CS 242 Final Project Proposal
<Pac-Man>

<Xinhang Chen> (<xinhang2>), <ZhengRu Qian>
<Yabo Li> (<yaboli2>), <Anjal Menoni>

1. Abstract

1.1. Project Purpose

Create a Pacman game for users to play.

1.2. Background/Motivation

This is the first time for us to create a game , especially a mobile game. This game is basically built from scratch without using any powerful game or networking library. However, this has been our motivation to become a programmer. In this course, we have been equipped with knowledge of mobile development and web development so it may be the best time for us to realize our ideas about gaming. Besides, a mobile game can be a suitable form to represent our ideas and skills. To perfect and optimize a game could be a nice practice of our coding skills.

2. Technical Specifications

2.1. **Platform:** Android App

2.2. **Programming Languages:** Java, Golang

2.3. **Stylistic Conventions:**commenting, naming conventions, camelCase

2.4. **SDK:** Android Studio SDK

2.5. **IDE:** Android Studio, Visual Studio Code, Goland

2.6. **Tools/Interfaces:** Junit

2.7. **Target Audience:** Android Users

3. Functional Specifications

3.1. Features

- Multi-player in one room to compete for a winner
- Low-latency among multi-players
- AI for ghost
- Randomly generated maze and food

3.2. Scope of project:

Can only support Android users

4. Timeline:

4.1. Week 1

4.1.1. Create a custom protocol for communication between client and server. (Yabo Li)

● Accept player JOIN request. Return randomized ID, unique color, and initial positions for PLAYER and FOOD. (TCP)

● Push updated player's position data to different clients continually. (UDP)

● Accept player EAT request and handle potential concurrency; Reply response and updated score data. (TCP)

● Push newly created FOOD data continually. (TCP)

Basic two-player game logic with user interaction. User can control pacmans' movements (Xinhang Chen)

● Send request to server and get player's ID , color , and initial position.(TCP) Render pacmans on canvas.

● Receive updated players' positions from server and render pacmans on canvas.(UDP) Pacman move on screen continuously.

● Keep sending updated players' positions after user interactions.

● keep rendering when both players enter the game

Functional Requirements (Xinhang Chen)

Requirement - Initial set up	2	1 points: After connecting to the server, players show up on the screen with randomly generated initial location and colors via server
Requirement - receive and send updated players' location	5	2.5 points: Open two devices and both devices can see other player's movements updating.
Requirement - User control	3	1.5 points: User can make pacman move by touching the screen.
Requirement - rendering	5	2.5 points: Keep rendering the game after both players join the game

Testing Requirements (Xinhang Chen) No testing 2.5 points unitest for sending data and receiving data Testing - Manual Test plan 5 0 points No testing 2.5 points manually test the GUI and have a test report in PDF

Testing Requirements (Xinhang Chen)

Testing - Unit testing	5	0 points No testing 2.5 points unitest for sending data and receiving data
Testing - Manual Test plan	5	0 points No testing 2.5 points manually test the GUI and have a test report in PDF

Functional Requirements (Yabo Li)

Requirement - Custom protocol design	2	0 point - No custom protocol design 2 points - Custom protocol with detailed explanation
Requirement - Handle player's JOIN	2	1 point - Accept and parse correct data from user. 2 points - Reply correct the required data.
Requirement - Synchronization	6	4 points - Push player's position continually; create new food and push accordingly. 6 points - Handle concurrent EAT or MOVE cases.
Requirement - Handle player's EAT	5	2.5 points - Accept and parse correct data from users. 5 points - Reply correct responses to different players. Store player's data properly.

Testing Requirements (Yabo Li)

Testing - Unit Tests	5	0.0 points - No testing 2.5 points - 80% coverage using unit test and libraries 5.0 points - 95% coverage using unit test and libraries
Testing - Manual	5	0.0 points - No testing 5.0 points - Detailed manual test plan for communication

4.2 **Week 2**

Start rendering mazes and food. More users can join in the game. Pacman can eat food and get scores. (Xinhang Chen)

Automatically generate mazes to the graph. More users can join in the game. (Yabo Li)

- Use data structure and algorithm to generate different mazes each time.
- Shape each maze wall to a rectangle.
- Generate newly created food, not on the wall.

Requirement - Render food	5	2.5 points: players can see and eat food by controlling pacman's movement. Newly generated food is rendered.
Requirement - Render Maze	4	2 points: render the generated maze from server correctly on screen
Requirement - score updating	3	1.5 points: send update to server when one of the players eat the food and get score back
Requirement - multiplayer	3	1.5 points: More players can join in the game

Testing Requirements (Xinhang Chen)

Testing - Unit testing	5	0 points No testing 2.5 points unitest for sending data and receiving data
Testing - Manual Test plan	5	0 points No testing 2.5 points manually test the GUI and have a test report in PDF

Functional Requirements (Yabo Li)

Requirement - Generate maze	7	2.5 points - Generate static maze each time 5 points - Generate different maze each time 7 points - New maze data contains rectangle info.
Requirement - Generate food	3	5 points - Generate new food that is not in the maze
Requirement - Pass maze and food data	5	5 points - Pass maze data and food data accordingly

Test Requirement (Yabo Li)

Testing - Unit test	5	5 points - Provide a test plan for newly added functions
Testing - Manual test	5	5 points - Provide a manual test plan for the client side

4.2. Week 3

Add Ghost in this game which is controlled by computer. Refurbish UI. (Xinhang Chen)

Requirement - Render Ghost	5	2.5 points: players are chasing by ghosts
Requirement - Lose and Win	5	2.5 points: Basic end condition for win and lose and let player know whether they are winning or losing
Requirements - Modify some of the UI make pacman nicer...	5	2.5 points: Pacman can have animation and the walls would have nicer looks

Testing Requirements (Xinhang Chen)

Testing - Unit testing	5	0 points No testing 2.5 points unitest for sending data and receiving data
Testing - Manual Test plan	5	0 points No testing 2.5 points manually test the GUI and have a test report in PDF

Add Ghost in this game which is controlled by computer. Add game logic. (Yabo Li)

Functional requirement

Requirement - Ghost	3	5 points - The ghost can chase other players in the maze; touched player will lose score
Requirement - Game logic	5	5 points - Judge the end of the game; players with 0 score will los; players with the highest score will win
Requirement - Ghost algorithm	2	2 points - Implement algorithm for the ghost to chase players instead of simple distance comparison.
Requirement - Refinement	5	5 points - Refine data structure; Refine the size of

		payload in the communication; Refine data passed on the protocol

Test Requirement (Yabo Li)

Testing - Unit test	5	5 points - Provide a test plan for newly added functions
Testing - Manual test	5	5 points - Provide a manual test plan for the client side

5. **Future Enhancements**

Chat functions, more delicate UI and graphics, multiple isolated rooms…

6. Custom protocol

TCP

1. User info

   When a user joins the game, it will receive its own information which includes ID, initial position, and color.

   USERINFO;[json format data]\n

   JSON format data:

   {ID, X, Y, Color:}

2. New user

   When another user joins the game, all users will receive the notification which includes ID, initial position, and color.

   NEWUSER;[json format data]\n

   JSON format data:

   {ID, X, Y, Color:}

3. New food

   When a portion of food was eaten, there will be newly created food distributed.

FOOD\n[json format data]\n

JSON format data

{ID X Y}

4. Score
5. Maze