Rapport du Projet de Programmation Avancée

Projet réalisé par :

Alex Audebert

Eléa Roche

Réalisé à l'Ecole Nationale Supérieure de Cognitique

Introduction **2**

Présentation du projet **2**
- Contraintes techniques 2
- Contraintes fonctionnelles 2
- Notre jeu 2

Notice d'utilisation **3**
- Les mécanismes 3

Diagramme des classes, conception et fonctionnalités **4**
- Diagramme de classe 4
  - Ce projet utilise la programmation orientée objet. Nous avons utilisé 4 classes et leurs classes filles ainsi qu'une structure 4
  - Classe 4
  - Structure 4
  - InConstructionBuilding 4
  - Lancement du jeu 5
  - Les colons 6
  - Les bâtiments 8

Justifications techniques **10**
- Les bâtiments en construction 10
- La construction des bâtiments 10
- La disponibilité des bâtisseurs 10
- Les vérifications de chaîne d'entrée 11
- Les tests 12

Gestion de projet **12**
- Organisation 12
- Commentaire 12

A vous de jouer **12**

Bilan **13**
- Amélioration 13

Introduction

Ce rapport réalisé dans le cadre d'un projet de l'UE Programmation Avancée ayant pour objectif d'apprendre la programmation orientée objet en langage C#.

Ce document vise à expliquer la conception et les fonctionnalités de notre projet, ainsi que de justifier les choix techniques faits et de détailler comment utiliser l'application. Il est aussi spécifié les aspects de gestion de projet et le bilan.

Présentation du projet

Contraintes techniques

Ce projet est un projet de programmation orienté objet codé en c#, en développement console. Le code source respecte les normes de programmation c# (convention de nommage, de disposition, de commentaire) et est en anglais.

Contraintes fonctionnelles

Ce jeu a pour principe de gérer une colonie, que ce soit dans le développement de cette dernière, ou dans sa survie. Les colons ne doivent pas être contrôlés directement par le joueur mais seront contrôlés automatiquement par le jeu en fonction des actions que réalisent le joueur. Ces actions sont réalisées en tour par tour, ce qui permet de mettre en place une notion de temps dans le jeu

Notre jeu

Le thème que nous avons choisi pour la réalisation de ce jeu est la création du village olympique (pour les Jeux Olympiques de 2022). Le village contiendra donc la colonie, ainsi que toutes les infrastructures nécessaires à ce village olympique. Parmi ces infrastructures il y aura des hôtels et des restaurants, qui permettront d'accueillir les colons. Il y aura aussi toutes les infrastructures sportives nécessaires pour l'entraînement et la compétition. Il sera par exemple possible de créer une piscine olympique, une piste d'athlétisme, une salle de musculation, un terrain de tennis, un terrain de football, un terrain de volley ainsi qu'un terrain de basket.

De plus, la colonie est composée de 3 types de colons différents. Les bâtisseurs qui construisent les infrastructures énoncées précédemment. Les sportifs qui s'entraînent dans leur sport afin d'augmenter de niveau, ainsi que les coachs sportifs qui permettent d'entraîner les sportifs plus efficacement. Chaque sportif recruté a obligatoirement un sport, correspondant à l'une des infrastructures sportives cités précédemment (footballeur, basketteur, tennismen, volleyeur, athlète, crossfitter ou nageur).

Le village est initialement composé d'un Hôtel, d'un Restaurant et de 3 bâtisseurs. Nous avons fait ce choix afin que le jeu puisse commencer. Sans ça aucune action ne serait possible car on ne peut pas construire de bâtiments si on n'a pas de colon et on ne peut pas recruter de colons sans bâtiments.

Notice d'utilisation

Les mécanismes

Il n'est possible de recruter un colons (peu importe que ce soit un bâtisseur, un coach ou un sportif) que s'il y a encore une place disponible dans un hôtel et un restaurant. Un hôtel peut accueillir jusqu'à 5 colons, et un restaurant 10.

De plus, il n'est possible de recruter un sportif que s' il y a l'infrastructure sportive correspondant à sa discipline présente dans le village (par exemple on ne peut pas recruter de basketteur s'il n'y a pas de terrain de sport collectif intérieur).

