The Icarus Protocol

Project Description and Clarification

Lincoln Middle School

Pluribus Doctrina

Collin Nelson;

Anna Ueti;

I. Introduction

In 2021, the games industry reported estimated revenues of about 198.40 billion dollars, solidly positioning themselves as one of the world's most immensely popular and valuable forms of entertainment. This growth is only projected to continue rising as well, with market research firms anticipating revenues as high as 339.95 billion by 2027 (*Gaming market size, share: 2022 - 27: Industry growth* 2021). The power of games to capture the minds and attentions of players across the world cannot be discounted, and given this power, there is also incredible potential for games to capture the attention of players and direct it towards learning and personal advancement as well as towards entertainment.

There is a powerful overlap between games and education, an overlap that innovators have only recently begun to truly explore. The feedback loops and reward models designed and honed to keep people engaged in modes of entertainment can be useful for intriguing students in classwork with a reputation for being uninteresting. Our team aims to explore the potential of games as a tool for education by producing a fully featured game designed to teach basic programming skills to students at a middle school or early high school level with no prior programming experience. While games of a similar nature exist, they often fall into one of two traps which our team sees as pitfalls. Some, while employing the surface level appearance of a game, fail to truly embody the game design principles that make games powerful for learning; Others use a proprietary scripting language that has lessened impact in teaching actionable programming skills. In the construction of this project (working title: "Icarus Protocol") we aim to solve this problem by producing a game that is a genuinely fun and interesting experience, while also serving as an effective tool for teaching real Python programming.

II. Background and Related Work

In order to evaluate current leaders in the same domain as our project, we must first define that domain. For the purposes of this document, we have narrowed down this definition to "*interactive gamified services designed to teach programming.*" Given this definition, our research highlights three representative and distinct examples of highly successful existing products in the same domain.

The first notable leader in the domain, and the closest analogue to our work, is the service CodeCombat (CodeCombat, 2013). They create a number of educational tools, but the original product "CodeCombat Classroom" is of the most relevance to us. Their game, originally released in 2013, teaches programming to a similar age group to our target demographic using a series of fantasy-themed levels covering programming concepts in Python, C++, and Javascript. They have also implemented several potentially useful features, such as integration with Google classroom for teachers. Due to the larger scale of their company, they have also been able to fund studies on the effectiveness of this learning model in classroom. The most recent of these, released in 2019, shows that 90% of teachers agreed that this learning model kept students engaged, and 97% agreed that the gain to students was worth their time. (Danks et al., 2019) It also contains several other useful insights about factors such as ideal session time which will be useful in constructing our project. While a survey of CodeCombat's curriculum shows some overlap with the intentions of *Icarus Protocol*, the structure of their lessons lends itself to AI programming, which while generally good for teaching basic concepts, makes it difficult to teach concepts such as data conversion or file operations. Our hope is that *Icarus Protocol* can innovate in this field by using a narrative context that lends itself to a broader array of curriculum.

The second state-of-the-art contemporary of note is the online coding school *Codecademy (2011)*, which teaches many different professional programming languages. While it doesn't specifically target middle schoolers, it also teaches lessons assuming no prior experience with the relevant coding languages. They are set apart by the variety of lessons they offer, and the breadth of curriculum available on their site. While they opt to not integrate full gamification into their service, they do employ progress bars and completion badges to strengthen game-like feedback loops.

The third related work that I would like to highlight in our research is the simple online game creation platform *Scratch (2007)*. This MIT project allows people to create games with a simple all-inclusive editor. It is not in and of itself a game, but it teaches programming concepts through the game creation process. It utilizes a proprietary building-blocks programming language with which the students can script the behavior of sprites created in the program. This tool is particularly important to our project as it is one of the tools currently being used by Lincoln Middle School during their units on programming. It has the advantage of being very powerful, and broad in the concepts it can teach, but cannot function autonomously since it doesn't come with a structured curriculum. It also teaches fundamental programming concepts, but opts not to use a real programming language to reduce complexity. With *Icarus Protocol*, we aim to provide an equally useful tool to accompany or serve as an alternative to Scratch in the classroom at Lincoln MIddle School, but one which is capable of autonomous function and which teaches true actionable programming skills.

