# COMP5005 ASSIGNMENT

Semester 1, 2023

*Zhao Xin (Charles) Ong*

*21450673*

## Table of Contents

## Overview

The purpose of the “spinal-tap.py” program is to produce a simulation of a concert stage for the band “Spinal Tap”. This simulation includes two plots – one that represent a view of the lights from above the stage and another that represents a view of the stage from the perspective of the audience. Within these plots, several features can be added including lights, smoke machines, props and a backdrop. All features can be pre-set using a csv file, which allow for the “choreography” of the stage to be customised.

## User Guide

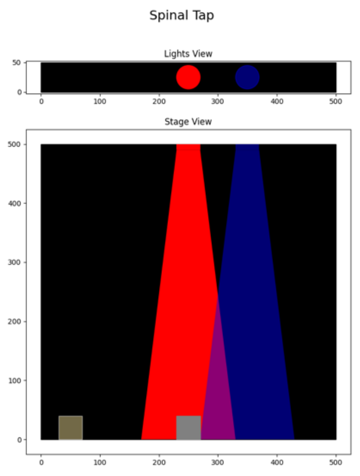
Before running the program, the following python packages need to be installed:

```
pip install matplotlib
pip install numpy
```

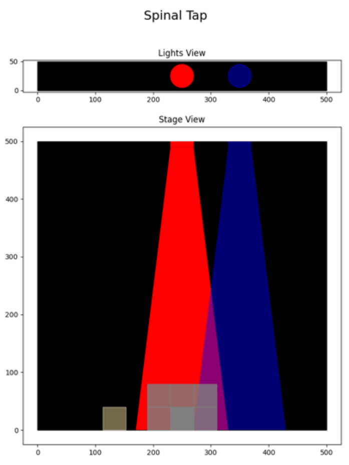
The program can be run by running the following command into the terminal:

```
python3 spinal-tap.py
```

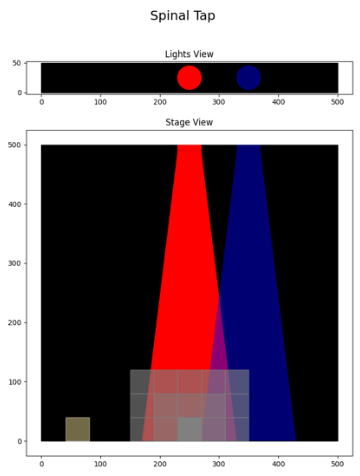
This will run the program with hard-coded features, including a red light, blue light, smoke machine, prop object and a black backdrop. Below is what the program should look like after it is run:



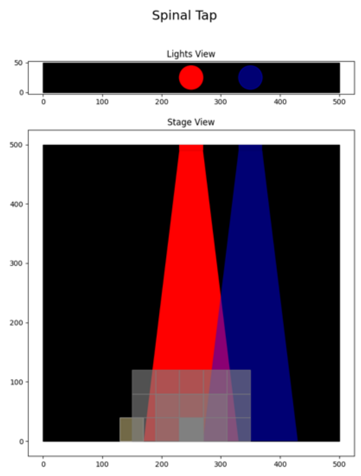
*Frame 0*



*Frame 1*



*Frame 2*

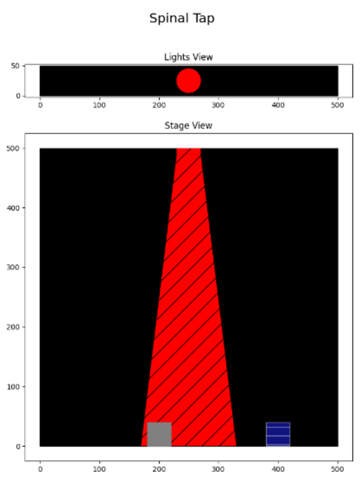


*Frame 3*

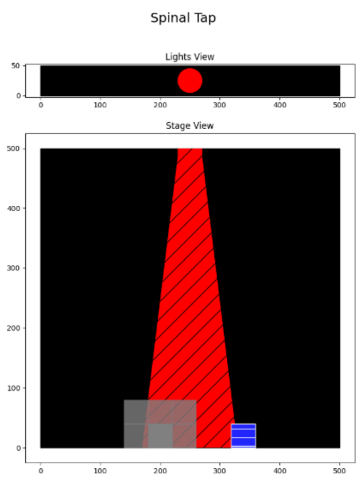
Alternatively, a csv file containing the choreography for the stage can be provided as a command line argument, like the example below:

```
python spinal-tap.py choreography1.csv
```

