Gamification in Computer Science: Assessing and Evaluating Its Effectiveness in Learning and Engagement Across Varying Experience Levels

Senior Honors Capstone Project

in partial fulfillment of requirements

for the Norbert O. Schedlers Honors College

by

Justin M. Stauffer

University of Central Arkansas

Conway, Arkansas

Spring 2025

Capstone Project Committee

Mentor: Stephen R. Addison, PhD

Professor and COSE Dean

Reader: Jonathan Baarsch, PhD

Dean: Patricia Smith, EdD

Associate Professor and Dean

of the Honors College

Abstract

This capstone project was designed to contribute to the understanding of the effectiveness of gamification in learning by undergraduates with differing levels of prior knowledge in computer science. The study uses a competitive, quiz-style game that reviews the syntax of the widely used computer language, Python. In the quiz, there are 30 questions, worth 10 points each, 300 points available within the game. A post game questionnaire was administered to participants using Google Forms. The questionnaire was designed to assess both the game and its effectiveness. The questionnaire has 21 questions in total, and there is a section that requires a screenshot of their final score. The goal of the research was to test the effectiveness of gamification in attracting students to computer science, to increase the knowledge of participants, and to measure the effect of prior knowledge.

Table of Contents

Acknowledgements

I'd like to sincerely thank my mentor, Dr. Stephen Addison, my second reader, Dr. Jonathan Baarsch, and the Honors College faculty at the University of Central Arkansas for their guidance and support throughout this project.

Grammarly used for assistance in grammar.

Introduction

Gamification: Computer Science Experience Relative to Understanding

WSAW-TV, a news station in Wausau, Wisconsin, stated that in 2019, in the United States, there were 389,000 computer science job openings, but only 72,000 graduates to fill those openings ("Demand for Computer Science Graduates…", 2020). More and more computer scientists are needed as computers are increasingly embedded in all aspects of our lives. While the field is growing, student enrollment is not, which is a problem. Universities need to find a way to attract more students to the Computer Science field. Similarly, high schools need to find a way to incorporate experiential learning into their Computer Science curriculum in order to improve their students' understanding before college. If it is also found that experience plays a role in learning and understanding computer science, then gamification should be pushed to get people to garner more computer science experience before college.

While gamification has the potential to help teach computer science, Dicheva and colleagues found that limited research has been conducted on the effectiveness of gamification in teaching (Dicheva et al., 2015). They state that important factors must be considered when incorporating gamification. The game must be tailored to the students, so finding something that would interest most young people will have to be looked into and researched further (Dicheva et al., 2015).

My study is significant because it could potentially show a way to prosper and thrive in the world of computer science, by having prior experience. This topic could also show how computer science can be fun and interesting. It can consist of fun games and program-to-play games. Computer science is not always simply data and numbers. The world of computer science is vast and students can take many different paths. My topic could also be interesting because it

can show how having experience in computer science can give you an advantage over someone who does not, by giving them more knowledge and skills. The study has ways of gathering interest in the ever-growing field of computer science and ways to better succeed in the field of computer science. Computer science is a crucial field since technology controls a lot of aspects of our everyday lives, so knowing more about the field and understanding it is crucial.

My capstone project is a case study with the intent to understand the effectiveness of gamification of computer science learning on college students with varying degrees of experience with computer science skills. I have developed a way to get students more interested by giving them a Kahoot-style quiz game to play, where they will try to get the highest score possible, which will attempt to deepen their understanding and hopefully interest in computer science. I will gauge this by analyzing the results of the game to see if there are any positive changes in their understanding and skills. I will also be giving them a Google Form review to fill out afterward to see how their skills and thought process changed, if at all.

Literature Review

Computer Science is a growing field as the world keeps shifting more and more towards technology-based systems, but the computer science field is lacking students, which is a problem. I will go on to review the teaching methods for computer science, the languages best for it, and how gamification plays a big role in both learning and gathering attention to computer science.

Choosing a Programming Language

Knowing programming languages is crucial in the realm of computer science. Each person has their own preferred language and some come easier than others. Each language has pros and cons, and the sources I used go into greater depth about Python and C++ specifically as they are two of the most well-known and widely-used languages.

Python is a high-level language, so it is much easier and is in a more human-like language compared to C++. Therefore, it is much easier to learn and program, especially for beginners. C++ is a lower-level language, meaning it is more computer-like code and less human language for the sake of helping the computer process it faster, so it is harder to learn and program. Python has even been recommended as an introductory language for computer science students. Wainer and Xavier conducted a study in 2018 to see if Python would be a good introductory language versus C++. (Wainer & Xavier, 2018). The course fail rate for Python was slightly less than the fail rate for C++ (Wainer & Xavier, 2018). Overall, there were no negative effects of having Python as an introductory language. However, their study had its limitations since it was not a randomized controlled experiment, there is variability among students and teachers for each class, along with the fact that each final exam was different (Wainer & Xavier, 2018). However, Python was still seen as much easier by the students to learn than C++, so it may be best to use it

instead of C++ when trying to get students interested in Computer Science because it is interesting and not so challenging that it becomes discouraging.

However, if one wanted to teach C++, there are ways to best go about it. Raj and colleagues implement a top-down method, which starts with the more complex aspects of C++ and then works its way down to the simpler aspects. They found that this teaching method had positive results and the students liked it (Raj et al., 2018). However, there is a gap in this research; while this study does support the notion that the top-down approach can be useful, additional research would be needed to show if it was useful for students with no computer science background, as this study used students who already had experience.

While C++, is an option, Sultana and Reed suggest that while picking a language to teach is controversial, Python is an excellent choice because it continues to increase in popularity. Its accessibility and versatility make it a strong choice, especially for courses with non-majors (Sultana & Reed, 2017). Therefore, Python would be an excellent language to take into consideration when it comes to teaching computer science. This is important to note because using Python may be the best option for gamification as its accessibility is high, and it is very versatile, appealing to students of more majors than just computer science. Appealing to students of other majors is good because it gives more perspective into the game itself, especially if you are looking to compare the effectiveness of games with people of varying experience, as someone from another major will typically have less computer science experience than a computer science major.

Gamification

In order to get people, especially younger people, interested in the world of computer science, common interests must be found. The best way to get them to learn would be by putting

them into a situation where they are playing some type of game or just simply competing with each other. Ahmad and colleagues present a study on the impact of gamification on the learning outcomes of computer science majors. Gamification is the art of application of elements from games into other areas, such as learning for this instance. The study involved 124 students who were randomly assigned to either a gamified or non-gamified version of a course on data structures and algorithms. The gamified version included elements such as points, badges, and leaderboards, while the non-gamified version used traditional teaching methods (Ahmad et al., 2020). The study found that the gamified version led to better learning outcomes, including higher exam scores and greater engagement. The authors concluded that gamification can be an effective tool for promoting learning and engagement in computer science courses. However, they also note that the design of gamification elements must be carefully tailored to the course content and student population to ensure maximum effectiveness (Ahmad et al., 2020). This study was mostly successful and shows how gaming can be used as an application to help students better learn computer science.

Another study by Dicheva and colleagues shows that gamification learning can be efficient if done properly. You must have the proper software, technical support, and you must come up with an idea on how to help students learn through gaming without simply giving them a reward every time they accomplish something (Dicheva et al., 2015). Potentially, leveraging student competitiveness can be effective. Dicheva and colleagues note that empirical research on the effectiveness of gamification learning is scarce, but the research that is out there does support that Gamification learning can help improve the knowledge and understanding of computer science students (Dicheva et al., 2015).

A study by Weintrop, an Associate Professor at the University of Maryland, and Wilensky, a Professor of Learning Sciences, Computer Science, and Complex Systems at Northwestern University and also the Director of the Center for Connected Learning and Computer-Based Modeling there, argues that program-to-play games are best for helping students develop interest in and gather information about the computer science field (Weintrop & Wilensky, 2016). A program-to-play game is when the game itself requires the players to code as part of the game. They claim that these types of games are beneficial as they help build productive programming practices and encourage players to use programming concepts (Weintrop & Wilensky, 2016). The authors go on to use several popular games that arose because of this unique program-to-play method, but their actual research into students is limited and more is required.

Boulden and colleagues conducted a study to see if gamification helped students gain a better understanding of computer science and programming (Boulden et al., 2021). Using a narrative game-based approach, their results showed that all students gained a better understanding of computer science, regardless of experience. They also found that their game with a narrative design helped students, regardless of experience, learn computer science and gain programming skills (Boulden et al., 2021). The results from this study indicate that the type of game that is implemented, such as a narrative game, can affect how much students learn, and prior experience with computer science does not always give you the advantage.

When developing the game, in order to make it effective, there were six skills that really stood out to me within the book Shell wrote: animation, brainstorming, games, sound design, visual arts, and listening (Shell, 2020). These are the things I especially focused on when I designed my game.

Animation is crucial as it gives life to the game, captivating the audience, and gathering their attention. Brainstorming is very important as well as things rarely go to plan, and you often find better or just simply different ways to do things along the way. So, brainstorming and having plans, even documenting them, makes the process a lot smoother. Another obvious, yet crucial skill, is to understand games. Understanding how games work, including their mechanics and design principles, is essential when creating your own game. This knowledge provides the foundational skills and insights needed to effectively code and design a game, even if the game you are working on is not directly related to those principles(Shell, 2020). Understanding games and how they work aids in the game development process, it is as simple as that. Sound design is another one that is crucial, as sound is what truly convinces the mind it is in a place, as Shell stated, “Hearing is believing” (Shell, 2020). For instance, if you wanted to make a competitive style game, you could insert music that seemed to be similar to Kahoot, a popular game-based learning platform. It would give a calm yet competitive sense to the game, letting the player remain focused yet competitive with not too much pressure.

