PORTFOLIO OF AI INSTANCES

Version 2.0
Computing
COMP702

Edward Powley

Introduction

In this assignment, you are required to **design** and **implement** a portfolio of **three** demo applications or game components, each implementing one or more artificial intelligence (AI) techniques. The three instances will demonstrate the following three categories of AI technology:

- Authored behaviours
- Search and planning
- Logic programming and constraint satisfaction

The form of these three instances is up to you. They may be standalone, purpose-built demo applications, or they may be components of larger pre-existing projects. They may be playable game experiences, non-interactive demos, or tools for game developers (standalone or integrated into a game engine). They may be three completely separate pieces of work, or three components of a single artefact. They may be implemented using any technologies, programming languages, platforms and third-party libraries that are appropriate. However they must demonstrate your knowledge of the AI techniques being used, as well as your ability to engineer robust and maintainable software systems.

Almost all types of games use AI in some capacity, and many genres rely on advanced AI techniques. Control of non-player characters is an important application of AI, and a wide-ranging one: enemy AI in a realistic stealth game is very different from in an arcade shooter, which in turn differs from a racing game. Other applications of AI can include adversaries in board, card or strategy games, procedural content generators, procedural narrative engines, assistive technologies, AI "directors", and many others. This portfolio of AI instances will showcase the breadth and depth of your knowledge on these techniques and applications.

This assignment is formed of several parts:

(A) **Prepare** a 1-page proposal for the first AI instance that will:
  (i) **outline** the concept;
  (ii) **describe** the key requirements;
  (iii) **identify** the AI technique(s) that will be implemented.
(B) **Implement** a draft version of your first AI instance
(C)–(D) As (A)–(B), but for your second AI instance
(E)–(F) As (A)–(B), but for your third AI instance
(G) **Implement** final version of your three AI instances that will:
  (i) **revise** any issues raised by your tutor and/or your peers.
(H) **Present** a practical demo of the AI instance to your tutor that will:
  (i) **demonstrate** your academic integrity;
  (ii) **demonstrate** your individual programming knowledge, understanding of AI techniques, and communication skills.

Assignment Setup

This assignment is a **programming** task. There is no template GitHub repository for this assignment; you are expected to use version control, but use whatever platform and repository is appropriate to your project. Remember to modify the .gitignore file (or equivalent on other version control systems) to exclude temporary build files from the repository.

This assignment consists of **three formative submissions**, followed by a **single summative submission**.

After each formative submission you will receive feedback from your peers and module tutor. You will also receive feedback through regular workshop sessions. You should note this feedback and feed this into subsequent prototypes.

This final submission is subject to the usual university policies regarding late submission or non-submission, as detailed in the course handbook — even if you have met all the formative deadlines, failure to make a submission via LearningSpace by the summative deadline will be subject to penalties.

Additional Guidance

As always, avoid underestimating the effort required to implement even simple software; always consider scope. From the proposal stage, you should consider very carefully what is feasible.

Your code will be assessed on **functional coherence**: how well the finished product corresponds to the user stories, and whether it has any obvious bugs. Correspondence to user stories runs both ways: implementing features that were not present in the design ("feature creep") is just as bad as neglecting to implement features.

Your code will also be assessed on **sophistication**. To succeed on a project of this size and complexity, you will need to make use of appropriate algorithms, data structures, libraries, and object oriented programming concepts. Appropriateness to the task at hand is key: you will **not** receive credit for complexity where something simpler would have sufficed. Likewise, if you are using an engine such as Unity or Unreal, you should make use of (and build upon) the AI functionality included therein; you will **not** receive extra credit for "rolling your own" without good reason to do so.

Maintainability is important in all programming projects, but doubly so when working in a team. Use **comments** liberally to improve code comprehension, and carefully choose the **names** for your files, classes, functions and variables. Use a well-established commenting convention for **high-level documentation**. The open-source tool Doxygen supports several such conventions. Also ensure that all code corresponds to a sensible and consistent **formatting style**: indentation, whitespace, placement of curly braces, etc. Hard-coded **literals** (numbers and strings) within the source should be avoided, with values instead defined as constants together in a single place. Where appropriate, values should be exposed as properties or variables in the Unity or Unreal editor so that they can easily be "**tinkered**" without changing the source code.

