KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

First Cycle, 15 credits

Easing the transition from block-based programming in education

Comparing two ways of transitioning from block-based to text-based programming and an alternative way to solve the transition problem

EDVIN NORDLING

JOAKIM ABDINUR IUSUF

Stockholm, Sweden 2023

Easing the transition from block-based programming in education

Comparing two ways of transitioning from block-based to text-based programming and an alternative way to solve the transition problem

EDVIN NORDLING

JOAKIM ABDINUR IUSUF

Degree Project in Computer Science and Engineering, First Cycle 15 credits
Date: June 11, 2023
Supervisor: Johan Jansson
Examiner: Pawel Herman
Swedish title: Att underlätta övergången från blockbaserad programmering inom utbildning
School of Electrical Engineering and Computer Science

Abstract

Many learners find the transition from block-based programming to text-based programming difficult. Consequently, research has investigated how block-based languages support learners when making the transition to text-based programming. It categorized the way in which block-based languages support the transition into one-way transition, dual-modality and hybrid environments. This research investigates how one-way transition environments compare to dual-modality environments with regards to learning a text-based language, and how the two modalities differ with regards to the motivational factors satisfaction, enjoyment and easiness. The results show that dual-modality environments could be a better alternative than one-way transition environment when learners make the transition from block-based to text-based programming. The results also show that solving a problem in dual-modality environments could be easier than solving them in one-way transition environments, which could potentially mean that learners experience more motivation when making the transition in a dual-modality environment. This study also investigated if there is an alternative to one-way transition, dual-modality and hybrid environments when helping learners transition from block-based to text-based programming, and what a learning activity in this alternative solution could look like. It found that Blockly Games is an alternative, and describes a learning activity built in Blockly Games. Future research should aim at gaining a deeper understanding of the differences between one-way transition, dual-modality and hybrid environments, and investigate if the approach taken by Blockly Games is a better alternative.

Keywords

Block-based programming, Block-based languages, Text-based programming, Text-based languages, One-way transition environments, Dual-modality environments, Hybrid environments, Blockly, BlockPy, Blockly Games

Sammanfattning

Många elever tycker att övergången från blockbaserad programmering till textbaserad programmering är svår. Följaktligen har forskning undersökt hur blockbaserade språk stödjer elever när de gör övergången till textbaserad programmering. En studie fann att blockbaserade språk stöder denna övergång med hjälp av one-way transition miljöer, dual-modality miljöer och hybrid miljöer. Denna forskning undersöker hur one-way transition miljöer jämför sig med dual-modality miljöer när det kommer till att lära sig ett textbaserat språk, och hur de två modaliteterna skiljer sig åt med avseende på motivationsfaktorerna tillfredsställelse, njutning och lätthet. Resultaten visar att dual-modality miljöer kan vara ett bättre alternativ än one-way transition miljöer när eleverna gör övergången från blockbaserad till textbaserad programmering. Resultaten visar också att det kan vara lättare att lösa ett problem i dual-modality miljöer än att lösa dem i one-way transition miljöer, vilket potentiellt kan innebära att eleverna upplever mer motivation när de gör övergången i en dual-modality miljö. Denna studie undersökte också om det finns ett alternativ till one-way transition miljöer, dual-modality miljöer och hybrid miljöer när elever ska övergå från blockbaserad till textbaserad programmering, och hur en inlärningsaktivitet i denna alternativa lösning skulle kunna se ut. Den fann att Blockly Games är ett alternativ och beskriver en inlärningsaktivitet byggd i Blockly Games. Framtida forskning borde försöka få en djupare förståelse för skillnaderna mellan one-way transition miljöer, dual-modality miljöer och hybrid miljöer, och undersöka om det tillvägagångssätt som Blockly Games använder är ett bättre alternativ.

Nyckelord

Blockbaserad programmering, Blockbaserade språk, Textbaserad programmering, Textbaserade språk, One-way transition miljöer, Dual-modality miljöer, Hybrid miljöer, Blockly, BlockPy, Blockly Games

Contents

1 Introduction

The rapid growth of digitalisation has led to an increased demand for Computer Science (CS) professionals. For instance, ITTelecomföretagen found that by 2024 Sweden will have a deficit of 70,000 IT professionals [7]. To fill this gap, it is important to increase the number of students who choose CS as a profession. This gives rise to the problem of how to teach programming and increase students' interest in CS, so that more choose to pursue it as a career.

Programming courses have grown in popularity in secondary education [13] and in Sweden, it has become mandatory for math and technology teachers to integrate programming into their teaching [12]. Research has found that teaching novices how to code using text-based programming languages leads to difficulties in their understanding of foundational programming concepts, such as variables, loops and expressions [5]. Moreover, previous research has also found that teaching novices how to code using block-based programming languages increases their interest in enrolling in future CS courses, whereas teaching them how to code using traditional text-based programming languages decreases their interest in taking future CS courses [16]. Block-based programming is seen as a good way to introduce novices, including children, to the basics of programming, as it allows novices to focus on concepts rather than the syntactic complexity of a text-based programming language [13].

However, research has shown that transitioning from block-based programming languages to text-based programming languages is a difficult challenge for learners [13]. In one study, researchers looked at the design approaches that are currently being used in block-based languages to support learners in the transition from block-based to text-based programming. The study found that most block-based languages are block-only environments, meaning that the languages do not support learners in transitioning to text-based programming. The other design approaches found were one-way transition environments, dual-modality environments and hybrid environments, which are explained in more detail in the background section. The question of how to best help students transition from block-based languages to text-based languages is an open problem, and the different design approaches that are meant to support this transition are largely

unexplored [9]. There is also the possibility that there are other ways of helping students make this transition than those highlighted by this study.

1.1 Research questions

In order to gain a deeper understanding of the differences between these design approaches we will compare a one-way transition environment with a dual-modality environment. We will explore which of these modalities learners find most effective in learning a text-based language, and we will also look at differences in the motivational factors satisfaction, enjoyment and easiness since these factors can increase learners' motivation and interest in continuing their computer science education [2]. Moreover, this study also explores an alternative to these design approaches in order to give a more complete picture of the ways in which block-based languages can help students transition to text programming, and aims to build a learning activity in this alternative design approach to show how this can be achieved. Thus, this study aims to investigate the following:

- How do one-way transition environments compare to dual-modality environments with regards to learning a text-based language, and how do the two modalities differ with regards to the motivational factors satisfaction, enjoyment and easiness?
- Is there an alternative to one-way transition, dual-modality and hybrid environments when helping learners transition from block-based to text-based programming, and what could a learning activity in this alternative solution look like?