Visual arts are another crucial element, and games use a lot of graphic elements. It is crucial to create the feeling you want their game to have (Shell, 2020). Graphic design can help the central theme of the game and is arguably one of the most important aspects as the game has to be visually appealing to the audience for them to want to play or stay attracted throughout the game. According to Jesse Shell, a very successful software developer and author, making the game visually appealing is crucial and can determine if someone even considers playing the game. (Shell, 2020). Shell argues that one of the most important skills for a successful game designer is to listen. Listening to yourself is crucial from beginning to end. Come up with an idea, test it out, and keep adjusting it until it fits you. Pay attention to yourself and figure out why

you're making what you're making. That's how you'll find your personal theme. The game theme is already set.(Shell, 2020). Find your reasoning for the game, as that can help add your personal flair and differentiate it from other games. It can even help fine-tune it to your purpose even further. Listening to the game is crucial as well. As the creator, you must understand how the game works; it is crucial for the development and debugging of the game. Listening to the audience/player is crucial as well as they have an outside perspective and can review the game, offering advice and improvements, because maybe the game did not give the delivery or purpose you wanted it to (Shell, 2020). Listening to the player can help gear a game more precisely toward your intentions because they are the ones feeling the "effects" of the game and therefore can help provide guidance.

Computer Science Education

Computer science education can completely determine people's interest in the field and their understanding of it, so finding a way to best teach computer science is crucial. Ahmad and colleagues present a thesis that a visual programming language can cause students to improve their understanding of computer science. They compare students who study computer science on paper versus students who study computer science on computers and gain experience coding. They found that the students who studied on computers performed better than the ones who studied on paper. So, using visual programming languages would be a way to effectively teach computer science.

Another important strategy to improve computer science education is using a collaborative approach. El-Hamamsy, and colleagues conducted a study that discusses the challenges associated with implementing a new subject area and highlights the benefits of a collaborative approach (El-Hamamsy et al., 2022). They used a case study and found that the

collaborative approach led to greater teacher engagement and enthusiasm, as well as more effective implementation of the new curriculum. A collaborative approach is when several people work together to achieve a common goal. The authors emphasize the need for ongoing professional development and support to ensure the successful integration of computer science in primary schools. They conclude that a co-construction (collaborative) approach can lead to more sustainable and effective curricular reform, particularly in emerging subject areas such as computer science (El-Hamamsy et al., 2022).

Sultana and Reed have sought to define the course content for a university introductory computer science course based on regional needs (Sultana & Reed, 2017). They found that the best ways to teach computer science are to provide students with a focus on programming along with training in professional soft skills, such as problem-solving, critical thinking, and teamwork (Sultana & Reed, 2017). They go on to say that teamwork and problem-solving is the way to help students to best learn computer science, which is slightly different from what the others suggested, as the others only mentioned specific methods but do not bring up whether the work is done on a team or individually. This brings in another key element to consider.

Banilover and Craven conducted a study, during the 2017–18 school year, based on a national probability sample of 2,000 schools and approximately 10,000 computer science, mathematics, and science teachers in grades K–12 (Banilower & Craven, 2020). Their study delves deep into computer science education and how the attributes of the teachers affect their teaching and student engagement. They found that in diverse classrooms and classrooms with excelling students, reform-oriented learning was the best method of teaching and kept the students engaged and their scores in reform-oriented classes were quite high (Banilower & Craven, 2020). Reform-oriented teaching methods move away from traditional, teacher-centered

instruction and towards methods that actively involve students in their learning process. This article shows that reform-oriented teachers are quite effective and can help keep their classroom engaged. It would be an effective way to teach, by giving the students an immersive computer science learning experience and giving them hands-on experience, rather than simply lecturing.

Conclusion

Regardless of the language, Computer Science should be implemented at a younger age, according to a paper by Liana Heitin, a managing editor at *Education Week*, a K-12 magazine, stated that only 20 of the 50 states in the United States require students to graduate from high school with at least one credit in computer science (Heitin, 2016). That is not even half of the total states in the United States. Computer Science plays a crucial role in how we advance and grow, whether it be in healthcare or even education. As technology advances, so will the need for computer scientists. Therefore, we need more people in this field, so we can all grow together as a people and become even more advanced as technology will be instrumental in tackling the great challenges of our times: climate change, global instability, care for the elderly, increasing food production, etc. Computer science should help up on a higher podium in this world. After all, ultimately, computer science is essential in driving progress and innovation in the modern world. The need for computer science is great, as the demand for workers outweighs the supply when it comes to computer science jobs. Some effective methods for teaching students can be using gamification, program-to-play games, and using python as an introductory language as it is simple and easy to understand. I want to pursue more research in the gamification area of learning as the research is scarce, and I would like to see more ways of utilizing gamification to help teach students.

Methodology

The purpose of this case study is to understand the effectiveness of gamification of computer science learning on college students with varying degrees of experience with computer

science skills. At this stage in the research, computer science learning will be measured through the game and a questionnaire via Google Forms afterward. I have a particular interest in this topic because I grew up in a small school that offered no computer science classes, so I came into college with no previous computer science experience. This put me behind most of the students in my classes because many of them came from larger schools that offered computer science classes, so they had prior experience in the field. This topic really resonates with me, and I am passionate about finding results. Gamification is a growing topic and form of learning computer science but not much is known about its effects and the results of it, so I plan to add more to the topic with this project.

Participants

The participants in this study were college students from the University of Central Arkansas. They chose to play. To solicit participants, I sent out a mass email to all UCA students via UCA research, so they can decide whether or not they would like to play it. If they decided to play, they clicked the link in the email, and it took them to a Google Form, which they could then use to access the game. I recorded only the years of computer experience when it comes to the participants.

Study Design

My study is experimental research and my intervention used in the study is the implementation of a Kahoot-style quiz game to see if it gathers students' interest in computer science. Since my project wants to look at the effect of gamification on people of varying computer science experience, there is one independent variable, computer science experience. The game will be used to see if varying experiences affect learning. Python is an easy-to-understand programming language. I then looked at their responses to questions

presented in a Google Form after they played the game, to see what their views, interests, and outlook on computer science are after the game. I compared their views before they played the game using close-ended questions and the five-point Likert scale.

When designing the game, there were several factors that I had to consider, such as making it visually appealing, and incorporating sound and animation. Some features I included are a blue central theme, for blue light engagement, PNGs displayed for visual engagement as well, 30 multiple choice questions, ability to go back and forth between questions: previous and next buttons. I also made sure that once answered, the question cannot be answered again, it can be navigated back to however. There is confetti and a green celebration screen for correct answers and a red screen is displayed with the correct answer shown, if the selected answer was incorrect. I also added background music for an immersive experience, and score tracking to be displayed at the end of the game. I made sure to incorporate mouse tracking as well, for highlighting the answer that the mouse is hovering over for ease of use. Additionally, the game will not end until all 30 questions are answered. Some resources I used were Python and the Pygame Library along with the Google Form for game review analysis. Several design decisions were made in order to increase the game's effectiveness. Among the most important were creating the overall game design and flow, leveraging the Pygame Python library, developing a question data type, and architecting the Scoring System.

Game Design & Flow: The goal was to create an engaging and educational experience, not just a basic quiz. I structured the game as follows to try to achieve this: Immersive screen: A visually appealing screen to draw players in. Question Navigation: Users can move between questions using Prev/Next buttons. Answer Selection: Clicking on an option records the answer

and provides immediate feedback. Scoring System: Players garner points, and their final score is displayed at the end.

Pygame: I chose the Pygame library because it allows for an interactive experience with smooth UI elements along with making the coding process smoother as it had a lot of pre-existing functions that were useful. Some of these features include Confetti Animation: When a player gets a correct answer, colorful confetti rains down to simulate a reward effect. This helps reinforce correct answers positively. Dynamic Hover Effects: Buttons and answer choices change color when hovered over to provide visual feedback improving usability and making it easier for the player. Music & Sounds: Background music keeps the game lively and engaging. These pre-existing functions enhanced the appeal of the game making it more immersive and effective.

Questions DataType: The questions are stored in a list of dictionaries, making them easy to update and manage. Each question has: The question text. A list of possible answers. The correct answer. A flag to track if it has been answered.

Scoring System: Correct answer: +10 points Incorrect answer: No points awarded Maximum possible score: 300 points. I chose this scoring system because my mentor and I determined that 30 questions were sufficient for this project in order to analyze and interpret. I then decided to make the questions worth ten points a piece, 300 points total, because it would make calculations easier and scores could be reduced down to our of 100, by dividing their score over total score possible by 3.These design decisions all contributed to making the game engaging for student learners while meeting the time constraints for creating the game and performing the study.

Materials/Instruments

The material that used for this study consisted of questionnaires with closed-ended questions, with the five-point Likert scale, after the game via Google Forms. Powershell and Github assisted in distributing and delivering the game.

Data Collection Procedures

My data collection procedure was to use questionnaires after the game itself. The study was done over the course of four weeks. The participants were UCA college students who consented to the study. The game was a mandatory part of the procedure and took a maximum of ten minutes to play. Their questionnaire results were anonymous and confidential. I also gathered the results of their game for examination.

Data Analysis Procedures

After gathering all the data from the questionnaire, I analyzed the data by interpreting the results of the way they answered the questions using the five-point Likert scale. Each question had the answers: Strongly Disagree, Disagree, Neutral (or Neither Agree nor Disagree), Agree, Strongly Agree. Several questions were asked within it, gathering their comparisons of attitudes. I also looked at their game data and analyzed it as well. When analyzing the data, I looked for how they scored out of the 300 total points, each question was worth 10 points, totaling 30 questions, and cross-referenced that with their questionnaire responses, specifically with how many years of computer experience they had.

Ethical Considerations

There were ethical considerations in this study since I used human subjects. We received IRB approval for this thesis(see appendix for approval memo). To make sure that everyone was treated fairly, and with respect and dignity, we did several things. We gained their informed consent. We were transparent about the purpose and format of my research project. All

participants were treated respectfully.We made sure they were completely comfortable and understanding of the project. We informed them that their answers would be anonymous and confidential, and stored on a protected drive. They did not have to worry about offending anyone with their answers. They received my mentor and my own contact information for any follow-up questions they had after the study was done. Additionally, if there were any questions on the questionnaire that they did not feel comfortable answering, I was sure to let them know they could leave it blank if necessary.

Results/Conclusions

Participants

A. **Recruitment:** I recruited students, initially and in January into February, via the University of Central Arkansas's(UCA) research department, where they sent it out as a mass email to every UCA student. However, after waiting weeks, there were only three responses. I then resulted to collaborating with a Computer Science professor in March. They recruited students from their courses, both freshman and upper-level courses, so the majority of the participants are computer science students, however, their experience still varied due to offering extra credit for varying levels of courses. They even offered extra credit as an incentive, and after a few weeks, I gathered thirty total responses.
B. **Delivery:** For the delivery of the game, I created a Google Form with the required powershell script to deploy the game for them to copy and paste into powershell. I also had one participant who opted to take it in person on my device, as that was an option in the Google Form. Some participants also downloaded the GitHub repository, as I made it public for this reason, and ran the

program with Python via the command line or an Integrated Development Environment(IDE).

