MorphoGenesis

CPTS 423 Final Report

Team S.H.A.P.E

Aaron Kwan

Andrew Lewis

Jonah Simon

Nick Strazis

Patrick Bowling

Customer

Mona Ghandi

Table of Contents

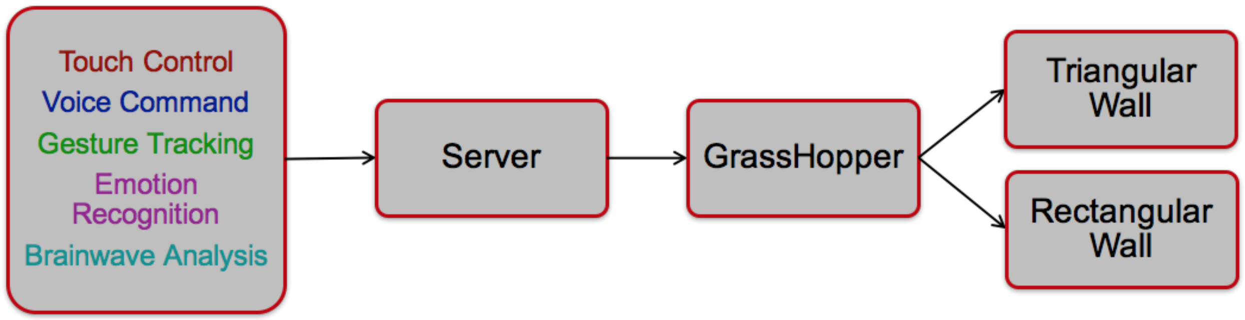
Executive Summary

In today's world, the environments that surround us in the workplace are static and do not adapt to our needs and desires as humans. Our aim is to create a dynamic environment that can respond to user input and change its own configuration to match the user's desires. Currently, our problem is trying to get a user to manipulate a wall based on their feelings and through simple actions such as voice and movement. We used a widespread approach to solve this. We built prototype applications including a Django Server, Mobile Application, Alexa Skill, Facial Emotion Recognition, Brainwave Analysis, and Kinect Gesture Recognition. All of these input methods allow the user to interact with a movable wall. The end goal for the project is to develop models that will provide an easy way for users to manipulate their environment through the use of smart home devices. Our project has the real possibility to change how we interact with our environment as emotional beings. This project could affect how all buildings are designed in the future.

Introduction

What if our environments changed to our needs as humans? Our project aims to make our environments adapt to us. As technology advances, we are able to interact more and more with the world around us. Our project aims to tackle the question of how we can have more control of our environment in a home setting based on our needs, desires and mood, at any given time. At the present time, no home exists that will morph and change shape to suit the occupants. To tackle this issue, we started small with creating a wall that will adapt it's shape to the user's needs. The wall would change shape by opening and closing windows and doors as needed by the user. For the wall to be intuitive and work for every human being, we developed a set of ways to interact with this wall. Currently, users can interact with our wall through touch, voice, face detection, gestures and dance, and even brainwave analysis.

Description of Culminating Design



The overarching design for the project is very simple. All user input applications work by sending configurations as JSON objects to the server. The server then stores the configurations. Next, the wall is manipulated by Grasshopper. We extended Grasshopper to ask the server for the most recent configuration and then update the wall based on that most recent configuration.

We found that this design was very efficient and effective. It allowed us to design the input devices around the user and their feelings instead of around the wall. All the clients had to do to change the wall was send a configuration to the server. Since nearly every modern language supports HTTP requests, we were able to use devices that supported nearly any modern programming language. This is how we are able to manipulate the wall through such a wide variety of devices. Additionally, by separating the server from the wall, we were able to try out many different wall models without making changes to the input devices or to the server.

