

Dragon Architect Goes to College: Teaching Carleton Students Computational Thinking Skills

Catalina Alvarez-Ruiz, Kate Grossman, Rie Kurita, Ellie Mamantov, Starr Wang
Advised by: Aaron Bauer

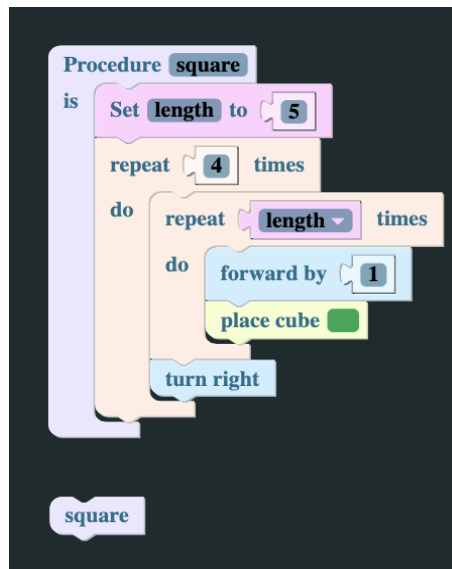
March 15, 2021

1 Introduction

1.1 Background

Dragon Architect is an educational game written in HTML, CSS, and JavaScript. It was created at the University of Washington by Aaron Bauer and his colleagues as a tool to investigate key open problems in the design of games that teach computational thinking (Bauer et al., 2017). To teach programming concepts, Dragon Architect utilizes block-based programming, which allows students to drag and drop blocks to create a program. The functionality of block-based programming is built from a library called blockly. Figure 1 displays an example of block-based code from Dragon Architect.

Figure 1: Snippet of block-based code from Dragon Architect that builds a square



Educational games like Dragon Architect can increase representation of minorities in computer science. Many students, especially non-male identifying, lack confidence in their ability to learn computer science. This lack of confidence often prevents them from ever trying. By playing educational games, students are able to build confidence in their own skills in a more accessible format. Playing a game can also change a student's perspective on the field of computer science in general. For example, if the perception of computer

science is that it is a field more appropriate for men, a game that is specifically created to appeal to all genders can help change that misconception (Hoegh & Moskal, 2009).

Finally, a game like Dragon Architect can increase interest in computer science by providing students an opportunity to discover their interest in computer science before diving head first into coding. It can provide a buffer between not knowing any programming concepts and becoming a more advanced programmer.

There were three major goals for the original game. The first was to teach computational thinking strategies. The original game taught students about loops and procedures, two fundamental computer science concepts. The second was to achieve a balance between creativity and direct instruction. In other words, the original designers strove to allow students to both explore creatively and learn on their own while also providing structured lessons. This goal was achieved in the original game by having a creative mode where players could build anything they wanted without limitations such as level goals. The third goal was to incorporate engaging social features to make students actually want to play Dragon Architect.

1.2 Goals of Comps Project

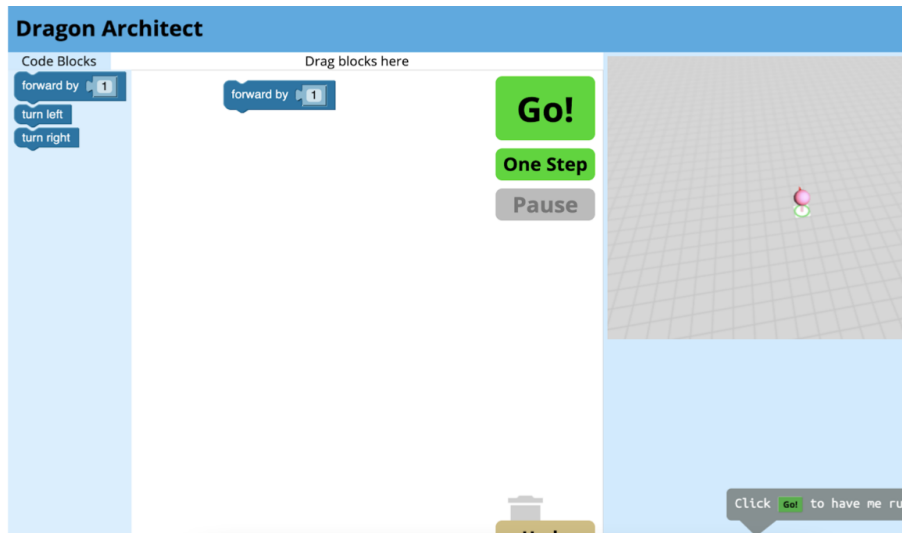
For our comps project, we decided to expand and alter Dragon Architect to tailor it to college students who are new to programming. Because we ourselves are currently college students, some of whom have experience prefecting and lab assisting for intro-level classes, we thought we would have strong insight into the needs of college students learning how to code. From our experience, we know that many students are intimidated and overwhelmed by coding when they take their first class. We thought that a block-based game like Dragon Architect would provide a less intimidating alternative to first learning programming concepts. Our goal was to make the game fun to play while also teaching key concepts, such as loops, variables, and functions.

