Catcoders

UNDERGRADUATE HONOURS THESIS
COMPUTER SCIENCE, FACULTY OF SCIENCE
ONTARIO TECH UNIVERSITY
OSHAWA, ON, CANADA

Julian Finley

SUPERVISORS:
Jeremy S. Bradbury
Randy Fortier

April 26, 2022

Abstract

From the perspective of a computer science student, or a software developer, it can be easy to forget about where we began in this career path. The very basics from when you first started coding are often taken for granted, not because it isn't important, but because it's something we intuitively understand and never have to think of again. From this, new learners are often first taught these simple concepts in academics if they're majoring in a similar field. But what if they're not?

Coding courses and serious games are powerful and common contenders when one wants to learn from the ground up, but don't necessarily have experience or knowledge on the subject. The difficulty is that they often assume the player has at least some prior knowledge, which leaves the rest in an odd position. The serious game Catcoders attempts to alleviate this gap.

Catcoders doesn't try to throw players into the deep end, it instead does the exact opposite; exposes players to the very first concepts available through a thematic and a light story-driven scenario with interactive and puzzle-like gameplay. The player must learn and apply certain beginning concepts in order to satiate the game's main antagonist: Little Spot.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

When thinking about learning programming concepts, we often don't think of where it all begins: the core concepts, the ideas, and the foundations that link everything together. We think of learning a new language, a new framework, or a new environment, which is fine for the experienced, but what about the beginners and inexperienced?

Technology has always been an effective teaching and instructional method for those who are already familiar with the platform, typically the younger generations [23], but what if we were to include video games as a part of that. In the space of games, there are a lot of different options to choose from; games geared towards beginners, ones towards intermediate and above, and others that focus on gameplay first, teaching later, and vice versa.

Being in the field of computer sciences and having a personal interest in the world of gaming, it only made sense to pursue the topic of a game that teaches the very thing I had learned myself: computer programming. There are many applications in this space which do exactly this; they're all different and unique in their own way, and they all fulfill different roles, however some areas are more lacking than others. When it comes down to games that are targeted at beginners who know absolutely nothing about programming at all; those in a different major, or simply have never thought of or encountered coding, there seems to be a deficiency.

1.2 Problem

Programming is often thought of as a difficult subject to break into as it requires not only prior knowledge in computers, but also requires one to have good problem-solving skills and a decent foundation in mathematics. [24] The real difficulty lies in learning the concepts you may have never fathomed before. What is a variable, what is an operator, a new learner might be wondering. These are arguably the first concepts one may want to learn when first coming into this space.

The issue here is this is often not the case. Young future programmers may often be shown live code, coding environments, or expected to jump into writing code when they may not even know or understand the very basics yet. This is an issue that not only exists in standard tutorials, but also in games that are designed to teach as well. Here lies the problem we face.

1.3 Contributions

The game created as a part of this thesis offers a new and more unique idea of how programming could be learned from the very beginnings of one's programming journey. We can see that a lot of other products available, either for free or for sale, have their own approach and solution they are trying to solve, but this goes a little simpler.

This thesis sets out to provide a game free of coding, programming language independent, and instead promotes the idea of computational thinking through the most basic programming concepts. Additionally, while the word 'fun' may be subjective to many players of video games world-wide, this thesis aims to not only provide a learning experience, but also intends to be a fun and enjoyable experience, offering interesting mechanics and a light storyline.

