Finger tracking and physically accurate 3D interaction

Table of Contents

Initial Goal 1
Planned Milestones 1
Survey of devices capable of finger tracking 2
Demo Application Infrastructure 3
Physical accurate hand reconstruction using Bullet 4
Test scenarios 6
Problems 10
Conclusion 10

Initial Goal

Goal of this project is the evaluation of finger tracking capabilities of touchless input devices and the suitability of the tracked fingers for 3 dimensional interaction inside a physics simulation. Primarily the Leap Motion will be investigated. Research of additional devices will also be conducted (e.g. Kinect) and compared with the Leap Motion.

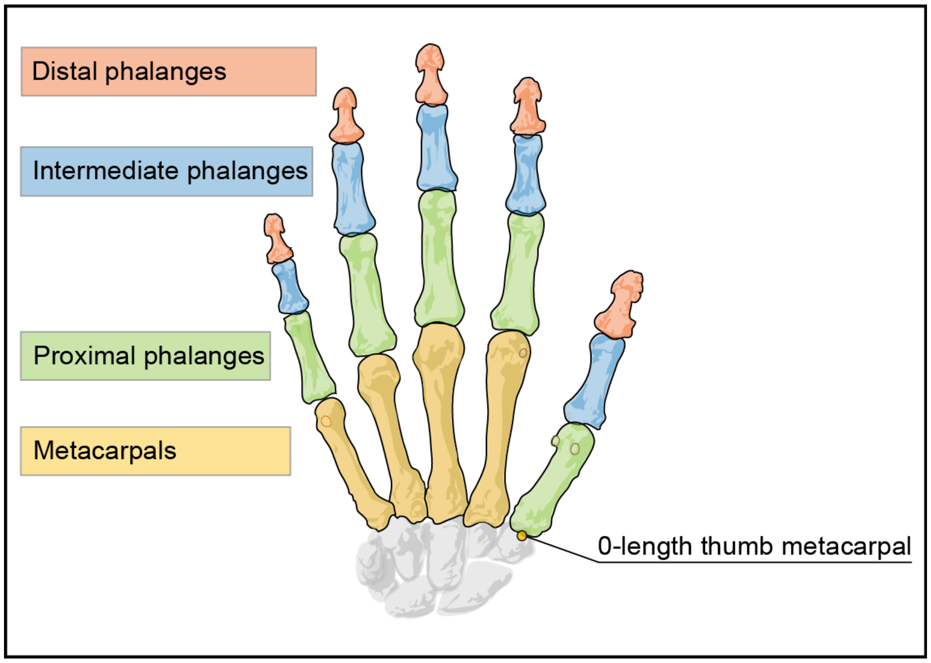
Planned Milestones

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 bones' centers (a matrix that describes the bone's transformation relative to the origin).
Initially we intended to use SDK 1.0 and calculate the bone positions ourselves. Luckily the SDK 2.0 offers this functionality out of the box. Furthermore SDK 2.0 simplifies the finger management as the API always guaranties five fingers per hand even if the Leap Motion has to interpolate fingers it cannot currently see.
The following image from the Leap SDK 2.0 documentation shows the available bones that can be queried from the Leap.



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

Data Glove

Considered but not recommended by Sebastian Pimminger.

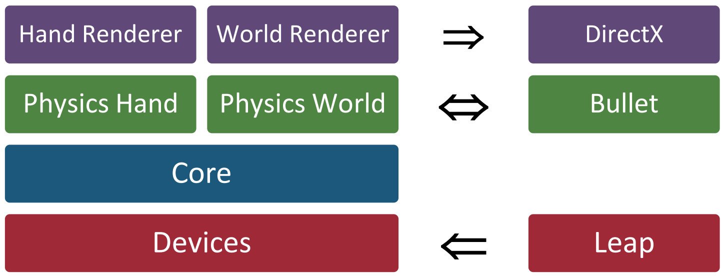
Conclusion

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, each containing 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



