Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A computer-based learning environment aimed for students at the 3rd and 4th grade level

Bachelor's Thesis

Florian Bütler
flbuetle@ethz.ch

Chair of Information Technology and Education
ETH Zürich

Supervisor:
Prof. Juraj Hromkovič

February 15, 2021

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.

Acknowledgements

I thank Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abstract

a brief introduction describing the discipline that the paper belongs to a clear and concise statement of your problem a brief explanation of your solution and its key ideas a brief description of the results obtained and their impacts

Contents

Chapter 1

Introduction

1.1 Motivation and Background

With the introduction of Lehrplan 21 Computer Science became an integral part of the Swiss education curriculum [**?**]. Pupils learn to understand the basic concepts of Computer Science and how to use them for problem solving. These concepts include methods on how to process, evaluate and summarize data, how to securely communicate and how to develop solution strategies for simple problems of information processing [**?**]. The Education and Counselling Center for Computer Science Education at ETH Zurich (ABZ) supports schools to teach these concepts among others by providing teaching resources and learning environments.

Todo: mention text book where the exercises are taken from

1.2 Goals

The main goal in this bachelor thesis is to implement tasks and riddles based on the textbook "einfach Informatik 3/4" in a computer-based learning environment that teaches the following concepts:

- representing information with symbols,
- keeping information secret and
- learning from data

for pupils in the second cycle. Along with solving tasks and riddles about the mentioned topics the ability of reading, writing, counting and calculating is trained as well.

1.3 Related Work

Todo: abz.inf.ethz.ch other learning environments

1.4 Outline

This report first explains how the aforementioned concepts are thought by hands-on exercises, then gives in-depth technical insight on how a learning environment is developed and how these exercises are implemented. Finally, the report ends with a conclusion with a review of the project.

Chapter 2

Concepts

This chapter is split into three parts. Each part discusses a basic concept of Computer Science and its corresponding exercises. Throughput all exercises beavers are used as protagonists to illustrate the concepts. The beaver comes from a project of the Swiss Association for Information Technology in Education and is intended to arouse interest in Computer Science for children and teenagers between 8 and 20 years old [**?**].

2.1 Representing Information with Symbols

Representing information with symbols is a fundamental concept of Computer Science as information should be represented clear and concisely. Words can be seen as a sequence of symbols, namely a sequence of letters.

2.1.1 Similar Words

Transmitting information includes representing it in a message, sending it to the destination and the receiver being able to make sense of it even tough the message might contain errors such as a spelling mistake. To achieve this sender and receiver agree to only send messages with a minimal editing distance [**?**] between each of them.

Editing distance

The editing distance is the amount of operations that need to be done to transform a message in another. Operations are deleting, inserting and changing a letter . A cost function exists that defines the cost of each operation, and in our case each operation has a cost of 1 (unit cost model). The editing distance is then the minimal cost to transform a message into another one by a sequence of operations. In our case a message is a word.

Todo: add some TI explanation

Example 2.1. The editing distance between `LIKE` and `BIKE` is 1, since changing the first letter from an `L` to and `B` transforms the first word into the second.

If sender and receiver agree to only transmit words with a minimal editing distance of e.g 3, then the receiver can still uniquely determine what word the sender has sent even when at most 1 spelling mistake has been made. The receiver calculates the minimal editing distance between the received word and each of the agreed words and chooses the word with the least editing distance. The receiver assume that this was the word the sender wanted to sent. This of course work only if there are not more than 1 error in the word. If this happens the editing distance to another word is closer than to the original word and hence the word is misinterpreted. To counter this, sender and receiver might agree on a bigger minimal distance, but this comes with a tradeoff. When chosing a bigger minimal distance for a fixed alphabet, the number of words that can be used shrinks. To maintain the same amount of words the size of the lphabet can be increased too, resulting in longer words.

The purpose of the `similar words` exercises is to learn these operations. Therefore an exercise is dedicated to each opertion: adding, changing and removing a letter from a word. Additionally, an exercise about swapping adjacent letters in a word is included. Swapping adjacent letters is not part of the mentioned operations and usually consists of 2 operations: removing and adding or changing twice. However, when typing on a keyboard typing mistakes happen often and most of the time only two adjacent letters are swapped. This exercises is supposed to train the ability to spot these sort of mistakes.

