Fundamentals of Programming

COMP1005 Assignment

Semester 2, 2022 v1.0

Discipline of Computing
Curtin University

1 Preamble

In practicals you have implemented and learned about simulations, object-orientation and (soon) how to automate the running of multiple simulations. In this assignment, you will be making use of this knowledge to extend a given simulation to provide more functionality, complexity and allow automation. You will then report on the results generated by the simulation.

2 The Problem

You will be simulating a concert stage for mythical rock band Spinal Tap. The stage will consist of **lights; smoke machine; props/band; backdrop and choreography**. We should have at least two plots – one of the lights alone and one looking at the stage from the audience. The model can be assumed flat/2-D – there are bonus marks for 3D, but not required.

We will provide some sample code to start this assignment, and additional code showing a range of approaches to assignments from previous semesters. For the assignment, you will develop code to model the lights using objects, and to add features to the simulation (e.g. more functionality, props, graphics). Your task is to extend the code and then showcase your simulation, varying the input parameters, to show how they impact the overall simulation.

Note: You do not have to use the supplied sample code, however, any other code that you have not written (e.g. sourced from others, online or generated etc.) will not receive marks. Lecture/practical and test materials from COMP1005/5005 are exempt, but must be referenced.

The required extensions are:

1. **Lights:** Represented as objects that "know" their colour, position, direction and intensity (0-11). We are mainly focussed on down-lighting, Lights can be grouped to work as synchronised sets. Lights may be solid colours, combinations of colours, patterns/shapes (gobo stencils) or lasers.
   Prompts: *How will you represent the "spread" of the light – the cone it shines through? How will you group lights and provide a mechanism to drive them as a collection? What types of lights will you include?*
2. **Smoke machine:** There should be one or more smoke machine objects, which should include position, direction, intensity (0-11). Smoke should diffuse using a choice of Moore or Van-Neumann neighbourhoods.

   Prompts: *How will the light interact with the smoke? What forces will apply to the movement of the smoke? How will the neighbourhood options affect your simulation? How will the timesteps for the smoke simulation sit within the overall program?*

3. **Props/Band:** The props/band should be objects with a position and shape, and may be stationary or moving. You should decide how they impact the lighting on stage
   Prompts: *How will the props/band be represented in the simulation? Will the stage view become back/white/coloured or have a halo around the props/band?*
4. **Backdrop:** The backdrop for the stage defaults to black or is read from a file. You need to decide a suitable file format and whether it defines the stage size, or is scaled to the pre-defined stage size.
   Prompts: *Will the backdrop be colour or monochrome (clack and white)? How will it be affected by the lights?*
5. **Choreography:** This allows a pre-set pattern of lighting and stage movement to be read in from a file. The file will need to define the required lights, smoke machine, props/bands and backdrop, then control these with references/commands.
   Prompts: *How will you reference the various items on/around the stage? Will there be an overall time control to allow timing of the movements/changes?*

Note: Parts 1-4 can be implemented and tested (mostly) independently of each other.

You can begin with hard-coded values and filenames, but should move to prompting for values, or a better approach is to use command line arguments to control the parameters of the experiment/simulation.

Your code should include comments to explain what each section does and how. This will affect your readability score in our marking.

It is useful to keep track of your changes in the comments at the top of the program. Feel free to re-use the code and approaches from the lectures and practicals. **However, remember to cite/self-cite your sources.** If you submit work that you have already submitted for a previous assessment (in this unit or any other) you must specifically state this.

Beyond the working program, you will submit a document: the **Project Report**, worth 40% of the assignment marks. This is described later in the specification.

There will be **bonus marks** for additional functionality and the use of more advanced programming techniques (e.g. interactivity, high quality visualisation, 3D space, parameter sweep etc.) but only if they are sensible and done well. Make sure to discuss the additional work in your Report, this will be easy if you make notes and keep old (incremental) versions of your code.

