Hochschule Augsburg
University of Applied Sciences

LeapWalking

Gesture-Based Walking

Corinna List, Daniela Neupert

Winter semester 17/18

Interaction Engineering

Prof. Dr. Michael Kipp

Abstract

Is it possible to map the walking of fingers to the movement of a game character? LeapWalking describes a gesture-based interaction technique for controlling the movement of a virtual character. A LeapMotion tracks the user's hand and maps the index and the middle finger to the movement of the in-game character. We wanted to find out, if the gestures can be mapped properly and if this interaction technique feels natural and intuitive to the user. To test our approach, we developed a simple 3D jump'n'run-game, where the character, depending on the hand posture, runs, jumps, ducks or moves left or right to avoid obstacles. We found out that the gestures are valid, but the hand position is tiring after a few minutes.

1. Motivation

A lot of games are developed nowadays, but few are truly innovative. Therefore, it is our goal to try out another way of interaction to create a new gaming experience and to explore if this interaction method is valid. We want to find out if it is fatiguing to control a character via the LeapMotion and if it is intuitive. Especially for physically disabled people, who cannot use interaction devices like the Kinect properly, our approach could be a fun way to play games.

Our idea for an application was inspired by the paper "**VR-STEP: Walking-in-Place using Inertial Sensing for Hands Free Navigation in Mobile VR Environments**". The authors of this paper thought about a technique, which allows the user to move in a VR-environment by walking in place. With this method, motion sickness is reduced and it is intuitive and immersive to the user (Tregillus and Folmer, 2016).

Instead of using the whole body of the person as the interacting part for walking, we decided to use the hand. We do not use the inertial sensors of a smartphone, but a LeapMotion. Moreover, our application is not specialized on VR Games.

Furthermore, we were inspired by the paper "**Motion Editing with Contact Based Hand Performance**". In this research the authors described a system where they used two fingers to pantomime leg movement. The idea was to create full-body animations based on the data they received from a touch-sensitive tabletop when a user "walked with their fingers" (Lockwood and Singh, 2012).

There are several differences to our idea. We collect the data using a LeapMotion. Therefore, we do not only get the data of the finger tips, but from all the joints in the user's hand. This gives us the possibility to add multiple movements to our application instead of only using the walking

movement. Furthermore, instead of only using the data to generate animations, we use it to control a character in a game to find out how natural this interaction method feels to the user.

2. Concept

We wanted to build an application, that uses the LeapMotion to make walking in a game more intuitive. Instead of using the “WASD”-keys to walk, our intention was to utilize the similarity of index and middle finger to the legs.

To test our idea, we wanted to create a jump and run game where the user controls a character. The goal is to avoid obstacles and reach a certain distance. The camera is positioned slightly above and behind the character (see figure 1). Consequently, the player can not only see the character and obstacles next to him but the stage which is positioned in relation to the setup of the LeapMotion and a cardboard in the real world. The character seems to run from the bottom to the top.

