MOURONVAL Laurane
LUTIC Enzo

TPD
SAE 1.02

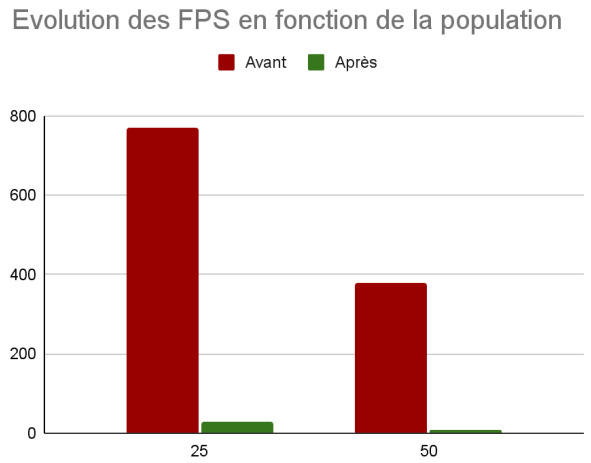
ETAPE 1:
(!! on a arrondi les fps à 5 nombres après la virgule pour plus de visibilité)

Objectif: Optimisation de l'affichage
synchroniser le déplacement des personnes pour avoir des FPS cohérents

AVANT	**APRÈS**
les déplacements des personnes ne sont pas synchronisés, une image est générée à chaque fois qu'une personne se déplace	changement dans pandemic.py: uniquement après avoir mis à jour toutes les personnes: on redessine la scène et sa population et on met à jour l'affichage à l'écran -> sortir la partie draw scene et la partie display update de la boucle for dans la boucle principale. changement dans bench.py: on sort de la boucle for dans la boucle principale le fait de compter une frame à chaque déplacement d'une personne (variable frameNumber)

Statistiques
(sont plus élevés dans le bench car il n'y a pas d'affichage à l'écran de la simulation)

exemple de statistique de pandemic.py	**exemple de statistique de pandemic.py**
FPS moyen durant la simulation : 467.6555 A la fin de la simulation, pour une population de 25 personnes, il y a : - 21 personnes en bonne santé, soit 84.0 % - 0 personnes infectées, soit 0.0 % - 4 personnes guéries, soit 16.0 % **exemple de statistique de bench.py** healthy;infected;immune;FPS 5;0;20;768.90965	FPS moyen durant la simulation : 29.48693 A la fin de la simulation, pour une population de 25 personnes, il y a : - 21 personnes en bonne santé, soit 84.0 % - 0 personnes infectées, soit 0.0 % - 4 personnes guéries, soit 16.0 % **exemple de statistique de bench.py** healthy;infected;immune;FPS 9;0;16;30.42447



Les FPS ont fortement chuté mais sont à présent cohérents.

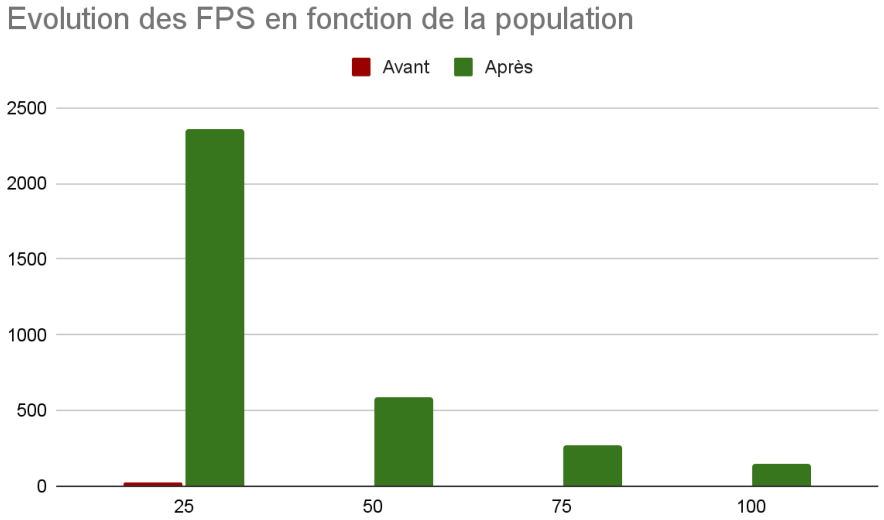
ETAPE 2:

