Concurrent Programming - Design Document

Authors

Shephard, Rachael Lorna(sherl005)

Yijian Yao(yaoyy020)

Pang, Bao Xiang(panby020)

Ring, Ring Dut(rinrd002)

Project Description

The purpose of the project is to create an application that runs a multiplayer game of snake's by simulating a multiplayer gamer server. The user will be required to login or create an account in order to get into the lobby where they will be presented with a list of available games to join. Since application will not include networking code, there will only be a single game in the lobby; the one on the local machine. In the game the players take control of a single snake, they must grow their snake and increase their score points by eating fruit while avoiding other snakes, their body and the wall.

Project Objectives and Success Criteria

The project shall be considered successful if it meets all the following objectives.

- **Performance**: The game has the typical functionality of a snake's game and is playable by 4 human players concurrently and 100 simulated players.
- **Scope**: A fully functional snake game with multiplayer capabilities is to be created, however the multiplayer game server which would host these games will be simulated with no networking code.
- **Cost**: The project costs nothing to complete.
- **Time**: The project is completed by the 14th of June, the end of week 13.

Requirements

ID	Requirement Name	Description
F01	Login	Players shall be able to login with a username and password in order to play
F02	Register	Players can create an account which shall be saved on an external database
F03	Number of players	The game shall support up to 4 human controlled snakes and.
F04	Snake control	Each human player shall have a unique key-mapping to control their snake
F05	Number of NPCs	The game shall support up to 100 simulated snakes
F06	Snake game functionality	The players shall earn points from eating fruit which causes their snake to grow by one unit
NF01	Simple interface	The interface shall be simple enough to navigate without instruction
NF02	Source code	Source code and all changes made to it shall be tracked on a private Git repository
NF03	Login threads	Logins must be thread-safe on the server
NF04	Concurrent login threads	The concurrent login code must not use a thread-safe standard Java collection to pass messages between the player logging in and the server [1]
NF05	Player-server communication	The system must use thread safe collections for the communication channels between players and the server for game playing and for the storage of the game state. [1]

NF06	Thread pools	The System must thread pool for game player interactions based on the java.util.concurrent.executor model. [1]

Constraints

- **Scope:** The project is constrained by the scope specified identified within the project report, in this case it is limited to creating a functional snake game with local multiplayer capabilities.
- **Cost:** No money has been allocated to complete the project, constraining the potential solutions to ones which incur no cost.
- **Time:** All solutions developed within the project are limited to ones which can be implemented by the 14th of June (2 months).

Assumptions

- The user understands the rules of a game of snake
- The user will not forget their username or password
- The game will be run on a device with a QWERTZ keyboard

Rules

The rules we chose are only slightly changed from the traditional snake rules:

- The snake will keep moving and cannot stop.
- Snake can change the direction of travel (up, down, left, right).
- Snake cannot immediately reverse direction (ie, turn up or down or turn left or right).
- If the snakes do not change the direction of travel, they will continue to advance in the same direction.
- When the snake crosses the boundary line, he will come out from the other side
- If the head of one snake collides with the body of another snake, it will die.
- If two snake heads collide, they will both die.
- Only when the snake head touches the fruit, it will be judged as eating fruit.
- Each fruit will increase the length of a unit for the snake
- When there is only one snake left or the length of a certain snake reaches 20, the game ends and the winner appears.
- There will only be one fruit at the same time in the game. When the fruit is eaten, the next fruit will be randomly generated in a new place

Project Risks

ID	Risk Description	Prevention	Likelihood	Impact	Severity
R01	Chance of not fulfilling all the technical specifications and requirement requested.	Implement a checklist system of specification and requirement requested by the client therefore having the ability of keeping track of the project's progress which help minimizes the risk as much as possible.	Possible	Major	High
R02	User registering with an account name that already exist in the database which may cause conflict in the database.	Implement a code to prevent user from registering with existing account name.	Likely	Major	High
R03	User logging in with an account that is already logged in.	Implement a code to prevent user from logging in if the account is already logged in regarding if the password was entered correctly.	Unlikely	Moderate	Medium

Class Specifications

Snake

Representable running objects for snakes in the game. In the game, the player can only control the movement (direction) of the snake's head, but the snake still contains the head and body. During operation, the player changes the direction of the snake head to control the movement of the snake by pressing the buttons, and then the whole snake moves in this direction. The snake is divided into player control and computer control. If the snake is controlled by the player, it will not randomly change its direction. The direction can only be changed with the arrow keys. If controlled by a computer, the computer will give the snake random directions at random times. The snake will calculate its next position based on its direction, check the collision and act accordingly (if it collects food, her body length will increase by one grid from the tail; if the snake head hits the border or other snake's body, it will die).

