

Combining Parsons Problem and Serious Games for Learning Programming

Onyedikachi Chimmuanya Kalu

April 2020

1 Abstract

Parsons problem involves providing a user or student with blocks or lines of code, the goal of reordering them to create a specific solution. Information Technology is constantly evolving. With newer ideas, fascinating implementations of several software products are rapidly being produced. With all this advancement, interest in learning programming fundamentals is at an all time high and is slowly becoming a necessity. Parsons problems are designed to test a students understanding of programming without the added pressure of creating code from scratch. It is also a useful tool for introducing basic programming concepts to newly aspiring programmers. For this reason, it is important to explore the different implementations of parsons problem; to really understand why it works and how effective it is, as a teaching tool.

This leads to the next key point, Serious Games. A serious game is a game designed for a specific or primary purpose other than pure entertainment. [6] A more relevant explanation would be, a game that was made to teach a certain idea or concept. extensive research has shown the effectiveness of Serious Games in teaching several concepts, both coding and non- coding concepts. Combining parsons problem with serious games is what led to the creation of "Coach Script". The goal, of the game is to use code blocks to execute a specific basketball sequence. The sequence varies in difficulty and all options of implementation were explored over an eight month period.

Keywords: parsons problem, serious game, programming, students

2 Motivation

For most people, learning to program can be very challenging. Drop out and failure rates in many introductory programming classes at the college level is astoundingly high. In a study with twelve colleges and fifty universities, the passing rate for introductory programming classes where 88 percent and 66 percent. The result of the colleges can be put aside because of the small sample size, while the university pass rate is more reliable. To break things even further,

the programming courses were broken down into object-oriented, imperative, functional or other. The results still produced the same identical pass rates.[1] Parsons problem can not only help new programmers, but novices as well. Their knowledge is deeply tested and could lead to them learning to fix errors and understand simple algorithms.[5] When analyzing the results of using parsons problem as exam questions, the results showed students performed significantly better than writing code. Most importantly, the students in the lower percentile performed better; 23 percent to 58 percent. The examiners claimed it was easier to spot the areas of struggle for students.[3] Parsons Problem allows students accomplish school work faster than students using traditional code writing, and they perform no worse at assignments.[10] Research suggests that engagement is the motivation for using serious games. When developing a serious game, the difficulty level needs to mimic the users intellect; harder for novices and easier for new users.[6] This reasoning applies for parsons problems; the exercise difficulty should increase at a rate that keeps the students engaged.

3 Problem

Game development comes with its different complications, one of which includes keeping the audience engaged. The addition of parsons problem implementation makes this even harder. The objective of teaching students would be a failure if the game fails to keep them engaged or is deemed too easy. A handful of thought and research needs to be done to design each level's difficulty and feedback. In addition, the choice of how to implement basketball terms in a parsons problem game is quite challenging. The sport is very complex and involves a lot of moving pieces that can be confusing or impossible to implement at this level.

4 Contributions

This thesis offers contributions, mainly, in the area of Computer Science education. We have designed and developed a serious game that can assist students in learning and understanding basic programming principles that could be hard to understand using regular teaching methods. Coach Script is unique because it is one of the few games that combine parsons problems and basketball into a serious game. The research done to create this game, opens up the doors for further designs of parsons problems games, with multiple sports, and different story lines or background.

5 Parsons Problem

Parsons problems are essentially puzzles, that students need to piece together to assemble a working program. It allows students to focus on solving the problem rather than constructing a solution, searching for needed components, and merging them together[10]

Another aspect to consider is Adaptive parsons problem. The Adaptive practice is where the problems are adapted based on the learners prior performance. Barabra Ericson invented two types of Parsons Problem; intra-problem and inter-problem. In intra-problem adaptation, if the student is struggling, the problem is dynamically modified, and made easier to solve. In inter-problem adaptation, the next problem's difficulty is modified based on well they did on previous problem.[5] For optimal learning, the problems should keep the students in Vygotsky's zone of proximal development. [2]

