System Overload Rule Book

Players: 2 **Average Duration:** 20 minutes

Introduction

System Overload is a serious competitive card game themed around computer systems programming. The game happens in a futuristic cyberpunk world. In this world, cities, transportation, healthcare, and energy—nearly every aspect of society is driven by codes, and the execution of the codes requires a large amount of Energy Cores to sustain it. In order to secure more resources, some people organized themselves into gangs and wage technological warfare under the city's networks against rival factions. Players will play as two rival computer masters, deploying and executing their own constructed "codes" within a limited Code Capacity, while strengthening their systems and disrupting their opponent.

Game Objective

In this game, players need to build, refine, and execute their own programs within a limited Code Capacity, aiming to deplete their opponent's Energy Cores to achieve final victory. Each round, players must make critical code optimizations to ensure their programs operate with maximum efficiency.

Learning Objectives

By playing System Overload, players will be able to:

1.Understand how code is executed during runtime.

2.Identify common code structures and their functions.

3.Engage in algorithmic thinking and optimize code under constrained conditions.

4.Develop and understand logical code construction skills.

Components

Character Cards (various) (now 2)

Code Cards (various) (now around 25)

Component Description

Character Cards

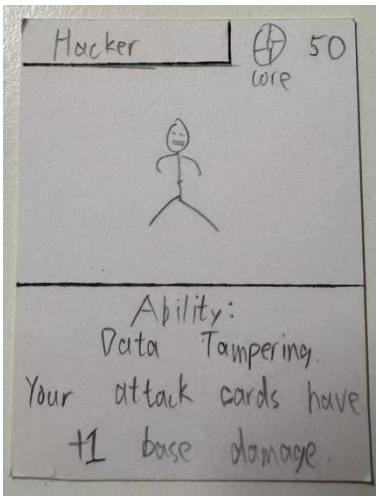
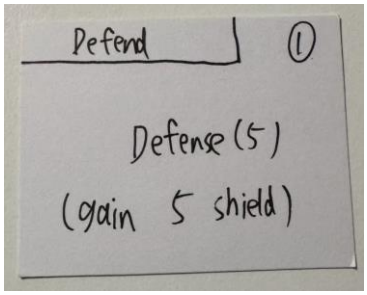


Figure1: prototype of a Character Card. Draw by hand.

Figure2: a processed Character Card. Created by ChatGPT.

Code Cards



Figur3: prototype of a Code Card. Draw by hand.

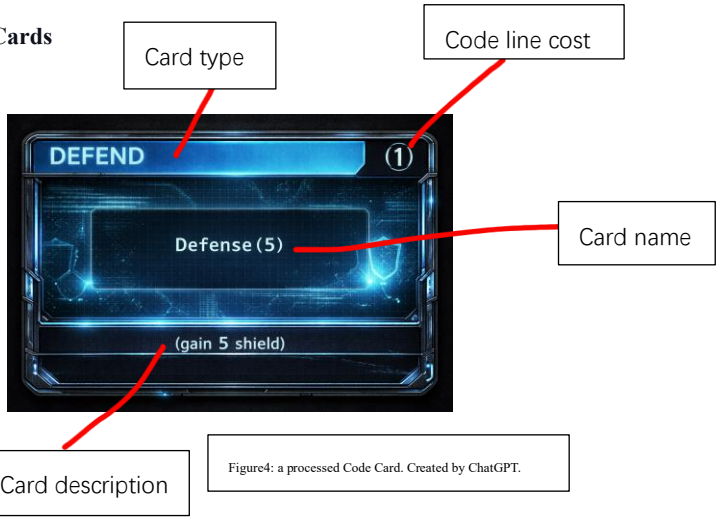
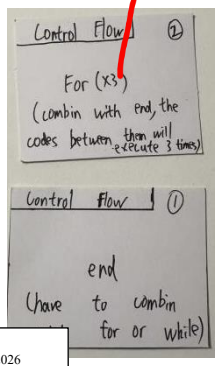


Figure4: a processed Code Card. Created by ChatGPT.

