UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

**Brigitta Ambur**

# Syntax Bee: An Interactive Prototype for Investigating the Impact of Code Visualisation on the Motivation in Learning Programming

**Bachelor's Thesis (9 ECTS)**

Supervisor:
Velvet Spors, PhD

Tartu 2025

# Syntax Bee: An Interactive Prototype for Investigating the Impact of Code Visualisation on the Motivation in Learning Programming

**Abstract:**

The purpose of this thesis was to create an interactive prototype to investigate how code visualisation affects students' intrinsic motivation while learning programming. The theoretical part discusses learning to program, motivation in learning to program, visualisation in education, and code visualisation. The practical part consists of interviews and a quasi-experiment. The interviews focused on participants' experience with learning programming. The experiment involved creating an interactive prototype based on the interview results, testing the prototype, and analysing the results. Motivation was measured before and after testing the prototype by using adapted version of the Intrinsic Motivation Inventory. The results showed that intrinsic motivation increased among participants who used the visualised prototype, whereas motivation decreased in the control group.

**Keywords:** learning programming, prototype, motivation

**CERCS:** S281 computer-assisted education

# Syntax Bee: interaktiivne prototüüp koodivisualiseerimise mõju uurimiseks programmeerimisõppe motivatsioonile

**Lühikokkuvõte:**

Lõputöö eesmärk oli luua interaktiivne prototüüp, et uurida, kuidas koodivisualiseerimine mõjutab tudengite sisemist motivatsiooni programmeerimise õppimisel. Teooriaosas käsitletakse programmeerimise õppimist, programmeerimisõppe motivatsiooni, visuaalset õppimist ning koodivisualiseerimist. Praktiline osa koosneb intervjuudest ja kvaasi-eksperimendist. Intervjuud keskendusid osalejate programmeerimisõppe kogemusele. Eksperiment hõlmas intervjuude vastuste põhjal interaktiivse prototüübi loomist, prototüübi testimist ning tulemuste analüüsi. Motivatsiooni mõõdeti enne ja pärast prototüübi testimist, kasutades kohandatud sisemise motivatsiooni küsimustikku. Tulemused näitasid, et visualiseeritud prototüüpi testinud osalejate sisemine motivatsioon kasvas, kontrollgruppi kuulunud osalejate motivatsioon kahanes.

**Võtmesõnad:** programmeerimise õppimine, prototüüp, motivatsioon

**CERCS:** S281 arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

# Table of Contents

# Introduction

Learning programming is considered difficult for beginners because understanding programming language structure and how a program is executed can be challenging [1]. Each language has its own unique logic, syntax and implementation methods. Struggling with comprehending new programming languages can lead to frustration and lack of motivation [2]. Maintaining motivation is an important aspect of learning, especially learning to program. Higher motivation leads to higher success rate and better understanding of programming concepts [3].

Many approaches and tools have been developed to enhance learning programming. One such approach is code visualisation. This refers to explaining programming concepts by using visual representation such as illustrations and diagrams [4]. Using visualisation to explain programming concepts could simplify the learning process [5]. Since learning to program relies heavily on understanding how a program works, this paper focused on describing coding concepts through walkthroughs and syntax explanations.

The goal of this thesis was to investigate how code visualisation impacts students' intrinsic motivation while learning to program. The research consists of two studies: interviews and quasi-experiment. Firstly, interviews were conducted to gather information about students' experience, problems and motivators while learning programming. The results were then analysed and prioritised. The second part of the research focused on creating and conducting the experiment. This included creating an interactive prototype based on the results of the interviews, testing the prototype on computer science students and measuring the participants' intrinsic motivation before and after the testing using the Intrinsic Motivation Inventory, and analysing the results.

The assistance of ChatGPT[1] by OpenAI was used for grammar checking, formatting and structuring some text sections. In addition, ChatGPT was utilised for troubleshooting Figma workflow problems and correcting or simplifying the code snippets used in the prototype. No participant data was processed by AI.

[1] https://chat.openai.com/ (version 4o)

# 1. Theoretical Background

## 1.1 Learning to Program

Learning to program is often described as challenging. According to Mohorovičić and Strčić [1] the concepts can be difficult to understand, and this leads to high dropout and failure rates. The authors bring out that programming is more than just understanding the syntax of a language – it also requires using cognitive skills. Calderon et al. [2] explain that besides knowing how to write a code, learning to program also requires understanding the problem and finding ways to solve it.

There are many methods of teaching and learning programming languages. Many higher education institutions use traditional approaches for teaching programming languages [2]. Zhang et al. [3] categorise traditional approaches as passive learning methods since students do not engage with understanding the concepts on a deeper level. Examples of traditional methods include discussion, where students can interact with each other by discussing questions, and demonstration, which is used for explaining the step-by-step process of how to do something.

Active learning is another approach for learning programming. Active learning integrates creative methods for teaching programming. Calderon et al. [2] bring out that some of the most used Active Learning Methodologies (ALMs) are Flipped Classroom and Gamification-Based Learning. Other innovative methods also include problem-based learning, puzzle-based learning, pair programming and peer tutoring, [1, 3]. Mixing multiple approaches is found beneficial since different methods complement each other [2].

AI code generation tools are changing how programming is taught [6]. Students can generate code with AI instead of writing it by themselves. These tools help students understand coding by providing multiple solutions, evaluating their code, explaining errors and getting started with coding. However, Becker et al. [6] explain that using AI can also make students dependant on it since they do not have to think by themselves.

### 1.1.1 Motivation in Learning to Program

According to Gomes et al. [7] motivation is a crucial aspect of learning process as it takes commitment and effort to reach success. The authors add that learning to program can be challenging for beginners since getting a code to work may take multiple attempts. Therefore,

motivation is essential for students to persist in learning to program. Higher motivation also increases performance and engagement in a subject such as programming [3]. For example, Gomes et al. [7] found that students who were more motivated received higher marks in assessments.

There are many factors that can increase motivation while learning to program. Calderon et al. [2] explain how using traditional methods in teaching programming concepts may cause loss of interest whereas using active learning approach has been found to improve students' performance and increase their motivation. Mohorovičić and Strčić [1] found that students' motivation depends on the value they see in a task and whether they believe they can succeed. The research added that choosing a form of motivation is individually different for each person and can therefore impact the outcome of the learning process.

Internal, external and social goals are classified as separate motivation types which affect learning efficiency [1]. Social motivation can be described as pleasing others, intrinsic motivation refers to how students value the task at hand with goals such as mastery or curiosity, and extrinsic motivation relates to motives like grades, rewards or competitiveness [8]. According to the study conducted by Gomes et al. [7], intrinsic motivation was the most common type of motivation in learning programming as it was chosen by half of the sample. The study also found that students who were intrinsically motivated received better results.

Intrinsic motivation can be evaluated using the Intrinsic Motivation Inventory (IMI). The Center of Self-Determination Theory explains that IMI is used for measuring participants' subjective experience and motivation during some activity or experiment [9]. It includes seven subscales: interest/enjoyment, perceived competence, effort/importance, value/usefulness, pressure/tension, perceived choice and relatedness. The Center states that each subscale varies in the number of statements it consists of, and which can be modified to fit the experiment at hand. Answering to IMI is based on a 7-point Likert scale. The Likert scale is an attitude measurement scale that provides feedback to statements using choices from strongly disagree to strongly agree [10].

## 1.2 Visualisation in Education

Visualisation in education means using visual elements to make knowledge more understandable [11]. It makes abstract concepts comprehensible by conceptualising them in visual forms [12]. Using visualisation also helps students develop critical thinking skills, simplifies understanding new materials, and enhances students' performance in assessments

[13]. Besides better understanding, another important aspect of visual learning is how it increases students' motivation and interest in the learning process.

