A Programming Board Game considered in Real Programming Language

Koki Watanabe[a], Shota Hayashi[a], Hayata Mori[b] and Koji Tajima*[c]

[a] NIT, Gifu college, Advanced Course, Gifu, Japan
[b] NIT, Gifu college, Dept. of Electronic Control Engineering, Gifu, Japan
[c] NIT, Gifu college, Electrical and Information Engineering Course, Gifu, Japan

Koji Tajima (ktajima@gifu-nct.ac.jp)

This paper describes a card-based board game developed for early elementary school students' programming education. The game aims to enable students to write in an actual programming language. The game has the following three features. (1) The game is designed for two players, and it makes players want to learn by competing. (2) The program can be developed simply by ordering the card. (3) The completed card program can be written in block programs, TypeScript, Java, Python, etc. We held a class for eight elementary school students to use the game we had created. We confirmed in this class the importance for elementary school students to understand the goal of developing program. The game has a simple, easy-to-understand goal. Creating a number larger than the opponent's by arranging the cards is the simple goal and it is for students to generate their own ideas. The implementation of their ideas helps them to improve their motivation to learn. Furthermore, we are developing a program that can automatically make the program by taking pictures of the cards of this game. We have now finished recognizing cards from images with high accuracy, and we plan to develop the code generation part in the future.

Keywords: *programming education, card game, gamification, image recognition*

Introduction

In 2020, Japan made programming education a requirement for elementary schools. We need to think about how to teach programming to elementary school students. There are two phases to obtaining programming skills. One is to learn to think logically, using logic such as addition and repetition to solve problems. Another is learning to write and describe languages such as C and Java. Programming tools for education, such as *Scratch* and *MakeCode*, focus mainly on the first. Block programming is used in these software. The problem was that while the students enjoyed creating logic, they could not develop an actual program.

This paper proposes a programming learning material that is easier to program than block programming, but is easier to convert to actual code. The proposed material is in the type of a card game, which will motivate students to learn by challenging them to play the game. On the other hand, this material is not designed for general-purpose programming as much as block programming. This material is designed to learn the logic focused on the task of making large numbers with some functions. In addition, this logic is easy to convert to the actual syntax of programming languages.

Materials of Education

1. Summary of the game developed

We developed a card-based board game for programming education for early elementary school students, which aims to enable them to write in an actual programming language. The game has the following three features. (1) The game is designed for two players, and it makes players want to learn by competing. (2) The program can be developed simply by ordering the cards. (3) The completed card program can be written in block programs, TypeScript, Java, Python, etc.

Fig.1 shows the cards for the games that we developed. In this game, the objective is to arrange the order of cards to build a larger number than the opponent's. Players play three matches with a pre-determined set of seven cards. The example in the figure shows a match with three cards. The bottom player makes 5 by +1 + (+2) x 2 times. In this game, red cards are for

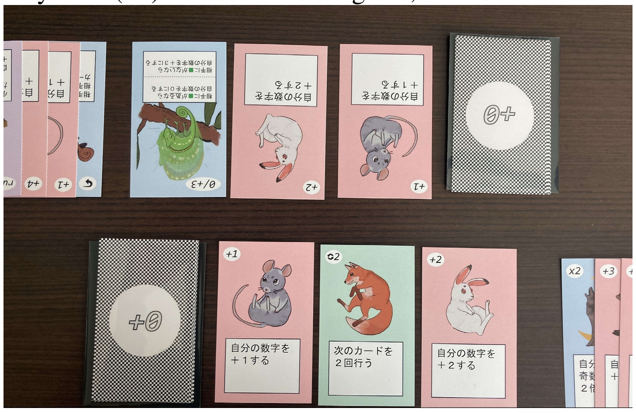

Fig.1 An example of the board game.

