



Universiteit
Leiden
The Netherlands

Computer Science

A cross-case analysis of the integration of computational thinking
in the English subject

Yunus İşleyen

Supervisors:

Sabiha Yeni, Anna van der Meulen & Xiaohua Jia

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

www.liacs.leidenuniv.nl

June 24, 2022

Abstract

Computational thinking is a set of skills that can be used to solve problems. It is also being integrated in education as a more effective way to teach students. The aim of this thesis is to analyse two case-studies where computational thinking was integrated in the English subject at a secondary school in two consecutive years.

In this thesis four dimensions are assessed. First and foremost the English learning outcomes of the students, since that is the actual subject. Secondly, the learning outcomes of computational thinking, how well did the students grasp the ideas of computational thinking. The last two dimensions are the students' attitudes towards this integration of computational thinking and the teacher's attitude.

The results show interesting differences, with as a most valuable finding the improvements in the second year in terms of the learning outcomes. The end projects of the students were more advanced and diverse in the second year. Even though the students found this integration difficult in both years, the teacher was very impressed with the (effects of the) integration, especially in the second year. Overall, this study strengthens the idea that computational thinking can be used as an effective way to teach.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem statement	1
1.3	Purpose	2
1.4	Overview	3
2	Literature review	4
2.1	Computational thinking	4
2.2	Integration of computational thinking in education	5
2.3	Digital storytelling	7
2.3.1	Digital storytelling in education	7
2.3.2	Connection with computational thinking	7
3	Methodology	8
3.1	Research design	8
3.2	Lesson design	8
3.3	Participants	10
3.4	Data collection	10
3.5	Data analysis	11
4	Results	12
4.1	Scratch projects	12
4.1.1	English learning outcomes of students	12
4.1.2	Computational thinking learning outcomes of the students	18
4.2	Students' attitude	25
4.2.1	Exit tickets	25
4.2.2	Students interviews	26
4.3	Teacher's attitude	28
4.3.1	Learning objectives	28
4.3.2	Students' understanding from the teacher's point of view	29
4.3.3	Instructional strategies	30
4.3.4	Assessment	30
4.3.5	Teacher's opinion and suggestions	31
5	Conclusions	33
5.1	Limitations, future direction and conclusion	35
	References	39

1 Introduction

1.1 Background

The modern world has become abundant with technology and automated processes, these require a logical and structured way of thinking. Recognising patterns, decomposing and abstracting problems, among others, are critical aspects of problem solving capabilities [AH21]. Problem solving capabilities and crucial thinking, which are closely related, are critical skills that we must have today and in the future [LC18]. The overarching term for this way of thinking is also known as computational thinking.

Computational thinking is a relatively new term, loosely meaning to solve problems like a computer would, though there exists other definitions. Computational thinking was first mentioned by Papert in his book *Mindstorms: Children, Computers, and Powerful Ideas* [Pap80] and has been becoming more widespread after 2006 by Wing [Win06]. The digitalization has seen a massive increase since the twenty-first century and does not seem to be slowing down any time soon. So it is only logical that computational thinking is becoming more of a trend.

Computational thinking is already being integrated in the curricula of primary school [CTHE19] and secondary school [ABBH10]. A lot of experiments and research is being done for this integration [TCT20], to such an extent that, according to Timmer and Tolboom, computational thinking will be fully integrated in the curriculum of mathematics within ten years [TT19]. We are already being called the digital society [MG06], which keeps proving true with new or rising trends like working from home due to the coronavirus pandemic and smart devices replacing almost every type of device.

A lot of study has already been done on integrating computational thinking in the curricula of STEM (science, technology, engineering and mathematics), but less so in other subjects, like languages. The reason is that, at first sight at least, there seems to be no connection between computational thinking and learning a language. But even languages consist of rules, like grammar rules and spelling [SDMJ18]. Wing thinks that computational thinking should be an essential skill for everyone, next to reading, writing and arithmetic [Win06]. And though the academic world seems to agree on this, based on the amount of research that is done on computational thinking, there is still no clear consensus on how to integrate it ([Gon15], [BR12]).

1.2 Problem statement

Due to the benefits of computational thinking in terms of problem solving, it is a very useful skill to have in the 21st century [MM16]. But the focus remains, as stated above, on designing a framework for teachers to integrate computational thinking in their subjects. This is also the case for the education in The Netherlands [VFG+15]. It is important to note that we want the students to use computational thinking not only during programming and other computer-related subjects, which is where most of the focus of this integration goes to, but everywhere. Consequently the goal is to integrate it in all applicable subjects. The reason is that computational thinking can help students even with language learning, as it can, for example, help them structure sentences [PCWH21].

Because computational thinking is a method to solve problems, students will still have the same subject learning objectives as before, so only the way they analyse and solve the problems will change. As teachers, in order to be able to teach and guide the students through the use of computational thinking, they also need to acquire the skills themselves [SKM+21]. The integration of computational thinking is also different per subject, so there is no one right way to integrate it. According to Guzdial [Guz08], in order to pave the way for computational thinking for everyone, we need to understand how non-computer scientists (teachers and students in this case) understand computing first. Then it is possible to find an effective way for teachers to integrate computational thinking in their subjects.

1.3 Purpose

The aim of this study is to find out how effective the integration of computational thinking is through the analysis of two data sets from different studies acquired from the teacher and this students through questionnaires and interviews. Also, this study is part of a larger research project that aims to integrate computational thinking into the Dutch curricula of different subjects. This research project is a collaboration between Leiden University and Radboud University. It is funded by the NRO (The Netherlands Initiative for Education Research) and started in 2018.

Computational thinking is a skill that has already been used in certain fields for years, mostly in the research and science world (and of course in the computer science fields) [BB19]. Consequently, this research project has an emphasis on developing guidelines and methods for teachers to be able to integrate computational thinking, introducing this concept to the future generation at a much earlier age. The research is done by means of practical experiments and data collection. A lot of experiments within this research project have already been done on different primary and secondary schools ([Nad21], [DB21b], [DB21a], [Har21]). The experiments can have different assignments for the students, but are all designed for the same goal. With the use of the PCK (pedagogical content knowledge) acquired from the teachers during interviews, this project researches how to best teach using computational thinking. With the teachers' PCKs, the integration of computational thinking can be viewed from the teacher's perspective. With that analysis a design principle can be put together to have a general basis that can be used by all teachers in the future to integrate computational thinking.

The experiments and data and that are relevant for this research are two projects that were done in subsequent years, both projects integrated computational thinking in the subject English. Students from both projects had a digital storytelling assignment. This is an effective approach to language learning, because it gives the students the opportunity to integrate technology with storytelling, allowing them to represent their understanding and combining their stories with images and verbal content. This method of language learning can increase the students' performance and motivation [PCWH21]. These assignments were done in Scratch, which is a (visual) programming language aimed at children to learn the basics of programming. After the experiments, data was collected from the teacher and students by means of interviews and questionnaires. The comparison will be done by analysing these two sets of data. The purpose of this research is to compare the data of those two projects through the use of a cross-case analysis, i.e. find similarities and differences between them to better understand the integration.

To be able to identify and analyse the similarities and differences, the following question will stand central in this research:

What are the differences between two computational thinking integrated English lessons?

With the following research questions:

1. How do the English learning outcomes of the students differ in two computational thinking integrated English lessons?
2. How do the computational thinking learning outcomes of the students differ in two computational thinking integrated English lessons?
3. How do the attitudes of the students differ in two computational thinking integrated English lessons?
4. How does the attitude of the teacher differ in two computational thinking integrated English lessons?

1.4 Overview

After this introduction, there will be a literature review (section 2), where previous studies will be explored. After that, in the methodology (section 3), the design and methods used in this thesis will be explained. The fourth chapter (section 4) will go over the data and describe them thoroughly. In the last chapter (section 5), it will be concluded by using the results from the previous chapter to provide answers to the research questions.

This study is a bachelor thesis made in cooperation with Sabiha Yeni and Anna van der Meulen as my first supervisors and Xiaohua Jia as my second supervisor from Leiden Institute of Advanced Computer Science (LIACS).

2 Literature review

In this chapter the following relevant topics will be reviewed: What is computational thinking and what are its characteristics. What are the effects of integrating computational thinking in education and more specific in languages, and what is digital storytelling and how can it be used to implement the integration of computational thinking.

2.1 Computational thinking

As stated shortly in the introduction, the term *computational thinking* was mentioned first by Papert in his book *Mindstorms: Children, Computers, and Powerful Ideas* [Pap80]. While computational thinking was mostly just another term for "algorithmizing", it was Wing who really made computational thinking an umbrella term for thought processes and techniques [LM21]. In her essay [Win06] she called computational thinking a fundamental skill for everyone, not just for computer scientists. According to her, computational thinking has the following characteristics:

- Computational thinking is not about programming, but about conceptualising. "It requires thinking at multiple levels of abstraction" [Win06].
- Computational thinking is a fundamental, not rote skill. According to Wing computational thinking is something every human must be able to do in the current society. It is not a rote skill, as in it cannot be learnt by repetition, but has to be understood and mastered in order to use it.
- Computational thinking is a way that humans think and solve problems, it is not trying to make us think like computers.
- Computational thinking complements and combines mathematical and engineering thinking. Mathematical thinking is involved because its foundations rest on mathematics whereas engineering thinking is involved because we build systems that interact with the real world.
- Computational thinking is about ideas, not just artefacts. The computational concepts that are used to approach and solve problems, not necessarily the end-products, but the process leading to it.
- Computational thinking is for everyone, everywhere. Computational thinking will only become a reality when it becomes a skill that is taught to everyone such that it disappears as an explicit philosophy.

The second definition of computational thinking is Selby and Woollard’s definition of computational thinking [SW13] from 2013. This definition incorporates all the characteristics of computational thinking for which there is a consensus. In short, “. . . computational thinking is a focused approach to problem solving, incorporating thought processes that utilise abstraction, decomposition, algorithmic design, evaluation, and generalisations”, so this definition exists out of:

- The ability to think in abstractions, hiding complexity.
- The ability to think in terms of decomposition. To break down large and/or complex problems.
- The ability to think algorithmically, a step-by-step set of instructions.
- The ability to think in terms of evaluations. Being able to evaluate and make trade-offs, think of the use of time and space.
- The ability to think in generalisations. Being able to move from specific to broader applicability.

Wing’s definition of computational thinking can be summarized as a thought process in an abstract way that helps humans to solve problems, without going too much in depth. Selby and Woollard’s definition of computational thinking, however, does go more in depth on the characteristics of the thought process. And although differences of opinion still exist, there appears to be some agreement that computational thinking refers to a thought process to solve problems at their core with abstraction.

2.2 Integration of computational thinking in education

Computational thinking is a technique that is already being taught to students on its own, especially in computer science education [LR18]. Additionally, the integration of computational thinking in the K-12 subjects is a famous topic for the exact sciences, also known as STEM (Science, technology, engineering, and mathematics) ([SDF18], [LSd+20], [WSC21]) and comes with benefits and disadvantages. According to Saidin, Khalid, Martin, Kuppusamy and Munusamy [SKM+21]: “CT has greatly benefited students in technological learning in the twenty-first century”. Increasing the students’ critical and analytical thinking skills, coming up with structured solutions for problems. Mohaghegh and McCauley [MM16] state that: “Computational terms may be more effectively understood if students are able to see them effectively demonstrated in areas they are already familiar with. . . . Students need to see computer science as more than just programming, but instead an immensely broad field and the initiation of a branch of thinking that may be used to solve many problems in numerous areas”. According to them, computational thinking enables the students to be more effective problem solvers. It also encourages them to *create* tools instead of using existing tools for their problems. Computational thinking also helps students define what can and cannot be solved. Lastly, they point out that computational thinking enables the students to understand *how* to construct solutions and why some solutions are more suitable and effective than others. This is exactly the evaluation dimension of computational thinking, as defined in the previous section.