Table 1: Feature Comparison chart

	Catcoders	code combat [6]	vim adventures [5]	codewars [3]	bitburner [16]	robocode [19]	botland [11]	while True: learn() [13]	Human Resource Machine [8]	CheckIO [1]	XSS game [14]	Screeps [20]	Swift Playgrounds [7]	Untrusted [18]	return true to win [10]	CodinGame [4]	7 Billion Humans [9]	JSbattle [12]	Coding Planets [15]	Lightbot [2]	Tanxy [17]
Expanations	●	◐	◐	-	◐	-	-	●	●	●	●	●	◐	◐	-	●	●	◐	●	◐	◐
Storytelling	●	●	-	-	-	-	-	◐	◐	-	-	-	-	◐	-	-	●	-	-	-	-
Puzzles	◐	●	●	●	●	-	-	●	●	-	◐	◐	●	-	-	●	●	-	●	●	-
Interactive	●	●	◐	-	●	-	-	-	-	-	◐	-	●	●	◐	-	◐	◐	◐	◐	-
Gameplay	●	●	●	-	-	●	●	●	●	-	◐	◐	-	◐	-	◐	●	◐	●	●	◐
Typed coding	-	●	-	●	●	●	-	-	-	●	●	●	●	-	●	●	-	●	-	-	●
Block coding	-	-	-	-	-	-	●	◐	●	-	-	-	-	-	-	-	●	-	◐	◐	-
Programming basics	●	-	-	◐	-	-	-	-	◐	◐	◐	-	◐	-	◐	●	◐	-	◐	●	-

● = property is sufficiently present; ◐ = property partially exists; - = property is not present;

2 Background

2.1 Overview

When we begin programming, we have no idea what is going on at first. Your very first programming class you may have attended taught the basics, maybe off a PowerPoint presentation, or maybe through the professor's experience, however, one thing is always consistent: the traditional

‘Hello World’ program. Usually, we are taught to write this code from lectures in our environment, and see if it runs fine, but what does it really mean? We are taught how to code, but not what those core concepts necessarily mean.

Lastly, what this very thesis is about is utilizing games as a platform to teach. [21] Programming is taught in different ways in a classroom; from being traditionally lectured to at the front, and given homework and assignments to do on your own time, to watching lectures on your own time and then receiving live instruction and help during lectures to complete schoolwork. [22] Additionally programming is taught on an online platform, most commonly through lessons which end in code executions.

2.2 The Beginnings of Programming

Basic computational concepts of programming are what this thesis project has sought out to teach to the prospective player. This starts with what I believe is important to every language and every application the player may go to create in their future. These being data types, variables, and iterations.

- Data types are typically divided into 4 separate groups, which are consistent across languages albeit with different names and uses:

    1. Integers represent whole numbers, which can be either positive, negative, or zero
    2. Decimals are fractional numbers, which can also be positive or negative
    3. String types store a sequence of letters and numbers

4. Booleans are denoted by being only true, or false

- Variables are used to store these data types, and once set cannot be changed in most cases

- Iterations, also known as loops, have two common types:

  1. While loops execute until the defined condition has been met

  2. For loops execute only for a predetermined amount of times

While planning the creation of the game this thesis is based on, the development Platform of Godot was initially chosen, and seemed to be the best choice for the application. Reasons being is that it is more specialized in 2d development, has faster run times and a simpler development language and environment. While these things all hold true, the community support for new game developers was seriously lacking, in addition to tutorial series being specific in scope. With this, it was time to try the Unity Engine.

Unity is a powerful tool, with a vast community of users in their forums, numerous tutorials for almost anything needed, and a great asset store to assist in a development journey. While the switch to Unity was difficult at first, it was an essential step and learning experience in the completion of this thesis project.

2.3 Serious Games

Serious games are what can be considered a sub-genre of regular video games. They are meant to have the familiar play and feel of a video game, but they include something much more important: education. Serious games are primarily a tool used for learning a wide range of topics and are specifically designed to teach a topic first and foremost. [25] However, just because education is the primary focus of a serious game, it doesn't mean that it cannot also be fun at the same time.

Being an alternative to other methods of study, serious games allow a different type of audience to find enjoyment in learning. Those who have grown up with video games or digital media may be able to relate with this style of learning better than in-class instruction. A game in this category typically includes feedback, telling the user whether they are on the right track or not. This can include pop-ups, scoreboards, or visual and audio cues. Enjoyability must also be considered when designing a game. If the player is not enjoying their time in the game, they are just learning another method they do not enjoy.

