DISTRIBUTED PROCESSING TASK

Version 3.0
BSc Computing for Games
COMP260

Gareth Lewis

Introduction

In this assignment, you are required to analyse existing Python applications to create a design proposal for a multiplayer dungeon, implemented as client and server applications.

"...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

This assignment is formed of several parts:

(A) **Analyse** existing 'SUD" and 'chat service' applications to determine the structure and functionality of those applications such that they can be combined and refactored to make a MUD service.

UML analysis will comprise of class hierarchy and flowchart / state diagrams for both applications.

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

— Luiz Henrique de Figueiredo

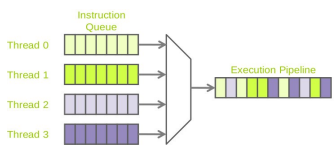
(B) **Design** a distributed processing architecture in UML for a MUD that will:

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. **Permit** players in the same room to communicate.
v. **Robustness** that allows the server to continue operation when a client is lost
vi. **Robustness** that allows a client to continue (limited) operation when server is lost
vii. **Create** a suitable wireframe mock-up of the client UI.
viii. **Use** appropriate UML techniques to capture:
    (a) The class hierarchy form of the client and server applications
    (b) The state-based function of the client and server applications
    (c) The data transmitted between client and server applications

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

— Ryan Tomayko

The solution will use the SUD and Chat Service as its core. Chat Service UI is modified to fit the needs of the multi-user dungeon.



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

Assignment Setup

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

https://github.com/Falmouth-Games-Academy/comp260-design

cont...

The repo contains two applications; 'SUD' a single user dungeon application, which was discussed in session 2 and 'Chat Service', a client/server-based chat application which was discusses in session 4.

Part A

Part A consists of a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis.

To complete Part A, analyse the code for both applications and create appropriate UML designs that capture the form (class hierarchy) and function (state diagrams, flowcharts).

Part B

Part B consists of a **single formative submission**. This work is **individual** and will be assessed on a **threshold** basis.

To complete Part B, use the analysis from part A that will implement a multiuser dungeon as a combination of the existing MUD and Chat Service components. In addition to form and functional diagrams, be sure to include sequenced diagrams that show how user input is sent from client to server and how server output is sent back to clients.

Take parts A & B, zip them up and upload them onto Learning Space.

You will receive formal feedback within three weeks

Additional Guidance

Part A of this assignment is a reverse engineering activity, a core software development skill that will allow you take working systems and develop suitable UML diagrams that describe the form, functionality and dataflow of complex systems.

Part B of this assignment is a design reuse and refactoring activity, another core activity that enables engineers to make 'new' software from existing application that perform a subset of designed functionality.

Both of this activities should demonstrate that software development is about design as much as it is about implementation, an hour of design can save a week of hacking around in code.

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:

cont...

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:**

  https://resourcelists.falmouth.ac.uk/

cont...

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. Submission of client and server applications in Python. Clear evidence of programming knowledge and communication skills. No breaches of academic integrity				
Analysis	20%	Little or no analysis of existing software applications No class hierarchy diagram of SUD application No	Existing functionality for SUD and Chat service is adequately captured within UML diagrams				
MUD Design	40%	Little or no design work. Design does not incorporate concurrency in either client or server No class hierarchy of client / server applications No state diagrams of client / server functionality No sequence diagram of client / server communications No client wireframes	Design is captured within class, state and sequence diagrams. Diagrams are laid out in a way where they are generally hard to follow and interpret. Little consideration is given to use cases. Wireframe shows basic layout of client UI				