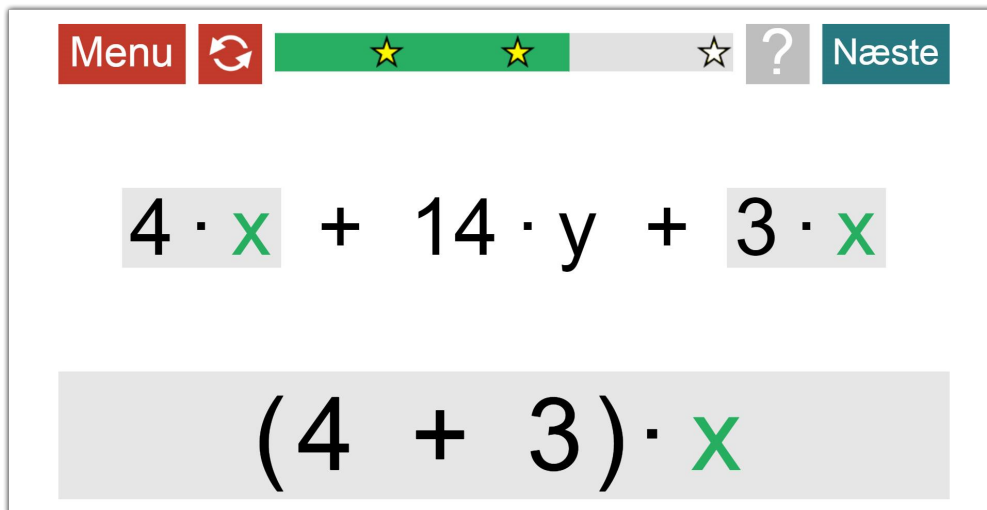


# Gamifying Math in Danish Primary Schools



The screenshot shows a user interface for a math game. At the top, there is a navigation bar with a red 'Menu' button, a red refresh icon, a green progress bar with two yellow stars, a grey question mark icon, and a blue 'Næste' (Next) button. Below the navigation bar, the expression  $4 \cdot x + 14 \cdot y + 3 \cdot x$  is displayed. The variables  $x$  and  $y$  are highlighted in green. Below this, a grey box contains the simplified expression  $(4 + 3) \cdot x$ , where the  $x$  is highlighted in green.

## Group A317b

Morten Rask Andersen

Anton Christensen

Lasse Fribo Gadegaard

Christian Mønsted Grünberg

Mathias Ibsen

Nikolaj Mariager

Mathias Rohde Pihl

24/05-2016

Aalborg University  
Software, 2. semester



Morten R.A.

---

Morten Rask Andersen

Anton Christensen

---

Anton Christensen

Lasse F. Gadegaard

---

Lasse Fribo Gadegaard

Christian M. Grünberg

---

Christian Mønsted Grünberg

Mathias Ibsen

---

Mathias Ibsen



---

Nikolaj Mariager

Mathias Rohde Pihl

---

Mathias Rohde Pihl



**Computer Science**  
Aalborg University  
<http://www.aau.dk>

# AALBORG UNIVERSITY

## STUDENT REPORT

**Title:**

Gamifying Math in Danish Primary Schools

**Theme:**

Gamification in Education

**Project Period:**

Spring Semester 2016

**Project Group:**

A317b

**Participants:**

Morten Rask Andersen  
Anton Christensen  
Lasse Fribo Gadegaard  
Christian Mønsted Grünberg  
Mathias Ibsen  
Nikolaj Mariager  
Mathias Rohde Pihl

**Supervisor:**

Giorgio Bacci

**Copies:** Online on Digital Exam

**Number of Pages:** 69

**Date of Completion:**

May 23, 2016

**Abstract:**

Boredom in primary school is a problem for Danish students in senior grades. In a survey done by the Danish Ministry of Education, about a third of the questioned 7th to 9th graders said that they frequently found classes boring. This project presents a solution for the problem statement: *"How can gamification be used to make the learning of reducing algebraic expressions by hand more engaging for students from 7th to 9th grade?"*. This is done by developing a program as an alternative to reducing algebraic expressions by hand. The program implements motivational affordances: *levels, point system, progress, achievement, feedback and leaderboard* to increase the students' motivation. The program has been tested at Vesterkærets skole where a group of 7th graders tried the program and filled out a questionnaire about it. Due to the low sample size of 12 students it was not possible to make a general conclusion, but from the questionnaire there were tendencies amongst the students, showing positive effects of the motivational affordances implemented and 11 out of 12 students agreed that the program was more fun to use than doing tasks by hand.

*The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the authors.*

---

## Preface

This report is written by group A317b from the department of Computer Science at Aalborg University, consisting of seven 2nd semester students. The theme of the project is gamification in education and the project has been supervised by Giorgio Bacci.

In the electronic appendix There are five solution that can be opened in Visual Studio. There are solutions for Bridge/Web, Android and Desktop. The last two solutions are for testing. In the folder 'App' an .apk file is located which can be used to install our solution on Android devices.

To test the web version open the solution "ThreeOneSevenBee.Development.Bridge.sln" in Visual Studio and set ThreeOneSevenBee.Frontend.Website as start-up project by left-clicking in the solution explorer. Finally rebuild the project and run it.

We would like to thank Vesterkærets skole of Aalborg municipality for allowing us to test our program on some of their students. We would also like to thank our supervisor Giorgio Bacci for his guidance throughout the project.

## Reading Guide

This report is written in a chronological order and should be read as such. Bibliographical sources are referred using numbers. The bibliographical sources can be found at the end of the report. Sections and figures are referred to by number which is set depending on in which order they occur.

It is expected that the reader possesses a basic knowledge of mathematics and object-oriented programming in C#.

The illustrations used in this report are made by group A317b unless otherwise specified.

Code snippets *can* be shorter compared to the source code. If a part of the code has been removed the missing part will be indicated with an ellipsis[...].

The report has been delivered online on Digital Exam where the program's source code has been attached. The code can also be found in our GitHub repository [1].

# Contents

<b>1 Introduction</b>	<b>9</b>
1.1 Problem Statement . . . . .	9
<b>2 Methods</b>	<b>10</b>
2.1 Gamification Definition . . . . .	10
2.2 Motivational Affordances . . . . .	10
2.3 Gamification as a Method . . . . .	12
2.4 Test-Driven Development . . . . .	12
2.5 Collection and Evaluation of Data . . . . .	13
<b>3 Problem Analysis</b>	<b>14</b>
3.1 Cause and Effect of Boredom . . . . .	14
3.2 Current Teaching of Math . . . . .	15
3.2.1 Different Ways of Teaching Math . . . . .	15
3.2.2 Digital Tools in Math Lectures . . . . .	15
3.2.3 Curriculum . . . . .	16
3.3 Current Educational Programs . . . . .	16
3.3.1 Discussion of Gamification in Educational Programs . . . . .	19
<b>4 Final Problem Statement</b>	<b>22</b>
<b>5 Idea</b>	<b>23</b>
5.1 Comparison of the Different Approaches . . . . .	23
5.2 Gamifying Math . . . . .	24
<b>6 Program Description</b>	<b>25</b>
6.1 Login Menu . . . . .	25
6.2 Main Menu . . . . .	26
6.3 Level Select Menu . . . . .	27
6.4 Level Menu . . . . .	27
<b>7 Development</b>	<b>28</b>

7.1	Choice of Environment . . . . .	28
7.2	Program Documentation . . . . .	30
7.2.1	Overall Code Description . . . . .	30
7.2.2	From String to Expression . . . . .	33
7.2.3	Size . . . . .	37
7.2.4	View . . . . .	42
7.2.5	Database and API . . . . .	45
<b>8</b>	<b>School Visit</b>	<b>49</b>
8.1	Purpose . . . . .	49
8.2	Method . . . . .	49
8.3	Data . . . . .	49
8.4	Observations . . . . .	51
8.5	Analysis . . . . .	52
<b>9</b>	<b>Testing</b>	<b>54</b>
<b>10</b>	<b>Discussion</b>	<b>55</b>
10.1	Gamification Without Reducing Time Spent on Learning . . . . .	55
10.2	Gamifying the Learning of Formulas and Algebraic Rules . . . . .	56
10.3	Adjusting the Challenges to the Individual Student's Skill Level . . . . .	56
<b>11</b>	<b>Conclusion</b>	<b>57</b>
<b>12</b>	<b>Future Development</b>	<b>58</b>
<b>13</b>	<b>Bibliography</b>	<b>60</b>
<b>14</b>	<b>Appendix</b>	<b>63</b>
A	Questionnaire . . . . .	63
B	Infix to Postfix Algorithm. . . . .	64
C	Calculations of Amount of Bored Students . . . . .	65
D	Number of Teaching Lessons . . . . .	66
E	Table of Raw Time Data Extracted from School Test . . . . .	67

F Questionnaire Data with ID Tags . . . . . 69



## List of Figures

1	Mihaly Csikszentmihalyi's flow diagram from his book Flow: The Psychology of Optimal Experience [14]. . . . .	14
2	Screenshot that shows points, levels and medals at MatematikFessor [24]. . . . .	17
3	Shows feedback for the question 'what is the product of 4, 7 and 5 ?' at MatematikFessor [24]. . . . .	18
4	Shows the idea behind the program where progress shows how much the expression is reduced. . . . .	24
5	An example of the login screen. . . . .	26
6	An example of a title menu. . . . .	26
7	An illustration of the level select menu. . . . .	27
8	An example of the level menu. . . . .	28
9	Describes the separation of platform dependant and independent code, as well as separation within the independent codebase into model and view. JQueryGameAPI and CanvasContext inherit from IGameAPI and IContext classes respectively. . . . .	33
10	The class hierarchy for expressions (part 1/2). - Generated by Visual Studio. . . . .	34
11	The class hierarchy for expressions (part 2/2). - Generated by Visual Studio. . . . .	35
12	The expression tree for $2 - a$ . . . . .	36
13	Shows the tree structure of $a * b * c$ . . . . .	39
14	Shows common parent for selections in expressions $a * 2 + a$ , $\sqrt{2 + 2}$ and $a$ . . . . .	41
15	UML-diagram of implemented views. - Generated by Visual Studio. . . . .	43
16	Database diagrams with table relations. - - Generated by MySQL Workbench. . . . .	48
17	Chart showing the answers from eight of the questions in the questionnaire. . . . .	50
18	Chart showing the amount of minutes used on each level. . . . .	51
19	All unit tests being run in Visual Studio. . . . .	54
20	Hours of education in danish primary school [38]. . . . .	66
21	Shows the questionnaire data with corresponding ID tags. . . . .	69

# 1 Introduction

In 2015 the Danish Board of Education asked approximately 470.000 students in primary schools about their well-being in class [2]. One of the questions in the survey was “Are classes boring?”. In appendix C it is shown that in the third bracket (7th to 9th grade) 39.903 out of 123.120 students answered ‘often’ or ‘very often’ which is approximately a third. It is interesting to find a way to engage these students.

Gamification is a method to engage people in doing activities that they would normally find boring or tedious [3]. Through the application of game-design principles, various activities are made more interesting [3]. Applying this to education could be a way to engage students and thereby reduce the amount of bored students. To study the effect of gamification in education, we focus on math. The Board of Education in Denmark has specified that out of about 930-960 school hours, the students in 7th to 9th grade are required to have, a minimum of 150 hours must be dedicated to math as seen on the table in appendix D. Because of this, we know math is a large part of the curriculum in the Danish primary school. Looking at the curriculum for math for the third bracket there are five topic areas under numbers and algebra. One of them is formulas and algebraic expressions [4]. This topic will be the starting point for this project, and we are now interested in seeing if gamification can be used to engage students in learning formulas and algebraic expressions. This leads us to the following problem statement.

## 1.1 Problem Statement

How can gamification be used to make the learning of formulas and algebraic expressions more engaging for students in 7th to 9th grade?

## 2 Methods

To analyze and solve the problem stated in section 1.1, we use different methods. This section describes the methods used in this project. This includes a description of gamification, how we collect and evaluate data and finally, how we are going to develop a solution.

### 2.1 Gamification Definition

Before giving a definition of gamification, it may be helpful to understand the definition of a game. A game can be perceived as being many things. In [5], Katie Salen and Eric Zimmerman define a game as being "[...] a system in which players engage in artificial conflict, defined by rules, that results in a quantifiable outcome." The artificial conflicts can be solving a puzzle, saving a princess, defeating the enemy and so on. According to [5], conflict can be found in all games and occurs when the user interacts with the game in order to pursue a goal. The user then has to overcome different obstacles in order to achieve this goal. Rules are another important aspect of games. The game developers should define specific rules to be followed in order to win the game. Rules within a game define the structure of the game. An example of a rule for solving a puzzle is that all pieces must fit perfectly together in order for the puzzle to be finished.

So what is gamification? Karl M. Kapp gives the following description of gamification: "*Gamification is using game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems*" [3]. Making something game-based consists of creating a system in which the players engage in an abstract challenge. The goal is to make a game where people want to invest their time and energy [3]. Game thinking is the key principle in gamification. It is the idea of taking common things like exercise and restructure them by means of game elements as competition, co-operation, exploration and storytelling [3].

In this way gamification can be used to promote learning, solve problems and engage people [3]. Gamification brings some unique qualities to help solve problems and engage students. With gamification, students can use the co-operative aspects of games to solve problems together. Motivational affordances such as points, leaderboards and achievements can be used to promote learning by making the students strive for a goal, helping them stay engaged.

### 2.2 Motivational Affordances

The term affordance refers to the properties between an object and an actor [6], and motivational affordances cover things that can support our motivational needs. When games involve our motivational needs, we feel engaged and when games satisfy our motivational needs, we feel enjoyment and as a result we want to keep playing [6].

Early examples of gamification are showed all the way back to when the Egyptians built pyramids. Here motivational affordances such as teams and challenges were used to motivate workers [7]. As gamification evolved, other motivational affordances came into play. These are e.g. points, leaderboards, achievements, levels, clear goals, feedback, rewards, progress and challenges [8].

The different motivational affordances in gamification are elements known from games that can be used

to engage people in doing activities as mentioned in section 2.1. The following is a brief description of each of them.

### **Point System**

Points are a numeric form of rewards that is used to rate a user's performance [9]. A player or user can earn points in a variety of ways depending on the genre of the game. In some educational games, one can earn points depending on how fast a given assignment is completed or according to the difficulty of the assignment .

### **Leaderboard**

The motivational affordance leaderboard promotes competitive behavior. In competitive games a player can compare his score with those of other players. Some games also have ranking systems, in the sense that when you win or lose points, you are placed accordingly on the leaderboard. Leaderboards can have a demotivating effect on players who are lagging behind if implemented incorrectly i.e. displaying the bottom of the leaderboard [9].

### **Achievements**

Achievements are a kind of reward given for completing a specific task. Achievements can either be given for completing essential tasks, but could also be given for completing less important parts of the game [9].

### **Levels**

Most often a level refers to how far you are in a game. Levels help make different stages of the game more manageable and provide the user with a better overview of the game, along with providing goals. Levels invoke a feeling of progress in the user [9].

### **Clear Goals**

Another aspect of games is to have clear goals where the players have a clear vision of what to accomplish, how to do it and why it has to be done. An example of this is a 'quest' where the player is tasked with a specific objective they have to achieve with some sort of purpose.

### **Feedback**

An example of motivational affordances that can be used in gamification is the ability to send direct and clear feedback to the users of the program. By doing so, users will be able to adjust their method for solving a problem, based on the feedback they receive [3].

## Rewards

In games you often receive a reward for accomplishing something. Examples of rewards can be virtual currency, titles, items and so on. The reason behind rewards is to 'give' something to the users for accomplishing something in the game. Rewards are similar to the point system, but a reward can be different things depending on the game. You can say that getting a point is a reward, but a reward does not necessarily have to be a point. Other examples of rewards could be achievements or badges, where badges are visual representations of achievements [9].

## Progress

Progress is a way of telling the players how far they are in a game. Progress is often showed with a progress-bar. This could e.g. be a bar that shows how far a player is from accomplishing something. The progress can be compared with other players and it gives a feeling of competition which is motivating for some players and can make them feel more engaged.

## Challenges

One of the motivational affordances that reoccurs in many games is challenges or obstacles to overcome. Challenges provide a level of difficulty to the games, and give the players something to achieve and overcome. A challenge can be a task that is difficult to do [9]. This could e.g. be a difficult level in a game.

## 2.3 Gamification as a Method

Our project is based on gamification and how it can be used to solve problems in education. With that in mind, we have been looking for problems of relevance to the society in education. We chose to look at problems regarding boredom in schools. To see how gamification can be used to solve the boredom problem, we will analyze the cause and effect of boredom. To get a better understanding of different motivational affordances we will look at different educational program used in education today. From this we can finally develop our own program using gamification to increase motivation.

## 2.4 Test-Driven Development

In order to ensure that our program works as intended we will use Test-Driven Development (TDD) which is a test centric approach to development. The following description of TDD is based on [10].

- 1. Quickly add a test** In this step a test is added. The purpose of this test is to be as small as possible and to fail right away. It is a test for a feature that has yet to be added, but a feature where the specifications and requirements are completely understood.