1.2 Scope

As mentioned in the background section, there are two types of one-way transition environments: read-only and editable. In this thesis, we are focusing on read-only environments. Likewise, there are two types of dual-modality environments: shared-view and distinct-view. We are limiting our research to shared-view environments in this thesis. Moreover, hybrid environments will not be studied. We will also limit our research to focusing on three of the four

motivational factors mentioned in the background section. Since the problem solved by the participants is fairly simple, measuring usefulness is deemed to be irrelevant.

2 Background

This section begins with explaining the problem with teaching text-based languages to novices, and then describes block-based programming, which is thought to be a solution to many of the problems associated with teaching novices how to code in a text-based language. After this, some of the problems that arise when making the transition from block-based to text-based programming is explained, and then two ways that block-based languages help learners make this transition is described. We also explain motivational factors commonly used in research, three of which are relevant to this study. Lastly, a learning environment called Blockly Games is described.

2.1 The problem with teaching text-based languages to novices

Programming is a complex cognitive activity, and research has shown that novice learners encounter several issues when learning how to program using a text-based language. Most of this research was conducted in the context of CS1 undergraduate work. Some of this research focused on issues associated with the broader understanding of programming and how to create working programs, whereas the other studies focused on issues pertaining to learning specific programming constructs. This research points to issues related to learning about variables, expressions and loops. Novice learners tend to have a flawed understanding of variable assignments, and difficulty understanding that a single variable can hold different values in different parts of a program. The research also shows that novices have problems distinguishing between what goes on inside a loop and what precedes or follows it, and that an expression containing the control variable of the loop can have different values in each cycle of the loop. It also shows that novices have problems understanding how to initialize variables, and knowing how and when to terminate loops [5].

2.2 Block-based programming

An alternative to teaching novices how to program using text-based languages, is teaching introductory programming using block-based programming. Block-based programming allows novices to create programs by using blocks that can be attached to each other in a puzzle-like form, as seen in Figure 2.1

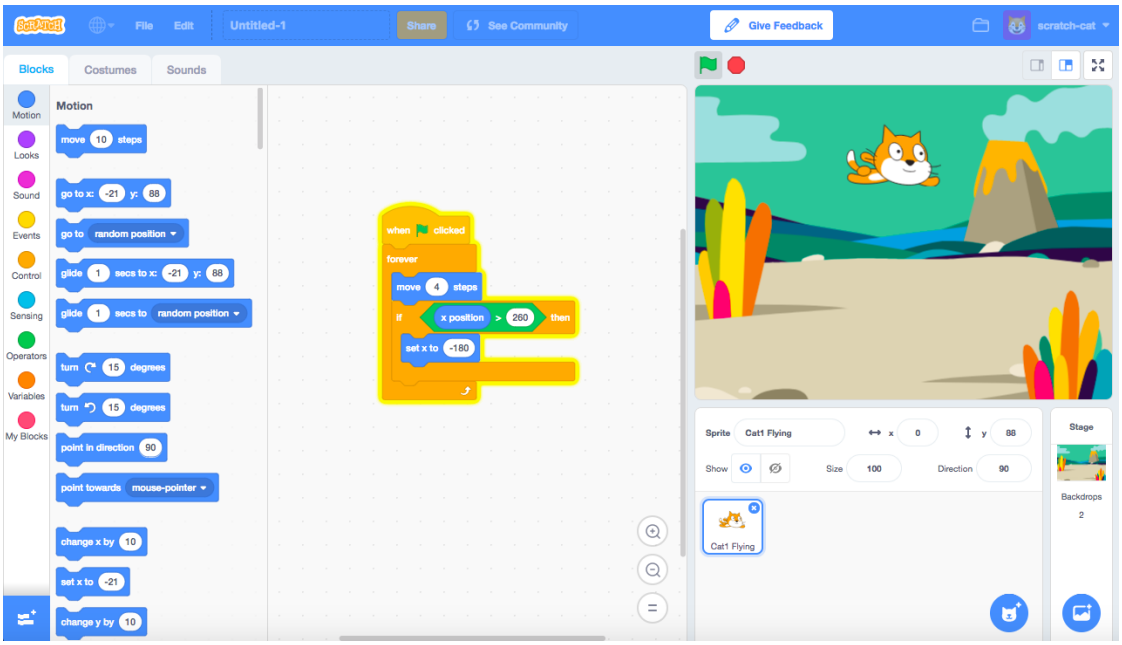

Figure 2.1: Example of a block-based language called Scratch.

Novices to text-based programming with previous experience in block-based programming make as many syntactic errors as novices who have no previous programming experience, which suggests that block-based programming simply delays this problem. However, everything cannot be learned at the same time, and research shows that novices' understanding of foundational constructs, such as loops and conditionals, is better in block-based languages. One advantage of teaching novices how to program in a block-based language is, therefore, that they can focus on learning foundational constructs and leave aside the syntactic complexity of a text-based programming language [4].

In one study [10] researchers focused on misconceptions about loops when using the block-based language Scratch and the two text-based languages Logo and Python. The study found that using a block-based language minimized misconceptions about loops as compared to using text-based languages. This

difference became more apparent when the tasks became more complex and students were required to use nested loops. When using Python, for instance, most students did not recognize nested loops, and instead recognized two independent loops. This was addressed to Python syntax being confusing to students, since all instructions within a nested loop have double indentations.

The same study also found justification for using block-based programming to teach novices to code by programming games. One justification for this, which is in line with the previously mentioned study, is that block-based languages remove syntax problems, which allows students to focus on using programming constructs and develop algorithms. Another justification for using a block-based language to teach novices to code by programming games is that younger students are in a pre-formal phase of cognitive development, and programming games in a block-based language gives them a concrete experience which facilitates learning and minimizes misconceptions when learning hard programming concepts. The main conclusion of the study is that programming games by using a block-based language facilitates the process of learning how to code.

2.3 Transitioning from block-based to text-based programming

At one point in their education, students need to make a transition to text-based programming if they want to pursue more advanced CS courses. Block-based programming is mostly used as an introductory tool, whereas text-based programming is being introduced to students in their teenage years [8].

