CS 2340 Sprint 4

Enemy Movement, Collisions, Game Over Screen, DCD, Observer Design and Factory Design
Due Date: See Canvas assignment

Background

For the iteration of this project, your team will be implementing some of the core functionalities of the 2D Dungeon Crawler game: enemy, enemy movements, player-enemy collisions and game over screen.

Purpose

This sprint expands upon the work up until Sprint 3. Your team will be able to work together to implement enemy, enemy movements, player-enemy collisions, and game over screens. Working on your Design Class Diagram (DCD) will help you plan out how to represent the additional features in this sprint into your system in a software-oriented manner.
Also, following software engineering principles by incorporating design patterns will allow your team to learn the importance of logical code organization, especially for larger-scale projects. Testing will occur alongside development, and you will be responsible for writing unit tests to verify your implementation's functionality.

Tasks

For Sprint 4, you are asked to create and submit to Canvas multiple design deliverables along with demoing the functionality implemented. For the design deliverables, you will perform object-oriented design (OOD) with the new sprint features. This will include updating your Design Class Diagram (DCD) using the Sprint 4 features and providing evidence of your team using two design patterns. For the implementation portion of this sprint, you will make distinct types of enemies with different movement patterns and handle player-enemy collisions. Other than the requirements outlined below, and that you must use Java (not Kotlin), the details of your implementation are up to you. Your application implementation and functionality will be graded during a demo, which will occur the week after sprints are due; see the class schedule for specific dates.

Pre-Sprint Agreement

Unlike previous sprints, we will be requiring you to create an itemized list of tasks to be completed by the team and assign them to individuals. This list should be submitted as a document to the Canvas assignment for your section. This list can contain additional requirements such as a completion date for the task or subtasks, if you so wish.

The most important element to this assignment is agreeing as a team to the work to be done by everyone. We will only be checking this assignment for completion but may refer to it in the future if there are any issues with team members not pulling their weight. Ensure that the work distribution is fair and agreed upon by everyone on the team. **This is due by Wednesday, October 23.**

Design Deliverables

There are two sets of design deliverables required in this sprint: a Design Class Diagram (DCD) and evidence of two design patterns. These are due **as a combined single PDF via Canvas submission (check**

course page for official date). **For diagramming, we require the use of an online diagramming tool like draw.io. Handwritten submissions will not be accepted.**

Revised Design Class Diagram

1. Update the previous design class diagram (DCD) to include the features from this sprint (an example might be enemies).
    a. You should include at least 2 new classes and their associated attributes
    b. Highlight or indicate the new classes in some way
2. Your DCD should include the following:
    a. Classes
    b. Attributes with basic types
    c. Access modifiers
    d. Operations/Functions
    e. Associations with multiplicities
    f. Relationships: Generalization/Aggregation/Composition/Dependency
3. Your diagram should include at least any classes necessary for the following:
    a. Displaying the enemy, their types and movement patterns
    b. Displaying player-enemy collisions.
    c. Displaying the game over screen and leaderboard.
4. **You do not need to include UI classes or getters/setters in your DCD**.

Design Pattern Evidence

1. Submit screenshots of your code abiding by the Observer pattern for the implementation of the player-enemy collision.
2. Submit screenshots of your code abiding by the Factory pattern for the implementation of the different types of enemies.
3. The screenshots should capture a clear view of how the design patterns were used for their implementation, including all relevant classes.
4. Write 2 paragraphs, one describing how your code properly utilizes the factory design pattern and another describing how it properly utilizes the observer design pattern.