Enfin, il faut un certain nombre de bâtisseurs pour construire un bâtiment (2 pour un hôtel, 3 pour un restaurant et 5 pour une infrastructure sportive). Il est donc possible de construire le bâtiment seulement s'il y a le bon nombre de bâtisseurs disponibles pour cela. Lorsqu'un bâtisseur est en train de construire un bâtiment, il est donc indisponible le temps de terminer son action.

Comme tout être humain, les colons ont besoin de se nourrir, ainsi que de dormir. Ils sont rendus indisponible le temps de se rendre à l'hôtel pour dormir, ou encore de se rendre au restaurant pour manger. La restauration et le repos sont gérés automatiquement par le jeu.

De plus, les colons doivent se déplacer avant de faire leurs activitées, ils doivent :

- Aller à l'emplacement du bâtiment pour le construir
- Aller sur l'emplacement de leur restaurant pour manger
- Aller sur l'emplacement de leur Hotel pour dormir
- Aller à l'emplacement de leur infrastructures sportive pour s'entraîner ou entraîner

Les colons sont indisponibles quand ils se déplacent vers leurs activités.

Chaque type de bâtiment à un nombre de tour avant d'être construit :

- Hotel : 2 tours
- Restaurant : 2 tours
- Infrastructure sportive : 4 tours

Ce nombre de tours est additionné au temps que mettent tous les bâtisseurs à arriver à l'emplacement avant que le bâtiment ne soit construit.

Les bâtisseurs ont pour rôle de construire les bâtiments, alors que celui des sportifs est de s'entraîner afin d'acquérir le niveau nécessaire pour participer aux Jeux Olympiques, et celui des coach d'aider les sportifs à améliorer leur niveau. Si un coach est disponible dans le village quand on demande à un sportif de s'entraîner, le coach accompagne le sportif

durant son entraînement. A la place de gagner 1 session, le sportif gagne alors 2 sessions. Au bout de 2 sessions, le sportif passe au niveau 1.

Pour gagner 2 sportifs doivent être au niveau 1 (on a choisi le niveau 1 pour que ce ne soit pas trop long à tester).

Diagramme des classes, conception et fonctionnalités

Dans cette partie nous allons vous présenter un diagramme de classe partie par partie ainsi que nos différentes fonctionnalités et choix de conception.

Diagramme de classe

Ce projet utilise la programmation orientée objet. Nous avons utilisé 4 classes et leurs classes filles ainsi qu'une structure

- Classe
  - Simulation
  - Village
  - Building
    - Hotel
    - Restaurant
    - SportInfrastructure
  - Settler
    - Athletic
    - Coach
    - Builder
- Structure
  - InConstructionBuilding

