Guillaume ASSAILLY
Romain KLODZINSKI
Antoine MURA
Paul QUINZI

Architecture Logicielle et Qualité

RIRE & EMPIRE

Enseignant : Benjamin ANTUNES

03/01/2024

Table des matières

Prévisionnel

Cahier des charges

Présentation du projet

Contexte

Ce projet prendra la forme d'une simulation géopolitique multi-agent de la terre durant la guerre froide. Les enjeux sociaux, idéologiques, économiques et militaires seront pris en compte pour restituer au mieux l'activité incessante du globe à cette période.

Objectif

L'objectif de ce projet sera de pouvoir obtenir des simulations les plus fines possibles de l'évolution de la guerre froide depuis 1966 (cf. Contraintes techniques pour le choix de la date). Cette simulation aura pour pas de temps un jour, sur une durée déterminée en amont. En ce qui concerne l'échelle géographique, nos agents représenteront des pays ou des entités territoriales indépendantes. Certaines graines de générations pourront être conservées à titre d'exemple. Il sera également possible de faire varier les données de départ comme l'idéologie majoritaire initiale ou la taille des forces armées des pays afin de simuler un aspect plus précis voire uchronique de cette période.

Besoins Fonctionnels

Fonctionnalités principales

L'application finale prendra la forme d'une interface graphique ergonomique depuis laquelle l'utilisateur pourra accéder aux informations suivantes :

- Evènements en direct
- Diagramme d'idéologie de chaque pays
- Courbe de population de chaque pays
- Situation économique de chaque pays
- Visualisation des alliances
- Visualisation des conflits

L'utilisateur pourra faire avancer la simulation au rythme de son choix et même revenir en arrière pour observer à nouveau un évènement important.

Les paramètres de départ devront être choisis de manière avisée. En effet, une fois la simulation lancée, l'utilisateur ne pourra plus influer sur son déroulement.

Diagramme de cas d'utilisation

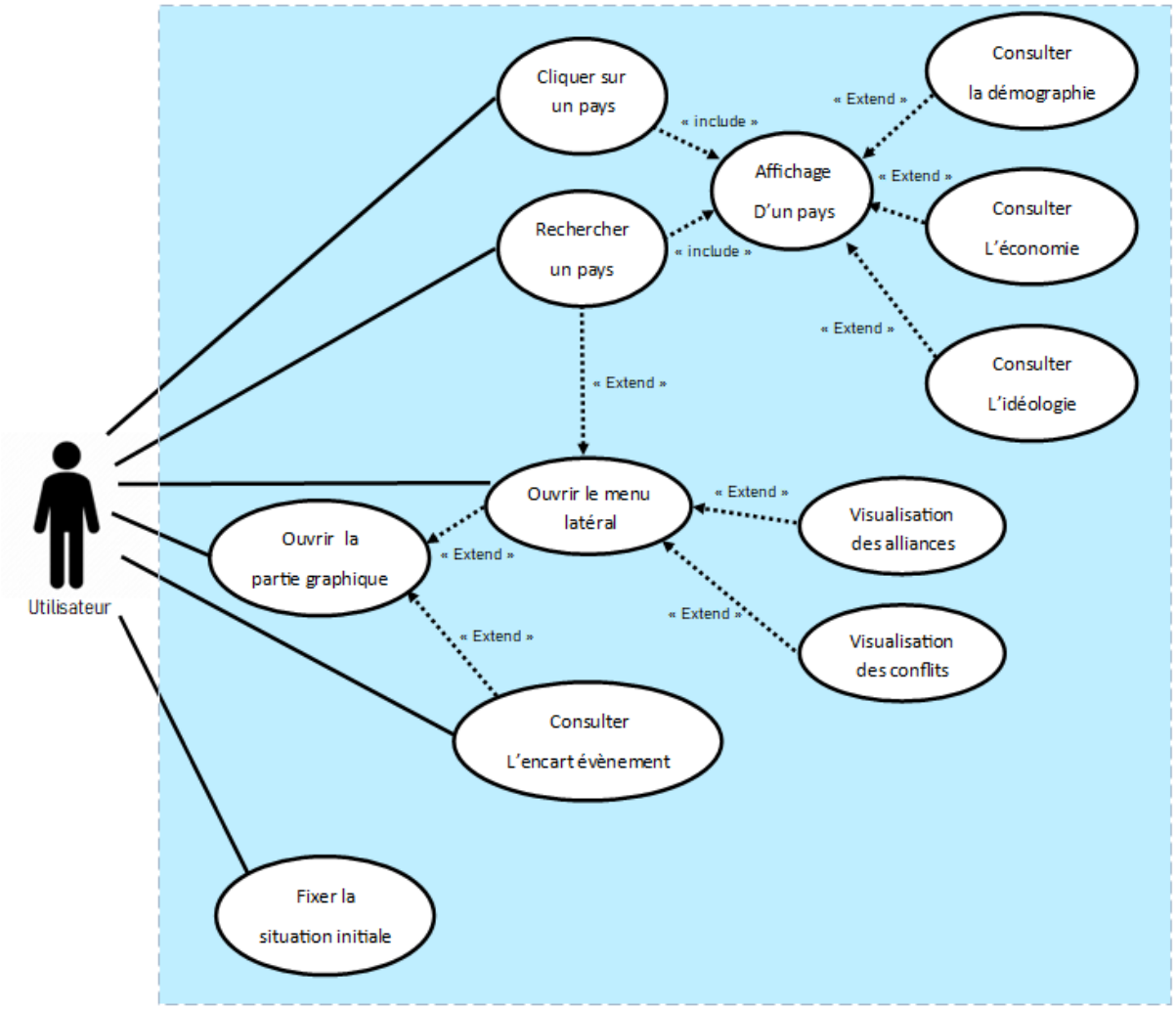


Figure 1 : Diagramme de Cas d'utilisation

Besoins Non Fonctionnels

Ergonomie

Chaque information devra être accessible avec le moins de cliques possibles sans pour autant que l'interface soit surchargée. Pour commencer, tout l'affichage sera effectué dans la partie graphique et non en console si ce n'est quelques informations de débogage. La navigation se fera sur un globe par des glissements de souris. Ensuite, un menu latéral permettra de visualiser toutes les informations des pays sous la forme de diagrammes et de courbes pour une meilleur lisibilité. Enfin, les informations relatives aux alliances et aux conflits seront sous la forme de calques pour éviter les effets de listes.

Charte graphique

Notre affichage principal s'inspirera des vues satellites telles que celle de Google Earth. Les frontières seront matérialisées seulement en dessous d'un certain niveau de zoom pour éviter la surcharge d'informations. Une attention particulière sera portée aux drapeaux des pays notamment en fonction de leurs changements d'idéologie.

Spécifications techniques

Back-End

Pour la simulation en elle-même, nous utiliserons une partie Back développée intégralement en C# grâce au framework .NET. Une version console de la simulation restera disponible directement depuis le Back. Elle utilisera une librairie DataSetExtensions. Une petite partie graphique de visualisation des données en 2D appelé Escalation existera aussi avec les bibliothèques LiveCharts ainsi que la package WPF-UI pour faire des applications graphiques sur windows. Cette application graphique est un outil de développement et en aucun cas le produit final. C'est cependant un outil de développement efficace pour le débogage et la visualisation facile des interactions entre agents sans avoir besoins d'attendre le développement de la partie Front.

Les versions minimums requises seront :

- CSharp : 4.7.0
- DataSetExtensions : 4.5.0
- LiveCharts : 0.9.7
- WPF-UI : 3.0.0

Front-End

La partie Front de notre application est intégralement développée en Python. Elle utilise Pillow, une bibliothèque d'affichage d'images ainsi que la bibliothèque Kivy initialement crée pour le développement d'applications Android avec sa collection principale de Material Design, KivyMD. Le framework Kivy3 sera utiliser pour les rendus 3D avec Kivy. Pour les graphiques, on utilisera Kivy_garden.graph.

Les versions minimums requises seront :

- Python : 3.11
- Kivy : 2.2.0
- KivyMD : 1.1.0
- Kivy3
- Pillow
- Kivy_garden.graph

Contraintes

Délais

Ce projet sera délivré à la date du dimanche 7 Janvier 2024 maximum selon les termes énoncés. Cependant, il est possible que des mises à jour interviennent et apportent de nouvelles fonctionnalités par la suite.

Contraintes Techniques

Revenons sur quelques contraintes techniques qui ont fixé des choix d'implémentation ou l'orientation générale du projet.

Le choix de la date de départ de la simulation en 1966 n'a pas été fait au hasard. En effet, la décennie 1960 est marqué par de nombreux changements notamment dû à la décolonisation et la prise d'indépendance de certains pays ou de leurs territoires. A partir de 1966, les pays ne sont pas exactement ceux que nous connaissons actuellement mais les tracés des frontières sont quasiment identiques à ceux qui nous sont contemporains. C'est pourquoi notre simulation sera réaliste au moins pour une durée d'environ 65 ans sur cet aspect là. Pour en rester sur les frontières, l'annexion partielle d'un pays ou la création d'un nouveau pays ne seront pas possibles en raison des incohérences que pourrait apporter la génération aléatoire de noms ou de drapeaux par exemple. Nous aurions pu utiliser notre connaissance jusqu'à aujourd'hui afin de séparer les pays comme dans le déroulé de l'histoire mais cela aurait donc faussé la simulation, la transformant en reconstitution.

Enfin, la contrainte centrale de notre projet est la communication entre la partie Back-end et la partie Front-end. Pour ce faire, nous utiliserons une sérialisation en JSON afin de transmettre les informations jour par jour au Front. Les fichiers JSON produis seront stockés dans un dossier permettant au Front d'avancer à son rythme et de revenir en arrière si besoin tandis ce que le Back simulera l'intégralité de la durée prévue. C'est pour cela qu'aucune action utilisateur ne pourra influer sur la simulation.

