UNIVERSITY OF TARTU

Institute of Computer Science

Software Engineering Curriculum

Jaan Janno

Introductory Game Development and Programming Course Materials

Master's Thesis (30 ECTS)

Supervisor: Raimond-Hendrik Tunnel, MSc.

Co-supervisor: Ljubov Jaanuska, MSc.

Tartu 2017

Sissejuhatav mänguarenduse ja programmeerimise kursuse õppematerjal

Lühikokkuvõte: "Teeme ise arvutimänge" on e-kursus Tartu Ülikoolis. Antud kursusel on kaks peamist eesmärki. Esimene eesmärk on programmeerimisega tutvumine õpilastele, kellel on huvi jätkata oma õpinguid kutsekoolis või ülikoolis programmeerimisega seotud õppekavadel. Teine eesmärk on gümnasistidele programmeerimise ning mänguarenduse õpetamine. Kursus on peamiselt mõeldud gümnaasiumiealistele õpilastele, kuid on sobilik ka vanematele ja edasijõudnud noorematele osavõtjatele.

Käesolev töö annab esmalt detailse ülevaate esialgsest kursusest. Tuvastatakse kursuse probleemid. Seejärel kirjeldatakse lahendusi ning lisasid kursusele. Suur osa tööst on uued videomaterjalid. Kursusel juurutati uusi keerukamaid ülesandeid ning tekstilisi materjale, et tutuvustada graafikat varasemalt. Selle eesmärk on kursuse varasema osa põnevamaks muutmine ning selliste õpilastega tegelemine, kes tulid kursusele peamiselt mänguarendust õppima. Töö käigus lisati kursusele uuendused ning kursust katsetati uute õpilaste peal. Viimaks arutletakse uuendatud kuruse kvaliteedi üle.

Võtmesõnad: Programmeerimine, arvutimänguarendus, õppematerjal, arvutigraafika, online kursus, e-kursus, Python, Thonny, Pygame, e-õpe.

CERCS:

P170 – Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

S281 – Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Introductory Game Development and Programming Course Materials

Abstract:

The course "Teeme ise arvutimänge" (translated "Let's Make Computer Games"), is an online course conducted in the University of Tartu. The course has two main aims. One is to teach programming and game development skills to secondary school pupils. The other goal is engaging pupils, who might be interested in continuing to learn in programming related vocational or university study programs. The course is foremost meant to suit pupils in secondary school age, but it is also suitable for older and more advanced younger pupils.

The thesis provides a thorough analysis of the course materials as they were before the work of this thesis. Problems are identified regarding the course. Then the thesis describes the solutions to the found problems and general additions to the course. A large part of the work is new video materials, which are described in detail. Several new complex tasks and textual materials were introduced to the course with the intent of introducing graphics early. The goal of introducing graphics early is to engage pupils, who came mainly for the game development part of the course, sooner and to make the first part of the course more exciting in general. During the work of the thesis, the additions and revisions were implemented in the course and tested on new pupils. Finally, the quality of the revised course is discussed.

Keywords: Programming, computer game development, course material, computer graphics, online course, e-course, Python, Thonny, Pygame, online learning.

CERCS:

P170 – Computer science, numerical analysis, systems, control

S281 – Computer-assisted education

Contents

1 Introduction

Online courses are becoming increasingly more popular. In Estonia, the number of online courses has risen from 14 in 1999 to several thousands today and the number of participants has been rising as well [DT].

The course "Teeme ise arvutimänge" (translated "Let's Make Computer Games") is an introductory online programming and game development course conducted in the University of Tartu. It was assembled in 2012 by Tiina Kull and has been successfully conducted since then with some organic changes happening during the years. The course is relevant, because game development is a good gateway to programming for pupils who could one day study programming related fields in the future. Several of the students, who have taken part in the course, have later started studying Computer Science in the University of Tartu, for example.

However, the course is showing signs of aging. This includes old versions of programming tools, libraries and programming language. The material of the course needs to be updated as well. This thesis covers the modernization of the course.

The first section, "The course", gives an overview of the course as it was before the work of the thesis. The target group, organization and the content of the materials and tasks are covered.

The second section discusses the problems in the course material. This includes description of the state of the technology used in the course, the content of materials and the grading system.

The third section covers the work done on the course as part of this thesis. This consists of the changes made in the technology, new tasks, changes in the textual materials and recording of new videos that support the material.

Then the next section discusses the results of the thesis. The quality of the new course content is measured by questionnaires, which the students have filled

weekly. The section concludes with some ideas about future developments in the course.

2 The course

"Let's make computer games" is an introductory programming and game development course. The course in its initial form was assembled by Tiina Kull in 2012 [Kul12a]. It has been conducted in the University of Tartu from 2012 onwards. The course has two main goals. Its one aim is to provide basic skills and knowledge in game development and computer programming in an accessible way for beginners with basic computer operating skills. The second goal is to create interest in programming in secondary school students. This aids students who might potentially decide to continue their studies in the University of Tartu to make an educated decision about whether the computer science curriculum would suit them.

This section describes the course structure before the changes had been made to it in the scope of this thesis. This includes changes made to the initial 2012 version before 2016 by other course instructors.

2.1 Target group

The target group of the course is secondary school pupils in Estonia. Most of whom have none or only basic former experience with computer programming. They should have skills in everyday computer use. However, the course is also suitable to more advanced primary school students, students in tertiary education and non students. Additionally the target groups should have an interest in mathematics.

2.2 Course organization

The course is conducted in an online environment – Moodle. This system is used for hosting and providing most of the materials, exercises, tests and organization information of the "Let's make computer games!" course. A large of the com-

munication also happens on the platform as forum posts and private messages. This includes communication between the students and instructors as well as peer feedback regarding various exercises. The students are, however, also able to communicate with the instructors over e-mail. The video are embedded in the Moodle textual materials, but hosted on YouTube.

The duration of the course is 8 weeks, of which the final work is presented by the students by the end of the 7th week. The last week is for peer feedback on final projects, grading and finishing remarks.

There are weekly mandatory exercises in the first 5 weeks of the course. These include 5 programming tasks for the students to solve each week. The first 3 weeks also contain multiple-choice tests, which measure the pupil's factual knowledge of the textbook material.

The 6th and 7th week are for finalizing the course project. These projects are assembled by teams of 2 to 5 students, however working alone is also permitted. Originally the project was designed to be individual for each student. It was later changed to be in teams to promote teamwork and to let students divide tasks in a way that each member would get a part suitable and interesting to himself.

A number of tasks in the course are such, where the results are submitted publicly. The purpose of this is to create more transparency in the course, meaning the pupils are aware of each others work and presence in the course. This helps create an environment, where the participants of the course are encouraged to become partners in the project phase of the course [DP].

2.2.1 Programming toolset and technologies

The course teaches programming based on the Python programming language. It is an interpreted high-level programming language.

It makes use of a syntax that makes expressing concepts more compact – the

language requires fewer lines of code when compared to widespread languages such as C++ and Java. The language is well suited for writing smaller programs. These characteristics make it well suited for teaching programming to beginners [She].

The Python language is also used in the introductory programming course for computer science students in the University of Tartu [Ül]. This makes it beneficial to also use the language in the "Let's make computer games" course, as one of the course's goals is to cater to future students in the University of Tartu.

Another important merit of the language is the portability of programming tools. Students with a wide variety of different operating systems are able to follow the course without a lot of differences in usability. Usage of operating systems alternative to Windows has been rising [W3S]. Therefore to include the students with different operating systems in the course, the tools have to be portable.

The Python programming language by default also includes an IDE in the install package. This is the IDLE IDE. It features a shell for quick writing and running of shorter code snippets and a text editor for writing larger scripts. It also features a basic debugger, as shown in the figure 1.

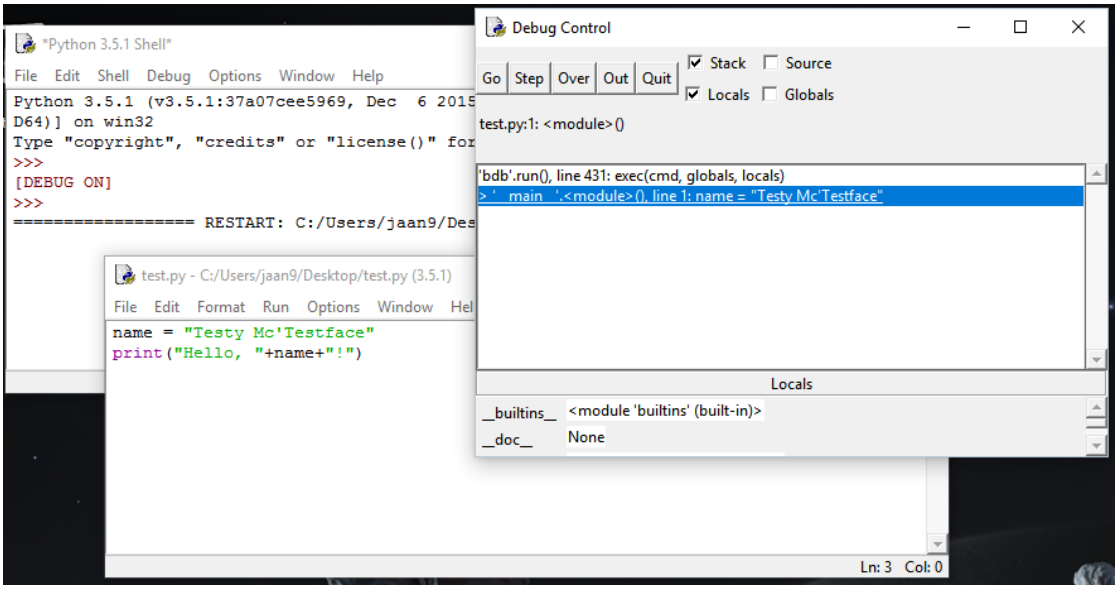

Figure 1: The IDLE IDE featuring the shell, text editor and debugger.

As the course contains topics regarding game development, an appropriate library is used in the course. The library used is the course is Pygame. This is an extra module for Python, that needs to be installed separately. It is used in the course for computer graphics, window management, user input, event handling and sound.

The course materials are built upon an older version of Python, which is instructed to be downloaded in the beginning of the course. This was the version 3.2.2, released in 4th of Septmenber 2011. This was justified with the fact that officially Pygame only offered compatibility with this previously mentioned version. At the moment of writing this, the newest version of Python is 3.5 with the latest stable release in 21th of March 2017.

2.2.2 Moodle

Moodle, acronym for Modular object-oriented dynamic learning environment, is a learning management system. It is an open-source software for conducting e-learning. It supports a wide variety of options for creating and managing an online course.

