Coding Ladder - Teaching a Structured Programming Language Through a Casual Level Game

A Master's Thesis Project
Submitted to the faculty of
Worcester Polytechnic Institute
In partial fulfillment of the requirements for the
Degree of Masters
in
Interactive Media and Game Development
By:

_____
Junfeng (Kimi) Guo

Date:
May 10th 2021

Thesis Advisor:

_____
Professor Charles Roberts

Committee Members:

_____
Professor Dean O'Donnell

1. Introduction

In recent years, video games have played a big role in education. Innovations in Internet technology and smartphones continue to fuel the accessibility of gaming. According to Ed Dieterle, the Senior Program Officer for Research, Measurement, and Evaluation for the Bill and Melinda Gates Foundation, students who use games in learning improve their results by 12 percent (Shapiro, 2014). “Games mirror the way the human mind was designed to learn,” says Mitch Weisburgh, partner at Academic Business. The popularity of educational games is getting attention from academics and professionals. Games in educational use are able to assist students in pursuing their intellectual interests but in a more efficient way than books with a personalized, engaging learning experience(Voss, 2015).

Computer Science also benefits from the use of games to provide students a more enjoyable way to learn the fundamentals of programming(Wong, and Mohamad Yatim, 2014). Learning programming by playing games is a fun way to encourage people to improve their programming skills. Programming games broadly separate themselves into two areas: single-player games where the programming elements either make up part or the whole of a puzzle game, and multiplayer games where the player's automated program is pitted against other players' programs (“Programming Game”, 2019). For puzzle games, one example we find compelling is CodeCombat(CodeCombat, 2013), which is a web app for puzzles and challenges that can only be solved by writing code. CodeCombat has a significant educational bent with a “Classroom Edition" that teachers can use to help their students learn how to code. For games involving competition, Robocode(Matthew A. Nelson and IBM, 2001) is a complex programming game where players need to code robot tanks that fight against each other by writing the artificial intelligence that drives robots to success—using real programming languages like Java, Scala, C#, and more(Lee, 2019).

However, there are negative aspects to using games to teach programming. Since the basic structure of each game has been settled by the game developer, the incorporation of programming content is hard to deeply integrate. When compared to traditional programming projects, students–or players–are usually restricted to a single assigned programming sequence, limiting the amount of programming knowledge that can be learned while playing such games.

For my project, my core idea was to make a game to motivate players with little to no background in computer science to learn foundational concepts in programming. Alternatively, it could help students who did not enjoy taking CS courses to regain their interest in programming. By using the Blockly coding language, this project intends to explore the use of an in-engine visual programming language for teaching programming. I designed a casual level game in the context of computer science education. The game interface consists of two areas: the Blockly programming area(see 2.1 for reference) and a set of puzzles that can be completed using Blockly and gradually increase in complexity. There are different structured coding blocks (See Section 2) for each level, and each level requires players to explore these specific blocks to craft a solution for an associated puzzle. We also provide a freestyle environment for players to experiment with all available coding blocks found throughout the gamer. After I developed the game, I conducted a user study of player experiences to analyze how this methodology could help them overcome the stress of learning to program.

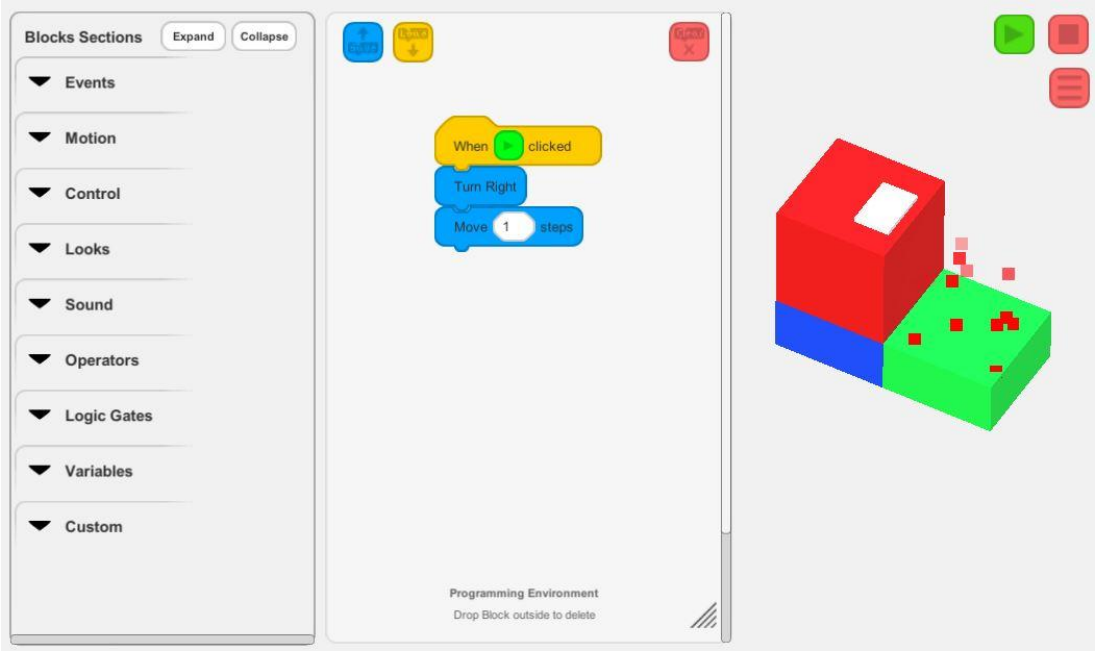

Figure 1. First look of the game

2. Related Works

2.1 Blockly

Blockly is a web-based editor that supports block-based visual programming. Blockly enables web pages to include a visual code editor for any of Blockly's five supported

programming languages, or a developer's own custom script. In Blockly Games, users can solve a maze using Blockly's editor as shown in Figure 2.1 shown below(Google, 2017).

