Designing a Computer Game to Teach Computer Science Concepts

Rhys Hanrahan

Abstract-This project is attempting to solve the problem of people struggling to learn computer science coding, in particular threshold concepts such as conditional statements, methods, and recursion. Which, without understanding, prevents learners from continuing progression. This project aims to solve this problem as a fun, engaging and interactive video game that uses constructivism techniques. The game contains blocks of code, with simple phrases, numbers, or words on them, that can be arranged to create functioning code structures. To test and evaluate the game, some level 400, 300, and 100 COMP/SWEN students have played the game, providing thoughts and feedback throughout development. Regular user testing and feedback has been very insightful for this project providing many new ideas, thoughts for improvement and finding problems and bugs through different perspectives. The current version of the project has 22 levels ranging in difficulty which contains simple code structures, infinite/looping code structures, logic comparisons, multiple simultaneous code structures, multiple controllable players entities, codable controlled entities, and method blocks which can be called, and cause recursion within themselves. Current constructivism ideas include simple mazes, Snake, and Pong levels where the user must create code for aspects (e.g. the ball movement in pong), with their own pre-existing knowledge, of the widely known game/idea.

1.1 INTRODUCTION

It is important to help promote the education of computer scientists as they are required and highly valued in various industries due to the world's increasing requirements in technology. One of the primary challenges in computer science education is overcoming the early threshold concepts faced in learning programming which causes the highest dropout rates in any field *[1][2][3],* with some of the reasons being “Lack of required skills”, “Not knowing how to ask for help” *[1]* and the course being believed to be too hard *[3]*.

This project is attempting to help solve the problem of students struggling to learn computer programming, in particular threshold concepts, by creating an additional learning resource targeting these concepts. “A threshold concept refers to core concepts in a subject where understanding these concepts is key to transforming the way students understand a whole subject, allowing them to move on in their learning” [4]. This project focuses on concepts such as conditional statements, methods, and recursion[5][6],and aims to be an easily accessible, simple, fun and interactive video game that beginners learning coding, in particular students that may be struggling to understand some of the first threshold concepts; can access and play as an additional learning resource to help understand the concepts so they can continue their learning.

1.2 EXISTING SOLUTIONS

A constructivism learning theory is one of many techniques to improve learning. It is where you “emphasize the active role of learners in building their own understanding” through reflecting on their own experiences and incorporating their knowledge to promote deeper learning and understanding [7]. Using a constructivism approach to designing levels such as simple mazes, Snake, and Pong levels, the user must create code for aspects (e.g. the ball movement in pong), with their own pre-existing knowledge, of the widely known game/idea without having to be told how it should function.

An existing solution for the development of interaction design is Jakob Nielsen’s usability heuristics, which is the most used usability heuristics for user interface design. These heuristics help set the standard expectation for the non-functional requirements of this project.

Three existing application solutions I investigated were the online platform “Scratch” *[8]*, to get an understanding of possible techniques to show, teach and use code in the project. I chose to analyze scratch as it was one of the tools that I used to start learning computer code and it showed me how simple and complex programs can get using very simple visual structures. The other two existing solutions are both problem solving, puzzle video games called “Baba is You” *[9]* and “Patricks Parabox” *[10]* which I analyzed to get an understanding of fun, interactive, and problem-solving traits.

Scratch

“Scratch is a high-level block-based visual programming language and website aimed primarily at children as an educational tool, which is used to create projects using a block-like interface.”

Pros	Cons
- Massive community with lots of support and resources.	- Block-like interface does not provide any bridge to real programming.

- Easy and fun introduction to coding. - Allows for deep understanding of background processes in video games. - Can be integrated into different subjects.	- It does not include text-based coding. - It is an online community targeted at kids and content moderation only consists of user reporting.

