1. Overview (2 marks) describe your program's purpose and implemented features.

The program simulates a stage that has moving lights, moving smoke machines, moving images(called objects) and a background. The light interacts with the smoke, objects and background.

The program works through generating and combining screens. A screen is a 2d array of values or another array. The first two dimensions represent the horizontal and vertical position. It then contains information about that point on the screen, such as the intensity of smoke or the colour of light present there.

2. User Guide (2 marks) how to use your simulation

Dependencies:

* matplotlib 3.7.1
* numpy 1.24.3
* opencv-python(also known as cv2) 4.7.0.72
* pandas 2.0.1

The program is controlled entirely by an excel file with several sheets. There must be a sheet labeled “init” that sets up each element of the simulation. To create something, add the type to the “Type” column, give it a name, then fill out the initial conditions. The types don’t have to be grouped but it is more legible when they are. Any row without a type won’t be read. Image Location and Background Image Location should be string paths to the relevant image.

For each element of the simulation except the stage there must be a sheet that’s named exactly what’s written as the name of the element. This sheet controls what the object does in the simulation. The first column is the instruction. It must be accompanied by an end time, which is the time that the instruction ends). Each instruction will run at least once.

The available instructions are:

* Move To: Will move to a specific state, reaching it when end time is reached. All columns except loop index must be filled out. Everything except color is smoothly interpolated. To move instantly set an end time of 0
* Loop To: Go to and execute a specific line. The internal time will be set to the endtime of the previous instruction(0 if first instruction). The loop index is two less than the excel row. End time is irrelevant
* Hold: The element will not change until the end time is elapsed.
* Stop: The element will not change again.
* End: Ends the entire program

To run the file, either rename it to be “Choreography.xlsx” and put it in the same folder as the program, or edit the program to change choreographyLocation to be the path of the new choreography file.

3. Traceability Matrix (10 marks) of features, implementation and testing of your code. The matrix should be a table with columns for: i. Feature - numbered for easy referencing ii. Code reference(s) – reference to files/classes/methods or snippets of code only, do not put the whole program in the report iii. Test reference(s) – test code or describe how you tested your feature was correctly implemented iv. Completion date - N/A if not implemented

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Feature | Code Reference | Testing | Completion Date |
| 1 | Light object generates light screen | Light.getNewLightScreen() | Used plt.im\_show() to display lightscreen for a few manually entered lights |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

4. Discussion (10 marks) of implemented features (referring to the Traceability Matrix), explaining how they work and how you implemented them. UML Class Diagrams should be included for objects and their relationships.

5. Showcase (10 marks) of code output, including three different scenarios: a. Introduction: Describe how you have chosen to set up and compare the simulations for the showcase. Include commands, input files – anything needed to reproduce your results. b. Discussion: Show and discuss the outputs/results of each scenario.

6. Conclusion (2 marks) reflection on your assignment with respect to the specification

7. Future Work (2 marks) further investigations and/or extensions that could follow.

Currently the laser width is based on its horizontal width. This means as the laser becomes closer to horizontal, it looks thinner. This could be resolved by dynamically setting the horizontal width of the laser based on the desired width and current angle.

Theoretically lights could have a width and a spread angle, this isn’t tested. It would likely work in the main view, but the light view would likely display incorrectly. This could be resolved with a few additional checks to determine if its needed and how to do it.

Movement currently has no acceleration, so is very jerky. Smoothing could be added to look more natural.

Currently the file to read the choreography from is hard coded into the program, it could be entered as a command line argument or a text input.

Currently only supports png inputs, could be made to support other formats of images.

Lights can’t fade between colours, can’t be told to follow another light, can only be given one color at a time and only display as a cone. All of these could be implemented.

The lights, smoke machine and objects could all inherit from a more general object that contains things like the position and much of the choreography are the same across each of them so should be put into a parent class.

8. References (2 marks)