Visual learning can include drawings, videos, charts, simulations – tools which help to explain something through visual representation [12]. To make visualisation more effective for learning, the visuals often follow principles such as clarity, simplicity, use of colours and displaying only the essential information [11]. Although most of the materials used in visualising concepts are made by designers or teachers, visualisation is found more useful when students participate in creating their own learning materials [11].

### 1.2.1 Code Visualisation

Code visualisation refers to using graphical representations to illustrate how a program works [5]. Supposedly, it simplifies learning and understanding programming concepts by displaying code workflow [5]. In other words, code visualisation means translating textual code into pictorial representation to make the structure and behaviour of programs more comprehensible [4].

Common code visualisations are diagram-based [4]. For example, there are UML diagrams, which use Unified Modeling Language (UML) to simplify visualising code architecture and program workflows [14]. Flowcharts are another example of diagram-based visualisation tools which involve using universal symbols and shapes to represent processes or algorithms [15]. Dependency graphs help display the dependencies between different classes, modules or components of a program [4].

Other tools and features for visualising code include syntax highlighting, debugging and stepping, live coding, and analogies. Syntax highlighting is a feature that displays code elements like keywords or variables in different colours [4]. This makes understanding the code easier and can also help with finding errors in the code. Debugging means going step-by-step through a code to identify and fix any bugs within the code [16]. Stepping is a method of debugging which refers to a walkthrough of the program execution to see what each code line does [17]. Live coding provides real-time visual feedback on code changes by updating the program output immediately [18]. Analogies can be used to make abstract concepts more understandable by explaining them through real world references [19].

Code visualisation can have positive effects on learning and its outcomes. For example, Dina et al. [20] found that students who had used education tools with visual representation received

higher grades compared to those who had not used visual aid, and using visualisation tools made learning coding more interesting and understandable. An experiment conducted by Olsson et al. [5] found that visualisation of loops highly improved students' understanding of the topic. Mladenović et al. [21] conducted an experiment that indicated that having visuals improved immediate understanding of programming concepts. Lian et al. [19] found that using analogies to explain Object-Oriented Programming (OOP) concepts enhanced comprehension and minimised confusion surrounding complex topics. Using walkthroughs and tracing program execution can simplify finding and fixing errors within a code [4].

# 2. Undertaken Studies

To investigate how visual representation affects students' intrinsic motivation while learning and solving programming tasks, two studies were conducted – interviews and visualisation experiment. This chapter gives an introductory overview of both studies.

## 2.1 Study 1: Interview

The first study focused on interviewing people who had previous experience in programming. The interviews collected data on participants' programming backgrounds, what problems they faced when learning to program, what motivated them while learning coding, and their thoughts on how to make programming education more engaging. Interview results were analysed and prioritised to identify the features that the interactive prototype should include.

## 2.2 Study 2: Visualisation Experiment

The second study was a quasi-experiment for testing how visualisation affects students' intrinsic motivation while learning programming. The experiment included creating an interactive prototype for a Java programming language quiz based on the results of the interviews. The quiz had two versions for each of the experiment groups – plain version for the control group and visualised version for the intervention group. Participants' intrinsic motivation was collected before and after solving the quiz prototype. Experiment results were analysed then analysed to evaluate how code visualisation impacted the motivation.

# 3. Study 1: Interview

The first study concentrated on interviewing people with previous experience in programming. The results were used for designing a prototype for the visualisation experiment. This chapter explains the methodology of creating and analysing the interviews, describes how the interviews were conducted, and gives an overview of the findings.

## 3.1 Methodology

This chapter gives an overview of the methods that were used in the first study. The methodology describes the interview type and creation process, participant selection, and how the interview results were analysed.

### 3.1.1 Semi-Structured Interviews

Interviews were designed to follow a semi-structured interview type. Semi-structured interview combines elements from both structured and unstructured interviews [22]. This type of interviewing consists of a previously chosen focus to explore but also enables the conversation to flow freely. More precisely, this means that the researcher has prepared an interview guide to cover a certain topic, but the questions do not need to follow a specific order, and the format allows expanding the answers by asking spontaneous follow-up questions. Karatsareas [22] describes that the questions in a semi-structured interview should be open-ended to encourage participants to express their personal experience and ideas.

Semi-structured interview type allowed participants as the potential end users to reflect on their point of view – for example, their thoughts and pain points while learning programming. The interview consisted of 13 questions which were divided into two sections. First 4 questions were about participant's previous experience in programming. The 9 remaining questions concentrated on participant's learning preferences. The format also allowed asking follow-up questions when elaboration was necessary.

The questions were open-ended and devised in a way that they would cover related background information, problems and motivators while learning programming, ideas on how to improve teaching a programming language. At first, the thesis was supposed to concentrate on game-based learning in addition to visualisation but later the focus shifted to how visualisation affects the motivation while learning new programming concepts. Therefore, some questions that were asked in the interview were more inclined towards gameful activities and learning.

### 3.1.2 Participants

Participants were chosen through convenience sampling. Even though convenience sampling was used, the author was keen on including a diverse group of people – the sample reflected different programming backgrounds and education levels.

There were five participants. Each of the interview participants were given pseudonyms as Int Number based on the order of the interviews, therefore Int 1, Int 2, Int 3, Int 4 and Int 5.

All participants had some previous experience in programming. There were four female participants and one male participant, all of them between ages 20-25. They were university students and had between 3 to 6 years of programming experience. Every participant had studied programming as part of their curriculum. Three of the participants, Int 1, Int 4 and Int 5, had taken programming courses in high school as well. Int 3 had taken internet courses in middle school to broaden their knowledge. All the participants were familiar with Python, and all except Int 3 had learned Java as well. Besides that, participants also mentioned their experience in C++, R, JavaScript and SQL.

### 3.1.3 Thematic Analysis

After all the interviews were completed, they were analysed using thematic analysis. Thematic analysis is a qualitative data analysis method used for identifying recurring patterns within a text. It uses a systematic approach, starting with coding the data and later organising it for identifying and interpreting larger themes [23].

Each interview was transcribed and reviewed, then each interview's main thoughts were summarised in four categories: previous experience, pain points, supporting factors and what additional tools/methods they had used. Then, the author sorted out common ideas and grouped them thematically.

After the themes were analysed and categorised, they were prioritised. The prioritisation was based on three criteria: how frequently was the theme mentioned, how practical it would be to implement, how beneficial it would be for teaching programming.

## 3.2 Interview Setup

The focus of the interviews was to find out participants' learning preferences regarding programming language studies – the problems they have faced and what has helped them understand new programming concepts. An interview guide (see Appendix I. Interview Guide)

was composed for managing the main themes and questions that were essential for the interview.

Five semi-structured interviews were conducted with people who had previous experience in programming. The interviews were conducted individually via in-person meetings or video calls if one-on-one meeting was not possible. Each interview lasted between 20-40 minutes.

The interviews were audio recorded, and later transcriptions were made from the recordings. Participants were informed of participation terms before attending the interviews. They received project information (see Appendix II. Interview Project Information), privacy notice, and consent form (see Appendix III. Interview Consent Form) ensuring that they understand the terms and give their permission to use the collected data for this bachelor's thesis.

## 3.3 Interview Findings

Firstly, problems of learning new programming concepts were discussed. Four of the interviewees, Int 1, 2, 3 and 5, described their difficulties with beginning to write code. The main concern was comprehending the logic and structure behind a language, and what should be the first step when starting to write a program. As a continuation to the previous thought, Int 2 and Int 3 brought out the struggle of understanding what should be the order of steps to complete a task.

