Membres :
KURDYK Louis (Info 6)
MORAND Paul-Emile (Jap-Info)
NGOY John (Jap-Info)
QU Julien (Info 5)

Groupe : Corona2

CORONA BOUNCE

Année 2020-2021

Encadrant : M.Jurski

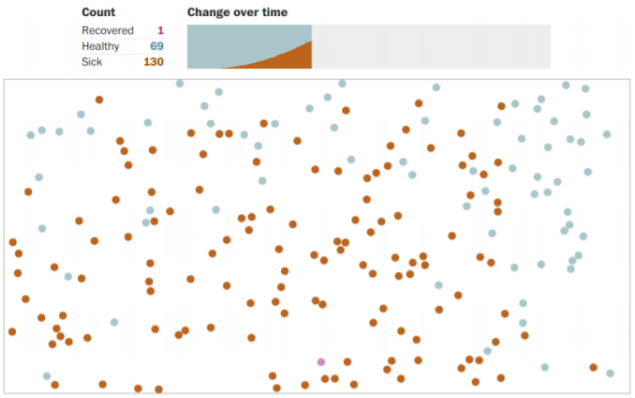
Sommaire :

I) Introduction :

Ce projet a pour but de pouvoir simuler la contamination d'une population par une maladie. Dans notre cas, la contamination s'effectue par contact direct entre un individu sain et un individu contaminé. Ce projet a également comme attribut d'appliquer un principe de collision entre différents objets. Dans notre cas, des individus. Ce projet avait pour but de montrer un utilisation de la programmation orientée objets. De plus, le sujet mis en avant s'accordait avec la situation sanitaire actuelle, même si cette simulation n'est pas aussi réaliste que celle-ci. Pour rendre cela possible, ce projet dispose de plusieurs grands éléments distincts : un panneau de commande où réside le centre de l'interaction entre l'utilisateur et le programme, une fenêtre où se déroule la simulation qu'on appellera board par la suite, et une fenêtre où l'on retrouvera un graphique représentant l'évolution de la contamination.

Pour débuter ce projet nous nous sommes inspirés d'un travail similaire réalisé par autrui est présenté par notre professeur (1).

Durant la construction de ce projet nous avons eu recours à la création de différentes classes pour créer les différents éléments de ce projet. L'architecture de ces classes vous sera présentée dans la partie II avec un diagramme puis une explication de celles-ci.



1. Objectif premier et modèle que nous devions atteindre

II) Diagramme de classe et de dépendances

Afin de présenter l'architecture du projet sous une forme de schématique, nous avons deux diagrammes, un représentant les différentes classes, et l'autre représentant les dépendances entre celles-ci, ce dernier étant un peu plus lourd. Pour les classes sont présentées textuellement dans la partie suivante et plus de détails sont disponibles dans la Javadoc.

