

# La propagation de la peste



## Doit-on craindre la peste de nos jours ?

9/05/2023

Présenté par :

Anta Ndiaye  
Margaux Tremier  
Lisa Bachelier  
Léa Perrot  
Eowyn Renoud-Hallereau  
Elvire Racineau  
Nicolas Liégard  
Lysa Picart



## 1. Remerciements

Nous souhaitons remercier l'équipe enseignante qui nous a accompagnés tout au long de ce projet en étant présente et en répondant à chacune de nos questions, en particulier, notre professeur référent M. MARTINOD Pierre. Nous remercions Mme COGREL Claire, Mme CAPALDI Nathalie et Mme GUENEGO Martine pour leur réactivité lors de nos demandes concernant la réalisation de notre sondage. Nous remercions également toutes les personnes ayant accepté de répondre à notre sondage au sujet de leur présence et de leurs déplacements sur le campus de la Chantrerie.

## 2. Résumé

Les êtres vivants ont toujours été confrontés à diverses maladies. Nous avons souhaité travailler sur la propagation de ces épidémies. Pour cela, nous nous sommes concentrés sur la propagation de la peste. En effet, au 14e siècle, la peste a tué entre 30 et 50 % des Européens en l'espace de 5 ans. Ainsi, elle fit 25 millions de victimes sur notre continent.

La peste est causée par le bacille *Yersinia Pestis* et touche aussi bien les humains que les animaux, et se transmet donc entre eux. Selon le mode d'infection, la peste humaine peut se développer selon trois formes primitives : septicémique, pulmonaire et bubonique. Dans notre cas, nous nous sommes concentrées sur les formes buboniques et pulmonaires, par manque d'information sur la forme septicémique.

Pour étudier ce phénomène, nous avons décidé de simuler une épidémie de peste au sein du campus de la Chantrerie. Nous avons commencé par réaliser une modélisation plus simpliste : des points, représentant les différents stades de la maladie dans une population, se déplacent dans un espace fermé. De plus, grâce aux diverses informations collectées, telles que les moyens de transports empruntés, les bâtiments les plus fréquentés (etc.), par la population du campus Chantrerie, nous avons essayé d'obtenir une deuxième modélisation, la plus réaliste possible, représentant ce secteur.

Au vu des conditions sanitaires actuelles, nous pouvons faire la conjecture suivante : *“Une très faible partie de la population décède, et la maladie est rapidement éradiquée.”* Le paramètre majeur de nos modélisations est le  $R_0$ , qui représente le nombre moyen de personnes contaminées par une personne infectée. Nous l'avons calculé en divisant le nombre de personnes contaminées par le nombre de personnes contaminantes. En comparant avec d'autres études et modèles existants sur ce sujet, nous avons pu vérifier cette conjecture.

Avec notre première modélisation, se concentrant seulement sur le déplacement aléatoire des individus dans un cadre restreint, nous obtenons un  $R_0$  de 2.5. Ce résultat est un peu élevé mais pas surprenant au vu de nos approximations. Les résultats de notre seconde modélisation sont, eux, beaucoup moins cohérents. En effet, on trouve un  $R_0$  atteignant en moyenne 19.42. Ce résultat irréaliste s'explique par le manque de considération de certains paramètres (notamment biologiques) ainsi que par les différents facteurs choisis qui nous permettent de calculer ce  $R_0$ .

# Sommaire :

<b>1. Remerciements</b>	<b>2</b>
<b>2. Résumé</b>	<b>3</b>
<b>3. Introduction</b>	<b>6</b>
<b>4. Les Recherches Bibliographiques</b>	<b>7</b>
I. La peste, une maladie ravageuse	7
A. Historique de la peste	7
a. La Peste Noire	7
b. La peste de Bombay	8
c. La peste de Madagascar	9
B. Différenciation des stades de la maladie	10
C. Exposition et Explication des chiffres (taux de mortalité, taux de guérison, temps...)	11
a. Les Temps	11
b. Les Taux	12
c. Les Animaux	12
D. Les recherches menées sur une amélioration des voies de guérison de la peste (vaccin, traitement...)	13
II. Les recherches sur les modélisations déjà existantes de la peste	14
A. Le modèle SI	14
B. Le modèle SIR	14
C. Modèles existants de simulation de la peste bubonique	15
a. Le modèle de réaction diffusion de Noble (1974)	15
b. Le modèle à métapopulations de Keeling et Gilligan (2000)	15
c. Le programme de simulation Plaguesirs	16
d. Le modèle SIMPEST	17
III. Explication du R0	18
A. Définition du R0	18
B. Formule du R0	18
<b>5. Modélisation et Analyse</b>	<b>19</b>
I. Première modélisation	19
A. Points importants de notre programme	20
B. Déroulement du programme	22
II. Modélisation graphique	23
III. Seconde modélisation	24
A. Nouveautés et changements	25
B. Déroulement du programme	29
IV. Analyse	32
A. Première modélisation	32
a. Exposition des résultats obtenus	32
b. Analyse et commentaire	32
B. Deuxième modélisation	33
a. Exposition des résultats obtenus	33

b. Analyse et commentaire	34
C. Comparaison aux modèles déjà existants	35
<b>6. Résultats complémentaires</b>	<b>37</b>
I. Algorithme de recherche de chemin	37
<b>7. Aspect gestion de projet</b>	<b>39</b>
I. Déroulement	39
II. Analyse des difficultés rencontrées	40
A. Gestion de l'humain et du relationnel	40
B. Gestion du temps	40
C. Gestion des risques techniques	41
<b>8. Conclusion</b>	<b>43</b>
<b>9. Table des figures</b>	<b>44</b>
<b>10. Sites internet / Bibliographie</b>	<b>44</b>
<b>11. Annexes</b>	<b>46</b>

### 3. Introduction

Dans le cadre de la conférence PEIP, nous avons choisi de travailler sur le thème de la propagation des virus. Pour ce faire, nous avons fait des recherches sur les différences entre les virus et les bactéries que ce soit dans leur forme ou bien dans leur propagation. La différence la plus marquante entre une bactérie et un virus est leur mode de reproduction. Une bactérie peut se reproduire d'elle-même contrairement à un virus qui lui a besoin d'une cellule hôte pour se multiplier et proliférer. On peut donc en déduire qu'une bactérie se propage plus facilement qu'un virus car elle le fait de manière indépendante alors qu'un virus devra lui trouver une cellule, y entrer et réussir à prendre le dessus pour contaminer la cellule. Les agents permettant la guérison face aux bactéries et aux virus sont eux aussi différents : pour les bactéries, il s'agit des antibiotiques, conçus pour interférer dans le développement des bactéries et pour les virus il s'agit des antiviraux, conçus pour ralentir leur propagation. La vaccination peut aussi être une voie de guérison pour les virus comme pour les bactéries, on peut par exemple citer le vaccin contre la varicelle pour les bactéries ou bien celui contre la COVID-19 en ce qui concerne les virus. Dans ce cas, certaines doses de rappel sont parfois nécessaires. Grâce à ces premières recherches, nous avons décidé d'étendre notre sujet à "La Propagation des Virus et des Bactéries".

La suite logique était de faire des recherches sur divers virus et bactéries afin de choisir sur lequel allait se concentrer notre modélisation finale. Voici la sélection de virus et de bactéries sur lesquels nous nous sommes portés : la rougeole, la COVID-19, la grippe, la peste, la varicelle, la salmonelle, Ebola et le VIH. Lors d'une réunion, nous avons chacun exposé nos recherches au groupe puis, nous nous sommes concertés afin de sélectionner le virus ou la bactérie qui correspondrait, selon nous, mieux à notre projet. En prenant en compte le fait que nous voulions représenter informatiquement la propagation d'une maladie dans une ville, les modélisations déjà existantes sur les virus et bactéries que nous avions retenus, les informations dont nous disposions sur ceux-ci comme par exemple le mode de propagation ou bien encore les chances de guérison, le groupe s'est finalement arrêté sur "la Peste".

Nous pouvons ainsi vous exposer notre problématique sur le thème "La Propagation des Virus et des Bactéries" : "Que se passerait-il aujourd'hui si nous lâchions sur le campus de la Chantrerie une population de rats infectés par la peste ? Arriveraient-ils à tous nous contaminer ? Serions-nous plus résistants à la peste que l'a été la population européenne au Moyen-Âge ? Y'aurait-il des survivants ?"

## 4. Les Recherches Bibliographiques

### I. La peste, une maladie ravageuse

Nous avons tous eu vent de la terrible épidémie de Peste Noire qui a envahi la France et toute l'Europe au Moyen-Age. Cette maladie était si puissante à l'époque qu'elle a décimé plus d'un tiers de la population européenne. Cependant, il n'est pas dit que si la bactérie apparaît aujourd'hui, elle fasse les mêmes dégâts. Là se trouve le noeud de notre projet, nous aimerais voir les conséquences d'une épidémie de peste de nos jours.

#### A. Historique de la peste

##### a. La Peste Noire

Les origines de la bactérie *Yersinia Pestis* sont encore floues, cependant les chercheurs s'accordent pour les placer en Asie (Himalaya, Kirghizistan ou Chine...). Elle aurait gagné l'Europe par l'intermédiaire de la Route de la Soie [1]. L'épidémie de peste en Europe commence alors son chemin en 1347 par le biais d'un bateau de commerce déchargeant ses marchandises infestées de rats et de puces portant le bacille de *Yersinia Pestis* (bactérie de la peste).

Le développement de ce bacille dans les intestins des puces provoque le déclenchement de la maladie : les puces deviennent porteuses de la peste. Si une puce infectée mord un rongeur (ici un rat), il y a de forte chance que celle-ci le contamine à son tour. Pour que la maladie arrive jusqu'à l'Homme, il y a plusieurs possibilités : un rat infecté peut mordre un Homme de son vivant et le contaminer, un Homme peut aussi toucher un rat mort infecté, or comme la peste est extrêmement contagieuse et ce pendant une longue durée, l'Homme est susceptible de contracter la maladie, ou bien une puce peut changer d'hôte à la mort de son rat, choisir un humain et le contaminer. Arrivé à un certain stade de la maladie, les humains peuvent se contaminer entre eux de leur vivant, et une fois mort, le schéma se répète comme avec les rongeurs : même inerte le corps reste contagieux un certain nombre de jours et le toucher est souvent synonyme de contamination fatale.

Ce cycle de contamination inévitable explique pourquoi la peste s'est répandue avec une telle rapidité dans la population européenne et mondiale, causant des millions de morts. Les chiffres parlent d'eux-mêmes, entre 1334 et 1353, on parle de :

- 50 000 à 80 000 décès à Paris [2]
- 7 millions de morts en France (soit 41 % de la population à l'époque) [2]
- environ 25 millions de victimes en Europe (ce qui représenterait entre 30 % et 50 % de la population européenne de l'époque) [3], [4]
- 75 à 200 millions de décès à l'échelle mondiale [1], [2], [4], [5]

Ces résultats désastreux font de l'épidémie de la peste noire, la maladie la plus ravageuse que le monde ait connue comme le montre le graphique suivant (figure 1) :

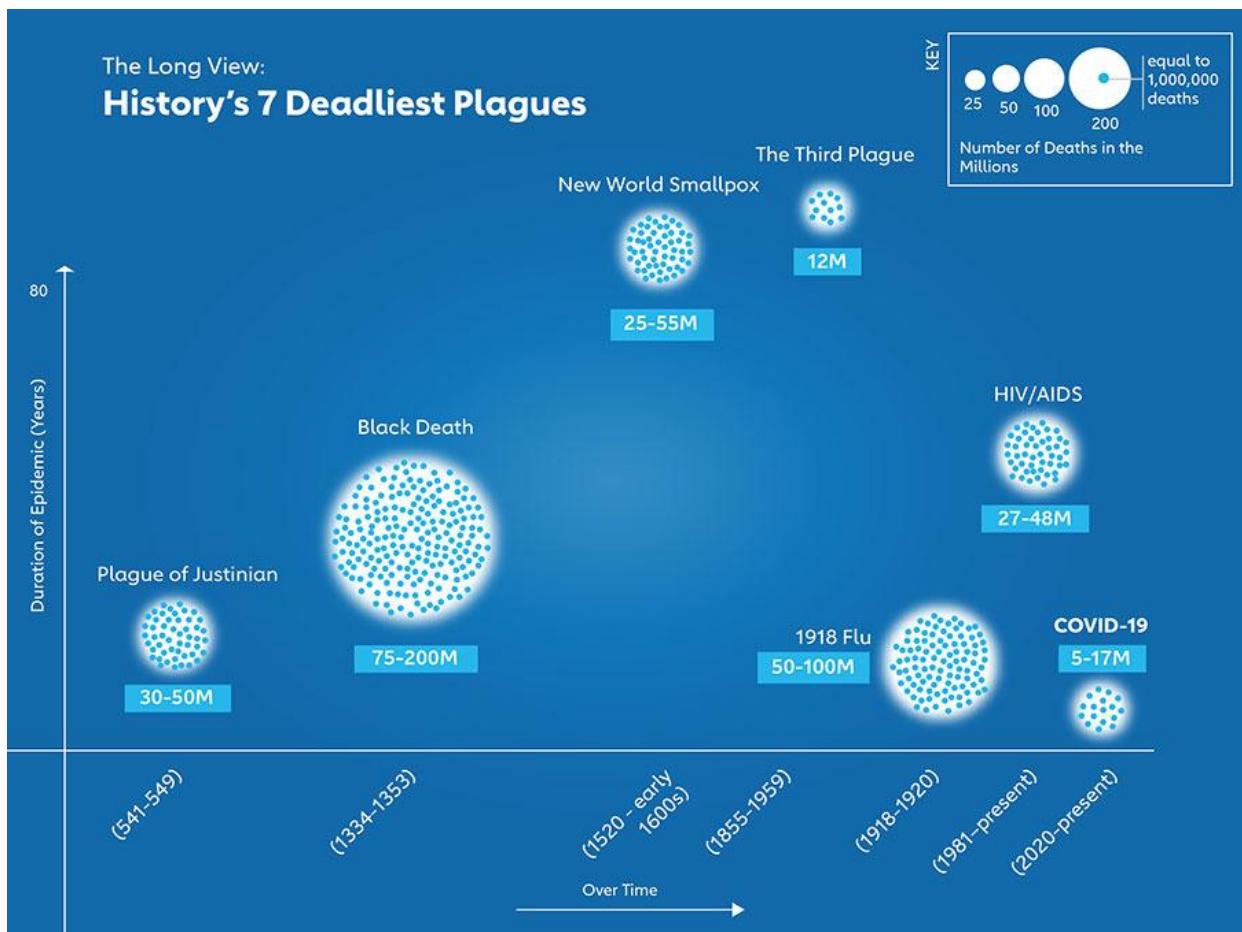


Figure 1: Représentation des différentes épidémies mondiales et de leur ampleur par rapport au temps [5]

### b. La peste de Bombay

Comme vous avez pu le constater sur le graphique précédent, à la fin du XIX<sup>e</sup> siècle, une nouvelle épidémie de peste envahit le continent asiatique. On l'appelle "The Third Plague". Elle a pris ses sources à Hong Kong, dans le sud-est de la Chine jusqu'à atteindre Bombay, l'une des plus grandes villes de l'Inde. Ainsi, entre 1894 et 1898, l'épidémie grandit et finit par se propager dans tout le pays ainsi que dans les régions alentours au fur et à mesure des années. Le tableau suivant (figure 2) répertorie le nombre de morts causées par la peste chaque décennie :

PÉRIODE	MORTS DE LA PESTE	MOYENNE ANNUELLE	POURCENTAGE DE MORTS DE LA PESTE SUR LA PÉRIODE (1898-1948)
1898 - 1908	6 032 695	548 427	47,88
1909 - 1918	4 221 528	422 153	33,51
1919 - 1928	1 702 718	170 272	13,52
1929 - 1938	422 880	42 288	3,36
1939 - 1948	217 970	21 797	1,73
TOTAL	12 597 789	247 015	100,00

Figure 2 : Mortalité liée à la peste en Inde sur la période 1898-1948 [6]

Comme on peut très clairement le voir dans ce tableau, l'épidémie de Bombay a provoqué plus 10 millions de décès [6], [7], [8]. Ce qui prouve bien que, même après le Moyen-Âge, la peste a continué de sévir dans certaines régions du globe et à faire plusieurs millions de morts.

### c. La peste de Madagascar

Alors que la peste est éteinte dans la majorité des pays, et ce, depuis longtemps, elle persiste à Madagascar. En effet, ce pays représente 75 % des cas mondiaux signalés à l'OMS. Depuis les premiers cas en 1850, la peste y est endémique et réapparaît chaque année entre septembre et avril à raison de 200 à 700 cas suspects de peste bubonique par an.

C'est en août 2017 qu'une véritable épidémie de peste pulmonaire se déclare au niveau des Hautes Terres centrales. Touchant principalement les grandes villes qui concentrent 45 % de la population nationale, on parle d'épidémie urbaine. Face à cela, le pays agit et décide de faire une campagne de prévention, et à mettre en place des désinfections dans les lieux à haut risque [9] (figure 3). À cela s'ajoute l'aide de l'OMS qui leur livre 1.2 million d'antibiotiques et 1.5 million \$ de son fond d'urgence.

L'épidémie prit fin en novembre 2017. Durant cette période, 2414 cas suspects ont été déclarés, dont 78 % de cas de peste pulmonaire.



Figure 3 : Désinfection d'un marché pour limiter la propagation de la peste à Madagascar [9]

## B. Différenciation des stades de la maladie

Selon le mode d'infection, la peste humaine peut se développer selon trois formes primitives : septicémique, pulmonaire et bubonique. Les premiers symptômes de la peste sont plutôt proches de ceux d'une grippe actuelle (fièvre, courbature, maux de tête...), cependant chaque forme de peste possède des symptômes particuliers permettant de les différencier :

- La peste bubonique est la plus fréquente. L'infection provoque l'enflure des ganglions lymphatiques qui deviennent douloureux. L'infection est causée par la morsure d'une puce infectée qui se nourrissait sur un rongeur infecté, tel qu'un rat.