Plan d'organisation

Contexte

Comme vous le savez, le rapport que vous lisez se rapporte à un travail qui est le fruit de la fusion de deux groupes. Par conséquent la plupart des diagrammes prévisionnels prévus au début du projet sont maintenant obsolètes. Par soucis de cohérence, nous présenterons ici des versions corrigées des versions de départ et nous expliquerons au fur et à mesure qu'elles sont les modifications qui sont intervenues.

Pour commencer, chaque groupe avait au départ une structure du type Back-end Front-end. Dans la version finale du projet, la majorité de la partie Back-end d'un des groupes

a été conservée tandis ce que la partie Front-end viens majoritairement de l'autre groupe. Voyons donc comment le projet devait s'articuler sur ces parties.

Diagrammes de Pert

Commençons par le Back. Voici la maquette prévisionnelle des tâches à effectuer. La fin prévue étant le 27 décembre, cela devra laisser une marge suffisamment confortable pour s'adapter à un problème ou changement important comme la fusion de deux groupes par exemple. En parallèle de cela, un travail sur une interface graphique (Escalation) était présent notamment pour visualiser les avancés dans la simulation. Ce travail n'apparaît pas ici dans cette version corrigée. On peut voir que la parallélisation de l'implémentation de certaines fonctionnalités devra nous permettre de faire fonctionner plus intuitivement ces aspects entre eux. Le comportement d'un agent indépendamment des autres pays doit être cohérent sur tous les aspects. Cela est plus simple à réaliser lorsque l'implémentions des composants de cet agents est récente. C'est aussi le cas des guerres dont le déroulement est forcément étroitement lié aux relations entre les pays et donc aux alliances.

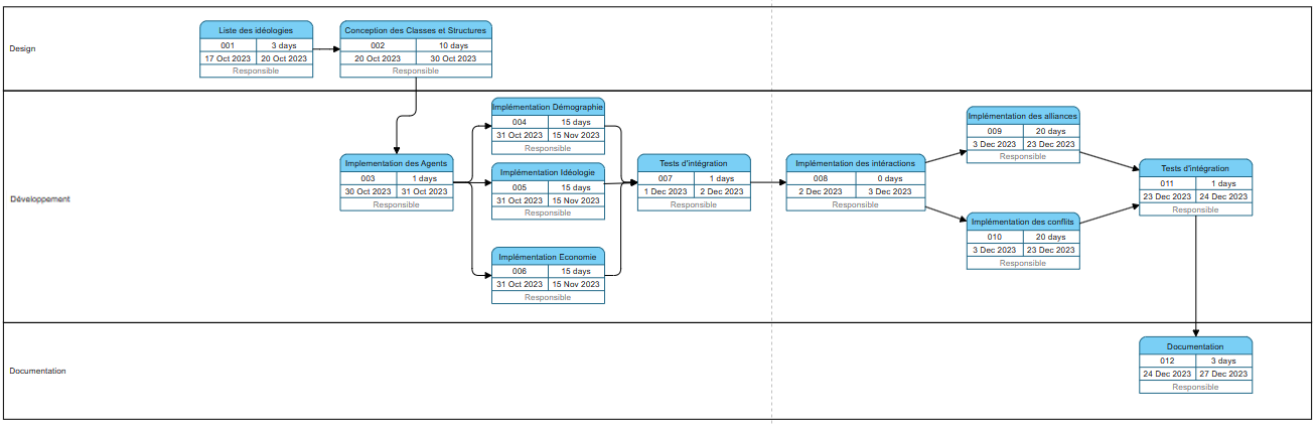


Figure 2 : Diagramme de Pert Back-End

En ce qui concerne la partie Front, l'évolution est plus linéaire. En effet, il est plus difficile de travailler sur des tâches en parallèle. Il n'est pas évident de rajouter de la texture si la forme n'existe pas ou de rajouter des sources de lumière sur un écran noir par exemple. Dans ce diagramme modifié, nous avons choisi de conserver la fusion avec le Back prévue puisque le déroulement des évènements coïncide avec cette version si ce n'est que la simulation n'est pas en python comme prévu initialement mais en C# ce qui est nécessairement un peu moins intuitif.

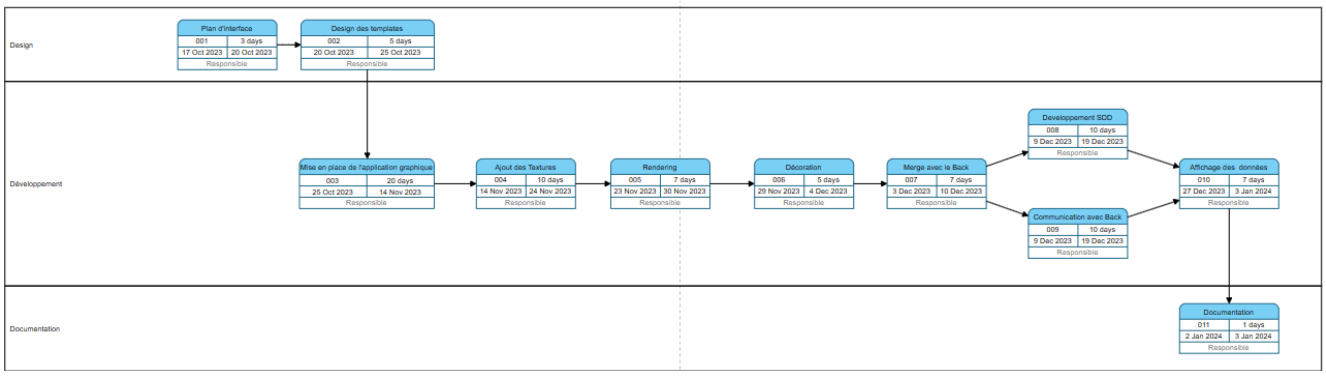


Figure 3 : Diagramme de Pert Front-End

Réel

Utilisation de l'interface

Figure 4 : Interface Principale

L'utilisation de notre application est assez intuitive. Toutefois, il est important de comprendre les quelques menus disponibles lors de son lancement. Pour commencer, il est possible que l'application mette un certain temps à se lancer en raison du chargement de l'image et des autres données. Là s'affichera la terre en rotation ainsi qu'un bandeau d'affichage d'évènements et un menu latéral à droite. Pour l'instant, la simulation n'a pas commencé. Au-dessus du bandeau d'évènements, vous trouverez un bouton « démarrer » pour lancer la simulation et son affichage. Vous pourrez ensuite naviguer dans cette simulation en cliquant sur la date pour ouvrir un calendrier. Dans le menu de droite. Vous trouverez dans l'ordre une barre de recherche des pays qui sera d'ailleurs le seul moyen d'accéder aux données de petits pays comme le Vatican par exemple. Vous trouverez aussi la possibilité d'afficher les caractéristiques générales du pays. Enfin, les deux derniers onglets du menu au démarrage permettront respectivement de visualiser les alliances que l'on pourra matérialiser en cochant la case en haut à droite et les guerres qui pourront être afficher de la même façon.

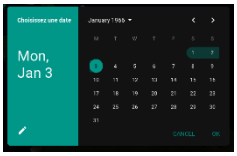


Figure 5 : Choix de la date

Figure 6 : Affichage des Alliances

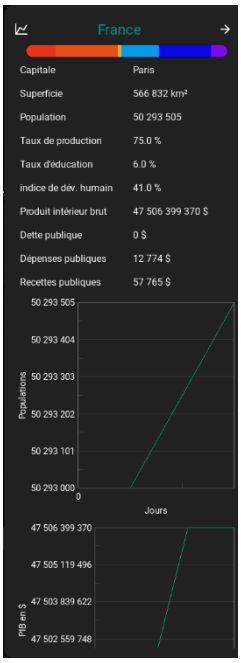


Figure 7 : Affichage Pays

Dans les onglets de guerres et d'alliances, on pourra retrouver la liste des pays concernés en développant grâce à la flèche à droite. Cela peut être une bonne façon d'observer nos drapeaux générés dynamiquement selon l'idéologie des pays.

Dans les caractéristiques du pays, on retrouve une barre indiquant l'idéologie. Il est possible de survoler cette barre avec la souris pour avoir plus d'informations. On y retrouve également les graphiques démographiques et économiques.

Figure 8 : Menu latéral

Recherche de pays

Informations sur le pays

Informations sur les alliances

Informations sur les guerres

Fonctionnement de la simulation

Fichiers d'entrées

```
RUS;0;1;1;232243000;1.15;1.327;7.6;370;255;102;6,92700000000;250;43
```

Figure 9 : Format fichier.csv entrée Russie

Les données relatives à chacun des pays sont codées en dur dans un fichier csv lu en entrée du programme. Chaque ligne correspond à un pays et on peut y retrouver les informations suivantes :

- Le code identifiant du pays
- Le numéro de son idéologie majoritaire (correspondant à l'énumération des idéologies, soit 0 pour communisme, 1 pour socialisme, etc
- La valeur min et max de cette idéologie majoritaire : par exemple, ici, la Russie commencera à 100% de communisme. Mais dans le cas des USA :

```
ETU;3;0.8;1;196560000;1.16;0.842;35.7;402;336;333;8,15000000000;300;30
```

Figure 10 : Format fichier.csv entrée USA

  Leur idéologie majoritaire est la démocratie de droite, avec pour valeur min 80% et max 100%. Cela signifie que sur les 7 idéologies présentes dans la simulation, seule l'idéologie majoritaire est fixée aléatoirement entre 80 et 100, laissant les 6 autres définies aléatoirement. Ce procédé permet de coder en dur toutes les idéologies principales sans avoir à stocker les 6 idéologies mineures de chaque pays, et de préciser avec ces deux indicateurs, à quel point elle est hégémonique.
- La population au début de la simulation (donnée historique)
- La croissance de la population au début de la simulation (donnée historique)
- La mortalité de la population (donnée historique)

- La densité de la population : indicateur qui pourrait permettre dans le futur d'être corrélé au calcul des victimes civiles des guerres, mais non utilisé pour le moment.
- Puissance industrielle : indicateur du niveau industriel du pays.
- Puissance agricole : indicateur du niveau agricole du pays.
- Puissance du tertiaire : indicateur du niveau de tertiarisation du pays (en 1966, les pays sont peu tertiarisés, mais tendront à le faire, si leur économie le leur permet, en temps de paix).
  Chacun de ces indicateurs ont été codés en dur pour singer le niveau de développement de tous les pays au début de l'an 1966. Ces valeurs permettent de calculer l'essor économique du pays. L'indicateur industriel, en plus de fournir de la croissance économique, est également « reconverti » en temps de guerre pour améliorer le niveau militaire d'une nation.
- Le PIB du pays en 1966.
- La puissance militaire, valeur arbitraire chargée de représenter la puissance et la léthalité d'une armée nationale.
- Les points de victoires d'un pays : autrement dit, ses points de vie avant de capituler et d'être annexé. Dans un futur proche, on pourrait avec une base de données, associer chacun de ces points à une ville majeure du pays/territoire.

