Épidém.io

Rendu final

Table des matières

Présentation du projet

Objectif

L'objectif du projet Épidém.io est de réaliser une simulation de propagation d'une épidémie. L'utilisateur peut régler différents paramètres pour adapter la simulation à ses critères. L'implantation a été réalisée en Python et des bibliothèques associées. L'utilisateur voit la maladie se répandre au fur et à mesure. Il y a des itérations successives de la propagation de l'épidémie que nous affichons sous forme de schéma.
Un point absolument central pour nous lors du développement a été de réaliser un logiciel complet, fonctionnel et pensé pour une vraie utilisation. L'interface est claire, documentée avec notamment des bulles d'information, et compte tous les outils nécessaires pour une utilisation intuitive et claire. Pour la simulation en elle-même, de nombreuses recherches ont été faites pour avoir une simulation réaliste contenant de l'aléatoire, avec une gestion des naissances, de la relance de la maladie, mais aussi des déplacements des personnes pour simuler des déplacements les plus « humains » possible.

Simulation

La partie principale de l'écran est la fenêtre d'affichage de la simulation. Les personnes sont placées dans un espace en 2D et sont représentées par un état. L'attribut état est obligatoire et exclusif : il y a quatre états, une personne en a nécessairement exactement un parmi ces choix. Les quatre états sont liés à une couleur pour la représentation visuelle :
- l'état sain, représenté en vert : la personne n'est pas malade ;
- l'état infecté, représenté en orange : la personne est infectée, mais n'est pas morte ;
- l'état mort, représenté en rouge : la personne est morte à cause de la maladie ;
- l'état immunisé, représenté en bleu : la personne ne peut pas être contaminée.

La simulation fonctionne donc avec des itérations, et nous mettons à jour toutes les x secondes l'affichage, faisant une forme d'animation.

L'utilisateur peut régler différents éléments grâce à un menu dans l'interface. Cela lui permet de simuler une épidémie qui correspond le plus possible à celle qu'il cherche à recréer. Par exemple, l'épidémie du COVID-19 ne peut pas être simulée comme celle de la peste au XIVe siècle. Parmi les indicateurs que l'utilisateur pourra donc régler, on compte trois types :

a. les indicateurs liés à la population :
   - le nombre de personnes totales ;
   - le pourcentage de personnes immunodéprimées : par exemple, au Japon 30.2% de la population est âgée de 65 ans ou plus, alors que c'est 1.7% de la population en Ouganda, c'est donc un facteur intéressant ;
   - le nombre de médecins : ce sont des personnes qui peuvent augmenter les chances de survie des gens autour (ça ne sera pas du 100%, mais on pourrait dire que les personnes proches de ce point à une certaine distance verraient leurs chances de survie multipliées par deux) ;
b. les indicateurs liés à la maladie :
   - le taux de létalité (ou pourcentage de mortalité) : la simulation ne sera pas la même si on représente la tuberculose, qui a un taux de létalité de 43%, et la grippe A, qui a un taux de létalité inférieur à 0.1% ;
   - la distance d'infection : la distance qu'il faut entre deux personnes pour que la personne infectée contamine l'autre ;

- le pourcentage de risque de transmission : ce n'est pas la même chose si une maladie a seulement 10% de risques d'être transmise, ou 95% ;

- l'immunité après guérison : il arrive que certaines maladies ne puissent plus s'attraper après qu'une personne l'ai attrapée et soit guérie, on peut donc mettre un indicateur qui voudra "oui" ou "non" pour l'obtention d'une immunité contre cette maladie une fois guéri ;

- le temps de guérison : certaines maladies ne nous affectent que quelques jours, comme la grippe, mais d'autres restent à vie par exemple, comme le VIH ;

c. les indicateurs liés à la simulation :

- le nombre de personnes infectées au début : le nombre de personnes qui seront les "patients 0" ;