Client

Client class allow a client or user to login to this game and play, client need to use the correct username and password, also the login event need to be passed by a buffer.

MPGS

This class will create a lobby and allow client to join, define how many Simulated player there will be (from 4 to 100)

Map

This class will limit the scope of the area and define the coordinates.

Fruit

This is a game mission prop. Only one fruit will appear on the map at the same time. When the head of a snake touches the fruit first, it will increase in length, and then the new fruit will appear at a random position on the map (no Repeat with snake body and snake head position)

Buffer

At the top is the physical memory storage area, which is used to temporarily store data when it is moved from one location to another. It has the largest allocated space. If the client tries to attach the login event to the shrink when the lock is full, it will continue Wait until the space in the bribe is available

Game class

Determine the game rules and victory conditions and make judgments

Login

The login server class is operational and is used for accounts that have already been authenticated and passwords that match it. Correct, can allow players to successfully enter the game.

Register

When the player cannot log in at the login interface, there will be a registration window at the bottom of the interface. After the player creates the account password in the player, it will be checked to ensure that the account name is not duplicated. Then the server will send the data to the buffer to save the record Login normally

Modelling

SNAKE = (move->sendmove->wait->updategame->SNAKE).

GAMESERVER = (receivemove->processmove->sendupdategame->GAMESERVER).

||SNAKEUSER1 = (player1:SNAKE).

||SNAKEUSER2 = (player2:SNAKE).

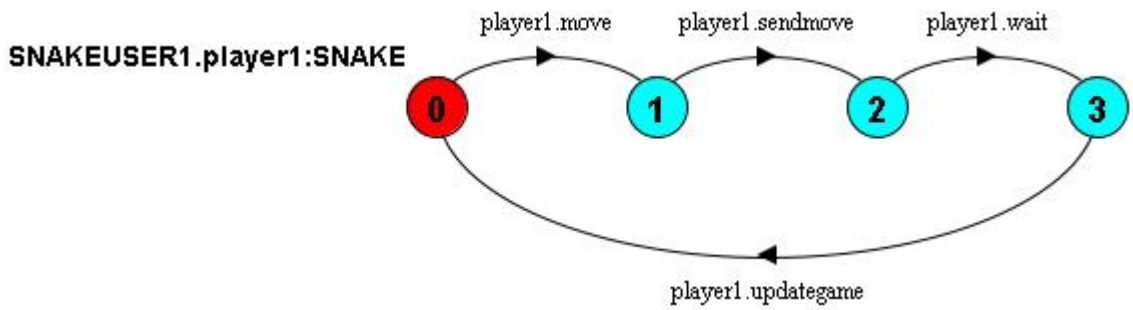
||SNAKEUSER3 = (player3:SNAKE).

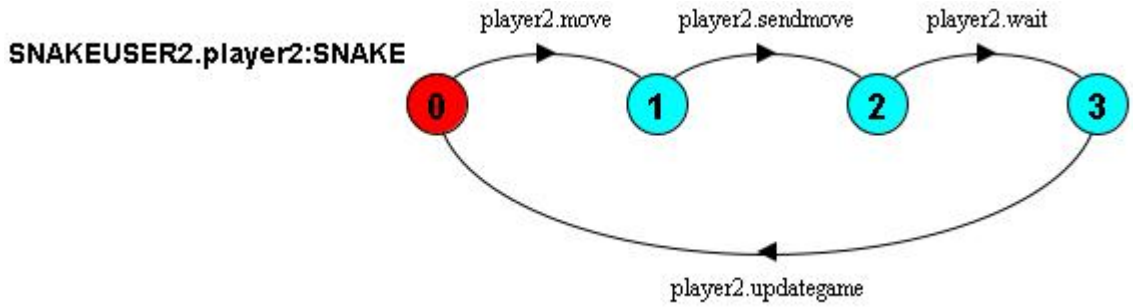
||SNAKEUSER4 = (player4:SNAKE).