This transition, however, can be difficult. Transitioning to text-based programming means students have to learn how to use an IDE and various editing and compiling tools, learn the syntax of a text-based language, debug syntax errors and other forms of errors, and translate their knowledge of programming from blocks to writing text commands. Many students get discouraged when their code does not compile, and draw the conclusion that they are inadequate programmers. This is exacerbated by the perception that block-based languages are aimed at younger audiences, so when text-based programming proved to be more difficult,

many students concluded that their success with block-based programming had not been “real programming”, and instead had just been fooling around with a toy environment. When making the transition, students also pay very little attention to syntax, thinking that it is acceptable to hand in work that does not compile. Few recognize the importance of precision when writing expressions. The transition to text-based programming is especially difficult for weaker students, many of whom lose confidence in their programming ability [11].

2.4 One-way transition environments

Given the difficulties associated with making the transition to text-based programming, previous research has investigated the ways in which block-based languages support learners when making this transition. As mentioned in the introduction, one study [9] found that there are four ways in which block-based languages do this. One way is simply that many block-based languages are only block-based and do not provide any support when making the transition. They analyzed 46 block-based environments, and found that 24 of them fell under the category of block-only environments.

The three other ways of supporting the transition to text-based programming were placed in the categories one-way transition environments, dual-modality environments and hybrid environments. One-way transition environments were divided into read-only environments and editable one-way transition environments. In one-way transition environments, block-based presentations of a program are transformed into a text-based presentation, but learners do not have the possibility to transform a text-based presentation into a block-based presentation. In read-only environments, learners can see the text-based version of the block-based program they have created but cannot edit the text commands. One example of a read-only one-way transition environment is Blockly, seen in Figure 2.2, which is the one-way transition environment we are studying in this thesis.

In editable one-way transition environments learners can transform their block-based program to text form, but do not have the possibility to convert it back to a block program. Once the transformation has been made, the two forms of the

Figure 2.2: Image of Blockly. Block commands are seen on the left, and autogenerated text commands are seen on the right.

program are no longer linked [9].

2.5 Dual-modality environments

In dual-modality environments, learners have the possibility to edit their program with block commands and text commands. Unlike one-way transition environments, authoring text commands updates the block-based version of the program. In the aforementioned study [9], dual-modality environments were divided into the categories shared-view and distinct-view environments. In shared-view environments, the block-based and text-based version of the program are presented side-by-side, with updates to one version of the program resulting in the corresponding version of the other. One such environment is BlockPy, showed in Figure 2.3 which is the dual-modality environment we will compare with the one-way transition environment Blockly.

Distinct-view environments are the same as shared-view environments, with the difference that only one interface is shown at a time. The switch from one interface to the other is often accompanied by an animation that “morphs” one modality to the other, reinforcing the isomorphism between the two modalities. This form of a dual-modality environment is not going to be studied in this thesis.

Figure 2.3: Image of BlockPy. Block commands are seen on the left, and editable text commands are seen on the right.

2.6 Motivational factors

Four motivational factors are commonly used when looking at children's intention to attend a similar activity in the future. These are: satisfaction, enjoyment, easiness and usefulness [2]. In this study we will measure participants' level of satisfaction, enjoyment and easiness when solving a problem in Blockly and BlockPy. As mentioned in the introduction, these factors are measured because they could give some indication as to which modality increases motivation and the intention to attend similar activities in the future, which we view as an indication of learners' interest in continuing their computer science education.

2.7 Blockly Games

Blockly games is an open source project developed by Google, which utilizes Google's own block-based programming language "Blockly". Blockly Games consists of a collection of mini-courses of programming challenges using block-based programming (Blockly). The challenges have different themes but are all visual and has a game-like nature. Some examples of activities are programming a character that traverses a maze, seen in Figure 2.4, and programming a bird that needs to eat a worm and then return to its' nest. One of the last courses is called Pond Tutor, seen in Figure 2.5, which is where text-based programming is introduced. The Pond Tutor challenges is the part of Blockly Games that will be used and looked into in this project due to its' unique combination of block-based

and text-based programming.