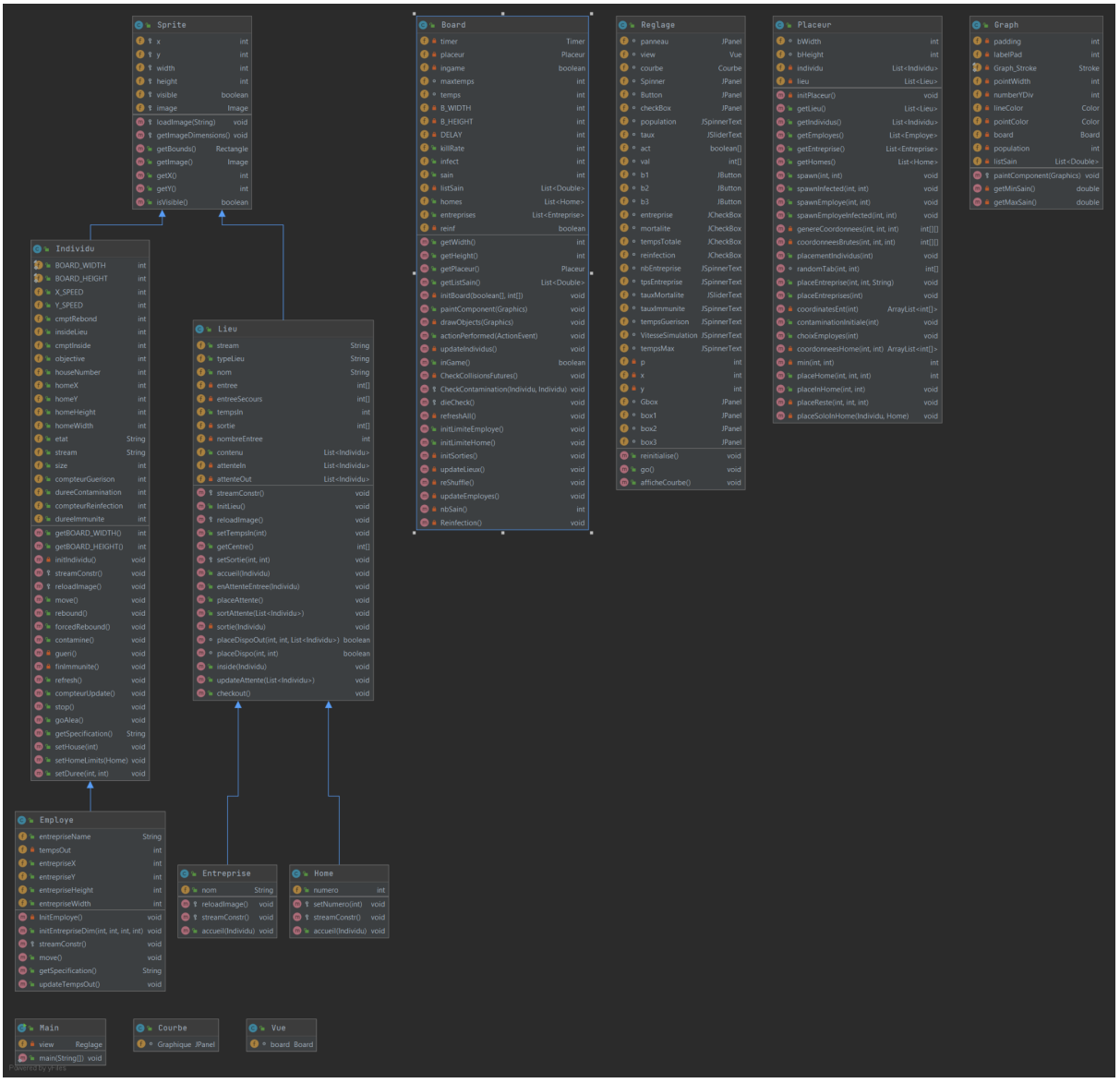


Diagramme des classes

Pour une version plus lisible :
https://drive.google.com/file/d/1g-aZfcOmAbomKj7iTNSYHDwieUA9Afrs/view?usp=sharing
Le second diagramme étant trop grand pour une page :
https://drive.google.com/file/d/1tQCQ_g4qVy4VErreo9DXvxxFnBgKXiU9/view?usp=sharing

III) Description des classes

Comme vous avez pu le constater sur les diagrammes de classes et de dépendances, les classes qui ont été programmées l'ont été pour répondre à des objectifs précis. Nous allons détailler quelles sont les principales fonctions et utilités de chaque classe.

A) Main

La classe Main de notre projet sert, comme son nom l'indique, de point de départ dans notre simulation, elle génère une fenêtre Réglages.