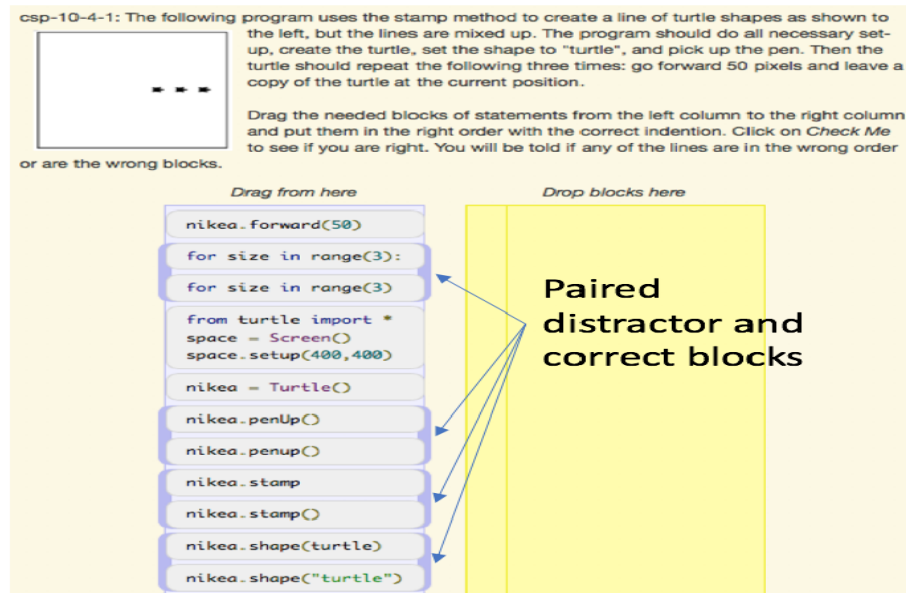
In order to properly achieve adaptive parsons problem, feedback and progress evaluation are very important. For a every mistake made by the student, the problem needs provide enough feedback to allow the them understand their mistake and make the adequate corrections. Evaluating progress can be done in different ways, some of which will be explored later.

6 Intra-Problem Adaptation

In Intra-Problem Adaptation, the problem is dynamically made easier if the student is struggling. The problem is simplified by disabling distractors, combining code blocks or providing indentation. The easiest way to implement this adaptation would be to have a "help" button available to the students.[5]

In this adaptation the key is feedback, the students need to see the transformation of the problem. Figure 1 shows a parsons problem; in this case the distractors are paired with the correct blocks. A perfect way to dynamically help, would be to slowly show a distractor unpaired from the correct block, or have it fade away. This way the students can visually track the changes and understand why that code block was incorrect.

csp-10-4-1: The following program uses the stamp method to create a line of turtle shapes as shown to the left, but the lines are mixed up. The program should do all necessary set-up, create the turtle, set the shape to "turtle", and pick up the pen. Then the turtle should repeat the following three times: go forward 50 pixels and leave a copy of the turtle at the current position.



Drag the needed blocks of statements from the left column to the right column and put them in the right order with the correct indentation. Click on *Check Me* to see if you are right. You will be told if any of the lines are in the wrong order or are the wrong blocks.

Drag from here

- nikea.forward(50)
- for size in range(3):
- for size in range(3)
- from turtle import *
- space = Screen()
- space.setup(400,400)
- nikea = Turtle()
- nikea.penUp()
- nikea.penup()
- nikea.stamp
- nikea.stamp()
- nikea.shape(turtle)
- nikea.shape("turtle")

Drop blocks here

Paired distractor and correct blocks

Figure 1. Intra-Parsons Adaptation [5]

7 Inter-Problem Adaptation

In inter-problem adaptation the user’s performance on the previous problem is used to modify the difficulty of the next problem. Assessing the user’s performance can be done by checking the attempts taken to solve the problem. If the user solved the previous problem in one attempt, the next problem could be made harder by adding more distractors and randomizing their placement. Figure 2 shows a problem with all distractor blocks randomly mixed in with the correct code blocks. This is an example of the highest difficulty version of this problem.

