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.

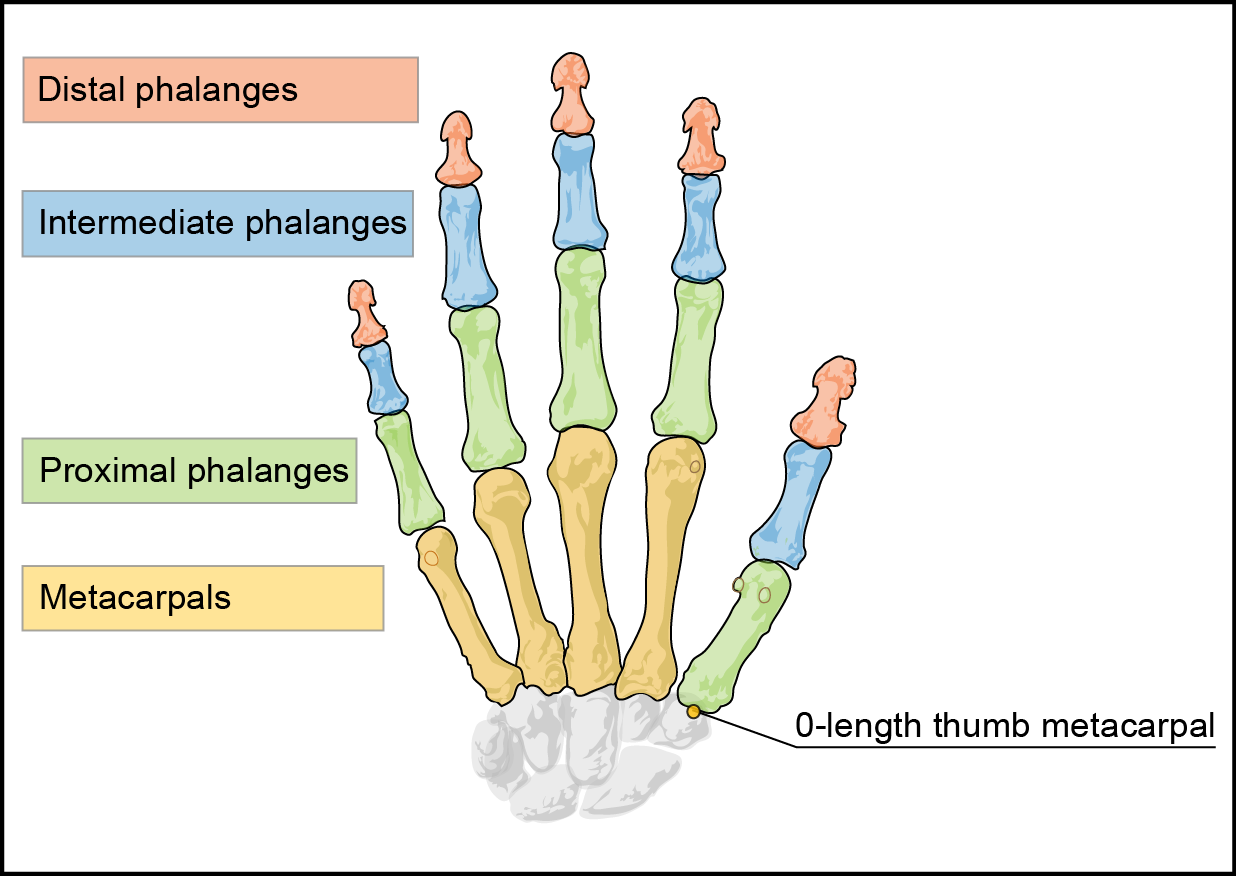
# Planned Milestones

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

Philipp, architecture and data flow from leap through bullet to visualization

# Physical accurate hand reconstruction using Bullet

Reconstructing the user's hand for interaction inside a physics simulation environment involves two steps:

1. Creating an approximated representation of the hand using bounding volumes provided by the physics engine. This stage will be called calibration in the following section.
2. Updating the transformation of the bounding volumes using live data from the Leap Motion controller. These updates are then processed by the physics engine to detect object collisions and forces.

