CS3357A COMPUTER NETWORKING

Assignment #3: Server-driven snake game.

Assignment Purpose

The objective of this assignment is to develop a server-based snake game where the server manages the game logic and the client interfaces with the server to interact with the game. The game in focus is the classic Snake game (shown in image below).

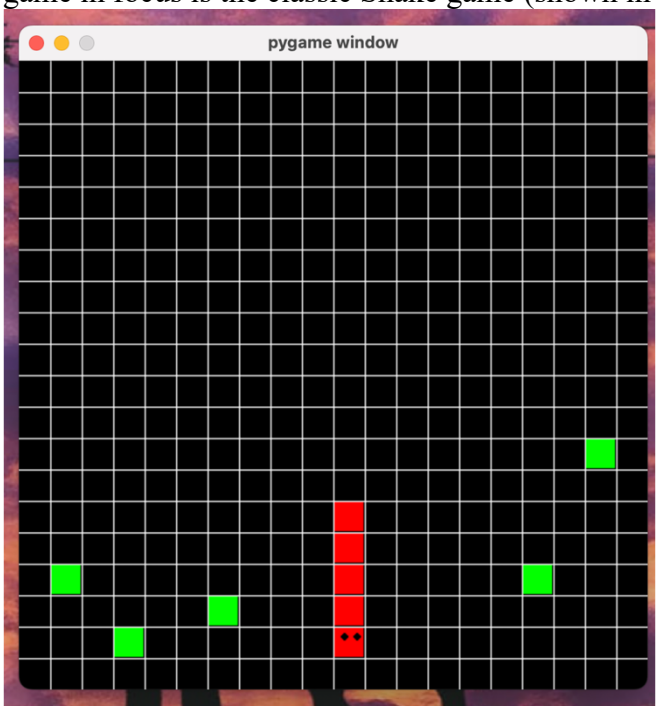

Figure 1 The snake game Graphical User Interface reconstructed at the client side using the game state sent by the server.

Assignment Description

Server-based gaming is a popular service provided over the internet nowadays. World of Warcraft, Conflict of Nations, and Clash of Clans are examples of server-based or server-driven games. In this assignment, you will implement a server-based snake game that is played over the network.
A server-based game is composed of a server side and a user side. The game model lives on the server; this is where the game variables are stored and where the rules to update these variables are implemented and executed. The user machine runs an application that connects to the server, sends the user controls inputs, and periodically receives the updated game state from the server, to draw an interface of the game.

You are given the server-side code (snake.py and snake_server.py), and you are required to implement the client side code (snake_client.py).

Files description

snake.py contains an implementation of the objects that constitute the snake game (e.g. snake, cube, snack) and all the functions that the server needs to run the logic of the game (The snake game is described below).
snake_server.py Performs two tasks simultaneously:

1- Runs the snake game logic including updating snake's movements, collisions, and scores; and
2- Waits for a client to connect on a TCP socket port. When a new client connects, a snake is instantiated and deployed on the board. The state of the game including the position of the snake and the locations of snacks and is sent to the client periodically. And when the client presses any of the defined control keys, the movement command is sent to the server which includes this command while updating the state of the environment before the new state is sent to the client.

snake_client.py (You will implement) should connect to the server, receive the state of the game periodically, and send the appropriate control commands when the user presses a control key to control the snake. The exact commands that should be implemented in the *snake_client.py* are provided in the **game controls** sub-section of the **snake game** section, or they could be inferred from the provided *snake_server.py* file. To reconstruct the game display on the user side, the user should parse the game state sent by the server to extract the positions of the snake and the snacks and draw a graphical interface like the one shown in figure 1. A detailed description of the format of the game state sent from the server is provided below.

The snake game server is responsible for:

1. Accepting a connection from a single client.
2. Receive and parse the controls sent by the client.
3. Managing the game's logic, including recording control inputs, and updating the game state according to the game rules.
4. Sending the game state to the client.

The snake game client is responsible for:

1. Connecting to the server.
2. Sending user inputs (movements) to the server.
3. Receive and parse the game state from the server.
4. Displaying the game interface based on the received game state.

Snake game

The snake game is a game featuring a snake roaming a squared arena; and snacks dispersed across the map. The arena can be viewed as a grid composed of an equal number of rows and columns. The snake is composed of cubes, and the size of each cell is the same as the size of a single cube. The size of a snack is one cell. Figure 1 shows a screenshot of the game interface that the client you are required to develop should reconstruct from the game state received from the server. It shows five snacks (in green) at random locations, and a snake (in red) which is five cubes long.