Lancement du jeu

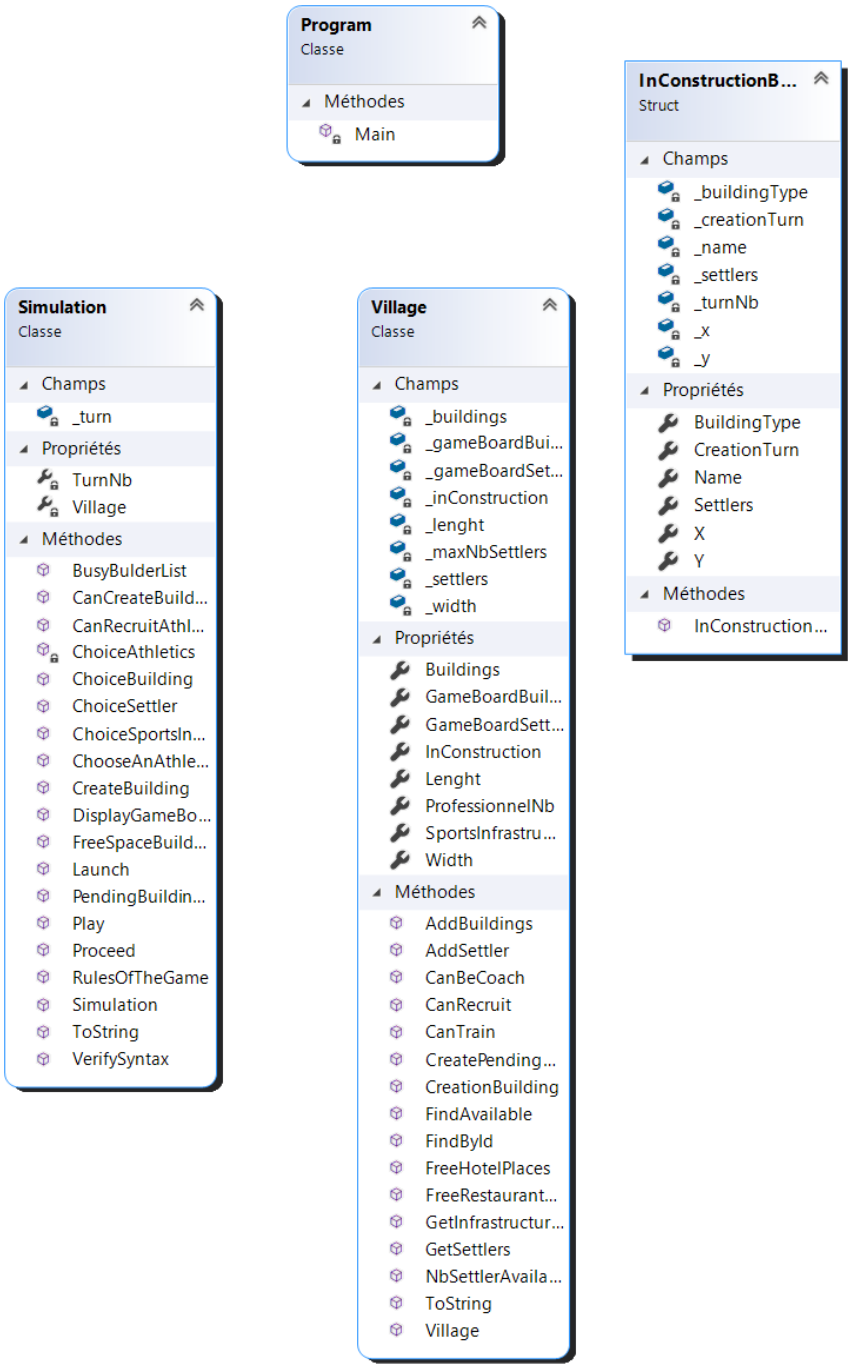

Diagramme des classes Simulation et Village et de la structure InConstructionBuilding

Le programme.cs permet de lancer une simulation, qui est ici une partie de jeu.

La simulation crée un village olympique composé de bâtiments (_buildings) et de colons (_settlers). Le village contient aussi un plateau de jeu contenant l'emplacement des bâtiments (_gameBordBuilding) et un second contenant l'emplacement des colons

(_gameBoradSettler). La simulation contient une méthode “Play()” qui permet de lancer le jeu et de gérer la partie.

Dans l’état actuel du jeu une simulation à un seul village et un village appartient à une simulation.

Un village peut avoir de 4 à n colons et de 2 à n bâtiments.

La classe Settler et la classe Building sont des classes abstraites en effet elles ne peuvent pas exister seul.

Les colons

Nous avons créé une classe colon (Settler) dont hérite trois classe pour nos trois types de colons différents :

- Bâtisseur (Builder)
- Athlète (Athletic)
- Coach

Chaque colon à des coordonnées (x et y), un niveau de faim et d'énergie (energyState et hungerState) et des variables booléennes pour savoir si le colon est disponible, si c’est l’heure de manger ou de dormir. Selon que ce soit des “Athlete” des "Bâtisseur" ou des “Coach” le type est différent ainsi que la valeur dont le niveau de faim diminue à chaque tour (hungerdecreasing) et celui d'énergie (energyDecreasing). Enfin la fonction play() de chaque type de colon diffère. Pour chaque type à chaque tour le niveau de faim et d'énergie diminue. Selon la demande du joueur, les athlète s'entraînent et les bâtisseur bâtissent.