When planning this thesis project, enjoyability, feedback, and player engagement were of the utmost importance for what was to be coupled with the primary goal: teaching programming concepts.

3 Approach

3.1 Overview

The game designed and crafted to display these concepts that have been mentioned thus far is: Catcoders. Catcoders is an example of a serious game and is designed to teach the player the basic topics we covered before.

The user assumes the control of their player character and must complete the levels of all the prerequisite levels to challenge the final boss. Both levels cover one of each of the concepts of data types variables, and iteration. Through explicit dialogue explaining the situation, the player must follow the advice of the non-player character and complete steps within the level until they reach the last part; where they should have the knowledge to finish the level.

The boss level combines both concepts together, and visually displays each in a non-explicit way, to enforce what they had previously learned. This final stage is longer than the previous levels, but allows the player to effectively take a shortcut in order to complete it quicker.

3.2 Design Procedure

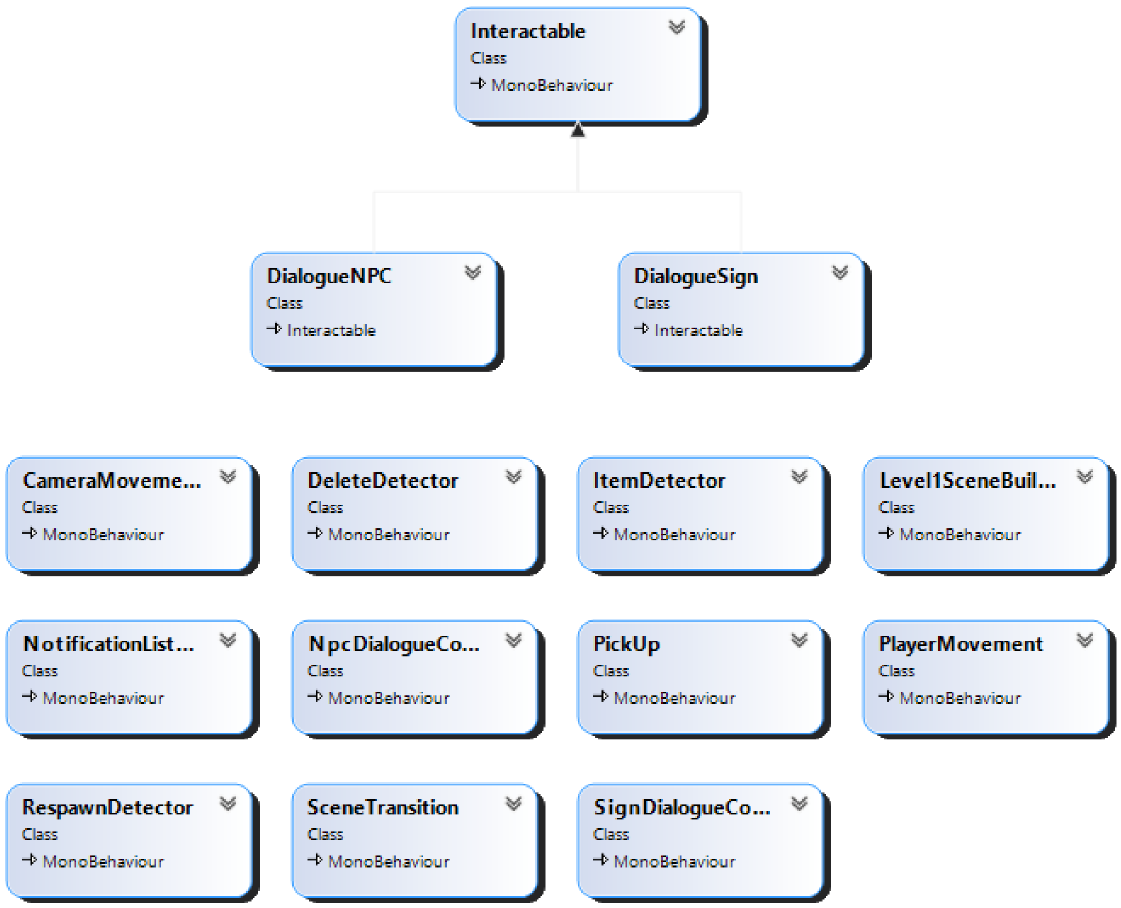

Figure 1: Class overview for level one

Before development even began, initial gameplay thoughts were written up, designs and UI ideas were drawn, and level designs for the different scenes were thought-out. These first drafts proved to be useful not because everything was fully included and fulfilled the design vision, but because it allowed the truly important parts to be picked out from the "nice to have" to the "must haves".