Project Management

Based on the project requirements, the team decided that the best way to complete the tasks assigned was to break up into smaller groups so multiple tasks could be completed at the same time. The new teams were created based on need and interest in the task and were as follows:

Project Task	Team Member(s) Assigned
Building the server	Andrew, Jonah
Upgrading the server	Andrew
Setting up server for machine learning	Andrew
Setting up grasshopper	Aaron, Jonah

Making the mobile app	Andrew, Nick
Making the Alexa app	Andrew, Patrick
Making the Kinect app	Aaron, Nick
Making the affectiva app	Nick
Making the emotiv app	Jonah
Exploring other possible applications	Aaron, Andrew, Jonah, Nick, Patrick

Because of how people could not be on two tasks at once, the team managed the time allocated for each task with gantt charts to ensure that there was enough time to complete each task. Each task was given at minimum 2 weeks to make sure that the assigned members wouldn't feel rushed about finishing projects. Any extra time was allocated to projects that were either falling behind on deadlines or to projects that the team wanted to explore further. We kept each other accountable through weekly check-ins with our instructor and customer.

Results

Final Prototype Description

Affectiva (Face Detection)

To read the user's facial emotions, we used facial recognition software from a company called Affectiva that provides a free SDK for open development. The software uses a webcam to read the user's facial emotions by placing anchor points on the image that is read in and using them to determine the user's emotion through affectiva's custom machine learning algorithm. In order to get the image of the user's face, the user must sit directly in front of the webcam, no more than 3 feet away. The image of the user's face is processed by the Affectiva SDK and we then send an HTTP Post request to the server based on the emotion that is returned from Affectiva's algorithm.

Currently the Affectiva application only supports configurations for four predefined emotions, but has the capability to detect 21 different facial expressions resulting in a total of 7 emotions.. By far the easiest emotion to read is 'Happy' as the Affectiva application does a great job in detecting your smile. Other emotions like 'Angry' rely on multiple different anchor points on the face and can be slow to respond to the emotion the user is showing.

The software is built on a web application using HTML and JavaScript that is hosted on our server, so it can be used by any electronic device that has internet access and a webcam, however the application will not currently work for the Google Chrome browser due to our server not being HTTPS compliant. As a workaround users must use FireFox to view the Affectiva page on our server.

Alexa (Voice Interaction)

To receive voice input, we created an Alexa Skill using Amazon's Alexa algorithm in conjunction with Amazon's Echo Dot speech input device. We decided to utilize Alexa because of the flexibility offered in their Alexa Skills Kit which allows anyone to easily create a custom, personalized Skill. This Skill is then linked with any Alexa-supported device where the owner of the Skill is logged in. The Skill also uses an Amazon Lambda Function to process the user's speech and interact with the server.

Users can interact with the wall by first calling upon the custom Alexa Skill. This is done by calling upon the Skill's invocation name, SHAPE Wall, by saying "Alexa, open SHAPE Wall" when within range of an Echo Dot with SHAPE Wall support. Alexa will process the input, recognize that the user wants to open the SHAPE Wall skill, and execute the desired action while greeting the user. The user can then say what emotion they are currently feeling (e.g. "I am happy.", "I feel sad."). Alexa receives the user's utterance and parses out the emotion word, classified as an "intent." This intent is then sent to the Lambda Function. The function reads the intent value (e.g. "happy", "sad") before sending an integer to our server based on the given value. This integer is sent through an HTTP POST request. From there, the wall is updated by performing an HTTP GET request to the server in order to get the new value. The user can repeat this process as many times as they like. When the user is finished, the Echo Dot will exit out of the Alexa Skill on its own after 15 seconds of inactivity.

Emotiv (Brainwave Analysis)

The Emotiv application allows the user to send data to the server via their thoughts. The user puts on a headset with sensors covered in saline solution. The user sends by thinking of emotions (happy, sad, etc) and by doing so will release electrical impulses that will fire off the headset sensors. Based on the combination of sensors being fired off, the headset will determine which emotion the user is thinking and do a HTTP POST request to send the data to the server.

Grasshopper (Wall Interaction)

Grasshopper was facing a problem where it was having compatibility issues with the new server. With the updated server, the data was being stored as JSON instead of XML. This became a problem since grasshopper can only take in XML files. In order to deal with this, we developed a script that will take the incoming JSON and convert it into a wall object. Once the wall object is created, the script will output the values of the class.