*Figure 4 : Représentation de la peste bubonique au Moyen-Age, vision traumatique des symptômes, illustration tirée de la Bible de Toggenburg, 1411 [10]*

Ainsi, comme le montre bien cette image (figure 4), la peste bubonique, en plus d'être très dangereuse, est aussi et surtout très spectaculaire. Lorsque les ganglions augmentent autant en volume, on les nomme bubons (d'où le nom peste bubonique), ils évoluent ensuite en pustules voire en ulcères et le malade se met alors à vomir du sang [1].

- La peste pulmonaire, aussi appelée peste pneumonique, est la forme la plus mortelle de cette maladie, mais aussi la plus rare. L'infection se produit lorsque la bactérie *Yersinia pestis* envahit les poumons, jusqu'à entraîner l'asphyxie du patient [11]. Celle-ci est transmise par les gouttelettes en suspension dans l'air qui sont expulsées lorsqu'une personne ou un animal infecté tousse ou éternue.
- La peste septicémique est, elle, due à une infection du sang qui peut faire apparaître des tâches noires sur la peau. Le terme de "Peste noire" est d'ailleurs certainement lié à ce fait [11]. Le stade septicémique peut être atteint de deux manières différentes : soit par évolution de la peste bubonique, soit en ayant contracté le bacille de la peste après une coupure [2].

## C. Exposition et Explication des chiffres (taux de mortalité, taux de guérison, temps...)

Au cours de notre projet, les recherches bibliographiques ont été plus qu'importantes car il était impératif pour nous de livrer une modélisation réaliste. C'est pourquoi nous avons essayé de comparer le plus de sources possibles afin d'avoir des chiffres parlants et corrects à entrer comme paramètres dans notre programme. Cependant, il est vrai que parfois la précision sur certains chiffres était floue, il nous est même arrivé de ne pas trouver le chiffre dont nous avions particulièrement besoin pour notre modélisation. Nous avons donc dû faire des choix arrêtés sur les chiffres qui vont suivre afin de pouvoir commencer l'analyse précise de résultats que nous étions capables d'obtenir grâce à notre modélisation. Nos recherches ont abouti à deux grands thèmes de chiffrage : les temps et les taux. Ces thèmes seront définis et expliqués à la suite.

### a. Les Temps

Lorsqu'une maladie comme la peste se déclenche dans notre organisme, bien qu'elle soit fulgurante, elle est éphémère. Nos recherches vont donc porter sur le temps que la maladie met à s'installer dans notre système immunitaire mais aussi sur le temps qu'elle met à le désérer (que ce soit par guérison ou par décès).

Intéressons nous, pour commencer, à la période d'incubation. Il s'agit du temps pendant lequel la personne à contracter la maladie, elle est donc contagieuse mais sans le savoir car les premiers symptômes ne se sont pas encore déclenchés. En ce qui concerne notre modélisation, ce n'est pas le chiffre le plus important car il n'y apparaît pas, cependant il correspond implicitement au moment pendant lequel les personnes infectées contaminent les personnes saines. En croisant les sources suivantes pour la peste bubonique [12]–[16], nous avons pu nous arrêter sur une durée d'incubation variant de 4 à 6 jours. Au sujet de la peste pulmonaire, nous arrivons à une période d'incubation plus courte que celle de la peste bubonique qui varie généralement entre quelques heures et 3 jours [13], [16]–[19].

Si le patient n'a pas été pris en charge assez rapidement, il a de fortes chances de décéder suite à ces symptômes. Pour pouvoir supprimer les malades morts de notre modélisation, nous avons dû rechercher un temps de décès en fonction du type de peste contracté. Pour la peste bubonique, les informations sont plus rares sur le temps de décès, toutefois nous savons que la mort n'advient pas aussi vite qu'après la contraction de la peste pulmonaire, le patient succombe généralement au bout de 5 jours environ [1], [13], [20]. À propos de la peste pulmonaire, les symptômes font que la mort du sujet arrive prématurément, c'est-à-dire au bout d'environ 2 jours [13], [17], [18].

Le temps de guérison est une information plutôt difficile à trouver. Après avoir ép杵ché énormément de livres et d'articles à ce sujet, nous n'avons pas vraiment trouvé un résultat précis. Ce que nous savons avec certitude est qu'il est possible de guérir de la peste seulement si le patient est pris en charge assez rapidement. On parle d'un délai d'un à deux jours pour la peste bubonique et de 18 à 24 heures pour la peste pulmonaire, et ceux après l'apparition des premiers symptômes [16], [17].

Sinon, la possibilité que le patient se rende compte trop tard de son état symptomatique et qu'il n'ait pas le temps d'être pris en charge pour être guéri peut le mener à sa perte. Il est aussi vrai que certains patients ont pu guérir de la peste bubonique (car elle est bien moins fulgurante que la pulmonaire). Parfois le bubon suppure ce qui permet le début de la guérison, cependant celle-ci ne prend fin qu'au bout d'une certaine période de convalescence, qui peut s'avérer très longue [12]. Si on parle maintenant du traitement par antibiotiques c'est ici que les informations à notre disposition sont les plus maigres. Nous allons donc nous baser sur les durées connues des divers traitements antibiotiques pour des

maladies plus courantes : il est de 5 à 10 jours [21]–[23]. Cependant il est aussi possible de considérer qu'une personne sous traitement n'est plus contagieuse car elle peut rester chez elle toute la durée de sa convalescence. Cela sera donc un premier choix à faire lors de notre fixation de paramètres dans notre modélisation.

### b. Les Taux

Afin de constater la transmissibilité d'une maladie dans une modélisation graphique, nous devons fixer certains taux. En effet, certaines personnes seront plus contagieuses que d'autres selon le stade de maladie auquel elles en sont. Dans les diverses formes de peste citées antérieurement, il y en a qui sont plus fulgurantes que d'autres et donc celles-ci provoquent une mortalité plus grande. Ces taux sont donc à différencier et à fixer dans les paramètres de notre programme.

Portons nous tout d'abord sur le taux de guérison. Bien que très peu renseigné dans nos sources nous savons tout de même que les traitements antibiotiques contre la peste sont considérés comme efficaces. Ainsi le nœud n'est pas tant sur la performance des traitements mais bien sûr le temps de prise en charge. Dans notre modélisation, le taux de guérison est évalué à 80% tant pour la peste bubonique que pour la peste pulmonaire à condition que celle-ci soit détectée et traitée dans les premiers jours des symptômes.

Toutefois on ne guérit pas toujours de la peste et cette maladie possède même un taux de mortalité plutôt élevé. Comme nous avons pu le voir dans le point historique, les épidémies de peste ont causé la mort de plusieurs dizaines de millions de personnes. C'est pourquoi dans notre modélisation le taux de mortalité associé à la peste bubonique est à fixer entre 40% et 65% [13], [17], [20], [24]–[26]. Comme on a pu l'évoquer précédemment, la peste pulmonaire, elle, est une forme beaucoup plus fulgurante. De ce fait, son taux de mortalité est beaucoup plus extrême étant donné qu'il atteint les 100% sans prise en charge [12], [15], [17], [25]–[27].

Nous pouvons aussi aborder le taux de contagion de la peste pulmonaire. Étant la forme la plus grave de celles décrites précédemment, c'est aussi la seule forme qui possède un taux de contagion interhumaine ce qui la rend d'autant plus dangereuse. En effet, la peste pulmonaire peut se transmettre par gouttelettes infectieuses, qu'elles soient salivaires ou bien respiratoires (toux ou éternuements) [13], [26], [27]. De plus, elle est décrite comme "très contagieuse" [13]. De ce fait, nous prendrons en paramètre un taux de contagion de 60%.

Pour finir, il y a aussi une forte chance d'évolution de la maladie : il n'est pas rare que des sujets atteints de la peste bubonique voient leur maladie évoluer au stade pulmonaire ou septicémique. Cela se produit dans environ 70% des cas [25], étant donné que notre programme ne représente pas la peste septicémique notre paramètre est réglé à 35%, bien que l'accès à cette information soit compliqué.

### c. Les Animaux

Il est aussi impératif de prendre en compte que la peste est une maladie initialement apportée par des corps autres qu'humains, les puces et les rongeurs notamment. Dans notre modélisation nous avons fait le choix de n'y représenter que les rats pour une question de simplicité, en partant du postulat que si un rat apparaît, il est déjà contaminé par la bactérie *Yersinia Pestis*. Voici les chiffres qui y sont liés :

Tout d'abord, parlons de la contagion des rats. C'est entre autres, à cause d'eux (et des puces), qu'il est possible que la peste bubonique se transmette. Ils disposent de deux voies différentes pour contaminer l'homme : la morsure ou bien la transmission par tissus et

liquides organiques (le fait de toucher un rat infecté, mort ou vivant peut vous mener à la peste bubonique) [13], [26], [27]. C'est pourquoi nous avons évalué la contagion du rat à 25% car même si une morsure peut paraître rare, le fait de toucher un rat l'est beaucoup moins (si l'on veut par exemple l'enlever de notre chemin). Dans notre modélisation un rat meurt au bout de 2.4 jours pour que la modélisation dure 60 jours.

En ce qui concerne leur quantité, à Nantes nous comptons environ 2.5 rats par habitant [28]. Cependant il faut garder en tête que les rats de notre modélisation sont absolument tous infectés par la bactérie *Yersinia Pestis*, ils sont donc tous susceptibles de faire circuler la peste. C'est pourquoi nous avons décidé d'en mettre strictement moins que le nombre annoncé précédemment, nous nous sommes arrêtés sur une proportion de 50 rats infectés pour 1000 habitants (soit 5 pour 100).

#### D. Les recherches menées sur une amélioration des voies de guérison de la peste (vaccin, traitement...)

Comme nous avons pu le constater, la peste a disparu dans certains pays mais reste bien active dans d'autres régions. On ne peut donc pas affirmer que nous sommes à l'abri d'un retour de cette maladie. Il est donc important de continuer à mener des recherches afin de trouver des voies de guérison. Comme dit dans l'introduction, les infections causées par les bactéries se soignent grâce à des antibiotiques, mais en existe-t-il certains efficaces contre *Yersinia Pestis* ?

Oui, et ils sont au nombre de 3 : les fluoroquinolones, les tétracyclines, et également la streptomycine. La streptomycine est le médicament de choix, mais il n'est pas disponible partout, il s'est alors avéré que "la gentamicine seule ou en combinaison avec une tétracycline pourrait traiter efficacement la peste humaine" [24]. Attention, ces derniers ne sont efficaces que s'ils sont administrés à temps au patient, et dans le cas de la peste le temps disponible pour agir est court car la maladie évolue très vite, le diagnostic doit donc être rapide. Ces antibiotiques sont aussi bons à donner aux proches du malade afin de prévoir et de contrer la possible infection.

En ce qui concerne les vaccins, une prévention avait été mise en place il y a de ça quelques années. Elle était adressée aux personnes qui prévoyaient d'aller séjourner dans un pays où le bacille de la Peste était encore actif. Cependant, elle a été arrêtée pour cause "d'effet indésirable, parfois sévère" [12]. Puis d'autres vaccins sont arrivés sur le marché mais ceux-ci se sont avérés inefficaces contre la peste pulmonaire (la plus foudroyante des trois formes). Aujourd'hui, l'heure est à la recherche et plusieurs vaccins sont à l'étude, toutefois aucun d'entre eux n'a encore été validé sur les humains. À l'institut Pasteur, une unité nommée "Unité de recherche *Yersinia*" et dirigée par Javier Pizarro-Cerdá, concentre tout son travail autour du bacille et de la maladie qu'il provoque. On peut notamment citer leurs recherches sur cette bactérie et toute sa génomique, sur la résistance de *Yersinia* aux antibiotiques ou bien encore leur vaccin breveté en 2014 qui en est à la phase des tests précliniques. Bien que national, l'unité assure s'investir dans "la lutte contre la peste au niveau international (Centre collaborateur de l'OMS ou CC-OMS)" [12].

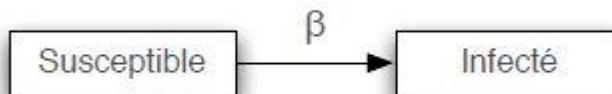
## II. Les recherches sur les modélisations déjà existantes de la peste

Afin d'avoir des outils pour la comparaison des résultats obtenus grâce à la modélisation, nous avons fait des recherches sur les différents modèles de modélisations existants à ce jour.

Nous nous sommes concentrés sur deux modèles bien connus : le modèle SIR et le modèle SI autrement dit, le modèle Hamer.

### A. Le modèle SI

Ce premier modèle apparaît en 1906 grâce à Sir William Heaton Hamer qui est un médecin épidémiologie anglais. C'est un modèle déterministe simple. Il consiste à prendre une population exposée à une maladie contagieuse qu'on va par la suite répartir dans deux groupes distincts. Il y aura tout d'abord les individus infectés, donc contagieux que l'on note "I" et les individus susceptibles d'être infectés que l'on note "S". Pour qu'il y ait une infection, il faut un contact direct entre les individus du groupe S et les individus du groupe I. Le facteur noté  $\beta$ , également appelé taux de contagion, est proportionnel à l'évolution du groupe I.



Le nombre total de population noté N s'exprime de la façon suivante  $N = S + I$ . Le nombre de nouveaux cas infectés pendant un temps  $dt$  sera  $S\beta I$ . Les variations des deux groupes pendant ce temps  $dt$  conduit à deux équations différentielles à résoudre :

$$\frac{dS}{dt} = - \beta SI \quad \text{et} \quad \frac{dI}{dt} = \beta SI$$

Ce modèle a notamment été appliqué pour l'étude de la peste à Bombay [6]. Cependant, lors de l'application, il a été défini que ce modèle était trop simpliste et qu'il manquait donc un paramètre qui prendrait en compte le fait qu'un infecté peut guérir immunisé sans être susceptible d'être de nouveaux infectés. C'est donc de là qu'apparaît le modèle SIR.[29]

### B. Le modèle SIR

Le modèle SIR est un modèle quasi identique au modèle SI. Néanmoins, il y a un paramètre en plus. C'est au début du XXème siècle que W.O. Kermack, un biochimiste et A.G. Mckendrick, un médecin de santé publique, propose ce modèle. Ils voulaient un modèle qui représente au mieux la peste. Donc comme vous pouvez le deviner un nouveau groupe, noté R pour retirés, qui désigne qu'un infecté peut soit guérir de la maladie et donc acquérir une immunité, soit mourir de la maladie ou de toute autre cause indépendante est ajouté. Ici la population est donc divisée en 3 groupes, donc N, la population totale, est égale à la somme de S, I et R.

Il y a deux paramètres : on retrouve le paramètre  $\beta$  le taux de contagion et un nouveau paramètre  $\gamma$  le taux de retrait, également appelé taux de guérison.

$$N(t) = S(t) + I(t) + R(t)$$



La dynamique épidémique est représentée par trois équations différentielles :

$$\frac{dS}{dt} = -\beta SI \quad \frac{dI}{dt} = \beta SI - \gamma I \quad \frac{dR}{dt} = \gamma I$$

Comme vous pouvez le voir ci-dessus, la variation des individus infectés par rapport au temps dépend maintenant de deux termes. Il y a tout d'abord le premier terme qui est commun au modèle SI ( $\beta SI$ ) et le deuxième terme ( $\gamma I$ ) qui désigne le produit du taux de retrait, compris entre 0 et 1, et le nombre d'individus susceptibles d'être infectés. Ce qui nous donne donc la différence entre le nombre de personnes nouvellement infectées et le nombre de personnes nouvellement guéries. [29]–[31]

Nous pouvons aussi aborder le modèle SEIR qui est un modèle plus complexe que le modèle SIR, car il intègre un nouveau paramètre, des personnes infectées non infectieuses. Cela veut dire qu'elles ne peuvent pas transmettre la maladie donc ne sont pas contagieuses. Cependant, ce n'est pas un modèle cohérent avec notre programme, car ce type de cas n'existe pas dans la propagation de la peste. Nous avons donc fait le choix de ne pas développer sur le SEIR.

### C. Modèles existants de simulation de la peste bubonique

#### a. Le modèle de réaction diffusion de Noble (1974)

Cette expérience de modélisation visait à reproduire la diffusion de la pandémie de peste noire en Europe au XIV<sup>e</sup> siècle. Cette simulation est un modèle de type réaction-diffusion, basé sur le modèle S.I.R. de Kermack-McKendrick (1927). Le but de ce modèle est de comprendre les conditions d'apparitions d'une onde de propagation épidémique, selon la densité de population humaine. Les conditions de modélisation du modèle de Noble sont les suivantes : la population est répartie uniformément et les déplacements humains sont courts et aléatoires. Seules les classes de la population humaine "susceptibles" et "infectées" vont varier, et on considère qu'il n'y a pas de renouvellement de la population. La méthode de diffusion de la peste (piqûres de puces infectées sur des rats) va être généralisée dans un paramètre appelé "coefficients de transmission". Cependant, on constate que ce modèle est peu représentatif de la réalité, car il génère une propagation de proche en proche, ce qui contredit les observations faites sur plusieurs foyers de peste, où on parle plutôt de diffusion hiérarchique. [29]

#### b. Le modèle à métapopulations de Keeling et Gilligan (2000)

Les précédents modèles de diffusion de la peste bubonique sont des modèles anthropocentriques, qui modélisent la peste comme si elle ne se transmettait qu'aux populations humaines. Cependant, les études biologiques ont montré que la peste était une zoonose, c'est-à-dire, une maladie des rongeurs qui se propagent grâce aux puces, et qui occasionnellement, contaminent les humains. C'est pour cette raison que Matt Keeling et Chris Gilligan, ont créé un modèle épizootique pour les populations de rats et de puces, qu'ils ont couplé avec un modèle épidémique standard, afin de comprendre les circonstances de multiplication de cas de peste chez l'Homme.

Les deux Hommes divisent la communauté de rats et leurs puces associées, en plusieurs ensembles de sous-populations, avec la possibilité de migrer vers une sous-population voisine. Cette modélisation met en avant deux phénomènes.

