Finger tracking using touchless input devices

# Initial Goal

Goal of this project is the evaluation of finger tracking capabilities of touchless input devices. Primarily the Leap Motion will be investigated. Research of additional devices will also be conducted (e.g. Kinect) and compared with the Leap Motion.

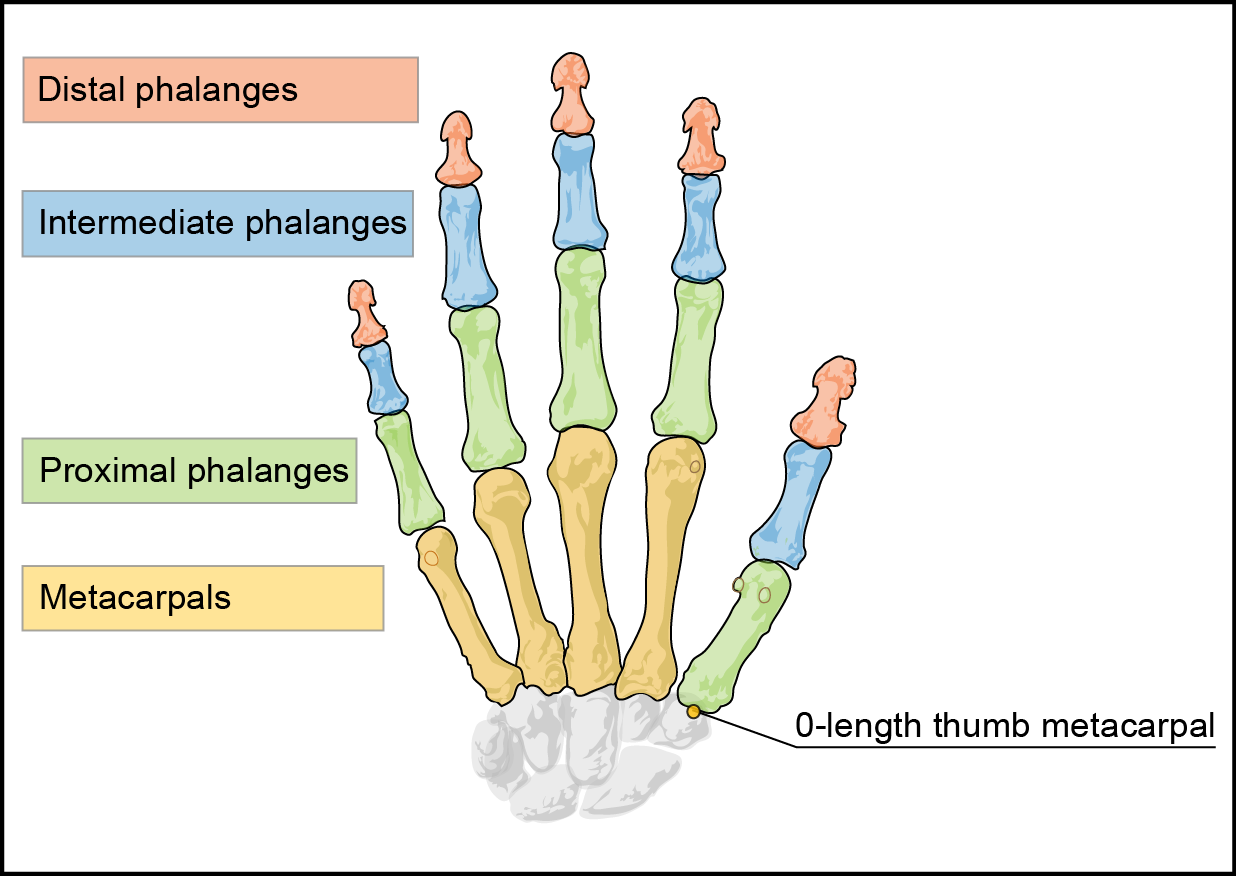
# Planned Milestones

1. First, a short survey about devices suitable for finger tacking is created.
2. Based on these devices, a small framework will be developed for generating the bounding volume hierarchy of the user's hand. This milestone includes some basic visualization for the bounding volumes.
3. Thirdly, the bounding volumes from the hand are loaded into a physics simulation environment. This includes a simple scenario where the user has to grab and move an object (bowls example).
4. After the simple scenario, more advanced scenarios with complex interactions and sophisticated physics constraints will be created (e.g. solving a Rubik’s Cube).
5. Based on the data gathered in these scenarios, a conclusion about the capability and suitability for finger tracking with the Leap Motion and other touchless input devices is drawn.