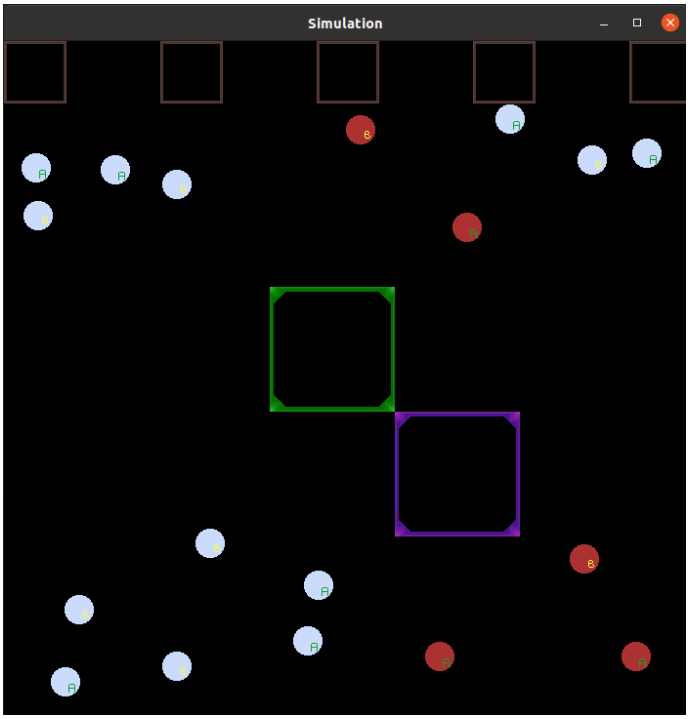
Cette classe contient:

En attribut un objet Réglage qui sera la première fenêtre qui apparaîtra à l'utilisateur. A partir de cette fenêtre il pourra utiliser l'ensemble du programme.

Un fonction main() qui va créer la simulation et l'afficher.

B) Vue

La classe Vue sert à créer une fenêtre contenant la simulation créée par Board(3). Cette fenêtre est créée à la suite de la demande de l'utilisateur lorsqu'il désire créer une simulation depuis une fenêtre créée par la classe Réglage.



3.Fenêtre construite par la classe Vue contenant une Board

La classe hérite de JFrame est contient:

En attribue une Board qui sera affiché pour l'utilisateur.

Un constructeur:

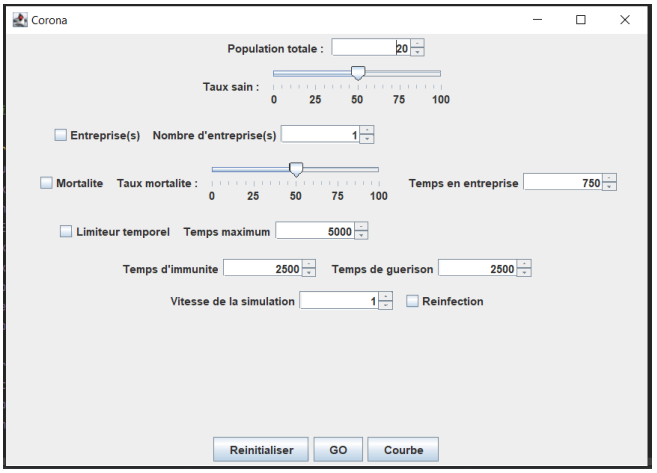
Ce constructeur va mettre en place les paramètres par défaut de la fenêtre: taille, position d'apparition, titre. De plus, ce constructeur prend en paramètre un tableau de

valeur booléenne et un tableau de valeur entière. Ces tableaux sont issus de la classe Réglage , paramétrés par l'utilisateur vont permettre de créer une board conforme à ces choix. Cette board sera ensuite ajouté à la fenêtre permettant son affichage.

C) Réglage

La classe Réglages, permet la création d'une fenêtre à partir de laquelle l'utilisateur va pouvoir paramétrer une simulation selon les fonctionnalités disponibles et désirées (4). Depuis cette même fenêtre, il pourra réinitialiser tous les paramètres sans relancer le programme, lancer une ou plusieurs simulations en simultanées avec des paramètres propres à chacune, et il pourra également afficher une courbe représentant l'évolution de la contamination de la simulation attachée.

4. Représentation du rendu la classe Réglage



Cette classe hérite de JFrame et contient:

En attribut on a :

Créations des checkbox , JPanel , JSpinner et JSlider en classe interne .

Obtention de ce rendu:

La fonction go permet de lancer la simulation de notre programme et cela affiche l'interface graphique .