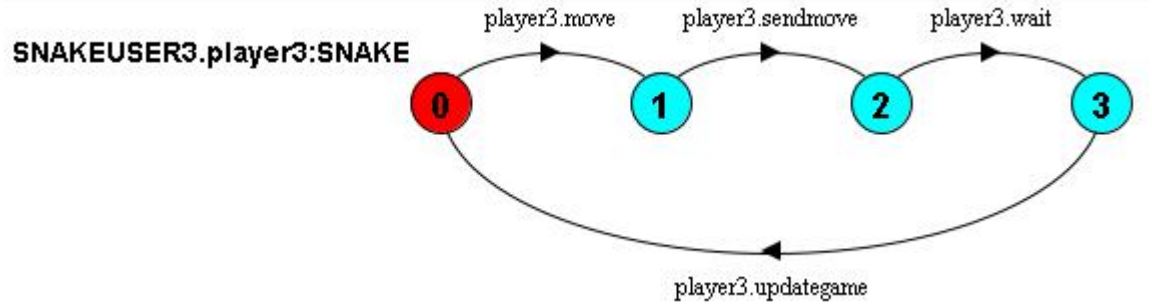
||SHAREDGAME = ({player1, player2, player3, player4}::GAMESERVER).

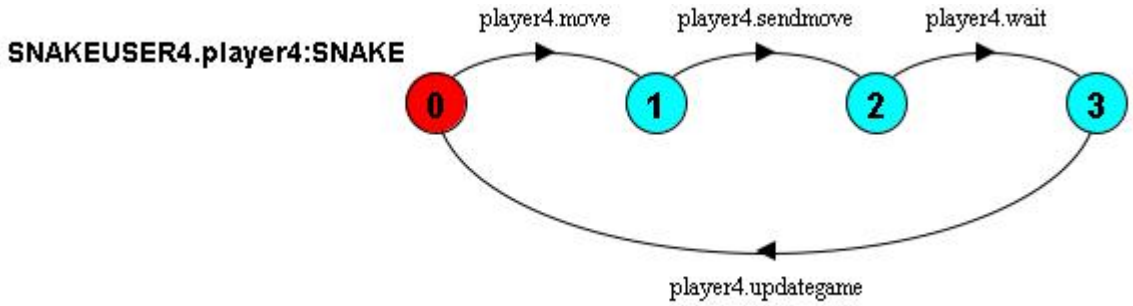
||GAMECLIENT = (SNAKEUSER1 || SNAKEUSER2 || SNAKEUSER3 || SNAKEUSER4 || SHAREDGAME).

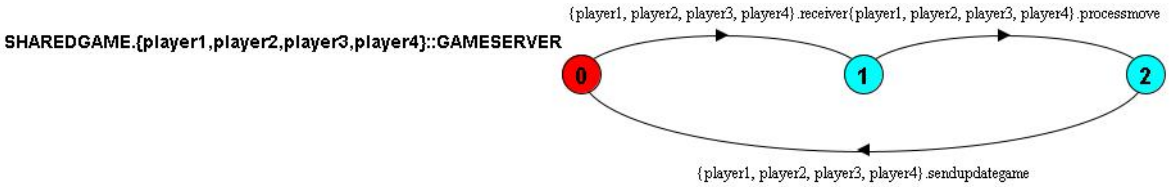
||SIMULATEDNETWORK = (GAMECLIENT || GAMESERVER) / {sendmove/receivemove, sendupdategame/wait}.