Dans un premier cas, la proportion de rats susceptibles d'être contaminés est faible, ce qui entraîne un comportement endémique chez les rats, et une très faible force d'infection chez l'Homme. Cependant, lorsque la proportion de rats pouvant être contaminés est très importante, la population de rats est alors très vite décimée et les puces se trouvant sur ces

rats changent d'hôtes. C'est ainsi qu'on observe une très forte contamination chez les humains. Le modèle de Keeling et Gilligan met en avant le fait que la maladie peut persister dans une sous-population de rats et se transmettre par la suite aux humains, ce qui peut entraîner une épidémie. Cela a permis d'expliquer que l'apparition d'une épidémie de peste ne coïncidait pas nécessairement à une importation extérieure de la maladie. Certaines épidémies de pestes buboniques se produisaient donc malgré la mise en quarantaine de certaines villes. De plus, ce modèle a aussi mis en évidence le fait que la vaccination ne permettait pas d'éradiquer la maladie, car celle-ci est transmise par les rats aux Hommes. Il est donc important de prendre l'évolution des populations de rats pour étudier la peste. [29], [32]

#### Équations du modèle :

$$\frac{dS_R}{dt} = r_R S_R (1 - T_R/K_R) + r_R R_R (1 - p) - d_R S_R - \beta_R S_R F [1 - \exp(-aT_R)]/T_R$$

$$\frac{dI_R}{dt} = \beta_R S_R F [1 - \exp(-aT_R)]/T_R - (d_R + m_R) I_R$$

$$\frac{dR_R}{dt} = r_R R_R (p - T_R/K_R) + m_R g_R I_R - d_R R_R$$

$$\frac{dN}{dt} = r_F N (1 - N/K_F) + d_F F [1 - \exp(-aT_R)]/T_R$$

$$\frac{dF}{dt} = (d_R + m_R [1 - g_R]) I_R N - d_F F$$

$$\lambda_H = F \exp(-aT_R)$$

SR est le nombre de rats pouvant être contaminés, IR est le nombre de rats pouvant infecter, TR = SR+ IR+ RR est le nombre total de rats, N est le nombre moyen de puces sur un rat, F est le nombre de puces contaminées recherchant un hôte,  $\lambda_H$  représente la force d'infection chez les humains.  $\lambda_H$  correspond donc dans le modèle au nombre de puces contaminées qui n'ont pas trouvé de rats comme hôtes. [32]

#### c. Le programme de simulation Plaguesirs

Le modèle précédent considère la présence d'une seule espèce de rongeurs, les rats. Mais en réalité, *Yersinia pestis* peut persister dans de nombreuses populations de rongeurs comme des souris ou des campagnols. Le but de ce programme de simulation est d'obtenir des informations sur la persistance de la peste. Plaguesirs est basé sur des modèles déterministes SIR (décris précédemment). Deux matrices vont décrire les taux de contacts contagieux d'un rongeur vers une puce et inversement, ce qui permet d'obtenir le taux de transmission de la maladie, ainsi que le taux de charge d'un hôte. Le taux de natalité des hôtes est estimé en étudiant les portées de rongeurs. Une étude a été menée avec ce programme. Plusieurs simulations ont été effectuées pour comprendre l'impact de la composition de la population de rongeurs, sur la persistance dans le temps de la peste, dans le camping Chuchupate, dans le comté de Ventura. Là-bas, plusieurs espèces de rongeurs interagissent. Les chercheurs se sont concentrés sur une communauté réservoir

composée de toutes ces espèces de rongeurs. Cette communauté a été divisée en 21 sous-ensembles, et les chercheurs ont simulé la dynamique de ces 21 sous-ensembles sur une durée de 20 ans. Cette simulation a montré qu'une communauté réservoir composée d'une seule espèce de rongeurs avait une persistance plus faible qu'une communauté composée de deux rongeurs. Il a également été mis en évidence deux points favorisant la persistance de la peste : l'hétérogénéité de la saison de reproduction des hôtes et l'augmentation du nombre d'hôtes. Concrètement, le fait d'avoir une population d'hôtes composée de plusieurs espèces qui ne se reproduisent pas toutes aux mêmes périodes, favorise la présence constante de la maladie dans certaines zones. [29], [33]

#### d. Le modèle SIMPEST

Ce modèle est un modèle individu-centré, c'est-à-dire qu'il représente l'ensemble des individus composant le système (voir figure 5), ce qui s'oppose au modèle dynamique.[34] Dans ce modèle, on utilise le comportement individuel de chaque entité composant le système, et on étudie les réactions individuelles et collectives face à une modification de l'environnement tel qu'une épidémie. Les précédents modèles simulant la propagation de la peste, développés jusqu'ici, sont des modèles dynamiques. Ils ont permis de mettre en lumière les conditions de persistance de la peste sur des espaces importants, mais également, l'impact de la présence de plusieurs espèces de puces et de rongeurs dans le cycle épidémiologique.[35]

Le modèle SIMPEST permet d'étudier la peste dans un foyer précis : le foyer de Madagascar. La peste de Madagascar est une peste bubonique qui a fait son apparition suite aux échanges commerciaux liés à la colonisation. Elle a été apportée par un rat clandestin qui a colonisé toute l'île et éliminé les espèces locales de rongeurs. Ce rat constitue à la fois le principal hôte et la principale victime de la peste de Madagascar. [36]

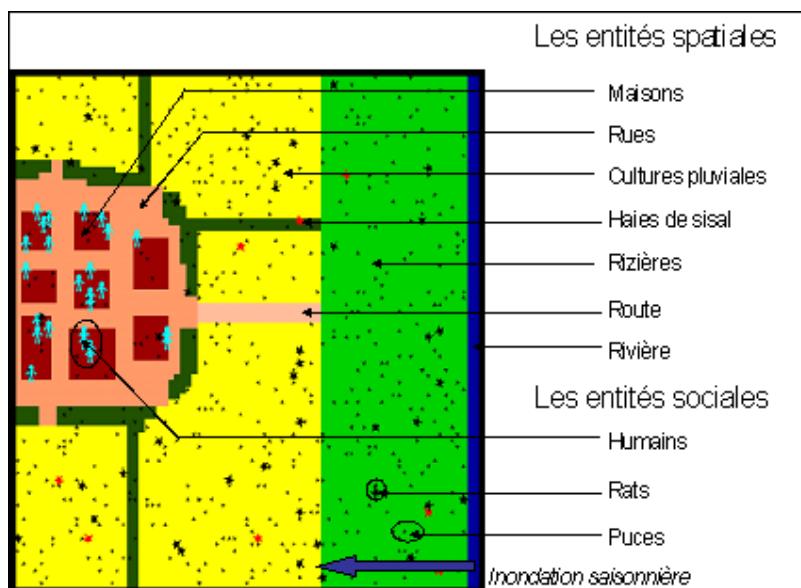


Figure 5 : Entités de base du modèle SIMPEST [36]

### III. Explication du $R_0$

#### A. Définition du $R_0$

Le  $R_0$  est un indicateur extrêmement important dans le cadre d'une épidémie. Cette valeur signifie "*taux de reproduction du virus*". Il indique le nombre moyen de nouveau cas d'une maladie qu'une personne infectée et contagieuse va générer en moyenne dans une population qui ne possède aucune immunité.

#### B. Formule du $R_0$

Ici, on s'intéresse donc au  $R_0$  de la peste. Ce dernier a pu être calculé grâce au modèle SIR (*Susceptible-Infected-Removed*) de Kermack-McKendrick vu précédemment. Pour rappel, ce modèle est régi par 3 équations différentielles que voici :

$$\frac{dx}{dt} = -k x y, \quad \frac{dy}{dt} = k x y - \ell y, \quad \frac{dz}{dt} = \ell y$$

où  $t$  est le temps,  $x$  est le nombre de personnes sensibles,  $y$  est le nombre de personnes infectées,  $z$  est le nombre de personnes qui se sont rétablies et ont développé une immunité contre l'infection,  $kx$  qui est le taux d'infection et  $\ell$ , le taux de guérison.

Dans cette partie, ce qui va nous intéresser est le  $z$ . En effet, le  $z$  correspond en fait à notre  $R_0$ . On en déduit donc que  $R_0 = kx/\ell$ .

En fonction de la valeur obtenue, on va pouvoir en tirer des conclusions. Si  $R_0 < 1$ , cela signifie qu'une personne ayant contracté la maladie va en moyenne contaminer moins d'une personne. Par conséquent, elle s'estompera jusqu'à disparaître. A l'inverse si  $R_0 > 1$ , alors l'individu contaminera en moyenne plus d'une personne et la maladie se propagera davantage. À noter que la formule s'applique uniquement dans le cadre du modèle SIR de Kermack-McKendrick. Sortie de ce modèle, cette formule perd de son sens.

Si nous nous sommes intéressés à ce  $R_0$ , c'était dans le but de l'intégrer à nos résultats. Plus précisément à notre modélisation. En effet, pour obtenir des résultats qui nous sont propres, il nous fallait faire nos propres calculs.

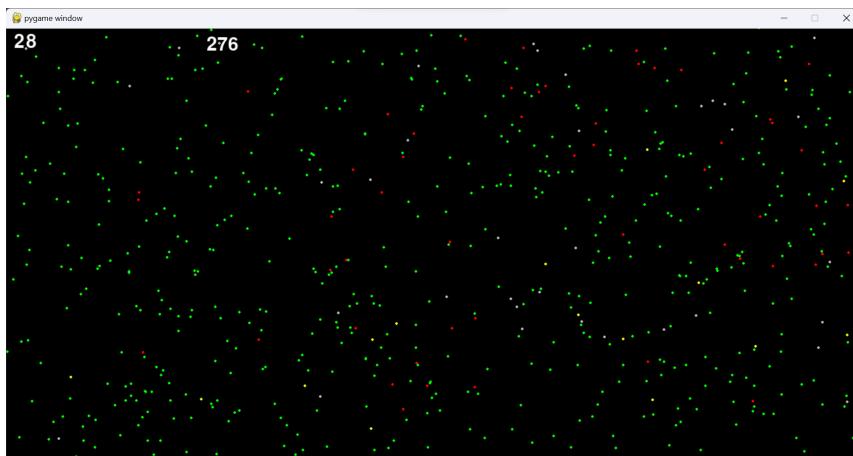
## 5. Modélisation et Analyse

Lors de la préparation de notre projet nous avions pour ambition de modéliser l'évolution de la peste au sein d'une ville. Cependant, après réflexion, nous nous sommes mis d'accord sur le fait de commencer par une modélisation dans un espace plus petit afin de limiter le nombre d'individus à représenter et simplifier le problème. C'est alors que nous avons choisi de modéliser la propagation de cette bactérie au sein du campus de la Chantrerie, un lieu que nous connaissons tous très bien.

Pour réaliser ce projet nous avons utilisé le langage de programmation Python. Nous avons eu recours à la bibliothèque Pygame, cette librairie permet d'afficher une interface graphique dans une fenêtre.

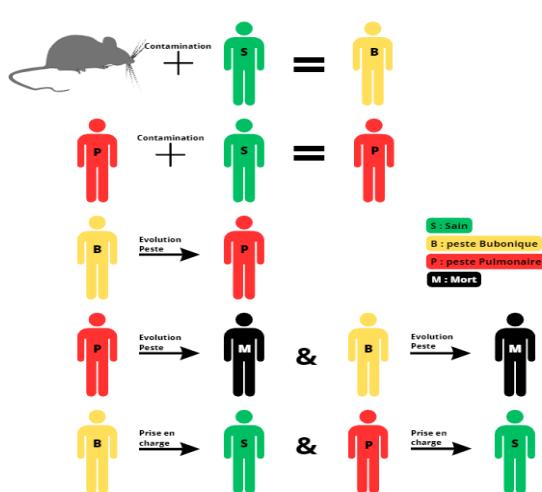
### I. Première modélisation

Le but final de notre projet était d'arriver à modéliser la peste au sein du campus de la Chantrerie. Mais avant d'arriver à cela, nous avons simplement modélisé des points de différentes couleurs, avec des déplacements aléatoires. Nous allons expliquer ici comment nous y sommes arrivés.



*Figure 6: Visuel final de notre modélisation*

Les points verts sont les habitants sains, les jaunes ceux qui sont atteints de la peste bubonique, les rouges sont ceux qui sont atteints de la peste pulmonaire, les points gris sont les rats, et les habitants morts sont des points noirs afin qu'on ne les voit pas (figure 10).



*Figure 7 : Schéma de la contamination entre les individus dans notre modélisation*

## A. Points importants de notre programme

Premièrement il a fallu définir les points (que nous appellerons dans la suite habitant), pour cela nous utilisons la fonction `pygame.draw.circle()`, ainsi que leurs déplacements. Nous avons fait le choix d'attribuer à chaque habitant une position en x et en y, et à chaque tour de boucle une valeur x et y est ajouté à leurs positions. C'est-à-dire, qu'on tire aléatoirement un réel x et y entre deux bornes (`-val_aleatoire, val_aleatoire`), puis on ajoute ces valeurs aux positions x et y des habitants. Par exemple si `val_aleatoire` vaut 20 (valeur finale), on tire aléatoirement en réel entre -20 et 20, pour x, puis pour y.

Après différents tests nous sommes arrivés à la conclusion qu'il serait utile de définir une classe habitant. Cette classe a pour attribut la couleur de l'habitant, sa position x et sa position y, ainsi qu'un temps (nous en parlerons plus tard) et le rayon de son cercle. Ceci nous permet de ne pas avoir plein de tableaux différents par couleur, et de ne pas avoir à différencier les fonctions selon la couleur des points. On stocke dans un tableau `Habitant`, tous les objets de la classe habitant, afin de pouvoir les parcourir et par exemple changer leur couleur. Et on stocke dans des variables telles que `nbvert`, le nombre de points présents de chaque couleur.

Nous avons aussi dû définir une fonction `VerifPos()` qui permet que les habitants ne sortent pas de la partie visible de la fenêtre. Pour cela, si l'habitant dépasse du côté des négatifs on lui ajoute la valeur absolue de x ou y, et s'il dépasse de l'autre côté on lui soustrait la valeur absolue de x ou y.

Une fonction très importante de notre modélisation est la fonction `collision()`. Elle est appelée à chaque tour de boucle. Pour tous les habitants, si leur couleur correspond à la première donnée en entrée, elle calcule la distance (norme euclidienne) avec tous les autres habitants de la deuxième couleur mise en entrée de la fonction, et si elle est inférieure à quatre fois le rayon du point, on considère qu'il y a collision. Nous avons choisi le facteur quatre afin d'avoir assez de collisions, pour la même raison le rayon des points est de deux pixels. De plus, avant de considérer qu'il y a eu collision, on tire un nombre aléatoire entre zéro et un et si ce nombre est inférieur au facteur décidé, alors il y a bien eu collision. Cela permet de dire par exemple que les habitants rouges ne contaminent qu'avec 60% de chance. Si la collision est validée, la couleur de l'habitant devient la deuxième, on met à jour le nombre d'habitants de chaque couleur (ces nombres sont renvoyés par la fonction), et on met à jour le temps de l'habitant. Par exemple, on parcourt tous les habitants, quand on en trouve un vert, on calcule sa distance avec tous les points rouges, s'il est suffisamment proche d'un, et que le nombre aléatoire tiré est inférieur à 0.6, alors le point vert devient rouge.

Nous avons également défini une deuxième fonction `collision2()`, qui nous est utile pour les rats. Dans cette fonction lorsqu'il y a collision entre deux habitants (un rat et un vert), le premier ne devient pas de la couleur du deuxième comme dans `collision()`, mais plutôt le deuxième devient d'une troisième couleur. Ainsi si un rat (habitant gris) rencontre un habitant vert, l'habitant vert devient jaune, et le rat ne change pas.

Mais la peste n'évolue pas uniquement par contamination, comme cela a été expliqué précédemment. En effet, il faut introduire le facteur temps, afin que les habitants puissent changer de couleur pas par collision mais au bout d'un certain temps. Pour cela nous avons défini trois fonctions : `VerifTempsSup()`, `VerifTempsInf()`, `rat()`, qui fonctionnent sur le même principe.

Commençons par la fonction `VerifTempsSup()`, elle débute par vérifier si l'habitant considéré est de la première couleur donnée en entrée de la fonction, si c'est le cas deux vérifications sont faites. Tout d'abord, on calcule la différence entre le temps qui s'est écoulé depuis le début de la simulation, et le temps depuis lequel l'habitant est de cette première couleur, si ce temps est supérieur au temps d'attente décidé la condition est validé. La deuxième condition est de même sorte que pour `collision()` : on tire un nombre aléatoire entre zéro et un et si ce nombre est inférieur au facteur décidé, c'est bon. Si les deux

conditions sont validées, l'habitant prend pour couleur la deuxième donnée en entrée de la fonction, et on met à jour le temps, ainsi que le nombre d'habitants de chaque couleur (nombres que la fonction renvoie). Prenons pour exemple l'évolution de la peste bubonique (habitant jaune) en peste pulmonaire (habitant rouge). Si l'habitant considéré est bien jaune, on tire un nombre aléatoire, si ce nombre est inférieur à 0.35, et que cela fait plus de 3.5 jours qu'il est jaune (si la différence, entre le temps écoulé depuis le début, et le temps auquel il est devenu jaune est supérieure à 3.5 jours), alors l'habitant devient rouge et son temps est réinitialisé.

La fonction `VerifTempsInf()` fonctionne presque pareil, mais à la place de vérifier que cela fait plus de tant de temps que le point est de la couleur une, on vérifie que cela fait exactement tant de temps que le point est de cette couleur, et alors on le change de couleur. Cette fonction nous sert pour les guérisons, ainsi si un point rouge, ou jaune, est de cette couleur depuis un jour, on tire un nombre aléatoire et s'il est inférieur à 0.8, alors l'habitant guérit (devient vert), car cela voudrait dire qu'il a été pris en charge dans les premières 24h.

La fonction `rat()` est un peu plus différente, elle sert à tuer les rats au fur et à mesure, pour que la modélisation s'arrête quand il n'y en a plus. Si le temps écoulé depuis le début de la modélisation est supérieur à  $n$  fois le temps d'attente décidé, alors le rat meurt (devient noir) et on augmente  $n$ . La variable  $n$  est donc un nombre entier qui permet qu'un rat meurt tous les multiples du temps d'attente défini au départ.