L'autre fichier essentiel en entrée est une matrice d'adjacence des pays, pour connaître leurs voisinages. On ne considère pas un pays voisin uniquement s'il partage une frontière, mais plutôt s'il fait partie de la même sphère d'influence (ex : pays d'Europe de l'Ouest, pays du Moyen-Orient, etc.). Ce fichier contient les caractères suivants : X si pas voisins, L si voisin. Une troisième lettre pourrait faire son apparition M : si on tenait compte des frontières maritimes.

Principe de la simulation :

Le parti pris pour la modélisation a été l'axiome suivant : les fluctuations idéologiques au niveau national expliquent l'intégralité des interactions géopolitiques sur le plan international, y compris les intérêts changeant d'une nation. Cette thèse peut faire sourire par sa simplicité, et son inexactitude réelle, mais elle nous a semblé être un bon point de départ pour réaliser une simulation qui produise des données cohérentes.

Les idéologies et luttes idéologiques comme force motrice de l'Histoire

Au sein d'une nation, parmi toutes les données économiques et industrielles, résident le plus importantes de toutes : la liste des idéologies par pourcentage dans ce pays. L'évolution de ces dernières au fil de l'histoire est la suivante :

- En temps de paix, elles évoluent en suivant ce comportement : une idéologie peut soit se renforcer, au détriment des autres donc (puisque tous leurs pourcentages se compensent), soit s'affaiblir, à la faveur d'une autre idéologie parmi les minoritaires de ce pays (ou les majoritaires d'un voisin aléatoire de ce pays. Le niveau de renforcement naturel et d'affaiblissement naturel de chaque idéologie dépend de sa nature : des idéologies plus totalitaires auront tendances à se renforcer plus une fois

qu'elles arrivent au pouvoir (quand elles deviennent les premières au classement du pays), tandis que les idéologies plus libertaires ou plus mondialistes ont tendance à s'affaiblir plus rapidement où être plus influencées par les pays voisins.

- En temps de guerre : un pays annexé perd ses idéologies et prend celles du conquérant. C'est comme cela que l'on peut observer les revirements les plus brutaux en termes d'idéologie.

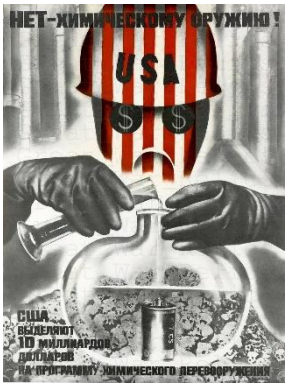


Figure 11 : Affiches de propagande 1951

Politique intérieure

Chaque pays, à chaque pas de temps, évolue, change, traverse des crises financières, sociales, etc. Plusieurs indicateurs représentent ces problématiques : certains évidents, PIB, natalité, etc. d'autres plus imagés comme un indicateur d'éducation, de santé publique, etc. Absolument tous ces indicateurs influent les uns sur les autres : un pays de forte productivité et de bon niveau d'éducation aura une très bonne croissance de son PIB, mais des dépenses publiques plus importantes. Pareillement, un pays d'un bon niveau de santé publique sera peut-être plus déficitaire, voir endetté (comme certains pays peuvent l'être au cours de notre simulation), mais aura une bien meilleure croissance de population, etc.

Toutes ces données ne sont pas vouées à être figées dans le temps. Elles vont bien évidemment évoluer au cours de ce que nous avons modélisé comme des plans politiques.

Un plan politique est une représentation imagée et arbitraire d'une action de gouvernance, destinée à impacter positivement ou négativement tout indicateur de la politique intérieure d'un pays. (Nous pourrions plus tard rajouter des plans politiques pour les relations internationales). Ces plans dépendent de deux choses :

- L'idéologie majoritaire du pays : c'est le propre des idéologies, elles n'offrent pas les mêmes réponses aux mêmes problèmes.
- La situation économique du pays : les dépenses sont-elles à l'instant T plus élevées que les entrées d'argent ?

Un plan politique prend une certaine durée avant d'être complété. Pendant toute sa durée de complétion, un pays ne peut pas changer de plan politique, et les effets de ce plan ne s'appliqueront qu'à sa complétion.

Ainsi, suivant ce petit principe, nous pouvons modéliser le « cerveau » d'un pays :

- Il a 50% de chance de prendre un plan lié à son idéologie majoritaire. Par exemple si l'idéologie majoritaire est le socialisme, il a une chance de prendre un plan de la famille du socialisme comme le plan écologique de démantèlement de l'industrie lourde : ce plan fait baisser le nombre d'industrie mais augmente la santé publique.

Ce n'est pas un plan rentable au niveau économique, mais cohérent avec la politique que pourrait mener un gouvernement à dominante écolo/socialiste.

- Il a 50 % de chance d'être pragmatique : dans ce cas, plus de pulsion idéologique et de beaux discours. Le pays pioche un plan générique (commun à tous les pays peu importe leur idéologie), qui dépend de la situation du pays. S'il est déficitaire, il prendra forcément un plan pour améliorer son économie.

Enfin, si une nation prend part à au moins une guerre, elle prendra différents plans exclusifs au temps de guerre, qui servent uniquement à améliorer son armée. Ce sont des plans d'urgences qui représentent les conversions de l'industrie civile en militaire.

Ce fonctionnement a deux avantages :

- Premièrement, il permet de simuler des politiques drastiquement différentes au sein d'un monde bipolaire (multipolaire par la suite). Chaque bloc idéologique fera face, même en temps de paix (qui peut représenter la majorité du temps durant une guerre froide) à de nombreuses problématiques qui pourraient même impacter son niveau opérationnel au moment où des guerres plus globales se déclencheraient.
- Deuxièmement elles permettent de « simuler » (c'est un bien grand mot) les deux principes économiques capitalistes et communistes de l'époque. Tous les plans libéraux sont très rapides à mener, avec peu de bénéfices chacun, mais s'adaptent donc parfaitement à la situation sociale et économique à l'instant T. Les plans politiques des pays communistes comme le plan quinquennal historique, sont beaucoup plus long à mener, mais offrent de grands bénéfices. Puisqu'un pays, une fois son plan choisi, ne peut pas le changer, un plan politique de longue durée est beaucoup moins approprié aux problématiques actuelles du pays (sur un plan de 5 ans, 3 guerres ont le temps d'éclater par ex, ce qui changerait drastiquement le contexte depuis le moment où il a initié ce plan). Ce principe permet de modéliser des économies plus « planifiées ».

Les relations internationales

Un pays n'est jamais seul au monde. Il est en relation, à plusieurs niveaux, de prés ou de loin, avec toutes les nations du monde. Ces relations se traduisent par une matrice géante qui représente le score d'amabilité entre deux pays. Plus ce score est élevé, plus ils sont en bons termes. Cette matrice est la seule manière que nous avons d'établir des guerres entre des pays, des alliances, des complots, etc.

Le facteur le plus important dans les relations est celui, évidemment, de l'idéologie principale du pays. Mais toutes les idéologies ne se détestent pas de la même manière, et suivent plutôt la matrice suivante :

```
{ {  100,  70,  20, -10, -50, -50, -100 },
  {   70,  100, 50,   0, -20, -60,  -80 },
  {   20,  50, 100,  20, -20, -60,  -80 },
  {  -10,   0,  20, 100,  50,   0,  -30 },
  {  -50, -20, -20,  50, 100,  40,   20 },
```

Figure 12 : Matrice d'entente des idéologies

Pour chacune des 7 idéologies, certaines se haïssent (à raison) : le communisme & le fascisme, mais certaines sont bien plus complaisantes : centre droit démocratique et centre gauche.

L'autre facteur déterminant est le voisinage. Nous avons modélisé ce principe comme suit :

- Deux pays voisins, qui ont la même idéologie s'adorent encore plus.
- Deux pays voisins qui ont des idéologies différentes ou opposées se détestent.

Cela permet de renforcer des pays voisins, leur donnant plus de chance de s'allier s'ils sont du même bord, ou plus de chance de se déclarer la guerre (pour modéliser des conflits territoriaux et locaux).

D'autres facteurs minoritaires sont également calculés pour établir le niveau d'entente, comme l'appartenance à une alliance différente, être en guerre contre un allié, etc.

Tension mondiale et guerre nucléaire

La période que nous avons choisi, la guerre froide, est excellente pour deux choses : décrire le conflit (virtuel ou réel) entre deux idéologies, et simuler une tension mondiale fluctuante (période de la détente ou au contraire de l'escalade).

Un indicateur tout simple de 0 à 100, permet de représenter cette tension mondiale, et est calqué sur le niveau DEFCON utilisé par l'armée américaine.