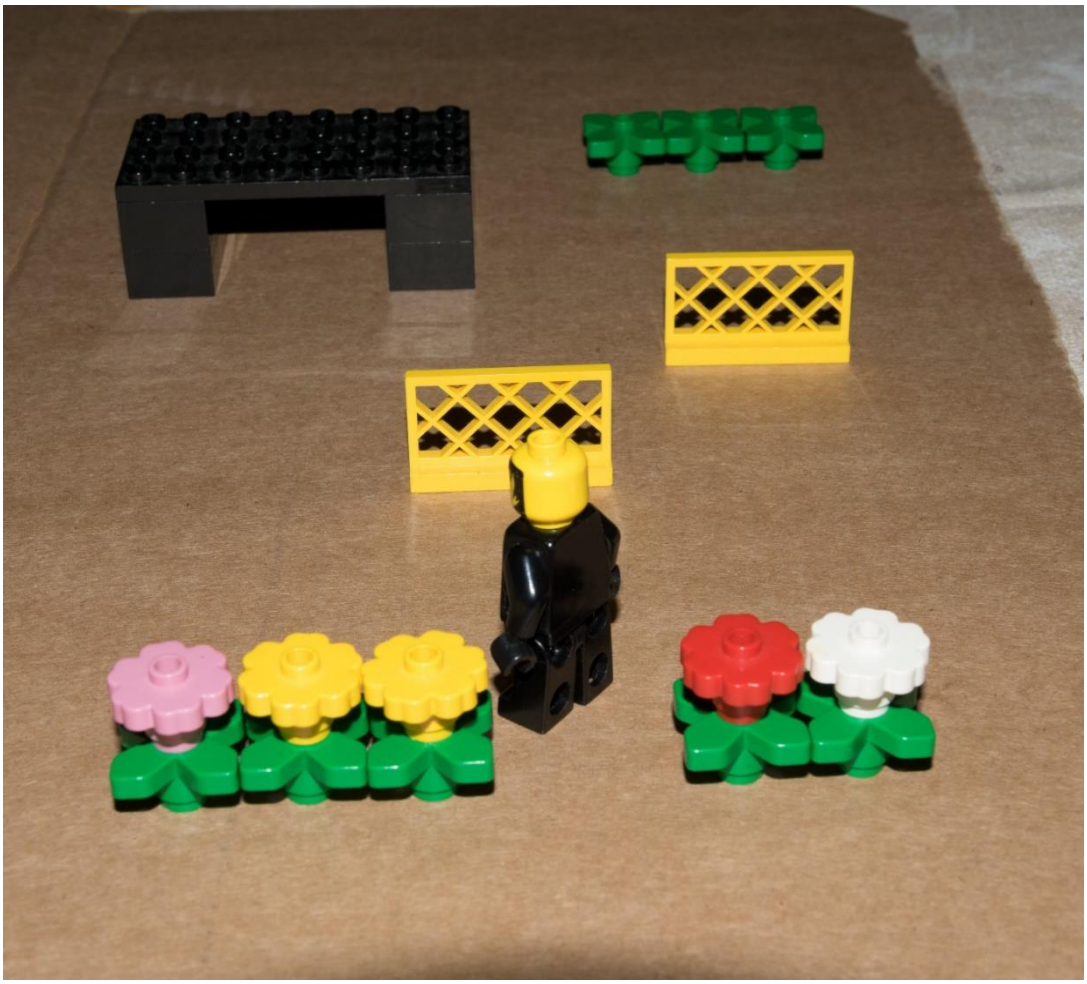


Figure 1: Game scenario

3. Setup

To give the user a sense of orientation, we created a frame (see figure 2). The LeapMotion is facing upwards and the user should place his hand above it. The fingertips of the index and middle finger should touch the vertical cardboard.

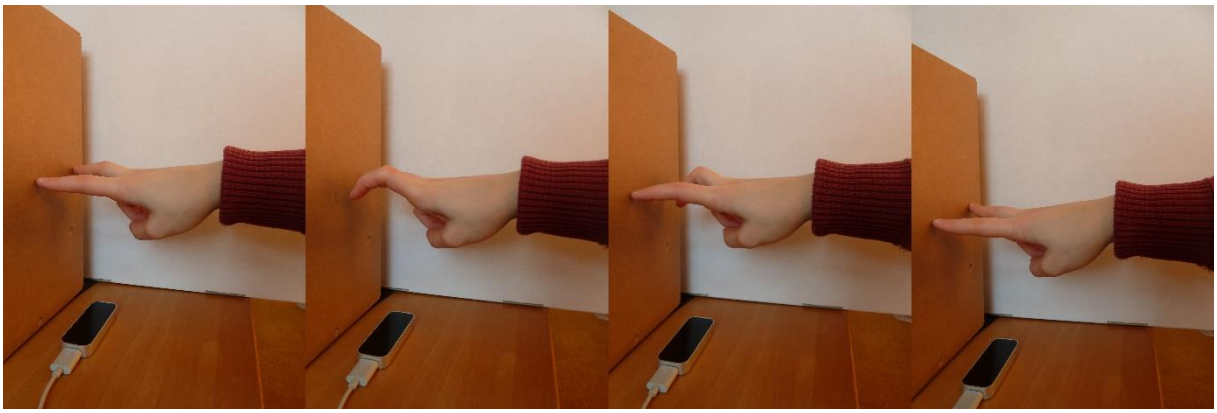
Figure 2: Frame for the LeapMotion

4. Gestures

To control the character in the game, the player can use four different gestures:

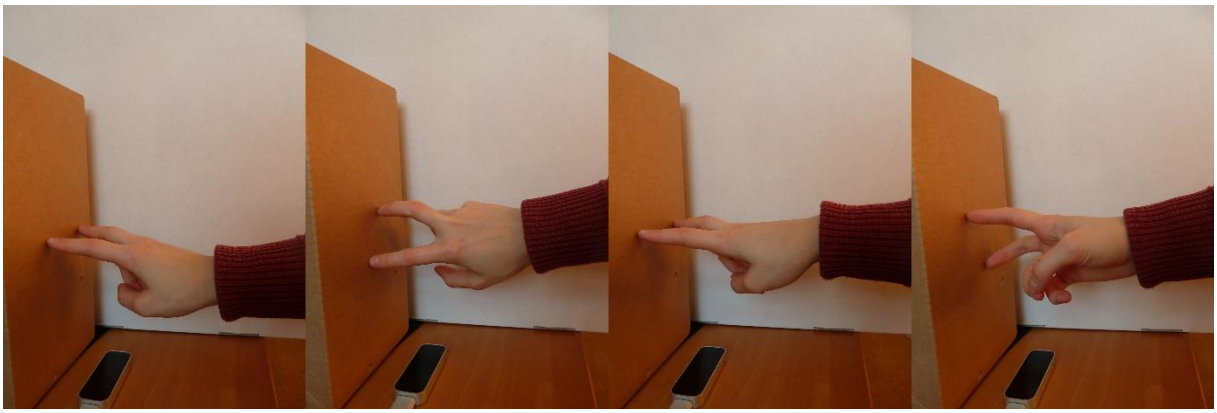
Walking

To walk, the user lifts up one of the “legs” (index or middle finger) vertically, while the other one moves down. To continue walking, index and middle finger now have to change their position.



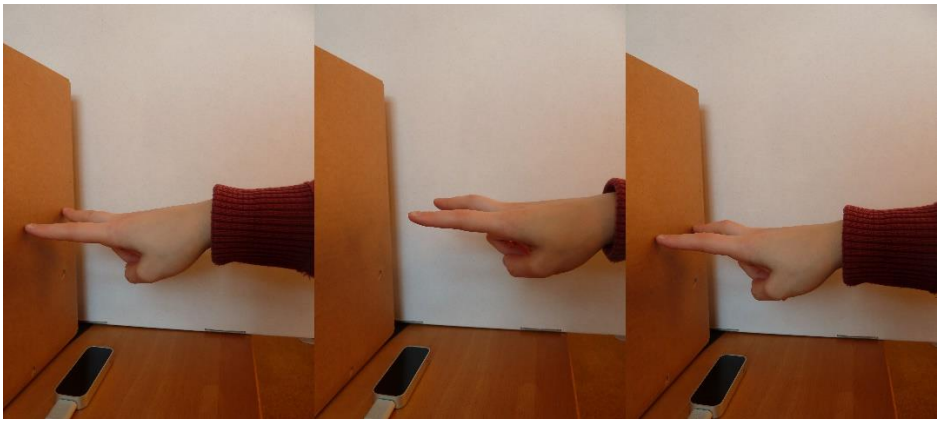
Curves

To pass an obstacle on the left or right side, the user rotates his hand to change the speed values in the x and z dimensions while doing the walking gesture. Therefore, the walking direction is changed.