# Survey of devices capable of finger tracking

## Leap Motion

Skeletal tracking of fingers is officially supported by Leap Motion SDK in version 2.0. The Leap API provides information about each bone of each finger, most notably the knuckle positions and the orthonormal basis of the bone's center (a matrix that describes the bones transformation relative to the origin).



## Microsoft Kinect

Finger tracking is not supported by Microsoft's Kinect SDK, but access to depth image stream allows using third party libraries, for example (suggested by Prof. Kurschl):

* frantracerkinectft
* KinectLibrary
* CandescentNUI

## Conclusion

Due to the superior finger tracking capabilities of the Leap Motion, Kinect support has been dropped in favor of a more in-depth investigation of the new Leap Motion 2.0 SDK.

# Demo Application Infrastructure

The demo application provides the user with a platform to interact with a virtual world with a touchless user device. The user can choose from a set of different scenes which each contains of multiple *scene objects*. To interact with the scene objects, the user has to use a touchless input device (e.g. the Leap Motion) to track the motion of his hands.

## Architecture Overview

The application infrastructure is designed to offer extensibility for multiple devices. Therefore it is organized in multiple layer to abstract different kind of data. The Core layer contains device independent data structures for representing the status of a user’s hand.

The users hand is then converted into bounding volumes (Physics Hand), which are placed into the Physics World (i.e. the scene). The scene, including the physics-model of the hands, is simulated with the Bullet Physics Framework. Bullet itself calculates the resulting *motion state* of all objects, which describes the position and rotation of an object after a simulation step.

Later, all objects in the physics world, as well as the hand, are rendered depending on their motion state (Hand Renderer, World Renderer). For the visualization, the physics shapes are converted into triangles and then rendered via DirectX.

## Data Flow

The application’s data flow consists of 4 major stages. The first stage contains the raw device data from the touchless device (i.e. the Leap Motion). This raw data is processed and converted into device-independent data, which contains data structures for hands, fingers and bones. For the next stage, bounding volumes for each bone and for the palm are generated. The bounding volumes from the hand as well as other *scene objects* are added to the physics simulation. With Bullet, the physics world is simulated and the *motion state* of each object is calculated. The visualization stage draws the 3D scene based on the results of the physics simulation.

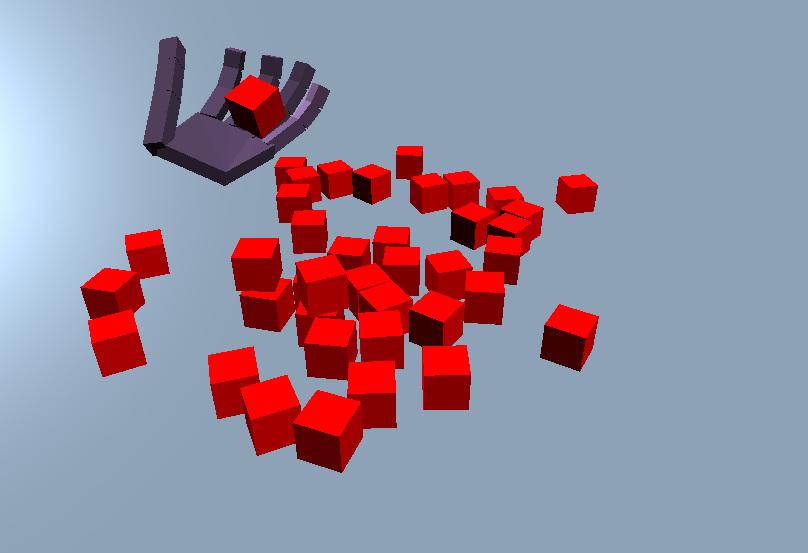
# Physical accurate hand reconstruction using Bullet

