Gotscha! - One Way To Support Learning

Franz Knobel

Bachelor Thesis
January 8, 2020

Supervising Professor:
Prof. Dr. Juraj Hromkovič

Supervisor:
Elizabeta Cavar

ETH zürich

Abstract

As the importance of computer science is rising, certain games that support the developement of abstract thinking, analytical skills and decision making, are becoming more and more interesting at an early age. Through identification and describing relationships between items, children develop a foundation to early math skills and basic concepts of computer science (e.g. combinatorics of finite affine and projective spaces, the theory of error-correcting codes, hashing, etc.) The pupils individual learning speed and lack of concentration, if not receiving the right amount of attention, is another challenge by itself. Without individual fostering, children are at high risk of losing interest. Through this thesis the teachers will be introduced to a tool for their pupils. Focused on classification of objects with certain properties, the pupils get introduced to a computer-based learning environment. There they can individually train and improve themselves in this field. In the mean time the teachers can concentrate on the majority of their pupils and have the opportunity to work with pupils on an individual basis, without feeling the pressure of having to support everyone at once. It has shown, that gamification and game based learning has enormous potential. It takes time to learn how to create games. Once overcome, creating further games is not as hard as one may think. In the created test environment the test subjects have shown more concentration, individual work behaviour and more willingness to learn than in the whole group. Keywords : phaser 3, typescript, debugging, testing, from a topic to the game, how to tackle a new framework

Acknowledgment

I want to thank my supervisor Elizabeta Cavar for supporting me in the last six months. Without her input and constant reminder of our deadlines, I would have had a lot more obstacles on the way. Also many thanks to Jaqueline Staub, Nicole Trachsler and Philippe Voinov for their help with specific topics on my work. Another very important role in my work played the debuggers, the ones who tested my program and searched for any kind of problems. Thank you for your time, effort and fun you had with my work: Dilana Kunz, Guido von Burg, Julia Badertscher, Ronja Koeppel, Jonathan Chen

Contents

1

Introduction

1.1 Introduction

1.2 Problem Statement

the background on and motivation for your research - technological trends of the area - open problems - recent promising developments - introduce more specific terminology which is not widely known. - Provide good motivation for your work, - explaining its technological, research or economic importance - The motivation simply two or three good reasons are enough to make your research important.

a summary and outline of your paper, telling readers what they should expect to find in it.

- a problem description, which is slightly more detailed than in the abstract. - a description of your solution - some arguments on its impacts - key concepts and categorize its approach.

Close your introduction with a description of your pa- per outline - what sections it contains - what the reader will find in each.

A proper flow 1. context 2. present your proposal 3. provide the verification 4. conclusions.

As the importance of computer science is rising, certain games that support the developement of abstract thinking, analytical skills and decision making, are becoming more and more interesting at an early age. Through identification and describing relationships between items, children develop a foundation to early math skills and basic concepts of computer science (e.g. combinatorics of finite affine and projective spaces, the theory of error-correcting codes, etc.) . The pupils individual learning speed and lack of concentration, if not receiving the right amount of

attention, is another challenge by itself. Without individual fostering, children are at high risk of losing interest. Through this thesis the teachers should receive a tool for their pupils. Focused on classification of objects with certain properties, the pupils get introduced to a computer-based learning environment. There they can individually train and improve themselves in this field. In the mean time the teachers can concentrate on the majority of their pupils and have the opportunity to work with pupils on an individual basis, without feeling the pressure of having to support everyone at once.

Background

The key contribution of computer science to general school education is rooted in the concept of *algorithmic thinking* . One way of introducing kindergarten and primary school pupils to algorithmic thinking and it's concepts consists in making them solve problems with and without computers. This can be achieved using age- and knowledge-appropriate learning materials. Several papers with corresponding online learning environments have been proposed. This work is going to be added to the implementation of "INFORMATIK BIBER in KG und 1/2" by Jil Weber . The focus of this work will not solely lie on the translation of the existing teaching material to a computer-based learning environment, but also on introducing learning methods in a gamified environment which not only complements the teachers with their teachings but also assisting them.