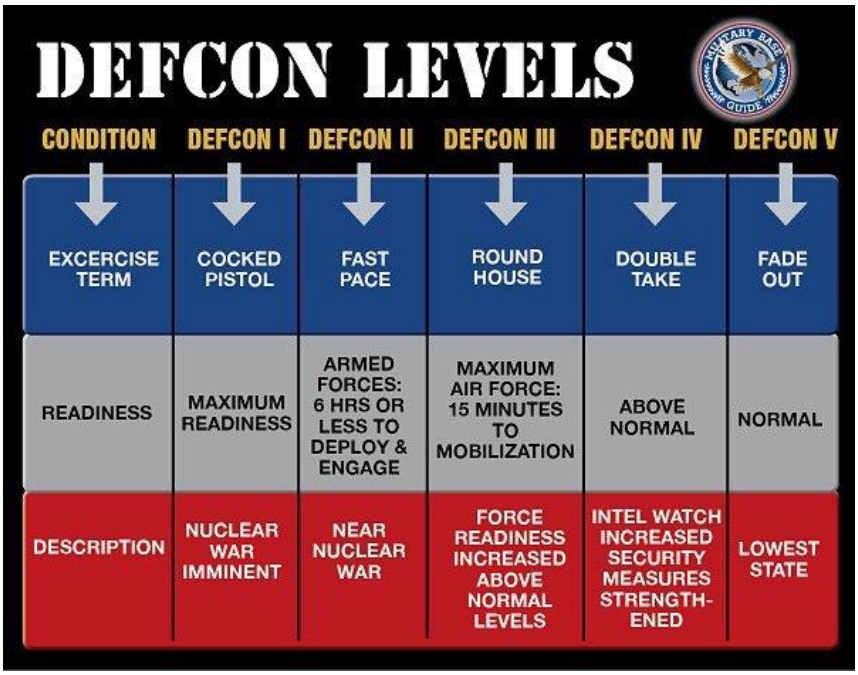


Figure 13 : DEFCON

Une tension mondiale de 0 à 20 correspond à DEFCON 5, de 20 à 40 à DEFCON 4, etc. Evidemment, en fonction du niveau de DEFCON, les règles du jeu ne seront pas les mêmes (ce sera décrit plus bas).

La tension mondiale tend à baisser naturellement (ce paramètre peut être modifié pour simuler plus facilement la probabilité d'une situation de détente plutôt que d'escalade).

Mais certains événements tendent à la faire monter : les guerres et les créations d'alliances. (NB : Les traités de paix font chuter la tension). Plus la tension augmente, plus les pays qui

ont de mauvaises relations tendent à entrer en guerre les uns les autres (où à rejoindre les guerres existantes pour profiter du chaos général).

Mais le niveau de DEFCON fait varier quelques règles :

- Prenons par exemple deux pays : l'Allemagne de l'Est et l'Allemagne de l'Ouest. Ces deux pays ont tout pour se détester : idéologies principales différentes, membres d'alliances militaires différentes, voisins l'un l'autre, etc. Il y a de fortes chances que ces deux pays se déclarent la guerre même en temps de paix (DEFCON 5), et cela ne serait pas logique et cohérent avec un contexte historique.

  C'est pourquoi, avant DEFCON 2, des pays membres d'une alliance militaire ne peuvent pas se déclarer la guerre l'un l'autre.

  Et cela a de très gros bénéfices. En situation de paix, les pays membres d'alliances ne se déclareront pas la guerre (NB : un pays même si membre par ex de l'Otan, pourra toujours entrer en guerre contre un pays non aligné type Vietnam). Les guerres apparaîtront donc d'abord sous la forme de conflits régionaux. Et là, plusieurs issues sont possibles : soit le conflit régional se termine par un accord de paix, soit il entraine d'autres états dans un conflit plus large. Dès qu'un pays rejoint une guerre, on ajoute proportionnellement sa puissance militaire à la tension mondiale.

- Pour reprendre notre exemple, un conflit régional au Moyen-Orient pourrait dégénérer jusqu'à faire augmenter la tension mondiale suffisamment pour que l'Allemagne de l'Est et de l'Ouest ait le droit de se déclarer la guerre, et d'embarque avec eux leurs alliés respectifs (et c'est parti pour une guerre nucléaire !).

Une autre règle par exemple est la suivante : entre DEFCON 5 et 3, les pays peuvent créer ou quitter des alliances existantes. A partir de DEFCON 2, les pays ne peuvent plus créer d'alliances et ont de très forte chance de rejoindre l'alliance avec laquelle ils ont le plus de sympathie.

- Puisque les alliances sont créées d'après notre matrice de relation, et que cette matrice dépend majoritairement de l'idéologie des pays, à chaque simulation, on peut observer que les alliances se créent par blocs idéologies : pacte fasciste, pacte despotique, pacte libéral, etc.

Guerres et annexions

C'est bien regrettable, mais nous voilà en guerre. Une guerre est constituée de nations attaquantes, et de nations défenseures. Au sein d'un même camp, ces nations peuvent être de la même alliance, ou même d'alliances différentes selon les circonstances (ou bien sûr non alignées).

Les guerres provoquent des morts civils chaque jour parmi les pays belligérants. Les guerres sont des ensembles de batailles, qui opposent des pays en comparant leur score militaire (avec un peu d'aléatoire pour plus de fun). Chaque bataille permet de faire regagner ou perdre un point de victoire à un pays.

Une nation sans point de victoire est annexée par le pays le plus puissant du camp opposé. Une annexion signifie plusieurs choses :

- Le pays change d'idéologie et prend celle du conquérant.
- Le pays entre dans l'alliance du conquérant (s'il est membre d'une alliance)
- Le pays entre en paix avec tous les pays du monde (mais généralement rejoindra quasi immédiatement la guerre au côté de son conquérant, puisqu'il se met à obtenir des relations très cordiales).

Une limitation de notre simulation est de ne pas encore prendre en compte le concept de rancœur d'une nation conquise pour son conquérant, et d'au contraire les rendre copains comme cochon (ce serait faire fi de plusieurs crimes de guerres un peu trop facilement…)

Conception

Contexte

Pour commencer rappelons le dérouler des évènements. Nos deux groupes étant respectivement intéressés par un simulateur géopolitique, nous avons fait le choix de partir dans des directions différentes. Le premier groupe partirait sur un projet centré sur la période de la guerre froide, avec un aspect idéologique et économique très poussé, tandis ce que le deuxième effectuerait une simulation depuis l'époque contemporaine, centrée sur des aspects plus démographique et militaire. Cependant, durant la dernière ligne droite du projet. Nous nous sommes aperçus que les faiblesses de chaque projet étaient compensées par les forces de l'autres. C'est pourquoi nous avons demandé une mise en commun des groupes que vous avez accepté et nous en sommes reconnaissant. Cela ne fut pas sans mal que nous avons créé ce nouveau projet, en utilisant majoritairement les projets initiaux comme Back et Front. Le titre « Rire & Empire » est issu de l'imagination débordante d'internet combinée au sens de la formule de ChatGPT. Il a été sélectionné parmi un grands nombre d'autres candidats.

Architecture Globale

L'architecture globale de notre application est plutôt simple. Une simulation s'effectue en Back-End sur une durée déterminée à partir d'un jeu de donnée dont nous décrirons le fonctionnement dans la rubrique suivante. Pour chaque jour simuler, les données du monde entier sont sérialisées dans un fichier JSON qui sera accessible par le Front. C'est ainsi que se fera la liaison entre le C# et le Python.

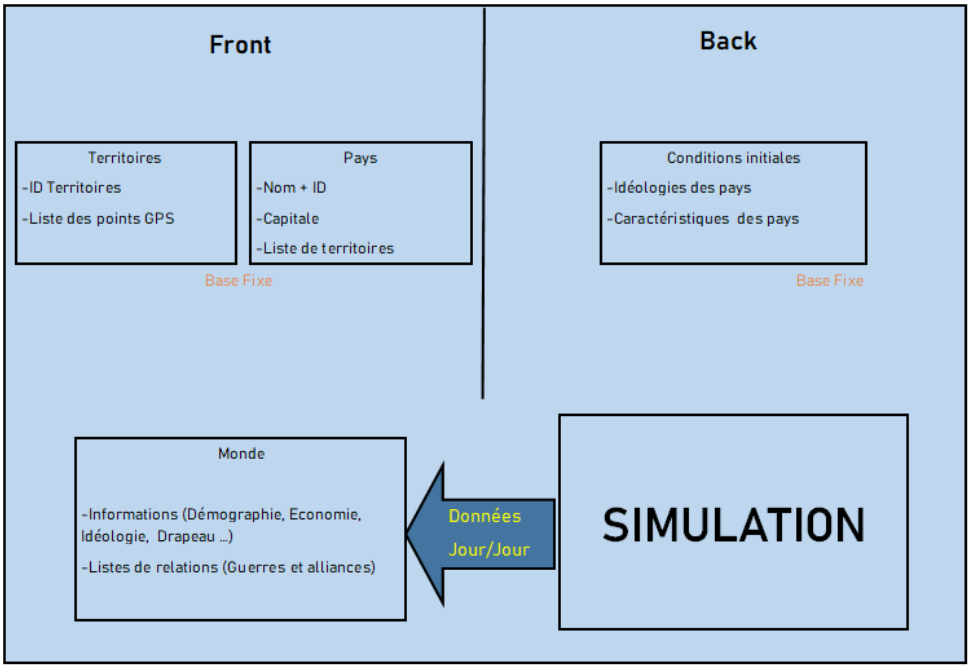


Figure 14 : Schéma architecture globale

Back

Le back, quant à lui, gère l'évolution des agents et produit quotidiennement (du point de vue de la simulation) en sérialisant un objet représentant le monde. Son fonctionnement détaillé et les étapes de sa création seront détaillés dans la partie consacrée de ce rapport.