- 2. Run all tests and see the new one fail** Next we run the tests and confirm the test is failing. The fail is important so we know right away that the test is working properly and that the feature is in fact missing. If this test passes, the feature is either already implemented or the test has errors.

**3. Make a little change** In this step we start to add bits of code, to get the test to pass. At this point, all worries about code-smell [11] or bad practices are ignored. The goal here is simply to get the test to pass in as little code as possible.

**4. Run all tests and see them all succeed** Next we run the tests, and confirm that the new addition has properly implemented the required feature. All other tests are run as well, to ensure that nothing was broken during the addition of this new feature.

**5. Refactor to remove duplication** Finally, the code is refactored and cleaned up. The code-smell and other bad parts added in step three are removed and the code is moved to its appropriate place in the solution.

## 2.5 Collection and Evaluation of Data

After we have developed a prototype of the program we want to determine how the program answers the problem statement. To do this we will visit a school and let students in 7th to 9th grade try the program. To get data from the school visit, we will use the two methods: behavioral analysis and questionnaires, both described below.

### Behavioral Analysis

We will register the use of our solution to get data about the behavior of the users of our solution. This is done in a passive way where the users are not directly conscious about the data collection. One part of these measures is the time spent on solving different levels as well as measure how much progress the users have made.

### Questionnaires

This will be used to get a quantitative evaluation of our solution. The questionnaires are intended for students from 7th to 9th grade and will contain general questions about the solution and its usage. Here we will focus on getting simple quantitative answers, such as a scale ranging from strongly disagree to strongly agree.

### 3 Problem Analysis

As described in the previous section we are going to develop a program that implements gamification. Before we begin developing the program it is first important to get an understanding of the problem's context. This section is used to get a better understanding of the problem statement and the issues that are connected to it. This is done by analyzing the cause and effect of boredom in primary schools. To understand the content that the program should present, we look at the current teaching in the Danish primary schools and what the 7th to 9th graders are supposed to learn in math. After this we look at current educational programs and how these use different motivational affordances. From this we can discuss which motivational affordances are suited to solve our problem statement.

#### 3.1 Cause and Effect of Boredom

To understand why boredom in school is a relevant problem, this section will look at what boredom actually is, what might cause it, and which consequences it can have on the development of the individual student. According to lecturer in psychology Einar Baldvin [12] boredom is an emotional feeling, like anger or love, which occurs when an individual is under-stimulated. Professor of Psychology and Management Mihaly Csikszentmihalyi agrees to this definition, and adds that boredom can also occur when the challenges one face, are too easy for their skill-level [13]. If the challenges are too tough for one's skill-level, anxiety will replace boredom, see figure 1. It is therefore important to keep students stimulated and to adjust the challenges to the individual person's skill-level.

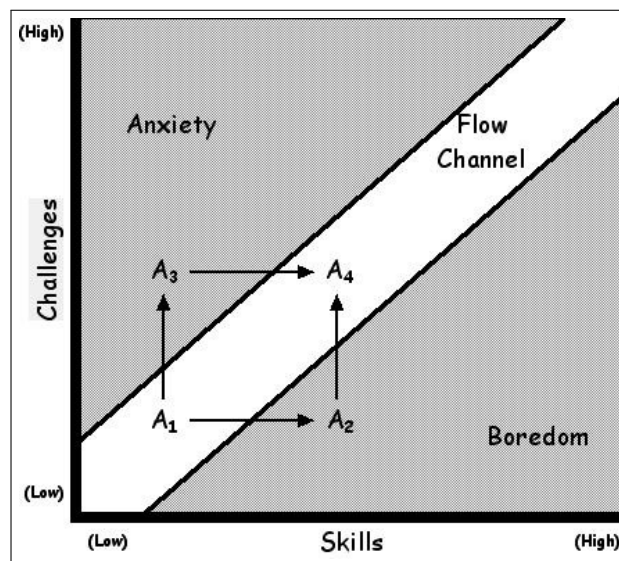


Figure 1: Mihaly Csikszentmihalyi's flow diagram from his book *Flow: The Psychology of Optimal Experience* [14].

Boredom amongst students in math class is an important matter to look at because there is usually a big gap between the students who are good at math, and the students who are less good at math [15]

and this makes it more difficult to stimulate every single student in the correct way. The math prone students do not need a lot of introduction to the subject that is reviewed, and they do not need a lot of repetition of the same exercises either.

According to Hans Jørgen et al. this will cause the smarter students to lose interest and will thereby also waste development opportunities. A survey done by "Dansk Industri og Danske Skoleelever" (Danish Industry and Danish Students), showed that 52,5% of the questioned 217 students have answered "yes, often" to the question "Are you bored in school, because you are not challenged enough?" [16].

Since a lot of students in the Danish primary schools are affected by boredom and since it has a great effect on the learning process of the students, it is important to try to solve this problem. According to Professor Csikzentmihalyi, this problem can be solved by making the students hit the state of flow. To help the students reach this flow, we will try and match the assignments with the student's skill level.

## 3.2 Current Teaching of Math

In the Danish primary school, different teaching methods are used to vary the way students are taught. According to EMU, a Danish educational portal [17], teachers tend to maintain the student's attention and interests better, if the teaching methods are varied [18]. Examples of different teaching methods in the Danish primary school includes: IT, social media and co-operative learning. This section will focus on IT and digital tools in the teaching of math in Danish primary schools. To do so, this section is divided into two: the curriculum of math from 7th to 9th grade and different forms of digital tools that can be used in education.

### 3.2.1 Different Ways of Teaching Math

In the Danish primary school mathematical problems are sometimes solved with the use of calculators or with computer programs such as Wordmat and GeoGebra [19]. When manipulating expressions by hand, the student spends a lot of time writing the expressions on paper, the same goes for typing a lot of intermediate results on the computer. Since the current way the Danish students are doing math can be time-consuming, we want to find a solution to this, where the student still learns the same way of manipulating mathematical expressions and learns different arithmetic rules, but in a less time-consuming manner.

### 3.2.2 Digital Tools in Math Lectures

With the increased focus on using digital teaching methods in primary schools, the Danish Ministry of Education allocated 500 million DKK to be used to increase the usage of IT in primary schools from in the years 2011 to 2015 (Extended to 2017) [20]. The digital tools used in the Danish primary schools are divided into three categories: Common tools, CAS-tools and educational programs [21].

The common tools are non-specialised programs that can be used in multiple classes. Examples of common tools are Microsoft Word and Microsoft PowerPoint.

CAS-tools (Computer Algebra Systems) are programs meant to help the user solve algebraic problems. CAS-tools are often used in math education and can be used to reduce expressions, solve equations and



draw graphs amongst other things.

Educational tools for math are programs that are build to be used as a tool for math education and may focus on a specific topic in math [21].

In this project we will try to create a digital tool that implements some functionality from CAS-tools in an educational program, so that the program has functionality for applying algebraic rules to an expression but it is the student that controls which rules to apply.

### 3.2.3 Curriculum

The purpose of this section is to clarify what the students from 7th to 9th grade are supposed to learn in the math topic *formulas and algebraic expressions*. To do this we look at the overall description of the topic made by the Danish Ministry of Education.

The topic *formulas and algebraic expressions* are divided into three categories [4]:

1. Describe relationships between simple algebraic expressions and their geometric representation.
2. Substitute variables and evaluate algebraic expressions.
3. Manipulate algebraic expressions showing knowledge about rules of algebra.

A key part of *formulas and algebraic expressions* is to gain knowledge about rules of algebra. Table 1 shows selected rules of algebra [22]:

Fractions	Exponents	Parenthesis
$\frac{a}{c} + \frac{b}{c} = \frac{a+b}{c}$	$a^{-n} = \frac{1}{a^n}$	$a \cdot b + a \cdot c = a \cdot (b + c)$
$\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$	$a^0 = 1$	$a^n \cdot b^n = (a \cdot b)^n$
$a \cdot \frac{b}{c} = \frac{a \cdot b}{c}$	$a^n \cdot a^p = a^{n+p}$	$(a^n)^m = a^{n \cdot m}$

Table 1: Shows selected rules of basic algebra.

## 3.3 Current Educational Programs

Before we develop a program that implements the rules shown in the previous section in combination with different motivational affordances, we will first review the use of motivational affordances in two existing educational programs used for math in Danish primary schools, as well as one of the most well-known free learning sites in the world, Khan Academy. Furthermore, we will discuss the pros and cons of the implementation of the different motivational affordances as well as pros and cons of implementing gamification in educational games. This will help us form ideas on how to implement motivational affordances into our application.

### Emat.dk

Emat.dk is an online digital learning program, that can be used on several electronic devices, such as PCs and iPads. It has about two million questions and more than 10 levels of difficulty. The program

provides the user with help for each topic and makes communication between teacher and student easy, by helping the teacher send out tasks to the students and helping the students send answers back to the teacher for feedback. The following motivational affordances is used at Emat.dk.

**Leaderboards** are used when the users competes against themselves or other students in e.g. multiplication and addition.

**Levels** are used in forms of various assignments with different levels of difficulty. There are 10 different levels assignments can be made on the basis of.

**Feedback** is used when the users can see their results in detailed statistics. This way the user can see where they performed better and where improvement is needed.

### Matematikfessor.dk

MatematikFessor is a digital educational system for math in primary schools. The system contains more than 2000 video lectures combined with tasks that are automatically graded [23]. The tasks can be completed in various manners, such as *SuperTræneren* (The Super Trainer) that trains the user in numerous subjects or *5 Skarpe*, where the user has to answer five questions in a limited amount of time.

**Point Systems** are used to award players after the tasks are completed, on the basis of the amount of tasks solved correctly and the time, it took to solve them. The points are shown in the top of the web-page and allow the access of certain rewards. Level and medals are calculated on the basis of the amount of points the user has earned.



Figure 2: Screenshot that shows points, levels and medals at MatematikFessor [24].

**Leaderboards** are used on the front page. Here statistics are used for showing 'boys versus girls' and their respective correct answers. It is also possible to change the view from all schools, to a specific school and even a single class in a specific school.

**Achievements** are used by listing certain titles based on the amount of correct answers. Examples of these are *TalentFessor* (0 correct answers), *JuniorFessor* (500 correct answers) and *MatematikFessor* (MathFessor, 2000 correct answers).

**Levels** are automatically implemented and adjusted in *SuperTræneren*. This tool makes up problems and assignments that fits the individual user's skills, abilities and progress [25].

**Feedback** is used in several ways. In *SuperTræneren* the user is instantly told whether the answer is true or not and the correct answer is provided, but not the way to do it. In *5 Skarpe* the user does not receive instant feedback, but after the test the questions are listed and a way to solve them is presented.



Figure 3: Shows feedback for the question 'what is the product of 4, 7 and 5 ?' at MatematikFessor [24].

**Rewards** are used by making the user able to change the theme of the Fessor (the mascot of MatematikFessor) based on the amount of correct answers. The different themes of the site's mascot is a reward.

**Progress** is shown when *SuperTræneren* is used. In the top left corner, it is stated how many questions need to be solved till the next star is awarded. The amount of awarded stars is shown in the main page of *SuperTræneren*.

**Challenges** are used in the main page of *SuperTræneren*. Here you can see the challenges within certain subjects. There are time-challenges and 'correct answers in a row' challenges.

### KhanAcademy.org

KhanAcademy.org is a world-wide known learning site, it offers free learning material in several different subjects such as math, programming, art and many more. The site is available in many different languages. Since the site is not built after the Danish school structure, it is not possible to sort it after the Danish curriculum for specific classes. In addition to all of the videos on the different subjects, there is a button after most of the videos, to 'Practice this concept' where the user is tested in the taught subject.

**Point Systems** are used to award players, what Khan Academy calls 'energy points'. According to Khan Academy, energy points *"[...] measure effort on Khan Academy. Learners earn more energy points for pushing the edge of their knowledge. They are not a measure of mastery or ability."* [26].

**Achievements** are used in form of badges. There are different types of badges such as the most common; meteorite-badges or the unknown and rare badges, black-hole badges.

**Feedback** is used in a very simple manner. After writing an answer and pressing the 'check answer' button, the user will be presented with a text either saying 'Not correct yet, please try again' or 'Correct! Next question'.

**Progress** is used automatically by Khan Academy. Depending on which subject the user is studying, Khan Academy recommends to resume the learning of that subject by having a list of recently studied subjects. The users can see their progress on the subjects they are currently studying.

### 3.3.1 Discussion of Gamification in Educational Programs

We have now looked at how three different educational programs implement motivational affordances, we will now discuss the pros and cons of the motivational affordances we find most relevant for our solution.

**Point System** Point systems are used in different ways as a motivational affordance. Matematik-Fessor uses them as a factor of how many tasks are completed correctly compared to the time it took to solve them, while Khan Academy use them as a measure of how much the users are pushing to the edge of their knowledge.

The user wants to obtain as many points as possible, especially because it, in most cases, gives some reward or places them higher on the leaderboard. Making points depend on the amount of solved tasks alone, makes it motivating for the user to solve as many tasks as possible, but that is not always what the optimal learning experience is for the user. If the user already knows how to add, but not how to subtract, there is a chance the user will do more addition problems to earn easier points instead of trying to learn how to subtract, since it will award points slower.

'Energy points' on the Khan Academy website are earned through various activities, such as watching educational videos, doing challenges, or earning badges. This motivates the user to watch more videos about different topics and thereby increase their knowledge, but in the case of the user only wanting to earn the most points possible, it will not necessarily be the optimal learning experience.

To refrain from making the user only solve the tasks that give the quickest and easiest access to the most points possible, it would be optimal to make it progressively more difficult to earn stars within a given category. Another approach is to have a limited amount of points for each category.

**Leaderboards** There are different ways of making leaderboards; between schools, against other students, against the user itself, or to make up 'fake' leaderboards with pre-set times.

Leaderboards between schools can be very motivating if you want to be the best school in the competition. Some students may not care about this, since their individual performance is not celebrated.

Leaderboards between other students can be more motivating for some, but can also have a negative effect on the student's motivation. Students want to beat their classmates, but watching yourself in the bottom of the leaderboard can be very demotivating. A work-out around this can be to do like Kahoot [27], where only some of the best students within a class is shown. This would decrease the demotivating part considerably and make leaderboards a motivating affordance for all students in the class.

Leaderboards against the user itself can be motivating by making the user want to beat their own records, but can also be boring to only compete with yourself.

Leaderboards with made up times can be motivating in regards to the user wanting to beat certain times (e.g. if certain times award gold, silver or bronze medals). It does not have the human competition factor, which may not motivate the user as much.

A good way to implement leaderboards in our solution could be for students within a class to compete against each other only showing the top students on the leaderboard. The leaderboard could be scored after amount of achievements earned, this way students who complete the game first will be shown at the top of the leaderboard.

**Achievements** Achievements can be motivating for the user, to make them get the feeling of being rewarded for their actions. Achievements could be awarded for a wide range of actions, but a few examples could be if the user completed all the levels in 'Fractions' with the maximum amount of points. This could, just like point systems, make the user motivated to only do the tasks that award the easiest way of getting achievements. To refrain the user from doing this, achievements could be hidden, progressively be harder to obtain or award a specific amount of 'achievement points' based on where the user needs practice instead of where the user is already well-practiced.

**Levels** Levels are pretty straightforward; they help the user get an overview of how far they are in a given topic. A way to implement levels could be to manually create all levels. They could also be implemented automatically as MatematikFessor does it. Here the difficulty is automatically adjusted to the user's skill level.

**Feedback** is important for showing the users if their answer is correct or not, and the user will thereby know if they have learned how to use the applied theory, but there are several ways to give feedback. Instant feedback, i.e., getting the feedback right after solving the problem, can be in the middle of a test or before the next problem to be solved. This is helpful; if the user is doing a certain problem wrong, they can correct it, and know the correct way to solve the rest of the similar problems. Another way to give feedback is to give it after the test or the list of problems. This can help the user get a better overview of the problems that were done correctly or incorrectly. If feedback is presented after a test full of addition problems and the student forgot how to add, the result would just be a lot of incorrect answers. There could also be no feedback, but feedback is necessary, since the user should know if their answers are correct or not. Not knowing if the answers are correct or not makes the user incapable of doing them right in the future.

The best way to give feedback would be to tell the user if the answer is correct or not after each answer with an option to see why it is correct or incorrect. This would help the user not repeatedly make the same mistakes.

**Progress** Progress is used to show the user how far they are within a given topic, test or problem. Progress can be used for various things, such as a progress bar, or progress within a given problem, to show the user how far they are to finishing it completely. This is relevant in cases of simplifying expressions, where it is difficult to know when an expression is simplified completely.