However, Saidin et al. [SKM⁺21] point out that the integration of computational thinking also comes with challenges. Most of these challenges have to do with the teachers. The first one they mention is the teachers' understanding of computational thinking. The teachers do not understand the concept of computational thinking and what facilities are necessary to integrate it. The second one is the teachers' lack of confidence in using computational thinking. The teachers may, because of lack of training, have not enough confidence to integrate and use computational thinking in their courses. The third one is the teachers' lack of skills to implement computational thinking. The teachers may not understand how these skills have to be used in the classrooms. The last one is about the students' acceptance towards computational thinking. Many aspects have to be considered (age, gender, etc.) in order to integrate computational thinking effectively for the students.

This overview provides some of the benefits and disadvantages of the integration of computational thinking in education. The benefits are proof that the use of computational thinking has a positive effect on the students. The disadvantages are mostly obstacles that have to be overcome by the teachers and students. Once there is a good infrastructure that deals with the obstacles, then computational thinking can be integrated most efficiently.

Integration of computational thinking in languages

So far this section has focused on the integration of computational thinking in the STEM courses. Saidin et al. [SKM⁺21] stated that STEM teachers should have no problem implementing computational thinking in their courses, not including teachers that teach languages. This paragraph will discuss the integration of computational thinking in the language subjects. Studies done by Sabitzer et al. [SDMJ18] and Jenkins [Jen15] both tried different approaches to integrate computational thinking into a language. The study by Sabitzer et al. [SDMJ18] tried to answer the question *How and where can we introduce modelling in primary and lower secondary language education?* In this (still ongoing) study the data showed that approximately half of the students would use modelling (through different types of diagrams) again in future learning situations. So modelling techniques seem to be an effective tool in language learning. But the study also concluded that the students had difficulty with abstraction and generalisation.

Jenkins' comparative quantitative evaluation [Jen15] had one group with an approach to teach English poetry to the students using computer-based microworlds or guided discovery and another comparative group. A microworld provides a means for learners to acquire knowledge in a natural way [Pap80]. The example Papert used is the early language development of infants, where language develops naturally through immersion within that linguistic environment. Jenkins came to the following conclusion: "There are potential quantifiable gains in performance that could be achieved as a result of incorporating elements of a microworld-based pedagogy into classroom practice." So this research showed that the students who received the microworld-based intervention made more improvement in computational thinking and poetic thinking than the comparative group.

Integrating computational thinking might not be something that first comes to mind when thinking of language subjects. This might be the reason that it is (relatively) less common [SDMJ18]. But the above results show that there is a benefit, though there is much work left to be done in this field.

2.3 Digital storytelling

Having discussed computational thinking in depth, this final section addresses (one of the many) ways of how it can be integrated in language courses. Digital storytelling is the art to create stories through the use of technology. These stories are created through images, sounds and animations [Rob], taking the form of a video, song, podcast, etc. Creating a digital story combines techniques to develop literacy and storytelling skills and some basic level of understanding about information and communication technology. Group exercises and individual processes can be used to develop confidence and build self-esteem.

2.3.1 Digital storytelling in education

Digital storytelling is a medium heavily used in education, both by teachers and students. According to Alismail [Ali15] multimedia is a powerful and beneficial tool to teach students, and teachers should aim to find ways to integrate it into their curriculum. This medium incorporates higher order thinking skills into the projects, because the students must create something. Digital storytelling is also beneficial for teachers, as it can save them time and effort. Smeda et al. [SDS14] first state that digital storytelling enhances engagement, the students are (more) motivated. Secondly, the study finds that students do more work in digital environments compared to printed media, such as books. It also increased the collaboration and the level of communication. Digital storytelling also increases digital literacy. Students improve their technical skills, next to the subject-related skills. Another finding is that digital storytelling allows for greater diversity by personalising students' experience. This can in turn help them with their confidence, which again enhances their social and psychological skills. And lastly, digital storytelling is suitable for a constructive approach to learning. Students work on their own digital stories after receiving basic instructions from the teacher. They can create their digital stories in their own way and with their own approach, based on the difficulty, use of (re)sources and their experience.

2.3.2 Connection with computational thinking

This section so far has discussed digital storytelling and its use in education. It is now necessary to explain the relationship between digital storytelling and computational thinking. A study by Parsazadeh et al. [PCWH21] compared two groups of students which were taught using different instructional strategies. One group, the experimental group, was taught using computational thinking through digital storytelling. These students were asked to decompose the problem into smaller, more manageable parts. They were also asked to recognize patterns (pattern recognition), a repeated process to get information. Then they were asked to use abstraction, whereby a process was generalised to be used in other cases. Lastly, they were asked to think algorithmically, where the students had to put the digital story together in a flowchart before implementation. "The experimental results indicate that CT-based English language learning is significantly more effective in enhancing students' learning performance and motivation for language learning than traditional learning. Understanding computational thinking will give students a foundation for solving problems, and will be one of the fundamental core abilities in all steps of life in the 21st century". Parsazadeh et al. also concluded that the use of computational thinking with digital storytelling is effective and that it increases curiosity, interest and interaction between students, which may lead to better

performance in English. This and other research ([TKHS21], [KK]) show that digital storytelling is an effective method to incorporate computational thinking in a course.

3 Methodology

This chapter describes the design and methods used to be able to answer the research questions. The first section describes the research design, what type of research it is along with the lesson design and how these are created with the research in mind. The data collection section describes who collected the data, how it was collected and who the participants are. In the last section, data analysis, it is described how the data was analysed and what tools were used.

3.1 Research design

This research is a qualitative research, a cross-case analysis. The two cases are parts of a larger project [nwo], these were implemented as case-studies for two consecutive years. In these case-studies there are multiple ways to collect data, such as observations, interviews and surveys.

Cross-case study

A cross-case study is an exploratory study where multiple case-studies are analysed. This study is better than a single case-study because it allows for a more intensive analysis of the same issue across multiple cases [LC12]. In this study four dimensions will be cross-analysed: English learning outcomes, computational thinking learning outcomes, students' attitudes and teacher's attitudes. The data will be described in the results (section 4) and then a summary will be given in the conclusion (section 5) along with related literature or logical outcomes using induction.

3.2 Lesson design

Case-study 1

Case-study 1 was a research done in the academic year of 2020-2021, this research was carried out within the scope of NWO's project. The aim of this research was to get a clear view of the effectiveness of the integration of computational thinking in the school subject English. The students should be able to use computational thinking concepts during the English lessons. Not only did this research analyse the data from the students, but also from the teacher, because most teachers do not have experience with computational thinking concepts, especially teachers from language subjects.

The English learning objectives were: The students are able to describe themselves, things, places, date and time. The assignment in this case-study was to create a digital story in Scratch where the students introduced themselves, giving their name, age, date and place of birth and appearance, but also a description of their family, city, school and hobbies. This information had to be included in the correct tense. The lessons of case 1 were divided into two parts, each part consisting of four lessons. In the first part of the lessons the focus was on the English learning objectives, the English content that the students had to implement in their digital stories. They had to describe what they look like, what they like or dislike and some other general information

about themselves. In the second part of the lessons the stories had to be implemented in Scratch, the software used to create the digital stories. It started with a research phase, then a design phase and then two lessons for the implementation phase. At the end of this lesson series the students had to submit their Scratch projects to the teacher and researcher for assessment and further analysis

Case-study 2

Case-study 2 was a research done in the academic year of 2021-2022, also carried out within the scope of NWO's project. The participants of this case-study were (a fraction of) the same students as case 1, but a year later. This case-study was similar to the first one in the sense that the integration of computational thinking through digital storytelling was the same, but the English learning objectives were different.

The English learning objectives in case 2 were: The students are able to create a story with foreshadowing and following Freytag's Pyramid using correct English. Foreshadowing is a method used in different media where a hint is given to the audience about what is to come or happen. Freytag's Pyramid is a story structure where the story works up to a climax (rise) and then works down to a catastrophe (fall), as seen in figure 1. The digital stories had very different requirements in this case-study. The lesson series consisted out of eight lessons in total again. This time the lessons were divided into three parts. The first part took place in normal classrooms where groups of three were formed and the groups had to come up with their story, but not in Scratch yet. These digital stories had to include an interaction, asking for user input which influences the actual story. In the second part of the lessons the students had to implement their stories that they created in the first part in Scratch. These students had experience with Scratch from the first case-study. In the last of the lessons the students had to present their stories to the class. After a brief introduction, they had to present a summary of the story and then the story itself was viewed to the class. At the very end the data and Scratch projects were collected for assessment and further analysis

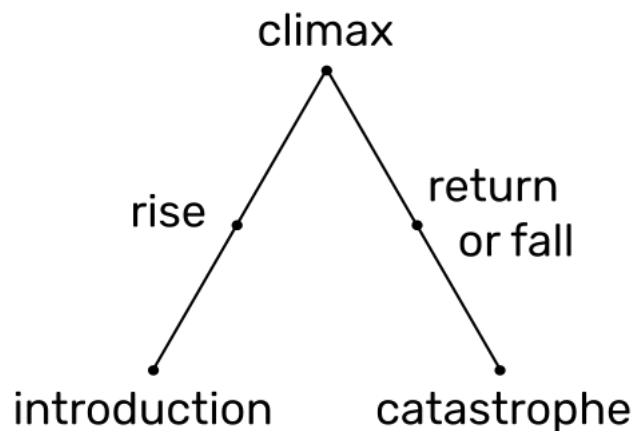


Figure 1: Freytag's Pyramid, source: [Wikipedia](#).

3.3 Participants

The participants of this study are the students of the same class in two consecutive years. There were 29 participants divided into groups of two in case 1 and groups of two, three or four in case 2. The students consisted out of 19 boys and 10 girls, being between 13 and 15 years old during case 2 and a year younger in case 1. These students were in the first grade of secondary school during case 1 and second grade during case 2. All the students have had programming lessons for either Python or Scratch.

The same teacher taught this class English in case 1 and case 2. This male teacher had 12 years of experience in teaching by the time of case 2. Before case 1, this teacher had no programming experience in any form.

3.4 Data collection

The researchers collected data in two different ways: surveys and interviews. Different data was collected between the different case-studies, but for this research only the data that are present in both case-studies, will be mentioned and considered. The data was either collected from the students or from the teacher. In addition, only the students that were part of both case-studies are mentioned and considered, which is one class (in case-study 1 multiple classes were part of the research). The data consisted out of student interviews, teacher interviews, exit tickets (survey) and the Scratch projects themselves. Each piece of data should be relevant for this research by answering (at least) one of the research questions.

Exit tickets

The exit tickets of case-study 1 consisted of five questions, but one of these questions changed in case-study 2, so only four are considered in this research. These questions ask about the opinions of the students, whether they enjoyed it, found it interesting, understood what they had to do and whether they found it difficult. The students could answer *yes*, *unsure* or *no*. This data was used to answer research question 3 (students' attitude). In case 1 there are 18 responses available. In case-study 2 the exit tickets consisted out of six questions, so an extra question was added. Again only the four questions that were asked in both cases are considered, so that extra question is also left out of this research. The exit tickets in case 2 also had some open questions after these, these are also not considered. In this case there are 28 responses available.

Scratch projects

The Scratch projects were the end products of the students. These data were used to answer research questions 1 and 2 (English learning outcomes and computational thinking learning outcomes, respectively). In case 1 there are 5 Scratch projects available. These Scratch projects were made in groups of two. With the Scratch projects, the story itself can be played and the code can be viewed to see how it is made. So in the Scratch projects the digital stories can be read and heard for the writing and speaking part of English. For computational thinking, the code itself can be analysed, how the students were able to write their story to the digital screen with Scratch. In case 2 the Scratch projects were also made in groups, but this time also having groups of three and four.