normal addition operations, and green cards are for repeats. On the other hand, the upper player makes 6 by +1 +2 +3. The value of this light blue card changes its effect depending on whether the opponent has a green card or not. In this example, the opponent has a green card, so it is +3. Therefore, the upper player wins this match. The difference between our card game and existing materials for learning programming is that we directly include a programming processing element and focus on numerical calculations based on the goal of creating larger numbers. There are many card-based programming games, such as *Robot Turtles* and *STEMON*, in which players line up instructions to move a specific object, leading the object to the goal along a designated path. These programming games do not focus on calculation directly, since the main point of these games is to be enjoy from pre-school age children. On the other hand, programming materials for post-school-aged children are often based on block programming. These games could be freely programmed, but had the disadvantage that they had no fixed objectives. It is an enjoyable learning experience for children with a clear objective, such as upper elementary school students. As a result, for younger children, the former had the disadvantage of being too easy and the latter too difficult. The proposed game we propose is positioned between these two types of games. It is simple in its operation of lining up cards, but it has a clear goal of calculating larger numbers. It also allows children who have just learned the four arithmetic operations to apply their knowledge to the game. Moreover, it can be considered as replacing the blocks in block programming with cards, and these cards can be converted into the actual programming language. In this way, this game can work as teaching materials between block programming and authentic programming language learning and can be used to help beginners in programming learn the basics of programming and a programming language.

2. The game cards and rules

Fig. 2 shows a summary of the cards used in this game. In this game, each player has seven of these cards. This combination of the cards is called the “deck”. We made 4 types of decks. Table 1 shows the cards in deck 1 and 4. Each deck has different characteristics, and the contents that can be learned are different. We adjusted any deck can win against all other decks with a good algorithm. Because we are targeting beginning students, we have decided to only focus on the simplest functions of programming at this time. In the future, we plan to increase the functions according to the level of learning.

There are three games, and the winner is the player who wins two games. The first and second matches are played with three cards chosen from seven cards; in the second match, the cards used in the first match cannot be used. The third match is played by selecting five cards from seven. In this match, all the cards can be used. This is strategic because a card unseen by the opponent can be used.

This game can be introduced to second-grade elementary school students because of the simple rule of making large numbers by calculation. Also, when creating a program in an actual programming language, there is no need for special functions, since it is sufficient to define variables and add numbers. In addition, some cards are an introduction to learning how to write looping (repeating) and conditionals. The color of the card is designed to identify its function of the card: addition is red, looping is green, and conditional is blue.

Table. 1 The card lint in the deck 1 and 4.

Card	function
Red 1	add 1
Red 2	add 2
Red 3	add 3
Red 4	add 4
Red 5	double
Green	Repeat 2 times
Blue 1	If your number is odd, double that number.
Blue 2	If opponent has a blue card, your number added 1 and double your number.
Blue 3	If opponent has a blue card, all other blue cards become zero, and your number added 1.
Blue 4	If opponent has a green card, all other green cards become zero, and your number added 1.
Blue 5	If opponent has a green card, your number added 3, otherwise zero
Blue 6	If opponent has a green card, the next card after that card is zero.
Purple	Change the rules for this match. The one with the lowest number wins.

3. The digitized method of the game

We considered a digitizing method to convert this card game into a computer game. Play the game with a computer, which could be automatically converted into a programming language, and it automatically calculates the execution results.

We consider the ability to freely remake the source code essential to understanding the algorithms of programming. For this reason, the code can change only by changing the order of the cards in the electronic

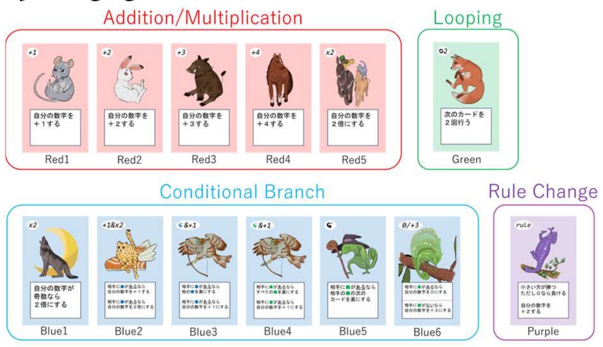

Fig.2 Card list of developed game.

version. When editing source code directly, this is difficult for beginning students, who are unfamiliar with keyboard usage, copying and pasting, etc.

There are two different methods to capture the sequence of cards into the electronic version. The first is to digitize the card game itself so that it can be played all on a computer. The other method is to play the game with cards, take a picture of the lined-up cards with a camera, and analyse this image. In this project, we tried to implement both methods.