In the context of the "Let's make computer games!" course, the following functionalities are important:

- Forum system and private messages for communication.
- System for creating and managing multiple-choice question tests.
- System for creating and managing exercises.
- System for adding a textbook with custom HTML elements such as embedded videos.
- Student feedback forms.
- A grading system.

As Moodle offers all of those, it is a good fit.

2.3 Week one - Introduction

The first week's aim is to make a basic introduction into the course for the students. This means explaining the structure of the course a what they will learn during it. This section gives a brief overview of the contents of the material in [Kul12a]. The learning outcome of this week is the following knowledge and skills:

- Installing Python.
- Running the IDLE IDE.

- Using the shell.
- Using maths and math expressions in Python.
- Using IDLE text editor for writing programs.
- Running Python programs.
- Recognizing the most common error messages.
- Utilizing variables.
- Using different data types – numeric types and string.
- Difference between ints and floats.
- Type conversion between ints floats and string.
- The e number notation in Python.
- Determining the type of a variable.
- Asking for input and doing output.
- Getting input from files and from the web.

2.3.1 Textbook

The textbook for the first week first starts by explaining how to navigate in the materials. It also proposes strategies for the student to get maximal value out of the course. An explanation is given about various visual aids used in the material for emphasizing important concepts.

The the first real tutorial in the textbook explains how to install the necessary tools for the course. This includes how to download and install Python. The

tutorial proposed that version 3.2.2 should be downloaded to be compatible with the latest Pygame version at the time.

The lesson continues with a short demonstration of the shell functionality in the IDLE IDE. The standard "Hello, world!" example is given as well as some simple samples with printing out numbers and different text. An example is also given about arithmetic operation with numbers and strings. Also the behavior of operators applied to a mix of numbers and strings is demonstrated – namely that strings can be "multiplied" in python, however addition and subtraction produce errors.

Next the tutorial introduces the text editor in IDLE, which is now used for writing a somewhat larger code than was done in the shell as an example. This example program is based on the same basic concepts as the ones the shell example made use of.

The tutorial then moves on to discuss error messages in the Python language. It explains that making errors is a normal part of programming. Several most common error messages are explained to the pupils. A distinction is explained between syntax errors and runtime errors.

Then an example is shown about a game, where the player must guess a number the computer is thinking of. The pupil is supposed to write out the program and try to make sense of it in general without understanding the specifics, as most of the programming concepts in the game have not yet been explained this week.

The next discussed topic is input and output. The material explains the basic concept of it and ties it to previously seen *input* and *print* functions.

The concept of variable is also explained in the first week. First a shell example is given and then it is explained how a programmer should think of variables in Python. The nomenclature of variable is also discussed.

The material then discusses the differences of data types in Python. This

includes the types learned so far – the numeric type int and strings. It is explained how they are represented in the Python language.

Another major topic is mathematics. The main mathematical operators are listed for the students and examples are given.

Subtracting and adding to variables is discussed subsequently. Then it is shown how extremely large and small numbers are represented in Python. This means the e notation of numbers. Also the module concept is brought in – namely with the math module of Python.

The tutorial moves on to data types. The data types learned so far are repeated and it is shown how to transform values from one type to another. It is also shown how to find out of which type a variable or value is.

The final topic in the first week's explains reading in data from files and from the internet. Short examples are given on how to use the appropriate Python functions for this purpose.

2.3.2 Exercises and test

The first week has 5 exercises for the students to solve. The exercises can be summarized as follows:

1. A program, which asks the user some personal information and then prints it on the screen. The goal of this exercise is to support understanding of the *print* and *input* functions.

2. A program that given the circumference of earth, must compute how many people fit on its diameter. The goal is to support the understanding of the mathematical operators in Python.

3. A program, which downloads and prints ASCII art from a text file in the web. The goal is to further support the understanding of the *print* function

and manipulation of a file on the web.

4. A number guessing game. The goal is to slightly modify the existing number guessing game mentioned previously. The goal of this is to give the student and overview of the slightly more advanced Python language concepts without discussing the specifics.

5. Time until TV show program. The program must compute the time until a TV show specified by user begins given current time. The goal is to mix the concepts of input/output and the mathematical operators.

The first week also contains one test. The test checks factual understanding of the textbook materials. For the first week, this includes the basics about using variables, different types of values and the maths operators.

2.4 Week two - Decisions, branching and loops

This section covers the contents of the material in [Kul12b]. The learning outcomes of this week are the following skills:

- How to use conditions & decisions using the if-elif-else statements.
- How to check multiple conditions.
- How to use *for* loops and the *range* function.
- How to use *while* loop.

2.4.1 Textbook

The textbook begins by explaining the principle of decisions in Python. It demonstrates the use of the *if* statement and follows up by introducing the *elif* and *else* keywords.

The next topic covered is loops. First, *for* loop are explained. It is shown, how the *range* function can be used along with the loop. Second, the *while* loop is introduced. The loop topic is concluded by teaching how to use the *break* and *continue* keywords.

2.4.2 Exercises and test

The second week also has 5 exercises for the students to solve. The exercises can be summarized as follows:

1. The first two exercises are part of the same task. The goal is to create a game belonging to the "choose your own adventure" genre. The first part is to make a scheme of the decisions in the game. The second part is to create and upload the game itself publicly. The students are also asked to give peer feedback to fellow students. The goal is to support the learning of decisions in Python and supporting students interaction with each other.

2. Writing a program, which asks the user a password and must react differently to the correct and false passwords. The goal is to support learning of decisions in Python.

3. Program, that does a count down from a given number to zero, decrementing by one each second. The goal is to support the leaning of loops.

4. Program, that counts numbers divisible by three in a given interval. The goal is to support the leaning of loops.

The second week also contains one test. The test checks factual understanding of the textbook materials. This includes decisions with the *if-elif-else* statements and loops.

2.5 Week three - Nested loops and lists

This section covers the contents of the material in [Kul12c]. The learning outcomes of this week are the following skills:

- How to use nested loops
- How to use lists in Python. Including how to create a list, remove elements, change elements, find elements and how to sort the list.

2.5.1 Textbook

The textbook starts with an example of a Snake game made using Pygame The goal of which is to make pupils acquainted with it before teaching the specifics. Then nested loops are explained. The goal of this is to show that the content of the for loop can contain any arbitrary code, including inner *for* loops. It is also shows, how for loops can be used to traverse combinations and combinations of elements. The last topic is lists. The following is explained about lists:

- How to create a list.
- How to remove elements.
- How to change the elements.
- How to find the position of elements.
- How to sort lists.

One video demonstrates how to use the Turtle graphics library. It is a module for Python, that can be used to draw basic shapes.

2.5.2 Exercises and test

The third week also has 5 tasks. However, one of them is not a programming task, but feedback writing, which is not covered in this section. The programming tasks ask to create the following programs:

1. A program, that prints an empty rectangle of asterisk characters in the shell. The goal is to support the learning of loops.

2. A turtle program, that draws the shape of a chocolate bar. The goal is to give the pupils practical experience with graphics in python and to further support the learning of loops.

3. A program, that stores lists of basketball players heights. The program has to ask the user for the height of the basketball players. The code then has to contain various sorting, adding and deleting operations on the list. The goal is to support the learning of operations applicable to lists.

4. The last exercise is very open ended. The goal is to write a program where it makes sense to compute all combinations certain set's elements using *for* loops. Such as all combinations of food from a restaurant. The goal is to let students independently use *for* loops for a purpose of their own choosing.

The second week also contains one test. The test checks factual understanding of the textbook materials. This includes nested loops and operations with lists.

2.6 Week four - Functions and objects

This section covers the contents of the material in [Kul12d]. The learning outcomes of this week are the following skills:

- How to use functions. This includes definition of functions, adding arguments to the functions and the *return* keyword.
- How to use variables in different scopes. The difference of global and local variables.
- How to use OOP in Python. This includes definition of classes ,initialization of objects and method calls.

Unlike in the previous weeks, in week 4 and onwards there are no mandatory tests.

2.6.1 Textbook

The textbook starts with an example of Snake game, which makes use of functions. This gives a general overview of defining functions. After the Snake example, the concept of functions is described in more detail. The following is demonstrated and explained about functions:

- Definition of functions.
- Adding arguments to the functions
- The *return* keyword.
- How to use variables in different scopes. The difference of global and local variables.

The next and last topic of the week is OOP. The following skills are taught to the pupils:

- How to define a class.
- How to create an object and initialize its attributes.
- How to do method calls.

2.6.2 Exercises

The third week has 5 tasks as well. However, again one of them is not a programming task, but feedback writing, which is not covered in this section. The programming tasks ask to create the following programs:

1. A program containing a function definition, that converts miles to kilometers. The goal is to solidify skills of defining a function and returning the outcome of computation.

2. A program, that draws a set of triangles using Turtle. The goal is to show the usefulness of function definitions for a task that contains graphics.

3. The third is a very open ended task. The goal is to define any class and create objects of that class using ones own imagination. The goal is to solidify the skill of using OOP.

4. The last task is to choose a topic for the project in the end of the course. The goal is to let pupils have an early start with the projects. It is important to note, that this is done publicly. The students announce their chosen topics as forum posts.

2.7 Week five - Graphics

This section covers the contents of the material in [Kul12e]. The learning outcomes of this week are the following skills:

- How to create a window using the Pygame module.
- How to create graphics using Pygame.
- How to represent moving objects in Python.
- How to define RGB colors.

2.7.1 Textbook

The textbook of this week focuses exclusively on the Pygame library - a graphics module for Python. The material contains instructions of how to achieve the following by using Pygame:

- How to create a window using the Pygame module.
- How to draw different shapes and images on the window.
- How coordinates are represented in Pygame.
- How to define RGB colors.
- How to make the positions of shapes interdependent. Such as drawing one image to the edge of another.
- How to generate and draw text.
- How to create moving objects.
- How to process user input from keyboard and mouse devices.

2.7.2 Exercises

The programming tasks ask to create the following programs:

1. A Pygame program, that draws a house. The goal is to make the pupil acquainted with using various shape drawing functions in Pygame.
2. A program, that draws rectangles on its window. A base code is given, which draws bubbles. The student also need to comment all the lines in the code. The goal of the task is to solidify the understanding of Pygame and the goal of the comments is to let instructors see the thought process of students and to give according feedback.

3. A program, that draws and hides a picture according to user input. This means defining a button, which toggles the visibility of an image on the window. The goal is to let the pupils connect the concepts of Pygame and conditional statements.

4. A program, that contains a moving picture. The goal is to solidify the understanding of textbook examples regarding moving images.