Student interviews

At the end of the lessons student interviews were taken by the researcher. The interviewees are the student groups. The interviews taken within the same case were asked the same questions, but case 1 and case 2 had different questions. These interviews were used to answer research question 3 (students' attitude), so only the questions that were relevant to that question were considered. These are questions that asked about the students' attitude. They were asked what they found to be difficult, what they thought of Scratch and the lessons themselves. In case 2 there is an extra question where they are asked to compare this case (case-study 2) to the previous case (case-study 1), this question is also considered, because it is relevant to research question 3 and the research question.

Teacher interviews

At the end of the lessons the teacher was also interviewed by the researcher. This data was used to answer research question 4 (teacher's attitude). The questions asked were largely the same. The teacher interviewed was the same teacher in both case-studies. The questions that provide a better view of his attitude towards these integrated lessons are considered. The teacher was asked about the learning objectives (both for English and computational thinking), his standard way of teaching, his opinion and feedback, among others.

3.5 Data analysis

In order for the data to be analysed, the exit tickets were transcribed to a spreadsheet and the students and teacher interviews were transcribed to text documents. The Scratch projects were left as they were, analysing the code and the digital story. This cross-case study was done by comparing the data. Consequently, all the data that was used in this research, there is a version for case 1 and an equivalent for case 2. These data are put next to each other to detect the differences and similarities. For some research questions, in order to be able to answer them, the data had to be assessed (in a different form) first. Consequently, the data of the exit tickets, which were given as frequencies, was also converted to percentages, because the number of responses differed between case-study 1 and case-study 2. With the percentages it was now possible to compare that data between case 1 and case 1. The Scratch projects were analysed with the help of tools in the form of tables, charts and rubrics. The number of Sprites and backdrops, which are key components from the Scratch projects were counted and put into tables and charts. Also the number of programming concepts and mistakes made were counted and put into tables. These were all manually counted. A rubric made by Maryam Vaezi and Saeed Rezaei [VR19], which focuses on creative storytelling, was used to assess the English content of the Scratch projects. This resulted in a certain amount of (total) points, which can be compared again between case 1 and case 2. All of these tables and charts were made in such a way such that it is very convenient to compare the data between case 1 and case 2.

4 Results

4.1 Scratch projects

Five Scratch projects from case 1 and six Scratch projects from case 2 will be used to analyse the students' knowledge of computational thinking and the subject English itself. Because the assignments were different between the cases, more general components of the Scratch projects will be analysed. So the components that will be analysed to measure the students' knowledge of computational thinking are: number of sprites, number of backdrops, use of code blocks (amount and variation) and interactions. Finally, the components to measure the students' knowledge of English are: pronunciation (speaking) and grammar (writing).

4.1.1 English learning outcomes of students

Content & vocabulary

Case 1	Case 2
Introduce yourself: name, age, place and date of birth, family	Freytag's Pyramid
Descriptions: city, school, appearance, clothing style	Foreshadowing
Hobbies and (dis)likes: movies, sports, music, food, social media	Interaction

Table 1: Content requirements of the digital stories

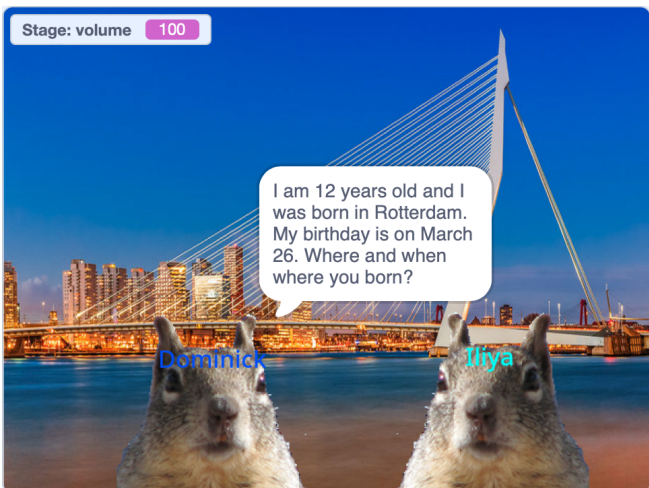
This first part of the English learning outcomes of the students is about the (digital) stories. The projects in case 1, the content in terms of storytelling were similar to each other. All the Scratch projects asked the same questions, so there is not much variety in that. Those questions can be seen in table 1, where it lists every information that is required to be discussed in the digital story. The only difference between the digital stories of case 1 is the order in which the information is presented. In figure 2 below there is a screenshot of each project from case 1.



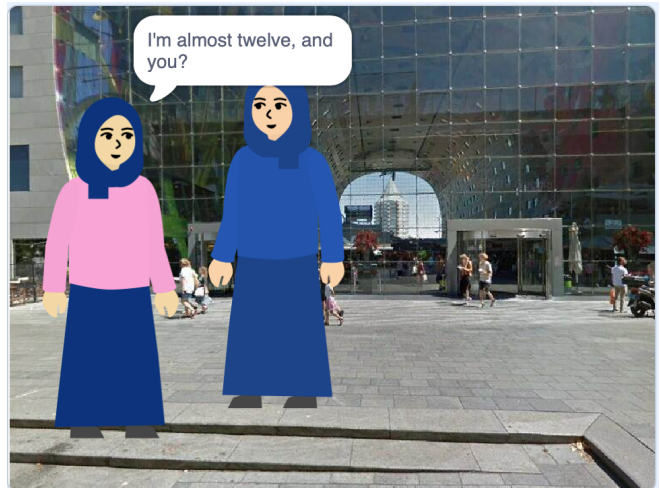
(a) C1P1



(b) C1P2



(c) C1P3



(d) C1P4



(e) C1P5

Figure 2: Screenshots of the digital stories from case 1

The English that was used in the Scratch projects of case 2 was more complex. The conversations were not of the questions & answers type anymore, but more like a real story, with character sprites actually talking to each other. Because the stories are rich and different from each other, there is also a summary per project in table 2.

Project + content	Screenshot
<p>C2P1: A babysitter arrives at the client’s house and after helping the brother with his homework, it is time for the twins to go to bed. Then the babysitter invites her friends over to the house and she picks up a gun. The twins decide to sneak out of their room and witness one of the friends getting shot by the babysitter with that gun. The twins try to escape from the house and when they get in the car a choice is given to the audience to run over the babysitter. In the end it turns out to be just a bad dream for both of them.</p>	
<p>C2P2: Two guys are playing basketball when a third guy comes in and catches the ball. He says that they can only get the ball back if they win. The ball gets taken from him and a choice is given to the audience to either dunk or shoot and win the game to get their ball back.</p>	
<p>C2P3: There are two aliens that are trying find a safe place to stay. They are met by another alien who refuses them to stay, after which he becomes aggressive and the two refugees flee back to their ship. On the ship the audience has a choice to go left or right, each direction going to a different planet. On this planet they get a call from that same alien from before saying that he wants to get revenge. In the end it is revealed that this is just a game that they lost and that they will try again tomorrow.</p>	




<p>C2P4: Two friends are rushing to make it to the bus to go camping. Eventually there, they get lost in the wilderness and they start to hear some noises. They see a close cave nearby and decide to run to it so they can hide in it. Once in the cave they find out that there is a bear in the cave, the noises being the roars.</p>	
<p>C2P5: A man and a woman bump into each other, the man dropping his wallet. The man keeps on walking and the woman picks up his wallet. She finds out where he lives and goes to his house to return the wallet. Once there she gets invited by the man to come in and they eventually decide to marry. The man asks the woman to go to a store to buy a wedding dress, on the way she sees on the TV that he is a wanted man. When she gets back home she asks him why he never told her. Eventually he is forgiven by the princess and they get married.</p>	
<p>C2P6: Two friends go home together after watching a movie. Once one friend gets home, the other one walks alone, she notices something strange but decides to keep on walking. The next day it turns out that the friend got killed. A mouse starts talking to the friend, saying that it wants the friend's help to get revenge. They go together to the murderer's hideout and fall into a trap that sets off a bomb. With the murderer, the friend and the mouse all being present at the hideout, the mouse asks the friend to set off the bomb, getting revenge at the cost of their own lives.</p>	

Table 2: Scenes and summaries from each digital story of case 2

The content of the digital stories in case 1 was basic and similar to the other ones. The questions were the same across all those projects and it was a question and answer type of conversation. In case 2 the students were given more freedom, so the content and vocabulary in case 2 are richer than in case 1. Whereas all the Scratch projects of case 1 adhered to the questions & answers type of conversation, the Scratch projects of case 2 felt more like a story. It had a beginning, middle and end (according to the Freytag’s Pyramid) and the story was proceeding more naturally, like a real life story. Finally, every story in case 2 was very different from each other.

Creative writing

Project	Voice	Characterisation	Atmosphere	Language	Dialogue	Story	Setting	Image	Plot	Total
C1P1	1	1	1	2	1	1	1	1	1	10
C1P2	1	1	1	2	1	1	1	1	1	10
C1P3	1	1	1	3	1	1	1	1	1	11
C1P4	3	3	2	3	2	2	2	2	2	21
C1P5	1	1	1	2	1	1	1	1	1	10
C2P1	3	3	3	3	3	3	4	4	3	29
C2P2	3	3	3	3	3	3	3	3	3	27
C2P3	2	2	3	3	3	2	2	2	2	21
C2P4	2	3	3	3	3	3	3	3	3	26
C2P5	3	3	2	3	3	2	2	2	2	22
C2P6	3	3	3	3	3	3	4	4	3	29

Table 3: Creative writing per Scratch project

For the second part, the student’s ability to write digital stories in a good way is measured. A (good) story has to fulfill certain criteria. For this research, the following criteria will be analysed, based on the rubric made by Maryam Vaezi and Saeed Rezaei [VR19]: voice, characterisation, atmosphere, language, dialogue, story, setting, image and plot. The points given are as follows: excellent = 4 points, above average = 3 points, developing = 2 points and needs improvement = 1 point. In table 3 the points are presented in a table, with the total amount per project in the last column. In case 1 there was no characterisation, the character sprites were chosen without taking the story into consideration. The physical attributes of the chosen character sprite had nothing to do with the creator or the story itself. The only exception being C1P4, which had two female character sprites with headscarves, representing the creators in that way. There was no atmosphere nor a story, only a simple greet at the beginning, straight-up questions and answers and then a farewell, always between two characters. The dialogue was the same in all the projects, outside of the required questions there was no dialogue. In C1P4 the flow of the dialogue was more natural, having more of a conversation type of questions & answers instead of the turn based questions & answers. The setting was different in all the projects, but irrelevant to the story. The backdrop did not add anything to the story and was randomly chosen by the creators. The exception was again C1P4, which had some settings that cohered with the story and dialogue, talking about their favourite movies and going to the cinema, with actual backdrops of a cinema.

The same criteria will be assessed for the creative writing in the digital stories of case 2. The characters this time reflected the story, players in actual basketball outfits playing basketball, aliens sprites because the digital story takes place in space, big sprites for older characters and smaller sprites for young characters, etc. The story was different in every project. Some stories had more than two characters, with protagonists and minor characters. Enough information was given to the audience at the beginning to know which sprite was who. Each digital story required some input from the audience, which most of the times had an impact on the story. Even though the stories were all different, the atmosphere was the same. This is because all the projects were required to use the Freytag’s Pyramid (1), so with a buildup to a climax and a fall a to a catastrophe. The dialogues in these projects had a natural flow, more like real world conversations. It carried emotion (through articulation, mostly) and moved the story forward towards an end. The settings of these projects reflected the story. Depending on the location of the story, the backdrop of the digital story changed along and accordingly. Even in minor cases, like when a character sprite moves to a closed door to go to another location, the backdrop changes to that same image but with the door open. Most of the times the foreshadowing part is hidden in the setting, like a gun in the background.

There was almost no criteria points for creative writing present in the digital stories of case 1 outside of language, except C1P4, which had at least some form of characterisation, setting and dialogue. These criteria, however, were present in all the digital stories of case 2. Thus, taking these criteria as a measurement of creative writing, it is clear that there is an increase in case 2 compared to case 1.