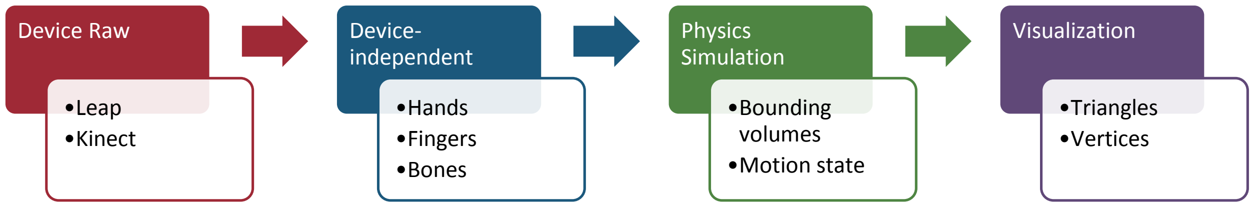
The application infrastructure is designed to offer extensibility for multiple devices. Therefore it is organized in multiple layers to abstract different kind of data. The Core layer contains device independent data structures for representing the status of a user's hand.

The user's 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

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.

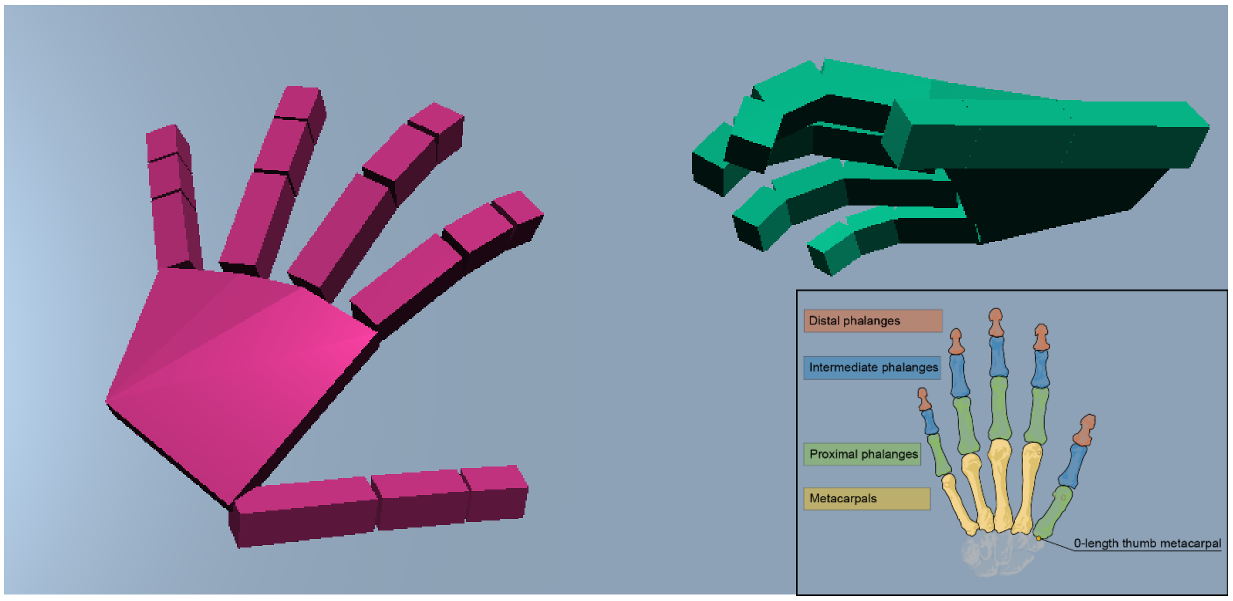
Calibration

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

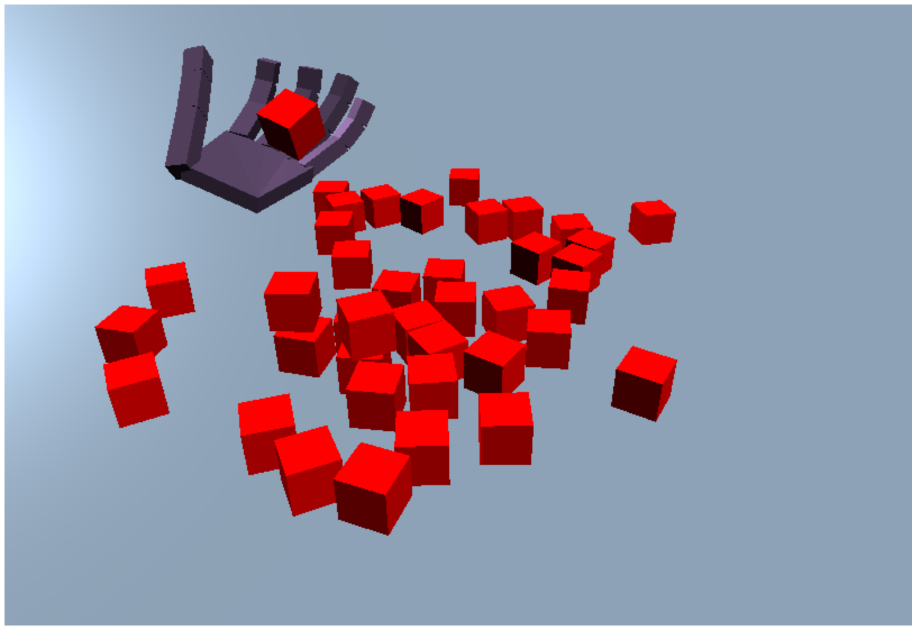
On each Leap frame, all calibrated hands are searched in the recognized hands by their ids. If a hand is found which was already calibrated in a previous frame, all motion states of all rigid bodies of the calibrated hand inside Bullet are updated using the values of the current frame. This concerns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