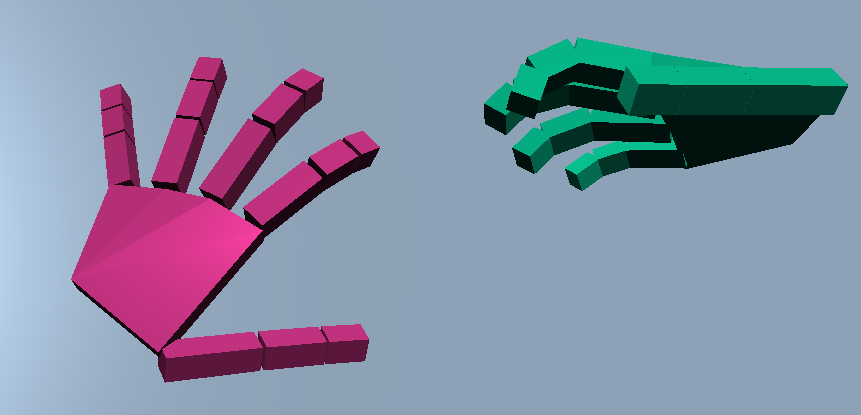
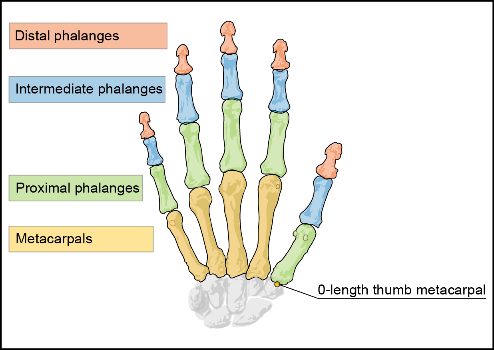
## Calibration

The calibration step is done each time a new hand enters the Leap Motion controller's field of view. The Leap SDK provides ids for each recognized hand which are used to detect new and gone hands at each frame. When a hand object is received from the Leap with a new id (an id not present in previous frames), this hand object is used to create a new hand representation inside Bullet.

Hands are represented using two kinds of bounding volumes provided by bullet. The bones of the fingers are approximated by boxes and the hand palm consists of a manually built triangle mesh.

For each finger the three bones not inside the palm (distal, intermediate and proximal) are used. The bone length is determined by the distance between the two adjacent knuckles of the bone (provided by the Leap via Leap.Bone.PreviousJoint and Leap.Bone.NextJoint). The finger's thickness is taken from Leap.Bone.Width and scaled down a bit. The position and orientation of the bone in world space is given by the orthonormal basis of the bone (Leap.Bone.Basis). This matrix describes the transformation of the bone local coordinate system (origin in the center of the bone) from the Leap Motion controller's coordinate system (origin at the top of the controller) and is set as initial transformation (motion state) when feeding the generated bounding volumes as rigid bodies into Bullet.

The hand's palm is generated using the positions of the metacarpal bones of each finger. These positions built up a polygon which is manually extruded into a triangle mesh. This mesh is also added to Bullet as a rigid body using the hands orthonormal basis in Leap.Hand.Basis as initial motion state.



As the calibration is done on the first frame of a new hand, the size and shape of the hand is fixed for the duration where the hand is visible. Therefore, the user should try to enter the Leap Motion controller's field of view with a flat hand, stretched fingers and preferably from the top.

## Updating

On each Leap frame, all calibrated hands are searched in the recognized hands by their ids. If a hand is found which was already calibrated in a previous frame, all motion states of all rigid bodies of the calibrated hand inside Bullet are updated using the values of the current frame. This concerns each bones and the hands orthonormal basis.

When a hand leaves the Leap Motion controller's field of view, its id is not found anymore in subsequent frames and all physics related data is removed from the physics simulation (unregistering and deletion of all rigid bodies).

# Test scenarios

Screenshot of each scenario with a short description about what shall be especially tested with this scene.

# 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)