Final Project

EGP405: Networking for Online Games (Fall 2017)

Assigned: **Monday, Sept. 18**

Due dates: **Thursday, Oct. 5 – Proposal**
Weeks 8 and 11 – Progress Checks (actual dates TBD)
Thursday, Dec. 7 – Final

Grade weight: **30%, out of 60 points**

Introduction:

The final project for this course is to create a portfolio-worthy, engineering-centric demo in the domain of online game networking. Your team has the creative freedom to design such a product, but it should somehow apply each of the following core topics discussed throughout the course:

- Networked game architecture (e.g. peer-to-peer; emulated server-client systems)
- Asynchronous messaging (e.g. turn-based gameplay; chat system)
- Synchronous messaging (e.g. real-time gameplay; some form of data coupling)
- Improving the experience (e.g. resolving latency; security)

Your contribution should ultimately fall into one of the following categories. The general scope of each is broken down here:

a) ***Build a networked game***: Using the framework of your choice, produce an online game from scratch.
   - Implement a networked game with both synchronous and asynchronous gameplay components. Alternatively, build upon an existing game concept and add synchronous and asynchronous networking elements.
   - Develop the systems required to create the networked game.
   - Integrate one or more experience-improving features.

b) ***Build a development tool***: Implement a tool that would help integrate networking into the game development pipeline.
   - Focusing on the *developer's* experience, implement a new standalone tool from scratch or as an extension to an existing engine.
   - The tool must use a networked architecture, and both synchronous and asynchronous elements.

c) ***Solve an engineering or architectural problem***: Focus on architecture and engineering by implementing a low-level system or advanced algorithm that would benefit a networked game engine.
   - The project's intent is to improve upon existing architectures or problems.
   - The project should at least *support* synchronous and asynchronous messaging, and have some experience helpers built-in.

Here are a few high concepts for potential project ideas:

- *Networked game*: Simple networked first-person shooter with chat room lobby.
- *Networked game*: Open-world exploration online game with turn-based duelling.
- *Tool*: Networked game debugging tool with state loader.
- *Tool*: Network data analytics tracking game events server-side.
- *Engineering*: Cross-platform networking framework, standalone or as plugin.
- *Engineering*: Render farm architecture.
- *Engineering*: Custom SCM or CI system.

Goals:

The project is broken up into four deliverables:

1. ***Proposal***: Form a team and come up with a project plan.
2. ***Technical design and justification***: Write an overview of code base and components; breakdown of how the project integrates course topics.
3. ***Code***: Implement a portfolio-worthy game, tool or engineering solution.
4. ***Demo***: Pitch your product, show it in action and discuss your implementation.

Instructions:

The above deliverables are broken down as follows:

1. ***Proposal***: This *short* document describes your idea and explains how it relates to the course materials. The purpose is to get you thinking about your requirements and developing an action plan early on.
    a. Describe your idea by providing the high-concept (1-2 sentences) and a breakdown of your product's purpose and requirements (a few paragraphs).
    b. Consider the features and systems you will need to implement your product and the tools that you want to use (e.g. frameworks). Using point form is fine, as long as each point has a couple of sentences about the intent of and plan for accomplishing each task.
    c. Provide a *milestone schedule* for the entire project ***and repository link***.
    d. Explain how your project will demonstrate the four main overall topics of the course: networked architecture, asynchronous messaging, synchronous messaging, and experience improvement.
2. ***Technical design and justification***: This *short* document describes the final product and its systems. The purpose is to help anyone who uses it navigate the code base; think of it as a more detailed (and professional) read-me.
    a. Provide UML diagrams of the core classes and systems.
    b. Explain the purpose of major features and justify system design choices. Provide code snippets if it helps you with your points.
    c. Show where to find the pertinent features and systems in the code base.

3. ***Code***: This is the entire code base developed, either for a game, tool, or framework or architecture solution.
    a. Your project should be *based* in ***C/C++***. Networking should be done using *RakNet*, unless otherwise justified.
    b. If using another engine for the front-end, ensure your project is clean and minimal. ***You may not use another engine's networking support!***
    c. Maintain workflow using version control.
    d. Document all contributions thoroughly.
4. ***Demo***: The final presentation in lieu of having a final exam.
    a. Team members will meet with me (privately or in front of a panel is to be determined) and provide an explanation and demonstration of what they have created. Slides or other supporting materials are optional.
    b. The team will be required to show one or two major contributions in code.
    c. After the demo, there will be a question and answers period.
    d. The presentation should take between 20 – 30 minutes. Actual time to be determined closer to the final week of classes.

Additional Requirements:
You may work in teams of up to ***four*** members for this project. Teams with more members will have higher expectations; be sure to scope your project accordingly. You will be required to complete a peer review of your team members; failure to do so by the final submission date will result in a ***zero*** grade on the project. The format of the peer review is to be determined.

