

Projet JAVA 2013

GREEN PLANET

BAKALOGLOU Lucas | BROCHARD Jeremy | LETESTU Lauren | SANDRAMOHAN Jean-Marc



Table des matières

INTRODUCTION	4
PRESENTATION DU PROJET	4
PRESENTATION DE L'EQUIPE	4
OBJECTIFS :	5
ORGANISATION ET REPARTITIONS DES TACHES	5
ANALYSES DES BESOINS ET PROPOSITION DE SOLUTION	6
CRITERE DE SATISFACTION DES CITOYENS	6
LE BUDGET	6
TEMPS DE REFLEXION (REPONSE AU SERVEUR)	6
RESPECT DU COTE ECOLOGISTE	6
DUREE DE VIE DES USINES ET DU PROBLEME DE STOCKAGE	7
DIAGRAMME DE CLASSES SUIVANT LE PATTERN « MODELE – VUE – CONTROLEUR	7
ANALYSE DES STRATEGIES	8
PREMIERE STRATEGIE : IA 1	8
DESCRIPTION	8
DIFFERENTS CHOIX ...	8
ORGANIGRAMME DECISIONNEL	10
DEUXIEME STRATEGIE : IA 2	11
DESCRIPTION	11
DIFFERENTS CHOIX ...	11
ORGANIGRAMME DECISIONNEL	11
CONCEPTION GRAPHIQUE	14
CONFIGURATION DU JEU ET CONNECTIONS A LA BASE DE DONNEES	14
LA FENETRE « NEW GAME » OU LA VITRINE DU JEU :	14
LE MENU BDD LOGIN OU COMMENT SAUVEGARDER SES SCORES :	15
PARAMETRER UNE NOUVELLE PARTIE VIA LA MENU GAME OPTIONS :	17
UN MOT SUR LA JMENUBAR	18
FENETRE DU JEU	19
HIGH SCORE	21
REGLE DU JEU	22
GESTION DE LA BASE DE DONNEE	22
PHASES DE TESTS	23
FREQUENCES	23
AFFICHAGES DES RESULTATS DANS LA CONSOLE	24
ENREGISTREMENT DES RESULTATS DANS UN FICHIER	24
CONCLUSION	26
BILAN GENERAL	26
BILAN INDIVIDUELS	26
LETESTU LAUREN (CHEF DE PROJET)	26
SANDRAMOHAN JEAN-MARC	26
BAKALOGLOU LUCAS	27
BROCHARD JEREMY	27
BIBLIOGRAPHIES	28
ANNEXES	29

Table des illustrations

Figure 1: Équipe de projet.....	4
Figure 2: Planning.....	5
Figure 3: Diagramme de classe	7
Figure 4: Organigramme décisionnel IA1.....	10
Figure 5: Organigramme décisionnel IA2.....	12
Figure 6: Fenêtre "new game"	14
Figure 7: Construction fenêtre "new game"	15
Figure 8: Fenêtre connexion BDD.....	15
Figure 9: Construction fenêtre BDD.....	16
Figure 10: JPanel "InPane"	16
Figure 11: Fenêtre "Game Option"	17
Figure 12: Construction fenêtre "Game option"	17
Figure 13: Panel information Game Option.....	18
Figure 14: JMenu Bar.....	18
Figure 15: Interfaces jeu	19
Figure 16: Maquette de la fenêtre du jeu.....	19
Figure 17: Maquette JTabbedPane : Panel événement.....	20
Figure 18: Maquette JTabbedPane: Panel statistiques	20
Figure 19: Maquette JTabbedPane : Panel graphique.....	21
Figure 20: Interface High score.....	21
Figure 21: Table GameData	22
Figure 22: Représentations des 10 meilleurs joueurs de la base de donnée	23
Figure 23: Test affichage console.....	24
Figure 24: Visualisation du fichier.....	24
Figure 25: Diagramme de classe package contrôleur	29
Figure 26: Diagramme de classe package vue	29
Figure 27: Diagramme de classe package modèle	30
 Listing 1: Insertion d'un joueur dans la base de donnée	22
Listing 2: Récupérer les 10 meilleurs joueurs.....	23

INTRODUCTION

Présentation du projet

Projet informatique du 2^{ème} semestre de l'année 2012 – 2013, « Green Planet » est un projet de développement en JAVA sur 6 semaines. L'objectif était de développer, par équipe de 4 personnes, un jeu ludique basé sur le thème de l'écologie.

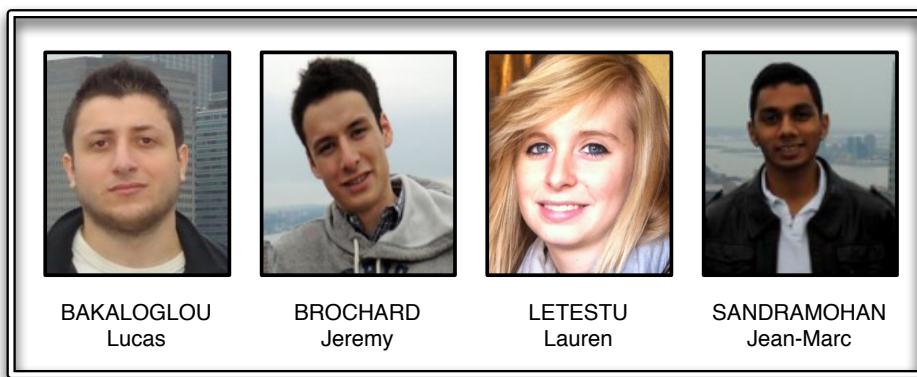
Le principe du jeu est de gérer au niveau d'un pays fictif la production d'énergie. Se jouant au tour par tour, le jeu met en compétition un ou plusieurs joueurs. Il est alors possible de jouer en « Offline » contre des bots ou en « Online » contre d'autres joueurs humains via le site internet du jeu. Le but du jeu est d'être le dernier survivant. Pour rester en vie, un joueur doit satisfaire en énergie 80% de la population de son pays, tout en gardant un niveau de cash positif. Pour cela, chaque joueur dispose d'une intelligence artificielle (IA) chargée de prendre des décisions à sa place en fonction d'un certains nombre de critères : prix de l'énergie, météo, etc. Pour produire de l'énergie, l'IA de chaque joueur peut construire plusieurs types de centrales ayant chacune des spécificités environnementales. Car oui le jeu se veut écologique ! Chaque joueur est classé selon son taux de pollution. Des bonus sont donc accordés au 1^{er} du classement, alors que le dernier reçoit une amende. D'autres évènements peuvent intervenir et changer le cours de la partie. Ainsi va la vie sur la « Green Planet » !

Pour ce projet, nous avons décidé de programmer deux IAs de manières différentes pour permettre aux joueurs de choisir entre deux stratégies différentes. Nous avons aussi décidé de réaliser une interface simple et stylisée, pour rendre notre jeu plus attractif et lui donner une identité visuelle unique.

Dans le présent rapport, nous présentons dans un premier temps notre équipe et son organisation puis notre interprétation du cahier des charges (CDC). Nous détaillons ensuite le fonctionnement de nos deux IA avant d'aborder l'interface graphique. Un bilan collectif et nos conclusions personnelles clôturent le rapport.

Présentation de l'équipe

Figure 1: Équipe de projet



Pour la réalisation de ce projet nous sommes composés d'une équipe de quatre étudiants. Actuellement en première année de cycle ingénieur à l'ECE-Paris. Nous avons nommé Letestu Lauren comme chef de projet pour ses qualités d'organisation et de rigueur. Étant tous très motivés et impliqués, nous avons su faire preuve d'efficacité pour la réalisation de ce jeu et chacun a su trouver sa place au sein de l'équipe.

Objectifs :

L'objectif principal du projet était de réaliser une application java capable de prendre des décisions toute seule afin de « satisfaire » au minimum à 80% la population grâce aux différents types d'usines en étant le plus « écologique ».

Nos principaux objectifs étaient :

- ✓ Créer deux intelligences artificielles afin de permettre à l'utilisateur de pouvoir choisir entre deux stratégies différentes.
- ✓ Créer une interface graphique attrayante et intuitive afin de faciliter les différentes manipulations pour l'utilisateur.
- ✓ Intégrer une base de données pour permettre à l'utilisateur de suivre ses scores et pour l'appliquer sur un projet concret.
- ✓ Réaliser des bons scores afin de valider nos stratégies.