Adding a letter

In the `adding a letter` exercise pupils are presented a word, the alphabet from A to Z and spaces between each letter of the word as well as at the beginnig and the end of the word, where a letter can be added. Pupils are supposed to choose a letter from the alphabet and add it to one of the mentioned spaces to form a new valid word.

Example 2.2. The word `PACE` is given. By adding the letter `S` before the first letter, the valid word `SPACE` is formed.

Changing a letter

In this exercise there is again a word and the alphabet from A to Z shown. Again pupils should select a letter from the alphabet, but this time, instead of adding it to a space, the selected letter should be replace a letter from the word itself to create a new valid word.

Example 2.3. The word `BIKE` is given. By changing the first letter from a `B` to a `L` the new valid word `LIKE` is formed.

Removing a letter

In this exercise pupils receive a word and they should select a character from within the word an move it to the trashcan to remove the letter from the word itself to form a new valid word.

Example 2.4. The word `SPACE` is given. By removing the first letter the new valid word `PACE` is formed.

Swapping two adjacent letters

To learn to recognize typing mistakes in a word pupils are presented a word with swapped adjacent letters and they are supposed to identify those and swap them back to restore to original word. Here are multiple difficulty levels possible, where on the easy level only one pair of adjacent letters is swapped and on harder levels multiple pairs of adjacent letters are swapped.

Example 2.5. The word `BKIE` is given. By swapping the second and third letter the original word `BIKE` is restored.

2.1.2 Representing Numbers like the Maya

Today one is used to the decimal number system (base number 10), with 10 symbols. The Maya used a different number system with with a base number of 20 i.e the vigesimal number system. It is believed that the Maya used their ten fingers and ten toes to count adding up to twenty. But instead of having a symbol for each number as in the decimal system, the Maya used three different symbols: zero (a turtle shell, belly side up), one (a dot) and five (a bar) [**?**]. The mayan representation of the decimal number from 0 to 19 can be found in figure **??**.

The following exercises are supposed to familiarize pupils with a new number system, the mayan number system. First converting numbers between the decimal number system and the mayan numbers is trained topped of an exercise about adding two mayan numbers.

Representing mayan numbers

A decimal number is presented to the pupils and they need to choose the correct amount of dots and bars representing the decimal number. For the purpose of simplicity, only decimal numbers between 1 and 19 are chosen.

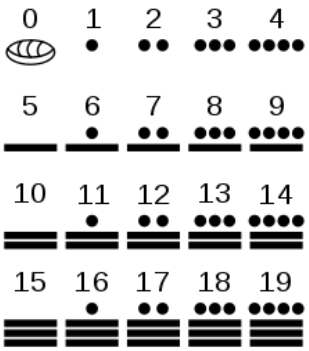

Figure 2.1: Representation of the Maya numbers from 0 to 19

Example 2.6. The decimal number 17 is writte as three bars and two dots.

Understanding mayan numbers

In this exercise the pupils learn to understand mayan numbers. Again only decimal numbers between 1 and 19 are used and the pupils need to understand what mayan number is shown and write down the decimal equivalent.

Adding mayan numbers

To sum this section up, adding mayan numbers is learned. This exercises combines the previous two, since the pupils need to understand the summands given in the mayan number system, add them and write the sum down in either decimal number system or the mayan number system.

2.1.3 Representing Numbers with Coins

The following exercises introduce a new number system and trains an already learned one: the binary number system and the decimal number system. Both number systems are praticed with coins with numbers. The binary coins include the following numbers: 1, 2, 4, 8, 16, 32 and 64 (Fig. **??**). The decimal coins include 1, 2, 5, 10, 20 and 50 (Fig. **??**). Overall, the same concepts are praticed for both number systems with the limitation that every binary coin can at most be used once. For both number systems the following exercises are given:

- Conversion of a decimal number to its coin representation
- Conversion of a number given in its coin representation to a decimal number

Additionally, for the decimal number system the following exercise is given:

Figure 2.2: Representation of the decimal coins

Figure 2.3: Representation of the binary coins

- Reducing the number of decimal coins in a given set of decimal coins

Conversion of a decimal number to its coin representation

This exercise includes two difficulty levels:

- Converting a decimal number to a coin representation, where the sum of all coins are equal to the decimal number
- Converting a decimal number to a coin representation, where the sum of all coins are equal to the decimal number **and** the amount of used coins is minimal.