- la vitesse de déplacement des points

Il y a également deux éléments algorithmiques invisibles, la relance d'épidémie et la propagation de l'épidémie. Pour la relance, si personne n'est malade depuis un certain nombre d'itérations, on relance la maladie sur quelques personnes. Par rapport à la propagation, on utilise un système de grille, elle sert à réduire la complexité de l'algorithme, car au lieu de regarder pour chaque infecté tous les voisins sains, on regarde seulement ceux qui sont à distance d'infection. Pour faire cela, on met en place une grille avec des carreaux carrés, dont la taille de chaque côté est la distance d'infection. Cela assure qu'une personne malade ne puisse pas infecter quelqu'un plus loin que sa case et les 8 bordant sa case. On fera ainsi les calculs de distance pour vérifier qu'une personne saine est à portée entre les coordonnées de la personne malade et les coordonnées des personnes saines dans les 9 cases mentionnées, et pas plus loin.

La simulation n'est pas prévue pour un nombre de personnes supérieur à 200. Comme les itérations sont très rapides, et que les étapes les plus longues sont le calcul des nouvelles positions ainsi que la propagation de l'épidémie, si la population est très grande, les calculs mettront plus de temps que le délai prévu entre chaque itération. Il y aura ainsi un ralentissement de la simulation qui sera également visible. Comme il y a des naissances, qui peuvent aller jusqu'à 5% de la population, si l'utilisateur génère une simulation pour une population de 200 personnes (le nombre maximum que l'on peut entrer manuellement) avec une maladie avec un taux de mortalité de 0.00%, la population va s'accroître rapidement. Nous avons choisi de ne pas interdire cela en capant au sein même de la simulation le nombre de personnes dans la population, mais la simulation va fortement ralentir au fur et à mesure que la population augmente. Nous indiquons ici à plusieurs endroits dans l'application que nous ne recommandons pas les simulations de plus de 200 personnes, et nous laissons aux utilisateurs la responsabilité de respecter ces guidelines ou non.

Travail prévu

Implémentation de l'interface graphique

- création de la fenêtre principale
- mise en place d'une toile pour afficher la simulation en 2D
- ajout d'un menu de réglage des paramètres :
    1. champs numériques pour les valeurs entières (la taille de la population, le nombre de médecins, rayon de contagion…) ;
    2. cases à cocher pour les options booléennes (immunité après guérison, présence ou non des médecins,…) ;

3. curseurs pour les valeurs continues (risque de transmission, taux de létalité, distance d'infection, vitesse de déplacement, pourcentage de personnes immunodéprimées…)

- ajout des boutons démarrer, mettre en pause et réinitialiser
- mise à jour de la toile à chaque itération pour animer les déplacements et les changements d'état (couleur selon l'état)
- ajout d'un bouton pour télécharger les statistiques
- gestion de la fermeture propre du programme

Implémentation du système de gestion des statistiques

- création de la classe statistiques pour enregistrer les données à chaque itération
- définition d'une structure interne pour stocker le nombre total de personnes totales et contaminées
- ajout d'une méthode appelée à chaque mise à jour de la population pour stocker les informations
- génération d'un graphique matplotlib affichant l'évolution de la contamination au fil du temps
- ajout de la possibilité d'exporter les données

Implémentation de la simulation

- création des classes principales personne, population et épidémie
- implémentation d'un système de calcul des distances entre individus pour déterminer les contaminations
- mise en place d'un pas de temps entre chaque itération où chaque individu :
    1. prend une nouvelle position ;
    2. se déplace selon sa vitesse ;
    3. vérifie les contacts à proximité pour une éventuelle contamination ;
    4. met à jour son état (guérison, mort, immunisation)
- test et comparaison de différentes approches pour le déplacement des individus, type algorithmes de simulation de déplacement de foule (exemple d'approches possibles : déplacement totalement aléatoire, déplacement avec inertie ou direction persistante, gestion des collisions pour éviter la superposition des points, déplacements inspirés de Boids…)
- choix et implémentation de l'approche la plus stable et visuellement claire pour la simulation
- synchronisation de la simulation avec l'affichage
- mise en place de comportements spécifiques (par exemple : les médecins augmentent les chances de guérison autour d'eux, les immunodéprimés ont un risque de mortalité plus élevé…)
- gestion de la fin de la simulation (tous guéris, morts, aucune infection active, ou au contraire régénération de la population avec des enfants : à déterminer lors du développement)

