GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

SWZ Clausthal – Göttingen Simulation Science Center

DATA FUSION

Practical Course: Data Fusion
Operation of a coffee machine in a virtual environment via Kinect v2

Hendrik Bitzmann, Hanna Holderied, Christoph Rauterberg, Maximilian Weß

Abstract

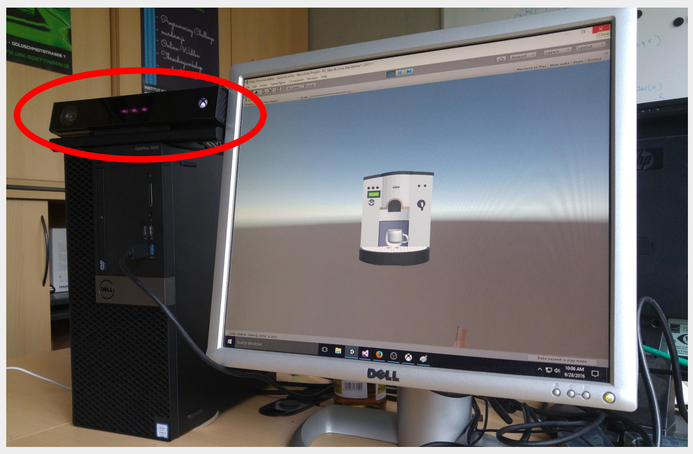
The aim of this project was to control a 3D model of a coffee machine via hand tracking with different technologies. This work is a prerequisite for the complete setup, where a user will wear a virtual reality headset. This in unison with the motion tracking can provide a deep immersive experience.

Introduction

The first technology we tried to enable hand tracking was the Microsoft Kinect v2 which was released in 2014 and comes with several features: an infrared depth sensor, a RGB sensor and 3D motion tracking. The last is especially important for this project.
In general the application is very easy: You place yourself in front of a monitor where you can see an 3D object and a 3D model of two hands. By moving your hands you can interact with the objects displayed on the screen. Therefore the motions are tracked by the Kinect sensors.
The second device used for a short period of time was a Leap Motion, also with a build-in infrared camera. This is a small device you place in front of your keyboard and via tracking of your hand gestures it allows you to control your PC or like in our case a coffee machine.

Methods: Kinect Setup

For our project we used a Windows 10 desktop. The first step to run the Kinect v2 was to download the official Kinect 2.0 Software Development Kit (SDK) from Microsoft which includes many useful tools and examples. The next step was to start the Kinect Configuration Verifier from the SDK browser, which will check the system requirements.



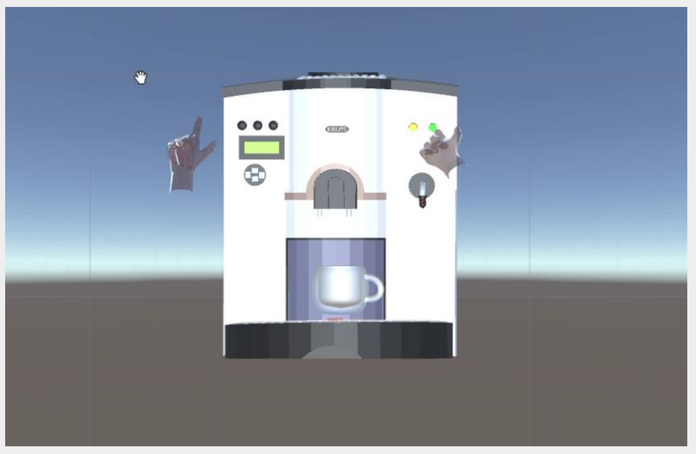
The setup with the Kinect (red) and the game view in Unity.

Methods: Development Software

We chose a running start by following a simple tutorial. This gave us a working development environment consisting of Unity3D, Visual Studio 2015 and the Kinect SDK, and we were already able to track our hand position and display them in Unity3D. Unity3D is a game development engine. It was a good choice for our application because of the ease which simple tasks can be executed. The import of assets into the project can be achieved via drag and drop, linking scripts for event handling and update mechanisms to this assets is handled the same way.

Methods: Tracking Procedure

To display everything as close to the reality as possible we looked for 3D hand models online and imported them into unity. We also got a model for the coffee machine from Patrick Harms.
For the correct visual feedback we had to track the hand movements. We used the skeleton information from the Kinect. These contain 3D coordinates from the palm, thumb, wrist and the tip of the hand, which should be the longest outstretched finger. To form a local coordinate system for the hand we used the palm and tip coordinates to form the forward vector. We used the palm and thumb coordinates for form a temporary side vector. The cross product of these two gave us the up vector. To obtain a orthogonal system we recalculated the side vector from the cross product of up and forward vector. For the positional tracking we just used the positional information of the palm.
To operate the coffee machine we had to detect when the hand is touching a button of the coffee machine. We pursued 2 approaches: The first was to track the euclidean distances from then hand to the buttons manually, so we could color the buttons according to the distance between the hand and the nearest button.
For the actual push of button we used the built-in collision detection in Unity. This is a mechanism which recognizes when an object marked as MeshCollider (the hands) hits another one marked as isTrigger (the buttons) and in a script you can then define what happens after this event.



The view in Unity. The coffee machine and both hands are visible. The right hand is pushing the button in the top right.

Results

We successfully established a development environment for the use of a Kinect v2 sensor and programmed a demo application where we are able to control a coffee machine with our tracked hands in a virtual environment.
Unfortunately we were not able to create a very robust platform due to several reasons: The buttons of the coffee machine were relatively small and close to each other. Additionally our motion tracking was a bit sluggish, therefore it was difficult to touch only one button.

Discussion

Apart from the problems mentioned above we were astonished how quickly we were able to use our setup, move hand models via the Kinect in Unity3D and implement collision detection. Even though some things were a bit tricky, the software works quite well and can - for easy purposes - be almost used as plug-and-play. Unfortunately this can't be said for the software available for Linux PCs, Windows is in the lead here.
There are several possibilities to improve the performance of our project. On the software side we could dig deeper into correction mechanisms to reduce the noise, also provide more hand gestures or a complete finger tracking. On the hardware side it would be nice to have some haptic feedback or fuse multiple sensors to improve the tracking and stability.
On top of that you could stream the image of the screen to a smartphone and use e.g. a Google Cardboard to create the full virtual reality effect. This could improve the usage of objects like our coffee machine even more and make it feel and look more real.