From the analysis of Scratch, I wanted to maintain the positives of being an easy and fun introduction to code, allowing for a deeper understanding of background processes in games, and easy integration for multiple different ideas. To implement this for my game, I keep the simplicity of each block of code. Through keeping each block of code simple it would allow for very easy levels to be created to ease users into different aspects of the game and allows for the game to have a very wide range of possibilities of what the code can create. Looking at the negatives of Scratch, I wanted to ensure my game can provide some transferable knowledge into real programming, so I keep the basic code blocks in the game the same as real code.

Baba is You
"Baba Is You is a puzzle game where the rules you have to follow are present as blocks you can interact with. By manipulating them, you can change how the game works, repurpose things you find in the levels and cause surprising interactions."

Pros	Cons
- Simple but fun art style. Responsive and smooth movement animations. - Unique idea for a puzzle game. - Challenging puzzles, with multiple different interactions. - Music fits gameplay and is not annoying for continuous gameplay.	- "Brutally hard puzzle game" level difficulty ramps too quickly which requires users to have some knowledge of programming to not become utterly confused. - Some levels feel "unintuitive and slow". - Concepts not being taught, with some solutions feeling like 'one-time exploits. - No hint system.

The main positive aspect from Baba is You I wanted to transfer into my game, is the aspect of allowing multiple different interactions and possibilities to solve puzzles. To do this I needed to create and provide the user with many different code blocks, so they can create their own solution and do not always just have to arrange one correct solution. There were a few cons from Baba is You that I wanted to avoid, such as levels getting too hard, progression of levels being too slow, and some solutions feeling like one-time exploits. So, to ensure the difficulty progression of the game's levels being balanced, getting user testing and feedback was crucial throughout development.

Patricks Parabox
"A mind-bending recursive puzzle game about boxes within boxes within boxes within boxes. Learn to use infinity to your advantage as you explore a deep and elegant system."

Pros	Cons
- Interesting and tricky problem-solving concepts - Some concepts and level designs require taking a new view of thinking. - Responsive and smooth movement and animations. - Simple starting levels, easy to understand the objective and start playing.	- A lot of the levels start to get repetitive with too similar solutions. - Quite basic and few game mechanics - Some users are prone to motion sickness with the amount of zooming in and out. - Dull and a little annoying soundtrack.

A key positive from Patricks Parabox is the overall game simplicity but level design still requires the user to think outside the box. I wanted to keep my game quite simple to ensure beginner users to the subject can succeed and progress through the game but still having puzzles requiring a little thought. Some cons I wanted to avoid from Patricks Parabox were levels getting repetitive with the same solutions, and the game being too simple that there are not enough unique game mechanics. User testing and feedback was a good indicator of this balance.

This project's proposed solution is quite different overall from existing solutions, but key aspects were heavily inspired from some aspects from each of the analysed existing solutions. Overall, the proposed solution is unique in its way of incorporating level design, user interaction and progression though video game aspects with programming structure and functionality.

1.3 Tools

For this project the programming language C# is used, and the software's Unity, Visual Studio, Photoshop, Illustrator, and GitHub Desktop were used. When selecting tools and technologies for this project it was important to keep in mind my limited knowledge of game development, its tools, and the time frame of the project. Therefore, I analysed the pros and cons of the two most well-known game development engines Unity and Unreal Engine for ease of access, ease of use and ease of learnability.

Unity

Pros	Cons
- Beginner-friendly, Unity is known to have a user-friendly interface and	- Unity has performance limitations which may not perform as well as

development tools for beginners to learn and create games. - Unity has a large community and documentation which allows for lots of sources for problem solving, support and tutorials. - It has cross platform development allowing games to be built for multiple platforms e.g. desktops and mobile phones. - Unity asset store contains a large amount of free and paid assets. - Unity supports 2D and 3D development, offering beginner friendly flexibility.	Unreal Engine in some scenarios with highly detailed graphics. - Unity does not provide advanced rendering capabilities like Unreal Engine. - Unity licensing requires paid subscription for certain features and services. - Some advanced features of unity have a learning curve and require time and effort to use properly.