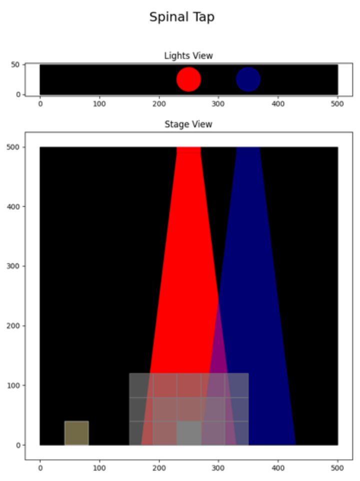
This example csv file produces the following results:



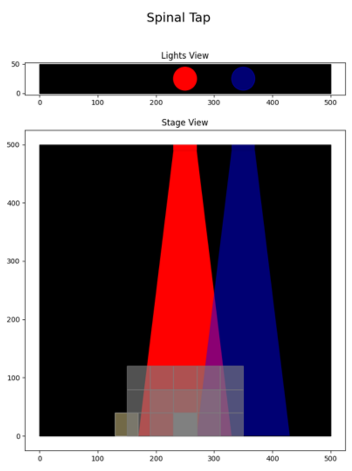
*Frame 0*



*Frame 1*



*Frame 2*



*Frame 3*

The following table shows the contents of choreography1.csv file:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Frames | 15 | | | | | | |
| Direction | left | | | | | | |
| | | | | | | | |
| | COLOUR | | | | | | |
| Backdrop | Black | | | | | | |
| | | | | | | | |
| | POSITION | COLOUR(S) | DIRECTION | INTENSITY | AMOUNT | TYPE | |
| Light | 250 | red | down | 11 | 1 | | PATTERN ('/', '\', '\|', '-', '+', 'x', 'o', 'O', '.', '*') |
| | | | | | | | / |
| | POS | DIRECTION | INTENSITY | NEIGHBOUR | | | |
| Smoke | 200 | up | 11 | m | | | |

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
|  | POS | SHAPE | MOVE | PATTERN ('/', '\', '\|', '-', '+', 'x', 'o', 'O', '.', '*') |  |  |  |
| Prop | 400 | square | y | - |  |  |  |

In order to create a file that can be read by the program, there must be labels written on the leftmost column of the csv file. There are two types of labels: variable initialisation labels and object labels.

The two variable initialisation variables are “Frames”, “Direction” and “Backdrop”, which tell the program how many frames the simulation will run, what direction props will try to move towards; and what colour the backdrop should be.

Object labels tell the program what objects need to be created and what their attributes need to be initialised to. As objects have multiple attributes, there are headers describing what each column’s attribute is. These headers are optional and do not affect the code.

Additionally, saving of the images produced in each frame of the program can be done with the following command line argument:

```
python spinal-tap.py -s
```

or

```
python spinal-tap.py choreography1.csv -s
```

When this is run, there will be a folder created in the same directory as the program, which will be used as the save location of the images made by the program. This folder will be named depending on the time and date (to the second) of the user’s device at the time the program is run. As can be seen above, this feature is not limited to the default running of the program, but can also be used in combination with the command line argument for using a custom choreography file. However, the name of the folder created by the program will include not only the time date, but also the name of the choreography file as well.

## Traceability Matrix

| Feature | Code | Test | Date |
|---|---|---|---|
| **Lights** |  |  |  |
| Lights know their colour | Class Light: self.colour in __init__ function | PASSED: Creates and displays Light object with an assigned colour. | 08/05 |
| Lights know their position | Class Light: self.pos in __init__ function | PASSED: Creates and | 08/05 |

