Alicia Cyphers
Tim Meehan
Bill O'Toole

Overview Statement:

We envision a game that gets non-programmers and beginners familiar with and teaches them about (the theoretical and practical aspects of) object oriented programming. This includes teaching them how objects, methods, and loops work with gameplay that makes each player use their method cards and premade lines of code to steal RAM from either players or the computer in order to win the game.

Other programming games are either too simple, teaching the player only the importance of ordering statements/directions to get to a specified goal, or too complicated for beginners, requiring the player to already have knowledge and/or learn specific methods and classes for the game. Byte Fights aims to be fun for non-programmers, beginners, or even programming experts but still be able to teach players about object oriented programming and more general programming concepts if they don't have experience with it. As programmers, our team realizes the importance of looking at code and thinking about the concepts often to have it really stick. We chose real methods for the cards to get players familiar with real methods (which are only tweaked to work as a card game but otherwise the concepts are exactly the same) and chose to limit creating our own methods only for the purpose of the game (accumulating RAM).

Another skill that we're trying to reinforce is reading API's. The instructions are written in Java doc style. Players should become familiar with the game's API as they will have to when they start their programming career or classes. All the game instructions are in the game's API so by having instructions in API the player should become more familiar with API style and should be more comfortable reading API after multiple game playthroughs.

What makes Byte Fights unique is that it is competitive and easy to play without previous programming knowledge so it is less intimidating than other competitors.

Audience and Platform:

Byte Fights' focused audience is middle to high school students. These are students around the age where they would start to learn about programming in school (as well as how it plays a part in the internet and its impact on the world) and it is also a good age to get a start with programming if interested. Although, it could be argued that elementary students would be the best to teach programming since the critical period is between age 5 and puberty (according to the Critical Period Hypothesis), asking students to look, assemble, and understand even simple code may be a little too complicated before the age of 10 or 11.

Byte Fights is a tabletop card game. For a programming game for beginners, asking players to handwrite or type code seems too tedious and more like (school) work rather than something someone would do for fun. Deciding to use code snippets on each card allows players to look at their options, rather than trying to grasp method names, classes, ect from memory, and then rearrange them which is much less time and labor intensive than typing/writing code themselves.

Littlecodr is another game that tries reinforcing programming concepts such as creating a sequence of actions to get to a specified end goal. It uses direction cards as statements and the cards can be used to get to the goal position. This is too simple and teaches concepts too

simple for older students who have a more solid grasp of what programming is and is capable of. Screeps is another game that uses programming in the game itself, however, it uses its own methods and classes that the players have to learn before being able to play. The players have to already have some experience with programming and we would argue it also scares players away from having to read through all the methods and classes to understand what they can do, and they have to memorize it all to be able to implement the code to play the game itself.

Gameplay:

Core mechanics/gameplay loop

Draw the amount of cards according to the size of your hand. A player can play any or all of the cards they possess in their hand. Each card that is played is placed into the discard pile. After the player is done playing their cards, they draw the amount of cards needed to fill their specified hand size. The next player plays their turn. After the center deck is empty, the center deck is shuffled and placed back in the center as it was in the beginning. Once the RAM pool is empty the game ends.

Objectives/goals

Capture - steal RAM from the computer and other players. Deter other players and keep them from stealing or keeping RAM. Accumulate the most RAM by the end of the game to win.

Procedures

Players are dealt 5 cards at the beginning of game. Player that deals goes first. Player plays as many cards as they can/want before ending their turn. After ending their turn, the player can replenish their hand. The next player plays their turn. Players take turns until no more RAM is left to take from the computer.

Rules

1. Players must compete for RAM until the computer's RAM is completely used up.
2. Pick up cards after completing turn until the player reaches their hand size specified on their mat.
3. Player cannot shuffle their cards without playing the shuffle card.
4. Players can only have one for loop in play at a time.
5. Popped cards go on the bottom of the player's deck.
6. If you try to steal RAM from another player and they have less than that amount, you cannot go through with the action and must discard the played card(s).
7. If statements execute once, discard the if statement and the cards played with/inside the if statement after the statement is triggered.
8. Breaks only work on loops.

Resources

RAM is the resource players are striving to get. This is basically the units used in the point system. Cards are used to actually take actions.

Conflict

Other players can steal your RAM. Players can stop your loops. Players can make you skip a turn.

Boundaries

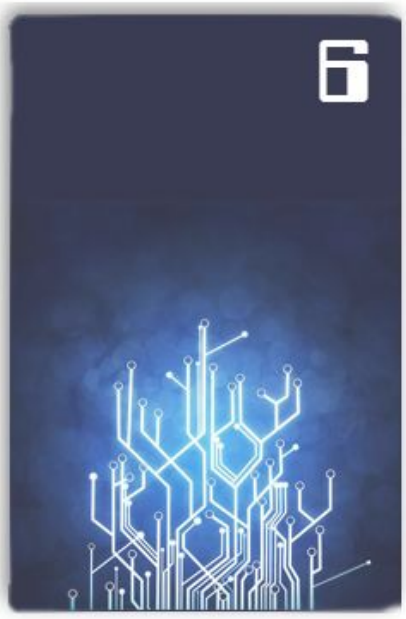
The magic circle is the table/area the game is played on/within.

