2805ICT/7805ICT System and Software Design

3815ICT Software Engineering

Assignment Specification: Milestone 1

Assignment Title: Enhanced Tetris Game Development

Submission: Milestone 1

Due Date: End of Week 5 (25/08/2024)

Objective

The primary objective of this assignment is to utilize Software Engineering principles and the Java programming language to develop an enhanced version of the classic Tetris game. This milestone will focus on the initial development stages, ensuring a solid foundation for the final product.

Background

Tetris is a classic puzzle video game originally designed and programmed by Alexey Pajitnov in 1984. The game involves players manipulating tetrominoes, shapes composed of four square blocks each, which descend onto a playing field. The goal is to fit these shapes into complete rows, which then disappear, earning points for the player. The game ends when the tetrominoes stack up to the top of the playing field and no more shapes can enter.

General Functions

To understand the general requirements and functionalities expected in the enhanced Tetris game, please study the provided demo video. This will give you a comprehensive understanding of the gameplay mechanics, user interface, and additional features required for your implementation.

Submission

The submission for this milestone is a Word or PDF document based on the template provided on the course website. The document must include the following:

1. **GitHub Project Link**:
    - Provide the link to your GitHub project repository.
    - Ensure that you have invited the course convenor to your project so they can access it:
        - GC students, please ask Elizabeth for her GitHub account.
        - Online and Nathan students, please add `LarryAtGU` as a team member in your GitHub project.
2. **Demo Video Link**:
    - Include a link to a video that demonstrates your game.

    - Detailed instructions for the video will be provided later in this specification.

Note: Each group only needs to make one submission. A student in the group will submit the document on behalf of the entire group.

Requirement and Marking Criteria

1. Project Plan (5 points)

Students are required to provide a comprehensive project plan. The project plan should follow the provided template and include the following elements:

- **Task Definition and Assignment**:
    - Clearly define all tasks necessary for the completion of the project.
    - Assign tasks to individual group members.
- **Time Estimation and Tracking**:
    - Estimate the time required to complete each task.
    - Record the actual time spent on each task.
    - Include the planned task finish date and the actual finished date.
- **Group Meetings**:
    - Document the frequency of group meetings.
    - Specify the software tools used for project management and communication.
- **Effort Summary Table**:
    - Complete the effort summary table to provide an overview of the work distribution and contributions of each group member.

2. GitHub Repository (5 points)

Students are required to provide the link to their GitHub repository. Ensure that the course convenor has access to the repository:

- GC students, please ask Elizabeth for her GitHub account.
- Online and Nathan students, please add `LarryAtGU` as a team member in your GitHub project.

Additionally, include a screenshot of the commit history in your submission. Ensure the screenshot is clipped so that the text is easy to read. Marks will be deducted for the following:

- Poor image resolution.
- Too few commits.
- Insufficient explanations in commit messages.
- Most commits occurring within a very short period.
- Too few group members contributing to the code.

3. Functional Requirements (10 points)

Based on the demo video and your study of Tetris games, provide a comprehensive list of the functional requirements for the system. Note that these functional requirements pertain to the completed product, not just the Milestone 1 submission. Ensure the quality of your functional requirements by avoiding incompleteness, inconsistency, ambiguity, and redundancy. Each functional requirement must be testable and should follow the precondition, event, postcondition, and constraints pattern (this will be explained in the lecture).

Key points to address:

- **Comprehensive Functional Requirements**:
  - Define all necessary functional requirements for the final product, including those not yet implemented in Milestone 1.
- **Scoring System**:
  - Detail how scores are calculated.
- **Game Levels**:
  - Define the criteria for advancing levels and how different levels affect gameplay.
- **Gameplay Screen**:
  - Specify additional information to be displayed, such as upcoming tetrominoes, current score, etc.
- **Level Advancement**:
  - Define how levels will automatically increase based on gameplay criteria.
- **Shortcut Keys**:
  - Define how shortcut keys will be used to enable music and sound effects.
- **Extended Mode**:
  - Specify additional features or modes that extend the basic game functionality.

Additional Requirements:

- **Unique IDs**:
  - Each functional requirement must have a unique functional requirement ID.
- **Proper Ordering and Grouping**:
  - Functional requirements should be properly ordered and grouped to enhance readability and organization.

Ensure that each functional requirement is clear, concise, and testable. Here are some examples of how to format functional requirements:

Example Functional Requirement:

- **ID**: FR-001
    - **Title**: Start to play
    - **Precondition**: The game is started and main screen is display.
    - **Event**: User click the play button.
    - **Postcondition**: The game will change to play screen, and a new game will be automatically started.
    - **Constraints(Optional)**: The new game will be started in less than 1 second and based on the configuration setting.

Please ensure the functional requirements are well-documented and include all aspects of the game as observed from the demo and your study.