Il faut détailler un peu plus comment fonctionne le temps dans notre modélisation. C'est la variable `test` qui le représente, elle est initialisée à zéro et incrémenté à chaque tour de boucle du "while" qui permet de faire tourner la modélisation, c'est-à-dire que la variable `test` compte le nombre de tour de boucle qu'il y a eu depuis le lancement de la modélisation. Chaque habitant a un attribut `temps`, il correspond à la valeur de la variable `test` au moment où il a été créé, ou au moment où il a changé de couleur. Par exemple, au début les habitants sont soit verts soit des rats, leur attribut `temps` vaut zéro, puis un vert devient jaune, et alors son attribut `temps` change, et prend pour valeur le numéro du tour de boucle (la valeur de `test`) auquel il a changé de couleur.

Nous avions d'abord commencé par compter le temps avec la fonction `pygame.time.get_ticks()`, donc en millisecondes, mais cela n'était pas pratique pour la fonction `VerifTempsInf()` car on ne pouvait pas mettre : "si le temps est égal tant de millisecondes faire quelque chose" car elles changent trop vite, alors qu'on peut dire "si le temps est égal à tant de tours de boucle", et faire une action.

Il a aussi fallu initialiser les temps d'attente (par exemple au bout de combien de temps les habitants jaune peuvent devenir rouges). Nous les avions d'abord imaginés de telle sorte qu'une seconde correspondait à un jour, donc quand il a fallu passer en tour de boucle nous avons fait le choix d'une seconde égale à neuf tours de boucle, donc tous les temps ont été multipliés par neuf, puis passé en entier, afin de ne pas avoir de problème avec les conditions "if ... == ...". Par exemple le temps avant que les habitants rouges aient une chance de mourir est de 2 jours et demi, donc 22 tours de boucle car  $2.5 * 9 = 22.5$ .

Enfin, une des dernières parties de notre modélisation est le calcul du  $R_0$ , c'est-à-dire du nombre d'habitants contaminés en moyenne par un habitant rouge. C'est grâce à ce nombre que l'on peut comparer notre modélisation, aux chiffres et modèles connus de la peste. Nous calculons le  $R_0$  en divisant la somme des contaminations par le nombre d'habitants contaminant. Pour cela on compte dans un tableau  $R_0$  le nombre d'habitants que chaque habitant contamine et dans un autre tableau couleur on enregistre si l'habitant a été rouge à un moment, même si après il devient vert ou noir, car on veut le nombre total d'habitant rouge qu'il y a eu dans toute la modélisation. Enfin, après que la modélisation soit finie, on parcourt tous les habitants, si la case du tableau couleur leur correspondant contient 'r' (l'habitant a été rouge à un moment) on ajoute sa case du tableau  $R_0$  à une variable somme  $R_0$ , et on ajoute un à une variable nbrougetot, et pour finir on divise somme  $R_0$  par nbrougetot.

Prenons l'exemple d'une modélisation sur dix habitants, avec au total 5 habitants qui ont à un moment donné été rouges. Le tableau couleur pourrait ressembler à ça :

[0,'r',0,'r','r',0,0,0,'r','r'], et le tableau  $R_0$  à ça : [2,0,0,0,1,0,0,1,0,0], car il est possible que certains habitants rouges ne contaminent personne. On a donc  $\text{somme}R_0 = 4$ , et  $nbr\text{ougetot} = 5$ , donc le  $R_0$  serait de  $4 / 5 = 0,8$ .

## B. Déroulement du programme

Décrivons le programme étape par étape. D'abord la fenêtre est définie, pour cela on utilise la fonction `pygame.display.Info.current_w` (ou `_h`) afin de connaître la taille de l'écran de l'utilisateur, et on définit une fenêtre pygame à l'aide de la fonction `pygame.display.set_mode()`. Puis les variables sont initialisées, voici les plus intéressantes :

- `val_aleatoire = 20` # correspond au déplacement max des points d'un instant à l'autre
- `rayon_cercle = 2`
- `test = 0` # nombre de tours de boucle depuis que la fenêtre est lancée
- # nombre d'habitants au départ
- `nbvert, nbjaune, nbroute, nbrat, nbmort = 1000, 0, 0, 50, 0`
- `nbtot = nbroute+nbvert+nbjaune+nbrat+nbmort`
- # facteurs de chance (ex : `factjr` = facteur de jaune en rouge) compris entre 0 et 1
- `factjr = 0.35`
- `factjv = 0.8`
- `factjn = 0.55`
- `factrn = 1`
- `factrv = 0.8`
- `factgn = 1`
- `factvr = 0.6`
- `factgvj = 0.25`
- # temps d'attente (ex : `tattjr` = temps d'attente de jaune en rouge) en secondes\*9 = nb de tours de boucles
- `tattjr = int(3.5*9)`
- `tattjv = int(1.0*9)`
- `tattjn = int(4.0*9)`
- `tattrn = int(2.5*9)`
- `tattrv = int(1.0*9)`
- `tattgn = int(3.5*9)`

Ensuite on initialise les habitants, pour cela on parcourt le nombre total d'habitants, et on crée un objet de la classe `Habitant`, de couleur blanche au début, pour chacun d'eux, on remplit le tableau `Habitant` avec tous ces objets. Puis on parcourt le tableau `Habitant`, couleur par couleur en utilisant les variables `nbvert, nbroute, ...` et pour chacun on attribue la bonne couleur et une position aléatoire dans la fenêtre.

Puis on peut rentrer dans la boucle `while`, qui s'arrête si la croix est cliquée ou s'il n'y a plus de rats. On affiche un fond noir et calcule le nombre de seconde. Ensuite la plupart des actions se font dans une boucle "for i in range(nbtot)" : on tire un nombre aléatoire pour `x` et `y` comme expliqué plus haut, on additionne ces nombres aux positions en `x` et `y` des habitants, on vérifie qu'ils sont toujours dans la fenêtre grâce à la fonction `VerifPos()`, et on affiche les points, à l'aide d'une fonction `dessin()` attribué à la classe `Habitant`, utilisant la fonction `pygame.draw.circle()`.

Par la suite, toujours dans la boucle `for`, on appelle la fonction `VerifTempsSup()` pour les habitants jaunes qui deviennent rouges au bout d'un moment, puis pour les rouges qui meurent (deviennent noir), et pour les jaunes qui meurent aussi. On appelle également la fonction `VerifTempsInfr()` pour les habitants rouge qui ont la possibilité de guérir (devenir vert)

à un moment, et de même pour les habitants jaunes. On appelle aussi la fonction rat() pour que les rats meurent peu à peu. Enfin on met à jour le tableau couleur, ajoutant un 'r' dans la case de l'habitant concerné s'il est rouge.

En dehors de cette boucle for, on appelle la fonction collision() pour que les habitants verts deviennent rouges s'ils sont près de l'un d'eux. Et on appelle la fonction collision2() pour que les habitants verts deviennent jaunes s'ils sont près d'un rat. Puis on affiche le nombre de secondes et le nombre de tours de boucle auquel on est grâce à la fonction screen.blit() (screen étant notre fond noir), et on n'oublie pas d'incrémenter d'un la variable test, correspondant au nombre de tours de boucle.

Quand la fenêtre pygane est fermée, on calcule le  $R_0$ , que l'on print, avec le nombre d'habitant vert restant. Enfin le programme ajoute dans une feuille excel extérieur les variables qui nous intéressent, afin de garder une trace des différentes fois où le programme a été exécuté. Les variables sauvegardées sont : le nombre d'habitant vert restant, le nombre de contamination qu'il y a eu (la valeur de sommeR<sub>0</sub>), le nombre d'habitant rouge qu'il y a eu en tout (la valeur de nbrougetot), le  $R_0$  calculé (sommeR<sub>0</sub>/nbrougetot), et le nombre de tour de boucle qu'il y a eu (la valeur de test).

## II. Modélisation graphique

Afin de représenter les déplacements des individus et leur contamination sur le campus de la Chantrerie nous avons dû réaliser une interface graphique.

Tous les éléments et fonctions cités dans la suite de ce paragraphe sont contenus dans une fonction draw\_window() qui sera appelée dans le programme principal pour afficher la fenêtre où se déroulera la modélisation.

Notre interface est composée d'un plan du campus comme image de fond sur lequel nous avons rajouté des éléments pour représenter les bâtiments et les chemins. Nous souhaitions initialement nous servir d'un plan de construction des bâtiments de Polytech mais nous n'avons pas trouvé de plan où tous les bâtiments apparaissent tous aux bonnes échelles. Nous nous sommes donc rabattus sur une simple vue aérienne du campus, issue de Google Maps. Nous avons modifié l'image à l'aide d'un logiciel de montage photo afin de la simplifier et de ne garder que les éléments utiles à notre modélisation (figure 8).

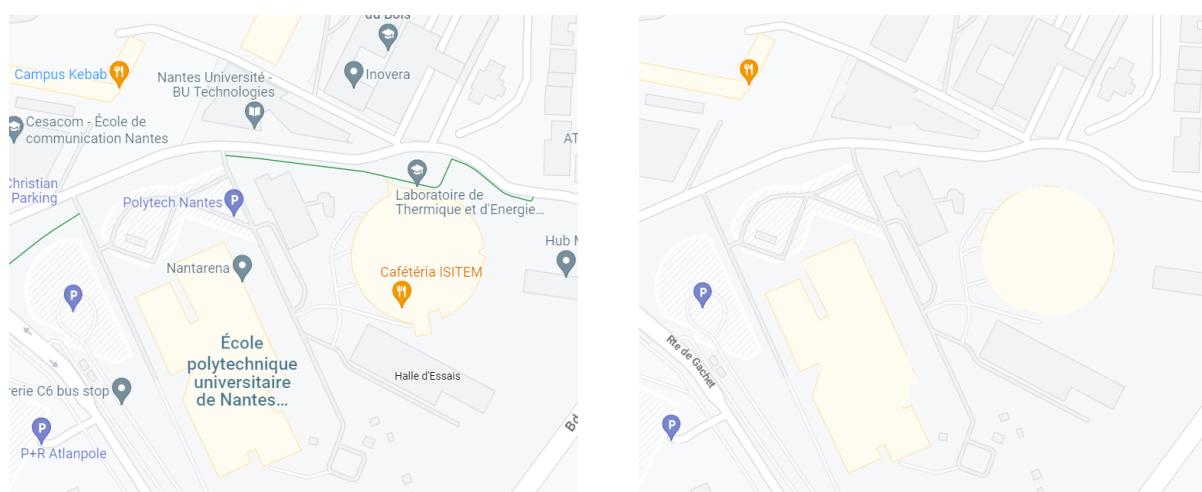
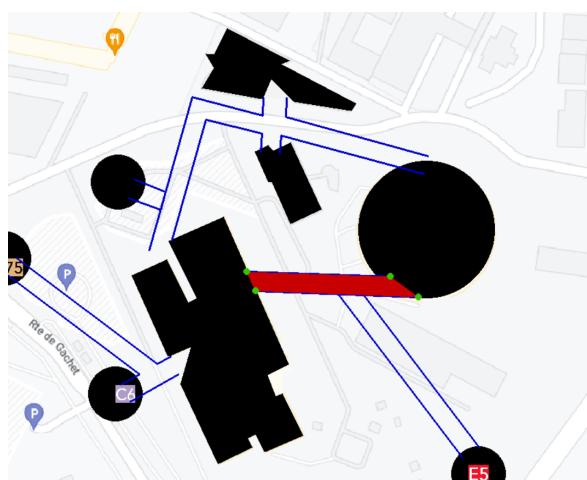


Figure 8 : Plan du campus avant et après modification

Sur cette image, nous avons commencé par tracer des bâtiments directement sur notre programme à l'aide de la fonction draw\_polygone. Au départ, nous avons approché les allures de Ireste, Isitem, IHT et du restaurant universitaire, par de simples rectangles, triangles et cercles. Nous avons ensuite pu relier ces derniers par des chemins. Chaque

chemin est défini par 4 points et peut être représenté par un quadrilatère comme le montre l'exemple de la figure 9. Il nous a donc fallu déterminer ces points. Pour faciliter les déplacements, nous avons pris la décision de faire des chemins droits, parallèles et tous de même largeur. Les bords de nos chemins sont donc modélisés par des fonctions linéaires. Cela nous a notamment permis de simplifier la localisation des étudiants au sein du campus (voir 4.III.A. Nouveautés et changements). Nous avons d'abord localisé sur notre plan les endroits où nous voulions que nos chemins passent en déterminant deux points pour chaque chemin. Nous avons ensuite, grâce à de simples lois mathématiques, calculer la pente et l'ordonnée à l'origine de la droite passant par ces points ainsi que les droites parallèles à celle-ci distantes de 25 pixels (distance calculée avec la norme euclidienne). Nous avons cherché les points d'intersection de chaque chemin avec les bâtiments et chemins adjacents. Finalement, nous avons obtenu l'ensemble des points définissant nos chemins, que nous avons stockés dans un tableau, p. Le nombre de chemins étant important, nous avons utilisé une boucle for associée à la fonction draw.line, pour tracer une ligne d'un point  $p[i]$  à un point  $p[i+1]$ . Pour satisfaire un besoin esthétique, nous avons utilisé une condition "if" afin que certaines lignes ne soient pas tracées.



*Figure 9 : Plan du campus avec, en vert, les 4 points définissant le chemin rouge*

La base de notre interface était ainsi créée. Nous avons pu l'améliorer en traçant les polygones représentant les quatre bâtiments plus fidèlement en ajoutant des points. Une de nos volontés était également que la taille de la fenêtre de notre modélisation puisse s'adapter à la taille de l'écran de chaque utilisateur. Pour cela, nous avons dû modifier les dimensions de tous nos tracés en créant des facteurs dépendants de la hauteur et de la longueur de l'écran ainsi que des dimensions de notre plan d'origine. Nous avons appliqué ce facteur à l'ensemble des coordonnées associées à nos bâtiments et chemins. Après avoir représenté les chemins et bâtiments, nous avons ajouté en tant qu'image les différents arrêts de bus présents sur le campus. Finalement, notre interface comporte 4 bâtiments, 11 chemins et 4 zones d'arrivées pour les individus modélisés : le C6, le 75, le E5 et le parking comme on peut le voir sur la figure 9.

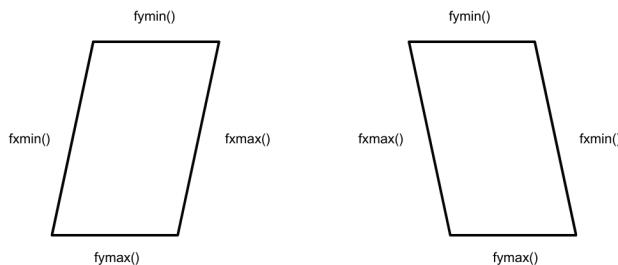
Une des étapes la plus longue dans la création de notre interface était de repérer les coordonnées de chaque point afin de réaliser les tracés de toutes nos figures. De plus, faire en sorte que nos chemins soient tous de la même largeur fut plus long que nous ne l'avions escompté. Bien que les mathématiques mises en jeu soient assez simples, cela nous a pris plus de temps que nous le pensions d'effectuer l'ensemble des calculs des coordonnées de nos chemins.

### III. Seconde modélisation

Après avoir fait cette première modélisation, nous l'avons adapté pour modéliser l'évolution de la peste au sein du campus de la Chantrerie. Nous allons expliquer ici les différentes adaptations que nous avons dû faire. Après nous être renseignées auprès de Mme CAPALDI et Mme GUENEGO pour connaître le nombre d'individus présents sur le campus, nous sommes parvenues au résultat d'environ 1200 étudiants et personnels sur le site. Cependant, nous avons d'abord choisi de commencer les tests de notre modélisation avec 500 individus pour ne pas surcharger le programme.

#### A. Nouveautés et changements

Premièrement, pour cette nouvelle modélisation il a fallu une interface graphique, présentée plus haut. Ceci implique que les points restent à des endroits définis par l'interface tels que les chemins ou les bâtiments, nous avons donc dû trouver un moyen pour cela. Nous avons fait le choix de définir chaque surface rectangle, tels que les chemins et les bâtiments (sauf Isitem), par quatre fonctions affines, pour les quatre côtés : voir la figure 6. Il faut préciser que même si certaines de ces fonctions s'appellent `fxmax()`, elles renvoient une position en y, et prennent en entrée une position en x.

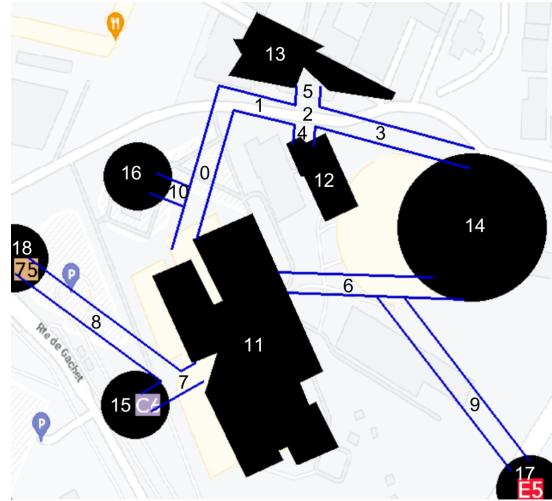


*Figure 10 : Schéma de définition des fonctions fx() et fy()*

Après avoir défini ces quatre fonctions pour toutes les surfaces, nous avons défini pour chacune une fonction `verifsurface()`, par exemple `verifchemin1()` ou bien `verifireste()`. Ces fonctions vérifient trois conditions : si la position x de l'habitant concerné se situe bien entre les abscisses minimum et maximum de la surface, si la position y est bien comprise entre  $fxmin(x)$  et  $fxmax(x)$  et de même si la position y est bien comprise entre  $fymin(x)$  et  $fymax(x)$ . Si toutes les conditions sont vérifiées, la fonction renvoie True. Ces fonctions sont stockées dans un tableau Verif afin qu'elles puissent être appelées facilement, de la même façon nous avons rassemblé les fonctions définissant les rectangles dans un tableau `fxmax`, `fxmin`, `fymax`, `fymin`.

