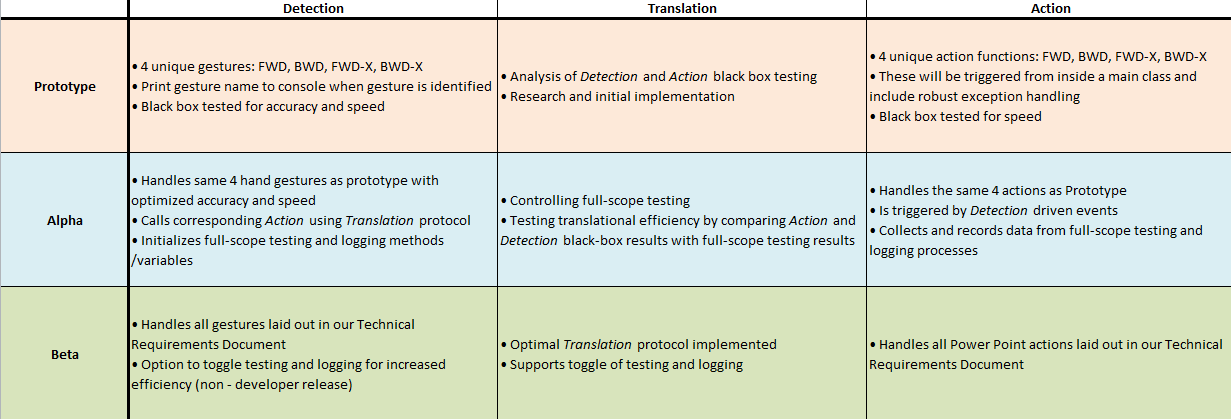
CSE 379: Project Implementation

Matt Raporte and Greg Potter

11/2/2015

Intro:

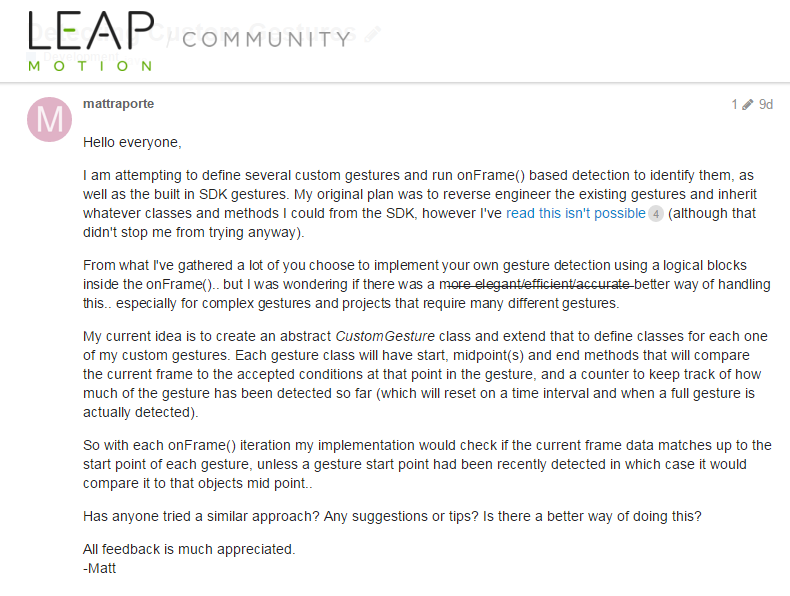
Since “Project Design” we have made great progress on each of our 3 components and have fully completed our Prototype version goals for each component. We’re also extremely close to completing Alpha release goals, however we’ve been busy with exams/conflicting schedules and haven’t been able to meet up to combine the 3 components yet, but we’re planning on doing that this Wednesday afternoon. In the paragraphs below we provide a technical breakdown of the progress we’ve made for each component. The diagram below the intro outlines our goals for each phase of the software development cycle, broken down by each component.



Detection:

We have 100% of our prototype (basic features) *Detection* component completed, with working detectable gestures for “FWD”, “BWD”, “FWD-X” and “BWD-X” which function by creating Listener and Controller objects, using those to check each data frame from the Leap Motion, detection SDK gestures “Swipe” and “Circle” and then extracting direction coordinates/number of rotations in order to distinguish inverse gestures and how many slides to skip (the “X”), respectively.

Additionally, we have briefly skipped over the Alpha release goals (which are to combine --the basic feature 4-gesture *Detection* and *Action* prototypes) due to scheduling issues and exams, and have instead gotten an early start on the Beta release goals for this component. We’ve been able to create 2 custom gestures: “Clap” and “Make Fist” by storing relevant tracking data from previous frames and comparing it to each frame’s current data using logical statements. These gestures are working with about the same accuracy as the Leap SDK ones, however we are experiencing latency issues. We have tried the following implementation:



However, latency was still a big problem so we’re working on implementing a variation of the $1 algorithm (<http://depts.washington.edu/aimgroup/proj/dollar/index.html>) developed by the University of Washington which should boost our accuracy and speed greatly.

Factoring in the amount of research we’ve had to do for this part, I would say we’re about 80-90% finished with our “reach” projected goal.

Translation:

We have a working queue to transport commands from the *Detection* phase on to the action phase. Instead of using some sort of file-based communication, which would have many open()/close() and thus high overhead, we are using Microsoft Message Queueing (MSMQ) to handle this communication. While the work to set up this queue is 100% done, we haven’t done any performance testing yet and, so far, time between dropping a message on the queue and *Action* has been negligible and perhaps we can look into optimizing it, if necessary, soon. We also currently log all of the data that is passed on it for debugging purposes and for the next iteration we will remove that (or at least disable it with some sort of flag).

Action:

Currently, control of the slides (next, previous, forward/back by *n*) are 100% working which is consistent with our alpha/prototype. Originally, we were going to integrate it with PowerPoint’s “ribbon” navigation as a VSTO add-in. This had issues with making sure that it could very simply access the current slide when running. Instead, it acts as a separate, console application which connects to the PowerPoint application using:

Additionally, we made a simple UI for testing without the *Translation* piece:



One interesting effect that we noticed is that this code works regardless of whether or not the presentation is running. In other words, motions can change the slides even when editing the presentation. We haven’t decided yet if we should allow this, but we are leaning towards adding a check before the project is finished.

Right now, we are 100% done with the *alpha* phase of this part. The code has been fairly straightforward which has allowed us to focus a lot of time and effort on *Detection*. Moving forward, we just need to implement some of the other commands. Overall, we consider this about 80-85% done.