4. Non-Functional Requirements (5 points)

Based on the FURPS+ model, create at least one non-functional requirement in each category. Each non-functional requirement should have a unique ID, must be testable, and should clearly indicate the non-functional requirement category (e.g., usability, security).

- **ID**: NFR-001
    - **Category**: Usability
    - **Description**: The difficulty levels of the games must be in a range that can be accepted by typical players.
    - **Testable Criteria**: Conduct user testing to play the game in all the difficulties levels and collect their feedback.

5. Use Case Diagram (10 points)

Based on the functional requirements, create a use case diagram. The use case diagram must comply with the UML use case diagram notation.

Key points to address:

- **Services to the User (Actor)**:
    - The use case diagram should describe what services the system will provide to a user (actor).
    - Do not include implementation details in the use case diagram.

Example:

- **Use Case**: "Play Game"
    - This is a use case that describes the service of playing the game.
    - Do not include specifics such as "press which key for what effect" in the use case diagram.

Guidelines:

- Ensure that all important use cases are included.
- Avoid missing significant use cases that the system should provide.
- Do not include implementation details in the use case diagram.

6. Activity Diagram (5 points)

Draw an activity diagram to illustrate a typical scenario of playing the game. The activity diagram should include the following activities:

- **Start the Program**: The initial activity when the user launches the game.
- **Play Game**: The core activity where the user engages with the game.
- **Finish the Game**: The activity representing the end of a game session.
- **Check the Top Score**: An optional activity where the user checks the high scores.
- **Exit the Program**: The final activity when the user closes the game.

Guidelines:

- Ensure the activity diagram follows standard UML notation.
- Clearly represent the flow of activities and decisions.
- Use appropriate symbols for actions, decisions, start, and end points.

Requirements:

- **Correct Use of UML Notation**:
  - Use standard UML symbols to represent activities, decisions, start, and end points.
- **Comprehensive and Clear Representation**:
  - Include all specified activities and ensure the flow is logical and easy to follow.

7. Peer Review (5 points)

Based on the provided template, complete the peer review process. Every group member must perform a self-review as well as peer reviews for all other members.

Key points to address:

- **Self Review**:
  - Each member must assess their own contributions and performance.
- **Peer Reviews**:
  - Each member must review the contributions and performance of all other group members.

Requirements:

- **Submission Document**:
  - The submitted document should contain:
    - Each member's self-review report.
    - Each member's peer review reports for all other group members.
    - A summary of the peer review findings.

Guidelines:

- **Use the Template**:
  - Ensure that all reviews are completed using the provided template.
- **Comprehensive and Fair Assessment**:

    - Provide detailed and fair assessments of each member's contributions.
    - Highlight strengths and areas for improvement.

8. Reflection (Master Students Only)

If your group includes master students, each master student must provide a one-page reflection on their learning experiences from this project and suggest ways to improve the work in the future.

Requirements:

- **Content of the Reflection**:
    - Describe what has been learned from participating in this project.
    - Identify areas where the work could be improved in future projects.

Guidelines:

- **Length and Detail**:
    - The reflection should be one page in length.
    - Provide thoughtful and detailed insights into your learning and improvement suggestions.

Note:

- **No Marks for This Section**:
    - This section does not carry any marks.
- **Penalty for Non-Compliance**:
    - Failure to provide a reflection or submitting a poorly written reflection may result in a deduction of up to 10 points from the overall project score.

9. Video Quality (5 points)

Students are required to submit a video to demonstrate the functionality of the game. The video can be hosted on YouTube or another platform; therefore, the report should only contain the link to the video.

Requirements:

- **Duration**:
    - The video should be no longer than 7 minutes. Videos longer than 7 minutes will not be watched or marked.
- **Content**:
    - The student should narrate the video, introducing and explaining all the functions of the game.
- **Quality**:
    - The video quality should be high enough to ensure that the text and visuals in the game are clear and easily visible to the assessor.

10. Splash Window (5 points)

The video should start from the startup of the game, showcasing a splash window as demonstrated in the demo.

Requirements:

- **Splash Window Design**:
    - You may design your own images for the splash window.
    - The splash window should include the identity of your group, the course code, and any other relevant information.
- **Display and Duration**:
    - The splash window should be displayed in the middle of the screen.
    - It should last for a few seconds before disappearing and transitioning to the main game screen.

11. Main Screen (5 points)

After the splash window disappears, the game should display the main screen centered on the screen.

Requirements:

- **Main Screen Design**:
    - The main screen should contain four buttons:
        - **Play**
        - **Configuration**
        - **High Scores**
        - **Exit**
    - You are encouraged to add artistic design elements to the main screen to enhance its visual appeal.

12. Configuration Screen (5 points)

In the main screen, clicking the Configuration button should replace the main screen with the configuration screen.