When talking about specific programming concepts which are difficult, Int 2 and Int 4 brought out recursion. Int 2 also mentioned data sorting algorithms, tree structure and loops as struggles. In addition, Int 5 brought out not knowing the meaning of code keywords like “public” or “static” and built-in methods which make programming easier but are not familiar to most people.

Besides previous problems interviewees 2, 4 and 5 found that getting stuck is a big problem since it may become quite frustrating when they do not know how move forward and it makes them want to give up on figuring out the code on their own. This problem matches with another thought that was discussed: not understanding where the mistake was made and spending a lot of time trying to figure out why something does not work.

Next, the factors that support learning programming were discussed. Every participant mentioned the value of having visual explanations. The aid used for visual representation varied within participants. For example, Int 1 had searched for illustrative examples of how different JavaScript array methods like push() or pop() worked. Int 3, 4 and 5 found that

watching explanatory videos made understanding how code works easier. Int 2 said they use debug on a regular basis for actually seeing how a program works.

When talking about motivation and what could help with comprehending programming concepts better, there were many discussion points. Int 2 mentioned that an explanation what each code line does would make understanding everything a lot easier. Interviewees 3 and 5 added that hints and guidance when getting stuck is something they would prefer to have while learning programming. All the participants noted that what motivates them is seeing the output and effect of their program and the feeling of accomplishment that accompanies a successful effort. Interviewees 3 and 5 explained that they prefer tasks having real-world relevance or engaging narrative involved. Besides that, Int 3 and 4 brought up that they would like to be able to choose between different topics and select based on their preferences. When talking about gameful activities, all except Int 1 found that having a competitive element, collecting points in a leaderboard or keeping streaks would motivate them to continue playing.

Gradual increase in difficulty was mentioned by Int 1 and Int 4. Starting with more simple tasks and advancing to more complex tasks, helps the users to feel and see the progress of their skills and offers feedback of their level of competence. Int 2, Int 3 and Int 5 preferred dividing a task into smaller steps rather than having to write one big program all at once. Lastly, the same interviewees mentioned that having a visually pleasing and engaging interface with colours and illustrations makes solving tasks more pleasing for them.

### 3.3.1 Prioritising Interview Findings

Interview findings were used in designing an interactive prototype. Here is an overview of which features were implemented in the prototype, and which were left for future development.

As all the participants mentioned some variation of visualisation and including visual representation has been the main focus of this thesis from the beginning, visual demonstration is included in the prototype as illustrations explaining code segments and concepts. The illustrations were displayed as clickable overlays so that every user could proceed through the visualisation at their own pace. Understanding what each code line does was mentioned in the second interview. To combine this idea with visual representation, illustrations were created so, that each row which has some effect within a code segment is explained.

To include the feeling of progress and accomplishment, prototype exercises were gradually changing in difficulty. Prototype was created so that it would start with simpler exercises and

go progressively harder as user continues solving tasks. When user chooses an answer, an explanation of the answer is displayed. In addition, the prototype included step-by-step approach to support understanding the structure and logic behind a code. This means taking a bigger exercise and breaking it down into smaller steps, which was done with questions 5-8.

While many good ideas emerged from the interviews, not all were implemented for this thesis. Instead, the following ideas could be used for future development. Participants mentioned game features such as leaderboards and keeping streaks which would motivate users with competitiveness. Besides that, interviewees described that having a possibility to see hints would make solving tasks easier and less frustrating. Lastly, thematic selection was mentioned since it allows personalised learning. All these features would be valuable additions to the tool since they could have a motivational effect on learning programming languages.

# 4. Study 2: Visualisation Experiment

After interviewing people and analysing their answers, the next part of the research was to conduct an experiment. The purpose of the experiment was to find out if and how visual representation motivates people who are learning programming concepts. The chapter includes the methods used for designing and analysing the experiment, how the prototype was created, how the experiment was carried out, and the findings.

## 4.1 Methodology

This chapter describes the methodology used in the second study. Subchapters give an overview of prototype development, what research type was used for the experiment, participant selection, and how the collected data was analysed.

### 4.1.1 Prototyping

The prototype was based on the Java programming language. IBM [24] describes Java as an object-oriented programming language. According to the source, it can be run anywhere, and its syntax is derived from C and C++. In software development Java is one of the most popular languages used [24]. Java was chosen because of its popularity and the fact that the Tartu University computer science curriculum is mostly based on Java language skills.

The quiz was developed using Figma prototyping tool. Staiano [25] describes Figma in his book as a cloud-based design tool used for creating and developing interactive wireframes and prototypes. One of its main features is real-time collaboration allowing multiple people to work on the same project simultaneously. Besides that, the book points out that Figma also provides a platform for creating design elements, vector graphics and plugins which can be shared via Figma Community for others to access. Since the platform is quite easy to use, allows creating more complex design flows and the author has previous experience in using it, Figma was chosen as the tool to create the prototype for this thesis.

### 4.1.2 Quasi-Experimental Research

Experiment followed quasi-experimental research design. Gopalan et al. [26] describe quasi-experimental research as means of evaluating the cause and effect of an intervention. In other words, it focuses on how a factor, or its absence affects the results of an experiment. To understand and compare the differences between having or lacking a certain factor, test subjects are assigned to control and intervention groups [26]. All that is very similar to true experiments,

but what distinguishes quasi-experimental research, is not having complete randomisation when assigning the groups [27]. Instead, participants are typically divided into groups through self-selection, or they are assigned by the researcher. Lam and Wolfe [27] explain that this sort of assignment can reduce the reliability of the results since unpredicted variables that influence either one of the groups may also affect the outcome.

To analyse participants' motivation before and after taking the quiz, the Intrinsic Motivation Inventory (IMI) was used. For this thesis, the inventory was narrowed down to 8 questions from the following subscales: interest/enjoyment, perceived competence, pressure/tension and value/usefulness. Pre- and post-tests consisted of the same questions, modified accordingly. When conducting experiment with intervention group, they received extra questions after the post-test to gather feedback on their experience of having visual explanations.

### 4.1.3 Participants

The experiment participants were selected using convenience sampling. The selection was made so that all the participants were third year computer science students at University of Tartu ensuring that all of them had similar experience in programming, especially in the Java programming language. The sample included 6 people – 4 female and 2 male participants.

Since the sample was small, participants were divided into two groups according to their availability. Each participant was given a pseudonym based on their group and order of taking part in the experiment. Therefore, control group participants were named ExpC 1, ExpC 2, ExpC 3, and intervention group participants were named ExpI 1, ExpI 2, ExpI 3.

### 4.1.4 Descriptive Statistics

The experiment pre- and post-test results were analysed using descriptive statistics. The goal was to observe how the motivation changed before and after doing the quiz and what were the main differences between control and intervention group responses.

Descriptive statistics is a form of quantitative analysis used for summarising and describing the main characteristics of a data set [28]. The different types of descriptive statistics include distribution, which refers to the frequency of each answer, central tendency, estimating the average value of answers, and variability, which shows how distributed the answer values are.

The intervention and control group IMI results were analysed by calculating the mean scores of both groups' pre- and post-test responses. The pre- and post-test scores were compared within both experiment groups and later the groups were compared to each other. The scores

were also compared according to the question type and subscale. The participants who were in the intervention group were asked additional questions about their experience with the visualised prototype. The answers to these questions were summarised to give feedback on the prototype. The additional questions along with pre- and post-tests can be found in experiment guide (see Appendix IV. Experiment Guide).

## 4.2 Prototype Development

The idea was to create an educational learning tool called Syntax Bee. At first, the prototype was supposed to include multiple gameful programming tasks about different topics based on the interview findings. After having complications while trying to make the tasks work, the format was changed. The refined idea included creating a multiple-choice quiz where each question has visualisation according to question type and answers.

