TDLog : Jelly

BATY Léo, BRUNOD-INDRIGO Luca, CAMPO Yannis,
LAURET Jéremy, LO Emmanuel

15/02/2019

Table des matières

L'objectif initial de ce projet était de créer un jeu multi-joueur en ligne, adaptation d'un jeu de plateau créé l'année dernière dans le cadre du cours de développement durable.

1 Présentation du jeu

Dans ce jeu, les joueurs incarnent chacun un pays. Ils doivent gérer le développement économique, social, et environemental du pays durant deux ères, depuis la révolution industielle jusqu'à nos jours. Chaque ère se décompose en tours appelés générations, dans lesquels les joueurs jouent simultanément en construisant des bâtiments, recherchant des technologies et concluant des accords, puis dès que tout le monde a validé on passe à la génération suivante.

1.1 Déroulement d'une génération

Chaque génération se déroule en trois phases :

1. Phase de revenus
2. Phase principale
3. Phase évènements

1.1.1 Phase de revenus

Chaque joueur possède trois champs principaux :

— ses ressources stockables : UM (unité monétaire) et hydrocarbures

— sa production excédentaire : UM, hydrocarbures, nourriture, pollution, déchets, électricité, régénération de l'environnement. Par exemple, une production négative de nourriture veut dire que le joueur doit importer les ressources correspondantes afin de satisfaire le besoin de son pays.

— l'état de son développement : social, économique, et environnemental (entier entre 0 et 100)

Durant la phase de revenus, chaque joueur reçoit son revenu en UM, doit importer nourriture et électricité si ils sont en production négative, ou les exporter si il sont en production positive.
Les revenus d'hydrocarbures fonctionnent d'une manière un peu particulière. Au début de la partie, on constitue trois piles d'hydrocarbures représentant la réserve mondiale.

Lorsque l'on produit des hydrocarbures, on ajoute des hydrocarbures aux resources du joueur de la manière suivante :

— Tant qu'une pile n'est pas vide on prend les ressources toujours dans la même zone en commençant par la première.

— Si au début de la phase revenus il y avait des ressources dans la pile 1, on prend 3 hydrocarbures par point de production que l'on posséde et on les place dans notre réserve.
— Si au début de la phase revenus la pile 1 était vide et la pile 2 non vide, chaque point de production rapporte 2 hydrocarbures.
— Si au début de la phase revenus les pile 1 et 2 étaient vides, chaque point production rapporte 1 ressource.
— S'il n'y a plus de ressources dans la réserve, on n'en prend pas.

1.1.2 Phase principale

Durant la phase principale, les joueurs effectuent des actions de manière simulatnée. Voici la liste des actions possibles :

— Construire un bâtiment débloqué. Chaque bâtiment a un coût en UM, des modificateurs au niveau de la production du joueur et de son développement, et un éventuel effet (ponctuel ou permanent).
— Acheter une technologie débloquable. Chaque technologie débloque une ou plusieurs technologies, un ou plusieurs bâtiments, et possède un éventuel effet (ponctuel ou permanent).
— Conclure des accords d'importation ou d'exportation. Chaque accord est proposé ou non de manière aléatoire au joueur, et réduit le coût des importations ou permet des exportations d'une ou plusieurs ressources pendant un certain nombre de générations.

Chaque joueur peut effectuer autant d'actions qu'il veut, dans la limites des fonds disponibles. Cependant, il ne peut plus construire de bâtiments dès qu'il a construit une technologie, cela simulant que la recherche met du temps à devenir effective.

1.1.3 Phase évènements

A la fin de chaque génération, on tire alèatoirement un évènement correspondant à l'ère en cours, et on l'applique à tous les joueurs. Les évènements ont des effets très variés dépendants de l'état du développement des joueurs. Certains s'appliquent de la même manière à chaque joueur, et donc regardent la moyenne (ou le minimum/maximum) de chaque état, d'autres s'appliquent individuellement. La fin de chaque ère est déclenchée par des évènements particuliers appelés évènements finaux. Ces évènements ont des effets plus radicaux que les évènements classiques, et déclenchent le début de l'ère suivante. A la dernière ère, l'évènement final déclenche un dernier tour sans évènements avant la fin de la partie.

1.2 Décompte des points

A la fin de la partie on décompte les points pour chaque joueur. Chaque joueur classe les trois échelles de développement dans l'ordre décroissant de sa position sur ces dernières,

et calcule son score de la manière suivante :

1. Chaque niveau atteint sur l'échelle de développement la plus haute rapporte 1 point.
2. Chaque niveau atteint sur l'échelle intermédiaire rapporte 2 points.
3. Chaque niveau atteint sur l'échelle la plus basse rapporte 3 points.

Le joueur possédant le plus de points remporte la partie.

2 Objectifs initiaux

Voici la liste des objectifs principaux fixés au début du projet :

1. Coder le jeu ainsi que le plus de fonctionalités possibles parmi celles décrites dans la section précédente.
2. Y ajouter un système qui permet de créer des parties, d'en rejoindre, d'en lancer, et d'en sauvegarder.
3. Créer un site internet hébergeant le jeu

3 Organisation/architecture du code

Nous avons utilisé un backend en python avec le framework Django, et un frontend Javascript avec la librairie React.

3.1 Backend Django

Nous avons codé le backend en Python à l'aide du framework Django. Le backend est composé de quatre applications. L'application principale est l'application *game*, dans laquelle se trouve les modèles représentant les différents composants du jeu, les fonctions associées aux règles du jeu, ainsi l'API django REST framework qui permet le lien avec le frontend à l'aide de requêtes d'URL.