Implementation

In this Sprint, you will be creating different types of enemies using Factory Design Pattern, implementing enemy movements, handling player-enemy collisions using Observer Design Pattern, and creating a Game Over Screen with a leaderboard. The following are the requirements:

1. Implement enemies using Factory Design Pattern
    a. There should be 4 different kinds of enemies for the game, with each room having at least 2 different kinds of enemy
    b. Different enemies should differ in at least 2 attributes i.e., enemy sprite, enemy movement speed, enemy size, etc.
2. Implement enemy movement
    a. The movement should be visually displayed
    b. The enemy should not be able to move off the screen
    c. Different types of enemies must have different movement patterns (could be speed or direction, etc.)
3. Implement Enemy and Player collision using Observer Design Pattern.

    a. All enemies must be updated with the player's position whenever player makes a movement
    b. Upon update of player movement, all enemies must check if they make direct contact with the player i.e., collision
    c. Upon collision, player health points (HP) is reduced based on the difficulty level selected. Damage must differ with difficulty
    d. Player HP must be updated and displayed in real time
4. Implement Game Over Screen
    a. Have an ending screen to show the player that they have lost and that the game is over
        i. You can either modify the existing ending screen in Sprint 3 to display an alternative graphic if the player loses or implement a completely new screen with the same functionalities and features as the first ending screen
    b. Automatically navigate to the game over screen if player health (HP) reaches 0 i.e., player dies
    c. The win screen and the game over screen must have the same features (restart button, leaderboard, etc.)

As mentioned in previous sprints, artistic liberty is permitted for groups to customize their project theme. You are also free to add functionality as long as you meet the requirements.

Testing Requirement

Starting this sprint, your team will be required to create 2 unit tests per team member. Make sure you create and develop your project with unit testing in mind. For Sprint 4, you must construct unit tests that test functionalities from Sprint 4. Some example unit tests that you could prepare from Sprint 4 requirements are:

1. Player and enemy collision are handled correctly.
2. The player's health is updated after collision.
3. The enemy cannot move past the coordinates of a wall.

For instructions on setting up JUnit 4 and running test cases, please refer to the Sprint 2 document.
For further knowledge on Android testing, refer to the documentation.
Note: The UI of your team's game should be separate from the game's functionality/logic. These tests should not be composed of UI mocking, but instead, your team's tests should be validating your game's logic and event handling. **If you create mock test cases, you will receive 0 credit for those tests.**

CheckStyle

For checkstyle requirements for this sprint, you can refer to the details mentioned in Sprint 1 for the set up. Make sure you follow the rules for the correct styling of your project's code.

Sprint 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 asked to checkout this tagged commit during demo. This is done with "git fetch --all --tags" and "git checkout tag name". **You will be required to pull this tag during the demo. If you forget to tag your commit or tag your commit late, you will be penalized with –5 points for each day late.**

Additional Resources

Under this section are some resources we think will help you and your team streamline the game creation process. If you have any questions about these resources, contact your TAs through Ed or office hours.

1. TA website: https://github.gatech.edu/pages/gtobdes/obdes/
2. Ed: https://edstem.org/us/courses/42802/discussion/3386189
3. Android Studio Video (located on your section's canvas lecture page).

Demos

Your implementation of sprint features will be graded through an in-person TA demo. Demos will be held in CCB 267 the week after the sprint is due. Please sign up for a demo slot in Microsoft Bookings. Note that not all demo slots may be open at sprint release.

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. TA demos will be able to be booked through Microsoft Bookings**.** You will submit your design deliverables as a combined PDF via the Canvas assignment. **You should also include a link to your GitHub project repository**. Your repository must be set to private. When you sign up for a demo, ensure that the shared TA GitHub account has been added to your repository **and you have tagged the code you intend to demo.** Points may be deducted if these guidelines are not followed.

Academic Integrity

Note on Plagiarism and the Use of AI Tools

We understand that technology and tools, including Artificial Intelligence platforms, have become readily accessible and can be valuable in various scenarios. However, students must be cautious about their usage in academic settings.

Please refer to the course policy regarding the use of AI tools for your projects. Using AI to generate or copy content for your project is not considered collaboration. Leveraging AI to produce work that you present as your own, without proper citation, is likely crossing the line into academic misconduct. It's essential to maintain integrity in all academic undertakings. If in doubt, always consult the course guidelines or reach out to the instructor on Ed Discussion for clarification.