For designing coding games, Blockly is a suitable tool for developers to consider. Blockly uses interlocking, graphical blocks to represent code concepts like variables, logical expressions, loops, and more through Google's shared online resources. As seen in the screenshot of Coding Ladder(see Figure 1), we have upgraded the game to use 3D blocks, more types of block commands, and an intuitive combination of game features & mappings.

However, an article from Katie Victoria points out that coding in Blockly is only performed in a visual format(Victoria, 2019). While this makes it easier for children to learn how algorithms work on a theoretical level, it does not teach more traditional, text-based coding languages. Despite this, Victoria stated that "Blockly has been used by a variety of developers to create games and apps that kids find enjoyable".

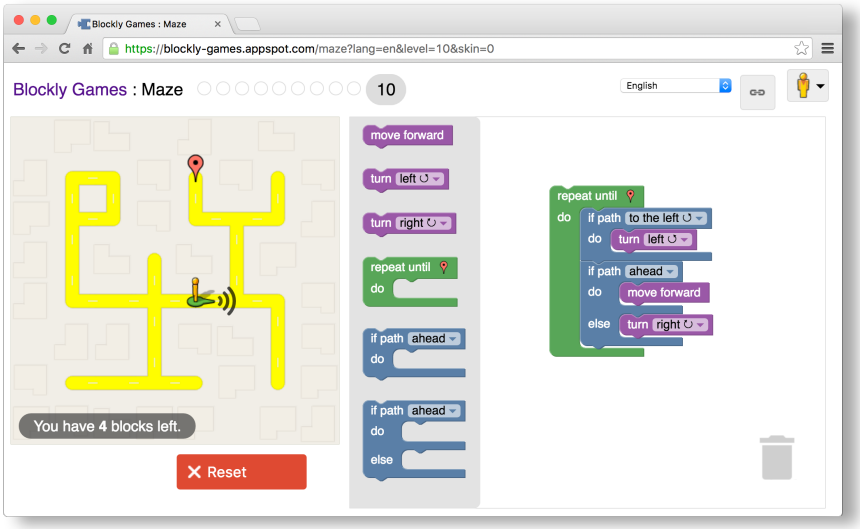

Figure 2.1 Sample works built with Blockly

2.2 Scratch

Scratch is a visual programming language developed by the MIT Media Lab. It aims to help children easily create and share their own interactive digital projects such as video games, animations, newsletters(Resnick et al. 2009). With different shaped programming blocks in Scratch's library, children are able to play with materials like sounds and images to build their own works. Specifically, programming is done by snapping commands(sometimes with parameters) together to move various objects on the background(Maloney et al., 2010). This is arguably similar to the way people play with LEGO bricks, and "Scratchers" can build their own projects freely with these components.

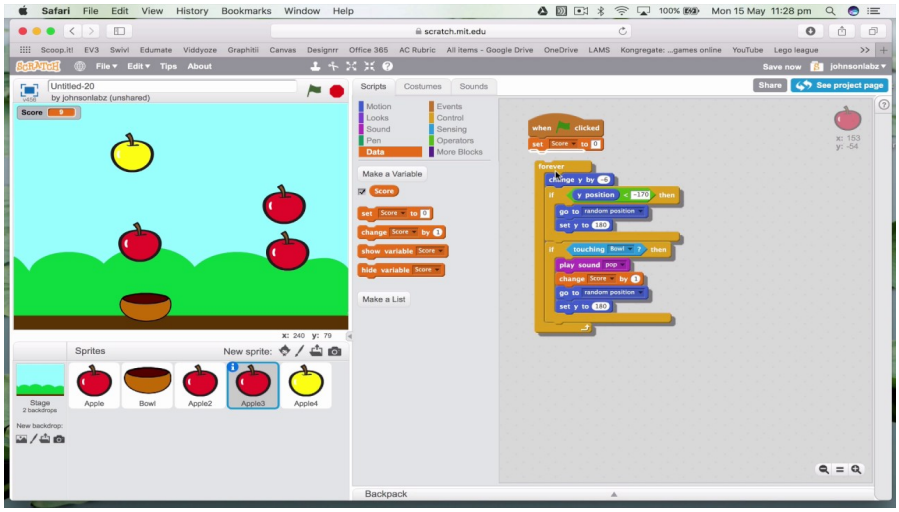

Figure 2.2 User Interface in Scratch

Scratch is similar to Blockly in that they both use visual blocks in place of text-based coding. The 3.0 version of Scratch was built and updated by using the Blockly library(Pasternak, 2019), but Scratch still differs from Blockly in that Blockly is designed for a slightly older audience with a more advanced skill set. Research shows that “students tend to find options and commands too complicated in Scratch, which causes difficulty in the process of building projects.”(Filiz KALELIOĞLU, Yasemin GÜLBAHAR, 2014) Moreover, some students are expecting “something cooler” in Scratch because the game art looks a lack of variety to play(no story background, lack of artistic design, single gameplay style). Therefore, the future plan for the Scratch team is to focus on lowering the technical floor and making it more approachable for kids.