Goals of the Thesis

The main objective of this thesis consists of planning, analyzing, implementing and testing a computer-based learning environment on the topic of classification. The student studies the already existing implementations of "INFORMATIK BIBER in KG und 1/2", analyses the capabilities of kindergarten kids and first graders, develops an interactive classification tool, implements it then on a platform compatible with the implementation of "INFORMATIK BIBER in KG und 1/2" and conducts an evaluation with test subjects. We expect the student to find a suitable implementation that integrates neatly into our existing system mentioned before. The outcome of this thesis is a well-documented, stable and reliable prototype, providing the functional elements to be used in schools.

1.3 Section

SECTION 0

2

Related Work

In this section we are going through some previous works in a the same area. Starting with mainly publications, in a second part we take look at gamification and game based learning and in a third part we look at existing educational software and highlight their pros and cons to build on the experiences made there.

2.1 Previous work

Sonja Tabea Blum Kevin Tang Sarah Kamp Jil Weber

2.2 Gamification and Game Based Learning (GBL)

4 5 6 7 8

2.3 Existing Educational Software

pro: player didnt recognize he was learning something con: every level needed the same level of knowledge in every category

====

2.4 Checklist

X - list of research works that are related to your paper—necessary to show what has happened in this field. X - critique of the approaches in the literature—necessary to establish the contribution and importance of your paper.

X distinguish and describe all the different approaches to the problem.

Critiquing the major approaches of the background work will enable you to identify the limitations of the other works and show that your research picks up where the others left off. This is a great opportunity to demon- strate how your work is different from the rest; for ex- ample, show whether you make different assumptions and hypotheses, or whether your approach to solving the problem differs.

DIE FRAGE: ein neues framework, einfach zu lernen, ja nein? aufwand, vorteile, nachteile

irgendwo mus ine, dass au lehrer abgneigt sind gege elektronik im taegliche gebruch. jedoch isches fact, dass dmehrheit vode chinder scho im junge alter mit dere war konfrontiert wird. de taeglichi gebruch isch sowieso da. wieso noed gad als guets bispil vorusgah und ihne spil schmackhaft mache wo ihne spass mached und si demit passiv lerned. motivation isch hoecher chind choennd scho hoechduetsch, bevor si ueberhaupt schwizerduetsch richtig choennd, will si mit duetscher sprach im gebruch vo elektronik konfrontiert werded. warum noed gad bruche als au soettigs und dete de vorteil drus zieh?! mr chan als bispil voragah ide informatik da das ersch jetzt gross ufchunnt, bi etablierte lehrmittel wird das schwiriger vorallem wills denne oftmals heisst, mit dem hanich scho glernt und das het funktioniert

3

Requirements

3.1 Boundary conditions

3.2 Evaluation of possible solutions

In this section we will capture the requirements for our learning environment. We will look into the cognitive and motor abilities of the target audience and some elements of gamification which will be analyzed and assessed. The existing environment our work should be implemented has conditions as well. Those will be taken into consideration while evaluating a suitable framework.

Target Audience

Elements of Gamification

Additional Conditions

Framework

human - kinder koennen nicht lesen und schreiben, keine groesseren zahlen und keine buchstaben

Game - capturing name - objects categories nad subcategories - easy level - easy level under harder difficulties - limited sorting -> hashing - set easy - set hard - overview - gamification - track progress - time based - correct and wrong - visual gratification - have fun dont recognize you are learning - optics - randomization and more options for variety - sound - story like - menu - levelmenu - fullscreen - flow - exitbutton - returnbutton -

3.3 Checklist

- explicitly describe all the hypotheses and assumptions of the environment on which the problem will be stated. - Put good effort in realizing all explicit and implicit assumptions that you make, and clearly state them. It is important to provide support for your assumption choices. The more valid and acceptable your assumptions are, the more valid and acceptable your work will be. The system model section should always have a figure. The figure should demonstrate the parameters of your system model. Prepare the figure so that it can later be reused or enhanced to demonstrate your solution.