| | | | |
|---|---|---|---|
| | | displays Light object with an assigned position. | |
| Lights know their direction | Class Light: self.direc in __init__ function | PASSED: Creates and displays Light object with an assigned direction. | 08/05 |
| Lights know their intensity | Class Light: self.inten in __init__ function | PASSED: Creates and displays Light object with an assigned intensity. | 08/05 |
| Lights can be grouped to work as synchronised sets | In createLightSet() function:<br>for i in range(no_lights):<br>        light_set.append(Light(int(line[1])+offset, colours[i], line[3], int(line[4]), line[6], line[7]))<br>        offset += 120 | PASSED: Creates and displays a set of Light objects. | 18/05 |
| Lights can be solid colours | Class Light:<br>    self.colour in __init__ function<br><br>    in plotLight() function:<br>    circle1 = plt.Circle([self.pos,25],self.radius,<br>        facecolor=self.colour, alpha=self.inten,<br>        hatch=self.hatch)<br>    ax.add_patch(circle1) | PASSED: Creates and displays a Light object using the assigned solid colour. | 18/05 |
| Lights can be combinations of colours | In createLightSet() function:<br>colours = line[2].split(" ")<br><br>for i in range(no_lights):<br>        light_set.append(Light(int(line[1])+offset,<br>            colours[i], line[3], int(line[4]), line[6],<br>            line[7]))<br>        offset += 120 | PASSED: Creates and displays a light set that can have a variation of colours. | 18/05 |
| Lights can be patterns/shapes | Class Light: self.hatch in __init__ function | PASSED: Creates and displays Light object with an assigned pattern. | 18/05 |

| | | | |
|---|---|---|---|
| Lights can be lasers | Class Light: self.type in __init__ function | PASSED: Creates and displays Light object with an assigned type of either a spotlight (default) or laser. | 18/05 |
| **Smoke Machine** | | | |
| Smokes have a position | Class SmokeMachine: self.pos in __init__ function | PASSED: Creates and displays Smoke Machine object with an assigned position. | 08/05 |
| Smokes have a direction | Class SmokeMachine: self.direc in __init__ function | PASSED: Creates and displays Smoke Machine object with an assigned direction. | 08/05 |
| Smokes have an intensity | Class SmokeMachine: self.inten in __init__ function | PASSED: Creates and displays Smoke Machine object with an assigned intensity. | 08/05 |
| Smokes can diffuse using a choice of Moore or Von-Neumann neighbourhoods | Class SmokeMachine: self.neigh in __init__ function | PASSED: Creates and displays Smoke Machine object with an assigned diffusion type. | 15/05 |
| Smokes last for a specific | Class SmokeMachine: self.max_steps in __init__ function | PASSED: Displays | 15/05 |

| | | | |
|---|---|---|---|
| number of timesteps | | Smoke Machine object for the duration of the timesteps defined. | |
| **Props/Band** | | | |
| Props have a position | Class Prop: self.pos in __init__ function | PASSED: Creates and displays Prop object with an assigned position. | 08/05 |
| Props have a shape | Class Prop: self.shape in __init__ function<br>  In plotProp() function:<br>    if self.shape.lower() == "circle":<br>    elif self.shape.lower() == "triangle":<br>    elif self.shape.lower() == "square":<br>    else: | PASSED: Creates and displays Prop object with an assigned shape. | 08/05 |
| Props can be stationary or moving | Class Prop: self.move in __init__ function<br><br>  In animate() function:<br>    if self.move == "y":<br>      if direction == "left":<br>        step = random.randrange(-100,0)<br>      elif direction == "right":<br>        step = random.randrange(0,100)<br>      else:<br>        step = random.randrange(-100,100) | PASSED: Creates and displays Prop object with an assigned movement type (y/n). | 19/05 |
| Props can be impacted by the lighting | Class Prop: self.inten in __init__ function<br><br>In setLighting() function in Class Prop:<br>for range in ranges:<br>      if (self.pos-20 > range[0] and self.pos-20 < range[1]) or  (self.pos+20 > range[0] and self.pos+20 < range[1]):<br>        new_inten = self.inten + range[2]<br>        if new_inten > 1:<br>          new_inten = 1<br>        self.inten = new_inten<br>      else:<br>        self.inten = 0.5<br><br># iff prop is within the range of the light beams, increase intensity with light's intensity (until it reaches the maximum intensity of alpha=1) | PASSED: Recalculates and reassigns intensity of prop objects depending on whether they are within the range of the light beams. | 20/05 |