When first starting with the Unity engine, it was abundantly clear that many things from those first drafts needed to be taken out for time to first learn the system and become proficient at it. Most, if not all, of the filler was removed from each level to focus on core mechanics, gameplay, feedback, and the teaching portions.

Besides teaching the user the computational concepts, feedback was decidedly an important feature to promote whether what the user was doing was correct or not. Each level includes feedback in either a pop-up, a visual cue, or a loss of points to display this accurately.

Designing the code at first was a difficult task. While being experienced in C was an asset, the way Unity utilizes C was entirely different to prior experience. At first code was messy, duplicative, and without much structure. (See Figure 1) As development went on, this changed and the design by the end was a significant improvement.

3.3 Learning Process

Catcoders, rather than being designed for a specific age group, is targeted specifically for users who have no experience in development or programming at all. While age isn't an important factor, being able to understand the concepts at a high level and having some problem-solving

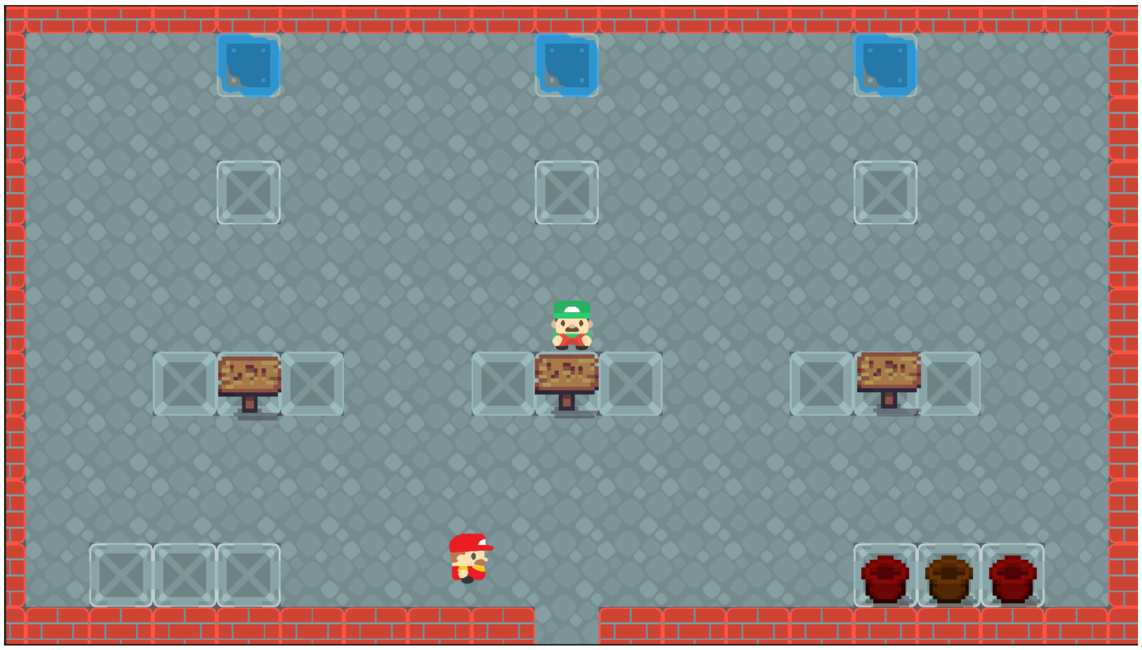

Figure 2: First level: first draft

abilities is a plus. Even though no prior experience or familiarity with the subject is required, this game's mechanics and gameplay could be enjoyable to the new and old in the world of computer science.