Grammar & pronunciation

Project	Mistakes	Length (s)	Mistakes per second
C1P1	8	99	0.08
C1P2	4	108	0.04
C1P3	8	178	0.04
C1P4	3	210	0.01
C1P5	2	121	0.02
C2P1	4	210	0.02
C2P2	0	40	0.00
C2P3	0	121	0.00
C2P4	3	124	0.02
C2P5	1	119	0.01
C2P6	2	202	0.01

Table 4: Number of mistakes and its length per project

For this last part, the number of mistakes per project are counted and the length of the digital stories is measured in order to make it comparable between the cases. These mistakes are either grammatical mistakes or pronunciation mistakes. In case 1, an example of a grammatical mistake is the wrong use of the plural form of hobby as hobby’s instead of hobbies (C1P2). Other examples are *Have you a pet?* instead of *Do you have a pet?* and *Do you have a hobby to?* instead of *Do you*

have a hobby too?. Overall, mistakes are present in every Scratch project of case 1, between two and eight, which can be seen in table 4.

In case 2 the number of mistakes are counted again along with the length. Only the dialogue parts are counted towards the length of a project, since the projects in case 2 have a lot of movement and transition sequences without any dialogue. There are two Scratch projects without any mistakes, C2P2 and C2P3. It is important to note that C2P2 was a very short digital story with fewer dialogue. The other Scratch projects have a maximum of 4 mistakes. Examples of some of the mistakes are *exicted* instead of *excited* (C1P4) and the pronunciation of *idea*. The mistakes per second is at most 0.02 in case 2.

A decrease in the number of mistakes and mistakes per second can be seen when looking at table 4. There were no project in case 1 that had no mistake, whereas case 2 has two of such projects. The average number of mistakes is lower in case 2 compared to case 1. Also the maximum of mistakes per second in case 2 (0.02) is below the average of case 1.

4.1.2 Computational thinking learning outcomes of the students

Sprites

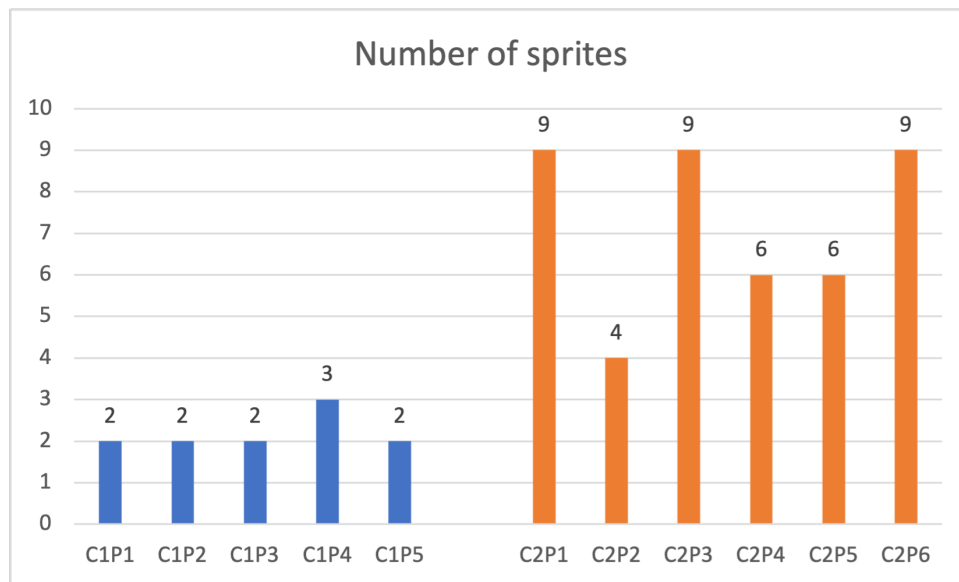


Figure 3: The number of sprites used per project in case 1 and case 2

The number of sprites were counted in every Scratch project of case 1 and case 2. In order for a sprite to count, it has to be a part of the digital story with at least one script. As can be seen in figure 3, all the Scratch projects from case 1 used exactly two sprites, except for C1P4, which had three sprites. In all of these projects, there are two character sprites that had a conversation with each other. The project C1P4's additional sprite was a headset on one of the character sprites, this headset was only visible during a relevant part of the digital story. The project C1P4 was also

the only project where the sprites had some form of movement, whereas in all the other Scratch projects of case 1 the sprites were static. In figure 4 the standard layout of every Scratch project from case 1 can be seen, a character sprite on the left having a conversation with a character sprite on the right.

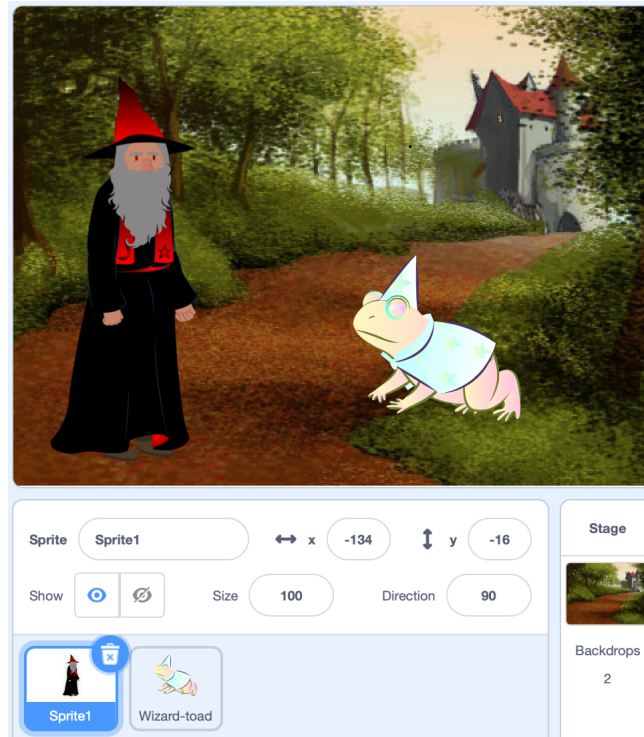


Figure 4: The sprites of project C1P4

The projects of case 2, however, had more sprites compared to the projects of case 1, which can also be seen in figure 3. The amount of sprites varied between 4 and 9. These were not just characters sprites that were part of the digital story, but there were also (more) object sprites present in these Scratch projects. These object sprites also serve a purpose in the digital story, but they are not characters. Examples are objects like a phone, wallet, basketball, etc. An example can be seen in figure 5, the character sprites being the first four (those are the ones with dialogue) and the object sprites being the other five sprites. The sprites from case 2 also changed during the digital stories through costumes, costumes are variations of the same sprites, e.g. facial expressions, a sprite being happy during one part of the story and then that same sprite being sad during another part of the story. These sprites also had movement and only the sprites that were a part of the scene were visible.

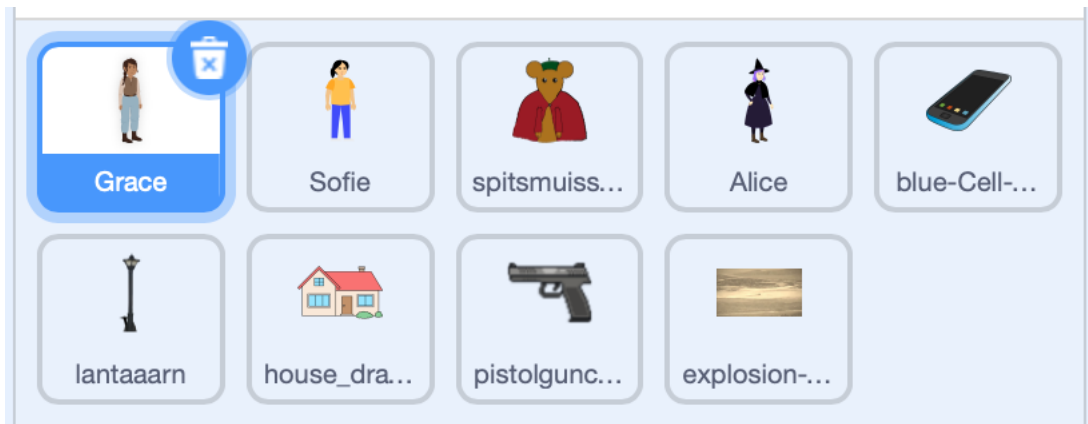


Figure 5: The sprites of project C2P6

To summarize, there is an increase in the number of sprites in the Scratch projects of case 2. The character sprites are increased in some Scratch projects, having more than two characters sprites that are part of the digital stories, whereas the Scratch projects in case 1 always had exactly two character sprites. And there are also object sprites present in case 2, just like the one project from case 1, C1P4, which are not present in any other Scratch project of case 1. These sprites from case 2 all had some form of movement, most of them actually moving and some of them appearing and disappearing during certain parts of the digital story, this also was only present in that same project of case 1, C1P4.

Backgrounds

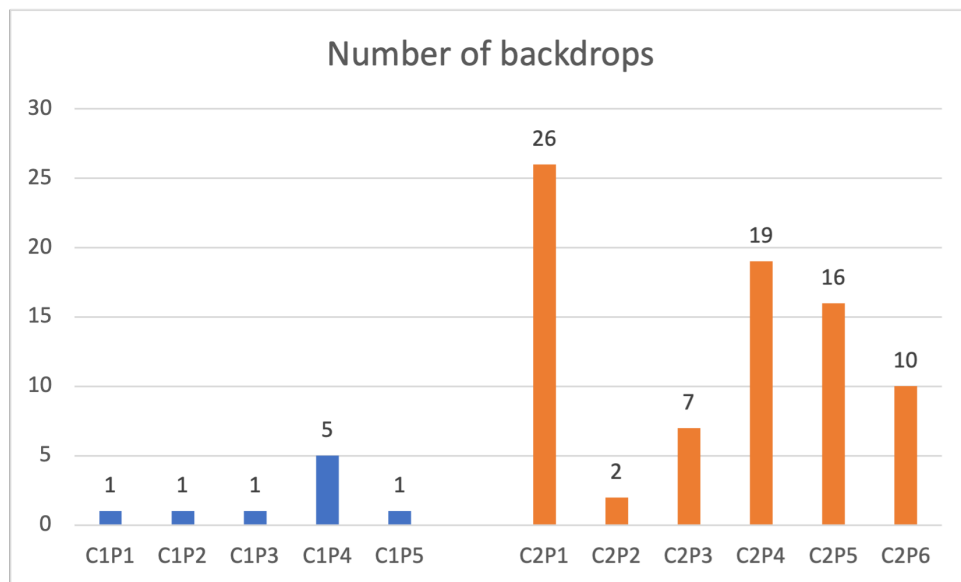


Figure 6: The number of backgrounds used per project in case 1 and case 2

Next, the number of backdrops were counted in every Scratch project of case 1 and case 2 of which the results can be seen in figure 6. There is a similar pattern with the number of backdrops and the number of sprites in the previous section. Again, all the Scratch projects from case 1 had just one backdrop during the whole digital story, apart from C1P4, which had five backdrops. Therefore, this was the only Scratch project from case 1 where the digital story took place at different locations. In figure 7 two different backdrops can be seen of this Scratch project, the one on the left being outside the cinema and the one on the right being inside the cinema.