Ces fonctions `verifsurface()` nous ont permis de définir un nouvel attribut `endroit` à la classe habitant, cet attribut est un numéro allant de zéro à dix-huit et correspondant au numéro de la surface dans lequel il est. Il vaut -1 si l'habitant se situe dans aucune surface. La figure 12 permet de voir à quelle surface ces numéros correspondent. Nous avons défini une fonction `endroit()` qui parcourt le tableau Verif, et si une des fonctions renvoie True alors l'attribut `endroit` de l'habitant devient l'indice de la case correspondante. Par exemple, la fonction parcourt le tableau Verif et si c'est la case 0 qui renvoie True cela veut dire que c'est la fonction `verifchemin1()` qui renvoie True (il y a un décalage dans les indices), et l'attribut `endroit` de l'habitant est mis à 0, ce qui veut dire qu'il se trouve dans le chemin partant d'Ireste, vers le haut.

Figure 11 : Plan de Polytech avec les surfaces numérotées



Mais contrairement à la modélisation précédente où les habitants n'avaient pas de contraintes sur leurs déplacements (autre que rester dans la fenêtre), il est important que les habitants (autre que les rats qui ont le droit de se balader partout dans la fenêtre) ne sortent pas des surfaces dans lesquelles ils sont, pour cela nous avons défini une fonction `collmur()`. Cette fonction ramène l'habitant sur les frontières de la surface s'il en est sorti. Si la position en x de l'habitant est plus grande que l'abscisse maximale de la surface, il est mis à cette abscisse; et si elle est plus petite que l'abscisse minimale il est mis dessus. Pour avoir accès aux abscisses maximales et minimales des surfaces, on a stocké dans un tableau points les coordonnées des quatre coins de chaque surface. Ainsi quand on appelle le tableau, le premier indice est le numéro de la surface, auquel on accède avec l'attribut `endroit` de l'habitant, et le deuxième indice permet d'accéder aux quatre coins de cette surface. Ensuite, si la position y est plus grande que `fxmax(x)` ou `fymax(x)`, elle est mise à cette valeur, et si elle est inférieure à `fxmin(x)` ou `fymin(x)` de même. La figure 13 illustre le fonctionnement de la fonction `collmur()`, dans le cas de la position y, pour deux cas différents.

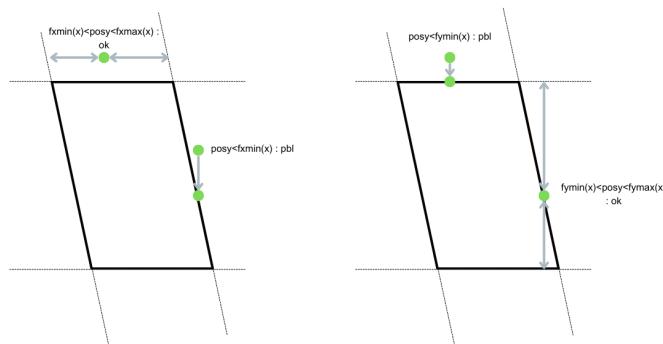
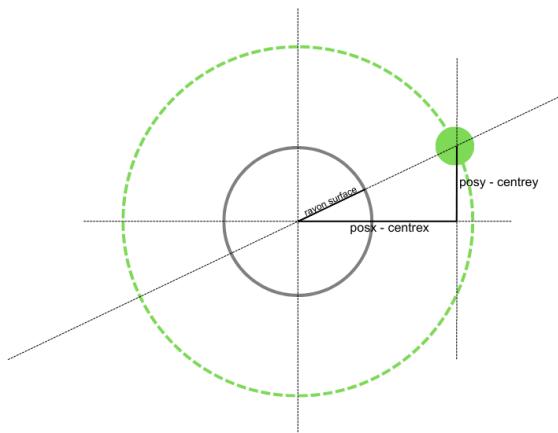


Figure 12 : Schéma du fonctionnement de la fonction `collmur` pour la position y d'un habitant

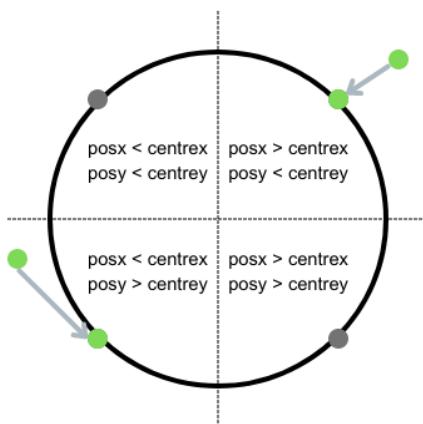
Tout ce qui a été dit plus haut a été principalement détaillé pour les surfaces rectangles, cependant plusieurs de nos surfaces sont circulaires. Ces surfaces ont été

définies différemment, tout d'abord les fonctions `fxmax()`, `fxmin()`, `fymax()` et `fymin()` n'ont pas lieu d'être. Cependant la fonction `verifsurface()` existe bien pour ces surfaces, au lieu des trois conditions mentionnées plus haut elle vérifie si l'habitant est dans le cercle grâce au théorème de Pythagore. En effet, en reprenant les notations de la figure 14, on calcule la distance entre l'habitant et le centre de la surface :  $(\text{posx} - \text{centrex})^{**2} + (\text{posy} - \text{centrey})^{**2}$ , et si elle est inférieure au rayon de la surface la fonction renvoie True. Ces fonctions sont stockées dans le même tableau `Verif` que celles mentionnées plus haut.



*Figure 13 : Schéma du fonctionnement des fonctions `verifsurface()` pour des cercles, ici l'habitant n'est pas dans la surface*

L'équivalent de la fonction `collmur()` pour ces surfaces est la fonction `dansisitem()` (car au début il n'y avait qu'`lsitem` qui était rond). De la même manière, si l'habitant sort de la surface, il est ramené sur la frontière. Les habitants sont ramenés en quatre points différents selon le coin duquel ils sont sortis, ces quatre points sont répartis autour de la surface, ils correspondent à des angles de  $\pi/4$  radians, la figure 14 illustre cela.



*Figure 14 : Schéma du fonctionnement de la fonction `dans isitem()`*

Sur la figure 14, on remarque la notation `centrex` et `centrey`, comme sur la figure 13, cela correspond aux coordonnées x et y des centres des surfaces. Ces informations sont stockées dans un tableau `infocercles`, qui comporte autant de cases que le nombre de surface totale, pour que les indices correspondent, les premières cases sont donc remplies de 0, mais cela permet qu'`lsitem` se situe toujours à l'indice 14, comme dans le tableau `Verif`.

Les cases correspondant à des surfaces rondes, contiennent les coordonnées des centres de ces surfaces et leur rayon. De la même manière que pour le tableau points, on peut accéder aux cases en utilisant l'attribut endroit de l'habitant, puis en précisant l'indice 0 si on veut les coordonnées du centre ou 1 pour le rayon.

Un autre point important de cette modélisation est que nous voulions qu'elle soit réaliste. Tout d'abord la population du campus n'est pas répartie de manière égale entre les différents bâtiments. Nous avons essayé de représenter cela en incluant un nombre d'habitants maximum autorisé par bâtiment, ces nombres reflètent environ les résultats de notre sondage qui sera détaillé plus bas, ces résultats sont : 50% des personnes évoluant sur le campus de la Chantrerie passent la plupart de leur temps dans le bâtiment Ireste, 30% en Isitem, et 20% en IHT. Ces données sont rentrées dans un tableau autorisation, dont chaque case correspond à une surface et chacune contient le nombre de personnes autorisées dans la surface. Ce nombre est calculé comme étant un pourcentage du nombre total d'habitant moins le nombre de rats.

On a programmé une fonction calculprop() qui remplit un tableau prop avec le nombre d'habitant dans chacune des surfaces, on parcourt donc le tableau Verif et si l'une des fonctions renvoie True on ajoute un à la case correspondante du tableau prop.

En plus de cette fonction on définit la fonction verifprop(). Cette fonction parcourt tous les habitants et les surfaces et si la case du tableau prop où l'habitant se situe est supérieure à la case du tableau autorisation, la case du tableau stop associé devient False. Chacune des cases du tableau stop correspond à une surface et contient un booléen qui vaut False tant qu'il n'y a pas trop d'habitants dans la surface, puis True. Ensuite le tableau prop est remis à zéro, puis pour chacun des habitants on inverse leur attribut endroit et preendroit, et on appelle les fonctions collmur() et dansisitem() pour mettre les habitants au bon endroit. Il faut après recalculer le tableau prop, et si maintenant la case de prop est inférieure à celle de autorisation alors la case de stop redevient False. Ainsi si un habitant était dans le chemin zéro et entre dans Ireste, mais qu'il y a déjà le nombre de personnes maximum dans le bâtiment, alors l'habitant est renvoyé dans le chemin zéro.

Nous avons introduit l'attribut preendroit, cet attribut est comme l'attribut endroit mais au lieu d'indiquer la surface dans laquelle l'habitant se situe, il indique la surface dans laquelle l'habitant était avant celle-ci. Cet attribut est mis à jour dans la fonction endroit et initialisé à -1, comme l'attribut endroit. Quand une fonction du tableau Verif renvoie True, si son indice est différent de l'attribut endroit actuel de l'habitant, alors preendroit devient endroit et endroit est mis à jour, mais si l'indice est le même que l'endroit actuel alors on ne change rien.

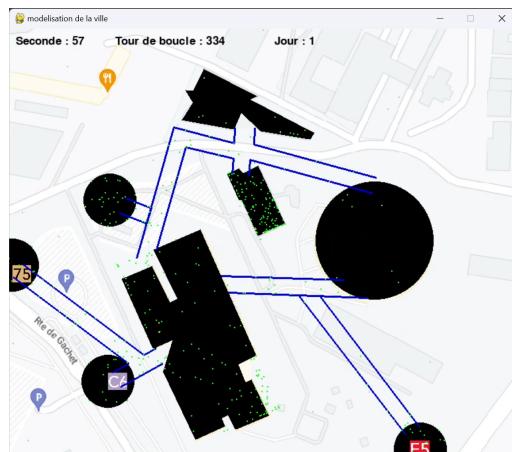
De plus, pour être encore plus réaliste, il faut que les habitants aillent au RU seulement le midi, et que l'on voit le passage des journées.

En ce qui concerne le RU nous avons essayé différentes choses, notre première idée était d'autoriser les habitants à y aller qu'entre midi et 14h à l'aide du tableau autorisation. C'est-à-dire que la case correspondant au RU (la 12ème du tableau autorisation) vaut 0 la plupart du temps, sauf quand le nombre de tour de boucle auquel on est correspond au milieu de la journée. Cependant ne faire que ça ne suffisait pas car bien qu'ils aient le droit d'aller dans ce bâtiment entre ces moments, ce n'est pas pour ça que les habitants y allaient, ou alors que quand certains passaient par là. Nous avons donc rajouté autre chose.

A chaque tour de while, pour tous les habitants, le programme vérifie si le tour de boucle auquel on est (valeur de test) est compris entre n2 fois un temps attente décider (temps de début de l'ouverture du RU), et n2 fois ce temps plus un autre temps décidé (temps pendant lequel le RU reste ouvert). Si c'est bien le cas, la case du RU du tableau autorisation ne vaut plus zéro mais un demi de la population totale (moins les rats), puis on tire un nombre aléatoire entre zéro et un. Puis si le nombre aléatoire un inférieur à un facteur décidé, que l'attribut ru de l'habitant vaut False et que l'habitant n'est ni un rat (gris) ni mort (noir), alors il est transporté dans le RU, ces positions sont mis à des coordonnées se situant dans le RU, et son attribut endroit est mis à celui du RU (douze).

Puis si le tour de boucle où on se situe est supérieur au temps d'ouverture du RU alors la case du tableau autorisation est remise à zéro. Et de la même manière que pour la variable n qui permettait de faire mourir les rats, on incrémente n2 de deux, pour que le RU soit ouvert tous les multiples du temps d'attente du RU. On ajoute deux car comme le temps d'attente du RU correspond à la moitié de journée, si on faisait plus un on se retrouverait à la fin de la journée et pas à la moitié de la journée suivante.

Nous avons ici introduit l'attribut ru de la classe habitant, cet attribut permet de savoir si l'habitant est déjà dans le RU ce jour. Il est initialisé à False et si l'habitant est transporté dans le RU il vaut True, ainsi l'habitant ne sera pas transporté au centre du RU à chaque tour de boucle, mais pourra continuer ses mouvements aléatoires dans le RU ou ailleurs s'il en sort. Cet attribut est remis à False à chaque fin de journée.



*Figure 15 : Illustration de notre modélisation : Le midi (du 302ème tour de boucle au 367ème pour le 1<sup>er</sup> jour) une partie des habitants est au RU*

Enfin il nous reste à faire en sorte que le passage des journées soit visible, pour cela nous avons fait le choix qu'à chaque nouveau jour, les habitants redémarreront à leur point de départ. Pour cela, à chaque tour de boucle while et pour chaque habitant on vérifie si le tour de boucle actuelle (la valeur de test) est égal à n3 fois le nombre de tour de boucle pour un jour (la valeur de nbTBPJ). Si c'est le cas et que l'habitant est ni mort (noir) ni un rat (gris) les positions de l'habitant sont remises à celle de départ, grâce au tableau depart, et l'attribut ru est mis à False. Enfin on n'oublie pas d'incrémenter n3 d'un pour que tous les jours ces opérations se répètent. On affiche les jours qui défilent dans la fenêtre mais attention ils commencent à zéro.

Le tableau depart est un tableau qui contient pour chaque habitant (sauf les rats), ses coordonnées de départ. Il est défini en compréhension, et utilise les pourcentages du sondage, soit 17% des habitants démarrent au E5, 39% au parking, 41% au C6 et les 3% restant au 75. Mais faire arriver pleins de points sur le même pixel n'est pas vraiment possible, nous avons donc défini de nouvelles surfaces : des cercles autour de ces points d'arrivée, ils sont visibles sur la figure 11. Ces cercles sont régis de la même façon et par les mêmes fonctions qu'ilsitem.

La variable nbTBPJ (nombre de Tour de Boucle Par Jour) a été introduite car contrairement à la première modélisation qui comptait 9 tours de boucle par seconde et donc par jour, nous avons ici testé différentes valeurs de tour de boucle par jour. Cette variable vaut 216 (valeur arbitraire,  $24*9$ ) et sert donc à multiplier les temps d'attente définis en jours pour avoir des tours de boucle, et sert aussi à compter et passer les jours.

## B. Déroulement du programme

Comme pour la première modélisation nous allons maintenant décrire le programme étape par étape. Il est divisé en 4 fichiers, le fichier modélisation qui contient les fonctions de la première modélisation ainsi que l'initialisation des variables qui sont les mêmes :

## 5. Modélisation et Analyse

- val\_aleatoire = 30 # correspond au déplacement max des points d'un instant à l'autre
- rayon\_cercle = 1
- n = 1
- test = 0 # nombre de tours de boucle depuis que la fenêtre est lancée
- 
- # nombre d'habitants au départ
- nbvert, nbjaune, nbrouge, nbrat, nbmort = 500, 0, 0, 25, 0
- nbtot = nbvert+nbjaune+nbrouge+nbrat+nbmort
- 
- # facteurs de chance (ex : factjr = facteur de jaune en rouge) compris entre 0 et 1
- factjr = 0.35
- factjv = 0.8
- factjn = 0.55
- factrn = 1
- factrv = 0.8
- factgn = 1
- factvr = 0.6
- factgvj = 0.25
- factru = 0.1
- # temps d'attente (ex : tattjr = temps d'attente de jaune en rouge) en tour de boucle
- nbTBPJ = 216 # nb de tour de boucle par jour
- tattjr = int(3.5\*nbTBPJ)
- tattjv = int(1.0\*nbTBPJ)
- tattjn = int(5.0\*nbTBPJ)
- tattrn = int(2.0\*nbTBPJ)
- tattrv = int(1.0\*nbTBPJ)
- tattgn = int(2.4\*nbTBPJ) # au lieu de 3.5 pour avoir 60 jours
- tattRUinf = int(0.5\*nbTBPJ)
- tattRUDedans=int(0.2\*nbTBPJ)
- n2 = 1
- n3 = 1

Puis il y a le fichier classes, qui existait aussi pour la première modélisation, mais celui-ci a les attributs endroit, preendroit, ru en plus. Dans le fichier fonctions les fonctions sont définies, et dans le fichier plan on retrouve le centre de notre programme. C'est ce fichier que nous allons détailler et qu'il faut lancer pour avoir la modélisation.

Tout d'abord la fenêtre pygame est initialisée, les images tels que celle des bus sont chargés, le tableau p est initialisé avec toutes les coordonnées des points des différentes surfaces, le nombre de surfaces (nbsurface) est initialisé à 19, les tableaux Verif, fxmax, fxmin, fymax, fymin, infocercles, points et stop sont initialisés. Puis les tableaux autorisation et départ sont initialisés comme suit :

```
autorisation = [50/100*(nbtot-nbrat-nbmort) if i == 11 else 0 if i == 12 else
20/100*(nbtot-nbrat-nbmort) if i == 13 else 30/100*(nbtot-nbrat-nbmort) if i == 14 else
nbtot+1 for i in range(nbsurface)]
```

```
depart = [[infocercles[17][0][0],infocercles[17][0][1]] if i < 17/100*(nbtot-nbrat-nbmort) else
[infocercles[16][0][0], infocercles[16][0][1]] if 17/100*(nbtot-nbrat-nbmort) <= i <
(17+39)/100*(nbtot-nbrat-nbmort) else [infocercles[15][0][0],infocercles[15][0][1]] if
(17+39)/100*(nbtot-nbrat-nbmort) <= i < (17+39+42)/100*(nbtot-nbrat-nbmort) else
[infocercles[18][0][0],infocercles[18][0][1]] if (17+39+42)/100*(nbtot-nbrat-nbmort) <= i <
(nbtot-nbrat-nbmort) else [hauteur/2,hauteur/2] for i in range(nbtot)]
```

Ensuite comme dans la première modélisation, on initialise les habitants, pour cela on parcourt le nombre total d'habitants, et on crée un objet de la classe habitant, de couleur blanche au début, et avec comme attribut endroit et preendroit égal à moins un, pour chacun d'eux, et on remplit le tableau Habitant avec tous ces objets. Puis on parcourt le tableau Habitant, couleur par couleur en utilisant les variables nbvert, nbroute,... et pour chacun on attribue la bonne couleur et la bonne position selon le tableau départ. Et pour les rats, leur position est mise au milieu de la fenêtre.

