AirHockey 3X

Distributed Systems – Project Proposal

Basile Maret, Philipp Rimmle, Etienne de Stoutz
Oliver Butz, Raphael Schnider, Valentin Venzin
ETH ID-1 13-937-974, ETH ID-2 13-913-595, ETH ID-3 13-920-418
ETH ID-4 XX-XXX-XXX, ETH ID-5 13-933-205, ETH ID-6 13-916-895
bmaret@student.ethz.ch, primle@student.ethz.ch, etienned@student.ethz.ch
oliverknu@student.ethz.ch, sraphael@student.ethz.ch, vvenzin@student.ethz.ch

ABSTRACT

We present a bluetooth based multiplayer game called AirHockey 3X. The game is based on the real-life game air hockey. In contrast to the original game, our adaption allows from two to four players to play together. The game is supported on Android™.

1. INTRODUCTION

Air Hockey.

Let us start by a few words of explanation about the game itself. Air hockey is played on specific air hockey table. The table consists of a big smooth surface, which is surrounded by a rail. There is a slot at either end of the table. Furthermore, there is a machinery which generates a cushion of air to reduce the puck's friction and increase its speed. Two players play against each other. Their main objective is to play the plastic puck into the opponents slot by using a mallet.

Our Adaption.

The main conceptual difference between our adaption of air hockey and the original game is the following: We allow for more than two players. In the case where three or four players decide to challenge each other, the pitch changes from rectangle to triangle or to a square respectively. The idea is that the participants of a specific instance of a game sit in a circular fashion next to each other. We recommend sitting around a table. Compare Figure 2.

Each player has his own mallet on his screen and may play the virtual puck to any of the other players. If player A play the puck to player B the puck leaves player A's screen and enters player B's screen after some delay for traversing the invisible pitch between the mobiles. Note that it is important for the players to remain in the same relative position to each other during the game.

Distributed Systems Component.

We base our underlying communication layer on a simplified peer-to-peer bluetooth network. In particular we distanciate ourselves from the perhaps more popular client-server model. In our setting, each player represents a node witch equal rights. The advantage of this approach is **[[(?)...]]** that each participant can run the same protocol and uses the same amount of resources.

Important aspects are to synchronize global state changes, e.g. player scores or wheter the game is being paused, among all participants and to send the information about the puck, i.e. speed or angle of incdence, to the right opponent. The challenge is to design a reliable protocol which guarantees real-time constraints and therefore makes a and dynamic user experience possible. **[[(?)...]]**

2. SYSTEM OVERVIEW

2.1 App Architecture

It follows a short description of the app flow. The reader may refer to Figure 1 for a visual representation. This section shall provide insight in how the application is set up and picture the effects of the most likely series of events.

Start Screen.

Upon opening the app, the user sees a start-screen (1) with two main buttons, namely the *start* button and the *settings* button.

Settings Screen.

Pressing the *settings* button, leads to the settings-screen. The user may select his usernamer which gets shown to other players. Furthermore, **[[(?)...]]**

Ready Screen.

If the user hits the *start* button, he gets directed to the ready-screen (2). The ready-screen serves the purpose of setting up the new game. Each player is first asked to select the number of players. As soon as the choice is done he sees, a schematic topology of the players next to each other, himself being at the bottom. This schema helps new users to understand how the game is to be played; compare Figure 2. The user needs to select his opponents in order to start the game. He does so, by tapping on each of the other players on the schema, and selecting his Bluetooth device from list, which appeard next to the player. All players must add all of their opponents; this way we can make sure that all the players consent to play against each other. Note at this point the neccessity that player A needs to add player B in the same relative position. For example, if player A adds B to the right B must add A to the left. If an error occurs, which includes a wrong topology such as A left to B and B left to A or not all of the players have the same list of participants, a respective error message is displayed. If everything went fine, meaning all players have the same player topology on their respective screens, the user taps the *ready* button and as soon as everyone did so, gets directed to the game-screen after a short countdown.

Game-Screen.

Once in the game-screen the participants play until the first hits the maximum number of points. All players see a dialog presenting the winner and get asked if they want to play a new game. If everyone selects this option, the new game starts after a short countdown. At any point in time any player may pause the game and all the the players see a pause dialog. As soon as everyone tapped the continue button, the game resumes. If a player decides to quit the game by closing the app, the game is ended and his opponents are shown an error dialog and directed back to the start screen.