**Discussion of Gamification in General** By implementing gamification in a certain subject, it is possible to make the subject more interesting for the users. This requires knowledge of the different motivational affordances in gamification, and how to use them correctly. Implementing gamification requires a certain amount of resources to do correctly, it takes time to make a game that is interesting enough to catch the users' attention. In our context, we would like to optimize the users' learning processes by making all the users hit the state of 'flow', as described in section 3.1. To make the students hit this state, we implement game elements such as levels, to match the individual student's skill-set, with the challenges they are going to face. Doing this would help the teacher save a lot of time that was previously used on helping all the students with different skill levels. When implementing gamification in education, it is essential to prioritize the focus on teaching the subject, rather than making the game as 'fun' as possible. Since the school has a specified time allotted for each class, where a certain curriculum has to be covered it is essential to make the game achieve these requirements rather than only implement game-elements that decrease time spent on learning.

## 4 Final Problem Statement

From the previous section *Problem Analysis* we now see that boredom in math can be the result of challenges that do not match the student's individual skill. As mentioned in section 3.3.1, it is important that the implementation of gamification does not remove focus from learning the subject. As explained in section 3.2.1, our goal is to find a new way for students to reduce algebraic expressions rather than doing it with pencil and paper. This leads us to the final problem statement:

*How can gamification be used to make the learning of reducing algebraic expressions by hand more engaging for students from 7th to 9th grade?*

This project will focus on answering the problem statement by developing a solution that answers the following questions.

- How can gamification be implemented in educational programs without reducing the actual time spent on learning?
- How can the learning of formulas and algebraic rules be gamified?
- How do we ensure that challenges meet the individual student's skill level?

## 5 Idea

The purpose of this section is to describe our idea for the solution of the problem statement. We will first discuss different approaches that we have come up with during the group discussions, then choose and sketch the idea which we want to develop.

### 5.1 Comparison of the Different Approaches

To find an approach for developing an educational program using different game elements, this section describes two different approaches to combine math and game elements.

**Hiding Math in a Game** The idea behind this approach is to change the representation of expressions. This could be variables represented as objects, such as diamonds, animals, gold, rubies and so on. In this way mathematical expressions are presented as things the students are familiar with, instead of using standard symbols such as  $x$ ,  $y$ , and  $z$ .

An example of hiding math in a game is DragonBox Algebra [28], where variables are represented as bugs or animals.

This approach has the advantage of enhancing the user's motivation without moving the focus from math. Changing the representation of expressions can have both motivational and learning advantages. The disadvantage of this approach is that it is required that the program makes the connection between the game representation and the traditional mathematical representation of expressions. Since we focus on 7th to 9th grade students that already know basic algebraic manipulation, it can be tedious for them to spend time on learning the new representation instead of using the traditional mathematical representation.

**Presenting Math as a Game** In this approach the traditional notation of expressions is used. The idea is to make mathematical expressions more visual and manipulative. Thereby, the goal is to make math more playable.

The advantage of this approach is that students learn to use the traditional representation and become more familiar with this representation. Therefore, they do not need to spend time on learning another game representation like it was the case in the first approach. The disadvantage of this approach is that it may be difficult to find a way to visualise and manipulate mathematical expressions in its original representation while keeping the motivation high.

**Comparing The Approaches** The first and second approach both focus on math. The first approach may be more motivating but also has to make the translation between the traditional representation and the game's representation of expressions. The first approach may be more well-suited for a younger audience because they have not yet learned the representation of expressions.

Since the first approach requires time spent on learning a new representation, we have chosen to focus on the second approach. This approach will teach the students how to manipulate and reduce algebraic expressions in a more playful manner while still using the traditional mathematical notation. The goal is



now to make an educational program that implements motivational affordances and uses the traditional representation of expression. Therefore, our goal is to make algebra more visual, manipulative and motivating.

## 5.2 Gamifying Math

The purpose of this section is to explain how we want to use different motivational affordances in order to gamify the way students are taught math. In this section we will introduce the motivational affordances that are going to be used, and how they will be implemented in accordance with the idea mentioned in the previous section. The motivational affordances, we want to implement are: *levels, point systems, progress, achievements, feedback and leaderboards*. The first sketch describing the idea behind our program is shown in figure 4.

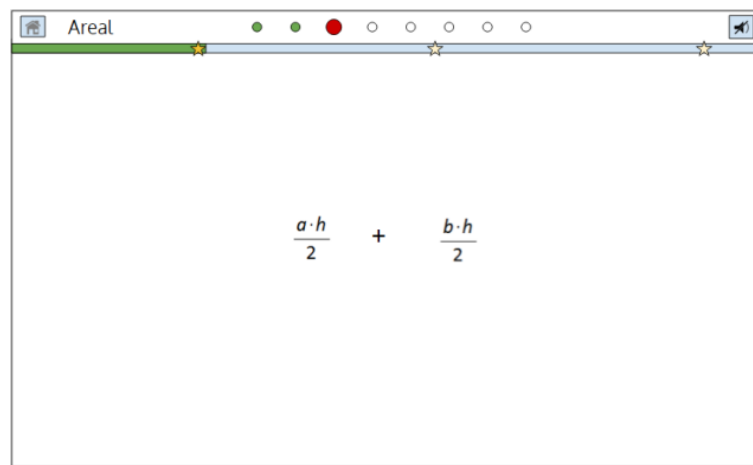


Figure 4: Shows the idea behind the program where progress shows how much the expression is reduced.

### Levels

In our solution we will have different categories e.g. fractions and exponents. Each category will contain different levels, and the goal is that the higher the level, the more difficult the assignments. A way to show this could be like on figure 4 which shows eight circles that indicate the number of levels in the current category and the progress made in each level. The color indicates how many stars the player gained in that level. A big red circle indicates the current level reached.

### Point Systems

We want to use point systems, because it is a way to measure the student's abilities, similar to MatematikFessor, described in section 3.3, because we think that getting points is important when you need to keep young people interested and or motivated in something. In figure 4 three stars are shown on a progress bar. You earn stars by reducing the equation. The point system could also help teachers in keeping track of the progresses of the students on each subject.

## Progress

As a part of learning math we want to show the students how far they are in completing an assignment. To do this we will implement a progress bar that illustrates the amount of progress made within a level. Figure 4 shows an example of a progress bar, where the green line indicates how far the students are from completing the level. The progress bar also serves as a part of the point system where stars are earned when you reach a certain point on the progress bar as illustrated in figure 4.

## Achievements

In our solution, achievements are implemented to motivate the students. An example of this is to finish an entire category with the maximum amount of stars. Another achievement could be to complete a level in the fewest possible steps.

## Feedback

As mentioned in section 3.3.1, feedback is important. We want the students to receive instant feedback when they manipulate expressions. The progress bar is used as feedback that shows the students when they reduce or expand the size of the expression.

## Leaderboards

The purpose of leaderboards is to motivate students by letting them compete with each other for a better rank in the leaderboards. Leaderboards are made from logged data from each user based on the achieved points and completed levels. From this, it will be possible to make leaderboards to show which schools are doing the best, which students are doing the best in each class or compare other groups. We want to only show about 25% of the best students in the leaderboards, to prevent the not so good students to be demotivated by being in the bottom of the class.

# 6 Program Description

Based on the idea described in the previous section we developed the program which we will now describe with all of its features. By reading this section one should get an understanding of how the program works and how to use it. As mentioned previously, this game is made for Danish schools which is why the text etc. in the pictures will be in Danish.

## 6.1 Login Menu

By accessing the website <http://webmat.cs.aau.dk/> the user will be presented with a login screen similar to the one shown in figure 5. Here the user has to write their username, typically first name followed by the initials of their last name, and their password. After this, the large green 'Log ind'

button can be pressed which will take the user to the main menu. If the username and/or password is incorrect, a message will displayed indicating the username or password is wrong

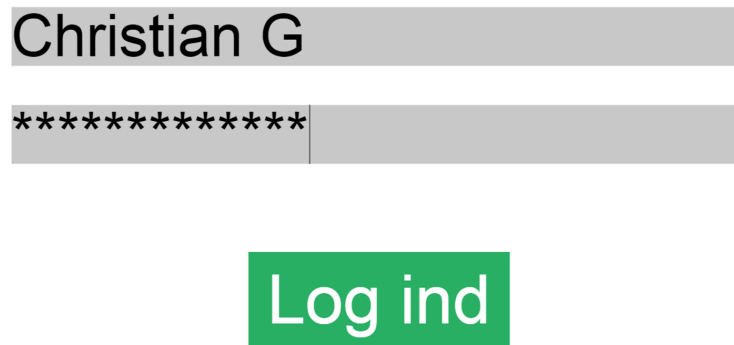


Figure 5: An example of the login screen.

## 6.2 Main Menu

After logging in, the user will be presented with the main menu. An example of the main menu is shown in figure 6. Here the user will be presented with a welcome message including their username along with the badges they have obtained. Two large buttons give the user two possibilities to play. The first green button is the quick-play button which jumps to the first level that the user has not completed. The other button gives access to the different levels of the game. This will be explained with more details in section 6.3. Lastly, on the right, the user is presented with a leaderboard that shows the seven users from the same class that have the highest amount of badges.

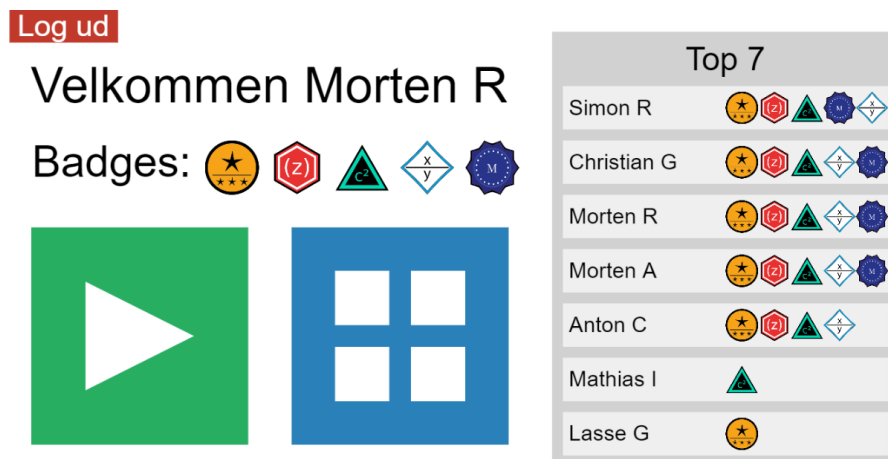


Figure 6: An example of a title menu.

### 6.3 Level Select Menu

If the blue button on the main menu is pressed, the user is brought to the level select menu which can be seen in figure 7. As mentioned in section 5.2 we considered showing circles for each level as shown at figure 4. Instead of this we decided to create a level select menu where the student can use two arrows to navigate between different categories. The title of each category is shown at the top, along with the badge that can be earned if the category is completed. At the top right of the screen, the user is presented with the resume of the total amount of stars earned so far for the selected category, and how many stars are left for the user to be awarded the badge relative to the selected category. Lastly, there is a red *Menu* button at the top left, which enables the user to go back to the main menu.

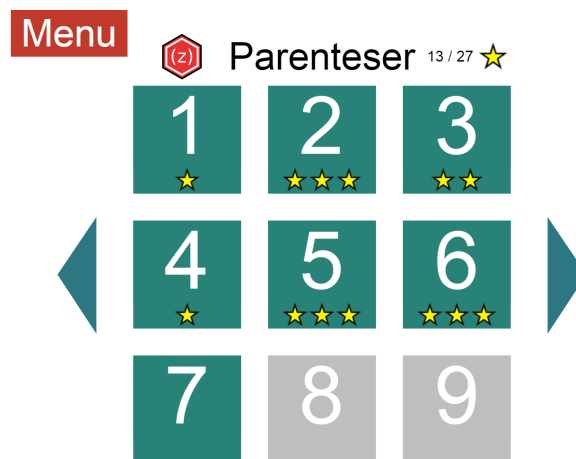


Figure 7: An illustration of the level select menu.

### 6.4 Level Menu

If the green button on the main menu is pressed or if a level from the level select menu is pressed, the user will enter a level. An example of a level is shown in figure 8. The red menu button at the top left, takes the user back to the level select menu. To the right of the menu button, is the restart button which restarts the current level if pressed. To the right of the restart button is the progress bar. This bar displays the user's progress in reducing the mathematical expression with a green color. The progress bar also has three stars which are grey/black when they are not obtained but will turn yellow when obtained. The stars are obtained when the user has made enough progress for the green bar to reach the star. The question mark box, if pressed, displays helping text for the current level. It can be closed again by pressing the box or 'Luk' (Close). Not all levels have this tooltip, if this is the case, the question mark box will be grey instead of blue. If a user has obtained at least one star for the specific level, the button 'Næste' (Next) will go from grey to blue, indicating that the current level has been reduced to some degree, and that it is therefore possible to go to the next level.

The mathematical expression to be reduced is displayed in the middle of the screen. In this case, the user, by having reduced the starting expression, has been awarded two stars, but since it can be reduced more, the maximum amount of stars has not yet been awarded. Pressing meaningful combinations of

variables/numbers will cause different suggestions to be displayed at the bottom of the screen. On figure 8 the user has selected the common factors to factor a outside of the parenthesis. Therefore the program suggests that  $8 * a + 2 * a$  can be rewritten as  $(8 + 2) * a$ .

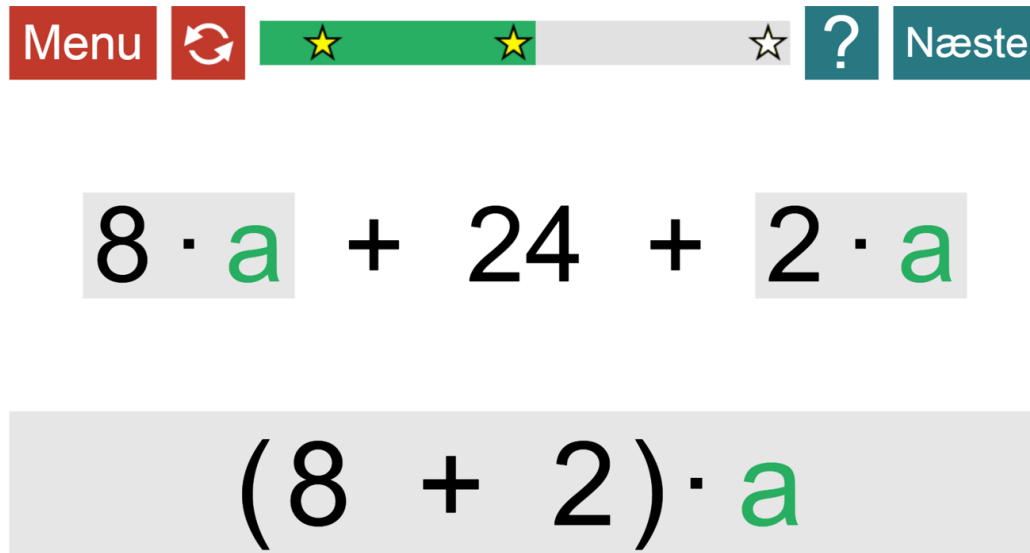


Figure 8: An example of the level menu.

## 7 Development

The purpose of this section is to give a description of the program's structure, by explaining essential parts of the code. This section will explain some of the methods and classes in the program and the choices behind making the program.

### 7.1 Choice of Environment

During the 2nd semester we had a course in object-oriented programming in C# and therefore decided that we wanted to write our program in this particular language. However we faced some challenges with this decision since C# mainly targets the Windows platform and we wanted to develop a program that could be used on multiple platforms. In connection with this we then researched which options we had targeting different platforms while still writing C# code. By contacting a school to test the program we discovered that some of the students at the school use Chromebooks and iPads. Therefore we had to target web because Chromebooks mainly use web applications. To develop in C# for web we found two possibilities the first being Unity3D and the second being Bridge.Net. In the sections below we will briefly describe the two environments and the thoughts behind the choice we made.

## Unity3D

Unity3D [29] is a cross-platform game engine which supports C# scripting. Unity3D offers many advantages, but also has some disadvantages. It offers an interface and tools which make developing games easier, but use more system resources when compared to more barebone approaches which can cause problems for mobile users [30].

There is a number of different communities and online resources, where one can get help or find information about how to develop games using Unity3D. They also have a store, where you can buy assets for use in your game [31], these could add sound/visual effects, offer more interesting physics effects without the need to create these yourself, and can be a compelling option if you do not have any graphics artists on your team. As mentioned earlier, we needed a developing environment where we could test our application across different platforms easily, and since Unity3D does not work very well on e.g. web and have no support for HTML5 [30]. Instead they have a webplayer but this does not work on Chrome, which is one of the most widely used browsers [34, 35]. The last web based option is Unity3D's WebGL version which does work in Google Chrome but still presents some resource overhead. We might consider getting a developer licence for e.g. Android and iOS, if we choose this approach.

## Bridge.NET

Bridge.NET [32] is a development environment which allows programmers to write C# code which is then compiled and translated into JavaScript. The advantages of Bridge.Net are that it allows the programmer to create platform independent applications that can be deployed on almost any device, since most devices support JavaScript through a browser, and because the C# gets transformed into JavaScript, there is no need for getting different development licences. Since Bridge.NET is C# all the way, it means that we can write the whole project in C# using the advantages of Visual Studio when writing the code. Like Unity3D, Bridge.NET also have a vast community [33], where you can seek for help for problems with the system. Bridge.NET also supports popular frameworks like jQuery, Bootstrap and WebGL [32].