C. **Population of Participants:** After gathering all the data, 30 responses total, it appears that 36.7% of the participants had 4 or more years of computer experience, while 10% had 3 years, 23.3% had 2 years, and 30% had 1 or less. So it appears that the majority of the participants were on opposite sides of the experience spectrum, which helped even further in determining how crucial a role experience plays in learning.

D. **Special Sub-Groups**

It is important to note that there is one special sub-group within this data, as two people played the game two times, as that was an option specified in the Google Form.

Survey Results

Table Key

A. Date of Participation
B. Score(s)
C. The game was easy to understand and navigate.
D. I felt more engaged with the learning material because it was presented as a game.
E. The game helped me improve my understanding of computer science concepts.
F. I found the challenge level of the game to be appropriate for my skills.
G. The game kept my attention throughout the entire experience.
H. I would recommend this game to others who want to learn about computer science.
I. The game helped me feel more confident in my ability to solve coding problems.
J. The feedback I received in the game helped me understand my mistakes.
K. Playing the game made learning more enjoyable than traditional teaching methods.
L. The visual design of the game enhanced my learning experience.
M. I found the competition aspect of the game motivating.
N. The instructions provided were clear and easy to follow.
O. I feel that the game could be beneficial for students with no prior computer science experience.
P. The game allowed me to apply theoretical knowledge to practical problems.
Q. The pacing of the game was well-suited to my learning speed.
R. I would be interested in playing similar educational games in the future.

S. The game helped me retain the computer science concepts better than other methods.
T. The game helped me improve my problem-solving skills.
U. I found the game's tasks to be relevant to real-world challenges.
V. About how many years of computer experience do you have? (Round if needed)

1 = Strongly Disagree
2 = Disagree
3 = Neutral
4 = Agree
5 = Strongly Agree

Survey Results

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
6-Jan	100	5	3	3	1	3	2	2	2	2	2	3	4	3	2	1	3	3	2	2	4+
6-Jan	70	5	4	2	2	4	4	4	4	3	4	4	4	4	4	5	4	3	3	4	4+
6-Jan	290	5	4	4	2	5	5	1	2	5	4	4	5	4	1	4	5	4	2	1	< 1
4-Feb	270	5	4	2	3	4	2	2	4	3	2	5	5	2	1	4	3	2	2	1	4+
4-Feb	240	5	5	3	4	4	2	2	3	4	5	5	5	4	5	4	4	2	2	4	4+
5-Feb	180,240	5	5	5	5	5	5	4	5	5	3	5	5	5	3	5	5	4	3	5	< 1
4-Mar	90	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	< 1
10-Mar	280	5	4	2	3	5	4	2	4	4	2	5	5	3	4	4	4	2	2	4	4+
10-Mar	160	5	4	4	4	5	5	3	4	3	5	4	3	4	2	2	4	4	5	4	4+
10-Mar	300	5	5	5	3	5	5	3	4	4	4	4	4	5	4	4	4	3	2	3	2
11-Mar	270	5	3	3	4	2	2	2	2	2	3	4	4	1	2	3	4	2	2	2	2
11-Mar	210	4	4	3	4	4	3	2	2	4	4	2	3	2	3	4	4	3	3	2	2
11-Mar	130	4	2	4	3	3	3	4	2	4	2	4	2	1	4	4	4	2	3	4	4+
11-Mar	100	4	4	4	4	3	4	3	4	4	4	4	5	2	4	3	3	4	4	4	3
11-Mar	190	4	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	4	5	5	3
11-Mar	220	4	5	4	4	5	5	3	4	5	5	4	3	4	4	3	4	4	4	3	3

Survey Results																				
11-Mar	**190**	**5**	**3**	**3**	**4**	**4**	**2**	**1**		**2**	**3**	**5**	**4**	**4**	**3**	**2**	**4**	**4**	**4**	**3**	**4+**
12-Mar	**180**	**5**	**4**	**5**	**3**	**4**	**2**	**3**	**4**	**4**	**3**	**3**	**5**	**3**	**5**	**4**	**3**	**4**	**3**	**2**	**4+**
12-Mar	**90**	**5**	**5**	**5**	**5**	**4**	**4**	**3**	**4**	**4**	**5**	**5**	**5**	**4**	**3**	**4**	**5**	**3**	**3**	**4**	**4+**
13-Mar	**120**	**4**	**5**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**5**	**4**	**4**	**4**	**4**	**3**	**4**	**3**	**< 1**
14-Mar	**270**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**4**	**5**	**4**	**4**	**4**	**5**	**3**	**5**	**5**	**5**	**3**	**5**	**2**
14-Mar	**170**	**5**	**5**	**4**	**2**	**4**	**4**	**3**	**4**	**5**	**3**	**5**	**5**	**4**	**3**	**3**	**5**	**3**	**3**	**3**	**< 1**
14-Mar	**130**	**4**	**4**	**4**	**4**	**5**	**4**	**3**	**3**	**4**	**4**	**3**	**4**	**3**	**2**	**4**	**4**	**4**	**4**	**4**	**2**
15-Mar	**290**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**3**	**4**	**5**	**5**	**4**	**5**	**5**	**4**	**4**	**4**	**2**
17-Mar	**100**	**5**	**4**	**3**	**2**	**5**	**4**	**2**	**4**	**3**	**4**	**4**	**4**	**3**	**4**	**4**	**5**	**4**	**3**	**4**	**< 1**
17-Mar	**60**	**5**	**4**	**4**	**4**	**4**	**3**	**4**	**5**	**3**	**4**	**5**	**5**	**2**	**4**	**4**	**5**	**2**	**4**	**4**	**2**
18-Mar	**110, 230**	**5**	**3**	**2**	**3**	**4**	**3**	**4**	**5**	**2**	**1**	**4**	**5**	**1**	**2**	**2**	**3**		**2**	**2**	**< 1**
19-Mar	**190**	**5**	**4**	**4**	**4**	**5**	**4**	**3**	**3**	**4**	**4**	**5**	**5**	**2**	**3**	**4**	**4**	**3**	**3**	**3**	**4+**
19-Mar	**200**	**5**	**5**	**3**	**3**	**2**	**4**	**2**	**2**	**5**	**5**	**2**	**5**	**4**	**2**	**5**	**5**	**5**	**4**	**3**	**< 1**
21-Mar	**60**	**5**	**4**	**4**	**4**	**5**	**4**	**4**	**5**	**4**	**4**	**4**	**5**	**4**	**3**	**4**	**4**	**4**	**3**	**5**	**< 1**

Summary of Collected Data(The Big Five).

Figure 1.

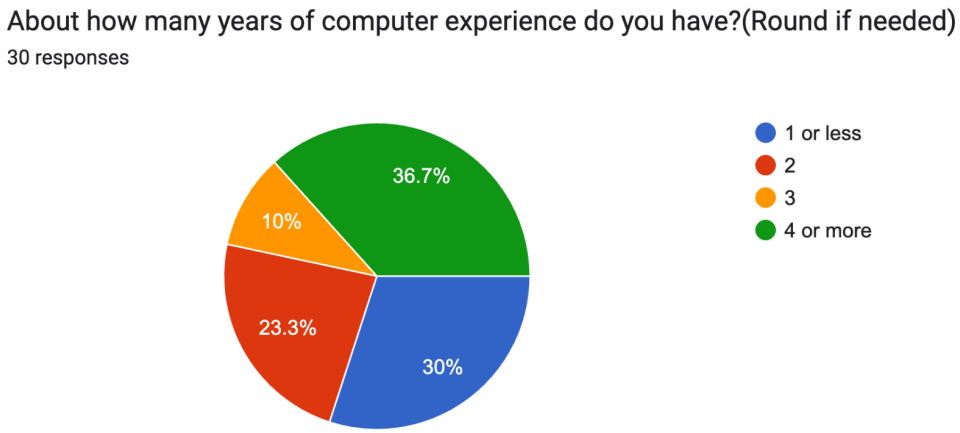


Figure 2.

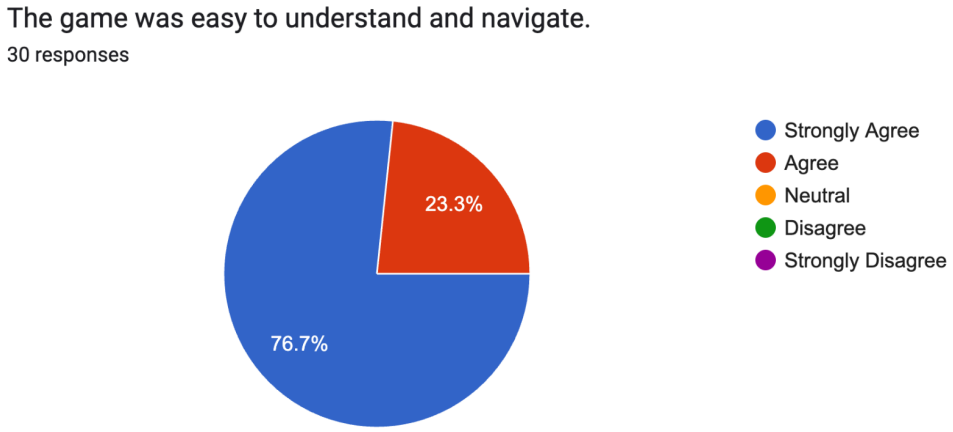


Figure 3.