2.3 Human Resource Machine

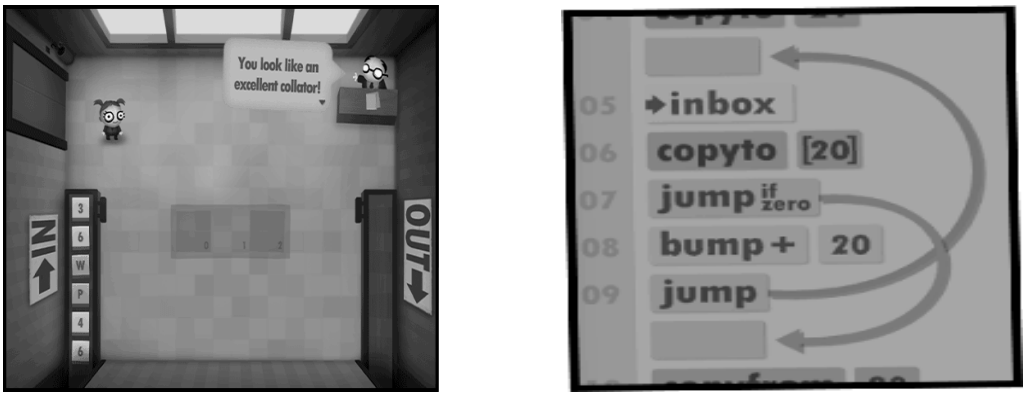

Figure 2.3. Sample screenshot inside the Human Resource Machine

The puzzle game is another popular type of programming game. For instance, Human Resource Machine(Tomorrow Corporation, 2015) can help teach programming and simulate computer algorithms in a simple game where the user has to solve puzzles. The main background image of the game is depicting the office as a simple computer, which has an inbox and an outbox (inputs and outputs), and a few slots on the floor to store stuff for later(memory) (nossy, 2018). In each level, after the description of a puzzle as the goal, there is a command box with simple drag n' drop commands for the player program with Players start the game with just two commands, and gradually earn more as they are promoted(See Figure 2.3).

3. Methodology

In order to evaluate the game I created, I applied the following methodology:

1) Design a game that revolves around coding with coding blocks. The categorized coding blocks will enable players to program solutions to in-game challenges using Blockly.
2) Implement scripting language detection system to determine win-lose condition, and exporting game from Unity and playtest the game.
3) Perform a comparative analysis between Coding Ladder and Blockly Games: Maze in order to reflect on differences between the two games and identify areas for future improvement in Coding Ladder.

3.1. Design a game that revolves around coding with coding blocks

My plan was to make an original tiled, map-based puzzle game. I decided to develop a game with a series of puzzle levels because I believed the design would be intuitive to players. I created puzzles for each level with increasing difficulty; levels would also differ by object layouts, such as structured combinations of scripting blocks.

My primary inspiration was Human Resource Machine, (see Section 2.3) which is also a puzzle game that teaches players programming skills and awareness of coding structure. Other related games include RoboCode (Matthew A. Nelson and IBM, 2001) and CodeCombat(George Saines, Scott Erickson, and Nick Winter, 2013) that were designed to improve effectiveness when teaching programming. My goal was to overcome the stress of learning to program by playing this game, and I added calm music to build a peaceful and relaxing experience.

The game has multiple levels of ascending difficulty and each puzzle builds from one to the next. In my project, most of the game uses drag and drop for interaction between Blockly

and the game's environment, which is suitable for beginners and encourages learners to use their senses to interact and learn (Yassine, Chenouni, Berrada, & Tahiri, 2017).

I made a game in Unity for one player on a desktop computer. The game could potentially be made more engaging by changing the complexity of coding functions by limiting coding blocks for different levels or by introducing a timer mechanic into gameplay.

3.2. Develop the game in Unity and playtest the game.

I implemented the procedurally generated tiled maps using a tool called SuperTiled2Unity(Seanba, 2019) to seamlessly import prefab grid assets and programming blocks into Unity. Each block is made by a C# script; these blocks are in turn managed by a block controller class. In addition, I designed & scripted the Blockly workspace with the Unity C# development environment. Major components of the game system included:

- the player character
- the blocks area containing the coding commands
- the programming environment area,
- win-lose condition
- mode/level selections depending on lock & unlock levels.

The examples below show how the game progresses with procedural levels of increasing difficulty.

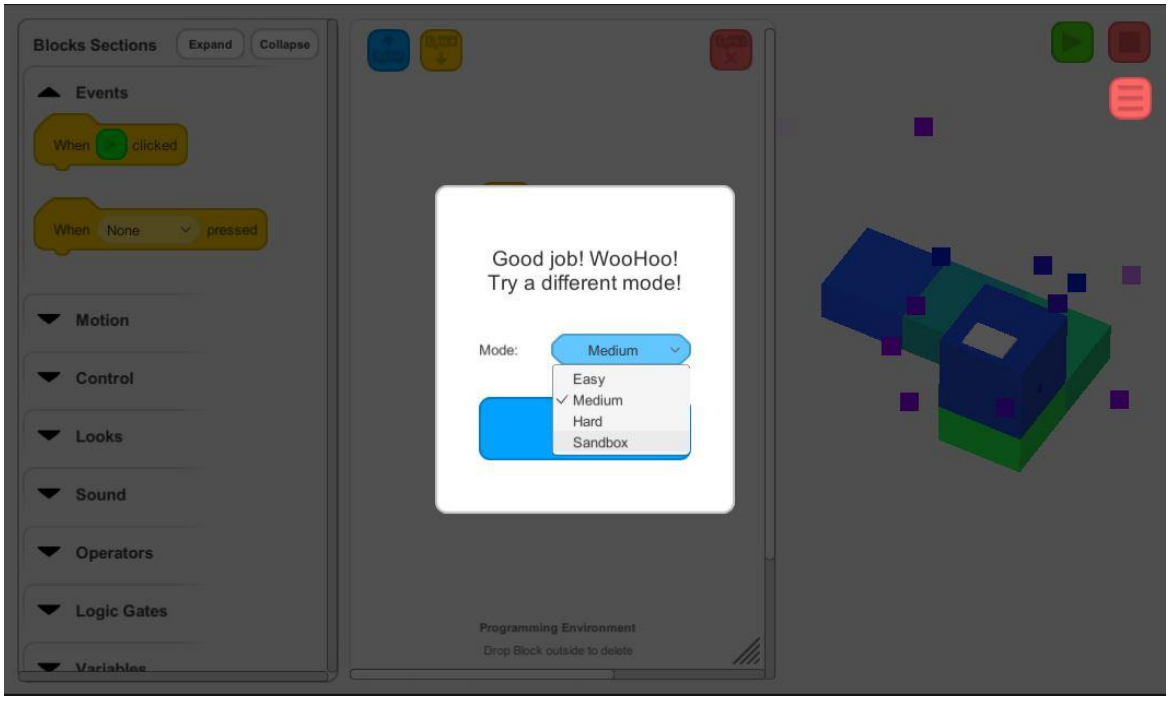

Figure 3.2.1 Difficulty Mode Selection

In Easy Mode 1, there is only one function deployed to the programming workspace, and the robot starts one tile away from the goal. However, the robot—represented by the large red cube—starts facing the wrong direction (the robot's current orientation is indicated by the white rectangle on top of the cube).

