# SHAPE Android App and Server Report

---

Patrick Bowling
Andrew Lewis
Nick Strazis
Jonah Simon
Aaron Kwan

## Android Application

For the front-end, we created an Android App that formats and sends data to our server that hosts an XML file. The XML file contains the values needed for grasshopper to build the wall. Our app takes input from sliders and touch location. The touch location signifies the center of the window and the sliders are the width and height of the window.

## Concept And Research

We wanted to create a natural way to control the wall through touch. In order to accomplish this we researched many different touch input systems. We decided to start with creating an Android App. We went with Android because there was lots of documentation and we had worked with Android in the past. Additionally, Android apps are built on Java. Java is very well documented and easy to work with.

## Development

We split the task of creating the app into small pieces. We started with making a simple slider based interface. The slider interface did not take much work and did not present any challenge. Next, we focused on creating a way for the app to send data to the server. This task presented many challenges. The first major challenge was to find a Java library that could allow us to send data to the server. Eventually we settled on the java.net.URLConnection library. This library had a function called stream that could send data to the server. Implementing this seemed simple, but the app kept crashing without throwing any errors. We then learned that on Android apps, certain functions must only be run from another thread. After learning how to utilize threads, our app was

finally able to connect to the server. For step 3, we wanted to make a more complete interface. To do this, we had to come up with the math to map pixels on the screen to the wall. In addition, we had to find a way to send those values to the function that send data to the server. This was difficult because the send data function had to be in a different thread. Our solution was to close the thread after each send. This made it possible to pass values to it on startup. Our app is a culmination of these three steps.

## Future Plans

We have many future plans for this app. We would like to add support for a visual representation of the wall. The representation of the wall would allow the user to preview the changes before sending them to the server. Another feature we would like to include is more accurate touch recognition. Currently, our touch canvas is very small. The small size can make it difficult for users to pinpoint where they would like the window.

Overall, we are very happy with the development of the app to this point. We faced many challenges and overcame them. The app is currently working flawlessly.

## Current State

In its current state the app supports-

- Place a window anywhere on the wall
- Change the width and height of the window
- Submit the window to the preview canvas
- Submit the wall to grasshopper

## Back end application

For the back-end team, we were tasked with creating a way for grasshopper to take in the information sent from the front-end app and convert the information to something that can create the window.

We broke the task into three smaller more manageable tasks. The first being trying to find a way to connect grasshopper to the server itself. The second task then became communicating with said server so information can be transferred, and the third task was to then take the information and use it to create the window. The back-end team decided that the best way to create the backend application would be through grasshopper itself instead of a third party application. That way, any changes made to the application can be done with just grasshopper. As the back-end team worked on the project, we kept the front-end team updated so any problems could be dealt with immediately.

As we developed the backend for the project, we encountered a few problems that inhibited us from completing the project. The first problem we encountered was with the type of file that was being read in. Based on the file, processing the information would have to be done differently. In order to deal with this problem, the back-end team had to communicate with the front end app team in order decide on the file type that would be most convenient for both teams. After a few days of communication and testing a few file types, we decided that XML files would be the easiest to work with since there was a grasshopper plugin known as GHowl that could read XML files from a passed in URL. Another problem that the backend team encountered was that there

was no way to efficiently update the grasshopper information after user input. At first, we attempted to create a messaging application that would send notifications to update after user input, but we encountered more problems trying to establish a reliable and secure connection with the server. We solved this problem by creating a timer that will renew the information every second, but this solution requires much more processing power as Rhino will have to redraw the window every second. We are still attempting to create the messaging application to make communication more efficient, but it is not complete. Other than these two problems, there were no other problems that are worth mentioning.

All in all, creation of the backend went relatively smooth and all problems encountered were dealt with. There are only two notable issues left; communication between the servers needs to be more efficient if we are to put Rhino and grasshopper on the server, and the backend application will need to be updated should the user want to put anything besides a location, a width, and a height.