5. The last task is to upload the current state of final projects. The goal is to have pupils receive feedback and advice regarding their projects based on the status of the project.

2.8 Week six - Bonus topics and sound

From week six onwards there are no weekly programming tasks and problems for the pupils to solve. Instead the week is for reading the textbook for advice on creating their projects and to start writing the project.

This section covers the contents of the material in [Kul12f]. This material is not mandatory for the pupils, but can be useful for some, whose game contains sound.

2.8.1 Textbook

The content of this textbook in essence an extra material in order to help the students with their course projects. The material contains information about the following:

- How to add and play sounds and music with Pygame.
- How to create a high score table.
- How to ask for a name input graphically.

- An example of a worm game which makes use of the previously mentioned points.

2.9 Week seven - Project

This week is the deadline for the students to present their projects. This is done in a separate forum in the Moodle page. The forums allows students to publicly upload their project files and to write a short intro message. The pupils can write peer feedback to each others submissions.

This week is also used to collect feedback from students about the overall performance of the course.

2.10 Week eight

During week eight the only obligation the students have is to present feedback to other students about their projects. Once this is done, the pupils who successfully completed the course receive a certificate of having completed the course. If they have not yet left feedback to the course, they can also do so as well.

3 Problem identification

This section discusses the aspects in the course which could be improved. The identification of problems is done based on several sources of information. The first is the authors own observations and personal experience when teaching in the course. Secondly, interviews were conducted with two students. Both of whom later decided to continue studying in the University of Tartu. Both of them are currently enrolled in the Computer Science curriculum in the Bachelor's level. The third source is the online course creating guide book [VKK+]. Also, both supervisors have conducted the course and have suggested ideas to the author.

3.1 Obsolescence of technologies

Technological advancement in programming tools and development environments is rapid. This means it should be assessed whether the technologies used in the course would benefit from changes.

Newer programming tools can contain features, which can aid in learning. This means that some particular tools can be utilized to aid in the teaching of some programming concepts, such as *for* loops. This is discussed in section 4.1.2, which explains why Thonny was chosen over IDLE.

Also, the tools should be up to date enough to be beneficial to the pupils, for whom this course is a step before studying computer science or similar curriculum in university. This is relevant, as one of the goals of this course was to cater to such students.

Libraries, also known as modules in Python, can also be rendered obsolete, if they do not receive continuous development. Section 4.1.1 discusses this regarding the Pygame game programming module for Python.

Another technological aspect that is discussed is the consistency of the mate-

rials and up-to-date tools. Naturally programming tools and operating systems change over time and this can cause there to be inconsistencies between the materials and what the pupils themselves experience when taking part in the course. Especially when deliberate changes are made in the tool sets of students, the inconsistencies are even more obvious. For this purpose the material was updated. This is discussed in section 4.

3.2 The grading system

An important aspect of an online course is the grading system. The pupils must be well informed about the way they are graded [VKK+]. In the authors experience when teaching the course before, the grading system has created a substantial amount of confusion. As the course is before the work of this thesis, the grading system in not well understandable. The criteria for successfully completing the course is completing 75 percent of the exercises according to the course material. However, in the middle of the course, it is marked in the materials, that the grading system has changed to mean 75 percent of points. This is a remnant from earlier iterations of the course.

To sum up, the grading system before the work of this thesis is the following:

- According to some documents, it is mandatory to complete 75 percent of exercises, but according to other documents it is 75 percent of points.
- Each programming task gives 2 points, when solved correctly.
- Each task gives 1 points, when marked as "mittearvestatud" (transl. "failed"). This is a major source of confusion, as *failed* could be interpreted to mean not receiving points at all.

- The exact number of exercises is not mentioned anywhere. The only way to count if one has enough points, is to manually count the tasks.

The grading system therefore needs to be rewritten. The improved grading system is described in section 4.2.

3.3 Interviews with former pupils

We conducted two anonymous interviews with the former students of the "Let's make computer games!" course. Both of the interviewees are currently enrolled in the Computer Science (BSc) curriculum in the University of Tartu. The original untranslated transcripts of the interviews can be found in Appendix I and II.

3.3.1 Interview one

The first interview was conducted in 21st of March in 2017. The following was discussed with the interviewee with the following answers:

- **Difficulty of the installation process, including Pygame.** The beginning was good and the materials helped with studying. It would be beneficial to teach, how to fit different concept together.
- **Usefulness of video materials.** The videos were very useful. If something did not make sense, then they helped. A problem was not being able to fast-forward the video to a specific topic conveniently.
- **Whether graphics should be introduced sooner.** First, the important concepts should be taught. On the other hand, some students want to create games right away. Would be good to tie exercises on programming concepts with tasks in a game. Introducing graphics sooner might make some students more motivated and give them an guess at how the end result of their own work might look like.

- **Hardest concept to learn.** The concept of classes was hard to understand. About what it does and what it is.
- **Is the concept of a OOP/class even necessary.** While creating a game it is right to use, because it is logical to create characters in a game using classes.
- **Are there enough exercises, where creativity is welcomed.** Prefer clearly defined exercises.
- **Balance between creativity and technology.** There were problems with the last project. Didn't know where to start. Would help to have some given topics and material for showing the process.
- **Fluidity of teamwork.** It is reasonable to divide tasks in such a way, where each team member understands each part.
- **Balance between teaching programming and game design.** Could have a bit more game design. Programming was well covered, but games were explained less.
- **The time investment in the course.** The course took a lot of time. About 7 hours per week. The exercises were a small part after the material. Could use more abstract exercises.
- **Necessity of extra materials.** There might be interested people, who want to advance themselves more. Graphics parts could give ideas for a project. The programming part is already pretty good.
- **Advice for instructors to make work easier for pupils.** Some people are afraid of asking questions and might have problems. A weekly questionnaire could be beneficial for sharing problems.

3.3.2 Interview two

The second interview was conducted in 2nd of April in 2017. The following was discussed with the interviewee with the following answers:

- **Difficulty of the installation process, including Pygame.** Set-up of Python was quite easy. Was easy compared to other Java courses. Was well explained, how things work.
- **Usefulness of video materials.** Watch mostly when the text was confusing. There were enough videos. Did not manage to watch all of them.
- **Whether graphics should be introduced sooner.** Would not have preferred that. Non-graphics part was more exciting.
- **Hardest concept to learn.** Creating a loop when creating a game. The snake game was quite large - too bothersome to write out.
- **Are there enough exercises, where creativity is welcomed.** Yes. Liked the exercises, where you could add something yourself.
- **Balance between creativity and technology.** There were enough creative exercises. The "choose your own adventure" game was especially exciting.
- **Fluidity of teamwork.** Did with people from my own school. Was good to discuss directly. Would have been harder over the internet.
- **Balance between teaching programming and game design.** It was quite balanced. The game part was sufficient. Had to search for extra information regarding connecting game development concepts and programming concepts.

- **The time investment in the course.** The beginning was less time consuming, as exercises were easier. Graphics part took longer as well as the "choose your own adventure" game. About two to four hours per week in the beginning, up to 6 in the end. First the time spent was the same for exercises and tasks. Later more for tasks.

- **Necessity of extra materials.** Rather not necessary. Would probably not have enough time.

- **Advice for instructors to make work easier for pupils.** Direct communication between instructors and students could be tried. Communicating over e-mail took a fair bit of time.

3.4 Final projects

The first interview mentioned, that it is difficult to get started with the final projects in the end of the course. He said, that he did not know where to begin with the project and that some materials and topic suggestions would be useful. According to the authors experience, getting started with projects has been difficult for many other pupils as well. During the previous iterations of the course, pupils have often had confusion regarding the projects, which needed extra communication to solve. This means it is necessary to provide some extra materials, which would advise pupils about how to approach the projects. This is discussed in section 4.4.

4 Changes and additions in the course

This section describes and explains the changes made in the course in detail. It is explained how the technologies used in the course have been made more up to date. Then changes in the materials are covered.

4.1 Changes in technology and tools

One problem identified in the initial course is the aging of technology used by the students for programming. The following subsections cover the decisions regarding the technology.

4.1.1 Re-evaluating Python and Pygame

The Python version suggested by the initial materials is 3.2.2. This choice is explained by the fact, that the official releases of Pygame only support Python up to this version.

A consideration was made to replace to Pygame module with a more supported module. This is due to the fact that it is not ideal, that the course materials instructs the students to acquire an old version of Python for making the Pygame library work. Luckily, during the work of this thesis the situation changed. The Pygame library received an update, which makes it possible to use Pygame on new releases of Python officially. For illustration, in [Pyga], only the old versions are shown, however [Pygb] now shows new versions supported.

4.1.2 Exchanging IDLE for Thonny

The option of replacing IDLE was considered to make the process of teaching easier and to make installation process less involved. A decision was made to use Thonny as the main IDE in the course. This is due to the very straightforward

installation process of the IDE and Python and Pygame along with it. Thonny provides a debugger, which can be used to visualize the state of variables. This is utilized in the new video material to explain how variables behave in certaion situations. These include for loops, where the looped variable in often not understood correctly. While IDLE also features a debugger, the one included with Thonny is more practical for teaching, because it displays less unnecessary information.

4.2 New grading system

To fix the problems mentioned in chapter 3.2, the following changes were made:

- The sum of points from all mandatory tasks is 100.
- It is necessary to collect 60 points to finish the course.
- The project must be completed to finish the course. This remains unchanged.
- Bonus tasks add the possibility to earn additional points.
- Each test gives 10 points. There are 3 tests in total.
- Each normal task gives 3 points.
- Larger tasks consist of 4 points, of which 2 is from a mandatory part and 2 is from a bonus part.

This was explained to the students in the beginning of the course and remained unchanged.

4.3 Revisions to the materials

The majority of the work done in the work of this thesis is the new video materials. All of the original videos in the textbooks were re-made to be consistent with the

chosen IDE – Thonny. Also additional videos were made to explain the topics more in-depth and to introduce the game development paradigms in more detail. These include the concepts of a game loop, handling the event queue, double buffering and OOP. The particular videos by week are described in the upcoming subsections.

Textual materials were made to accompany the video materials. A large portion of the code sample images in the textbook were remade to be consistent with the new tools. The number of these samples is large and listing them all one by one separately is redundant, as they are covered in the video materials chapters.

The new versions of textbooks can found in the Appendix IV.

4.4 Project example textbook

A new textbook was written to help pupils get started with the final projects. This is a solution to the problem described in section 3.4. The material can be found in Appendix IV.