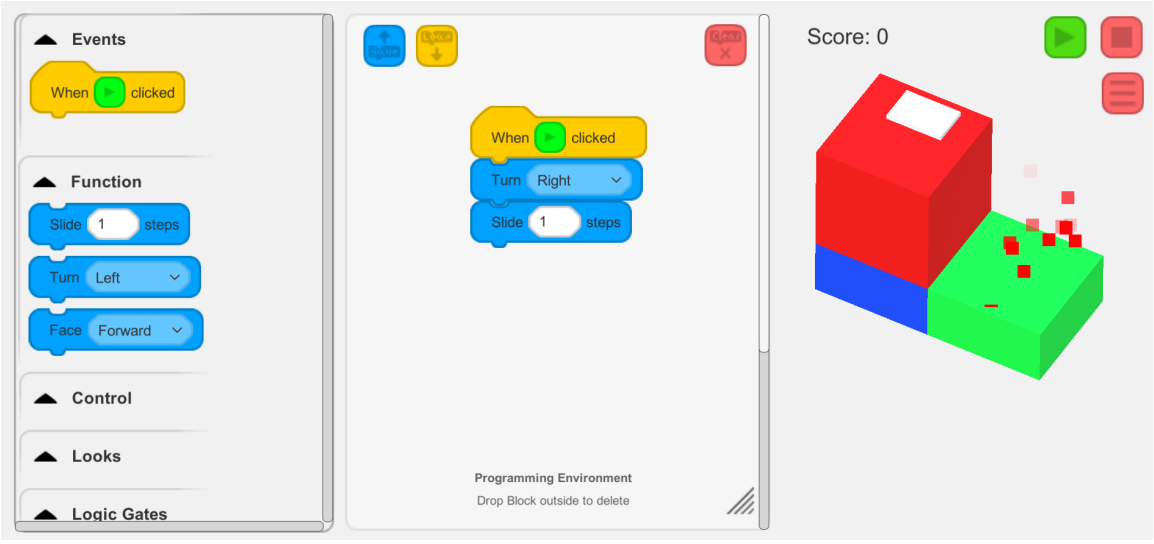

Figure 3.2.2 Easy Mode 1

In Easy Mode 3, which requires nested commands to solve, the map has a longer path and increases in complexity, which requires players to deploy new types of blocks to reach the goal:

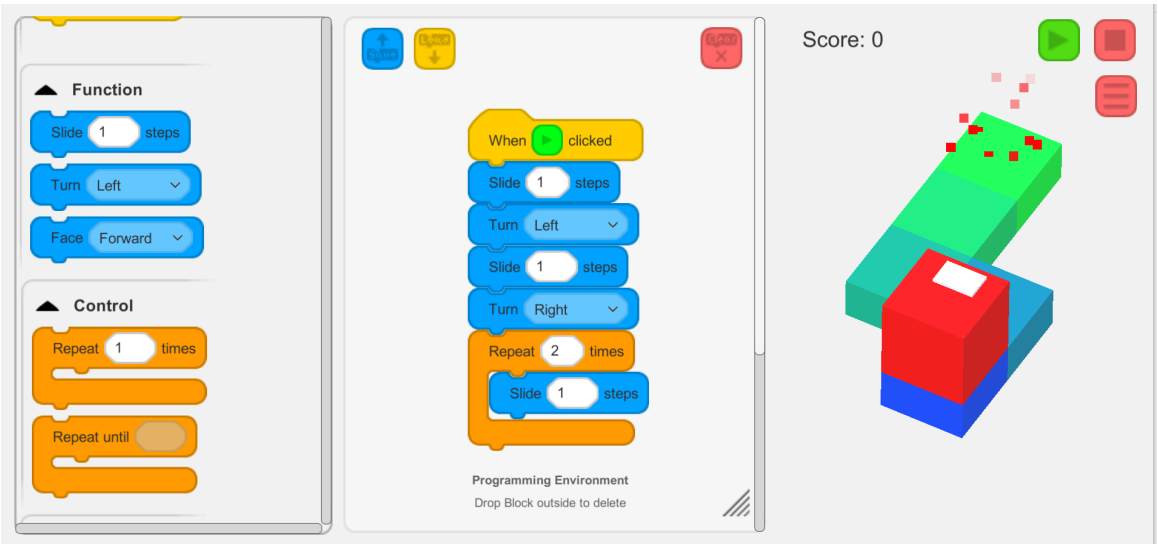

Figure 3.2.3 Easy Mode 3(Correct solution)

I designed a scoring mechanic to detect whether players used new commands to finish each puzzle (for example from image above, players needed to use a repeat block in the final command combination). If players follow this requirement, their score increases by 100, and a new level appears; if not, the score increases by 50(See Figure 3.2.5), and text instructions appear to notify players that the new block command must be used in order to complete the level. By gradually introducing new blocks and requiring their use, we scaffold learning the Blockly programming language.