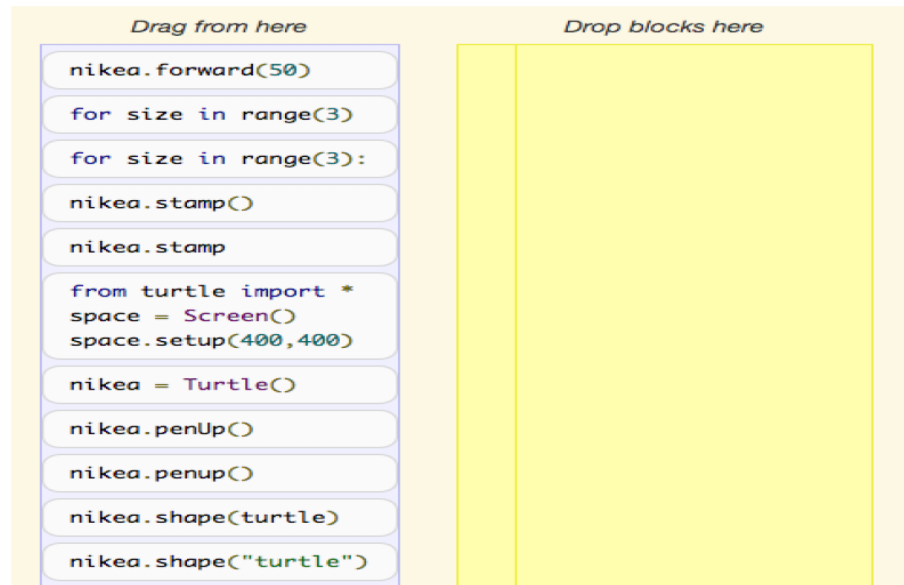


Figure 2. Inter-Parsons Adaptation [5]

8 Serious Games

Serious games have been around for a long time.[4]. The best definition is, “any form of interactive computer-based game software for one or multiple players to be used on any platform and that has been developed with the intention to be more than entertainment” [9]

Studies like the Teachers Evaluating Educational Multimedia report show that, teachers and parents have recognised that computer games can develop skills like; strategic thinking, planning and communication, application of numbers, negotiating skills, group decision-making and data-handling. With that being said, computer games have been effective at improving performance in

areas such as Science and Math, Language and Computer Science. [7] In an attempt to help high school students understand variables in programming, the Super Mario Collaborative game was designed. "This game is structured in 4 levels, and students are divided into 4 groups, where each group focuses on a specific level that is dedicated for the learning of a specific class of variables. The activities are interactive, constructive, collaborative, and structured so as to excite the curiosity of students to experience and understand different aspects of variables in programming." [8]



Fig. 3a. A snapshot of level 3

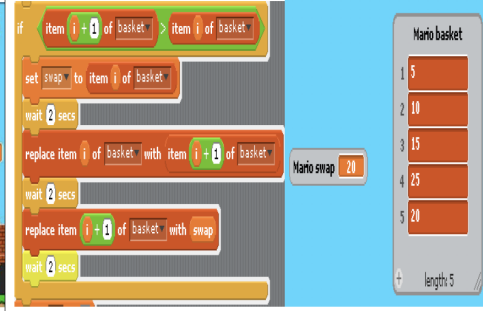


Fig. 3b. A snapshot indicating that position of value 25 replaced with value 20, which is saved into the variable swap

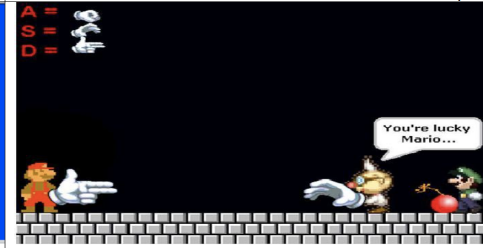
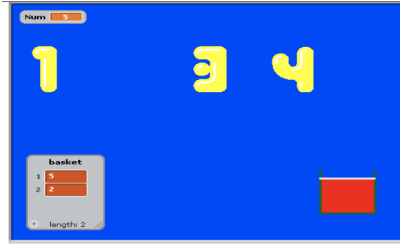