Kinect (Gesture and Movement Interaction)

The Kinect application is now split into two applications. One which allows the user to manipulate the coordinates for the wall via gestures and One which has the user sending commands to the wall via dancing. With the addition of the full body movement application the user will have their entire body tracked and will send messages based on hitting specific poses with their body as they move or dance. These two applications gives us an inside look at the possibility of sending information about a person via their body motion.

The application that moves the wall via coordinates is done by taking the user's body and drawing their head and their hands. Once the head and hands are drawn, anchor points are placed on the hands and head to track the movement of other anchor points. The user can send commands to the server by making sliding and swiping gestures. Once the user manipulates the wall to their liking they submit by putting their hands down. This lets the program know that the user is finished making changes and will use a HTTP POST request to send the data to the server.

The application takes the user's body and draws their skeletal structure. Once the skeleton is made, anchor points are placed at key joints/body parts and are used to track the movement of other anchor points. The user fires off commands by moving their body into specific poses which triggers a conditional statement letting the program know which pose the user chose. Once the user picks a pose the program will notify the server using a HTTP POST request.

Mobile App (Touch Control)

The mobile application with touch input (see image 1) has reached its final stage of conversion from a coordinate based approach to the preset emotion approach. The user is now able to interact with the application quickly and efficiently by selecting and confirming (see image 2) an emotion from a list of preset emotions.This allows for an altogether easier and faster experience for the user by cutting down on the time required to select the configuration for the wall. This does come at the expense of not being able to customize exactly what shape the wall will make. We think that creating the app in this way ultimately affords an overall better experience to the user.

Furthermore, the mobile application now sends a preset integer to the server based on the emotion that was selected by making an HTTP Post request to the server. Currently these emotions have been set to certain presets that map to preset configurations for the wall. These preset values can be changed and customized to any integer, or any other value, completely at the discretion of the mobile app programmer or the server programmer. We have designed it this way so that the values can be easily changed in the future if the specification for the program changes.

The user interface for the mobile application also supports the ability to select two different emotions (see image 3) and send a preset value based on the two selected emotions. An example of this is if the user is feeling both 'Sad' and 'Angry', they can chain the two emotions together and send a compounded value to the server. This value can be changed easily in the code, allowing for the programmer to easily customize the wall configuration for multiple emotions depending on changes in the specification for the program.

The last and biggest change that we made to the mobile application was to change it from a native Android application developed using Java and Android Studio to being a web based application that uses JavaScript, HTML, CSS, and the Bootstrap framework for responsive web design. Making this change has had two very important impacts to how the mobile application runs and how it looks. To begin with, because the mobile application is now a web application, it can be run on any electronic device that has a web browser and internet access. This means that by writing this one application we can now run it on Android and iOS devices, Windows and Mac devices, and most other computers. This is a huge improvement to our program because in our previous design, the app was built specifically for Android devices and therefore could only be run by devices that run the Android OS. The Bootstrap framework is important as well because it provides a responsive web design that will change to correctly fit the size of the screen that is accessing the website. This means that the web application will look like a native application to whichever device is accessing it. Also, iOS and Android devices provide the ability to develop a native application that loads a web page, called a webview, so we can create native applications for both iOS and Android devices without needing to change our existing code, and only needing to write very easy webview applications for each specific operating system. Overall this is a huge improvement to how our mobile application is designed because it allows for many more mobile input devices such as iPhones and Android phones, without needing to change any of our core code.

Server (Wall Interaction and Machine Learning)

Early in the project, we determined that we needed some sort of server that would act as intermediary between the many input devices and the wall. We started by creating a simple system where wall coordinates could be sent to the server through the url and a php script would write the coordinates to a file. This early server architecture worked well enough at the beginning of the project, but it’s limitations arose quickly. It was limited in how it could only handle a few requests in a row before shutting down. Additionally, there was not an efficient way to store the history of the wall usage. Therefore,in the second semester we adopted an entirely new server.