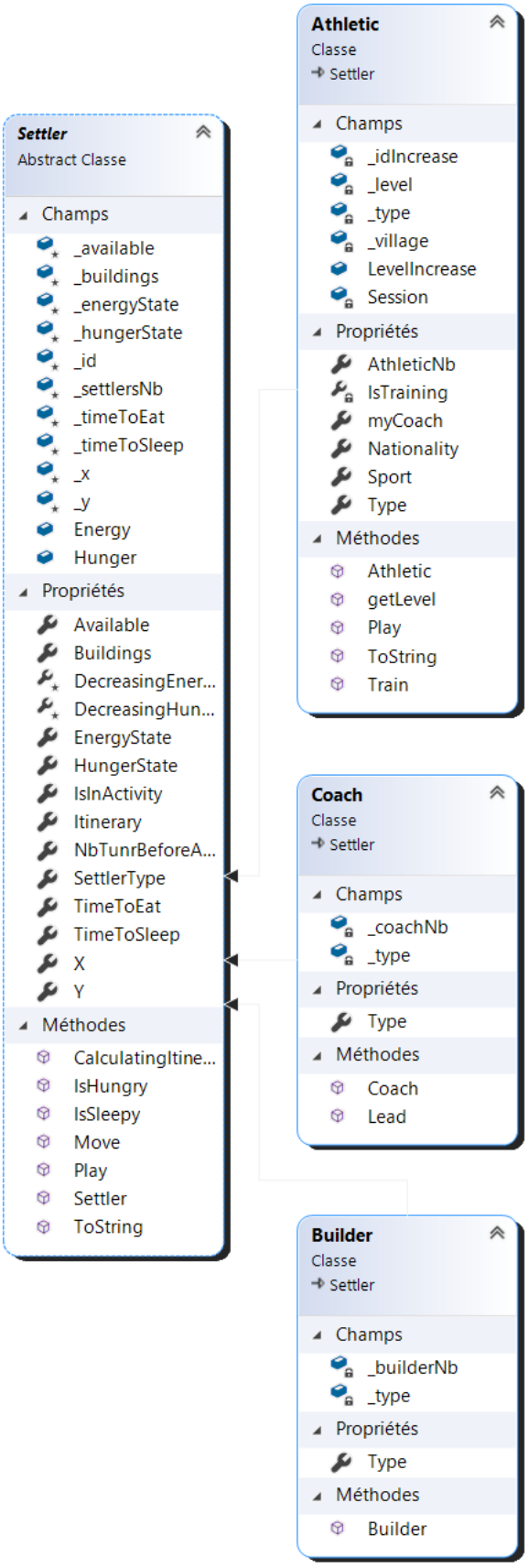

Diagramme de la classe Settler et de ces classes filles

Les bâtiments

Nous avons créé une classe bâtiment (Building) dont hérite trois classe pour nos trois types de bâtiments différents :

- Hotel
- Restaurant
- Infrastructure sportive (SportInfrastructure)

Chacun de ces bâtiments à des coordonnées (x et y) correspondant aux coordonnées de leur angle en haut à gauche. Chaque bâtiment à de plus un identifiant, un type, un nombre de tours de constructions (turnNb), un nombre de bâtiments, une liste de colons ainsi qu'une longueur et d'une largeur.

La longueur, la largeur, le type, le nombre de tours avant la construction et le nombre de builder dépendant du type de bâtiment.