Figure 3. Super Mario Collaborative game [8]

Serious Games and Parsons have proven to be helpful in teaching programming to students. Compared to traditional learning, they have proven to increase motivation and engagement levels. Coach syntax is an attempt to combine the benefits of both, to facilitate better understanding of programming.

9 Coach Syntax



Figure 4. Coach Syntax

This version of Coach Syntax, was designed to teach sequential programming. The game plays the role of a 'coach', teaching the players how to code. The coach shows you a basketball sequence to replicate with programming. The game focuses on high level moves that any passive basketball fan will understand; pass, dribble, run, shoot, layup, dunk, etc.

Each level begins with a video of a basketball sequence, the player needs to replicate with code. The sequences are represented in code blocks that need to be re arranged. The player's job is to drag the code blocks in the drop-zone area in the right order. On the right side of the screen, there is a replay center. This serves as a visual aid, to sequentially run through the code block order, the player arranged. Each code block is represented by a corresponding video showing the action on the replay centre, as well as a 'wrong' video indicating the code sequence is incorrect.

As the levels increase, the length of the basketball combinations increase and so does the number of distractors. The distractors became the primary method of assessing understanding; in order to differentiate the distractors, players would need to watch the basketball sequence videos multiple times.

10 Game Design

Coach Syntax was designed in Unity, using C programming language. Early versions of the game were text based, and involved a lot of brainstorming to create the basketball sequences for the levels.

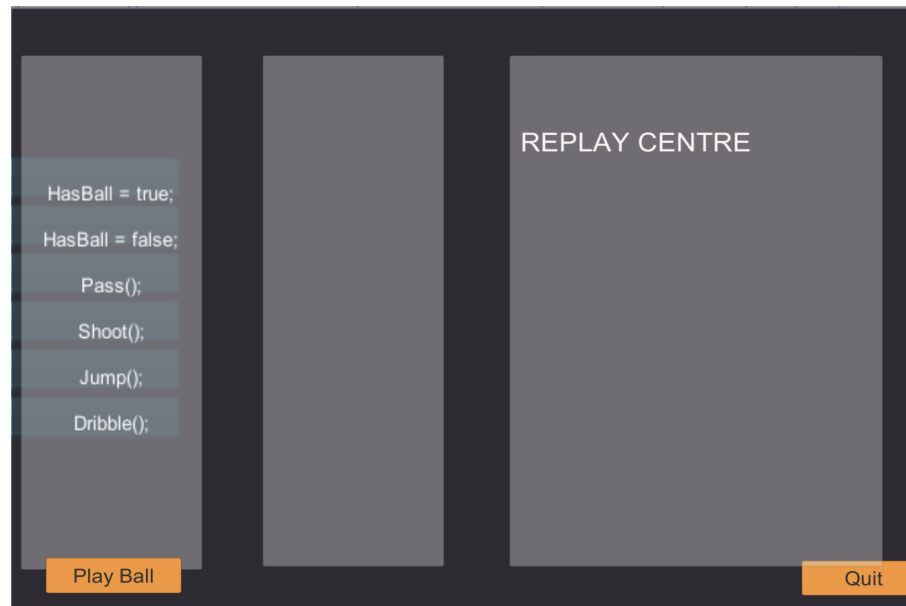


Figure 5. Coach Syntax: Midterm Prototype

This design, was made to facilitate the basic requirements for a parsons problem game. The scene was divided into three panels; pickup, drop-off and replay panel. This version had no video feedback, and operated purely text based.

11 Learning Objectives

Coach Syntax was designed for first year Computer science or introductory programming classes. My rendition of the game focuses on sequential programming; this topic is one of the first concepts new programmers need to understand and often have problems grasping. Without this important foundation, students are almost destined to fail, especially with no prior programming background.