We broke down this overarching goal into two parts: the front end and the back end. The front end refers to the user interface and “look” of the game, whereas the back end refers to the more structural components of the game such as the level design and concepts taught. Our main goal for the front end was to change the design from looking like an outdated educational game to looking like a modern, cool video game. Our main goal for the back end was to update the concepts to better suit a college-level audience. There were a few major issues with the front end in the original game, which is shown in Figure 2. The main one, in general terms, was that it looked like an educational game created by researchers as opposed to a fun game created by designers. One way that this showed was in the color scheme: the original game had a bright white background. Modern interfaces have transitioned away from white themes in favor of dark themes. Dark themes not only look cooler, in our opinion, but they are also much easier on the eyes. For a user playing this game for a long period of time, we wanted to make sure that the colors were not distracting or uncomfortable.

Another major issue with the front end was the graphics. There was basically no design within the graphics window: it was just a blank white background with a little pink ball in the middle as the avatar. Where was the dragon? Well, there was actually a dragon avatar but not in the graphics window; rather, the dragon floated around the screen to give instructions and hints. The dragon was not doing any architecting at all. Coincidentally, we also disliked the floating dragon because it was distracting, unpredictable, and would cover up parts of the screen. So we decided to scrap the floating dragon and redo the graphics. Instead of a pink ball we wanted a real dragon avatar, and instead of a blank white background we wanted a realistic nature scene. And, to replace the function of the floating dragon as the giver of instructions and hints, we wanted to add a stationary “goals and hints” box. There were a few other small problems with the original front end, such as the fonts and a jarring cheering sound that played at the end of each level. Our front end goals included fixing these smaller issues as well.

In the back end, we wanted to update the flow of the game, which was confusing to users in the original design. The original game had a “sandbox mode” feature, which was a free-form creativity mode in which the user could build anything they wanted. This creativity mode seemed cool to us in theory, but college students ignored it and were annoyed when they were automatically placed into sandbox mode. The game launched right into sandbox mode immediately after the tutorial levels, which always threw users off guard because they had no idea what to do. There was no clear information about what the intention of this mode

Figure 2: The Original Front End Interface of Dragon Architect



was or how to exit it and re-enter the levels. The user had to search all over the screen in order to figure out how to complete more levels, and this issue came up again and again when transitioning between level packs. We quickly realized that a clearer and more directed level flow was necessary.

The original game also introduced a limited number of computing concepts due to the fact that it was designed for middle schoolers. There were certain concepts that are important in college-level computer science classes that were not covered in the game. One of our major goals was to make the game feel more relevant for college students, so we wanted to add more concepts that are taught in Intro CS courses. We also decided that we wanted to add the option to see the Python code that corresponds to the blocks. This way, college students would be able to more explicitly create the connection between this game and actual coding.

