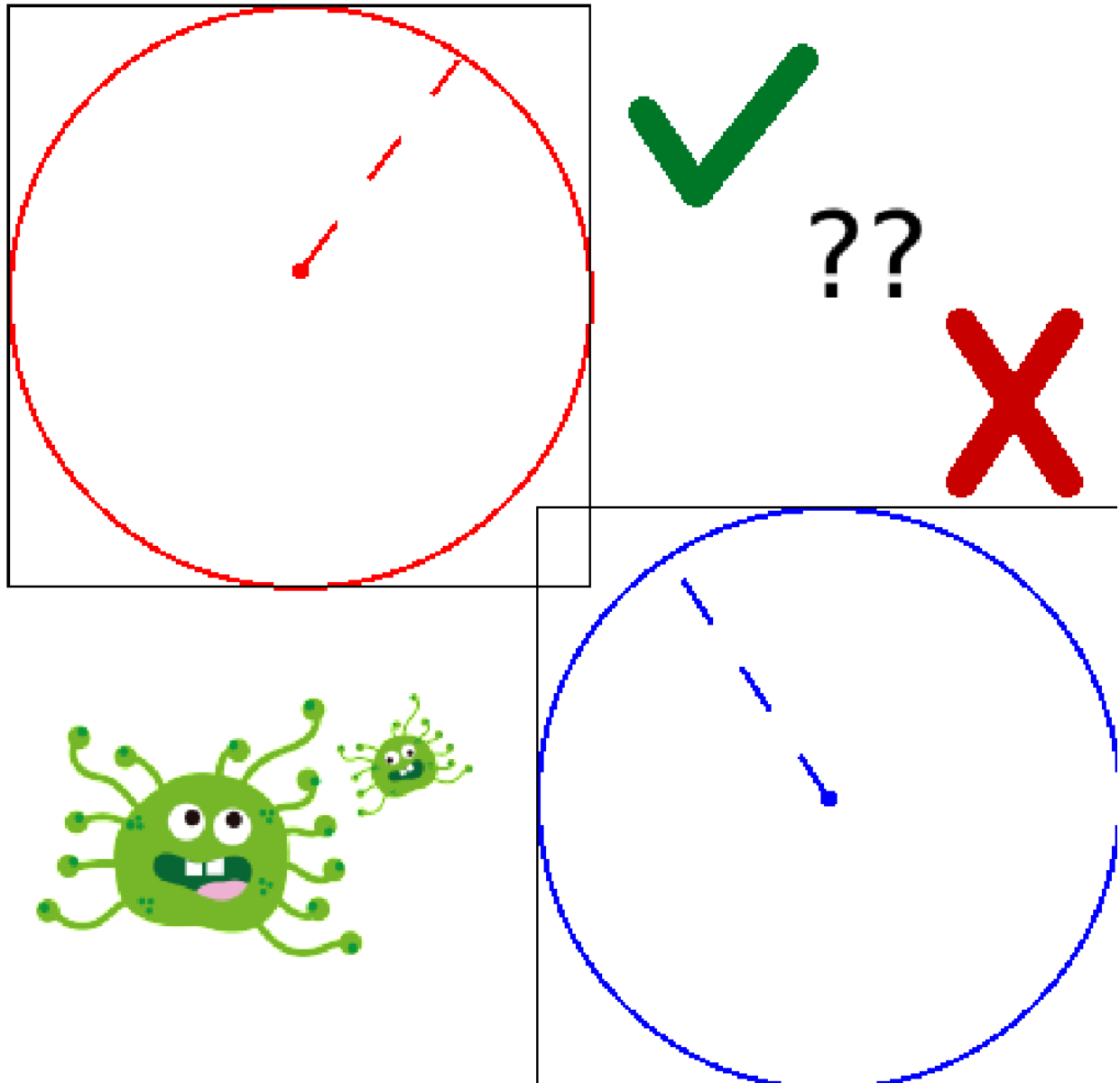


SAÉ 1.02 - Comparaison d'approches algorithmiques

*Optimisation d'algorithmes pour une simulation
de diffusion de la du covid*

There's a file for every step, they are called "etapeN_file" in the
folder !



!!! Every underlined and Italic word is a reference to the code. !!!

Step 1: Display optimisation

In this step we were demanded to optimise the way that he the algorithms displays the simulation.

The problem is that the refreshment of the images is made after every single movement of a ball/person. Wich makes the whole simulation laggy and jerky. Not optimised!