Diagramme de classe

Passons maintenant au diagramme de classe. Il s'agira d'une version légèrement simplifiée dans lequel certains attributs ou certaines fonctions n'apparaitrons pas en raison de leurs utilisations locales ou de leur aspect standard. C'est notamment le cas des constructeurs, des accesseurs ou des propriétés publiques des attributs privés.

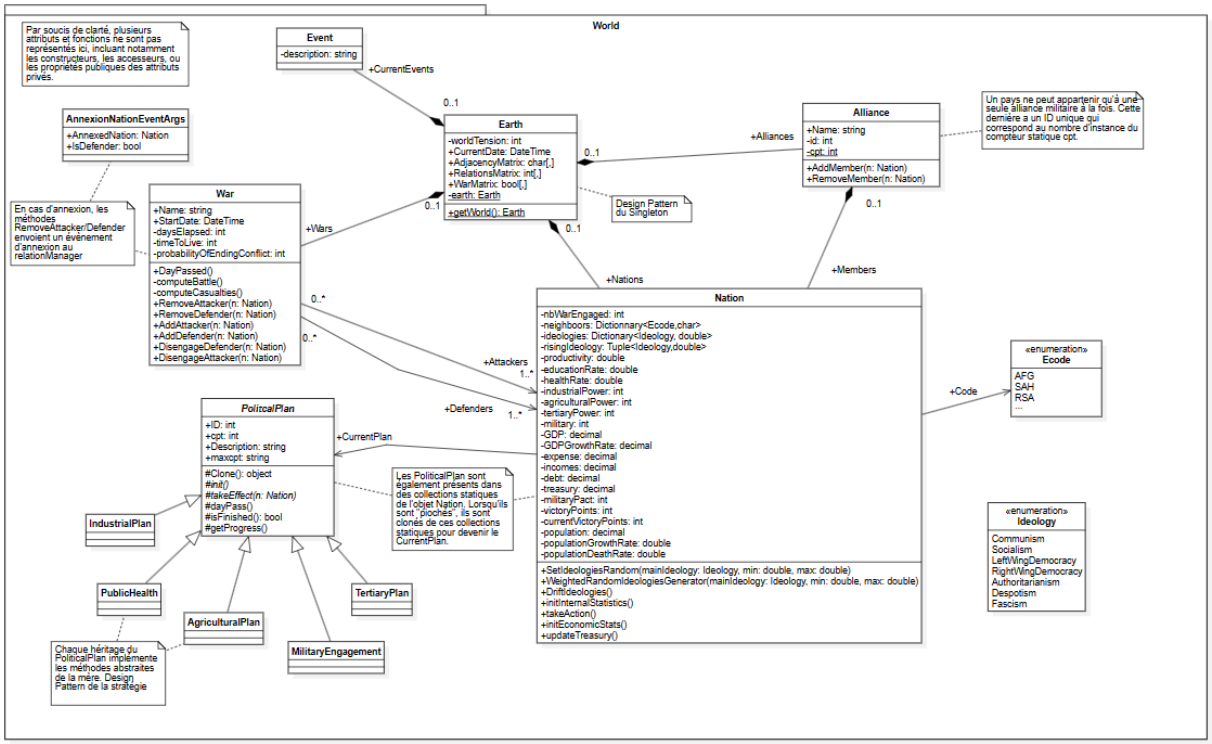


Figure 15 : Diagramme de classe Package World

Le package le plus important de la conception de notre Back en terme sémantique est celui qui représente le monde. C'est d'ailleurs cette partie qui, une fois sérialisée sera envoyé au Front pour l'affichage. On y retrouve une classe représentant la terre (*Earth*) dans laquelle on calcule le niveau de tension global ou encore la matrice d'adjacence des pays. Cette classe est composé d'un ensemble de nations mais également d'une classe de gestion des alliances (*Alliance*)et d'une autre de gestion des conflits (*War*).

La classe alliance sert à instancier des objets représentant des convergences entre pays pour des raisons diverses (idéologie, stratégie …). Les pays concernés dans des alliances auront plus de chances de se rallier en cas de conflit généralisé. Pour éviter tout problème de conflits internes aux alliances, chaque pays ne pourra appartenir qu'à une seule à la fois.

En ce qui concerne les guerres, on y retrouvera une liste d'attaquants et de défenseurs ainsi que les données propres au conflit. C'est également par cette classe que se fera l'accès au système d'annexion. En effet, en cas de défaite absolue d'un des camps, certains pays pourront être entièrement annexés. Tout cela grâce à la classe d'annexion (*AnnexionNationEventArgs*).

Au-delà de son implication dans les autres classes, les instances de la classe nation ont également leur comportement propre notamment d'un point de vue politique. C'est le rôle de la classe abstraite représentant le plan politique (*PoliticalPlan*). De nombreuses classes dérivent de celle-ci et représente les méthodes propres à chaque stratégie conformément au patron de conception du même nom.

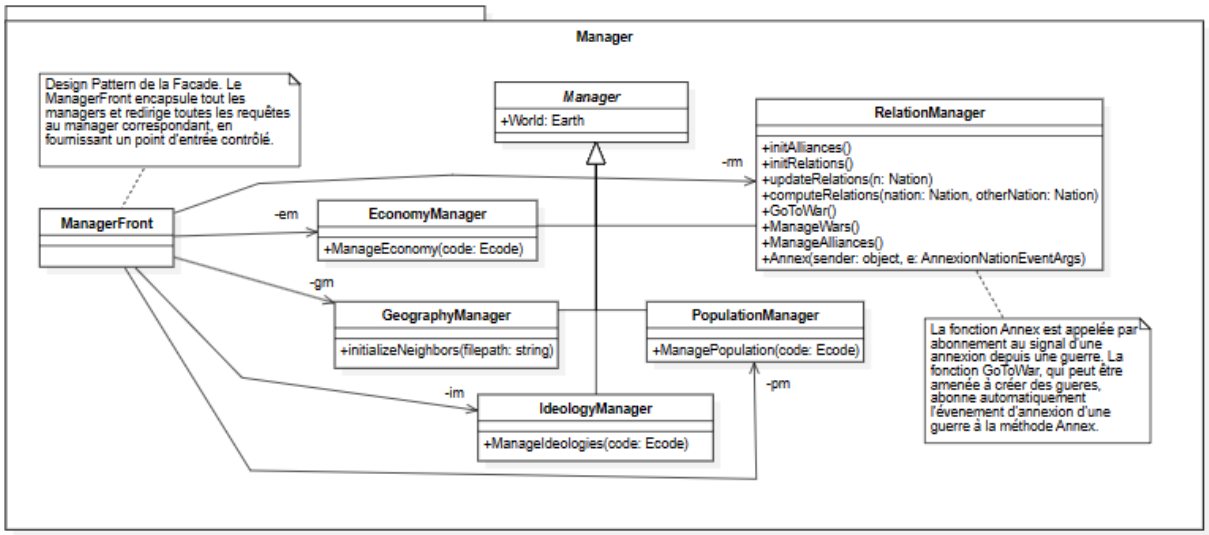


Figure 16 : Diagramme de classe Package Manager

Ensuite, on dispose d'une couche de gestionnaire. Ces managers permettent de coordonnées les fonctionnalités de la couche inférieur et de gérer de manière plus modulaires le fonctionnement du système. On y retrouve un gestionnaire pour chaque aspect important de notre simulation vu précédemment qui appellera et validera les données obtenues par nos fonctions. Certains managers s'occupent de plusieurs types de données, en l'occurrence les guerres et les alliances. Certains managers permettent par exemple de gérer des caractéristiques des nations comme celui dédié à l'économie, la géographie et l'idéologie. Un manager global contiendra une instance du monde courant.

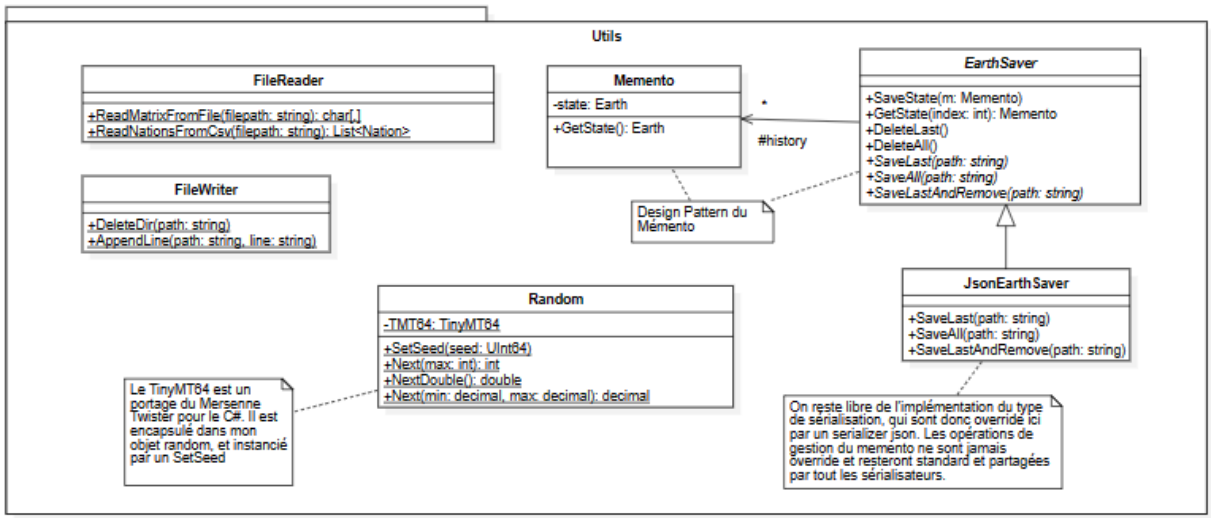


Figure 17 : Diagramme de classe Package Utils

Enfin, on retrouve dans cette partie tout un ensemble de classes utilitaires. Notamment le système de sauvegarde du monde dont hérite la classe de sérialisation JSON qui permet de communiquer avec le Front. On y retrouve également les classes englobant le system de lecture et écriture en fichier afin de pouvoir concevoir et récupérer les valeurs d'entrée de la simulation.