| | | | |
|---|---|---|---|
| **Backdrop** | | | |
| Backdrops are default to black | In main() function:<br>Backdrop.append("black") | PASSED: Backdrops are black by default. | 08/05 |
| Backdrops' colours can be determined by a file | In main() function:<br>Backdrop.append(splitline[1])<br>setBackdrop(ax0, SIZE, SMALL_HEIGHT, Backdrop[0])<br>setBackdrop(ax1, SIZE, SIZE, Backdrop[0])<br><br>In setBackdrop() function:<br>ax.set_aspect("equal")<br>ax.fill([0,width,width,0],[0,0,height,height], color=col) | PASSED: Backdrop colours can be assigned by file. | 13/05 |
| Stage sizes are pre-defined | Global variables:<br>SIZE = 500<br>SMALL_HEIGHT = 50<br><br>In main() function:<br>setBackdrop(ax0, SIZE, SMALL_HEIGHT, Backdrop[0])<br>setBackdrop(ax1, SIZE, SIZE, Backdrop[0]) | PASSED: Stage sizes are pre-defined to a specific size. | 08/05 |
| **Choreography** | | | |
| Allows for pre-set patterns of lighting, by defining the required lights, smoke machine, props/bands and backdrop | In main() function:<br>elif splitline[0].lower() == "backdrop":<br>Backdrop.append(splitline[1])<br><br>elif splitline[0].lower() == "light":<br># One Light<br>if int(splitline[5]) < 2 or splitline[5] == "":<br>Lights.append(Light(int(splitline[1]), splitline[2], splitline[3], int(splitline[4]), splitline[6], splitline[7]))<br># Light Set<br>else:<br>Lights.extend(createLightSet(splitline))<br><br>elif splitline[0].lower() == "smoke":<br>Smokes.append(SmokeMachine(int(splitline[1]), splitline[2], int(splitline[3]), splitline[4]))<br><br>elif splitline[0].lower() == "prop":<br>Props.append(Prop(int(splitline[1]), splitline[2], splitline[3], splitline[4])) | PASSED: Adds light, smoke machine and prop objects; movements and backdrop colours depending on the contents of the file. | 13/05 |

| | | | |
|---|---|---|---|
| Allows for pre-set stage movements | In main() function: DIRECTION = splitline[1] | PASSED: Contents in the file determines how appropriate objects (props and smokes) should move on the stage. | 20/05 |
| Uses a specific file format | In main() function: elif ".csv" in i: FILE = True elif "." in i: print("WARNING, UNKNOWN FILE, '" + i + "' DETECTED IN COMMAND LINE ARGUMENTS. PLEASE USE .csv FILE IF YOU ARE INTENDING TO USE A PRE-SET CHOREOGAPHY") In getChoreography() function: try: file = open(file_name, "r") return(file.readlines()) except OSError as err: print("Error with opening file: ", err) except: print("Unexpected error: ", err) | PASSED: Warns the user if the file provided in the command line arguments are not of "csv" format or cannot be opened. The default pre-defined stage will be executed if warnings are present. | 13/05 |

## Discussion

### Light Objects

All light objects in this program have parameters of pos (position), colour, direc (direction), inten (intensity), type, range and hatch (pattern).

The position of the light is the x coordinate of the light's midpoint. This position value is checked by the checkPos() function, which warns the user if the position of the light may be outside of the stage's boundaries.

The colour of the light requires colour formats recognised by matplotlib, such as RBA float, hex RGB, RGB string, float value between 0 and 1; single character shorthand notation and named colours.

The direction of the light is the direction of the light's beam and should not be confused with the global direction provided by choreography files. This direction can be set to any string value, but will only affect the light if it is of the values "left" or "right". These will change the "skew" attribute of the light, which is used to calculate the coordinates of the light's beam.

The light's beam coordinates are also checked using the checkPos() function for the potential of plotting outside of the stage's boundaries.

The inten (intensity) attribute  of the light determines the brightness of the light. The method used to determine brightness is the "alpha" parameter in matplotlib. As only float values between 0 and 1 can be passed to this parameter, the intensity range of 1-11 (as provided by the project description) must be converted to a value between 0 and 1. This has been calculated using the formula below.

$$alpha = inten \times \frac{1}{11}$$

After calculation, a check of whether alpha is within the required range between 0 and 1 is conducted using the checkInten() function, which returns the value if it is within the range; or 1 if it is not.

The type of the Light can also be any value, but will affect the light's beam if the value is of type "laser". As the name suggests, this will cause the light's beam to be equal in width from the source to the end of the light's beam. The range of the light is the coordinates of the light's beams, which are used in the calculation of the lighting of the Prop objects on the stage.