The material starts with the idea of developing a "Who wants to be a millionaire?" computer game. The following is discussed in the textbook:

- The planning phase. How to divide the work into smaller pieces. This is an important step, because projects are written in teams, excluding some, who prefer to work alone.
- How to solve the task without graphics. This is done in four parts to emphasize splitting the work into several parts. It is important to begin without graphics, because some less advanced pupils prefer to do text-based projects. This keeps the examples simple for such pupils.
- How to extend the game with graphics. This is also done in 4 parts. It is shown how to write parts of the game graphical and how to connect the

parts.

4.5 Week one video materials

Nine videos were recorded for the materials of the first week. The materials covered the most basic concepts in Python programming and also the installation process of the required tools.

4.5.1 Installation

This video described the process of installing the Thonny integrated development environment. [1] First it was shown, where the installer could be downloaded from. Then it was demonstrated how to run the installer and after this the program itself.

4.5.2 First experiments in the shell

This video goes through the very basic concepts in Python just to get a feel of the language before explaining anything in detail.[2] This includes the classic printing of "Hello, world!". The programming in the video was done entirely in the Shell part of the Thonny IDE. The following was demonstrated to the viewers:

- The *print* function.
- The importance of quotation marks in strings.
- The addition and multiplication operations on numbers and strings.
- Printing several values with multiple arguments to the *print* function.

[1] `https://youtu.be/TG87bBU271s`
[2] `https://youtu.be/HAYgYVcx17M`

4.5.3 First Python program

This video moved on from the shell used in the previous video.[3] Now the text editor in Thonny was used for demonstrating the writing of a small program.

First it was shown, how to save a new file. This is necessary to run a written program in Thonny. Then the actual programming was initiated. The program consisted of elements very similar to the concepts shown in the previous video on the shell. The following was demonstrated to the viewers.

- print function
- print function without arguments for skipping a line
- the multiplication operation on numbers and strings to produce repetitive strings
- how to save and run the written program

4.5.4 Variables

This video introduced the concept of variables to the students.[4] The video starts with a sample code, where it is applicable to use a variable instead of writing integer literals.

The following was demonstrated to the viewers.

- a situation, where it is recommended to use variables
- how to define a new variable
- how to change the value of a variable

[3] `https://youtu.be/WwW78TQbm50`

[4] `https://youtu.be/8siRpH3w6uE`

- how to conduct mathematical operations with a variable with an example of addition

4.5.5 Operations with variables of varying types

The video starts with a sample with 3 distinct variables of different types - an integer, a float and a string. The following was shown in the video to demonstrate which operations are valid between the different types.[5]

- all mathematical operations between integers and floats are valid
- strings can only be a multiplied by integers and added to other strings
- string multiplication with a float instead of integer produces a crash

4.5.6 Error messages

This video demonstrates the common error messages one deals with when programming in Python.[6] The distinction between runtime errors and syntax errors was explained. Although several other videos deal with their sepcific errors as well, this video tries to summarize the most common ones and to explain errors are a natural part of programming.

The first mentioned error type is "NameError", a runtime error, which was shown by deliberately forgetting to add quotation marks around a string literal. As the error messages chapter in the textbook was now moved to be later than the variables chapter, it was possible explain that the "NameError" in this case means that Python interpreted it as an undefined variable. Two solutions for this were proposed to the student: if it was meant to be a string literal, then quotation

[5] `https://youtu.be/W1FuD139KkQ`
[6] `https://youtu.be/ll7idbmtmiA`

mark must be added or alternatively a variable with the name should be define if that was the intent.

The second runtime error time was "TypeError", which occurs when operations are done with incompatible types. This was demonstrated with an example of adding a string and an integer.

Also, an example of a syntax error was made. It was down with an example of an invalid operator sign in the program. It was shown, how Thonny adds a link to move to the exact line with the error.

4.5.7 Maths operations

This video shows example of the main mathematical operators and their order of operations and how to use them in Python.[7] A demonstration is done on the Thonny shell. This included the following:

- the main operators: addition, subtraction, multiplication and division
- integral division (// operator)
- exponentiation
- type conversion from float to int
- rounding of floats

4.5.8 Incrementing and decrementing of variables

In this video it was demonstrated how various operators can be used to change the values of variables.[8] The following operators where demonstrated in the Thonny shell:

[7] `https://youtu.be/fy_lZOShNUk`

[8] `https://youtu.be/Ox3-PoGf8O8`

- +=, incrementing a variable
- -=, decrementing a variable
- *=, multiplying a variable
- /=, dividing a variable
- **, exponentiating a variable

4.5.9 Data types conversion

The main data type conversion functions in Python are summarized in this video.[9] The following is explained:

- The *int()* function for conversions to integer type. It is explained how this can be applied to floating point numbers and strings. It is shown, how the floats are always rounded down. It is done to prevent the use of this functions where the *round()* function is appropriate.
- The *float()* function for conversion to floating point numbers. Again, it was explained how this can be applied to floating point numbers and strings.
- The *str()* function for conversion to string.

The video also demonstrates a common use for type conversions. It shows how the type conversion functions are used for converting user input to the required type. It is first shown how the *input()* function returns a string. The use of strings instead of numeric types can cause errors and mistakes when doing operations between numbers. Then it is shown how the type conversions fix it.

[9] `https://youtu.be/ksbUUiNWj3Y`

4.6 Week two video materials

Seven videos were recorded for the materials of the second week. The materials covered the topics of decisions, branching and loops. This includes boolean values, *if-elif-else* statements, *for* loops, the *range()* function, *while* loops and *break-continue* keywords.

4.6.1 Boolean values

Theis video introduces the concept of boolean value and how they are represented in Python.[10] The clip demonstrates various operators that produce boolean values as a result.

This includes the following common operators in Python:

- ==, !=, <, >, <=, >= operators applied on numbers.
- ==, !=, <, >, <=, >= operators applied on strings. It was said, how the <, >, <=, >= operators compare the alphabetic position of the string instead of its length as could be mistakenly assumed. It truth, it is actually lexicographical comparison, but this was not explained to avoid confusion. It was only said that strings starting with different cases should not be compared this way - this is to assure lexicographical and alphabetical orders are identical.
- It was also shown, how the length of strings can be compared. The *len()* function was introduced for this purpose.

[10] `https://youtu.be/fsNeFd54DfU`

4.6.2 If statements

This video makes an introduction to the concept of *if* statements.[11] This is demonstrated starting from an example code with some defined variables and a set of print statements.

This includes a variable for the "cost of ice cream" and various messages that could be displayed regarding it as shown in figure 2, depending on the cost and money available. The video incrementally shows how the code is supplemented by *if* statements to make decisions. The end result is seen in figure 3

```
raha = 2.0
jäätise_hind = 0.0

print("Sul on täpne raha!")
print("Sul ei ole täpne raha.")
print("Kahjuks jääb rahast puudu. :(")
print("Saad osta jäätise ja üle jääb", raha - jäätise_hind, "eurot.")
```

Figure 2: Initial code in the if statements video.

[11] https://youtu.be/QHopOKDEv1g

```
raha = 0.5
jäätise_hind = 1.0

if raha == jäätise_hind:
    print("Sul on täpne raha!")
if raha != jäätise_hind:
    print("Sul ei ole täpne raha.")

if raha < jäätise_hind:
    print("Kahjuks jääb rahast puudu. :(")

if raha >= jäätise_hind:
    print("Saad osta jäätise ja üle jääb", raha - jäätise_hind, "eurot.")
```

Figure 3: Final code in the if statements video

4.6.3 If-elif and if-else statements

This video expands the concept of *if* statements explained in the last video.[12] It introduces the *if-elif* and *if-else* statements.

This time a different example code was used as a basis, which is shown in figure 4. The initial code has a "score" variable and messages, which print a "grade" depending on the score's value. This is done in such a way that using only if statements alone would be cumbersome. The resulting code after else and elif statements have been introduced is shown in figure 5

```
punkte = 55

if punkte > 90:
    print("Hinne 5!")
if punkte > 70:
    print("Hinne 4!")
if punkte > 50:
    print("Hinne 3!")
```

Figure 4: Initial code in the if-elif-else statements video.

[12] https://youtu.be/atyJG_xdXB4

```
punkte = 55

if punkte > 90:
    print("Hinne 5!")
elif punkte > 70:
    print("Hinne 4!")
elif punkte > 50:
    print("Hinne 3!")
else:
    print("Mittarvestatud!")
```

Figure 5: Final code in the if-elif-else statements video

4.6.4 For loop

The *for* loop concept is explained in this video.[13] The video starts from a base code of a small loop over a list of integers.

It should be noted, that list have not yet been explained in-depth at this part of the course. However, it is assumed pupils can understand the list intuitively after a short explanation.

The following is explained about the concept of *for* loops:

- The *for* and *in* keywords that are used for defining the loop.
- The fact that the loop iterates over a collection specified after the in keywords.
- The loop variable is introduced. It is shown how the elements in the collection are consecutively assigned to the value of the loop variable.
- How changing or re-assigning the loop variable's value does not affect its value in the next iteration.

[13] `https://youtu.be/1eslC3MyYtc?list=PLVB62V-IO-1neE50Xj7v-aHdenwj6NrQs`

4.6.5 Range function

The utilization of the *range()* function in *for* loops is shown in this video.[14] The clip starts from an example with a *for* loop over a list. An unreasonably long list literal is used for illustration, as shown in figure 6.

The example demonstrates with a excessively long list the fact, that literal lists can be cumbersome. The video offers the *range()* function as a remedy for overly long list literals in cases, where the numbers are representable as an arithmetic sequence. The final result is shown in figure 7

It is shown how to:

- Make a simple range from zero to a given numeric parameter. This means *range()* function with one parameter.
- Make a simple range from a given numeric parameter to another given parameter number. This means the use of the *range()* function with two parameters.
- How to add a step size for the generated sequence of numbers. This means the use of the *range()* function with three numeric parameters.

```
for i in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
    print(i)
```

Figure 6: Initial code in the range video.

[14] https://youtu.be/GmI-uMsrpfo

```
vahekoht = 10

for i in range(5, vahekoht):
    print(i)
    print("Väike arv!")

for i in range(vahekoht, 15):
    print(i)
    print("Suur arv!")
```

Figure 7: Final code in the range video

4.6.6 While loop

The while loop concept is explained in this video.[15] The video starts from a base code without a loop.

The idea of the base code is the following. There is a variable depicting the "number of cakes", which is shown in figure 8. The code prints, that one is eaten and the number decremented by one. The goal is to do this with a loop numerous times. The video explains how this can be achieved with a while loop, the result of which is shown in figure 9.

```
koogikesi = 10

if koogikesi > 0:
    print("Sõin ühe koogi!")
    koogikesi -= 1
    print("Kooke on nüüd alles", koogikesi)
```

Figure 8: Initial code in the while loop video.