In summary, while there are several notable existing services that fall within the provided definition of the project domain, very few accomplish the true intersection of game, educational tool, and real programming simulation that we aim to create in *Icarus Protocol*. Our project is distinguished from services like CodeCombat, which do also accomplish these things, by the curriculum that it is enabled to tackle, and the platforms it is designed to operate on. It is also distinguished by its distinct theming and narrative elements, which are unique from any prior service or tool uncovered in my research.

In order to accomplish the technical aspect of the project there are a few tools and frameworks we will have to master or improve our skills in. The most important is Unity. As the engine for the game, experience with its setup, debugging, and optimization tools will be required to make the game function, and function efficiently enough to be smoothly playable on the often outdated computers that are available to students. In addition, in order to integrate real Python programming into the game we intend to use the IronPython framework. Our team does not currently have any experience among us with this framework, so skills will need to be learned as the project progresses. The final notable technology with which our team is not already familiar is the Chrome OS and the process for building executables for it. While we will certainly be able to build for Windows, our desire to make the project as available to students as possible will require us to look into the requirements for Chrome applications, and potentially the process for submitting work to the Chrome Store.

III. Project Overview

There is incredible potential in games and game design principles to forward the field of effective education. Since the early 2000s, researchers have been talking about how educational games should be "like the school corridor, where kids experiment, interact, create, and share" (Squire & Jenkins, 2003), and even earlier science fiction authors like Orson Scott Card envisioned a world in *Ender's Game* where students learned complex battlefield strategy through interactive simulations. (Card, 1985). *Icarus Protocol* aims to explore the use of games

to engage kids in educational settings by placing them into the role of a starship AI on a quest to repair its malfunctioning vessel and discover the source of the damage.

In doing this there are several key overarching objectives that define the nature of the project. The first of these is that the game must be *fun*. Many other similar games fall into the pitfall of being coding simulations first, and games second. They ignore many of the feedback systems and incentive structures that make games effective at prompting player learning, and this causes them to lose some of the impact they could otherwise have. *Icarus Protocol* needs to employ these systems in full, existing as equal parts educational tool and game, and using systems like level scoring, completion percentages, secrets, and challenge modes to make the experience exciting and engaging to play.

Additionally, *Icarus Protocol* would be failing in its purpose if it isn't an effective tool for teaching the intended programming skills. While being fun to play, the game must also be demonstrably effective at allowing students to gain transferable and extendable coding skills without needing to be accompanied by traditional lectures or homework assignments. It should be able to operate independently (i.e. without assistance from a teacher) and help students to solve their problems and correct their mistakes.

The project will be designed and built from the ground up in the Unity Engine, which will serve to accelerate the process of game development, improve the general quality of the final project, and allow for simultaneous builds to multiple key platforms. Design and development will take place on Windows, and the game will be built for Windows PCs, optimized to be smoothly playable on laptops available to students. In addition, if features are ultimately compatible, a secondary build will be constructed and deployed for Chrome OS, intended to be played on the student-issued chromebooks used by Lincoln Middle School.

The game itself will consist of a sequence of levels, each themed after a different malfunctioning system of the starship. Each level represents a single coherent concept that the game intends to teach to the students. As a baseline for simple programming skills, the game will have levels on ideas such as arithmetic statements, function calls, if statements, loops, and lists. Each level will be constructed of a series of "phases", of increasing complexity, beginning with an introduction to the concept and culminating to a programming task that requires the synthesis of the new concepts with concepts introduced in previous levels.

The game should use reward systems to incentivize the player to achieve full level completion, and should also use systems which reward novel, intelligent, or especially efficient solutions to the more complex programming tasks. In addition, the game should be flexible in accommodating optionally difficult challenges for students who feel they have a firm grasp of the material and wish to stretch their abilities.