Unreal Engine

Pros	Cons
- Unreal engine has powerful graphics and rendering capabilities. - Uses a blueprint visual scripting allowing for beginners to create gameplay without much coding knowledge. - Robust physics engine for realistic physics simulations. - Unreal engine is free until a revenue threshold is reached. - Unreal engines marketplace and community allow access to existing assets and tutorials.	- Unreal engine has a steeper learning curve than other game engines. - Unreal engine initially did not support 2D games at the start of this project's development. - Longer compilation time and interaction speeds cause slower development. - Limited cross platform support, requiring more effort to achieve cross platform compatibility. - Unreal engine has a smaller asset marketplace compared to unity.

Overall, Unity was the obvious choice due to it being more beginner friendly, better suited to the project as it supports 2D game development whilst Unreal Engine did not at the start of development. The cons of Unity were not as critical as the cons of Unreal Engine, as Unity has primary cons relating to rendering highly detailed graphics, which this project does not have. This project did not need any features or services requiring the paid subscription and there are plenty of online resources to help learn some of the advanced features of Unity. Unreal Engines' cons include a steeper learning curve, did not support 2D game development, and would require longer compilation time as it usually handles deep rendering, which all conflict with this project. Additionally, I already had a small amount of existing knowledge with Unity game development.

The choice of picking Unity also choose the programming language of C# and the integrated development environment of Visual Studio as they are both the default and recommended options for working in Unity. C# is a good fit as it provides strong object-oriented programming suited for game development, and I used the default of Visual Studio as it is currently my preferred IDE and provides robust tools for writing and debugging code.

I ended up using Photoshop and Illustrator for my tools to create and modify the image sprites of the objects, as I already had them and have experience using them. They also provide a very strong ability to create and modify 2D images with their range of tools.

Lastly, I used the GitHub Desktop software to push and pull GitHub commits as it can be integrated with Unity projects allowing for easy modification, version control and update commits to GitHub.

1.4 Final product key Aspects

The game contains blocks of code, with simple phrases, numbers, or words on them, that can be arranged to create functioning code structures. The current version of the project has 22 levels ranging in difficulty which contains simple code structures, infinite/looping code structures, logic comparisons, multiple simultaneous code structures, multiple controllable players entities, codable controlled entities, and method blocks which can be called, and implement recursion. This table contains the levels and what concepts they introduce, concepts they contain:

Level	Concept Included/Introduced
1-2	Simple If statement
3-6	Simple If statement with logic functions such as Or and And.
7	Multiple If statements on one line
8-9	If Else statements
10	Multiple If and Else statements on one line
11-12	Door state variables, door objects
13	Multiple controllable player entities, multiple simultaneous code structures
14	Multiple simultaneous existing code structures
15	Codable controlled entities
16-17	Infinite/looping code structures, codable controlled entities
18-19	Pong: Combines multiple simultaneous looping code structures and many variables.

20	Snake: buttons to control and play snake.
21-22	Snake: Combines multiple simultaneous looping code structures, many variables, method and call method functions, recursion.

1.5 Method

I choose to work with an agile method due to its suitability for game development in the sense that you want to continuously get user testing and feedback throughout development to adjust the difficulty, flow of levels, find bugs, and understand how real users might interact with the game. Using iterative design, cycling prototyping, testing, analyzing, and refining allows for regular refinements and improvement to existing aspects of development from user feedback and evolving requirements. It allows for flexible and adaptable development which helps in game development due to ideas and plans always changing for a better experience. Continuous testing allows for quality assurance and smooth gameplay as user tests will allow the game to be seen and played from different perspectives to find problems of features and mechanics. In my case I had six students who gave user testing and feedback, three level 300 students over five stages throughout development, two level 400 students and one level 100 student towards the end and final development stages to get fresh perspectives on a more complete product, who don't have previous knowledge from past development stages.

2.1 Conceptual Design

The game is designed to firstly introduce the simplest concepts that are then built upon, starting with very simple levels containing a single If statement, as seen in figure 1, then level by level introducing new blocks and concepts such as logic comparisons, multiple if statements, If Else statements, level objects, object state variables, multiple simultaneous code statements, codable entities, looping statements, and gradually building up knowledge and complexity, then ending up with method functions and recursion within constructivism approaches of old school games of Mazes, Pong and Snake to enable easier understanding with provided blocks for logic and concepts, as seen in figure 2.