## Comparing the Environments

We could have chosen either environment. Unity3D utilizes a game engine that supports all manner of tools used for game development and an interface that makes level editing and all sorts of effects easy to visualize and use as opposed to having no visual representation of the game in the interface with Bridge.NET. Unity3D might for the same reasons present some overhead. We have decided that our game should first of all be available on a web platform, and for this reason we chose Bridge.NET. Unity3D can run in a web context but its webplayer is not supported in the Google Chrome Webbrowser [34]. Chrome is the most used browser according to w3schools.com [35]. Unity3D also has a WebGL version that can run in Chrome but the large engine in general will at least present some amount of overhead and therefore Bridge.NET is the environment we will be using.

## 7.2 Program Documentation

In this section we will describe how we have structured and developed the program.

### 7.2.1 Overall Code Description

The purpose of this section is to describe the overall structure of the program, and the differences between platform dependent code and platform independent code. To illustrate these principles this section will be based on our web application (ThreeOneSevenBee.Development.Bridge.sln) which can be found in the electronic appendix.

#### App

App is the class that contains the program's Main methods which is the entry point that contains the first code to be executed. Furthermore, as illustrated in code snippet 1, the main method is responsible for creating an instance of IContext, IGameAPI and Game. Lastly, the game is started by calling the game.Start() method. When the game is started the Start method will connect the platform dependent code implemented with interfaces IGameAPI and IContext with the cross platform code as described later. It will then check if the user is logged into the system and then either load the game data or direct the user to the login menu.

```
1 class App
2 {
3     void Main(String[] args)
4     {
5         [...]
6         IContext context = new CanvasContext(canvas, input);
7         IGameAPI gameAPI = new JQueryGameAPI();
8         Game game = new Game(context, gameAPI);
9         game.start();
10    }
11 }
```

*Code Snippet 1: Shows the main class.*

Both IContext and IGameAPI are interfaces. The reason for this is that it allows us to change platform by creating other classes that implement these two interfaces. Because we are able to change the classes containing platform dependent code we were able to create an implementation for Web (using Bridge.NET), Desktop (using MonoGame) and Android (using Xamarin). We will now briefly describe the two interfaces IContext and IGameAPI.

#### IContext

IContext is used by all the visual components (Views) to interact with the under-laying graphics-framework. This enables us to easily port the code to other platforms than web. Because Bridge.NET

and C# is targeted at Windows, we had problems compiling on Linux and OS X, resulting in some members running Windows 7 on a virtual machine, to be able to compile which caused some problems. This problem was solved by creating a Desktop version of the program using the MonoGame framework.

```
1 public interface IContext
2 {
3     double Width { get; }
4     double Height { get; }
5     void Clear();
6     void SetContentView(FrameView view);
7
8     void Draw();
9
10    void DrawPNGImage(string fileName, double x, double y,
11                    double width, double height);
12    void DrawRectangle(double x, double y, double width,
13                    double height, Color fillColor);
14    void DrawLine(Vector2 first, Vector2 second, Color lineColor,
15                double lineWidth);
16    void DrawRectangle(double x, double y, double width,
17                    double height, Color fillColor,
18                    Color lineColor, double lineWidth);
19    void DrawPolygon(Vector2[] path, Color fillColor);
20    void DrawPolygon(Vector2[] path, Color fillColor,
21                    Color lineColor, double lineWidth);
22    void DrawText(double x, double y, double width, double height,
23                string text, Color textColor,
24                TextAlignment alignment);
25
26    Vector2 GetTextDimensions(string text, double maxWidth,
27                            double maxHeight);
28 }
```

*Code Snippet 2: Shows the IContext interface.*

As shown in code snippet 2, IContext contains methods to draw polygons, text and images on the screen. IContext has two properties, Width and Height which have the screens actual width and height in pixels. We have also made a method called GetTextDimensions which renders a string, without putting it on the screen, and returns the dimensions of the string. A Clear, Draw, and SetContentView is required to be implemented. When a FrameView is passed to the SetContentView then the class that implements IContext has the responsibility to call the FrameView's DrawWithContext method as well as Click and Keypressed methods when the user clicks or presses a key.



## IGameAPI

IGameAPI is our interface for the application programming interface (API) which enables the communication between our program and the web server. As shown in code snippet 3, the interface contains methods for e.g. saving user level progress, adding badges and getting a list of all players. The interface is designed as a callback pattern. This comes from a requirement for some platforms such as web running JavaScript, where requests can be handled asynchronous. It also keeps implementations responsive by having all calls be non-blocking. This is an advantage since web implementations tend to have a comparatively long response time. The interface can still complete the functions instantly if possible, and simply call the callback right before returning.

```
1 public interface IGameAPI
2 {
3     void IsAuthenticated(Action<bool> callback);
4     void logout(Action<bool> callback);
5     void Authenticate(string username, string password,
6                       Action<bool> callback);
7     void GetPlayers(Action<List<Player>> callback);
8     void SaveUserLevelProgress(int levelID,
9                                string currentExpression,
10                               int stars, Action<bool> callback);
11    void GetCurrentPlayer(Action<CurrentPlayer> callback);
12    void UserAddBadge(BadgeName badge, Action<bool> callback);
13 }
```

*Code Snippet 3: Shows the IGameAPI interface.*

## Game

When the Game class is instantiated it takes two interfaces as parameters the IContext and IGameAPI which contains the platform dependent code. When the Game's Start method is called in the programs entry point, IGameAPI's method IsAuthenticated is used to determine whether or not the LoginView should be shown. When the user is logged in, the IGameAPI is used to retrieve the user data and a list of players from the database, which is then used to construct a GameModel. The GameModel is used to construct a GameView which is passed to IContext's SetContentView to display it on the screen.

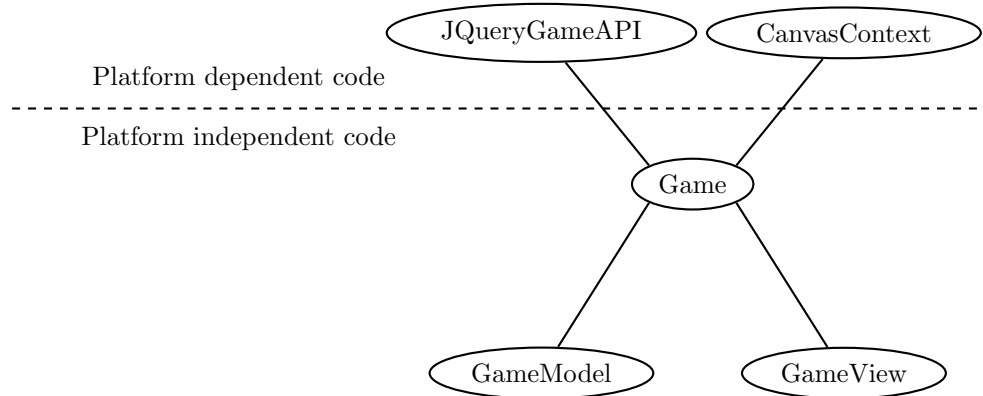


Figure 9: Describes the separation of platform dependant and independent code, as well as separation within the independent codebase into model and view. JQueryGameAPI and CanvasContext inherit from IGameAPI and IContext classes respectively.

Figure 9 shows the five main components of our program each with their own responsibility. The core logic and data behind the program is managed by the GameModel. GameView then represents the data in GameModel and makes this controllable. The Game combines GameView and GameModel with the platform dependent code through IContext and IGameAPI. This separation of the program is somewhere similar to the Model View Controller (MVC) design pattern. In MVC a program can be divided into three main components: the model, view and controller. In MVC the view is a graphical representation of the data in the model and the controller is responsible for all actions specified in the view [36].

### 7.2.2 From String to Expression

Before describing the model and view behind the program it might be helpful to understand an essential part of our program which is how we represent expressions in the code. This section describes how expressions are represented in our program and how the parsing mechanisms works.

The arithmetic expressions that we deal with contains, variables, numbers, operators and constants. These elements constitute the atomic entities of an expression and will be referred to as tokens. An example of an expression is the following:

$$x + 4 - 3$$

In our program an expression is represented in the form of an expression tree which is a specific type of data structure for a handy representation of expressions. In an expression tree each node represents an expression, and the root of the tree contains the entire expression. To further understand how an expression is represented as an expression tree, we must first look at the individual parts of an expression tree and see how different tokens are represented in the program.

The figure below illustrates the class hierarchy for the implementation of the following expressions: constants, variables, functions, delimiters and numeric expressions.

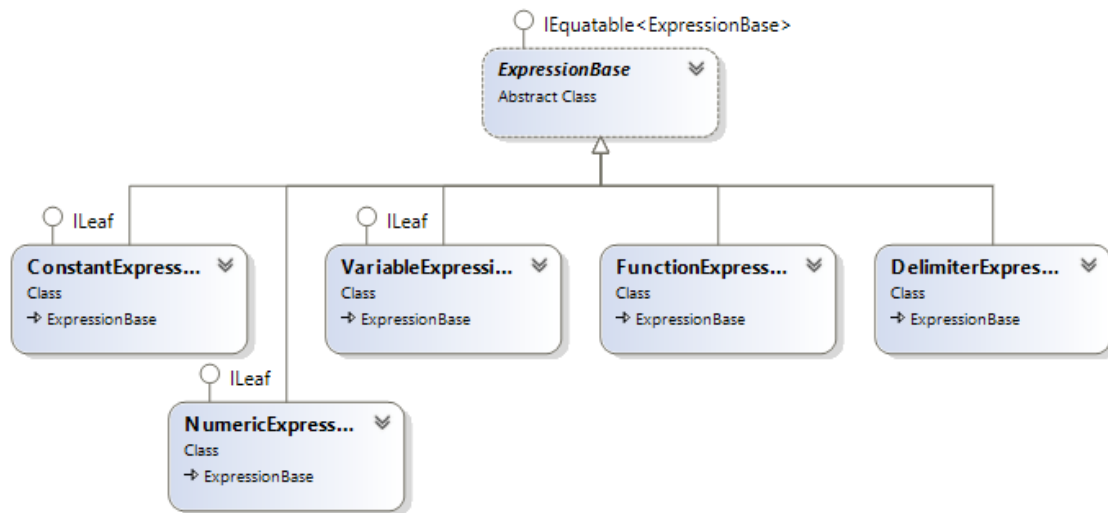


Figure 10: The class hierarchy for expressions (part 1/2). - Generated by Visual Studio.

As illustrated in figure 10, the super class for all expressions is the ExpressionBase class which is an abstract class containing implementations that are common for all expressions. The figure also shows that the classes ConstantExpression, VariableExpression and NumericExpression all implement the ILeaf interface which means that whenever one of these types of expression is represented in the expression tree, they are represented as leaves, because none of these classes have any child nodes.

In the program we chose to represent different operators differently in our tree structure. This is done to ease the implementation of different mathematical rules which is described later in table 5. As a result of this, all operators inherit from the class OperatorExpression which derives from ExpressionBase. The class hierarchy for all operator expressions are shown in figure 11.

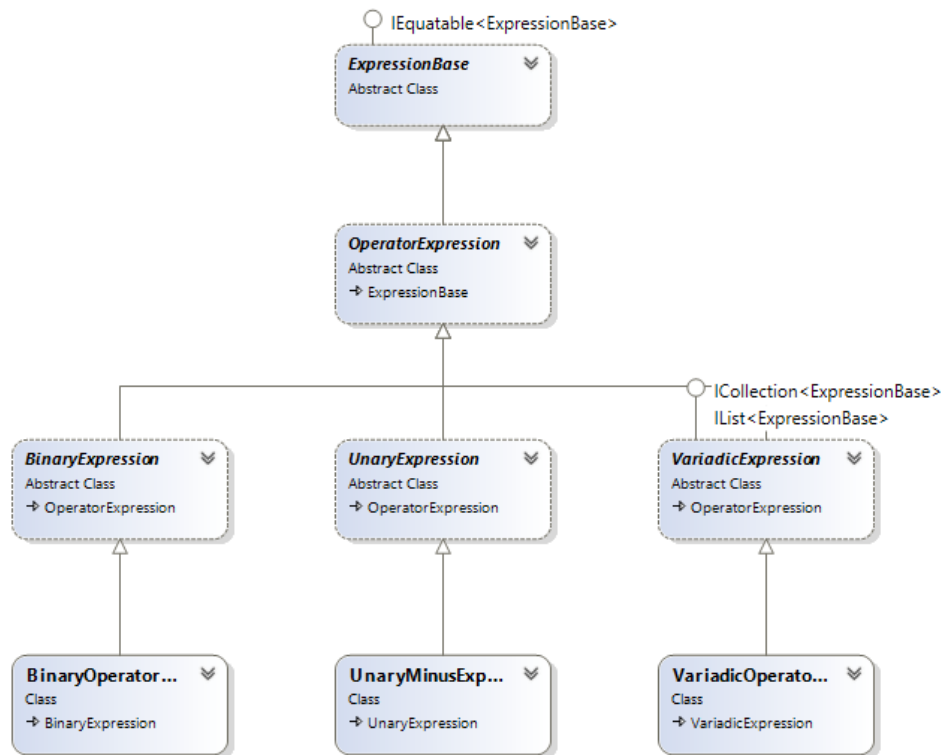


Figure 11: The class hierarchy for expressions (part 2/2). - Generated by Visual Studio.

The class `BinaryExpression` is the base class of all binary operators. A binary operator consists of an operator type and two child nodes. In our representation of a binary expression the operator type is of type division or power.

Another type of operator expression is `VariadicExpression` which is the base class to all variadic operator expressions. What sets the `VariadicExpression` apart from the `BinaryExpression` is that the `VariadicExpression` contains a list of child nodes. As with the `BinaryExpression` the `VariadicExpression` must contain at least an operator type and two children. In our representation of a `VariadicExpression` the type is either multiply or addition. Note from table 2 that the main difference between `BinaryExpression` and `VariadicExpression` is that all operator types in `VariadicExpression` satisfy the commutative law.

The last type of operator expression is a `UnaryMinusExpression` which is an expression of operator type minus that only has one child.

The table below shows the connection between classes and operators.

Class	Operators
BinaryOperatorExpression	$\wedge /$
VariadicOperatorExpression	$+ *$
UnaryMinusExpression	$-$

Table 2: Shows which classes that represents different operators.

A special-case in the structure of our expressions is that binary subtraction is represented in the expression tree by using a combination of a VariadicOperatorExpression with OperatorType Add and a UnaryMinusExpression. Lets say we have the algebraic expression  $2 - a$ , the figure below shows its equivalent representation in form of an expression tree.

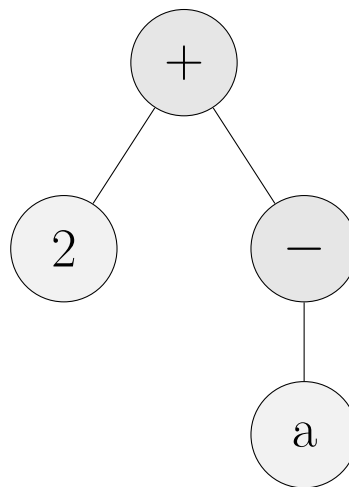


Figure 12: The expression tree for  $2 - a$ .

As we can see in figure 12 the root of the tree is a variadic expression of operator type addition which has two child nodes, where the left child in this case is a numeric expression. The right child of the root is a UnaryMinusExpression containing a VariableExpression. As it can be seen on the figure this particular expression tree has two leafs which all consist of expressions that implement the ILeaf interface.

## ExpressionParser

In our program the class ExpressionParser contains the necessary methods to convert a string representation of an algebraic expression in to an expression tree. One of the main methods in this class is called InFixToPostFix and is a method which takes an infix representation of a expression string and then converts it to a postfix representation of an expression. The return type of this method is a list of tokens.

The following illustrates a postfix and an infix expression:

Infix:  $2 + 3$

Postfix: 2 3 +

The method for converting an infix expression into a postfix expression is based on the "Shunting yard algorithm" [37]. The pseudo code in appendix B describes the InFixToPostFix method which is our variation on the "Shunting yard algorithm".

When the expression has been converted into a postfix expression the ExpressionParser's Parse method converts it into an expression tree by dequeuing tokens off the postfix expression queue, and using the classes derived from ExpressionBase to build the tree as described previously.

### 7.2.3 Size

We wanted to have a progress bar of how much a given expression could be reduced, to do this we introduced a property called Size that associates with each expression an integer measuring the arithmetic complexity of the expression. In order to design which expression should get a lower Size value than others we made some rules, which can be seen on table 3. These are statements that have to be true for the program to give the correct Size values, we have given the expression points so they correspond with the statements where  $[expression]$  denotes the Size value of  $expression$ .