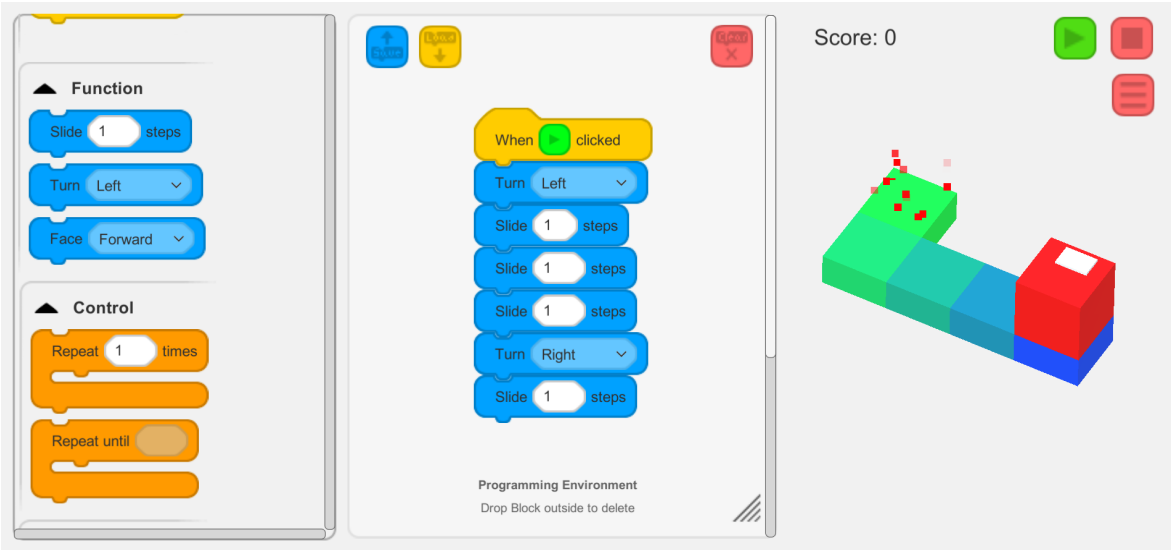

Figure 3.2.4 Easy Mode 3(Incorrect solution)

Figure 3.2.5 Easy Mode 3(Result game scene for half-score solution)

In Hard Mode, players are required to using commands that were recently introduced, which in turn requires players to learn new programming concepts.

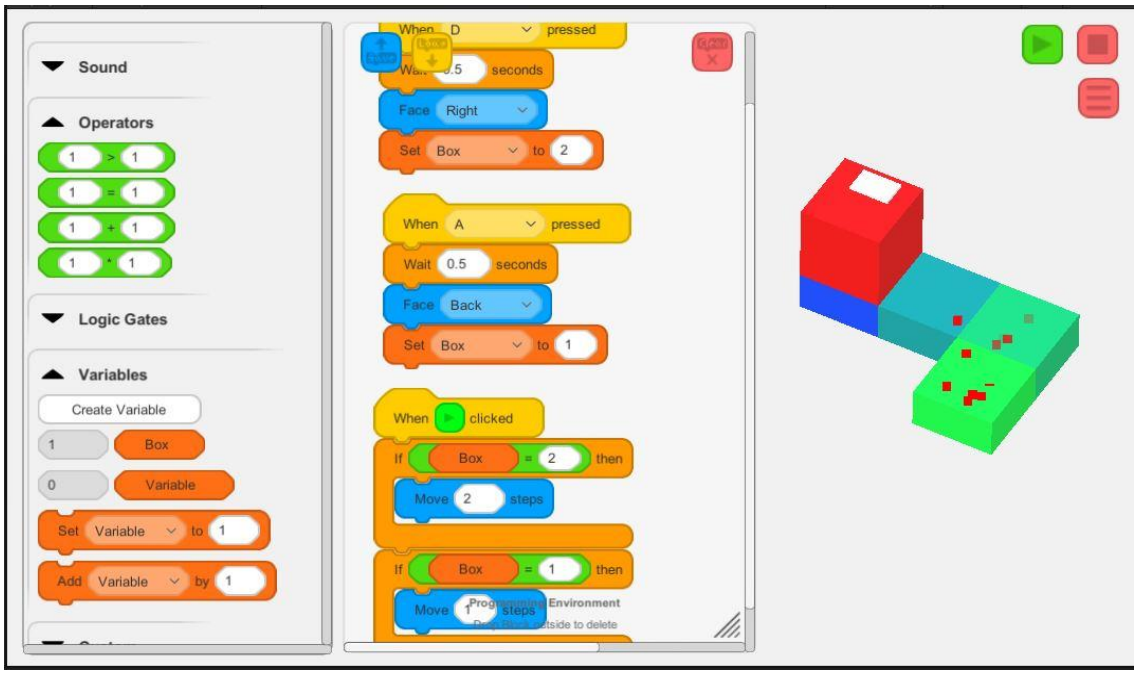

Figure 3.2.6 Hard Mode

In Sandbox Mode, player can freely experiment using all available Blockly commands., This can help help players review basic game mechanics and encourage them to experiment with combinations of blocks, as shown in Figure 3.2.7.