[15] https://youtu.be/-H-h5eXTWVA?list=PLVB62V-IO-1neE5OXj7v-aHdenwj6NrQs

```
koogikesi = 10

while koogikesi > 0:
    print("Sõin ühe koogi!")
    koogikesi -= 1
    print("Kooke on nüüd alles", koogikesi)

print("Koogid on nüüd otsas!")
```

Figure 9: Final code in the while loop video

4.6.7 Break and continue

The video explains, how the *break* and *continue* keywords can be used to control the flow of loops.[16] An example code is explained regarding *while* and *for* loops separately.

The for loop example starts from a for loop over a list of numbers the are summed up to a variable. The demonstration consists of two parts. First is done by excluding negative numbers from the loop using *continue*. The second is to stop the entire loop when the first negative number occurs using *break*.

4.7 Week three video materials

Four videos were recorded for the materials of the third week. The materials covered the topics of debugging with Thonny, nested loops, lists and Turtle graphics.

4.7.1 Thonny debugging

The video shows how to use two of the debugging capabilities of the Thonny IDE.[17] These two options are: the debug mode and the variables view.

Regarding the debug mode, it was demonstrated, how Thonny allows the user

[16] https://youtu.be/bF0i7hgQLzM

[17] https://youtu.be/DvtXqqs0w-A

to run their program step-by-step. This also includes the possibility to see the mid-steps in the evaluation and execution of the lines of code. Essentially the user can see how the statement is traversed by Python during interpretation.

For the variables view, two things can be seen. First, all the variables values at the termination of the program are shown. Second, in the debug mode the view shows the change of variable values in real time.

4.7.2 Nested loops

The video shows how to use nested loops in Python.[18] This is demonstrated with an example of printing a multiplication table.

The video makes use of the previously covered debug mode of Thonny along with its variable view. This is used for the purpose of showing how the loop variables change with each iteration in the loops.

4.7.3 Deeper nested loops

The video shows how to use nested loops in Python with 3 layers of for loops.[19] This is demonstrated with an example of printing an asterisk grid multiple times.

The purpose of this video is to imply how the depth of the nesting can be arbitrary. This is again done with the help of the debug mode of the Thonny IDE.

4.7.4 Turtle graphics module

An example of using the Turtle graphics module of Python is shown in this video.[20] First the clip lists and demonstrates all the essential commands when drawing shapes with Turtle:

[18] `https://youtu.be/c2fQd2wZ4n0`
[19] `https://youtu.be/IdYtBIwttr8`
[20] `https://youtu.be/QEllf7AS9os`

- *forward*() and *back*()
- *left*() and *right*()
- *up*() and *down*()
- *speed*()

The second part of the video shows a more complex example. A more intricate spiral shape is drawn using a *for* loop.

4.8 Week four video materials

Eight videos were recorded for the materials of the fourth week. The materials covered the topics of functions, scope of variables and objects.

4.8.1 Defining functions

The goal of this video is to teach how to define new functions.[21] The video starts with a simple base code with input and print function calls. It ties the concept of functions to the functions used in the course so far.

The base code is modified to define a new simple function. It is shown how to do so using the *def* keyword. It is shown how the newly defined function can be called just like the functions used in the course previously. It is explained, that functions can be called multiple times.

4.8.2 Function arguments

This video shows and example of defining a more complex function.[22] This time the function has defined arguments.

[21] `https://youtu.be/x9ASbAe2ci8`
[22] `https://youtu.be/unk8MH19MgQ`

The example is made with a function that draws a grid of asterisks based on given parameters. The parameters dictate the dimensions of the grid. The working of the function is illustrated by using the built-in debugger of Thonny.

4.8.3 Returning a value

The return keyword is introduced in this video.[23] The video starts with a base code example.

The example contains a function, that only prints the result of a computation, but does not return anything. It is shown, how the result of the computation cannot be reused at all in the future, if it is only printed. This is an important example, because it is one of the common beginner mistakes to use the print statement instead of return [VKDL]. It is then shown how the return keyword can be used for returning results from a function. It is demonstrated, that the result can be assigned to a variable and used like any other value in a variable.

4.8.4 Scope

This video explains the difference between global and local variables.[24] It is shown in which conditions the variables can be used in different scopes:

- Local variables not accessible nor seen in global scope.
- Global variables are visible in and local scope, but are not modifiable without the *global* keyword.
- Why to use global variables at all in functions? - To consolidate repetitive constants.

[23] `https://youtu.be/evUwVTiT63U`
[24] `https://youtu.be/9WvVyPZSqzo`

4.8.5 Classes and attributes

The first introductory video about classes and objects.[25] The example start with a game-based base code.

The base code consists of a description of some game enemies. This description includes variables for name and life count and a print statement of the information describing the enemy. It is shown, how these separate variables can in stead be described using a class.

The solution starts with the definition of an empty *Enemy* class. Attributes are added to it one by one accordingly to the initially described enemy.

4.8.6 Classes and methods

This video continues the same example shown in the last video.[26] It is continued by supplementing the defined *Enemy* class with a simple method, that prints out the enemies information.

4.8.7 Object initialization

The video again proceeds to improve the class from the previous video.[27] This time the addition is the initialization of attributes of objects.

It is shown, how the $\_\_init\_\_()$ function can be defined to assign initial values to the created object's attributes. In the end it can be seen, how the method defined in the previous videos makes use of the initialized attributes.

[25] `https://youtu.be/zyMGOwRSx3M`
[26] `https://youtu.be/ZgN-TBtbg3U`
[27] `https://youtu.be/i4ttML5z0y0`

4.8.8 Class use example

This is the last video of the objects concept.[28] It concludes the example from the previous few videos. It is shown, how multiple instances of a class can be constructed. It is shown how to add multiple of the objects to a list and how to call the same method from them using a for loop.

4.9 Week five video materials

Seven videos were recorded for the materials of the fifth week. The materials cover various topics about game development using the Pygame module.

4.9.1 Basic window

The first Pygame video starts with a basic example of a window.[29] A simple window is created and closed right after. It is explained, what each of the Pygame functions is for. The things meant to be observed by pupils from this part are:

- Importing Pygame
- The *pygame.init()* function, which initializes the module, making it possible to use its functionality.
- Creating a window with a set size.
- Closing Pygame and the window along with it using the *pygame.quit()* function.

The video continues by next showing how to make the window remain on the screen, instead of closing immediately as in the previous sample. For this, the event loop concept is introduced. The following is demonstrated:

[28] `https://youtu.be/pv-sEwndQ6s`
[29] `https://youtu.be/TGNdz1gVgx0`

- The *pygame.event.get()* function is used acquire a set of events. Usually these events originate from user input.
- It is shown, how the events can be handled using a *for* loop.
- An example is given on checking the type of an event. The example is this video is the quit event, which is triggered when the window is being closed.

4.9.2 Game loop

This video introduces the concept of a game loop, a pattern widely used in game development.[30] Is is explained, how this pattern should be used in Pygame. The game loop traditionally consists of three main steps and optionally one more auxiliary step:

- The first is the events and input processing phase. This stage is for detecting all of the input originating from the user and sometimes the computer. This is illustrated with the same event loop example from the last video - the loop, which detect the *pygame.QUIT* event, which is thrown when the window is closed.
- The second is the update phase. This step is used for computing the next step of the game. This includes physics, AI and any other game logic. In the example, this step is used for randomly modifying a variable, that represents a color.
- The third step is the draw phase. As the name suggests, this step is for drawing the current state of the game and displaying it on the window and/or screen. In the example, only one thing is drawn - the background. This is

[30] `https://youtu.be/zsd4w4bDujY`

done by making us of the variable, that is constantly changed in the *update* phase. The result is a window randomly blinking background.

- The last optional step is to sleep a set amount of time. This is done to not let the game run is an unnecessarily fast tempo. The make the blinking bearable yet keep the example simple, the wait is 300 milliseconds in this example.

4.9.3 Moving image

The video shows, how moving objects can be created in Pygame.[31] The explanation is done based on the game loop pattern explained in the previous video. The following is done to demonstrate the concept:

- X and y variables are set for representing the position of an image.
- During the update phase, the x variable is incremented by one in each iteration.
- As the game loop pattern advises, the draw phase draws displays the new state on screen after each iteration.
- The waiting is done for 17 milliseconds to hold the frame-rate around 60, which is a typical screen refresh rate. This is a simplification, because the producing of a frame would also take some time, which should be considered. To keep the example simple to understand for pupils, this was not included in the tutorial.

[31] `https://youtu.be/QQ8-5A10spg`

4.9.4 Movement by keyboard action

Keyboard input is explained in this video.[32] A base code is provided, where an image is movable with the arrow buttons on the keyboard. The same code is shown in the textbook as well. The video explains all the steps done in the code:

- A string variable is defined to represent the movement direction of the image
- During the event loop it is detected, when arrow buttons are pressed. Each key press changes the direction variable.
- During the update phase, the x and y coordinates of the image are modified according to the direction variable.
- The draw phase draws displays the new state on screen after each iteration, where the image has moved.

4.9.5 Advanced movement by keyboard action

The second video regarding keyboard input makes a more advanced keyboard input example.[33] This means that several drawbacks of the previous example are eliminated - it is possible to move diagonally with two buttons and the movement is stopped by releasing the keys. To conclude, this is done as follows:

- As before, the image has x and y coordinates.
- In addition, the image now also a a velocity, which is represented by speed on the x and y axis in two variables.
- During the event loop, pressing a key adds speed along the appropriate axis and releasing it negates the added speed.

[32] `https://youtu.be/oYtJxqqdSVE`
[33] `https://youtu.be/elKEnW1jnnU`

- During the update phase, the x and y coordinates of the image are modified according to the velocity.
- The draw phase draws displays the new state on screen after each iteration, where the image has moved.

4.9.6 Mouse position

This video shows how to use the mouse position.[34] It is explained, that the mouse position, once received from Pygame, is just a numerical value that can be used just like any other numerical value in Python. The example demonstrated in the video explains the following:

- How to acquire the x and y values of the mouse position using the function of *pygame.mouse.get_pos()*.
- The position of an image on the screen is updated accordingly to the mouse position. Essentially the image moves along with the mouse position.
- The draw phase draws displays the new state on screen after each iteration, where the image has moved to the mouse position.

4.9.7 Mouse click

This video improves upon the same example shown in the last video.[35] However, instead of always moving along with the mouse cursor, it is moved only when a click is made on the screen. The new steps done in the example are:

- Assigning a variable to keep track of whether the mouse is currently held down and the image is *dragged*.