Finally, the hatch (pattern) attribute determines whether the light will have a pattern. The attribute may be empty to represent a solid colour or a value that is recognised by matplotlib's "hatch" parameter, such as the following: '/', '\', '|', '-', '+', 'x', 'o', 'O', '.', '*'. In order to check whether the hatch provided by the user is a valid hatch, the checkHatch() function is called after initialisation and will reassign the hatch attribute if the hatch value is not valid.

## Light Sets

Light sets are not separate objects as of themselves, but rather light objects added the light array using a choreography file and the createLightSet() function. In fact, single lights are created in the same way light sets are created, with only the value of the "AMOUNT" column (in the choreography file) set to below 2 or empty.

If the "AMOUNT" column is not below 2 or empty, the createLightSet() function will be called and multiple light objects will be created using details specified by the row in the choreography file. The "synchronised" trait specified in the project description has been interpreted to mean that all light objects created by the createLightSet() function will have the same direction, intensity, type and pattern. However, the position of each light is calculated depending on an offset that allows for all lights to fit within the stage.

Additionally, colours of light sets fulfill the criteria in the project description where lights may be a "combination of colours". This is because the user can provide multiple colours separated by spaces in the "COLOUR" column, which will be converted to a list for colours of the lights in the light set to be set iteratively.

## Smoke Machine

All smoke machine objects have parameters of pos (position), direc (direction), inten (intensity) and neigh (diffusion type).

The position attribute is similar to the light object and is also checked using the checkPos() function for validity.

The direction attribute should not be confused with the direction of the global direction specified by the choreography file. Instead, it represents whether the smoke machine should be placed on the floor of the stage and diffuse “up” (upwards) or placed on the ceiling and diffuse “down” (downwards). If the value provided for direction is empty or not recognised, the value will default to “up”.

The intensity attribute is also similar to the light object, as it is also calculated by the “alpha” parameter from matplotlib. Therefore, the formula shown above in the light object section of the discussion was also used for the calculation of the smoke machine’s intensity. However, it should be noted that there is also a “fade” variable in the plotSmoke() function, which is used to gradually reduce the intensity of the smoke as the timestep increases.

The diffusion type determines the neighbourhood algorithm that will be used for the smoke machine object. There are two options recognised – “m” for Moore neighbourhood diffusion and “vn” for Von-Neumann neighbourhood diffusion. Additionally, each neighbouring smoke “block” created will have a slightly lower intensity than the previous (for better realism). The algorithm used to diffuse the smoke in the stepChange() function. The algorithm is an original implementation created by analysing the changes in indexes, the location of neighbours and the number of neighbours at every iteration using each diffusion type. The calculations can be found in the file “Smoke Calculations.xlsx” in the root folder of the project.

Additionally, smokes are also affected by the global direction specified by the choreography file. All smoke “blocks” will shift left or right by its own width depending on whether the global direction is “left” or “right”, respectively.

## Props

All prop objects have the parameters pos (position), shape, move (movement) and hatch (pattern).

The position attribute is implemented in the same way as the lights and smoke machines; and are also checked using the checkPos() function.

The shape attribute determines the shape of the prop and can either be a “circle”, “triangle” or “square”. If an unrecognisable shape is provided, then the default shape of “square” will be used instead. Each shape has a white border for better clarity for the viewers.

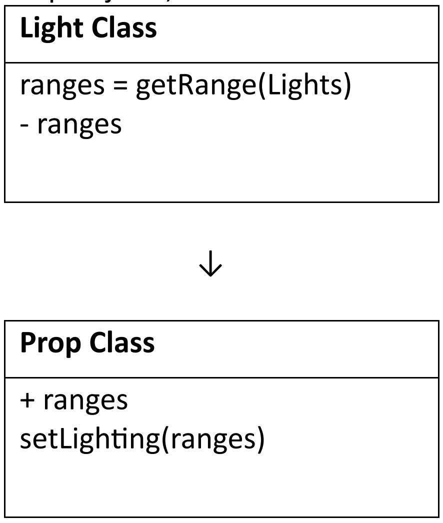
The movement attribute determines whether the prop will move or be stationary. For better readability in the choreography file, a string input of “y” and “n” can be provided (instead of Boolean 0 and 1/True and False). If an unrecognisable movement is provided, the default value of “y” will be used instead. Specific movements of the prop is determined in the animate() function. This animate function also calculates whether the prop will move left, right or either depending on the global direction variable.

