Portfolio assignment 2: Showtime

Group members:

- Nikola Dordevic, s341839
- Jørund Topp Løvlien, s341822

Running the program:

Technology choice

The assignment is programmed in Python 3.9.2
Libraries needed for the client and server to work are gRPC, tkinter and PyGame
To install them run:

```
pip install grpcio
pip install grpcio-tools
pip install pygame
```

Running the game

To get started you first run the server this can be done through a dockerfile or manually through the terminal. The server must be running to play the game singleplayer or multiplayer, else the client will be prompted with an error message.

Afterwards you can run the game as a client. The game supports endless amounts of players.

Example run:
NOTE: Each step is run on a separate terminal.

Step 0: gRPC
To generate the grpc files required to run the client and server, run the command while in the same directory as data.proto, server.py and client.py:

```
python -m grpc_tools.protoc -I./. --python_out=. --grpc_python_out=. ./data.proto
```

Portfolio assignment 2: Showtime

Step 1: Running the server
This can either be done via a dockerfile or manually:

Dockerfile:
In the assignment folder copy over the docker folder to the desired virtual machine

```
# The build name can't be "server"
# Default address is "localhost:10000"
docker build -t <build-name> --build-arg address=<address:port> .
docker container run -it <build-name>
```

Manually:

```
cd <path-to-project-folder>/
py server.py <address:port> # (All in one argument)
```

If done correctly "Server has started on <address:port>", is printed in the terminal

Step 2: Running the client

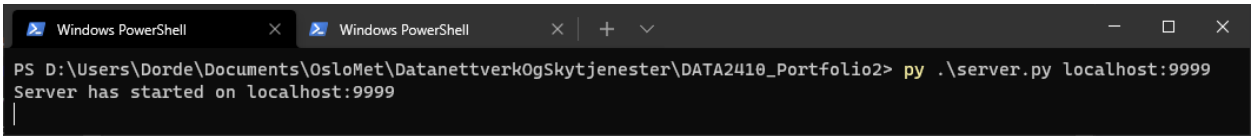
```
cd <path-to-project-folder>/
py client.py <address:port>
```

If done correctly the user should be prompted with a screen looking like figure 1 further down below.

- "Run as bot" checkbox decides if the user joined should be played as a bot
- "Start game" button puts you in the game session.
- "How to play" button displays a window with instructions on how to play the game and control the snake

Image showcase of software:

Server:



Client:

Figure 1: Start menu

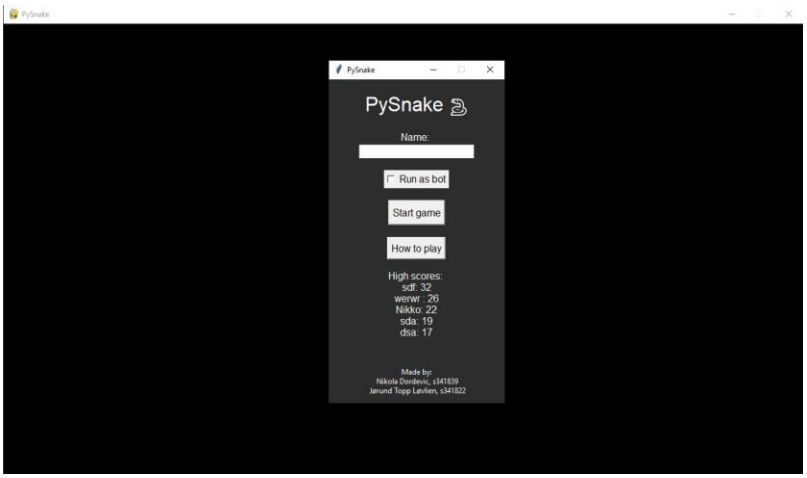
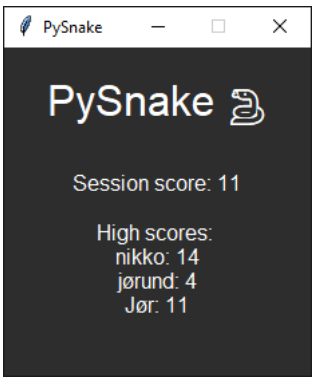


Figure 2: Whilst in a game session

Figure 3: End of a session



Bots:

The game supports bots. To play as a bot you must tick off a checkbox on the menu titled: “Run as bot”. Once that is done, input the name and start the game as normal.

Algorithm

The bots use the A* search algorithm.
Firstly it has information on all the fruits on the board. It compares each fruit by the distance between its head and the fruit:

```
abs(p1[0] - p2[0]) + abs(p1[1] - p2[1])
```

Where p stands for point, and [0] for x, [1] for y, in a point (x, y)
After it has chosen the closest fruit possible it starts calculating the path.

The bot treats every tile on the board as a node, using a linked list. Similarly, to Dijkstra’s Algorithm it compares each path possible by the weight of the distance. However, since each node is part of a grid, we can assume that the weight is exactly 1. Using this knowledge, we compare each neighbor to the snake head using this formula:

$$f(n) = g(n) + h(n)$$

g(n) is the weight between the node and the head, while h(n) is the distance between the new point and the fruit closest to the snake head. What this does is, if the neighboring node we are testing is further away from the point than the other neighboring nodes, we can safely throw it away knowing that it does not contain the fastest path. After finding the closest node, first it assigns the node we came from as a parent then it recursively checks its neighbors until finally reaching the fruit. Finally, we get the path by putting each node’s coordinates into an array and iterating through the linked list via their parents.