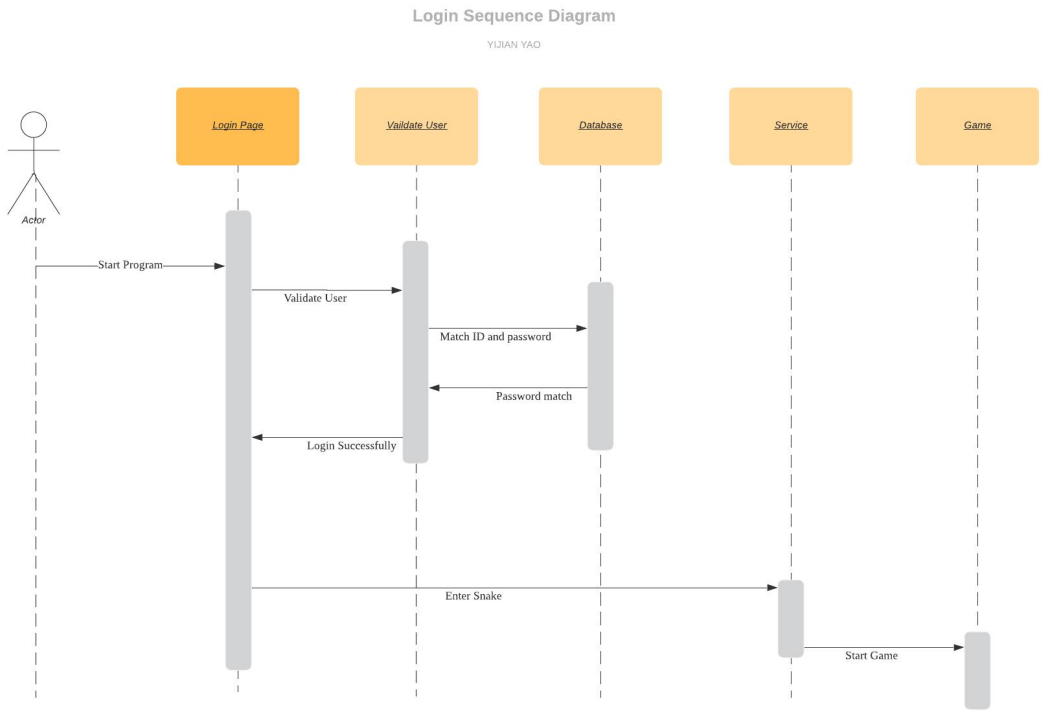



SNAKEUSER4.player4:SNAKE
player4.move
player4.sendmove
player4.wait
0
1
2
3
player4.updategame

