Fundamentals of Programming

COMP5005 Postgraduate Assignment
Taking Care of Beezness

Semester 1, 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 bee hive and the activities of the honey bees on a property. Bees live in colonies where most bees are workers (female), but there are also drones (male) and a single Queen. Worker bees will build the comb structure to hold honey and baby bees. They can then collect nectar and pollen from flowers and bring it back to the hive to be transformed and stored as honey.

You will develop (at least) two views of the simulation: the **hive** and the **property.** The hive will include one or more frames to support the comb which has cells to store the honey. Bees will begin in the hive, fly out of the hive to seek pollen and nectar and return to store it. The proerty will have various plants and landmarks, which the bees with seek or avoid. The model can be assumed to be flat/2-D, although some 3D aspects can be factored in (e.g. Item height). There is a lot of flexibility in how you can approach this problem, noting that the focus is to model bee movements and the storage of honey, and models are always wrong (i.e. manage your expectations).

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 bees, creatures, target objects 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 **beeworld.py** include:

1. **User Interface**: Your program should have two modes - interactive and batch. To run in interactive mode, the program is run with `python3 beeworld.py -i`. Interactive mode will ask for number of bees and any other variables you include. In batch mode, you will use command line parameters yo get the terrain and parameters e.g.: `python3 beeworld.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?
2. **Bees**: Represented as objects that "know" their position and their state. One aspect of this will be whether they are in the hive or out on the property. A starting point for the state was given in the sample code. Bees will undertake **missions** to seek and bring back nectar from the property (world).
   Prompts: How will the bee know when it is starting a.mission? Will it move randomly, or have a strategy or way to seek nectar? How will it know it is on the return phase of the mission?
3. **Flowers**: Represented as objects that "know" their position, name and colour. Flowers may also be grouped into a Tree object.
   Prompts: How will you represent the items themselves, and differentiate between them in the simulation? How will the Bee know it has found a Flower? Will the nectar on a Flower/Tree be limited or unlimited?
4. **Strategy (postgraduates)**: When bees find a nectar source, they can communicate the location to other bees in the hive. Postgraduate students will explore the difference in hive effocoency between random and directed missions.
   Prompts: How will nectar locations be shared? How will they spread between bees? As post-graduate students are not modelling the internals of the hive, you may use random/statistival models to communicate locations to bees in the hive.
5. **Terrain**: The map of the bee world will have flowers and trees, and also barriers where the bees wont fly - including water and buildings. 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 nectar sources on the map? How will the bees know what regions they can fly onto?
6. **Simulation**: On each timestep, the bees will move and/or interact with the hive/world/other bees. This behaviour is driven through the **step_change( )** method which is called for each bee on each timestep.
   Prompts: How will the bee know if it is in the hive or outside? How will it know it is on a mission and/or carrying nectar? Will there be interaction between bees?

NOTE - undergraduate students will be focussing on the Frames, postgraduates will work on Bee communication and strategy. *Your assignment will explore the impact of changing parameters for the mission (seek) strategy and the communication of nectar location(s). The intevals of the hive will not need to be modelled or plotted. You may assume the hive has an overall size and use an overall count for the amount of honey stored. Bees will need to spend an amount of time in the hive for nectar location to be communicated.*

There are marks allocated for flexibility and usability. For example, using an input file, or varying numbers of trees/houses/roads 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.