Muhammad Moughal
Kais Baillargeon
Susan Tan
Matthew Prince

CS 3110 A7 Final Report

Vision

Our team's vision has remained consistent through A7: to develop, test, and implement the game of Monopoly. In a game of monopoly, there are a maximum of 4 players. The player rolls two dice and moves to a certain location on the board. There are "tiles" with different "functions": property tiles which can be bought, card tiles which draws a card with a specific function, income tax tile to pay $200, luxury tax tile to pay $100, passing go gives $200, railroad tiles which can be bought (rent is varied), utilities tile (electric company and water works) which can be bought, go to jail tile which takes the player to jail, jail tile, and a free parking tile. The player can also build houses and hotels on properties. If a player lands on an owned property that isn't theirs, the player has to pay a rent fee. A player can also mortgage and buy/sell houses and hotels. For more monopoly rules, see wiki. The board.json file is now adjusted to be Cornell-based. One specific change to our vision from this sprint is that we altered some rules that we felt were clunky, and/or unnecessary to enjoy the game (this included some changes with jailing and rolling doubles) but we retained the overall, original monopoly rules and structure. This will be expanded upon in the following sections, and specifically the last paragraph in the next section. Note that besides the normal "roll" command, we kept the [rolln position] command to make testing of functionality easier for everyone including the graders. This wouldn't be on a version for regular users. The same idea goes for testing specific chance and community cards.

Summary of progress

In this sprint, we continued building up Monopoly functionalities. We expanded upon our last sprint by implementing the more complicated features of monopoly such as jailing (moving into jail, actions in jail, and leaving jail) and eliminating players due to failing to pay jail, rent, or money from chance and chest cards. We also added the ability to buy and sell houses on properties once all properties in a set (particular color) are owned by one player. If a player owns all 4 houses on a property, then the player can buy and sell a hotel. Furthermore, with the addition of houses/hotels came the functionality of scaled rent, so rent would be different depending on the number of houses or hotels on a property. Other types of unique rent were added as well, including the railroad rent which scales with the total number of railroads owner, and the utility rent which multiplies based on the die roll that caused the player to land on the utility. The ability to mortgage properties was also added, enabling players to receive funds from the bank in exchange for losing the ability to collect rent or construct houses or hotels on any property of the same color. Un-mortgaging repays the loan with 10% interest and lifts all restrictions Finally, we added chance and community chest cards which have functions including going to tiles moving spaces, taking or giving money, and losing money based on houses/hotels. In addition to more complex gameplay, the visuals of the entire game were completely overhauled, with a larger, more colorful board and a bright title screen, with the ability to use multiple boards.

Regarding some specific changes from original Monopoly, you may have noticed that some things were altered. For example, jailing works a little bit differently: instead of forcing to pay the fine or rolling doubles, the player is eliminated if they fail to pay the fine after their last turn. Overall, we felt that this was more aligned with our vision of the game, and we also did not want to implement a complete one-to-one Monopoly game (hence we also have a custom, Cornell-based monopoly map that we expect users to play). Hence, by eliminating the player after their third turn, this naturally allows the player to forfeit if they believe they do not have a chance to win and speeds up the game - after all, we wouldn't want users to sit in front of the same screen for hours play Monopoly. The faster pace is our small twist on the game and we hope you enjoy it. Following this idea, we also decided to do away with doubles. We weren't using them in getting into and out of jail like regular Monopoly either way.

Activity Breakdown
Everyone developed **specific functions**, **tested,** and **documented** them in this sprint. Everyone also **helped write the report**.

Muhammad Moughal

State	*Implemented jailing and player elimination, and income/luxury taxes.*
Main	*Improved make play related functionality and play-tested all the above.*

Kais Baillargeon

Monopoly/Json	*Created a colorful monopoly board and json files.*
State	*Implemented color, board printing and mortgage features,*

Susan Tan

State	*Implemented Chance cards, community chest, more elimination, and other special tiles (free parking, etc).*
Command/Main	*Added commands for cards, get out jail card, and implemented these in main.*

Matthew Prince

State	*Wrote functionality for building and selling houses, displaying the houses and hotels on the board, and varied rent.*
Command/Main	*Wrote commands for houses/hotels and implemented features on board.*

Productivity analysis
As the team became increasingly comfortable working together, our productivity actually increased during A7. Close to daily meetings were held in Duffield hall at 9PM where - inspired by the "sprint" system - we discussed our individual progress and roadblocks together. This allowed us to quickly resolve issues with specific features and merge conflicts. When we were not meeting in person, our slack channel continued to be a crucial avenue of communication. In this sprint, we had a lot more functionality to implement as monopoly is a huge game and the rules are extensive. As our program became more complex, we encountered many bugs as each of us are implementing so much and

need to put it all together. As a team, we needed to take a step back and implemented tests that caught bugs as well as put the game as a whole. This led to better developed game and higher confidence in using one another's code. Ultimately, we believe that the increase in feature complexity, as well as overall scope, in A7 versus A6 is a testament to the improved productivity and chemistry of our team. This also is one of our main ideas regarding grades in the next section.

Coding standards grades
Standards wise, as we got a better sense of our project, we definitely improved across all four categories of documentation, testing, comprehensibility, and formatting. Testing wise, we made to sure test for necessary functions and implementations across the game. We had decent coverage for all files except for state.ml, but for good reasons. For state.ml, we felt that it was better to test the functionality by play-testing, rather than unit testing - and ultimately, we believe that it was for the better in terms of productivity and correctness. Still, we made sure to test basic functionality in state through unit tests. Documentation also improved: we had a ton of functions and we made sure to document them clearly and efficiently across our large project. The same goes for comprehensibility and formatting. Due to the increasingly extensive project and the fact that we improved in all four categories between A6 and A7, we believe we should receive a "1" for the four categories.

Scope grades
We based our scope grades off the scope grades for A3; we believe this is fair because just as A3 was a continuation of A2 with more functionality, our final project is a continuation of our beta with more complex features, and we based our scope for the beta on the scope for A2.

Satisfactory:
The solution compiles and make play launches the game. It contains all of the functionality of the beta, and implements the ability to buy houses and hotels on properties with scaled rent.

Good:
The solution implements chance and community chest cards which may have many different functions. The solution gives the user the choice of how many players will be in the game.

Excellent:
The solution implements the following augmentations: full jail functionality, including the Go to Jail tile and Jail itself, where the player stays for up to three turns until a fee is paid; a brightly colored board and title screen; the ability to mortgage properties, which gives the player half the price back but removes the ability to buy buildings or gain rent from the tile.

Following the above guidelines, we believe we deserve a grade of "Excellent" due to the fact that we satisfied the requirements posed under "Satisfactory", "Good", and we finished all the requirements of "Excellent" that we set up ourselves during the end of the first sprint. Overall, we have completed all the goals we wrote in the last report.