By changing the time when the algorithm refreshes. We have fewer refreshments but more unity on the movement, coming from the refreshment of the movement only when the whole balls moves.

Changes:

The changes that we have made are on the Pandemic.py file. We simply decremented the draw. On the Main.

We putted the draw on the line 93 of the main. After all the movements are processed.

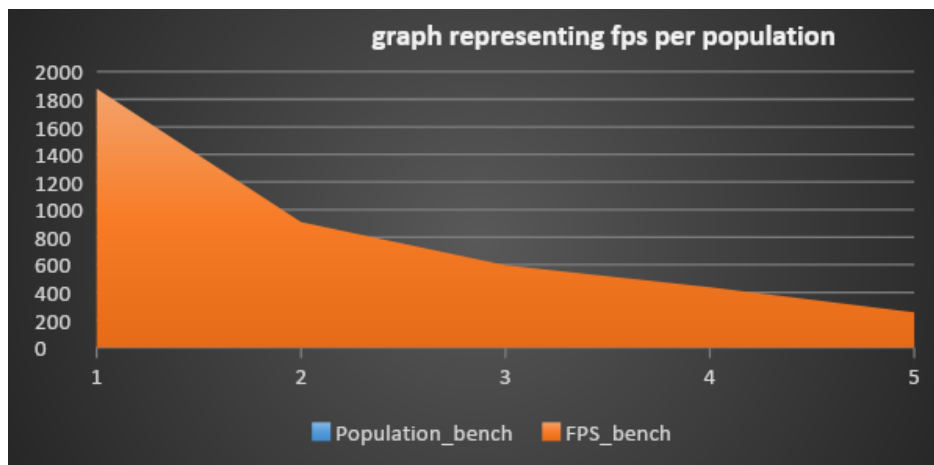
We did the same thing on the FrameNumber, so that the frames are more realistic.

Results:

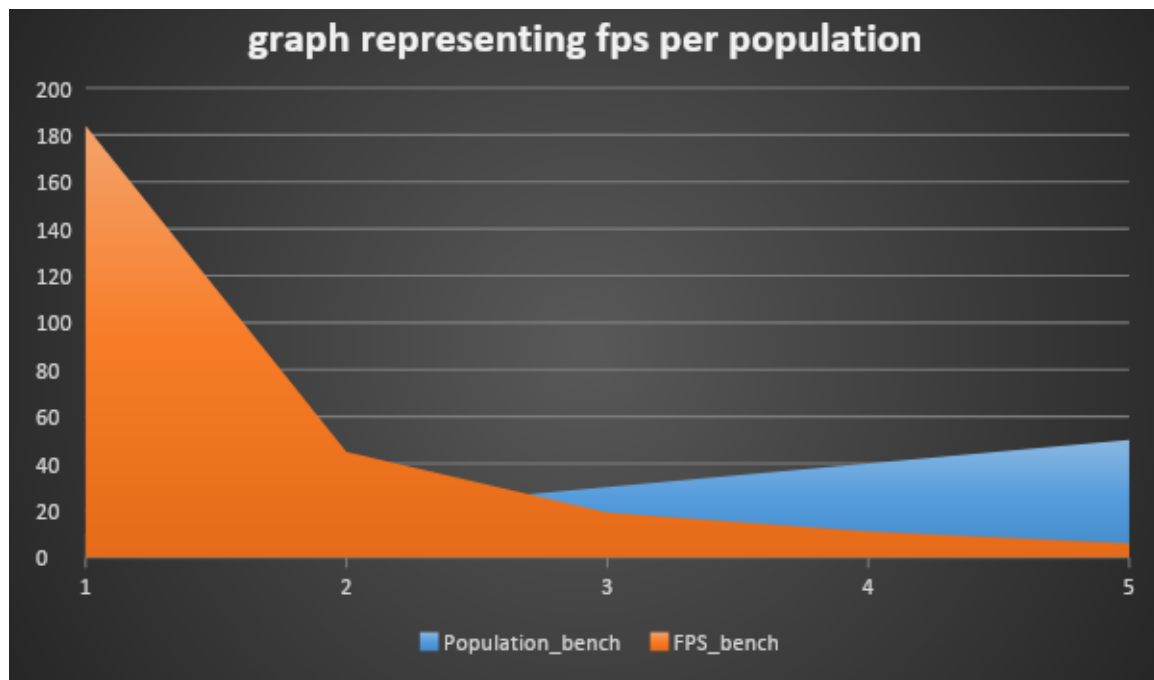
After the changes, we have had as results decreasing of the fps, which was predictable because we do not refresh, as oftener, in the other hand it is more realistic.