The prototype development process started with creating interactive base components. Figma components are reusable elements that help to keep consistency throughout designs and manage time more efficiently since the instances of a main component can be managed and updated all at once [29]. The component Answer State determines the design of an answer option square – no stroke for unanswered view, green stroke for correct choice, red stroke for incorrect choice. The component Answer Base uses four instances of component Answer State to create the design of multiple-choice options. The Answer Base component has two properties: one for managing whether the view is Default or Answered, second for setting the correct answer option. These properties create 8 variants of possible views. The Default and Answered variants with the first option set as correct can be seen in Figure 1.

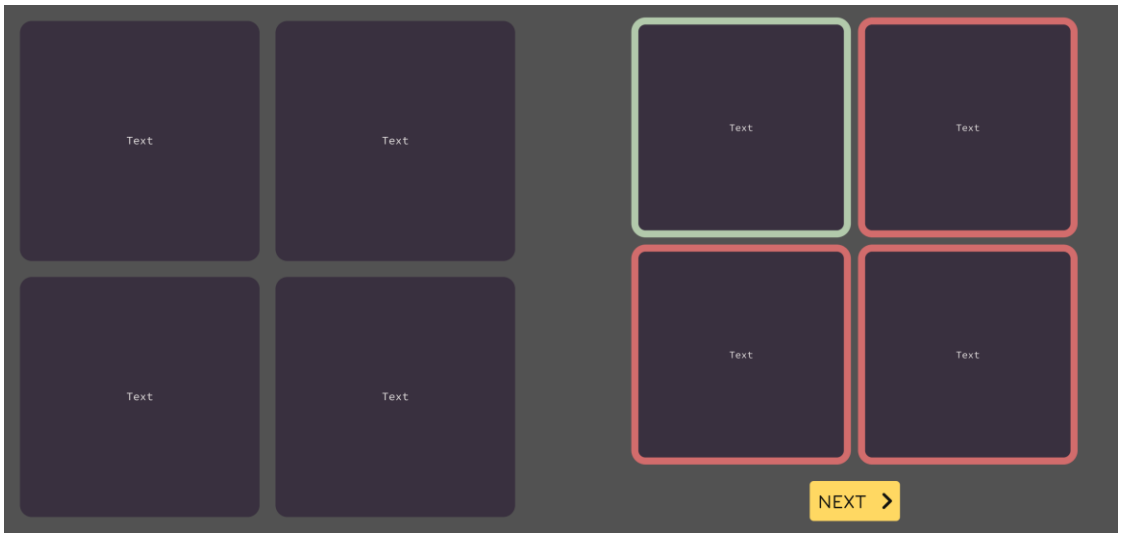


Figure 1. Answer Base variants “1, Default” and “1, Answered”.

Next, the layout for the prototype was made. This included creating frames Question Base and Overlay Base which were not added as components since these were different with each question. The Question Base was used for creating the question layout – left side includes question text and code box, right side has the clickable answer options. An example of using Question Base for the second question can be seen in Figure 2. In total, eight questions pages were created, and each of them leads to the following one when clicking the Next button.

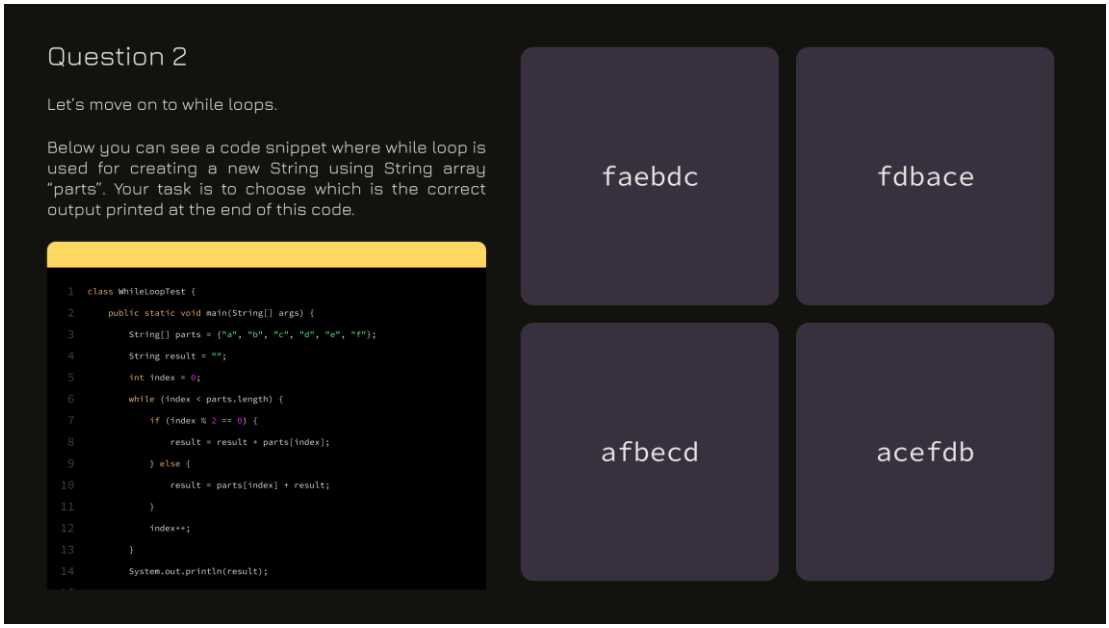


Figure 2. Question Base layout used in question 2.

After the question pages were created, the content was added. The quiz was based on Java programming language and the themes that the quiz included were recursion, keywords, HashMap built-in methods and while loops. The questions were divided into two parts within the quiz. The first four questions focused on separate topics, starting with shorter and easier code segments. The last four questions were story-based and included more complex code segments. The idea was to introduce the topics in the first four tasks and then move on to one larger task that is divided into smaller doable chunks. The idea of having step-by-step approach and moving from easier to harder questions was mentioned in the interviews.

For visualising the answer options, overlays were used. Each question had four answer options and for each of these, an explanatory illustration was made. There was one exception – in question 2, only the correct answer was visualised because otherwise the explanation would have given away the correct answer. Most visualisations consisted of step-by-step code explanations, displaying what each code line does. For showing the current code line, a bee

character was used as a pointer. An example of an answer visualisation can be seen in Figure 3. The users could navigate through the visualisation by using left and right arrow buttons at the bottom of the overlay. To exit the illustration, users could click the X button on the top right corner of overlay or click anywhere on the screen outside of the overlay.


Figure 3. Question 6 answer 1 visualisation.

Finally, each answer option was linked to open when its respective overlay was clicked. Each overlay was connected to the answer in a way that it could be reopened even when the answer had already been clicked on. The experiment required two versions of the quiz to be created, and therefore a duplicate without the overlay interactions was made as well. Instead of having visual explanations for each answer option, this quiz only gave feedback on whether the answer was correct or not.

The prototype can be accessed from the following Figma link: https://www.figma.com/design/GraSv27ixIogFfKxkXYygA/Syntax-Bee---Quiz-Prototype?node-id=0-1&t=GjY5BSvkmJc3GErY-1. Both prototype versions are included in this Figma file. Either one, Quiz Visualised or Quiz Plain, can be selected by clicking on the Pages section in the left sidebar.

## 4.3 Experiment Setup

The goal of the experiment was to evaluate how code visualisation affects the motivation of learning programming languages. The experiment followed an experiment guide (see

Appendix IV. Experiment Guide) which included asking participants to solve the Java programming language quiz and collecting data on their intrinsic motivation before and after solving the quiz.