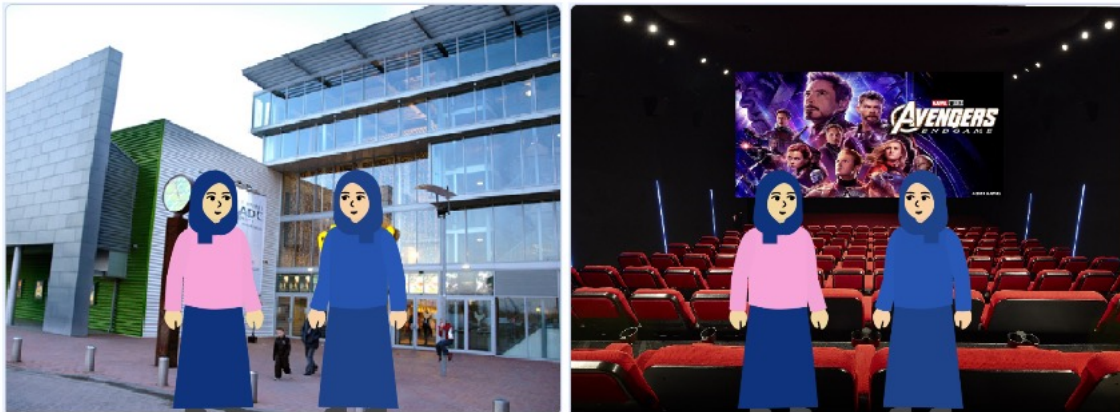


Figure 7: Two consequent backdrops of project C1P4

In case 2, all the Scratch projects had at least two backdrops, as seen in figure 6. Some backdrops are variations of other backdrops, with a slight change (e.g. open and closed door or night and day). The Scratch projects do not always use all the backdrop in their digital stories. Since the assignment in case 2 was to have some form of interaction, sometimes this interaction can lead to different story outcomes (branches). Therefore, some backdrops are only used depending on those interactions.

As said before, these results are similar to the results of the number of sprites. There is that same Scratch project in case 1 (C1P4) again that sticks out with the number of backdrops, other than that all the Scratch projects just had one backdrop from start to finish. In case 2 there is a greater number of backdrops and there is also more variety, some Scratch projects sticking out more than others (for example C2P1, C2P4 and C2P5). The least one being two backdrops and the most one being 26 backdrops. In both cases the backdrops were acquired from the Scratch itself, uploaded from the internet or custom made.

Computational thinking concepts

For this section the code itself is central: What programming techniques were used to create the digital stories and what are the differences between case 1 and case 2? This thesis will adhere to the techniques as described by Karen Brennan and Mitchel Resnik [BR12]: sequences, loops, parallelism, events, conditionals, operators and data. For sequences the longest script (sequence of code blocks) will be counted. For parallelism and data, those techniques will be marked with a *yes* or *no*, depending on whether that Scratch project contains any form of parallelism or use of variables. The rest of the techniques will be given as a frequency, representing the number of times that particular technique is used within that Scratch project. Loops are a technique where a body of code blocks is repeated, instead of adding that same body of code blocks multiple times. Events are interactions between scripts, one thing causing another thing (or things) to happen, the simplest form of event being the green flag (start button). Conditionals are if (else) statements, where a body of code blocks is only executed if the condition is true. Lastly operators, these are mathematical and logical operations and some string operations, in Scratch these are the green code blocks.

Programming concepts	C1P1	C1P2	C1P3	C1P4	C1P5	Average	C2P1	C2P2	C2P3	C2P4	C2P5	C2P6	Average
Sequences	40	48	41	35	50	42.8	43	21	33	21	98	32	41.3
Loops	0	0	0	3	0	0.6	0	0	0	0	1	1	0.3
Parallelism	Yes	No	No	Yes	No	N/A	Yes	Yes	Yes	Yes	Yes	Yes	N/A
Events	4	2	4	19	2	6.2	38	27	43	52	9	49	36.3
Conditionals	0	0	0	0	0	0	1	4	2	2	1	2	2
Operators	0	0	0	0	0	0	1	4	2	2	1	2	2
Data	No	No	No	No	No	N/A	No	No	No	No	No	No	N/A

Table 5: Programming concepts per Scratch project

In table 5 above the Scratch projects were assessed based on the previously mentioned programming techniques. The sequences were lower in case 2, but C2P5 is a big outlier, which had a very long sequence of 98 code blocks. There was one project in case 1, C1P4, that used loops, though these loops were set to one (so nothing was repeated). In case 2 two projects, C2P5 and C2P6, used loops that actually repeated code block(s). In case 1 three out of five Scratch projects used parallelism and in case 2 all of them did. The Scratch projects in case 1 used an average of 6.2 events and in case 2 this was increased to 36.2 events. In case 2 there seems to be a connection between the sequences and events, the more events the project has, the shorter the sequences of that project are. This makes sense because the code blocks are spread among the events. This is also visible, but to a lesser extent, in case 1. None of the Scratch projects used conditionals or operators in case 1. In case 2 all of them did, this has mostly to do because the digital stories required some input from the audience, giving them a choice that influenced the story. Outside of the conditionals there were no operators used, that is why the amount of conditionals and operators are equal. And lastly none of the projects, both from case 1 and case 2, used data.

The Scratch projects of case 1 had sounds in it and the codes were sequential in every project. Thus, after every interaction, which were dialogue in the form of text and speech, a wait-block was added with a timer. After this wait-block the next interaction is added, until the end. Thus each sprite of case 1 has two long scripts of blocks, with wait-blocks between each interaction. One script for the visual part and one script for the audio part. C1P2, C1P3 and C1P5 (the Scratch projects without parallelism, table 5) had one script per sprite, in these scripts the audio and visual code blocks were placed under each other instead of being separated in two scripts. Therefore, there was also not much variation in the use of different block colors, they were all the same. Similarly, all the scripts had the same pattern, interaction and wait-blocks. This pattern can be seen in figure 8. Except for C1P4, which also had blocks for the movement of the Sprites. In terms of interaction, all the projects had a two-way conversation between the two sprites. This was done with the wait-blocks, setting the timers in such a way that the interaction (speech bubbles and voice) of one sprite begins after the interaction of the other sprite.

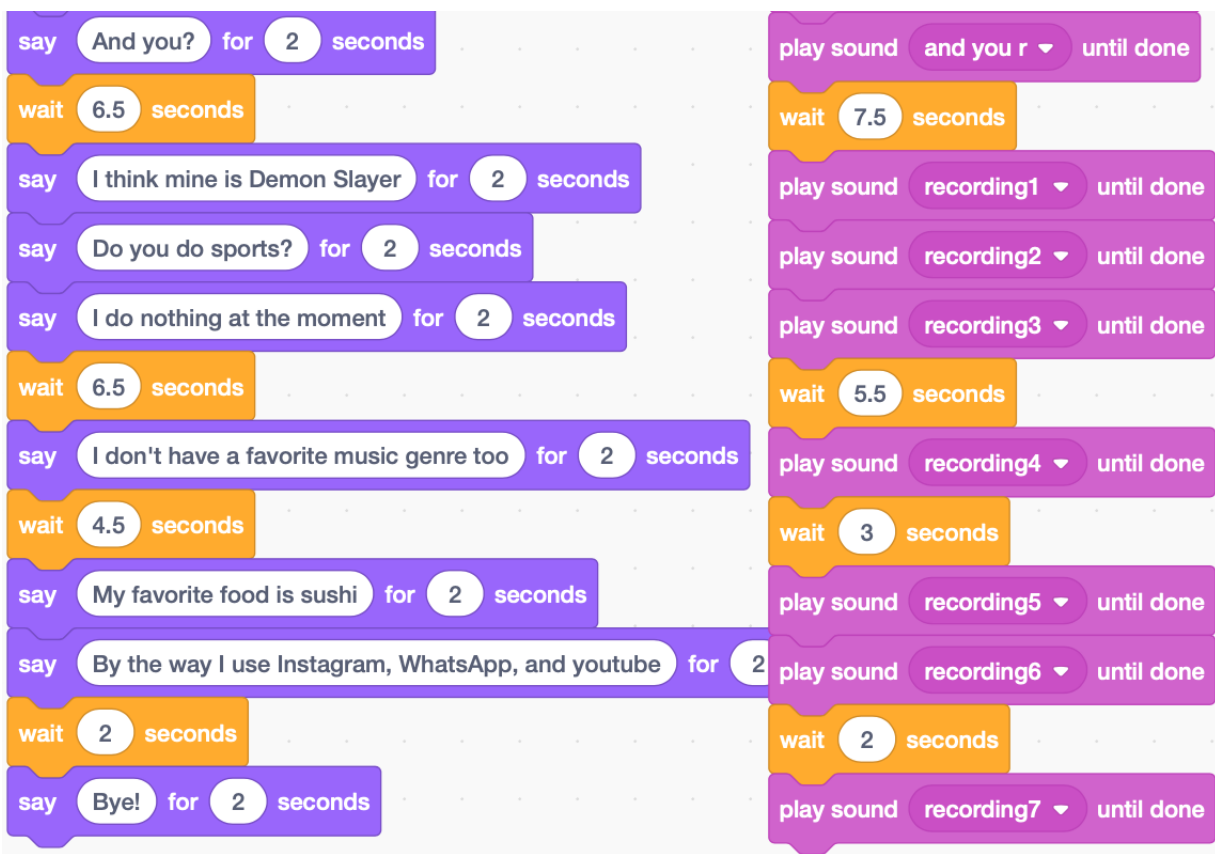


Figure 8: A script for the text (left) and a script for the audio (right) from C1P1

Because the Scratch projects of case 2 had more sprites, as mentioned in the previous section, the total amount of code blocks in case 2 is also more than the amount of blocks in case 1, since each sprite has its own script(s). The Scratch projects of case 2 also made use of broadcasts, giving a start signal to other sprites to start their script(s). This is an alternative way of the wait-blocks that were heavily used in case 1. These Scratch projects also made use of sounds, just like the ones of case 1. These sounds, however, were not only used for speech, but also for sound effects (e.g. doorbell). There are also code blocks used in case 2 that were not present in the Scratch projects of case 1, such as control flow commands (e.g. if-statements for the interaction part). So these Scratch projects had more interactions, with code blocks (from different sprites) interacting with each other (with the use of broadcasts) and the sprites themselves having more interaction (moving, changing their appearance, etc.). A code snippet can be seen in figure 9, where the scripts start when a broadcast is received, these scripts have sprite movement and backdrop & costume changes.

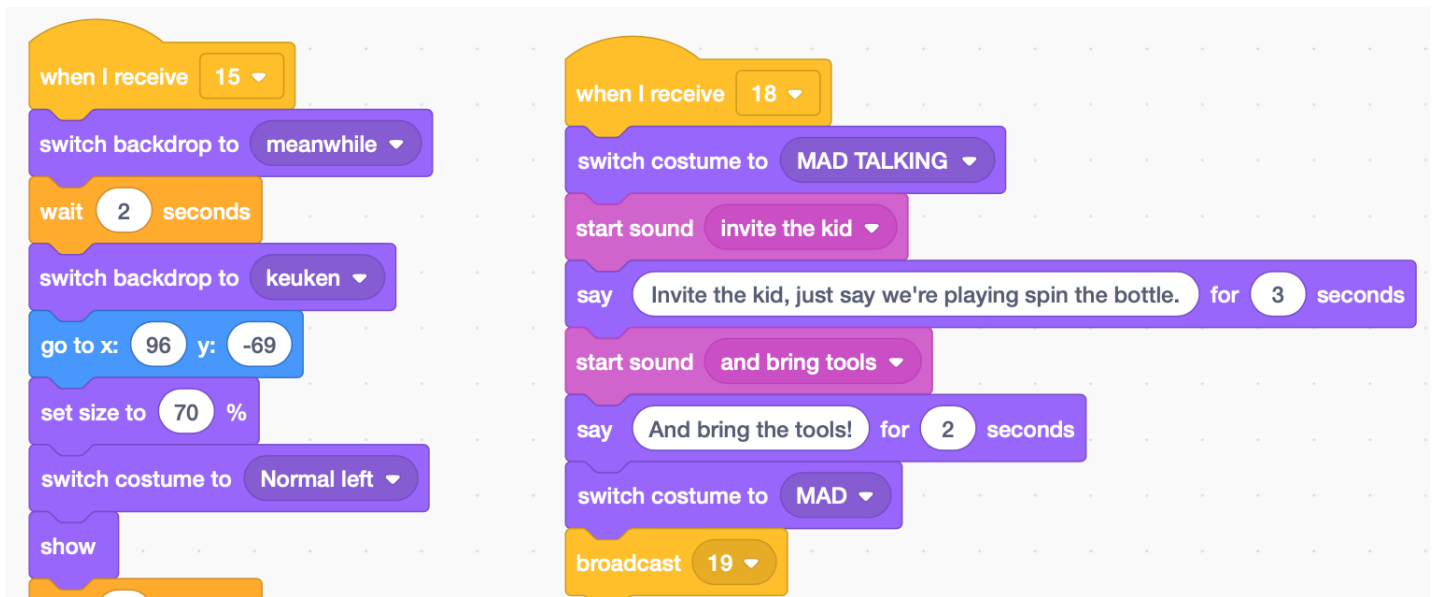


Figure 9: A snippet of a script for a sprite in C2P1

There is more variation of code blocks in the Scratch projects of case 2. These extra code blocks are used to add more interaction to the Scratch projects, compared to the very limited interactions of sprites in case 1. The Scratch projects of case 1 had a familiar structure that was used in almost every sprite of every project, speech bubbles and voice sound followed by wait-blocks. The Scratch projects of case 2 also made use of more complicated programming concepts, such as if-statements, to implement interaction. The total amount of code blocks is also greater in the Scratch projects of case 2 than the ones of case 1. Lastly, there is more separation in the use of code blocks, having more scripts in contrast to the few but very long scripts of the Scratch projects of case 1.