Due to the users being assumed to have no prior knowledge of computer science, the game is designed to hold the hands of these individuals during the first introduction to the previously introduced concepts present in each level. Mainly through the user of the NPC dialogue system, the user will first learn what the concept is, what it means, and then what to do with it in the context of the level. This process may happen on the first and second dialogue encounter, in order to reinforce what they have learned.

3.4 Gameplay

The first level of Catcoders features the concepts of data types variables.

- The level starts with the friendly NPC explaining the concept, the thematic situation involved, and how the player must help them to stop it. The player is given the opportunity to pick up a cat food bowl with a label on it and is prompted to place it down underneath a pipe. Afterwards the player should then interact with the corresponding control panel and select the correct option. Given the prior information, the player should know which to select. (See Figure 3)
- The second part has the player sort an additional three bowls after receiving additional information from the NPC.(See Figure 4)
- Lastly, with all the knowledge of data types and variables being taught, the final part of the level comes into play, and the player must then score 20 points to pass. Each correct answer awards 2 points, and each incorrect answer subtracts 1 point. When reaching the threshold, the player is then able to speak with the NPC one last time to complete the level.

The feedback associated with this level is based upon correct or incorrect choices, a notifier if no bowl is present under the pipe when the control panel is interacted with, and a status bar for points. The second level features iterations.

- This level starts in a similar way, there is another unfortunate situation the player must solve by listening to their explanations and learn the new content. First, the player will learn an

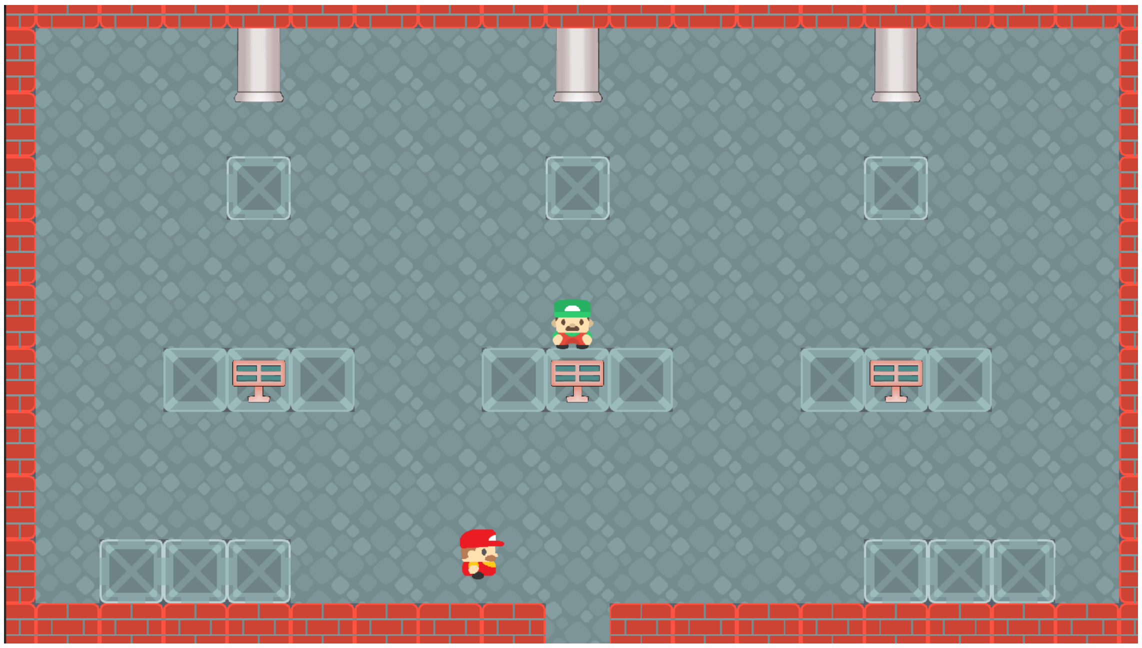

Figure 3: First level: final design

infinite loop is happening, and must press a button to shut down the operation of the factory they are presently in.

