Curtin University

FUNDAMENTALS OF PROGRAMMING

SEMESTER 1, 2023

ASSIGNMENT

BY: SYED MUHAMMAD AHMED ZAIDI

STUDENT ID: 20972008

Contents

OVERVIEW

The requirement of this project was to simulate a concert-like scenario considering an old and famous rock band called Spinal-Tap. The project aimed to assess the students on their learning about the fundamentals of programming by giving them a practical assignment that allows implementation of the skills learned during classes and labs. Hence, the purpose of this program I produced, is to create a plot that shows a stage, props, band, lighting, backdrop, smoke machine, and choreography that animates with the use of different libraries, methods, and functions. The program comprises different features that are set aside using comments to show a full understanding of how each of them is adding their part in the overall structure of the concert scenario. These features include the creation of classes, the import of data files, the creation of lists and arrays, producing plots, attaching pictures and finally making everything animate with iterations and functions. In this report, every feature will be discussed thoroughly in terms of its code and also stepwise implementation.

USER-GUIDE

The code itself is stored in a file called spinal-tap.py that includes all classes, lists, iterations,and final output. To achieve the final output, this file needs to be put on the desktop of my Oasis login (20972008) and then run through python3. This will automatically bring up the chart showing the concert scenario. The code imports multiple libraries including, matplotlib, numpy, and PIL. These are done to get the required functions so that the code can be completed and run smoothly. In order to see the output, we just need to run the complete code in this file through python3, and we will be able to see the plot for the stage with all implemented features within. However, as the pathways are set with my oasis account, it will not be able to run with any other desktop.

In order to create parameter sweeps for the top lights and front lights, there is a file called lights.csv included in the folder, which is a comma-separated file with multiple rows and columns. The first 5 columns of this file represent the colors associated with each of the top lights in the concert scenario. Column one shows the colors for Light1; column 2 shows the colors for Light2, and so on. For ease, I've also included the intensity of lights within the same file. The intensities show float numbers that are actually being used as the value for alpha in the spinal-tap.py file. Every column after the first 5 represents the intensities of the top lights. This is not affecting the lights on the first plot as it's a different view perspective but it does show a difference from the audience's point of view. For example, column 6 shows the intensity of Light 1 while column 7 shows the intensity of Light 2, and so on. The total rows for this file are 100 as they are representing each iteration. To explain further, each row gives one new color and intensity of the light. For example, Row 1 shows the colors for each light that it will be showing in iteration 1. This row will also be showing the intensity of each light. Once read all at the same time through iterations, the colors and intensities would change, showing animation and movement.

There are other features in the spinal-tap.py file that have parameter sweeps. For example, in the class of smoke, there is a method created called plotBubbles. This is actually the bubble being created to represent one element of the complete smoke. The parameter of size and shape can be changed in order to get a different result. The shape can be turned into a triangle or any other based on the requirement. Also, the size can be changed with the market size, while the color can also be changed as preferred.

The program also allows changing the parameter sweeps for the smoke machine. As we can put the position anywhere we want depending on what x and y axis we set(recommended to put at x=0). Similarly, we can change the number of smoke particles being produced to show the same. Too many smoke particles usually crash the plot so in my program I have used two machines each producing 50 particles.

TRACEABILITY MATRIX:

For this section of the report, the feature column gives out the different additions completed to the file to make it look realistic and appealing. The second column shows the main features within the code and the reference to it that can be found in the main file (spinal-tap.py). Please find the reference code in the comments of the file to read through. In the third column, it explains what was the aim for the feature and how would it be recognized as passing the test.