The hatch attribute is implemented in exactly the same way as the hatch attribute in the light object; and is also checked using the checkHatch() function.

An attribute of prop objects that cannot be changed by the user via parameters is the inten (intensity) attribute. Instead, this is set with the value of 0.5 (alpha value, like in light and smoke machine objects) and is changed when the prop object is within the range of the light beams (i.e., shined by the lights). This is calculated at every timestep using the setLighting() function.

### Class Relationships

The only objects of this program that have relationships with other objects are Lights and Prop objects, as can be seen in the UML Class Diagram below:



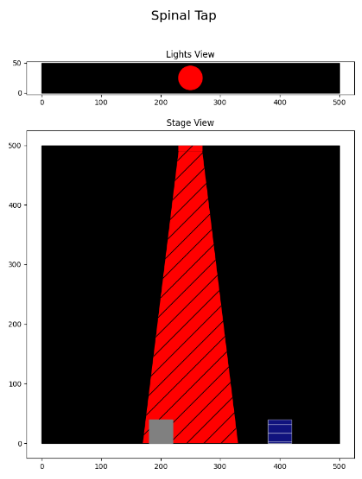
## Showcase

Besides the default scenario without a choreography file (as shown in the “User Guide” section), three other scenarios have been produced.

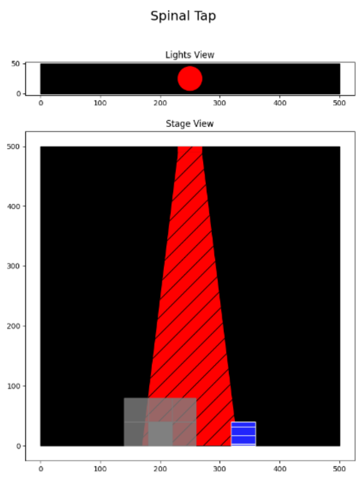
### Scenario One

The first scenario uses the file “choreograhy1.csv” and its contents have been shown previously in the [“User Guide” section](). Furthermore, frames 0 to 3 have also been shown in the same section. Below, we can see frames 0 to 1 once again. (Note that other frames can be found in the “choreography2.csv 2023-05-24 22/36/22 images” folder”

### Frames (0 to 1)



*Frame 0*



*Frame 1*

### Discussion

As can be seen, there is one light object with a position: 250, colour: “red”, direction: “down”, intensity: “11”, amount: 1, and pattern “/”. This all corresponds to the contents choreography1.csv file.

There is one smoke machine with position: 200, direction: “up”, intensity: 11 and diffusion type (neighbour): “m”. This also corresponds to the choreography1.csv file.

There is also a prop object with position: 400, shape: “square”, move: “y” and pattern “-”. Once again, this also corresponds to the choreography1.csv file. Also note that the intensity of the prop increases in frame 1, as it is within the range of the light.