Special Code Cards

For Loop

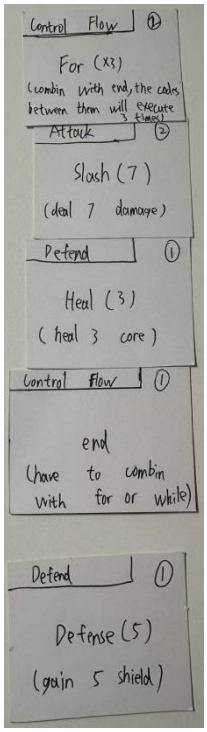
A loop consists of one ***for*** card and one ***end*** card, and any other code cards can be placed between them. After the entire loop finishes running (once the end card has resolved), all code cards between the for and end cards will immediately executed multiple times. The total number of executions is determined by the number shown in parentheses on the for card.



composition of for loop. Shoot on 13/01/2026

Example

In this case, the code on the second and third lines won't run immediately. Once the ***end*** on the fourth line has resolved, the code on lines 2 and 3 will be executed immediately, still following the top-to-bottom order: first ***slash (7)***, then ***heal (3)***, then ***slash (7)*** again… repeating this loop three times in total. After the loop finishes, proceed to run line 5. If the for or end becomes invalid for any reason, then all four lines of code are invalidated. If slash (7) or heal (3) becomes invalid, it does not affect the loop as a whole.When resolving this loop , that line is simply treated as having no effect.



6: example of a program with for loop. Shoot on 13/01/2026

While Loop

A loop consists of one ***while*** card and one ***end*** card, and any other code cards can be placed between them. After the entire loop finishes running (once the end card has resolved), all code cards between the ***while*** and ***end*** cards will immediately executed one time, and this effect triggers once after each line of code is executed by either player, until the end of execute stage, or until its execution condition is no longer met.

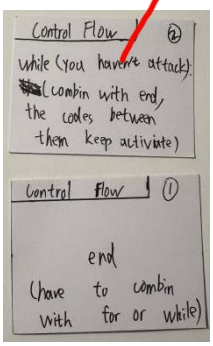
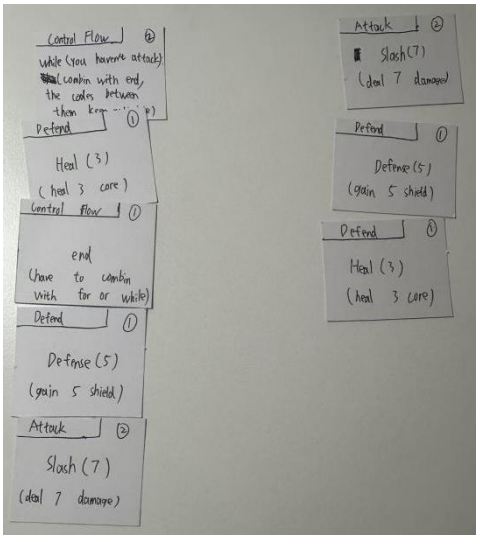


Figure7: composition of while loop. Shoot on 13/01/2026

Example



In this case, the code on the second won't run immediately. Once the end on the fourth line has resolved, the code on lines 2 will immediately execute once, still following the top-to-bottom order, then proceed to run line 4. At this time, although your opponent's program has already finished, your program will continue to run, and the while loop effect will remain active. This rule also applies when your program has finished but your opponent's program has not. After resolving the fourth line, ***defense (5)***, the code inside the while loop triggers once again, which is run ***heal (3)*** one time, and then proceed to resolve line 5, which works in the same way. After both players' programs have finished, the while loop effect ends and will not carry over into the next round.

op. Shoot on 13/01/2026

Setup

1. Choose a Character

Each player selects one Character Card and places it face-up in front of them as their character for the game. Players may also prepare paper and pens to keep track of their remaining Energy Cores.

2. Prepare the Deck

Shuffle all Code Cards to form a Deck, and place it face-down in the center of the table.

3. Draw Starting Code Cards

Each player draws five Code Cards randomly from the Deck to form their Starting Hand.

4. Build Your Initial Program

Players examine the five Code Cards they have drawn. Based on the Code Capacity (maximum of 10), they select any number of these cards and deploy them to their Program Area to construct an initial program. Code Cards deployed to the Program Area occupy the corresponding number of code lines. Different Code Cards require different numbers of lines. Any Code Cards not deployed remain in the player's hand and may be used in later turns.

The figure below shows an example setup.

