Mac Clevinger and Sienna Collette

Object Oriented Design

Dr. Lauri Ricker

December 4, 2018

Report on the Incorporation of Milestones 4 and 5 for the Mr. Jack Game

The codebase chosen for usage in this second part of the project was Mac’s, which required some additional work to make function at the desired baseline. This was not much work, as what was necessary was the connecting of our View to the Controller to the Model; each was individually established a great deal, but the wiring-up of each to the other was simply too much for an exhausted mind. What follows now is the description of what changes were made for Milestones 4 and 5, as well as a user’s guide on how the program works.

After that initial connecting-up, working out the base-errors once the game was playable to transition from round-to-round became the new task; once rested, this was not too difficult. The existing structure was a system of numerical values being assigned special values for representing states in the program, which was changed to constant values for readability and consistency. More explicit methods were used in the Controller for communicating the View to the Model, as opposed to trying to encode intention via numerical values in a single method, so that our View could now communicate, via the Controller, more directly with the Model. Some work was also put into the method by which we translate the Model into a format that the View can interpret to understand how to represent the state of the game at that moment.

For the additional implementation of Milestone 4, we chose to implement the characters Joseph Lane and Madame. Ignoring their abilities for the moment, their inclusion was a simple matter of extending the abstract class MrJackCharacter and specifying a name and movement distance for the two new classes. For Joseph Lane, the abstract class for Tile was adjusted to hold an instance variable that contains a reference to the index of the other Tile in the list of Tiles for the board, between which his barricade is placed; if no barricade, hold -1 instead to show that there is no barricade. We can now, via Joseph Lane’s ability, assign a barricade between two Tiles by providing his class with a reference to the two Tiles that have the barricade; when a new position is desired for the Barricade, he can remove it from the old Tiles and replace their references. The barricade is interpreted by the MrJackCharacter class (as it limits movement) via the canMove method that is used to request whether or not the current character can move to the Tile provided to the method; the adjustment was to include a reference between the tentative Tile being moved to and the one they were already on. This permitted a comparison of the barricade value stored by the two; if the starting Tile had the new Tile’s index as its Barricade (representing that there lay a barricade between the two), it would not permit that specific movement to occur. (The breadth-first general approach would allow it to be reached by other avenues if such would be legal.)

Overall, Joseph Lane was easy to implement, and only required small tweak to accommodate his existence. Madame was a tad trickier, as she had a specific relationship to the Manhole Tile type, and the current canMove did not have any kind of interpretation of the kind of Tile outside of the Tile’s intrinsic ability to be shared (i.e, a building or a lamp cannot be shared with a character, but a manhole or the exit could.) The implementation for Manholes had been to include all other Manholes as neighbors of each Manhole, so during the breadth-first traversal to find viable Tile locations, the Manhole network was modeled as simply being ‘shortcuts’, where neighbors were respected despite not being physically beside one another. Given the previous amendment to provide the canMove method with the preceding and target Tiles, Madame could have her ability to not use Manholes be implemented by querying whether both Tiles were Manhole tiles (an unfortunate place to request the identity of a class, but one side had to budge to accommodate the other, and overriding one method for Madame was much easier than making all Tiles maintain a blacklist for which Characters could use them.) If both were Manholes, then reject the movement, otherwise proceed as normal for verifying a potential movement position.

The way in which the game proceeded was complicated by this ability, however; a three-step system had been put into place wherein the user would select a character, move them, and use their ability. Sometimes the ability was in place of their moving, and, as in the case of Madame, sometimes the ability was passive and would not be explicitly used by the user. This had already led to MrJackCharacters maintaining values specifying the nature of their ability, but the specific interpretation for observing whether they could do their ability after moving/could move after doing their ability permitted finesse in the handling of edge-case characters who needed to be described as having done their ability to satisfy the continuation of the game despite not being able to be directly commanded to do their ability. Implementing a check for the character’s ability attributes allowed for this case to be handled, marking the character as either having moved or done their ability if one invalidated their doing the other.

Moving on to Milestone 5, we were instructed to add an additional character, permit the choosing of the location of the four characters used in the game by the user(s), and dispermit Mr. Jack from escaping in the first turn as well as having a display of the witness component of the game. The third condition was considered to be trivially implemented already by the ‘suspect’ marker each character has; by simply observing who was still suspected and whether or not they were in the light, this information was easily derivable from the game’s current visual state (and in our simplification of the game, we do not have ‘cards’ as, mechanically, such is not necessary when it merely informs the player of information we can communicate in other ways.) The second condition required a minor overhaul; not so much refactoring as adding additional user-interaction and a new method in the Controller to explicitly place a character which the View could call. Now, the user is prompted with the four characters to be used in that game in the space where the Select/Move/Ability buttons had been until each have been placed and the game can begin for real. By simply selecting the character’s name and the tile on the game board, each can be placed, and once all are placed the game begins.

As the game is currently a single-instance assumedly shared between two people, there is no explicit labeling of who should be placing each character (or the order of placement, to be more accurate.) It is assumed that a rule-guide would be provided, and that the users can be trusted to understand what feat they are undertaking. (There is no way for us to tell who is using the single mouse/keyboard that interacts with the game.)

