Serious Tabletop Game Design Essay

Haoxuan Wang

1. Overview

System Overload is a two-player serious tabletop card game. The game is primarily targeted at primary programmers, who frequently encounter challenges in structuring code and planning execution logic when engaging with programming tasks. To solve these learning challenges, the game abstracts the programming process into structured in-game decisions, thereby supporting learners in both understanding and remember common programming functions.

2. Learning Outcomes

As a serious game, the main idea of this game is to enable players to turn concrete tasks into clear execution logic and to construct the most efficient and functional structures if possible. Therefore, there are 3 core learning objectives. First, help players understand the fundamental mechanisms of code execution and execution order, from deployment to execution and final resolution. Second, guides players to identify different programming functions and their respective roles, and to learn how to select and combine them according to specific goals (such as for and while loops). Third, under the constraint of Code Capacity, the game trains structured and algorithmic thinking by encouraging players to plan, make trade-offs, and optimise their programs in order to achieve objectives efficiently. Based on these objectives, this essay adopts Bloom's Taxonomy[1] as a framework to categorise the learning outcomes, illustrating how players progressively develop their learning.

- Remembering

Players are able to identify and memorise the abstracted in-game representations of fundamental programming concepts, such as code capacity (memory limit), code cards (functions), and execution order. Players can also remember commonly used programming functions during the play.

- Understanding

Players construct programs based on their own intentions, demonstrating the ability to explain the aim of their constructed programs in their own words, as well as to explain why the same set of cards can produce different outcomes when arranged in different orders.

- Applying

Players are able to apply their understanding of the game to a range of new situations. Based on both their own state and their opponent's state, they need to determine the objective for the current turn, such as focus on defend or attack, select appropriate combinations of cards and decide whether to avoid or strategically exploit the risks and rewards associated with overload.

- Analysing

Players need to analyse the relationships between internal modules within a program, distinguishing between attack, defend or other components, then take appropriate actions based on this analysis.

- Evaluating

Players are able to reflect on and deconstruct their decisions made in each turn after gameplay, analysing which choices contributed to victory or led to defeat, and then envision alternative possibilities while proposing potential directions for improvement.

- Creating

Players are able to create different types of programs across multiple gameplay sessions by combining different kind of cards into program frameworks with specific functions. In response to different situations, they need to make real time adjustments, gradually developing their own construction logic. As a result, the entire gameplay

process functions as an iterative process of program creation.

3. Design Rationale

Research shows that many novice programmers do not struggle primarily with syntax itself, but rather lack the transferable problem solving strategies characteristic of expert programmers.[2] Thus, the design rationale of this game is to transform the act of writing code into a repeatable, gamified process of logical construction. This allows players to learn programmatic logic without requiring extensive prior familiarity with specific programming languages or syntax. Compared with traditional learning approaches that emphasise memorising syntax before understanding program logic, this approach lowers the entry barrier to programming. By skipping the relatively tedious stage of syntax memorisation, the game aims to make programming more accessible and engaging, thereby encouraging a broader range of learners to develop an interest in coding. The Code Capacity mechanic is designed to help players develop an understanding of resource management while highlighting the importance of code optimisation in programming. This mechanic is inspired by the mana crystal system in Hearthstone and the charm overload system in Hollow Knight, and integrates these ideas into a capacity model that allows players to deliberately enter an overload state. Through this design, players are encouraged to weigh short-term gains against long-term costs, reinforcing strategic decision making under constrained resources. The game's basic mechanics, such as the health and shield systems, are inspired by Slay the Spire. In the initial design concept, the game was also intended to include a single-player Roguelike mode, allowing solo players to experience the game independently. The deck drawing mechanic is designed as Open Drafting, allowing players to make more meaningful choices[3] while reducing randomness and increasing the emphasis on skill-based decisions. The game design also consider Bartle's Player Types[4] as reference. PVP interaction and error cards that directly disrupt opponent's codes provide enjoyment for killer type players. The need to adapt strategies to different game states offers enjoyment for explorer type players, while the gradual optimisation of programs and clearly defined victory conditions appeal to achiever type players. However, as the game is designed as a 1v1 PVP game, it currently offers limited opportunities for social interaction and is therefore less engaging for socializer type players. In addition, when using AI tools to create the visual design of Code Cards, two alternative design approaches were considered: one featuring a central illustration, and the other placing the card name at the centre of the card. The latter approach was ultimately selected. While the former design benefits from the Picture Superiority Effect[5], which can enhance memorability through visual imagery, providing an enlarged display of the card name was considered more effective in helping players remember the card's name(which is related to one of the learning outcomes) rather than its illustration. Moreover, this design choice reinforces the metaphor of assembling code into executable programs, preserving the programming oriented aesthetic and conceptual coherence of the gameplay experience.