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

Test scenarios

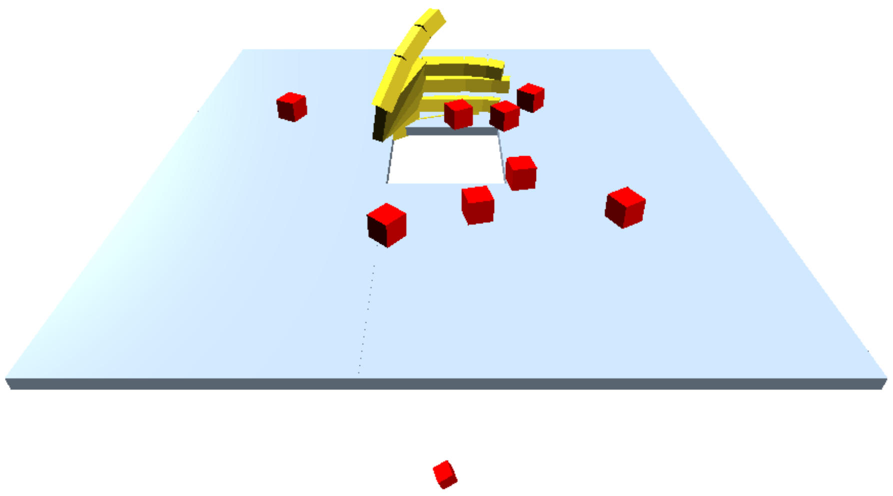
Cubes

Description	In the Cubes 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.



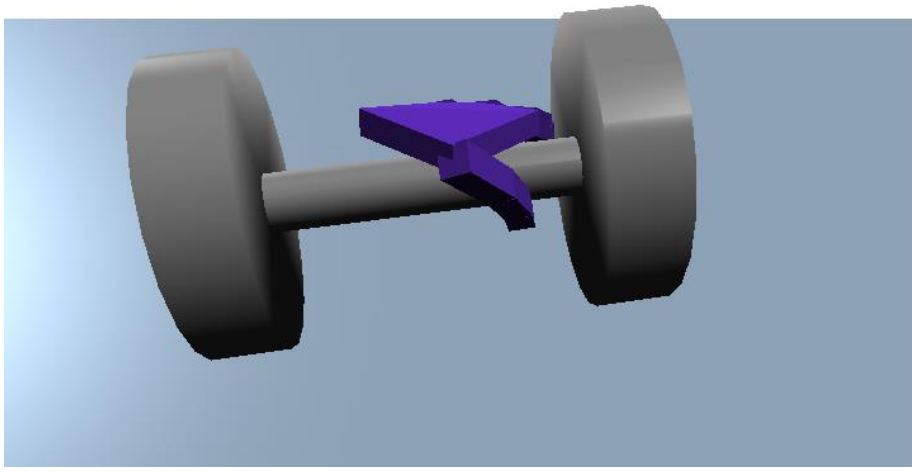
Hole

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.