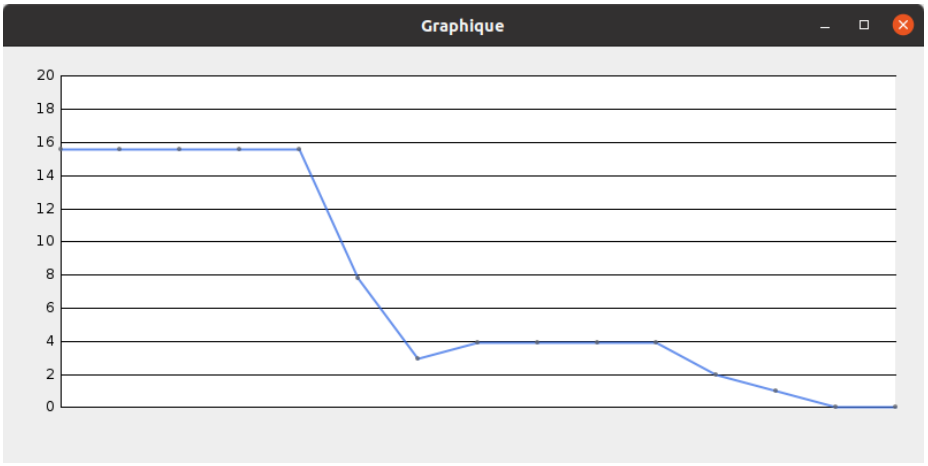
Ensuite la fonction reinitialise() remet les valeurs des checkbox et des Jspinner/Jslider par défaut .

Classe interne JSpinnerText , qui permet la construction de JSpinner avec un texte directement intégré lors de sa création.

Classe interne JSliderText, qui permet la construction de JSlider avec un texte directement intégré lors de sa création.

D) Courbe

La classe Courbe, permet de créer une fenêtre qui contient un Graph qui représente l'évolution de la contamination d'une simulation(il faut donc pour cela avoir une simulation en parallèle)(5). Cette fenêtre est créée à la demande de l'utilisateur depuis la fenêtre créée par la classe Réglages.



5.Fenêtre construit par la classe Courbe et contenant un graphique de la simulation

Cette classe hérite de JFrame et contient:
un JPanel qui va contenir un graphique.
Le constructeur met en place les paramètres par défauts de la fenêtre: taille, position d'apparition, titre, et un Layout.
De plus, le constructeur de cette classe prend un paramètre un objet de la classe Vue.
A partir de ce paramètre, il va pouvoir récupérer les informations nécessaires pour construire un objet Graph correspondant à la simulation courante créer par l'utilisateur. Cet objet Graph est également ajouté à la fenêtre

E) Graph

La classe Graph est destinée à créer le graphique qui représente le nombre d'individus contaminés durant la simulation. Elle hérite de la classe JPanel du JComponent. Dans un élément Graphics2D, on trace la courbe ainsi que les axes et les échelles. La courbe est tracée en fonction du temps, plus précisément toutes les 100 ticks du temps de la board. La classe Graph fonctionne en prenant la liste du nombre d'individus sains de la classe Placeur qu'elle convertit en une liste de flottants pour par la suite créer des points à cordonnée via la classe Point de java qui prend par la suite des x entier et y entier.

F) Board

La Board est l'élément central de la simulation en elle-même. L'interaction avec l'utilisateur y est limitée car tous les paramètres pouvant l'influencer ont été demandés à l'utilisateur avant la construction de la Board. Ces principaux intérêts sont la gestion des relations entre les Individus dans tous les cas de figures, la gestion des collisions entre les

Individus et les Lieux, la récupération des données de la simulation pour pouvoir les envoyer à la classe qui va s'occuper des les afficher à l'utilisateur. Le temps de la simulation utilise le timer de Java Swing, ainsi dès que nous parlerons de temps ou de durée, cela correspondra à des nombres de ticks de ce timer.

G) Sprite

La classe Sprite est la classe mère de tous les éléments affichables à l'écran. Elle permet à ces éléments d'avoir des coordonnées, une image qui leur est associée. Pour la gestion des collisions qui tient une part importante de notre projet, chaque Sprite est lié à un rectangle dont ses dimensions sont données par celle de l'image correspondante. Par la suite, chaque classe fille a son propre constructeur de chemin d'accès pour aller chercher l'image correspondante dans le dossier ressources.