Bernhard, from Leap data to the rigid bodies

# Test scenarios

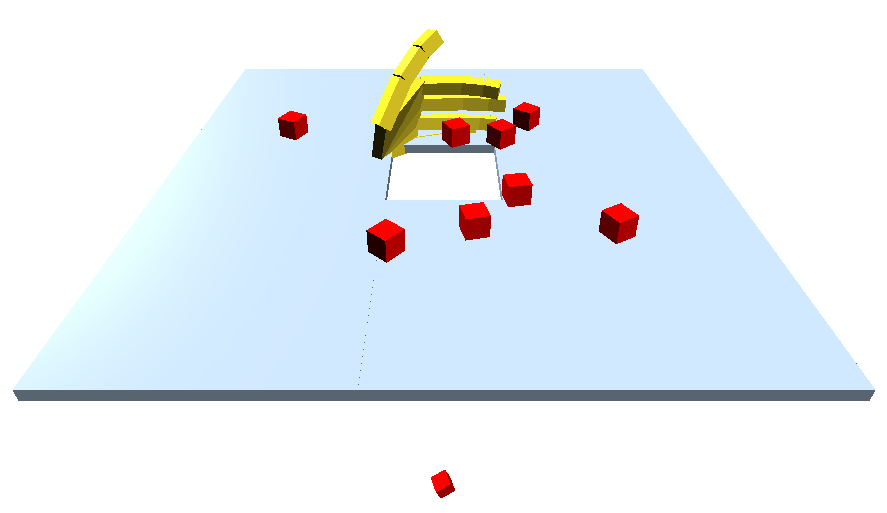
## Cubes

|  |  |
| --- | --- |
| Description | In the Cube scene the user can move, grab and throw small cubes. |
| Task | The user should familiarize with the representation of his hands in the virtual world. |
| Difficulties | Precision issues from the device. |



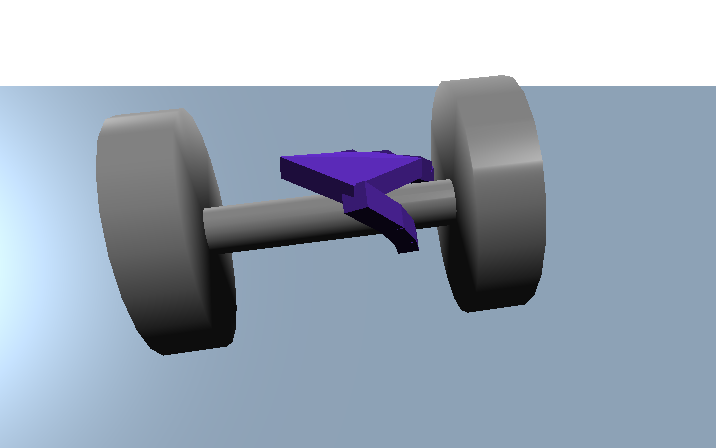
## Hole

|  |  |
| --- | --- |
| Description | Similar to the cube scene, but the plane has a hole. |
| Task | The user has to move all cubes into the hole. |
| Difficulties | Cubes which are too far away cannot be reached. |



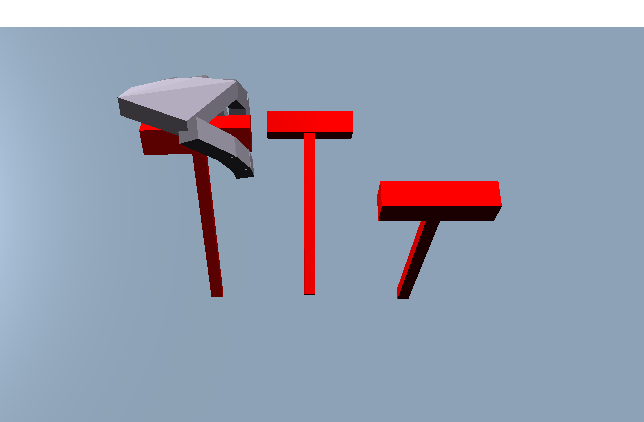
## Dumbbell

|  |  |
| --- | --- |
| Description | A single dumbbell is placed onto a plane. |
| Task | The user has to grab the dumbbell and move it upwards. Skilled users can move the dumbbell from one hand to the other hand. |
| Difficulties | Grabbing an object can be difficult because of the missing haptic feedback. If the hands are grabbed to tight, the dumbbell jumps through the hand. Another difficulty is the orientation and distance estimation in the three dimensional space. |



