CS 4999 Independent Research

Midterm Report

Cornell University Program for Computer Graphics

Kenneth Lim (kl545), Leart Albert Ulaj (lau8)

April 3, 2014

Contents

1 Overview

Microsoft's new Kinect for Windows, released under the Kinect for Windows Development (K4WDev) Program incorporates a time-of-flight (TOF) laser for depth-sensing at an effective range of 1–15 feet. The higher level of precision and increased range achieved by the TOF laser (compared to the older Kinect, which uses a pseudo-random infrared dot pattern) makes the new Kinect a suitable candidate for gesture-recognition applications in (1) environments where it is not always possible for the user to be situated immediately in front of the sensor; and in (2) environments where the user may not be the only body in the sensor's field-of-view (FOV).

1.1 Objectives

1.2 Literature Review

2 Project Specifications

The software is conceptualized as a hybrid client capable of supporting multiple connections. The back-end is written in a low-level language that interfaces directly with the Kinect (via the Kinect API) and uses the workstation's GPU to convert raw sensor data into well-formed intermediate output. The intention is to offload processor-intensive computations to the workstation, and stream the output to the front-end, which is a lightweight application with little to no dependencies. The front-end consumes the output and renders graphical content to the user's display.

2.1 Hardware

We utilize the new Kinect sensor, provided to the Cornell Program of Computer Graphics by Microsoft under the K4WDev program. Interaction with the application was done on an 82-inch Perceptive Pixels Inc. (PPI) touchscreen display at HD resolution. Testing was performed on a secondary, 52-inch PPI touchscreen display at 4K resolution.

2.2 Back-End

2.3 Front-End

Content schema and human interaction feedback were rendered in real-time in a modern web browser. The application leverages the HTML5 WebSockets API to connect to the back-end, then renders incoming data on a GPU-accelerated HTML canvas layer overlaid on the application user interface. Serving the application via a web interface eliminates

the need for users to install any software on their devices (since all assets can be hosted on a remote server) and allows for multiple connection instances in a large classroom setting without additional infrastructure overhead.

3 Road-map

3.1 Phase I — Core Functionality

3.2 Phase II — Establishing Coherence

3.3 Phase III — Gestural Interaction

3.4 Phase IV — Content Development

3.5 Phase V — Testing and Robustness Evaluation

3.6 Phase VI — User Recognition

3.7 Phase VII — Gesture Learning

4 Deliverables

4.1 27 January

4.2 24 February

4.3 31 March

4.4 28 April

4.5 End of Semester

5 Extensions