PORTFOLIO OF GAME ENGINE COMPONENTS — AI (TASK 2)

Version 2.0
BSc Computing for Games
COMP250

Dr Ed Powley

Introduction

In this assignment, you are required to design and implement an **AI bot** for the **MicroRTS** game.

In most games, the role of AI agents is to provide an enjoyable experience for the player. However, creating agents that instead try to win the game is an important area of research: in some genres of games there is demand for such AI, and it allows games to be used as a testbed for AI techniques that can be applied to real-world decision problems. Real-time strategy (RTS) games are particularly challenging for AI to play well, involving complex tactical and strategic decisions. MicroRTS is a simplified RTS which was developed for benchmarking AI techniques against each other. In this assignment you will develop a "bot" (an AI agent) that plays MicroRTS. Your bot will compete in a tournament against your classmates' bots.

*"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) **Implement**, over a course of several iterative sprints, draft versions of your bot. **Upload** these to the tournament server.
(B) **Implement** a final version of your bot that will:
  (i) **revise** any issues raised by your tutor and/or your peers.
(C) **Present** a practical demo of your bot that will:
  (i) **demonstrate** your academic integrity;
  (ii) **demonstrate** your individual programming knowledge and communication skills.

Assignment Setup

This assignment is a **programming** task. Begin by forking the following GitHub repository:

`https://github.com/Falmouth-Games-Academy/comp250-microrts`

Modify the provided template code to implement your bot. The base repository's `readme.md` file contains important instructions for submitting your bot to the tournament server; be sure to read and follow these instructions carefully.

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

Part A

Part A 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 A, develop your bot in an iterative fashion. Push your code to GitHub regularly; if you have forked the base repository correctly, every push to the `master` branch will automatically submit your bot to the tournament server. Use the results from the tournament server to refine your bot design.

Ensure that a sufficiently advanced draft is uploaded to GitHub, and attend the scheduled peer review session.

Part B

Part B 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 B, revise your bot based on the feedback you have received. Then, upload it to the LearningSpace. Please note, the LearningSpace will only accept a single `.zip` file, which must include your submissions for **both** Task 1 (Game Component) **and** Task 2 (MicroRTS Bot).

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

Part C

Part C 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 C, prepare a practical demonstration of your bot. 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. Consider allowing some literal values, where appropriate, to be "tinkered" without changing the source code, e.g. by defining them in an external file read at startup.

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. The student is disqualified from the tournament.	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. Tournament rules have been adhered to. The student participates in all sprint reviews.				
Functional Coherence	5%	The code fails to compile or run. Many obvious and serious bugs are detected.	Some obvious bugs are detected.	Few obvious bugs are detected.	Some minor bugs are detected.	Some superficial bugs are detected.	Almost no bugs are detected.
Algorithmic Sophistication	10%	Little insight into the appropriate use of AI techniques is evident from the design of the bot.	Some insight into the appropriate use of AI techniques is evident from the design of the bot.	Much insight into the appropriate use of AI techniques is evident from the design of the bot.	Considerable insight into the appropriate use of AI techniques is evident from the design of the bot.	Significant insight into the appropriate use of AI techniques is evident from the design of the bot.	Extensive insight into the appropriate use of AI techniques is evident from the design of the bot.
Sophistication of Implementation	10%	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.
Performance	10%	No bot is entered into the tournament, or the entered bot is disqualified.	Marks to be allocated according to the bot's Elo score, as measured in a round-robin tournament.				
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.

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
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).

Appendix: Tournament Rules

1. Bots will play in a round-robin (all-versus-all) tournament. Bots are ranked by Elo score, based on wins and losses.
2. A rolling tournament server is available, to which you may upload your bot for testing. Final leaderboard position will be based on the version of your bot that is submitted to LearningSpace.
3. If the submitted bot cannot be compiled on the tournament server, it receives the lowest possible score.
4. There is a time limit of 100 milliseconds per game tick. If a bot takes longer than 100 milliseconds to choose an action, it forfeits the current match (which is recorded as a loss).
5. If a bot crashes by raising an uncaught exception, it forfeits the current match.
6. Bots **must not** create additional threads or processes, perform computations on the GPU, or perform network access. Any bot found to be doing these will be disqualified from the tournament.
7. A bot **may** read and write files in its specified working directory, or subdirectories thereof. It **must not** access files outside its working directory.
8. **Any attempt** to compromise the security of the tournament server, or otherwise deliberately interfere with the smooth running of the tournament and the fairness of the results, will result in **permanent disqualification** from the tournament and **failure of the Basic Competency Threshold** for this assignment. This is at the discretion of the Module Leader.