Multiplayer Pac-Man

Distributed Systems – Project Proposal

Student One, Student Two, Linus Fessler
ETH ID-1 XX-XXX-XXX, ETH ID-2 XX-XXX-XXX, ETH ID-3 14-924-203
one@student.ethz.ch, two@student.ethz.ch, fesslerl@student.ethz.ch

1. INTRODUCTION

Since 1980, the game Pac-Man has fascinated players around the world. Starting as an arcade game, it was adapted for many platforms while technology was improving and is still played today on mobile devices. While graphics have improved significantly over time and new features were added to the original idea of the game, there is still a major drawback to most of its versions as the game only provides single player user experience.

Our goal is therefore to exploit the opportunities of modern mobile devices in order to create a new user experience. This new user experience will consist of the well known Pac-Man game combined with a distributed multi player approach. The idea will be implemented as an android application allowing the user to create a new instance of the game by starting a server on an android device. Other users can then connect to this server with their own devices.

2. SYSTEM OVERVIEW

Figure 1 gives an overview of the Pacman app Activities. The app starts in the Main activity. From there, the user can choose to either edit the game settings, host or join a game by pressing corresponding buttons.

If the user presses the "Settings" button, the app will switch to the Settings Activity, where the devices local IP address is listed and the following properties can be changed:

1. The username that will be used in the game.
2. The port on which the local server would listen to.
3. The IP address and port on which the app would try to connect to a server.

If the user chooses the "Host" button, the app will switch to the Lobby Activity (Host). Here, the user can await new clients and see which ones are currently connected (identified by usernames). Furthermore, the user can decide to start the game by pressing a "Start" button. This will launch the Game Activity and start the actual game.

If the user chooses the "Join" button, the app will switch to the Lobby Activity (Client). Here, the user can see the host's username and which clients are currently connected (identified by username). In contrast to the host's view, the client can not see a "Start" button, but instead has to wait for the host to start the game.

Independent from the device's role as host or client, the app will automatically switch to the Score Activity after the game has ended. This Activity displays each users score (determined by the gameplay performance). The only following action is to return to the Main Activity by pressing a button.

Figure 1 gives an overview of the Pacman app. The starting activity is the Main activity. From there the user can either choose to host a game or to join a game. In the former case, the user needs to specify in the Settings activity to which port should the other players connect. In the latter case, the user needs to configure in the Settings activity the IP address of the host and the port number of the host. Furthermore, before joining (or hosting) a game, the user should specify an (unique) name.

After choosing to join a game, the app will switch from the Main activity to the Lobby activity. From there on, the user/app needs to wait for the host to start the game. Of course, if the user has not configured his/her settings appropiately (for instance, invalid port number for the host), then the app will notify the user. Conversely, if the user is the host, the app will make sure he has a (valid) port number. Moreover, it should check that all names are unique. If there is a name clash, the app can and should kick those players out. Once the host is ready, he/she can start the game.

Once the game started, a Score activity keeps track of the current performance of the players. If Pacman (i.e. the host) manages to take all the points that are positioned on the map, he wins the round and gets one point. Otherwise, if a ghost manages to capture Pacman, then the ghost gets one point. [??? is this the way we should keep track of the score? any other proposals?]