Your project may be completed using existing frameworks and engines. However, ***any existing networking support in your selected frameworks and engines may not be used***. Any such use of existing networking technology will result in a ***zero*** grade on the project.

Submission Guidelines:
1) Include the following header information at the top of all source files ***modified***:
    a. "This file was modified by <*names*> with permission from author."
2) Include the following header information at the top of all source files ***created***:
    a. Team member names and student IDs.
    b. Course code, section, project name and date.
    c. Certificate of Authenticity (standard practice): "*We certify that this work is entirely our own. The assessor of this project may reproduce this project and provide copies to other academic staff, and/or communicate a copy of this project to a plagiarism-checking service, which may retain a copy of the project on its database*."

3) All sections of code modified or written by your team should be commented, explicitly stating ***who*** made the change or wrote code and ***why***, and what the code ***does*** in the context of the program.
4) Your code base will be submitted using version control. ***Do not track garbage files in your repository!*** Ignore filters exist for a reason!
5) In the *final* Canvas submission, add:
    a. Your initial proposal document as a PDF file (also submitted week 6).
    b. Your design justification document as a PDF file.
    c. Any supplementary materials used in your demo; e.g. PowerPoint slides, handouts, etc.
    d. A link to your public repository.
    Note: Grades will not be discussed or released through public repositories, for any reason, in compliance with FERPA.
    Note: Zipped submissions will not be accepted for this project; I should be able to pull your repository and build at the tip revision.

Grading:
Grading for this project is per deliverable, with each broken into categories, with specific objective expectations listed in the respective tables below.

1. ***Proposal***: 10 points (5%), due *week 6*.

Points **Category**	**0** **N/A**	**1**	**2** **Fair**	**3**	**4** **Meets Expectations**	**5** **Exceeds Expectations**
Concept Justification	Concept is weak or does not reflect the objectives.		Concept not entirely clear and needs refinement.		Concept is clear and thought-out.	Concept is thought-out and within scope.
Plan & Milestones	Concept does not outline plan; no repo link provided.		Plan is incomplete or raises concerns.		Complete plan with all features considered.	All features considered in detail.

2. ***Technical design and justification***: 10 points (5%), due *week 14*.

Points **Category**	**0** **N/A**	**1**	**2** **Fair**	**3**	**4** **Meets Expectations**	**5** **Exceeds Expectations**
Design	Design not provided.		Diagrams & descriptions provided but do not fully supplement the system implemented.		Diagrams and descriptions provided; adequately summarize features.	Diagrams and descriptions adequately summarize features, and help navigate code.
Justification	Justification not provided.		Design choices are not entirely clear.		All design choices clearly justified.	All design choices justified with arguments for and against.

3. ***Code***: 20 points (10%), due *week 14*.

Points Category	0 N/A	1	2 Fair	3	4 Meets Expectations	5 Exceeds Expectations
Functionality	Application does not serve the purpose or goals of the project.		Application serves its purpose with some bugs or issues.		Application serves its purpose with no bugs.	Application serves its purpose with "contingency plans" for user errors.
Features	Several required features not implemented.		Some of the required features implemented.		Most of the core features implemented; one or more delighters.	All required features and multiple delighters implemented.
Architecture	Code is not coherent, organized or modular.		Code has some organization and modularity.		Code is clean, coherent and organized into functions and modules where applicable.	Project makes use of appropriate file structure with full modularity.
Continuous Integration	Milestone checks lack evidence of progress.		One progress check met with some evidence of progress.		Both progress checks met with evidence of progress.	Strong evidence of progress throughout life of project.

4. ***Demo***: 20 points (10%), due *week 14*.

Points Category	0 N/A	1	2 Fair	3	4 Meets Expectations	5 Exceeds Expectations
Presentation Style	Team was not professional; presentation was not coherent.		Somewhat coherent articulation of ideas; team was somewhat professional.		Team articulated ideas and discussed contributions.	Team was confident and professional; discussed contributions from all members.
Evidence of Preparation	No indication of team being ready to present.		Some indication of preparation; team stumbled through presentation.		Demo and presentation were seamless.	Demo and presentation were seamless; supporting materials provided.
Code Walkthrough	Team did not present code features.		Full team did not seem to know the code.		Detailed explanation of at least one major feature.	Detailed explanation of multiple major features.
Questions Answered	Team did not answer fielded questions.		Team responded poorly or struggled to respond to questions.		Team responded confidently to questions.	Team responded confidently and added discussion points.

Specific penalties:

- **–20 points**: Submission does not build and run *correctly* and *immediately* out of the box (test using both debug and release modes).
- **–20 points**: Code not documented and commented, including contributions, explanations, certificate of authenticity, etc.
- **–20 points**: Lacking or weak evidence of having used version control to maintain project. Please ask for help if needed.
- **–20 points**: Repository tracks junk files. Use an ignore filter appropriately tailored to your framework to ensure you have removed the correct files, and ask for help if needed.

Good luck, learn lots and have fun! ☺