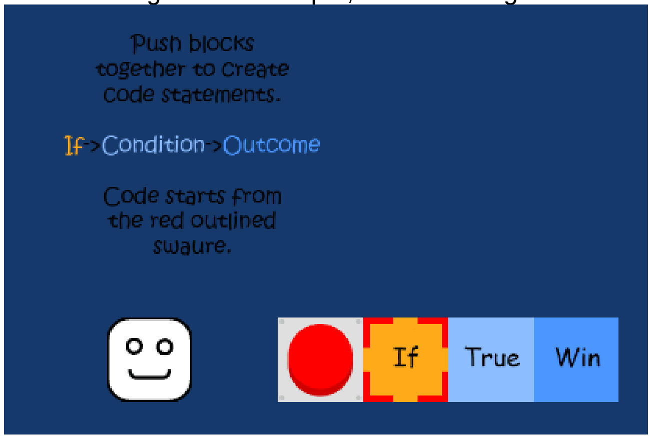
Push blocks together to create code statements.
If > Condition > Outcome
Code starts from the red outlined swaure.
If
True
Win

Fig. 1. Level 2 of game.

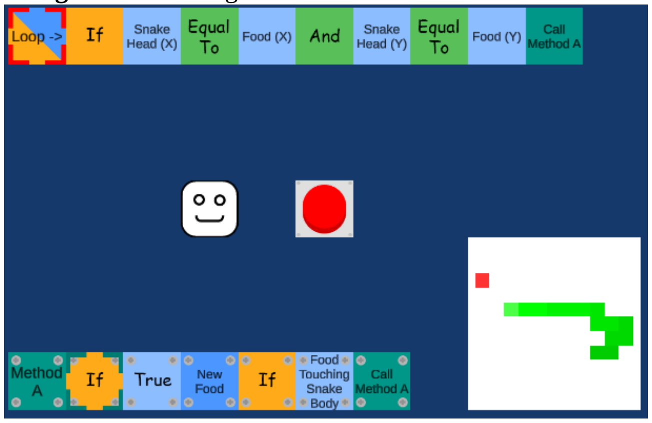

Fig. 2. Level 21 of game.

2.2 Requirements and Constraints in Design Choices

The main requirements and constraints considered when making design choices were the eight non-functional requirements that I set at the beginning of the project to evaluate performance. These eight-performance metrics were derived from Jakob Nielsen's five components of usability *[11] and* are:

- **Learnability** -The game must be able to efficiently convey ideas and levels to enable the user to learn and use ideas in the environment. Using a constructivism approach to create levels that the user will have some existing understanding of. Learnability has the highest significance in evaluation as the project's goal is to teach concepts, and if the project cannot do that, it is not a solution to the problem.
- **Performance** -Fast and responsive, with minimal lag or delay in processing user inputs and displaying results. Performance is significant as if the game does not perform well and is very slow to open and use, it will not maintain user focus and enjoyment.
- **Usability** -Easy to use system, with a simple and intuitive user interface that requires minimal training. Usability is another focus, as if users struggle to use the system, they will not be able to learn anything.
- **Reliability** -Minimal downtime, with error prevention, robust error handling and fault tolerance mechanisms. Preventing bugs and errors is crucial to keep user attention as no one wants to play a game that keeps breaking and it will prevent the user from being able to enjoy and learn anything.
- **User Experience** -Engaging visuals and multimedia content. Simple and clear examples or scenarios to help the user understand how to complete levels. Good user experience is required to maintain user attention, to encourage replay ability and expand awareness of the game to others.
- **Accessibility** -System built or procedures in place to allow users with disabilities to use the system, such as development with color blind friendly colors. Accessibility is required to evaluate to ensure the solution can be used by anyone and not exclude any persons or groups.