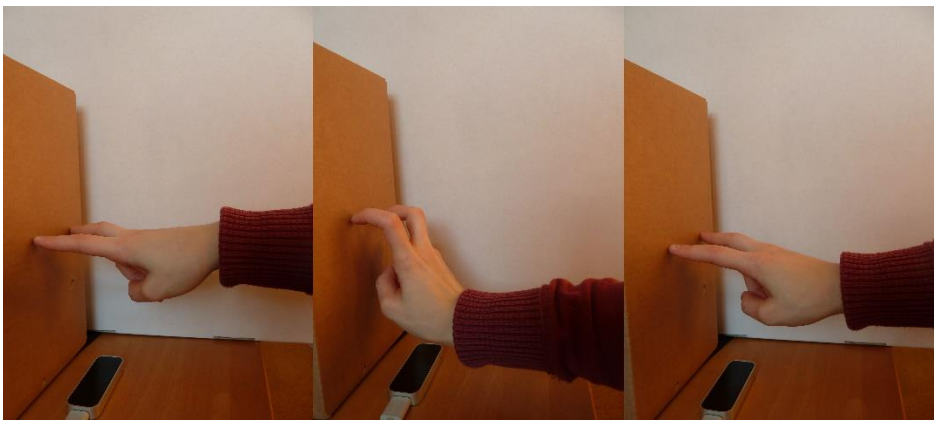
Jumping

For jumping the user lifts up both “legs”, so they do not touch the cardboard anymore. Then he has to return the fingertips to their initial position.



Ducking

To evade high obstacles, the character has to duck. To achieve this, the user moves his wrist towards the cardboard and then returns to his initial position.

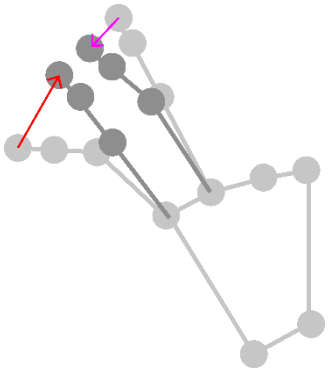


5. Implementation

To be able to use the LeapMotion in combination with Processing we used the library "LeapMotionForProcessing" by Darius Morawiec. For our application we needed to detect different points of the fingers and the arm, for example the index finger tip or the wrist.

Our gestures are detected as follows:

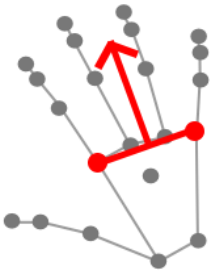
Walking



Since walking is a dynamic gesture we needed to detect the movements of the index and middle finger in relation to time. After each frame we look at the movement of the index and middle finger tips by calculating the vector between their old and their new position. Our walking gesture is as the opposite movement of those two fingers. We calculate the angle between the indexFingerMovement-vector and the middleFingerMovement-vector. If the angle is greater than 90° and smaller than 270° we define the vectors as being opposite. As long as this condition is fulfilled the character is walking.

In Addition, we calculate the character's speed by counting the direction changes of the indexFingerMovement per second.

Curves



To define the walking direction, we calculate the vector between the index finger and pinky finger knuckle. This orthogonal vector faces the direction in which the character is walking. We multiply this vector with the calculated speed from the walking gesture.

Jumping

To jump, the user has to lift the index and the middle finger from the cardboard. Both finger tips have to reach a certain distance to the frame to activate the jump. This leads to an automatic movement, so the user does not have to keep performing the walking gesture during the jump.

Ducking

For the ducking gesture, the user has to move his wrist towards the cardboard. To trigger the action, the wrist position has to be lower than a certain value. This, as well as the jumping gesture, starts an automatic movement.

Game and GUI

As explained in the concept, we wanted to create a simple jump'n'run-game where the user has to avoid obstacles and reach a certain distance. We programmed a 3D game built with simple geometrical objects. To help the user understand the game, a simple GUI shows the distance to the goal, the amount of times the character has hit an obstacle, the walking speed and the walking direction (see figure 3). Before the game starts a short counter gives the user enough time to position his hand and focus on the game. At the end, the interface shows the user's result with the time that it took him to reach the end as well as the number of collisions.