The aim of coach syntax is to link peoples understanding of sports, to potentially understand programming. So as long as they can comprehend basic basketball skills, they should be able to understand programming. For example the image below shows a video of someone shooting a basketball. This is a simple and basic skill that any passive basketball is familiar with. In this scenario, it would be easy to translate the video into 2 sequential functions jump and shoot.

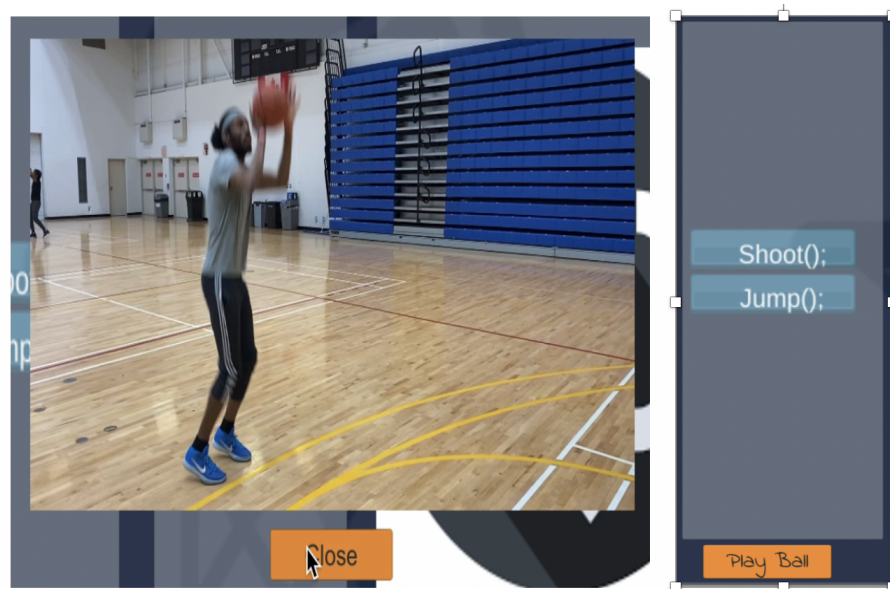


Figure 6. Jump shot Code Blocks

12 Gameplay

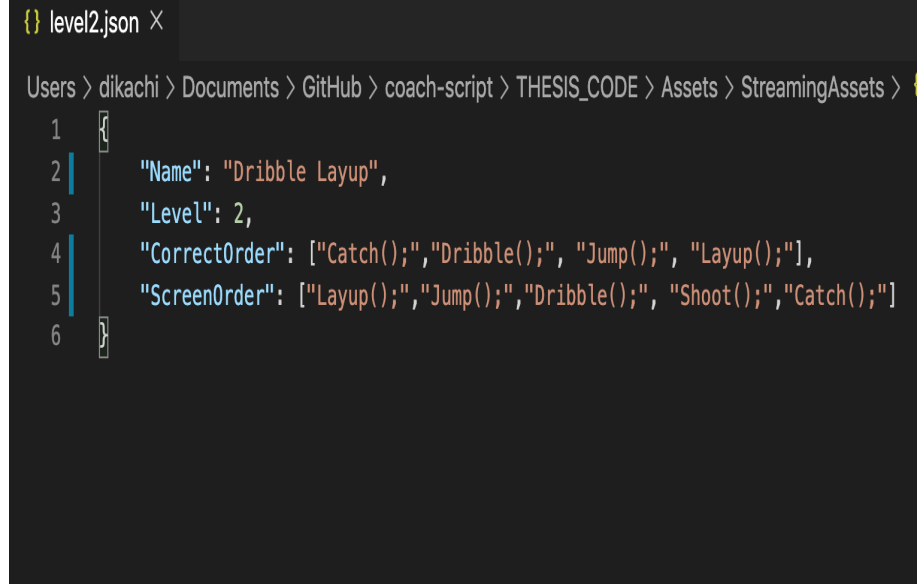


Figure 7. Coach Syntax Game

Analyzing the gameplay screen, it was important to add certain features we see. There is a pick up and drop off area. The initial plan was to have only one

area, where the players would simply re-arrange the code blocks. That plan was scrapped because of the importance of distractors. The game's teaching relies heavily on the players ability to differentiate between the correct blocks and the distractors, so a drop off zone was created.