- gestion de l'approche concernant le temps entre les itérations (temps que peux déterminer l'utilisateur, temps fixe, temps variable en fonctions d'indices comme le nombre d'individus…)
- détermination des limites de l'algorithme (par exemple : pas plus de x personnes car risques trop élevés que la simulation ne supporte pas une quantité trop élevée)

Tests de l'application

- vérification des données entrées par l'utilisateur dans les champs à remplir
- vérification des effets de bord
- surveillance des bugs algorithmiques
- tests de performance (par exemple : limitation de la population à simuler, mise en place d'une limite minimum pour le temps entre chaque itération…)
- vérification de l'intégrité des données récupérées entre chaque itération

Rédaction de la documentation et du support de présentation

- rédaction de la documentation technique des classes et méthodes
- ajout de commentaires dans les parties principales du code
- rédaction d'une notice utilisateur décrivant le déroulement de la simulation et les options disponibles
- création d'un schéma UML final mis à jour selon les choix d'implémentation effectués
- rédaction d'un support de présentation
- réalisation de tests finaux et correction des éventuels bugs avant la présentation et la livraison du projet

Travail réalisé

Toutes les tâches listées ci-dessus ont été réalisées, soit l'ensemble des tâches prévues dans le cahier des charges. Voici un détail sprint par sprint des tâches réalisées :

- Sprint 2 :
    1. Ajout des états : sain, infecté, immunisé, mort
    2. Ajout des paramètres simples : nombre initial d'infectés, probabilité de transmission…
    3. Mise en place de la propagation (première étape sans déplacement)
    4. Gestion de la mortalité, immunité après guérison (oui/non), immunodéprimés…
    5. Premier déplacement aléatoire temporaire simple pour chaque personne (mouvement limité par les bords) pour tester pour la contagion
    6. Organisation des données collectées par itération pour le module statistiques
    7. Ajout de la visualisation dans l'interface de la simulation
    8. Ajout de la partie extraction et visualisation des données dans l'interface
    9. Liaison entre le modèle et l'interface : toile affichant les points colorés selon l'état
    10. Rafraîchissement automatique de la toile toutes les x seconde(s) (approche à déterminer) avec mise à jour des paramètres
    11. Boutons fonctionnels pour lancer, mettre en pause et réinitialiser

12. Info bulle pour que les utilisateurs puissent comprendre exactement à quoi correspondent les paramètres

- Sprint 3 :
    1. Tests de plusieurs modèles de déplacement (par exemple : aléatoire simple, inertiel, avec évitement, Boids ajusté aux foules humaines…)
    2. Comparaison leurs impacts sur la propagation, leur fidélité par rapport au phénomène à représenter et la stabilité visuelle
    3. Implémentation de l'approche choisie
    4. Mise à jour de la documentation
    5. Mise en place des tests
    6. Documenter les différentes classes du package view
    7. Ajouter des tests de non régression
    8. Améliorer l'interface et corriger les bugs éventuels
    9. Adapter la toile représentant la simulation en fonction de la taille des personnes
    10. Ajout de cercles rouges autour des points pour montrer la distance d'infection entre deux points lorsque l'utilisateur règle la distance d'infection pour qu'il puisse bien se représenter la distance
- Sprint 4 :
    1. Ajout des actions des médecins sur leurs voisins et leurs paramètres associés (pourcentage de survie augmenté, distance nécessaire pour soigner les personnes…)
    2. Graphiques Matplotlib : schéma avec les itérations en abscisse et le pourcentage d'infectés (morts inclus) en ordonnée…
    3. Téléchargement du graphique réalisé
    4. Amélioration visuelle en fonction de ce qui est nécessaire : taille des points, contraste, lisibilité, disposition générale…
    5. Implémentation visuelle des médecins
- Sprint 5 :
    1. Tests du modèle complet : cohérence propagation et mouvements
    2. Ajout de relance de l'épidémie
    3. Vérification cohérence des statistiques enregistrées
    4. Rédaction de la documentation technique
    5. Diagramme de classes final à jour
    6. Tests de l'interface : réactivité des boutons, gestion des états (pause, reprise, réinitialisation)
    7. Correction des bugs potentiels
    8. Rédaction de la documentation utilisateur
- Sprint 6 :
    1. Nettoyage et commentaires du code côté modèle
    2. Rédaction de la partie modèle du rapport
    3. Correction des bugs
    4. Peaufinage du rendu
    5. Nettoyage et commentaires du code côté interface
    6. Rédaction de la partie interface du rapport

