DISTRIBUTED PROCESSING TASK

Version 2.0.1
BSc Computing for Games
COMP260

Gareth Lewis & Dr Michael Scott

Introduction

"...a folk definition of insanity is to do the same thing over and over again and to expect the results to be different. By this definition, we in fact require that programmers of multithreaded systems be insane. Were they sane, they could not understand their programs."

— Edward A. Lee

"No one can write correct programs in a language where a=a+1 is not deterministic."

— Luiz Henrique de Figueiredo

"Frameworks don't solve scalability problems, design solves scalability problems."

— Ryan Tomayko

In this assignment, you are required to design and implement algorithms that process data in a *distributed* manner.

Games are resource intensive. Compounding this issue, players are sensitive to performance issues. It is critical, then, to leverage available resources to ensure adequate performance. Distributed processing is one solution. Apply the principles of coordination and agreement, and you will be successful.

This assignment is formed of several parts:

(A) **Design** a distributed processing architecture in UML for a MUD that will:
  i. **support** multiple client instances on a single computer;
  ii. **enable** players to navigate **multiple** locations in a virtual dungeon;
  iii. **allow** players to be aware of other players in the same location;
  iv. and **permit** players in the same room to communicate.

(B) **Implement** a MUD prototype that will:
  i. **support** multiple clients using socket-based networking;
  ii. **incorporate** distributed processing using threads;
  iii. and be **realised** as a client-server architecture.

(C) **Implement** a more refined design **and** MUD prototype that will:
  i. **revise** any issues raised by your tutor and/or your peers.

(D) **Present** a practical demonstation of the MUD prototype that will:
  i. **show** academic integrity **and** technical communication skills.

Assignment Setup

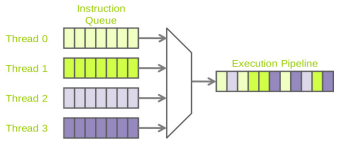
This assignment is a **programming task**. Fork the GitHub repository at:

`https://github.com/Falmouth-Games-Academy/comp260-server`

Use the existing directory structure and, as required, extend this structure with sub-directories. Ensure that you maintain the `readme.md` file.

Modify the `.gitignore` to the defaults for **C#** and **Visual Studio**. Please, also ensure that you add editor-specific files and folders to `.gitignore`.

Part A



Multi-threading is commonly used to improve performance in games.

Part A consists of a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis. This deliverable is not assessed and is intended to be advisory at this stage.

To complete Part A, incorporate the design, using UML, into the `readme.md` document. Show this to your tutor in-class. If acceptable, it will be signed-off.

You will receive immediate **informal feedback** from your **tutor**.

Part B

Part B 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 B, prepare a draft version of the MUD. Please ensure that the source code and related assets are pushed to GitHub and are made available prior to the scheduled peer-review workshop. Then, attend the scheduled session.

You will receive immediate **informal feedback** from your **peers**.

Part C

Part C 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 C, revise the MUD based on the feedback you have received. Then, upload it to the LearningSpace. Ensure that you include the `readme.md` document containing the design that you developed in Part A. 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 D

Part D 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 D, prepare a practical demonstration 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 viva session. Then, attend the scheduled viva session.

You will receive immediate **informal feedback** from your **tutor**.

Additional Guidance

A common pitfall is poor planning or time management. Many underestimate the work involved in designing and implementing multiplayer games. It simply cannot be crammed into a last minute deluge just before a deadline. There is a critical and time-consuming phase of testing! It is, therefore, very important that you begin work early and sustain a consistent pace: little and often.

The first deadline is close to the start of the module and not much material will have been covered by this point. Please rest assured. This first formative submission is supposed to be a simple analysis of design. It is advisory—and a way to kick start the project such that you recieve early feedback to give you some direction and to encourage you to practice your programming skills.

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 raise an issue and comment accordingly.

Additional Resources

- Additional resources have been migrated to the Talis Aspire system, which is available at: `http://resourcelists.falmouth.ac.uk/`.

Marking Rubric

Criterion	Weight	Refer for Resubmission	Basic Proficiency	Novice Competency	Novice Proficiency	Professional Competency	Professional Proficiency
Threshold	40%	At least one part is missing or is unsatisfactory.	Parts A—D are complete and timely. Enough work is available to hold a meaningful discussion. Provided a meaningful review of a peer's work. Clear evidence of programming knowledge and communication skills. Apprioriate use of GitHub for version control. No breaches of academic integrity.				
Design	20%	Little to no design work. Design does not incorporate concurrency and/or a databasing. Little to no UML.	Design has some merit. Design sufficiently incorporates concurrency and databasing. Sufficient use of UML.	Design has modest merit. Design adequetely incorporates concurrency and databasing. UML and other notation is leveraged adequetely. Scale and performance are briefly mentioned.	Design has much merit. Design appropriately incorporates concurrency and databasing. UML and other notation is leveraged somewhat appropriately. Scale and performance are considered.	Design has considerable merit. Design, with high appropriateness, incorporates concurrency and databasing. UML and other notation is leveraged appropriately. Issues such as scale and performance, with reference to CAP theorem, are considered.	Design has significant merit. Design, with high appropriateness, incorporates concurrency and databasing. UML and other notation is leveraged highly appropriately. Issues such as scale and performance, with reference to CAP theorem, are carefully considered.
MUD Architecture	20%	Little to no evidence of distributed processes.	Some distributed processes. Player chat system and/or space navigation operable. Supports at least two client instances.	Modest number of distributed processes. Player chat system and space navigation operable. Basic commands implemented. Supports serveral client instances.	Many distributed processes. Player chat system and space navigation implemented well. Interesting commands implemented. Evidence to suggest scalability to support many client instances.	Considerable number of distribued processes. Player chat system and space navigation implemented effectively. Sophisticated commands implemented. Evidence to suggest scalability to support a great many client instances.	Significant number of distributed processes. Player chat system and space navigation implemented highly effectively. Highly sophisticated commands implemented. Evidence to suggest scalability to support hundreds of client instances.
Demo	20%	No demo. Little to no ability to articulate either networking or concurrency concepts.	Demo somewhat sufficient to illustrate key distributed processing concepts. Some ability to articulate either networking or concurrency concepts.	Demo sufficient to illustrate key distributed processing concepts. Modest ability to articulate either networking or concurrency concepts. Some ability to articulate design decisions.	Demo adequete to illustrate key distributed processing concepts. Much ability to articulate either networking or concurrency concepts. Modest ability to articulate design decisions.	Demo appropriate to illustrate key distributed processing concepts. Considerable ability to articulate either networking or concurrency concepts. Much ability to articulate design decisions.	Demo highly appropriate to illustrate key distributed processing concepts. Significant ability to articulate either networking or concurrency concepts. Considerable ability to articulate design decisions.