$[a + a] > [2 * a]$	$[a * a] > [a^2]$	$[a * a * a] > [a^2 * a]$
$[a * b + a * c] > [a * (b + c)]$	$[(a)] > [a]$	$[- - a] > [a]$
$[1 * \frac{a}{b}] > [\frac{a}{b}]$	$[a^2 * b^2] > [(a * b)^2]$	$[1 * a] > [a]$
$[\sqrt{4}] > [2]$	$[\frac{a}{a}] > [1]$	$[2 + 2 + 2 + 2 + 2] > [2 * 5]$

Table 3: Shows the relation between different expressions' Size values denoted with  $[expression]$ .

With these rules we can now begin to assign a size value to expressions that is in agreement with the rules. Notice the table below which consists of different expressions that implement the Size method, what the expression consists of and how many points it gives.

Expression	Represents	Size
VariableExpression	Variables (a, b, c etc.)	2
NumericExpression	Numbers (1, 2, 3 etc.)	1
ConstantExpression	Constants ( $\pi$ )	1
FunctionExpression	Sqrt, sin, cos, tan	1 + <i>size of the subexpression</i>
DelimiterExpression	Parenthesis	1 + <i>size of the subexpression</i>
UnaryMinusExpression	Unary minus	1 + <i>size of the operand</i>
BinaryOperatorExpression	Divide, power	1 + <i>size of operands</i>
VariadicOperatorExpression	Add, multiply	( <i>number of operands</i> - 1) + <i>size of operands</i>

Table 4: Shows how size of expressions are evaluated.

The size of an expression is calculated according to the rules described in Table 4. For example with a

simple division  $\frac{1}{2}$ , since a fraction goes under BinaryOperatorExpression we can see that the size of this expression is three and is computed as follows:

$$[1/2] = 1 + [1] + [2] = 1 + 1 + 1 = 3$$

**NumericExpression and ConstantExpression** These two are some of the expressions that give the least amount of points, because numbers and constants can not be further reduced.

**VariableExpression** Variables has a bigger Size than numbers or constants, because otherwise the expressions  $a + a$  and  $2 * a$  would give the same amount of points and following the statements in table 3 then  $2 * a$  should be the most simple of the two.

**FunctionExpression** In our program FunctionExpression is only used for square root. An example of this could be  $\sqrt{a}$ , which computes as follows.

$$[\sqrt{a}] = 1 + [a] = 1 + 2 = 3$$

**DelimiterExpression** We gave DelimiterExpression one point since it could not be zero because then  $[(a)] = [a]$  and if we gave it two or more it would interfere with some of the parenthesis statements so that  $a^2 * b^2 > (a * b)^2$ .

**UnaryMinusExpression** This expression's size is the size of its operand plus one and are used in expressions like  $-a$  or  $a^{-2}$ . For example the size of  $a^{-2}$  which have both a power and a unary minus computes as follows.

$$[a^{-2}] = 1 + [a] + [-2] = 1 + 2 + 1 + [2] = 1 + 2 + 1 + 1 = 5$$

**BinaryOperatorExpression** The binary operators 'division' and 'power' are treated in the same way. I.e.  $[\frac{1}{3}] = 1 + [1] + [3] = 1 + 1 + 1 = 3$  as described under Table 4. The definition of the size of power is, because we wanted  $a^2$  to be simpler than  $a * a$ . An example of how power is computed is as follows:

$$a^2 = 1 + [a] + [2] = 1 + 1 + 1 = 4$$

**VariadicOperatorExpression** When calculating the size of a VariadicOperatorExpression you sum the sizes of the operands and add the number of operands minus one (Number of plus or multiplication symbols). This is seen in code snippet 4.

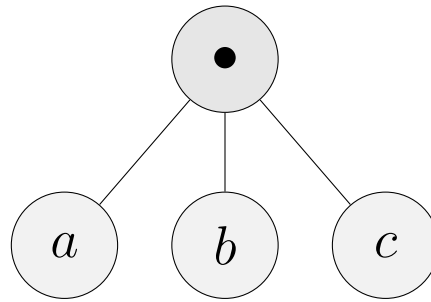


Figure 13: Shows the tree structure of  $a * b * c$ .

As seen in figure 12 when we have an expression like  $2 - a$  a `UnaryMinusExpression` is put in a `VariadicExpression` in these situations we have decided to give the unary minus zero points, because we wanted expression like  $2 - a$  and  $2 + a$  to give the same amount of points. Following the code in code snippet 4 the if statement ensures that only the size of the expression behind the minus is added to result. An example is shown below.

$$2 - a + 4 = (3 - 1) + [2] + [a] + [4] = 2 + 1 + 2 + 1 = 6$$

```

1 // Count-1 gets all operators in the variadic expression
2 int result = (Count - 1);
3 foreach (ExpressionBase expression in this)
4 {
5     // Cast to get what is behind the unaryminus
6     UnaryMinusExpression minus = expression as UnaryMinusExpression;
7     if(minus != null)
8     {
9         result += minus.Expression.Size;
10    }
11    else
12    {
13        result += expression.Size;
14    }
15 }
16 return result;

```

Code Snippet 4: Size implementation in `VariadicExpression`.



## GameModel

The GameModel class has multiple properties to handle the input it receives. These can be seen in code snippet 5. As shown in code snippet 5, GameModel has properties that hold information describing the state of the program such as the User which contains all the information describing the student's state in each level. Another property is Players which is a list of all students' public information such as name and badges. Properties CurrentExpression, StartExpression and StarExpressions are used to store information about the level that the student is currently trying to complete. CurrentExpression is manipulated with the class ExpressionModel as described later.

```

1     public class GameModel
2     {
3         public CurrentPlayer User { get; }
4         public IEnumerable<Player> Players { get; set; }
5         public ExpressionModel ExprModel { get; private set; }
6         public ExpressionBase CurrentExpression { get {
7             return ExprModel.Expression; } }
8         public ExpressionBase StartExpression { get; private set; }
9         public List<ExpressionBase> StarExpressions { get; private set; }
10        [...]

```

*Code Snippet 5: Properties of the GameModel class.*

After getting information from the controller, and with the information that GameModel already has through User, GameModel updates the view. This is done through various methods, such as SetLevel, RestartLevel and UpdateLevelData. Part of the UpdateLevelData method is shown in code snippet 6.

```

1     private void UpdateLevelData()
2     {
3         if (ProgressBar.ActivatedStarPercentages().Count() >
4             User.CurrentLevel.Stars)
5         {
6             User.CurrentLevel.Stars = ProgressBar.ActivatedStarPercentages().
7                 Count();
8             if(GetNextLevel() != null)
9             {
10                GetNextLevel().Unlocked = true;
11            }
12        }
13    }

```

*Code Snippet 6: Shows how the UpdateLevelData method in GameModel determines if next level should be unlocked.*

In code snippet 6 it is shown in line 3 that if the amount of reached stars is more than what the user already had reached, it changes the amount of stars the user has to the new amount. Furthermore, if the method GetNextLevel is not null, it unlocks next level which is shown in line 8-11.

## ExpressionModel

ExpressionModel holds an instance of an ExpressionBase, and it is responsible to update and insert identities of the current selection depending on a set of rules. This is done whenever ExpressionModel's Select Method is called which also invokes an OnChanged event, which is used to inform other classes about changes. ExpressionModel uses an instance of a class named ExpressionAnalyzer to get identities of the current selection.

## ExpressionAnalyzer

ExpressionModel uses ExpressionAnalyzer's methods GetCommonParent and GetIdentities. GetCommonParent is an algorithm that finds the common parent of all selected subexpressions in an expression. Examples of selections and the corresponding common parent are shown in figure 14.

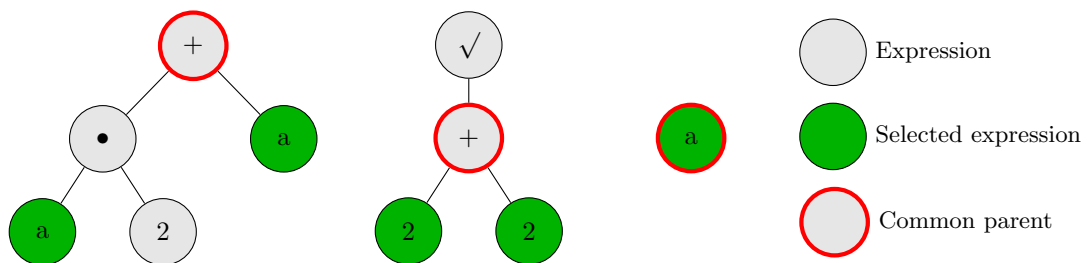


Figure 14: Shows common parent for selections in expressions  $a * 2 + a$ ,  $\sqrt{2 + 2}$  and  $a$ .

The functionality of an ExpressionAnalyzer can be changed by adding or removing an ExpressionRule. An ExpressionRule is a delegate that represents a mathematical rule like  $a * a = a^2$ . A list of ExpressionRules are used in the method GetIdentities to get a list of identities to the common parent of the given subexpressions.

## ExpressionRule

For each algebraic rule in table 1 there is a corresponding ExpressionRule. Names of these are shown in table 5. All implemented rules are located in a static class named Rules. All ExpressionRules check if a given expression can be substituted with another expression according to a specific rule. If a substitution is possible it returns the rewritten expression. If it is not possible it returns null.

Rule	Method Name
$\frac{a}{c} + \frac{b}{c} = \frac{a+b}{c}$	AddFractionWithCommonDenominatorRule
$\frac{a}{b} \cdot \frac{c}{d} = \frac{a \cdot c}{b \cdot d}$	ProductOfFractions
$a \cdot \frac{b}{c} = \frac{a \cdot b}{c}$	ProductOfConstantAndFraction
$a^{-n} = \frac{1}{a^n}$	VariableWithNegativeExponent
$a^0 = 1$	ExponentToProductRule
$a^n \cdot a^p = a^{n+p}$	ExponentProduct
$a \cdot b + a \cdot c = a \cdot (b + c)$	ProductParenthesis
$a^n \cdot b^n = (a \cdot b)^n$	CommonPowerParenthesisRule
$(a^n)^m = a^{n \cdot m}$	ParenthesisPowerRule

Table 5: Shows names of some of the implemented rules.

### Minimizing ExpressionRules responsibility

In an earlier version of the program, the common parent with all its subexpressions was passed to each rule. If the common parent of a selection was a VariadicOperatorExpression like  $a * 5 * a$  where the two a's are selected then each rule had to manage the unselected 5 even though it was not important to a rule like  $a * a = a^2$ . To prevent this, the ExpressionAnalyzer now removes all operands that do not contain any selected subexpressions before it passes it to the rule and inserts them again in the expression returned from the rule.

#### 7.2.4 View

In the previous section we described the model of the program which contains all the core data and logic behind the program. Following is a description of the view part of the program, which presents the data in the model. All classes that serve as a visual components like buttons, text or images derive from the View class. The View class itself represents a rectangle with its color described in the BackgroundColor property. In the constructor the View takes its dimensions X, Y, Width and Height. The View has a virtual method for drawing called DrawWithContext which takes an IContext and uses its methods to draw itself on the screen. Classes that derives from View overrides this method to draw different visual components. To describe the boundary of the View the method ContainsPoint determines whether or not a given point is inside the view, using the View's dimensions. Because our view is part of the cross platform code, we have implemented the Scale method to target different devices' screen sizes. The Scale method simply scales the View's dimensions with a given factor.

#### KeyPressed and Click methods

The View class contains two methods called KeyPressed and Click along with two events called OnKeyPressed and OnClick. If Click is called and the given click coordinates are inside the View then the OnClick event is called. The Click method controls a boolean variable called Active which tells whether or not an object is in focus. If the object is clicked, it will gain focus and if a click happens outside the

object it loses focus. Other objects can assign methods to the OnKeyPressed and OnClick events to get notified when an object is clicked or a button is pressed while the object is in focus. Both the click and KeyPressed methods take a context to be able to use graphical methods. KeyPressed takes a string containing what letter was pressed or a string representation of what button was pressed. Special keys could be "Space", "Back" and "Right", representing the space-key, back-space and the right arrow-key.

Figure 15 shows the class hierarchy for 14 out of 22 view classes.

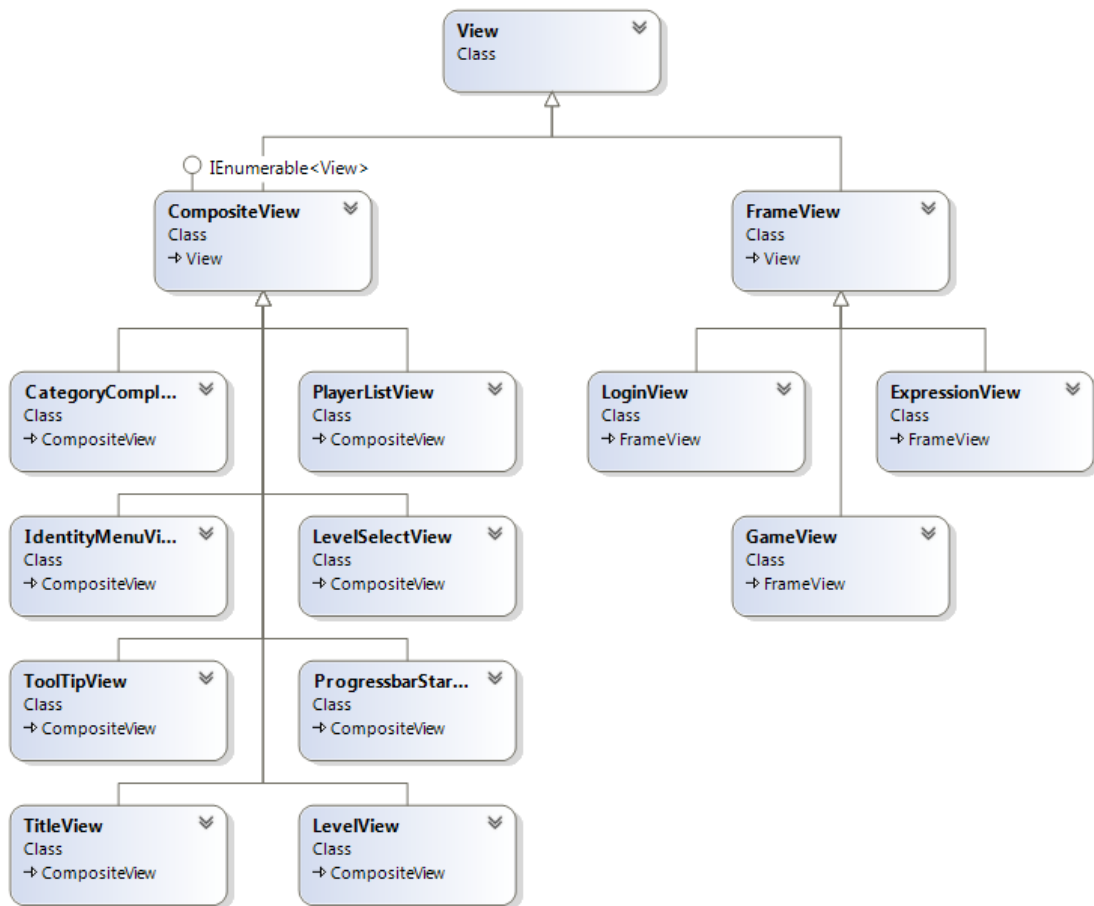


Figure 15: UML-diagram of implemented views. - Generated by Visual Studio.

The following sections give a brief description of some of the classes that derive from View.

### CompositeView

CompositeView inherits View and contains a list of Views called Children. When CompositeView's Click and KeyPressed methods are called it propagates these down to all its Child-views, as shown in code snippet 7.

```
1
2 public override void Click(double x, double y, IContext context)
3 {
4     if (base.ContainsPoint(x, y))
5     {
6         Active = true;
7         if (PropagateClick)
8         {
9             foreach (View child in Children)
10            {
11                child.Click(x - X, y - Y, context);
12            }
13        }
14        if(OnClick != null)
15        {
16            OnClick();
17        }
18    }
19    else
20    {
21        Active = false;
22    }
23 }
```

*Code Snippet 7: Click implementation in CompositeView.*

The Active property is true whenever one of the Views in Children is active. CompositeView is also enumerable to be able to iterate through all Children.

## FrameView

FrameView inherits View and is used to enclose another View stored in the property Content. When the FrameView's method SetContent is called, the given View is scaled to fit inside the dimensions of the FrameView.

## GameView

GameView is responsible for showing the different menus of the game as described in section 6. GameView inherits from FrameView to be sure that the GameView fits the size of the device's screen. The GameView acts as a visual representation of the data in the GameModel and is used to interact with the model. GameView contains two events OnExit and ReloadGame. OnExit is called when the user clicks the logout button in the Title Menu and ReloadGame is called whenever the Game data from the database should be reloaded.

## ExpressionView

ExpressionView inherits from FrameView. ExpressionView visualizes an ExpressionModel using e.g. OperatorView and SqrtView to visualize different expressions. This is achieved by the recursive BuildView method which takes an ExpressionBase and ExpressionModel as parameters and returns a view representing the given expression. BuildView uses the passed ExpressionModel to set OnClick events so that the user can select parts of the expression. If the passed ExpressionBase is UnaryMinusExpression, BinaryOperatorExpression, VariadicOperatorExpression, DelimiterExpression or FunctionExpression then the BuildView will be called recursively to build views for each subexpression/operand which is then combined in a CompositeView together with operators, parenthesis or square root. If the passed expression is none of these types, then it hits the base case which just creates a LabelView showing the text representation with the ExpressionBase's ToString method.