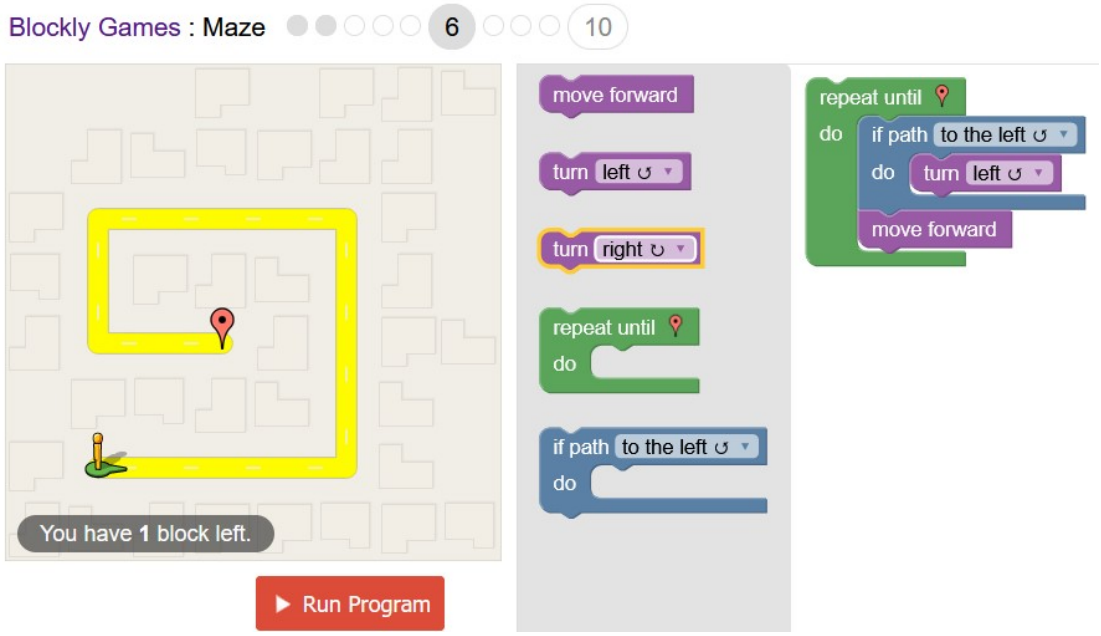

Figure 2.4: The Maze Activity in Blockly Games

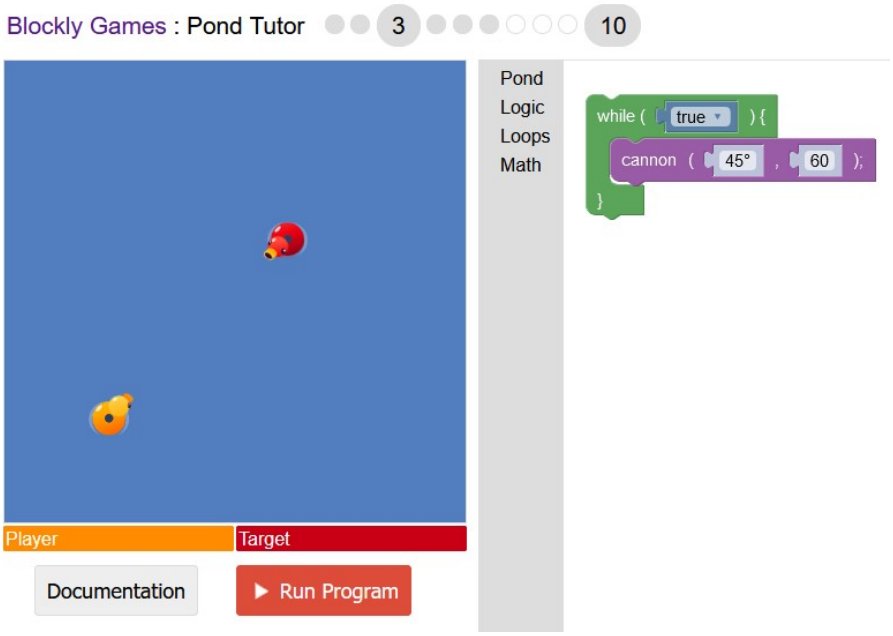

Figure 2.5: Pond Tutor in Blockly Games

3 Method

The research in this study consists of a quantitative and exploratory part. The first part of this section describes the experiment that was conducted and the statistical analyses that were performed. After this, we describe how we found and built our own learning activity in the open-source project Blockly Games.

3.1 Experiment

In the experiment, a comparison between Blockly and BlockPy was made. Blockly is a read-only one-way transition environment, whereas BlockPy is a shared-view dual-modality environment, as explained in the background.

Ten research participants where gathered by asking people sitting in computer room *Spelhallen* at KTH if they wanted to be a part of our bachelor thesis. All data was gathered in the span of one day. Each participant was sat in front of a computer. We then explained that their task was to solve a problem, which they would solve in both Blockly and BlockPy. We explained how to use Blockly and BlockPy by showing that blocks could be dragged on to a white canvas and that other blocks could be attached to it, and also showed that text commands could be written in BlockPy. The participants were then given the following instructions:

- Initialize a variable called “age” and assign it an integer value between 1-100.
- Use at least one “if”-statement to print “Under 30” if the variable “age” has a value under 30, and print “Not under 30” if the variable “age” is not under 30.

The participants were then left to their own devices when solving the problem. In order to minimize the effect of confounding variables, half of the participants solved the problem in Blockly first before solving the problem in BlockPy, and the other half solved the problem in BlockPy before solving it in Blockly.

Once the participants had solved the problem in both Blockly and BlockPy, they got to answer survey questions. Google Forms was used to create the survey since it is cheap, easy to use and automatically generates graphs of the responses. The

age of the participants spanned from 18 to 27, and nine of the ten participants had taken more than one programming course whereas one participant has only taken one. The participants were then asked to answer the question of which modality (Blockly or BlockPy) they thought would help them learn Python programming the most. The reason BlockPy was picked as a shared-view dual-modality environment is because it supports the transition to Python, which is an appropriate text-based language to teach highschoolers [13]. They were also asked to provide a written explanation as to why they chose that particular modality. After this, they were asked to rate the level of satisfaction and enjoyment they experienced when solving the problem in Blockly and BlockPy, and they were also asked to rate how easy it was to solve the problem in each modality. A five-level Likert scale [1] was used to measure satisfaction, enjoyment and easiness, where 1 represented low levels of satisfaction, enjoyment and easiness, and 5 represented very high levels of satisfaction, enjoyment and easiness.

In order to measure whether there were any significant differences in satisfaction, enjoyment and easiness when solving the problem in Blockly and BlockPy, a t-test was performed in Google Sheets. The formula T.TEST was used [14]. It takes four parameters. The first two take the data that is used to perform the t-test. We used the answers to the participants level of satisfaction when solving the problem in Blockly and compared these to the level of satisfaction when solving the problem in BlockPy. Likewise for enjoyment and easiness. The third parameter specifies the number of distribution tails. Since we assume that our data is normally distributed, our t-test used a two-tailed distribution [6]. Also, we assume that the variance of the measured variables are not extremely different, so a two-sample equal variance test was performed. We view a p-value of less than 0.05 to be statistically significant, indicating a significant difference in satisfaction, enjoyment or easiness, and a value that is greater than 0.05 to be statistically insignificant.

3.2 Developing a learning activity in Blockly Games

Blockly games is an open source project which we found by using Google and forked from GitHub. The project is written in JavaScript, HTML and CSS [3].

Only the pond tutor part of Blockly games was used and changed in this project. We created our own version of the pond tutor by changing the source code, and changed the following parts of pond tutor:

- Start positions for each duck.
- Health of ducks.
- Functions for the movements of enemy ducks.
- Help instructions which are displayed before each challenge.
- Availability of code blocks for each challenge.

4 Results

In this section, we first describe participants answer to the question of which modality they think is most useful when trying to learn a text-based language. We then show the participants answers to the level of satisfaction, enjoyment and easiness they experienced when solving the problems in Blockly and BlockPy, and describe the results of the t-tests. Lastly, we describe how Blockly Games differs from other ways of helping learners transition to text-based languages, and describe a learning activity that was built in the open-source repository.

