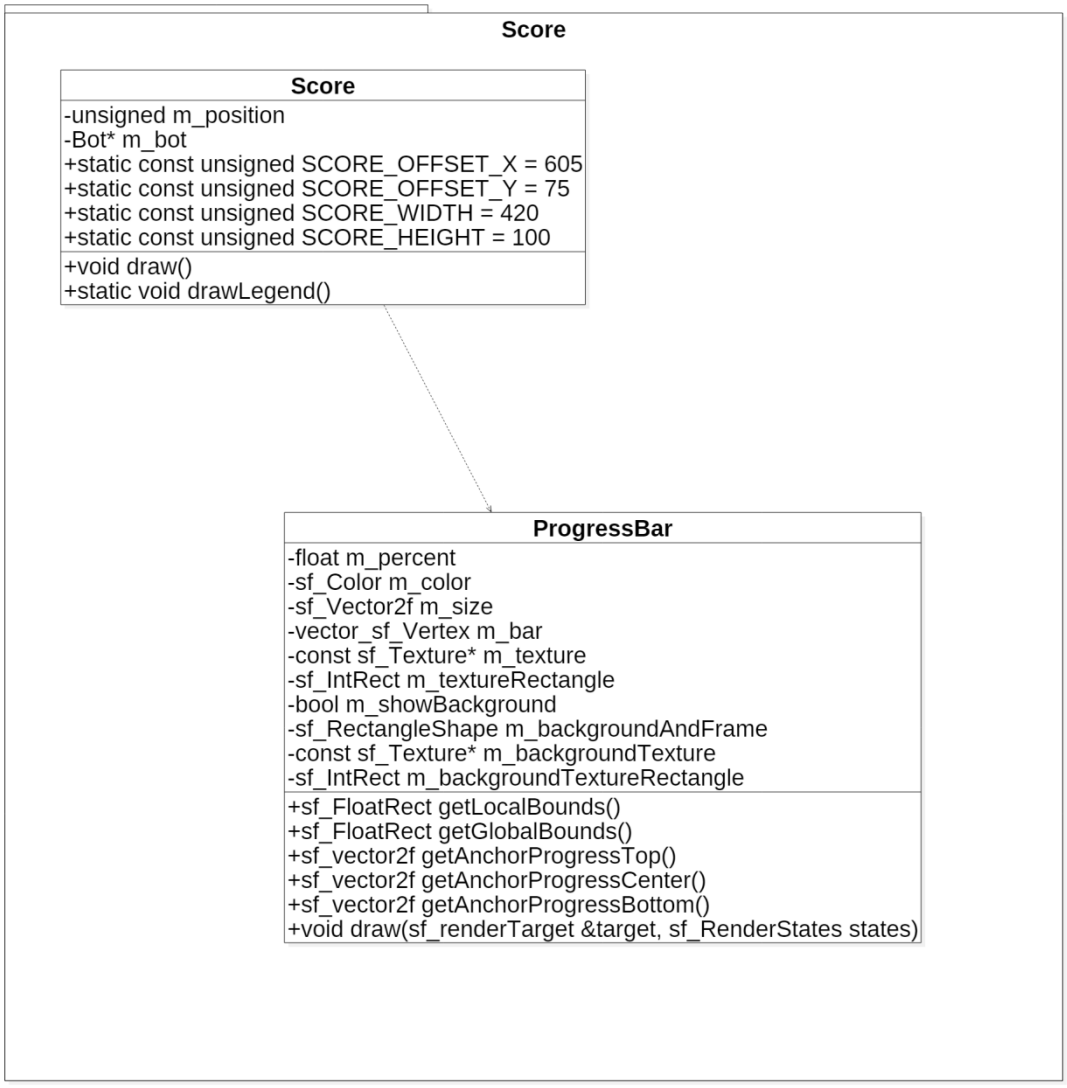


Chaque IA reçoit au début du jeu une couleur pour représenter son camp ainsi qu'un mineur de la même couleur.

Elle doit alors vaincre les autres IA ou bien devenir la plus puissante avant le nombre maximal de tours soit 1500 tours.

Voici les classes Scores et Bot :





3.3.5. Sprint 5 : Test :

L'objectif :

Tester les différentes fonctionnalités créées ainsi que leurs mises en relation avec le jeu.

Résultats :

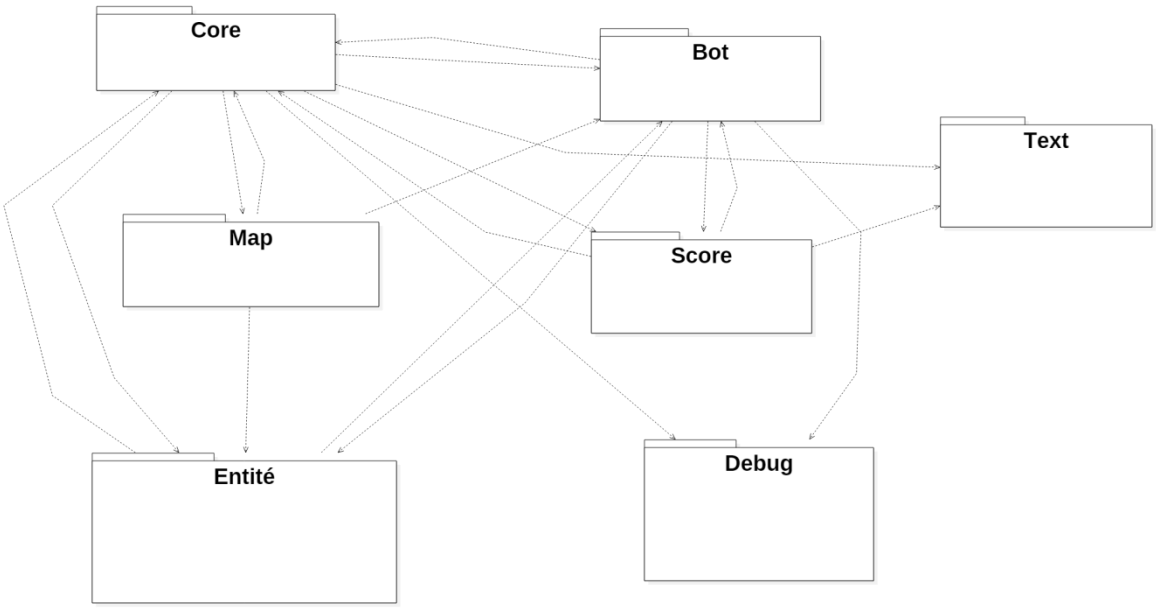
Nous avons testé plusieurs cas d'utilisation telle que