[34] `https://youtu.be/m-uGd7fUchQ`

[35] `https://youtu.be/K-ztioAq2f8?list=PLVB62V-IO-1lkOOpAIkJv7pKvqo9wr-gx`

- During the event loop, the mentioned variable *dragged* is updated when mouse clicks and releases are captured.
- During the update phase, the x and y coordinates of the image are modified when the *dragged* variable is true.
- The draw phase draws displays the new state on screen after each iteration, where the image has potentially moved to the mouse position.

4.10 Changes in tasks

Some new tasks were introduced into the course with the purpose of introducing simple graphics earlier. This is to make the beginning more exciting to the students. The new tasks consisted of 2 parts - the first being mandatory and the second a bonus. A simultaneously written Bachelor's thesis by Mark Muhhin also introduced a number of new tasks to the course. The tasks from the Bachelors thesis are not covered in this work.

4.10.1 Graphics for a "Space invaders" game

A new task was introduced in the first week, which had two goals. The first aim was to make sure the student had properly configured the Thonny environment along with an installation of the Pygame module. This was relevant, because it has been a recurring problem in the course, that some students had not succeeded in installing Pygame by the time it was needed, even though the textbook contains optional Pygame samples. The second aim was to make the first week more exciting by demonstrating graphics in Python and Pygame.

A set of base files were given for solving this exercise. This consisted of a Python source file and a set of images. The source was for a simple space invaders style game and the image files were used in it.

The exercise consisted of a main part and an extra credit part. The main task consisted of two necessary steps as well. The first step was to just make the game work properly by having Thonny and Pygame installed as instructed. The second step was to replace the image files by something the student found interesting. An option was given to either find new fitting images from the internet or the pupils could also draw their own images.

The extra credit part of the exercise describes of a problem in the image files. Namely that the images were lacking an alpha channel, which means their background is not transparent. This causes graphical anomalies in the game in some cases. The pupils were instructed to independently research about transparency in images and to replace to existing image files with new ones, where proper transparency was present. The students again had a set of possibilities to choose from, when solving this exercise. They could do any of the following:

- find new images from the internet with proper transparency
- draw new images themselves using graphics software whey are familiar with
- reconstruct the original images by adding an alpha channel to them, if the student feel confident enough with some piece graphics software

Figure 10 demonstrates the base game the students were given. The spaceship and asteroid images were the ones the students were instructed to replace according to their own preference.

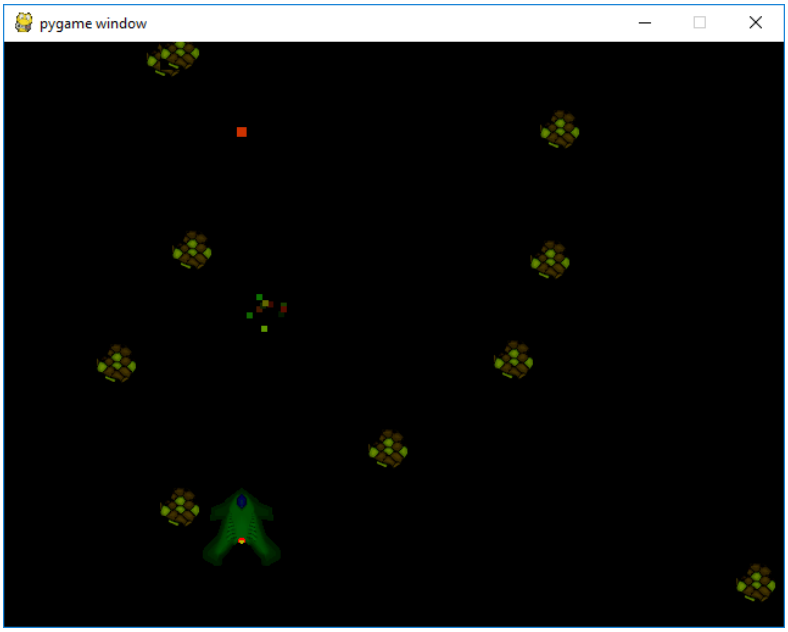

Figure 10: Base program from the exercise.

4.10.2 Numbers divisible by 3

Unlike the other exercises added during the work of this thesis, this exercise does not feature computer graphics. This is due to the fact that this week's textbook featured a large number of new topics, which needed corresponding exercises. This meant there was no room for adding a graphical task.

This exercise already existed in the original materials as described in section 2.4.2, but was extended with a bonus part. The added bonus part instructed to find the count of number that satisfied at least one of the following criteria:

- The number is divisible by 5
- The number is divisible by 7

It was mandatory to do this using loops. The key part of solving this is understanding that numbers divisible by 35 should not be counted twice. The bonus

part's purpose is to let more advanced pupils apply their knowledge of mathematics when writing loops.

4.10.3 Graphical "choose your own adventure" game

An exercise was added to the course, which asked the pupils to write a game in the "choose your own adventure" style. This task was introduced into the third week. They were provided with a base code with 2 Python source files and some template pictures. One of the Python source files was a template source code of the game with examples, which the students had to fill with their game logic.

In essence, the idea of this exercise is very similar to the "choose your own adventure" type game described in section 2.3.2. However, the second Python source file provided was an engine for creating the game in a graphical environment. It provided the students with 2 new functions. One for displaying a message and the other for letting the player choose between a set of answers to a question.

The exercise consists of a main part and an extra credit part. The main task is to be creative and modify the template to represent a new story. The extra credit part instructs the students to create at least some more intricate logic in the game. This means the existence of a list that is modified during game-play or a loop in the game.

The two new functions provided to the students for creating the game are the following:

- *show_message(image_address, message_text)*, the show message function, which displays an image with given texts. The name is translated. The function produces a message window, which is demonstrated in figure 11.

- *ask_choice(image_address, question_text, choice_list)*, the multiple choice question function. The name is translated. The function produces a question

window, which is demonstrated in figure 12.

The *image_address* parameter defines, which picture the message or question displayed. The *message_text* and *question text* set, which text is displayed for the user. The *choice_list* is a set of strings to choose from. The function returns the chosen string, which the could then be used to make decisions in the game.

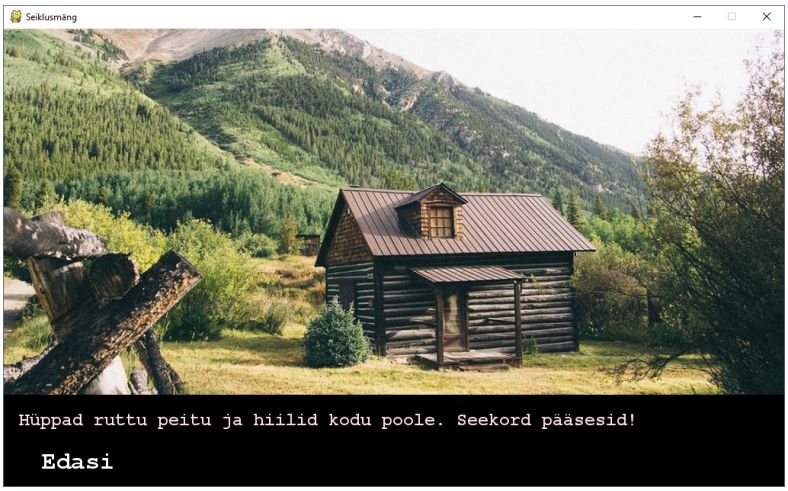

Figure 11: Example of a message displayed by the game.

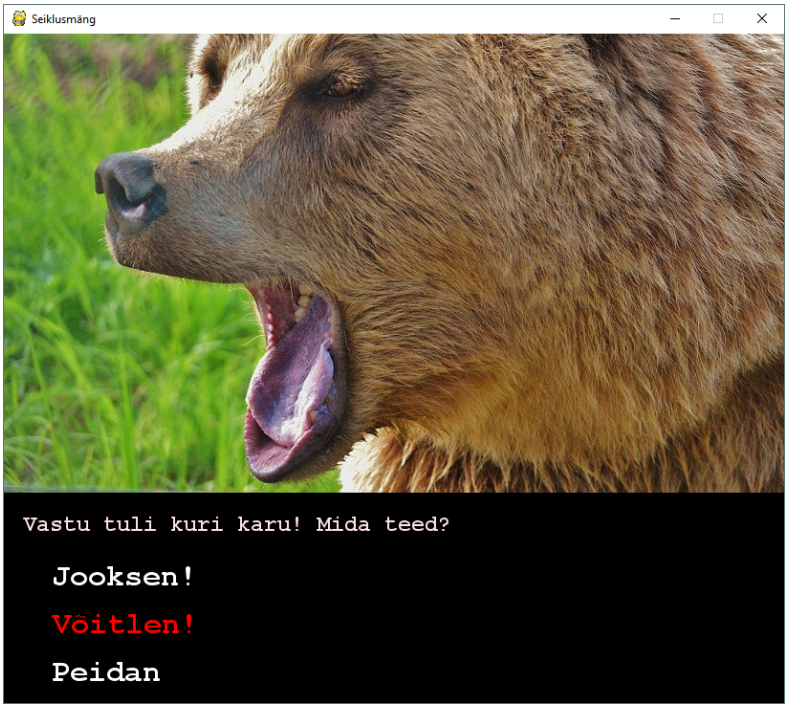

Figure 12: Example of a multiple choice question displayed by the game.

The following template of code was provided for the students. The purpose of this was to explain how to use the two new functions for creating a game.

```python
from alus import *

# Siin on ette antud väike näidis, kuidas kasutada funktsioone
# küsi_valikut() ja näita_teadet(), mis tulevad alus.py failist.
# Tee oma seiklus näidise abil. :)

# Esimene parameeter määrab pildi nime, teine küsimuse, kolmas on list valikuvariantidest.
# Valik väärtuseks saab see variant, mis valiti.
valik1 = küsi_valikut("metsatee.jpg", "Oled metsateel. Kas lähed otse koju või uurid metsa?", ["Uurin metsa", "Lähen koju"])

if valik1 == "Uurin metsa":

    valik2 = küsi_valikut("karu.jpg", "Vastu tuli kuri karu! Mida teed?", ["Jooksen!", "Võitlen!", "Peidan"])

    if valik2 == "Jooksen!":
        näita_teadet("majake.jpg", "Karu oli kiire, aga sina kiirem. Jõudsidki oma väiksesse majja. :)")
    elif valik2 == "Võitlen!":
        näita_teadet("karupeidus.jpg", "Karu ehmus su julguse üle. Seekord pääsesid!")
    elif valik2 == "Peidan":
        näita_teadet("majake.jpg", "Hüppad ruttu peitu ja hiilid kodu poole. Seekord pääsesid!")

elif valik1 == "Lähen koju":
    # Esimene parameeter on pildi nimi, teine on teade, mida näidatakse.
    näita_teadet("majake.jpg", "Jõudsidki oma väiksesse majja. :)")

# See järgnev jäta alles, et aken kenasti kinni läheks.
pygame.quit()
```

Figure 13: Python code of the example and template file given to students.

