AirHockey 3X

Distributed Systems – Project Proposal

Basile Maret, Philipp Rimle, 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 13-921-069, 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^{TM}.

1. INTRODUCTION

Air Hockey.

Let us start by a few words of explanation about the game itself. Air hockey is played on a 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 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 a rectangle to a triangle or to a square respectively. Refer to Figure 2 for a schematic setup of a game with three players. 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 with 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, that is speed or angle of incidence, to the right opponent. The challenge is to design a reliable protocol which guarantees real-time constraints and therefore makes a 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 username which gets shown to other players during the game. Further game specific settings may be set here.

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. One player clicks the start button and gets directed to the ready-screen. He gets shown a schematic view of the game topology as seen on Figure 2. The name of the player will be shown inside the screen and not any of the game action. The player can now choose his opponents from a list of Bluetooth devices and assign them to their position around the pitch. Every player, who has been chosen, receives an invitation in form of a confirmation dialog. If he accepts this invitation, he sees an "uneditable" ready-screen with the game setup. When the game setup is complete (all the players have been chosen and accepted the invitation), every player can mark himself as ready by toggle the *ready* button. As soon as all players are ready, the game starts after a short countdown.

Game-Screen.

Once in the game-screen, the participants play until the first reaches 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

We discuss in this section how the communication between the players works in a general fashion. Our communication layer is based on the Bluetooth technology; we use the provided android Bluetooth API.

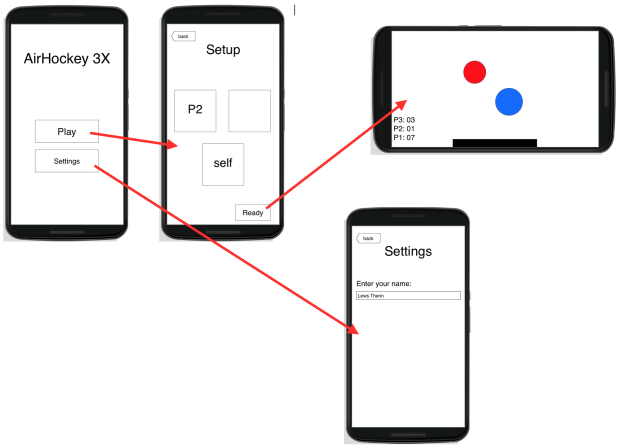

Figure 1: Application flow

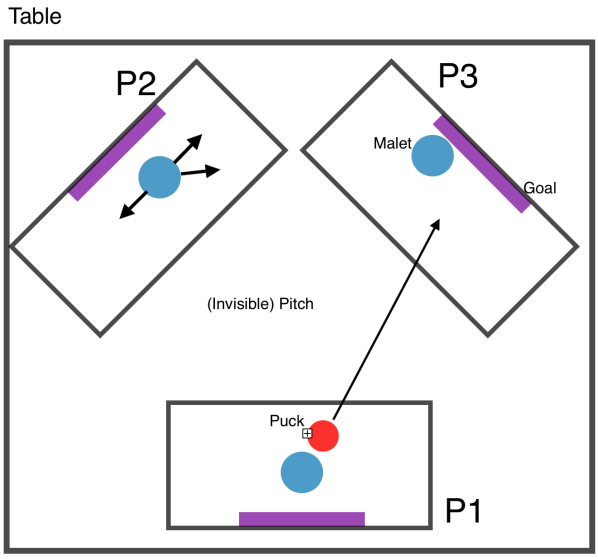

Figure 2: Schematic setup of the game. This topology gets also shown on the ready-screen, altough without any of the game action. The names of the players will be displayed inside the screen instead.

The communication layer serves as crucial level of abstraction. The objective of our design is to provide a realiable and real-time able underlying message passing system. The upper layers of our application, such as the game logic or the view controllers, can use the communication layer as simple interface to send messages to other participants, without having to know anything about Bluetooth.

Messages.

We use JSON to encode our messages. Each message is composed of a header and a body. The header contains the senders id and the message type. 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. All participants maintain a local copy of the schematic game topology. The goal is that at the end of the phase, each player knows where all the other players are and is paired with all of them. That way quick connections to the devices are possible.

Sender's Perspective One of the players, which need verbaly be agreed upon, enters the ready-screen by tapping on the *start* button. Let us call this player thereafter the 'sender'. His device starts scanning for other Bluetooth devices. (Naturally everyone needs to turn on Bluetooth on their device to be able to play). As he taps on one of his opponent's position on the schematic game topology displayed on his screen, a list with Bluetooth devices shows up. He can now choose one of his opponents; a Bluetooth connection gets established and an inivtation messege gets sent. The message carries the current state of the setup to the other player, this includes the position of said player and the the position of the sender. If the other player responds positively to the invitation, the sender assigns the other player to the respective position in the topology and may proceed adding players until all are set.

Receiver's Perspective All the other participants of the game need to wait for an invitation of the sender. As soon as they get an invitation from the sender, they get shown a invitation dialog and can either accept or deny. If they accept, the Bluetooth connection gets established to the sender and they receive the information about the current state of the setup phase, e.g. which position was assigned to them and at which position the sender is. The user gets directed to an "uneditable" ready-screen, where he sees the schematic topology of the game with his position and the sender's position already filled in.