Organisation et répartitions des tâches

Figure 2: Planning



Afin d'organiser au mieux toute la durée de notre projet, nous avons commencé par réaliser un planning prévisionnel décrivant les principales tâches :

- ✓ Étude du cahier des charges
- ✓ Mise en place de la base du projet
- ✓ Réalisation des stratégies (Intelligences artificielles)
- ✓ Réalisation de l'interface graphique

La répartition des différentes tâches c'est faite selon les compétences et les goûts de chaque membre de l'équipe. Sandramohan Jean-Marc et Bakaloglou Lucas se sont occupés de la réalisation des stratégies. Comprenant la mise en place d'un algorithme afin de gagner sans intervention de l'utilisateur au cours de la partie et la gestion et les prévisions des données de notre joueur. Nous avions deux idées d'intelligences artificielles, la première basée sur la méthode dite « simplex » et la deuxième sur le « système expert ». Étant tous les deux doués aussi bien en informatique qu'en mathématiques, notre choix nous a semblé. Ces deux stratégies sont détaillées dans la suite de notre étude. Brochard Jeremy et Letestu Lauren se sont occupés de la conception générale du projet en appliquant la méthode « Modèle-Vue-Controller ». De plus, Jeremy a assuré la conception graphique : organisation et mise en place des différentes fenêtres et la gestion des « listeners ». Enfin, Lauren avait en charge la gestion des données au tour actuel avec l'enregistrement de celles-ci dans un fichier pour nos différents tests, sauvegarde les joueurs gagnant dans un base de données et conception de la fenêtre du « jeu ».

Analyses des besoins et proposition de solution

Pour que le ministre de l'énergie et de l'environnement puisse correctement assurer la « survit » du pays Green Planet, il est primordial que la stratégie de l'État puisse valider les quatre critères suivantS :

- ✓ Respect du critère de satisfaction des citoyens
- ✓ Respect du budget
- ✓ Respect du délai d'attente pour une prise de décision
- ✓ Respect du côté écologiste
- ✓ Respect de la durée de vie des usines et du problème de stockage

Critère de satisfaction des citoyens

Selon le cahier des charges il est nécessaire que le nombre d'unités d'énergies produites sur le nombre d'habitants soit égal à 0.8. Cela implique donc que pour chaque tour, il n'est pas nécessaire de produire exactement la quantité d'énergie demandée par les citoyens. Néanmoins cela implique aussi que 80% de la population soient alimentées en énergie sans quoi le ministère sera remplacé.

Pour respecter ce critère il suffit tout simplement de connaitre avec précision la production de la ville à tout instant t et ainsi en fonction de cette production, le ministère devra soit acheter des usines soit acheter directement l'énergie manquante à la banque centrale de l'énergie.

Le budget

Comme dans tout organisme aussi bien politique que de crédit, il est très important d'apporter une attention particulière au **budget**. En effet de nombreuses sociétés doivent déposer bilan suite à une mauvaise manipulation ou à une mauvaise administration des comptes. C'est pourquoi le ministère va devoir d'une part répondre au 80% de satisfaction des citoyens tout en évitant de vider les caisses.

Afin de réaliser cela notre équipe a défini deux plans qui pourraient d'une part économiser des sous mais aussi apporter un bénéfice à l'État. Il n'est pas aisés de trouver des usines dont la somme des productions apporte exactement la quantité d'énergie nécessaire pour remplir le prorata de 80%, c'est pourquoi il serait intéressant de vendre le surplus d'énergie à la banque centrale de l'énergie et ainsi rentabiliser cette dépense. Par ailleurs, les usines ne sont pas le seul moyen d'avoir de l'énergie, en effet la banque propose un service « ventes d'énergie ». Ce service peut être intéressant à utiliser lorsque le prix de l'énergie est très faible car il offre parfois une quantité d'énergie achetée égale à celle produite par l'achat des usines mais pour un prix 1000 fois inférieur.

Temps de réflexion (réponse au serveur)

Selon une enquête portée par « The institute of satisfaction » il est très important que toutes les décisions prises par le ministère soient effectués avant une période t, au-delà de cette période les citoyens considèrent que l'État est incomptént et boycotte celle-ci.

C'est pourquoi afin de ne pas mettre en faute le ministère notre groupe va développer une nouvelle méthode de management qui va d'une part faire en sorte de réduire au MINIMUM de temps chaque opération et ainsi permettre de satisfaire les citoyens avant qu'ils ne se révoltent.

Respect du coté écologiste

Comme il a été signalé dans le cahier des charges, Green Planet est une planète **green**. Cela implique que toutes les futures décisions de l'État doivent respecter l'aspect écologique sous peine d'être sanctionnée par l'organisme au-dessus de l'État. Deux manifestations sont à prévoir, le premier apparaît sous la forme d'une prime de bonne écologiste et le second correspond plutôt à une taxe à payer pour ne pas avoir respecté l'aspect écolo.

Pour remédier à ce problème il est important de toujours favoriser les usines écologiques au profit des usines polluantes pour ainsi éviter la taxe.

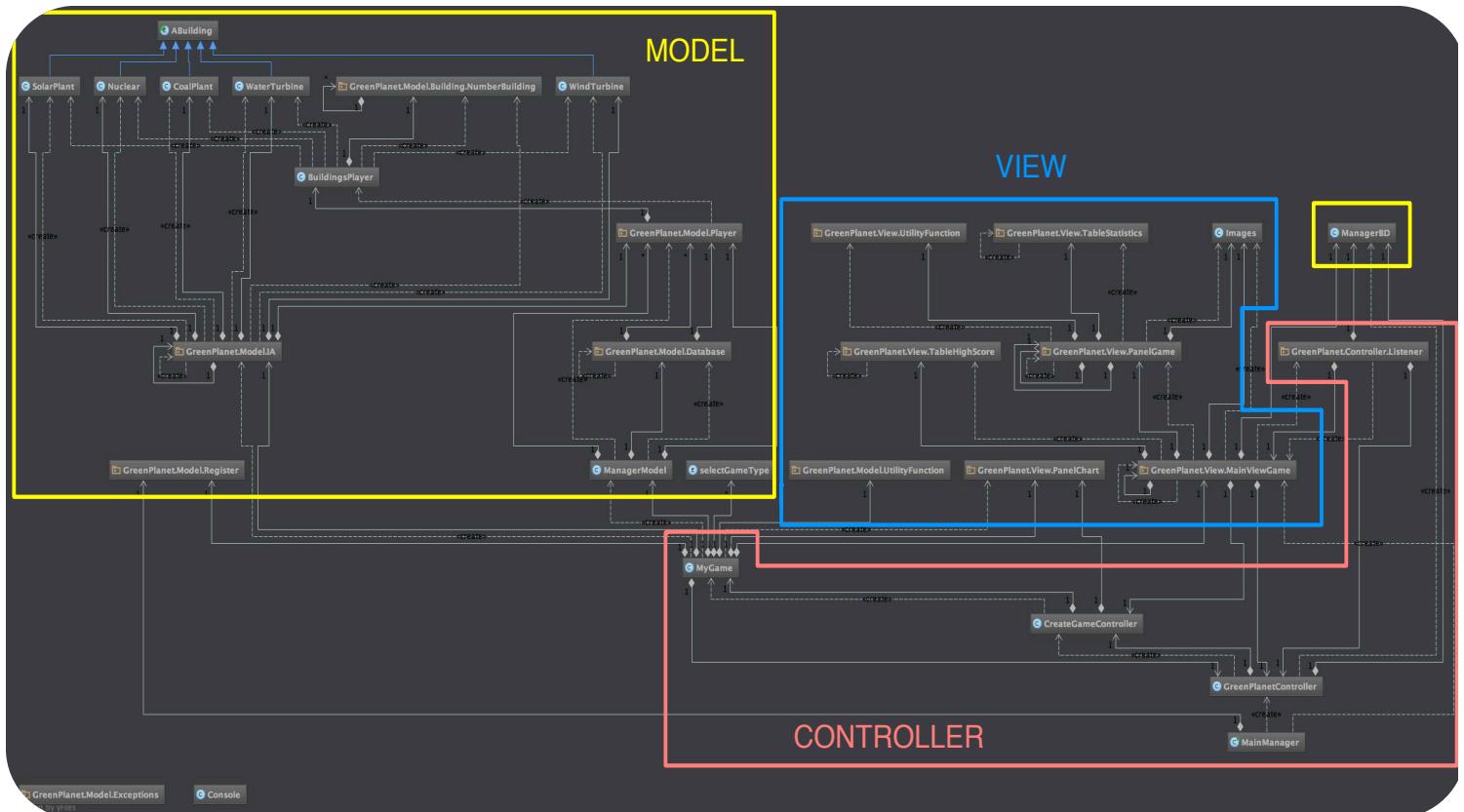
Durée de vie des usines et du problème de stockage

Les usines qui sont construites disposent d'une durée de vie de 10 ans et chacune d'elles ne peut stocker d'énergie. Cela implique que toute l'énergie qui n'a pas été utilisé par les citoyens va être perdue indubitablement.

C'est pourquoi notre équipe va faire en sorte de toujours vendre la quantité supplémentaire qui ne peut être stockée et ne jamais rien gaspiller. De plus, il faudra toujours prendre en compte la durée restante de vie d'une usine afin d'éviter d'avoir de mauvaise surprise.

