

**Leap Walking**

**Gesture-Based Walking**

Corinna List, Daniela Neupert

Winter semester 17/18

Interaction Engineering

Prof. Dr. Michael Kipp

**Abstract**

It is our goal to implement an application, where a character will be controlled by the hand of the user. Two of his fingers are mapped to the legs of the in-game character. We want to find out, if the fingers can be mapped properly to the body and if this interaction technique feels natural to the user. To test our approach, we want to develop a simple jump-and-run-game, where the character, depending on the hand posture, runs, jumps, avoids obstacles moving left or right or ducks.

(10-12 lines)

****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 the physically disabled people, who cannot use interaction devices like the Kinect properly, our idea 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. As so, motion sickness is reduced and this method 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. Also, 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, as so, we not only get the data of the finger tips, but from all the joints in the users hand. This gives us the opportunity to add multiple movements to our application instead of only using the walking movement. Also, 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.

Concept

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

To test our idea we want to create a jump and run game where the user has to control 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). As so, not only the player can see the character and obstacles next to him but the stage which is positioned in relation to the setup of the LeapMotion and the board in the real world. The character seems to run from the bottom to the top.



Figure 1: Game scenario

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.

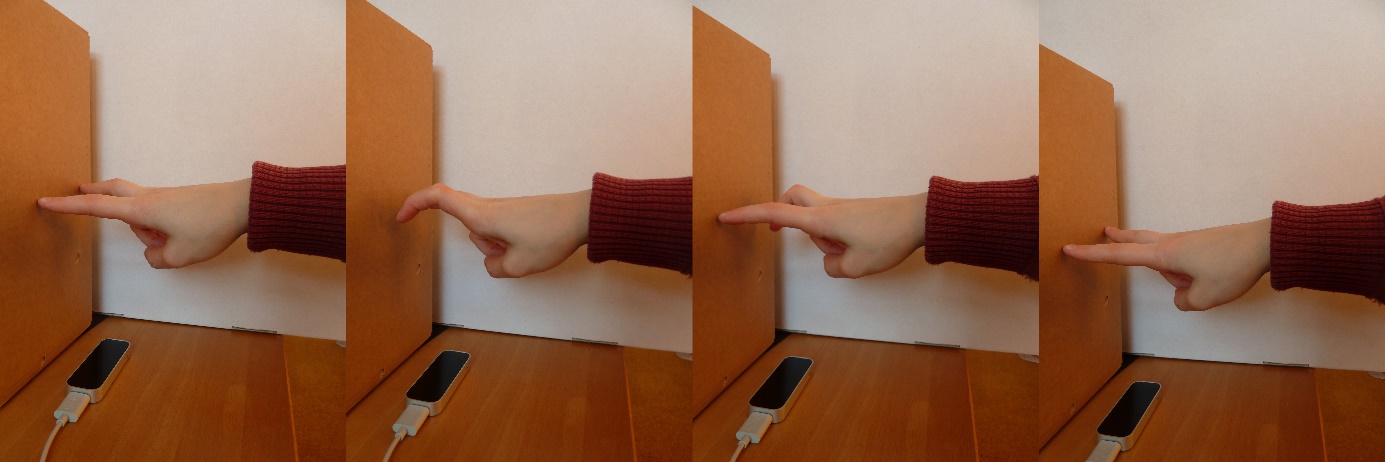
Figure 2: Frame fpr the LeapMotion

Gestures

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

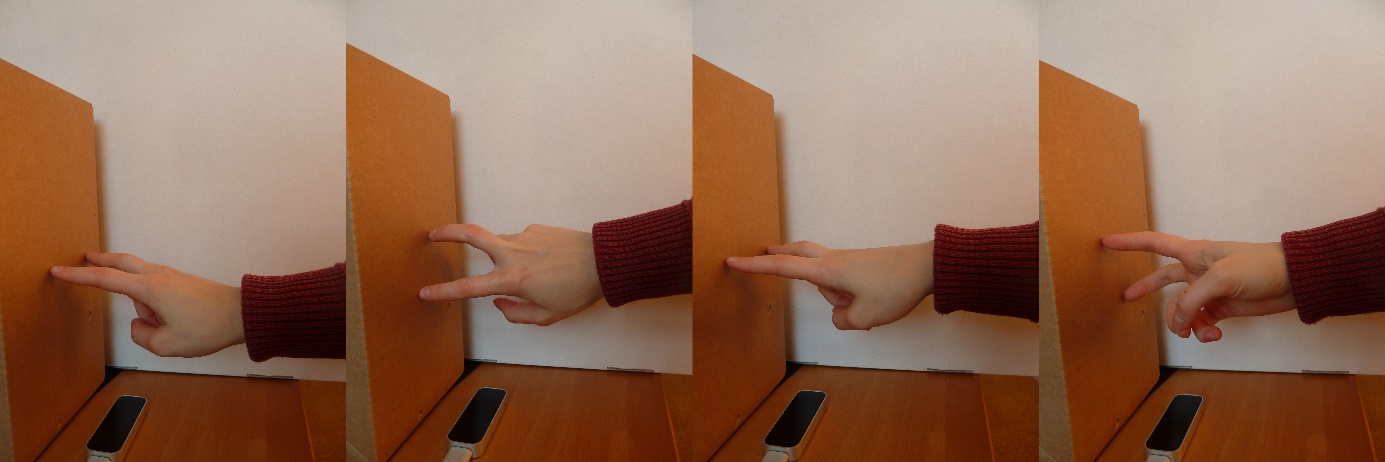
**Walking**

To walk, the user has to lift 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.



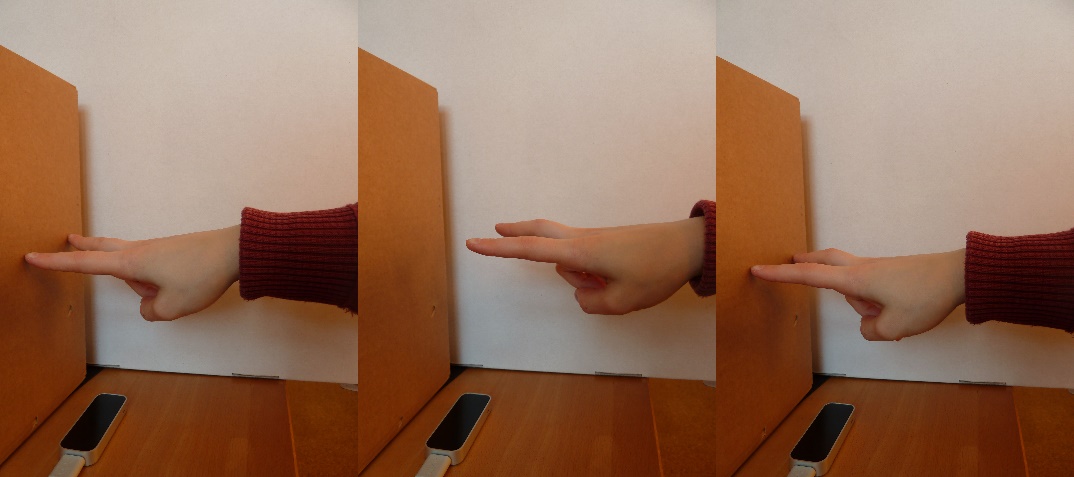
**Curves**

To pass an obstacle on the left or right side, the user has to rotate his hand to rotate the body of the character and as so change the walking direction, while doing the walking gesture.