The participants were divided into two groups: intervention and control group. Both groups received identical pre-tests and post-tests, and all of them were required to take the quiz as well. The quiz had identical questions and answers but what set the two groups apart was visual representation. Members of the intervention group received the quiz with visualisation, meaning that when they chose an answer option in the quiz, an illustration explaining the answer was displayed. The intervention group was also asked additional questions after the post-test to reflect on their experience of having visual representation while solving the quiz. These questions can be found in experiment guide as well (see Appendix IV. Experiment Guide). The control group received the plain version of the quiz without visualisation of answers and were not asked additional questions after filling the post-test.

In total, 6 experiments were conducted. Three of the experiments included solving the quiz with visual representation, the other three participants received the plain version. Each experiment was done individually through in-person meetings and each session lasted between 30-60 minutes.

Each attempt of solving the quiz was screen recorded. Later these recordings were analysed to gather feedback on how much was the visualisation used while solving the quiz. Participants were informed of the screen recording beforehand. The participants received project information (see Appendix V. Experiment Project Information), privacy notice and consent form (see Appendix VI. Experiment Consent Form) to inform them about the experiment and ensure their permission to use the collected data in the thesis.

## 4.4 Experiment Findings

Both groups' IMI results before and after solving the quiz were formatted as a table. The intervention group's pre- and post-test mean scores for each question can be found in Table 1.

Table 1. Mean scores of motivation questionnaire results for the intervention group

| IMI subscale | Question id | Pre-test mean | Post-test mean |
|---|---|---|---|
| interest/enjoyment | 1 | 4.33 | 6 |
| | 2 | 3.66 | 5.33 |

| | | | |
|---|---|---|---|
| perceived competence | 3 | 5 | 4.33 |
| | 4 | 5.33 | 5.33 |
| pressure/tension | 5 | 4.66 | 4 |
| value/usefulness | 6 | 5.66 | 6 |
| | 7 | 6 | 5.66 |
| | 8 | 5.66 | 5.66 |

According to the IMI results, for half of the questions participants' motivation had grown or stayed the same. The biggest difference could be seen in the interest/enjoyment subscale, where for both questions the motivation had significantly increased. The perceived competence subscale on the other hand showed decrease in motivation. When talking about pressure/tension, the score had also decreased, which is specific for this subscale since a lower score shows less pressure and tension in this case. Therefore, participants felt less tense after having solved the test. When focusing on the value/usefulness subscale, all of the three questions' scores showed different results. The motivation about the quiz being useful for learning increased, the score of finding the quiz beneficial stayed the same, and the score indicating that the quiz would help learn more efficiently decreased.

The control group's pre- and post-test mean scores for each question are displayed in Table 2. The results showed that for each question the score had decreased. This also includes the pressure/tension subscale which, as mentioned before, shows reversed results since higher score indicates more tension.

Table 2. Mean scores of motivation questionnaire results for the control group

| **IMI subscale** | **Question id** | **Pre-test mean** | **Post-test mean** |
|---|---|---|---|
| interest/enjoyment | 1 | 4.67 | 3.67 |
| | 2 | 4.33 | 3.67 |
| perceived competence | 3 | 4.67 | 4 |
| | 4 | 4.67 | 3.67 |
| pressure/tension | 5 | 4 | 2.67 |
| value/usefulness | 6 | 4.33 | 4 |
| | 7 | 5.67 | 3.67 |

| | | | |
|---|---|---|---|
| | 8 | 5 | 4.67 |

Both groups had similar pre-test scores with the average answer value being 5.04 for the intervention group and 4.67 for the control group. The average post-test answer value was 5.29 for the intervention group and 3.75 for the control group. Although the intervention group had a higher motivation score to begin with, their motivation increased after solving the quiz prototype. The control group's motivation showed an overall decrease in its value. For both groups, the average difference between pre- and post-test question scores was smaller than 1. The intervention group had both increases and decreases between pre- and post-test scores and the average difference was 0.67. The control group on the other hand received only decreased scores and the average difference was 0.92 which is approximately one-third more than the intervention group average.

In addition to intrinsic motivation inventory questions, the intervention group were also asked to answer additional questions concerning the visualised quiz prototype. All intervention group participants (ExpI 1, ExpI 2 and ExpI 3) brought out how visualisation made understanding the tasks easier. When asked about what bothered them, both ExpI 1 and ExpI 2 answered that the text was too small and too closely together. ExpI 3 brought up that having too many different colours was distracting, while the ExpI 2 said that they preferred having a colourful interface because it made the information easier to grasp. Both ExpI 1 and ExpI 3 also mentioned that when the overlay opened, they did not see if the answer they clicked was correct. Therefore, they would have liked the overlay to contain a text or coloured border showing whether their answer was correct or not.

# 5. Discussion

The study results showed that visualisation could be a helpful addition for teaching and understanding programming languages. All of the interviews showed that students would like to have code visualisation to explain programming concepts. For example, Int 2 specifically mentioned using debugging because it helps understand what each code line does, while other participants brought up that videos and illustrations can also give a better understanding of different topics. When conducting the experiment, it was found that the intervention group who solved the quiz with visual aid had higher intrinsic motivation after the experiment with the average post-test answer value being 5.29. The control group participants, who received the plain version of the quiz prototype, had lower intrinsic motivation with the average answer value of 3.75 after solving the quiz. These results are supported by experiment feedback as well. When asked about the visualisation quiz prototype, all three intervention group participants said that they enjoyed having visualised answers and that visual representation makes programming concepts more understandable.

Since the experiment sample size was small, including three participants in both groups, it can be argued that the results are not too accurate. The results mirrored participants' personal experience as the value increases or decreases altered with each question and subscale. Still, even with the small sample size, it became clear that answer visualisation mostly increases the motivation of learning and understanding programming concepts. Using visualisation can provide an alternative approach for teaching coding concepts in an engaging and motivational way.

In the future, the prototype could be developed into a real education tool which could provide help with explaining different programming languages and concepts. Both interviews and experiment feedback gave ideas for future developments. From the interviews, it was established that people enjoy gameful factors to support their learning. The interviewees brought up competitiveness and leaderboards, daily challenges, and preference-based topic selection. Besides these ideas, the quiz feedback showed that people would like to see whether the answer was correct or not in the overlay. All of these could be implemented to create a more personal learning tool for students to use.

# Conclusion

The goal of the thesis was to investigate how having code visualisation affects the intrinsic motivation of students who are learning programming. The thesis included theoretical background, the undertaken study and discussion. The theory chapter gave an overview of how programming is studied, how motivation affects learning to program, what is visual learning and how is it used in programming education. The practical part included a study which was divided into two parts: interviews and experiment. The interviews were used to gather information on what features the visualised prototype should include and the experiment was conducted to see what effect code visualisation has on participants' intrinsic motivation. Since the research used a quasi-experimental design, the prototype had two versions – one with answer visualisation for the intervention group and a plain version for the control group. Before and after solving the quiz, both group's participants were asked to fill a questionnaire to reflect their motivation. These responses were then analysed and compared to explore the impact that visual representation had on participants' intrinsic motivation. The discussion described the results and possible further developments of the research.

The interviews highlighted students' problems and motivators while learning programming. For example, these included pain points like not knowing where to start or getting stuck with solving tasks, and helpful aspects such as visual explanations or seeing the output of a program. The study results showed that using visual representation in teaching programming languages can increase students' motivation. The participant feedback also reflected that having visualisation was helpful for them and made understanding answers easier. Code visualisation could make learning programming more engaging and increase students' motivation.

# References

[1] Mohorovičić, S., Strčić, V. An Overview of Computer Programming Teaching Methods. *Proceedings of the 22nd Central European Conference on Information and Intelligent Systems*, 2011, pp. 47–52.

