CS 2340 — Milestone 5: Project Iteration 4

Tower Defense Combat, Money, and DCD

Due November 14th

BACKGROUND: For the fourth iteration of the project, your team will implement the tower defense combat and the ability to gain money. When enemies come within proximity to a tower, the tower should attack the enemy to defend the monument. Different types of towers and enemies provide gameplay variety for the player. The player gains money throughout gameplay to buy more towers to provide a more robust defense for the monument.

PURPOSE: Design class diagrams allow you to map the structure of your application through the specification of its software classes and their relationships.

TASK: For this milestone there is one design deliverable, one testing deliverable, a required feature set, and an extra-credit opportunity. The details of the implementation requirements are outlined below. Your app implementation/functionality will be graded during a demo.

Design Deliverable: Design Class Diagram

1. Create a design class diagram for your implementation from M4 that includes the following:
    - Classes
    - Attributes with basic types
    - Operations
    - Visibility marks
    - Associations with multiplicities
    - Inheritance and Interface notation where necessary
2. Your DCD should include any classes necessary to implement enemies and the tower's ability to attack enemies.
3. You do not need to include UI classes or getters/setters in your DCD.

Implementation Requirements

1. Implement **tower defense combat**
    - At least one type of tower should attack enemies when they enter the tower's proximity, and if a tower type does not attack enemies, it should have some distinct gameplay behavior that benefits the player.
    - Enemies should *visibly (or descriptively)* lose health when attacked by a tower.
    - Each different tower should have some distinct gameplay behavior (e.g., differing damage, speed, range, area of effect)
    - Each different enemy should have a distinct gameplay effect (e.g., differing travel speed, damage to monument, different levels of enemy health)
    - The player should be able to interact with the game as enemies come in, whether by placing new towers, manipulating tower position, or in some other way.
2. Implement **money gain**
    - There should be some method during normal gameplay by which the player can gain money to buy more towers (e.g., from enemies, over time)

Testing Requirements

1. Write **unit tests** to verify the functionality of the newly implemented features.
    - There is no code coverage requirement, but you should make sure that your unit tests cover meaningful functionality of the implementation requirements.
    - ***Each*** *team member should add at least* ***two*** *unit tests.*
2. **Testing Deliverable:** Include with your submission a brief writeup describing your testing process for the milestone. Explain which components were chosen for testing and why. Additionally, explain how your tests verify that the code functions as expected. This writeup should be present on your GitHub repository, so that you can edit it up to your demo.

Checkstyle

During your demo, your team will be required to run the **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 M5 grade. Be sure to run the checkstyle script prior to submission to avoid unforeseen deductions.

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. *You will be required to checkout this tagged commit during demo.* This is done with **"git fetch --all --tags"** and **"git checkout tag_name".**

Submission Requirements

In addition to your diagrams, ensure that you include a link to your GitHub repository in your submission. Also, ensure that you have added your grading TA(s) and the professor as collaborators so that they may view your private repository. **Repositories must be located on the Georgia Tech GitHub and must be set to private!** Points may be deducted if these guidelines are not followed!

CRITERIA: You will be graded according to the successful and correct completion of the design deliverables, implementation requirements, and testing requirements above. **Groups are required to demo in order to receive credit for the features they have implemented.**

Extra Credit Requirements [+15 points]

Choose one of the following types of diagrams:

- Domain Model
- System Sequence Diagram
- Sequence Diagram
- Robustness Diagram
- Design Class Diagram

Enter your team's number into one of the available slots here: Extra Credit: Diagram

For the chosen type of diagram, provide the following:

- Problem description
- Rubric
- Solution in draw.io format

The extra credit requirements are worth up to 15 extra points in total.