- For the second part, the player must take part in restarting the factory to produce higher quality scratching posts. The goal of this part is to fill the loading spots with a certain number of objects which are moving along the conveyor belts. This number is received by reading the control panels beside their corresponding loading spot. When the conditions are met, the while loop producing the objects shuts down once again.
- For the final part, the player is shown an example of a for loop. One item at a time comes down the conveyor belt, and the player must sort it accordingly, like the previous stage. After fulfilling the requirements marked out by the control panels, the level will be completed.

The feedback associated with this level are status bars for each required item for the loading

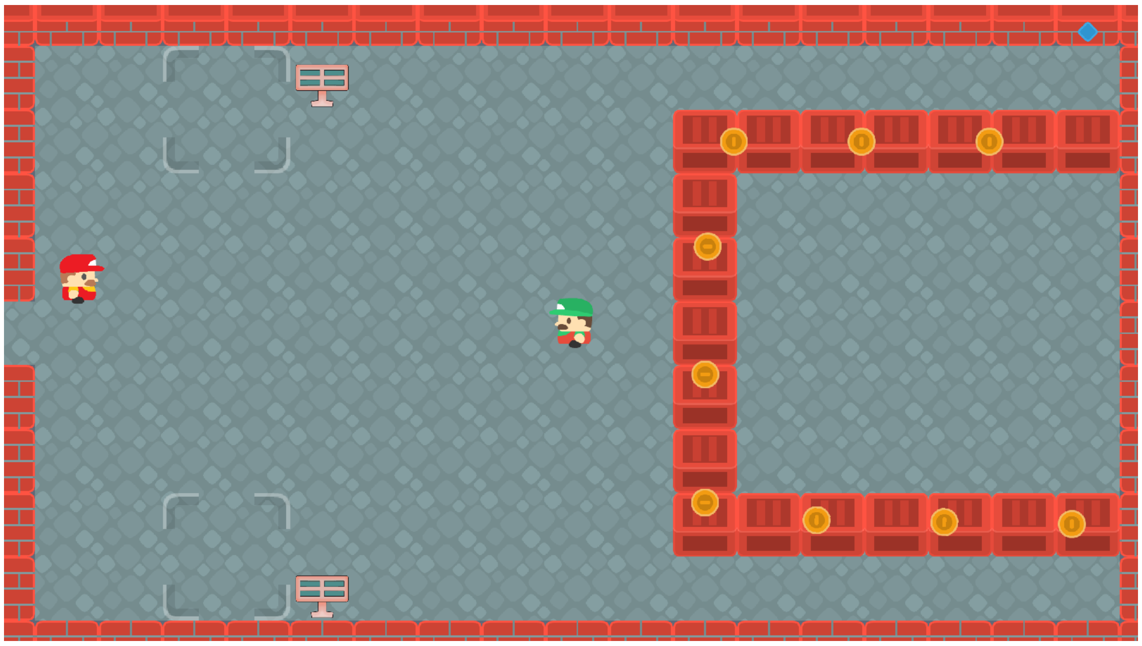

Figure 4: Second level: final design

spots. The players will know if they performed correctly when it increments, and when they complete a stage.

Lastly, the boss level will test the players familiarity with what they have learned. There is no dialogue present, and the level starts immediately after entry.

The level features the boss in the middle of the scene, with four looping orbits around the center of the boss. This represents a 4-layer nested for loop. The objective is for the player to match the corresponding items to whittle down the amount of orbit objects present, representing data types matching with variables. After eliminating a layer in its entirety, the player will be able to remove an object from the next layer – the trick is, after removing one or more, the previous layer will always regenerate unless the removed item was the last one in the orbit. After playing through to the point of removing everything, the boss will be defeated, and the player may enter the center to trigger the end game screen.