These eight requirements were derived from five components of usability user interface design of Jakob Nielsen's heuristics *[11].* These heuristics were the requirements and constraints, for example the heuristic of performance required fast and responsive interactions, with minimal lag or delay in processing user inputs and displaying results, and learnability and usability requiring the game to be able to efficiently convey ideas and levels, and being an easy to use system with a simple and intuitive user interface that requires minimal training to enable the user to learn and use ideas in the environment. To analyze these heuristics throughout testing, I watched testers play through the game to see what, where, and why they get confused, struggle, or progress easily. Personally, watching and discussing their thought processes allowed for an equal and in-depth analysis of my requirements. Additionally, I asked testers questions after completing tests to get overall feedback about their final thoughts of learnability, usability, and user experience.

2.3 Implementation

Implementation currently includes basic player controls, levels, obstacles, basic GUI, tutorials, win conditions, and conditions functionality. Code blocks implemented include blocks of If, Else, True, False, And, Or, Equal To, Not Equal To, Greater Than, Less Than, Win, Lose, Open Door, Close Door, Loop, Call Method, and many unique blocks relating to the interactions in Pong, Snake, and Robot controls. The implementation of current code blocks allows for an endless code structure, assuming the code is arranged correctly, in an endless level with an endless number of blocks. Multiple code structures can be created and executed on a level from select positions.

3.1 Results

User testing and feedback was performed throughout development on level 300 COMP/SWEN students and some final user testing and feedback was performed by some level 400 and 100 COMP/SWEN students to assess the solution without any previous knowledge from past development tests.

All the six final testers found the concept and functionality of the game fun and interactive, and all managed to complete the game from start to finish without any issues showing the levels do not become too hard for any user of the target audience. All testers agreed the tool is believed to be a good new resource for learning and getting familiar with the beginner concepts within computer science programming, with some testers playing, discussing, and providing thoughts on the game over multiple hours.

The user feedback from the initial stages of development, helped shape the design of the game in terms of the learnability, usability, and user experience metrics, as the final decisions to keep all blocks the same sizes and shapes ultimately kept the design simple and easy to learn. Feedback through intermediate stages helped find the correct flow and incrementing difficulty of levels and provide feedback on what elements are less clear and require changes or visual explanation/hints. Testing and feedback throughout the later stages helped direct the game's final complex features in the shape of method blocks for elements of levels and recursion, whilst deciding the value of implementing an initial idea of a compression block not to be needed or able to provide any further context or learnability to the game.

Comparing final user testing with the mentioned performance metrics, all users were able to play and complete the game without issue showing the accessibility, reliability, and usability are at a suitable level. There were no performance problems throughout testing, but some users believed the movement functionality was a little clunky due to movement being limited to one input per movement and cannot be held down. All the testers said the learnability and user experience was enjoyable and easy to understand. The final average difficulty rating for the levels were: (0 being little user input, little time taken, and/or little thought required and 10 being a large amount of user input, time, and/or thought required).

Level	Average Rated Difficulty
1-2	1
3-6	1
7	2
8-9	3
10	4
11-12	3
13	2
14	5
15	1
16-17	4
18-19	7
20	3
21-22	7

3.2 Sustainability

The design of this unity game project addresses social sustainability considerations. The design of an online resource for teaching computer science concepts provides an alternative accessible learning method to a wider range of people in the community regardless of their socio-economic status or geographical location allowing for equal access and opportunity to a learning resource. This learning resource tool attempts to teach users about the beginner threshold concepts in the computer science field and can enable learners to gain a better understanding of programming, which is an important sustainability consideration as the importance of electronics in society is always increasing.

Looking into the 17 Sustainable Development Goals (SDGs) *[12]* quite a few do not overlap with this project as it is entirely composed of software, but there still are some overlaps due to the project's goal of education.