4.2 Students' attitude

The results for the students' attitude come mainly from the exit tickets, but also from student interviews. In this section, the students' attitude and opinions about these two computational thinking integrated English lesson series are central. This first part will be about the exit tickets. The students were given five statements about their opinions of these lessons. The exit ticket of case 2 had some extra questions on top of the ones of the exit ticket of case 1. Since there is no data on those extra questions for case 1, they are not considered in this thesis. One of the five statements was changed for case 2 and since there is no data of these two statements in both cases, they are also not considered in this thesis. The remaining four statements were of the ordinal type, where the students could answer *yes*, *no* or *unsure*. These questions were asked in Dutch. In case 1 the exit ticket was filled in by 18 students and in case 2 it was filled in by 28 students. The overall results of these exit tickets can be seen in table 6 below.

4.2.1 Exit tickets

Statement		Yes	Unsure	No	Total
I enjoyed the lessons	Case 1	18 (100%)	0 (0%)	0 (0%)	18
	Case 2	17 (61%)	7 (25%)	4 (14%)	28
I found the lessons to be interesting	Case 1	15 (83%)	3 (17%)	0 (0%)	18
	Case 2	11 (39%)	13 (46%)	4 (14%)	28
I understood what I had to do during the lessons	Case 1	15 (83%)	3 (17%)	0 (0%)	18
	Case 2	22 (79%)	5 (18%)	1 (4%)	28
I found the lessons to be difficult	Case 1	4 (22%)	4 (22%)	10 (56%)	18
	Case 2	3 (11%)	11 (39%)	14 (50%)	28

Table 6: Total overview of the exit tickets

I enjoyed the lessons

In order to make the data comparable between case 1 and case 2, the data of the statements are also converted to percentages, as can be seen in table 6. All the students in case 1 said *yes* to the statement "I enjoyed the lessons". For case 2, about half of the students, 61%, said that they enjoyed the lessons. A quarter of these students were not sure whether they enjoyed the lessons or not. 14% of the students did not enjoy lessons. Since the student size of case 2 is almost double the student size of case 1, the amount of *yes* answered are about as equal in both cases (18 and 17 students, table 6).

I found the lessons to be interesting

The second statement was to measure whether the students found the lessons to be interesting, the results of this statement can be found in table 6. In case 1 there were no students that did not find the lessons to be interesting, in case 2 14% of the students found the lessons to be uninteresting. 17% of the students in case 1 were unsure whether they found the lessons to be interesting or not. In case 2 this was more than doubled, where 46% of the students were unsure. The majority of the students, 83%, did find the lessons to be interesting in case 1. But in case 2 this was decreased to 39% of the students.

I understood what I had to do during the lessons

The third statement was to find out whether the students understood what they had to do during the lessons, so what was expected from them by the teacher(s). The outcome of this statement can be found in table 6. No students in case 1 did not understand what they had to do during the lessons. In case 2 there was one student who did not, making it 4% of the total. About the same percentage of students in case 1 and case 2 were unsure, 17% and 18% respectively. The majority of the students, both in case 1 and case 2, did understand what they had to do during the lessons. This was 83% for case 1 and 79% for case 2.

I found the lessons to be difficult

For the last statement in the exit ticket, the students were asked whether they found the lessons to be difficult. The percentages can be found in table 6. Around half of the students in case 1, 56% to be exact, answered *no*, not finding the lessons to be difficult. In case 2 this was exactly the half, 50% of all the students. 22% of all the students were unsure, so they did not find it difficult and also not easy, but somewhere in between. For case 2 this percentage was increased to 39% of the students. And another 22% of all the students answered *yes* to this statement, those students did find the lessons to be difficult. This percentage was halved in case 2, sitting at 11% of all the students.

According to the results of these exit tickets, the students were more positive towards the computational thinking integrated English lessons in case 1. This is especially evident when looking at the first two statements "I enjoyed the lessons" and "I found the lessons to be interesting". The results of these two statements had an increased percentage of *yes* and a decreased percentage of *unsure* and *no* in case 1, compared to case 2. So a higher percentage of students enjoyed the computational thinking integrated English lessons of case 1 than case 2, also a higher percentage of students found these lessons to be interesting. The results of the statement "I understood what I had to do during the lessons" are roughly the same between case 1 and case 2, with only a slight increase in percentage of the amounts of *no* and *unsure* in case 2. In case 2 of the last statement, "I found these lessons to be difficult", there is an increase in percentage of the amounts of *unsure* while there is a decrease for both *yes* and *no*. So the students in case 2 were less sure whether they found the lessons to be difficult or easy, whereas in case 1 they were more decisive.

4.2.2 Students interviews

Scratch

In the second part of the students' attitude interviews were analysed from the students. The students were asked, among others, about their opinions, what they liked and disliked and whether they found the project to be difficult. There are three group interviews from case 1 and four group interviews from case 2. In case 1 the students were asked about what they liked and disliked about the program Scratch. The student from group 1 said that he/she did not like to copy paste blocks of codes and then changing the small parts individually, saying: "You need to do it all over again, that was a bit annoying". The student from group 2 said that working with Scratch is too time consuming, but "when it is done it is very fun". The student from group 3 said that he/she liked working with Scratch, but would not use it again for a future project unless he/she had to.

In case 2 the students were not directly asked a question about their opinion on Scratch, but the students still shared their experience. One of the students from group 1 said that he/she liked this different approach and working with Scratch in general. A student from group 2 wanted to use Python instead of Scratch, since he/she had prior experience with Python and he/she found that more useful than Scratch. These students from group 2 also ran into the issue where Scratch's auto save function was not working properly. Another student from group 2 found Scratch to be the most difficult part, because they found it to be inconvenient. In group 3 a student said that it was frustrating that things did not work out the first time, but through trial and error. In the last interview a student said that Scratch was too difficult for him/her.

Lesson series

In both cases the students were asked what they thought of these lesson series. In group 1 of case 1 the student said that he/she really enjoyed it because it was fun and he/she learned a lot from it. The student from group 2 simply said: "It is fun to do this". And lastly, the student from group 3 said that he/she found it difficult but still liked it.

In case 2, when the students were asked what they thought of the lesson series, group 1 said that they found it interesting to be learning this way and that they rather do this than do exercises from books. They also stated, however, that they did not like that this project has a heavy weight towards their English grade, putting too much pressure on them trying to succeed. These students also found the programming and planning to be difficult. The students from group 2 said that the programming was the most difficult part and coming up with the script was the easiest part. The students from group 3 enjoyed these lessons. They enjoyed doing it as a group, dividing the tasks between them and working on them together, also outside of school. The students from group 4 said that they found the explanation of the teacher really good, that everything was explained in a such a way that they could understand it. They found working with Scratch the hardest part and collaborating the easiest part.

From these results of the student interviews it is clear that the students from both cases had at least something positive and something negative to say about the computational thinking integrated English lessons. Technical issues and it being too slow and difficult were the issues student of case 1 had with Scratch itself. In case 2 the students still had (the same) problems with Scratch. Again the technical issues, preferring other programs and repeatedly saying that Scratch was the most difficult part of this project. However, there was also a positive thing that was said about Scratch. It was said that they, despite the difficulties, prefer this different approach with Scratch over the usual approach with books. Looking at the lesson series overall, the students from case 1 enjoyed the lesson series, even though they found it difficult. In case 2, the students liked the collaboration as a group and said that the teacher explained everything in a clear way, but wish they got more help with planning and programming and did not like that this project weighed towards their English grade.

Comparison

Lastly, in case 2 the students were asked to give their opinion on these lessons compared to last year's. The students from group 2 said that the digital stories looked better and more beautiful, because there was more animation and movement. Those students also added that they did not divide the tasks like they did this year and that they got more explanation compared to last year. The students from group 6 said that the sounds (sound effects and pronunciation) and programming were better this year. They were more creative and the collaboration was better, but they felt like there was still room for improvement for the collaboration and planning parts. In group 8 the students said that last year there were only two characters in their digital story who had a conversation without any movement or interaction, but this year they created an actual story with interactions and multiple backdrops. And because this assignment had a deadline and just working during the lessons was not enough, they had to learn to plan better and work from home to be able to finish it in time. The students from group 9 said that the stories were more original, because everyone had their own unique story, this also made the assignment more difficult. Then they said that this year they put more energy in the project compared to last year. So the students were more positive this year, despite the increase in difficulty.

4.3 Teacher's attitude

In both cases an interview was held at the end of the experiments. The researcher was the interviewer and the teacher was the interviewee. During both interviews the following subjects were discussed:

- The computational thinking learning objectives.
- The English learning objectives.
- The teacher's usual way of teaching.
- The students' experience from the teacher's point of view.
- The assessment of the projects.
- The teacher's overall opinion on the integration of computational thinking.

4.3.1 Learning objectives

In the first part, the goal is to find out what the teacher wanted to teach his students. There were two types of learning objectives, English-related learning objectives and computational thinking-related learning objectives. In both interviews questions were asked about both of them. The English learning objectives were different between case 1 and case 2. The English learning objectives of case 1 were grammar, vocabulary and the present simple. The students had to introduce themselves by asking and answering questions about their name, age, place and date of birth and what their families look like. These questions had to be asked and answered in the present simple. Then the students had to describe their city, school and their own appearances and clothing style. Finally they had to ask and answer questions about their own hobbies. They had to use the vocabulary that they had learnt during the first part of the lessons. This vocabulary covered the keywords of the things that the students had to talk about. The computational thinking learning objectives

were roughly the same. In case 1 the learning objective was to be able to create a digital story in Scratch. He mentioned a "step by step" process, which can be thought of as algorithmic thinking.

The English learning objectives of case 2 also included grammar and vocabulary. However, instead of talking about themselves, the students had to come up with their own story. This story had to be built step by step, because the story had to include the Freytag Pyramid and some form of foreshadowing. The teacher said the following about the Freytag Pyramid: "You know, once you read more stories you get how the majority of the stories are built". Instead of learning this through reading stories, however, the students had to learn this technique by writing their own story. Case 2 separated the somewhat broad computational learning objective of case 1 into smaller ones, following Selby and Woollard's definition of computational thinking [SW13], so abstraction, decomposition, algorithmic thinking, evaluation and generalisation. In this case he also mentioned the step by step process. In case 2 the teacher adds that this step by step working process makes the students think more long term, because every step has an influence on the story. The teacher also added the following: "I thought it was a really cool one too, because then they had two options, so if that goes through then this happens. And you know, it was not a big complicated thing that they did, but it gives them the idea of how things work and hopefully they - you know, with English and everything - understand that".