Graphics:

Before:



After:



As we can observe on the first graphic the fps were abnormally high. After the changes the fps are more coherent. However, they are low and we cannot have a high number of people in the population, this problem leads us to the second step.

STEP 2 : Mathematic optimisation

One of the problems of this algorithm is that it calculates the intersection on two persons but it does it X times depending on the number of people. In addition, this is supposed to be a simulation of Covid spreading so the number of people must be consequent. With the calculations of the intersections, it can rapidly be tricky. The current calculations are too heavy. We need to make them simpler.

Changes:

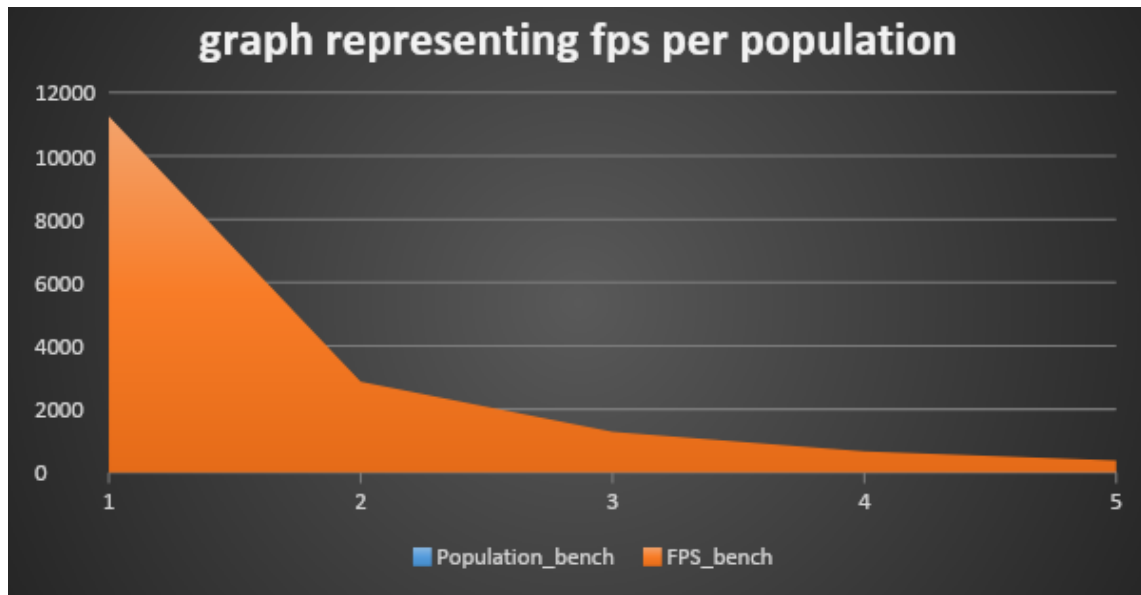
We eliminated the square root by squaring the radius. This helped the program to take less time on making the calculations.

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

Code:

```
def circleCollision(c1, c2):
    """
    Determines if two circles intersect or not
    """
    # We pick a point on the first circle
    d = ((c2[0]-c1[0])**2) + ((c2[1]-c1[1])**2) #this is the
    # We then move to another point close to the previous one, until we loop a
    # around c1
    # If this length is less than the radius of c2, then this point is inside
    # the circle
    if d < ((constants.PERSON_RADIUS**2)*2):
        return True
    else:
        return False
    # No collision detected : c1 and c2 do not intersection
```

Graphics:



We perceive a great increase of the fps but a grand decrease on the increasing of the population.

Step3: optimization by approximation

We can now to optimize even more considerate the persons hitbox as a square.

That helps a lot, because the intersection of two squares that does not rotate is a lot easier to calculate.

Changes:

From our researches, we've found out this rule.