Conversion of a number given in its coin representation to a decimal number

This exercise is straigh forward. A number given in its coin representation, either binary or decimal coins, have to be converted to a decimal number.

Reducing the amount of decimal coins in a given set of decimal coins

Similar to the previous exercise, a number in its coin representation is given and pupils have to display the same number but with less coins.

2.2 Keeping Information Secret

The previous section **??** was about representing information with symbols. This section is about keeping information secret. Ciphers haven been used for thousands of years [**?**]. They are used to keep information secret from people, that are not supposed to have knowledge of it. Not encrypted information is called

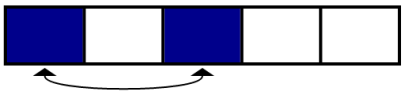

Figure 2.4: Pattern

Figure 2.5: Cipher text of an encrypted number

clear text. Once one encrypted a clear text, it is called a cipher text and only people who know how to decrypt the cipher text can read originally encrypted information. The exercises in this section are introducing pupils to the concepts of ciphers.

2.2.1 Cipher Texts from Reversed Letters

The cipher used in these exercises is a simple mix up of letters and both directions are trained: encryption and decryption. In the decryption exercise, the pattern, on which the clear text was encrypted with, is shown. The pupils need to understand the pattern and move the letters in the cipher text accordingly to retrieve the clear text. The encryption exercise is set up analogously. Multiple difficulty levels are possible by changing the amount of moved letters.

Example 2.7. The cipher text is `ULFSS` and the pattern is shown in figure **??**. By moving the letter in the cipher text according to the pattern, the clear text can be retrieved: `FLUSS`.

2.2.2 Cipher Texts from New Characters

Sometimes, only moving symbols in not enough to keep information secret. A better why is to substitute symbols with new symbols. These symbols may be letters, numbers or completly new symbols, that are solely invented for the purpose of encrypting information. In the following exercises the last approach is followed. Again both direction, encryption and decryption, are trained. But this time, instead of having a pattern, there is a symbol table showing how the letters are encrypted.

Example 2.8. The cipher text is shown in figure **??** and the symbol table in figure **??**. By using the symbol table one can decrypt the cipher text to `52`.

	·	:	⁝	⁘
□	0	1	2	3
○	4	5	6	7
△	8	9		

Figure 2.6: Symbol table to encrypt numbers

Figure 2.7: Trees from height 1 to 3

2.3 Learning from Data

The following exercises are logic-based, combinatorial puzzles. The idea is that pupils need to conclude the solution by a partially completed grid.

2.3.1 Row of Trees

The row of trees is a one dimensional grid i.e a row either of size 3 or 4. In every row there is exactly one tree of every height between 1 and 3 (Fig. **??**) or 4 (Fig. **??**) respectively. At both ends of the row the amount of tree visible from this end of the row is given. Pupils are given an empty row with only the number of trees seen from both ends of the row and need to place a tree of each height such the aforementioned rules are met.

Example 2.9. For a row of size 3, if given the value 2 for both ends of the row, then one possible solution to the puzzle is shown in figure **??**.

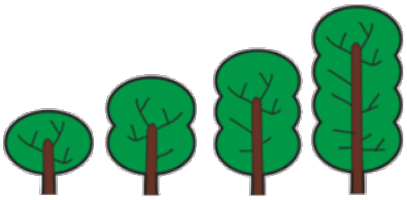

Figure 2.8: Trees from height 1 to 4

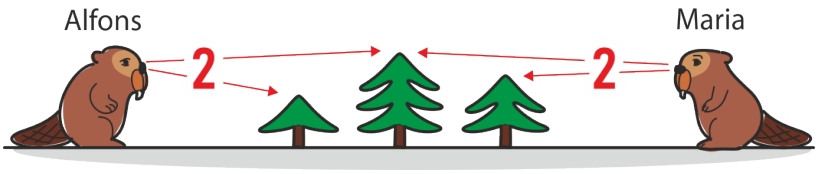

Figure 2.9: Visualization of the puzzle solution from the tree row example

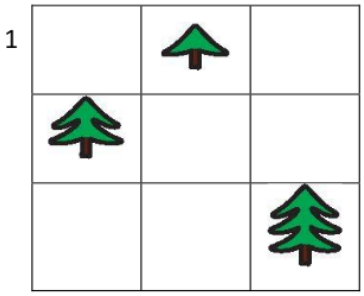

Figure 2.10: Example of an unsolved 3x3 tree sudoku

2.3.2 Tree Sudoku

Tree sudoku is similar to the well known traditional sudoko with the difference that trees of different heights are placed instead of numbers, and for end of every row and collumn the number of visible trees is given. Otherwise, the puzzle follows the same rules as the row of trees and is either of size 3x3 or 4x4.

