Week 4

Completed November 18

Trush had the video manipulated since week 1, and had the video displayed on the figure. All we had to do now was to add the data analysis part to the program. Shane was able to get the original video to show up on the top left and the manipulated video to show up on the right. Below both video is a bar graph that shows the color proportions for every frame until the video was over. Always it calculated the differences live! Below the bar graphs, we had the 3D cluster to show up. It would show the 3D cluster being updated every frame for the original video and manipulated video. Also we included the color line variation in order to have it look similar to the Disney Animated app created by Theodore Grey. We struggled with this, but the man himself was able to help us create it! Here are the links to the Manipulated video Data Analysis: <https://www.youtube.com/watch?v=2DLCQYCy_aw> <https://www.youtube.com/watch?v=oD7uPBSo3xM>

Completed November 18

Shane worked very hard on this. This was possibly the hardest thing he encountered while coding. He flew by the other parts of the project easily, but this was slowing him down a bit. Therefore we contacted Theodore Grey. Theodore Grey was kind enough to figure us part of his code in order to help us create the color line variation. Originally we only had 6 colors, but after adding more colors, we ended up with a total of 30+! Now, the color line variation looks way better than it did before. With the 6 colors, it looks very low, quality wise. When we added more colors, it looked more modern. We hope to carry this project one day in order to get our color line variation to show up just like how Theodore Grey had his color line variation to show up. Here is what Shane has made so far! Pretty cool!

Here is what we had before:

<https://www.youtube.com/watch?v=2DLCQYCy_aw>

Here is what it looks like now:

<https://www.youtube.com/watch?v=oD7uPBSo3xM>

Completed on November 21

This week, we have gotten a lot done. We researched how to get more colors into our program. We want more colors so customers can see a more detailed analysis. The reason we I say, customers, is because we are trying to sell this idea, not just created because it's a project that we will get graded on. If we treated like this project was to get a good grade on the final then we would have low expectations. If we treated it like we were selling this, then we would be able to put more effort. So to make it a product worth selling, we included more colors to get a more detailed analysis. So far what we did was, we researched the RGB values of different colors, then we looked at the proportions of each RGB value to get it to display specific colors. Yao did a very nice job while researching the different color values. All we had to do essentially was to find more colors to add and then use the proportions. Wasn't too difficult, but was time-consuming as we had to make sure the values were correct.

Completed November 21

Here is what Trush found in the API's that we can use for MATLAB. The first one gives a detailed explanation of how we can use this specific API.

MATLAB API

.dom.\* api: You can use this API to create reports. Reports such as Word, PDF, HTML. This way a user and access code written outside of their home. This can be accessible outside of MATLAB. Here is an example of how the MATLAB .dom api works:

1) Import the package .dom.\* next below this code you will have to type rpt\_type to specify the type of report you want. If you want a pdf you type it in single quotation marks. If you want word it would look like ‘Word’

Import mlreportgen.dom.\*

rpt\_type = ‘pdf’;

2) Next you create a variable name. For example I have made the variable name called doc. After the equal sign, it is specified as document. This does **NOT** mean It is going to be a word document. It is basically going to create a empty pdf document. The document will be called mydoc. The append function will display the next. In this case Hello World.

doc = Document(‘mydoc’, rpt\_type);

append doc, ‘Hello World’);

3) The close method will close the variable and document name doc. The rptview(doc.OutPath); will open the document in in-built PDF viewer.

close(doc);

rptview(doc.OutputPath);

4) To create a paragraph, start off by naming the variable. The variable name for this code below is paraObj. Next to the equal sign is Paragraph. This will create the paragraph. Inside the parameters, the message gets displayed. Append(doc,paraobj); basically will append the paragraph made to the document

paraObj = Paragraph(‘This is a paragraph’);

append(doc,paraObj);

5) To insert an image use the code below. First create the variable name imageObj. After the equal sign write Image to have it display an image. The which part is to specify which image you want to use. For our example we are using image image123. Below this code we specify the width. The width we have set is 1.52 inches. This will be the with in pixels. Below the width, we have the height specified. The height will be 1 inch total. The append(doc,imageObj)); will add the image to the report.