4.10.4 Moving image

An exercise was added to the fifth week of the course, which asks the students to write a program featuring a movable image. The exercise consist of a manadatory and a bonus part.

The mandatory part asks for a program that features the following:

- A window created with Pygame.
- The window must contain an image, that can be moved by keyboard input.
- When the image reaches and crosses an edge, the image must reappear from the opposite edge.

The bonus part asks for the following improvements in addition to the previously mentioned list of requirements:

- The image must be affected by gravity. This means the speed on the y axis is increased in each step of time.
- The image must stop falling when reaching the bottom.
- There's a button for jumping. This means the speed on y axis is increased upwards.

The purpose of this exercise is to give students experience with the game loop pattern and the solidify the understanding of the coordinate system used in Pygame. The bonus part of the exercise lets more advanced students apply their previous knowledge of physics in programming to a larger extent.

5 Results and discussion

The updated course was conducted in the spring of 2017. The duration of the course was from the 27th of March to 21st of May. The final works of the students were submitted by the 14th of May. 33 students were registered in the course. 19 of the students passed.

The quality of the course materials and exercises was assessed by weekly questionnaires. There was another questionnaire at the end of the course, which covered the whole course.

5.1 Questionnaire

For each week in the first 5 weeks, a questionnaire was given to the students. The pupils were asked to assess the quality of the materials and exercises on a 10 point scale. They were also given open text fields to add additional comments. The questionnaire was not made mandatory for the students. There is no direct comparison of the feedback from the earlier iteration of the course, because such weekly questionnaires had not been conducted previously.

The remaining last weeks of the course were for finalizing projects, so there were no mandatory exercises or materials to asses. The feedback for the project materials and organization was included in the final feedback of the course.

For the material, the students were asked to give a numeric assessment to the textbook and the included video materials in range from 0 to 10, where 10 is the highest score. Additionally they were given fill-in text boxes for adding further comments. The following questions were asked each week:

1. Was the textbook adequate for explaining the topics? (0 - 10) (0 - insufficient, 5 - satisfactory, 10 - excellent)

2. How well did the video materials support the explanation of topics? (0 - 10) (0 - not at all, 5 - moderately, 10 - greatly)

3. Are there any comments or suggestions regarding the materials? (fill-in field)

For the new exercises, similarly constructed questions were asked:

1. How difficult was the exercise? (0 - 10, higher number means harder)

2. How exciting was the exercise? (0 - 10, higher number means more exciting)

3. How much time did the exercise take? (arbitrary number in minutes)

4. Additional comments. (fill-in fields)

At the end of the course, the following was asked regarding the materials:

1. Was the materials adequate for supporting the implementation of the project? (0 - 10) (0 - insufficient, 5 - satisfactory, 10 - excellent)

2. Please comment the last answer. What was good and what was bad? (fill-in field)

The following was asked in the conclusion of the course regarding the course as a whole:

1. How would you evaluate the course? (very good, good, satisfactory, non satisfactory) A fill-in field was provided for clarifications.

2. Would you recommend the course to your friends? (yes, no, don't know) A fill-in field was provided for clarifications.

3. Did the course meet expectations? Was something missing or redundant? (fill-in field)

4. Did the forum system suit you? (yes, no, don't know?

5. Did forum posts by peers help understanding topics? (yes, no, don't know) A fill-in field was provided for clarifications.

6. How would you evaluate the course materials (books and videos)? (very good, good, satisfactory, non satisfactory) A fill-in field was provided for clarifications.

7. How would you evaluate the work of instructors? (very good, good, satisfactory, non satisfactory) A fill-in field was provided for clarifications.

8. Did interest in programming increase or decrease after the conclusion of the course? (fill-in field)

9. Would you like to continue studying programming in university? (yes, no, don't know)

10. Additional comments. (fill-in field)

11. Additional feedback to instructors. (fill-in field)

The figures showing results of the questionnaires in the following subsections have been translated. The questionnaire also contained questions regarding exercises created in the work of the Bachelor's thesis by Mark Muhhin. These are not discussed in this work, with one exception. The exception is the following question, which from now on will be considered as part of the additional comments for exercises: **Which exercise did you like the most and why?**

All of the questionnaires can be found through Appendix III, where all the questions and answers are unchanged and not translated.

5.2 Week one

Figures 14 and 15 shows the numeric feedback to the textbook and video material. It can be observed, that the textual materials peak at the score of 8. The video materials peak at the score of 10. The fill-in question also produced several relevant observations, that can also give context to the numbers. Those observations and suggestions were:

- One mentioned problem was regarding the use of the OS X operating system. This also corresponds with the fact, that during the week several e-mails were received regarding getting Pygame to work on OS X. A solution was found and the textbook was edited to work on the OS X platform.

- "The videos don't teach anything new compared to the textbook. Both at the same time are unnecessary." This point could be argued using the scores in figure 15. The textbook's score has less variance, so it is a good baseline for the material. However, a substantial number of students gave the videos a higher score, which confirms it can be considered necessary.

- The time spent when working through the material interferes with other school work. Regarding this, the course instructors tried to be accommodating with the deadlines if given notice and a reason ahead of time.

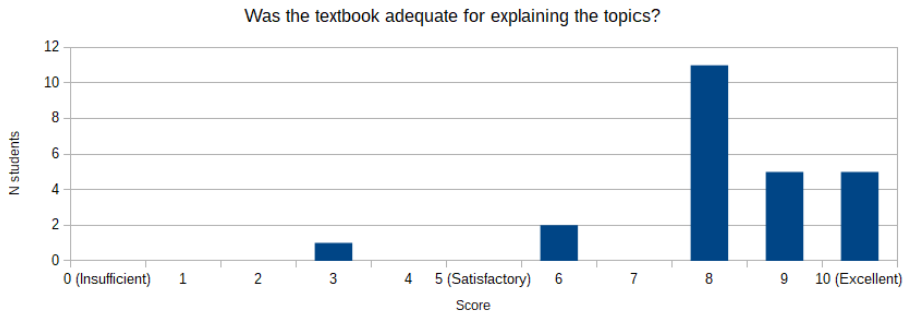

Figure 14: Feedback to the textbook material of week 1.

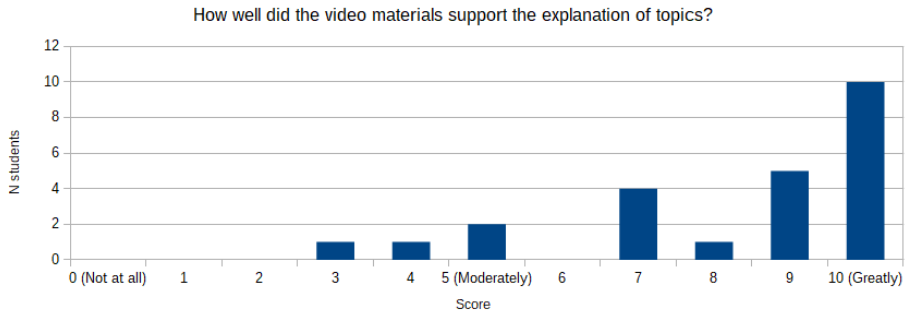

Figure 15: Feedback to the video material of week 1.

Figures 16 and 17 show the numeric feedback to the exercise described in section 4.10.1. The difficulty assessments peak at around 4 and 5. For most of the students the exercise is of moderate difficulty. However, there is a number of votes on the high and low difficulty assessments as well, meaning for some students the exercise was very easy or difficult. The people assessing this exercise to be hard were most probably students using OS X. As mentioned before, installing Pygame caused problems for people. This exercise needed the Pygame module to work. The exercise is generally considered to be moderately exciting, peaking at a score of 7. On average, the students reported the exercise took 31 minutes to solve.

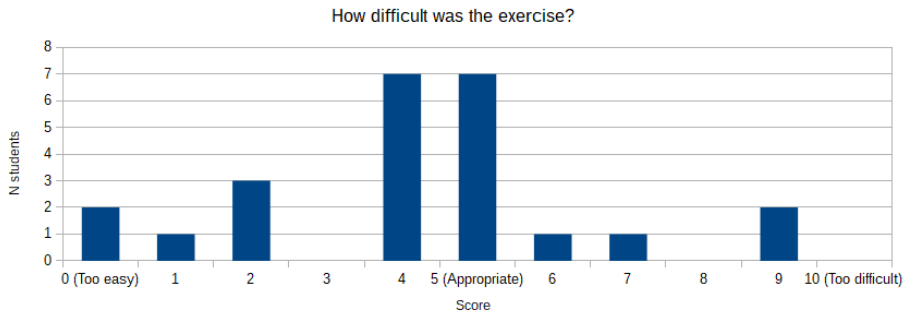

Figure 16: Difficulty of the new exercise added to week 1.

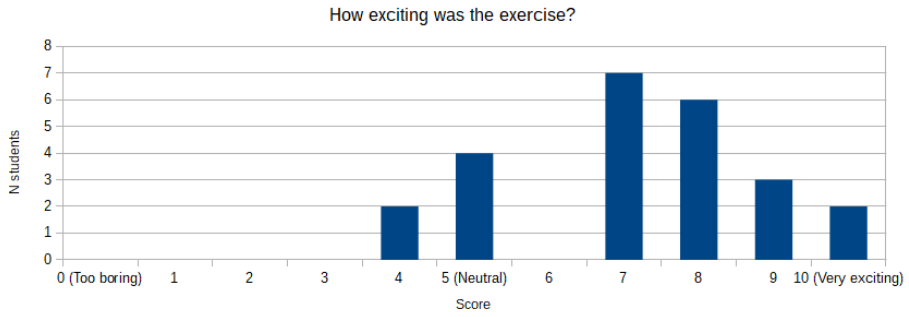

Figure 17: Excitement at the new exercise added to week 1.

The additional comments contained the following thoughts about the exercise:

- Liked that is showed using external files.
- Was exciting to do.
- It was practical.
- Liked, because deals with making games.
- Liked testing different pictures.
- Liked drawing his own pictures.

- Pygame difficult to configure in OS X. This complaint was addressed by updating the course material with instructions for OS X.