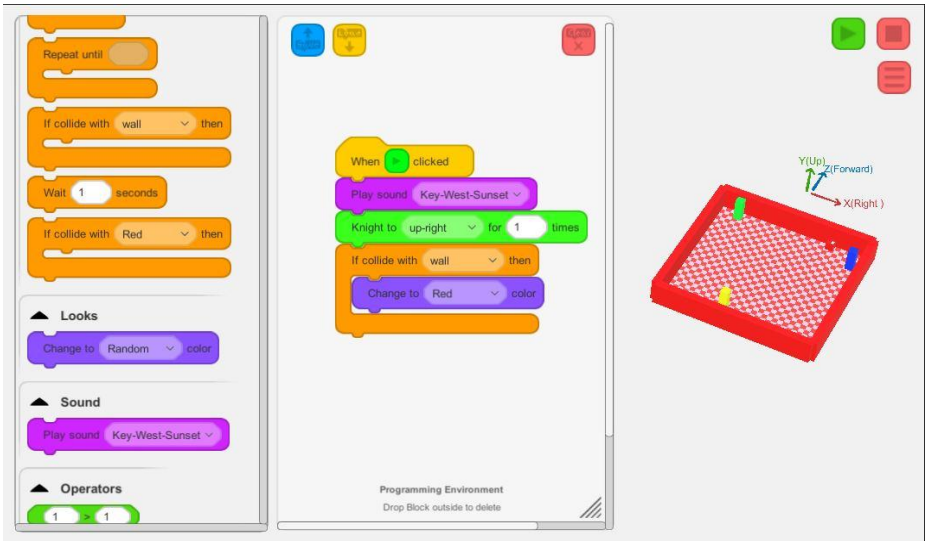

Figure 3.2.7 Sandbox Mode

3.3. A comparative analysis between Coding Ladder and Blockly Games: Maze

My thesis product was inspired by two specific aspects of Blockly Games: Maze.

1. The Blockly game layout lets the user to intuitively create code by via a drag and drop mechanism.
2. Blockly games can make programming concepts easy to understand, because the game have clear goals and rules and an easy-to-use visual block interface.

My game Coding Ladder had some similarities with Blockly Games: Maze. However, I will explain 3 major differences (graphic, sound, and level design) between them. I will also provide a future level design mockup at the end of this section.

For graphics, I used 3D graphics in Coding Ladder to enrich the visual experience. I changed the game character(robot) movement to make it realistic and fun to watch compared to the movement in Blockly Games:Maze, so players will be encouraged to explore more possibilities In addition, I also added a particle effect showing where players finish the puzzle; this effect is difficult to achieve using 2D graphics alone, especially using Unity Engine. However, the 2D graphics of Blockly Maze bring less distractions to players so they can quickly

start with clear instructions. 2D graphics also have less latency to respond to the player's in-game decisions..

The Blockly Games: Maze did not include any background music and contained very few sound effects for the drag and drop coding mechanic. In Coding Ladder, I composed a series of music tracks for levels, and tracks changed as players advanced. Music during the easiest level is calm and melodic while levels of medium difficulty feature fast rhythmic music. I added sound effects to most key events, including robot movements and win-lose triggers. I also added customized coding blocks with sound lists for players to experiment with and explore. I think the game audio libraries can not only bring a much more joyful experience to players, but can also help players understand how to interact with the game.

Compared to the level design of Blockly Games: Maze, Coding Ladder provides explorable puzzles thanks to its procedural generated levels. The restricted condition for showing new categorized types of blocks in different levels helps lead players to systematically practice new concepts by setting rules, as players are required to use newly introduced blocks to advance. Levels in Blockly Games:Maze can be solved using a much smaller, more limited set of blocks in comparison. Furthermore, Coding Ladder has many customized types of coding blocks (e.g. colors, sounds, and customizable movements in the sandbox level) to encourage players to explore and learn.

Map designs in Coding Ladder use colored tiles to generate paths, and the number of tiles increases as levels proceed. Character and goal particle effects are located above the grid (wee Figure 3.3.1). In contrast, Blockly Games: Maze uses a linear design, by highlighting designated yellow lines with start point(character) and end point(goal)(See Figure 3.3.2).