Diagramme de classes suivant le pattern « Modèle – Vue – Contrôleur

Figure 3: Diagramme de classe



Analyse des stratégies

Première stratégie : IA 1

Description

La première intelligence artificielle est basée sur le principe du système expert. Son raisonnement prend en compte trois critères pour la prise de décision. D'une part le prix, ensuite le besoin et enfin le cash disponible.

Ces trois critères sont par la suite couplés à la notion de prévision qui permet ainsi d'imposer réellement les futures valeurs du besoin de l'énergie produite au tour suivant.

Afin de mieux comprendre le raisonnement de notre système expert, chaque étape de l'algorithme va être explicitée.

a) Phase d'initialisation

Dans un premier temps toutes les données **utiles** (prix, besoin et production) renvoyées par le serveur sont stocké dans des variables temporaires.

Après leurs stockages ces mêmes variables passent par une phase de test de corruption. En effet lorsque le serveur reçoit trop de requête simultanément, il commet des erreurs allant d'un besoin négatif à une somme d'argent disponible erronée. C'est pourquoi si ces valeurs sont erronées alors des valeurs par défaut sont attribuées pour ainsi ne pas perturber l'algorithme.

Puis à partir de ces valeurs, on définit ce que nous appellerons les variables de *respect des critères de victoire*. Ces variables sont calculées par rapport aux données reçues après le test de corruption. Il s'agit de trois variables qui représentent respectivement la quantité d'énergie possible à dépenser, la quantité minimale de la production afin d'obtenir un ratio de 100% et enfin la valeur du bonus Eco souhaité.

b) Phase de d'analyse de la situation

Après avoir effectué ces divers calculs, notre algorithme fait un choix très simple est-il plus avantageux d'acheter des usines ou de d'acheter les valeurs énergétiques à la banque ?

Pour répondre à cela cinq paramètres entre en jeu, le numéro du tour courant, la quantité d'argent disponible, le besoin des citoyens, la production actuelle et le prix de l'énergie.

Selon la combinaison, l'algorithme choisira soit d'acheter des usines soit de ne rien faire ou soit de tous acheter à la banque.

Différents choix ...

À la différence d'une IA classique, notre première IA élaboré à partir du principe du système expert n'émule pas uniquement un seul comportement mais plutôt 3 :

- Comportement de maximisation de l'écologie
- Comportement de maximisation de la pollution
- Comportement de maximisation dû cash

C'est trois comportements se chevauche durant le déroulement du jeu et permet à notre IA de réellement s'adapter au marché et donc de ne pas subir les fluctuations des prix et de la demande.

Comportement de maximisation de l'éologie

Ce comportement s'active lorsque le besoin est trop proche de 0, en effet l'une des conditions de perte est d'avoir un besoin négatif ou égal à 0. C'est pourquoi lorsque nous atteignons cette limite, l'IA essaie de faire son maximum pour faire augmenter son besoin en achetant notamment des usines non polluantes.

Comportement de maximisation de la pollution

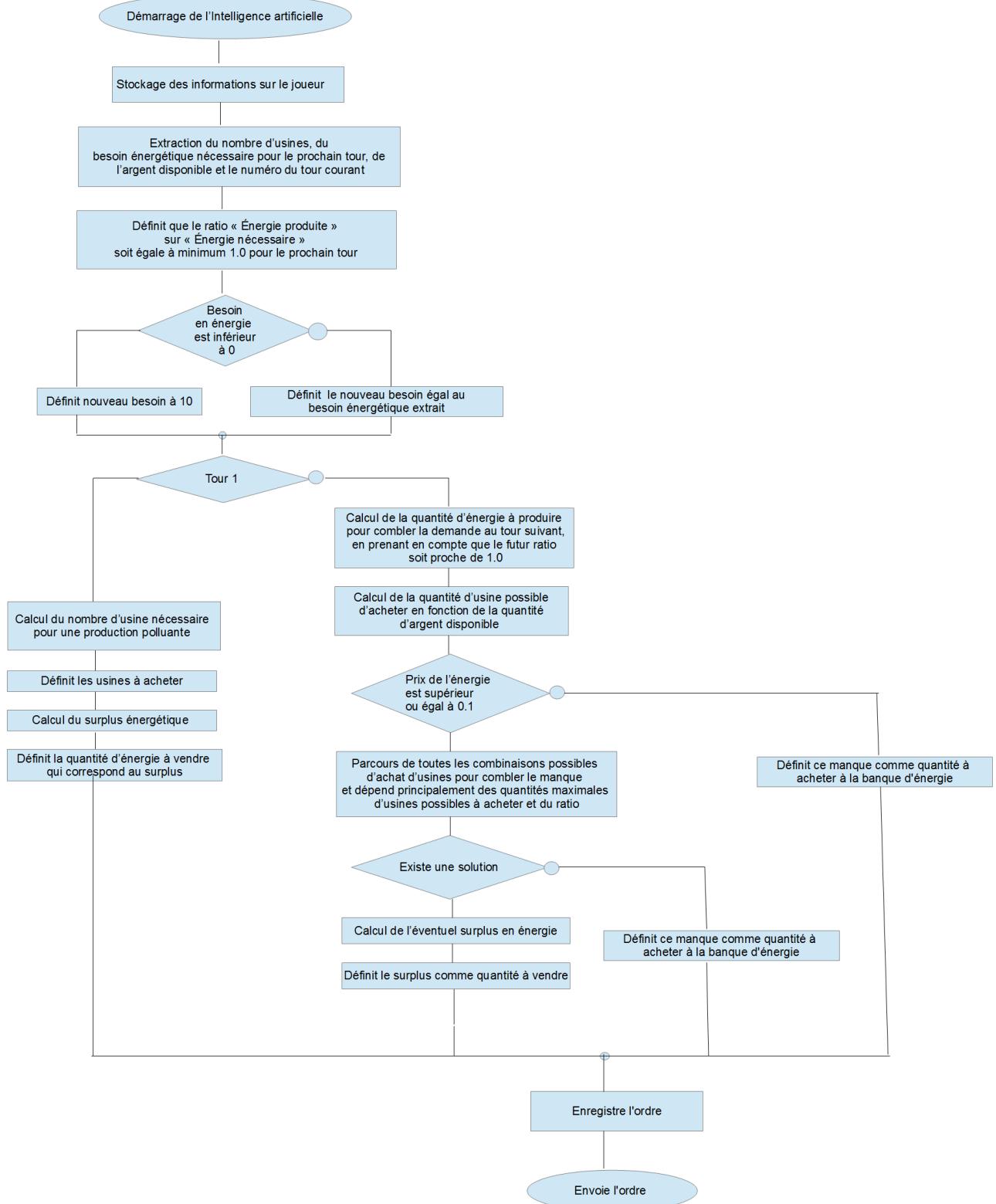
Lorsque l'on remarque que le besoin augmente de manière trop importante, l'IA fait en sorte d'abandonner le côté écologie et focalise sa production sur les usines non « green ». L'avantage de cette méthode est de pouvoir réduire certes la demande des citoyens mais a contrario, le danger est de se voir attribuer le malus du pire pollueur. Pour réellement permettre à la ville de ne pas trop produire dès le démarrage de la partie, ce comportement est favorisé dès le premier tour.

Comportement de maximisation dû cash

Afin d'obtenir tout une rentrée d'argent régulière, le comportement de maximisation de cash est appelé à chaque tour. Il permet notamment de vendre avec exactitude le surplus en énergie produite pas nos usines pour ainsi avoir un ratio d'exactement 100%

Organigramme décisionnel

Figure 4: Organigramme décisionnel IA1



Deuxième stratégie : IA 2

Description

Cette IA est basée sur la vente de l'énergie et la manipulation du « powerNeed » (qui correspond au besoin énergétique) grâce à des observations de son évolution en fonction des ordres, elle est aussi basée sur l'achat d'usine grâce à un algorithme mathématique dit d'optimisation connu sous le nom de Simplex, puis de l'achat d'énergie.