The Third section is the replay centre. A large amount to time and effort was put into designing the logic of this panel. The replay centre takes the order of the code blocks placed in the drop off zone and sequentially plays the corresponding video of the function.



```
{ } level2.json X
Users > dikachi > Documents > GitHub > coach-script > THESIS_CODE > Assets > StreamingAssets > level2.json
1 {
2   "Name": "Dribble Layup",
3   "Level": 2,
4   "CorrectOrder": ["Catch();", "Dribble();", "Jump();", "Layup();"],
5   "ScreenOrder": ["Layup();", "Jump();", "Dribble();", "Shoot();", "Catch();"]
6 }
```

Figure 8. JSON level2 file

The 'Play Ball' button was designed to check the player's answers. This button collects the order given and compares it to the correct order stored in a JSON file on the backend. It creates a list that the Replay Centre uses to display the videos, including or excluding the 'wrong' video.

Lastly, the 'PlayBook' button was created as an everlasting guide, to refer back to the basketball sequence video, the player is required to program. As the basketball combinations get more complex, the need to refer back to the video is also increases. The final level of the game is a prime example of the importance of being able to watch the video multiple times.

13 Level Design

Designing the levels for coach syntax, might the been the hardest thing about the project. To specify, this refers to defining the basket ball sequence for each level. It needed to follow a specific pattern that the players would understand, and keep more importantly, keep them engaged.

There were many drafts that didn't make the final product, simply because

they were either too simple or too hard. The combinations needed to appeal to average basketball fan understanding of the game.

Level	Number of Distractors	Number of Code Blocks
1	0	2
2	1	5
3	2	10

Figure 9. Levels Breakdown

The key was to increase the difficulty of the levels gradually. The first level was designed to be as simple as possible; it has no distractors and only 2 blocks to arrange. There is literally a 50 percent chance of getting this right, and if the player gets it wrong, the 'PlayBook' can show them why their solution is incorrect.

Level 2 was the hardest to construct. The jump in difficulty needed to be minimal, but still stay connected to the principle of level 1. A distractor was added; this tests how well the player understand the basketball video. Discovering the distractor block might require multiple looks at the 'PlayBook' but once it is located, it becomes very obvious why it was placed there.

Once the first two were created, the pattern had been set for how the final level would be designed. There are 2 distractors and ten code blocks to sort through. In this level the distractors do not play as big of a role as before, this level tests how well, the player can breakdown the singular basketball moves of the video. This is the final test to see if the players understand how the basketball moves relate to the programming blocks, and it sets the stage for possibly adding loops and functions in following levels.