The new server is much more capable and overcomes the limitations of the original. It is a RESTful API built on Django and the REST API Extension. The server interacts with the clients and the wall by supporting HTTP GET and HTTP POST Requests. The input applications all send configurations and coordinates formatted as JSON objects to the server via HTTP POST requests. The wall and grasshopper get the current configuration they should be at by asking the server for the current configuration every 0.5 seconds with HTTP GET requests.

In addition to serving and changing wall configurations, the server is also the start to the machine learning aspect of the project. Currently, the server stores the data required to support a sophisticated machine learning algorithm in the future in a MySql database. It stores the wall configuration, the time changed, and the device that changed it. Even though the server is not using full-blown machine learning, the server currently uses this data to predict the user’s prefered wall configuration, given a time of day. It does this prediction by analyzing the users usage and changing the configuration to the one they most often use for the time of day. For example, if the user is changing the wall to angry in the mornings the majority of the time, the server will start to change the wall to angry in the mornings.

Final Prototype Test Results

After meeting with our customer, we determined that adding automated testing was not the best use of our time. Our customer preferred that we focused on creating more prototypes. Below, we discuss the manual testing of each application.

Affectiva (Face Detection)

The code that we added to the application is relatively simple, with only short statements and an HTTP Post request. Therefore our team came to the conclusion that the best way to test the Affectiva application was to use manual testing by cycling through the different emotions in real time and confirming that the correct configurations were sent to the wall.

Alexa (Voice Interaction)

The Alexa Skill was tested by reciting the list of possible utterances and confirming that the wall was updated accordingly. We decided that it was a more efficient use of our time to do this rather than create automated tests due to the low amount of possible utterances.

Emotiv (Brainwave Analysis)

The Emotiv application allows the user to send data to the server via their thoughts. The user puts on a device with sensors drenched in saline and begins to think of emotions. The headset will detect the electrical impulses in the brain and based on the number of sensors that fire off will send a command to the server. When testing this application, the team found that live users were the most efficient way for us to test it.

Grasshopper (Wall Interaction)

Based on the results of the test, the team decided that Grasshopper worked as intended. The only flaw the team could find was that grasshopper wasn't requesting data from the server very efficiently, but this should have no effect on performance.

Kinect (Gesture/Movement Interaction)

After some consideration, it was determined that the kinect applications did not need any automated testing. This is because all inputs are based on the user and their body movement. Attempting to do automated testing would be more difficult than bringing in a real person and having them move around. After testing with an assortment of users, the team decided that while the gestures/poses were acceptable, improvements were needed for triggering commands.

Mobile App (Touch Control)

The Mobile Application does not have any automated testing because the functionality is very simple and relies purely on the user input gathered by the user interacting with the application. Instead we use a form of manual testing where we go through each of the configurations and determine that each is working correctly and sending the correct information to the wall. Our team found that this is the best approach to testing the mobile application software.

Server (Wall Interaction and Machine Learning)

As with the other applications, the server does not have any automated testing. All testing is done by users. The server has been very thoroughly tested through our user testing. Every application in the project uses the server in some capacity. In addition to testing the server with the other applications, we can test it directly by going to its web address and making HTTP Requests. Throughout the semester, we tested every server change by making these direct requests.

Final Prototype Validation Results

The results were validated as we used a wide variety of users to make sure that any issues that needed to be addressed were ones that were produced by every type of user. The users ranged from male to female, young to old, and knowing of the technology to not knowing the technology. This wide range of people made it so that the team was able to verify that all potential users could properly use the devices. Since all of the users were able to properly manipulate the wall using our devices, it is safe to validate the results.

Analysis, Modeling, and Implementation/Simulation Results

Our Analysis, Modeling, and Implementation/Simulation Results were very promising. All of our prototypes worked as intended. We demoed each of the input devices to our customer throughout the semester. With each demo, we showed the application working with a model wall in Rhino and a real life wall. Additionally, with each input demo, we were effectively showing the server and grasshopper working. Overall, our customer was very happy with our results.