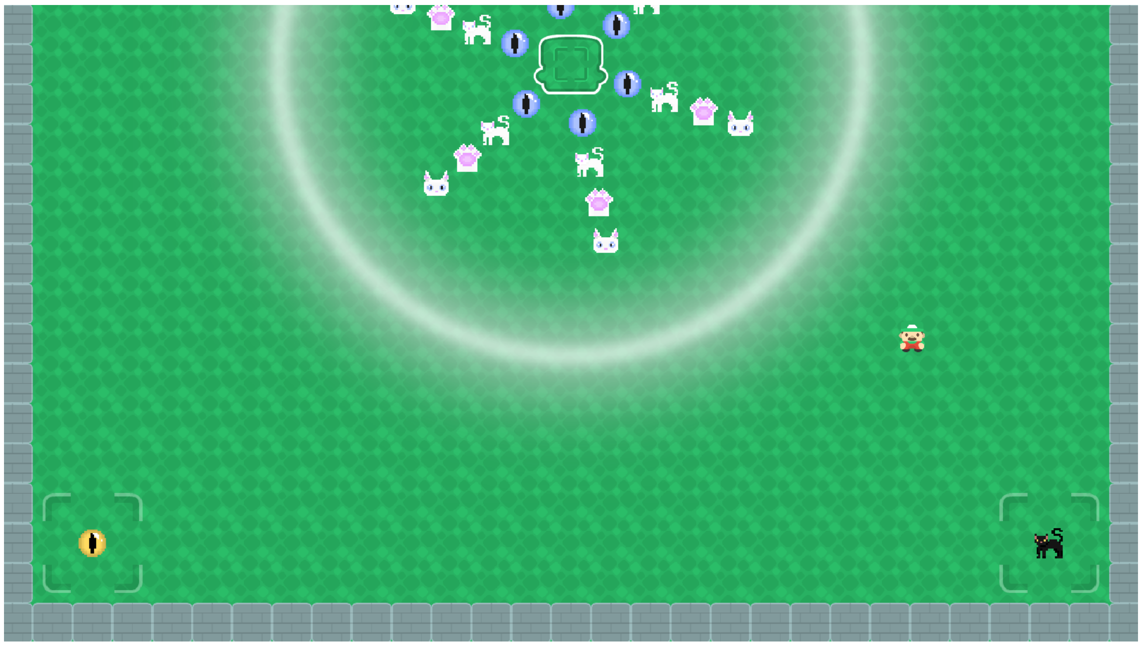

Figure 5: Final level: final design

3.5 Level Design

Each level displays a different type of game mechanic unique to that particular level, however the outcomes of each are different. While the intent is still to learn, what and how they learn are different.

The outcome for level 1 and level 2 are designed to display the previously mentioned beginner concepts: data types variables and iteration. For the final level, the outcome is crafted to combine both concepts from the prior levels,thus the player is only able to attempt it once both are completed. Using both prior concepts, the player must be able to figure out what to do on their own without any help.

There was an additional level originally planned to be placed after level 2, and before level 3, which was cut due to time constraints.. This level planned on including information pertaining to conditional statements. The goal, like the other learning levels, was to first learn and then display this knowledge. The final level would have also incorporated these learnings into the fight.

Lastly, an optional level explaining operators was included in initial drafts and would not be required to complete the game.

Both cut levels did not have any mechanics or gameplay planned out and were only concepts.

4 Conclusions

4.1 Summary

The design thesis Catcoders overall has been a rewarding experience. The three levels that have been crafted to teach players the concepts sought out from the beginning, have been made in a way that is thoughtful and entertaining, and hopefully one is able to learn from this final product.

While there were portions of the game that had to be cut out for reasons mostly associated with time, this thesis project did what it sought out to do: make a proof of concept for something the space of serious games needs.