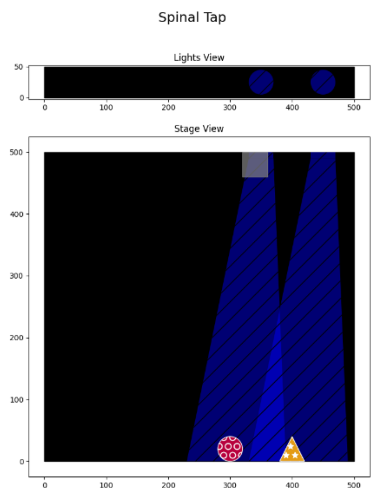
## Scenario Two

### Choreography file (choreogaphy2.csv)

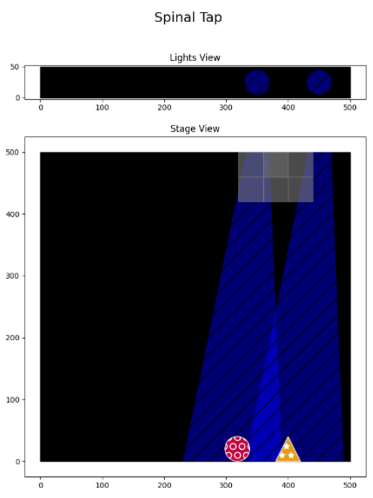
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Frames | 20 | | | | | | |
| Direction | right | | | | | | |
| Backdrop | Black | | | | | | |
| | | | | | | | |
| | POSITION | COLOUR(S) | DIRECTION | INTENSITY | AMOUNT | TYPE | PATTERN ('/', '\', '\|', '-', '+', 'x', 'o', 'O', '.', '*') |
| Light | 450 | blue | left | 5 | 2 | | / |
| | | | | | | | |
| | POS | DIRECTION | INTENSITY | NEIGHBOUR | | | |
| Smoke | 300 | down | 8 | vn | | | |
| | | | | | | | |
| | POS | SHAPE | MOVE | PATTERN ('/', '\', '\|', '-', '+', 'x', 'o', 'O', '.', '*') | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Prop | 300 | circle | y | O | | | |
| Prop | 400 | triangle | n | * | | | |

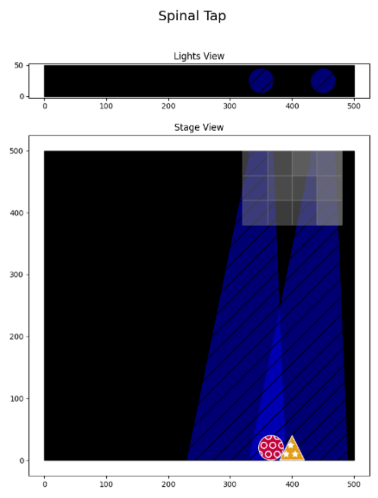
## Frames (0 to 3)



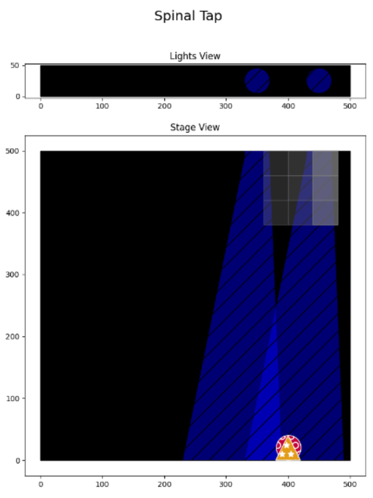
*Frame 0*



*Frame 1*



*Frame 2*



*Frame 3*

## Discussion

As can be seen, there are now two lights, which are skewed towards the left due to the “left” parameter provided. These lights are created using a the createLightSet() function, as can be seen by the “AMOUNT” column for the light object set to “2”. Also notice that even though only one colour was provided (“blue”), two lights were detected, so both were given the same colour.

The smoke machine now has a direction of “down”, which causes it to be placed on the roof and diffuse downwards.

The movement of the circle prop is set to “n”, so it does not move as the timestep increases.

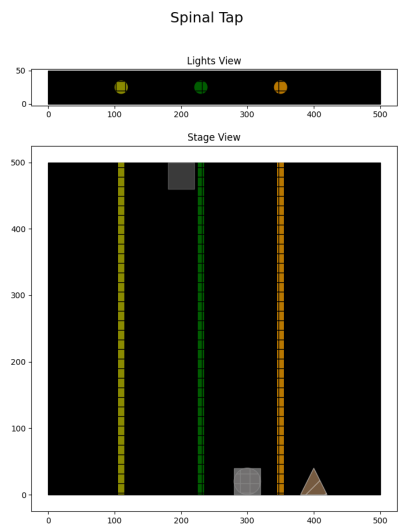
Also notice that the movement of the smoke and the triangle prop tends towards the right, due to the global movement variable set to “right”. The triangle prop does go left to the end, as the animate() function detected that it could not go right anymore.

