Team Name: SHAPE
Team Members: Patrick Bowling, Andrew Lewis, Nick Strazis, Aaron Kwan, Jonah Simon
Instructor: Behrooz Shirazi
Mentor: Hassan Ghasemzadeh
Sponsor: Mona Ghandi
Course Number: CptS 421
Date Submitted: 3/7/17

Concepts, Algorithms, or Other Formal Solutions Generated

The idea for the user interface for the wall is for it to be both simple to use and convenient for the user. Thus, our vision for the wall aligns with this idea by having a simple Android application that the user can use to adjust the wall precisely and in a matter of seconds. This works especially well because all of our users will have smart phones, and this design can also be easily ported to iOS devices.

The concept behind the Android application is for all users to have very easy access to changing the wall with minimal effort and button clicks. The design that we have used for this fits the design that Mona has outlined for us, giving the user a rectangular box to choose where the wall will be open, as well as having two size adjusters for adjusting the width and height of the wall (see Image 1 in appendix). This application has been developed with Mona's design in mind, as well as Hamid's working model of the wall that is currently in GrassHopper. Hamid has given to us the dimensions of his working model, and the maximum values that can be taken in.

Solution Selection Process - Discussion of Design Choices

Our design follows the principle of having grasshopper pull formatted information from a server. This is because in this scenario we can arbitrarily add more input devices and sensors, and have them send the formatted data to the server. Grasshopper will treat that new data the same because it will be formatted properly, and so adding a new device or input method simply requires getting the required information on the server, with no extra changes to the grasshopper model.

Specifically this server-based design choice allows us to approach the overall problem in stages, where we first set up the structure, and then add new input methods. Once touch input is completed via our android app and we have full verification that the proper data is being communicated to and from the server, we can move on to additional input methods. For voice command for example we will collect raw data (the user's voice) translate it into commands using software, and send formatted data to the server based on those commands. Because the voice commands and touch input are used to make the same types of changes to the wall they were be formatted the same and grasshopper will not differentiate on what input method provided the data it is pulling from the server.

Theoretical or Formal Description of Selected Solution, Including Considerations for Software Testing (both Alpha and Beta Versions)

Our decision to pass the information via an app and server is based on the fact that an app is a platform that is easy for user's for interact with. By letting the users use an app to alter the wall, we can look at how fast our wall reacts to user input as well as determine if the wall is able to plot the locations properly based on the user's input. As far as testing this would allow us to interact with the wall remotely (as a final product might be used) so it would make it easier to to test ourselves, and to have others test.Similarly we could have sensors for voice or video that are not physically located near the wall during testing which would ease testing as well. Lastly with this solution choice we can directly manipulate the formatted data on the server to test how worst case scenarios, or unexpected input affect the system, we can locate bugs easier, and we can easily design test cases for any situation we need.

Summary of the State of the Project

The current state of the project is on schedule. As of this moment, touch has been developed to a level that is acceptable based off of comments made by Hamid (see Appendix A). As for the back end part of the project, development of the back end is on track to be finished within the 4 week time period given to us. Currently the back end part of the project has accomplished pulling information from the server, parsing the data to obtain the information we want, and sending that data to the object to move it. The last step we need to accomplish is to set up an event handler that will notify grasshopper of changes so grasshopper can update the object live. With the current mini projects wrapping up, the team will now prepare the next mini product which is the voice aspect of the project.

Future Work for this Semester

We still need to add an event that will allow Grasshopper to update when a change is made to the data on the server. Right now, Grasshopper pings the server on a set schedule, every second or so, using a timer. The solution to this is still being decided, but our best bet at the moment is to create an event on the server that will trigger when new data is sent to the server. This event will tell Grasshopper that new data has been sent so that Grasshopper can make the corresponding changes.

Once we have finished our methods for reading and writing of data, we will be able to add extra functionality to our prototype. The next thing to add will be voice recognition; in theory, the user will be able to manipulate the wall by saying a command. For this, we will most likely use Amazon's Alexa because it has an open-source API, the "Smart Home Skill API". This will involve adding skills to Alexa that respond to voice commands like "Alexa, open wall" or "Alexa, make window". One limitation of this is that we have to begin the voice commands with "Alexa". This means that our voice commands will not be as natural as intended for the final product. However, since this is only a prototype, we think that this is the best option to get the basic functionality working as fast as possible.

We also need to add the option to interpret data from camera sensors in order to detect the user's gaze. We do not have as clear of a path for this as we do for voice recognition, but our first thought is to use Microsoft's Kinect. The Kinect, like Alexa, has a built-in API that we can read video data from (It also has an API for audio which could potentially be used for voice recognition, but Alexa's API will interpret voice commands much more easily). This will eventually be able to track the user's gaze and react accordingly.

Appendix A: Screenshot of User Interface Using Touch Controls