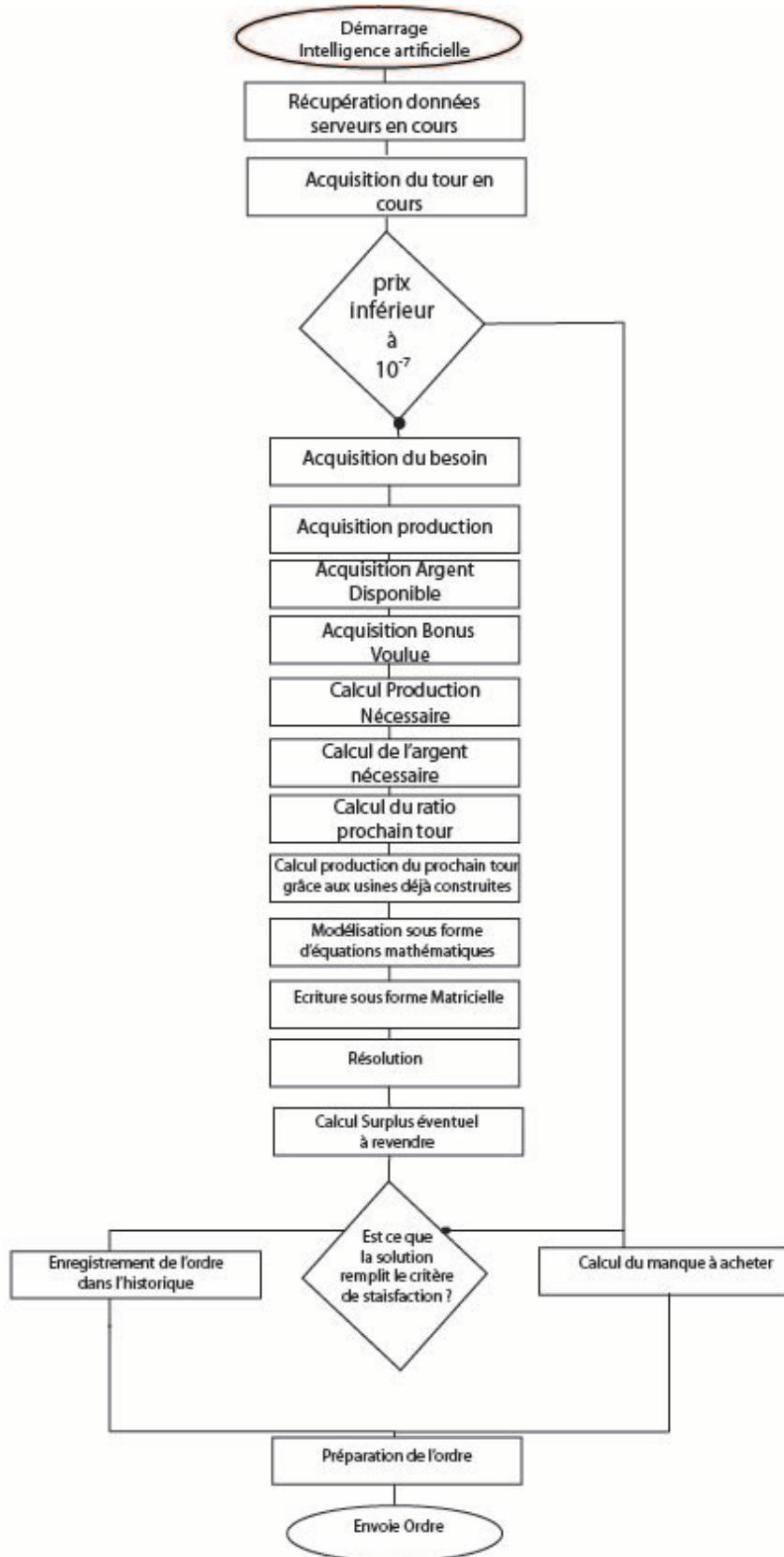
Différents choix ...

Comme pour la première Intelligence Artificielle, la stratégie n'est pas statique mais change en fonction de deux paramètres qui sont le numéro du tour et le prix de l'énergie.

- Le premier choix est d'avoir un malus écologique afin que le powerNeed diminue au maximum puis en même temps en achetant beaucoup d'usine avoir une surproduction pour revendre l'énergie ainsi il y a deux conséquence pour la suite du jeu :
 1. L'énergie sera à un prix faible donc facile à acheter pour un powerNeed élevé.
 2. On aura du Cash pour la suite et on pourra passer à un bonus écologique à la fin du jeu pour gagner en étant le meilleur joueur écologique.
- Le deuxième choix de stratégie s'effectue à partir du Tour 40, basé après avoir effectué plus d'une cinquantaine de partie en Online. En effet le prix étant extrêmement faible on peut ne faire qu'acheter avec très peu d'argent ce qui nous garantit un nombre énorme de tour ou on reste en vie.

Organigramme décisionnel

Figure 5: Organigramme décisionnel IA2



L'IA 2 est composé de 2 classes chaque classe correspondant à un moteur de décision bien particulier.

La Classe Simplex :

Cette classe permet d'utiliser l'algorithme d'optimisation à savoir le Simplex, qui permet de maximiser la production en nous donnant un couple de valeur correspondant au type d'usine à acheter et le nombre respectif pour chaque usine.

En effet cette optimisation s'effectue sous contraintes, c'est-à-dire qu'une fois la fonctionne économique écrite ici on a :

$$\begin{aligned}
 \text{Max}[Z] &= 1000x_1 + 100x_2 + 100x_3 + 300x_4 + 500x_5 \\
 x_1 &: \text{Nombre de centrale nucléaires à acheter} \\
 x_2 &: \text{Nombre d'éoliennes à acheter} \\
 x_3 &: \text{Nombre de centrale solaires à acheter} \\
 x_4 &: \text{Nombres de centrale hydroélectrique à acheter} \\
 x_5 &: \text{Nombre de centrales de charbon à acheter}
 \end{aligned}$$

On peut définir des contraintes, qui ne sont rien d'autre que des inégalités reliant ces relations, pour ma part j'ai retenu 3 contraintes.

- La première étant que la somme des usines achetées devait être inférieur à un certain cout fixé lors des essais et en fonction des tours.
- La deuxième étant que la somme des usines achetées coefficientées de leur bonus/malus écologique ne doit être supérieur à une certaine valeur, ainsi on peut régler le paramètre écologie, puis la dernière étant que la production que l'on aura grâce aux usines achetées, ajoutées à la production déjà obtenue grâce au tour précédent respecte la condition de satisfaction.

Afin de procéder au simplex on a quelques itérations à effectuer, les 4 premières méthodes sont les quatre étapes du simplex avec méthode des tableaux vus avec M. LAZRACK au premier semestre.

- La méthode « DoweStop » nous dit si on doit recommencer car une fois les quatre méthodes effectuées, on regarde les coefficients économiques (dernière ligne du tableau) puis s'ils sont négatifs on a atteint le maximum, sinon on recommence, c'est ce à quoi répond cette méthode.
- La méthode « initializing » permet de créer la table et de l'initialiser.
- Puis « mainloopSimplexTable » permet l'appel de ses différentes fonctions dans l'ordre nécessaire.
- Les fonctions « get Costs », « getBonus », et « getProductionAmount » permettent d'avoir les informations nécessaires pour chaque bâtiment. Par exemple « getCost » renvoie le cout, « getBonus » renvoie le bonus.
- La fonction « getProductionatTurnAsked » permet en regardant l'historique des achats effectués, de calculer la production pour le tour d'après afin de ne pas trop acheter, ainsi on tient compte des tours d'avant.
- La fonction « getPowerNeedAugmentation » permet de calculer l'augmentation du powerneed la formule se base sur des observations effectuées lors de parties, on a pu corrélérer ce que l'on pense être la formule d'augmentation, et on voit qu'elle dépend du nombre de joueurs en vie ce qui paraît logique.
- Ce qui nous amène à la dernière méthode « getAliveNumber » renvoie grâce à l'arraylist de player, le nombre des joueurs dont le statut est « Alive ».

Cet algorithme possède donc les avantages suivants, il nous permet par itérations successives d'étapes de calculs matriciels et algébrique de trouver le couple d'usine à acheter, couple qui nous garantira une production maximale pour les contraintes imposées ; de plus on peut régler beaucoup de paramètres tels que l'argent disponible, le bonus voulu. Cependant elle possède quelques défauts mais non dus à sa

conception mais dus au joueur donc au concepteur directement. Le joueur ne doit pas obligatoirement maximiser sa production à chaque tour mais juste respecter certaines contraintes, et le fait que les contraintes du simplex se basent sur des inégalités et non égalités impose qu'on ne peut avoir des coefficients fixes comme le ratio par exemple. La solution a donc été d'ajouter une deuxième classe IA qui elle va gérer l'achat d'énergie, la vente d'énergie et l'utilisation du simplex à bon escient.

La Classe IA :

La classe IA va elle appeler le Simplex, c'est-à-dire acheter les usines en fonction du tour dans lequel on se trouve. De plus, elle va acheter des énergies, quand le prix de celle-ci sera faible et vendre quand l'énergie sera au prix le plus fort. On ne peut faire une liste exhaustive de tous les choix faits car cela prendrait plus d'une dizaine de pages, on peut aller se référer au code directement. Un exemple serait que si on est au tour 40 et un prix très faible de l'ordre de 1 milliardième, l'IA n'achète plus du tout d'usine car seul l'achat d'énergie est rentable.

Conception graphique

L'interface de notre jeu a été conçue pour être simple et épurée. L'idée était de s'inspirer du thème arboré par le site du jeu pour réaliser une interface à la fois stylisée et simple d'utilisation. Nous avons donc repris la police du site internet et les principales couleurs.

