Gameplay Programming Patterns

Professor: Mattia Romeo ***mromeo@giganticmechanic.com***

Course Description

Gameplay programming is a mess. Once you've factored out engine-level systems like graphics and physics, what remains is a complex tangle of concepts and relationships - often unique to your game - that can be difficult to express clearly in code.

The goal of this class is to provide students with a set of techniques - applicable across different languages, genres and game engines - that can help tame that complexity. To achieve that goal student will develop a game in Unity over the course of the semester. Most weeks we will introduce a new technique and add a feature to the game that highlights the utility of the technique. Along with developing new features students will also be responsible for reviewing each other's code, as well as maintaining and revising their codebase. There will also be guest lectures by experienced developers who will discuss common issues they face during development and the techniques they use to resolve those issues.

Course Objectives

By taking this course students will:

- Learn how to write code that is more flexible and adaptable to changes in the game's design.
- Learn approaches to managing the complexity of gameplay programming.
- Develop a "toolbox" of useful C# code that they can use in other Unity projects and adapt to other languages.
- Develop a small game and refine it over the course of the class.

Grading and Assignments

Students will have weekly assignments. All assignments are due at the beginning of the next class. Students will be judged on the code they commit to their projects and on class participation. Late work will not be accepted unless expressly discussed with the instructor.

Final grades will be determined according to the following breakdown:

Class Participation:	25%
Weekly Assignments:	25%
Student Project #1:	20%
Student Project #2:	30%

Prerequisites

- **Foundation in object oriented programming:** You should be competent in at least one language that supports object oriented programming (e.g. ActionScript, Java, C#, C++, Python, etc.)

- **Game Development Experience:** You should have programmed at least one game of non-trivial scope (i.e. a game that has multiple different types of objects interacting with one another).
- **Game Engine/Library Experience:** You should be familiar with at least one game development/creative coding library (e.g. Processing, Cinder, OpenFrameworks) or engine (e.g. Unity, Unreal)

Statement of Academic Integrity

Plagiarism is presenting someone else's work as though it were your own. More specifically, plagiarism is to present as your own: A sequence of words quoted without quotation marks from another writer or a paraphrased passage from another writer's work or facts, ideas or images composed by someone else.

In the context of this class, in particular, plagiarism is presenting someone else's code as your own. Working together to solve problems, using the internet as a resource to answer questions, etc. is allowed, even encouraged, however, it should be done in the service of developing your own implementations of algorithms, not duplicating existing work.

Accessibility

Academic accommodations are available for students with documented disabilities. Please contact the Moses Center for Students with Disabilities at 212 998-4980 for further information.

Attendance

Attending and arriving on time to all class sessions is required and expected. This includes all labs, recitations, and critiques. If you will be missing a class due to illness, or unavoidable personal circumstances, you must notify your professor in advance via email for the absence to be excused.

Unexcused absences and being late to class will lower your final grade. Three unexcused absences lower your final grade by a letter. Each subsequent unexcused absence will lower another letter grade. Two tardies will count as one unexcused absence. Arriving more than 15 minutes late to class will also count as an unexcused absence.

See departmental guidelines for more information.

Schedule

Below is the planned schedule for this class. It is subject to change at the instructor's discretion.

Week 1: Introduction

Introductions

Course overview
Review Syllabus
Review Assignment
Review Git and Bitbucket/Github

ASSIGNMENT: Project Setup & Player Controlled Ship

- Create a project in your platform of choice.
    - You're going to be using this project for the bulk of the semester so keep that in mind when choosing your toolset.
    - This project should be playable on a PC/Mac (i.e. your project doesn't need to be cross platform but it must be runnable on a desktop/laptop)
    - This project is going to be implementing a top-down arena shooter (think of *Geometry Wars* or *Robotron*).
- Implement the following:
    - A player controlled ship:
        - The ship should be controllable via either mouse or keyboard controls.
        - The ship should fire bullets.
- Push your project to a public online repository and send the professor a link to the project.

Week 2: Subclass Sandbox

Lecture on Subclass Sandbox
Review assignment requirements
Workshop

ASSIGNMENT: Multiple Enemy Types

- Use the subclass sandbox pattern to implement two different enemy types. These enemies need to be differentiated by:
    - Ship Graphics: Each enemy type should use a distinct ship graphic.
    - Movement: Each enemy type should have distinct movement (e.g. different speed, acceleration, movement patterns like moving in a zig-zag pattern, etc.).
    - Audio: Each enemy type should play different sounds when they are created and destroyed.
- Submit assignment by committing work to project's Git repository.

Week 3: Manager/System

Lecture on Manager/System
Present assignments
Workshop
Review assignment requirements

ASSIGNMENT: Enemy Manager

- Use the manager pattern to implement an enemy manager that implements enemy “waves”. The manager will need to:
  - Keep track of how many enemies are alive at any given time.
  - When there are no enemies alive, the enemy manager should emit a new wave of enemies.
  - A wave of enemies is defined by:
    - A list of enemies that make up the wave.
    - The rate at which enemies are emitted.
- Submit assignment by committing work to project’s Git repository.

