CS 2340 Milestone 2 (M2)

New Game Configuration, Domain Modeling, Use Cases + SSDs

Due February 20th

Background

Over the next five milestones, you will design and build a Tower Defense game. You'll be responsible for implementing some of the functionalities to play the game. The player will be able to buy and upgrade towers and place them onto the game map. Enemies will spawn and travel along a designated path to attack the player's monument. The player's towers will attack the enemies to defeat them to protect the player's monument. The ultimate goal of the game will be for the player to survive long enough to fight and defeat the final boss. Future milestones will build upon your work and detail which new features to add. For more detailed requirements, see the project outline.

Purpose

There are two primary goals for this project. The first goal is to provide you with experience collaborating with a team to develop a product with specific requirements. The second goal is to increase your aptitude with several key software engineering principles and technologies, namely SOLID, GRASP, Design Patterns, TDD, and Git. Over the course of the project, these concepts will be introduced to you so that they can be incorporated into your future milestones. With each project milestone after the first, testing will occur alongside development, and you will be responsible for writing unit tests to verify your implementation's functionality.

Task

For this milestone, you are asked to create and submit to Canvas two design deliverables in addition to the first portion of your application and its accompanying tests. For the implementation portion of this milestone, you will create three screens: a welcome screen, a player configuration screen, and an initial game screen. Other than the requirements outlined below, the details of your implementation are up to you. Your app implementation and functionality will be graded during a demo which will occur the week after milestones are due.

Design Deliverables

There are two design deliverables that are required in this milestone: a Domain Model and a set of SSDs. These are due as a combined PDF on **Sunday, February 20th at 11:59pm via Canvas submission.**

Domain Model

- Identify and list at least ten potential nouns which could be used in your project.
  - Example: Player, Enemy, Tower
- Sort the ten nouns into two categories
  - Game Objects (classes): require their own methods, associations, and attributes.
    - Example: Player, Enemy
    - A minimum of 5 nouns must be Game Objects.
  - Attributes: do not require a whole class.
    - Example: Price, Difficulty

- Draw a Domain model for the nouns you brainstormed.
- Connect each class within your Domain Model with at least one other class using associations.
    - Example: Player "owns" Tower
- Include multiplicities for each association, one on each side of the association.

The submission of the Domain Model itself suffices for the inclusion of the listed and categorized nouns.

Use Cases and System Sequence Diagrams

- Identify a unique use case per group member for the semester-long project.
    - Example: "Buy Tower"
    - Use the background section of this document to inform your use cases.
- Model this action as a System Sequence Diagram.
    - Each System Sequence Diagram must have at least one ALT, LOOP, or OPT frame.

When submitting your System Sequence Diagram, make sure the title on the diagram is the corresponding use case for it. The submission of the diagram with that title will count for both the use case and the SSD.

Project Setup

Milestone 2 through Milestone 6 all incorporate this semester-long project, therefore it is imperative that your set up and utilize your repository in the correct manner.

- The project must be implemented as either a JavaFX desktop application or an Android mobile application. Alternate implementation backings must be run by the professor and head TA, and TAs may not be able to answer questions or support projects with alternative backings.
    - Recommended IDE's for development are IntelliJ and Android Studio, respectively.
    - Any external libraries must be cleared with the TAs by making a public post on Piazza so that others may know of this library as well.
- Your project must be hosted on the GT GitHub.
    - Create a new repository within your organization for this application. You can name your project however you want (the more fun the better). *Do not use the repository you set up for M1.*
    - We highly recommend including a .gitignore file for your project. This allows you to exclude certain files that shouldn't be tracked (e.g., ".DS_STORE", "local.properties"). For more information, see the following example gitignores:
        - https://github.com/github/gitignore/blob/main/Android.gitignore
        - https://github.com/akshaybabloo/JavaFX/blob/master/.gitignore
    - We also recommend looking at the Milestone 1 repositories to get an idea of where your Git top-level directory should generally be.
        - With IntelliJ and Android Studio, you can create a new project within the empty repository, and everything should be set up correctly.

Implementation

In this milestone, you will be creating the new game sequence, including configuration and customization. The following are the requirements:

- Starting the application will open to a welcome screen for the application which:

    - Has some way to start the game (e.g., a start button).
    - Has some way to quit the game within the application (e.g., a quit button).
        - The user closing the window does not count as a method to quit.
- Starting the game should take the player to an initial configuration screen which:
    - Requires input for the player to enter their name.
        - Players should not be allowed to pass in an empty, null, or white-space only name.
    - Requires the player to choose their difficulty from at least 3 different options.
    - Allows the player to continue to the game screen after they input both a valid name and chosen a difficulty.
        - The player must be able to visually see the name and difficulty they have chosen *before* they are allowed to move to the game screen.
- The game screen should display textually or graphically the first map the player will see. This is where most of the functionality in later milestones is implemented. For now, the screen must:
    - Display starting money.
        - Starting money should vary based on difficulty.
    - Display/describe a distinct path over which enemies will travel in future milestones.
    - Display/describe a monument at the end of the path which in future milestones will be attacked by monsters and protected by towers.
    - Display/describe monument health.
        - Monument health should vary based on difficulty.

Any ambiguity or extension is allowed for the group to customize their project with. Similarly, specific nouns (e.g., money, monument) may be replaced by some other thematic noun (e.g., energon cubes, moon base).

Testing Requirement

This milestone does not have a testing requirement, though subsequent milestones will. Create and develop your project with unit testing in mind. It may be wise to write some unit tests for the implementation requirements to practice for future milestones. Some example unit tests that one could write off of Milestone 2 requirements are:

- Detection of whitespace-only, null, and empty names
- The starting money and monument health of each difficulty should each differ

Checkstyle

During your demo, you will be required to run he checkstyle.py script (located on Canvas under "Files" > "checkstyle" > "Java Guide.pdf"). This script will give your project a score out of 10 and will account for 10 points of your final milestone grade. Be sure to run the checkstyle script prior to submission to avoid unforeseen deductions. *You must run the* ***python script*** *during your demo. Running the check within IntelliJ/Android Studio does not count.*

Milestone Tagging

Tags are a way of marking a specific commit and are typically used to mark new versions of software. To do this, use “git tag” to list tags and “git tag –a tag name –m description” to create a tag on the current commit. Important: To push tags, use "git push origin –tags". Pushing changes normally will not push tags to GitHub. Although not explicitly graded, you will be asked to checkout this tagged commit during demo. This is done with “git fetch --all --tags” and “git checkout tag name”.

Summary

You will be graded according to the successful and correct completion of the design deliverables and implementation requirements above. Groups are required to demo to receive credit for the features they have implemented. This will be done during TA office hours after the due date of the design deliverables.

You will submit your design deliverables as a combined PDF via the Canvas assignment. You should also include a link to your Georgia Tech GitHub project repository. Your repository must be set to private. Points may be deducted if these guidelines are not followed.