cardComposer: A Functional Programming Card Game

Maria Hwang
Fashion Institute of Technology
New York, NY, USA

Mark Santolucito
Barnard College, Columbia University
New York, NY, USA

ABSTRACT

We introduce a card game for teaching basic functional programming concepts - specifically maps and filters. The game uses a standard deck of playing cards and the underlying computational concepts can be introduced to students within a one-hour lecture period. We tested this game (informally) with CS-101 students and found it to be an engaging activity. We describe the complete set of instructions for the game and outline future directions of development.

CCS CONCEPTS

• **Applied computing → Interactive learning environments**.

KEYWORDS

computer science education, card game, functional programming

ACM Reference Format:
Maria Hwang and Mark Santolucito. 2021. cardComposer: A Functional Programming Card Game. In *26th ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2021), June 26-July 1, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3456565.3460029

1 INTRODUCTION

As functional programming has become more widely used, for example in the Map-Reduce framework, there have been efforts to use games to make functional programming more accessible to beginning computer science students [1, 3]. While promising, these games have remained in the digital domain and the use of manipulatives for teaching functional programming has remained underexplored. The value of manipulatives has been proven to be a useful resource for introductory computer science courses, for example with the "CS Unplugged" project [2]. To fill this gap, we introduce a card game that we have developed to teach the building blocks of functional programming language concepts.

Our game, *cardComposer*, can be played with a standard deck of playing cards, making it accessible to many different classroom environments. This game was inspired by the *cubeComposer* game [4], which also teaches functional programming using a physical analog (though is still a digital game). In *cubeComposer*, a player must solver a puzzle by constructing a functional program to manipulate cubes into a target configuration. In contrast, *cardComposer* is a two-player (or two-team) game that introduces an interactive and competitive component to the basic model of *cubeComposer*.

2 HOW TO PLAY THE GAME

The game is played in three rounds: the initial placement (Sec. 2.1), the code writing (Sec. 2.2), and the battle phase (Sec. 2.3). The players initially draw a hand of cards, then write a small functional program to manipulate the layout of that hand, then "battle" their hand against their opponent.

2.1 Dealing the Cards

To begin the game, each player draws eight cards, placing four in each row, alternating between face up and face down. The board should be arranged as shown in Fig. 1.

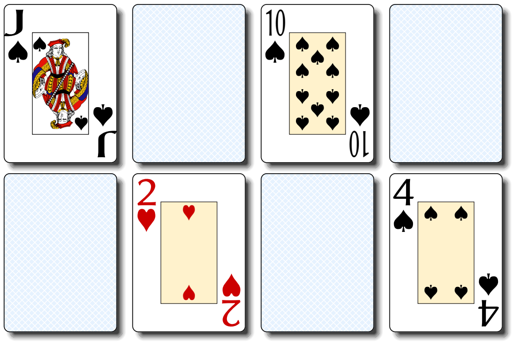


Figure 1: An initial starting draw example for one player

2.2 Applying Functions

Each player (or team) is given time to write down the code that they will use to rearrange the cards. Their code will be in a functional style - utilizing `map`s and `filter`s. The higher order functions (map and filter) *only* apply the first-order functions (those listed in Fig. 2 and Fig. 3) to the front row of cards. The exception is "`map swap`" which swap the front row and back row.

If a card is filtered out, all cards move to the left (your zero index of the list) to fill this hole. The code each player writes should be a composition of maps and filters.

Once each player has written down the set of functions they want to apply to their cards, the players show their code to their opponent. The players then walk though the application of the functions together. This is a useful step to check each players understanding of the functions.

ITiCSE 2021, June 26-July 1, 2021, Virtual Event, Germany
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8397-4/21/06.
https://doi.org/10.1145/3456565.3460029

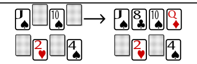
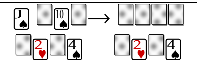
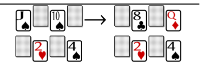
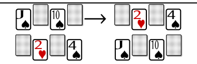
function f	description of f	example of `map f`
faceUp	turn all cards face up (regardless of their previous state)	
faceDown	turn all cards face down	
flipOver	flip the cards from face up to face down or vice versa	
swap	swap the front and back rows	

Figure 2: The functions to be used with `map`

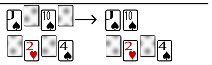
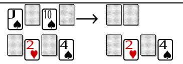
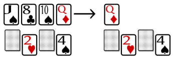
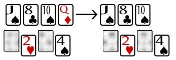
function f	description of f	example of `filter f`
isUp	returns True if the card is face up	
isDown	returns True if the card is face down	
isRed	returns True if the card is red AND face up	
isBlack	returns True if the card is black AND face up	

Figure 3: The functions to be used with `filter`

program	evaluation

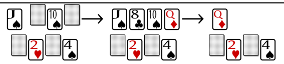
`filter isRed`
`(map faceUp hand)`
	

Figure 4: An example program applied to a hand.

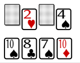
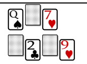
player 1 hand	
player 2 hand	

Figure 5: The battle stage - player 2 wins by one point.

2.3 Battle Phase

Once the cards have been rearranged according to the functions, the players can move to the battle phase. In this stage players compare the cards directly across from each other.

A high card wins by the point differential (cf. Fig. 5 where the Queen of Spades beats the Ten of Spades to earn player 2 two points). A face down card is in "defense" mode, and automatically results in a draw (cf. Fig. 5 where the player 2 defends the Eight of Clubs resulting in zero points for both players). If there is no card in the first row, the card in the back row is used in the comparison (cf Fig. 5 where the Nine of Hearts is compared to the Ten of Diamonds, earning player 1 one point). If there is no card in either row on a player's field, it is counted as zero, and any face up card in the opponent's field (directly across) wins by the value of that card. Sum the total of the points in each of the four comparisons to determine the winner of the round. A typical game should have three rounds.

3 DISCUSSION

This game was first tested in a classroom of approximately 40 CS-101 students at Bucknell University on Jan 19, 2020. We split the class into groups of four students so that two students could be on each opposing teams. Each group of four students can use half a deck of cards (well-mixed) instead of a full deck. The activity took approximately one hour including an explanation of the functional programming concepts.

An implementation of the game logic is available online at **https://github.com/santolucito/cardgame**. The code still lacks a complete front-end implementation, but we believe this game is more ideally realized as a physical learning activity with tangible cards. This game is also a good setting for which to use the playing card deck of Notable Women in Computing [5]. Although this is an activity that will need to wait for in-person teaching to resume, our hope is that this card game can be utilized as an engaging introduction when teaching basic functional programming concepts.

REFERENCES

[1] José Bacelar Almeida, Alcino Cunha, Nuno Macedo, Hugo Pacheco, and José Proença. 2018. Teaching how to program using automated assessment and functional glossy games (experience report). *Proceedings of the ACM on Programming Languages* 2, ICFP (2018), 1–17.

[2] Tim Bell, Ian Witten, and Michael Fellows. 2021. CS Unplugged. Retrieved March 7, 2021 from https://csunplugged.org/en/

[3] Marco T Morazán. 2013. Functional video games in CS1 III. In *International Symposium on Trends in Functional Programming*. Springer, 149–167.

[4] David Peter. 2021. Cube Composer. https://github.com/sharkdp/cube-composer.

[5] Susan Rodger, Katy Dickinson, and Jessica Goodman. 2021. Playing Card Deck of Notable Women in Computing. Retrieved March 7, 2021 from https://www2.cs.duke.edu/csed/wikipedia/cards.html