2 Methods

2.1 Implemented Changes

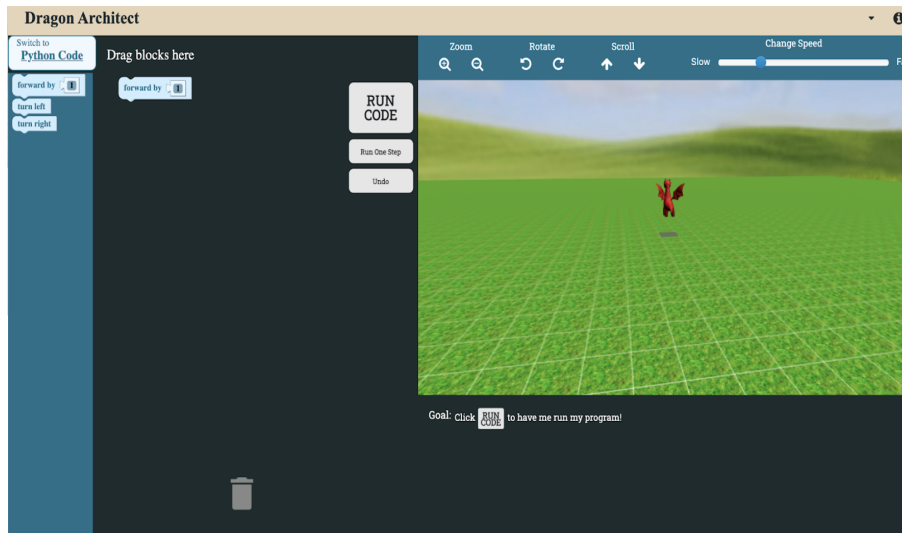
2.1.1 Front End

We updated the color scheme from a white theme to a dark theme, which included changing not only the background color but also the complimentary colors. We added a real dragon avatar and nature background to the graphics window, along with the ability to zoom in and out and change the “camera” perspective of the graphics. We removed the floating dragon and added a goals and hints section to the area below the graphics window. We also changed the overall layout to favor the graphics window (the rationale was to make the game more visually appealing, but we ended up not liking this change because it left too little room for the blocks). We changed the fonts and some of the wording (e.g., “Go” → “Run Code”) to feel more appropriate for an older audience. We also redesigned the navigation bar to be clearer and look sleeker. The final product of these changes can be seen in Figure 3.

2.1.2 Back End

After speaking to Carleton CS professors, who gave their perspectives on what concepts were challenging to intro CS students, we ended up adding variables as our new concept. This meant adding a new level pack dedicated to teaching the concept of variables through multiple levels. We also implemented the

Figure 3: The Updated Front End Interface of Dragon Architect



display Python code feature to allow for dynamic presentation of the Python version of whatever blocks are currently being used. To streamline how users navigate through the game, we chose to remove sandbox mode altogether. We felt that having creative "challenge" levels at the end served a similar purpose without being too open-ended and confusing. We also restructured the way that the level flow was presented to the user. Instead of the user having to figure out how to navigate to the next level pack, the game now automatically leads the user through. This structural change has been crucial to the user experience by allowing the user to independently move through the game without confusion.

2.2 Formal User Study

To evaluate whether the changes we made to the game accomplished our goal of developing a version of Dragon Architect that can be used with a college student population, we conducted a formal user study.

2.2.1 Design

Our user study had a single-group pretest-posttest design, which means that each participant is tested twice, once before and once after a treatment. In our case, this took the form of participants completing a "before" survey, playing the game for 30 minutes, then completing an identical "after" survey. During analysis, each participant is only compared to their past self.

2.2.2 Participants

We recruited 30 Carleton students from class years 2024 through 2020 who had never taken a computer science class and had little to no experience with programming to participate in our study. We decided to recruit students who had never taken a CS class before because, by the time we were ready to run our study (middle of winter term), they were the closest sample to our target population: students unfamiliar with computer science or students just starting their first computer science class.

2.2.3 Materials

The surveys that were completed by participants before and after playing were almost identical (see the appendix for the exact materials). They had two sections. The first contained questions that assessed

participants' attitudes towards computer science. This section contained 11 Likert-Scale questions that ask participants how much they agree with a given statement about computer science, computing, or computational thinking. Participants are given four possible responses that range from strongly disagree to strongly agree. These questions were drawn from two validated surveys developed by computer science education researchers (Hoegh & Moskal, 2009; Cetin & Ozden, 2015).

