Fundamentals of Programming

COMP1005 Assignment
Puppy Simulation

Semester 1 2024 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 the behaviour and interaction of one or more puppies/dogs. The yard the dogs live in consists of grassed area, garden, fences, gates and house. We will view it from above. Your simulation will include **dogs/puppies** (multiple types possible), **squirrels** (a major distraction), **humans** (various types), **yard/house** (sectioned by surface and accessibility), **toys** and **food sources**, and **senses** (sight, smell and sound). This will be plotted as a top-view of the activities. The model can be assumed flat/2-D – there may be 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 dogs using objects, and to add features to the simulation (e.g. food, toys, humans and interactions). 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 as a guide and checklist as you work through the assignment.

The assessable features for **snoo.py** are:

1. **Animals**: Represented as objects that "know" their position, name, colour and age and can strategise on their next activity. You should have at least **Dogs/Puppies** and **Squirrels**
   Prompts: How will you represent the animals themselves, and differentiate between them in the simulation? How will they move and decide between movement options? Will they get hungry/thirsty/lonely/bored?
2. **Humans**: Humans will have varying relationships with the Animals. Some will be owners/friends, some strangers/intruders. Humans will bring food and may play with the dog(s).
   Prompts: How will you differentiate the humans? How will the dog sense the types of humans? Will there be a regularity of interaction, e.g. feed/play at certain times?
3. **Food Sources/Toys**: The main food sources for the dogs are given by humans. Your animals should have a value for energy/hunger that is increased by eating and decreases over time. Food sources should deplete as they are eaten. Toys can be played with and moved from place to place. Items may be buried, then found via smell.
   Prompts: How will the animals find and respond to food sources? How will you track energy and the food source(s) being eaten? How will a toy be "carried"?
4. **Senses**: Each creature will have a way of sensing the world around them. You should have sight and smell as a starting point, then potentially add hearing.
   Prompts: How will you code the "sight" of the animals? Many aspects of the simulation will have a smell, which may be in a trail that fades over time. You might do this as a parallel grid... Which senses/events will take priority?
5. **Terrain and Obstacles**: There should be at least two types of area in the terrain – back yard and house. Different animals will traverse their terrains in their own manner, although within a class they should have the same patterns. Obstacles might be fences, walls or doors/gates. These can be built in the code, however they will be better if read from a file.
   Prompts: How will the animal know what terrain it is in? How will this affect their choice of movement for each time step? How will they get around obstacles?
6. **Collisions/Interaction**: How will your creatures seek out or avoid each other? They will need to detect each other and take action, with a decision being made on the outcome of any interactions.
   Prompts: How will you detect a collision is imminent? What strategies will you have for the animals to avoid each other?

There are marks allocated for flexibility and usability. For example, changing terrain input file, or numbers of dogs/humans/squirrels/food can give very different simulations. 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. 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 discouraged in the unit – even if you see examples of their use online.

It may be useful to keep track of your progress/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 in Section 3.1.

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.

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 – snoo.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 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. **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 codeoutput, including **three** different scenarios:
    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 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 (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.