PORTFOLIO OF GAME ENGINE COMPONENTS — AI (TASK 1)

Version 2.0
BSc Computing for Games
COMP250

Dr Ed Powley

Introduction

In this assignment, you are required to **design** and **implement** a game component implementing one or more artificial intelligence (AI) techniques. Note that "component" here may also include tools to help game designers to author AI behaviours or systems.

It is strongly recommended that your component is integrated into your COMP240 group game development project, however if this is not appropriate you may integrate it into another of your current or previous development projects. You may use any programming language and game engine you deem appropriate, including visual scripting languages such as Blueprints; this choice will probably be dictated by your choice of game.

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. Your final product will be a portfolio piece, which you can use in future to demonstrate your mastery of these techniques.

*"To be enjoyable, an AI must put up a good fight but lose more often than win. It must make the player feel clever, sly, cunning, and powerful. It must make the player jump from his seat shouting, 'Take that, you little s**t!' "*

— Mat Buckland

"Real stupidity beats artificial intelligence every time."

— Terry Pratchett

This assignment is formed of several parts:

(A) **Write** a 2-page handout that will:
  (i) **outline** the concept of your component;
  (ii) **identify** the game into which your component will be integrated, and how it will fit into the overall game concept;
  (iii) **describe** the key requirements of your component.
(B) **Populate** a task board with the user stories for your component.
(C) **Implement** a draft versions of your component that will:
  (i) **implement** the user stories from your Trello board.
(D) **Implement** a final version of your component that will:
  (i) **revise** any issues raised by your tutor and/or your peers.
(E) **Present** a practical demo of the computer program to your tutor that will:
  (i) **demonstrate** your academic integrity;
  (ii) **demonstrate** your individual programming knowledge and communication skills.

Not all video games portray AI in a positive light. However, many games rely on AI to create a satisfying gameplay experience.

Assignment Setup

This assignment is a **programming** task. There is no template GitHub repository for this assignment; you should work within the existing source control repository for your game or create a new GitHub repository as appropriate. Either way, remember to modify the `.gitignore` file (or equivalent on other version control systems) to exclude temporary build files from the repository.

Part A

Part A consists of a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis. Answer the following questions to pass:

- What is the title and high concept of the game or demo into which your component will be integrated?
- What functionality will your component include?
- How does your component fit into the overall concept of the game or demo?
- What are the key requirements?
- Is the scope appropriate for the product development time-frame?

To complete Part A, prepare the handout using any word processing tool. Your handout may include images and/or links to online videos.

Show the handout to the **tutor** for immediate **informal feedback**.

Part B

Part B consists of a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis. Answer the following questions to pass:

- What are the user stories for your component?
- Which user stories are required for a minimum viable product, and which are stretch goals?
- Is the scope appropriate for the product development time-frame?

To complete Part B, populate a task board with your user stories. If you have an existing task board for your game development project (e.g. a Trello board or a physical task board), use labelling or colour-coding to identify those user stories which comprise your AI component — it is **not** recommended that you create a separate task board for your component.

Show the task board to the **tutor** for immediate **informal feedback**.

Part C

Part C is a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

(a) Submission is timely;
(b) Enough work is available to conduct a meaningful review;
(c) A broadly appropriate review of a peer's work is submitted.

To complete Part C, prepare draft versions of the computer programs. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to the scheduled sprint review sessions. Then, attend the scheduled sprint review sessions.

Part D

Part D is a **single summative submission**. This work is **individual** and will be assessed on a **criterion-referenced** basis. Please refer to the marking rubric at the end of this document for further detail.

To complete Part D, revise the computer program based on the feedback you have received. Then, upload it to the LearningSpace. Please note, the LearningSpace will only accept a single `.zip` file.

You will receive **formal feedback** from your **tutor** three weeks after the final submission deadline.

Part E

Part E is a **single summative submission**. This work is **individual** and will be assessed on a **threshold** basis. The following criteria are used to determine a pass or fail:

(a) Enough work is available to hold a meaningful discussion;
(b) Clear evidence of programming knowledge **and** communication skills;
(c) No breaches of academic integrity.

To complete Part E, prepare a practical demonstration of the computer program. Ensure that the source code and related assets are pushed to GitHub and a pull request is made prior to the scheduled viva session. Then, attend the scheduled viva session.

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 deadlines must only be specified on the MyFalmouth system.
- **What should I do to seek help?**
  You can email your tutor for informal clarifications. For informal feedback, make a pull request on GitHub.
- **Is this a mistake?**
  If you have discovered an issue with the brief itself, the source files are available at:
  `https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs`.
  Please make a pull request and comment accordingly.