FEATURE	REFERENCE	TESTING	COMPLETION
Front Lights	Class Creation = Sec02 File Reading = Sec07 List Creation = Sec06 Calling the functions = Sec17	The aim of this feature was to show lights seen from the bottom point of view. The first test was to create a separate subplot that would capture this. Then it was required to have a black background indicating it be a ceiling and lights coming down. The colors of the lights were taken from the file called lights.csv and were animated using the function called Funcanimation. Once every light was shifting on every iteration it was regarded as passing the test.	8/05/23
Top Lights	Class Creation = Sec03 File Reading = Sec07 List Creation = Sec06 Calling the functions = Sec18	Very similar to the Front lights, top lights used the same files and needed to be shifted after every iteration. However, for these lights, the intensity also needed to change as they are shown on a new plot which is the audience point of view. Once every iteration had different colors and intensities, the feature passed the test.	10/05/23

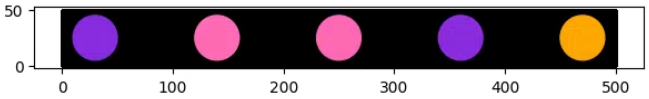
Smoke Particle	Class Creation = Sec04 Functions used = Sec22	This feature aimed to create a single smoke particle. This was done through the creation of class where the attributes of the particle like color,size shape were set. Once the particle was seen using smoke machine, it had passed the test.	18/05/23
Smoke Machine	Class Creation = Sec05 Creation of Machines = Sec10 Functions used = Sec22	The smoke machine was created with the help of Practest3. The aim was to create a machine that produces particles(bubbles) so small and many that it actually looks like its smoke. The particles also needed to change their position randomly after every iteration to pass the test.	18/05/23
Plot	Plot Creation = Sec09 Plot for front lights = Sec15 Plot for audience view = Sec16	It was required by the assignment to have at least two plots hence, subplots were used to pass the test	07/05/23
Band	Imaging = Sec11 Plotting = Sec20 Bonus Feature = Sec25	The band was originally created using shapes. (BONUS FEATURE). However, in order o make it look more appealing, I imported a png file. Once it was shown on the plot at the right position with the right size it had passed the test.	14/05/23
Backdrop	Imaging= Sec12 Plotting = Sec20	The background was aimed to be black. But, once the band worked quite nicely, I imported a background image as well to give backdrop that shows depth. Once it showed on the plot, it had passed the test.	14/05/23
Logo	Imaging = Sec13 Plotting = Sec20	The logo of the brand was also similarly imported as a png. Once it showed the right position and size, it passed the test.	14/05/23
Stage	Stage Creation = Sec19 (Bottom, top, and circles)	It was aimed to show a stage on which the band and props would be situated. The stage was created in two layers. One as a floor and the other as a wooden floor. Once it was shown on the plot, it passed the test.	10/05/23
Props	Speaker creation = Sec21	The props were created just like the stage. Once the speakers were placed correctly and the colors matched reality, the test was passed.	10/05/23
Choreography	Iterations selected = Sec08 Function creation = Sec14	This was to show the overall movement of lights and smoke machine to make it animated. Once all errors were removed,	18/05/23

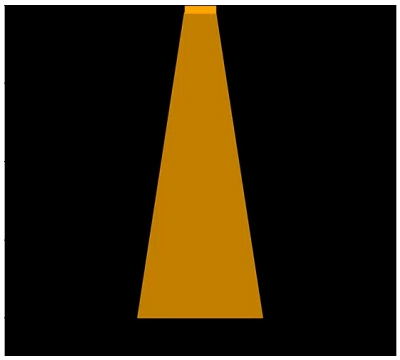
	Animation plotting = Sec24	the choreography feature passed the test.	

DISCUSSION

Front Lights and Top Lights

The first step in creating the front lights and top lights was to have a separate plot for each of them so they can show two different view perspectives. I first used the code given in the samplestage.py file and tried changing it. I understood from that what each part of the code changes and then started changing it further so that I could express my imagination of the concert. The values of the positions were changed, and some extra lights were added to make it more appealing. The size of the front lights was kept the same but for the top lights, the beams were made longer so they can come down till the stage.