For 2 given squares or rectangles.

```

1
2  +-----+
3  |         |
4  |         +-----+
5  |         | |         |
6  +-----+--+
7  1         |         |
   2+-----+
8

```

maxleft=max(x1,x2)

minright=min(x1+width1,x2+with2)

maxdown=max(y1,y2)

minup=min(y1+height1,y2+height2)

There is an intersection if
:

maxleft<minright and maxup<minup

This rule in python and with our variables gives:

```

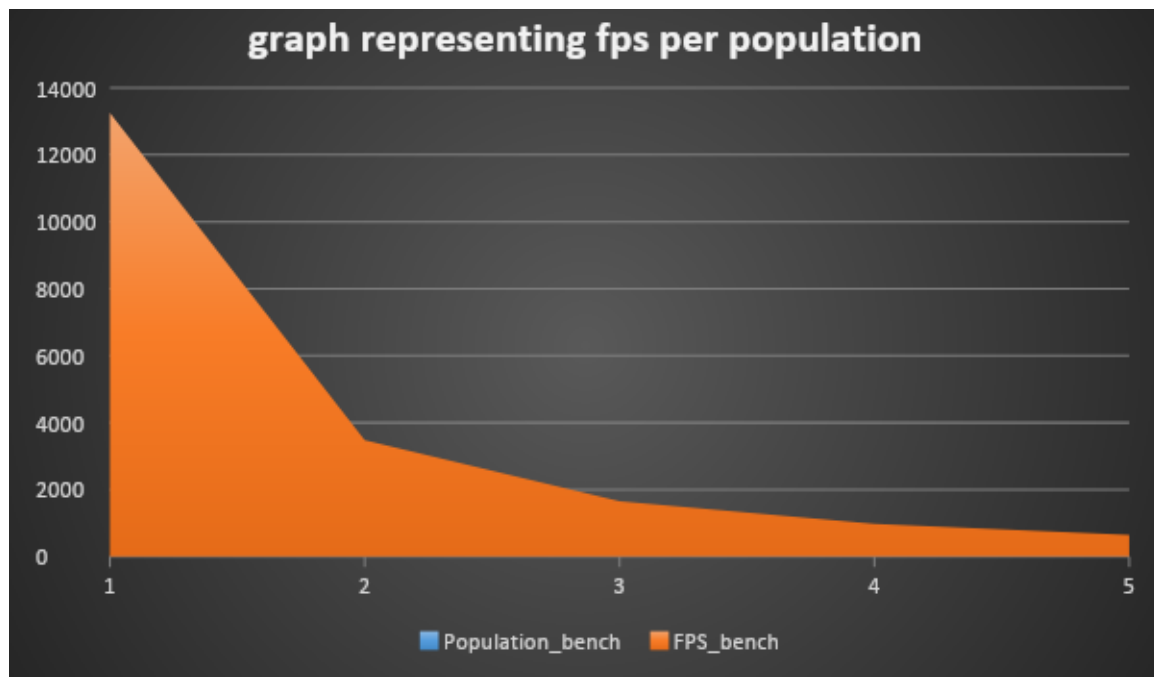
if max(c1[0], c2[0]) < min(c1[0]+(constants.PERSON_RADIUS*2),
c2[0]+(constants.PERSON_RADIUS*2)) and max(c1[1], c2[1]) <
min(c1[1]+(constants.PERSON_RADIUS*2),c2[1]+(constants.PERSON_RADIUS*2)):
    return True
else:
    return False

```

Results:

We found out that the fps on the bench.py have increased again. Caused by the simplicity of the calculations.

Graphic:



As we can see here the fps are a lot better even if the population increases a lot. This is already a lot better.

Nevertheless, we can do better.

STEP 4: algorithmic logic optimization

We did a good job on optimizing the algorithm. However, we still check for collision between all the circles/persons. We are simply going to verify if the circle/person is infected before validating a collision.

Changes:

We simply added the verification in the function `compute collisions`, knowing that `p` and `q` are persons and that `p[2]` and `q[2]` refers to the state [HEALTHY = 0 ; INFECTED = 1 ; IMMUNE = 2].