Additional Resources

- `http://aigamedev.com`
- `https://docs.unity3d.com/Manual/Navigation.html`
- `https://docs.unrealengine.com/latest/INT/Gameplay/AI/`
- `https://google.github.io/styleguide/cppguide.html`

Marking Rubric

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
Basic Competency Threshold	40%	At least one part, or at least one sprint review, is missing or is unsatisfactory.	Submission is timely. Enough work is available to hold a meaningful discussion. Clear evidence of programming knowledge and communication skills. No breaches of academic integrity. The student participates in all sprint reviews				
Appropriateness of User Stories and Sprint Plans	5%	Few user stories are distinguishable and easily measured. Sprint plans provide little support for the project.	Some user stories are distinguishable and easily measured. Sprint plans provide some support for the project.	Most user stories are distinguishable and easily measured. User stories correspond to the product design. Sprint plans provide much support for the project.	Nearly all user stories are distinguishable and easily measured. User stories clearly correspond to the product design. Sprint plans provide considerable support for the project.	All user stories are distinguishable and easily measured. User stories clearly and comprehensively correspond to the product design. Sprint plans provide significant support for the project.	All user stories are distinguishable and easily measured. User stories clearly and comprehensively correspond to the product design. Sprint plans provide extensive support for the project.
Functional Coherence	5%	Few user stories have been implemented and/or the code fails to compile or run. Many obvious and serious bugs are detected.	Some user stories have been implemented. Some obvious bugs are detected.	Many user stories have been implemented. There is some evidence of feature creep. Few obvious bugs are detected.	Almost all user stories have been implemented. There is little evidence of feature creep. Some minor bugs are detected.	All user stories have been implemented. There is almost no evidence of feature creep. Some bugs, purely cosmetic and/or superficial in nature, are detected.	All user stories have been implemented. There is no evidence of feature creep. Few to no bugs are detected.
Sophistication	15%	Little insight into the appropriate use of programming constructs is evident from the source code. The program structure is poor or non-existant.	Some insight into the appropriate use of programming constructs is evident from the source code. The program structure is adequate.	Much insight into the appropriate use of programming constructs is evident from the source code. The program structure is appropriate.	Considerable insight into the appropriate use of programming constructs is evident from the source code. The program structure is effective. There is high cohesion and low coupling.	Significant insight into the appropriate use of programming constructs is evident from the source code. 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. The program structure is extremely effective. There is very high cohesion and very low coupling.
Maintainability	15%	The code is only sporadically commented, if at all, or comments are unclear. Few identifier names are clear or inappropriate. Code formatting hinders readability.	The code is well commented. Some identifier names are descriptive and appropriate. An attempt has been made to adhere to a consistent formatting style. There is little obvious duplication of code or of literal values.	The code is reasonably well commented. Most identifier names are descriptive and appropriate. Most code adheres to a sensible formatting style. There is almost no obvious duplication of code or of literal values.	The code is reasonably well commented, with appropriate Doxygen-compatiable documentation. Almost all identifier names are descriptive and appropriate. Almost all code adheres to a sensible formatting style. There is no obvious duplication of code or of literal values. Some literal values can be easily "tinkered".	The code is very well commented, with comprehensive appropriate Doxygen-compatiable documentation. All identifier names are descriptive and appropriate. All code adheres to a sensible formatting style. There is no obvious duplication of code or of literal values. Most literal values are, where appropriate, easily "tinkered" outside of the source.	The code is commented extremely well, with comprehensive appropriate Doxygen-compatiable documentation. All identifier names are descriptive and appropriate. All code adheres to a sensible formatting style. There is no duplication of code or of literal values. Nearly all literal values are, where appropriate, easily "tinkered" outside of the source.

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
Creative Flair	10%	Little or no creativity. The work is a clone of an existing work with mere cosmetic alterations. The work delivers little or no fun and/or engagement.	Some creativity. The work is derivative of existing works, with only minor alterations. The work delivers some fun and/or engagement.	Much creativity. The work is derivative of existing works, demonstrating little divergent and/or subversive thinking. The work delivers much fun and/or engagement.	Considerable creativity. The work is somewhat novel, demonstrating some divergent and/or subversive thinking. The work delivers considerable fun and/or engagement.	Significant creativity. The work is novel, demonstrating significant divergent and/or subversive thinking. The work delivers significant fun and/or engagement.	Extensive creativity. The work is highly original, with strong evidence of divergent and/or subversive thinking. The work delivers extensive fun and/or engagement.
Portability and Navigability	5%	Product will not execute at all on another machine, for reasons related to code portability, even if they are trivially resolvable. The directory structure inside the submitted zip file is unclear. Provided template has not been followed well, if at all.	Several portability issues are present. The directory structure inside the submitted zip file is somewhat confusing. The provided template has mostly been followed.	Some portability issues are present. The directory structure inside the submitted zip file is adequate. The provided template has been followed.	Few portability issues are present. The directory structure inside the submitted zip file is mostly sensible. The provided template has been followed.	Almost no portability issues are present. The directory structure inside the submitted zip file is sensible. The provided template has been followed.	No portability issues are present. There is cross-platform compatibility. The directory structure inside the submitted zip file is sensible. The provided template has been followed.
Use of Version Control	5%	Material has been checked into GitHub less frequently than once per sprint.	Code has been checked into GitHub at least once per sprint.	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. There is some evidence of engagement with peers (e.g. code review).	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. There is much evidence of engagement with peers (e.g. code review).	Code has been checked into GitHub several times per sprint. Commit messages are clear, concise and relevant. There is significant evidence of engagement with peers (e.g. code review).	Code has been checked into GitHub several times per week. Commit messages are clear, concise and relevant. There is extensive evidence of engagement with peers (e.g. code review).