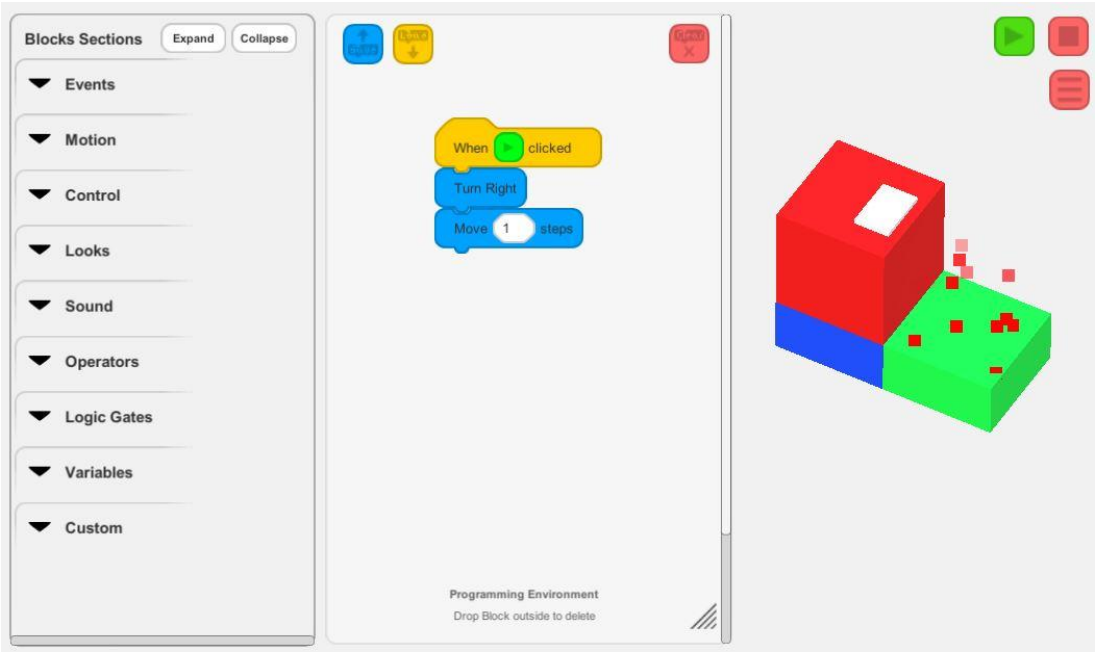

Figure 3.3.1 Game Scene of Coding Ladder

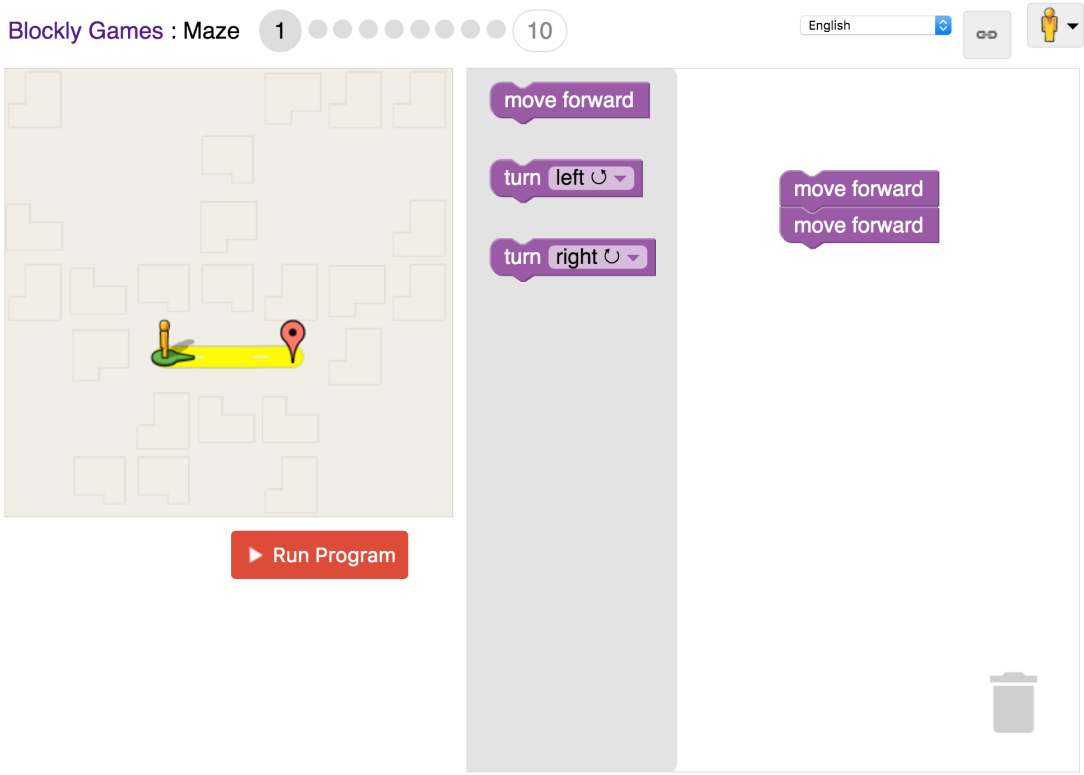

Figure 3.3.2 Game Scene of Blockly Games: Maze

However, Coding Ladder has lots of room to improve. I learned that the level implementation for Coding Ladder was much harder than Blockly Games: Maze. Due to time

constraints during development, I only implemented a few programming concepts (loop, variable, and basic function) with simple puzzle layouts. As I continue to develop Coding Ladder, I plan to develop a series of levels with new types of coding commands that relate to practical coding problems (e.g if-else statements, data types). In addition, if I add a competitive two-player mode, the game could be more fun to play.

Figure 3.3.3 shows a sample mockup for a future level, which has a more complex design. In this example, the robot must pick up bagels when it encounters them, move towards the goal, and check if all bagels within the path are collected.