The software, which can be played on a computer, was created using Unity. This software is designed to be played by two people on different computers and enable them to communicate with each other. The communication is carried out in real time. The library for this communication used PUN2. PUN2 can synchronize objects using the Cloud. In this case, the cards are kept as network objects on the server and synchronously placed as objects on both game screens.

The flow of the electronic version of the game is as follows. (1) Register the project of the game on the PUN2 server and output the project. (2) Two players run the output project. (3) The running projects automatically connect to the server and display each other's network objects and names. (4) Once the two players are connected, the card selection screen appears. (5) After both players have finished selecting their cards, the result of the match is displayed. As there are three matches, the card selection and display of the result are carried out 3 times.

Fig. 3 shows the card selection screen. This screen places the cards that can be selected in the current match. This screen is not synchronized over the network but locally, as card selection is not done openly to the opponents. The cards can be selected by mouse-clicking on the object. When a card is clicked, the selected card number is displayed in the "Selected Cards" section at the

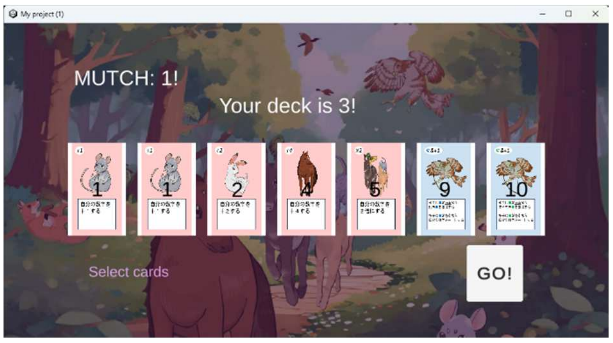

Fig. 3 The card selection screen of the game.

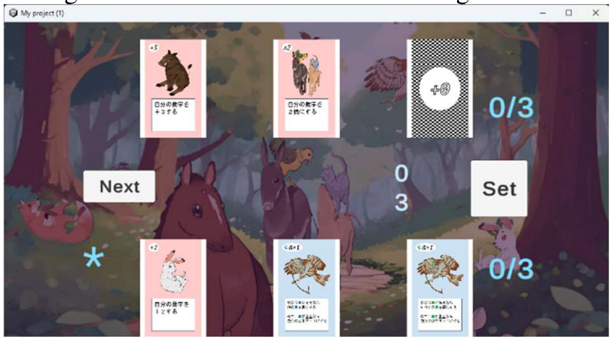

Fig. 4 The match-up screen of the game.

bottom of the screen. To deselect a card, simply click on it again. The newly selected card is added to the last of the cards list already selected. When the player finishes selecting the cards, pressing the "GO!" button allows. Press this button, the application moves on to the match-up screen.

Fig. 4 shows the match-up screen. When the screen moves to the match screen, the player is connected to the network again. Two players are connected to the network and press the 'Set' button, both players' cards will open. An asterisk is displayed on its side to make it easier to identify. The player processes the cards in order from left to right by pressing the space key, and the calculation results are displayed. Processes such as turning over an opponent's cards are done before this calculation. Wins and losses are stored locally, and the end result is displayed after three matches.

Method of Experimental

For the evaluation experiment, we tested the understanding test in an IT course for elementary school children and checked the accuracy of the image recognition. The reason for experimenting with image recognition is that we plan to develop a software system that generates a program corresponding to the card contents and calculates the score automatically by photo images of the board. This system shows how to write code; it helps beginners in programming to understand how to express corresponding rules of the game. Students could develop logical thinking and programming skills, in this process.

(1) The IT learning class for primary school

This game was used in a local IT class, and we confirmed the effect of this game for elementary school students. We invited elementary school students to participate in this course. As a result, eight students attended the learning class: one 4th year student, three 5th year students, three 6th year students in elementary school and one 1st year junior high school student. The title of this class is "Learning Programming and IoT via competitive games" ("対戦型ゲームで学ぶプログラミングと IoT" in Japanese). After learning about programming development using our proposed card game, the students will develop a programme to obtain temperature sensor and share the temperature using the IoT device Obniz. The value obtained from the sensor is 0 to 1,024. It cannot be used as temperature as it is, so the programming technique learned in the card game is used to convert the value obtained from the sensor to temperature there. At the end of the course, we surveyed the level of enjoyment and programming difficulty by questionnaire. Free-text answers were also collected to obtain information on what was interesting and difficult.