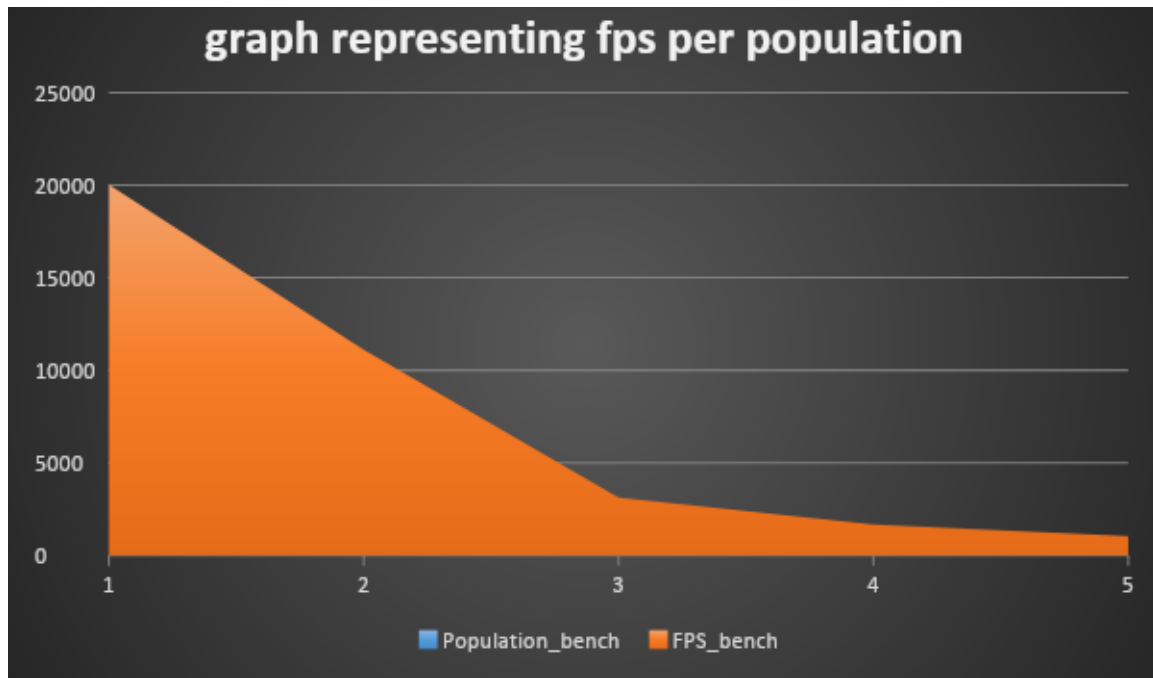
Code:

```
if p[2]==1 or q[2]==1:
```

Results:

We can see now that the fps are a lot better even if the population is at high rate.

Graphic:



Conclusion :

Although we didn't know (almost) nothing about pygame, this was a great first approach. Thanks to the document and the teachers, we were able to learn and start to organize and optimise some code. We learned about code optimisation, maths application and mostly about code performance understanding.

We know that our code isn't the best but we are able to understand it, now. We've created something that we're proud of. Thanks for reading us !!!

Th

The Sources :

- <https://www.developpez.net/forums/d776649/general-developpement/algorithme-mathematiques/algorithmes-structures-donnees/detecter-l-intersection-entre-rectangles/>
- <https://www.it-swarm-fr.com/fr/algorithm/intersection-de-deux-rectangles/1070665969/>
- https://les-mathematiques.net/vanilla/index.php?p=discussion/comment/1453278#Comment_1453278
- <https://stackoverflow.com/questions/25068538/intersection-and-difference-of-two-rectangles>
- <https://stackoverflow.com/questions/25068538/intersection-and-difference-of-two-rectangles>
- <https://www.developpez.net/forums/d776649/general-developpement/algorithme-mathematiques/algorithmes-structures-donnees/detecter-l-intersection-entre-rectangles/>
- <https://www.developpez.net/forums/d776649/general-developpement/algorithme-mathematiques/algorithmes-structures-donnees/detecter-l-intersection-entre-rectangles/>
- <https://www.devoirs.fr/4eme/mathematiques/comment-calculer-une-diagonale-sans-pythagore-216252.html>
- <https://www.devoirs.fr/4eme/mathematiques/comment-calculer-une-diagonale-sans-pythagore-216252.html>
- <https://www.geometrictools.com/Documentation/IntersectionOfEllipses.pdf>
- <https://qastack.fr/programming/115426/algorithm-to-detect-intersection-of-two-rectangles>
- <https://qastack.fr/programming/115426/algorithm-to-detect-intersection-of-two-rectangles>
- <https://www.developpez.net/forums/d910093/autres-langages/python/general-python/gerer-l-intersection-d-objets-cercles-ellipses-etc/>
- http://dossierslmm.cher-alice.fr/fiche/intersection_droites.pdf
- <http://math.15873.pagesperso-orange.fr/IntCercle.html>
- <https://codes-sources.commentcamarche.net/forum/affich-1524867-fonction-pour-calculer-l-intersection-de-deux-cercles>
- <https://stackoverflow.com/questions/55816902/finding-the-intersection-of-two-circles>
-

Thanks for Reading!