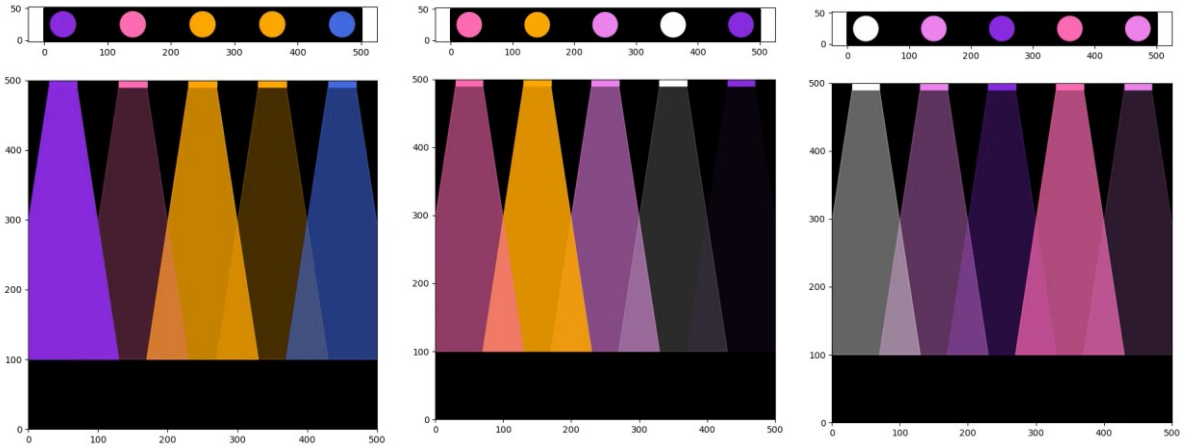




Once the structure looked good, I converted both of them into their respective classes to make the readability of the code much more easier. It included the option of changing colors. however I added the option of intensity into the class which is based on alpha values for the fill function. Then I wanted these lights to shift after every iteration hence, I looked up to the internet and was able to find a function within the matplotlib library that allows animation through iteration. (*Matplotlib.Animation.FuncAnimation Class in Python - GeeksforGeeks*, 2020). I tested it out on another file by shifting colors by giving it an IF function. Once it successfully created the movement with colors shifting, I brought it back to the main file. I then used the learning from PracTest 3 and created a csv file with different colors and intensities (see user guide).

	A	B	C	D	E	F	G	H	I	J
1	white	royalblue	violet	orange	royalblue	0.1	1	0.5	0.4	0.3
2	orange	blueviolet	royalblue	hotpink	orange	0.2	0.1	0.4	0.3	0.1
3	blueviolet	violet	hotpink	violet	white	0.3	0.4	0.7	0.2	0.4
4	hotpink	hotpink	blueviolet	orange	blueviolet	0.4	0.7	1	0.5	0.2
5	royalblue	white	hotpink	white	white	0.5	0.8	0.6	0.9	0.5
6	violet	orange	white	blueviolet	hotpink	0.6	0.9	0.2	0.1	0.6
7	white	orange	blueviolet	hotpink	violet	0.7	0.5	0.1	0.7	0.9
8	orange	hotpink	orange	royalblue	blueviolet	0.8	0.3	0.3	0.6	1
9	blueviolet	white	white	white	hotpink	0.9	0.2	0.9	0.8	0.8
10	hotpink	blueviolet	orange	blueviolet	orange	1	0.6	0.8	1	0.7
11	white	royalblue	violet	orange	royalblue	1.1	2	1.5	1.4	1.3
12	orange	blueviolet	royalblue	hotpink	orange	1.2	1.1	1.4	1.3	1.1
13	blueviolet	violet	hotpink	violet	white	1.3	1.4	1.7	1.2	1.4
14	hotpink	hotpink	blueviolet	orange	blueviolet	1.4	1.7	2	1.5	1.2
15	royalblue	white	hotpink	white	white	1.5	1.8	1.6	1.9	1.5

This was used to allocate different colors to different lights, also the last 5 column shows different intensities for the lighting. The end result shows different lights, colors, and intensities in each iteration.