H) Individu

La classe Individu est une classe fille de la classe Sprite précédemment présentée. Chaque Individu de notre simulation a pour objectif de correspondre à une personne. Ces principales méthodes sont move(), permettant de gérer les déplacements de l'Individu en fonction sa position ; rebound(), faisant rebondir les Individus si besoin est ; contamine(), qui s'occupe faire passer les Individus de l'etat "Neutral" à "Infected" ; ainsi que les méthodes qui permettent regarder depuis combien de temps l'Individu est contaminé ou guéri afin de le faire changer d'état selon l'ordre "Neutral" -> "Infected" -> "Recovered" -> "Neutral".

I) Employe

La classe Employe est une classe fille de la classe Individu précédente. Elle hérite de toutes les méthodes d'Individu en modifiant la méthode move() pour qu'elle assure des bons déplacements en Entreprise. Chaque Employé est lié à un Lieu Entreprise (si l'utilisateur à demander de tels Lieux) ce qui lui permet de rentrer à l'intérieur où la contamination s'y effectue normalement.

J) Lieu

La classe Lieu est une classe fille de la classe Sprite (cf. la sous partie G). Les Lieux sont des parties closes de la Board où les Individus se déplacent à taille réduite mais conservent leur vitesse, cela permet ainsi d'augmenter fortement le nombre de collisions sans pour autant les rendre obligatoires, simulant ainsi l'effet des lieux clos sur les contaminations réelles. Chaque Individu doit être compatible avec le Lieu avant de pouvoir y

pénétrer, sinon il rebondit simplement dessus. Pour gérer l'accueil des Individus au sein des Lieux, la principale méthode est inside() qui va, selon la disponibilité de l'entrée du Lieu (un champ défini comme le centre de celui-ci) va : ou bien réduire la taille de l'Individu, le replacer, recharger sa nouvelle image et ses nouvelles dimensions et l'ajouter à la liste contenu du Lieu ; ou bien va le mettre en liste d'attente pour rentrer quand la place sera disponible. Les sorties s'effectuent de manière similaire avec une liste d'attente pour si la place définie comme la sortie du Lieu (définie en fonction de la place du Lieu sur la Board pour s'assurer de son bon fonctionnement). Tous les Individus passent une certaine durée au minimum dans le Lieu appelée tempsIn, ils peuvent donc entamer le processus de sortie une fois cette durée écoulée.

K) Entreprise

La classe Entreprise est une classe fille de la classe précédente Lieu. Elle hérite de toutes les méthodes et adapte l'accueil pour l'accueillir que des Employés dont le champ nomEntreprise coïncide avec le champ nom de l'Entreprise.

L) Home

La classe Home est aussi une classe fille de la classe Lieu. Elle a pour vocation de représenter les maisons des Individus, ce qui en fait le point de départ de chaque Individu de la simulation. Les deux sont alors liés pour que la méthode d'accueil soit modifiée pour n'accepter que les Individus qui sont apparus en premier dans cette maison.

M) Placeur

La classe Placeur est une classe qui fonctionne comme initialiseur de la Board. Il va dans un premier temps faire apparaître le nombre d'Individus souhaiter à des places données puis il fait apparaître et initialise les Lieux. Une fois que tout cela est effectué, il va pouvoir placer les Individus dans les maisons (avec au maximum quatre par maison). Il possède la liste de tous les Individus et Lieux, variables régulièrement transmises à la Board pour qu'elle puisse s'occuper des interactions.

IV) Fonctionnalités et prise en main

A) Installation

Tous les fichiers .java sont présents dans le dossier com.company, il s'agit donc de les compiler puis d'exécuter le résultat.

