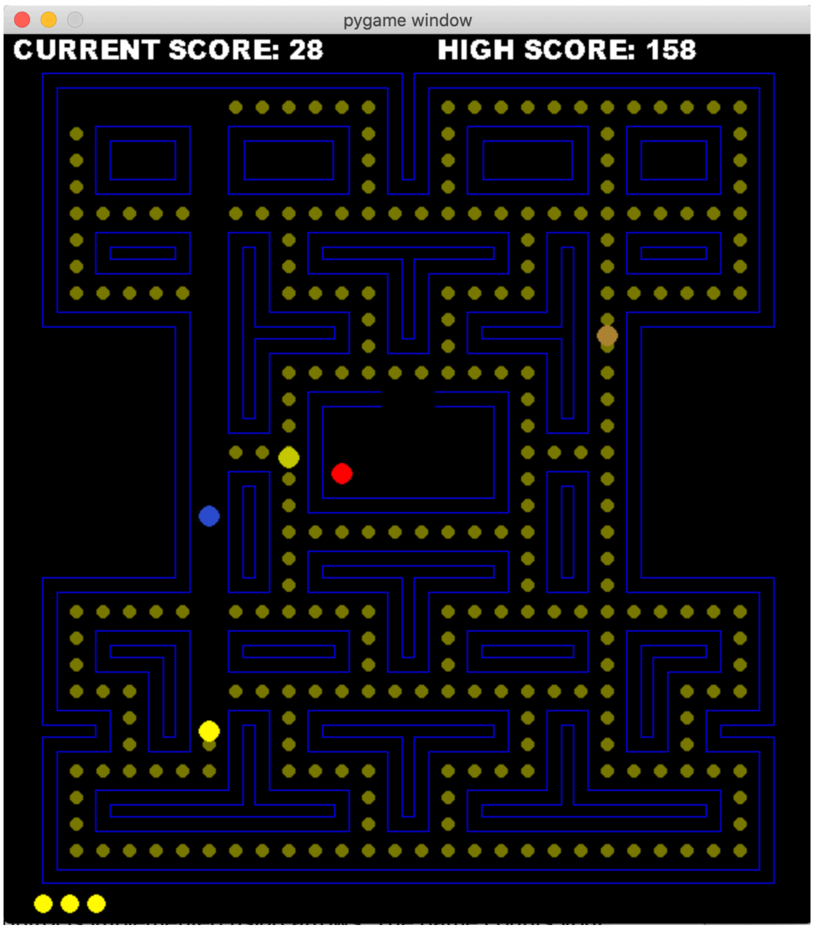
Semester work report

FIT CTU BI-PYT ZS2020



A simplified copy of the 1980 pacman game (Nintendo).
Made in Python version 3.8.3 using the paygame engine version 2.0.0 by FIT CTU student Andrey Cherkasov.

The entire algorithm of the game is built on OOP and is divided into modules. The game class is instantiated from the __main__ module, which runs in a while loop until the player exits the game. The playing field is implemented by a txt foal loaded at startup. Main methods are events, update and draw. Events using engine methods, it tracks keyboard presses and implements control. Update provides various calculations and updating ingame objects. Draw depending on updated data redraws game_screen.

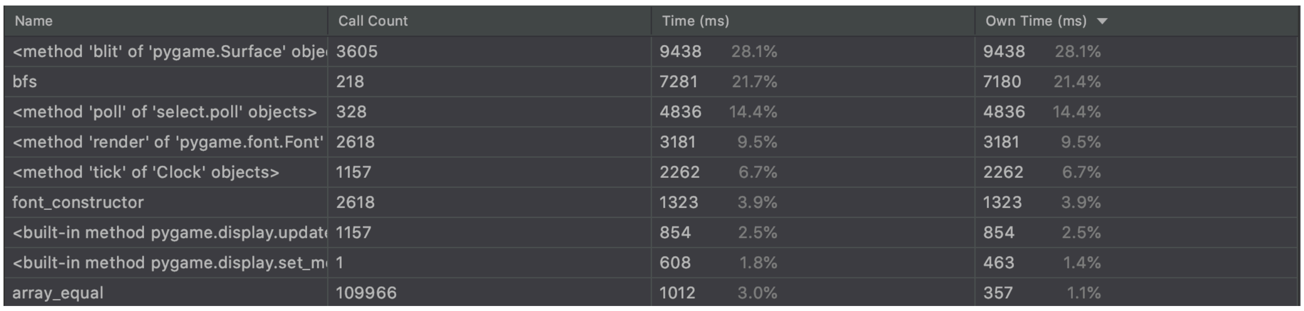
The game has 4 states, switching between which is implemented on the keyboard. Player control is implemented using arrows. The game counts your score and displays it on the screen. The high score is written to a file and updated on load.

To implement correct tracking and calculations of enemies and the player, the field is implemented with a 28 by 30 matrix. For calculations, python lists and numpy arrays are used. The methods of the player and enemies in the while loop calculate the presence of walls and do not allow to leave the playing field.

The game features 4 types of enemies. Speedy, slow, scared and random. Fast and slow find the player's position using the BFS algorithm and follow him with the shortest route. The scared one always goes to the opposite corner from the player. The random one randomly chooses the direction.

The game works almost stable, does not crash. No critical errors were detected.

The main problem with the game is performance. The calculations of the movement of enemies are not optimized, which can cause the application to slow down during loading and during the game.



Name	Call Count	Time (ms)	Own Time (ms) ▼
<method 'blit' of 'pygame.Surface' obje	3605	9438 28.1%	9438 28.1%
bfs	218	7281 21.7%	7180 21.4%
<method 'poll' of 'select.poll' objects>	328	4836 14.4%	4836 14.4%
<method 'render' of 'pygame.font.Font'	2618	3181 9.5%	3181 9.5%
<method 'tick' of 'Clock' objects>	1157	2262 6.7%	2262 6.7%
font_constructor	2618	1323 3.9%	1323 3.9%
<built-in method pygame.display.updat	1157	854 2.5%	854 2.5%
<built-in method pygame.display.set_m	1	608 1.8%	463 1.4%
array_equal	109966	1012 3.0%	357 1.1%

The main reason for this error is poor planning of the algorithms and technologies used. By mixing Numpy arrays with python lists some methods were written according to the principle ‘just work’

Various methods of tracking the position of objects in the matrix, as well as my system of modules, can be used in almost any simple 2D game with a static playing field. (for example bomberman or tetris)

Further, it may be possible to improve performance, as well as add functionality with the evil pacman mode. Also, if modify the algorithm , the game will be compatible with other playing fields, and not only with the original.

Resources used:
https://www.pygame.org/docs/
https://numpy.org/doc/
https://www.youtube.com/channel/UC-pLL_SFEm6B9Eeyr-V424Q

Semester work report

FIT CTU BI-PYT ZS2020