Voici le diagramme UML représentant les modèles de l'application *game* représentant une partie ainsi que leur organisation :

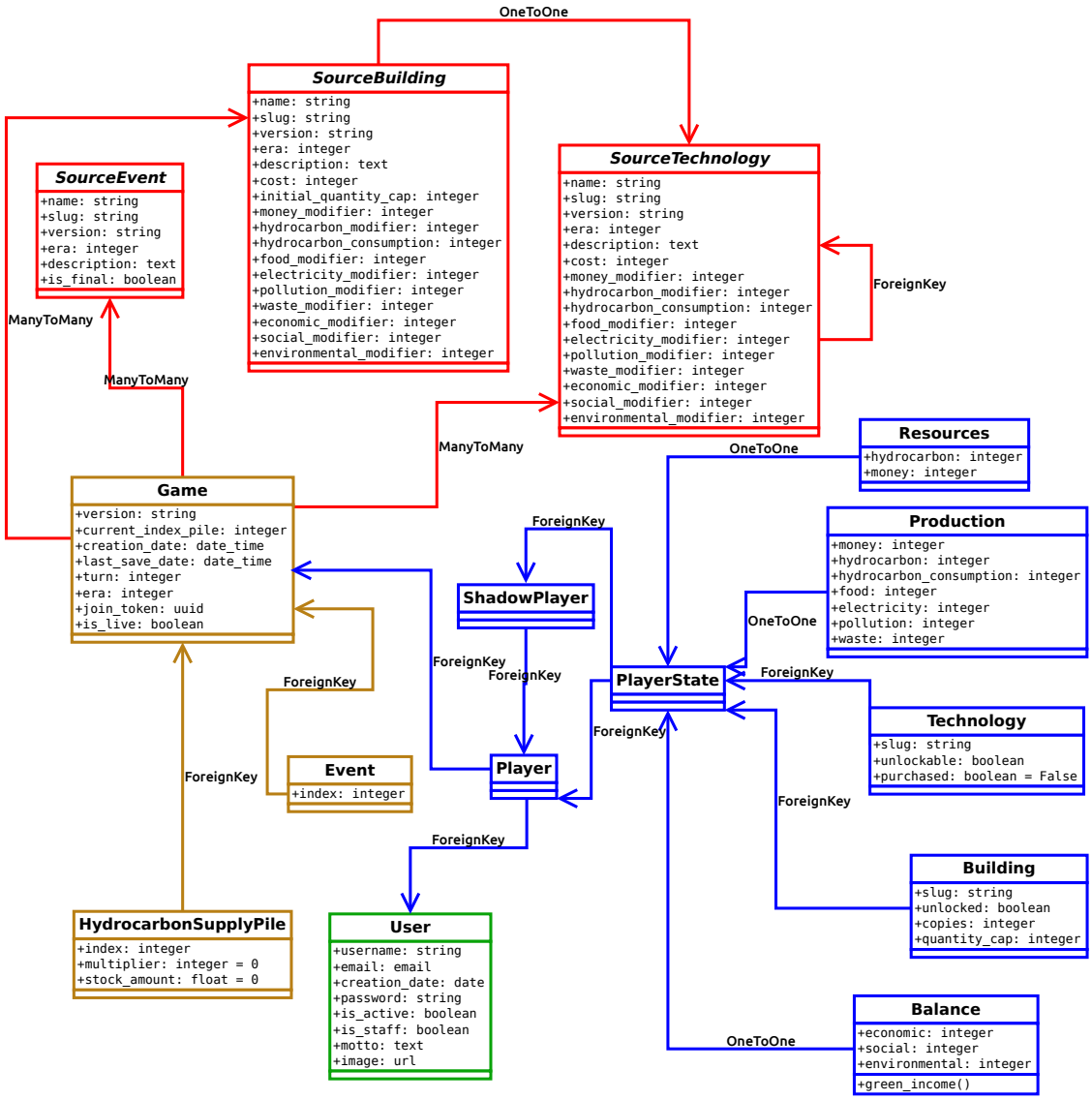

FIGURE 1 – Diagramme UML des modèles

Comme indiqué sur le diagramme, on a tout d'abord une classe centrale **Game**, à laquelle sont reliées les fixtures (écrites dans des fichiers .json puis chargées en tant qu'instances de modèles) représentées par les modèles **SourceBuilding** (bâtiments), **SourceTechnology** (technologies), et **SourceEvent** (évènements). Puis, à **Game** sont reliés avec une relation ForeignKey (many to one) les modèles **HydrocarbonSupplyPile** (piles d'hydrocarbures, qui sont ordonnées par leur index) et **Event** ("deck" d'évènemements pour cette partie, ordonnées par leur index, et liées en OneToOne aux SourceEvent). Le deuxième modèle central de l'architecture du backend est le modèle **Player** qui est relié à **Game** par une ForeignKey. Chaque **Player** possède un **PlayerState**, auquel sont reliés les modèles **Resources**, **Production**, **Balance**, **Technology**, et **Building**. De plus, chaque

le **Player** est lié à un **ShadowPlayer**, qui est lui même lié à un **PlayerState**. Cela permet d'implémenter la pile d'action : au début de la phase principale de chaque joueur, on copie le Player dans le ShadowPlayer, puis pendant toute la phase on effectue les actions sur le ShadowPlayer et on copie à nouveau le Player dans le ShadowPlayer à chaque annulation des actions. Enfin, à la fin du tour un met à jour le Player avec le ShadowPlayer.

3.2 Frontend React

4 Eléments réalisés

(il est usuel d'avoir un delta avec les objectifs initiaux)

5 Problèmes réalisés

(afin que nous capitalisions de l'expérience en vue des prochaines promos)