Chaque partie de notre jeu arbore une image de fond bien spécifique pour séparer visuellement nos parties et leur donner une identité visuelle propre et facile à mémoriser. Pour chacune d'entre elles, nous avons volontairement choisi des images de paysages, pour coller au thème du jeu, qui se veut écologique. Notre interface se veut donc elle aussi « green » avec une dominante de couleur bleu, verte et beige.

À noter que la plupart des indications dans les fenêtres du jeu ont été écrites en anglais dans un souci de cohérence avec le titre du jeu « Green Planet » qui est lui aussi en anglais. Cela permet aussi de rendre notre programme plus universel.

Configuration du jeu et connections à la base de données

Pour créer une nouvelle partie, l'utilisateur doit passer par plusieurs étapes.

La fenêtre « New Game » ou la vitrine du jeu :

Figure 6: Fenêtre "new game"



« Splash Window » il s'agit de la première fenêtre qui s'affiche au lancement du jeu. On y découvre deux boutons permettant de lancer une nouvelle partie ou de quitter le jeu.

En lançant une nouvelle partie, on arrive ensuite à la fenêtre permettant de se connecter ou non à sa base de données (BDD).

A noter que la fenêtre possède une icône spécifique.

Focus sur la construction de la fenêtre :

Figure 7: Construction fenêtre "new game"



La JFrame principale se compose de deux JPanel ainsi que de deux JButton :

- ✓ Le Font JPanel qui englobe tout ce qu'il faut afficher dans le ContentPane de la JFrame. Sa disposition est en BoxLayout vertical et centré. Il possède une image de fond personnalisée. Les titres « Green Planet » et « When JAVA saves Green » sont des images qui sont ajoutées sur l'image de fond.
- ✓ Le « gameConf » JPanel qui servira à afficher les autres menus (pour la configuration de la BDD et du jeu) Sa disposition est de même en BoxLayout vertical et centré.
- ✓ Les 2 JButton « Play » et « Quit » avec une taille, une couleur et une police d'écriture personnalisée.

Le menu BDD LOGIN ou comment sauvegarder ses scores :

Figure 8: Fenêtre connexion BDD



Le menu de configuration de la base de données permet à l'utilisateur de se connecter à celle-ci. Cela permet au programme d'enregistrer les meilleurs scores de chaque partie pour que l'utilisateur puisse ensuite y accéder ultérieurement via la fenêtre « High Scores ».

Pour que cela fonctionne, il faut néanmoins que la BDD soit préalablement correctement configurée.

Pour activer la BDD, il est nécessaire de rentrer son URL (champ pré rempli), son nom d'utilisateur ainsi que son mot de passe. Il suffit ensuite de cliquer sur le bouton « Submit ! » pour passer à l'étape suivante : la configuration d'une nouvelle partie. Si les informations d'authentifications sont erronées, un message d'erreur apparaîtra, et il faudra resaisir les informations.

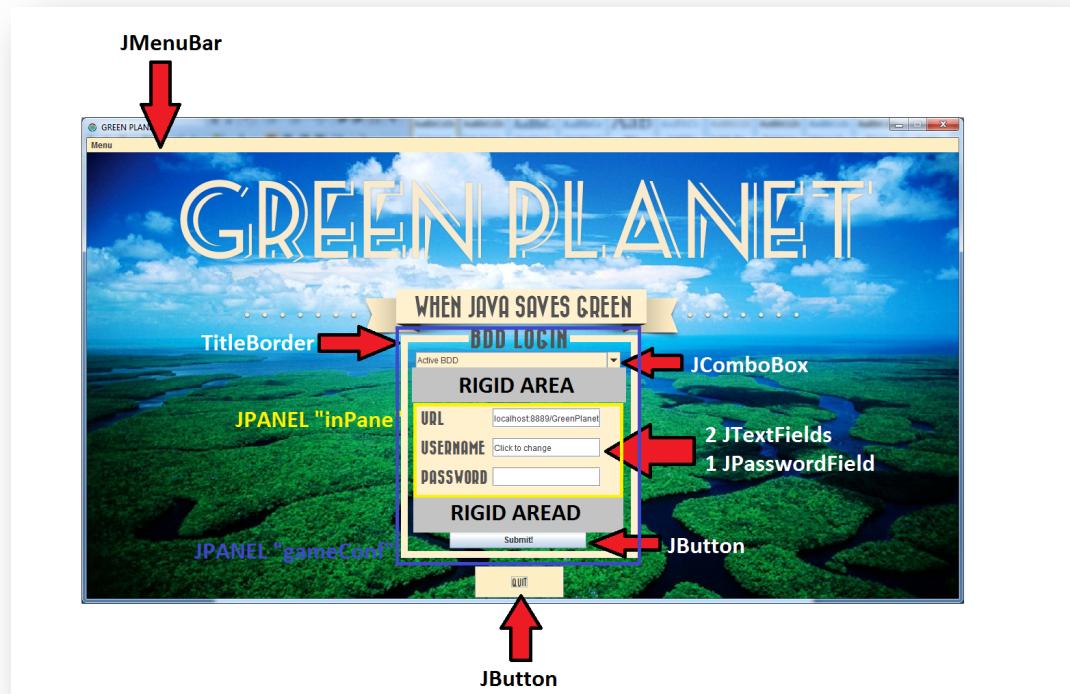
Pour passer à l'étape suivante sans se connecter à la BDD, il faut en sélectionner « Skip BDD » dans la JComboBox de sélection. Dans ce cas, tous les champs seront alors grisés ; il sera impossible de les modifier. Là encore, il suffit d'appuyer sur le bouton pour passer à la suite.

Focus sur la construction de la fenêtre :

Le menu « BDD Login » se place dans le JPanel « gameConf » déjà présent dans la fenêtre. Le bouton « Play » disparaît et une JMenuBar s'ajoute en haut de la fenêtre.

Le JPanel du menu se compose alors comme suit :

Figure 9: Construction fenêtre BDD



Le JPanel « inPane » est un panel en BoxLayout vertical, intermédiaire, se composant de trois autres panels :

Figure 10: JPanel "InPane"



Ces trois JPanel sont composé de JLabels puis de JTextFields (excepté le 3^{ème} vu qu'il a un JPasswordField) en FlowLayout. La taille des labels et des textfields est égale pour que l'alignement soit correct.

Paramétrer une nouvelle partie via la menu GAME OPTIONS :

Figure 11: Fenêtre "Game Option"



Le menu Game options permet à l'utilisateur de configurer une nouvelle partie avant de la lancer. Il peut indiquer son nom (ou pseudo) qui sera utilisé pour la partie ainsi que l'IA qu'il souhaite utiliser pour jouer la partie.

En fonction du type de la partie (Online ou Offline), il est possible de modifier plusieurs paramètres supplémentaires. Si la partie est Offline, il est alors obligatoire d'indiquer le nombre de bots. À l'inverse, si elle est Online, il est nécessaire de remplir le numéro de la partie indiqué sur le serveur internet du jeu. Pour aider l'utilisateur à remplir ces informations des informations lui sont indiquées en pointeur le curseur de la souris sur les différents champs à remplir.

Pour débuter la partie, il suffit ensuite, de cliquer sur le bouton « Launch the game ! ». Des messages d'erreurs sont susceptibles d'apparaître si une ou plusieurs des infos saisies par l'utilisateur sont erronées. Le cas échéant, une nouvelle partie se lance.

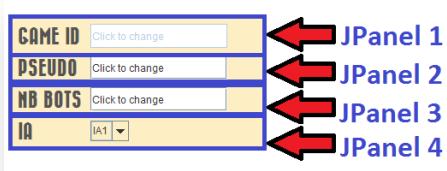
À noter la présence d'un bouton « Go to the website » permettant d'accéder directement au site internet du jeu.

Focus sur la construction de la fenêtre :

Le menu « Game Options » est construit de la même manière que le menu précédent à la différence près qu'il comporte plus de composants.

Figure 12: Construction fenêtre "Game option"

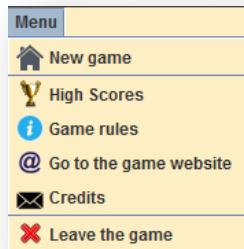


Figure 13: Panel information Game Option


Un mot sur la JMenuBar

Présente dans toutes les fenêtres du jeu (hormis la Splash window), elle permet d'effectuer plusieurs actions à n'importe quel moment :

- « New Game » renvoie à la page d'accueil et permet de créer une nouvelle partie.
- « High Scores » permet d'accéder à la fenêtre des meilleurs scores.
- « Game Rules » permet d'accéder à la fenêtre expliquant les règles du jeu.
- « Go to the game website » ouvre la page du site internet du jeu dans le navigateur par défaut.
- « Credits » permet d'accéder à la fenêtre des crédits du jeu.
- « Leave the game » quitter le jeu.

Figure 14: JMenuBar


Fenêtre du jeu

Pour l'interface principale du jeu, nous avons décidé de la réaliser de manière claire et attrayante pour ces utilisateurs. De plus, nous voulions une fenêtre intuitive pour qu'elle soit facile d'utilisation, même pour une personne n'ayant pas de compétences particulières en informatique.