[2] Calderon, I., Silva, W., Feitosa, E. Active Learning Methodologies for Teaching Programming in Undergraduate Courses: A Systematic Mapping Study. *Informatics in Education*, 2024, 23(2), pp. 279-322. https://doi.org/10.15388/infedu.2024.11

[3] Zhang, A., Olelewe, C. J., Orji, C. T., Ibezim, N. E., Sunday, N. H., Obichukwu, P. U., Okanazu, O. O. Effects of Innovative and Traditional Teaching Methods on Technical College Students' Achievement in Computer Craft Practices. *SAGE Open*, 2020, 10(4), pp. 1–11. https://doi.org/10.1177/21582440209829

[4] Codesee. Code Visualization: 4 Types of Diagrams and 5 Useful Tools. https://www.codesee.io/learning-center/code-visualization (26.04.2025)

[5] Olsson, M., Mozelius, P., Collin, J. Visualisation and Gamification of e-Learning and Programming Education. *The Electronic Journal of e-Learning*, 2015, 13(6), pp. 441-454.

[6] Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., Santos, E. A. Programming Is Hard– Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, Toronto, Canada, March 15–18, 2023. New York: ACM, pp. 500–506. https://doi.org/10.1145/3545945.3569759

[7] Gomes, A., Ke, W., Lam, C.-T., Marcelino, M. J., Mendes, A. Student Motivation Towards Learning to Program. *2018 IEEE Frontiers in Education Conference (FIE)*, San Jose, USA, October 3–6, 2018. Piscataway: IEEE, pp. 1–9. https://doi.org/10.1109/FIE.2018.8659134

[8] Silva, L., Gomes, A., Borges, A. R., Vasconcelos, V., Mendes, A. J. A Study on the Motivation of Computer Science Students to Learn Programming. *2023 International Symposium on Computers in Education (SIIE)*, Setúbal, Portugal, November 16–18, 2023. Piscataway: IEEE, pp. 1–7. https://doi.org/10.1109/SIIE59826.2023.10423706

[9] Center for Self-Determination Theory, Intrinsic Motivation Inventory (IMI). https://selfdeterminationtheory.org/intrinsic-motivation-inventory/ (09.04.2025).

[10] APA Dictionary of Psychology. https://dictionary.apa.org/ (09.04.2025).

[11] Fadiran, O. A., van Biljon, J., Schoeman, M. A. How Can Visualisation Principles Be Used to Support Knowledge Transfer in Teaching and Learning? *2018 Conference on Information Communications Technology and Society (ICTAS)*, Durban, South Africa, March 8–9, 2018. Piscataway: IEEE, pp. 1–6. https://doi.org/10.1109/ICTAS.2018.8368739

[12] Kim, M., Jin, Q. Studies on Visualisation in Science Classrooms: a Systematic Literature Review. *International Journal of Science Education*, 2022, 44(17), pp. 2613–2631. https://doi.org/10.1080/09500693.2022.2140020

[13] Shatri, K., Buza, K. The Use of Visualization in Teaching and Learning Process for Developing Critical Thinking of Students. *European Journal of Social Science Education and Research*, 2017, 4(3), pp. 134–140. https://doi.org/10.26417/ejser.v9i1.p71-74

[14] Miro. The ultimate guide to UML diagrams. https://miro.com/diagramming/what-is-a-uml-diagram/ (13.05.2025)

[15] GeeksforGeeks. Introduction to Flowcharts. https://www.geeksforgeeks.org/an-introduction-to-flowcharts/ (13.05.2025)

[16] GeeksforGeeks. What is Debugging in Software Engineering? https://www.geeksforgeeks.org/software-engineering-debugging/ (13.05.2025)

[17] IBM. Stepping through a program. https://www.ibm.com/docs/en/debug-for-zos/17.0.x?topic=introduction-stepping-through-program (13.05.2025)

[18] Santolucito, M., Hallahan, W. T., Piskac, R. Live Programming By Example. *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems (CHI EA '19)*, 2019, pp. 1–4. https://doi.org/10.1145/3290607.331326

[19] Lian, V., Varoy, E., Giacaman, N. Learning Object-Oriented Programming Concepts Through Visual Analogies. *IEEE Transactions on Learning Technologies*, 15(1), 2022, pp. 78–92. https://doi.org/10.1109/TLT.2022.3154805

[20] Dina, N. Z., Wuryanto, E., Marijanto, R. S. Evaluation on the Effectiveness of Visual Learning Environment on Programming Course From Students' Perspectives. *IIUM Engineering Journal*, 2019, 20(1), pp. 100–107. https://doi.org/10.31436/iiumej.v20i1.993

[21] Mladenović, M., Žanko, Ž., Čuvić, M. A. The Impact of Using Program Visualization Techniques on Learning Basic Programming Concepts at the K–12 Level. *Computer*

*Applications in Engineering Education*, 2021, 29, pp. 145–159. https://doi.org/10.1002/cae.22315

[22] Karatsareas, P. Semi-Structured Interviews. *Research Methods in Language Attitudes*. Ed. by R. Kircher, L. Zipp. Cambridge: Cambridge University Press, 2022, pp. 99–113.

[23] Clarke, V., Braun, V. Thematic analysis, *The Journal of Positive Psychology*, 2016, 12(3), pp. 297–298. https://doi.org/10.1080/17439760.2016.1262613

[24] IBM. What is Java? https://www.ibm.com/think/topics/java (10.04.2025)

[25] Staiano, F. Designing and Prototyping Interfaces with Figma: Learn essential UX/UI design principles by creating interactive prototypes for mobile, tablet, and desktop. UK: Packt Publishing Ltd., 2022, pp. 4–5.

[26] Gopalan, M., Rosinger, K., & Ahn, J. B. Use of Quasi-Experimental Research Designs in Education Research: Growth, Promise, and Challenges. *Review of Research in Education*, 2020, 44(1), pp. 218–243. https://doi.org/10.3102/0091732X20903302

[27] Lam, C., Wolfe, J. An Introduction to Quasi-Experimental Research for Technical and Professional Communication Instructors. *Journal of Business and Technical Communication*, 2022, 37(2), pp. 174–193. https://doi.org/10.1177/10506519221143111

[28] Scribbr. Descriptive Statistics https://www.scribbr.com/statistics/descriptive-statistics/ (29.04.2025)

[29] Figma. Guide to components in Figma. https://help.figma.com/hc/en-us/articles/360038662654-Guide-to-components-in-Figma (22.04.2025)

# Appendices

## Appendix I. Interview Guide

**Introduction (3-5 min)**

Hello and thank you for joining me. My name is Brigitta and I have asked you to come in today to answer some questions regarding my Bachelor's thesis. I am developing an interactive prototype for a game-based education tool for learning programming. The goal of this interview is to determine which features should the prototype include. For that I would like to ask you some questions concerning your experience with learning programming and how to make the learning process more efficient.

As mentioned in project information sheet, taking part in this interview is voluntary. If you do not wish to continue at any point, just let me know. This meeting will also be audio recorded.

Do you have any questions before we begin?

I am going to start the recording.

**Programming experience and background (5-10 min)**

Let's start with your experience in programming.

1. Where and when did you first start learning programming? *(e.g. at high school, university, self-taught)*
2. For how long have you been practising it?
3. Which languages are you familiar with and to what level?
4. What resources have you used for learning programming? *(e.g. online courses, university courses, books, websites, games, bootcamps, etc.)*

**Learning preferences (10-20 min)**

We will now move on to your learning preferences.