imageObj = Image(which(‘image123.jpg’));

imageObj.Width = ‘1.52in”;

imageObj.Height = ‘1in’;

append(doc,imageObj));

6) The code below will be used for creating tables. First name the variable like always, next write Table to specify that it is a table. Inside the parameter allows you to create a 6 by 6 table. Append(doc, tableObj)); will create the table into the document.

tableObj = Table(magic(6));

append(doc, tableObj));

Some other useful API’s:

<https://www.mathworks.com/videos/upgrade-advisor-api-121582.html>-This is video shows you the demonstration. To summarize what I learned from this video is that it allows you to update the models. It will not only fix the models, but also show recommendations on what should be updated. So for Shane and I, we encountered an error that didn’t allow us to see the pixels displayed on the 3D graph. What this API allows you to do is to basically prevent that from reoccurring. It will analyze things such as the frame, model, functions, and properties. This will prevent us from having another problem to occur the day of the meeting with Professor Martin!

<https://www.mathworks.com/videos/faster-simulations-with-performance-advisor-90544.html> This allows you run models much faster. If the program finds any sort of way to speed things up it will do it. However, I have only seen logic gates being used. Couldn’t find any information on whether it can work on our 3D model.

Completed November 21

Shane was kind enough to explain his code. Since he is the lead programmer, most of the code that we have comes from him. He played a big factor in getting this project where it is right now. Trush helped him document some of his code, but Shane did most of the code commenting since he is the one who wrote the majority of the code. He knows his code better than any of us. Therefore, to help us understand his thought process when he was creating all his code, he made a video to help explain what was going on. He made an hour long video right after he came back from work at night. He was able to go over almost every single line of code with detail. And was totally honest when he said that even he himself didn't know what every code does. Shane explained everything thoroughly for both Yao and me to know what is going on. Both of us feel more confident with coding in MATLAB now after he made the video for us explaining the code and watching it.

Here is the link to his code explanation:

https://www.youtube.com/watch?v=NfeG2EDdjmE

Completed on November 21

This part was very cumbersome. Converting in MATLAB code to JAVA code was impossible. Literally. MATLAB can only convert to either C++ or C. Unfortunately, we couldn't do this either if we wanted to. We would have to buy a subscription in order to do this. Trush met with professor Martin on Wednesday to have a chat with her about this, since he was assigned to this part. She mentioned how if I couldn't get the code to convert from MATLAB to JAVA then I could at least make it so JAVA can run MATLAB code to have it display the figures. However, even after this being said, we had to make sure the school offered the JAVA Package SDK otherwise, this part would also be impossible. Trush contacted MATLAB through email and called them up. Both said it was impossible to convert the code, but it was possible to run MATLAB code on JAVA. The email provided the steps to take in order to have it do that. Trush doesn't feel comfortable with JAVA, and Shane didn't feel comfortable with Arduino. So they switched roles. Shane was able to get MATLAB output of the color proportions saved as a text file and then used that to create bar graphs on JAVA. Trush, on the other hand, took responsibility for getting the Arduino servo to spin in different directions, and have it display a "This is (Color)".

Completed on November 21

What we have in mind for this Arduino project, is to first create a wheel. Now, we cant have all 30+ colors onto a wheel because this servo is about the size of half of a thumb. Therefore we have limited the color wheel down to just 10 colors. Red, Orange, Yellow, Green, Blue, Pink, Purple, Black, White, and Grey. We also have to make it so the servo pointer is bigger. Right now the pointer is as big as a USB port. It will be hard to tell which color it lands on as the colors are already crammed together and the servo can only rotate one hundred eighty degrees. Next, we plan on making it so the servo spins at different angles. Trush was able to do this and also have it say which color the needle lands on. He did this while swapping positions with Shane as Shane was working on getting MATLAB code to run on JAVA. Now, all we need to do is have the color wheel made and have it so Arduino can retrieve the color proportions that JAVA got from the text file it got from MATLAB's color proportion output. We will create an ArrayList that will give the data of the color proportions to Arduino.