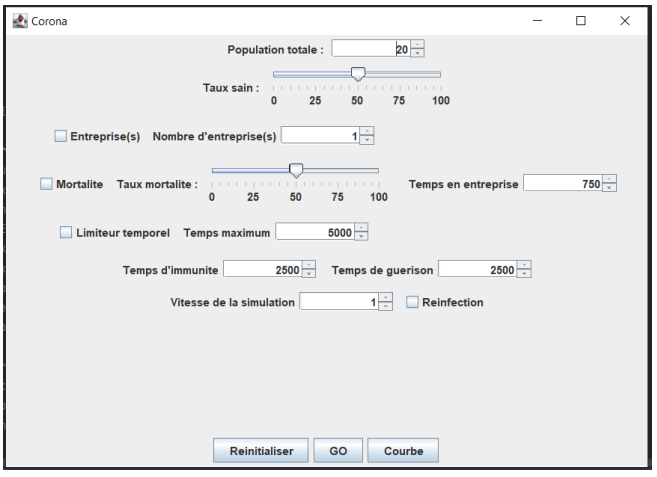
Ou d'utiliser le .jar qui se situe dans out/artifacts/Corona2Bounce_jar.

B) Utilisation

La classe Réglage affiche le panneau de contrôle de la simulation.
Il y a 2 JSliders : taux sain qui indique le pourcentage de la population saine et le taux mortalité qui définit le taux de mortalité moyen du virus.
Nous avons 3 checkboxes qui dès qu'elles sont cochées, les sliders et spinners correspondant ont un effet sur la simulation : Entreprise(s) avec le slider Nombre d'entreprise(s), cela donne le nombre d'entreprises dans la simulation, ensuite on a Mortalité avec le spinner taux mortalité expliqué ci-dessus, et enfin Limiteur temporel avec le slider Temps maximum qui décide du temps maximal de simulation si une limite est voulue.

Ensuite il reste les JSpinner :

- Population totale : le nombre de personnes dans la simulation.
- Temps en entreprise : le temps que les personnes restent dans les entreprises, puis ressortent après le temps indiqué.
- Temps immunité : les individus restent immunisés au temps mis dans le spinner.
- Temps de guérison : même chose que Temps immunité, sauf qu'il est guéri au lieu d'être immunisé
- Vitesse de la simulation : Accélération de la vitesse de la simulation, le délai du timer Swing est divisé par la valeur indiquée.



Enfin, pour lancer la simulation avec les paramètres souhaités, reste à appuyer sur le JButton Go pour lancer la simulation et Courbe, de manière optionnelle, si l'on veut afficher la courbe représentant les Individus contaminés (en bleu).

V) Etapes de la programmation

La programmation de ce projet s'est faite en plusieurs paliers. La première étape était l'étude d'un tutoriel qui visait à créer un jeu de tir dans un vaisseau spatial type Space Invader mais sur la largeur de la fenêtre au lieu de la hauteur. Nous avons donc pu nous aider des classes qu'implémentait ce tutoriel pour concevoir les nôtres, en particulier Sprite, Individu et Board.

A partir de ce moment nous avons pu nous éloigner de ce tutoriel afin de faire coller nos classes à ce que nous voulions pour notre simulation. Nous avons donc donné des mouvements en diagonale aux Individus puis avons implémenté les rebonds sur le bord de la Board dans un premier temps avant de nous occuper des collisions entre Individus. L'étape suivante de notre projet fut une nouvelle version de l'apparition des Individus réalisées jusque-là par une action de l'utilisateur. Après cela il nous avons programmé les différents états de la maladie pour les Individus et la transmission de celle-ci tout d'abord marquée par un simple affichage textuel puis par des modifications d'image comme dans la version finale.

Une fois que cette base de notre projet a été faite, nous avons pu commencer à nous consacrer à des ajouts hors du premier modèle (1) que nous avions. Nous avons pensé à ajouter des lieux pour que notre Board ressemble un peu plus à une ville et à donner un peu de sens à l'existence de nos Individus (certes qui se contentent d'aller au travail, c'est un peu triste). Pour cela nous avons commencé par travailler sur une classe Lieu car nous n'avions pas encore une idée particulière de Lieu. Nous avons ensuite choisi les Entreprises qui nous ont donc incités à faire la classe Employé pour conserver une certaine logique. Une pour implémenter une seconde classe fille à Lieu, les Homes ont été implémentés permettant une meilleure apparition des Individus sur la Board. Nous avions dans un premier temps pensé à rendre les maisons optionnelles mais elles remplissent un bon rôle dans l'initialisation, nous les avons donc conserver au démarrage de toutes les simulations.

