SNAQE player's guide

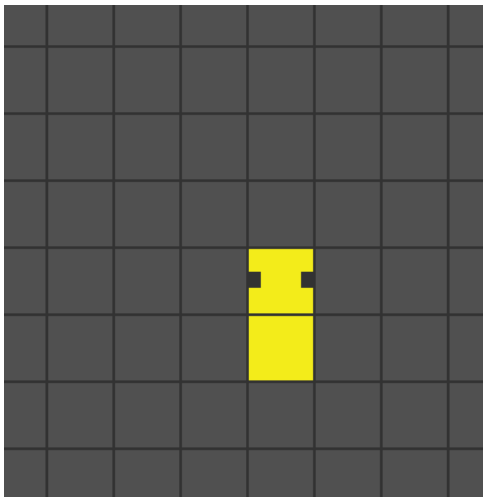
October 1, 2022

1 Running SNAQE

To run SNAQE on a Windows machine click on the file snaqe.exe

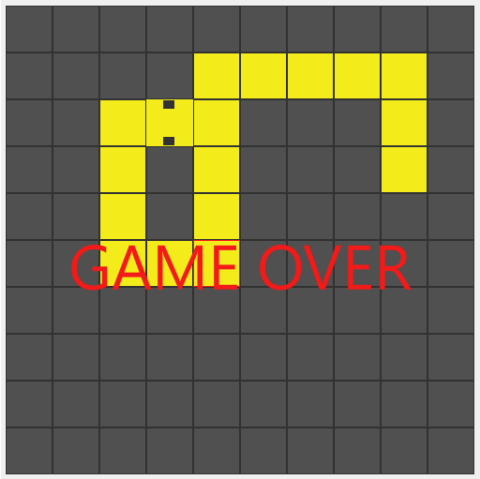
2 Playing SNAQE

The object of SNAQE is to grow your snake. Your snake grows by finding, striking, and eating the hidden prey.



2.1 Moving the snake

Use the keys W, A, S, and D keys to move your snake around the grid. Avoid moving the snake into its own body. If the snake collides with itself you lose!



2.2 Finding prey

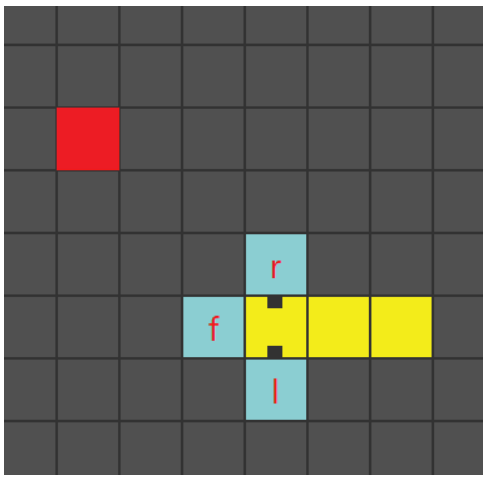
The prey is hidden on one of the grid squares. Your snake is equipped with a probing function that gives it clues to where the prey is hidden (reminiscent to how a real snake uses its tongue). You can choose to probe any square adjacent to the snake's head and will receive back a number telling you how far the prey is from that square. For example you could choose the square in front of the snake. In reply, an oracle with access to the hidden prey location will return, say 5 if the prey is 5 squares away from the square in front of the snake.

In fact, because this is a quantum snake, your choice of square to probe can be a superposition of squares to be probed. Say $|\mathrm{f}\rangle$, $|\mathrm{r}\rangle$, and $|\mathrm{l}\rangle$ represent the squares in front of, to the right of, and to the left of the snake, respectively. You could then choose to probe an arbitrary superposition $\alpha\,|\mathrm{f}\rangle + \beta\,|\mathrm{r}\rangle + \gamma\,|\mathrm{l}\rangle$.