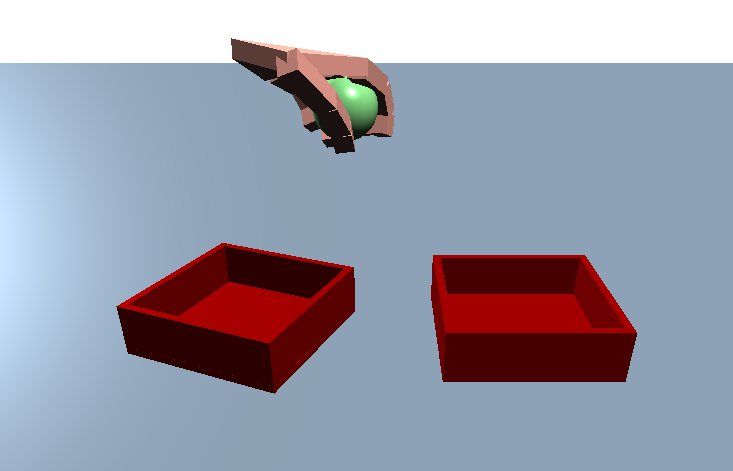
## Leavers

|  |  |
| --- | --- |
| Description | The scene contains three leavers with the end attached to the ground. In addition the leavers can only be moved along the z-axis. |
| Task | The user has to switch all leavers on and off again, i.e. move the position from. |
| Difficulties | Similar to the dumbbell scene, the orientation in the three dimensional space is difficult. |



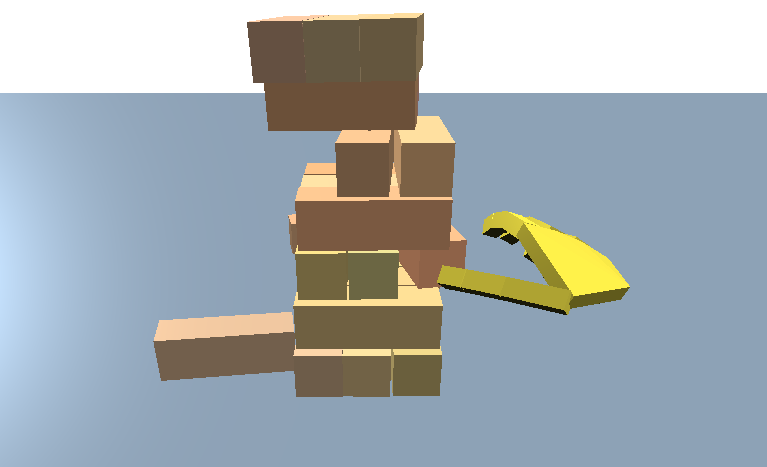
## Bowls

|  |  |
| --- | --- |
| Description | This scene contains of two boxes, and a ball inside of the left box. |
| Task | The user has to grab the ball and move it into the other box. |
| Difficulties | Correct grabbing of the ball is very difficult. When the hand is too much opened, the ball falls through, when the grab is too tight the ball jumps out of the hand as well. In addition, grabbing the ball without moving the boxes is difficult because the boxes move away easily. |



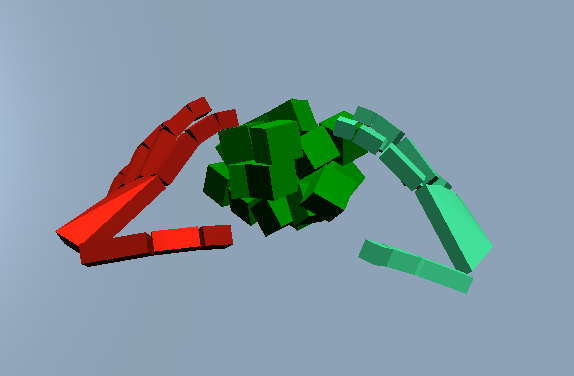
## Jenga

|  |  |
| --- | --- |
| Description | This scene multiple boxes which forms a Jenga-tower. |
| Task | The user has remove blocks out of the tower without crushing it. Can be played with multiple people. |
| Difficulties | Because of the size of the scene and the restricted area of the input device (in our case the Leap) not all positions can be reached. Also the hand can spawn inside the tower, destroying it completely. The orientation in the three dimensional state is difficult and therefore picking the right box is difficult. Furthermore to point with one finger to slide out a box is very difficult. |



## Rubic’s Cube

|  |  |
| --- | --- |
| Description | This scene contains a Rubic’s Cube |
| Task | The user has to solve the Rubic’s Cube by rotating the right layers of the cube. |
| Difficulties | Unfortunately the physics constraints to model a Rubic’s Cube are very complex. We did not manage to set them up correctly, therefore the Rubic’s Cube remains a chaotic bundle of cubes. |



# Problems and conclusion

Bla bla bla,

We thought this shall be easy

Bullet is extremely badly documented.

Even simple scenes turned out to be extremely challenging,

Problems with friction and movement of kinematic objects (the hand)

LeapSDK+