Une fois que tout ce cadre a été placé, il nous a fallu s'assurer que tout fonctionnait bien ensemble, ce qui nous a permis de retravailler plusieurs fois la classe Lieu, plus problématique que les autres.

En ce qui concerne l'interface, si dans un premier temps, elle s'est contenté d'afficher tous les éléments de la simulation au sein de la même fenêtre, elle a par la suite changer. De plus, ces premières versions de l'interface ne permettait que d'avoir des fonctionnalités simples de la simulation(indiquer population, lancer simulation, l'arrêter, réinitialiser les valeurs). Elle a par la suite été étayer avec l'avancement du projet. L'interface a également changé, puisque d'une simple fenêtre contenant toute la simulation, nous sommes passés à trois fenêtres distinctes rendant l'ergonomie plus agréable et claire pour l'utilisateur. permettant aussi le lancement en simultanés de plusieurs simulations, pouvant être distincte les unes des autres.

Suite à ça dans une des fenêtre la principale qui permet de lancer la simulation , nous avons ajouté des checkbox, des Jspinner et Jslider pour augmenter les choix possibles sur la simulation ,indiquer le nombre d'individus saints, de morts , le temps qu'ils guérissent du virus , l'immunité au virus et l'accélérer.

VI) Compétences acquises

A) Julien

Ce projet de programmation m'a permis d'améliorer mes compétences dans plusieurs domaines d'informatique : les principales sont git/gauffre , l'utilisation de JavaSwing , les checkBox, les JSpinner/JSlider . En début de semestre sur une mauvaise communication , j'ai travaillé sur les mêmes classes ce qui m'a fait perdre beaucoup de temps, du coup la communication en équipe s'est améliorée de mon côté aussi.Le semestre précédent la matière prépro2 , le git n'était pas maitrisé et grâce à ce projet j'ai pu mieux comprendre et utiliser git sans causer de bug sur les commit ou commit des mauvaises versions contenant des bugs.

La classe Réglage m'a entraîné sur les thèmes informatiques que j'ai cité au-dessus. Je remercie notre professeur du 4e semestre , M.Jurski de nous avoir encadré , nous aider en donnant quelques pistes au début du projet , des conseils sur nos difficultés rencontrées.

B) Paul-Emile

Ce projet de programmation m'a permis d'améliorer mes compétences, ainsi que d'en acquérir de nouvelles et ce dans différents domaines de l'informatique. L'une des premières est la connaissance et l'utilisation de git/gauffre, qui m'ont fait connaître une nouvelle méthode pour aborder et effectuer un projet informatique en groupe.

Ensuite, il y a eu le renforcement de mes compétences sur la programmation orientée objet. En particulier dans le domaine de l'interface graphique. Si au début du projet, mes premiers travaux étaient assez grossier, puisque je me contentait d'afficher la simulation en une unique fenêtre, j'ai par la suite appris et fait en sorte de diversifier cette aspect du projet pour le rendre plus compréhensible, facile d'accès et ergonomique pour l'utilisateur. J'ai également aidé mes camarades en reliant leur différents travaux pour que ceux-ci montrent leur réelle utilité dans le projet.

Ce projet m'a donc permis d'apprendre de nouvelles choses dans le domaine de l'informatique, ainsi que le confort dans les compétences que je connaissais auparavant. Je remercie notre professeur encadrant M.Jurski de nous accompagner durant la réalisation de ce projet. Et je remercie également mes camarades avec qui j'ai pu réaliser ce projet.

C) John

Le projet, pour ma part, m'a permis d'acquérir un nouveau moyen de travail en équipe qui est l'outil git, avec beaucoup de mal à appréhender au début, l'outil est très pratique une fois pris en main. Pendant que je n'arrivais pas à relier git et mon espace de travail, j'ai travaillé de mon côté sans avoir donc vision sur le projet du groupe. Lorsque j'ai dû relier mon travail au projet du groupe cela à donc créé beaucoup de problème et ralenti le rythme du projet. Le projet m'a donc fait prendre conscience de l'importance d'être sur la même longueur d'onde avec tous les membres.
Techniquement, le projet m'a apporté beaucoup de bons points et une meilleure compréhension des interfaces graphiques de Java. Il m'a également rappelé l'importance des mathématiques en programmation, étant en japonais-informatique, j'ai complètement délaissé les mathématiques même fondamentales.