Ensuite la boucle while commence, elle se finit si la croix est cliquée ou s'il n'y a plus de rats. On appelle la fonction draw\_window() qui dessine le plan de Polytech, et on affiche le nombre de tours de boucle et de jour. Le tableau prop est initialisé à zéro. Puis on rentre dans la boucle "for i in range(nbtot):", on tire un nombre aléatoire x et y, et on fait différentes vérifications. Si le tour de boucle se situe au moment du midi, on fait ce qui a été décrit plus haut au sujet du RU, si le tour de boucle est supérieur au midi on remet la case RU du tableau autorisation à zéro, si le tour de boucle correspond à la fin de la journée l'habitant est remis au départ. Et enfin soit l'habitant est mort (noir) et sa position est mise à -1000,-1000 afin qu'il n'apparaisse pas sur la carte, soit on additionne x et y à sa position actuelle. Puis on appelle la fonction endroit() pour mettre à jour les attributs endroit et preendroit, et les fonctions collmur() ou dansisitem(), selon la surface dans laquelle l'habitant se situe, pour vérifier sa position. Ensuite, si l'habitant est un rat (gris) on vérifie qu'il ne sorte pas de la fenêtre.

Comme pour la première modélisation on appelle les fonctions VerifTempsSup(), VerifTempsInf() et rat() pour les différentes couleurs. On remplit ensuite le tableau couleur d'un 'r' dans la case de l'habitant s'il est rouge, on appelle la fonction calculeprop() pour remplir le tableau prop, et on dessine les habitants.

Puis on sort de la boucle for et on appelle la fonction verifprop() pour pas qu'il y ait trop d'habitants dans les surfaces, puis les fonctions collision() et collision2() avec les bonnes couleurs (voir la première modélisation). Ensuite si le temps du RU est dépassé, on incrémente de deux n2, et si le tour de boucle auquel on est correspond à la fin de la journée, on incrémente d'un n3. Et enfin, on incrémente d'un le nombre de tours de boucle (variable test).

Pour finir, lorsque la fenêtre pygame est fermée, on calcule le  $R_0$  et enregistre les données importantes dans une feuille de calcul, comme expliqué pour la première modélisation.

## IV. Analyse

### A. Première modélisation

#### a. Exposition des résultats obtenus

Après avoir fait tourner le programme plusieurs fois nous avons regroupé tous les résultats dans le tableau suivant :

	A	B	C	D	E	F	G	H	I
1	nbvert départ	nbrat départ	temps de vie rat	nbvert final	nb contamination	nb rouge max	R0	nb de tour de boucle	
2									
3	1000	50	3,444444444	470	2666	978	2,72597137		
4	1000	50	3,444444444	466	2305	967	2,383660807		
5	1000	50	3,444444444	451	2647	980	2,701020408	2328	
6	1000	50	3,444444444	470	2377	968	2,455578512	1743	
7	1000	50	3,444444444	472	2528	965	2,619689119	2036	
8	1000	50	3,444444444	476	2360	961	2,455775234	1834	
9	1000	50	3,444444444	468	2595	968	2,680785124	1825	
10	1000	50	3,444444444	452	2581	977	2,641760491	1662	
11	1000	50	3,444444444	462	2527	976	2,589139344	1951	
12	1000	50	3,444444444	460	2468	974	2,533880903	1578	
13	1000	50	3,444444444	447	2719	982	2,768839104	1553	
14	1000	50	3,444444444	452	2543	978	2,600204499	1553	
15	1000	50	3,444444444	456	2475	967	2,559462254	1553	
16	1000	50	3,444444444	455	2419	977	2,475946776	1553	
17	1000	50	3,444444444	439	2635	968	2,722107438	1553	
18	1000	50	3,444444444	475	2536	969	2,617131063	1553	
19	1000	50	3,444444444	460	2593	975	2,659487179	1553	
20	1000	50	3,444444444	477	2375	975	2,435897436	1553	
21	1000	50	3,444444444	491	2378	956	2,487447699	1553	
22	1000	50	3,444444444	466	2355	978	2,40797546	1553	
23	1000	50	3,444444444	468	2470	971	2,54376931	1553	
24									
25				moyenne	moyenne	moyenne	moyenne		
26				463,4761905	2502,47619	971,9047619	2,574549025		
27				46% de vivants					
28									

Figure 16 : Tableau des résultats du premier programme

Voici donc les résultats obtenus en ayant fait tourner 21 fois le programme. Ceux-ci sont faits sur la base de trois paramètres fixes : 1000 personnes saines au départ et 50 rats infectés qui disparaissent progressivement et qui ont un temps de vie de 3.44 jours, donc 31 tours de boucle (c'est-à-dire qu'un rat meurt tous les 31 tours de boucle).

#### b. Analyse et commentaire

Les résultats les plus pertinents et compréhensibles sont les moyennes calculées en bas du tableau de la Figure 16. A la fin de notre modélisation, nous obtenons, en moyenne, 46% de personnes ayant survécu à l'épidémie de peste. Ce chiffre est certes un peu faible comparé à ceux des pestes les plus récentes mais cela nous paraît plutôt cohérent si nous gardons en tête qu'ici aucune mesure n'est prise pour ralentir l'épidémie et que les points peuvent se déplacer comme ils le veulent sur l'écran (sans suivre de chemin). Une maladie fulgurante comme la peste qui provoque la mort d'un peu plus de la moitié de la population initiale reste probable.

Si nous regardons maintenant le nombre de contaminations, il s'élève en moyenne à 2502. Si au premier abord il peut paraître élevé étant donné qu'il n'y a que 1000 personnes dans notre population de départ, il peut s'expliquer par le fait qu'une personne n'est pas immunisée après avoir contracté la peste et si elle guérit une première fois, elle peut de ce fait attraper la maladie une seconde fois. Le nombre de contaminations correspond ici aux points verts passant rouges par contact avec ces derniers. Ce nombre est donc élevé mais

les points sont quand même confinés dans un espace très restreint au vu de leur taille et du nombre qu'ils sont.

Passons maintenant au nombre de rouge max qui correspond au nombre total de personnes atteintes de la peste pulmonaire lors d'une simulation (qui ont été rouge à un moment). En moyenne, il est de 972 personnes, mais ce chiffre ne prend pas en compte le fait qu'une personne peut être deux fois atteinte de la peste pulmonaire, cela insinue donc que la contraction d'une forme pulmonaire est presque inévitable. Cependant en se rappelant du nombre de survivants qui se situe autour des 463 personnes, on se rend compte que notre taux de guérison est efficace et reflète bien les progrès de la médecine aujourd'hui si toutefois le patient est pris en charge assez rapidement. En effet les habitants sont très atteints de la peste pulmonaire mais y survivent plutôt bien.

Nous avons aussi trouvé pertinent de calculer notre propre  $R_0$  à chacune de nos simulations. Il est calculé en faisant la division du nombre de contaminations par le nombre de contaminants (donc le nombre total de personnes atteintes de la peste pulmonaire). Sa valeur moyenne est de 2.57, comparé au  $R_0$  des précédentes épidémies de peste se situant vers les 1.20 (qui ont été très difficiles à trouver lors de nos recherches) [37] celui-ci est un peu trop élevé. Nous pouvons donc en conclure qu'il y a un peu trop de contaminations lors de notre simulation mais cela n'est pas surprenant en tenant compte du manque d'action pour ralentir l'épidémie, aujourd'hui, lors d'une telle épidémie, la population serait certainement confinée et les malades mis en quarantaine afin de stopper l'épidémie.

On peut donc conclure que notre modèle est assez réaliste car il y a presque 50% de survivants et le  $R_0$  est assez proche de ceux connus. Mais nous sommes conscients qu'il a des limites, par exemple du fait que nous ne nous appuyons pas sur des équations mathématiques, mais seulement sur une modélisation graphique.

## B. Deuxième modélisation

### a. Exposition des résultats obtenus

En ce qui concerne les résultats de notre deuxième modélisation qui est donc notre simulation sur le campus de la Chantrerie, voici ce que nous avons obtenu :

	A	B	C	D	E	F	G	H	I	J	K
40	nbvert départ	nbrat départ	temps de vie rat	nbvert final	nb contamination	nb rouge max	R0	nb tour de boucl	Xtourdeboucle=	nb jaune max	stat finale
41	500	25	518	5	10388	500	20,776	12953	24		
42	500	25	518	4	15059	500	30,118	12953	24		
43	500	25	518	8	16331	500	32,662	12953	24		
44	500	25	518	4	12697	500	25,394	12953	24		
45	500	25	518	7	17071	500	34,142	12953	24		
46	500	25	518	9	13075	500	26,15	12953	24		
47	500	25	518	6	12098	500	24,196	12953	24		
48	500	25	518	6	11293	500	22,586	12953	24	18	
49	500	25	518	3	16675	500	33,35	12953	24	18	
50	500	25	518	5	13961	500	27,922	12953	24	23	
51	500	25	518	3	16440	500	32,88	12953	216	35	
52	500	25	518	5	11198	500	22,396	12953	216	24	
53	500	25	518	4	12103	500	24,206	12953	216	15	
54											
55			moyenne	moyenne		moyenne			moyenne		
56	500	25	518	6,714285714	14113,35714	500	28,22671429	12052,64286	216	19,42857143	

Figure 17: Tableau des résultats du second programme

Sur la figure 17, le nombre de personnes saines initialement présentes sur le campus est de 500, elles seront en compagnie de 25 rats infectés qui meurent un par un tous les 518 tours de boucles, ce qui correspond à 2.4 jours, ainsi la modélisation se termine au bout de deux mois. Cette fois-ci, le programme mettant environ 35 minutes à s'exécuter entièrement, nous nous baserons sur les statistiques de 13 runs.

## b. Analyse et commentaire

Comme on peut le constater, ici les résultats sont nettement moins concluants car nous y voyons des nombres drastiquement plus extrêmes.

Commençons par le nombre de survivants moyen : 7 sur 500. C'est très peu et nous en sommes conscients. Nous pensons que cela est dû au déplacement des points qui est maintenant restreint par des chemins ou des bâtiments. Nos points sont désormais confinés dans de plus petits espaces et cela implique que les contacts augmentent, donc la propagation augmente, et avec elle la mortalité liée à la gravité de la peste.

Cela se constate encore lorsque l'on regarde le nombre de contaminations qui dépasse les 14000 en moyenne. On remarque aussi que chaque personne contractera la peste sous la forme pulmonaire sans exception, et de toute évidence au vu des chiffres, certains attraperont la bactérie plusieurs fois. C'est alors qu'on se rend vraiment compte du manque de réalisme de notre modélisation. En effet, dans ce programme nous ne prenons pas en compte le fait qu'une personne malade qui se rend compte de ses symptômes restera normalement chez elle. Dans notre modélisation, cette personne reviendra à chaque fois sur le campus. De plus, comme la plupart de nos habitants vont manger au RU tous les midis et reviennent à leurs points de départs (arrêts de bus et parking) tous les jours, cela les oblige à être dans des espaces encore plus restreints et donc à se toucher, ce qui augmente considérablement le nombre de contaminations.

Le  $R_0$ , comme les autres résultats à analyser, est beaucoup trop élevé puisqu'il atteint les 19.42 en moyenne. A titre de comparaison, le  $R_0$  de la varicelle, maladie extrêmement contagieuse notamment chez les enfants, est situé entre 10 et 12 [38], [39]. On se rend donc bien compte que ce  $R_0$  est irréaliste, surtout si on le compare à celui plutôt cohérent trouvé avec notre première modélisation. Cela s'explique par les nombres, énormes, de contaminations et d'habitants contractant la peste pulmonaire (rouge), qui nous permettent de calculer ce  $R_0$ .

Ces chiffres se remarquent en regardant la modélisation. En effet, les premiers jours, tout ce déroule conformément à nos prédictions, les rats contaminent les habitants sains, qui attrapent la peste bubonique et on remarque donc une proportion normale d'habitants jaunes qui apparaissent peu à peu. Cependant à partir du troisième jour et demi la peste bubonique évolue en peste pulmonaire. Ainsi les premiers habitants rouges apparaissent, mais comme leurs déplacements sont confinés dans un espace très restreint, ils contaminent très vite les habitants verts et dès le quatrième jour presque tous les habitants sont rouges (figure 18). Or comme les habitants atteints de la peste pulmonaire ont la possibilité de guérir, jusqu'à la fin de la modélisation ils alternent entre l'état sain (vert) et l'état de peste pulmonaire (rouge). On voit donc que tout se joue dès les premiers jours et que les chiffres élevés de contaminations et d'habitants rouges s'expliquent par ces comportements.

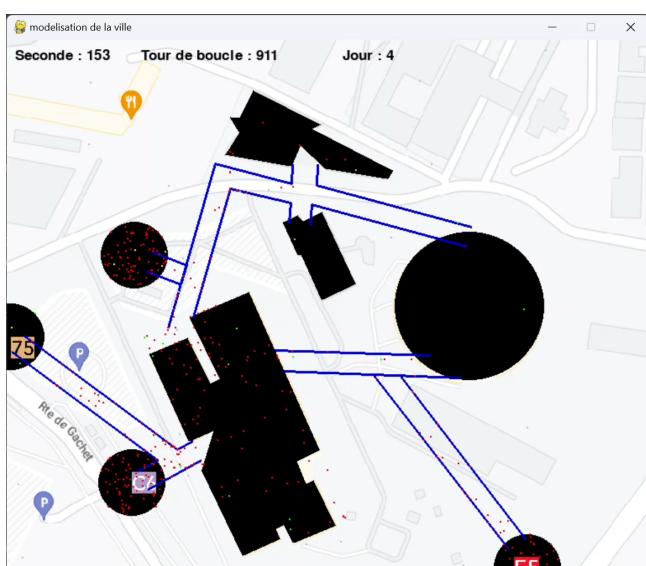


Figure 18 : Illustration de notre modélisation : au 4ème jour, presque tous les habitants sont rouges

Malgré ces résultats non cohérents nous n'avons pas changé les taux que nous avions fixés lors de la première modélisation, car nous les avons choisis d'après nos recherches et nous pensons qu'ils sont le plus proches possibles de la réalité. Cependant, nous avons réduit la distance nécessaire entre deux points pour qu'il y ait collision par rapport à la première modélisation. En effet, comme expliqué plus haut, elle était dans la première modélisation de quatre fois le rayon des points, elle est maintenant égale au rayon des points. Ainsi nous souhaitons être plus proches de la réalité où les individus doivent être très proche pour se contaminer et nous espérions diminuer un peu le nombre de contaminations. Nous avons également changé la valeur de val\_aleatoire qui correspond au déplacement maximum que les points peuvent effectuer à chaque tour. Au lieu d'être de 20 pixels, elle est maintenant de 30. Nous ne voulions pas non plus que cette valeur soit trop grande sinon cela donnerait l'impression que les points se téléportent d'un endroit à l'autre. Augmenter cette valeur avait pour but que les points se déplacent plus rapidement et qu'ils aient moins de chance de se toucher.

Ces résultats n'étant pas concluant, nous n'avons pas testé d'augmenter le nombre de points de la modélisation à 1200 qui serait plus proche du nombre réel d'individus présents sur le campus. En effet, augmenter le nombre de points augmentera le nombre de collisions donc de contamination d'autant plus que les points sont confinés.

Ainsi il est normal d'avoir un regard critique concernant notre deuxième modélisation. Etant donné que nous savions qu'il y avait plusieurs facteurs que nous n'avions pas pris en compte pour simplifier le problème, nous nous doutions que les résultats perdraient en pertinence. Cependant, ceux-ci sont quand même bien loin de la réalité et sont beaucoup plus extrêmes que prévu. Cette deuxième modélisation a donc de multiples limites. La première et sans doute la plus importante, est que contrairement à la première modélisation les points sont ici confinés dans des espaces (chemins et bâtiments) beaucoup plus restreints par rapport à leur taille et sont donc obligés de se toucher les uns les autres de manière très fréquente. De plus, peu de mesures sont prises pour restreindre l'épidémie. Les personnes atteintes de la peste ne rentrent pas chez elles, aucunes mesures de distanciation sociale n'est appliquée et la seule mesure mise en place est le traitement des patients malades. Dans la réalité, si cette situation se présentait, l'école serait rapidement fermée et toutes les personnes présentes sur le campus seraient confinées chez elles.

### C. Comparaison aux modèles déjà existants

L'une des premières remarques que l'on peut faire vis-à-vis de notre modélisation, est le fait que nous avons considéré que tous les rats lâchés dans le campus de la Chantrerie sont tous contaminés. Or, dans tous les modèles présentés dans ce rapport, la propagation de la peste au sein de populations de puces et de rats est modélisée. Nous avons fait le choix de négliger cet aspect, car nous n'avions pas les connaissances suffisantes en biologie pour pouvoir modéliser la propagation de la maladie au sein des populations de puces et de rats de manière réaliste.

D'autre part, notre simulation ne représente qu'une seule espèce de rongeurs : le rat. Nous n'avons pas pris en compte le fait qu'il existe probablement d'autres espèces de rongeurs présentes sur la Chantrerie.

De plus, bien que les modèles présentés précédemment visent à simuler la propagation de la peste, ceux-ci se concentrent davantage sur la façon dont elle se propage chez les puces et les rongeurs, pour en déduire ensuite la manière dont elle se propagera chez les êtres humains. En comparaison, notre modèle se focalise davantage sur la propagation chez les êtres humains et la façon dont la maladie va évoluer suite à la contamination d'un être humain.