Diagramme de Gantt

Diagramme initial

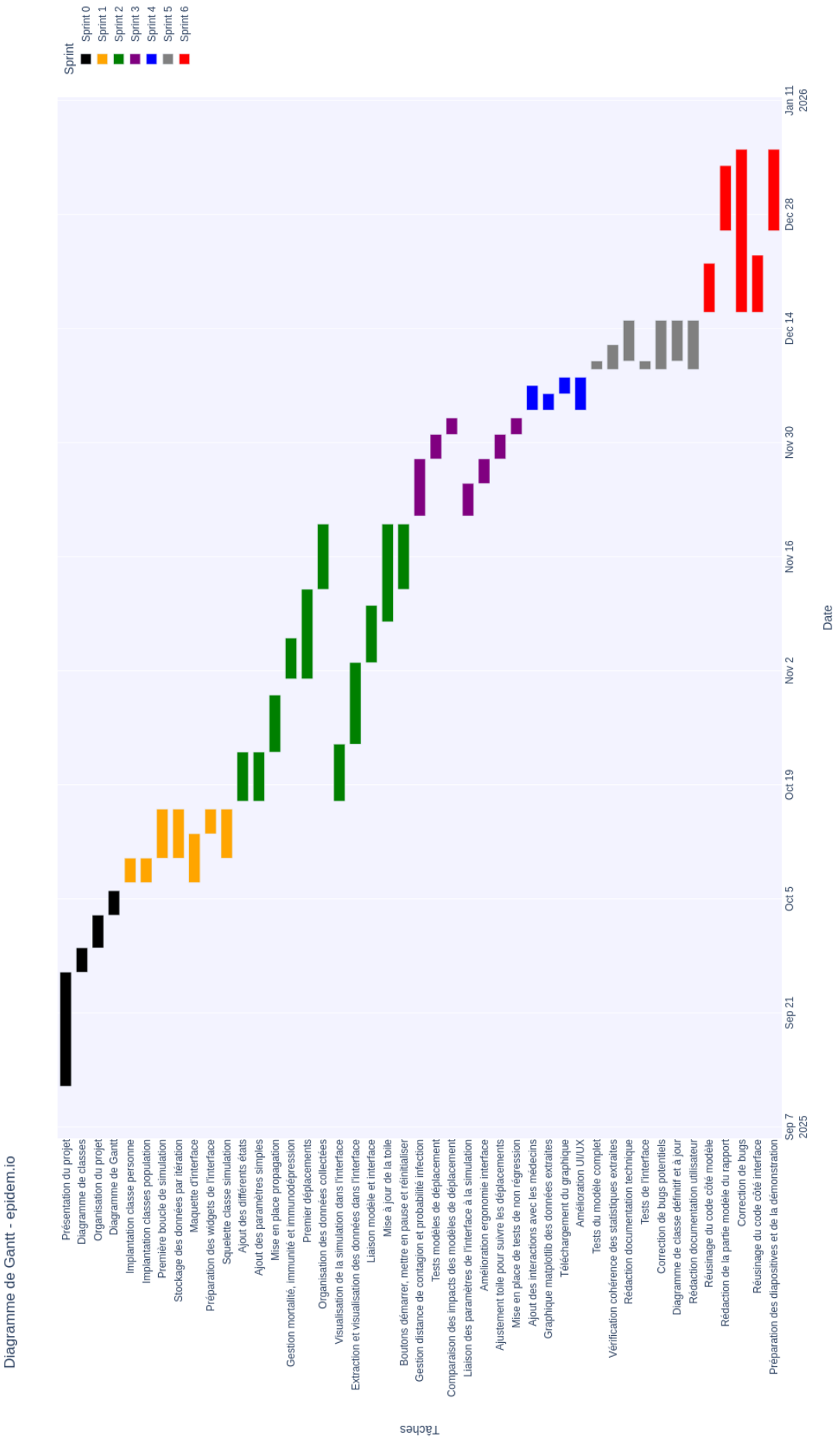
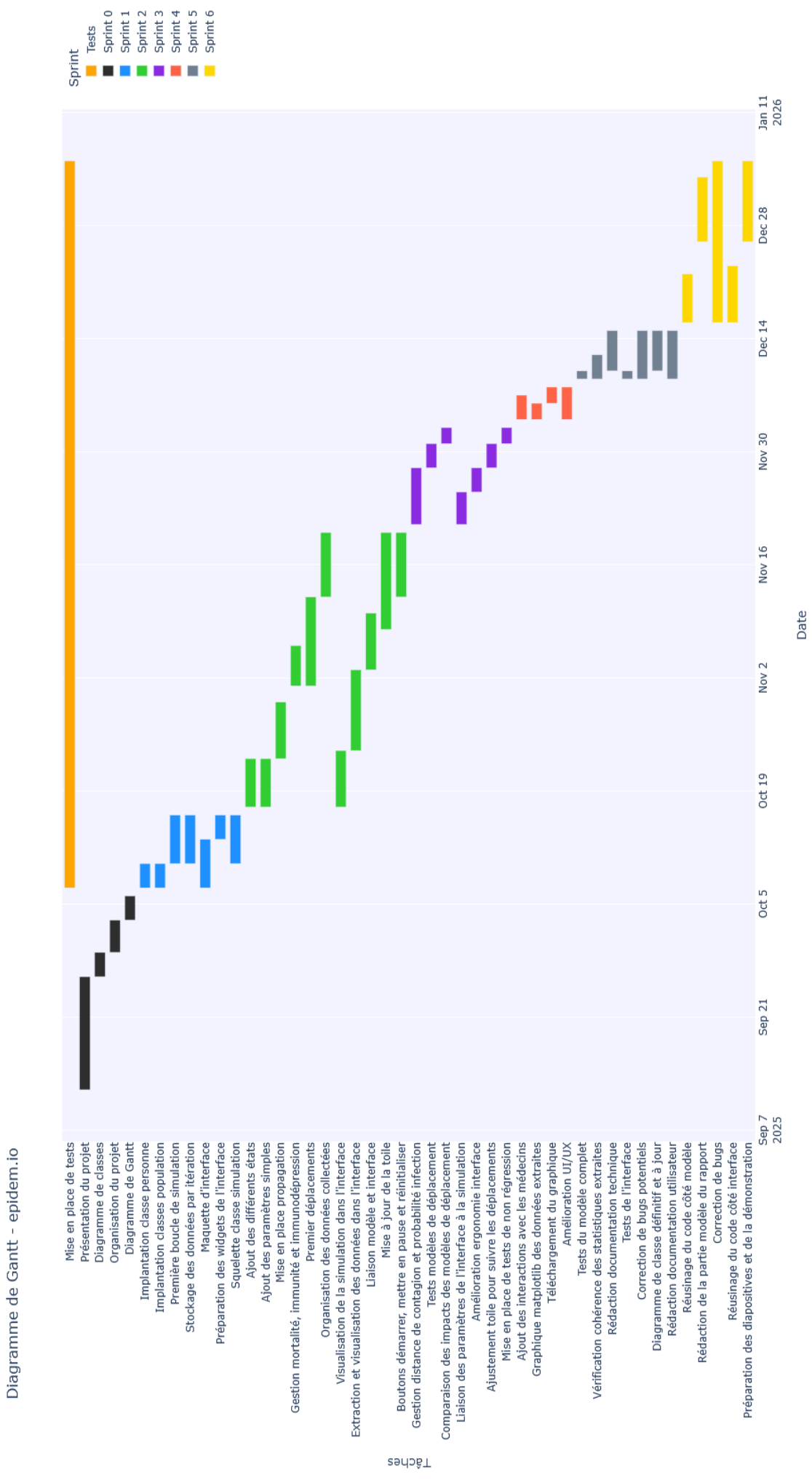


Diagramme final



On observe que notre diagramme de Gantt n'a pas beaucoup changé. L'ajout des tests s'est au final déroulé tout au cours du projet depuis son commencement.