The quantum version of the oracle will perform the linear map

$$|\text{chosen square}\rangle \mapsto |\text{chosen square}\rangle \otimes |\text{distance from chosen square to prey}\rangle$$

For example, consider the configuration

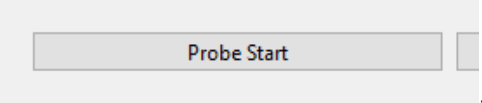


where the red square is the prey (usually hidden). Probing square f or square r would give distance 5, while probing square l would give 7. Probing the *superposition* $\alpha\,|\text{f}\rangle + \beta\,|\text{r}\rangle + \gamma\,|\text{l}\rangle$, would result in the superposition $\alpha\,|\text{f}\rangle\,|5\rangle + \beta\,|\text{r}\rangle\,|5\rangle + \gamma\,|\text{l}\rangle\,|7\rangle$.

The game UI allows the player to perform a limited number of simple quantum operations on this output. First, the distance tensor factor is always measured, with the result of the measurement communicated to the player. After this, the player can choose to perform an arbitrary unitary on the first tensor factor or to measure the first tensor factor. In our example, the oracle outputs $\alpha\,|\text{f}\rangle\,|5\rangle + \beta\,|\text{r}\rangle\,|5\rangle + \gamma\,|\text{l}\rangle\,|7\rangle$. The second tensor factor, representing distance is measured, resulting in 5 with probability $|\alpha|^2 + |\beta|^2$ or in 7 with probability $|\gamma|^2$. Say the measurement results in 5. Then the remaining state would be $\alpha\,|\text{f}\rangle + \beta\,|\text{r}\rangle$ (up to normalization). The user can

then choose to do unitary transformations on this state $\alpha\,|\mathrm{f}\rangle + \beta\,|\mathrm{r}\rangle$ or to measure it. If the player chooses to measure it they will receive the result f or r with probabilities according to the Born rule. If the player does not choose to measure the remaining state, this state is saved and can be used as input for a future probe step. (A future, more complete version of the game will allow the player to do more manipulations on the output of the probe.)

When the player desires to make a probe they click the probe button:

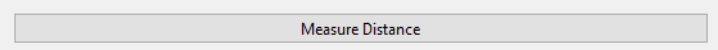


.

The player can enter the coefficients defining a superposition of squares to be probed:

.

Alternatively, if there is a previously saved output state, the player can use that as a probe superposition. Once a superposition of squares to be probed has been chosen, the player clicks



.

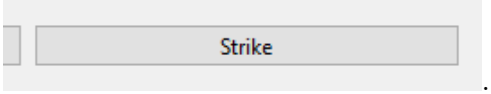
The squares are probed and the distance tensor factor is measured. Options to either enter and perform a unitary on the remaining state or to measure the remaining state then appear. Finally, when a player is finished manipulating the state, they click the continue button.

The player should move the snake around and probe squares until they think

they know where the prey is located.

2.3 Striking and eating the prey

Once the player thinks they know where the prey is they can attempt to strike they prey. Striking the prey means guessing the prey's location. The player clicks on the square they wish to guess and presses the strike button:



.

If the guess is correct the prey is revealed as a red square on the grid. If the guess is incorrect, the snake loses a life. The snake starts with three lives.

Once the prey is revealed, the player can freely move the snake to the prey to eat it. When the snake reaches the prey square, the prey disappears and the snake grows in length by 1.

3 Some details

3.1 Inputting complex numbers

The numbers the player enters for probe vectors or for unitaries can be complex. To enter a complex number the player should use python complex number notation. For example i is notated by

```
1j
```

and $2.57 + 3.89i$ would be notated by

```
2.57+3.89j
```

3.2 Edge cases

If the head of the snake is on the edge of the grid, the number of probeable squares will be less than three. In this case the probe vector will have fewer entries. The unitaries acting on the output will also be smaller than 3 by 3. The player is not allowed to use previously saved outputs as new probe superpositions if the number of entries do not match. E.g. if the player makes a probe while the snake is in the center of the grid, the output will be a three dimensional vector. If the player saves this output, and moves to an edge square where only two squares are probeable, they will not be able to used the saved output as input for a new probe.

3.3 Ways to play

One possible goal is to get each new prey with as few probes as possible. Another is to fill make your snake as long as possible, filling up the grid if possible.