Because ExpressionView derives from FrameView, the size of the ExpressionView can be set to a fixed size instead of depending on the number of elements in the expression.

## LabelView

LabelView inherits from View and contains a string which gets printed in the view. This is used in inputbox and ButtonView, to render text on to the screen. The string contained in LabelView gets scaled to fit the dimensions of the LabelView.

## Inputbox

Inputbox inherits the LabelView and shows an editable text. It uses the inherited Text property from LabelView to render text on to the screen and alters this according to the input. A default text to show when nothing has been entered yet and showing asterisk symbols instead of letters for passwords is also implemented and set using parameters in the constructor.

### 7.2.5 Database and API

To be able to store data between sessions, retrieve other players' data for making the leaderboard and to make it easy to update and create more levels, we use a database with an API interface.

**Request Structure** Requests to the API can be either POST or GET requests (Types of web requests) with data. The only required fields are the action and token (token is not required for the log-in). Action is the name of an API method that the request wants to execute, see table 6. The token is used for authenticating a user, so that, e.g. only the user itself can update his/her own progress in a level.

Action	Parameters
user_login	username, password
user_logout	token
get_users	token
get_levels	token
save_user_level_progress	token, level_id, current_expression, stars
get_current_user	token
is_authenticated	token
user_add_badge	token, badge_id

Table 6: API methods.

**Response structure** All responses are formatted as JSON. The response will always contain a success flag that is either true or false. If the success flag is false, there will usually be a 'message' entry that describes the error. Otherwise the 'data' entry will contain the data that was asked for, if any.

```

1 {
2   "success": "true",
3   "data":
4   {
5     "id": "49",
6     "name": "Anton C",
7     "token": "Q19M1GJ9933ZX6DU",
8     "badges": ["4", "3", "2", "0"]
9   }
10 }
```

Code Snippet 8: Response from get\_current\_user request.

**Token-system** In early stages of development we used a cookie to store a session-id that authorized users after they had logged in. To support more clients than just web, (see interface descriptions in section 7.2.1), we later chose to associate a token, represented by a random string, with a user when they log in. This token is sent back from a successful log-in request, and is then stored by the client and sent as parameter of later requests to authenticate the user.

**Actions** are the different functions that the API offers. Here follows a short description for each of the actions in table 6.

- **user\_login** : If a user with the provided username and password exists then a token is returned that should be used for all future requests in a given session.
- **user\_logout** : Invalidates the users token so that it cannot be used anymore.
- **get\_users** : Returns a list of users in your class and their badges.

- **get\_levels** : Returns a list of levels. A level consists of its unique id, a string representation of the levels expression, the expression that the user had left the level, the number of stars the user has gotten in that level and a list, called `star_expressions`, which are used to mark the simplicity of the expression needed to achieve different amounts of stars in that level.
- **save\_user\_level\_progress** : Given the id of a level this API call writes the users progress and number of stars achieved in that level to the database.
- **get\_current\_user** : Returns the users username and the badges the user has earned.
- **is\_authenticated** : Success is true for this request if the users token is valid.
- **user\_add\_badge** : Adds a badge to the user's list of badges.

**Database structure** Figure 16 shows the tables of the database. Table columns with name in the format `[table_name].id` are used to chain table data together in one-to-many relations. An example of this could be, a school has many classes and a class has many users.

The privilege and supervisor tables did not end up getting used but would have been used to restrict users, teachers and administrators to different access/privilege levels, so that only an administrator could edit levels and teachers could monitor student progress and access class statistics.

The school and class tables are mainly used to show the leaderboard table populated with people from a user's own class, see an example of this in figure 6. It could easily be used for showing statistics based on school or classes.

Level and level\_category contain the string-formatted levels and associated data, like `star_expressions` which describes the simplicity an expression, is required to be simplified to in order to unlock stars.

User\_level\_progress contains the number of stars obtained in a level and the expression the user ended with the last time they played the level.

The user table mainly organizes authorization data like a hashed password string, username and the valid token assigned to the client when the user is logged in, but it also stores the badges a user has obtained as the only game state related information in the user table.



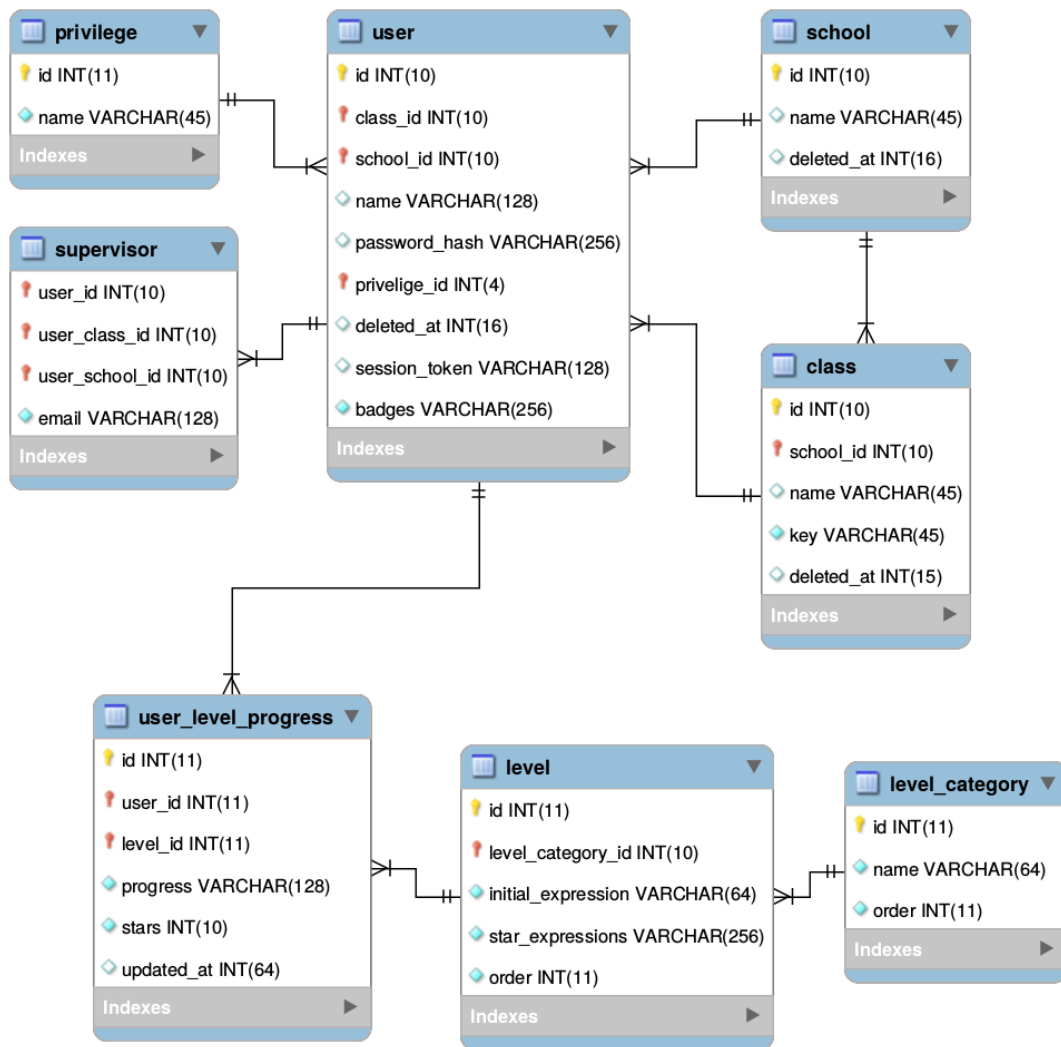


Figure 16: Database diagrams with table relations. - - Generated by MySQL Workbench.

## 8 School Visit

This section describes a test of the program at a school, the data we collected, and lastly an analysis of the collected data.

### 8.1 Purpose

We wanted to test our program by having a group of students in our target group 7th to 9th graders try the program. The purpose of this was to collect data through questionnaires and behavioral analysis to get a better understanding of the end users' experience with our program. Moreover, the test is also meant to discover to what extent the motivational affordances we implemented increase the incentive of the users to prefer our solution rather than doing tasks by hand. Finally the test date was also a motivational deadline for our development phase that pushed us to work efficiently to reach a stable prototype of the program.

### 8.2 Method

To test our program we had contacted several schools and ended up making an appointment with Vesterkærets Skole, where we were able to test the program on 7th grade students. Before the school visit we prepared a short questionnaire for the students to fill in when they finished using the program. The questionnaire can be seen in appendix A. Before meeting the class, we met with the teacher, Lars Gross Pedersen, to discuss the basics of the program, and the questionnaire, we wanted the students to fill in. We agreed that Lars would gather the student's attention and present us along with the purpose of trying out this program, after which we would make a short introduction to the program, e.g. mentioning the website where the program can be accessed and handout usernames and passwords printed on the questionnaire that they answered after the test.

To test the usability of the program, we wanted the students to run the program without any influence from us, this way we would be able to see which elements of the program causes problems and which of the tasks were too hard or too easy. But after seeing multiple students stuck on the same kind of levels, we decided that it was necessary to help them to some degree, to make sure we would get the most out of the testing (get all levels tested). This did, however, affect the conclusions we could draw from the testing, since the students did not do all the work themselves.

### 8.3 Data

As part of our project, we wanted to test our program to get a better idea of what problems users run into. Therefore we made a short questionnaire with a number of statements, the students could then say if they agreed or disagreed with the statements. The data from the questionnaire can be found in appendix F and are visualised in figure 17.

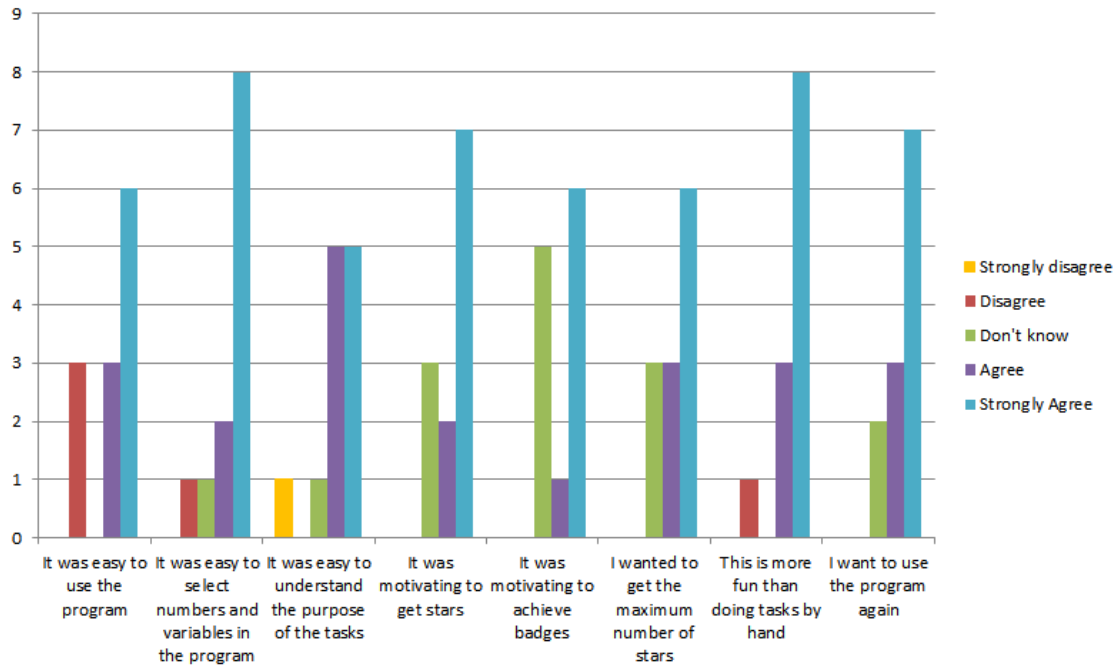


Figure 17: Chart showing the answers from eight of the questions in the questionnaire.

The last question was: "Do you have any ideas or suggestions for improvements of the program?" Which was intended to get more qualitative data from the students. A couple of students answered this question. One asked for some background music, another asked for more levels and a third student, said it would make it easier, if there was a voice to explain individual levels. If we had more time after the school visit then we could have used comments like these to improve the user's experience of the program by following a user centered design.

To analyze the behavior of the students when using our program, we measured how much time the students used on the different tasks on average. Figure 18 compares the amount of time it took to solve each level, broken down in the different categories. Each bar represents the estimated average time from all students that has completed the specific level. The times used to make the chart has been taken from our server and is listed in the table of appendix E. The table in appendix E may seem to have some odd values. Some fields are empty and some values are simple too low, representing an unreasonably short time taken to complete a level. Empty fields are mostly levels that the user has not played, otherwise due to the way we measured time, the empty fields can also be the level the user played the longest time ago which for our test is mostly the first level. Some values like that of user 6 in level "Parenteser 8" are very low compared to the average, this likely means that a user did not complete the level but returned to the menu partway through, as this action would also store the current time in the database.

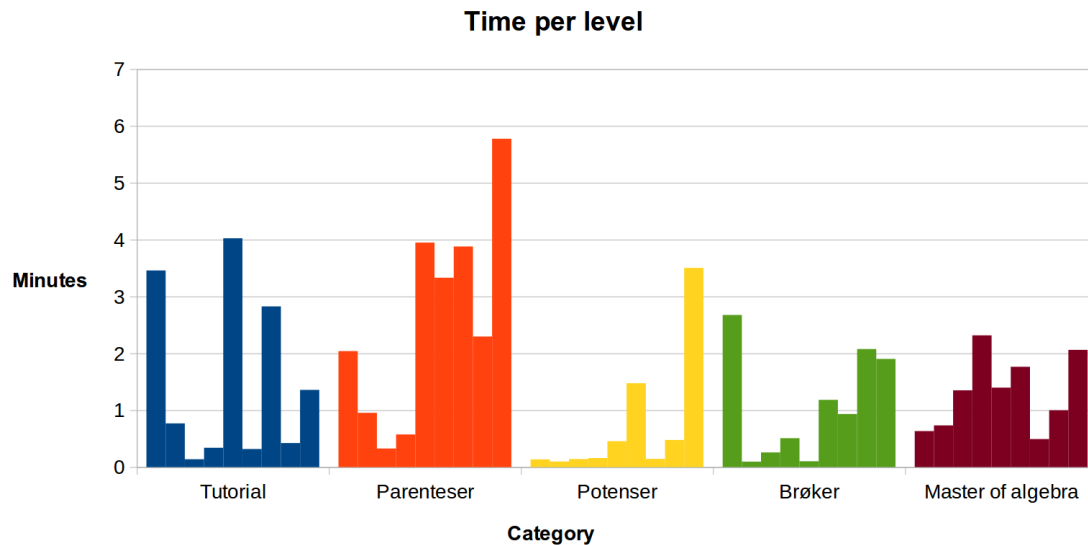


Figure 18: Chart showing the amount of minutes used on each level.

The following equation is used to calculate the estimated time used to complete a level. To measure the time a level took to complete, we recorded the unix time at the moment students completed the  $n$ th level and then calculate the time the  $n$ th level took to complete  $T_n$  as the timestamp  $t_n$  recorded for the level, subtracted the timestamp  $t_{n-1}$  recorded from level  $n - 1$ th level. Notice here that the  $n$ th level refers to the levels in the order of which the individual user played them and as such might not be the same as the actual order of the levels in the game.

$$T_n = t_n - t_{n-1}$$

However there can be some deviations from the actual time spent on a level, since this could also include time the students spend e.g. in the level-select menu. If a student for example complete the first category and uses two minutes looking at badge on the Main Menu, then these two minutes will be added to the time it takes to complete the next level.

## 8.4 Observations

There were multiple students that did not realise that they were done with a level. Since the progress bar is already large, this could be fixed by explaining when a level is finished by text, or a pop-up on the first level, so the users of the program, would know in later levels, how to know if they were done with a certain level.

The tooltip was used by a few of the students, some students requested tips at some levels that they found too difficult, but did not have tips. Some students did not use the tooltip when they were stuck on a level for an extended amount of time. This problem could be solved, by drawing attention to the tooltip button after an extended amount of time, where the student has not made any progress.

During the testing by the students, we wrote down notes of their reactions, frustrations and more. Here we saw a tendency growing, where multiple students had problems with the same kind of levels. An example of this, is a level, where they needed to solve the problem:  $3 * x - (4 * x - 5)$ . The negation in front of the parenthesis was not handled correctly, and therefore caused great trouble for some students.

The majority of the students sometimes guessed by clicking randomly on objects and suggestions instead of thinking about the best action. This worked in the introductory levels, but not in later levels. The introductory levels that were meant to teach some of the basic ways to solve levels did not stick with the students as they rushed through them without learning from them, and therefore they were not able to solve other levels that were based on understanding the introductory rules.

## 8.5 Analysis

In this section we will analyze the data collected during the school visit as seen in section 8.3.

