Milestone 1– Architecture and Testing Framework

Tips: Learn Visio Now! There are downloadable UML templates to make your job easy and professional looking.

Objectives:

1. Develop the model implementation that supports:
    a. Supports initialization of the game's floor either randomly or from a prescribed file
    b. Supports movement of one or more players around the board, including revealing and exiting the door
    c. Does not support bombs or enemies.
        i. Consequently, we cannot have a door hidden under a box at this milestone.
    d. Demonstrates compliance with the agreed-upon communication protocol for game updates and player controls.
2. Develop a testing framework for systematic injection and logging of the game's execution.

1. Design:

Develop two major UML diagrams:

1. The architectural subsystem diagram
    - Use either a class diagram or a component diagram to illustrate the major subsystems and the associations between these subsystems
2. The system's class diagram(s) – Develop as far as possible the classes that you will need in your system. Include the testing framework as part of your diagram.
3. Document the Communication Protocol as you understand it. Give the message format, and also use a sequence diagram or state chart to illustrate state-dependent message exchanges.

2. Testing Framework

In this first milestone, you are not going to implement a Player Client. Instead, we will build a testing framework made up of a test driver and a test logger.

- The testing logger will always be deployed on the same machine as the model server. You can choose to implement it as a thread or a separate process
    - The logger should log (as text, in a file) all player controls as they are processed, and all game state updates as they are sent.
    - Thought: Instead of simply printing the whole state, the logger could print a summary of the differences between the previous and current game state.
- The test driver should be implemented as a separate process, because it will play the role of many player controllers, sending UDP player control messages (such as UP, LEFT, GATE_START). The list of messages to be sent should be stored in a text file, read in by this test driver. In this way, a particular text file contains one (automated, repeatable) test sequence; multiple test sequences will be store in different text files.
    - For this milestone, the test sequences will contain movement only, no bombs.

    - Consider: Along with each player movement, perhaps there should also be a time (offset) to slow down the execution of the test driver. In this first milestone, the time delays could all be zero because only MOVEMENT controls are being supported; however, in later milestones, when bombs are deployed, the timing of movements vis-à-vis bomb detonations changes the outcomes of the game. (i.e. whether a player moves out of the way of the bomb in time)
- The model server should start up in two modes: with random initialization of the floor, or by reading in a prescribed initialization of the floor from a file. Suggestion: Use command-line arguments

3. Implementation, Version 1.0

The model server shall be implemented as far as described above. It is suggested that you implement the server in stages (with corresponding test cases)

- Assumption: The server starts with a known initial board state
- Note: No timing is needed. Can go at "computer" speed.

1. One player joins, starts the games and moves about, eventually revealing the door and exiting
2. Two players join, one starts the game, and both move about without touching until the game is finished.
3. Two players move about and touch each other.

3. Implementation, Version 1.1

The spectator client shall be implemented to display the game state as a GUI.

- Note: Timing will be needed in the test sequences so that changes to the GUI can be followed.

Milestone 1 Marking

As you work through the assignment, examine the rubric and attempt to determine how well you are doing.

Weight	F	D	C	B	A
Out of 5	0	1	3	4	5
Out of 10	0	5	6	7	10
Out of 15	0	10	12	14	20
Out of 20	0	8	9	11	15

Milestone 1 Rubric

Category	Criterion	Weight	Points	Comments
Sub-mission	Deliverables submitted on time. Legibility and completeness of README instruction.	/5		
Design	Sub-system diagram – Including test framework. Class Diagram(s) – Showing threads, buffers, synchronization points, Communication Protocol – Format, State-Dependencies Diagrams should always be accompanied by descriptive text helping the reader navigate your diagram. Marked for completeness, clarity, and consistency with project description, classroom discussion as well as your own designs	/10		
Testing Framework	Test logger – provides good detection of wrong behavior and hopefully clues for debugging that behavior. Test driver – provides for automated repeatable test sequences. Includes assessment of the sufficiency and extensibility of the format of your test input files. (Should be documented either in a file or as comments in the programs' headers)	/ 20		
Testing Sequences	A suitably complete set of tests showing normal and exceptional behavior for minimally the test cases described. Floor patterns should include boxes (so players must navigate around) and power-ups (that players can pick up).	/ 20		
Test	Successful execution of other test driver, demonstrating	/10		

Compliance	compliance and correct implementation of the communication protocol.			
Server Model	Code Inspection: Implementation corresponds to design document Implementation makes proper use of concurrency and synchronization mechanisms • Producer-Consumer model for Queue of Player Controls. • Concurrency Expectations met (as described in Project Description)	/20		
Spectator Client	A graphical view of the game	/5		
Overall Code Evaluation	Through Code Inspection: Using of coding standards (indentation,naming,documentation); Code Maturity (minimal,clean,robust). Good handling of exceptions. Concurrency Lessons: No sleeps, spins. Good choice of and correct use of synchronization components.	/5		
Overall Documentation Evaluation	Documentation: Grammar, Spelling, Organization UML: Consistent and clear, expresses idea.	/ 5		
Total		/100		