Lari Kunnasmaa

1 Map optimization explanation

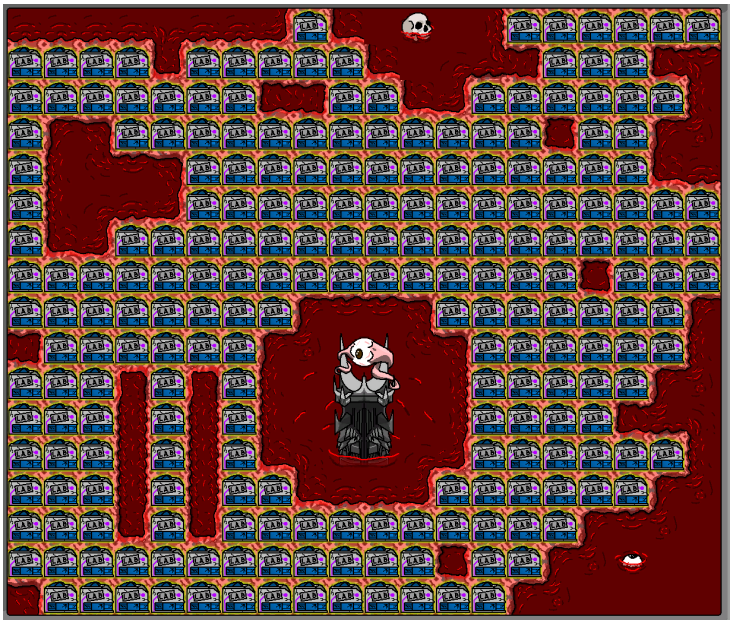
Hello again, you just can't keep away from these pdfs am I right? I am a huge fan of at least decent documentation. This pdf contains an explanation and some visuals of the code related to a free-time RPA project. The project has image recognition, some optimization and basic 2-d array manipulation, followed, by a pinch of pyautogui mouse manipulation. While this coding project is again based on optimizing a game I don't see why it would be any different from a real-world application. The code still allows for a great showcase of my problem-solving and knowledge of python, when it comes to automation and I think it perfectly sums up my passion for both. Oh and again don't take it too seriously just a fun project. I hope you enjoy it.

Kuva 1: You can use 2 min or 2 hours, don't lie to me we have all done it. Althought there is a 100 % guarantee that the 2 hours will be more fun and enjoyable.

2 Problem

The game consists of maps shown in figures 1 a and b. These maps have tiles that produce, let's say "fairy dust"for laboratories. We want to maximize the amount of fairy dust. One lab produces one "fairy dust", this can be increased by placing upgrade beacons that increase the production by 40% for adjacent tiles. The picture of a beacon at work can be seen in figure 3.

(a) Map 1 filled with laboratories, its total production is 132



(b) Map 2 filled with labs total production 226, because it has 226 labs

Kuva 2: Maps 1 and 2.

Kuva 3: the upgrade beacon upgrades adjacent tiles by 40 %, so the production of these 9 tiles is is now $1.4 \cdot 8 = 11.2$ instead of 9

So, as we can see the problem is very simple optimization. The code might be a bit unsophisticated in places since the code originally should have machine learning, which will be finished after I have finished my thesis. However, the code, shows a simple yet effective approach when it comes to solving the problem.

3 Solution

Before we can let the computer place any upgrade beacons we must give the computer a way to interpret the map, to find the best places for them. I used pyautogui for the solution, it recognizes all of the labs placed on the map and adds them to a 2D array that is the size of the map. Their position on the array is determined by the coordinates they found by the image recognition. After this, the program is supposed to use machine learning to try to learn how to place the beacons using several neighbouring tiles. The code for machine learning (unsupervised) is not ready yet or at least not presentable, so I improvised a solution that is almost as effective. It simply takes a random tile and checks would a beacon or a lab creates better production. It does this a number of times, for random tiles. Yes, it does sound very simple and that is because it is, but as will see it's very effective. After finding the optimal tiles on which to place the beacons the code calculates the total power and uses pyautogui to place the beacons in the positions that were found optimal and plot a nice heatmap to show that the positions are correct. Just to help to debug and some visual flare if you know what I mean.