**It was easy to use the program** To understand the difficulty of our program seen from a 7th grader's perspective, we wanted them to answer this question. 9/12 answered 'agree' or 'strongly agree' that the program was easy to use, while three students disagreed. The three students who disagreed all obtained less than three badges.

**It was easy to select numbers and variables in the program** This question goes more in depth with the difficulty of selecting the numbers and variables when reducing the expression. Here 8/12 students strongly agreed with the statement. There was a minor learning curve, but after the tutorial, most of the students knew how the select mechanism worked and used it properly.

**It was easy to understand the purpose of the tasks** It is essential, in math, to understand the purpose of calculating problems. Therefore we wanted to make sure that the students understood this purpose. 10/12 students either agreed or strongly agreed that they had understood this purpose. One student answered 'strongly disagreed'. This student only obtained the tutorial badge.

**It was motivating to get stars** This statement was to examine whether the students felt motivated or not. Here 2/12 agreed and 7/12 strongly agreed, while 3/12 answered "don't know". This could be because the stars were always visible, so during a task the students could see how many stars they had obtained within a level.

**It was motivating to achieve badges** This question was also made, to examine the motivation, but this time it is regarding badges. Here 7/12 students agreed or strongly agreed that they found the badges motivating, but there were still 5/12 students, who said "don't know". Comparing the individual student's survey answers with the amount of badges they received, a clear tendency is shown, where 5/6 students, who got all five badges voted 'strongly agree' and 4/5 who got 2 badges or less voted 'don't know'. This tendency could show that while badges work great for some, it is not enough to simply hand out rewards with no inherent value in order to motivate people. Perhaps if the badges unlocked special content or could otherwise be used to some purpose they would have a greater effect.

**I wanted to get the maximum number of stars** This is similar to the question 'It was motivating to get stars', but here it is focused on, if the individual students were motivated to receive all three stars per level, before moving on to the next level. 6/12 students strongly agreed while 3/12 students agreed while three students answered "don't know". This indicates that the majority of the students wanted to get all stars and thereby all badges.

**This is more fun than doing tasks by hand** We wanted to know if this form of learning was more fun, than their usual problem solving in math class. Because if this was not the case, then the program would have failed its purpose. The data shows that 8/12 students strongly agreed, 3/12 students agreed while 1/12 disagreed. This indicates a tendency among students to prefer the program over doing tasks by hand. It also indicated that the program has met some of the requirements we set to gamify the learning of formulas and algebraic rules.

**I want to use the program again** Lastly we wanted to know if the students would use the program again. 10/12 would use the program again. We did not want the game to be something that the students would just forget, but we wanted them to want to play again on their own accord. The answers from the students indicate that with these students the program has been a success and could help them supplement their math classes in the future.

Unfortunately there were only 12 students in the class, so we can not conclude anything concrete from this questionnaire. The ideal situation would be to test the program in multiple classes at different schools, because the time students spend learning the different things in the curriculum (see section 3.2.3), could vary from school to school.

We can see that this small number of students find it motivating to get stars and badges, and think that using the program is more fun than doing the mathematical tasks by hand and that the majority want to play the game again.

## Timetable

The amount of time the levels in the tutorial took, was varying quite a bit, but this was expected, since the tutorial levels were meant to teach the students how to use the program. The other categories and their time per level was also varying, with the parenthesis levels taking the longest time. This could be because the students had to get to know the program and try to remember the things they went through in the tutorial.

Figure 18 shows that the level with the expression  $3 * x - (4 * x - 5)$  took, on average, almost six minutes for the students to solve, which is a long time to use on one task compared to the time used on the others. We had also expected the 'Master of Algebra' category to be the most difficult, but this did not seem to be the case, looking at the times it took the students to solve it, this could be because when the students reached this point they had learned how to use the program and knew how the rules worked.

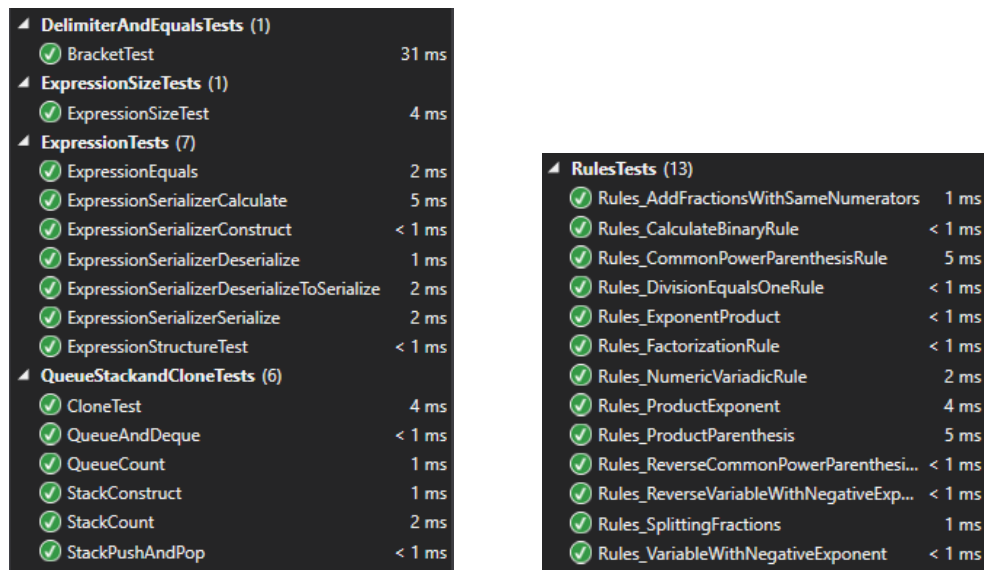
Looking at how long it takes for students to solve certain levels, helps us find levels that take an unnecessarily long amount of time to solve, and improve them. It also made us realise that the 'Master of Algebra' category which was meant to be the most difficult, may not be difficult enough. It is essential to mention that time spent on solving a task is not a direct way of measuring the task's difficulty.

## 9 Testing

This section describes how we used unit testing to ensure that the code kept working as intended when changes were committed.

### Unit Testing

To help facilitate unit testing, we make use of the tools available to us through Visual Studio. First we set up a range of tests based on our requirements. This covers features and supposed problems we can come up with. This resembles TDD as explained in section 2.4, but less structured, that is, we did not use TDD exclusively. From there, more tests are added as bugs show up in the code, each test addressing the particular bug found. This prevents the bug from returning in the future due to changes. This can be seen in our ExpressionSerializerCalculate test in ExpressionTests.cs. Most of these tests were made to make sure parsing was done correctly and that it would produce the correct tree that calculates to the correct value. We experienced various issues especially with regards to the function and power parsing, where they would get evaluated in an incorrect order. The tests helped us fix the issues and made sure that a similar issue would not return later. These tests are not exhaustive, but represent the issues we ran into and the situations we thought relevant to test.



(a) *DelimiterAndEqualsTests*, *ExpressionSizeTests*, *ExpressionTests* and *QueueStackandCloneTests*

(b) *RulesTests*

Figure 19: All unit tests being run in Visual Studio.

In figure 19a and 19b it is shown how all our 22 tests succeeded in the end of the project. Each unit test class has its own drop-down option, where its tests and the name of the tests are displayed. Here is either

a green checkmark or a red cross showing whether the test completed successfully or not. Furthermore, the time it took to complete the test is shown.

The most important tests of our program were RulesTests and ExpressionSizeTests. RulesTests tested most of our rules in different situations we could come up with, to make sure they worked. ExpressionSizeTests tested if the way we chose to measure the arithmetic complexity of an expression, as explained in section 7.2.3, worked correctly. Because unit testing is a time consuming process and some of the parts like views are difficult to test we did not cover all parts of the code, these parts was instead tested by running the program and trying to provoke any errors manually.

## 10 Discussion

This section discusses to what extent this project answers the three questions in the problem statement in section 4, to be able to evaluate the overall question: *"How can gamification be used to make the learning of formulas and algebraic expressions more engaging for students?"*.

### 10.1 Gamification Without Reducing Time Spent on Learning

To maximize the time spent on learning while implementing gamification in math, we chose the second approach, described in section 5.1. We chose to keep the traditional mathematical representation because our target users are 7th to 9th graders that are already familiar with this representation. This also teaches the students to manipulate mathematical expressions in the traditional way which they are going to use throughout their education.

If we compare our program to the traditional way of reducing algebraic expressions by hand, then the advantage of our program is that the students do not need to spend time on writing the different steps of reducing expressions.

In our program the students are able to apply algebraic rules by selecting parts of an expression. Our goal was that this would help the students to understand the usage of algebraic rules as well as discovering rules they have not yet learned. We designed the first category as a tutorial where the user learns the controls of the program as well as using basic rules of algebra. Learning the controls of the program requires some time, but not all this time is wasted because there is a close connection between the controls of the program and the algebraic rules that can be applied.

Because the program implements motivational affordances such as a leaderboard and badges, there is a risk that the students spend time on looking at these or showing them to their friends instead of completing the levels. To minimize the time spent on looking at these elements we only show leaderboards and their badges at the main menu to motivate the students before they begin to complete levels. To maximize time spent on learning we only implement the the progress bar for motivation when the students reduce expressions in the level menu. The advantage of displaying a progressbar is that it shows the user when an expression gets reduced; the disadvantage is that we could not find an objective definition of when one expression is more reduced than another. Therefore, we have implemented a value called Size which is our evaluation the arithmetic complexity of an expression, as described in section 7.2.3.



## 10.2 Gamifying the Learning of Formulas and Algebraic Rules

In an effort of making the learning of formulas and algebraic rules more engaging for students from 7th to 9th grade, we gamified our program using the motivational affordances described in section 5.2. To investigate if students actually felt motivated by using our program, we collected data from a school as described in section 8. The data we collected from the school indicated that some students felt more engaged and wanted to achieve the maximum amount of stars and badges in the game and that it was more fun using our program than doing tasks by hand. However, we must also keep in mind that only a small amount of data was collected, and that the program was only tested in one 7th grade class. We can therefore not be sure that other students would feel motivated in the same way. In order to check if we succeeded in gamifying the learning of formulas and algebraic rules it is therefore necessary to collect a bigger data set to see if more game elements should be added to the program. Furthermore, more badges and levels need to be designed to check if the program manages to keep students engaged over a longer time-span, or if the game only manages to keep the students motivated in a short amount of time. Another aspect we must keep in mind, is that there is a balance between making the program into a game and ensuring that the students learn something. Since we want to gamify the learning of formulas and algebraic rules we must therefore test if the students have become better at reducing algebraic expressions after using our program. The result of these tests should indicate whether future development of this program should focus more on gamifying the program or other features.

## 10.3 Adjusting the Challenges to the Individual Student's Skill Level

We wanted to find a way so all students at some point could be challenged with tasks that fit their individual skill level. This is important to make them hit the flow of learning as mentioned in section 3.1. In our program we tried to achieve this by implementing the motivational affordance *levels*. We tried to implement levels so the difficulty of the levels increases progressively. With this design the students will easily solve tasks until they reach the level that fits their skill. The difficulty in this was to make a common level design for all students' skill levels. Assume student A is good at fractions but less good at exponents, and student B is good at exponents, but less good at fractions. How would a common level design then have an increasing difficulty for both student A and B? To solve this we designed the program so the students only need to reduce the expression until they achieve one star which unlocks the next level. This is done to prevent a student from being stuck at a specific level. With this design it allows some levels to have a bigger span of difficulty and thereby targeting different student's skill levels. Looking at the data presented in section 8, it shows that the time spent on each level does not follow an increasing pattern. Therefore, if we interpret time spent per level as an indicator of the level's difficulty, the test shows that the level design needs to be improved or at least reordered. This cannot be fully concluded as time per level is not directly correlated to the level's difficulty. Time per level depends on other factors such as number of operations needed to solve the level as well as the difficulty of controls of the game.

## 11 Conclusion

In this project we have developed a program that tries to solve the overall problem statement: "*How can gamification be used to make the learning of reducing algebraic expressions by hand more engaging for students from 7th to 9th grade?*". The program implements the motivational affordances: *levels, point system, progress, achievements, feedback and leaderboard*. The program contains five categories with nine levels in each category. The goal of each level is to reduce a given expression by selecting sub-expressions that activate different algebraic rules. To help motivating the students we implemented a progress bar. The progress bar serves as both *progress* and *feedback*. The current progress shows how much the students have reduced the expression. Progress also controls our *point system* in form of stars which are obtained when a given progress has been reached. When the students have obtained one out of three stars the next level is unlocked. This enables the students to complete a level in different difficulties and thereby each level targets students with different skill levels. To motivate the students in completing all levels with three stars we use *achievements* in form of badges that are earned when all three stars are obtained in all levels within a category. Number of achieved badges are used to control our *leaderboard* that shows the top seven students with highest amount of badges.

To develop the program we used the object-oriented programming language C#. This enabled us to develop the classes GameModel, GameView and Game to follow a design pattern somewhere similar to the model-view-controller pattern as explained in section 7.2.1. To target different devices we used interfaces to implement platform specific code for rendering the GameView and communicating with the database through our API. From this we have developed classes that implement these interfaces for Web (using Bridge.NET), desktop (using MonoGame) and Android (using Xamarin).

To test the usability of our program, we contacted a Danish primary school where we got in contact with a 7th grade. We arranged a date to meet the class and had them try out the program. Although the class only consisted of 12 students, and thereby made our test unrepresentative, it gave us an overview of troubles and flaws of the program, and presented us with certain tendencies amongst the students. For instance, we noticed that multiple students clicked seemingly randomly in order to finish levels, and some did not even notice, when they were done with a level. Using these observations, we could improve the program, or write down what could be improved in future work in section 12. Furthermore, we made the class fill in a questionnaire. From this questionnaire we observed that 11/12 of the students found the program more motivating than doing tasks by hand, and 10/12 were motivated to try out the program again. Additionally, the majority of the students agreed that the motivational affordances had an effect on their motivation while using the program. 9/12 students wanted to get the maximum amount of stars in each level, and felt like it was motivating. 7/12 found that it was motivating to achieve badges. As mentioned earlier, our sample size was not large enough to give us a representative result, but what can be concluded from the observations of the small group who tried out our program, is that most students think that the program is more motivating than reducing expressions by hand.

## 12 Future Development

While testing our program, multiple ideas regarding improvements were discovered. These improvements were not considered worth the time they would take to implement, since we also had to focus on writing the report. Therefore, the ideas will be listed below as future work instead.

### The Program's Usability

To increase the program usability, the well-known controls for moving the focus to a different textbox could be implemented. These controls are e.g. the tab key to jump from username to password and arrows to move one letter back or forward when typing. When testing the program at the school we noticed that some students had trouble closing the tooltip box because they did not know where to click. This can be fixed by automatically closing the tooltip box if the user clicks anywhere on the screen when the tooltip box is open.

### Rules

While analysing the data from the testing, we noticed that some of the levels were unintentionally difficult to solve, one of these was a level, where one had to do a problem involving a minus parenthesis. The way the program handles a minus parenthesis is not consistent with the way the students are taught to do them by hand. Therefore, we could implement a rule in our program to handle this type of problems, and make them more intuitive to do.

Another rule that would make expression handling easier, would be to implement a rule that allows the user to remove equal terms in the numerator and denominator in a fraction like  $\frac{a*b}{a} = b$  instead of forcing the user to work with powers which is currently the case.

### Overview of the Program

While playing the game, the student can only see which category he/she is in, if they go back to the main menu and then go to the level select menu. After completing a category, a screen displays which category has been completed and congratulates the user. Therefore, there is not any logical way for the students to know which category they are currently playing. This problem could be solved by making a small label in the top of the program displaying which category the user is currently solving problems in like in our sketch of the idea in figure 4. This box could also be used to show how far a user is in a specific category, e.g. tutorial, level 3/9.

### Errors

While testing our program, we noticed that when player A finished the game, and got all badges, he/she went to the top of the leaderboard. But when player B also finished the game, later than player A, player B went above player A on the leaderboard. Future work could be to fix the leaderboard so it takes time into account when ranking players in the leaderboard.

## Target a Wider Audience

By implementing more levels, and more ways to manipulate with mathematical notation, the game could target a wider audience of different ages. This could e.g. be by using diamonds, hearts and cakes instead of variables to hit a younger target group, or adding more features such as logarithm, other function operators or matrices to hit an older target group such as high school students or even university students.