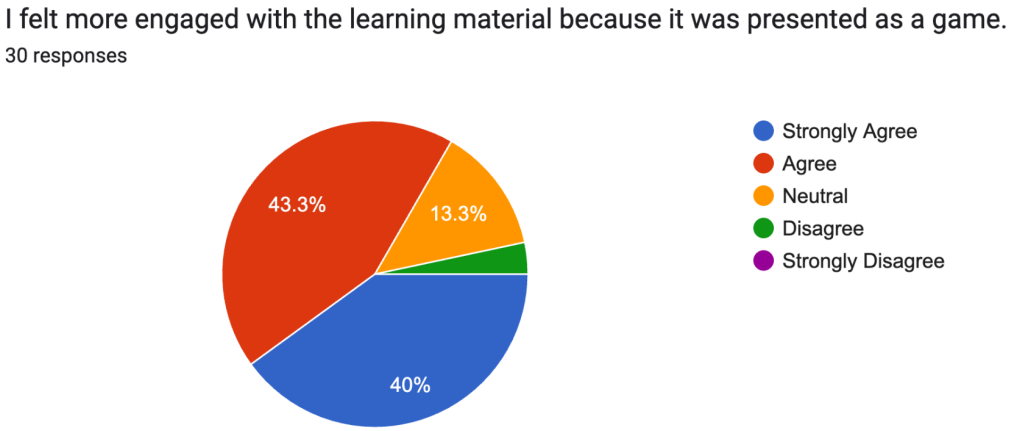


Figure 4.

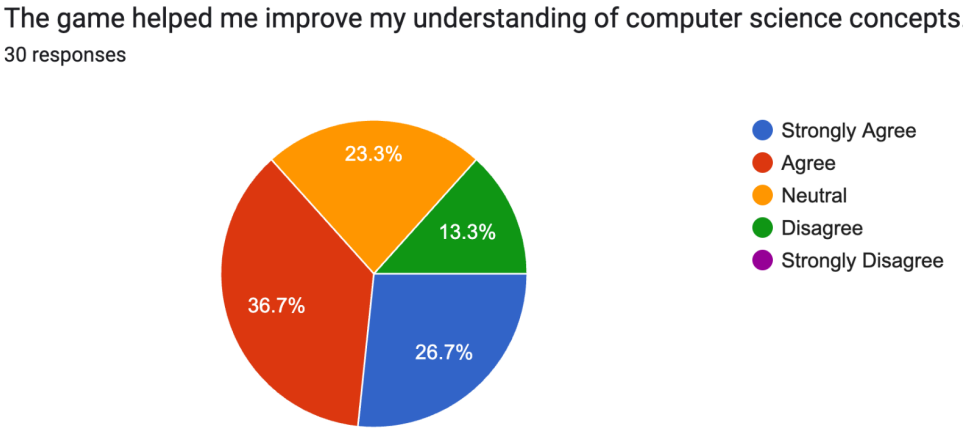
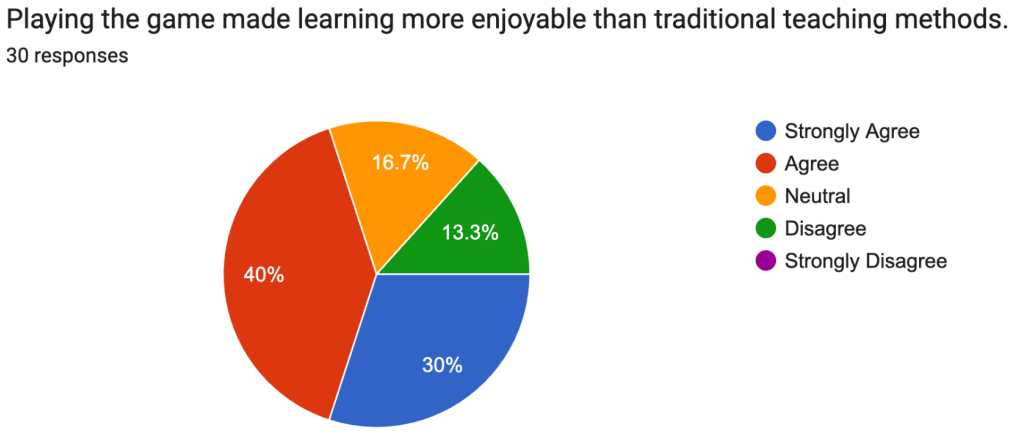


Figure 5.



Analysis

Gamification's Effectiveness

The majority of participants, 76.7%, strongly agreed that the game was easy to understand and navigate, with 100% of the participants either agreeing or strongly agreeing. Because the learning material was presented as a game, 83.3% of the participants agreed they felt more engaged, 13.3% neutral, and 3.3% disagreeing. 63.4% of the participants strongly agreed that the game helped improve their understanding of computer science concepts, 23.3% felt neutral, with 13.3% disagreeing. 70% of the participants agreed that the game made learning more enjoyable than traditional teaching methods, with 16.7% of them neutral and the other 13.3% disagreeing. From the data given above, the general consensus is that participants felt the game was easy to understand and navigate. Based on the data, it also kept them more engaged, improved understanding of the material, and made learning more enjoyable compared to traditional teaching methods. Two students even felt engaged enough to take it twice.

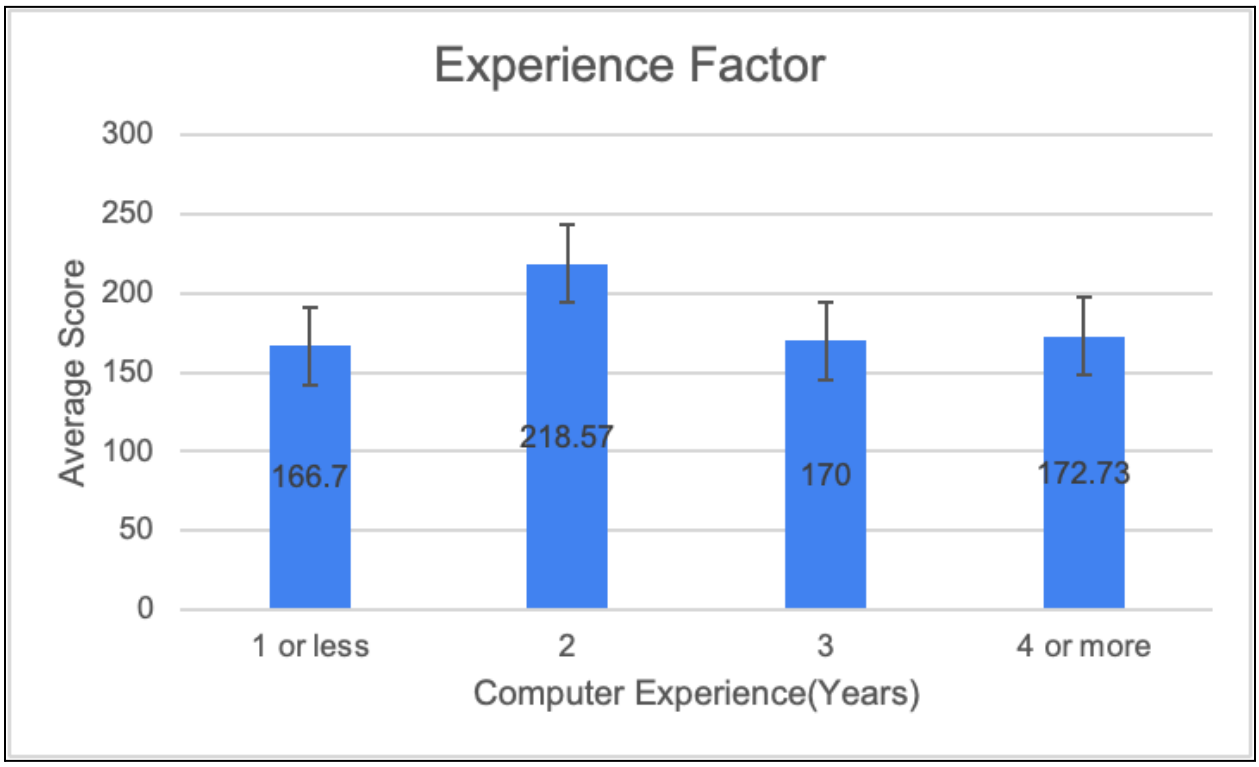
The Experience Factor

Programming experience was the only demographic information gathered. The participants with 1 year or less averaged a score of 166.67/300(55.56%). The participants with 2 years averaged a score of 218.57/300(72.86%). The participants with 3 years averaged a score of 170/300(56.67%). The participants with 4 years averaged a score of 172.73/300(57.58%). The calculated standard deviation is 24.50, and the range is 51.87 for the data. The chart for this data can be found at the end of this section.

In addition to experience in programming, two participants, both with 1 or less year experience, played the game twice, and their scores improved from the first to the second trial, with one increasing their score from 180 to 240, a 33.3% increase. The other participant's scores increased from 110 to 230, a 109.1% increase. For the purpose of this study, the latest scores of participants that have multiple attempts are taken into this calculation and not their first attempt, as it reflects their current knowledge from the game.

Additionally, the sole in-person participant noted that thanks to their experience in the Computer Science One course at UCA, which covers C++, some concepts were translated and they were able to answer some questions correctly.

Graph 1.



Overall Conclusion

The results show that gamification can effectively enhance both learning and engagement in computer science. Participants found the game easy to navigate, which helped reduce cognitive load and allowed them to focus on the content rather than the mechanics of the game. This ease of use along with the competitive nature of the game contributed to an increase in engagement. As many participants reported that they were more involved with the material compared to traditional teaching methods.

Additionally, the game was successful in improving participants' understanding of computer science concepts. Many participants indicated that the game deepened their comprehension of the material. This goes to show that gamification can be a powerful tool for

reinforcing learning. The improvement in scores from participants who played the game more than once further supports this. It shows that repeated engagement with the game helped solidify their understanding.

The results also highlighted the impact of prior programming experience. While participants with more experience tended to perform better, even those with little or no prior experience showed notable improvement. This suggests that gamification can be an inclusive learning tool that benefits students regardless of their background, which can potentially help beginners catch up with more experienced peers.

In conclusion, the study suggests that gamification can significantly enhance learning outcomes in computer science. Gamification can also make the subject more engaging and accessible to a wider range of students. The positive effects seen in this study show that gamified learning methods could serve as an effective teaching method compared to traditional teaching. Thereby, helping to bridge the gap for students at various levels of experience.

Discussion

Importance of Findings

The data gathered from this study highlight several key findings, all indicating that gamification can be an effective learning tool for computer science education. The general consensus is that participants consistently found the game was easy to understand and navigate, which is crucial for learning because a clear and easy to understand platform reduces cognitive load. This allows participants to focus on understanding the content instead of having to figure out how to interact with the game. Participants could focus more on learning the material and less on learning the platform. Participants also reported that their engagement was increased, which is another important finding. The competitive and interactive nature of the game was

shown to keep participants more involved with the content of the game, compared to traditional lecture-based learning. The data suggest that when learning is given as a game, students are more engaged, and motivated to perform and learn better. This level of engagement can be beneficial in computer science.