Broader Impacts and Contemporary Issues

The concepts used to manipulate the wall have the potential to solve many current issues. Currently, physically disabled people struggle to interact with their homes. Many of them can’t even open doors or windows on their own. Our project has the potential to solve that issue. The disabled person could open or close their windows/doors through many different devices. This would allow many disabled people to feel and be more independent. Additionally with our data collection and machine learning, the disabled user may not have to interact with any of our input devices. Instead the window/doors will learn to move when the user would like them to.

Another issue that our project impacts is those who feel depressed and alone. Our project aims to make the environment, such as a home, feel like a living thing. If a person is lonely, they could simply interact with any of our devices and their homes will respond. For example, if the user would like to dance and is sad because they have no one to dance with, they could simply dance with their homes. One of our devices (kinect) recognizes dances and can make the environment dance with the user.

Limitations and Recommendations

Affectiva (Face Detection)

Currently the main limitation of the Affectiva application is that the user needs to intentionally turn on the application and look directly into the webcam in order for the application to update the wall based on the users facial emotion. Our team would recommend that future teams create a user environment with multiple face tracking cameras so the Affectiva application can see the user from most or all angles and can immediately update the wall when the user changes their facial emotion. In this case, if a user were to smile while not looking directly into a specific location, the Affectiva application could then immediately send the emotion to the wall and the wall would quickly change to the new configuration. Our team thinks that this will be a much more natural and immersive experience for the user.

Alexa (Voice Interaction)

Right now, there are only six emotions supported by our Alexa Skill: happy, sad, angry, surprised, scared, and disgusted. The Skill also only accepts utterances in the form of “I am [intent]” or “I feel [intent]”. Future improvements could include support for a wider range of emotions as well as reception of those emotions in more natural language. To take it a step further, the Skill could listen for utterances that are not just declarations of current feelings (e.g. “I just aced an exam!”, “This meal is gross.”) and determine what the user is feeling based on that.

Emotiv (Brainwave Analysis)

As of right now, the Emotiv needs to be trained for each specific user. There is also the need to have each sensor heavily drenched in saline so saline will need to be applied every so often when using the headset. The team recommends finding a way to have the sensors be receptive of electrical impulses without the need for constant application of saline.

Grasshopper (Wall Interaction)

Currently, the grasshopper plugin is limited in the sense that the only way Rhino can get data from the server is if it pings the server for data every second. Even though the server can handle the load, the team feels that it is putting unnecessary stress on the server. In the future, the team recommends changing it to a push notification so grasshopper won’t ping the server every second.

Kinect (Gesture/Movement Interaction)

At the moment, the conditions for firing off each gesture/pose are too broad and could create problems in the future when new additional gestures/poses are added. If the conditions for triggering commands is not fine tuned, there is the possibility that body movement could trigger two different conditions at the same time which is not good.

Mobile App (Touch Control)

Currently, the biggest limitation of the application is that we have a list of four preset emotions for the user to select from. This is not a realistic amount of emotions for the user to choose from, so it would make sense for the user to be able to customize the options that they can choose from and be able to customize the configuration that the selection will send to the wall.

Secondly, there is currently limited support for the user to be able to choose two emotions and send a compounded configuration to the wall. One way to improve this is to let the user choose two emotions and choose whether they feel more of one emotion or the other. For example, if the user chooses the emotions ‘Sad’ and ‘Angry’, we could take a more dynamic approach and let the user choose to what degree they are feeling ‘Sad’ or ‘Angry’, and which one they feel more of. Then we could send a dynamic configuration to the wall based on the user’s customized selection. We feel that this would be an important feature of the application since it is common for people to be feeling a mix of emotions, and this would help personalize the users interaction with the wall.