In case 2 the students were given some freedom in case 2, which is in contrast to the more static requirements of case 1. Also the Freytag Pyramid from case 2 is an abstract concept, which can be difficult to understand. This is in contrast to the more straightforward story structure of case 1.

4.3.2 Students' understanding from the teacher's point of view

For this part, this thesis looks at the teacher's knowledge about the students' understanding. In both cases the teacher was asked about the students' reactions. The teacher was asked whether they liked it or disliked it and whether they found the assignment easy or difficult. In case 1 the teacher said that "they liked it a lot", but the teacher also thought that the students found it difficult. The reason for this was that, besides the English components, other skills were required for this assignment. Referring to collaborating in a group and working in Scratch. "So it is a lot of skills that have to come together to create something. I guess that was the hard part for them" The teacher also said that because of this, it was more difficult for him to guide the students. Furthermore, the teacher is not a programmer, so it is new for him too.

In case 2 the teacher's view was again that the students really liked it. However, the difficulties from case 1 are not mentioned this time. Instead, the teacher mentions the following: "The issues that you have in all types of projects - so it was not like an issue specific to this project - those are things that you can expect". The teacher also mentions what he thought the students enjoyed the most about this project: "They were really creative with it, I thought. So what I have got back from them is that they enjoyed it. It was not like "Hey, this is grammar and listen!", but it was more. I think the creative part, that is what they liked the most about it." So, according to the teacher, the students enjoyed it more than usual because it was more than just the teacher explaining something to them. They actually had to make something and had complete freedom within the domain of the project.

From this, it seems like the students found it less or not difficult the second time around. No difficulties were mentioned at all in case 2, unlike case 1.

4.3.3 Instructional strategies

For the instructional strategies, the objective is to find out which instructional strategies the teacher used to achieve his learning objectives. In both cases the teacher was explicitly asked about this during the interviews. In case 1 the teacher starts by saying that normally he would use direct instructions and explicit teaching to teach his students these learning objectives. However, with the digital storytelling the students would need a plan to create the story, which he compared to "creating an algorithm for that story to be complete". The teacher then agreed with the researcher that this planning could be in the form of a storyboard. The teacher did not mention any specific instructional strategy by name.

In case 2 the teacher mentioned multiple instructional strategies. The first one he mentioned was project-based learning. The students were given some freedom to be creative while being very clear to them what was expected in the end: "For me, the most important strategy was to be very clear, but also put limits to it but also give them freedom within those limits". The second one he mentioned was collaborative learning. The students had to make these digital stories in groups of at least two. Dividing the assignment into subtasks for each individual and eventually having a complete story as a group. The teacher also indirectly mentioned that normally he would use direct instructions, "just tell them about it".

There was no mention of collaborative learning in case 1. There was also no mention of project-based learning in case 1. He did, however, mention in case 1 that the planning is important for digital storytelling.

4.3.4 Assessment

For the assessment part, the goal is to find out how the students' projects were evaluated by the teacher. This includes what strategies the teacher used and what aspects of the students' projects he paid attention to. A rubric was used in case 1. This rubric assesses the students' content, English grammar and vocabulary, creativity and collaboration. How did the students make their story in Scratch? Are there moving elements in the story? How did the speaking go?

In case 2 the teacher first explains what he saw in the end products and during the presentations, where the students had to present to the class: "I think that went well, because in that sense I did not see any mistakes, they all understood what they had to do. Some were actually a bit too obvious with the foreshadowing ... But they got it and that is what you want, that they get it". Then the teacher mentions that they also used a rubric this year, different than the one from last year. This rubric was more detailed compared to the one from case 1, assessing structure, graphics, programming, English grammar, pronunciation/vocabulary and organisation. The teacher stated that all the students scored well in the rubric. Comparing these assessment methods to the one that the teacher normally uses, which is isolated and just focuses on grammar and vocabulary, these are more challenging for the students, because more skills are assessed than usual. English is graded based on four skills: reading, writing, listening and speaking. Usually reading and listening tests

are very accurate, because the questions are multiple choice, the student either answers correct or wrong. Writing and speaking, however, are more subjective, according to the teacher, that is when a rubric becomes necessary. However, students at this age do not get assessed by rubrics often, because they are still building knowledge. The rubric usually starts getting used at the third year, while these students were second year.

In both cases a rubric was used to assess the work of the students. However, the rubric of case 2 was more in-depth, better distinguishing between the different levels per category. Case 2 also considered the presentations as part of the assessment.

4.3.5 Teacher's opinion and suggestions

In this part, the teacher's opinion and suggestions are central. In case 1 the teacher stated that in order to integrate computational thinking in English in a good way, he himself has to improve his computational thinking and programming skills as well. In order to compensate for this, the teacher let the students that were more advanced in programming help out other students. The teacher also states that, because more skills are required, the planning could have been better: "Giving them the tools so that they can actually plan everything better. And give them more detailed steps, create more steps for them . . . I think the jump from English to computational thinking was not too big, but it could have been way smoother than it was, I think that would be something that I would have liked too". Even though the students really enjoyed it, it required a lot of organising. However, if there is an infrastructure, then less organising is required. So it costs a lot of energy for the teacher. Lastly, the teacher wondered in what year it is best to integrate computational thinking. He thought about third year students, since those students can program already.

In case 2 the teacher starts by saying that he really enjoyed this experiment and he thinks it is a good way to teach English. The teacher has a saying that "kids consume a lot nowadays but they do not produce a lot", but in this experiment the students were really producing. With these type of assignments, the teacher states that he has to be very clear on what to do, but also what they do not have to do. That area in between is where the students' freedom to be creative lies. So put clear limits, but also give them the freedom within those limits. This is a type of project-based learning. Normally the teacher would explain the concepts Freytag's Pyramid and foreshadowing to the students and then make them recognise it within different stories, but in this project they had to create their own stories with these concepts included. This was a better way of teaching according to the teacher: "So I think this was a different way and it probably is going to stick more because when you produce something it tends to go into your long term memory. So I think this was a stronger assignment, I guess."

Freytag's Pyramid is something that is usually taught in the fifth or sixth year. Some students do not even get this during high school at all, but later at the university when they study literature. However, with this experiment it was possible: ". . . because in actuality it is not that difficult, people just put too many things around it. But we found a simple way of teaching them the principle and then you can build on that and make it how difficult you want."

The teacher wants to incorporate more of these types of assignments: "... because then you are really busy with the language. And it is not that abstract, you are really applying the language and it does not even matter if they make mistakes, because they communicate using the language." When asked about the pronunciation, the teacher thought it was a good idea to record their voices, because in real time it is always more nerve-wrecking. It also gives the students to correct and hide their faults, which cannot be done in real time. In addition, this was a collaborative project, it also encouraged students to help each other spot mistakes and correct them. When asked what aspects could be improved, the teacher said that students were struggling with Scratch. However, if the students fully "dominate" Scratch, less time would be wasted on figuring out how it works and more time could be spent on the digital story, the actual assignment. At some point, the students were learning Scratch as they were working on the assignment. Consequently, the teacher thinks that the better the students understand Scratch, the higher they would score. Also, providing the students with an example model is something the teacher would add. With this example the teacher shows the students what they eventually need to reach. Lastly, the teacher suggest some from of weekly evaluations. This should let the students (and also the teacher) know whether they are on track in terms of progression. Every evaluation should be a step closer to the goal. These evaluations can also provide points for the students, to motivate the students. The example the teacher gave was: "The third week you need to be here and then we are going to evaluate and see what you need and then give you new information so you can continue".

Next, the teacher was asked to compare the assignment of case 1 to the assignment of case 2. The teacher preferred the assignment of case 2: "I think the assignment connected better with computational thinking. Because the other one, it was broader and they had to do the same steps as well. I think because it is a specific thing, the assessment for us was clearer. The differences are not that huge, because it is more limited". The teacher also said that he would like to integrate more computational thinking in his lessons, because "it helps them think about their steps". This can be especially valuable with these students, because it is not something that they do naturally.

In short, the teacher was generally more positive about case 2 than case 1, both in terms of English itself and computational thinking. In case 1 the teacher had some concerns about these integrated lessons and doubted his own skills to be able to teach this to his students. The teacher had a lot of question marks in case 1, maybe because this was his (and the students') very first time with the computational thinking integrated lessons. In case 2 the teacher had a lot more positive things to say and some still negative things. Scratch could take less time in if the students are more familiar with it. But in the end the students learnt what they were supposed with the use of Scratch and computational thinking. The teacher was so enthusiastic in case 2 that he even wanted to keep on incorporating computational thinking into his future lessons.

5 Conclusions

The aim of this thesis was to find out and describe the differences and/or similarities between two cases in which computational thinking was integrated in the subject English. In the first case students were instructed to create a digital story on Scratch where they introduced and described themselves. In the second case the same class was again instructed to create a digital story on Scratch, but this time creating a unique story including user interaction, foreshadowing and Freytag's Pyramid (from an introduction to a climax to a catastrophe). Four research questions (section 1.3), each of them covering a dimension of the lesson series, were created to help find that out: English learning outcomes, computational thinking learning outcomes, students' attitude and the teacher's attitude. This chapter will provide an answer to each research questions. Then the limitations will be listed, lastly, there will be recommendations for future studies.

English thinking learning outcomes

The first research question in this thesis was: *How do the English learning outcomes of the students differ in two computational thinking integrated English lessons?*

The results of this research show that the digital stories of case 2 were far richer than the digital stories of case 1. The students were offered more freedom in case 2, in contrast to all the requirements of case 1. With this freedom, the students in case 2 all came up with unique and distinct digital stories, fully using their creativity. The digital stories of case 1 were questions & answers, whereas the digital stories of case 2 were actual stories with a plot where each character played a role in that plot. The students of case 2 also used this freedom to improve the presentation of the digital stories. Those students used their voices with emotions as part of the story, instead of using it as just a medium to communicate in case 1. Also visually, the students of case 2 stood out, the locations actually mattered in the story and the characters interacted with it. Lastly, the number of mistakes was less in case 2 than in case 1, even though the average length of a project was higher in case 2 than in case 1. Overall, the English used by the students in case 2 was more diverse, creative and had less mistakes.

A possible explanation for the increase in creativity is the given freedom to the students in case 2. Chan et al. [CCC+17] state that the freedom to select themes promotes creativity. They also state that if stories are created in a group, people tend to be more involved because they want to include their ideas into the stories. If the students are more involved, they tend to pay more attention and possibly make less mistakes.

Computational thinking learning outcomes

The second research question of this thesis was: *How do the computational thinking learning outcomes of the students differ in two computational thinking integrated English lessons?*

In almost every way the Scratch projects of case 2 were more complex than the Scratch projects of case 1. The students in case 2 decomposed their stories into smaller subplots and this is clearly visible in the (number of) scripts. In case 1 the students had very long scripts, which went on from beginning until the end of the story (linear), while the students from case 2 had multiple scripts per sprite, like a flow-chart. There was also an increase in the number of sprites and backdrops per

project. There was just only one project in case 1 (C1P4) that stood out a bit, having multiple sprites and backdrops and having movement. This was also the only project where the digital story was not in the form of a questions & answers, the questions were integrated in a story. The students from C1P4 might have had previous experience with Scratch (or any other programming language), which can be the reason why their project was a standout. The students in case 2 also made use of code blocks of Scratch and programming concepts that the students did not use in case 1, like conditions.