Globalement le projet m'a donc montré des points négatifs et positifs mais ces points négatifs me semblent importants pour m'améliorer.

D) Louis

Pour ma part, ce projet de programmation m'a permis de développer un nouveau champ de compétence : la programmation orientée objet. En effet, venant de licence de mathématiques après une réorientation entre le S3 et le S4, je n'ai pas suivi les cours de POO du semestre dernier. Il m'a donc fallu apprendre les concepts de base de ce domaine pendant les premières semaines de mon travail. Cela a donc donné place à du code surement plein de balbutiements en particulier sur les premières versions de certaines classes. Le fait d'avoir travailler sur les classes Individus, Board, Lieu, Entreprise, Employé, Sprite, Placeur et Home m'a permis de comprendre d'une meilleure façon la pratique de la programmation orientée objet, et je remercie notre professeur encadrant, M.Jurski, pour sa patience et ses conseils.

En ce qui concerne Git, j'avais déjà eu l'occasion de travailler avec cet outil mais sans en exploiter le plein potentiel. J'ai donc pu apprendre à me servir des branches, un outil très puissant pour le travail de groupe.

Enfin, pour pouvoir faire un rendu propre, j'ai pu m'intéresser aux diagrammes de classes et de dépendance pour pouvoir présenter la structure du projet. Il m'a également paru important de faire une Javadoc (retrouvable dans le dossier Git) pour présenter plus précisément le travail de chaque classe et méthode sans être trop lourd dans ce dossier.

VII) Conclusion

Ce projet, partant d'un modèle existant, nous a permis de créer notre propre vision du sujet. Nous sommes partis sur des bases de code préexistantes que nous avons par la suite modifiées, voire construites. Nous avons commencé en nous intéressant au mouvement et à la collision de différents éléments. Puis nous avons voulu permettre à l'utilisateur de pouvoir facilement modifier les paramètres sans passer par la modification directe du code. Tout en continuant d'améliorer la simulation en elle-même, nous avons également affiné l'interface pour qu'elle soit plus intuitive. Nous avons apporté de nouveaux éléments permettant d'effectuer des simulations avec des cas plus divers que le simple qu'un individu infecté puisse contaminer d'autres individus. Nous avons fait en sorte que ces mêmes individus aient la capacité de se rendre dans des lieux plus contraignants en termes de mouvement que la board dans son entièreté avec le principe des entreprises. Nous avons également fait en sorte que la maladie ait un impact plus important sur les individus que simplement les contaminés en la rendant mortelle si l'utilisateur le désirait . En parallèle de cela, nous avons ajusté l'interface pour que l'utilisateur puisse permettre cela ainsi que les autres options ajoutées a posteriori sans avoir à rentrer dans le code en lui-même. Nous avons également permis à l'utilisateur d'observer un graphique représentant l'évolution d'une simulation en temps réel.

Ce projet nous a donc permis d'utiliser nos compétences acquises durant notre cursus, les renforcer, mais aussi d'en obtenir de nouvelles. Et cela en nous appuyant sur des modèles préexistants ainsi que sur les conseils de nos encadrants.

Après réflexion, si nous avions eu plus de temps, une liste de fonctionnalités supplémentaires nous est venue à l'esprit :

- Améliorer l’aspect de la simulation graphiquement
- Améliorer l’aspect “vivant” de la simulation en ajoutant des transports en commun
- Avoir un vrai pathfinding pour que les Individus puissent aller directement de chez eux à l’entreprise et inversement
- Un cycle jour nuit qui correspond au pathfinding
- Trouver le moyen de faire en sorte que les sliders correspondant à des checkboxes n’apparaissent que si les checkboxes ont été cochées.