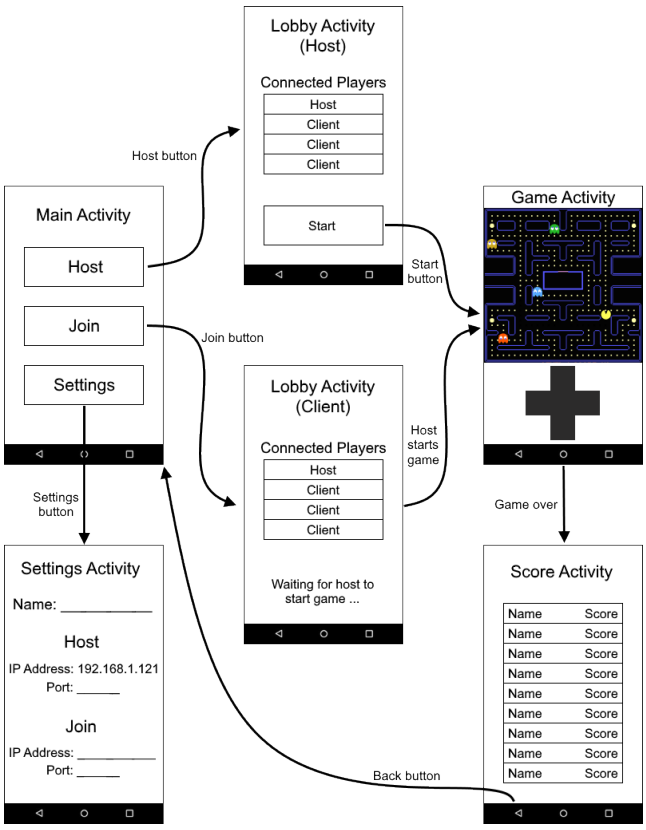

Figure 1: Activity Overview

3. REQUIREMENTS

1. The game can be played on multiple Android devices (at least two).
2. The gameplay should work as follows:
   (a) Coins are distributed evenly on the game map (board).
   (b) One player plays as Pac-Man.
   (c) One or multiple other players play as ghosts.
   (d) Pac-Man wins, if he collects all coins on the map.
   (e) The ghosts win, if they capture Pac-Man (simply modelled by collision).
3. Each player must use one Android device in order to control his figure (Pac-Man or ghost).
4. The map (board) on which the players move should provide the following features:
   (a) Pac-Man starts on a predefined location (Pac-Man spawn).
   (b) The ghosts (one ore multiple) start on predefined locations (ghost spawns).
   (c) Player figures can only move up, down, left and right.
   (d) The only structuring elements of the map are walls.
   (e) Walls have eiter horizontal or vertical orientation.
   (f) Player figures can not move through walls.
   (g) The map has rectangular shape and is limited by walls at its borders (horizontal walls along the left and right border, vertical walls along upper and bottom border).
   (h) Every position at the map that is not occupied by a wall must be reachable for the player figures.
5. The protocol used for Device-to-Device communication will will be implemented atop UDP, TCP or a higher-level protocol. [???]
6. The app will be optimized for Android version [???] and run on Android version [???] and higher. [???]
7. The host will play as Pac-Man.
8. Clients will play as ghosts.
9. On the hosting player's device, a service will be started to run the server which the clients can connect to.
10. As a game development engine the ANDEngine is used. This is a free Android 2D OpenGL game engine [1].

4. WORK PACKAGES

- **WP1**: Define appropriate model for game situation (state) and methods to change the state.
- **WP2**: Implement a map generator.
- **WP3**: Design graphical representation for Pac-Man player figure.
- **WP4**: Design graphical representation for ghost player figure.
- **WP5**: Design graphical representation for map elements (coins, walls, unoccupied positions).
- **WP6**: Implement Game activity that connects model and graphical representation. It should also allow the user to control its figure.
- **WP7**: Define a communication protocol to synchronize the game's state across devices. Choose appropriate transport protocol.
- **WP8**: Implement the communication protocol - part 1: Server.
- **WP9**: Implement the communication protocol - part 2: Client.
- **WP10**: Combine Game activity with the communication protocol.
- **WP11**: MainActivity, LobbyActivity, SettingsActivity, ScoreActivity design.
- **WP12**: MainActivity functionality:
  1. Host button starts LobbyActivity with boolean intent extra "is host" set to true to then enable Start button and disable "Waiting for host to start game" text in LobbyActivity.
  2. Join button starts LobbyActivity with boolean intent extra "is host" set to false to then disable Start button and enable "Waiting for host to start game" text in LobbyActivity.
  3. Settings button starts SettingsActivity.
- **WP13**: LobbyActivity functionality:
  1. ListView to list all connected player (might needs to be scrollable depending on how many player we allow in one game).
  2. Host can press Start button to start game for all players.
- **WP14**: SettingsActivity functionality:
  1. Name EditText: default name is "Anonymous" or device name.
  2. Host IP Address TextView: Shows the device's LAN IP Address.
  3. Host Port Address EditText: Allows to change the server's port.
  4. Join IP Address EditText: Allows to change the IP Address to connect to when joining a game.
  5. Join Port Address EditText: Allows to change the port at which to connect when joining a game.
- **WP15**: ScoreActivity functionality:
  1. ListView (possibly scrollable) to show Highscores list of the host (Pac-Man) which is saved locally on the host's device. When the game ends, every player will land in the ScoreActivity, receive the list from the host and display it.
  2. The Back button returns to the MainActivity.

5. MILESTONES

Work package distribution:

- **Stefan Oanceas**:
- **Johannes Beck**:
- **Linus Fessler**:
- **Markus Hauptner**:
- **Tiziano Zaramoni**:
- **Florian Morath**:

6. ADDITIONAL AUTHORS

7. REFERENCES

[1] AndEngine. http://www.andengine.org/. Accessed on 16 Nov 2016.