- Quality Education (SDG 4) is the most direct overlap as the project aims to teach basic concepts through a fun and interactive game. SDG 4 aims to ensure inclusive and equitable quality education and this project's approach makes learning more engaging and accessible, contributing to quality education.
- Decent Work and Economic Growth (SDG 8) overlaps with the project as computer science is a field with significant job opportunities and economic potential. By teaching these skills, the project could indirectly contribute to inclusive and sustainable economic growth.
- Industry, Innovation, and Infrastructure (SDG 9) Developing a Unity game project involves innovative use of technology and can inspire learners to pursue careers in industries that rely on such skills. This aligns with the goal to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.
- Reduced Inequalities (SDG 10) The project can help reduce some inequalities in access to education and skills development opportunities as it could be freely provided and accessed by anyone wanting to learn some basic computer science concepts.

3.3 Limitations

Having to learn and use multiple tools with my limited knowledge was also a limitation within this project, as whilst I had some existing experience with Unity, Photoshop, and Illustrator, I was still limited in having to learn some basic features and could not use more complex features and options that may have made the game look or function at a higher standard. With my limitation in knowledge and experience with visual design the game’s appearance is restricted to being quite basic.

User testers were also another limitation for this project as the primary target audience is level 100 students learning computer science, and I only managed to get one student from this demographic. The level 300 and 400 students did provide suitable testing and feedback, but they will always be somewhat biased due to their own greater knowledge than the target audience. So, a greater amount and variance in testers could have been more beneficial for development.

3.4 Further Development

While the current state of the project is a functioning game that contains aspects of computer science concepts, the current depth of the concepts explored could be further developed with many more levels covering each concept, providing more challenges and puzzles to go through. Many more games could be developed within the game to provide aspects for the user to explore in each game, for example a tower defense game, sudoku, tic tac toe, etc.

Further development for the user experience could be done by improving the quality of the visual design, adding more interactive menus and title screens, making the levels locked for a feel of progression, and development on music and animation would be good places to start.

References

[1] "Why Do Students Drop Out of Tech Studies?" Wawiwa Tech, [Online]. Available: https://wawiwa-tech.com/blog/why-do-students-drop-out-of-tech-studies/. Accessed on: 08/10/2023.

[2] "Dropping Out of Computer Science Studies: Let's Talk About It," Infinity Labs, [Online]. Available: https://infinitylabs.co.il/dropping-out-of-computer-science-studies-lets-talk-about-it/. Accessed on: 08/10/2023.

[3] "Computer Science Undergraduates Most Likely to Drop Out," Computer Weekly, [Online]. Available: https://www.computerweekly.com/news/252467745/Computer-science-undergraduates-most-likely-to-drop-out. Accessed on: 08/10/2023.

[4] "The Role of Usability in Learning Environments: A Conceptual Framework for Understanding the Impact of Usability in Educational Contexts," Technology, Education and Learning Research, SpringerOpen, [Online]. Available: https://telrp.springeropen.com/articles/10.1186/s41039-020-0122-3. Accessed on: 11/10/2023.

[5] "A Systematic Review of Usability Evaluation in Web Development," Journal of Computer Assisted Learning, Wiley Online Library, [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1111/jcal.12498. Accessed on: 11/10/2023.

[6] "Conditional Statement," Study Smarter, [Online]. Available: https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/conditional-statement/. Accessed on: 11/10/2023.

[7] "Constructivism," Simply Psychology, [Online]. Available: https://www.simplypsychology.org/constructivism.html. Accessed on: 11/10/2023.

[8] Scratch, MIT Media Lab, [Online]. Available: https://scratch.mit.edu/. Accessed on: 09/10/2023.

[9] Baba Is You, Hempuli, [Online]. Available: https://hempuli.com/baba/. Accessed on: 09/10/2023.

[10] Patrick's Parabox, Patrick Traynor, [Online]. Available: https://www.patricksparabox.com/. Accessed on: 09/10/2023.

[11] "Usability 101: Introduction to Usability," Nielsen Norman Group, [Online]. Available:

https://www.nngroup.com/articles/usability-101-introduction-to-usability/. Accessed on: 09/10/2023.

[12] "Goals," United Nations Sustainable Development Goals, [Online]. Available: https://sdgs.un.org/goals. Accessed on: 08/10/2023.