Another key finding is the impact of the game on participants' understanding of computer scientific concepts. Participants consistently reported that their understanding was improved, especially comparing their performance on the game before and after utilizing it. This supports the idea that gamification doesn’t just make learning more enjoyable, but it also helps with the retention of that knowledge. The game's ability to reinforce concepts interactively and more enjoyably indicates that gamification could play a crucial role in enhancing learning outcomes in computer science fields. Also, the participants who played more than once improved their scores greatly, which suggests that repeated exposure may further solidify knowledge.

Finally, the diversity of participant experience levels ranged from beginners to more experienced participants also offered valuable insights. Even participants with little to no prior programming experience showed improvement after engaging with the game. This demonstrates that gamification can be an inclusive and effective method for students at various stages of their learning journey. These findings support the potential of gamification as a tool for reinforcing knowledge among experienced students and also as a bridge for beginners to gain a better understanding of computer science concepts.

Connections to Previous Research

The results of this study are consistent with existing research on the benefits of gamification in education, particularly in technical fields like computer science. Several studies have shown that gamified learning environments can increase student engagement, improve

retention, and lead to better academic outcomes (Ahmad et al., 2020; Boulden et al., 2021). For example, Ahmad et al. (2020) found that gamification in a data structures and algorithms course resulted in higher exam scores and greater engagement. This aligns with the positive effects reported by participants in this study. This suggests that the incorporation of game-like elements such as points, real-time feedback(positive or negative) can be highly effective in motivating students to learn and perform well.

The results from this study also support the work of Dicheva et al. (2015), who found that gamification can enhance knowledge retention and student motivation. Especially when the game is designed to be both enjoyable and challenging. In the computer science field, students may often feel overwhelmed by the subject's complexity. The game used in this study served as a way to break down barriers and make learning more accessible and enjoyable. The study participants, like those in Dicheva et al.'s research, showed increased motivation and interest in the subject, which led to improved learning outcomes.

Finally, the inclusion of a game that was designed to be both easy to navigate and interactive reflects best practices in gamification, as highlighted by Shell (2020). This study embraced the holistic design principles outlined in the literature by focusing on key elements such as animation, sound design, and visual appeal. The game's effectiveness, along with the participants' positive feedback, indicates that well-designed gamification can play a crucial role in improving computer science education, offering a fresh and engaging way to learn technical content.

External Factors

The results suggest that gamification positively impacted participants' engagement and learning, but several external factors could have influenced the outcomes. For instance, some

students may have had prior exposure to game-based learning or might have been taught using similar techniques in other courses. These experiences could have affected their perception of the game and its impact on their learning.

Verifiability Factors

One limitation of this study that could impact the verifiability of the findings is the sample size. It is difficult to generalize these results to a larger population with a somewhat small group of participants(30). Additionally, self-selection bias could have played a role, as participants in the study may already have a higher level of interest in or comfort with technology and gamification. This could have skewed the results, because more engaged or motivated students might have performed better. Platform limitations should also be considered. While the game was designed to be accessible, there were certain limitations, such as students not knowing how to properly use PowerShell or GitHub. Tries to deploy the game via website were unsuccessful and due to time constraints, these two methods were the best-fit. This could have impacted some participants' experience or even drove them away from playing the game. These factors may have influenced the data, and future studies with a larger, more diverse sample and improved technical platforms would help increase the reliability and generalizability of the findings.

Notes for the Future & Implications of Gamification

The findings from this study open the door for further research into the role of gamification in computer science education. Future studies could explore different types of gamification techniques, such as incorporating more complex game mechanics, narrative-driven experiences, or interactive, program-to-play elements as suggested by Weintrop & Wilensky(2016). Future studies could also find better ways of deployment via an app or website.

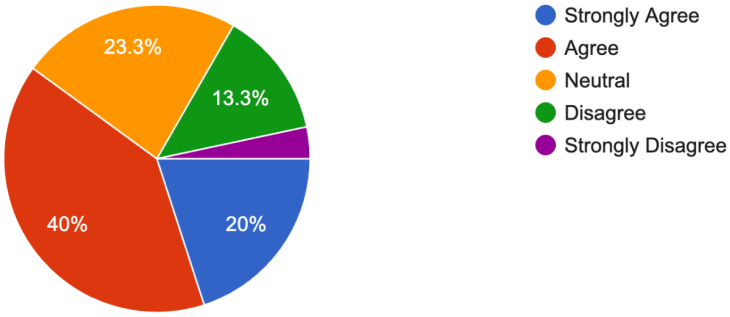
Research could focus on the deployment of gamified learning in various environments. For instance, in virtual classrooms or hybrid learning environments, to see if varying environments play a role in learning as well.

These results suggest that gamification could be an effective way to increase engagement and retention in computer science programs. Educators should consider incorporating gamified elements into their curricula to make learning more dynamic and accessible. Additionally, the findings indicate that students with varying levels of prior experience can benefit from gamified learning. This suggests that such methods could help bridge the gap between beginners and more experienced learners. Gamification has the potential to improve students' understanding of computer science concepts and also to inspire greater interest in the field, thereby contributing to the growth of the discipline as a whole.

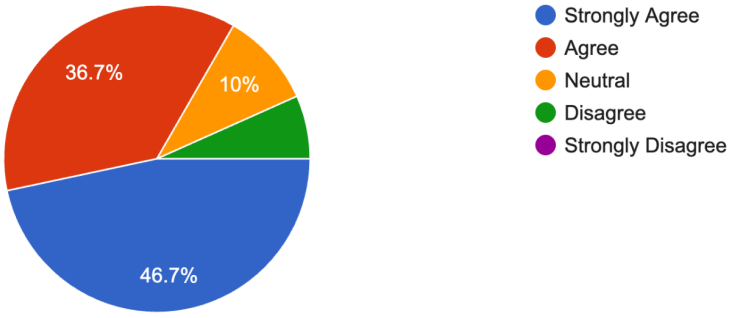
Appendix

Other Figures:

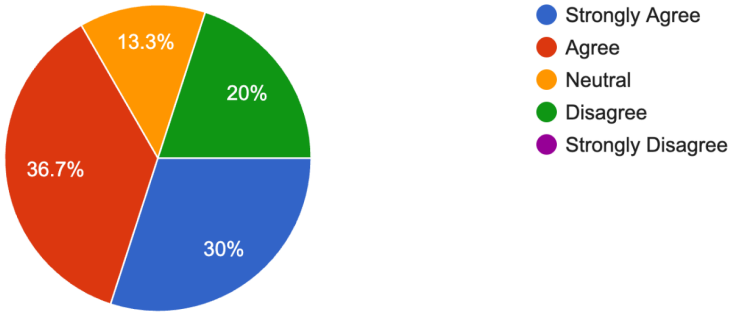
I found the challenge level of the game to be appropriate for my skills.
30 responses



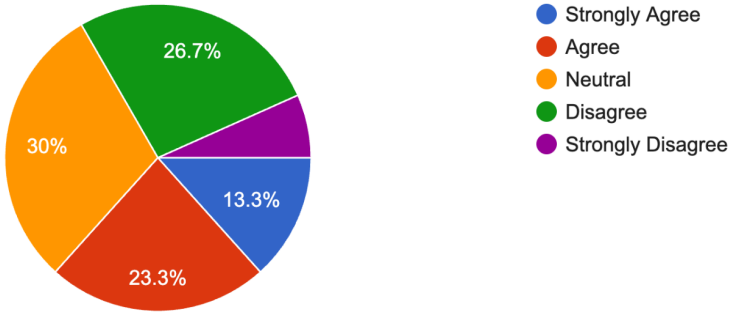
The game kept my attention throughout the entire experience.
30 responses



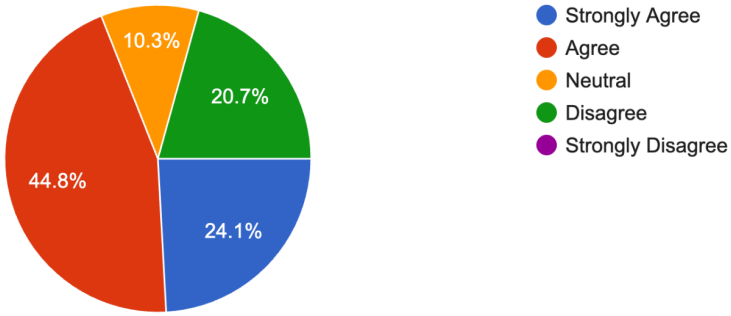
I would recommend this game to others who want to learn about computer science.
30 responses



The game helped me feel more confident in my ability to solve coding problems.
30 responses

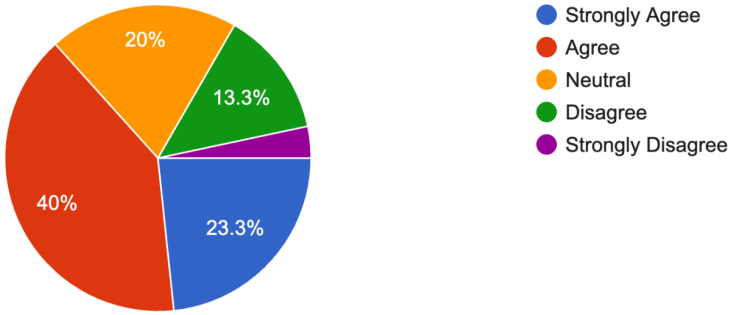


The feedback I received in the game helped me understand my mistakes.
29 responses



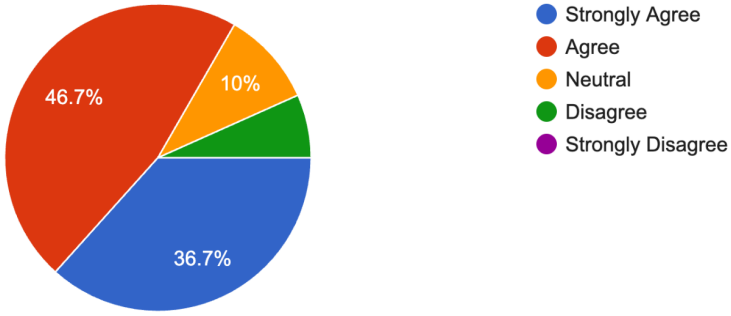
The visual design of the game enhanced my learning experience.