(2) The Image recognition accuracy

In the image recognition experiment, the cards are lined up and captured by a camera. We confirm the accuracy with which the cards can be recognised from the images captured. The cards for identification in this case are the 13 types of cards shown in Fig. 2. This

Table. 2 Accuracy of feature matching

Card	Number of trials	Correct	Accuracy
Red 1	39	39	100%
Red 2	26	26	100%
Red 3	28	28	100%
Red 4	30	30	100%
Red 5	42	42	100%
Green	30	30	100%
Blue 1	15	15	100%
Blue 2	15	15	100%
Blue 3	15	14	93.3%
Blue 4	15	14	93.3%
Blue 5	15	15	100%
Blue 6	15	15	100%
Purple	15	15	100%

time, the cards were printed from images created on a PC, so that all accurate comparison images existed. We performed card identification by feature matching between the comparison image and the captured image.

It is necessary to cut out the cards from the line-up into individual card images. The cut-out position was fixed for this by placing the cards on the play mat.

For each card comparison image and cut-out image, we extract local features using the SIFT algorithm. For matching between the extracted features, we used FLANN. For feature matching, the cut-out images are matched with all comparison images. Then we identified the image with the highest degree of matching. The comparison image with the highest degree of agreement is then selected as the recognition result. We prepared 30 images like Fig. 1 for the evaluation experiment; 10 cards were arranged in each image, so the total number of cards was 300.

Results and Discussion

In a questionnaire-based evaluation, all students selected 5 out of 5 for enjoyment, which indicates that they enjoyed learning programming using games. As for the level of difficulty, four students answered easy, three answered standard and one answered difficult. In the free-text comments, the students again noted that the game helped them learn in an enjoyable way, and also wanted to develop a more complex programme with a programming language. For the electronic application, the students tried the paper-based card game and understood the rules before using it. There were no problems of difficulty in operating this application if the students had understood the game rules before using it. Also, we observed that many students showed a strong interest in the way the game's controlling code works. However, since we could not collect enough younger elementary school students, we are planning to try the experiment again at a different location.

The results of the image recognition experiment are shown in Table 2. The mouse card is Red 1, and the rabbit card is Red 2. Out of the 30 images prepared for the experiment, the number of trials for Red 1 and Red 5 are over 30, This is because two of the same cards can appear in one deck.

The results show that more than 90% of all cards were identified, although Blue 3 and Blue 4 were misidentified once each. This is due to the very similar images and the light conditions that made it difficult to see some of the images.

Conclusions

This paper describes a programming education material for early elementary school students. This material is a card-based board game that line up cards with a process written on them and make a number greater than their opponent's. There are 13 types of processes, representing addition, multiplication, looping, and conditionals. A deck is a set made from seven of these cards. The player selects from 3 or 5 cards from the deck to make the program with their own ideas.

We used in a local IT class, and we confirmed the effect of this game for elementary school students. At the end of the class, we surveyed the level of enjoyment and programming difficulty by questionnaire. The results shows all students enjoyed learning programming using games. As for the level of difficulty, half of the students responded that it was easy. The age of the elementary school students who attended this class was a little older than the estimated age. Therefore, we will design that allows for changing the difficulty level.

On the other hand, we also tried to digitalize this game and to recognize the card images by photo. The digitized software was also used in the course. The application was easy to operate for elementary school students. The recognition of cards was accurate, but in this card set, there were two cards that were only different colors, and there was a case of mistaking these two cards.

In the future, we plan to create a system to automatically generate a program based on the recognized cards and their order, and hold the class for elementary school students again.

References

Mitchel, R., John, M., Andrés, M. H., Natalie, R., Evelyn, E., Karen, B., Amon, M., Eric, R., Jay, S., Brian, S., Yasmin, K. (2009). *Scratch: programming for all.* Communications of the ACM, Volume 52, Issue 11, pp 60 – 67.

Lowe, David G. (1999). *Object recognition from local scale-invariant features*. Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157.

Marius, M. and David G. Lowe (2009). *Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration.* International Conference on Computer Vision Theory and Application VISSAPP'09, pp. 331-340.