Objectif: Optimisation mathématique
On essaie de supprimer un maximum de calculs complexes

AVANT	**APRÈS**
La fonction circleColissions calculait un point du cercle et testait cela pour tout le périmètre du cercle.	changement dans engine.py: on a retiré le système de calcule qu'on a remplacé par un plus simple qui nous fait pas testé plusieurs calculs. Le principe est simple, si la distance entre le cercle 1 et le cercle deux est plus petite que le rayon les cercle se touche.

Statistiques
(sont plus élevés dans le bench car il n'y a pas d'affichage à l'écran de la simulation)

exemple de statistique de pandemic.py	**exemple de statistique de pandemic.py**
FPS moyen durant la simulation : 29.48693 A la fin de la simulation, pour une population de 25 personnes, il y a : - 21 personnes en bonne santé, soit 84.0 % - 0 personnes infectées, soit 0.0 % - 4 personnes guéries, soit 16.0 %	FPS moyen durant la simulation : 1052.24103 A la fin de la simulation, pour une population de 25 personnes, il y a : - 23 personnes en bonne santé, soit 92.0 % - 0 personnes infectées, soit 0.0 % - 2 personnes guéries, soit 8.0 %
exemple de statistique de bench.py healthy;infected;immune;FPS 9;0;16;30.42447	**exemple de statistique de bench.py** healthy;infected;immune;FPS 17;0;8;2355.89837



Les FPS ont été multipliés par 1000 en moyenne, l'optimisation des calculs est donc efficace.

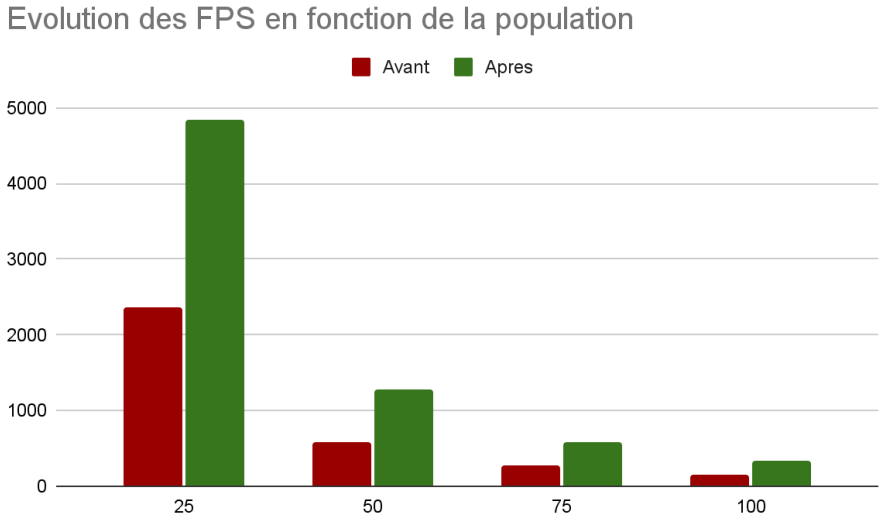
ETAPE 3:

Objectif: Optimisation par approximation
effectuer le moins de calcul possible pour la fonction de test d'intersection

AVANT	**APRÈS**
On fait des calculs pour savoir si les cercles se touchent.	changement dans engine.py: On calcule d'abord si deux carrés autour des cercles se touchent, si oui alors on calcule pour les cercles, sinon non. Cela permet de faire beaucoup moins de calculs complexes.