30 responses



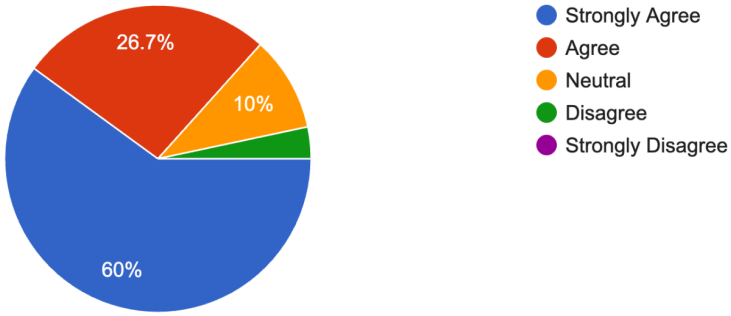
I found the competition aspect of the game motivating.

30 responses



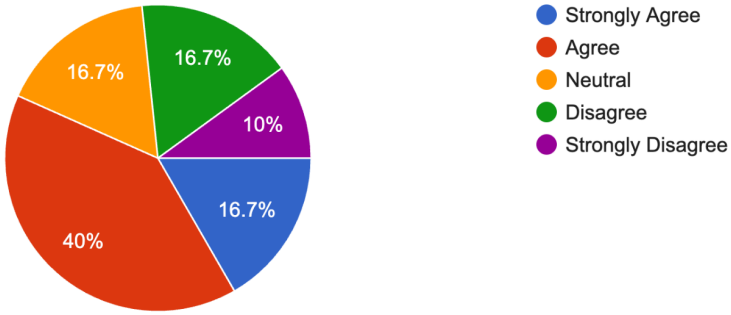
The instructions provided were clear and easy to follow.

30 responses



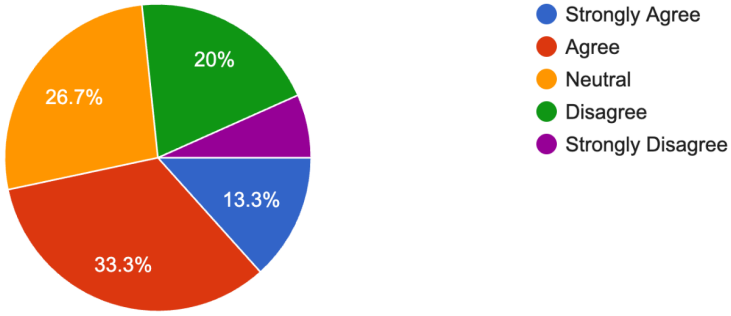
I feel that the game could be beneficial for students with no prior computer science experience.

30 responses



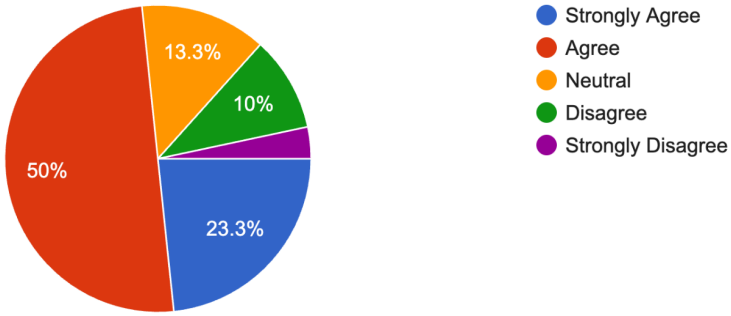
The game allowed me to apply theoretical knowledge to practical problems.

30 responses



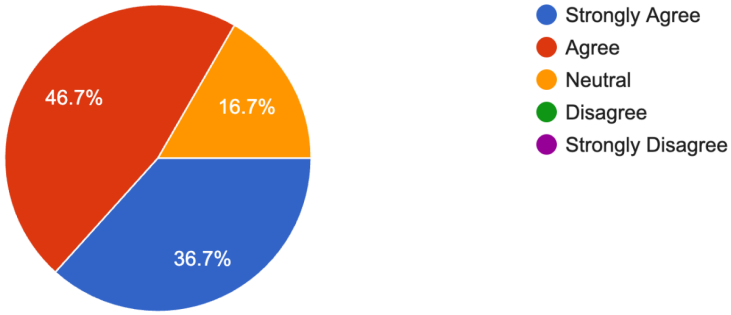
The pacing of the game was well-suited to my learning speed.

30 responses



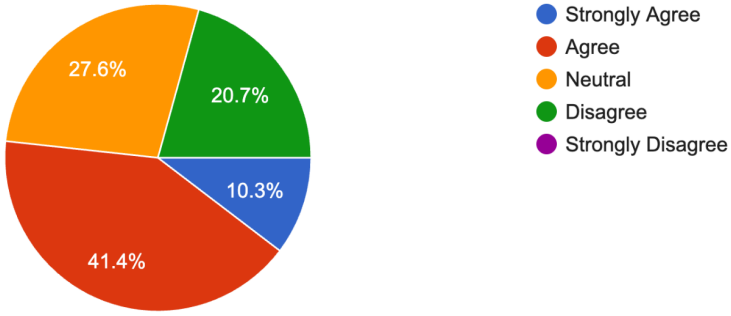
I would be interested in playing similar educational games in the future.

30 responses



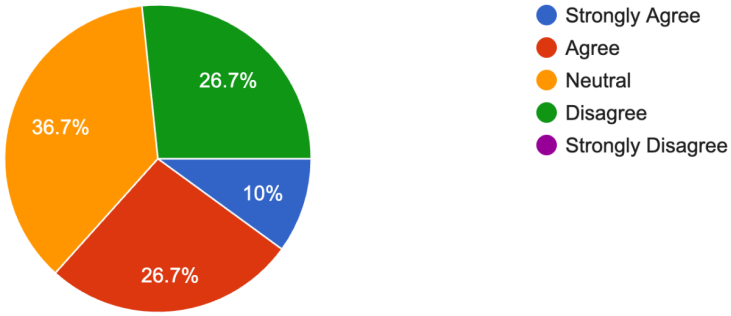
The game helped me retain the computer science concepts better than other methods.

29 responses



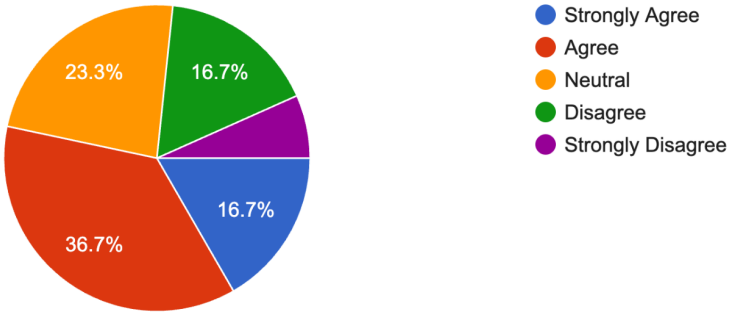
The game helped me improve my problem-solving skills.

30 responses



I found the game's tasks to be relevant to real-world challenges.

30 responses



IRB Approval Memo:

UNIVERSITY OF CENTRAL ARKANSAS™ | RESEARCH COMPLIANCE

IRB Exempt Approval Memorandum

To: Stephen Addison
Justin Stauffer

From: Research Compliance Office

Date: November 27, 2024

Subject: Exemption Review of **IRB #24-210**

Title: *Gamification-Computer Science Experience relative to Understanding*

A member of the UCA Institutional Review Board (IRB) has reviewed your application requesting exemption from further IRB review.

The research, as presented in your application, qualifies for exemption from further IRB review. Exemption for this research is approved from the date of this memo until a final report is submitted to the office of Research Compliance.

Note: Any changes to the original application with revisions must be submitted to the Research Compliance Officer before implementation as they could change the exempt status of your project.

If you have any questions, please contact us at 501-450-5789 or researchcompliance@uca.edu.

CC: Lasi McGhee

Github: https://github.com/JustinStauffer03/Honors-Capstone

Code:

```
# Import necessary modules from pygame for game development

import pygame

import sys

import random

from pygame.locals import *


# Initialize pygame modules and the mixer for sound effects and music

pygame.init()

pygame.mixer.init()


# Define basic color constants for easy reference and to improve readability

WHITE = (255, 255, 255)

BLACK = (0, 0, 0)

BLUE = (0, 0, 255)

CYAN = (0, 255, 255)

GREEN = (0, 255, 0)

LIGHT_BLUE = (173, 216, 230)

RED = (255, 0, 0)


# Set the dimensions of the game window

WIDTH, HEIGHT = 1150, 825
```

```
screen = pygame.display.set_mode((WIDTH, HEIGHT))


# Define fonts for different text elements in the game
FONT_NAME = "arial"
font = pygame.font.SysFont(FONT_NAME, 20)
largefont = pygame.font.SysFont(FONT_NAME, 100)
bangerfont = pygame.font.SysFont("papyrus", 28)
sfont = pygame.font.SysFont("star jedi", 28)
big_font = pygame.font.SysFont(FONT_NAME, 50)


# Defines a list of dictionaries containing the quiz questions, their options, correct answers,
and a flag for answered status
questions = [
    # Each dictionary in the list represents a quiz question with its details
    {'Question': 'Which is the correct way to output "hello"? ',
     'options': ['print(hello)', 'print("hello")', 'output(hello)', 'output("hello")'],
     'answer': 'print("hello")', 'answered': False},
    {'Question': 'Which is the correct way to create a for loop that loops through three
times? ',
     'options': ['for i in range(3):', 'for i in range(1,3):', 'loop(3):', 'for(i = 0, i <3, i++)'],
     'answer': 'for i in range(3):', 'answered': False},
    {'Question': 'Which of the following is used to comment a single line in Python? ',
```

'options': ['// This is a comment', '/* This is a comment */', '# This is a comment', '<!-- This is a comment -->'],

'answer': '# This is a comment', 'answered': False},

{'Question': 'What is the purpose of the continue statement in a loop in Python?',