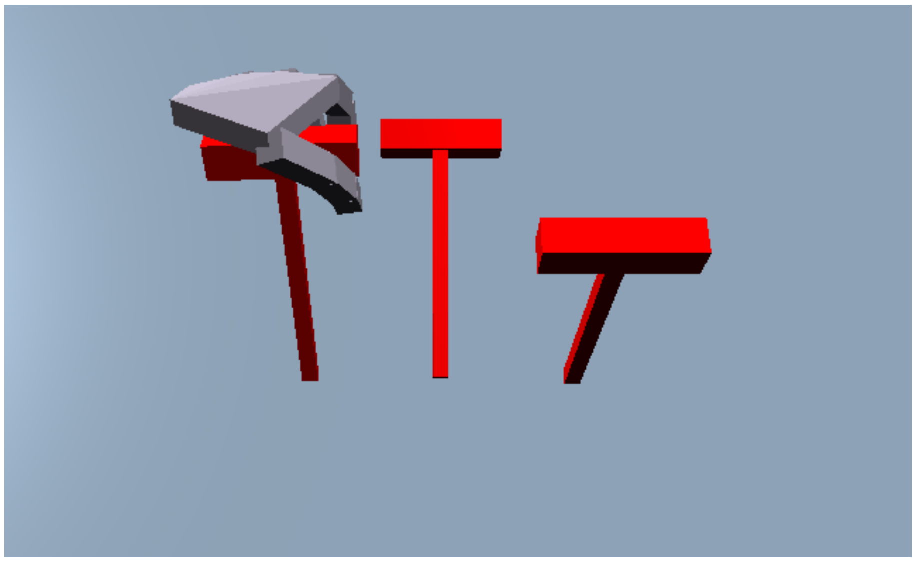
Dumbbell

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.



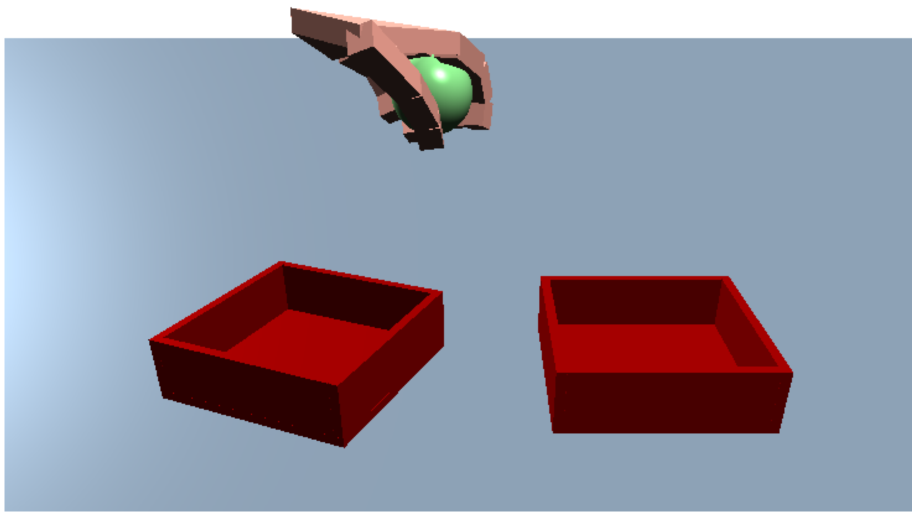
Leavers

Description	The scene contains three levers 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.
Difficulties	Similar to the dumbbell scene, the orientation in the three dimensional space is difficult.



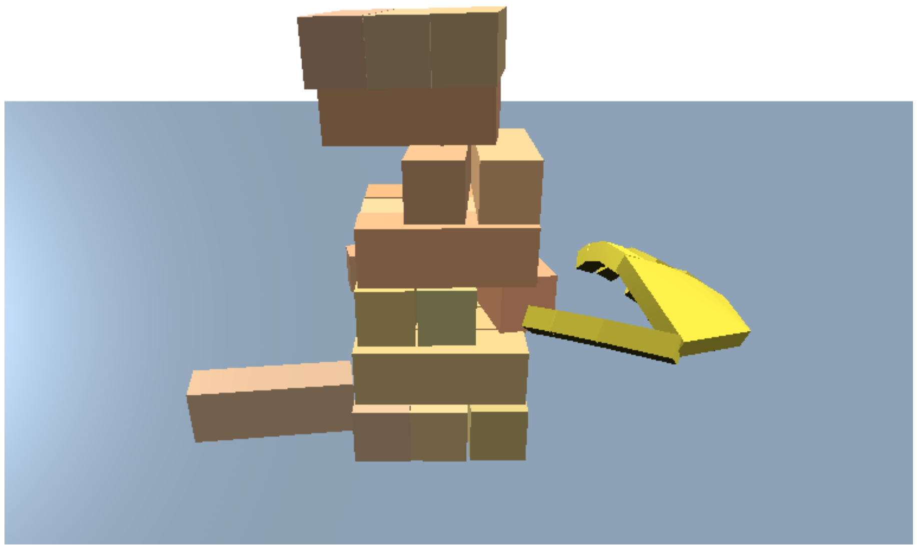
Bowls

Description	This scene consists of two boxes and a ball inside 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.