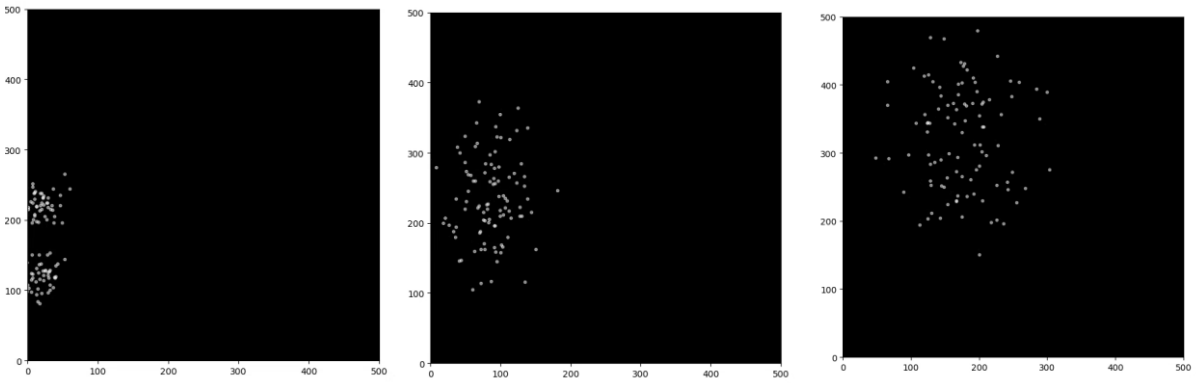


Smoke Particles and Smoke Machine

The smoke particles were created first using the learning from PracTest3. I first created a class of the particle that sets the size and color. This allowed me to use the creation of particles into the Smoke machine. The change in particle position was also included within this class by creating a method of stepChange. It actually pickups a random value using the numpy method of randint and moves the particle on x and y axis with a random value. The code snippet for Smoke Particle is below.

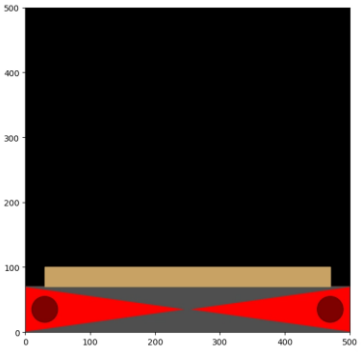
```
#Sec04# Creating a class of Smoke particle
class Smoke:
    def __init__(self, position):
        self.position = position

    def stepChange(self, row_range, col_range):
        self.position = (
            self.position[0] + np.random.randint(-(row_range/2), row_range+1),
            self.position[1] + np.random.randint(-(col_range/2), col_range+1)
        )

    def plotparticle(self, ax, alpha):
        ax.plot(self.position[0], self.position[1], 'o', markersize= 3, color= "white" , alpha=alpha)
```

Once this was created, I moved toward the creation of the smoke machine, which was once again influenced by the bubblemachine we created in Practest 3. The class was created that allowed where the machine should be positioned and how many smoke particles would be coming out. It was intended to create numerous particles to make it look more like a smoke but as the program kept crashing I had to reduce the number of particles. But to counter this, I created two machines giving out 50 smoke particles each. The output can be seen below.



Stage

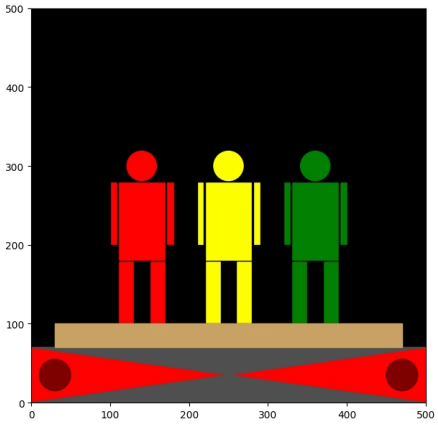
One of the most important parts of a concert is the stage. Hence, I did not import a stage from the internet but tried to make one on my own. I used the fill functions and gave it the right coordinates for x and y which allowed two to create a bottom stage that acts like a floor while another one on the top acts like a wooden stage on which the band and props would stand. I then gave them the right colors along with some detailing like making circles with with lowered alpha and also crossing lines to make it more appealing. Adding on to it I wanted to make it very unique so I added a text box that shows the number of the crowd however I used my student ID as as the counter. The end result for the stage can be seen below.