4.2 Limitations

The major impediment that ended up limiting the scope of Catcoders was a general lack of time. With more time, or less exterior distractions, it would have been very well possible to complete this project to its fullest extent.

When considering time, drafting plans for a product that will be completed 8 months into the future seemed doable, like there was all the time in the world – but this couldn't be further from the truth. While planning too much too early wasn't tremendously detrimental, a lot of plans ultimately did not come to fruition. The lessons to take from this is: plan simple then expand.

A lot of developmental time taken was put aside for learning. Learning the engine, and all the systems involved were vital to making a successful product in the end, and this could easily account for a couple months or more on the project length. During those first few months, not much content was added, changes were slow, and components added may have never made it to a commit. This time was not for nothing though, it allowed the later stages of the project to be completed in a much faster timeframe than expected.

4.3 Future Work

There are six avenues of improvement that should be explored for when considering the future work on this project: The omitted level involving conditionals, new features and difficulties, UI and HUD upgrades, art and assets, a full project rewrite and optional additional content.

- The previously mentioned level involving a new concept of conditionals will be the first thing added to the completed product, as well as alteration of the final level to include aspects of this as well.

- Introduction of new difficulties, timed challenges, and randomization would add much needed variety to mix up the gameplay for each level. For instance, having the player make no mistakes during level one, or having the player complete the final parts of level two within a certain time would be examples of good additions.

- Originally intended to have UI features for an inventory, menus, achievements, wardrobe, and a journal; these cut menus and the features corresponding with them would enrich the player experience and add rewards for playing the game. Additionally, a character creator could be added to customize the character appearance, pending asset acquisition.

- Asset and art are something that would add a unique and thematic feel to the game which is not currently present using placeholder assets. Ideally, art which is not used elsewhere would be preferable, and would additionally likely incur a cost to commission and implement.

- Coming upon optional changes, a total project overhaul and rewrite would make coding and project structure consistent across the whole platform. Using a modular approach would allow additions to be made more easily, and may additionally allow the game to run smoother and quicker than before

- Lastly, optional additional content, which was not already planned or thought of before could be added after everything else. This could include further levels, combat, collectables, etc. – the possibilities are endless

References

[1] Checkio: coding games and programming challenges for beginner and advanced.

[2] Lightbot: Solve puzzles using programming!, 2008.

[3] codewars: Achieve mastery through challenge, 2012.

[4] Codingame: step up your coding game, 2012.

[5] Vim adventures: Learning vim while playing a game, 2012.

[6] Code combat: Learn to code through the power of play, 2013.

[7] Swift playgrounds: Learn serious code, in a seriously fun way, 2014.

[8] Human resource machine, 2015.

[9] 7 billion humans, 2018.

[10] Return true to win, 2018.

[11] Botland, 2019.

[12] Jsbattle: Javascript programming game, 2019.

[13] while true: learn(), 2019.

[14] Xss game, 2019.

[15] Coding planets, 2020.

[16] bitburner, 2021.

[17] tanxy, 2021.

[18] Untrusted, 2021.

[19] Robocode: Build the best - destroy the rest!, 2022.

[20] Screeps: Mmo sandbox game for programmers, 2022.

[21] Ashish Amresh, Adam R. Carberry, and John Femiani. Evaluating the effectiveness of flipped classrooms for teaching cs1. In *2013 IEEE Frontiers in Education Conference (FIE)*, pages 733–735, 2013.

[22] Sébastien Combéfis, Gytautas Beresnevičius, and Valentina Dagienė. Learning programming through games and contests: overview, characterisation and discussion. *Olympiads in Informatics*, 10(1):39–60, 2016.

[23] Rania Elshiekh and Laurie Butgerit. Using gamification to teach students programming concepts. August 2017.

[24] Tony Jenkins. On the difficulty of learning to program. 2002.

[25] Michael A. Miljanovic and Jeremy S. Bradbury. A review of serious games for programming. In *Proc. of the 4th Joint Conference on Serious Games (JCSG 2018)*, Nov. 2018.