Maintainability is also important when using **visual scripting** systems such as **Blueprints**. Pay special attention to the **layout** of your Blueprints, which should be tidy and should make clear the flow of control and data. Use **grouping**, **macros**, **functions**, **routing nodes** etc. to achieve this. Blueprints which resemble bowls of spaghetti will not achieve high marks!

As with all assignments on this course, you are expected to display a level of **innovation and creative flair** befitting Falmouth University's reputation as a world-leading arts institution. One approach to promoting creativity is **divergent thinking**: generating ideas by exploring many possible solutions.

Often the most interesting ideas are **subversive**: they deliberately go against convention or obvious solutions.

FAQ

- **What is the deadline for this assignment?**
  Falmouth University policy states that summative deadlines must only be specified on the MyFalmouth system.
- **What should I do to seek help?**
  You can email your tutor for informal clarifications.
- **How will I receive feedback on my work?**
  You will be given verbal feedback on your work during the session in which it is marked. If you require more in-depth feedback or discussion, please book an appointment with your tutor.
- **Is this a mistake?**
  If you have discovered an issue with the brief itself, please inform the module tutor.

Marking Rubric

Criterion	Weight	Near Pass	Pass	Merit	Distinction
Choice of concepts	20%	Choice of projects is inappropriate to demonstrate the required AI techniques. Scope of projects is either too trivial or too ambitious for the time and resources available. Project concepts are uncreative and unengaging.	Choice of projects is appropriate to demonstrate the required AI techniques. Scope of projects is appropriate for the time and resources available. Project concepts are somewhat creative and engaging.	Choice of projects is highly appropriate to demonstrate the required AI techniques. Scope of projects is appropriate for the time and resources available. Project concepts are creative and deliver an engaging experience.	Choice of projects is highly appropriate to showcase advanced knowledge of the required AI techniques. Scope of projects is appropriate for the time and resources available. Project concepts are exemplary in terms of creative thinking and/or delivery of an engaging experience.
Functional Coherence	20%	Few requirements have been implemented and/or the code fails to compile or run. Obvious and serious bugs are detected.	Almost all requirements have been implemented. There is little evidence of feature creep. Some minor bugs are detected.	All requirements have been implemented. There is almost no evidence of feature creep. Some superficial bugs are detected.	All requirements have been implemented. There is no evidence of feature creep. No bugs are detected.
Sophistication	40%	Little insight into the appropriate use of programming constructs is evident from the source code. Solutions show little or no understanding of appropriate AI techniques. The program structure is poor or non-existent.	Considerable insight into the appropriate use of programming constructs is evident from the source code. Solutions show good working knowledge of appropriate AI techniques. The program structure is effective.	Significant insight into the appropriate use of programming constructs is evident from the source code. Solutions show extensive up-to-date knowledge of appropriate AI techniques. The program structure is very effective. There is high cohesion and low coupling.	Extensive insight into the appropriate use of programming constructs is evident from the source code. Solutions build upon cutting-edge AI techniques with advances of the student's own. The program structure is extremely effective. There is very high cohesion and very low coupling.

Criterion	Weight	Near Pass	Pass	Merit	Distinction
Maintainability	20%	Code formatting, commenting and structure are ineffective and hinder readability.	Code is somewhat consistently formatted. Code is adequately commented. There is little unnecessary duplication of code or of literal values.	Code is consistently formatted. Code is appropriately commented. There is no unnecessary duplication of code or of literal values. Literal values can be tweaked easily in code, or if appropriate through a configuration file or editor interface.	Code is consistently formatted. Code is appropriately and meaningfully commented. There may be auxiliary documentation to aid maintenance. There is no unnecessary duplication of code or of literal values. Literal values can be tweaked easily, and attention has been paid to the appropriate user experience for this.