This result may again be explained by the fact that the requirements were better fit with the programming and computational thinking concepts. The teacher also mentioned this in the interview, saying "I think the assignment connected better with computational thinking" when he was asked to compare case 1 to case 2. The students were, for example, instructed to create a storyboard first, which is a flow-chart on paper, before creating their story in Scratch. This requirement can be seen as a form of decomposition and a step-by-step thinking process. Another requirement was to include user input, which had to have an effect on the story. This requirement forced the students in case 2 to use conditionals, where they had to think (in the previously mentioned storyboard) how the story should proceed according to the given input of the user. Another possible explanation for this might be the fact that in case 2 it is not their first time using Scratch. The students in case 1 were in the first grade of secondary school, so it is very likely that they did not have (that much) experience with Scratch. Thus the students had more experience with Scratch a year later in case 2, which might have resulted in higher confidence because they were (more) familiar with it.

Students' attitude

The third research question in this thesis was: *How do the attitudes of the students differ in two computational thinking integrated English lessons?*

The results of this part showed a mixed change in their attitudes, both positively and negatively. The result of the exit tickets indicate that the students enjoyed and understood the lessons more in case 1 and that they found it more interesting. In the students' interviews the results were moderately similar. All the students in case 1 and case 2 had something positive and negative to say about Scratch, the negative comments concerning technical issues and Scratch being too difficult. Similar comments were made when they were asked about the lessons. Lastly, the students in case 2 were asked to compare it to case 1. The students were more satisfied with their digital stories and the teacher's explanations. They were also forced to plan and work better, because the time given was not enough to complete their digital stories. They also noted that there was still room for improvement when it comes to group collaboration.

There may be several possible explanations for the fact that the students' attitude did not change much. Decomposition is, according to Selby [Sel15], the most difficult computational thinking skill to master. Since decomposition had a (greater) emphasis in case 2, this might be the reason that the students still found it difficult, despite the increase of quality of their end products in case 2. Saidin et al. [SKM⁺21] also stated the students' acceptance of computational thinking is one of the challenges of integrating it into education. Many aspects need to be considered, such as gender, thinking ability, education level and the teacher's ability to implement computational

thinking. The teacher has stated multiple times, directly and indirectly, in the interview that he had difficulties with teaching this way. Within a classroom of 30 students it can be expected that there is a difference in programming experience between them. Both cases however did not take this into consideration, every student got the same package.

Teacher's attitude

The last research question in this thesis was: *How does the attitude of the teacher differ in two computational thinking integrated English lessons?*

On this question, the results show that the teacher was more satisfied with the assignment of case 2. The teacher said that in case 1 the students found it difficult, because additional skills (not directly related to English) were required to do this assignment and this also made it more difficult for him to guide his students, as this was new for him as well. In case 2, the teacher also noticed that the students had less difficulty and enjoyed it more. The teacher was more positive about the integration in case 2 than in case 1 and he wants to keep integrating it into his future lessons because it helps them think about their steps. In case 1, the teacher had concerns about this integration and he also doubted his own skills.

The observed change could be attributed to the teacher's better understanding of computational thinking. According to Sands et al. [SYG18], teachers have incorrect ideas about computational thinking. That study showed that teachers have very little knowledge about what computational thinking skills are and how they can implement these skills in their subjects. Consequently, case 2 being the second time for that teacher to experience the integration of computational thinking might have given him a better idea of what computational thinking actually is. Likewise, the teacher might have a better understanding of the benefits that come with the integration, which might be the reason why he was more positive towards the integration of computational thinking in case 2.

5.1 Limitations, future direction and conclusion

During this research, there were some limitations. First, these case-studies were not specifically designed to be analysed cross-case later on. Hence, when the case-studies were designed and executed, the researchers did not envision doing a cross-case analysis. As a result of this, the data of case 1 and case 2 were not the same. Therefore, a lot of data was not used for this research because there was no equivalent for it to compare it to from the other case. In addition, the participants of case 2 were just a fraction of the participants of case 1, which had a lot more participants. Consequently, the data of the participants that were not present during case 2 are also not considered. These unused data may have provided this research new insights and/or (slightly) different results. Second, for the teacher's attitude this research had only access to the data of one teacher. If the teacher's data are outliers, the corresponding results are not representative of the average.

A further study in the form of a pre-designed cross-case analysis is suggested. This cross-analysis could benefit from a larger case size. The data of such a research could benefit if all cases have the same data, in that case all the data that is collected will be used. The cases can be from other school subjects or education levels. This would be a cross-case analysis between different subjects of the same students or comparing it at different education levels, where each case is an education level.

In conclusion, this cross-case analysis showed an interesting difference between two subsequent integrations of computational thinking in English. Especially when looking at the learning outcomes of the students, there is a substantial increase in performance. The students were able to create more complex stories in a digital environment with the use of computational thinking compared to the first year. There is still room for improvement on the attitudes of both the teachers and students. This might however disappear over time, since this integration is still a relatively new phenomenon. On the whole, the evidence of this study suggests that, if done right, integrating computational thinking in English is an efficient way of teaching.

References

- [ABBH10] Vicki Allan, Valerie Barr, Dennis Brylow, and Susanne Hambruch. Computational thinking in high school courses. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 390–391, 2010.
- [AH21] Rian Andrian and Rizki Hikmawan. The importance of computational thinking to train structured thinking in problem solving. *Journal Online Informatika*, 6(1):113–117, 2021.
- [Ali15] Halah Ahmed Alismail. Integrate digital storytelling in education. *Journal of Education and Practice*, 6(9):126–129, 2015.
- [BB19] Erik Barendsen and Martin Bruggink. Het volle potentieel van de computer leren benutten: Over informatica en computational thinking. Available at <https://www.van12tot18.nl/het-volle-potentieel-van-de-computer-leren-benutten-over-informatica-en-computational-thinking> (accessed 24/06/2022), Dec 2019.
- [BR12] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, volume 1, page 25, 2012.
- [CCC+17] Banny SK Chan, Daniel Churchill, Thomas KF Chiu, et al. Digital literacy learning in higher education through digital storytelling approach. *Journal of International Education Research (JIER)*, 13(1):1–16, 2017.
- [CTHE19] Niklas Carlborg, Markus Tyrén, Carl Heath, and Eva Eriksson. The scope of autonomy when teaching computational thinking in primary school. *International journal of child-computer interaction*, 21:130–139, 2019.
- [DB21a] Lucas De Bruin. Analyse van de opvattingen van docenten in het nederlands basis en voortgezet onderwijs over de vakintegratie van computational thinking. 2021.
- [DB21b] Lucas De Bruin. Integrating computational thinking in economics through the use-modify-create strategy within the context of computer models. 2021.
- [Gon15] Marcos Román González. Computational thinking test: Design guidelines and content validation. In *Proceedings of EDULEARN15 conference*, pages 2436–2444, 2015.
- [Guz08] Mark Guzdial. Education paving the way for computational thinking. *Communications of the ACM*, 51:25–27, Aug 2008.
- [Har21] Alinda Harmanny. Computational thinking in context. 2021.
- [Jen15] Craig Jenkins. Poem generator: A comparative quantitative evaluation of a microworlds-based learning approach for teaching english. *International Journal of Education and Development using ICT*, 11(2), 2015.

- [KK] Maria Kordaki and Panagiotis Kakavas. Digital storytelling as an effective framework for the development of computational thinking skills.
- [LC12] Kien S Lee and David M Chavis. Cross-case methodology: Bringing rigour to community and systems change research and evaluation. *Journal of Community & Applied Social Psychology*, 22(5):428–438, 2012.
- [LC18] Yifan Liu and Min Chen. From the aspect of stem to discuss the effect of ecological art education on knowledge integration and problem-solving capability. *Ekoloji*, 27(106):1705–1711, 2018.
- [LM21] Michael Lodi and Simone Martini. Computational thinking, between papert and wing. *Science & Education*, 30(4):883–908, 2021.
- [LR18] Anna Lamprou and Alexander Repenning. Teaching how to teach computational thinking. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 69–74, 2018.
- [LSd⁺20] Yeping Li, Alan H Schoenfeld, Andrea A diSessa, Arthur C Graesser, Lisa C Benson, Lyn D English, and Richard A Duschl. On computational thinking and stem education, 2020.
- [MG06] Allan Martin and Jan Grudziecki. Digeulit: Concepts and tools for digital literacy development. *Innovation in teaching and learning in information and computer sciences*, 5(4):249–267, 2006.
- [MM16] Dr Mahsa Mohaghegh and Michael McCauley. Computational thinking: The skill set of the 21st century. 2016.
- [Nad21] Amine Nadif. Computational thinking integreren in het schoolvak engels: verhaal vertellen via programmeren in scratch. 2021.
- [nwo] Computational thinking in context: A teaching and learning trajectory for primary and secondary education. Available at <https://www.nwo.nl/projecten/40518540153-0> (accessed 24/06/2022).
- [Pap80] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.
- [PCWH21] Nadia Parsazadeh, Pei-Yu Cheng, Ting-Ting Wu, and Yueh-Min Huang. Integrating computational thinking concept into digital storytelling to improve learners’ motivation and performance. *Journal of Educational Computing Research*, 59(3):470–495, 2021.
- [Rob] Bernard Robin. What is digital storytelling? educational uses of digital storytelling. Available at <https://digitalstorytelling.coe.uh.edu/page.cfm?id=27&cid=27> (accessed 24/06/2022).
- [SDF18] Pratim Sengupta, Amanda Dickes, and Amy Farris. Toward a phenomenology of computational thinking in stem education. *Computational thinking in the STEM disciplines*, pages 49–72, 2018.

- [SDMJ18] Barbara Sabitzer, Heike Demarle-Meusel, and Maria Jarnig. Computational thinking through modeling in language lessons. In *2018 IEEE Global Engineering Education Conference (EDUCON)*, pages 1913–1919. IEEE, 2018.
- [SDS14] Najat Smeda, E Dakich, and Nalin Sharda. The effectiveness of digital storytelling in the classrooms: a comprehensive study. *Smart Learning Environments*, 1(1), 2014.
- [Sel15] Cynthia C Selby. Relationships: computational thinking, pedagogy of programming, and bloom’s taxonomy. In *Proceedings of the workshop in primary and secondary computing education*, pages 80–87, 2015.
- [SKM⁺21] ND Saidin, F Khalid, R Martin, Y Kuppusamy, and NA Munusamy. Benefits and challenges of applying computational thinking in education. *International Journal of Information and Education Technology*, 11(5):248–254, 2021.
- [SW13] Cynthia Selby and John Woollard. Computational thinking: the developing definition. 2013.
- [SYG18] Phil Sands, Aman Yadav, and Jon Good. Computational thinking in k-12: In-service teacher perceptions of computational thinking. In *Computational thinking in the STEM disciplines*, pages 151–164. Springer, 2018.
- [TCT20] Kai-Yu Tang, Te-Lien Chou, and Chin-Chung Tsai. A content analysis of computational thinking research: An international publication trends and research typology. *The Asia-Pacific Education Researcher*, 29(1):9–19, 2020.
- [TKHS21] Karin Tengler, Oliver Kastner-Hauler, and Barbara Sabitzer. Enhancing computational thinking skills using robots and digital storytelling. In *CSEdu (1)*, pages 157–164, 2021.
- [TT19] Mark Timmer and Jos Tolboom. Computational thinking—een vooruitblik op het wiskundeonderwijs van de toekomst. *Nieuw archief voor wiskunde*, 5(1):42–45, 2019.
- [VFG⁺15] Joke Voogt, Petra Fisser, Jon Good, Punya Mishra, and Aman Yadav. Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4):715–728, 2015.
- [VR19] Maryam Vaezi and Saeed Rezaei. Development of a rubric for evaluating creative writing: a multi-phase research. *New Writing*, 16(3):303–317, 2019.
- [Win06] J.M. Wing. Computational thinking. *Communications of the ACM*, 49:33–35, 2006.
- [WSC21] Changzhao Wang, Ji Shen, and Jie Chao. Integrating computational thinking in stem education: A literature review. *International Journal of Science and Mathematics Education*, pages 1–24, 2021.