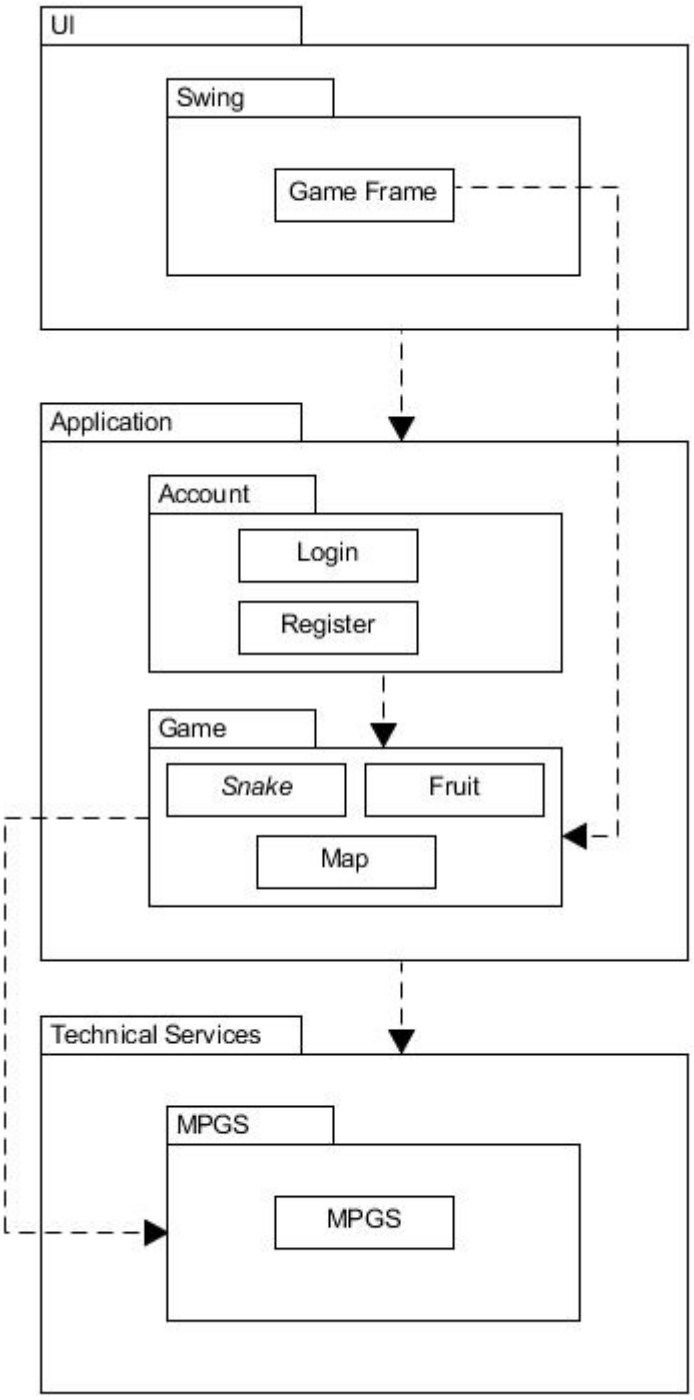

SHAREDGAME.{player1,player2,player3,player4}::GAMESERVER
{player1, player2, player3, player4}.receive{player1, player2, player3, player4}.processmove
0
1
2
{player1, player2, player3, player4}.sendupdategame

Login sequence diagram

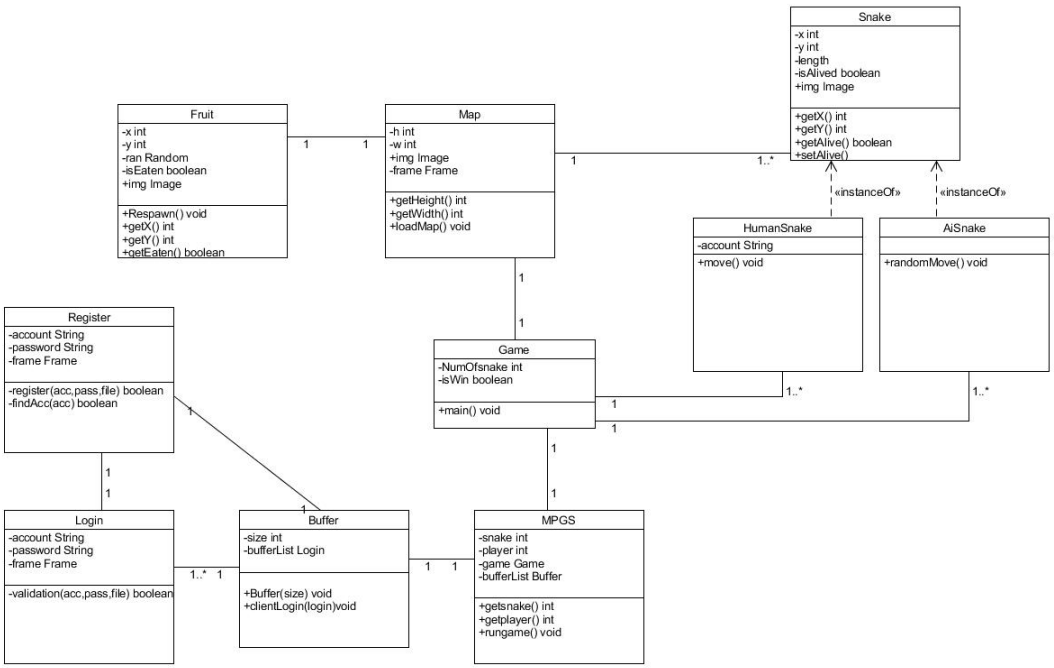
The player enters the login interface and starts to enter the account password to log in. If the account password matches, you can successfully enter the game, otherwise the player needs to enter the registration interface and register a new account and password, and then enter the game after success



System architecture layered package diagram



Class design diagram -UML



Java Class Assignments/Allocations

- MainGame:

Pang, Bao Xiang(panby020)

- Client:

Pang, Bao Xiang(panby020)

- Snake:

Shephard, Rachael Lorna(sherl005)

- LoginGUI:

Yijian Yao(yaoyy020)

- SnakeGUI:

Shephard, Rachael Lorna(sherl005)

- RegisterGUI:

Yijian Yao(yaoyy020)

- Buffer:

Shephard, Rachael Lorna(sherl005)

- Login:

Yijian Yao(yaoyy020)

- Register：

Yijian Yao(yaoyy020)

- Fruits：

Shephard, Rachael Lorna(sherl005)

- Mpgs:

Ring, Ring Dut(rinrd002)

- Map：

Ring, Ring Dut(rinrd002)