Server (Wall Interaction and Machine Learning)

The server is currently hosted at pythonanywhere.com on their free hosting plan. This means that the server only has a limited amount of storage and can handle a limited amount of traffic. That means that the current server implementation would not support a high volume of users. We recommend that if the wall ever grows to have more users, that the server hosting be upgraded to a paid plan. This would allow the server to support a high volume of users.

Conclusion

In conclusion, we have proven that the technology is there to make our environments adapt to us instead of the other way around. We have built ways to interact with the environment with voice, gestures/dance, facial emotion, touch, and brainwaves. Additionally, we have created an infrastructure that allows many more future input devices to interact with the wall. Our project shows the potential to SHAPE how we as humans interact with our environments.

Future Work

Although over the last two semesters we have made much progress, there is still a lot to be done before we really have the household of the future. Currently, our input devices are all prototypes. They all could benefit from refinement and further usability improvements. Additionally, all of our applications are currently lacking automated testing. This will need to be implemented before the project leaves the prototype phase. More future work can be done by adding more clients such as a emotion reading arm band and supporting more types of walls. Finally, the server would greatly benefit from a machine learning algorithm as it currently has the data stored and formatted for the algorithm to learn from.

Acknowledgements

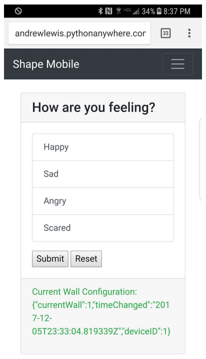
We would like to thank Yang Zhang for his contributions in linking Grasshopper with the wall, creating the triangular wall, and the Tobii app. Additionally, we would like to thank Hamid Esmaeillou for his contributions in creating a physical wall and virtual model for us to work with. Also, we would like to thank Professor Shirazi for his project planning guidance. Finally, we would like to thank Professor Ghandi for her contributions to the concept and her ideas for input devices.

References

https://docs.djangoproject.com/en/2.0/
http://www.django-rest-framework.org/#tutorial
https://help.pythonanywhere.com/pages/FollowingTheDjangoTutorial
http://getbootstrap.com/docs/4.0/
https://knowledge.affectiva.com/v4.0.0/
https://developer.amazon.com/alexa-skills-kit/alexa-skill-quick-start-tutorial
https://emotiv.github.io/cortex-docs/#introduction
https://msdn.microsoft.com/en-us/library/microsoft.kinect.tools.aspx

Appendix

Image 1**:** User interface for Mobile Application (running on Android device)



34% 8:37 PM

andrewlewis.pythonanywhere.cor

Shape Mobile

How are you feeling?

Happy

Sad

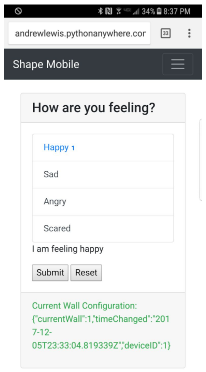
Angry

Scared

Submit Reset

Current Wall Configuration:
{"currentWall":1,"timeChanged":"2017-12-05T23:33:04.819339Z","deviceID":1}

Image 2: User interface for Mobile Application (running on Android device)



34% 8:37 PM

andrewlewis.pythonanywhere.cor

Shape Mobile

How are you feeling?

Happy 1

Sad

Angry

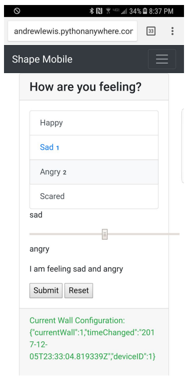
Scared

I am feeling happy

Submit Reset

Current Wall Configuration:
{"currentWall":1,"timeChanged":"2017-12-05T23:33:04.819339Z","deviceID":1}

Image 3: User interface for Mobile Application - user selected emotion 'Happy'



34% 8:37 PM

andrewlewis.pythonanywhere.cor

Shape Mobile

How are you feeling?

Happy

Sad 1

Angry 2

Scared

sad

angry

I am feeling sad and angry

Submit Reset

Current Wall Configuration:
{"currentWall":1,"timeChanged":"2017-12-05T23:33:04.819339Z","deviceID":1}