Week 4: Observer

Present assignments
Lecture on Observer/Event Manager
Workshop
Review assignment requirements

ASSIGNMENT: Accelerating Enemy

- Implement a synchronous event dispatcher (a concrete implementation of the observer pattern).
- Use the resulting event manager to implement a new enemy type that increments its speed when any enemy is destroyed.
- Submit assignment by committing work to project’s Git repository.

Week 5: Tasks/Processes

Present assignments
Lecture on Tasks/Processes
Workshop
Review assignment requirements

ASSIGNMENT: Start Work on Boss

- Implement a task manager and base task class.
- Using tasks implement a boss enemy. The boss enemy should sequence through the following states:
  - Appear: The boss appears in the center of the screen at 1% scales and quickly scales up to 100% scale. When at 100% scale it enters the spawn phase.
  - Spawn: Spawns waves of enemies from its location in sequence (i.e. it will emit the second wave when the first one is destroyed, etc.). When the boss reaches 50% health it enters the fire phase.
  - Fire: Quickly fires bullets in random directions. When it reaches 15% health it enters the chase phase.
  - Chase: Chases the player quickly while spawning waves of enemies.

- The assignment is due in its entirety on week 8 but students are required to submit at least the code for the task manager and base task class by week 7.

Week 6: Service Locator

Lecture on Service Locator
Workshop

ASSIGNMENT: Complete Boss

- Finish implementing boss.
- Submit assignment by committing work to project's Git repository.

Week 7: State

Present assignments
Lecture on State Machines
Workshop
Review assignment requirements

ASSIGNMENT: Basic AI Enemy

- Implement state manager
- Using states implement an enemy that transitions between the following states:
  - Seeking: Enemy will move slowly towards the player. If it comes within X distance it will enter the attack preparation state.
  - Attack Preparation: Enemy stops moving and pulses 5 times. If it is hit while pulsing it will enter the fleeing state. If not, it will enter the attack state.
  - Attack: Enemy moves quickly in a straight line towards the player for X distance. If it hits the player it will explode for Y damage. Otherwise it will enter the seeking state.
  - Fleeing: Enemy picks a random position X distance away from player and moves quickly towards it. When it reaches the position it enters the seeking state.
- Submit assignment by committing work to project's Git repository.

Week 8: Behavior Tree

Present assignments
Lecture on Behavior Trees
Workshop
Review assignment requirements

ASSIGNMENT: Reimplement AI

- Reimplement enemy AI using behavior trees.
- Submit assignment by committing work to project's Git repository.

Week 9: Modes/Scenes

Present assignments
Lecture on Mode/Scene pattern
Workshop
Review assignment requirements

ASSIGNMENT: Intro and Game Over Screens

- Using the mode pattern implement the following screens:
    - Game Intro: A screen that displays:
        - Player ship (extra credit if it has some form of AI control)
        - One or more enemies (extra credit if they have some form of AI control)
        - Play button. Pressing this button starts the game.
    - Game: This is the game itself (i.e. what you had prior to the assignment)
    - Game Over: A screen that displays:
        - The final score
        - OK Button. Pressing this button returns the player to the Game Intro screen.
- Submit assignment by committing work to project's Git repository.

Week 10: Command

Present assignments
Lecture on Command
Review and discuss options for student designed feature #1
Workshop

ASSIGNMENT: Spec Student Project #1

- Each student is responsible for producing a one-page document specifying the requirements for a feature they will implement in the game the following week.
- The proposed feature should be a non-trivial addition to the game and focus on game logic rather than art or design. For example, modifying the enemy manager to allow for dynamic difficulty adjustment, or enemies that coordinate with each other would be a good fit for the assignment, while adding animated cut-scenes would not.

Week 11: MVC (Model/View/Controller)

Lecture on MVC
Workshop

ASSIGNMENT: Implement Student Project #1

- Implement all the features specified in the previous week's assignment.
- Submit assignment by committing work to project's Git repository.

Week 12: Test Driven Development

Lecture on TDD
TDD demonstration
Workshop

ASSIGNMENT: Spec Student Project #2

- Each student is responsible for producing a one-page document specifying the requirements for a feature they will implement for week 14.
- Just like the previous week's assignment, implementing the proposed feature should primarily require programming and not art and design. Since there is twice as much time provided for completing this assignment, it should be substantially more ambitious than the previous week's assignment.

Week 13: Review

Review techniques and patterns discussed during course
Status report on assignment
Workshop

ASSIGNMENT: Work on Student Project #2

- Continue implementing student project #2.
- Submit assignment by committing work to project's Git repository.

Week 14: Final

Present final assignments
Discuss patterns covered in class, situations where they were or were not helpful.
Discuss additional resources and potential improvements for the class.

ASSIGNMENT: Finish Student Project #2

- Finish implementing student project #2.
- Submit assignment by committing work to project's Git repository.