Diagramme de classe

Diagramme initial

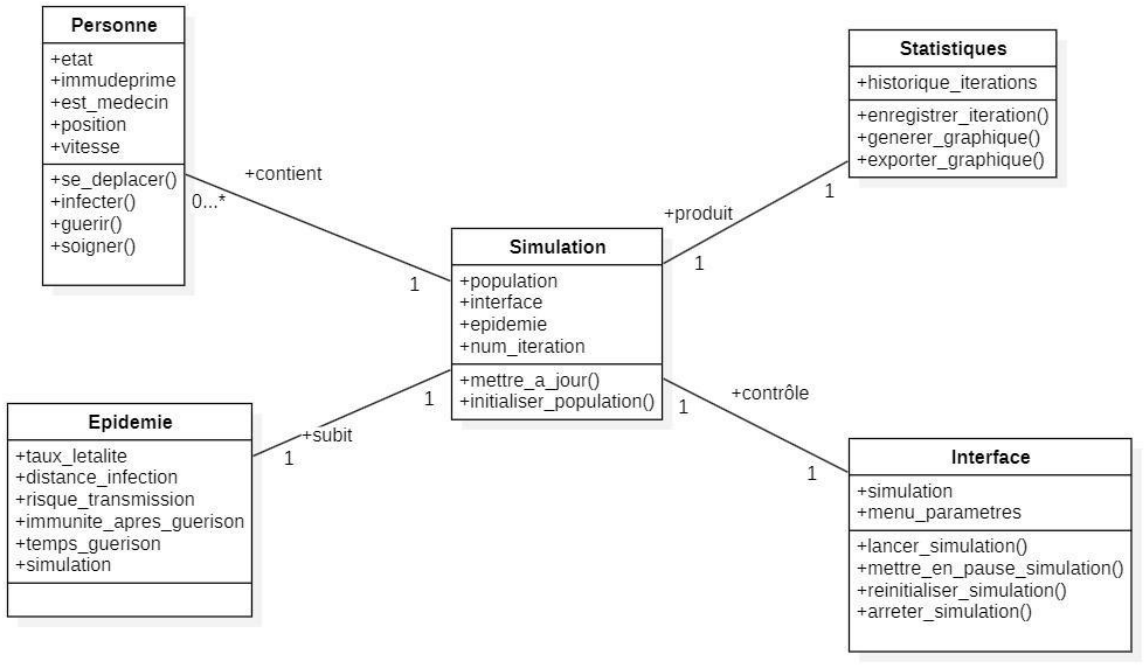
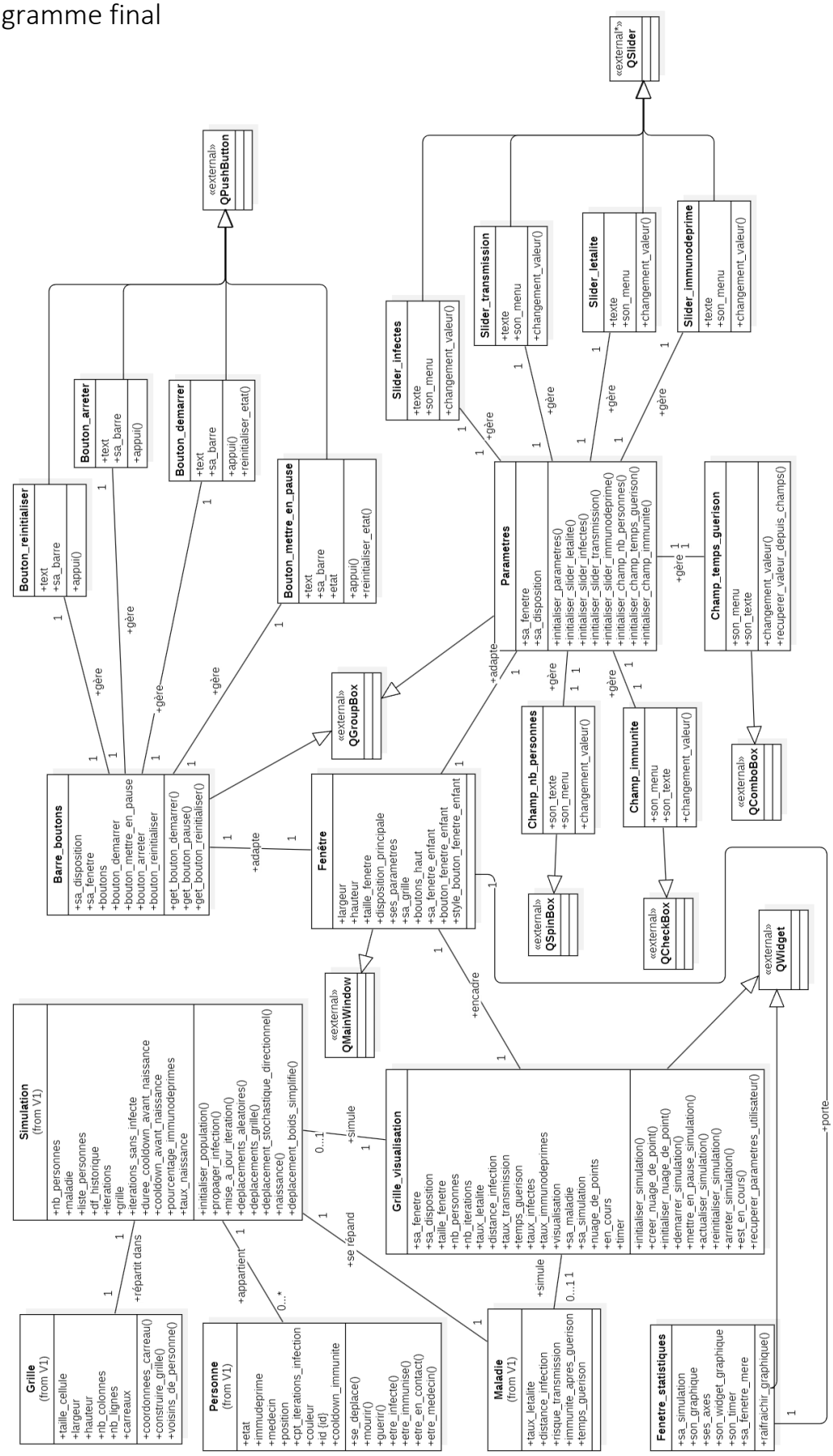