All of the above objectives should be shaped and guided, though not inherently constrained by the Washington State 6-8th grade learning standards for computer science, particularly educational standards 2-AP-10 through 2-AP-19, which describe the learning goals and standards in computer science classes for students between the 6th and 8th grade in Washington State.

IV. Client and Stakeholder Identification and Preferences

Our primary client is Lincoln Middle School, where Mr. Davis, the school's primary technology teacher, will be our contact liaison. The final project will be predominantly used by LMS as an introductory tool for students to gain experience with the Python programming language. There are multiple stakeholders within Pullman School Districts, which include but are not limited to Lincoln Middle School.

In order to enhance core programming fundamentals retention in middle school students, we're working with Lincoln Middle School to create an educational video game that uses positive feedback loops and a reward model framework. This project will require a released build of the game, and the deployment of the game to Windows PC for the Lincoln Middle School computers.

Potential clients would include other K-12 educational institutions. To appeal to these institutions, it would be important for the final build of our software project to be easy to download and install. Additionally, it would be greatly beneficial if the curriculum adheres to the CSTA's guidelines for 6-8 grade education. (Ospi, 2018)

Finally, all stakeholders of the project would benefit from clean and well-documented code that is easy to deploy and to extend. The *Icarus Protocol* team will endeavor to treat these needs as prerequisites as we fulfill the preferences identified above. The needs of our primary client (LMS) will be prioritized first, but the needs of other institutions will be considered throughout the design and development process.

V. Glossary

Feedback Loops: A term in game design which refers to using the players success or failure to adapt and adjust the output of other systems. This could mean adjusting up the difficulty of challenges if the player appears to be succeeding too easily, or providing hints only when the player shows signs of struggling.

Reward Models: A game's reward model is the structure and array of visual, auditory, and mechanical rewards that are given to the player to incentivize success, as well as the specific situations in which rewards are given to incentivize the desired behavior.

Scripting Language: A programming language, usually an interpreted language, which can be integrated in with a compiled executable to allow for the adjustment or extension of existing features or content without recompilation of the primary executable or exposure of its source code.

Gamified: Incorporating certain aspects of game design, such as lives, points, levels, or skill trees in an attempt to make ordinarily non-entertainment activities more engaging or rewarding to perform.

Building-blocks Programming Language: A method of scripting or programming behavior that involves connecting predefined behavior blocks. While it may utilize coding logic, it requires little to no handwritten code on the part of the user.

VI. References

"Coding games to learn python and JavaScript," *CodeCombat*, 2022. [Online]. Available: https://codecombat.com/home. [Accessed: 20-Sep-2022].

Computer Science Teachers Association (CSTA), "Computer Science 6-8 Standards," *Ospi*, 2018. [Online]. Available: https://www.k12.wa.us/student-success/resources-subject-area/computer-science/computer-science-k-12-learning-standards. [Accessed: 20-Sep-2022].

"Scratch Homepage" *Scratch*, 2007. [Online]. Available: https://scratch.mit.edu/. [Accessed: 20-Sep-2022].

"IronPython," *IronPython.net /*. [Online]. Available: https://ironpython.net/. [Accessed: 20-Sep-2022].

"Gaming market size, share: 2022 - 27: Industry growth," *Gaming Market Size, Share | 2022 - 27 | Industry Growth*, 2021. [Online]. Available: https://www.mordorintelligence.com/industry-reports/global-gaming-market. [Accessed: 20-Sep-2022].

K. Squire and H. Jenkins, "Harnessing the power of games in education," *InSight*, vol. 3, 2003.

"Codecademy Homepage," *Codecademy*, 2011. [Online]. Available: https://www.codecademy.com/. [Accessed: 20-Sep-2022].

O. S. Card, *Ender's Game*. New York: Tor, 2021.

S. Danks, B. Fraumeni, and M. Smrekar, "CodeCombat Implementation Study," *McREL International*, Feb. 2019.