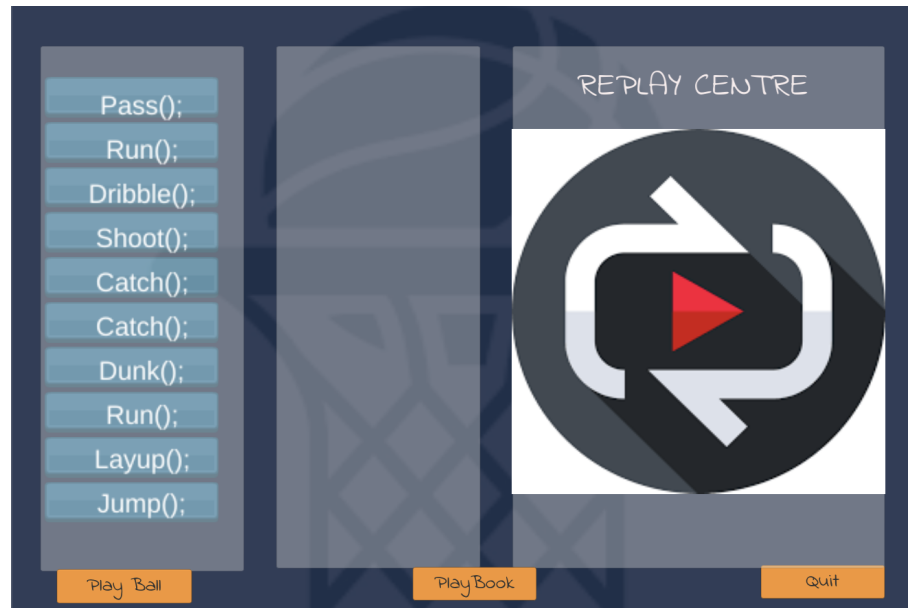


Figure 10. Level 3

14 Summary

Due to important issues, there was not an opportunity for Coach Syntax to be reviewed by first year students or new programmers. However, Coach Syntax opens the gate for Parsons Problem research involving sports. It harnesses people's obsession with sports to learn programming fundamentals. The understanding of each problem relies heavily on how well the players understand what is happening in the videos. Coach Syntax intentionally focuses on basic sequential programming, in order to establish an adequate baseline with the players understanding of basketball. With more practice and research Coach Syntax can be greatly improved, especially regarding the types of code blocks and possibly focusing specific computer science concepts.

15 Limitations

Initial designs involved Coach syntax using Two dimensional Animations instead of human videos, but that was not feasible. It would take a lot of effort and money, that supersedes the scope of the project. Although the basketball videos were used, it took a long time and involved a lot of time managing to the footage, and for further levels would require even more people in the videos. A more evolved version would either be costly or require outside video crew help.

Future Work Coach Syntax is a basketball game, but can easily be applied to different sports. The implementation would be similar but the approach

would need to match the selected sport’s style. Basketball was selected as the core of Coach Syntax because of its single player notoriety, using a different sport would need require a keen understanding of how the sport translates to sequential programming.

Coach Syntax is a single player game, but a multiplayer feature could be a possibility. Players could either play against or with each other. Time based scores could dictate the winner or it could be an opportunity to teach pair programming to the students.

Lastly, ‘help’ features could be included. Coach Syntax could be amended to a Intra or Inter Parsons Problem, where the difficulty of the level could be changed during the game depending on the players performance. A help button could be added to the game, to indicate the player is struggling with the question. This would serve as a way to evaluate students performance during the game, and pin point the areas they may be struggling with.

References

- [1] Jens Bennedsen and Michael E Caspersen. Failure rates in introductory programming. 2007.
- [2] Laura E. Berk and Adam Winsler. *Scaffolding Children’s Learning: Vygotsky and Early Childhood Education*. National Association for the Education of Young Children. 1995.
- [3] Paul Denny, Andrew Luxton-Reilly, and Beth Simon. Evaluating a new exam question: Parsons problems. 2014.
- [4] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining “gamification”. 2011.
- [5] Barbara Ericson, Austin McCall, and Kathryn Cunningham. Investigating the affect and effect of adaptive parsons problems. 2019.
- [6] Michael A. Miljanovic and Jeremy S. Bradbury. A review of serious games for programming.
- [7] M. Papastergiou. *Digital game-based learning in high-school computer science education: Impact on educational effectiveness and student motivation*. Computers and Education. 2009.
- [8] C. Theodorou and M. Kordaki. ”super mario: A collaborative game for the learning of variables in programming,” international journal of academic research, vol. 2. 2010.
- [9] Ritterfeld U., Cody M., and Vorderer P. Serious games: Mechanisms and effects. 2009.
- [10] Rui Zhi, Min Chi, Tiffany Barnes, and Thomas W. Price. Evaluating the effectiveness of parsons problems for block-based programming. 2019.