Statistiques
(sont plus élevés dans le bench car il n'y a pas d'affichage à l'écran de la simulation)

exemple de statistique de pandemic.py	**exemple de statistique de pandemic.py**
FPS moyen durant la simulation : 1052.24103 A la fin de la simulation, pour une population de 25 personnes, il y a : - 23 personnes en bonne santé, soit 92.0 % - 0 personnes infectées, soit 0.0 % - 2 personnes guéries, soit 8.0 % **exemple de statistique de bench.py** healthy;infected;immune;FPS 17;0;8;2355.89837	FPS moyen durant la simulation : 1218.57564 A la fin de la simulation, pour une population de 25 personnes, il y a : - 23 personnes en bonne santé, soit 92.0 % - 0 personnes infectées, soit 0.0 % - 2 personnes guéries, soit 8.0 % **exemple de statistique de bench.py** healthy;infected;immune;FPS 22;0;3;4849.37785



Les FPS ont été globalement multipliés par 2. On remarque que les FPS étaient à environ. Cette optimisation à donc été efficace.

ETAPE 4:

Objectif:

AVANT
On avait le code pandemic.py qui testait tous les cercles s'il touchait quelqu'un. Le problème c'est qu'on ne change que si les personnes sont infectées.

APRÈS
Le changement est très simple:
On a créé une condition dans la boucle for qui ne va lancer les fonctions `computeCollisions` et `processCollisions` que pour les personnes infectées. Cela ne test que les personnes infectées ce qui empêche de faire des choses inutiles.

Statistiques
(sont plus élevés dans le bench car il n'y a pas d'affichage à l'écran de la simulation)

exemple de statistique de pandemic.py

FPS moyen durant la simulation : 1218.57564

A la fin de la simulation, pour une population de 25 personnes, il y a :
- 23 personnes en bonne santé, soit 92.0 %
- 0 personnes infectées, soit 0.0 %
- 2 personnes guéries, soit 8.0 %

exemple de statistique de bench.py

healthy;infected;immune;FPS
22;0;3;4849.37785

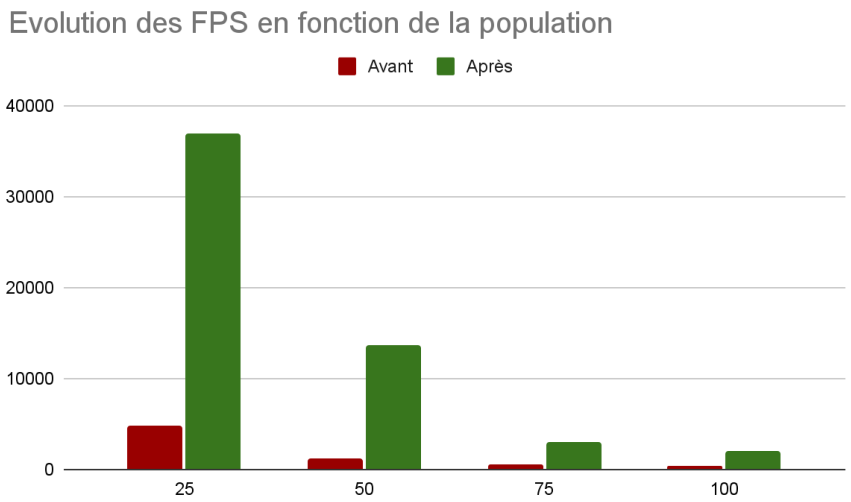
exemple de statistique de pandemic.py

FPS moyen durant la simulation : 901.01099

A la fin de la simulation, pour une population de 25 personnes, il y a :
- 23 personnes en bonne santé, soit 92.0 %
- 0 personnes infectées, soit 0.0 %
- 2 personnes guéries, soit 8.0 %