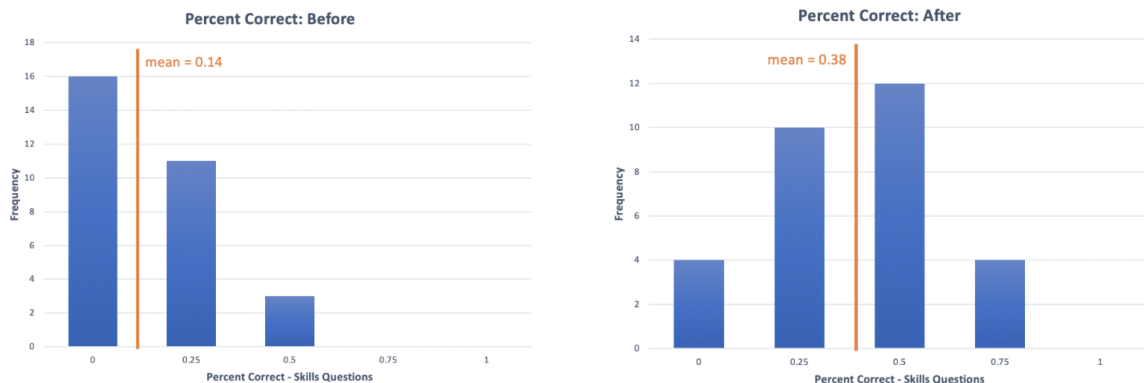
The second section of our survey contained four questions that assessed the participants' understanding of a few computational thinking skills, including loops, variables, and procedures. These questions presented a snippet of python code and asked the participant to choose the image that displays what the cubes and dragon would look like after the code is run. We designed these questions to test concepts that are taught in Dragon Architect and are a challenge for students in Carleton's Intro CS class, such as nested loops and calling functions. See the Appendix for the full materials from the user study

3 Analysis and Results

3.1 Computational Thinking Skills

To analyze any change in the participants' computational thinking skills, we calculated the percent of skills questions that each student answered correctly before and after playing the game. There were four skills questions on our survey, so each participant only had 5 potential scores: 0%, 25%, 50%, 75%, 100%. Before playing the game, participants answered an average of 14% questions correctly. As you can see in Figure 4, about half of the participants answered none of the questions correctly before playing the game. After playing the game, participants answered 38% of the questions correctly, on average. Because of the limited range of possible scores, we considered using a nonparametric statistical test (Wilcoxon signed-rank) to analyze the data, but because the skew of both distributions was less than 1 (before, skew = 0.80; after, skew = -0.11) and because their variances were similar (before, var = 0.03; after, var = 0.05), we determined the data adequately met the assumptions of parametric tests. A t-test of dependent means revealed a significant difference in percent of questions answered correctly from before to after playing, $t(29) = 4.69, p < .0001$. The effect size related to this change (Cohen's d) is 0.86. For reference, the cutoff for a small effect size is 0.2, and effect sizes larger than 0.8 are seen as large.

Figure 4: Distribution of Percent of Skills Questions Answered Correctly Before and After Playing Dragon Architect