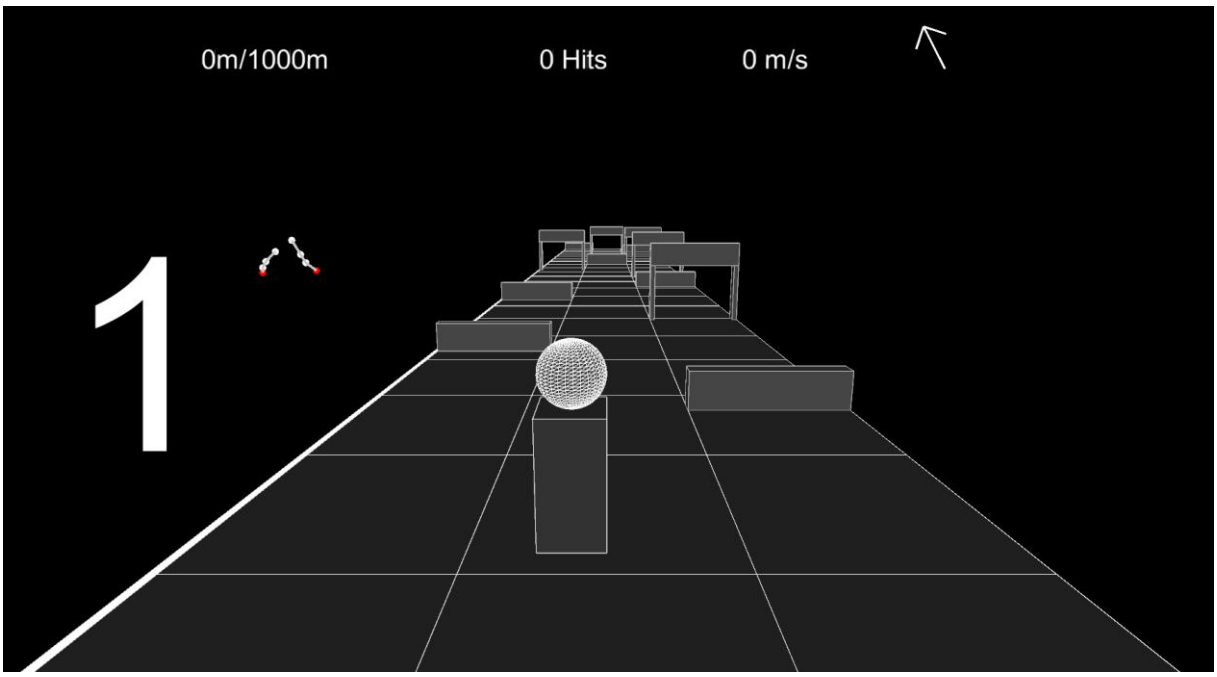


Figure 3: The game-GUI

6. Evaluation

For a user test, we planned to have three runs through the game per person. Hence, the user has time to get used to the gestures and the game environment. During these runs, we collect the following data: The time needed to reach the goal, the number of collisions with obstacles and how often the user jumps or ducks. We want to find out, whether the users used all gestures or tried to avoid some. For each run, the users have the task to reach the end of the level as fast as possible while minimizing the number of collisions.

After the users finish all three runs, they have to answer a few questions about the gestures and the game. We want to know for each gesture if it is intuitive and fun to use. Another important question is how fatiguing the gestures and the hand positioning are.

In our pretest with one user we found out, that three runs are not enough to get used to the game, so we will increase the number of runs to five. Furthermore, the one person thought that the ducking gesture was not intuitive. He would have preferred to bend the fingers instead of moving the wrist towards the cardboard, but we need more data to make a statistically valid statement about this gesture. Furthermore, the hand positioning was described as fatiguing and tiring for the hand.

In general, the game was described as a mini-game, which is fun to play but not for a longer period of time.

7. Conclusion

For the future, the game has to be tested by a larger group, consisting of at least six users, so we can gather more data about the gestures. In addition, the gaming experience could be enhanced by adding more visual feedback for the user, for example by changing the color of the obstacles during a collision to red. Moreover, another hand position might be less fatiguing, so we have to test different variations, maybe with a supporting frame, that the user can rest his hand on. Furthermore, we want to explore, if there are other possibilities to map the hand to the body, for instance for a soccer game. Thus, new gestures have to be explored.

8. References

Lockwood, N., and Singh, K. (2012). “Finger Walking: motion editing with contact-based hand performance,” *SCA '12 Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 43–52.

Tregillus, S., and Folmer, E. (2016). “VR-STEP,” the 2016 CHI Conference, Santa Clara, California, USA, May 07-12, 2016, pp. 1250–1255.