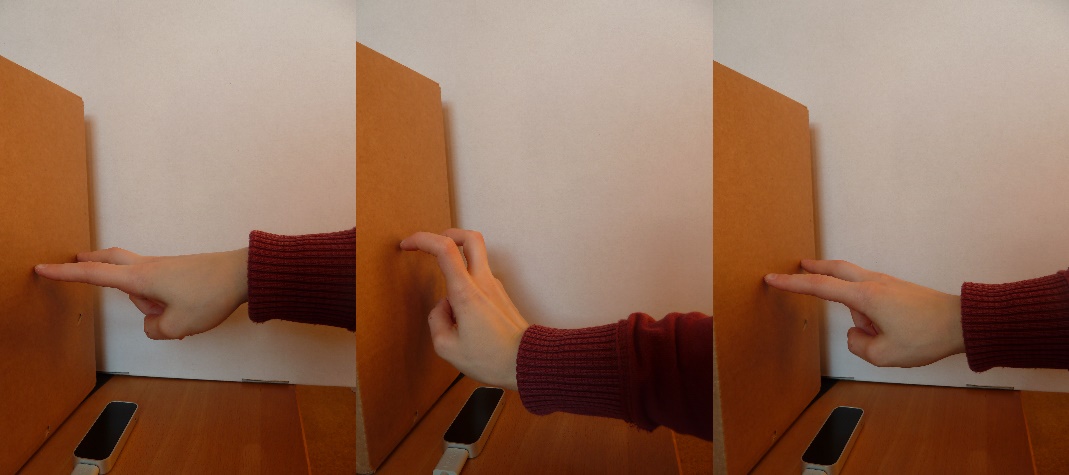


**Jumping**

For jumping the user has to lift up both “legs”, so they don’t touch the cardboard anymore. Then he has to return the fingertips their initial position.

****

**Ducking**

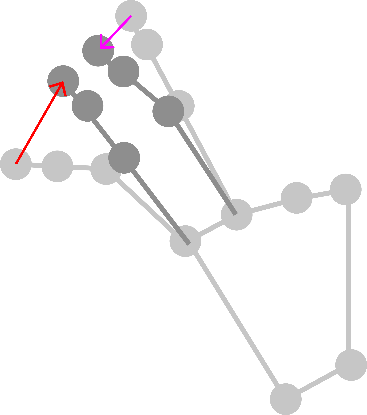
To evade high obstacles, the character has to duck. To achieve this, the user moves his wrist towards the cardboard and then returns to his initial position. 

Implementation

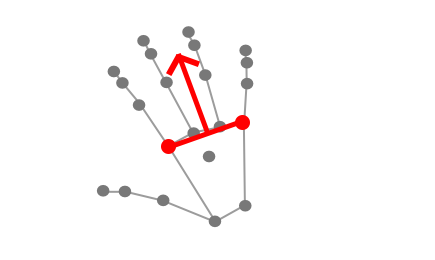
To be able to use the LeapMotion in combination with Processing we used the library “LeapMotionForProcessing” by Darius Morawiec. For our application we needed to detect different points of the fingers and the arm, for example the index finger tip or the wrist.

Our gestures are detected as following:

Walking

Since walking is a dynamic gesture we needed to detect the movements of the index and middle finger during time. After each frame we calculate movement of the index and middle finger tips by calculating the vector between their old and new position. Our walking gesture is defined to be an opposite movement of those two fingers. As so, we calculate the angle between the indexFingerMovement vector and the middleFingerMovement vector. Is the angle greater than 90° and smaller than 270° we define the vectors as opposite and as so the character starts walking.

Therefore, we have to calculate the characters speed. We count the direction changes of the indexFingerMovement per second to measure the speed.

Curves

To define the walking direction, we calculate the vector between the index finger and pinky finger knuckle. Its orthogonal vector faces the direction that the character walks to. We multiply this vector with the calculated speed from the walking gesture.

Jumping

To jump, the user has to lift both index and middle finger. Both finger tips have to reach a certain distance to the frame to activate the jumping. This leads into an automatic movement where the character jumps to a certain distance during a certain time. The user does not have to keep doing the walking gesture to move while jumping.

Ducking

For the ducking gesture the user has to move his wrist towards the cardboard. As so, the arms wrist position has to fall underneath a certain value. This, as well as the jumping gesture, starts an automatic movement, that ends after a certain time. Also, the user does not have to keep doing the walking gesture to move while ducking.

Game and GUI

As explained in the concept we wanted to create a simple jump and run game where the user has to avoid obstacles and reach a certain distance. We programmed a 3D game build with simple geometrical objects. To help the user understanding the game, a simple GUI shows the distance to the goal, the walking speed, the walking direction and the amount of times the character hit an obstacle. Before the game itself 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 his result with the time that it took him to reach the goal and the amount of hits. We store this data, as well as the number of jumps and ducks.

Evaluation

Pros:

* Intuitive
* Fun

Cons:

* Tiring for the hand
* The animation is not running smoothly yet

Conclusion

* More visual feedback for the user -> color while collision with obstacle
* Comparison of the smoothness of different walking-calculation-algorithms
* Are there other possibilities to use this interaction method? -> football/skateboard game

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.