4 Results

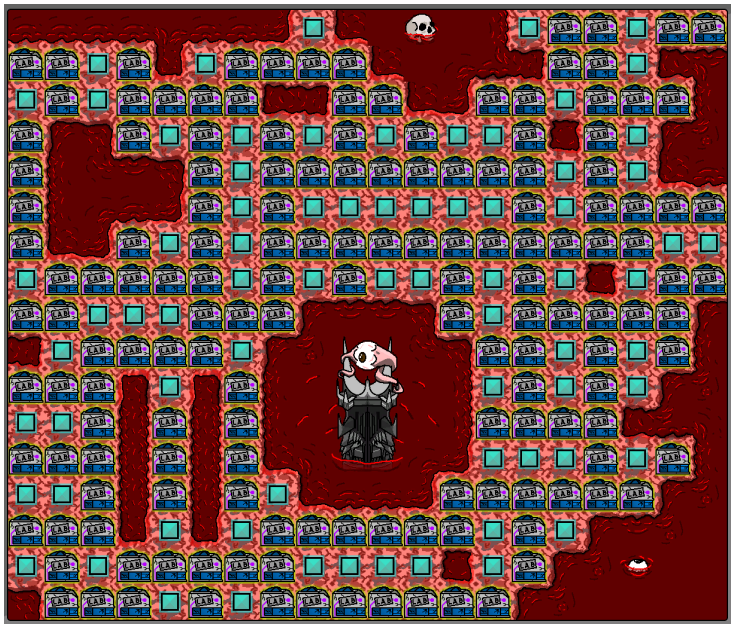
After the placement, the maps look like figures 5 a and b. The production power of each map increased by 35 % and 40 % respectively. The maximum value that the maps can achieve is 182 and 328. So, we are not too far from the most optimized value. Fair enough the solution is not perfect, but it does not have to be, it just needs to be better than a normal human who is placing the beacons. This placement will take and placing a new map would mean

doing it all over again. (There are 5 maps). So, the time saving is pretty big since the code takes around 10 seconds. Now some people might say "Why don't you just randomly throw beacons at the map". Well in the next section I demonstrate why that is not a great idea.

Taulukko 1: This table show the amouont of power each gets before and after the code. There is alos the maximum amount that is possible in the map.

	Before	After	Max
Map 1	132	178	182
Map 2	226	315	328

(a) The result of the code on map1, total power from 132 to 178,2.