This left the new character implementation, which was a bit of a doozy. We chose Inspector Abberline, whose ability is a passive one that limits the movement of adjacent characters to only one space. This feat was performed by introducing the Decorator, which would take a MrJackCharacterDec class object (an extension of MrJackCharacter) and wrap it in RestrictedMovementDec; this limits the viable movement from whatever their default was to the reduced one space. Unwrapping was made possible by the Abberline object possessing a reference to each other active character in the game in the form of the array of characters stored by the Model, permitting that class to edit its stored array to affect the characters in the game. By overriding one with the wrapped or unwrapped versions, that same change would be seen made to the original character object as well.

The implementation in when to perform these feats hit a snag and was ultimately resolved by an unfortunate explicit request for the type of class being observed; each time a character is selected, the list of active characters is parsed to find whether or not Abberline is part of that game. If so, we search amongst his neighbors for other characters, and if they are found, we wrap them to limit their movement. At the end of their being moved, we unwrap all characters to maintain consistency in changes made between movements. (A turn could start with a character adjacent to Abberline, but, if he moves, then they would be free again.) This is not a desirable approach; something more holistic would be better to allow for smooth interactions between objects in the future that possess a similar trait, but compromises have to be made when working in a group while a deadline looms.

In addition, while not explicitly a part of Milestone 4 or 5, a personal accomplishment of getting a program to be exported as a runnable .jar file while maintaining the validity of its included file references was accomplished, and in such a way that it was still able to be developed within an IDE on local file storage as well as in the exported .jar internal storage formats. This has allowed for a lone executable .jar file to be provided to ease the process of being able to run the program without finding the ‘main’ class from which the program starts. As a personal note from Mac, this was the exact issue that quite terribly bugged me a good while back and has continued for several projects, and that I may finally be rid of the annoyance is a great relief.

Overall, once the fundamentals for Milestone 3 were lain down, all future implementations were rather simple affairs made difficult by mental and physical fatigue as well as general course burn-out. While not a perfect product, if the youthful vigour we once possessed still burned in our veins, it could very easily be made such; but, alas, time slips away, and we can only provide to you what fleeting time permitted. The user guide shall follow, and we hope you enjoy the workwomanship that went into making this project. (Both in the design of the program/code, and in the in-house designed art for the game. Mac had made an art program, so she was able to pretty quickly get some nice-looking stuff; some UI elements could be more clear, especially on the board itself, but we have already made comments on time restraints.)

User Guide

* Upon starting the program, you will see a title screen and a button with the words ‘Start Game’. Do with this what you will.
* You are now greeted by the game proper; immediately, you can see a board of hexagons in the dominant position of the screen. This is the Board, which represents the Game’s Board. It is composed of several kinds of Tiles, and will hold Characters.
* Tile Types:
  + R – Road: Empty position that can be stood on
  + E – Exit: Mr. Jack can escape if in the dark when standing here at the end of their turn; can be stood on.
  + M – Manhole: Permits travel between all Manholes on the board; can be stood on.
  + B – Building: A building. Cannot be stood on.
  + L – Lamp: Provides light to adjacent tiles. Cannot be stood on.
* Several of these tiles will have additional labels, describing them as being ‘open’ or ‘off’; these are to be interpreted in the context of their home Tile. (So a Lamp would be ‘on’ or ‘off’ describing the status of its providing light, and a Manhole may be covered or available.)
* On the right side of the screen you can see the Clock; it informs which Player’s turn it is, what numerical turn number it is, what actions have been performed the currently selected character, and which characters have been selected (as well as an implied order for character choosing; in the context of the game’s rules, UI intuition should be evident.)
* On the bottom of the screen you see the Interaction region; this is where the game’s buttons are for doing specific actions. You will be required to click on the Tiles on the Game Board, but for what context that is done in, this region provides that insight.
* To begin, you are given four character names (in initials for space considerations). You will want to select a name (at which point it disappears from its box) and a legal Tile on the Board to place them on. Upon doing this properly, you will see them appear on the Board and their name disappear from the Interaction region.
* When all names have been placed, the game will begin. Note: The Menu button will allow you to return to the Title Menu, from which a new game can be started if desired. (This button is always present.)
* When everyone has been placed, four new buttons appear:
  + Select – Pick a character on the board to be used by the current player.
    - If a character has already been selected (viewable by the list of names on the right), you cannot choose them again.
    - Click on Select and then the Tile of the Character you want.
  + Move – Move the selected character to a legal position
    - The board will show viable Tiles you can move to by marking them with a Red patch; click on any of them to move there.
  + Ability – Use the selected character’s ability
    - Upon selection, click whatever sequence of Tiles are necessary for their ability to function. This may require knowledge of the game’s rules and a keen eye to know which Tiles to click on; better UI feedback is necessary.
    - Some characters may not have an Ability, in which case this has no purpose, and they are immediately marked as having done their ability upon movement.
  + Cancel – Undo the previous selection (Not Character Selection, but choosing to do Move or Ability.)
  + Menu – Return to the Menu to start a new Game.
* For a round to progress, each character must be moved and have their ability used. This is shown by the three red dots in the Clock region of the screen.
* When a Character is on a Tile, several UI symbols appear:
  + The open/closed eye – represents whether or not the character is suspected by the Detective. If open, they are. If closed, they are not. (Low-poly art, be open to interpretation of what an eye is.)
  + Yellow splotch – represents whether a character is in the light or not.
  + The number in the top left – Represents the tile’s number
  + The number (sometimes) in the top right – Represents the location of a barricade in relation to the current Tile (so the barricade is between the Tile and that which has the numerical value shown in the corner.)