Patrons de conception

Voici une explication un peu plus détaillée de nos choix de conception concernant les Design Pattern vues en cours

Patron stratégie

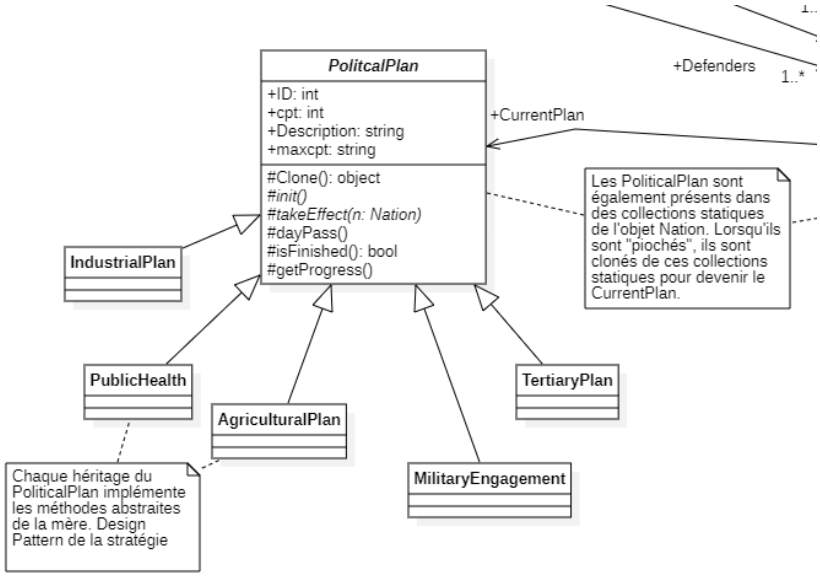


Figure 18 : Patron de conception Stratégie

Chaque plan politique implémente une classe abstraite *PoliticalPlan*. Leur cycle de vie est le même : *init()*, *dayPass()*, *takeEffect()*, *isFinished()*. Mais leur comportement est différent à chaque fois (c'est le propre d'ailleurs d'un plan politique). On utilise donc un patron stratégie pour encadrer les implémentations possibles concrètes de chaque plan, et unifier la manière dont on va y accéder.

Patron de la façade

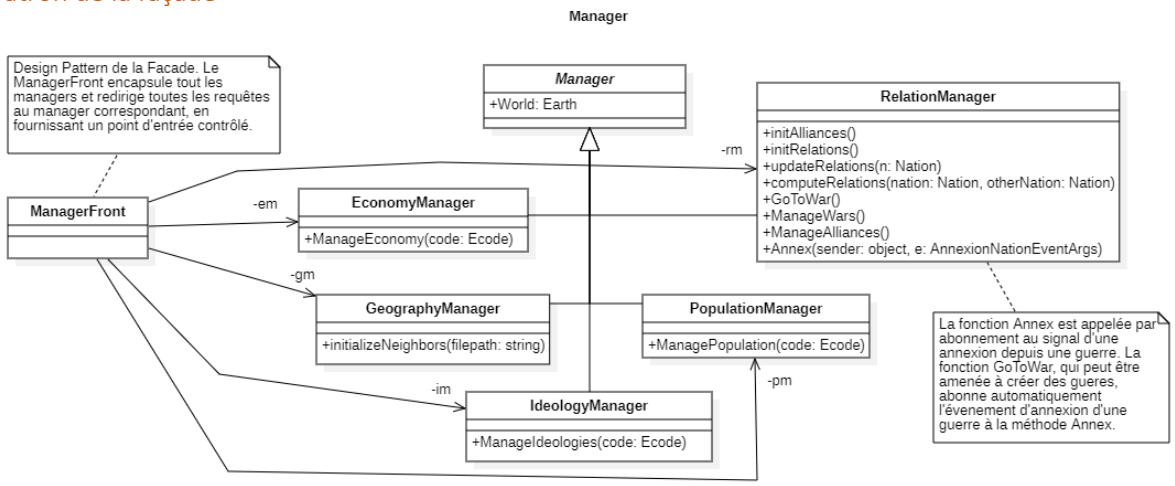


Figure 19 : Patron de conception Façade

Pour régir la multitude d'algorithmes qui paramètrent la simulation, on passe par un *ManagerFront*, principal, qui sera l'unique point de contrôle du programme sur les managers, et qui redirigera chaque requête au manager correspondant. Séparer et déléguer des responsabilités différentes à chaque manager permet également de simplifier le code, améliorer la lisibilité du projet et favoriser la maintenabilité voire la modularité future des objets.

Patron du Memento

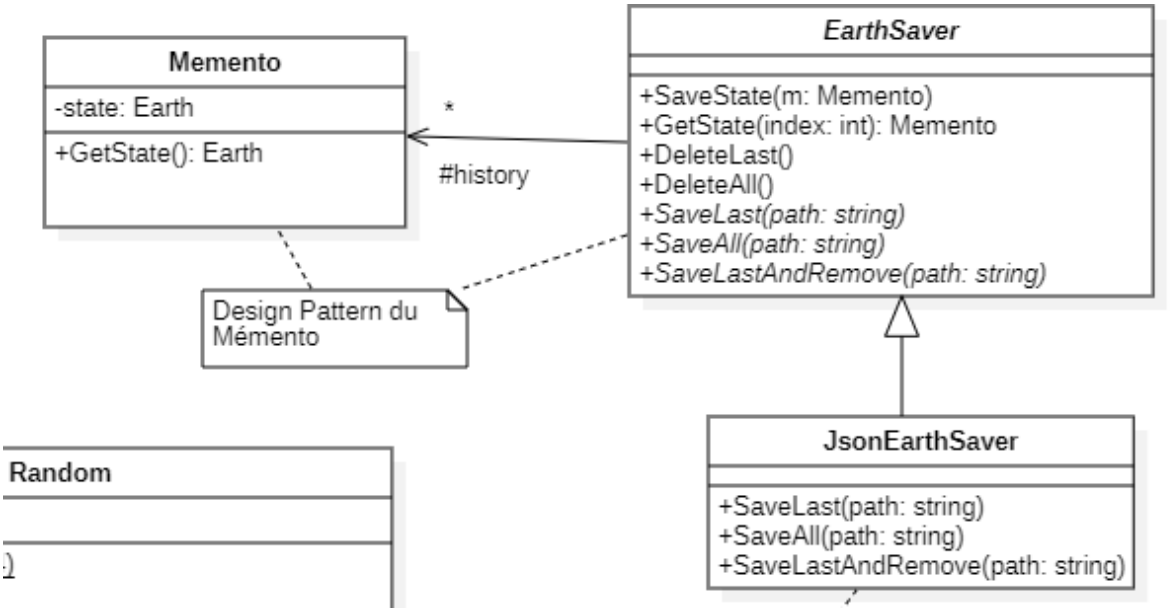


Figure 20 : Patron de conception Memento

Pour conserver plusieurs états et versions de l'objet Monde qui sera sérialisé, on les stocke dans un objet Memento. Le surveillant de cet état sera également un objet abstrait qui sera chargé de la sérialisation. Le sérialiseur garde avec lui une collection d'états et les sérialise grâce à une implémentation qui reprend le principe de la stratégie (une implémentation par technologie de sérialisation, on pourrait très bien imaginer un sérialiseur binaire). Pour l'instant, notre programme sauvegarde le Monde dans un Memento, le sérialise, l'écrit dans un fichier, et supprime le Memento. Mais on pourrait imaginer dans le futur le besoin suivant :

- Générer X jours sans les sérialiser, puis tout sérialiser et écrire dans un fichier, à la fin de la simulation. Par exemple, le Front pourrait demander de générer un historique pour 1000 jours et les sérialiser, attendre que l'utilisateur arrive au 999ème jour avant de sérialiser les autres, plutôt que la méthode actuelle qui sérialise sans interruption.

Patron du Singleton

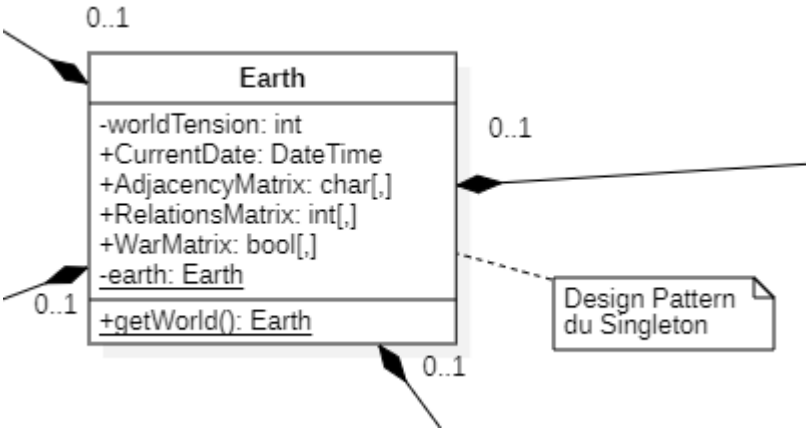


Figure 21 : Patron de conception Singleton

Ici très simplement, on ne souhaite conserver qu'une seule instance unique du monde que l'on observe. On passera ensuite uniquement par des références sur cet objet.