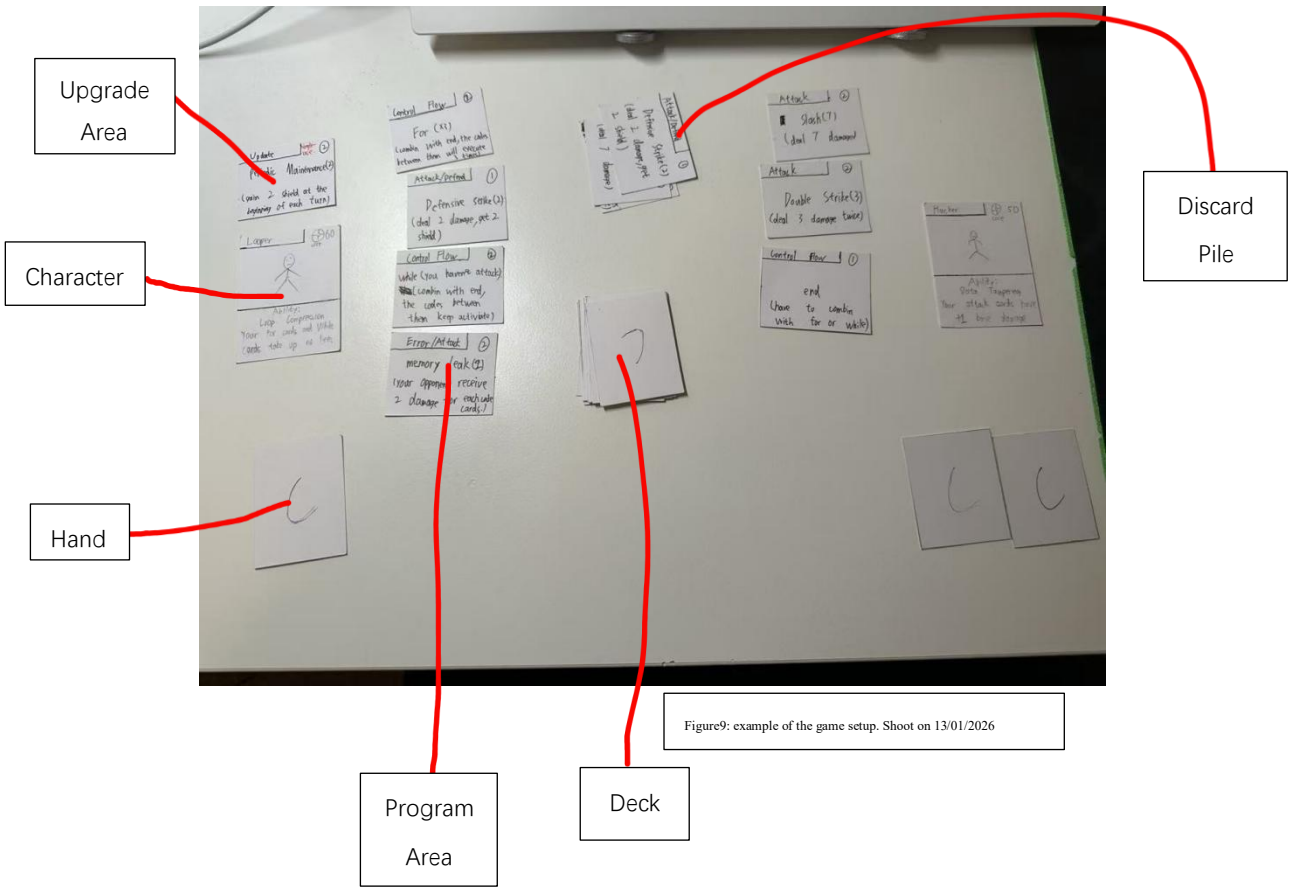


Figure9: example of the game setup. Shoot on 13/01/2026

Core Concepts

- Code capacity

Each player's system has a fixed Code Capacity, represents the amount of code that a program can safely execute.. The default starting Code Capacity is 10. Each Code Card occupies a specific number of Code Lines. Once deployed to the Program Area, it immediately consumes the corresponding capacity.

- Hand

Code Cards that are acquired but not used are kept in the player's hand for future use. The hand size limit is 5. Any cards in excess of this limit must be placed into the Discard Pile.

- Card Types

Attack: Deals damage to the opponent.

Defend: Grants shields to reduce incoming damage, shields will be cleared at the end of each round.

Error: Apply Debuffs to the opponent.

Update: Provides a persistent Buffs. After being executed once during a round, this card is placed in the Update Area, where its effect remains active.

Control Flow: Functional cards, enable combo effects when played together with other cards.

- Code priority

Error > Defence > Attack > Update > Control Flow

Code Cards on the same line are executed according to the priority listed above, with higher values resolving first. In addition, debuffs have higher priority than buffs.

- Overload

Players may choose to continue adding codes after reaching the Code Capacity limit. When this occurs, the player enters Overload. For each point by which the Code Capacity is exceeded, the player gains +1 Overload Level. At the end of the turn, the player suffers damage equal to twice their current Overload Level. Additionally, if the Overload Level is 3 or higher, the player will randomly lose one Code Card from their Program Area.

Round Structure

1. Start stage

The beginning of a round, certain buffs and debuffs can be triggered at this stage.

2. Draw stage

Both players draw three cards from the Deck and choose one to add to their hand. The unchosen cards will be placed into the Discard Pile. Players may choose who draws cards first. If there are not enough cards in the Deck, shuffle the Discard Pile and add them to the Deck.