Often, this section is merged with the system model. State your problem clearly. Be as exact as possible into stating what the question of the problem is. It reflects poorly upon an author if he cannot describe or does not know what problem his solution addresses. But most im- portantly, it will be easier for successive researchers to classify your work.

4

Design of the Learning Environment

In this section all implementation tools and approaches are explained. First our used framework will be explained, then how our different scenes and objects are generated and at last how these two are combined to our final solution.

4.1 Phaser 3

4.1.1 Scenes

4.1.2 Managers

4.2 Object generation

Our objects can have up to four properties with at maximum one from each of the following categories:

- Geometrical shape (square, triangle, circle, ellipse, rhombus, octagon)
- Color (yellow, orange, red, purple, green, blue)
- Holes or dots (one, two, three, four, five, six)
- Filling (filled, striped, dotted)

We need all possible objects with one, three and four properties.

With a python script [EXAMPLE] scalable vector graphic (SVG) files are generated [EXAMPLE]. After that they are converted to portable network graphic (PNG) files with the GNU image manipulation program (GIMP), so that they can be imported into Phaser.

4.3 Introduction Animation Generation

To produce animated instruction for the task later on, the screen is recorded with OBS Studio while someone (here me) is playing the game. Parts of the final video are then taken for the respective scenes and converted into single pictures. These single pictures are saved in one picture (called spreadsheet [EXAMPLE]). The phaser framework animates those spreadsheets.

4.4 Concept of the Learning Environment

Through the boundary conditions and requirements of the learning environment, we split our environment into multiple parts or here called scenes.

4.4.1 Drop Down Menu

Throughout the whole experience, the user can open a menu with a button. This button is always visible. Once the menu is open, the user may close the menu and return to the current scene, exit the current scene and return to the level menu and go into fullscreen and back. The current scene is paused while the menu is open as it can be distracting for the user, if suddenly something pops up and covers parts of the visible and running scene.

4.4.2 Welcome Screen

The welcome screen is the starting point of the user experience. Through this screen the user will be greeted by showing the name of the game. Through a click he shall commence to the level menu.

4.4.3 Level Menu

In the level menu, the user is able to choose between different levels and games. He can also access the object summary and track his progress/score. The progress of each individual level is shown on the respective buttons.

4.4.4 Object summary

The Object summary allows the user to get a feeling for all the different properties an object can have. For that reason the number of objects per category is restricted to five, as there are over 1000 different objects. Objects are draggable and sortable by each category with a click on a respective button.

4.4.5 Sorting with one category

Here the user has to sort the static objects with one category by the given category. As the user should be able to experience all categories, they are split into different levels. Each level represents a category. For motivation, the user can track his progress. The Progress is defined by objects sorted the right and the wrong way.

The amount of objects to sort, as well as the amount of properties of the category is randomly selected each time you start the game.

4.4.6 Sorting with one category under difficult conditions

Here the user has to sort falling objects with one category by the given category. As the user should be able to experience all categories, they are split into different level. Each level represents a category. For motivation, the user can track his progress. The progress is defined by objects missed, sorted the right and the wrong way. To make the task harder, dummy objects are added. Those objects look similar to the original ones but with a succinct characteristic. There are no negative points for missing such an object but negative points for sorting them in any way.

The amount of falling objects to sort, as well as the amount of properties of the category is randomly selected each time you start the game.

4.4.7 Sorting with restricted space

In this scene the user has to sort a given number of objects with all properties shown into boxes with limited space. The objects in one box must have at least one property incommon and can be put into the box and taken out an infinite amount of times.

To make the game more difficult, this level is split into two. The first level has boxes with the size 6, 4, 2 and the second one boxes with the size 6, 5, 4.