Requirements:

- **Configuration Screen Elements**:
    - The configuration screen should include the following settings:
        - **Field Width**: Display the current field width and allow adjustment.
        - **Field Height**: Display the current field height and allow adjustment.
        - **Game Level**: Display the current game level and allow adjustment.
        - **Music On/Off**: Toggle music on or off.
        - **Sound Effect On/Off**: Toggle sound effects on or off.
        - **AI Play**: Enable or disable AI play mode.
        - **Extend Mode**: Enable or disable extended mode.
    - There is a back button at the bottom of this screen, click it will bring back to the main screen.
- **Functionality**:

    - All components on the configuration screen must be functional and reflect the current settings.
    - Default values and valid ranges for all settings should match those shown in the demo.
    - The settings do not need to be saved or applied to the gameplay in this milestone.

13. High Score Screen (5 points)

In the main screen, clicking the High Scores button should navigate to the High Score screen.

Requirements:

- **High Score Screen Elements**:
    - The screen should list the top 10 scores.
    - Each score entry should include:
        - **Player Name**
        - **Score**
    - Include a **Back** button at the bottom of the screen to return to the main screen.
- **Data**:
    - The displayed data on this screen are dummy data for this milestone.
    - You may either hard-code these scores or generate them randomly.

14. Game Play (10 points)

In the main screen, clicking the Play button will navigate to the play screen where the game of Tetris automatically starts.

Requirements:

- **Play Screen Elements**:
    - The field size should be 10 x 20.
    - A randomly generated tetromino will start moving from the top of the field downward.
    - The movement of the tetromino should be smooth, not jumping from one row to the next.
- **User Controls**:
    - **Left Arrow**: Move the tetromino left.
    - **Right Arrow**: Move the tetromino right.
    - **Down Arrow**: Speed up the dropping of the tetromino. If the down arrow is released before the tetromino hits the bottom, it should return to normal dropping speed.
    - **Up Arrow**: Rotate the tetromino 90 degrees clockwise.
- **Collision Handling**:
    - When a tetromino hits the bottom of the field or another settled tetromino, it should settle into place.
    - Ensure that moving left, right, down, or rotating does not cause any part of the tetromino to go outside the field boundaries or overlap with any other settled tetrominoes.

15. Erase Full Rows (5 points)

During gameplay, if a row of the field is fully occupied by tetrominoes, the row should be erased and all settled blocks above it should move down by one row.

Requirements:

- **Full Row Detection**:
    - Detect when a row is fully occupied by tetrominoes.
- **Row Erasure**:
    - Erase the fully occupied row and move all settled blocks above it down by one row.
- **Multiple Row Erasure**:
    - Handle cases where multiple rows become fully occupied and need to be erased simultaneously.
    - Demonstrate the erasure of multiple rows in your video.
- **Color Maintenance**:
    - Each type of tetromino should have a distinct color.
    - Settled blocks and blocks that are moved down after row erasure should maintain the color of their original tetrominoes.

16. Pause Function (5 points)

When the game is running, which means there are still tetrominoes moving on the screen, pressing the P button should pause the game.

Requirements:

- **Pause Functionality**:
    - Pressing the P button should pause the game, stopping all tetromino movement.
    - When the game is paused, display a message on the field indicating that the game is paused and instructing the player to press P again to continue.
- **Resume Functionality**:
    - Pressing the P button again should resume the game, continuing the tetromino movement from where it left off.

17. Back from Play Screen (5 points)

The play screen should include a Back button at the bottom with the following functionalities:

Requirements:

- **Back Button**:

    - The Back button should be located at the bottom of the play screen.
    - When the game is over (i.e., the generated tetromino cannot move), clicking the Back button should return to the main screen.

- **In-Game Back Button Behavior**:

    - If the game is running, clicking the Back button should pause the game and display a confirmation dialog asking the user if they want to stop the game.
    - If the user clicks **Yes** in the dialog, the game stops, and the user is returned to the main screen.
    - If the user clicks **No** in the dialog, the game continues.

- **Paused Game**:

    - If the game is paused before the user clicks the Back button, clicking the Back button should maintain the paused status when returning from the confirmation dialog.

- **Running Game**:

    - If the game is running normally when the user clicks the Back button, it should return to its normal running state after the confirmation dialog if the user chooses not to stop the game.
    - You should demonstrate all the situations in your video.

18. Exit the Program (5 points)

In the main screen, clicking the Exit button should bring up a dialog box to confirm if the user wants to exit the program.

Requirements:

- **Exit Confirmation Dialog**:

    - Clicking the Exit button on the main screen should display a confirmation dialog box.
    - The dialog box should ask the user to confirm if they want to exit the program.

- **Dialog Options**:

    - **Yes**: Exits the program.
    - **No**: Returns to the main screen.