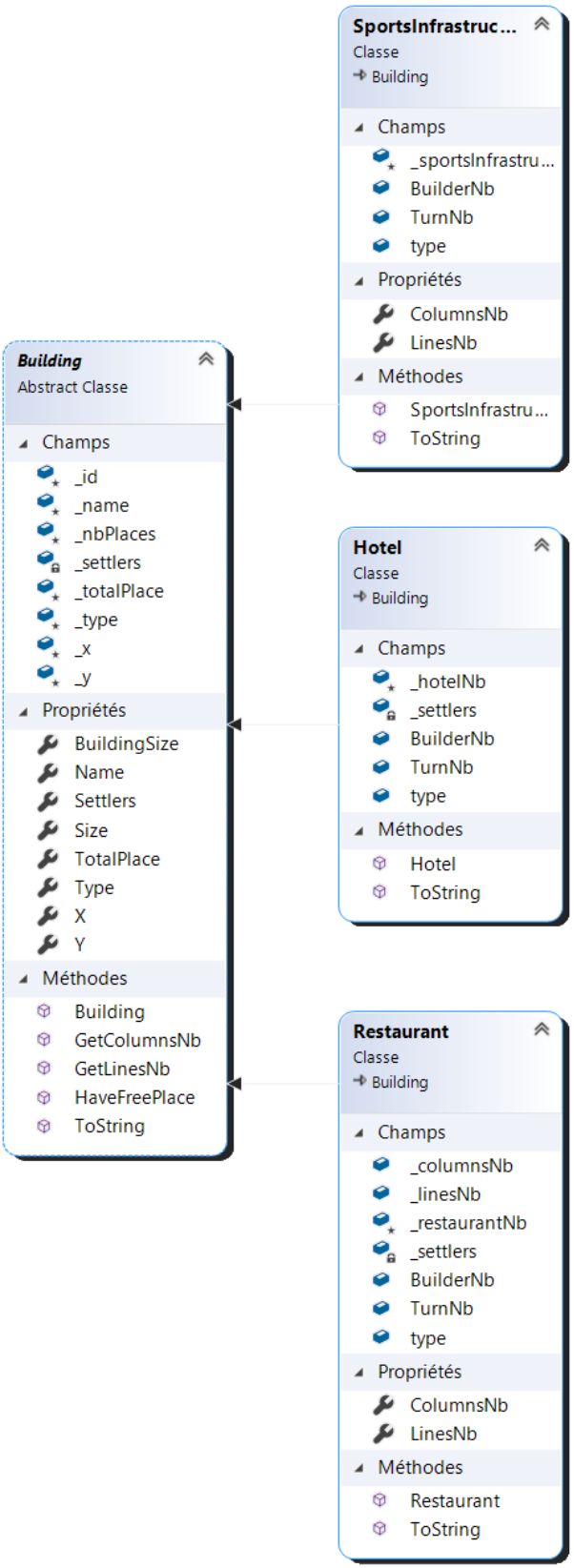

Diagramme de la classe Building et de ces classes filles

Justifications techniques

Dans cette partie nous allons expliquer quelques choix techniques que nous avons fait.

Les bâtiments en construction

Afin d'enregistrer les bâtiments en cours de construction nous avons décidé de créer une structure. En effet, les bâtiments en construction ont besoin d'informations supplémentaires que les bâtiments construits.
Ils ont besoin :

- D'un numéro indiquant le nombre de tours avant la construction
- D'une liste de bâtisseur

Une seconde option s'offrait à nous consistant à ajouter des informations dans la classe des bâtiments ainsi que d'ajouter un booléen pour savoir si le bâtiment est en construction. Cette seconde option nous semblait moins pertinente car les paramètres seraient alors inutile la plus grande partie du temps de jeu. De plus pour passer d'un bâtiment en construction à un bâtiment construit il aurait fallu à chaque tour parcourir tous nos bâtiments pour comparer leur tour de construction et le tour actuel du jeu. Avec la solution de la structure on en parcourt que les bâtiments en cours de construction et non tous les bâtiments du jeu.

La construction des bâtiments

Grâce à la structure expliquée précédemment nous avons pu enregistrer chaque demande du joueur. Avec une méthode PendingBuilding appelée à chaque tour nous avons comparé le tour auquel doit se construit un bâtiment et le tour actuel afin de constituer le bâtiment en question si ces deux chiffres correspondent. Grâce à cette méthode on ne stocke pas des bâtiments avant qu'ils ne soient réellement construits et on ne parcourt pas la liste de tous les bâtiments du village pour savoir lesquels construire.

La disponibilité des bâtisseurs

Nous avons décidé que le niveau de faim et d'énergie n'impactent pas les activités d'un colon. Cette décision implique qu'un colon qui à faim ou est fatigué alors qu'il est entrain de construire, de s'entraîner ou d'entraîner ne va pas arrêter son activité pour se restaurer ou se reposer.

Pour déterminer si un bâtisseur est disponible ou non, nous avons utilisé la propriété "Available". Cette propriété vérifie que le colon n'est pas en activité (construction,

entraînement) et qu'il n'a ni faim ni n'est fatigué. Elle renvoie "true" ou "false" en fonction du résultat de ces vérifications.

Les vérifications de chaîne d'entrée

Nous avons mis en place un fonction "VerifySyntaxe" afin de vérifier toutes les chaînes entrées par le joueur et vérifier qu'elles sont au bon format. Ainsi le programme ne s'arrête pas à la moindre erreur de saisie. De plus, l'explication de l'erreur s'affiche en rouge.

```
11 références
public int VerifySyntax(string phrase)
{
    Console.WriteLine(phrase);
    bool valid = int.TryParse(Console.ReadLine(), out int resultat);
    while (!valid)
    {
        Error("La syntaxe est incorrecte veuillez entrer un nombre");
        Console.WriteLine(phrase);
        valid = int.TryParse(Console.ReadLine(), out resultat);
    }
    return resultat;
}
```