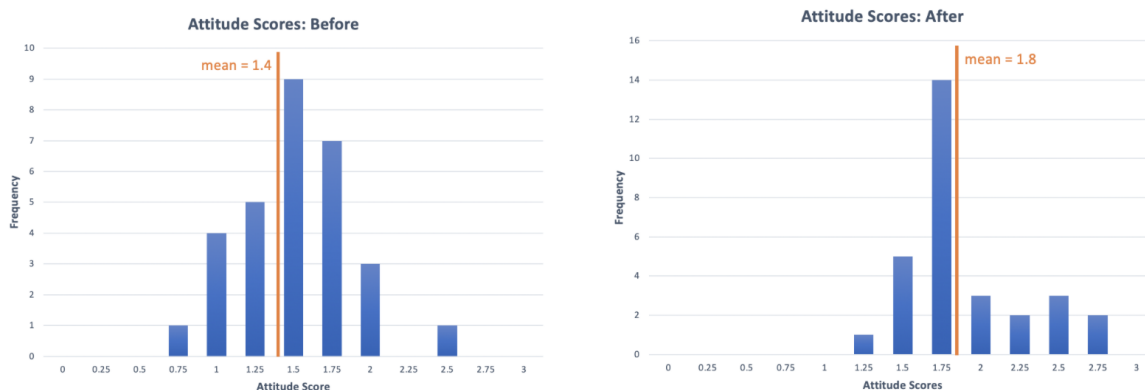


3.2 Attitudes Towards Computer Science

To analyze any change in the participants' attitudes towards computer science, we calculated an average attitude score for each student from before and after playing the game. We first flipped the scale for statements that were negative towards computer science. An example of a negative statement towards computer science is "I find programming frustrating", while an example of a positive statement towards computer science is "I can learn to understand computing concepts". Then, we converted each response into an integer value. For example, a response of "strongly disagree" to a positive statement about computer science was coded as 0 while a "strongly agree" response to positive statement about computer science was coded as 3. Finally, we averaged across the 11 statements to calculate an attitude score for each participant. Possible attitude scores ranged from 0 to 3, with 3 indicating positive attitudes towards computer science.

Before playing the game, participants had an average attitude score of 1.4, and you can see the distribution of scores in Figure 5. The distributions' skew (before, skew = 0.27; after, skew = -0.78) and variance (before, var = 0.134; after, var = 0.143) once again indicated that a parametric test would be a suitable choice to analyze the data. After playing the game, participants had an average attitude score of 1.8, and you can see the distribution in "after" scores in the histogram on the right. A t-test of dependent means revealed that this is a significant difference in attitude score from before to after playing, $t(29) = 6.98, p < .0001$. The effect size related to this change (Cohen's d) is 1.3, so we can conclude that this effect is also large.

Figure 5: Distribution of Attitude Scores Before and After Playing Dragon Architect



4 Future Work

Overall, these are exciting preliminary results that suggest that our updated version of the game effectively introduces Carleton students to computational thinking and improves Carleton students' attitudes towards computer science. Potential future directions of work include incorporating more computational skills into the game, such as conditionals or divide and conquer algorithms, exploring when and where it is most effective to deliver instructions and hints, or adding a social component to the game, such as allowing two players to work at the same time. Future user studies should include a control group, in which participants do not play the game, to allow for more definitive conclusions to be drawn about what caused the improvement in computational thinking skills and attitudes towards computer science. Finally, future work could explore the use of Dragon Architect with students taking their first computer science class.

5 References

- Bauer, A., Butler, E., & Popović, Z. (2017, August). Dragon architect: open design problems for guided learning in a creative computational thinking sandbox game. In Proceedings of the 12th International Conference on the Foundations of Digital Games (pp. 1-6).
- Cetin, I., & Ozden, M. Y. (2015). Development of computer programming attitude scale for university students. *Computer Applications in Engineering Education*, 23(5), 667-672.
- Hoegh, A., & Moskal, B. M. (2009, October). Examining science and engineering students' attitudes toward computer science. In 2009 39th IEEE Frontiers in Education Conference(pp. 1-6). IEEE.

6 Appendix

6.1 Surveys

Empty copies of the google forms used for the experiment are available here (pre) and here (post).

Table 1: Computational Thinking Skills Survey Questions

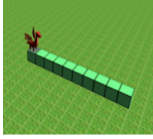
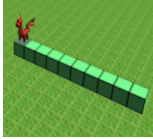
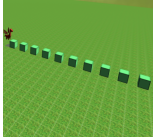
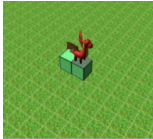

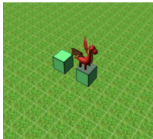
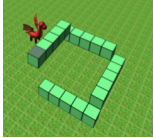
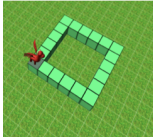
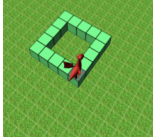
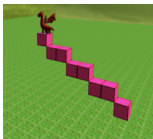
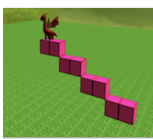
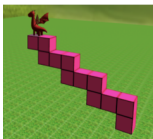
Code Snippet	A.	B.	C.	D.
<pre> 1. from DragonArchitect import forward, turnLeft, placeCube x = 10 for i in range(x): forward(1) placeCube(green) left() left() . </pre>				Unsure
<pre> 2. from DragonArchitect import forward, placeCube def myProc(): placeCube(green) forward(1) placeCube(green) . </pre>				Unsure
<pre> 3. from DragonArchitect import forward, turnLeft, placeCube x = 4 y = 5 for i in range(x): for i in range(y): forward(1) placeCube(green) left() . </pre>				Unsure
<pre> 4. from DragonArchitect import forward, up, placeCube def myProc(): forward(1) placeCube(red) for i in range(4): myProc() up(1) myProc() </pre>				Unsure