Example 2.10. An example of a 3x3 tree sudoku is shown in figure **??**

Chapter 3

Implementation

3.1 Introduction

3.2 TypeScript

JavaScript had its publication in 1996, is a scripting language and is intended to be used in browsers to extend the possibilities of HTML and CSS. It can dynamically manipulate HTML and CSS, validate user data, send and receive data without reloading the page and much more. TypeScript extends JavaScript by adding types and can be used anywhere JavaScript runs, because TypeScript code is transformed to JavaScript code by the TypeScript compiler. It provides a way to describe what type a variable has and helps to catch errors before the code is run. Moreover, TypeScript adds among others the concepts of method signatures, type interference, interfaces, enummerations and tuples. [**?**].

3.3 Vue.js

Vue.js is not developed by a company, but by Evan You and was first published in 2014. It is an alternative to Angular and React and was supossed to be a lightweight version of Angular. Vue.js is also based on reusable components, each having its own HTML, Javascript and CSS. At the point the development of this project started, Vue.js version 3 was already published. However, Vue.js version 2 is used in this project, because the ecosystem has no caught up yet and many libraries only work with version 2 at the moment [**?**].

3.3.1 Components

Components are named resuable Vue.js instances and have the advantages that the structure, functionality and style of an element is implemented once and can then easily be used multiple times. Therefore each component has a parent (ex-

cept from the root component) and possibly multiple child components forming a tree structure [?].

3.3.2 Communication between Components

Communication between a parent and a child component happens from the parent to the child by properties and from child to parent by events. Properties are custom attributes that pass data from the parent to the child component. Events are emitted by a child component, can carry data and a parent can listen and react upon receiving an event from a child component [?].

3.3.3 Best Practices

The following list represent the best practice that were followind in this project. To see examples for each best pratice visit the Vue.js style guide [?].

- Properties should be as detailed as possible and at least have a type.
- Always use `key` with `v-for` to maintain the internal component state.
- Avoid `v-if` with `v-for`
- Only the top-level `App` component and layout components should have global styles. All other components should always have scoped styles i.e the styles is only used within the component.
- Each component is in its own file.
- Filenames are in Pascal-Case.
- Components without any content should be self-closing e.g `<Component />` instead of `<Component><Component/>`.
- Components name casing in templates is PascalCase.
- Properties name casing is camelCase.
- Elements with multiple attributes should span multiple lines, with one attribute on each line.
- Component templates should only contain simple expressions. Complex expressions should be moved into computed properties or methods.
- Element attribute values should be quoted.
- Directive shorthands are always used.

- Element attributes should be ordered consistently.
- Components should be ordered like `<templates>`, `<script>` and `<style>`.
- Element selectors should be avoided with `scoped`
- Properties and events should be used for parent-child communication.

3.4 Structure

The code for the project is located in the **src** folder. The source folder has the following content:

- **assets** - all the images and videos used in components
- **components** - all components of the project that are not a view
- **router** - everything in Vue.js is a component, but not everything is page. A page needs a route e.g. `/settings` and this folder contains all routes of the project. Components with a route are called **routed**
- **views** - all routed components

3.5 Basic functionality

3.5.1 Home Screen

The home screen is a view and is the first page seens when visiting the learning environment. It has a simple structure: For each available exercise there is a card with an image to illustrate the exercise and its title. The basic ideas for the images are taken from the text book, but most of the time needed some simple photoshopping to make it represent the task reasonably.

3.5.2 Game Mixin and Game Interface

Game Mixin

Mixins can be used to reuse functionality over different Vue components. When a component uses a mixin, all functionalities of the mixins are mixed into the component. All exercises use the `GameMixin` containing mostly the functionality on start and evaluate an exercises. Additionally, a function to generate a random number exists to mock random number generation in tests. More on that later.