4.1 Learning a text-based language

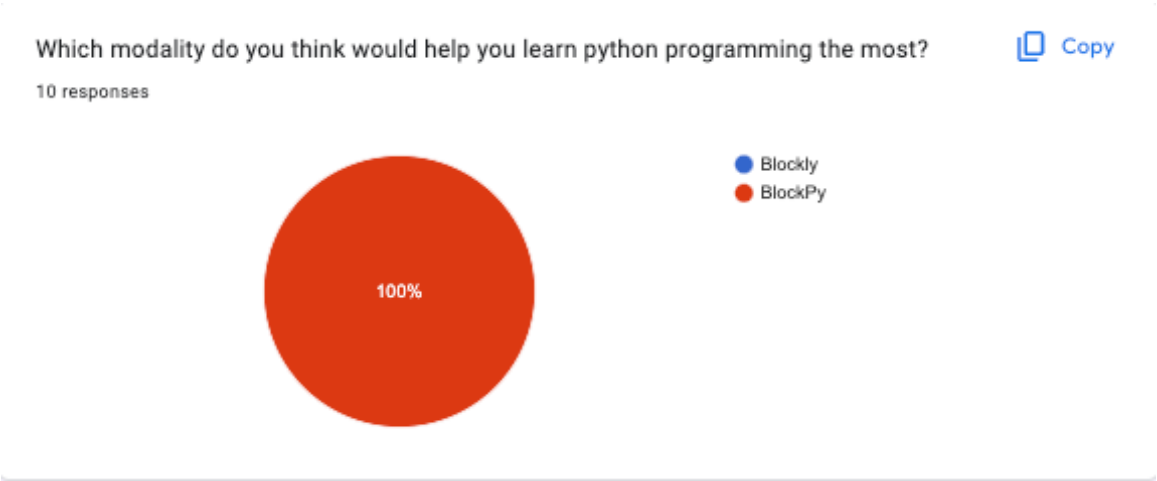

Figure 4.1: Percentage of participants that thought BlockPy is better than Blockly for learning python programming.

As seen in Figure 4.1, when answering the question of which modality is better for learning python programming, all the participants thought that BlockPy, a dual-modality environment, is more effective than Blockly, a one-way transition environment.

One participants said that *"Making changes in the code changed the blocks which is helpful when playing around and learning by reverse engineering"*. One reason for using a dual-modality environment over a one-way transition environment, therefore, might be that it is easier to play around and learn text-based programming by reverse engineering. Similarly, another participant said that BlockPy *"lets user explore both actually writing code and experiment*

with how the "block logic" might translate into actual python code.". A dual-modality environment could potentially be more effective than one-way transition environments when learning how to translate knowledge from block-based programming into text-based programming. Another participant thought that BlockPy is better for learning python syntax, writing *"Because I can edit code, and at the same time learn the syntax of Python."*.

4.2 Satisfaction

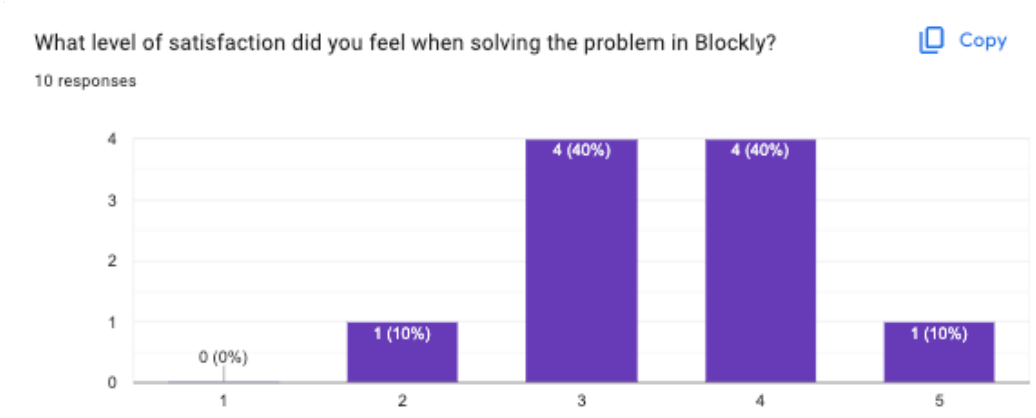

Figure 4.2: Level of satisfaction when solving a problem in Blockly.

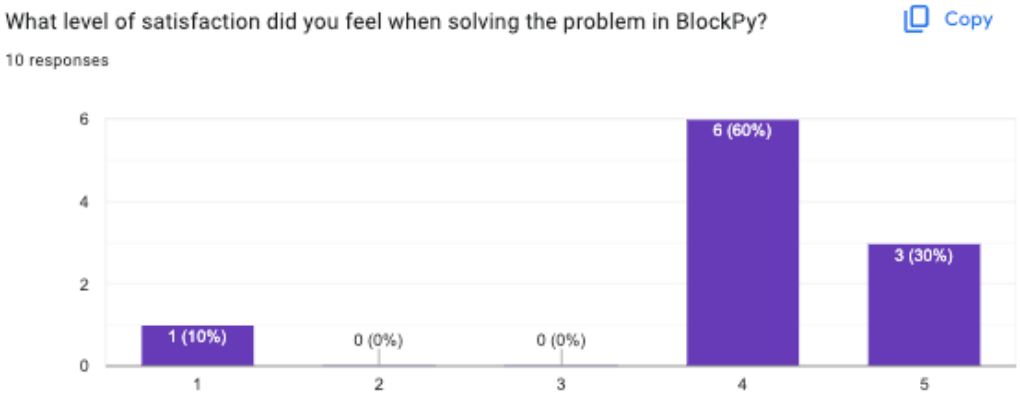

Figure 4.3: Level of satisfaction when solving a problem in BlockPy.

Figure 4.2 and Figure 4.3 show the participants grading of the satisfaction motivational factor. The t-test yielded a p-value of 0.28, meaning the result is

statistically insignificant and we have no evidence for any difference in the level of satisfaction when solving the problem in Blockly or BlockPy.