Front

Le Front quant à lui dispose d'une base fixe de territoires sous la forme de listes de coordonnées GPS. Ces territoires sont associés à des pays dans une autre base fixe mais facilement modifiable. On y retrouve aussi leurs capitales et leurs coordonnées. La partie dynamique est capable d'interpréter un monde sérialisé par la simulation en lisant le fichier JSON correspondant au jour voulu et en attribuant ses valeurs aux bons objets.


```
[
    {
        "outer": [
            [
                -155.005582,
                19.328882
            ],
            [
                -154.832073,
                19.455964
            ],
```


Figure 23 : Format JSON Territoires


```
{
    "id": 113,
    "name": "Porto Rico",
    "territories": [
        136,
        137
    ],
    "capital": "San Juan",
    "capcoord": [
        -66.1057,
        18.4655
    ]
},
```


Figure 22 : Format JSON Pays

Diagramme de Gantt

Contexte

Revenons maintenant sur le déroulé des travaux avec ce diagramme de Gantt. La version intégrale du diagramme fait avec GanttProject sera à retrouver avec ce rapport. Il est d'abord important de préciser certains points. Premièrement, la durée des tâches représente leur étalement dans le temps et pas réellement le nombre d'heures passées dessus. Par exemple, le développement de l'outil de génération de drapeaux ne constitue pas la tâche principale de ce projet. De plus, c'est un diagramme indicatif dans les grandes lignes mais cela ne veut pas dire que les tâches étaient strictement effectuées dans ces intervalles. De nombreuses retouches plus ou moins importantes ont été effectuées en dehors de ces dates.

Back

Pour commencer, on voit que l'avancée des travaux parait bien plus linéaire que dans le Pert prévisionnel. Cependant, des changements quasiment permanents sont intervenus sur les résultats des premières tâches. On retrouve donc un peu la parallélisation. On voit également que la date de fin est légèrement supérieure à la date prévue par le prévisionnel bien qu'elle rentre encore dans les délais prévus par le cahier des charges.

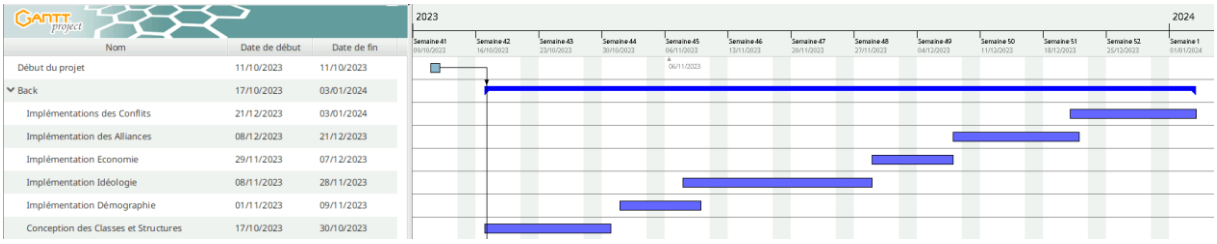
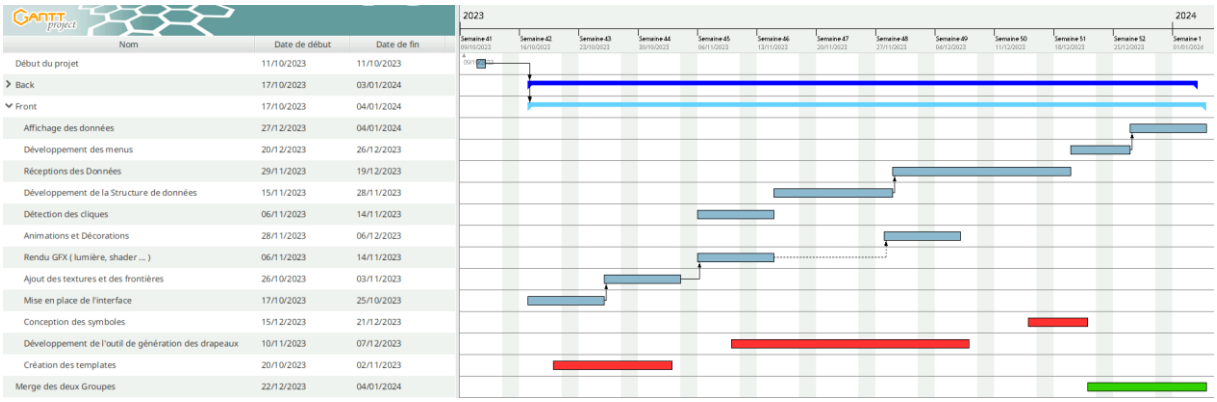


Figure 24 : Diagramme de Gantt Back-End

Front

A la manière du Back, on voit que la date du prévisionnel est dépassée. Cela s'explique par les nouveaux besoins apparus lors de la fusion des groupes ainsi que par l'apparition d'un système de génération de drapeaux qui n'était pas présent dans le prévisionnel du Front (il était en réalité présent dans le prévisionnel de la partie graphique du groupe finalement chargé du Back). Cette partie sur les drapeaux est différenciée en rouge dans le diagramme. Cette fusion apparait d'ailleurs dans le diagramme avec la tâche verte en bas.



Nom	Date de début	Date de fin
Début du projet	11/10/2023	11/10/2023
Back	17/10/2023	03/01/2024
Front	17/10/2023	04/01/2024
Affichage des données	27/12/2023	04/01/2024
Développement des menus	20/12/2023	26/12/2023
Réceptions des Données	29/11/2023	19/12/2023
Développement de la Structure de données	15/11/2023	28/11/2023
Détection des cliques	06/11/2023	14/11/2023
Animations et Décorations	28/11/2023	06/12/2023
Rendu GFX (lumière, shader ...)	06/11/2023	14/11/2023
Ajout des textures et des frontières	26/10/2023	03/11/2023
Mise en place de l'interface	17/10/2023	25/10/2023
Conception des symboles	15/12/2023	21/12/2023
Développement de l'outil de génération des drapeaux	10/11/2023	07/12/2023
Création des templates	20/10/2023	02/11/2023
Merge des deux Groupes	22/12/2023	04/01/2024

Figure 25 : Diagramme de Gantt Front-End

Quelques outils

Git

Evidemment, un des outils centraux pour la coordination de notre projet est Git. Cependant, il a fallu composer avec plusieurs projets sur différents gestionnaires de versions au moment de la fusion des groupes. En effet, les deux projets initiaux avaient respectivement un Gitlab pour le front et un Github pour le Back, celui si étant plus simple à utiliser avec Visual Studio. Finalement, les deux projets ont été compilés dans un seul projet Gitlab qui sera rendu. Cela permet en plus de la gestion de version d'accéder à des fonctionnalités d'intégration continue si besoin. C'est un outil d'autant plus indispensable qu'un groupe de 4 est plus difficile à coordonner et que la fragmentation du projet en plusieurs services le rend difficile à suivre sans système de commit.

Figure 26 : Logos Git

.NET

Le prochain outil est un framework du CSharp grâce auquel nous avons réalisé l'intégralité de notre Back. Cela permet un développement intuitif. De plus, .NET est connu pour sa polyvalence ce qui en fait un outil très apprécié pour le développement de jeux ou d'applications graphique. C'est notamment grâce à un des gros composant de .NET : WPF. Il permet de créer des applications graphiques avec un modèle de rendu vectoriel. C'est cet outil qui a permis de développer la partie graphique de débogage de la partie Back-End. Il dispose également de la librairie LiveCharts permettant l'affichage de graphiques dans l'application mentionnée précédemment.

Figure 27 : Logos .NET

Kivy

L'outil suivant est une bibliothèque peu connue de python conçue initialement pour développer des applications multi plateformes et en particulier des applications mobiles. Elle dispose de nombreuses extensions comme kivy3 qui est un wrapper expérimental sur la 3D d'openGL, kivy_garden.graph que nous utiliserons ici pour les graphiques ou encore kivyMD pour une collection un peu expérimentale également de Material Design. C'est un outil très complet que nous avons utilisé pour les menus grâce aux widgets de kivyMD comme pour l'affichage du globe en console même si cela nécessite un peu de travail, notamment en trigonométrie car la 3D de kivy3 est un peu expérimentale. La sphère ainsi que les shaders et plus globalement l'affichage 3D sont réalisés de manière tout à fait artisanale dans ce projet.

Figure 28 : Logos Kivy

CShapes

Cshapes est un outil simple mais qui nous a été extrêmement utile. Il s'agit d'une partie immense d'une base de données géographiques crée par des chercheurs de l'école polytechnique fédérale de Zurich. La base en elle-même est très complète mais la partie Cshapes regroupe la liste des entités territoriales et de leurs capitales mais surtout l'intégralité des frontières année par année entre 1886 et 2019 dans de nombreux formats.

Figure 29 : Cshapes Visualizer

Cela a été un des points clé au moment de la fusion des deux groupes puisque les deux simulations ne se passait pas à la même époque. Il a donc fallu adapter les tracés des frontières du Front aux pays présents à l'époque du Back.

JSON

Figure 30 : Logo JSON

Nous ne savons pas en quelle mesure un format de structure de donnée peut être considéré comme un outil mais une chose est sûre. Le JSON (ou JavaScript Object Notation) est partout dans notre projet. En effet, dans le Front, il est déjà présent pour la liste des territoires ainsi que pour le stockage des pays. Mais surtout, c'est ce format qui nous permet de faire le lien entre le Back et le Front. Les données de la simulation calculées selon la durée prévue sont sérialisées jour par jour et stockées dans des fichiers JSON qui seront lu par le Front. Un résultat similaire aurait pu être obtenu avec d'autres types de structures de données mais le JSON à de nombreux avantages ici. Il est facile à lire et donc à modifier ce qui est pratique pour une base de taille raisonnable comme celle des pays. De plus elle est également facilement parsable c'est évidemment indispensable pour l'utilisation que nous en faisons ici.