Game Interface

Each exercises implements the GameInterface. This ensures that exercise specific functionality like starting and evaluating the exercise is implemented and can be used in the `GameMixin`.

```
1 interface GameInterface {
2     isInitialized(): boolean;
3     start(): void;
4     isCorrect(): boolean;
5 }
```

Listing 3.1: GameInterface

3.5.3 General Purpose Components

The following presented components are components that are heavily reused. Most of them are part of the user interaction system since all exercises need some kind of user interaction elements.

Todo: images for each component

Game

The Game component is an essential

Todo: maybe elaborate on technical details like passing events

3.5.4 Event Bus

Todo: image about component tree structure

Sometimes components are not in a direct parent-child relationship and still need to communicate. This can still be achieved by passing properties and emitting events, but evolves quickly into an unclear structure. To tackle this problem one can use an event bus. The event bus is a vue instance that allows to to emit an event in one component and listen for that event on another component. To use the event bus, a component simply needs to import the instance and can emit or listen for event on the usual way. This functionality is necessary between the game buttons and each exercise.

Game Buttons

Every exercise needs a **check exercise** and a **next exercise** button to first check a given solution to an exercise and let the system check it and second to get to the next exercise.

Undo Button

The Undo button simply restores the current exercise initial conditions, so one can retry it again.

Trashcan

The Trashcan is an area where elements can be dropped to remove them. For example when pupils are asked to remove a letter from a word, they can either move the letter to this area and drop it or first click the element and then the trashcan are to remove it.

Difficulty Levels

Some exercises have multiple difficulty levels. For those exercise the Difficulty component is used to change the level. This component gives the possibility to choose to up to three different difficulty levels indicated by an increasing amount of beavers on the button and a title.

Tutorial

To introduce an exercise to the pupils the title of the exercise and a short instruction is not enough. Therefore for each exercise there is a detailed explanation of the background and purpose and a tutorial video. The tutorial video show an example run of first give a wrong solution, restart the exercise and finally give the correct solution. The tutorial video were recorded by a screen capture tool and the moves on how to solve the exercise are done programmatically. The reason for this is that moving the mouse by hand to solve the exercise introduces a jitter to the mouse movement. The tutorial on the other hand should show clearly and without any hesitation the way on how to solve the exercise. Since this cannot be done by an average human being, the mouse movement is done programmatically. Each graphical element part of the exercise has an ID and one can give a list of IDs that should be visited in a run. For some this is straight forward.

Remark: possible to elaborate more on how to generate the ID list

3.6 Styles

3.7 Representing Information with Symbols

3.8 Keeping Information Secret

3.9 Learning from Data

3.10 Testing

3.10.1 Unit and Snapshot Testing

3.10.2 End to End Testing

3.11 Continous Integration

Chapter 4

Conclusion, Drawbacks and Further Work

4.1 Conclusion

4.2 Drawbacks

- Not all exercises per topic
- Own learning environment

4.3 Further Work

- Inegrate into existing learning environment
- Add additional exercises

Appendix A

First Chapter Title

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Todo: This is a TODO annotation.

A.1 First Section Title

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

A.1.1 First Subsection Title

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Remark: This is a REMARK annotation.

Theorem A.1 (First Theorem)**.** *This is our first theorem.*

Proof. And this is the proof of the first theorem with a complicated formula and a reference to Theorem **??**. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

$$\frac{\mathrm{d}}{\mathrm{d}x}\arctan(\sin(x^2)) = -2 \cdot \frac{\cos(x^2)x}{-2 + (\cos(x^2))^2} \tag{A.1}$$

□

Lemma A.2. *lorem ipsum dolor sit amet*

Corollary A.3. *lorem ipsum dolor sit amet*

Observation A.4. *lorem ipsum dolor sit amet*

Definition A.5. lorem ipsum dolor sit amet

Problem A.6. lorem ipsum dolor sit amet

Assumption A.7. lorem ipsum dolor sit amet

Example A.8. lorem ipsum dolor sit amet

Claim A.9. lorem ipsum dolor sit amet

Remark A.10. lorem ipsum dolor sit amet

Note that in LaTeX, "quotes" do not use the usual double quote characters.