Par ailleurs, la plupart des simulations de peste existantes modélisent une épidémie de peste s'étant réellement produit, comme par exemple le modèle de réaction-diffusion de

## 5. Modélisation et Analyse

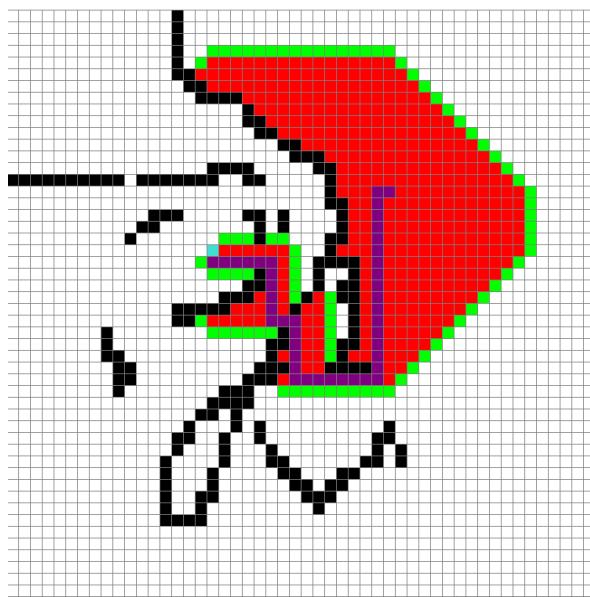
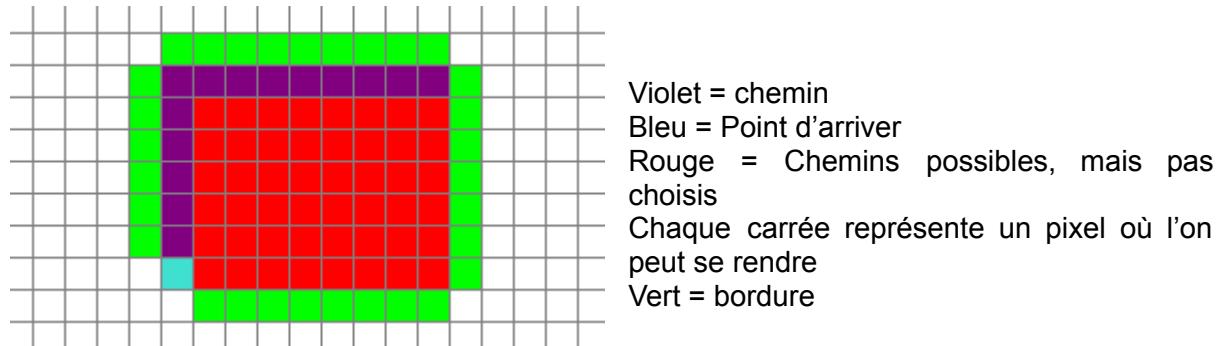
Noble qui simule l'épisode de peste de Bombay, ou encore, le modèle SIMPEST, qui modélise l'épidémie de peste à Madagascar. De notre côté, nous avons voulu simuler un épisode de peste fictive à la Chantrerie. La recherche de données pour écrire notre programme de simulation était d'autant plus compliquée.

Enfin, les modèles présentés précédemment sont basés sur des équations mathématiques comme par exemple le modèle à métapopulations de Keeling et Gilligan ou encore le programme Plaguesirs où les taux de contact sont décrits avec des matrices. Pour notre simulation, nous ne nous sommes pas basés sur des modèles mathématiques particuliers pour programmer la propagation de la peste au sein de la Chantrerie

## 6. Résultats complémentaires

### I. Algorithme de recherche de chemin

Pour simuler des mouvements humains de manière plus réaliste, nous avons décidé de créer un algorithme de recherche de chemin afin de déplacer nos points de manière déterministe plutôt qu'aléatoire. Le premier algorithme qui nous a venu en tête est le « A\* » qui est un algorithme trouvant à chaque fois le chemin le plus rapide. Il se base sur l'algorithme de Dijkstra et rajoute une fonction heuristique qui 'stock' les différents chemins et les compare à la fin afin d'être sûr de trouver le meilleur chemin. (Fichier astar.py)



Ici dans un cas plus complexe on voit qu'il trouve le chemin le plus court (les points noirs sont les obstacles).

*Figures 19 et 20 : Illustrations de l'algorithme*

Malheureusement, cet algorithme pose un problème assez majeur de temps. La fonction heuristique garde en mémoire tous les chemins qui peuvent être le chemin le plus court, or dans certains cas où il y a beaucoup de possibilités de chemin, l'algorithme met énormément de temps à calculer le prochain pixel où le cercle pourra se déplacer. En effet, la complexité temporelle de l'algorithme A\* est «  $O(b^{(d/2)})$  », où  $b$  correspond au nombre moyen de choix possibles pour l'algorithme à chaque étape de la recherche et  $d$  est la profondeur de la solution optimale.

## 6. Résultats complémentaires

Pour résoudre cela, on a décidé de changer d'algorithme et de le simplifier le plus possible afin de le rendre plus efficient. On a supprimé toute la partie fonction heuristique afin de passer sur un algorithme dit 'Glouton', c'est-à-dire un algorithme qui fait que des choix locaux, de pixel en pixel sans aucune remise en cause du chemin. Avec une complexité temporelle de «  $O(b \log b)$  » on a donc ici un algorithme qui sur-performe l'ancien algorithme sur la quasi-totalité des trajets possible de notre modélisation. (Fichier my\_algo.py)

Le code de cet algorithme a une structure en 3 points :

- La création de notre surface de travail : consiste en la définition d'une liste contenant l'ensemble de nos pixels avec leurs coordonnées (fonction creation\_pixels du code).

- L'analyse des prochaines étapes : faites à l'aide d'une fonction qui donne la position de chaque voisin de notre pixel et renvoyant ensuite une liste de ceux où l'on peut se déplacer, elle élimine alors les obstacles ou les bords de la fenêtre (fonctions obstacle et voisins du code).

- Et pour finir, l'algorithme en lui-même : qui vient prendre le pixel de début et d'arrivée et va renvoyer une liste contenant l'ensemble de pixels constituant le trajet optimisé. L'algorithme commence par trier dans l'ordre décroissant les scores des différents voisins, cela veut dire qu'il classe les meilleurs pixels où l'on peut se déplacer, en commençant par le meilleur pour finir par le pire. Le score est défini en fonction de la distance entre ce pixel et le pixel d'arrivée. Cela est fait grâce à la fonction h\_scores, dans notre cas, nous avons choisis une norme euclidienne. Une fois ceci fait, la liste triée des voisins rentre dans une seconde phase : celle qui permet d'éviter de se rendre plusieurs fois au même point pour éviter que l'algorithme se "coince" dans une zone. Une fois tout ceci fait, on renvoie donc un pixel qui est la prochaine étape optimale de notre algorithme. On recommence jusqu'à ce qu'on arrive à notre pixel final.

Lors de la phase de test, tout se passait bien pour un seul étudiant modélisé, mais dès lors qu'on passait à l'ensemble des personnes modélisées le programme plantait car trop lent à exécuter. Nous ne pouvions plus simplifier l'algorithme de recherche de chemin car nous avions déjà choisi une des options les plus optimales.

Face à ce problème, nous sommes venus avec une potentielle idée de pré-calculer tous les trajets possibles au préalable et, dans la modélisation, de chercher dans un grand tableau quel serait le trajet à suivre pour se déplacer d'un pixel A vers un pixel B. Le programme de pré-calcul fonctionnait bien pour une quantité moindre de trajets à faire mais, dans notre cas, cela faisait plusieurs millions de trajets différents à calculer et nous n'avions pas assez de puissance de calcul pour le faire à temps. (Fichier testing.py)

En résumé, nous avons dû abandonner l'idée de faire se mouvoir les points à l'aide d'un algorithme car nous n'avons pas réussi à résoudre le problème de performance du code.

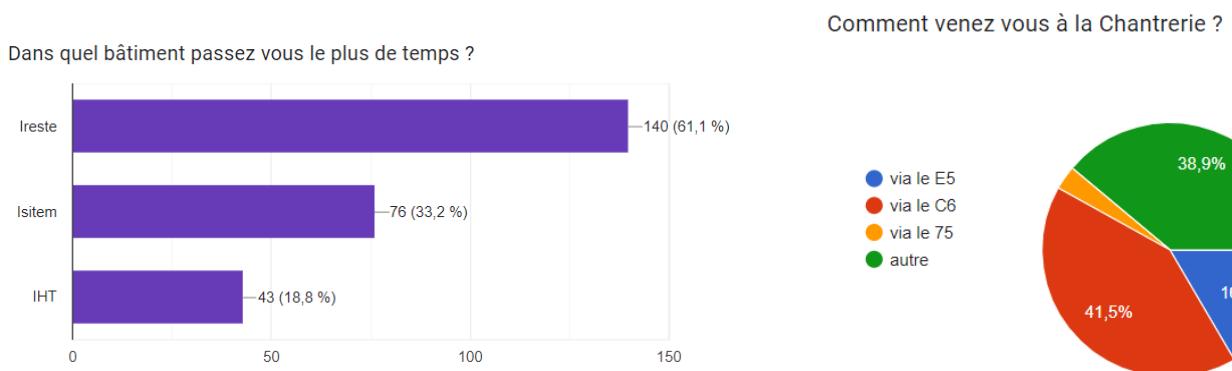
## 7. Aspect gestion de projet

### I. Déroulement

Dès les premières semaines du lancement de la conférence Peip, nous avons décidé de nous réunir afin de permettre une bonne gestion de ce projet. Lors de la première réunion, nous nous sommes rapidement approprié le sujet donné et chaque personne du groupe a dû faire des recherches sur un virus ou une bactérie choisis. Nous avions choisi sept bactéries et virus tels que la peste, la rougeole, la salmonelle, la varicelle, la grippe, le VIH et la pneumonie. Suite à cela, nous nous sommes réunis une semaine après afin de mettre en commun nos recherches. L'idée de base était de modéliser une ville avec plusieurs virus ou bactéries choisis lors de la première phase de recherche cependant, nous avons convenu que c'était un peu trop complexe pour un premier essai. L'idée s'est alors transformée en la modélisation d'un virus, ici la peste, au sein de Polytech. Avec l'accord commun du groupe, nous avons décidé de choisir la peste grâce à son taux de mortalité élevé à l'époque et son impact. Après ce choix, nous nous sommes posé la question des paramètres de la modélisation.

En-dehors de l'aspect bibliographique et de la programmation, d'autres étapes ont été nécessaires dans le déroulement du projet. Tout d'abord, afin que notre modélisation soit la plus réaliste possible, il nous a fallu déterminer le nombre d'individus présents à la Chantrerie à représenter. Pour cela, nous avons contacté plusieurs personnes afin d'obtenir les effectifs des élèves, des professeurs, et du personnel travaillant chaque jour sur le campus. Après plusieurs échanges par e-mail avec les services administratifs, nous avons obtenu ces chiffres. De la même façon, lors de la création de l'interface graphique, nous avons contacté le service technique pour nous procurer les plans du site.

Le campus de la Chantrerie est vaste et en tant qu'étudiants, nous savons que tous les bâtiments ne sont pas fréquentés de la même façon. Pour connaître les habitudes de déplacements de chacun et pouvoir les adapter à notre modélisation, nous avons décidé de créer un sondage. Dans ce questionnaire, nous avons cherché à savoir dans quels bâtiments il y a le plus de passage et dans quelles proportions. Nous avons également demandé comment les personnes se rendaient sur le campus ; nous nous intéressions avec cette question à l'utilisation des différentes lignes de bus qui représentent des lieux de contaminations importants. Nous avons envoyé ce sondage à l'ensemble des personnes travaillant sur le site de la Chantrerie et nous avons obtenu 229 réponses. Quelques-uns de nos résultats sont visibles ci-dessous. Si l'on peut observer dans le premier diagramme un total bien supérieur à 100 %, c'est parce que certaines personnes ont voté pour deux bâtiments différents.



*Figures 21 et 22 : Diagrammes des réponses au sondage*

## II. Analyse des difficultés rencontrées

### A. Gestion de l'humain et du relationnel

Dans l'optique d'avoir une bonne dynamique lors de ce projet, nous avons dans un premier temps décidé de diviser le groupe en deux grandes parties : un groupe qui s'occupe de la programmation composée de Nicolas, Eowyn, Elvire et Lisa et le groupe qui gère la recherche de données pertinentes pour ce projet avec Lysa, Léa, Margaux et Anta. Des sous-groupes ont été par la suite créés dans le but de mieux répartir les tâches. Dans la team modélisation Nicolas s'occupait de créer un programme cherchant à trouver le chemin le plus rapide pour se rendre à un point, Eowyn de créer le programme "mère" avec les collisions et Lisa et Elvire de créer l'interface graphique de notre modélisation.

Nous avons essayé de nous réunir tous les mardis après-midi pour avoir un bilan sur l'avancée des recherches et des programmes. Comme dans tout groupe, nous avons dû nous confronter à quelques problèmes d'organisation et de communication. Cependant, nous avons su les gérer assez rapidement. L'organisation de ce projet en équipes de 8 personnes n'est pas chose aisée puisque nous avons l'habitude de travailler au sein de groupes plus petits. Cette configuration a donc amené à des confrontations de points de vue et d'idées. Nous avons pu régler ces divergences en débattant tous ensemble. Individuellement, nous n'avons pas tous la même façon de nous organiser. Pour certains, les réunions hebdomadaires étaient un rendez-vous important, d'autres ressentaient moins le besoin de mise en commun chaque semaine. Pour ne pas que le projet se disperse nous avons donc fait attention de ne pas couper la communication même lorsque des personnes n'étaient pas présente physiquement.

### B. Gestion du temps

Pendant la réalisation de ce projet, la gestion du temps a été un aspect primordial de notre organisation. Certaines étapes ont respecté le temps prévu lors de la phase d'avant-projet, d'autres ont pris du retard. Nous avons tous commencé à travailler dès le début du mois de janvier et durant les premières semaines, nous n'avons pas rencontré de difficultés particulières. Cependant, l'avancement du projet s'est vu ralentir au milieu du semestre, l'enthousiasme du début s'essoufflant. De plus, il y a eu des semaines plus compliquées que d'autres. En effet, la conférence PEIP était certes un projet important, mais d'autres échéances en parallèles venaient ralentir son avancée (CC, oraux, etc). Les réunions lors desquelles nous nous retrouvions toutes les semaines ont donc été bénéfiques et ont permis de garder tout de même le projet actif tout du long. La phase des recherches bibliographiques fut un peu plus longue que ce que nous avions imaginé, dû à des problèmes techniques expliqués dans le paragraphe qui suit. Différentes parties du programme ont été mises en commun tardivement, nous avons alors rencontré des difficultés à les coordonner. En effet, si on se penche un peu plus sur le diagramme de Gantt fait au préalable, on remarque des écarts de temps significatifs. Par exemple, sur la partie programmation et modélisation. De toute évidence, nous avions initialement prévu 9 semaines pour ces parties. Or, il s'avère qu'il nous en aura fallu 4 de plus de ce qui avait été prévu au départ. Ainsi, ce retard a été payé. Nous n'avons pas pu régler certains problèmes et nous avons donc dû choisir de ne pas utiliser une de ces parties dans notre modélisation finale.

La rédaction de ce rapport a pour autant, elle, été prise en charge tôt, nous avons pu respecter les temps fixés sur le diagramme de Gantt pour cette phase. Enfin, pour une gestion du temps optimale, il nous a paru évident de nous répartir les tâches selon nos compétences. Nous avons très vite constaté que cette stratégie était payante. Effectivement,

travailler par groupe de deux ou trois à permis d'avancer plus vite tout en alliant cohésion et efficacité.

### C. Gestion des risques techniques

Lors de projet, nous avons dû faire face à plusieurs problèmes plus ou moins conséquents. Dans la partie de la modélisation de l'interface graphique, nous avons dû envoyer un mail pour obtenir les plans de l'établissement et faire un sondage qu'on a par la suite envoyé à tous les étudiants et personnels de Polytech. Cependant, après avoir envoyé le sondage, nous nous sommes rendu compte qu'il n'a pas été envoyé à tout le monde, mais qu'à une seule partie des élèves. En effet, le manque de retour nous a alertés. Nous avons donc dû avoir une certaine réactivité pour comprendre ce problème et le renvoyer à tout l'établissement.

Au moment de la conception du programme qui gère les collisions entre les points et les transmissions, nous avons dû mettre en place des paramètres avec les chiffres obtenus pendant les phases de recherches avec des taux de contagion et de guérison ou encore des paramètres qu'on a choisis comme le nombre de rats au début de la modélisation et le nombre de personnes. Suite aux choix des paramètres, nous avons effectué plusieurs essais plus ou moins concluants afin de fixer les bons paramètres. Cela nous a pris tout un après-midi.

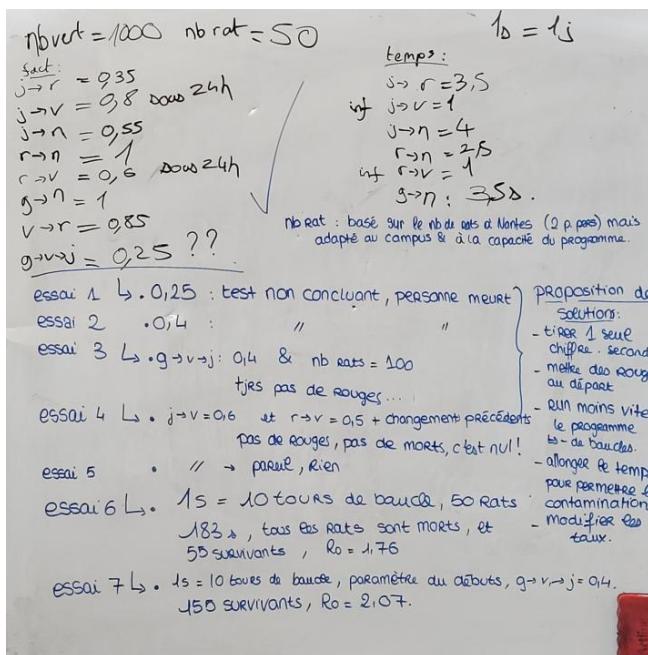


Figure 23 : Photo représentant la première partie des tests sur les paramètres

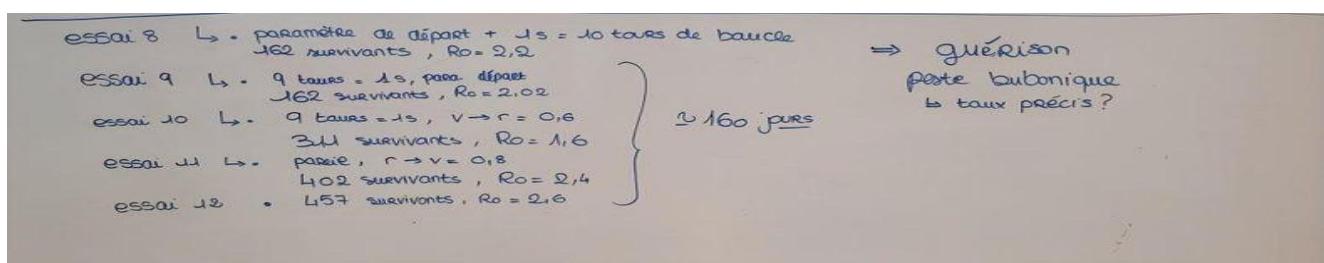


Figure 24 : Photo représentant la deuxième partie des tests sur les paramètres

Comme vous pouvez le voir sur les images ci-dessus (figure 22 et 23), plusieurs essais ont été faits avec plusieurs changements de paramètres.