4.4.8 Object pairing - easy version

In this game, twelve objects with three categories are shown. the user has to select three objects which have to fulfil the following. For each category (color, shape and number of holes/dots) one of the following conditions has to hold:

- They must be the same (blue, blue, blue)
- They must be completely different (square, triangle, circle)

[EXAMPLE]

If the user needs help, there is a helper bar which can be accessed by a button with a question mark on it. The helper bar shows which categories fulfil the conditions and which do not by coloring the category symbols on the bar in green or red.

The game is time limited. The remaining time is shown by a bar. If you select three onjects which fulfil the conditions, more time will be added. After a set amount of correct selected objects, the game will end.

If there are no three objects that fulfil the conditions, the playfield will be generated anew.

4.4.9 Object pairing - normal version

In this game, twelve objects with four categories are shown. the user has to select three objects which have to fulfil the following. For each category (color, filling, shape and number of holes/dots) one of the following conditions has to hold:

- They must be the same (blue, blue, blue)
- They must be completely different (square, triangle, circle)

[EXAMPLE]

If the user needs help, there is a helper bar which can be accessed by a button with a question mark on it. The helper bar shows which categories fulfil the conditions and which do not by coloring the category symbols on the bar in green or red.

The game is time limited. The remaining time is shown by a bar. If you select three onjects which fulfil the conditions, more time will be added. After a set amount of correct selected objects, the game will end.

If there are no three objects that fulfil the conditions, the playfield will be generated anew.

4.4.10 Score Screen

After a task has been completed, there is going to be a score. This score is represented here with a displayed number of stars. The minimum of starts is zero and the maximum is three. If the user is unhappy with his results, he can replay the game by clicking on the replay button.

4.4.11 Introduction

Before each game starts, a sample animation of the task beforehand is being shown. The current scene is paused in the meantime, so that the user has enough time to find out what he has to do.

4.5 Code structure of the learning environment

[IMAGE OF STATE DIAGRAM]

Our learning environment consists of different scenes each one inheriting the base scene:

4.5.1 BaseScene

Listing 4.1: *BaseScene.ts*

```
1   export class BaseScene extends Phaser.Scene {
2       /**
3        * Name of the scene
4        */
5       protected key: string;
6
7       /**
8        * Level of the scene
9        */
10      protected level: number;
11
12      /**
13       * Transition graphic
14       */
15      private transition: Phaser.GameObjects.GameObject[];
16
17      constructor(key: string) {
18          super({
19              key: key
20          });
21
22          this.key = key;
23          this.level = 0;
24          this.generateNewSeed();
25      }
26
27      /**
28       * Method for returning the key of this scene
29       */
30      public getKey(): string {
31          return this.key;
32      }
33
34      /**
35       * Method for returning the key of this scene
36       */
37      public getLevel(): number {
38          return this.level;
39      }
40
41      /**
42       * Method for generating a new seed so that pseudo randomness is
             guaranteed
43       */
```

```
44          private generateNewSeed(): void {
45              const rndStr: string = Phaser.Math.RND.realInRange(Math.pow
                    (10, 2), Math.pow(10,10)).toString();
46              Phaser.Math.RND.sow([rndStr]);
47          }
48
49          /**
50           * Method for initializing the shape, position and properties of
                 the graphical scene transition
51           */
52          private transitionInit(): void {
53              // Shape of the graphical transition
54              const circle: Phaser.GameObjects.Graphics = this.add.
                    graphics();
55
56              // Shape of the screen
57              const rectangle: Phaser.GameObjects.Rectangle = this.add.
                    rectangle(0, 0, this.cameras.main.width, this.cameras.
                    main.height, 0x000000);
58
59              // Define circle as the mask
60              const mask: Phaser.Display.Masks.GeometryMask = circle.
                    createGeometryMask();
61
62              circle.setPosition(this.cameras.main.width / 2, this.cameras
                    .main.height / 2);
63              circle.fillCircle(0, 0, 0.1);
64              circle.setDepth(0);
65
66              mask.setInvertAlpha(true);
67
68              rectangle.setDepth(1);
69              rectangle.setOrigin(0, 0);
70              rectangle.setMask(mask);
71
72              circle.fillCircle(0, 0, 0.1);
73
74              this.transition = [circle, rectangle];
75          }
76
77          /**
78           * Opening transition. Normally used to visually introduce a new
                 scene
79           */
80          protected transitionIn(): void {
81              // Generating a new seed, so that randomness is guaranteed
                    in every repetition of a scene
82              this.generateNewSeed();
83
84              this.transitionInit();
85
86              this.children.bringToTop(this.transition[1]);
87
88              const tween: Phaser.Tweens.Tween = this.add.tween({
89                  targets: this.transition[0],
90                  scale: 10 * 0.5 * Math.sqrt(Math.pow(this.cameras.main.
```

```
                    width, 2) + Math.pow(this.cameras.main.height, 2)),
91                  ease: 'linear',
92                  duration: 700
93              });
94
95              tween.on('start', () => this.sound.volume = 0);
96              tween.on('complete', () => this.introduction());
97              tween.on('update', () => this.sound.volume += 1/tween.
                    duration);
98          }
99
100         /**
101          * Closing transition. Normally used to visually close or stop a
                scene.
102          * @param scene The scene you want to start next.
103          * @param data Additional data you want to give to the next
                scene.
104          */
105         protected transitionOut(scene: string, data?: any): void {
106             this.children.bringToTop(this.transition[1]);
107
108             const tween: Phaser.Tweens.Tween = this.add.tween({
109                 targets: this.transition[0],
110                 scale: 0,
111                 ease: 'linear',
112                 duration: 700
113             });
114
115             tween.on('complete', () => this.sceneChange(scene, data));
116             tween.on('update', () => this.sound.volume -= 1/tween.
                    duration);
117         }
118
119         /**
120          * Helper method for starting a new scene and stopping the
                current one
121          * as the behaviour of the current scene when starting a new one
                is
122          * not clearly defined in the framework at this point of time.
123          * @param scene The scene you want to start next
124          * @param data Additional data you want to give to the next
                scene.
125          */
126         protected sceneChange(scene: string, data?: any): void {
127             this.sound.stopAll();
128             this.game.scene.start(scene, data);
129             this.game.scene.stop(this.key);
130         }
131
132         /**
133          * Helper method for playing the introduction and pause the
                current scene
134          */
135         protected introduction(): void {
136             this.scene.pause();
137             this.game.scene.start("IntroScene", {'pausedScene': this.
```

```
                getKey(), 'level': this.getLevel()});
138         }
139
140         /**
141          * Returns the correct scaling factor for the wanted image size
                in relation to the real image size.
142          * @param wantedImageSize Image size you want to have for a
                dimension
143          * @param realImageSizeWidth The image width you want to scale
144          * @param realImageSizeHeight The image height you want to scale
145          * @param scaleToHeight Boolean for scaling height or width of
                image to the wanted size. Default to false.
146          */
147         protected imageScalingFactor(wantedImageSize: number,
               realImageSizeWidth: number, realImageSizeHeight: number,
               scaleToHeight: boolean = false): number {
148             let ret: number;
149             if (scaleToHeight) {
150                 ret = Math.max(wantedImageSize / realImageSizeWidth,
                       wantedImageSize / realImageSizeHeight);
151             } else {
152                 ret = Math.min(wantedImageSize / realImageSizeWidth,
                       wantedImageSize / realImageSizeHeight);
153             }
154             return ret;
155         }
156     }
```

4.5.2 PreloadAssets

4.5.3 DropDownScene

4.5.4 WelcomeScene

4.5.5 LevelMenuSceneScene

4.5.6 IntroScene

4.5.7 SortingScene

4.5.8 PropertySortingScene

4.5.9 RestrictedSortingScene

4.5.10 GameScene

4.5.11 ScoreScene

4.6 Final look of the learning environment

4.7 Checklist

- overview of your solution. Give a good explanation of its rationale, concepts and mechanisms. If your solution relies on a theorem or some other undocumented concept, make sure that you explain them before you carry on to the detailed description.