In the game, the user controls the movement of the snake with the goal of collecting as many snacks as possible. With each snack collected, the snake grows one more cube. In this version of the snake game, the logic of the game runs on the server (implemented in the snake_server.py file) while the interface of the game (shown in figure 1) is displayed on the client side (which you are required to implement). To draw the interface of the game and update it several times each second, the client has to receive the game state (the position and length of the snake, and the location of all snacks). Since the game state is updated each

cycle/loop based on the inputs of the user, the server and the user must exchange the controls and the game state each cycle/loop to play the game.

Game controls

The client script you will implement controls the snake in the game environment handled on the server. For the user/client to be able to influence the state of the game, it has to agree with the server on a set of control inputs each has a specific impact on the game state.

There are seven control inputs that the snake game server understands. The client side should implement these seven control inputs to be able to "talk" to the server. These commands are used by the user to control the snake or the game. For example, changing the direction of the snake, reset, or quit the game. The list of control inputs that you should implement is given below:

1. up: changes the direction of the snake to up/north
2. down: changes the direction of the snake to down/south
3. left: changes the direction of the snake to left/west
4. right: changes the direction of the snake to right/east
5. reset: reset the length of the snake and start from a random location.
6. quit: quits the application.
7. get: a default command. Requests the current game state.

Upon receiving the control inputs from the client, the server parses the control messages, updates the game state (e.g. change the direction of the snake), and returns the updated game state to the client.

Game state

The game state is a set of values that work as a snapshot of the game at a certain moment (like figure 1). This snapshot contains all the information needed to reconstruct the game view on the client side. The string representing the game state is generated and returned using the game.get_state() function in the SnakeGame class in the provided snake.py file. The game state contains the position of the snake (the locations of all its body cubes) and the locations of the snacks.

The client that you will implement should parse the returned game state to extract the values that represent the locations of the snake and the snacks, to reconstruct the game interface, while at the same time sending game controls to the server to update game state and send it to the client.

The game state has a specific format. The client is expected to know the format the server uses to compose its game state and parse the received game state accordingly to extract the information it coveys and construct the game interface.
To show the format of the game state used: the following colored string shows an example of a game state returned by the function game.get_state() on a 20x20 grid, and sent by the server to the client:
(2, 17)*(3, 17)*(4, 17)*(5, 17)*(6, 17)*(7, 17)*(8, 17)*(9, 17)*(10, 17)|(18, 1)**(10, 14)**(12, 1)**(4, 1)**(10, 11)

The shown game state has two sides, a green side and a blue side separated by a "|" character. The first part (in green) represents the positions of the cubes of the snakes. Snakes are separated by "**" while cubes are separated by "*". Since we only have one snake, there is no "**" in the first part. The first part has nine "*"-

separated positions, which means it represent a single snake with length nine cubes. The second part contains the locations of the snacks separated with a "**". Each snack is represented by a single position.

The client script should receive the game state, parse it to extract the locations of the snake and the snacks, and draw the game interface (like in figure 1).

Testing Your Code (Important)

After implementing the snake game client-side script, you should use the provide server-side scripts to run the game by running the server (using the command python snake_server.py) and running the client (using the command python snake_client.py). If your implementation is correct, the game window (such as the image in figure 1) will be displayed, and the user will be able to control the snake using keyboard control keys.

In addition to the client-side script, you are also required to submit a screenshot of the game window (like figure 1) which pops up if the game is working properly.

Deliverables

Submit two files:

1. snake_client.py: client-side code of the game which is compatible with the provided snake_server.py.
2. game_screenshot.png: A screenshot of the game window showing the snake and the snacks (similar to figure 1).

Late submission

- Late assignments will be accepted for up to two days after the due date, with weekends counting as a single day; the late penalty is 20% of the available marks per day. Lateness is based on the time the assignment is submitted.
- Extensions will be granted only by your course instructor. If you have serious medical or compassionate grounds for an extension, you **must** take supporting documentation to the Academic Counselling unit of your faculty, who will contact the instructor.

Rubric

This assignment is out of 40. The assignment marks as follows:

1. Client sends movement commands to the server. 10 marks
2. Client receives the game state and parses it appropriately. 10 marks
3. Client displays the game interface. 20 marks

System Requirements

For this assignment, you will need to install python 3. You will also need a python package manager like pip or Anaconda to install packages such as:

- Pygame: Used for building the game Graphical User Interface
- Socket: Used for socket programming

Resources

An offline version of the snake game can be found here:
https://www.techwithtim.net/tutorials/game-development-with-python/snake-pygame/snake-tutorial-1

Good Luck!