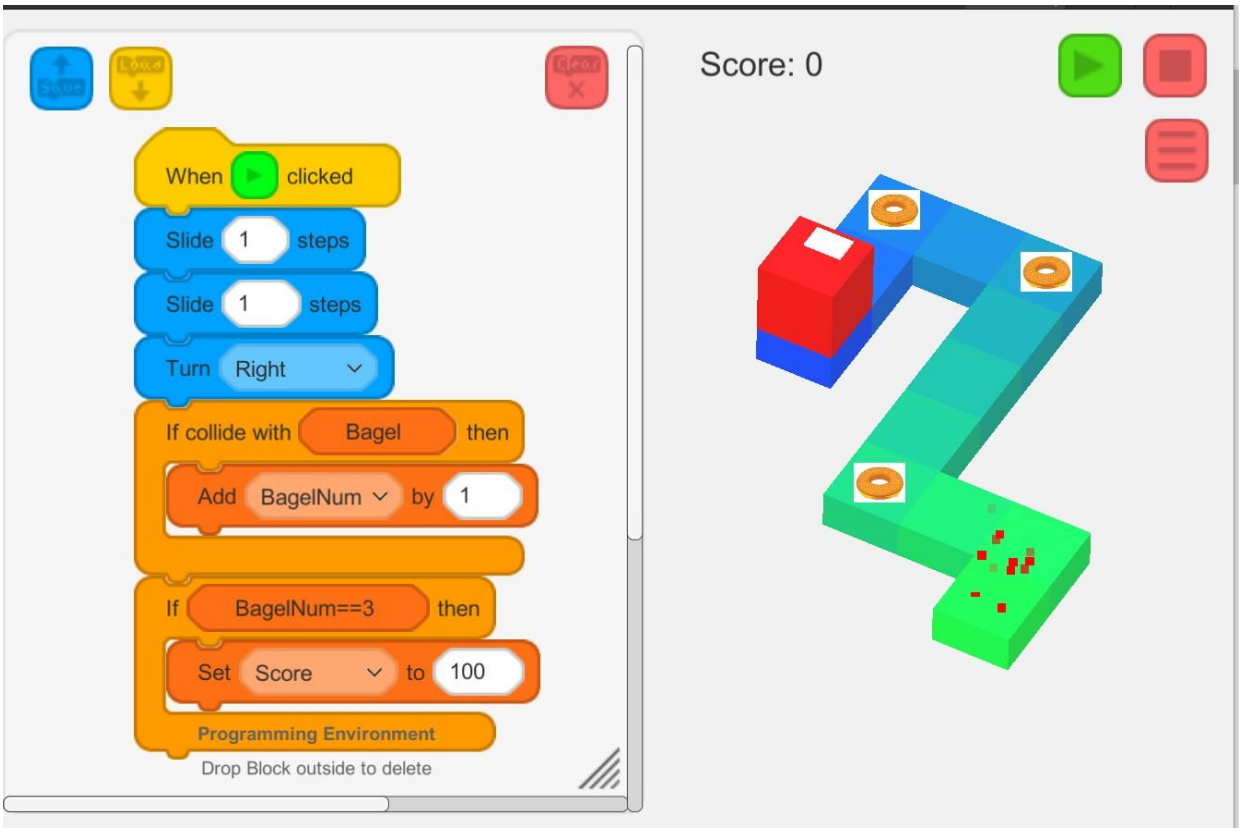

Figure 3.3.3 Mockup idea for future level design

4. Conclusion

My primary goal in creating Coding Ladder was to motivate non-CS background users to learn foundational concepts in programming. In addition, I tried to explore the use of an in-engine visual programming language to teach programming topics. I created a puzzle-based single player game in Unity to help teach players to program. The game incorporated Blockly, the visual programming toolkit used in Scratch, and uses procedural level design with clean

layouts. Procedurally generated levels can help improve enjoyment over repeated play sessions, but I found that implementing this feature quickly made building more complex levels technically challenging.

To improve this game, I will implement more levels that explore new programming concepts, and add game elements such as obstacles that require more complex programs to overcome. I may also consider deploying a countdown timer to limit game time so that players feel a bit more pressure, which might make the game more fun to play.

The complete Unity project can be downloaded at:
https://drive.google.com/drive/folders/1NQBz89tAqI-oHqAl5GA97qik4kIeQuZE?usp=sharing

5. References:

A. Shapiro, J. (2014, June 27). Games In The Classroom: What the Research Says. Retrieved October 05, 2020, from https://www.kqed.org/mindshift/36482/games-in-the-classroom-what-the-research-says [Accessed 1 Oct. 2020].

B. Games4ed.org. (n.d.). *Why Games are Important to Education | Games4Ed*. [online] Retrieved from http://www.games4ed.org/who-we-are/why-games-are-important-to-education/ [Accessed 28 Feb. 2020].

C. Voss, K. (2015). *The Growth of Gamification: What it Means for Schools and Districts*. [online] Getting Smart. Retrieved from https://www.gettingsmart.com/2015/11/the-growth-of-gamification-what-it-means-for-schools-and-districts/ [Accessed 26 Feb. 2020].

D. Victoria, K. (2019, March 17). *The best kids coding languages*. Teaching your kids code. https://teachyourkidscode.com/kids-coding-languages/ [Accessed 5 Mar. 2020].

E. Pasternak, E. (2019, January 17). *Scratch 3.0's new programming blocks, built on Blockly*. Google Developers Blog. https://developers.googleblog.com/2019/01/scratch-30s-new-programming-blocks.html [Accessed 5 May. 2021].

F. Wong, Y. and Mohamad Yatim, M. (2014). *Computer Game as Learning and Teaching Tool for Object Oriented Programming in Higher Education Institution*. [ebook] Perak: ScienceDirect, pp.215-224. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877042814014554#!

G. En.wikipedia.org. (2019). *Programming game*. [online] Retrieved from https://en.wikipedia.org/wiki/Programming_game [Accessed 3 Mar. 2020].

H. Rugelj, J. and Zapušek, M. (2013). *Learning programming with serious games*. [ebook] Ljubljana: ResearchGate, pp.1-7. Retrieved from https://www.researchgate.net/publication/256668036_Learning_programming_with_serious_games.

I. Santos, A. L. D., Souza, M. R. D. A., Figueiredo, E., & Dayrell, M. (2018). Game Elements for Learning Programming: A Mapping Study. *Proceedings of the 10th International Conference on Computer Supported Education*, 89. Retrieved from https://pdfs.semanticscholar.org/65f9/a8c93529c77fa92061248c2eaa639ef6ec27.pdf.

J. Resnick, Mitchel, Brian Silverman, Yasmin Kafai, John Maloney, Andrés MonroyHernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, and Jay Silver. "Scratch." Communications of the ACM 52, no. 11 (2009): 60. doi:10.1145/1592761.1592779.

K. Maloney, John, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. "The Scratch Programming Language and Environment." ACM Transactions on Computing Education 10, no. 4 (2010): 1-15. doi:10.1145/1868358.1868363.

L. Nossy. (2018). *GameCentral #3 Human Resource Machine*. Steemit. https://steemit.com/gaming/@nossy/2cxx729g [Accessed 5 May. 2021].

M. Filiz KALELIOĞLU, Yasemin GÜLBAHAR, The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. Informatics in Education, 2014, Vol. 13, No. 1, 33–50.

N. (2018, February 7). How Do We Best Teach Programming to Beginners? Retrieved from https://casabona.org/2018/02/how-do-we-best-teach-programming-to-beginners/ .

O. Yassine, A., Chenouni, D., Berrada, M., & Tahiri, A. (2017). A Serious Game for Learning C Programming Language Concepts Using Solo Taxonomy. *International Journal of Emerging Technologies in Learning (IJET)*, *12*(03), 110.

P. COMBÉFIS, S., BERESNEVIČIUS, G. and DAGIENĖ, V. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. *Olympiads in Informatics*, [online] 10(1), pp.39-60.

Q. Lee, J. (2019, July 5). The 9 Best Coding Games to Build Your Programming Skills. Retrieved from https://www.makeuseof.com/tag/best-programming-games/ [Accessed 2 Apr. 2020].

R. W3schools.com. (2017). *W3Schools JavaScript Tutorials*. [online] Retrieved from https://www.w3schools.com/js/default.asp [Accessed 22 May. 2020].