Wrapping up We arrived at the point where the sender knows the position of all the other players. All the other players must have accepted the invitation and have sent back a message to the sender saying so. The sender now iterates through the list of all his counterplayers and sends them the complete topology by again establishing a Bluetooth connection to the respective devices and sending messages. Each player is now able to update his knwoledge about the topology and also to show this to the user on the screen. The recipients have now the possibility to press the *ready* button. Then a message gets sent back to the sender, saying that this particular player is ready. As the sender got all ready messages, the sender's user can himself hit the *ready* button and a start message gets broadcasted to all players. Once received a start message, each player gets shown a short timer after which the game starts. This concludes the setup phase.

During the game.

Each player is in the situation that he knows where all his opponents are located 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 which 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; in general: global events.

2.3 Game Physics and Graphics

We forward touch events to a refresh rate independent game loop where we compute dynamics and collision detection between our local physical objects, the puck, the mallet, and the rails. These game elements are rendered with OpenGL ES 2.0 as simple geometric shapes. The physics en-

gine does not simply keep a single state up to date. It must compute the future trajectory of the puck after each shot, in order to determine which player to notify about handling the puck next.

3. REQUIREMENTS

A short listing of our requirements.

- We will be targeting the minimum SDK version: API 18: Android 4.3 (Jelly Bean) to be sure that we can use the Bluetooth API without any problems also for BLE.
- The commuincation layer is based on Bluetooth low energy (BLE). To that end, we make use of the provided android Bluetooth API.
- We use the built in android JSON API for the messages.
- The graphical part of the game is based on the openGL ES 2.0 library.

4. WORK PACKAGES

It follows a breakdown of the work to be done. We group the different tasks into work packages. Each work package may either belong to the category Communication (1), Application (2), Game (3) or Miscellaneous (4).

- **WP01: Basic app architecture.** Set up all the activities with their buttons, images and further view objects. Make sure that *settings-screen* and *main-screen* work also in portrait format. (2)
- **WP02: Desgin *ready-screen*.** Provide schematics of the game topology. Each player position can be displayed as empty or occupied. Also make sure that the sender can tap on empty positions and gets shwon a list, which can be filled by Bluetooth devices. (2)
- **WP03: Design *settings-screen*.** Fine tune all the settings. Also handle storing and loading from shared preferences. (2)
- **WP04: Messages.** Model messages based on JSON. Provide methods to initialize each message type and methods to extract the appropriate information again. (1)
- **WP05: Basic Bluetooth.** Design interface to client classes, in particular methods to get a list of all the players, or to send messages to a specific player at a specific position. Implement scanning for devices, establishing connections and sending messages. (1)
- **WP06: Setup Phase.** Handle all the logic happening during the setup phase. This includes scanning for Bluetooth devices, pairing with them, maintain a list of all the player's devices. Also, several rounds of message passing need to be done. The game topology needs to be passed on to the game logic, once the setup phase is done. (1)
- **WP07: Implement coherent physics engine.** Design a framework to be used for the puck's position, speed and direction. Collisions on the rails and mallets must be handled. (3)
- **WP08: Game graphics.** Design the *game-screen*. This includes the pitch, displaying the scores of each player and further graphic elements. Also important are the images of the mallet and the puck. (3)
- **WP09: Game logic.** Actually make the game happen. Start the game, then use the physics engine and the communication layer to notify the appropriate players as needed. Display the puck, with its speed, on the right player's device. Also manage the game state, e.g. all the points, or when to end the game because someone won. (3)
- **WP10: Testing** Aside from continueous testing, which is done throughout the course of the project, this final testing phase is used to conquer all remaining problems. (4)
- **WP11: Preparation for the presentation.** Create slides and prepare demo session and rehearse. (4)

5. MILESTONES

We provide our work plan in form of the table seen in Figure 3 below. Note that the length of the period, during which the wroking package is to be worked on, does not indicate the amount of work that is to be done. The time assigned to each package depends on the dependecies among the tasks. For instance, WP01 surely needs to be done before WP03. WP07 on the other hand is mostly independent from the other packages and can thus be worked on during a longer period of time.

	14.11	15.11	16.11	17.11	18.11	19.11	20.11	21.11	22.11	23.11	24.11	25.11	26.11	27.11	28.11	29.11	30.11
BM												WP08	WP08	WP08	WP09	WP09	WP09
PR			WP05	WP05	WP05	WP05	WP05	WP06	WP06	WP06	WP06	WP06	WP06	WP06			
ED	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP07	WP09	WP09	WP09
OB			WP02	WP02	WP02	WP02	WP02	WP06	WP06	WP06	WP06	WP06	WP06	WP06			
RS	WP04	WP04	WP04	WP04				WP06	WP06	WP06	WP06	WP06	WP06	WP06			
VV	WP01	WP03	WP05	WP05	WP05	WP05	WP05										

	1.12	2.12	3.12	4.12	5.12	6.12	7.12	8.12	9.12	10.12	11.12	12.12	13.12	14.12	15.12	16.12	17.12
BM	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP10	WP10	WP10	WP11	WP11	WP11	
PR											WP10	WP10	WP10	WP11	WP11	WP11	
ED	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP09	WP10	WP10	WP10	WP10	WP10	WP11	
OB											WP10	WP10	WP10	WP10	WP10	WP11	
RS											WP10	WP10	WP10	WP11	WP11	WP11	
VV											WP10	WP10	WP10	WP10	WP10	WP11	

Figure 3: **Work Plan.**
Legend: BM=Basile Maret, PR=Philipp Rimle, ED=Etienne de Stoutz, OB=Oliver Butz, RS=Raphael Schnider, VV=Valentin Venzin