3. Deploy stage

Players may choose to deploy Code Cards from their hand to the Program Area, rearrange or remove Code Cards already deployed in the Program Area. Any Code Cards that are removed are placed into the Discard Pile. During this phase, players can not see their opponent's Program Area.

4. Execute stage

Once both players have completed their code deployment, the game proceeds to the Execute stage. During this stage, both programs are executed simultaneously from top to bottom. Code Cards on the same line are resolved according to their priority values. The following are some example scenarios.

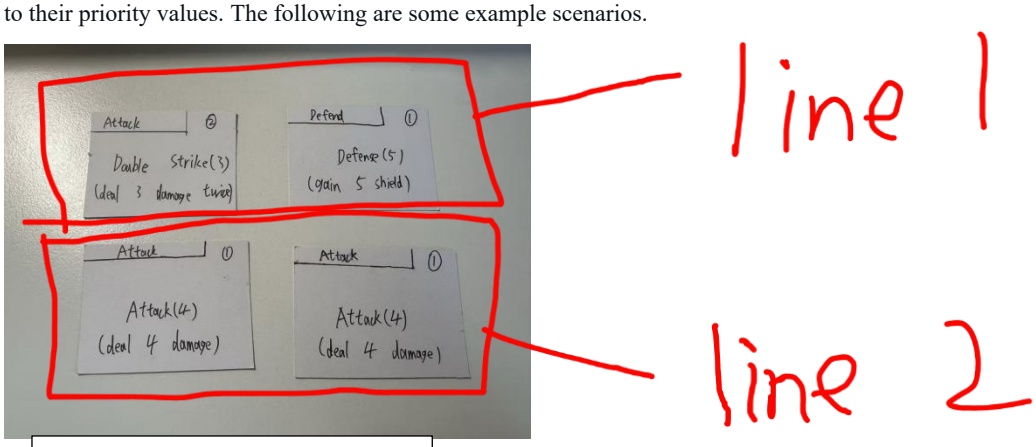


Figure10: example of a game scenarios . Shoot on 13/01/2026

In this situation, Line 1 runs first. Since Defend has a higher priority than Attack, the player on the right side gains 5 Shield first. Then, the player on the left deals 2 × 3 = 6 damage to the right player. Next, Line 2 runs. Because both Code Cards are Attack, they run simultaneously, and both players take 4 damage.

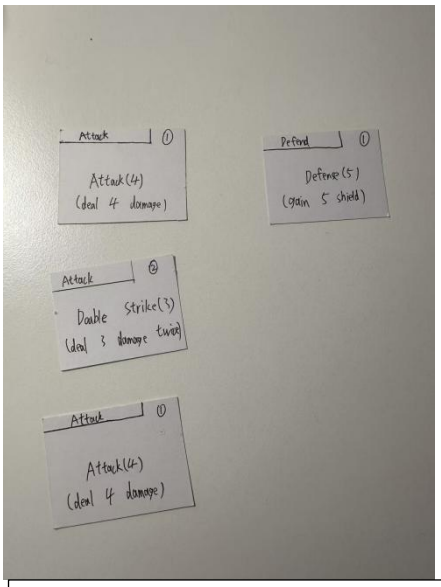


Figure11: example of another game scenarios . Shoot on 13/01/2026

In this situation, since the player on the right has only one line of Code Cards, after run the Line 1, the player on the left immediately proceeds to run Lines 2 and 3.

5. End stage

After both players finished their programs, enter the End stage. At this time, all players' Shield will be cleared. Then, any player in Overload suffers the corresponding penalties based on their current Overload Level. Once all effects have been resolved, the game proceeds to the Start stage of the next round.

End of Game

At any time, if a player's Energy Core is reduced to 0 or below, their opponent wins immediately. If both players' Energy Cores goes empty at the same time, the game ends in a draw.

Teacher Notes

Target Audience: Beginner to Intermediate Programming Learners

Tips: 1. Emphasize the core idea of capacity limits. Help students understand that computing resources have a limit, so they should optimize their code to use less space whenever possible.

2. Encourage students to plan the overall logic before modify their program. This helps students better understand how a program executes and plan their logic in advance, which is especially valuable when they later write larger programs.

Debrief Questions: After the game, you can guide reflection with the following questions:

1. What happens when the system overloads? How can you avoid overload?
2. What parts of your code could be improved to make it more efficient?

AI-generated Prompts

This is a character card for a board game. I want to enhance its appearance. The theme of this board game is about programming, so the style is cyberpunk-like.

Now generate this card. What I hope is that this card remains horizontal, but it can be a bit larger. The layout can follow the example of the character cards