## Scenario Three

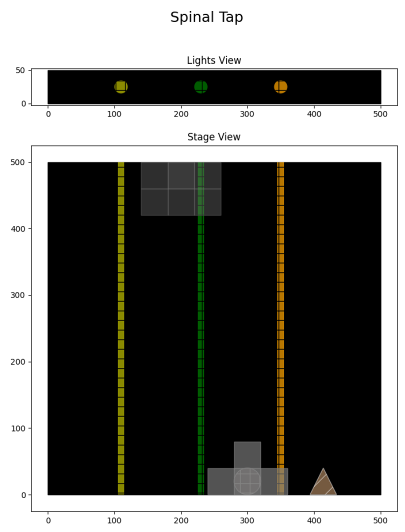
### Choreography file (choreography3.csv)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Frames | 10 | | | | | | |
| Direction | | | | | | | |
| Backdrop | Black | | | | | | |
| | | | | | | | |
| | POSITION | COLOUR(S) | DIRECTION | INTENSITY | AMOUNT | TYPE | PATTERN ('/', '\', '\|', '-', '+', 'x', 'o', 'O', '.', '*') |
| Light | 350 | y g orange purple | right | 8 | 3 | laser | + |
| | | | | | | | |
| | POS | DIRECTION | INTENSITY | NEIGHBOUR | | | |
| Smoke | 200 | down | 5 | m | | | |
| Smoke | 300 | up | 9 | vn | | | |
| | | | | | | | |
| | POS | SHAPE | MOVE | PATTERN ('/', '\', '\|', '-', '+', 'x', 'o', 'O', '.', '*') | | | |
| Prop | 300 | circle | n | + | | | |
| Prop | 400 | triangle | y | / | | | |

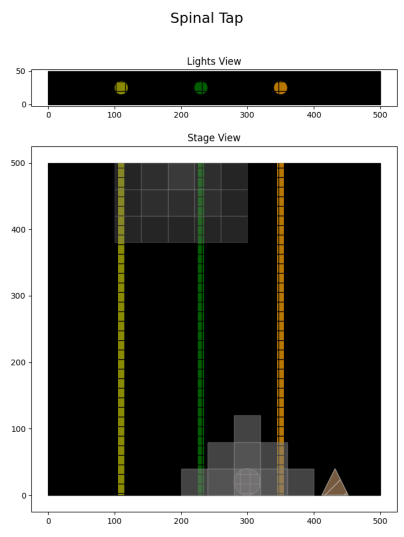
### Frames (0 to 3)



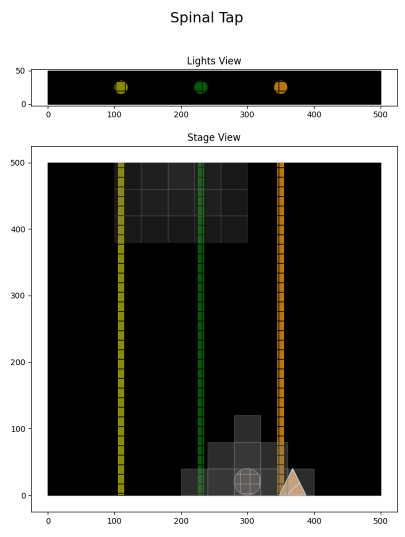
*Frame 0*



*Frame 1*



*Frame 2*



*Frame 3*

### Discussion

As can be seen, there are now 3 lights (created using createLightSet()), two smokes and two props.

The three lights have a "laser" type, so they now have a uniform width from the ceiling to the floor. Each light also has different colours depending on the different colours set (y, g and orange). The lights also have a direction, but this is overridden by the fact that they are lasers and cannot be skewed in any direction. Also notice that the lights in the "Lights View" also change in radius due to the change in type.

The two smokes are differing in direction, intensity and also diffusion type (neighbour).

The two props are similar to the second scenario where one is moving (triangle) and one is not (circle). It is important to note that the triangle goes in range of one of the lasers in frame 3, so it increases in intensity.

## Conclusion

The final program "spinal-tap.py" has incorporated the features required by the project description. The user is able to customise the stage in various ways using a choreography file in the csv format. Many additional features, such as image saving, multiple types of smoke diffusion and other types of customisations have also been implemented. Most if not all implementations have error checks, allowing for users to easily find out about any mistakes that they have made.

## Future Work

The future work of this project may include allowing for the choreography to control the exact timings of movements. Additionally, graphics can be improved by making more realistic objects and by creating a 3D view of the stage.

## References

*Matplotlib: Python plotting — Matplotlib 3.3.4 documentation*. (2023). Matplotlib.org. https://matplotlib.org/stable/index.html. Accessed 08/05/2023

Python Software Foundation. (2019). *3.7.3 Documentation*. Python.org. https://docs.python.org/3/. Accessed 13/05/2023

*datetime - Getting today's date in YYYY-MM-DD in Python?* (2015). Stack Overflow. https://stackoverflow.com/questions/32490629/getting-todays-date-in-yyyy-mm-dd-in-python. Accessed 22/05/2023.