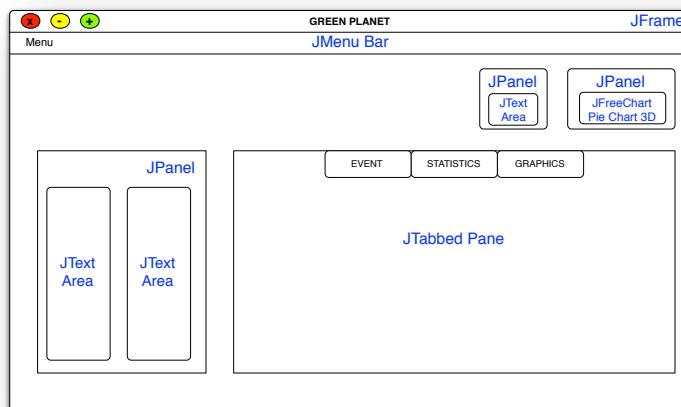
Figure 15: Interfaces jeu



Cette image représente la fenêtre de jeu, elle est composée de:

- JPanel Player : affichage des données propres à notre joueurs
- JTabbedPane : information principales du jeu
- Un panel affichant le tour actuelle et les prévisions météos

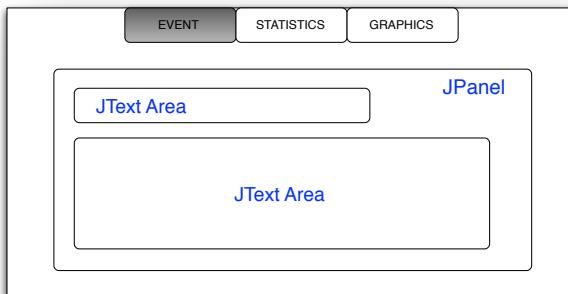
Figure 16: Maquette de la fenêtre du jeu



La maquette de l'interface « jeu » est organisée en deux grandes parties :

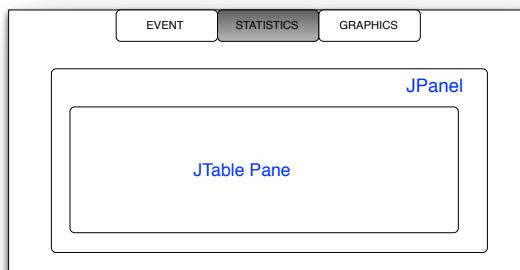
- le haut de page qui contient les informations du jeu : tour actuel mis à jour dans une JText Area à chaque tour et les prévisions météo afficher avec un graphique du type « Pie chart 3D » provenant de l'API Jfreechart.
- Et d'une seconde partie indiquant les informations de tous les joueurs présents lors d'une partie.
 - Un panel « player » qui composé de deux JText Area : l'une pour les « titres » des différentes catégories et l'autre pour afficher les données correspondantes actualisées à chaque tour.
 - Une « JTabbed Pane » qui est composée de trois panels différents, qui sont actualisés à chaque tour du jeu. Voir ci-dessous la description de chacun d'eux.

Figure 17: Maquette JTabbed Pane : Panel événement



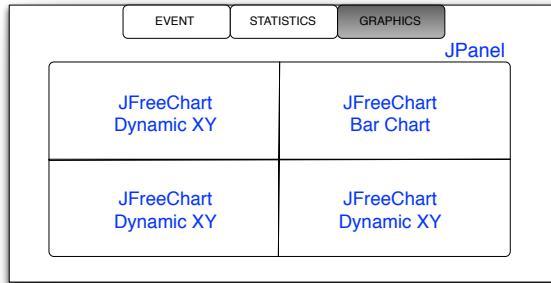
Maquette représentant le panel où sont affichés les différents événements au cours d'une partie : les joueurs pollueurs et écolo, les évènements liés au temps, par exemple « une explosion solaire » et en fin de partie, on y affiche le joueur gagnant. La première JText Area contient « le titre » du panel et la seconde contient les informations.

Figure 18: Maquette JTabbed Pane: Panel statistiques



Pour les statistiques, on a décidé d'afficher toutes les informations des différents joueurs sous forme de tableau, afin de permettre à l'utilisateur d'avoir accès à son classement et comparer sa stratégie à celle des autres joueurs à chaque tour.

Figure 19: Maquette JTabbed Pane : Panel graphique



Enfin, le panel « graphique » est composé de quatre « panel chart » pour permettre à l'utilisateur de suivre facilement l'évolution de son joueur. Nous avons choisi de regrouper les données suivantes : (de gauche à droite)

- Le premier graphique est du type « Dynamique XY ». C'est-à-dire que l'on affiche une courbe en fonction du temps et de la production d'énergie.
- Un deuxième panel sous forme de « Barre 2D » pour afficher les différentes usines que le joueur possède.
- Un troisième graphique correspondant à la satisfaction de la population. (Dynamic XY)
- Un quatrième qui affiche l'argent dont dispose le joueur (Dynamic XY).

High score

Nous avons décidé d'enregistrer à chaque partie le joueur « gagnant » afin de permettre à l'utilisateur de se positionner par rapport aux anciennes parties. Nous utilisons une base de données pour enregistrer le meilleur joueur.

Figure 20: Interface High score



Cette fenêtre comporte un Jtable Pane qui affiche les 10 meilleurs joueurs classés selon leur rang.

Règle du jeu

La fenêtre consacrée aux règles du jeu se veut simple et suffisamment complète pour comprendre le jeu dans son ensemble. Elle reprend et synthétise les principales mécaniques de gameplay.



Gestion de la base de donnée

Figure 21: Table GameData

GameData
Gameld : Int(11)
IdPlayer : Int(11)
NamePlayer : Varchar(25)
Cash : Float
GreenRanl : Int(11)
PowerNeed : int(11)
NBTurn : int(11)

Création d'une base de données « GreenPlanet » contenant la table « GameData ». Lorsque l'utilisateur lance une nouvelle partie, il décide oui ou non de se connecter à la base de données. Si l'utilisateur choisit de se connecter, à chaque fin de partie une requête est envoyée à celle-ci afin d'enregistrer le joueur gagnant. Enfin, lorsque l'utilisateur souhaite accéder à la liste des meilleurs joueurs une nouvelle requête est envoyé afin d'obtenir cette liste.

Requête pour insérer un joueur :

Listing 1: Insertion d'un joueur dans la base de donnée

```
PreparedStatement stmtPrepared = conn.prepareStatement("INSERT INTO GameData (IdPlayer, NamePlayer, Cash, GreenRank, PowerNeed, NBturn) " +
VALUES (?, ?, ?, ?, ?, ?);");

stmtPrepared.setInt(1, idPlayer);
stmtPrepared.setString(2, namePlayer);
stmtPrepared.setFloat(3, cash);
stmtPrepared.setInt(4, gRK);
stmtPrepared.setInt(5, powerNeed);
stmtPrepared.setInt(6, nbturn);

stmtPrepared.executeUpdate();
```

On utilise un « Statement Prepared » avec six champs, afin de stocker les différentes données du joueur reçues en paramètre de la fonction.

Requête pour récupérer les dix meilleurs joueurs :

Listing 2: Récupérer les 10 meilleurs joueurs

```
rset = stmt.executeQuery("select * from GameData order by NBTurn desc, Cash desc limit 10");
rsetMeta = rset.getMetaData();
```

Pour sélectionner les meilleurs joueurs on commence par classer les données par ordre décroissant de nombre de tours réalisés, puis par l'argent gagné et on limite les résultats à dix joueurs.

Figure 22: Représentations des 10 meilleurs joueurs de la base de donnée

	← ↑ →	Gamed	IdPlayer	NamePlayer	Cash	GreenRank	PowerNeed	NBTurn
<input type="checkbox"/>		26	1	test	6957.05	1	104971	33
<input type="checkbox"/>		36	1	ades	6148.8	1	77077	33
<input type="checkbox"/>		35	1	ades	5176.14	1	129499	33
<input type="checkbox"/>		38	1	ades	2141.54	1	82938	33
<input type="checkbox"/>		31	1	d	656.634	1	38725	33
<input type="checkbox"/>		17	5	Corkiadd	39.0566	1	29282	31
<input type="checkbox"/>		29	1	jjjj	1127.12	1	33447	30
<input type="checkbox"/>		21	3	Ryzeadd	322.473	1	18987	30
<input type="checkbox"/>		28	44951	GreenP	36005.5	1	3025	29
<input type="checkbox"/>		27	44951	GreenP	36005.5	1	3025	29

Ci-dessus, un exemple des joueurs présents dans notre table « GameData ». Ici, on affiche les dix meilleurs joueurs selon les critères de notre requête précédente.