Diagramme final



Travail du sprint 6

Travail prévu

Athène

1. Nettoyage et commentaires du code côté modèle
2. Rédaction de la partie modèle du rapport
3. Correction des bugs
4. Peaufinage du rendu

Adrien

1. Nettoyage et commentaires du code côté interface
2. Rédaction de la partie interface du rapport
3. Correction des bugs
4. Peaufinage du rendu

Travail réalisé

Athène

1. Nettoyage et commentaires du code côté modèle : code entièrement nettoyé et commenté avec réalisation de fichiers pydoc disponibles dans le fichier doc/documentation
2. Rédaction de la partie modèle du rapport disponible ci-dessus
3. Correction des bugs : réalisation de fichiers ptest avec le coverage suivant

```
Name                          Stmts   Miss  Cover   Missing
-----------------------------------------------------------
src\__init__.py                   0      0   100%
src\algorithmie\__init__.py       0      0   100%
src\algorithmie\grille.py        33      0   100%
src\algorithmie\maladie.py        7      0   100%
src\algorithmie\personne.py      36      0   100%
src\algorithmie\simulation.py   239      3    99%   135-136, 295
src\tests\__init__.py             0      0   100%
src\tests\test_classes.py       425      1    99%   572
-----------------------------------------------------------
TOTAL                           740      4    99%
```