4.3 Enjoyment

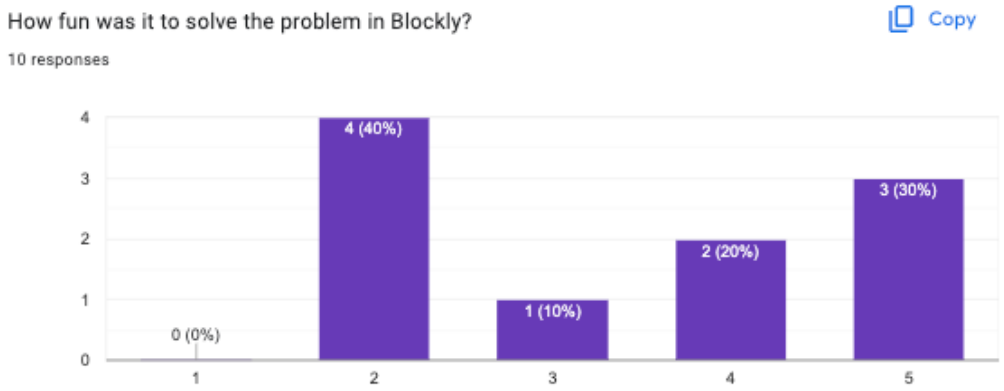

Figure 4.4: Level of enjoyment when solving a problem in Blockly.

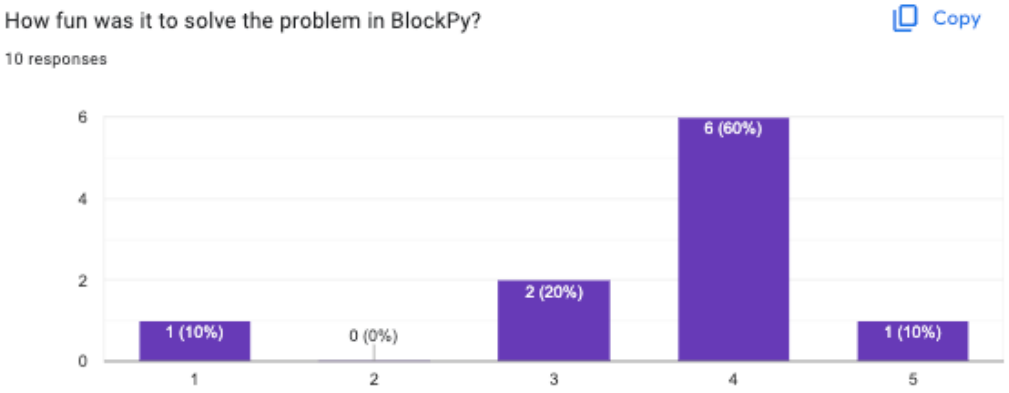

Figure 4.5: Level of enjoyment when solving a problem in BlockPy.

Figure 4.4 and Figure 4.5 show the participants grading of the enjoyment motivational factor. The t-test yielded a p-value of 0.72, meaning the result is statistically insignificant and we have no evidence for any difference in the level of enjoyment when solving the problem in Blockly or BlockPy.

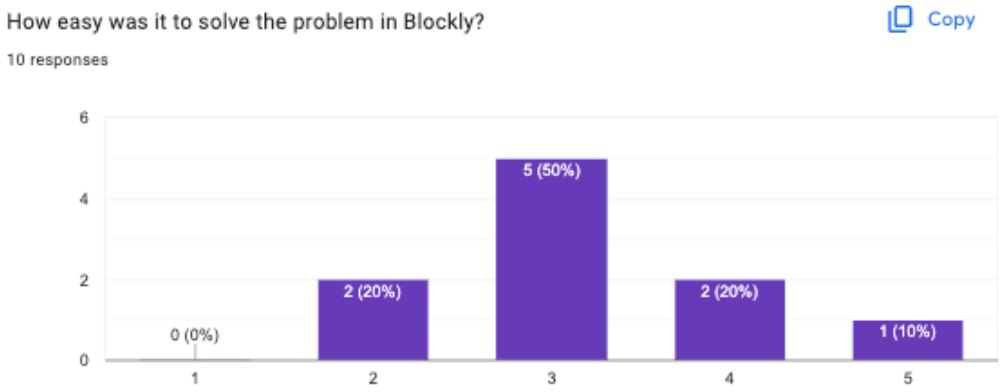

Figure 4.6: Level of easiness when solving a problem in Blockly.

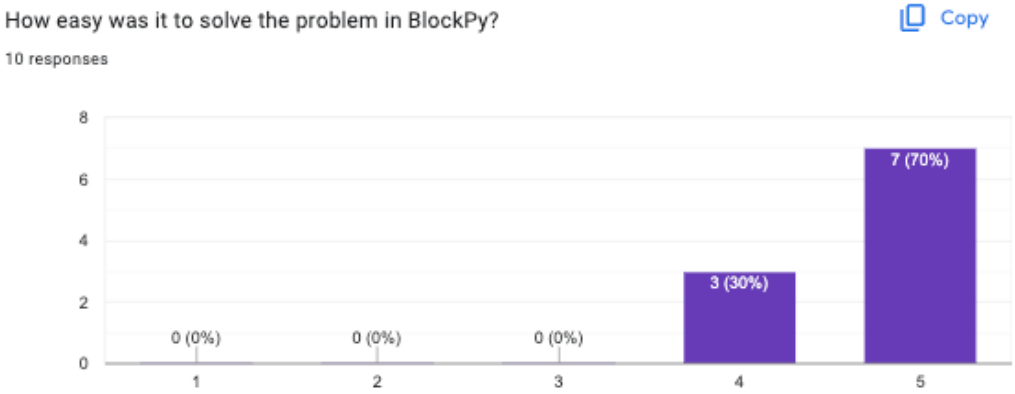

Figure 4.7: Level of easiness when solving a problem in BlockPy.

4.4 Easiness

Figure 4.6 and Figure 4.7 show the participants grading of the easiness motivational factor. The t-test yielded a p-value of 0.00024. This means that there is evidence that there is a difference in the level of easiness when solving the problem in Blockly or BlockPy.

4.5 A learning activity in Blockly Games

The learning activity consists of ten levels. In the first level, the user has to solve a problem with block-based programming. When moving on to the next level, the user has to solve the exact same problem with text-based programming. This is the same for all ten levels. An example of this is seen in Figure 4.8.

In level one and two, the user learns how to use the cannon command to hit a target a cannon ball. The user has to find the right angle and range to hit the target successfully. In level three and four, the user has to do the same thing as in the first two levels, but the coordinates that the cannons start in (and hence the angle and range needed to hit the target) is different. In levels five and six, the user has to do the same thing as in the first levels, but in addition to this the user has to use a while loop to hit the target repeatedly since it has more health. In levels seven and eight, the target moves vertically. The user has to do the same thing as in level five and six, but has to use a function called swim to swim alongside the target while hitting it repeatedly with cannon balls. In levels nine and ten, the target is too far away, so the user has to swim closer and then fire cannon balls when the user is close enough. This introduces the use of conditional logic since an if-statement can be used to solve the problem.