Phases de tests

Fréquences

On peut décomposer les tests en trois périodes distinctes. Durant la première étape, il s'agissait de déterminer l'algorithme qui pourrait réellement faire tomber le marché de l'énergie. C'est pourquoi durant les deux premières semaines, Lucas et Jean-Marc ont chacun essayé d'élaborer des IA différentes qui auraient une chance de vaincre la version offline du jeu. Durant cette période, 5 IA ont été conçus, et des chaque IA a été testé contre différents nombres de bots allants de 3 à 25 bots.

Puis on entre dans phase de test sur la version online : lorsque les deux IA les plus prometteuses ont été sélectionnées, la seconde période de test a démarré. Celle-ci était notamment contre la version du jeu online mais uniquement contre des bots. Les tests étaient quotidiens, chaque jour, nous avons essayé de battre le record de tours en vie. Aussi bien contre un bot que 25 bots.

Lorsque nous avons réussi à attendre une moyenne de 150 tours contre les IA des serveurs, nous avons effectué les dernières séries de tests d'IA contre IA. Cela nous a occupés toute la dernière semaine mais nous a notamment permis de simuler un comportement en réelle compétition et ainsi pouvoir adapter notre IA en fonction des stratégies des autres IA.

Affichages des résultats dans la console

Figure 23: Test affichage console

Screen shot console

Play turn #2 Tour actuel

ID	Login	State	Cash	GRK	Poll	powerA	powerN	Ratio	Nuclear	Solar	Water	Wind	Coal	Production
59474	Bot Ashe	ALIVE	9800.0	1	200	150	100	150.0	0	0	0	0	1	500
59473	Bot Jayce	ALIVE	9800.0	1	200	150	100	150.0	0	0	0	0	1	500
59476	colo	ALIVE	6250.0	3	5000	100	100	100.0	0	0	0	0	25	12500

```

Cash 6250.0
Data given by the server
Previous PowerAvailable : 100
New PowerNeed : 100.0
Our operation :
Compute production : 12500.0
Additional Production produced by our order : 0
How much we sold or we buy : -12400.0
The theorical value of the next ratio : 1.0
The Price : 0.25
The building that will be buy for the next turn []
Play turn #3
Price0.125

```

Information du tour actuel pour chaque joueurs

Information du tour suivant pour notre joueur

Afin de pouvoir tester, améliorer et valider nos stratégies nous avons décidé de visualiser les informations via la console. Comme l'image l'indique, nous affichons le numéro du tour correspondant, les données principales de tous les joueurs pour le tour actuel et les prévisions calculées pour le tour suivant pour notre joueur.

Enregistrement des résultats dans un fichier

Nous avons décidé d'enregistrer les parties dans un fichier texte afin de pouvoir étudier en détail l'évolution de notre IA et la stratégie des autres joueurs. Afin de ne pas encombrer notre fichier avec de nombreuses parties, nous avons décidé d'utiliser une fonction qui permet de créer et supprimer un dossier dans le projet : utilisation de la fonction « `makedirs()` » et « `delete()` ». Ce dossier contient nos fichiers de sauvegarde.

Figure 24: Visualisation du fichier

```

Turn : 15
ID      Login State   Cash  GRK    Poll   powerA   powerN   Ratio  Nuclear Solar Water Wind Coal  Production
1       d     ALIVE  21221.854 2      0      1031    1146     0.9   0      0      0      0      0      0      0
2       Dr. Mundoadd ALIVE  975.1675 3      800    6890    5392     1.28  3      5      1      4      6      0      6555
3       Ireliaadd ALIVE  7784.369  1     -800    3234    2133     1.52  3      3      3      2      0      0      3862
*****
Turn : 16
ID      Login State   Cash  GRK    Poll   powerA   powerN   Ratio  Nuclear Solar Water Wind Coal  Production
1       d     ALIVE  21281.78  2      0      1340    1489     0.9   0      0      0      0      0      0      0
2       Dr. Mundoadd ALIVE  1725.1675 3      800    6442    6470     1.0   3      4      1      4      6      0      6768
3       Ireliaadd ALIVE  4641.2603 1     -400    5673    2986     1.9   5      3      3      2      1      0      6652
*****
Turn : 17
ID      Login State   Cash  GRK    Poll   powerA   powerN   Ratio  Nuclear Solar Water Wind Coal  Production
1       d     ALIVE  21169.322 2      0      1741    1935     0.9   0      0      0      0      0      0      0
2       Dr. Mundoadd ALIVE  825.1675 3      200    6590    7764     0.85  2      5      3      5      7      0      6872
3       Ireliaadd ALIVE  2742.4907 1     -100    7368    4180     1.76  5      4      3      3      3      0      7679
*****
Turn : 18
ID      Login State   Cash  GRK    Poll   powerA   powerN   Ratio  Nuclear Solar Water Wind Coal  Production
1       d     ALIVE  21957.58  3      0      2263    2515     0.9   0      0      0      0      0      0      0
2       Dr. Mundoadd ALIVE  325.16748 2     -100    6815    9316     0.73  2      5      4      5      7      0      7167
3       Ireliaadd DEAD   -507.50928 1     -800    8965    5852     1.53  6      5      5      5      3      0      9425
*****
```

Ci-dessus une image représentant quelques lignes de notre fichier. Pour la réalisation de celui, nous enregistrons les informations dans une « `HashMap` » ayant pour clé le numéro du tour et pour valeur une deuxième « `HashMap` » contenant les informations de chaque joueurs. Ainsi, on a accès à toutes les

informations pour la partie entière. Enfin les données des joueurs sont enregistrés dans une classe PlayerData où nous avons réécrit (« overriding ») la méthode « `toString()` » afin d'afficher les donnée sous forme de « colonnes ».

Conclusion

Bilan général

Ce projet a été l'occasion de pouvoir mettre en application les bases de programmations Java vu en cours. Mais aussi par la même occasion de toucher à la librairie graphique SWING et ainsi développer des applications aussi bien soignées esthétiquement qu'algorithmitquement.

De plus les différents problèmes du site avec notamment de nombreuses « Exceptions » renvoyées pour n'importe quelle raison, nous a contraint à effectuer des maintenances régulières sur nos versions du jeu pour afin de pallier ce genre de problème.

Cependant ce projet a aussi été l'occasion de pouvoir faire un projet avec un groupe motivé et bucheur. Malgré les différents que certains membres du groupe ont rencontrés entre eux, le professionnalisme de chacun a été grandement apprécié par toute l'équipe et nous vraiment permis de nous mettre dans un cadre de travail professionnel qui notre futurs carrières d'ingénieurs nous réservent.

Bilan individuels

Letestu Lauren (Chef de projet)

Les objectifs principaux étaient de créer une application java capable de prendre des décisions afin d'approvisionner nos « villes » en énergie et d'avoir une interface attrayante pour son utilisateur. J'ai pris plaisir à travailler sur ce projet car nous nous étions fixé des objectifs précis afin de répondre à toutes les attentes du cahier des charges. Aussi, nous avons su dès le début nous organiser et définir les tâches de chacun, ce qui nous a permis d'ajouter des options à notre jeu en plus de celles attendues.

Je suis satisfaite de notre réalisation finale car nous avons réalisé une application fonctionnelle qui en plus est dotée de stratégies performantes nous permettant de dépasser 200 tours ! De plus, ce projet m'a permis d'améliorer mes compétences en JAVA. Pour ma part, c'est un langage informatique très documenté qui offre de nombreuses API et une exécution multiplateforme ce qui rend son application intéressante et pratique (sachant que nous avons travaillé à la fois sous Mac et Windows).

Avec plus de temps, j'aurais aimé améliorer la base de données afin de créer différentes relations pour stocker toutes les informations de chaque partie. En effet, même si nous avons intégré cette option à notre projet, son utilisation reste très basique. De plus, un point faible important de notre organisation aura été la gestion des mises à jour des modifications de chacun. En effet, nous avons passé « trop » de temps à assembler nos parties respectives, quasiment tous les jours sur la dernière semaine. Je pense qu'il aurait été préférable d'utiliser une autre méthode (Github ?) afin de ne pas prendre du retard à cause des nombreuses versions différentes.

Enfin, je suis ravi d'avoir travaillé au sein de cette équipe car nous avons su créer une forte cohésion de groupes ce qui nous a permis d'atteindre tous nos objectifs.

Sandramohan Jean-marc

Une nouvelle expérience ! C'est trois mots résument mon sentiment globale sur ce premier projet du cycle ingénieur. Durant cette collaboration j'ai eu l'occasion pour la première fois de travailler avec une équipe motivée et dont « les nuits blanches » n'ont pas fait peur. Grâce à ce groupe j'ai pu notamment me focaliser sur mes missions au sein du projet sans avoir à repasser devant chaque membre pour vérifier l'exactitude de leur algorithme. C'est une expérience toute nouvelle pour moi car depuis mon entrée en école d'ingénieurs j'ai toujours dû jouer le rôle de « chef de groupe ». Et pour la première fois j'ai pu laisser ce rôle à Lauren qui a d'ailleurs très bien joué son rôle de chef de groupe.