- la communication entre les IA et le jeu.

- les Affichages des différentes entités et des zones de scores et de jeu.

3.4.Modélisation

Notre modélisation bien que assez complexe peut se représenter via un diagramme de packages.



On voit qu'il y a énormément de dépendance entre nos packages, seul les quelques packages de Débug et de Text sont assez indépendant. Pour le reste le jeu nécessite vraiment chaque composant que ce soit les entités, les scores, les bots et la map.

4. Difficultés et Solutions

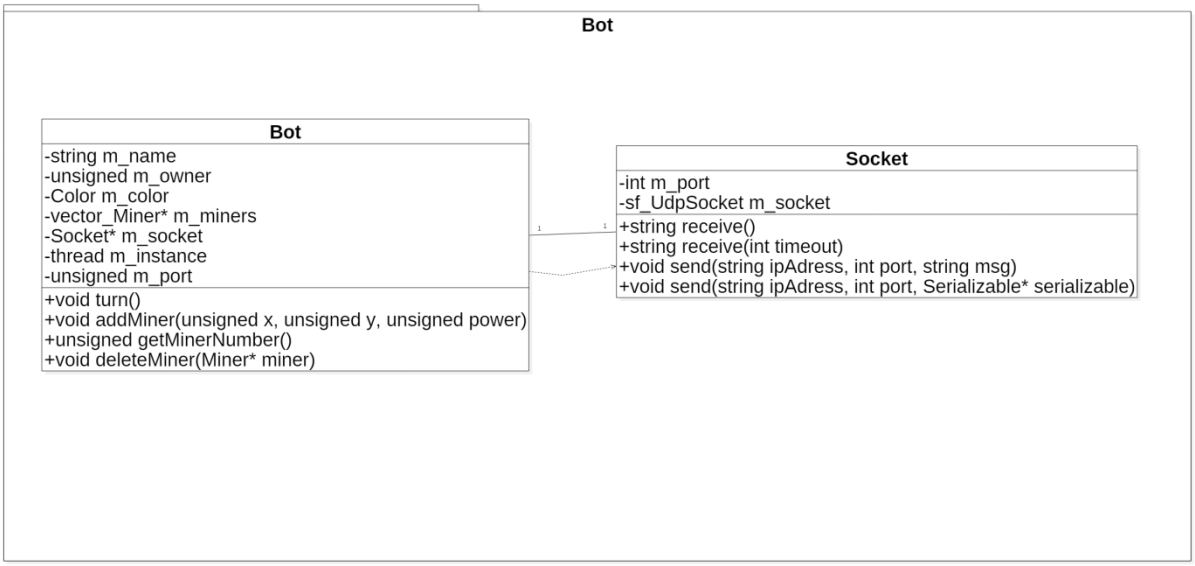
4.1.Les IA

Notre difficulté sur les IA a été le lancement des IA ainsi que la communication avec le jeu.

Nous avons utilisé pour cela les sockets internes permettant la communication en UDP sur l'adresse 127.0.0.1

Pour Lancer les IA nous avons choisi que notre projet va lancer les IA dans des threads différents et garde l'instance dans l'objet Bot pour pouvoir agir sur celui-ci

Par exemple, Arrêter le thread pour arrêter l'IA lorsqu'elle perd.



Nous avons alors utilisé des méthodes de Sérialisation (qui transforme les commandes et informations de jeu en chaines de caractère) pour pouvoir communiquer plus facilement entre les IA et le Jeu. En effet les chaines de caractères sont plus exploitables sur des paquets de transfert.

Exemple : Serialize de Map :

```
54  std::string Map::serialize() {
55      std::ostringstream os;
56      os << "Map:" << m_width << ";" << m_height << ";";
57
58      for(unsigned i = 0; i < m_width; ++i) {
59          for(unsigned j = 0; j < m_height; ++j) {
60              os << m_entities[i][j]->serialize() << ";";
61          }
62      }
```

Exemple : Serialize de Miner :

```
44  std::string Miner::serialize() {
45      std::ostringstream os;
46
47      os << "Miner:" << m_x << ";" << m_y << ";" << m_power << ";" << m_owner;
48
49      return os.str();
50  }
```

5. Conclusion

En conclusion ce Projet en Programmation Orienté Objet nous a permis de mettre en œuvre la Programmation orientés Objet dans la création de jeu de type « jeux de la vie ».

Cela nous a permis d'apprendre à mettre en place un projet Agile ainsi que les atouts majeurs d'un processus agile

- Excellente réactivité vis-à-vis du client
- Qualité accrue du fait de la présence des PO
- Favorise et facilite la communication avec les autres membres de l'équipe
- Développements hors sujet très peu probable

De plus nous avons eu une vraie plus-value sur la création de documents de projet telle qu'un cahier de Spécification, les documents de Conception, les fiches de tests, toute la partie Génie Logiciel qui sera un vrai plus pour notre métier futur.