2.2 Communication via Bluetooth

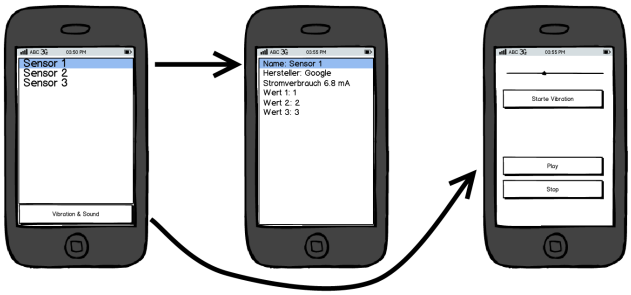

Figure 1: Application flow

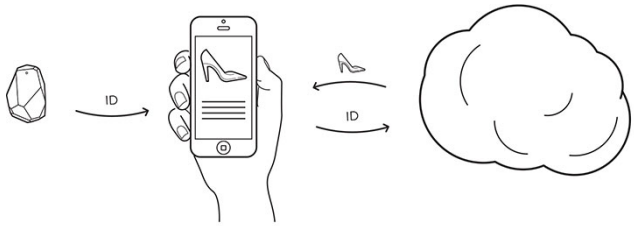

Figure 2: Schematic topology of players as shown on ready-screen

We discuss in this section how the communication between the players works in general fashion. Our communication layer is based on the Bluetooth technology; we use the provided android Bluetooth API. Furthermore, we encode all messages that are sent among the devices as JSON.

Messages.
[[elaborate]] We use JSON encoding for our messages. Each messages is composed of a header and a body. The header contains the senders id, the message type and ... . The body contains the payload which is specific to the message type.

Setup Phase.
This is the phase, during which the players add each other to their game. In the following we explain the behaviour for three players. The setup phase gets started as soon as the user got directed to the *ready-screen*. Each participant wants to fill its own representation of the player topology whith his counterplayers. The goal is to reach a state where all the participant have stored the same topology and are paired with each other. This way everyone knows where the others are and may choose to connect to either of them in order to send a message.

Topology Imagine a local topology as shown in Figure 3 [[Add figure]]. Each participant sees himself as number one. It is important to see, that a participant can reason out who, i.e. which position in the topology, sent a message, provided that the message contains the participants position as seen from the sender. This is easy to verify, because there is exactly one possible sender, which has the participant at this particular position.

A specific position in the local topology can attain three different states. The are *vacant*, *remembered* and *set*. A position is *set* only if an opponent has succesfully been assigned to it. *Remembed* denotes the state when a participant received a request and replied with an *acknowledgment*. In this case the player, who has sent the request, has successfully added the recipient to his topology. The recipient stores the other players id into his own topology and marks it as *remembered*. All positions that ar neither in the *set* nor in the *remembered* state, are in the *vacant state*. Note that the own position is always in the *set* state.

Consider following example: Player A's device sends a message to player B's with the information that B is at position 2. B knows now that A must be at position 3. Note that this is only possible because all players know the number of total players.

Sending a Request Each device starts with scanning its local area for other Bluetooth devices. The list of found devices gets displayed to the user as he taps to choose his opponents. Once the user seleceted one opponent from said list and the position is in the *vacant* state, the user sends a *request* message to the selected opponent. In the case where the position is in the *remembered* state, the user must first check if the selected opponent matches the stored player at this position. If this is the case, it is made sure that both parties store each other at the correct relative position; the request gets send. If however, the players do not match, the request must be aborted, because otherwise a wrong topology gets established. The message carries the opponents relative position to the user and the users player name.

Receiving a Request Upon receiving a request, the recipient reasons out at which position the sender must be. He then checks if the respective position in his local topology is still free. If it is still *vacant*, he sends an *acknowledgment* back to the sender and remembers the sender's id in the respective position. On the other hand, the position in in the topology might not be *vacant*. In this case the recipient replyies to the sender by an *error* message.

Ready As soon as a participant has its topology filled, he sends a *ready* message to all the other players. Once done this, and received the ready message from the other players, the participant knows that all players are ready. The player with the lowest id broadcasts a *start* message and the game starts after a short countdown. This concludes the setup phase.

During the game.
Each player is in the situation that he knows where all his opponents are locatet relative to him. Furthermore, the player's device is paired with all the other devices and thus ready to initiate a connection for sending a message to either of them.

Player A sends a message to player B if the puck leaves player A's screen and is about to enter player B's. The information sent along is such that player B knows when to display the puck where, in which angle, and with wich speed. Note, that only player B gets notified about the arriving puck.

There are broadcast messages being sent, if a player scores a goal or wins, a player pauses the game or someone leaves the game.

2.3 Game Physics and Graphics

We will use [[library ...]]

3. REQUIREMENTS

Describe system setup, components, external libraries, hardware etc.

4. WORK PACKAGES

Breakdown the work to subtasks to meet the project requirements. Define and describe these tasks.

- **WP1**: XYZ ...
- **WP2**: Set and Configuring Backend Serve ...
- **WP3**: Integration ...
- **WPx**: ...

Stick to a concise, scientific writing style.

5. MILESTONES

The milestones section provides a work plan for carrying out the project. This is your schedule for getting the project done. Clearly state how the work packages will be distributed among the team members.