Équipe équilibre, notre groupe avait l'avantage de ne pas avoir de « maillon faible », ce que je sous-entends par là c'est que chacun de nous savions coder et que l'aide d'une tierce personne n'était utile pour personne. Cela a notamment permis à chacun d'améliorer ses connaissances dans la programmation en Java, dans la manipulation de nouvelles librairies graphiques et dans la manipulation d'une base de données.

Cependant je pense que certains points seraient à travailler, notamment je pense surtout au partage des versions et de leurs fusions. Je pense qu'il aurait fallu trouver peut-être un autre moyen que celui utilisé pour ce projet car il a causé de nombreux retards dans la réalisation d'une version régulière complète. Et de nombreuses versions ont dû être remanié régulièrement faute de pouvoir les fusionner correctement.

Bakaloglou Lucas

Pour ma part, je trouve que ce projet a été vraiment bénéfique et m'a apporté beaucoup de plaisir notamment lorsque j'ai codé une des intelligences artificielles, car enfin on a pu voir comment aller agir des concepts écrits sur papier. De plus on peut voir directement quelles sont les conséquences directes de notre code. On peut voir l'effet de chaque paramètre et essayer de les modifier en fonction de la réponse du serveur mais surtout des autres IA.

Ce projet est aussi le premier, et je trouve ça très attrayant, pour moi et pour mes coéquipiers, de coder un programme qui se rapproche énormément d'un vrai logiciel, de par son interface graphique mais aussi par son intelligence artificielle, et pour se faire on a dû obligatoirement travailler ensemble 24h/24h. J'ai donc trouvé un intérêt énorme pour celui-ci. Et je trouve que la manière dont on l'a mené a vraiment été formidable de notre organisation mais aussi de par notre communication.

Brochard Jeremy

« Un résultat presque parfait ». C'est au final le sentiment que je retiens de ce projet auquel j'aurais quasiment consacré toutes mes vacances.

Pourquoi « parfait »? L'objectif initial était d'associer une interface claire et simple à une IA efficace. Au final nous avons réalisé une interface très complète, nous avons intégré la gestion d'une base de données et nous avons développé deux IAs très efficaces. Autant dire que nous avons largement dépassé nos objectifs !

Personnellement chargé de concevoir l'interface graphique, je suis très content du résultat. J'ai globalement réussi à faire ce que je voulais obtenir : c'est clair, sobre et complet. Par ailleurs, ce projet m'a permis de découvrir à quel point le langage JAVA était puissant et bien documenté ; ce qui m'a énormément aidé.

De plus, j'ai apprécié travailler avec une équipe motivée. On a très tôt défini un cap et des objectifs précis et cohérents. De même, on a très bien réussi à se partager le travail. C'était très appréciable et je pense que ça nous a permis d'atteindre nos objectifs, et plus encore.

Pourquoi « presque » ? Parce qu'avec le recul, je pense qu'on aurait pu aller plus loin si on avait mieux géré notre temps. En effet, j'ai personnellement perdu beaucoup de temps à naviguer entre les différentes versions de chacun. On a rajouté plein de détails et d'optimisations à notre code au fur et à mesure. C'était bien certes, mais avec une nouvelle version presque tous les jours, c'était difficile à suivre.

Bibliographies

- Salib Michel et Fancelli Charlie, « Green Planet », liens : <http://www.green-planet-project.comSalib>
- Oracle, tutoriels et documentations, liens : <http://www.oracle.com>
- Librairie et Tutoriels JFreechart, liens : <http://www.jfree.org/jfreechart/>
- Mig InfoCom librairie et tutoriels « MigLayout », liens : <http://www.miglayout.com>
- Paul Wheaton, forum Java liens : <http://www.coderanch.com/forums>
- Developpez.com, forum et tutoriels java, liens : <http://java.developpez.com>

ANNEXES

Figure 25: Diagramme de classe package contrôleur

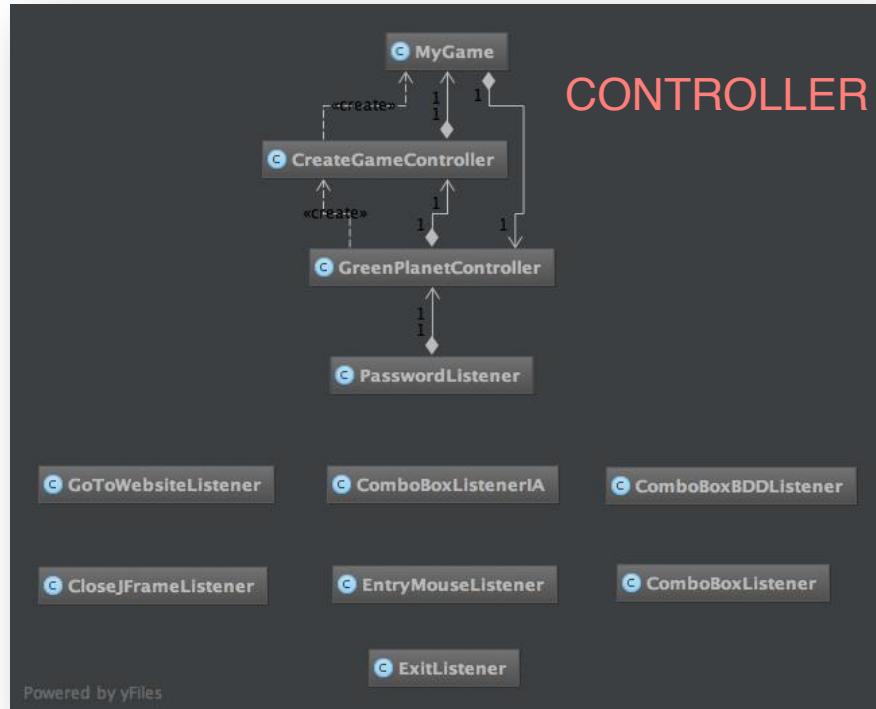


Figure 26: Diagramme de classe package vue

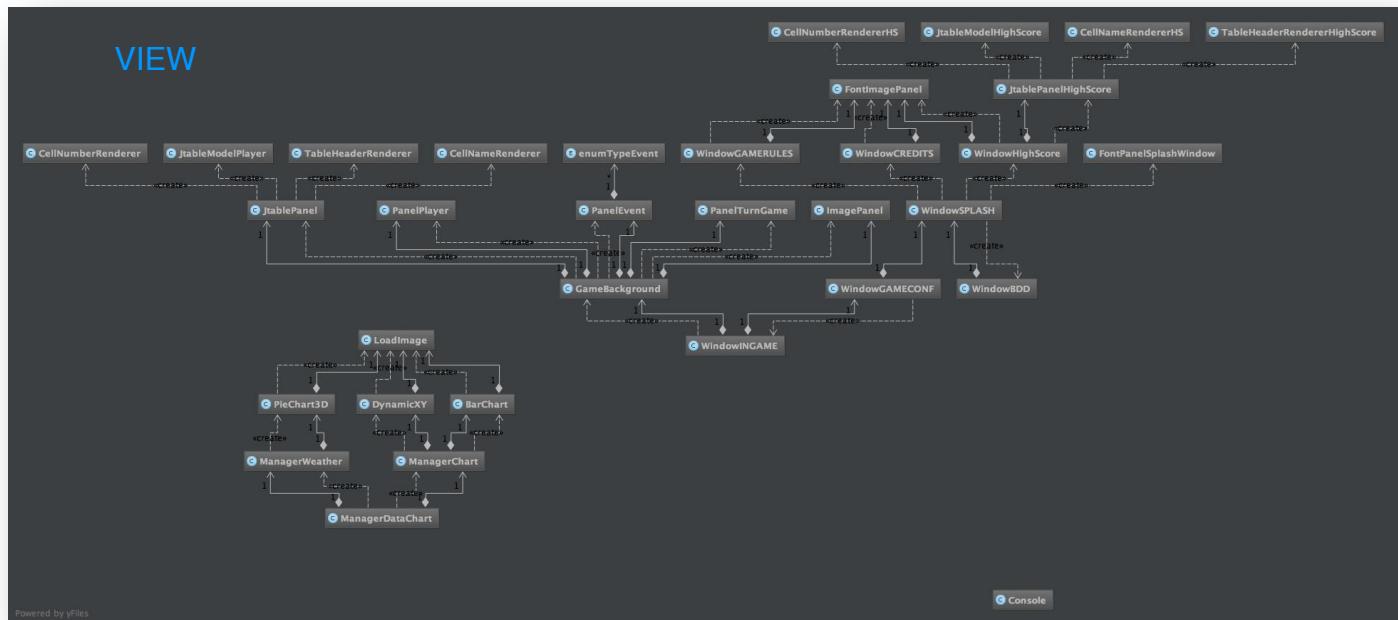


Figure 27: Diagramme de classe package modèle