À la fin du premier test, nous avons constaté qu'il n'y avait aucun mort. Ce n'était pas le résultat attendu donc nous avons décidé que ce premier test n'était pas concluant donc de modifier des paramètres. Après plusieurs tests, nous avons obtenu des résultats cohérents avec ce à quoi nous nous attendions et les recherches faites, ce qui nous a permis de fixer ces paramètres-là et donc de ne plus faire de modification.

D'autres problèmes techniques ont fait surface au cours de la seconde modélisation du problème. En effet, tout ce qui a été expliqué plus tôt depuis le début de la sous-partie "nouveautés et changements" nous posa plusieurs problèmes. Premièrement, comme vous pouvez le voir sur la figure 13, les fonctions `fxmax()`, `fxmin()`, etc, ne représentent pas toujours la même chose selon l'orientation des surfaces, et nous avons eu du mal à nous les représenter correctement. De plus, il a fallu trouver un moyen d'automatiser un peu tout ça pour que ce ne soit pas juste des conditions les unes après les autres, c'est pourquoi nous avons créé les tableaux `Verif`, `fxmax`, `points`, `infocercles`, etc, afin d'accéder au bonne case avec l'attribut `endroit`. Et il a été compliqué de trouver les bonnes solutions pour les fonctions `collmur()` et `dansisitem()`, surtout que nous avons fini par manquer de temps, donc ce que nous proposons ici sont des solutions mais peut être pas les meilleures. De surcroît, on peut remarquer que la fonction `verifprop()`, qui empêche les habitants (sauf les rats) d'entrer dans une surface s'ils sont déjà trop nombreux, ne fonctionne pas parfaitement. En effet, on remarque notamment que de temps en temps des habitants entrent dans le RU à des moments où il ne devrait pas être autorisé. Cependant nous avons manqué de temps pour régler entièrement ce problème.

Après ces problèmes, nous avons dû en faire face à d'autres, mais dans la partie recherches. Lors de nos recherches bibliographiques, nous avons rencontré un gros problème sur les chiffres concernant la peste. Étant donné que c'est une ancienne maladie, il nous a été compliqué de trouver des chiffres exacts sur le taux de transmission à l'époque, le nombre exact de décès et de cas. Pour résoudre ce problème, nous avons dans un premier temps essayé de trouver des maladies épidémiques semblables, dans les méthodes transmissions, à la peste comme le covid-19 et nous avons essayé de voir le taux de transmission que ces maladies avaient afin d'avoir une idée sur celui de la peste. Nous avons ensuite cherché les facteurs pour le calcul du taux de transmission,  $R_0$ . Cependant, il nous manquait des chiffres, nous avons donc décidé de calculer le  $R_0$  avec le programme. Nous avions trouvé des  $R_0$  plus ou moins logiques. Un autre problème que nous avons eu durant cette phase de recherche est le choix de nos articles. Nous avions trouvé à plusieurs reprises des articles qui disaient certes la même chose, mais n'avaient pas les mêmes chiffres. Il a fallu faire des choix sur quels articles placer notre confiance ou non.

## 8. Conclusion

Pour conclure, ce projet s'est déroulé en plusieurs étapes. Nous avons tout d'abord effectué de nombreuses recherches bibliographiques sur la peste. Cependant, lors de nos recherches, nous avons été confrontés aux limites de l'étude de cette maladie. En effet, ce fut compliqué de trouver des chiffres identiques parmi les différentes sources, ce sujet étant peu documenté. De plus, la peste est une maladie complexe à étudier de par ces différents stades d'évolution et son mode de transmission.

Les résultats de ce projet nous ont montré l'impact de nos approximations sur les valeurs obtenues avec notre modélisation. Ces valeurs sont plus ou moins éloignées de la réalité en fonction des différentes simulations. En effet, lors de la première modélisation, dans laquelle nous représentons le déplacement aléatoire des individus dans un cadre restreint, nous obtenons un  $R_0$  égal à 2.5. Nous étions, dans un premier temps, surpris des données récoltées. Cependant, avec une prise de recul nous nous sommes rendu compte que ce  $R_0$  n'était pas si irréaliste. Effectivement, nous n'avons pas utilisé tous les paramètres représentant la réalité de nos jours comme le confinement d'un individu après contamination ou encore l'immunité acquis par ce dernier.

Les résultats du second modèle mettent en scène les individus sur le campus de la Chantrerie. Contrairement aux premiers, les données obtenues à l'aide de ce dernier sont plus inquiétantes. Dans ce second cas, le  $R_0$  prend une valeur moyenne de 19.42. Nous sommes conscients que ce chiffre est beaucoup trop élevé par rapport à la valeur attendue. Cela est notamment lié à nos lacunes dans le domaine de la biologie qui nous aurait permis de décrire le problème plus fidèlement. De plus, contrairement au premier modèle, les habitants sont restreints dans leurs déplacements, ils sont également trop nombreux dans de petites surfaces et sont donc amenés à se toucher et à se contaminer trop fréquemment.

Nous nous sommes focalisés sur ce projet dans le but de connaître l'évolution de la population dans le cas de la réapparition d'une maladie contagieuse et dangereuse comme la peste. Ces événements bouleversent notre quotidien, comme nous avons pu le vivre avec le Covid-19. En effet, avec l'évolution climatique actuelle, nous serons confrontés de plus en plus à un retour de maladies comme celle-ci, voire même, à l'apparition de nouveaux virus ou bactéries inconnues.

## 9. Table des figures

- Figure 1: Représentation des différentes épidémies mondiales et de leur ampleur par rapport au temps [5] p8
- Figure 2 : Mortalité liée à la peste en Inde sur la période 1898-1948 [6] p9
- Figure 3 : Désinfection d'un marché pour limiter la propagation de la peste à Madagascar [9] p9
- Figure 4 : Représentation de la peste bubonique au Moyen-Age, vision traumatique des symptômes, illustration tirée de la Bible de Toggenburg, 1411 [10] p10
- Figure 5 : Entités de base du modèle SIMPEST [36] p17
- Figure 6 : Visuel final de notre modélisation p19
- Figure 7 : Schéma de la contamination entre les individus dans notre modélisation p19
- Figure 8 : Plan du campus avant et après modification p23
- Figure 9 : Plan du campus avec, en vert, les 4 points définissant le chemin rouge p24
- Figure 10 : Schéma de définition des fonctions  $fx()$  et  $fy()$  p25
- Figure 11 : Plan de Polytech avec les surfaces numérotées p26
- Figure 12 : Schéma du fonctionnement de la fonction `collmur` pour la position y d'un habitant p26
- Figure 13 : Schéma du fonctionnement des fonctions `verifsurface()` pour des cercles, ici l'habitant n'est pas dans la surface p27
- Figure 14 : Schéma du fonctionnement de la fonction dans `isitem()` p27
- Figure 15 : Illustration de notre modélisation : Le midi (du 302ème tour de boucle au 367ème pour le 1er jour) une partie des habitants est au RU p29
- Figure 16 : Tableau des résultats du premier programme p32
- Figure 17 : Tableau des résultats du second programme p33
- Figure 18 : Illustration de notre modélisation : au 4ème jour, presque tous les habitants sont rouges p34
- Figures 19 et 20 : Illustrations de l'algorithme p37
- Figures 21 et 22 : Diagrammes des réponses au sondage p39
- Figure 23 : Photo représentant la première partie des tests sur les paramètres p41
- Figure 24 : Photo représentant la deuxième partie des tests sur les paramètres p41

## 10. Sites internet / Bibliographie

- [1] « La peste noire : histoire d'une épidémie meurtrière », *Geo.fr*, 23 mars 2021.  
<https://www.geo.fr/histoire/la-peste-noire-histoire-dune-epidemie-meurtriere-204081> (consulté le 2 mai 2023)
- [2] « Peste noire : date, en France, symptômes, combien de morts ? », 21 juillet 2022.  
<https://sante.journaldesfemmes.fr/fiches-maladies/2755715-peste-noire-morts-en-france-date-symptomes/> (consulté le 2 mai 2023).
- [3] « Rapide et fatale : comment la Peste Noire a dévasté l'Europe au 14e siècle », *National Geographic*, 24 avril 2020.  
<https://www.nationalgeographic.fr/histoire/2020/04/rapide-et-fatale-comment-la-peste-noire-devaste-l-europe-au-14e-siecle> (consulté le 2 mai 2023).
- [4] « Peste noire », *Wikipédia*. 27 avril 2023. Consulté le: 2 mai 2023. [En ligne]. Disponible sur: [https://fr.wikipedia.org/w/index.php?title=Peste\\_noire&oldid=203734206](https://fr.wikipedia.org/w/index.php?title=Peste_noire&oldid=203734206)
- [5] « Les sept pandémies les plus meurtrières de l'histoire | Gavi, the Vaccine Alliance ».

- <https://www.gavi.org/fr/vaccineswork/sept-pandemies-meurtrieres-histoire> (consulté le 2 mai 2023).
- [6] J. Monteilh, « La modélisation épidémiologique (2/3) : 1906, la peste de Bombay », *Petites histoires des sciences*, 23 avril 2020.  
<https://petiteshistoiresdesciences.com/2020/04/23/la-modelisation-epidemiologique-2-3-1906-la-peste-de-bombay/> (consulté le 2 mai 2023).
- [7] SPF, « La peste, un bilan après l'épidémie en Inde ».  
<https://www.santepubliquefrance.fr/maladies-et-traumatismes/maladies-transmissibles-de-l-animal-a-l-homme/peste/la-peste-un-bilan-apres-l-epidemie-en-indie> (consulté le 2 mai 2023).
- [8] « The Bombay plague | National Army Museum ».  
<https://www.nam.ac.uk/explore/bombay-plague> (consulté le 2 mai 2023).
- [9] « Madagascar: l'Etat confirme le ralentissement de l'épidémie de peste », *Sciences et Avenir*, 31 octobre 2017.  
[https://www.sciencesetavenir.fr/sante/madagascar-l-etat-confirme-le-ralentissement-de-l-epidemie-de-peste\\_117944](https://www.sciencesetavenir.fr/sante/madagascar-l-etat-confirme-le-ralentissement-de-l-epidemie-de-peste_117944) (consulté le 8 mai 2023).
- [10] « Ce que l'histoire nous apprend sur le boom économique post-pandémie ».  
<https://fr.linkedin.com/pulse/ce-que-lhistoire-nous-apprend-sur-les-booms-naully-nicolas> (consulté le 8 mai 2023).
- [11] D. Tisserand, « La peste noire : résumé et histoire de la grande peste », 1 mars 2023.  
<https://www.linternaute.fr/actualite/guide-histoire/2616939-la-peste-noire-resume-et-histoire-de-la-grande-peste/> (consulté le 2 mai 2023).
- [12] « Peste », *Institut Pasteur*, 6 octobre 2015.  
<https://www.pasteur.fr/fr/centre-medical/fiches-maladies/peste> (consulté le 2 mai 2023).
- [13] « Peste et autres infections à Yersinia - Maladies infectieuses », *Édition professionnelle du Manuel MSD*.  
<https://www.msmanuals.com/fr/professional/maladies-infectieuses/bacilles-gram-n%C3%A9gatifs/peste-et-autres-infections-%C3%A0-yersinia> (consulté le 2 mai 2023).
- [14] « Peste bubonique : transmission, morts en France, vaccin », 2 novembre 2021.  
<https://sante.journaldesfemmes.fr/fiches-maladies/2658801-peste-bubonique-transmission-symptomes-morts-en-france-traitement-vaccin/> (consulté le 2 mai 2023).
- [15] « OMS Madagascar - A SAVOIR SUR LA PESTE: SIGNES ET... ».  
<https://www.facebook.com/OMSMadagascar/photos/a.525245400999193/728177077372690/?type=3> (consulté le 2 mai 2023).
- [16] « Peste : causes, symptômes & traitements | Creapharma ».  
<https://www.creapharma.ch/peste.htm> (consulté le 2 mai 2023).
- [17] « Peste ». <https://www.who.int/fr/news-room/fact-sheets/detail/plague> (consulté le 2 mai 2023).
- [18] « Peste pneumonique », *Wikipédia*. 4 mars 2023. Consulté le: 2 mai 2023. [En ligne]. Disponible sur:  
[https://fr.wikipedia.org/w/index.php?title=Peste\\_pneumonique&oldid=201970184](https://fr.wikipedia.org/w/index.php?title=Peste_pneumonique&oldid=201970184)
- [19] « PESTE, Les signes de la peste humaine - Encyclopædia Universalis ».  
<https://www.universalis.fr/encyclopedie/peste/3-les-signes-de-la-peste-humaine/> (consulté le 2 mai 2023).
- [20] F. Guinet et E. Carniel, « Faut-il encore craindre la peste aujourd'hui ? », *médecine/sciences*, vol. 24, n° 10, Art. n° 10, oct. 2008, doi: 10.1051/medsci/20082410865.
- [21] I. Bonnemain, H. Sarrasin, F. Sierro, et D. Genné, « La durée de l'antibiothérapie diminue : restons à jour ! », *Rev Med Suisse*, vol. 578, p. 1725-1731, oct. 2017.
- [22] « Feelgoods Pharmacies », *Feelgoods Pharmacies*. <https://feelgoods-pharmacies.ch> (consulté le 2 mai 2023).
- [23] « Bien prendre ses antibiotiques », *VIDAL*.  
<https://www.vidal.fr/medicaments/utilisation/antibiotiques/bien-prendre.html> (consulté le 2 mai 2023).
- [24] M. Galimand, E. Carniel, et P. Courvalin, « Resistance of *Yersinia pestis* to Antimicrobial

- Agents », *Antimicrob. Agents Chemother.*, vol. 50, n° 10, p. 3233-3236, oct. 2006, doi: 10.1128/AAC.00306-06.
- [25] É. Carniel, « La peste », *C. R. Biol.*, vol. 325, n° 8, p. 851-853, août 2002, doi: 10.1016/S1631-0691(02)01493-2.
- [26] H. Mollaret, « Le cas de la peste », *Ann. Démographie Hist.*, vol. 1989, n° 1, p. 101-110, 1989, doi: 10.3406/adh.1989.1733.
- [27] A. de la santé publique du Canada, « Pour les professionnels de la santé », 9 mai 2018. <https://www.canada.ca/fr/sante-publique/services/maladies/peste/professionnels-sante.html> (consulté le 2 mai 2023).
- [28] « À Nantes, comme dans toutes les grandes villes, les rats d'égouts prolifèrent », *Ouest-France.fr*, 19 octobre 2021. <https://www.ouest-france.fr/pays-de-la-loire/nantes-44000/les-rats-pullulent-a-nantes-b9af0ff2-2b35-11ec-bfa1-8289e5d95d8d> (consulté le 9 mai 2023).
- [29] V. Laperrière, « Apport de la modélisation individu-centrée spatialement explicite à la compréhension de l'expression d'une maladie transmissible : la peste bubonique à Madagascar », phdthesis, Université de Pau et des Pays de l'Adour, 2009. Consulté le: 2 mai 2023. [En ligne]. Disponible sur: <https://theses.hal.science/tel-00445563>
- [30] Joanna, « Modéliser la propagation d'une épidémie », *Interstices*, 27 novembre 2011. <https://interstices.info/modeliser-la-propagation-dune-epidemie/> (consulté le 2 mai 2023).
- [31] « Les types de modèles — Site des ressources d'ACCES pour enseigner les Sciences de la Vie et de la Terre ». <http://acces.ens-lyon.fr/acces/thematiques/sante/epidemies-et-agents-infectieux/sida/les-types-de-modeles> (consulté le 5 mai 2023).
- [32] M. J. Keeling et C. A. Gilligan, « Metapopulation dynamics of bubonic plague », *Nature*, vol. 407, n° 6806, Art. n° 6806, oct. 2000, doi: 10.1038/35038073.
- [33] P. Foley et J. Foley, « Modeling Susceptible Infective Recovered Dynamics and Plague Persistence in California Rodent–Flea Communities », *Vector Borne Zoonotic Dis. Larchmt. N*, vol. 10, p. 59-67, févr. 2010, doi: 10.1089/vbz.2009.0048.
- [34] « Écologie / Ecology - Modèle individu centré ». <https://www.bonobosworld.org/fr/glossaire/modele-individu-centre> (consulté le 7 mai 2023).
- [35] V. Laperrière, « Modélisation dynamique de la peste à Madagascar, entre théorie et observations », *L'Espace Géographique*, vol. 39, n° 4, p. 345-359, 2010, doi: 10.3917/eg.394.0345.
- [36] D. Badariotti, A. Banos, et V. Laperrière, « Vers une approche individu-centrée pour modéliser et simuler l'expression spatiale d'une maladie transmissible : la peste à Madagascar », *Cybergeo Eur. J. Geogr.*, juill. 2007, doi: 10.4000/cybergeo.9052.
- [37] N. Bacaër, « The model of Kermack and McKendrick for the plague epidemic in Bombay and the type reproduction number with seasonality », *J. Math. Biol.*, vol. 64, n° 3, p. 403-422, févr. 2012, doi: 10.1007/s00285-011-0417-5.
- [38] « La varicelle – GERES ». <https://www.geres.org/autres-pathologies/la-varicelle/> (consulté le 5 mai 2023).
- [39] P. T. Debord, « Moyens de lutte contre les risques épidémiques ».

## 11. Annexes

[https://drive.google.com/file/d/11cf7leGbBzp7PJ4XC5ALFN8Ty0unwoAs/view?usp=share\\_link](https://drive.google.com/file/d/11cf7leGbBzp7PJ4XC5ALFN8Ty0unwoAs/view?usp=share_link)