5. What struggles have you faced when learning programming?
6. Which programming concepts need more thorough explanation in your experience?
7. What would you like to see improved in teaching programming? *(e.g. more repetition, real-life examples of coding, etc.)*

8. Which activities best help you understand new coding concepts? *(e.g. reading theory first, hands on coding exercises, debugging to see how the code works, flowcharts, etc.)*
9. Do you prefer understanding theory before doing exercises or trying to solve exercises in parallel with learning theory?
10. What would motivate you while learning programming? *(e.g. challenges, advancing on levels, rewards, solving real-life problems, seeing immediate results, etc.)*
11. Have you tried any game-based tools for learning programming?
    a. If so: Which one(s) have you used? What parts/components did you like or dislike about them?
    b. If not: Why have you not used any? What would you like to see in a game for learning programming?
12. Have you used additional tactics/tools to understand programming better; what specifically?
13. If you could design a programming language learning game, which features would you tredefinitely like to include? *(e.g. mini-games, scoreboards, daily challenges, quizzes for theory)*

**Closing (2-5 min)**

That concludes my questions.

Do you have anything else to add regarding learning programming or game-based education?

Thank you for coming!

## Appendix II. Interview Project Information

# Project Information

**Date:** 05.03.2025

**If you have any questions about the project information sheet and/or are unsure what it means for you personally, please email Brigitta Ambur at brigitta.ambur@ut.ee.**

## Purpose of the Research

**What is this research about?**

This research explores how code visualisation enhances learning programming.
For my Bachelor's thesis I am creating a game-based education tool prototype that will focus on teaching programming through gameful activities and code visualisation. To determine which features and concepts should be included in the prototype I am conducting interviews with people who have studied programming and would like to share their experience.

## Nature of Participation

**What will happen during the study?**

Taking part in this project is entirely voluntary.
It involves:

- Having a 20-40-minute interview regarding your experience and preferences for learning programming.

You can leave the study at any point, without the need to give a reason, by informing the main researcher via email at brigitta.ambur@ut.ee.

## Benefits of the Research

**What will I get out of it?**

- Supporting science! You help to create new knowledge regarding how visual representation enhances the learning process of programming.

## Risks of the Research

**What could go wrong?**

- We will do our best to ensure your anonymity, but people who know you well might identify you based on quotes in publications from this project.
- Since the interview will be recorded, the resulting data might get accessed and/or stolen by malicious third parties that are beyond our control (examples: computers get

stolen, the university server gets hacked). University of Tartu has firm security and privacy policies in place to minimise this risk.

# Use of Your Data

**What happens to my data?**

You can read in more detail about how we process your data in the Privacy Notice for this project. The following section gives a summarising overview.

**Data Processing**

With your consent, data gathered for this project (audio recordings and written notes) will be used …

... to discuss the project (between University of Tartu researchers and affiliate researchers).

... analysed and showcased in University of Tartu's research papers and presentations in academic spaces.

... analysed and showcased on websites and social media (reports, blog posts, twitter).

We will also anonymise you to the best of our ability by removing identifying information. We will never publish or share *raw* data, e.g., the recording file of a video call.

**Data Confidentiality**

All personal data collected during the study will be processed in compliance with the EU's General Data Protection Regulation (GDPR) and the data protection laws of Estonia.
Data will be anonymised, e.g., by assigning you a pseudonym and expunging identifying information, e.g., if you mention where you live.

**Data Storage**

Your data will be stored in a folder in Brigitta Ambur's Tartu University OneDrive cloud storage.
More information about how your data is handled, can be found in this project's privacy notice.

# Withdrawal from the Research

**How can I leave the study?**

You can withdraw from the study at any time and do not have to give reasons for why you no longer want to take part. If you wish to withdraw, please contact Brigitta Ambur via email brigitta.ambur@ut.ee.

## Research Background and Funding Sources

This research study is undertaken as part of a Bachelor's degree thesis in Computer Science at the University of Tartu.

## Researcher Contact Details

Main Researcher - Name: Brigitta Ambur, Email: brigitta.ambur@ut.ee

Supervisor(s) – Name: Velvet Spors, Email: velvet.spors@ut.ee

## Contact Details of the Ethics Committee

If you wish to file a complaint about this study or exercise your rights, you can contact University of Tartu's Ethics Committee at the following email address: eetikakomitee@ut.ee

## Appendix III. Interview Consent Form

# Consent Form

**Date:** 05.03.2025

**If you have any questions about the project information sheet and/or are unsure what it means for you personally, please email Brigitta Ambur at brigitta.ambur@ut.ee.**

Thank you for wanting to take part in this research study. This consent form asks you questions to check that you are comfortable with the research and to confirm that ...

... you are happy to participate in the interview and...

... the ways in which you would like to participate in the research.

**The form assumes that you've checked out the following study documents:**

- **Project Information**: This document describes the project in greater detail and explains what happens during the study.
- **Privacy Policy**: This document explains what happens to your data during the project, your rights and shows which tools and services we will use during the jam.

These documents make sense to you, and you do not have any questions. If you have any questions, please email brigitta.ambur@ut.ee.

| Question | Your Answer |
|---|---|
| What's your name? | (please write) |
| What's your email? | (please write) |
| I understand what the study is about.<br>I have read the project info sheet. | Yes / No<br>(please circle) |
| I know that I can ask questions about the study before taking part by emailing the main researcher at brigitta.ambur@ut.ee and all questions that I have asked, have been answered in detail. | Yes / No<br>(please circle) |
| I understand that taking part in this study means creating data in the form of:<br>- talking<br>- written notes<br>- audio recording | Yes / No<br>(please circle) |

| | |
|---|---|
| I understand that the interview will be audio recorded, transcribed, and analysed.<br>The recording is made during the interview with OBS recording platform. Later a transcription is made for analysing the collected data. The transcription will be uploaded to university-controlled cloud storage. | Yes / No<br>(please circle) |
| I agree with my data, what I say in the interview, being used ... | |
| ... to discuss the project (Brigitta Ambur, Velvet Spors). | Yes / No<br>(please circle) |
| … for research papers and presentations in academic spaces. | Yes / No<br>(please circle) |
| I confirm that I have read the project's privacy notice which describes how my data is safe-guarded.<br>I understand that safeguards will be put into place to protect my identity, and data are acceptable to me. | Yes / No<br>(please circle) |
| I understand that no computer system is completely secure.<br>I understand that there is always a risk that someone else might be able to get an unauthorised copy of my data, e.g. the university gets hacked etc. | Yes / No<br>(please circle) |
| I give permission for data gathered during this project to be used, copied, excerpted, annotated, displayed and distributed for the purposes to which I have consented. | Yes / No<br>(please circle) |
| I agree to voluntarily take part in this study. | Yes / No<br>(please circle) |
| Date and signature: | |

# Appendix IV. Experiment Guide

**Introduction (00:00 – 00:05)**

Hello and thank you for joining me. Today we are going to do an experiment for testing a prototype I have created for my bachelor's thesis. The prototype consists of a multiple-choice quiz with questions about Java programming language. The goal is to test the prototype to evaluate its efficiency for learning programming.

Everything said and done in here will only be used for research purposes. This experiment is voluntary, if at any point you wish to leave, just let me know.

Do you have any questions before we begin?

**Pre-test (00:05 – 00:10)**

Before starting the quiz, I would like you to fill out this short questionnaire regarding your motivation for the upcoming tasks.

**Not at all true** **Somewhat true** **Very true**

| **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|
| 1. I expect to enjoy solving the Java based programming quiz. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2. I believe I will describe solving the Java based programming quiz as very interesting. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3. I believe I will be pretty skilled in answering Java based programming quiz questions. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4. I expect that after doing the Java based programming quiz for a while, I will feel pretty competent. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 5. I expect to feel tense while solving the Java based programming quiz. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 6. I think that solving the Java based programming quiz will be useful for learning. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 7. I think the Java based programming quiz could help me learn more efficiently. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8. I believe solving the Java based programming quiz could be beneficial to me. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

**Intro to task (00:10 – 00:15)**

We will now move on to the prototype testing. Your task is to solve a quiz. Let me know when you have finished. Also, if at any point you have problems or questions don't hesitate to reach out to me.