Fonction permettant de vérifier que la chaine d'entrée est bien un nombre

```
Vous n'avez pas de sportif à entrainer
Entrez 0 pour passer au tour suivant sans effectuer aucune action
Entrez 10 pour avoir le détail des colons de votre colonie
Entrez 20 pour voir les regles du jeu
Entrez 1 pour créer un batiment
Entrez 2 pour recruter un colon
p
La syntaxe est incorrecte veuillez entrer un nombre
Entrez 0 pour passer au tour suivant sans effectuer aucune action
Entrez 10 pour avoir le détail des colons de votre colonie
Entrez 20 pour voir les regles du jeu
Entrez 1 pour créer un batiment
Entrez 2 pour recruter un colon
```

Exemple d'erreur

Les tests

Afin de tester notre code nous avons fait des tests fonctionnels. Après le développement de chaque fonctionnalité nous testions chacun des cas critiques et normaux. De plus nous testions les anciennes fonctionnalités pour vérifier que leur fonctionnement n'avait pas changé.

Gestion de projet

Organisation

Afin de s'organiser dans le projet nous avons utilisé l'outil github pour se partager le code.

Nous avons de plus utilisé l'outil trello pour se partager les tâches et se tenir au courant de ce que chacune faisait. Sur Trello nous avons créé 4 colonnes: une "Todo" contenant les tâches à faire, une "Doing" contenant les tâches en cours, une colonne "Bug" contenant les tâches normalement finies mais dans lesquelles nous avons trouvé des erreurs, et enfin une colonne "end" pour les tâches finies. Lorsque que l'une de nous trouvait une anomalie, elle passait la tâche dans la colonne "Bug" et mettait en description le bug. Ensuite la tâche devait repasser par la colonne Todo avant d'être fini.

Voici le lien du trello : https://trello.com/b/9N1sMEbv/olympique-village
Voici le lien github: https://github.com/alaudebert/colonie

Commentaire

Nous avons toutes deux participé à ce projet à la hauteur de nos compétences, nous pensons que le travail à été équitablement répartie. Et que l'équipe à eu un bon fonctionnement du début à la fin notamment dû à une bonne communication. Chaque membre du groupe tenait au courant l'autre de ces avancées et de ces difficultés pour ne pas rester trop longtemps bloqué sur un problème.

A vous de jouer

Nous vous proposons ici un scénario vous permettant de gagner une partie.

Ce scénario est sous forme de vidéo, il explique rapidement les règles et montre un scénario de victoire en expliquant les mécanismes.

https://drive.google.com/file/d/1pIP3OOWSIKmQ96xQ3OAhQI7LX6BtFvrX/view?usp=sharing

Bilan

Amélioration

Pour améliorer notre organisation nous aurions pu réfléchir à quelles tâches étaient indépendantes l'une de l'autre pour éviter les conflits de merge. Cependant nous pensons que dans ce projet il y avait beaucoup de tâches dépendantes entre elles, on aurait donc pu limiter les conflits mais il y en aurait eu dans tous les cas. De plus, le projet nous a permis de nous rendre compte qu'une bonne conception préalable aide énormément au développement d'un jeu. En effet le diagramme de classe nous à bien aidé à ne pas nous perdre dans nos idées.

Côté fonctionnalités le jeu reste à améliorer en effet on pourrait par exemple faire en sorte que ce soit les bâtisseurs les plus proche de l'emplacement du bâtiment qui aille le construire. La condition de victoire reste de plus à étoffer, actuellement la nationalité des joueurs n'est pas utilisée mais le but final serait qu'on puisse participer aux jeux olympiques à condition d'avoir un sportif entraîné au maximum dans chaque discipline et de chaque nationalité. Cependant cette version de jeu serait trop longue à tester pour un rendu tel que celui là.

Enfin, nous aurions pu tester notre jeu à l'aide de test unitaire afin d'automatiser nos tests.

Conclusion

Ce projet nous a permis de mettre en pratique de manière très concrète le principe de Programmation Orientée Objet. La programmation orientée objet nous a notamment permis de mieux organiser notre code, sans cette dernière nous aurions eu bien du mal à représenter l'ensemble des composantes de notre jeu de manière claire et structurée. Nous avons de plus appris à utiliser git et à coder en binôme.