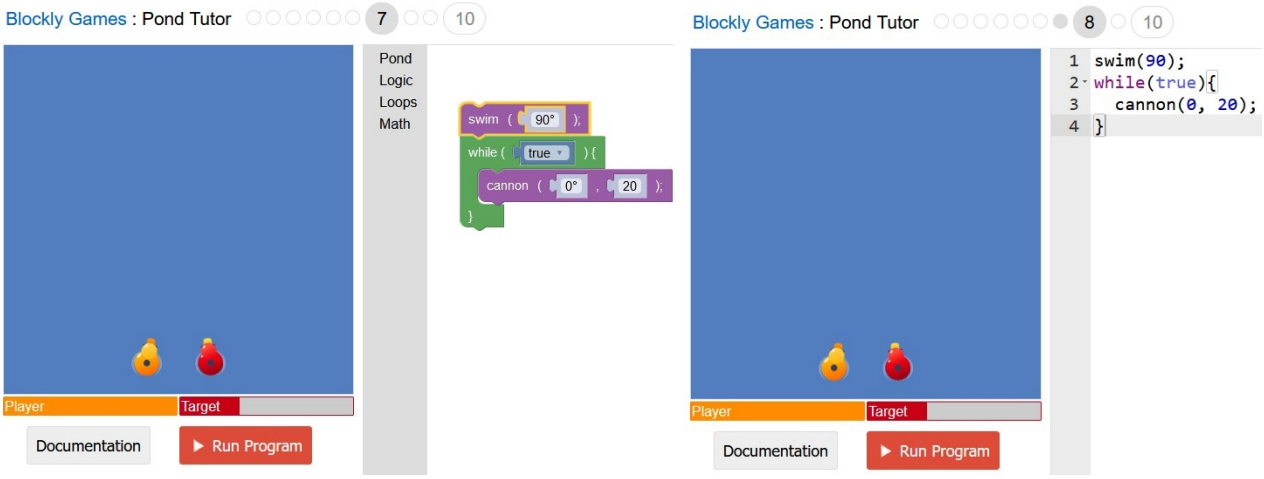

Figure 4.8: The same problem has to be solved twice. First with block-based programming (Blockly) and then with text-based programming (JavaScript).

4.6 How Blockly Games is different from other ways of transitioning to text-based languages

Blockly Games is similar to one-way transition environments, in that when a problem is solved with block commands, the corresponding text-based program is shown before progressing to the next level, as seen in Figure 4.9. The way Blockly Games differentiates itself from the other forms of supporting the transition to text-based programming, is that the user has to solve the same problem that was

solved with block commands from scratch with text-based programming.

Figure 4.9: After solving the problem with block-based programming the corresponding text-based code of the solution is shown.

5 Discussion

In this section, we will discuss the results and give suggestions for future work, and we will also discuss limitations with this study.

5.1 Discussion of the results from the experiment and implications for future work

One purpose of this study was to compare one-way transitions to dual-modality environments with regards to learning a text-based language, and how the two modalities differ with regards to the motivational factors satisfaction, enjoyment and easiness. Our results show that all of the participants believe that the dual-modality environment BlockPy is better for learning python (a text-based language) than the one-way transition environment Blockly. Our results also show no significant difference in the motivational factors satisfaction and enjoyment when solving a problem in Blockly and BlockPy, but that solving problems is significantly easier in BlockPy.

It may be unsurprising that a modality that has everything a one-way transition environment has, but that also allows learners to write text commands which results in a change in the corresponding block-based program, is seen as more effective for learning a text-based language. As one of the participants wrote, BlockPy allows the learner to edit code and at the same time learn the syntax of the text-based language (python in our case). It also allows learners to experiment with how block logic translates into actual python code, as another participant noted. More research into the differences between one-way, dual-modality and hybrid environments is needed, but our results suggest that a dual-modality environment might be preferable to a one-way transition environment when helping learners make the transition from block-based to text-based programming.

On another note, investigating how to help learners transition from block-based to text-based programming in the best way is an open problem [9], and it might be the case that simply going directly from a block-based to text-based language is more effective than using one-way, dual-modality or hybrid environments.

Future research could do a comparison between one group of students that are taught python directly, and another group that uses a dual-modality environment, and see which group has the best performance on a test that assesses python programming at the end of the study. In other words, it could be the case that the assumption that block-based languages need to provide features that help students make the transition to text-based programming is false. It might instead be more effective to teach a text-based program directly.

With regards to the motivational factors, it is unsurprising that the participants in this study found it significantly easier to solve the problem in BlockPy, which allowed them to use python. The participants had previous programming experience, and presumably little experience with block-based programming, which might be the reason for this difference in easiness. We believe the results still suggest that dual-modality environments are a better alternative than one-way transition environments, but future research should use learners that already know block-based programming and are transitioning to text-based programming. Future work could, for example, test highschoolers that have previous experience with block-based programming. Studying differences in motivational factors is important because, as explained in the background, they measure the intention to participate in similar activities in the future and could thus be a way of measuring the likelihood that students will continue their journey in computer science.

5.2 Discussion of Blockly Games and implications for future work

The second question we aimed at answering in this study was whether there is an alternative to one-way transition, dual-modality and hybrid environments when helping learners transition from block-based to text-based programming, and if so, how a learning activity in this alternative solution could look like. We found that Blockly Games differs from one-way transition, dual-modality and hybrid environments. It is like Blockly, a one-way transition environment, in that when one has solved a problem, the corresponding text-based program is shown. But, it adds the feature of having to solve the same problem solely using text-based

programming. This presumably forces the learner to learn how to code using text-commands. Future research could investigate if this is an effective way of helping students with previous experience in block-based programming to learn text-based programming.

Interestingly, one study mentioned in the background [10] argued that there is justification for using block-based programming to teach novices how to code by programming games. Since younger students are in a pre-formal phase of cognitive development, teaching them how to code a game gives them a concrete experience and minimizes misconceptions when learning hard programming concepts. This suggests that Blockly Games could be an effective way of teaching younger students how to code in a text-based language, since they would be using text-based programming to code a game which would make it less abstract and more concrete.

