CS 2340 Sprint 3

Player Movement, Collisions, DCDs, SDs, and Design Patterns
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: player movement and collisions. Players should be able to move four-directionally, move to the next room when reaching an exit, and not be able to move through walls. The ending screen should also be updated to be reached when the player reaches the end as well.

Purpose

This sprint expands upon the work started in Sprint 2. Your team will be able to work together on creating the basic game screens for the project, where future sprint requirements can be implemented.

When designing the Sequence Diagrams (SDs), your team will learn how important analysis is when developing large systems. Diagrams like SDs are very helpful tools for making decisions about how to inspect your code. Additionally, creating a Design Class Diagram (DCD) will help you plan out how to represent your system in a more software-oriented way than the Domain Model you previously made.

Also, following software engineering principles like adhering to a specific architecture for application development and 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 3, 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 analysis (OOA) and object-oriented design (OOD) with the new sprint features. This will include updating producing a set of Sequence Diagrams (SDs), a Design Class Diagram (DCD), and providing evidence of your team using two design patterns. For the implementation portion of this sprint, you will make your player able to navigate through the rooms that you have already created so they can reach exits and complete the game. 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.

Design Deliverables

There are three sets of design deliverables that are required in this sprint: a Design Class Diagram (DCD), three Sequence Diagrams (SDs), 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.**

Design Class Diagram (DCD)

1. As a group, create a DCD for your project based on your implementation. This can be created from existing and from future features your team plans to add.
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 game screen and any game rooms
    b. Displaying and moving the player
    c. Displaying the player's score
    d. Displaying the leaderboard
4. **You do not need to include UI classes or getters/setters in your DCD**.

Sequence Diagrams

1. Create three use cases from previous sprints or Sprint 3 requirements and turn each of them into a UML Sequence Diagram.
    a. Although you are not required to submit your use cases, it may be helpful to write them out in a brief or casual format.
    b. Please provide a title for each use case / SD to help explain the SD.
    c. Make sure each SD is properly modeled using the conventions of a UML diagram.
    d. Collect all 3 use cases and SDs and append them to the pdf used for the DCD.
2. Each diagram must contain at least one ALT, LOOP, OPT, or Ref structure, so choose your user stories carefully!
3. For each sequence diagram, ensure that it illustrates the dynamic interactions between the user of the user story and your application's classes.
    a. Make sure that the system calls in your SD correspond logically to the DCD you created. **There must be conceptual integrity between your DCD and SD.**
    b. Keep in mind which class stores each method and how that is represented on the Sequence Diagram.
    c. You should disregard UI while designing the SD for this sprint since the actor's interaction with the user interface is no different from directly interacting with the system.

Design Pattern Evidence

1. Submit screenshots of your code abiding by the Strategy pattern for the implementation of the player's movement.
2. Submit screenshots of your code abiding by the Observer pattern for the implementation of the wall collisions.
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 strategy design pattern and another describing how it properly utilizes the observer design pattern.

Implementation

In this sprint, you will be implementing player movement, room entering functionality, and game ending scenarios. In addition to the implementation of the requirements, you will be required to demonstrate the use of the Strategy and Observer patterns in your code.

1. Implement movement.
    a. The player should be able to move left, right, up, and down.
    b. The movement should be visually displayed.
    c. The player should not be able to move off the screen.
    d. Players should be able to move to different rooms by reaching an exit.
        i. Remove temporary next button from last sprint to navigate to different rooms.
    e. You should use the strategy design pattern for movement and the observer design pattern for separating movement logic from rendering.
2. Implement wall collisions.
    a. Character should not be able to pass through walls in the dungeon.
3. Update ending screen.
    a. Remove the temporary next button to navigate to the ending screen.
    b. Automatically navigate to the ending screen if player successfully reaches the final exit.
    c. Visually update the ending screen to show the player that they have won. In future sprints, you will implement an alternate ending screen for when the player loses.

As mentioned in Sprint 1 and Sprint 2, artistic liberty is permitted for groups to customize their project theme. For example, the game can be implemented in a Sci-fi theme where the player traverses through rooms such as the medical bay, helm, and crew quarters of a spaceship. You are also free to add functionality as long as you meet the base 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 3, you must construct unit tests that test functionalities from Sprint 3. Some example unit tests that you could prepare from Sprint 3 requirements are:

1. The player movement direction corresponds to the desired user input.
2. A player 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/
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.