4. Peaufinage du rendu : capture de vidéos de simulation ainsi que réalisation du contexte globale de la présentation ainsi que du rendu global

Budget points : 60

Adrien

1. Nettoyage et commentaires du code côté interface : retrait des éléments de débogage, et vérification de la cohérence du format du code entre la partie algorithmie et interface.
2. Rédaction de la partie interface du rapport disponible ci-dessus

3. Correction de bugs : ajout d'explications plus précises et claires sur le fonctionnement du programme, ajout d'une barre de défilement sur les paramètres afin de rendre leur visibilité responsive
4. Vérifications sur le bon fonctionnement des exécutables

Budget points : 40

Conclusion finale sur le projet

Athène

Ce projet m'a apporté beaucoup sur de nombreux points. Tout d'abord, étant en charge de la partie algorithmie du projet, j'ai testé de nombreuses approches pour faire la gestion des déplacements de la population. C'était une approche avec un état d'esprit de recherche que j'ai beaucoup appréciée. De plus, j'ai eu l'occasion de prendre en charge les missions d'organisation des tâches et de planification du projet. Finalement, je suis très contente du résultat que nous avons produit, car le logiciel correspond exactement à ce que j'avais envisagé en début de développement quand j'ai proposé ce sujet à Adrien.

Budget points : 55

Adrien

Au cours de ce projet j'ai appris à utiliser les librairies standards PyQt. J'ai également pu approfondir mes connaissances sur le fonctionnement de la programmation orientée objet notamment avec la syntaxe propre au langage Python. Ce projet, m'a grandement challengé car nous avons dû composer avec la ligne directrice que nous nous sommes nous-même imposés. Cela me sort de ma zone de confort, car je préfère avoir un cadre de travail prédéfini me permettant de bien situer d'où on part et où on va.

Budget points : 45

Commun

Ce projet nous a apporté beaucoup à tous les deux, et nous a permis d'allier nos compétences spécialisées pour créer un projet qui a mis en commun nos connaissances. Nous avons un logiciel complet, avec beaucoup de variables que l'utilisateur peut régler ce qui donne de nombreuses possibilités de personnalisation, qui correspond exactement à ce que nous avions en tête en début de projet. Nous sommes très contents d'avoir fait un projet qui allient des compétences dans des domaines différents, car notre projet nous semble plus complet techniquement.

Nous avons beaucoup aimé notre organisation, nous avons pris soin de préparer nos compte-rendu de sprint au minimum un jour avant la séance en classe pour limiter les évènements imprévus. Nous avons communiqué régulièrement par rapport à nos avancées mutuelles quant au projet. Nos missions étaient à la fois disjointes et liées : le développement des deux parties sont globalement indépendantes, mais par exemple pour tester l'affichage de la population

côté interface, il faut qu'elle soit fonctionnelle côté algorithmie, et pour tester les algorithmes de déplacement il faut pouvoir voir la population dans l'interface. C'était une bonne approche : aucun ne pouvait prendre du retard car ça impacterait le travail de l'autre, mais nous n'avions pas entièrement besoin que les tâches de l'autre soient complétées à chaque étape pour avancer sur ses missions.

Si nous devions recommencer un nouveau projet, nous ferions comme nous l'avons fait pour ce projet : nous avions comme avantage que ce n'était pas le premier projet que nous réalisions ensemble, donc répartir les missions en fonction de nos compétences n'était pas nouveau. Nous utilisons systématiquement Github en projet depuis deux ans donc nous n'avons pas non plus découvert l'outil, et nous étions alignés sur le fait que nous voulions réaliser un projet de qualité et tenir absolument nos deadlines sans prendre de retard. L'organisation a donc été très logique et intuitive en découlant de cela. Le fait que nous soyons d'accord sur ces éléments-là a fait notamment la plus grande force de notre organisation.