---

## 13 Bibliography

- [1] A317b GitHub repository, GitHub, seen 23-05-2016.  
[https://github.com/anton-christensen/p2\\_software](https://github.com/anton-christensen/p2_software)
- [2] Survey of Primary School Students and their well-being, UVM, seen 23-05-2016.  
[http://www.uvm.dk/-/media/UVM/Filer/Udd/Folke/PDF15/Juni/150617-Notat-Resultater-fra-trivselsmaalingen-med-tabeller-2015\\_FINAL.ashx?la=da](http://www.uvm.dk/-/media/UVM/Filer/Udd/Folke/PDF15/Juni/150617-Notat-Resultater-fra-trivselsmaalingen-med-tabeller-2015_FINAL.ashx?la=da)
- [3] The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. Karl M. Kapp.  
ISBN: 9781118096345
- [4] Primary school 7th to 9th grade curriculum math, EMU, seen 23-05-2016.  
<http://www.emu.dk/omraade/gsk-1%C3%A6rer/ffm/matematik/7-9-klasse/tal-og-algebra>
- [5] Rules of Play: Game Design Fundamentals, Katie Salen Tekinbas and Eric Zimmerman. Chapter 7 page 11 and chapter 20 page 1-2  
ISBN 0-262-24045-9
- [6] Zhang, Ping, Technical opinion Motivational affordances, Communications of the ACM.  
ISSN: 0001-0782.
- [7] Examples of gamification, Marta, February 28 2014, Seen 23-05-2016.  
<https://www.userlike.com/en/blog/2014/02/28/6-inspiring-examples-of-gamification>
- [8] Does Gamification Work? - A Literature Review of Empirical Studies on Gamification, IEEE Xplore, seen 23-05-2016.  
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6758978>
- [9] A Taxonomy of Motivational Affordances for Meaningful Gamified and Persuasive Technologies, Paul Weiser et al., Pages 5-6, seen 23-05-2016.  
<http://www.goeco-project.ch/wp-content/uploads/downloads/TaxonomyMotAffordances.pdf>
- [10] Test Driven Development: By Example, Kent Beck.  
ISBN: 978-0-321-14653-3.
- [11] Description of the term 'code smell', Martin Fowler, seen 23-05-2016.  
<http://martinfowler.com/bliki/CodeSmell.html>
- [12] "Kan man kede sig ihjel?" - An article about boredom, Videnskab DK, seen 23-05-2016.  
<http://videnskab.dk/sporg-videnskaben/kan-man-kede-sig-ihjel>
- [13] Applications of Flow in Human Development and Education, Mihaly Csikszentmihalyi.  
ISBN 978-94-017-9094-9
- [14] Flow: The Psychology of Optimal Experience, Mihaly Csikszentmihalyi.  
ISBN: 9780061339202

- 
- [15] Page 126, Matematik i Læreruddannelsen.  
ISBN: 87-00-36206-9
- [16] Competent Danish students are held down in school, di.dk, seen 23-05-2016.  
<http://di.dk/SiteCollectionDocuments/Opinion/Uddannelse/Unders%C3%B8gelse%20blandt%20skoleelever.pdf>
- [17] About EMU, EMU, seen 23-05-2016.  
<http://www.emu.dk/modul/about-emu>
- [18] Different teaching methods in the danish primary school, EMU, seen 23-05-2016.  
<http://www.emu.dk/modul/forskellige-undervisningsformer-i-skoledagen>
- [19] Different programs used in danish primary school, folkeskolen.dk, seen 23-05-2016.  
<https://www.folkeskolen.dk/517653/laererne-anbefaler-her-er-programmerne-som-loeften-vores-undervisning>
- [20] More focus on IT in the Danish primary school, UVM, seen 23-05-2016.  
<http://www.uvm.dk/Uddannelser/Folkeskolen/Laering-og-laeringsmiljoe/It-i-undervisningen/Oeget-anvendelse-af-it-i-folkeskolen>
- [21] Digital tools in math lectures, EMU, seen 23-05-2016.  
<http://www.emu.dk/modul/digitale-v%C3%A6rkt%C3%B8jer-i-matematik>
- [22] Formelsamling folkeskolens afsluttende prøver i matematik, Hans Jørgen Beck, Thomas Kaas and Klaus Fink,  
ISBN (WWW): 978-87-92140-60-9
- [23] Description of MatematikFessor, EduLab, seen 23-05-2016.  
<https://edulab.dk/matematikfessor-dk/>
- [24] MatematikFessor.dk, MatematikFessor, seen 23-05-2016.  
<https://www.matematikfessor.dk/>
- [25] Description of MatematikFessor, Clio Online, seen 23-05-2016.  
<http://www.clioonline.dk/om-os/samarbejdspartnere/matematikfessor/>
- [26] Description of energy points at Khan Academy, Khan Academy, seen 23-05-2016.  
<https://khanacademy.zendesk.com/hc/en-us/articles/202487710-What-are-energy-points->
- [27] Quiz game, Kahoot, seen 23-05-2016.  
<https://getkahoot.com/>
- [28] WeWantToKnow, Dragonbox, seen 23-05-2016.  
<http://dragonbox.com/>
- [29] Unity3D homepage, Unity, seen 23-05-2016.  
<https://unity3d.com/>
- [30] Advantages and disadvantages of Unity3D, Unity, seen 23-05-2016.  
<http://logicsimplified.com/games/2015/09/22/unity-3d-game-development-advantages-and-disadvantages/>

- 
- [31] Asset store for Unity3D, Unity, seen 23-05-2016.  
<https://www.assetstore.unity3d.com/en/>
- [32] Bridge.NET homepage, Bridge, seen 23-05-2016.  
<http://bridge.net/>
- [33] Bridge.NET forums, Bridge, seen 23-05-2016.  
<http://forums.bridge.net/>
- [34] Unity3d download page for webplayer, note: it is not available for chrome Unity3d, seen 23-05-2016.  
<http://unity3d.com/webplayer>
- [35] W3schools.com page on browser usage statistics, W3Schools, seen 23-05-2016.  
[http://www.w3schools.com/browsers/browsers\\_stats.asp](http://www.w3schools.com/browsers/browsers_stats.asp)
- [36] Model-View-Controller and Object Teams: A Perfect Match of Paradigms seen 23-05-2016.  
[http://delivery.acm.org/10.1145/650000/643618/p140-veit.pdf?ip=130.225.198.198&id=643618&acc=ACTIVE%20SERVICE&key=36332CD97FA87885%2E1DDFD8390336D738%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=619261473&CFTOKEN=76081714&\\_\\_acm\\_\\_=1464018980\\_18eaae5af3f93af7d495d9b053773007](http://delivery.acm.org/10.1145/650000/643618/p140-veit.pdf?ip=130.225.198.198&id=643618&acc=ACTIVE%20SERVICE&key=36332CD97FA87885%2E1DDFD8390336D738%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=619261473&CFTOKEN=76081714&__acm__=1464018980_18eaae5af3f93af7d495d9b053773007)
- [37] From infix expression to postfix, Wikipedia, seen 23-05-2016.  
[https://en.wikipedia.org/wiki/Shunting-yard\\_algorithm](https://en.wikipedia.org/wiki/Shunting-yard_algorithm)
- [38] Hours of education, UVM, seen 23-05-2016.  
<https://uvm.dk/Uddannelser/Folkeskolen/Fag-timetal-og-overgange/Undervisningens-samlede-laengde/~media/997F8A9AEAE64A77AC58B4312BB7DC91.ashx>


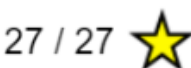
## 14 Appendix

### A Questionnaire

**Brugernavn:**

**Kode:**

Baseret på dit brug af programmet i dag, hvor enig er du i følgende udsagn? (sæt kryds)

	Meget uenig	Lidt uenig	Ved ikke	Lidt enig	Meget enig
Det var nemt at benytte programmet.					
Det var nemt at vælge tal og bogstaver i programmet.					
Det var nemt at forstå formålet med opgaverne.					
Det var motiverende at få stjerner for at lave en opgave i programmet					
Det var motiverende at kunne opnå badges: 					
Jeg fik lyst til at få det maksimale antal stjerner i hver opgave i programmet. 					
Det er sjovere at regne opgaver i dette program, end i hånden.					
Jeg har lyst til at benytte programmet igen.					

Har du nogle idéer / forslag til forbedringer af programmet?

Tak for din deltagelse!



## B Infix to Postfix Algorithm.

**Data:** infix expression string

**Result:** postfix token list

```

while there are tokens to be read do
    read a token;
    if the token is a number OR a constant OR a variable then
        enqueue the token to output;
    else if the token is a function then
        push the token to stack;
    else if the token is an operator AND the operator is unaryminus then
        enqueue the token to output;
    else if the token, o1, is an operator AND the operator is NOT unaryminus then
        enqueue token to output;
        while there are tokens in stack do
            if the token, o2, at the top of stack is a mathematical operator then
                if o1 is left-associative and its precedence is less than or equal to that of o2 OR o1
                    is right-associative, and has precedence less than that of o2 then
                        pop o2 from stack and enqueue it to output;
                    end
                end
            end
            push o1 to stack;
        else if the token is a left parenthesis then
            push the token to stack;
        else if the token is a right parenthesis then
            while there are tokens in stack do
                if the token at the top of stack is a left parenthesis then
                    break;
                else
                    enqueue token to output;
                end
            end
        else
            signal an error has occurred;
        end
    end
while there are tokens in output do
    add token to result;
end
while there are tokens in stack do
    add token to result;
end

```

**Algorithm 1:** Infix expression to postfix expression

## C Calculations of Amount of Bored Students

The following calculation is done by using numbers from question 31: *Are classes boring?* which can be found on [2] at page 27/29. The numbers used are from the age group we have chosen to focus on which is the 7th to 9th grade.

First the total amount of students in the specific grades are calculated:

$$\text{seventh} = 677 + 6118 + 23754 + 8956 + 4729 = 44234$$

$$\text{eighth} = 503 + 5160 + 23091 + 9247 + 4467 = 42468$$

$$\text{ninth} = 23 + 3997 + 19494 + 8380 + 4124 = 36418$$

$$\text{total} = \text{seventh} + \text{eighth} + \text{ninth} = 44234 + 42468 + 36418 = 123120$$

Then the amount of students in the three grades who answered that they find classes boring often or very often is calculated:

$$\text{bored} = 8956 + 4729 + 9247 + 4467 + 8380 + 4124 = 39903$$

At last the percentage of students who find classes boring is calculated:

$$\text{percentage} = (\text{bored}/\text{total}) = (39903/123120) \sim 32,41\%$$

## D Number of Teaching Lessons

Timetal (minimumstimetal og vejledende timetal) for fagene i folkeskolen											
Klassetrin	Bh.	1.	2.	3.	4.	5.	6.	7.	8.	9.	Timetal i alt
<b>Humanistiske fag</b>											
Dansk (minimumstimetal)		330	300	270	210	210	210	210	210	210	2.160
Engelsk (vejledende timetal)		30	30	60	60	90	90	90	90	90	630
Tysk eller fransk (vejledende timetal)						30	60	90	90	90	360
Historie (minimumstimetal)				30	60	60	60	60	60	30	360
Kristendomskundskab (vejledende timetal)		60	30	30	30	30	60		30	30	300
Samfundsfag (vejledende timetal)									60	60	120
<b>Naturfag</b>											
Matematik (minimumstimetal)		150	150	150	150	150	150	150	150	150	1.350
Natur/teknik (vejledende timetal)		30	60	60	90	60	60				360
Geografi (vejledende timetal)								60	30	30	120
Biologi (vejledende timetal)								60	60	30	150
Fysik/kemi (vejledende timetal)								60	60	90	210
<b>Praktiske/musiske fag</b>											
Idræt (vejledende timetal)		60	60	60	90	90	90	60	60	60	630
Musik (vejledende timetal)		60	60	60	60	60	30				330
Billedkunst (vejledende timetal)		30	60	60	60	30					240
Håndværk og design samt madkundskab (vejledende timetal)					90	120	120	60			390
<b>Valgfag</b>											
Valgfag (vejledende timetal)								60	60	60	180
<b>Årligt minimumstimetal</b>											
Årligt minimumstimetal pr. klassetrin	600	750	750	780	900	930	930	960	960	930	8.490 inkl. bh.
<b>Undervisningstidens samlede længde</b>											
Undervisningstidens samlede længde (inkl. pauser, understøttende undervisning mv.)	1.200	1.200	1.200	1.200	1.320	1.320	1.320	1.400	1.400	1.400	12.960 inkl. bh.

Note: Timetallene er angivet i klokketimer og uden pauser. Dette gælder dog ikke for sidste række, der dækker undervisningstidens samlede længde i klokketimer, inkl. pauser, tid til understøttende undervisning mv.

Note: Bh. : Børnehaveklasse.

Figure 20: Hours of education in danish primary school [38].

**E Table of Raw Time Data Extracted from School Test**

Level	1	2	3	4	5	6	7	8	9	10	11	Average
Tutorial 1			246	169								207.5
Tutorial 2	10	19			8	319	19	17	10	4	9	46.11
Tutorial 3	6	11	5	1	11	12	12	5	12	5	11	8.27
Tutorial 4	7	6	6	12	11	121	13	6	13	4	17	20.40
Tutorial 5	67	791	265	84	258	263	110	55	457	34	272	241.45
Tutorial 6	8	1	12	75	23	17	16	8	14	12	24	19.09
Tutorial 7	32	279	73	143	138	381	91	35	51	253	389	169.55
Tutorial 8	10	16	10	14	15	28	127	7	8	33	11	25.36
Tutorial 9	7	98	323	17	122	16	40	36	181	8	48	81.45
Parenteser 1	15	23	300	693	26	21	28	67	29	45	99	122.36
Parenteser 2	33	29	16	71	12	47	42	33	31	9	307	57.27
Parenteser 3	14	16	2	36	22	44	13	31	22	15	1	19.64
Parenteser 4	29	70	88	31	20	3	17	17	90	12	2	34.45
Parenteser 5	135	119	72	1051	68	239	555	238	46	49	34	236.91
Parenteser 6	170	195	131	81	115	413	77	191	507	129	189	199.82
Parenteser 7	127	241	69	100	64	1164	68	290	181	108	149	232.82
Parenteser 8	124	107	111	357	65	7	93	212	144	158		137.80
Parenteser 9	557	261	93		370		415	523	236	317		346.50
Potenser 1	6	8	10		7		11	6	7	10		8.13
Potenser 2	6	4	6		18		4	3	4	2		5.88
Potenser 3	6	4	7		18		10	4	11	8		8.50
Potenser 4	19	7	6		9		5	8	9	13		9.50
Potenser 5	20	19	23		48		26	15	60	8		27.38
Potenser 6	34	110	46		104		67	165	144	38		88.50
Potenser 7	5	6	19		11		5	4	10	9		8.63
Potenser 8	14	51	51		37		26	15	18	17		28.63
Potenser 9	21	152	72		643		487	92	139	76		210.25

Table 7: Shows the estimated amount of time each user spent on every level, measured in seconds - Continued on next page.

Level	1	2	3	4	5	6	7	8	9	10	11	Average
Brøker 1	88	140	73		192		437	134	141	79		160.50
Brøker 2	8	5	7		7		6	4	5	4		5.75
Brøker 3	9	5	11		37		7	22	8	25		15.50
Brøker 4	23	15	13		30		19	30	84	30		30.50
Brøker 5	6	8	5		8		5	5	6			6.14
Brøker 6	36	41	49		300		23	80	32	7		71.00
Brøker 7	37	28	22		231		25	52	31	22		56.00
Brøker 8	74	88	63				164	146	272	65		124.57
Brøker 9	177	88	112				178	71	131	42		114.14
Master of Algebra 1	42	31	81				19	44	45	4		38.00
Master of Algebra 2	44	82	31				66	46	26	13		44.00
Master of Algebra 3	83	75	177				28	50	68	86		81.00
Master of Algebra 4	92	53	494				72	95	120	47		139.00
Master of Algebra 5	53	235	118				27	71	63	20		83.86
Master of Algebra 6	56	86	36				175	100	36	252		105.86
Master of Algebra 7	22	30	30				30	40	31	24		29.57
Master of Algebra 8	93	54	23				132	65	29	24		60.00
Master of Algebra 9	82	77	345				101	85	77	99		123.71

Table 8: Shows the estimated amount of time each user spent on every level, measured in seconds.

### F Questionnaire Data with ID Tags

Questions	1	2	3	4	5	6	7	8	9	10	11*	11*
It was easy to use the program	A	SA	SA	SA	D	D	A	A	SA	SA	D	SA
It was easy to select numbers and variables in the program	A	SA	SA	SA	DK	A	SA	SA	SA	SA	D	SA
It was easy to understand the purpose of the tasks	SA	SA	A	SA	DK	SD	A	A	A	SA	A	SA
It was motivating to get stars	SA	SA	SA	SA	A	SA	SA	DK	SA	DK	DK	A
It was motivating to achieve badges	DK	SA	SA	SA	DK	A	SA	SA	SA	DK	DK	DK
I wanted to get the maximum number of stars	A	SA	SA	SA	DK	SA	SA	A	A	DK	SA	DK
This is more fun than doing tasks by hand	A	SA	SA	SA	D	A	A	SA	SA	SA	SA	SA
I want to use the program again	SA	SA	SA	SA	DK	DK	A	A	SA	SA	A	SA
* The ID tag is identical because two students used the same computer												
SD = Strongly disagree, D = Disagree, DK = Dont know, A = agree, SA = Strongly agree												
Do you have any ideas or suggestions to improve the program?												
1 = Not really												
6 = It would be easier if there was a voice to explain it												
7 = More levels												
8 = Find some background music												
9 = Not :)												
11* = Nope												

Figure 21: Shows the questionnaire data with corresponding ID tags.