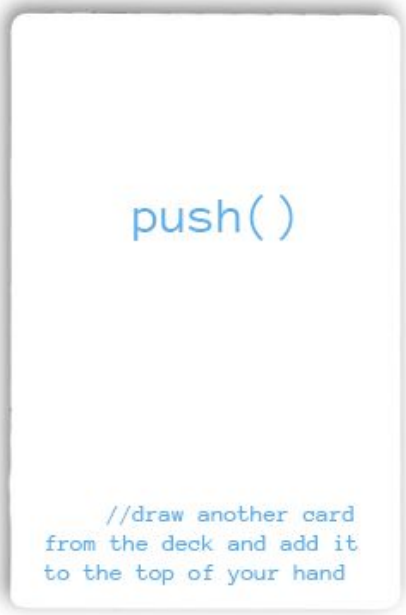
Outcomes

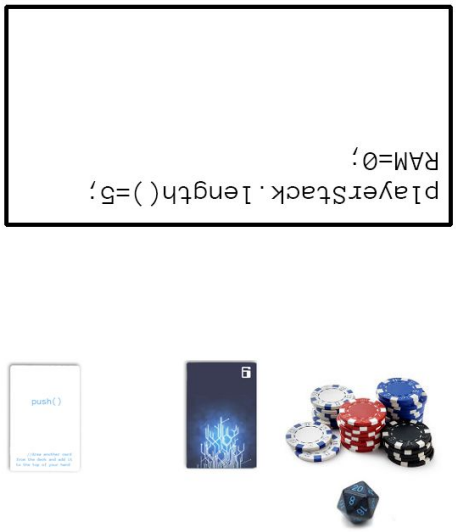
You win, lose, or (in rare cases) tie depending on who has the highest amount of RAM once the computer RAM supply is fully exhausted.

Card/Table Design:

Each card has the deck design on the back of the card and a number (which is used if the card asks for the card value). The front of the card has the code/statement in blue if it's a method and green if it's a full statement or loop. Blue and green were chosen because they're two colors often used as colors that represent technology. The deck design was chosen because it represents a circuit board pattern and the image isn't too distracting. Monotype was used for the font of the code and comments on the front of the card because it's easy to read and it also happens to be the font used by Eclipse (an open source IDE used frequently for Java programming).







The table is setup to have a draw and discard pile in the middle of the table. A 20 sided die and chips that represent the amount of RAM the computer has sit beside the draw and discard cards in the middle of the table. Each player has their own mat which displays their accumulated RAM and the current size of their hand (playerStack).

Story:

Because the game is designed to be a simple card game, it doesn't have much need for a story. However, we did come up with a simple concept story that the players are malware competing for a computer/server's resources. By stealing and using RAM, the malware try to use up all the RAM in the system so they can incapacitate the computer and cause the user distress as malware likes and is designed to do.

Research:

https://www.researchgate.net/publication/283637460_Learning_Practice_and_Theory_in_Programming_Education_Students'_Lived_Experience

As Anders Berglund and Anna Eckerdal of Uppsala University of Sweden write in their paper Learning Practice and Theory in Programming Education, "In learning to program, there is a complex interplay between the learning of practice and the learning of theory." (1)According to their experience, if one of these aspects is taught separately from the other it will lead to an inferior learning outcome, possibly even to fail.(1) Byte Fights teaches both aspects of

programming education by reinforcing the theoretical aspects of programming through the mechanics of the game and reinforcing practical aspects by including proper syntax on the playing cards.

Card Types

Deck manipulation

shuffle() - shuffle own deck

pop() - take card from the top of another player's deck
push() - draw another card from the draw deck
clear() - forces another player to discard entire deck
peek() - allows player to look at another player's top card
reverse() - reverses player turn order

Code Statements

For loop - get the number of turns it loops from the value on the back of the card
Break - can break for and while loops
While loop - runs until specified condition is broken
If statement - runs once statement condition is met
aPlayersStack = Arrays.copyOf(playerStack, length) - changes a player's hand to contain a specified number of cards which is retrieved from the card value
randomNum = rand.nextInt(20) - generates a random number between 0 and 20, roll the 20 sided die to determine randomNum
Malware.stealRAM(cardVal) - steals the number of RAM specified by the card value from another player
Malware.stealComputerRAM(5) - steal 5 units of computer RAM
Malware.getPlayerRAMVal() - gets the amount of RAM a player currently has