Arun Atchuthananthan
Dylan Li
Jason Tang

# CS 246 A5 - DD1

---

## Preamble

We decided to make RAIINet for our final project because we wanted to try something new and the abilities seem fun to create. In developing this game, we intend to employ a MVC framework where we have a Controller that interprets and validates player input, a Model that stores and manipulates the board's gamestate, and a View that allows one to display the game.

## Progress Schedule

Joint Github Repository……...November 25

- Consolidate all empty class files in repository

Setup simple Main.cc file…….……November 26

- May initially include unimplemented classes; to be updated as game develops

Write full Test Suite…….....November 26

- To check board setup, movement, download/reveal, player perspective, abilities etc.

Create Board Superclass……..November 26

- Declare / define necessary interface fields, methods

Decorate and implement Board Class……..November 27

- Add Game Objects class
    - General game object methods
    - Link functionality
        - Data / Viruses subclasses
    - Starting obstacle functionality
        - ServerPort, BoardEdge subclasses

Add Board display functionality……..November 27

- Create abstract observer class

        - Implement concrete text observer
        - Implement concrete graphics observer

Create Controller class…….November 28

- Implement command input and interpretation
- Utilize Board to Observer interface as necessary
- Format input to be passed through Board and Observer interfaces

Test and debug basic functionality……...November 28

- Test board text and graphics display
- Test basic game actions: setup, movement, download

Create Player class……...November 29

- Declare / define associated fields and base get methods

Add Player to display functionality…….November 29

- Concurrent player statistics
- Distinct player perspectives

Create Abilities class……November 30

- Declare and define standard Ability subclasses
    - Firewall,
- Add use-ability functionality
    - Parse customized input from controller
    - Allow abilities to edit/add game objects
- Declare and define ability Game Objects (eg. firewall)

Link Ability functionality to Board…….November 30

- Write virtual overridden, overlord, and helper methods to enable ability usage

Test and debug complete display and ability functionality……December 1

- Game display for both player perspectives
- Ability usage
- Game win/loss detection

Add custom abilities and perform custom testing……..December 2

- Barrier, swap position, randomize strength, freeze

Prepare presentation……...December 3

- Write and run final test cases to demonstrates all functionality

## Task Breakdown & Assignment

Arun:

- Create Board super class with general interface so game objects and players can be added
- Create and implement game object class hierarchy as a foundation of the game / Board
- Create controller class to validate and parse input as needed by Board and View interface
- After validating game functionality, adjust controller for players, turns, and abilities
- Create game object subclasses that are necessary for ability implementations
- Outline, discuss, and add implementation for one new player ability

Dylan:

- Create and setup joint repository
- Write/outline general main.cc file that uses the MVC classes to implemented after
- During creation of game objects, add Board methods to manipulate game objects
- After validating basic functionality, add Player class to be constructed and used in Board
- Create and implement ability class hierarchy to be used by Player to change Board
- Outline, discuss, and add implementation for one new player ability

Jason:

- Write comprehensive test suite with varied cases before any class implementation begins
- Create view/observer class to display only playing board using Board implementation
- Test, document, and resolve any errors in basic game setup, display, and movement
- After validating basic functionality, add Player fields/perspectives to display methods
- Conduct testing for additional player and ability functionality as well as regression tests
- Outline, discuss, and add implementation for one new player ability

## Question Responses

1. In this project, we ask you to implement a single display that flips between player 1 and player 2 as turns are switched. How would you change your code to instead have two displays, one of which is player 1's view, and the other of which is player 2's?

We would modify the View objects to display both player's perspectives on every turn such as through the iomanip setw() function for text display. Furthermore, rather than provide these objects with information on whose turn it was to determine what perspective to display, we would split our methods to display user information (abilities, links, downloads, etc.) from both perspectives. For example, could even have each line output be created by displaying the first player's perspective, formatting its width to separate the second display, and then similarly displaying the second player's perspective for that specific line. All logic regarding the differences in perspective (revealed links) would remain the same, as the View would simply have to repeat the same process of checking which link objects have been revealed. To orient the board, which will likely be a 2D vector of gameobjects, we can use a vector iterator to print each row from begin to end or rbegin to rend.

2. How can you design a general framework that makes adding abilities easy? To demonstrate your thought process for this question, you are required to add in three new abilities to your final game. One of these may be similar to one of the already present abilities, and the other two should introduce some novel, challenging mechanic.

Abilities are encapsulated within an abilities abstract base class which provides a pure virtual "use" method that is overridden by concrete subclasses to accommodate for differences in how the abilities change the game states. The "use" method will take in its parameters within a single string that has been processed by the controller, which will have already confirmed that the provided input is appropriate for the chosen ability. So, to add a new ability, one would have to add a new ability subclass (which is very concise and should override the "use" method), modify the controller to accept its correlated letter as well as parameters, and add a new obstacle subclass is the ability is meant to place something on the field. Although this requires up to three

steps, each such step is relatively short and ensures that existing blocks of code do not need to be interfered with.

One ability we want to implement is a swap ability that will swap the position of two links. Another ability we are considering is a 2x1 wall that links can not move on top of. One more ability we might add is increasing the strength of a link.

3. One could conceivably extend the game of RAIInet to be a four player game by making the board a plus shape (formed by the union of two 10x8 rectangles) and allowing links to escape off the edge belonging to the opponent directly adjacent to them. Upon being eliminated by downloading four viruses, each links and firewall controlled by that player would be removed, and their server ports would become normal squares. What changes could you make in your code to handle this change to the game? Would it be possible to have this mode in addition to the two-player mode with minimal additional work?

We would adjust the bounds of the board and add two player additional objects, with new characters representing their respective links. When a player loses, we would find all of that player's gameobjects (links, firewalls) to delete from the board. We would also need to reorganize what is displayed by the observer to make room for the information of all four players. Beyond redefining the boundaries of the board, the game state logic would remain relatively intact as all interactions are conducted through comparison of game object players, so any additional players would be treated equivalently as opponents for a given player.

It may be possible to put two boards on top of each other in a plus shape and add the ability to move between boards. This would allow us to add code instead of modify code and leave the original two player functionality untouched.