Fundamentals of Programming

COMP1005 Postgraduate Assignment

Adventure World

Semester 2, 2025 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 your design and implementation, and the results generated by the simulation.

2 The Challenge

You will be simulating a theme park with a range of rides with varying movement patterns. The rides will be plotted using matplotlib, with each ride represented as an object that knows how it changes on each timestep and how it is plotted. An extension of the project is to also have patrons queuing for rides, riding for a duration, then moving to another ride or leaving the park.

Practical Test 3 provided code for a rudimentary Pirate Ship ride, with the ship swinging left and right in an arc. This code could be extended and modified to provide for Ferris Wheels, Spider/Hurricane and similar rides. Plotting can be done as a side or top view, depending on the view that is of most interest.

The rides will be placed in a theme park. They should have a bounding box, which is checked against existing rides to ensure there is no overlap. The park should have one or more entrances/exits where patrons can spawn and leave (despawn). Movements between rides can be random or targeted, but should not go into the bounding boxes of the rides. Patrons will be in varying "states" when roaming, queuing and riding the attractions.

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 represent different types of rides, patrons and terrain features. Your task is to extend the code and then showcase your simulation, varying 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, however they must be referenced.

Remember : Think before you code!

You can do a lot of the assignment planning on paper before any coding. The Feature column of the **Traceability Matrix** should be filled in before coding, then used for planning the coding project and as a checklist as you work through the assignment.

The assessable features for **adventureworld.py** include:

1. **Rides**: Represented as objects that "know" their size, movement and state. A starting point for the state was given in the sample code. Rides will take a number of patrons for a set duration. ***Note**: plotting Patrons on Rides is not required (bonus points)*
   Prompts: How will you represent the state of the ride? How wil the state affect the plot? What will be the trigger to change state - will you need to fill the ride completely?
2. **Patrons**: Represented as objects that "know" their position, name and representation on the plot.
   Prompts: How will you represent the items themselves, and differentiate between them in the simulation? How does the Patron move around the park? How will the Patron know it has found a Ride?
3. **Queues/Rider Management**: These keep track of the Patrons waiting for or riding attractions. Similar to the Pet Shelter, you can stage the Patrons to move together through various statuses.
   Prompts: Consider which Python data structures are suited to implementing this feature. Could there be limits on the queue-length? Will the number of Patrons on a ride be limited/unlimited? The change in status for a ride should update the internal status for each Patron (they shouldn't keep roaming when in a queue!.
4. **Terrain**: The map of the theme park will have Rides and Patrons, and also barriers where the Patrons wont walk - the boundary of the park and other items that you can develop. The terrain for the world can be built in the program, and/or read in from files(s) for more variety.
   Prompts: How will you represent the barriers and other items on the map? How will the Patrons know what areas they can roam around?
5. **User Interface**: Your program should have two modes - interactive and batch. To run in interactive mode, the program is run with `python3 adventureworld.py -i`. Interactive mode will ask for number/type of rides and any other variables you include. In batch mode, you will use command line parameters to get the terrain and parameters e.g.: `python3 adventureworld.py -f map1.csv -p para1.csv`
   Prompts: How will you format the parameter file? How might you use the batch mode with an automated script?
6. **Simulation**: On each timestep, the Patrons will roam, queue or ride, depending on their state. The Rides will be moving or idle, depending on their state. This behaviour is driven through the step_change method which is called for each Patron and Ride on each timestep.
   Prompts: How will you manage working through all Patrons and Rides on each timestep? In what order will you consider: spawn/leave; movement; riding and plotting?
7. **Statistics (POSTGRAD ONLY)**: Your program should provide realtime and summary statistics of each of your simulations. This should use subplotting to have the park map in one subplot and statistics plotted in one or more subplots. A final summary printed/plotted/saved to file can give summary information for the scenario.
   Prompts: What interesting statics could you track and plot? Which statistics are likely to be of interest based on the scenarios?

There are marks allocated for flexibility and usability. For example, using an input file, or varying numbers of rides/ride lengths-speeds/patrons can give very different simulations. You can begin with hard-coded/generated 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. Configuration files can also be used.

Your code should include comments to explain what each section does and how. Apply PEP-8 and other style guides throughout - this will affect your **readability** score in our marking. Also beware of using while/True, break, continue and global variables – these are all strongly discouraged in the unit and will significantly reduce your code quality score.

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. Use of generative AI and other tools for coding or report-writing is not permitted.

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.

Beyond the working program, you will submit a document: the **Project Report**, worth 40% of the assignment marks. This is described in Section 3.1.

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** – python code 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.
- **You will also need to submit the Report to TurnItIn.**

Make sure that your zip file contains all items to be marked is required. Anything not included in your zip submission will not be marked. It is your responsibility to make sure that your submission is complete and correct – submitted to the main assignment link 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. **Test result** - Pass/Fail/Partial Pass/Not Implemented
    v. **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. A UML Class Diagram should be included for objects and their relationships.
5. **Showcase** (10 marks) of code output, including **three** different scenarios, e.g. different terrain, strategies and or numbers of objects to demonstrate your code:
    a. **Introduction:** (4 marks) 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:** (3x2 marks) Show and discuss each scenario's outputs/results.
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)

A report template is available on Blackboard.

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 lost 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 a reasonable attempt on the assignment in order to pass the unit. Your assignment must score at least **30%** (before penalties) to be considered a reasonable attempt. 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 (e.g. 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. If you submit your assignment late (without an extension), you will be penalised based on the number of days it is late.

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.

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.