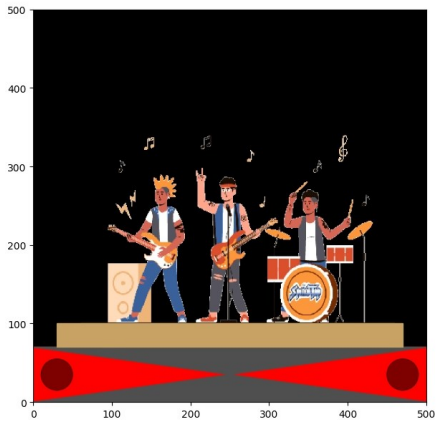
Band

I first created the band myself. It was done using the methods of fill, patch, and circle. I created the Class for them with the name of “Person” and individually created their face, torso, legs and hands. It took hours of work; however, I later realized it could look much

more appealing by importing a png. Hence, I let this stay in the main file as a bonus feature and continued towards importing a band from the internet. Below you can see the output.

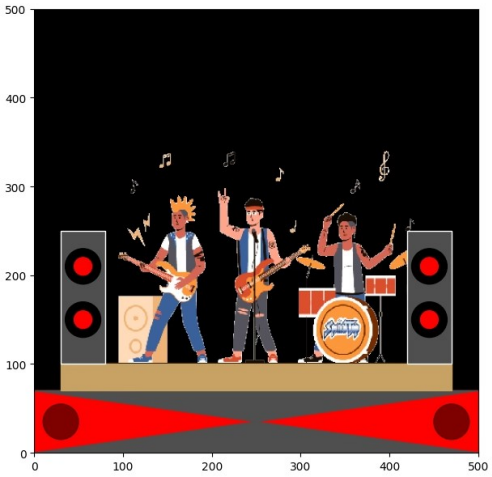


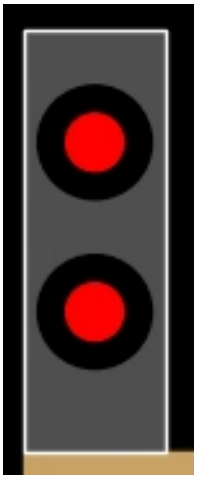
Once I decided to import png I went up to the internet, downloaded an image of a bad and imported it into Photoshop to attach a logo of the spinal-tap. This was then brought into my main code using the library of PIL and also imshow.



Props

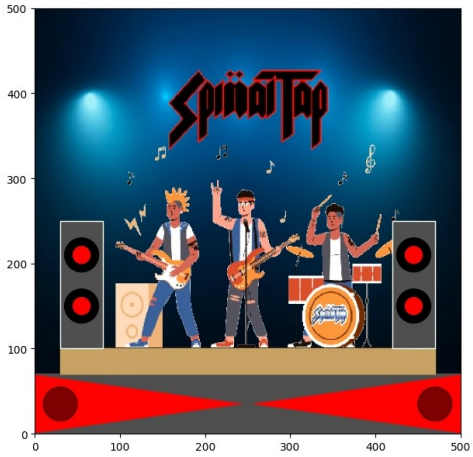
For props, I went a step ahead and did not import a png of speakers but tried to create each of them myself. Once again, with the use of circle, rectangle, patch, and fill functions I was able to add speakers into the concert scenario. I then played around with the colors to give them the best look so they actually do look like speakers.