Table 2: Attitudinal Survey Questions

Question	Strongly Disagree	Disagree	Agree	Strongly Agree
1. I am comfortable with learning computing concepts.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2. I can learn to understand computing concepts.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3. I am confident that I can solve problems by using computer applications.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4. I would voluntarily take additional computer science course if I were given the opportunity.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5. The challenge of solving problems using computer science appeals to me.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6. I think computer science is interesting.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7. I find programming frustrating.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8. Programming-related activities make me nervous.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9. Programming improves your problem-solving skills.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
10. I feel nervous while writing programs.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
11. Program-writing is boring.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

After completing the post survey, participants were taken to a separate page and given the option to answer the following question:

Do you have any feedback or comments about Dragon Architect?

6.2 Participant Recruitment

6.2.1 Advertising Message

The following message was posted in campus announcements and on each of the Carleton Class Years Facebook pages:

Want to make money and help a comps group? The Dragon Architect Comps Group is recruiting participants for a user study. To be eligible, you must not have taken a Computer Science class before, and you must be 18 years or older. You will participate remotely. Participants will complete surveys and play an educational game. You will earn a \$5 Amazon gift card for 40 minutes of participation! To sign up, choose an appointment here ...

6.2.2 Confirmation Email

The following message was sent to each student who signed up to confirm their eligibility and help prepare them to participate:

Hello *Student!*

This email is confirming your appointment to participate in the Dragon Architect comps group's user study. Thank you so much for your willingness to help us! To be eligible for this study, you cannot have taken a computer science class (either at Carleton or elsewhere). If you have taken a computer science class (or are in one currently), please let us know, we will need to cancel your appointment.

Your appointment is scheduled for: *Day, Time*

At your scheduled time, join the appointment using this zoom link ...

During the appointment, you will fill out a survey, play a game for 30 minutes, then fill out a second survey. While you play the game, you will be monitored by a member of our comps group. Your monitor will be: *comps group member*.

If you need to cancel or reschedule your appointment, please reach out to your monitor by replying to this email.

The game you will be playing is only available on Carleton servers. If you are currently off-campus,

you will need to use a VPN to access the game. Setting up a VPN should only take about 5 minutes and is very easy. We would appreciate it if you could do so before your appointment time. However, if you are unable to, we will be able to help you at the beginning of your appointment. This may cause your appointment to run a little over 40 minutes. Instructions detailing how to set up a VPN can be found here...

After you complete the study, you will receive a \$5 Amazon gift card, sent to your Carleton email address. By participating, you are consenting to having your anonymized data aggregated and shared as part of a comps presentation. We will only keep identifying information to send you your gift card. Thanks again for your participation! Please reach out to your monitor if you have any questions or concerns. If you would rather talk to our comps advisor, you can reach Aaron Bauer at awb@carleton.edu

See you soon!

6.3 Experiment Protocol

1. Hello, introduce yourself
2. Confirm that they have not taken any computer science classes before
 - (a) If they have, thank them for their time, but tell them that they cannot participate
3. Check if they have Google Chrome, or at least Safari
 - (a) If they only have Firefox and aren't willing to download chrome, tell them they cannot participate, but thank them for their time.
4. Ask if they are on-campus
 - (a) If not, ask them if they have set up a VPN. If they have not, help them using Carleton's instructions
5. Have them fill out the pre-survey
6. Ask them to share their screen
7. Tell them you will be monitoring their game play, but will be muted
 - (a) Start a timer!
 - (b) If they don't have chrome, have them use safari and hope that it works
8. Have them use chrome to go the live link
9. After 30 minutes, have them stop sharing their screen and fill out the post-survey.
10. Thank them and tell them that they should be receiving their gift card to their Carleton email in the next few days
11. Add their email to this spreadsheet
 - (a) If they made it past the pre-survey but something happened (they had to leave, the game broke, etc.), we will still pay them.

6.4 Data Analysis

To see fully anonymized data, see here (pre) and here (post).

To see the details of the statistical analysis, see the source code here.