Abstract

Visual programming can be cited as a learning support for programming beginners. Visual programming includes block-based programming, in which programming is done by combining puzzles and blocks, flow-based programming, in which programming is performed by connecting the nodes of each program, and a unique rule type performs programming using a method different from the type. All three have in common that they are visually easy to understand, easy to operate, less likely to cause errors, and allow learners to select from a set of predetermined code fragments.

On the other hand, examples of text-based programming required in an increasingly information-oriented society include Javascript, Python, Java, C#, and PHP. By learning these skills, learners will be able to handle the front-end to back-end processes in system development, and the demand for human resources in the information society will increase. However, while learning text-based programming allows them to utilize data effectively, it requires writing from scratch and error handling and is characterized by high learning costs in terms of time, teaching materials, and environmental construction.

Although Scratch has been successful as a visual programming tool for elementary school students, there is a gap in connecting it to text programming. Although it is not known exactly where the gap is, it is thought that a major reason for the gap is that it is impossible to convert to text type, and it is difficult to visualize the transition from visual type to text type.

Google Blockly, a block-based visual programming language, is an example of a language compatible with visual and text types. Google Blockly allows learners to convert the created visual program into a text program (JavaScript, Python, PHP, Lua, Dart) and run it. This feature makes it easy for the image of text-based programming to become firmly established, but there is an issue in which beginners who have completed visual programming may not be able to confirm the visual representation during text-based programming. To solve this problem, it is necessary to further solidify the connection between the images of visual and text programs by implementing conversion from text programs to visual programs.

In this study, we focused on the problem that it is difficult for learners who have finished visual programming to transfer to text programming, and in order to solve this problem, in addition to converting existing visual programming to text programming, we Implement the conversion function from a type program to a visual type program. By doing so, we will promote the learning of text-based programming while taking advantage of the characteristics of visual programming, which is easy to see and less likely to cause errors.

To achieve this, we will build an environment that makes it easy to compare visual and text-based programming, implement mutual conversion between visual and text-based

programs, and conduct experiments to demonstrate the usefulness of the developed support system.

We mainly used Google Blockly to build the visual programming environment. The reason is that it is compatible with text-based programs and that learners can add original visual-based programs.

MonacoEditor was used as an editor to construct the text-based programming environment. The reason was that it was necessary to display syntax errors in the program to the learner. The programming language in MonacoEditor was JavaScript. The reason is that it is one of the programming languages compatible with Google Blockly that can be easily executed on a browser.

To convert from a visual type to a text type program, we used the functions of Google Blockly. In Google Blockly, each visual program has a built-in text program, and the built-in programs are combined through connections between visual programs, ultimately completing the entire text program. Using this function, we converted a visual program to a text program.

AST(AbstractSyntaxTree)is a declaration statement, a statement (Statement), an expression (Expression), an operator, an identifier (Identifier), a value (Literal), and a block statement (BlockStatament) that is the body of if, for, function, etc.) is the tree structure data of the program. The text-based program was converted to a visual-based program based on this AST information.

We built an environment that displays visual and text environments within the same interface and allows programming. A visual environment is built on the left half, a text environment is built on the right half, and a conversion button is placed between them, making it easy to compare. Additionally, the system runs on a browser and can be executed, converted, saved, and loaded with the click of a button.

Using the developed system, we conducted experiments with graduate students who do not specialize in information systems. The subjects learned using the mutual conversion function in one of the two learning processes. A mastery comprehension test was conducted before and after learning to measure the subjects' level of understanding of text-based programming, and a questionnaire was administered before and after the experiment. We also collected programming learning logs during the experiment and obtained the error rates of program execution and conversion. In addition, we conducted an experiment on the accuracy of conversion from text type to visual type using reference books.

As a result of the mastery level comprehension test, we were able to show that learning using the system is significant in text-based programming learning with a p-value = 0.0009119, but a comparison between with and without mutual conversion shows that the p-value =0.89, the result is not significant. Errors occurred with a probability of

approximately 8% when running text-based programs, resulting in many definition errors. The conversion accuracy was poor for function call statements in text-based programs.

There are still areas for improvement in this system. In order to make it easier to compare visual and text types, the colors of characters and blocks in both types should be unified, and the number of programming languages that the system can support should also be increased. Also, since only certain visual type programs can be converted from text type programs, it is necessary to implement a function that automatically generates visual type programs from text type programs or the AST of text type programs.