The quiz consists of 8 multiple-choice Java programming language related questions. You will see different code segments and need to answer based on these.

You may now start the quiz.

**Task (00:15 – 00:45)**

*Give the participant the quiz to solve.*

**Post-test (00:45– 00:50)**

Thank you for participating. Before you leave, I would like you to answer this questionnaire to reflect your motivation after solving the quiz.

**Not at all true** **Somewhat true** **Very true**

**1 2 3 4 5 6 7**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1. I expect to enjoy solving the Java based programming quiz. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2. I believe I will describe solving the Java based programming quiz as very interesting. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3. I believe I will be pretty skilled in answering Java based programming quiz questions. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 4. I expect that after doing the Java based programming quiz for a while, I will feel pretty competent. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 5. I expect to feel tense while solving the Java based programming quiz. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 6. I think that solving the Java based programming quiz will be useful for learning. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 7. I think the Java based programming quiz could help me learn more efficiently. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8. I believe solving the Java based programming quiz could be beneficial to me. | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

*Additional questions for intervention group:*

1. Did the visualisation of answers make understanding the code easier?

2. What bothered you about the visualisation?

3. What would you like to improve about the visual illustrations?

**Closing (00:50– 00:55)**

That concludes today’s experiment.

Would you like to add something, some thoughts about the whole experience or other ideas?

Thank you so much for coming!

## Appendix V. Experiment Project Information

# Project Information

**Date:** 18.04.2025

**If you have any questions about the project information sheet and/or are unsure what it means for you personally, please email Brigitta Ambur at brigitta.ambur@ut.ee.**

## Purpose of the Research

**What is this research about?**

This research explores how different approaches of explaining and solving coding tasks impact the experience of learning programming.

For my Bachelor's thesis I am creating an interactive prototype for learning Java programming language. To understand what enhances and increases motivation while learning programming concepts I am conducting an experiment where people can engage with an interactive prototype for learning programming.

## Nature of Participation

**What will happen during the study?**

Taking part in this project is entirely voluntary.

It involves:

- Having ~1-hour long experiment for testing the prototype.
- Answering two questionnaires about motivation.

You can leave the study at any point, without the need to give a reason, by informing the main researcher via email at brigitta.ambur@ut.ee.

## Benefits of the Research

**What will I get out of it?**

- Supporting science! You help to create new knowledge regarding what approaches motivate learning programming concepts.

## Risks of the Research

**What could go wrong?**

- We will do our best to ensure your anonymity, but people who know you well might identify you based on quotes in publications from this project.

- Since the experiment will be recorded, the resulting data might get accessed and/or stolen by malicious third parties that are beyond our control (examples: computers get stolen, the university server gets hacked). University of Tartu has firm security and privacy policies in place to minimise this risk.

# Use of Your Data

**What happens to my data?**

You can read in more detail about how we process your data in the Privacy Notice for this project. The following section gives a summarising overview.

**Data Processing**

With your consent, data gathered for this project (audio recordings and written notes) will be used …

... to discuss the project (between University of Tartu researchers and affiliate researchers).

... analysed and showcased in University of Tartu's research papers and presentations in academic spaces.

... analysed and showcased on websites and social media (reports, blog posts, twitter).

We will also anonymise you to the best of our ability by removing identifying information. We will never publish or share *raw* data, e.g., the recording file of a video call.

**Data Confidentiality**

All personal data collected during the study will be processed in compliance with the EU's General Data Protection Regulation (GDPR) and the data protection laws of Estonia.
Data will be anonymised, e.g., by assigning you a pseudonym and expunging identifying information, e.g., if you mention where you live.

**Data Storage**

Your data will be stored in a folder in Brigitta Ambur's Tartu University OneDrive cloud storage.
More information about how your data is handled, can be found in this project's privacy notice.

# Withdrawal from the Research

**How can I leave the study?**

You can withdraw from the study at any time and do not have to give reasons for why you no longer want to take part. If you wish to withdraw, please contact Brigitta Ambur via email brigitta.ambur@ut.ee.

## Research Background and Funding Sources

This research study is undertaken as part of a Bachelor's degree thesis in Computer Science at the University of Tartu.

## Researcher Contact Details

Main Researcher - Name: Brigitta Ambur, Email: brigitta.ambur@ut.ee
Supervisor(s) – Name: Velvet Spors, Email: velvet.spors@ut.ee

## Contact Details of the Ethics Committee

If you wish to file a complaint about this study or exercise your rights, you can contact University of Tartu's Ethics Committee at the following email address: eetikakomitee@ut.ee

## Appendix VI. Experiment Consent Form

# Consent Form

**Date:** 05.03.2025

**If you have any questions about the project information sheet and/or are unsure what it means for you personally, please email Brigitta Ambur at brigitta.ambur@ut.ee.**

Thank you for wanting to take part in this research study. This consent form asks you questions to check that you are comfortable with the research and to confirm that ...

... you are happy to participate in the experiment and...

... the ways in which you would like to participate in the research.

**The form assumes that you've checked out the following study documents:**

- **Project Information**: This document describes the project in greater detail and explains what happens during the study.
- **Privacy Policy**: This document explains what happens to your data during the project, your rights and shows which tools and services we will use during the jam.

These documents make sense to you, and you do not have any questions. If you have any questions, please email brigitta.ambur@ut.ee.

| Question | Your Answer |
|---|---|
| What's your name? | (please write) |
| What's your email? | (please write) |
| I understand what the study is about.<br>I have read the project info sheet. | Yes / No<br>(please circle) |
| I know that I can ask questions about the study before taking part by emailing the main researcher at brigitta.ambur@ut.ee and all questions that I have asked, have been answered in detail. | Yes / No<br>(please circle) |
| I understand that taking part in this study means creating data in the form of:<br>- answering two questionnaires<br>- screen recording of testing the prototype | Yes / No<br>(please circle) |

| | |
|---|---|
| I understand that the interview will be audio recorded, transcribed, and analysed.<br>The recording is made during the experiment with OBS recording platform. No audio will be recorded. Later, the screen recording is analysed for research purposes. The screen recording will be uploaded to university-controlled cloud storage. | Yes / No<br>(please circle) |
| I agree with my data, what I write, discuss, do during the experiment, being used ... | |
| ... to discuss the project (Brigitta Ambur, Velvet Spors). | Yes / No<br>(please circle) |
| … for research papers and presentations in academic spaces. | Yes / No<br>(please circle) |
| I confirm that I have read the project's privacy notice which describes how my data is safe-guarded.<br>I understand that safeguards will be put into place to protect my identity, and data are acceptable to me. | Yes / No<br>(please circle) |
| I understand that no computer system is completely secure.<br>I understand that there is always a risk that someone else might be able to get an unauthorised copy of my data, e.g. the university gets hacked etc. | Yes / No<br>(please circle) |
| I give permission for data gathered during this project to be used, copied, excerpted, annotated, displayed and distributed for the purposes to which I have consented. | Yes / No<br>(please circle) |
| I agree to voluntarily take part in this study. | Yes / No<br>(please circle) |
| Date and signature: | |

# License

**Non-exclusive licence to reproduce the thesis and make the thesis public**

I, Brigitta Ambur,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis **Syntax Bee: An Interactive Prototype for Investigating the Impact of Code Visualisation on the Motivation in Learning Programming**, supervised by Velvet Spors;

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

*Brigitta Ambur*

***14.05.2025***