Escalation

Cet outil est un peu particulier puisque nous l'avons intégralement développé. Pour réaliser une simulation si complexe, il a fallu concevoir un outil de développement graphique plus explicite que des lignes de commandes. Voici le moteur ESCALATION, développé en WPF, qui tourne sur le projet C# .NET 6 central de la simulation. Ce projet utilise les nuggets LiveMap pour visualiser des graphiques et des camemberts en temps réel.

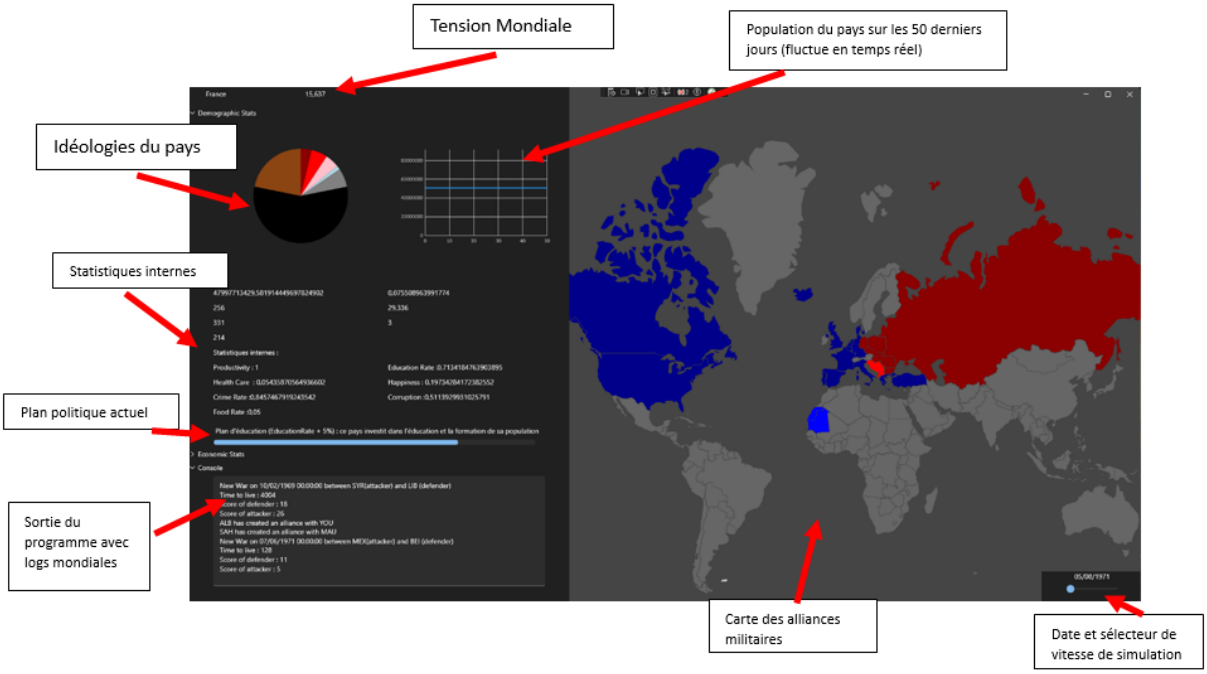


Figure 31 : Présentation Escalation

Il y a également un onglet pour le graphique de l'économie du pays sélectionné :

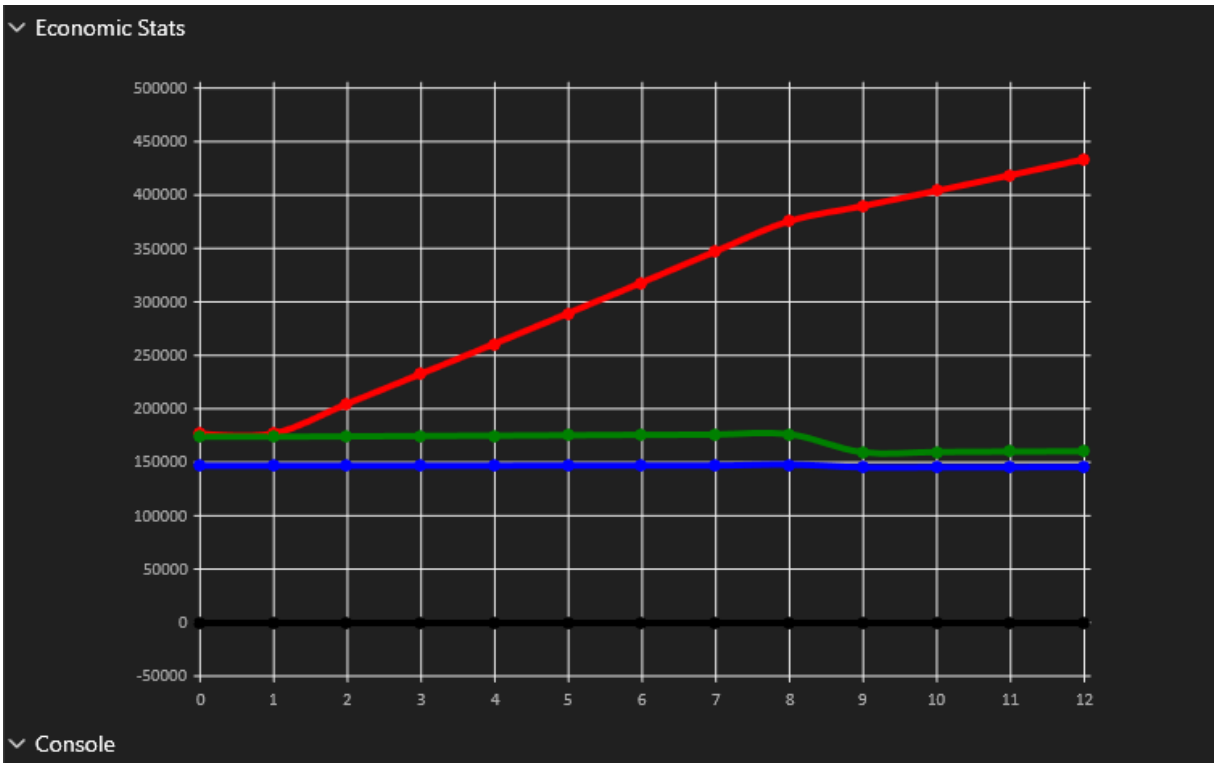


Figure 32 : Escalation Economie

Avec en bleu les dépenses, en vert les entrées, en noir la dette et en rouge la trésorerie.

En utilisant les touches F1, F2, F3, on peut alterner entre les différents modes de calque sur la carte principale :

Figure 34 : Escalation Calque Alliance

Figure 33 : Escalation Calque Guerre

Cet outil graphique s'est avéré être le compagnon de route idéal pour vérifier que toutes les interactions développées pour la simulation se déroulaient correctement, avant de les porter vers l'interface graphique en 3D sous python.

Pistes abandonnées ou archivées

Un projet de cette envergure donne forcément des idées difficilement réalisables dans le temps imparti. De plus le raccordement des deux groupes a conduit à la perte d'un certain nombre de tâches préalables. C'est pourquoi il nous semblait intéressant d'aborder rapidement ces éléments qui serviront peut-être un jour dans la suite du cercle de vie du projet.

Génération de chefs d'états

A la manière dont nous avons généré les drapeaux, nous aurions aimé crée un système de génération de chefs d'états fictifs selon le pays, la date et l'idéologie à partir d'un générateur d'image basé sur l'intelligence artificiel du type MidJourney ou Dall-E. Cependant, cette piste n'a pas abouti dans le temps imparti puisqu'elle n'est pas apparue comme prioritaire et qu'elle requiert un important travail de vérification et de modération afin que les images générées ne soient pas trop problématiques.

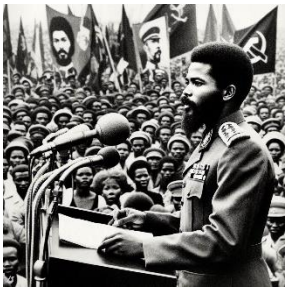


Figure 35 : Dall-E Chef d'état

Matrice d'alliance et d'entente

Voici un vestige de la fusion des deux groupes qui restera inutilisé dans la version actuelle du projet. Il s'agis de deux matrices carrées complètes dans lesquelles on retrouve pour chaque couple de pays un indice d'alliance représentant la coopération dans l'histoire entre ces deux pays ainsi que leur indice de proximité calculé à partir des distances haversiennes entre les deux pays ainsi que les langues et dialectes parlés et les devises et monnaies utilisées. Malheureusement, ces données étaient valides pour la période moderne et donc plus cohérentes dans le nouveau cadre du projet : la guerre froide.

Implémentation de graphismes supplémentaires

Il était question de rajouter des animations sur le globe lors de guerres ou d'évènements particuliers comme des catastrophes naturelles. Cependant, la charge de travail colossal ne nous a pas paru indispensable comptes tenus des bénéfices seulement cosmétiques que cela apporterait.

Sources

Atlas de la guerre froide *22/10/2023*

Un conflit Global et Multiforme par Sabine Dullin, Stanislas Jeannesson , Jérémie Tamiatto , Aurélie Boissière

Kivy *22/10/2023*

https://pypi.org/project/Kivy/

Pillow Imaging Library *22/10/2023*

https://pypi.org/project/pillow/

Live Charts *29/10/2023*

https://livecharts.dev/

Conseil Marketing *22/12/2023*

https://openai.com/chatgpt

Données des frontières *29/12/2023*

Schvitz, Guy, Seraina Rüegger, Luc Girardin, Lars-Erik Cederman, Nils Weidmann, and Kristian Skrede Gleditsch. 2022. "Mapping The International System, 1886-2017: The CShapes 2.0 Dataset." Journal of Conflict Resolution 66(1): 144–61.

Génération d'images *07/01/2024*

https://openai.com/dall-e-3

Images de Propagandes *07/01/2024*

https://dh.scu.edu/

Table des Illustrations