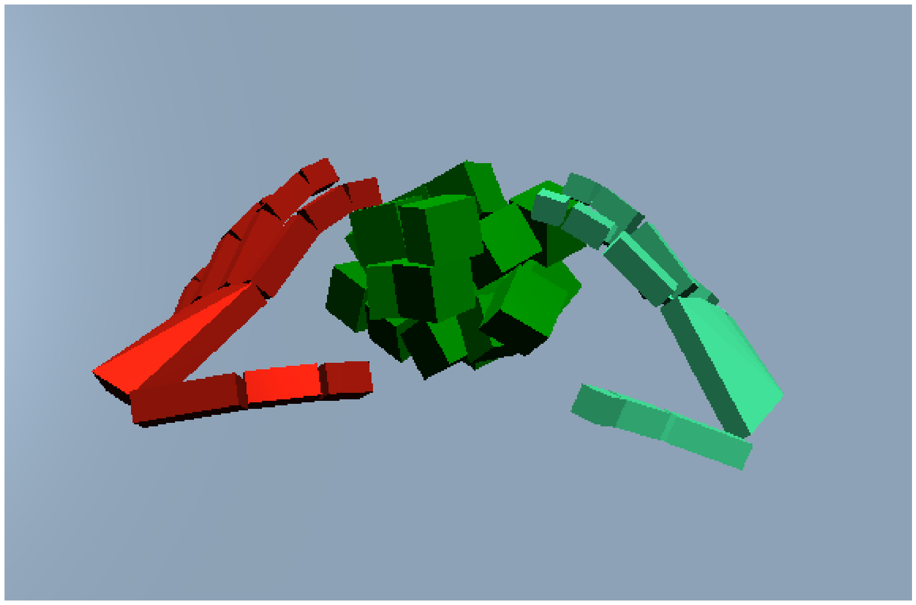
Jenga

Description	This scene contains multiple boxes which form a Jenga-tower.
Task	The user has to remove blocks from the tower without knocking it over. 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 were too complex. We did not manage to set them up correctly, therefore the Rubic’s Cube remains a chaotic bundle of cubes.



Problems

Concerning interaction

- The orientation in three dimensional space turned out to be harder than expected. Specially the estimation of depth on a two dimensional display is difficult (e.g. touching boxes in Jenga). Maybe the integration of shadows in the visualization could improve depth perception.
- As the user is not limited in the motion of their real hand the reconstructed hand might be placed invalidly (e.g. placed into objects). As a result the physics simulation might behave unrealistically (e.g. object jumping out of hand). This is especially a problem when holding objects inside the hand or between fingers.
  This problem is probably a result of the chosen implementation, as collisions are not reflected back to the user's virtual hand. The desired behavior of reactive collisions of the hand seems to be supported by Bullet but is more difficult to implement.

Generally speaking, even simple scenes turned out to be unexpectedly challenging.

Concerning implementation

- Bullet is a sophisticated and comprehensive framework but unfortunately lacks badly needed documentation.
- Implementation of the visualization with SharpDX (DirectX wrapper) turned out to be more time consuming than expected. Furthermore the integration into WPF was tedious and using WinForms would probably have been easier.

Conclusion

The Leap Motion Device offers comprehensive finger tracking capabilities. Especially with SDK 2.0 the input precision and quality is high and stable. The API of the Leap Motion is easy to use compared to the Kinect API. Furthermore it offers a wide range of data values (e.g. individual bone data).

The tested interaction scenarios (e.g. grabbing objects, moving levers) showed partial success. We believe the tested interaction scenarios might be applicable in games but are likely not suitable for traditional software.

Despite the time-consuming implementation (roughly 50h per person) the researched topics were interesting and we learned a lot. We believe there is still a huge potential for improving usability and applicability.