Remember : Think before you code!

You can do a lot of the assignment planning on paper before any coding.

3 Submission

Submit electronically via Blackboard. You can submit multiple times – we will only mark the last attempt. This can save you from disasters! Take care not to submit your last version late though. Read the submission instructions very carefully.

You should submit a single file, which should be zipped (.zip). Check that you can decompress it successfully. The submission file must be named FOP_Assignment_<id> where the <id> is replaced by your student id. There should be no spaces in the file name; use underscores as shown.

The file must contain the following:

- **Code – spinal-tap.py** and supporting files, i.e. all files needed to run your program, including input files.
- **README** file including short descriptions of all files and **dependencies**, and information on how to run the program.
- **Report** for your code, as described in Section 3.1.
- **Cover Sheet** - signed and dated. These are available on Blackboard. You can sign a hard copy and scan it in or you can fill in a soft copy and digitally sign it.
- **You will also need to submit the Report to TurnItIn.**

Make sure that your zip file contains what is required. Anything not included in your submission will not be marked, even if you attempt to provide it later. It is your responsibility to make sure that your submission is complete and correct – submitted as a **single zip file**.

3.1 Project Report

You need to submit your Report in Word **doc or pdf** format. You will need to describe how you approached the implementation of the simulation, and explain to users how to run the program. You will then showcase the application(s) you have developed, and use them to explore the simulation outputs. This exploration would include changing parameters, simulation time and perhaps comparing outcomes if you switch various features on/off.

THE REPORT MUST BE SUBMITTED THROUGH TURNITIN AND IN THE ZIP FILE

Your **Project Report** will be around 10 pages and should include the following:

1. **Overview** (2 marks) describe your program's purpose and implemented features.
2. **User Guide** (2 marks) how to use your simulation (and parameter sweep code, if applicable)
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
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.
8. **References** (2 marks)

3.2 Marking

Marks will be awarded to your submission as follows:

- **[30 marks] Code Features.** Based on your implementation and documentation
- **[30 marks] Demonstration.** Students will demonstrate their code and respond to questions from the markers. Marks are assigned for each feature implemented and for the usability and flexibility of the code.
- **[40 marks] Project Report.** As described in section 3.1.

Marks will be deducted for not following specifications outlined in this document, which includes incorrect submission format and content.

3.3 Requirements for passing the unit

Please note: As specified in the unit outline, it is necessary to have attempted the assignment in order to pass the unit. As a guide, your assignment must score at least 15% (before penalties) to be considered to have attempted this assignment. We have given you the mark breakdown in Section 3.2. Note that the marks indicated in this section represent maximums, achieved only if you completely satisfy the requirements of the relevant section.

Plagiarism is a serious offence. This assignment has many correct solutions so plagiarism will be easy for us to detect (and we will). For information about plagiarism, please refer to http://academicintegrity.curtin.edu.au.

You will be asked to explain parts of your code and the reason for choices that you have made during the demonstration. A failure to display knowledge required to have produced the code will most likely result in being formally accused of cheating.

Finally, be sure to secure your code. If someone else gets access to your code for any reason (including because you left it on a lab machine, lost a USB drive containing the code or put it on a public repository) you will be held partially responsible for any plagiarism that results.

3.4 Late Submission

As specified in the unit outline, you must submit the assignment on the due date. If there are reasons you cannot submit on time, you should apply formally for an Assessment Extension.

Students with a Curtin Access Plan should include a submission note to indicate the extra time they have taken, ensuring they have submitted the CAP to Blackboard for us to check.

In the event that you submit your assignment late, you will be penalised based on the number of days it is late. Note that if you are granted an extension you will be able to submit your work up to the extended time without penalty – this is different from submitting late.

3.5 Clarifications and Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced in the lecture and on the unit's Blackboard page. These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks. It is your responsibility to be aware of these, either by attending the lectures, watching the iLecture and/or monitoring the Blackboard page.