'options': ['To exit the loop immediately and move to the next line of code after the loop.', 'To skip the current iteration of the loop and continue with the next iteration.', 'To repeat the current iteration of the loop indefinitely.', 'To pause the execution of the loop and wait for user input.'],

'answer': 'To skip the current iteration of the loop and continue with the next iteration.', 'answered': False},

{'Question': 'How would you create a variable, abc, that holds the value 123? ',

'options': ['abc = 123', 'int abc = 123;', 'abc == 123', 'var abc = 123'],

'answer': 'abc = 123', 'answered': False},

{'Question': 'Which of the following is important in Python syntax regarding scope? ',

'options': ['Curly braces { }', 'Semicolons ;', 'Parentheses ()', 'Indentation'],

'answer': 'Indentation', 'answered': False},

{'Question': 'How do you import the math module in Python? ',

'options': ['include math', '#import math', 'import math', 'using math'],

'answer': 'import math', 'answered': False},

{'Question': 'What is the purpose of the len() function in Python? ',

'options': ['Returns the size, in bytes, of object.', 'There is no len() function.', 'To return the size of the program.', 'Returns the length of an object as an int.'],

'answer': 'Returns the length of an object as an int.', 'answered': False},

{'Question': 'In Python, what keyword is used to declare a global variable? ',

'options': ['Public', 'There is no keyword.', 'Local', 'Global'],

'answer': 'Global', 'answered': False},

{'Question': 'What is the default return value of a function in Python? ',

'options': ['Zero', 'False', 'None', 'One'],

'answer': 'None', 'answered': False},

{'Question': 'Which of the following is the correct way to define a function in Python? ',

'options': ['function myFunc() {}', 'def myFunc():', 'function myFunc():', 'define myFunc():'],

'answer': 'def myFunc():', 'answered': False},

{'Question': 'What does the *args syntax do in a Python function? ',

'options': ['Allows the function to accept any number of positional arguments.', 'Allows the function to accept any number of keyword arguments.', 'Specifies the number of arguments the function will accept.', 'Creates a list of arguments the function will accept.'],

'answer': 'Allows the function to accept any number of positional arguments.', 'answered': False},

{'Question': 'What is the purpose of the __init__ method in a Python class? ',

'options': ['To initialize the class attributes.', 'To define the main method of the class.', 'To initialize the instance of the class.', 'To destroy the instance of the class.'],

'answer': 'To initialize the instance of the class.', 'answered': False},

{'Question': 'How do you concatenate two strings in Python? ',

'options': ['string1 + string2', 'concat(string1, string2)', 'string1 & string2', 'merge(string1, string2)'],

'answer': 'string1 + string2', 'answered': False},

{'Question': 'How would you handle an exception in Python? ',

'options': ['try-catch', 'try-except', 'catch-finally', 'except-finally'],

'answer': 'try-except', 'answered': False},

{'Question': 'What is a dictionary in Python? ',

'options': ['A mutable sequence of objects.', 'An immutable sequence of objects.', 'A collection of key-value pairs.', 'A collection of ordered items.'],

'answer': 'A collection of key-value pairs.', 'answered': False},

{'Question': 'How do you check the type of an object in Python? ',

'options': ['type(object)', 'object.type()', 'checktype(object)', 'object.type'],

'answer': 'type(object)', 'answered': False},

{'Question': 'Which of the following is used to create a new list in Python? ',

'options': ['list()', 'create_list()', 'new_list()', '[]'],

'answer': '[]', 'answered': False},

{'Question': 'What will be the output of print(2 ** 3) in Python? ',

'options': ['8', '6', '9', '16'],

'answer': '8', 'answered': False},

{'Question': 'Which statement is used to terminate a loop in Python? ',

'options': ['stop', 'break', 'exit', 'continue'],

'answer': 'break', 'answered': False},

{'Question': 'How do you check if a value is in a list in Python? ',

'options': ['value in list', 'list.contains(value)', 'list.has(value)', 'list.includes(value)'],

'answer': 'value in list', 'answered': False},

{'Question': 'What will be the result of [1, 2, 3] * 2 in Python? ',

'options': ['[1, 2, 3, 1, 2, 3]', '[1, 2, 3, 2, 3, 4]', '[2, 4, 6]', '[1, 2, 3] * 2'],

'answer': '[1, 2, 3, 1, 2, 3]', 'answered': False},

{'Question': 'Which keyword is used to define a class in Python? ',

'options': ['class', 'def', 'new', 'type'],

'answer': 'class', 'answered': False},

{'Question': 'How do you remove a key from a dictionary in Python? ',

'options': ['dict.remove(key)', 'dict.del(key)', 'del dict[key]', 'dict.pop(key)'],

'answer': 'del dict[key]', 'answered': False},

{'Question': 'What does the len() function do when applied to a list? ',

'options': ['Returns the number of items in the list.', 'Returns the size of the list in bytes.', 'Returns the first item of the list.', 'Returns the last item of the list.'],

'answer': 'Returns the number of items in the list.', 'answered': False},

{'Question': 'How do you create a tuple in Python? ',

'options': ['()', '[]', '{}', 'tuple()'],

'answer': '()', 'answered': False},

{'Question': 'What will be the output of len("Python")? ',

'options': ['6', '7', '5', '8'],

'answer': '6', 'answered': False},

{'Question': 'How do you define a lambda function in Python? ',

'options': ['lambda x: x + 1', 'def lambda(x): return x + 1', 'lambda x: {return x + 1}', 'function lambda(x) { return x + 1 }'],

'answer': 'lambda x: x + 1', 'answered': False},

{'Question': 'Which function is used to convert a string to an integer in Python? ',