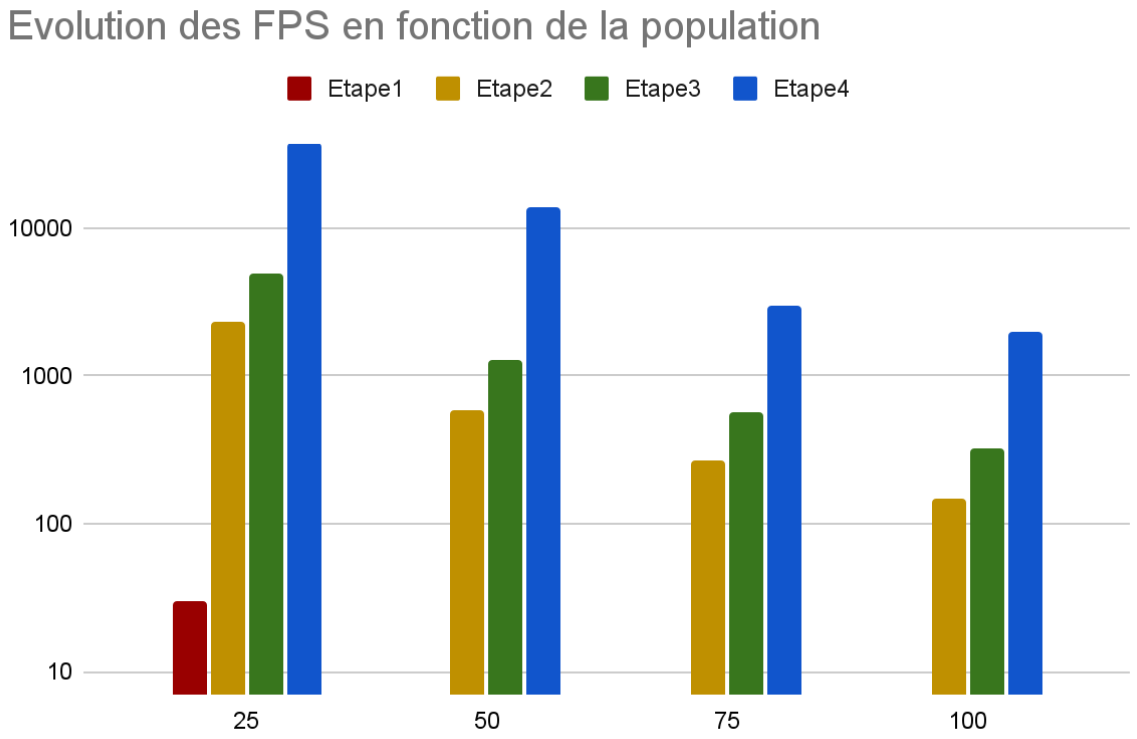
exemple de statistique de bench.py

healthy;infected;immune;FPS
24;0;1;37041.72541



Les FPS ont été globalement multipliés par 7 environ. Cette optimisation à donc été efficace.
MAIS il y a un **inconvénient** important à cette optimisation remarqué au lancement du pandemic.py:
Le jeu subit des ralentissements quand il commence à y avoir plusieurs personnes infectées, de plus il devient trop rapide.

CONCLUSION



We had to do a project in python which talked about the impact of a virus in a given population. We could see the simulation with a graphic interface but we realized that the frames per second, also called FPS, were very low even when we tested without a big population. Therefore, the aim was to optimize the code in order to raise the FPS.

First, we had to make the current frames coherent. We had to synchronise the population movements to make them move at the same time. So, the frames dropped from more than 700 to only 30 for 25 people.

Then, we had to minimize the number of complex calculations. Originally, the program did 3200 calculations to determine if there was a collision with a population of 100. After the change we made, the number of calculations reduced to 32 for the same population. We can see the significant change with the column chart, the FPS increased from 30 to about 2.300 for 25 people. This optimization is consequently very efficient.

Next, we wanted to reduce the number of calculations the program needed to do in order to know if two people touched. A person is characterized by a circle and it's more difficult to compare two circles than two squares. So, we added a function to know if the squares touched and only if they were, we would calculate if the circles touched. We can see that this optimization is efficient because there is a general rise in FPS in comparison with the second step.

Finally, we decided that we only wanted to test the collision of the infected people. The FPS climbed considerably from about 5.000 to more than 37.000 for 25 people. The

FPS are very high but we can see while we run the graphic interface that the more there are infected people, the more there are slowdowns in speed.

This project allowed us to see that the way of programming has a strong impact on the performance of our code.