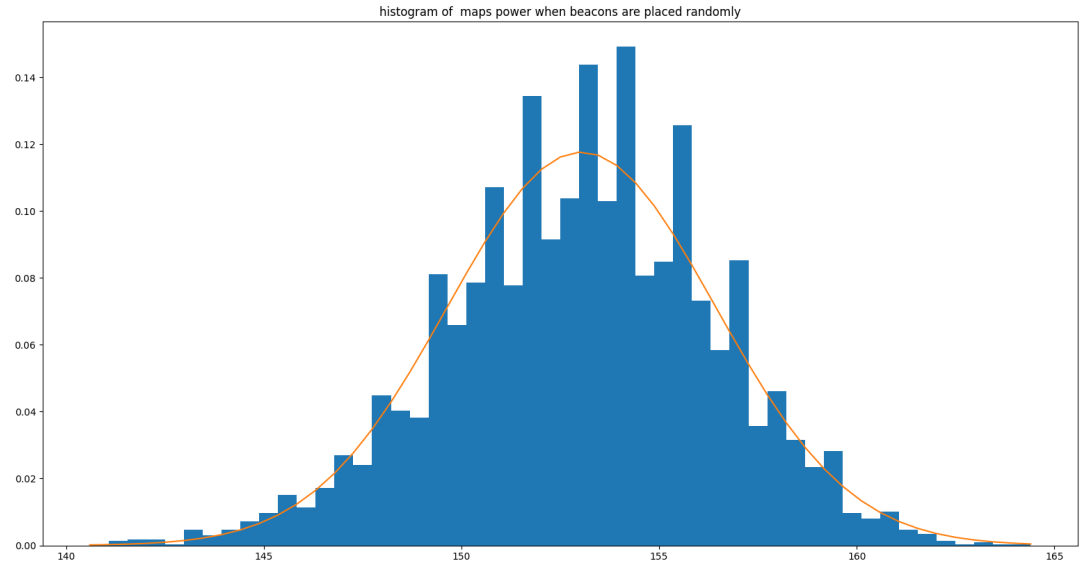


(b) The result of the code on map2, total power from 226 to 315.

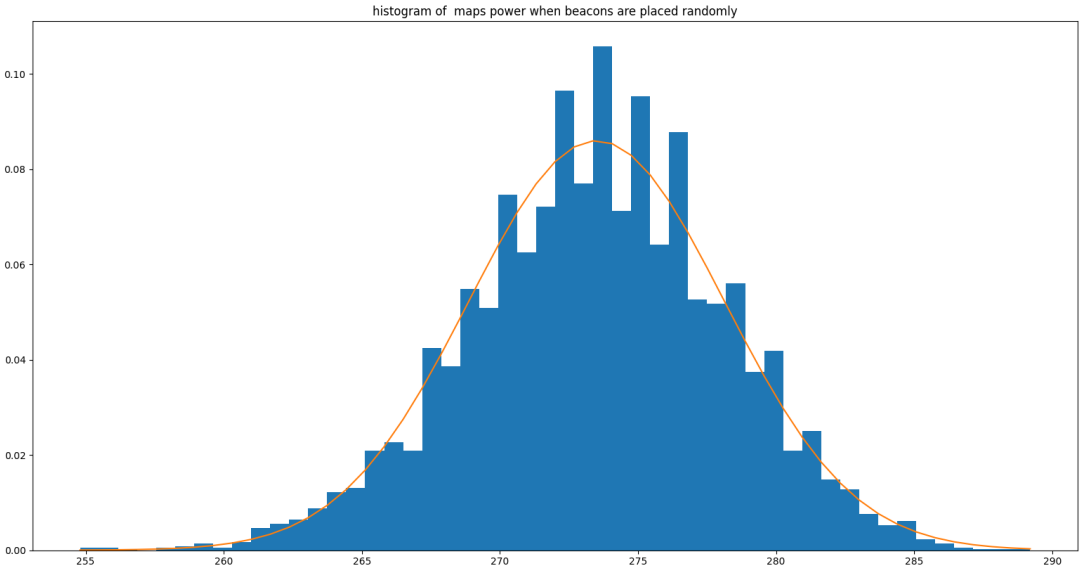
Kuva 4: Maps 1 and 2 after running the code on the map.

5 Random beacon, placement

So, to prove that my solution is good enough, when compared to random placement I created a random beacon placer and let it give the average power created. The results are shown in 5 and as we can see the even the best random placements are not near my other solution.



(a) Histogram of powers when beacons are placed randomly in map 1. Peak at 155 so, nowhere near the other solution



(b) Histogram of powers when beacons are placed randomly in map 2. Peak at 275 so nowhere near the other solution

Kuva 5: Maps 1 and 2 after running the code on the map.

6 Thank you

If you have any question about the code/project, in general feel free to ask me. Also I would be honored to recieve any feedbacb, wherher good or bad. The projects original goal was to apply machine learning and I will update it when I get it finished which would be after my thesis, but even with the machine learning still missing it still works as an excellent example for problem-solving when and python knowledge and even knowledge in statistics. As one famous programmer said "don't play games, make them", to that I would say "don't play games, break them".