Backdrop and Logo

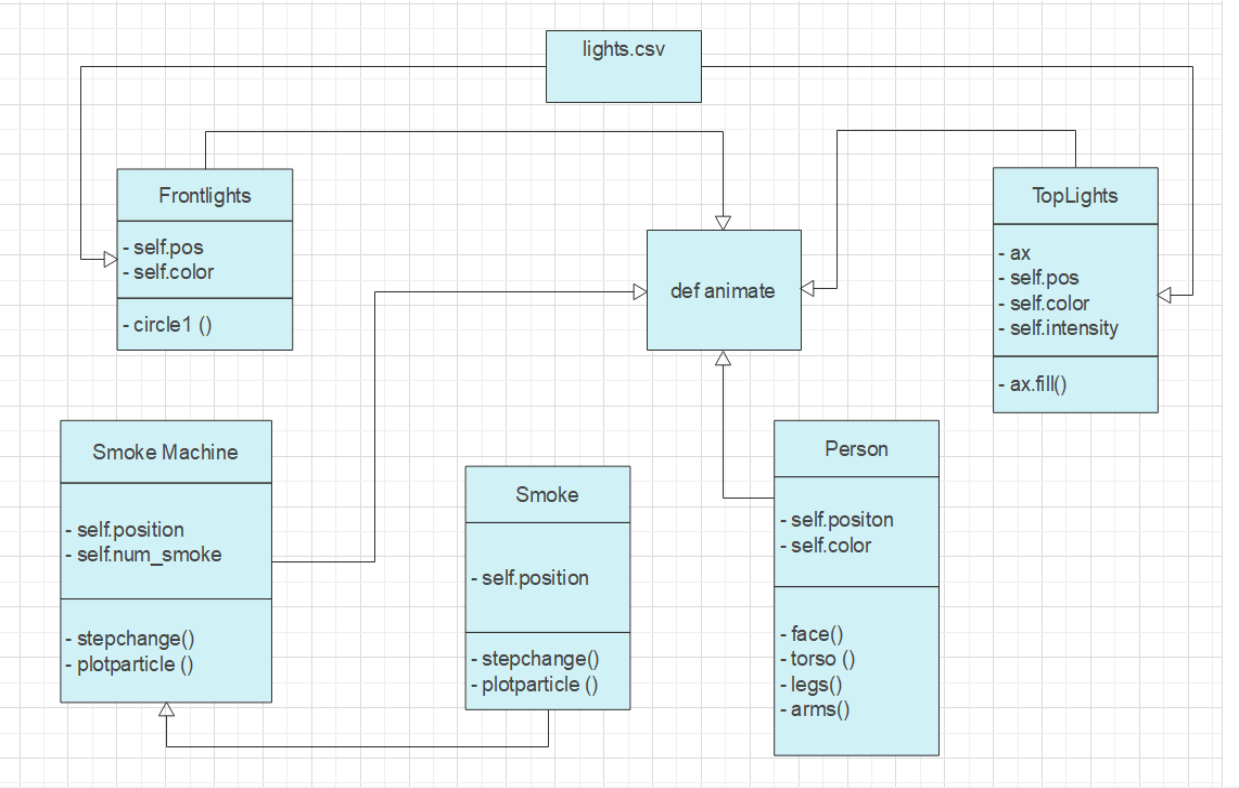
For this particular part I first stayed with the actual back face color however, when experimenting with the same use of PIL and imshow, I tried to attach a background image at which changed the whole scenario and added a lot more depth. With this, I also added the Logo of the band so it looks like its floating above the band in the air adding more colors and attraction to the scene. The output with both of them looked like this.



Choreography

For choreography and animation, I had to jump onto the internet to see if there's any way the whole concert could be shown through iteration but in an animated way. I successfully found a function within matplotlib called funcanimation. This allowed me to use as many iterations as I require and create a moving scene out of it. Through this I was able to animate the lighting and the smoke machines which as a whole completes the concert scenario scene. The snippets of the final product will be shown in the showcase section of the report.