The main part of this section is the thorough descrip- tion of the solution and its functionality. The description should not contain arguments on correctness or design de- cision debates; simply, describe the mechanisms of your solution and avoid explanations of the 'why so' type. Dedicate a separate paragraph or two on the latter, if you deem necessary.

Disassemble your solution to its functional components and explain them separately. For example, if you describe a distributed algorithm, explain the protocol-specific part (message format, etc.) separately from the semantics and decision-making part of the algorithm. It is both important and useful to provide figures demonstrating the functionality of your solution. Make the figures look similar to the system model figure, if ap- plicable, and exploit the similarities and differences to point out important aspects of your solution.

Analysis can be of two types: qualitative and quantitative. The former means to show some properties (qualities) of your solution, while the latter means to show some performance aspects of your solution. Qualitative analysis is usually proof of correctness, however it could be

proof that the solution possesses some desired property. For algorithms or protocols, a proof of correctness is always welcome. Quantitative analysis is mostly performance analysis. It is important to explain what performance metric you use and why you have selected the specific metric. Choos- ing a metric that has been widely used will make the comparison to other solutions easier.

5

Evaluation

In this section the environment the game was tested in will be presented and analysed. First every game has to be tested on different devices for code errors or functional disbehaviour which was not intended. Second and third a target and various audience will test the game and significant reaction and behaviour will be stated.

5.1 Debugging

5.1.1 Devices

firefox, chrome, microsoft edge computer (apple, windows 10), tablet (apple, android), smartphone (apple, android)

5.1.2 Differences and obstacles

5.2 Target Audience Test

5 Test subjects age between 4 and 7 with less to none knowledge of use of electronic devices. 6 Test subjects age between 4 and 7 with strong knowledge of use of electronic devices.

5.3 Various Audience Test

11 Test subjects age between 18 and 40 equally distributed with daily usage of electronics (computer, tablet, smartphone) 13 Test subjects age between 30 and 70 equally distributed with only daily usage of their smartphone

5.4 Checklist

Depending on your budget and available time, you may have performed simulations or even some experiments. In either case, it is important to describe the environment of your experiments or simulations. This includes stating the parameters and conditions of the environment (sim- ulated or real), what measurements were taken and how they were taken. You need to establish the fact that your simulation or experiment results are statistically stable, meaning that they are representative of the space of possible results. Performing experiments and simulations is a subtle mat- ter, always putting the validity of your data at risk in many aspects. Before you perform the simulation or ex- periment, educate yourself on how to perform simula- tions, how to interpret the results and how to present them in graphs and figures. Each figure (or graph) should be well explained. Dedi- cate at least one paragraph for each figure. Describe what the reader sees in each figure and what he should notice. Moreover, reason on the results - are they the way they were expected to be? Avoid giving tables of numerical data as means of presenting your results. Compare the performance of your solution to the per- formance of one or two competing solutions. Usually, when you simulate or experiment on your solution, you simulate it in contrast to a competing solution. You have to make sure that the test scenarios are fair and make an argument about the fairness of your comparison in the paper. A special case of experiment is the usability test. Many times, although the performance of a solution can be im- pressive, the applicability of it can be minimal. Usability test is a type of experimentation, which determines the acceptance of a solution by end-users or its suitability for certain applications. Usually, papers on software product solutions contain usability tests.

Conclusion

limitation - design - knowledge of the framework and examples for complete funcitonality "der gedanke dahinter" -

disadvantages - lehrer mues iharbeite will beschrenkt durch keine zahlen und woerter - durch eine webaplikation schwer zu beschraenken auf gewissen geraeten (apple)

6.1 Future Improvements

6.2 checklist

The conclusions section, similar to the introduction and related work sections, serves two purposes. The first is to elaborate on the impacts of using your approach. The second is to state limitations or disadvantages of your so- lution, thus enabling you to provide directions for future research in the field.

Bibliography

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.