5.3 Limitations

One limitation of this study is that the sample size is small. Small sample sizes are linked to the replication crisis in science [15] and larger sample sizes are desirable. This study can serve as an initial exploration of the differences between one-way and dual-modality environments, but future work should use a larger sample size. Another limitation is that the participants in this study are not entirely representative of the learners that need to make a transition from block-based to text-based programming. As mentioned in the previous section, future research could study highschoolers who are transitioning from block-based to text-based programming as this would give more useful information about the group that would most benefit from having an effective way of transitioning from block-based to text-based programming.

6 Conclusion

This thesis investigated how one-way transition environments compare to dual-modality environments with regards to learning a text-based language, and how the two modalities differ with regards to the motivational factors satisfaction, enjoyment and easiness. The results show that BlockPy, a dual-modality environment, is seen as a better alternative than the one-way transition environment Blockly with regards to learning a text-based language, and that it might be easier to solve problems in dual-modality environments which could increase motivation for learning programming. Further research should investigate the differences between one-way, dual-modality and hybrid environments to gain a deeper understanding of which modality is the best alternative when helping students transition to text-based programming. Future research could also investigate if it is more effective to go straight from block-based to text-based programming instead of using a tool that eases the transition. Importantly, future research should use larger sample sizes and a more representative sample, since these are two major limitations with this study.

This study also investigated if there is an alternative to one-way transition, dual-modality and hybrid environments when helping learners transition from block-based to text-based programming, and aimed at creating a learning activity in this alternative solution. It found that Blockly Games differs from said environments, and showed how a learning activity in Blockly Games could look like. Future research could investigate if the type of learning activities that we developed in Blockly Games is an effective way of helping learners transition to text-based programming.

References

[1] Contributors, Wikipedia. *Likert scale*. 2023. URL: https://en.wikipedia.org/wiki/Likert_scale.

[2] Giannakos, Michail N. and Jaccheri, Letizia. "What Motivates Children to Become Creators of Digital Enriched Artifacts?" In: *Proceedings of the 9th ACM Conference on Creativity Cognition*. CC '13. Sydney, Australia: Association for Computing Machinery, 2013, pp. 104–113. ISBN: 9781450321501. DOI: 10.1145/2466627.2466634. URL: https://doi.org/10.1145/2466627.2466634.

[3] GitHub. URL: https://github.com/google/blockly-games.

[4] Gomez, Marcos J., Moresi, Marco, and Benotti, Luciana. "Text-Based Programming in Elementary School: A Comparative Study of Programming Abilities in Children with and without Block-Based Experience". In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '19. Aberdeen, Scotland Uk: Association for Computing Machinery, 2019, pp. 402–408. ISBN: 9781450368957. DOI: 10.1145/3304221.3319734. URL: https://doi.org/10.1145/3304221.3319734.

[5] Grover, Shuchi and Basu, Satabdi. "Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic". In: *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '17. Seattle, Washington, USA: Association for Computing Machinery, 2017, pp. 267–272. ISBN: 9781450346986. DOI: 10.1145/3017680.3017723. URL: https://doi.org/10.1145/3017680.3017723.

[6] Hayes, Adam. "What Is a Two-Tailed Test? Definition and Example". In: (2022). URL: https://www.investopedia.com/terms/t/two-tailed-test.asp.

[7] "IT-Kompetensbristen". In: *ITTELEKOMFÖRETAGEN* (2020). URL: https://www.almega.se/app/uploads/sites/2/2020/12/ittelekomforetagen-it-kompetensbristen-2020-online-version.pdf.

[8] Kolling, Michael, Brown, Neil C. C., and Altadmri, Amjad. “Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming”. In: *Proceedings of the Workshop in Primary and Secondary Computing Education*. WiPSCE ’15. London, United Kingdom: Association for Computing Machinery, 2015, pp. 29–38. ISBN: 9781450337533. DOI: 10.1145/2818314.2818331. URL: https://doi.org/10.1145/2818314.2818331.

[9] Lin, Yuhan and Weintrop, David. “The landscape of Block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming”. In: *Journal of Computer Languages* 67 (2021), p. 101075. ISSN: 2590-1184. DOI: https://doi.org/10.1016/j.cola.2021.101075. URL: https://www.sciencedirect.com/science/article/pii/S259011842100054X.

[10] Mladenovic, Monika, Boljat, Ivica, and Zanko, Zana. “Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level”. In: *Education and Information Technologies* 23 (2018), pp. 1483–1500. ISSN: 1573-7608. DOI: https://doi.org/10.1007/s10639-017-9673-3. URL: https://link.springer.com/article/10.1007/s10639-017-9673-3#citeas.

[11] Powers, Kris, Ecott, Stacey, and Hirshfield, Leanne M. “Through the Looking Glass: Teaching CS0 with Alice”. In: *SIGCSE Bull.* 39.1 (Mar. 2007), pp. 213–217. ISSN: 0097-8418. DOI: 10.1145/1227504.1227386. URL: https://doi.org/10.1145/1227504.1227386.

[12] Schönborn, Konrad and Höst, Gunnar. “Många utmaningar när programmering integreras i undervisningen”. In: (2022). URL: https://www.skolverket.se/skolutveckling/forskning-och-utvarderingar/artiklar-om-forskning/manga-utmaningar-nar-programmering-integreras-i-undervisningen.

[13] Seraliduo,
Eleni and Douligeris, Christos. “Investigating the Transition from Block-based to Text-based Programming Techniques in Secondary Education in

Greece". In: *European Journal of Engineering and Technology Research* (2022). DOI: http://dx.doi.org/10.24018/ejeng.2021.0.CIE.2753.

[14] Support, Google. "T.TEST". In: (2023). URL: https://support.google.com/docs/answer/6055837?hl=en.

[15] Turner, B.O., Paul, E.J., and Miller, M-B. et al. "Small sample sizes reduce the replicability of task-based fMRI studies". In: *Communications Biology* 1 (2018). DOI: https://doi.org/10.1038/s42003-018-0073-z.

[16] Weintrop, David and Wilensky, Uri. "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms". In: *ACM Trans. Comput. Educ.* 18.1 (Oct. 2017). DOI: 10.1145/3089799. URL: https://doi.org/10.1145/3089799.

Appendix

A Blockly Games

The edited version of the Blockly Games Pond Tutor can be found at: https://github.com/Edvinnordling/blockly-games-KEX

TRITA-EECS-EX-2023:279