UML Diagram



SHOWCASE

Scenario 1

In scenario 1, I've kept the same code discussed in the above report. This shows a concert of Spinal-tap taking place in the night with blue lights in the background. The toplights are all kept colorful to reflect an understanding of how a csv file is read. The lights change colors and intensity at every iteration. The smoke machine produces small particles with a markersize of 3 and 0.5 alpha. They also have a the color of white to make them prominent. There are two smoke machines both set at x = 0 but the y values defer by 100. They are set at random so each particles adds an extra x and y axis at of random number. The band was picked up from a google picture then later transferred to Photoshop to get a spinal-tap logo edited on the drumset. The stage was created with a red and grey background that reflects the colors of the logo. There is a text box added on the stage saying giving the number of people in the crowd. I have used my Student id to show the uniqueness of my creation. Lastly, a big logo of the band is also placed at the top to make it more appealing and engaging.

Scenario 2

In Scenario 2, I've made a separate folder that has all the files, particularly for this scenario. In this, I've tried showing a Spinal-Tap concert in the morning. The background is taken from Google in the jpg format. This file shows clouds and a form of sunrise. The overall theme is set to blue, violet and white. It is reading these colors from a new CSV file called lights2.csv which only has a combination of blue and purple colors. The same file is also giving the intensities for the file in every iteration. Apart from that, the smoke machine is producing smoke particles that are a lot more bigger than the first scenario. The number of particles was reduced from 50 to 30 as with too many particles, the program gets very slow. The stage color was also changed to blue, while the speakers were left unchanged.

Scenario 3

In this scenario, another folder was created with only the files used for scenario 3. The concert of spinal-tap is taking place in a night was lots of golden and yellow lighting. The top lights and front lights are customized to reflect the theme. They are taking values from a new file called lights3.csv. The code made further changes to the smoke machine with alpha values increased so that the smoke particles were a lot more evident. The size of the particles is also increased further to 15, and the quantity is also changed to 50 for more depth in the particles. The stage colors are now set according to the theme of yellow. Everything is set bright so that it looks more engaging and animated. The text at the bottom now says, “Welcome to the concert, SPINAL-TAP”. The background logo was also changed from black and red to white and blue.

CONCLUSION

This overall assignment has been a great way to understand everything that was taught in the class. When we were doing practicals, we used to edit most of the code that was pre-written, which was helpful but not as much as this one. At the start, this assignment seemed to look impossible to complete, and was very difficult to interpret what was required. However, as time passed, one feature after another allowed to build commands on creating codes from the scratch and the end result show a lot of improvement. Throughout the assignment, trial and error, even though it was frustrating at times, allowed learning multiple concepts that will help me throughout the course.

FUTURE WORK

In the future, if I were given a chance to improve this code, I would try working on each iteration. This would want to make it look like a completely smooth animation that has something new happening in every couple of frames. For example, I would first like to add in some rock music from spinal tap to make it unique and engaging. Then I could try to add in more props, maybe some drones that are giving light on the stage and moving at the same time. My band could also be replaced with the persons I made in the bonus features so that I can make every part of their body move in every iteration.

REFERENCES

Animations and Movies — Python Numerical Methods. (n.d.). Animations and Movies — Python Numerical Methods. https://pythonnumericalmethods.berkeley.edu/notebooks/chapter12.04-Animations-and-Movies.html#:~:text=You%20can%20create%20animations%20in,the%20animation%20functionality%20is%20built.

Requirements Traceability Matrix — Everything You Need to Know | Perforce Software. (n.d.). Perforce Software. https://www.perforce.com/resources/alm/requirements-traceability-matrix

All You Need to Know About UML Diagrams: Types and 5+ Examples. (2018, February 22). Tallyfy. https://tallyfy.com/uml-diagram/

Python, R. (n.d.). *Image Processing With the Python Pillow Library – Real Python*. Image Processing With the Python Pillow Library – Real Python. https://realpython.com/image-processing-with-the-python-pillow-library/

Matplotlib.animation.FuncAnimation class in Python - GeeksforGeeks. (2020, April 5). GeeksforGeeks. https://www.geeksforgeeks.org/matplotlib-animation-funcanimation-class-in-python/