4. Play Test

An initial version of the game was tested in the class setting to evaluate its comprehensibility and level of engagement. Due to time constraints, only two play test were conducted. During these sessions, testers noted that the core rules of the game were relatively clear and easy to understand. This suggests that the game is accessible even to players with little to no prior knowledge of programming. However, some mechanics had not yet been fully developed, leading to potential conflicts. For example, although code cards on the same line are intended to execute simultaneously, the rules did not sufficiently clarify how simultaneous attack and defence actions should be resolved. As a result, a priority system between different card types was introduced in subsequent iterations of the design. In addition, another issue identified during the play test was that the limited number of cards in the prototype led to repetitive and less engaging gameplay. However, due to time constraints, this issue was not fully

addressed within the scope of the project. Furthermore, both testers responded positively to the for and while loop mechanics, suggesting that these elements enhanced player engagement and introduced a higher degree of creative freedom within the gameplay.

5. Evaluation

This game can be improved in several respects. First, due to time constraints, both the overall scope of the game and its visual design remain underdeveloped. These aspects could be refined through future evaluations by expanding the range of cards to incorporate additional programming concepts, such as stacks and arrays. Furthermore, the visual quality of the cards could be enhanced through the use of AI-assisted tools or collaboration with professional illustrators. If the game were to be commercially released, additional expansion packs could also be developed to extend the gameplay. Second, the single-player Roguelike mode discussed in the third section of this essay could be further developed by introducing a set of enemies and boss, and a relic system. Such a system could later be integrated into the existing PvP mode which add appropriate randomness to the game and is particularly well suited to conveying computational and programming-related concepts. This would also provide greater enjoyment for explorer type players. In addition, a cooperative competitive mode supporting a larger number of players could be introduced. For example, teammates could be allowed to exchange Code Cards, and certain cards could be designed to affect the entire team. These features would strengthen the game's social dimension and encourage collaborative learning through peer interaction, which make the game more suitable for socialiser type players. Additionally, introducing a time limit for each round could help players more easily enter a state of flow by maintaining an appropriate level of challenge and engagement[6].

Reference

[1] Krathwohl, D.R., 2002. A revision of Bloom's taxonomy: An overview. Theory into practice, 41(4), pp.212-218.

[2] de Raadt, M., Toleman, M. and Watson, R., 2004. Training strategic problem solvers. ACM SIGCSE Bulletin, 36(2), pp.48-51.

[3] Deci, E.L. and Ryan, R.M., 2000. The" what" and" why" of goal pursuits: Human needs and the self-determination of behavior. Psychological inquiry, 11(4), pp.227-268.

[4] Bartle, R., 1996. Hearts, clubs, diamonds, spades: Players who suit MUDs. Journal of MUD research, 1(1), p.19.

[5] Paivio, A. and Csapo, K., 1973. Picture superiority in free recall: Imagery or dual coding?. Cognitive psychology, 5(2), pp.176-206.

[6] Desurvire, H., Caplan, M. and Toth, J.A., 2004, April. Using heuristics to evaluate the playability of games. In CHI'04 extended abstracts on Human factors in computing systems (pp. 1509-1512).