'options': ['int()', 'str()', 'float()', 'convert()'],

'answer': 'int()', 'answered': False},

{'Question': 'How can you read a file line by line in Python? ',

```
    'options': ['with open(file) as f: for line in f: print(line)', 'file.read_lines()',
'file.readlines()', 'open(file).read_lines()'],
    'answer': 'with open(file) as f: for line in f: print(line)', 'answered': False}
]



# Initialize variables for tracking the current question and score
current_question = 0
score = 0
user_answers = [None] * len(questions)  # Initialize user answers to None

# Function to load an image, optionally resize it, and remove its white background
def load_image_remove_white_background(filename, size=None):
    image = pygame.image.load(filename).convert()  # Load and convert the image
    image.set_colorkey(WHITE)  # Set white as the transparent color
    if size:  # If a size is provided, resize the image
        image = pygame.transform.scale(image, size)
    return image

# Function to create a single confetti particle with random properties
def create_confetti():
    x = random.randint(0, WIDTH)
```

```
    y = random.randint(0, HEIGHT)
    color = random.choice([(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (0, 255, 255), (255, 0, 255)])
    size = random.randint(3, 6)
    fall_speed = random.uniform(0, 1)
    return {"x": x, "y": y, "color": color, "size": size, "fall_speed": fall_speed}

# Function to handle the animation and movement of confetti particles
def handle_confetti(confetti_particles, screen):
    for confetti in confetti_particles:
        confetti["y"] += confetti["fall_speed"]  # Move the confetti down
        pygame.draw.circle(screen, confetti["color"], (confetti["x"], confetti["y"]), confetti["size"])  # Draw the confetti
    return [confetti for confetti in confetti_particles if confetti["y"] <= HEIGHT]  # Remove off-screen confetti

# Function to animate confetti for a specified duration
def animate_confetti(duration, background_color, message):
    confetti_particles = [create_confetti() for _ in range(400)]  # Create a large number of confetti particles
    end_time = pygame.time.get_ticks() + duration
    while pygame.time.get_ticks() < end_time:
        for event in pygame.event.get():
```

```
            if event.type == pygame.QUIT:  # Allow quitting during the animation
                pygame.quit()
                sys.exit()
        screen.fill(background_color)  # Fill the background
        screen.blit(message, (460, 300))  # Display the message
        confetti_particles = handle_confetti(confetti_particles, screen)  # Update and draw
confetti
        pygame.display.flip()  # Update the display


# Function to draw text on the screen
def draw_text(text, font, color, x, y):
    text_surface = font.render(text, True, color)  # Render the text
    text_rect = text_surface.get_rect(center=(x, y))  # Get the text rectangle
    screen.blit(text_surface, text_rect)  # Draw the text on the screen


# Function to draw the choice options for a question
def draw_choices(question, mouse_pos):
    gap = 10  # Gap between options
    box_height = 50  # Height of each option box
    box_width = WIDTH // 2 + 200  # Width of the option box
    startx = WIDTH // 2 - box_width // 2  # Starting x-coordinate for the options
    total_gap = (len(question["options"]) - 1) * gap  # Total gap space between options
```

```
    total_box_height = len(question["options"]) * box_height  # Total height of all option boxes combined
    start_y = HEIGHT // 2 - (total_box_height + total_gap) // 2  # Starting y-coordinate for the options

    for i, choice in enumerate(question['options']):  # Iterate through each option
        rect_y = start_y + i * (box_height + gap)  # Y-coordinate for this option
        rect = pygame.Rect(startx, rect_y, box_width, box_height)  # Option rectangle

        if rect.collidepoint(mouse_pos):  # If mouse is over the option, highlight it
            color = CYAN  # Highlight color
        else:
            color = LIGHT_BLUE if i % 2 == 0 else BLUE  # Alternate colors for options

        pygame.draw.rect(screen, color, rect)  # Draw the option rectangle
        pygame.draw.rect(screen, WHITE, rect, 2)  # Draw a white border for the option
        draw_text(choice, font, BLACK, WIDTH // 2, rect_y + box_height // 2)  # Draw the option text

# Function to draw navigation buttons (Prev and Next)
def draw_navigation_buttons():
    prev_button_rect = pygame.Rect(50, HEIGHT - 100, 100, 50)  # Rectangle for the Prev button
```

```
    next_button_rect = pygame.Rect(WIDTH - 150, HEIGHT - 100, 100, 50)  # Rectangle for the Next button

    pygame.draw.rect(screen, BLUE, prev_button_rect)  # Draw the Prev button
    pygame.draw.rect(screen, BLUE, next_button_rect)  # Draw the Next button

    draw_text("Prev", font, WHITE, 50 + 50, HEIGHT - 75)  # Text for Prev button
    draw_text("Next", font, WHITE, WIDTH - 150 + 50, HEIGHT - 75)  # Text for Next button

    return prev_button_rect, next_button_rect  # Return the rectangles for button handling

# Function to handle clicks on navigation buttons
def handle_navigation_buttons(prev_button_rect, next_button_rect, mouse_pos):
    global current_question  # To modify the global current_question variable

    if prev_button_rect.collidepoint(mouse_pos) and current_question > 0:  # If Prev is clicked and not on the first question
        current_question -= 1  # Move to the previous question

    if next_button_rect.collidepoint(mouse_pos) and current_question < len(questions) - 1:  # If Next is clicked and not on the last question
        current_question += 1  # Move to the next question
```

```
# Function to check if all questions have been answered
def all_questions_answered():
    return all(answer is not None for answer in user_answers)  # Check if any answer is
None

# Function to calculate the final score based on correct answers
def calculate_score():
    score = 0
    for i, question in enumerate(questions):
        if user_answers[i] == question['answer']:  # If the user's answer matches the correct
answer
            score += 10  # Increment score
    return score

# Main game loop
def game_loop():
    global current_question, score
    confetti_particles = []
    backgroundimage = load_image_remove_white_background('backgroundmain.png',
(1150, 825))
    robot_image = load_image_remove_white_background('robot1.png', (250, 250))
    gamestart = load_image_remove_white_background('gamestart.png', (250, 250))
```

```
    python = load_image_remove_white_background('python.png', (250, 250))


    while not all_questions_answered():  # Main game loop, runs until all questions are
answered
        screen.blit(backgroundimage, (0, 0))  # Display the background image
        screen.blit(robot_image, (900, 50))  # Display the robot image
        screen.blit(gamestart, (10, -15))  # Display the game start image
        screen.blit(python, (500, 550))  # Display the Python logo


        robot = bangerfont.render("Think you got it??", True, BLACK)  # Render a text
        screen.blit(robot, (900, 110))  # Display the rendered text
        question = questions[current_question]  # Get the current question
        draw_text(question['Question'], font, BLACK, WIDTH // 2, HEIGHT // 4)  # Display
the question text
        mouse_pos = pygame.mouse.get_pos()  # Get the current mouse position
        draw_choices(question, mouse_pos)  # Draw the choice options for the current
question
        pygame.display.flip()  # Update the display


        waiting_for_input = True
        while waiting_for_input:  # Wait for the user to select an option or navigate
            prev_button_rect, next_button_rect = draw_navigation_buttons()  # Draw
navigation buttons and get their rectangles
```

```
        for event in pygame.event.get():  # Event handling loop
            if event.type == pygame.QUIT:  # If the user closes the window
                pygame.quit()
                sys.exit()
            mouse_pos = pygame.mouse.get_pos()  # Update mouse position
            draw_choices(question, mouse_pos)  # Redraw choices to update hover effects
            pygame.display.flip()  # Update the display
            if event.type == pygame.MOUSEBUTTONDOWN:  # If the user clicks the mouse
                mouse_x, mouse_y = pygame.mouse.get_pos()  # Get the click position
                # Check if the user clicked on navigation buttons and handle accordingly
                if prev_button_rect.collidepoint(mouse_x, mouse_y) or
next_button_rect.collidepoint(mouse_x, mouse_y):
                    handle_navigation_buttons(prev_button_rect, next_button_rect, (mouse_x,
mouse_y))
                    waiting_for_input = False
                    break

                # Skip if the question is already answered
                if questions[current_question]['answered']:
                    continue

                # Check which option was clicked
                for i, choice in enumerate(question["options"]):
```

```
                gap = 10
                box_height = 50
                total_gap = (len(question["options"]) - 1) * gap
                total_box_height = len(question["options"]) * box_height
                start_y = HEIGHT // 2 - (total_box_height + total_gap) // 2
                rect_y = start_y + i * (box_height + gap)
                rect = pygame.Rect(WIDTH // 4, rect_y, WIDTH // 2, box_height)

                if rect.collidepoint(mouse_x, mouse_y):  # If the user clicked an option
                    pygame.draw.rect(screen, CYAN, rect, 5)  # Highlight the selected option

                    if all_questions_answered():
                        final_score = calculate_score()  # Calculate final score if all questions
answered
                        return

                    user_answers[current_question] = choice  # Update the user's answer for
the current question
                    questions[current_question]['answered'] = True  # Mark the current
question as answered

                    waiting_for_input = False
                    if choice == question["answer"]:  # If the user selected the correct answer
```

```
                    textright = big_font.render("Correct!", True, BLACK)  # Render
"Correct!" message
                    animate_confetti(3000, GREEN, textright)  # Animate confetti for
correct answer

                    screen.fill(GREEN)  # Fill screen with green to indicate correct answer
                    pygame.display.flip()  # Update the display
                else:
                    textwrong = big_font.render("Incorrect", True, BLACK)  # Render
"Incorrect" message
                    correctanswer = font.render(f"The correct answer was:
{question['answer']} ", True, BLACK)  # Show the correct answer
                    screen.fill(RED)  # Fill screen with red to indicate incorrect answer
                    screen.blit(textwrong, (460, 300))  # Display "Incorrect" message
                    screen.blit(correctanswer, (250, 370))  # Display the correct answer
                    pygame.display.flip()  # Update the display

                    pygame.time.wait(3000)  # Wait for a few seconds before continuing

                if current_question < len(questions) - 1:  # If there are more questions
                    current_question += 1  # Move to the next question

                waiting_for_input = False  # End waiting for input
```

```
    # End of the game - calculate and display final score
    final_score = calculate_score()
    backgroundimage = load_image_remove_white_background('backgroundmain.png',
(1150, 825))
    scoreimage = pygame.image.load('finalscore.png')  # Load final score image
    scoreimage = pygame.transform.scale(scoreimage, (250, 250))  # Scale the final score
image
    gameover = pygame.image.load('gameover.png')  # Load game over image
    gameover = pygame.transform.scale(gameover, (200, 200))  # Scale the game over image
    screen.blit(backgroundimage, (0, 0))  # Display background image
    screen.blit(scoreimage, (430, 300))  # Display final score image
    screen.blit(gameover, (460, 80))  # Display game over image
    pygame.display.flip()  # Update the display
    score_text = largefont.render(f" {final_score}", True, BLACK)  # Render the final score
text
    screen.blit(score_text, (485, 450))  # Display the final score
    pygame.display.flip()  # Update the display

    while True:  # Keep the final score displayed until the user closes the game
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
```

```
            sys.exit()


# Load and play background music

pygame.mixer.music.load('Solve The Puzzle.ogg')

pygame.mixer.music.play(-1)  # Loop the music

pygame.mixer.music.set_volume(0.75)


# Start the game loop

game_loop()
```

References

Ahmad, A., Zeshan, F., Khan, M. S., Marriam, R., Ali, A., & Samreen, A. (2020). The impact of gamification on learning outcomes of computer science majors. *ACM Transactions on Computing Education*, *20*(2), 1–25. https://doi.org/10.1145/3383456

Banilower, E. R., & Craven, L. J. (2020). Factors Associated with High-Quality Computer Science Instruction: Data from a Nationally Representative Sample of High School Teachers. *Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3328778.3366831

Boulden, D. C., Rachmatullah, A., Hinckle, M., Bounajim, D., Mott, B., Boyer, K. E., Lester, J., & Wiebe, E. (2021). Supporting Students' Computer Science Learning with a Game-based Learning Environment that Integrates a Use-Modify-Create Scaffolding Framework. *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education v. 1*. https://doi.org/10.1145/3430665.3456349

El-Hamamsy, L., Bruno, B., Kovacs, H., Chevalier, M., Dehler Zufferey, J., & Mondada, F. (2022). A case for co-construction with teachers in curricular reform: Introducing computer science in primary school. *Australasian Computing Education Conference*. https://doi.org/10.1145/3511861.3511883

Dicheva, D., Dichev, C., Agre, G., & Angelova, G. (2015). Gamification in Education: A Systematic Mapping Study. *Journal of Educational Technology & Society*, *18*(3), 75–88. https://www.jstor.org/stable/10.2307/jeductechsoci.18.3.75

Heitin, L. (2016). States' Push for Computer Science Grows: Computer Science in High School Graduation Requirements. *Education Week, 36(5)*

https://ucark.idm.oclc.org/login?url=https://www.proquest.com/trade-journals/states-push-computer-science-grows/takeoff/1825230452/se-2

López, J., García, M., & Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: educational use of mBot. *Educational Technology Research and Development*, *67(6)*, 1405–1425. https://doi.org/10.1007/s11423-019-09648-5

Microsoft. (n.d.). Introduction to PowerShell. *Microsoft Learn.* Retrieved from https://learn.microsoft.com/en-us/powershell/scripting/learn/ps101/00-introduction?view=powershell-7.5

Pygame(n.d). Pygame Front Page - pygame v2.6.0 documentation. *Pygame front page*. https://www.pygame.org/docs/

Raj, A. G. S., Naik, V., Patel, J. M., & Halverson, R. (2018). How to teach "modern C++" to someone who already knows programming?. *Proceedings of the 20th Australasian Computing Education Conference*. https://doi.org/10.1145/3160489.3160503

Schell, J. (2020). *The Art of Game Design: A book of lenses* (3rd ed., Ser. Tenth Anniversary). CRC Press, Taylor & Francis Group.

Sultana, S. G., & Reed, P. A. (2017). Curriculum for an Introductory Computer Science Course: Identifying Recommendations from Academia and Industry. *The Journal of Technology Studies*, *43*(2), 80–92. https://www.jstor.org/stable/10.2307/90023144

Wainer, J., & Xavier, E. C. (2018). A Controlled Experiment on Python vs C for an Introductory Programming Course. *ACM Transactions on Computing Education*, *18(3)*, 1–16. https://doi.org/10.1145/3152894

Weintrop, D., & Wilensky, U. (2016). Playing by Programming: Making Gameplay a Programming Activity. *Educational Technology*, *56*(3), 36–41. https://www.jstor.org/stable/44430491

WSAW.*(*2020, December 10) Demand for computer science graduates continues to outpace supply. https://www.wsaw.com. https://www.wsaw.com/2020/12/10/demand-for-computer-science-graduates-continues-to-outpace-supply/