5.3 Week two

Figures 18 and 19 show the numeric feedback to textbook and video material. This week received the lowest ratings compared to the other weeks. The textual feedback suggests that this comes from the fact, that the week's material and exercises are very difficult. The second week indeed consists of a number of complex new topics: *boolean* values, *if* statements, *for* loops and *while* loops. The textual observations and suggestions included:

- Suggestion to be flexible with the deadlines, as the amount of topics covered is large. As in the previous week, the course instructors tried to be accommodating with the deadlines if given notice and a reason ahead of time.

- Suggestion to cover less topics. This is indeed worth considering when conducting the next iteration of the course. This is discussed in section 5.2.

- Suggestion to have more examples that are relevant to the set of exercises. As most of the exercises and the materials were compiled as parts of separate theses, there can indeed be discrepancies. Effort was made to do minor adjustments to the material during the course, but further work would be beneficial regarding the discrepancies.

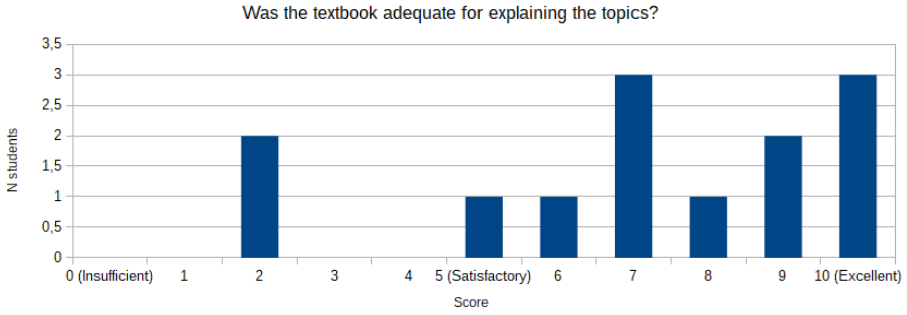

Figure 18: Feedback to the textbook material of week 2.

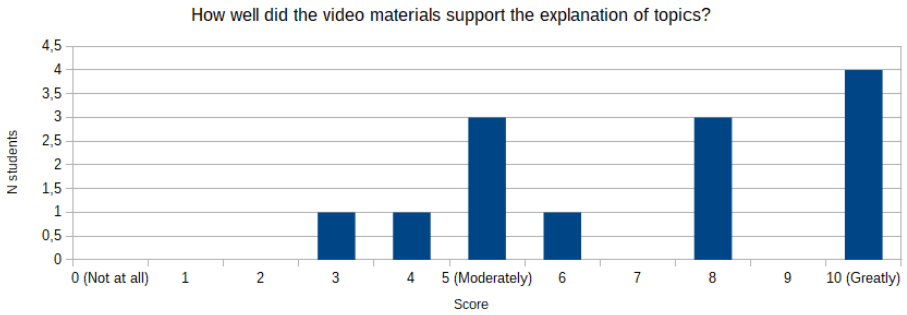

Figure 19: Feedback to the video material of week 2.

Figures 20 and 21 show the numeric feedback to exercise described in section 4.10.2. The difficulty of the exercise is generally considered to be appropriate. On the excitement scale, the exercise is slightly over neutral, peaking at scores of 6 and 7. This is slightly lower than the graphical exercise added to week one. On average, the students reported the task took 43 minutes to solve.

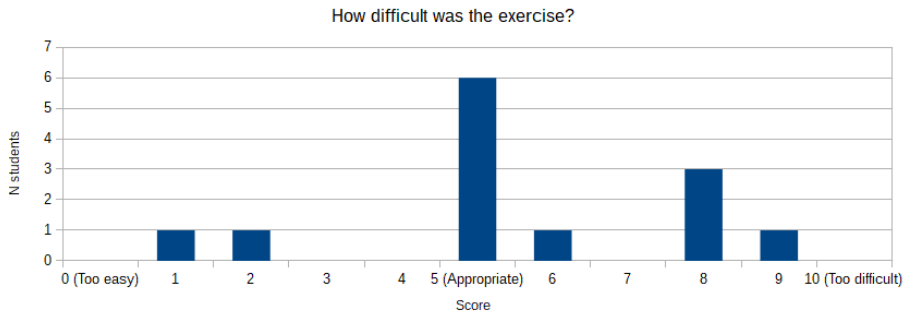

Figure 20: Difficulty of the new exercise added to week 2.

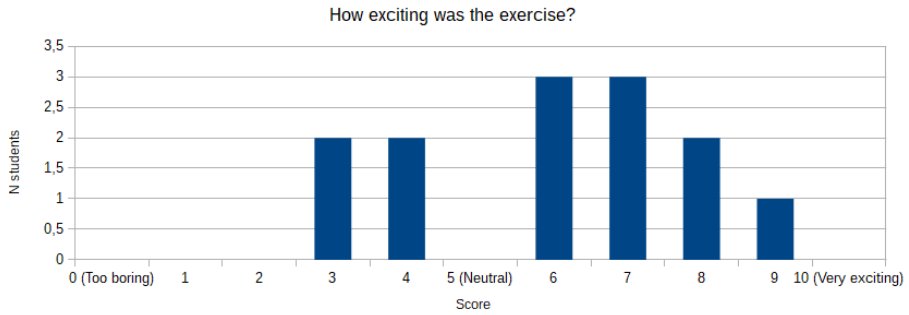

Figure 21: Excitement at the new exercise added to week 2.

The additional comments contained the following thoughts:

- Unclear which numbers to to find. One student also e-mailed about this. As a result the wording of the exercise was improved in the midweek
- Liked, that solving the bonus exercise did not require much changes to the mandatory part.

5.4 Week three

Figures 22 and 23 show the numeric feedback to textbook and video material. A positive change can be observed when compared the the previous week. The

following point was brought out regarding the materials in the fill-in feedback form: one exercise required the use of the *max* function, which is not covered in the materials. This created a discrepancy between the materials and exercises.

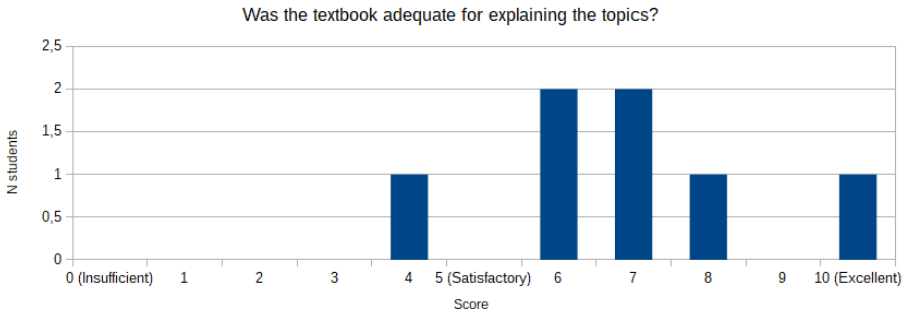

Figure 22: Feedback to the textbook material of week 3.

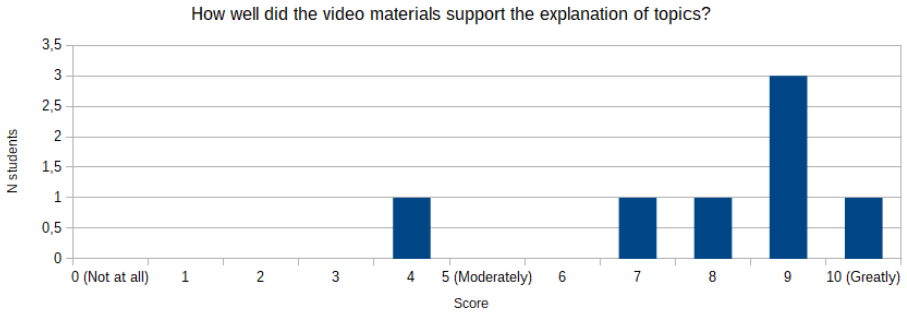

Figure 23: Feedback to the video material of week 3.

Figures 24 and 25 show the numeric feedback to exercise described in section 4.10.3. The task is considered to be exciting by a majority of the students, with the score peaking at the highest value of 10. However, the difficulty level of this exercise is also considered to be higher than average, peaking at about 7. On average, the students reported the task took 59 minutes to solve. This is a substantial amount of time for one exercise. This is probably the reason for the above average difficulty rating.

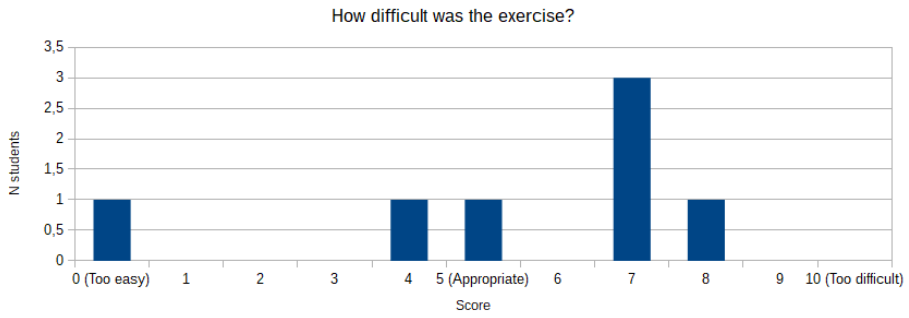

Figure 24: Difficulty of the new exercise added to week 3.

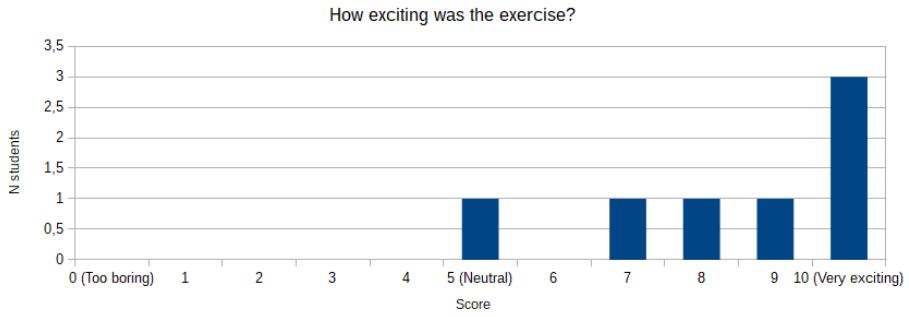

Figure 25: Excitement at the new exercise added to week 3.

The textual comments included the following thoughts:

- Liked the use of pictures in the exercise.
- It was interesting.

5.5 Week four

Figures 26 and 27 show the numeric feedback to textbook and video material. The feedback is generally positive. The following was covered in the feedback textually:

- The summary in end the of the material claimed, that students know how to use the __*str*__ method of objects, although this was not covered. This mistake in the summary was fixed.
- The functions part was difficult to understand.

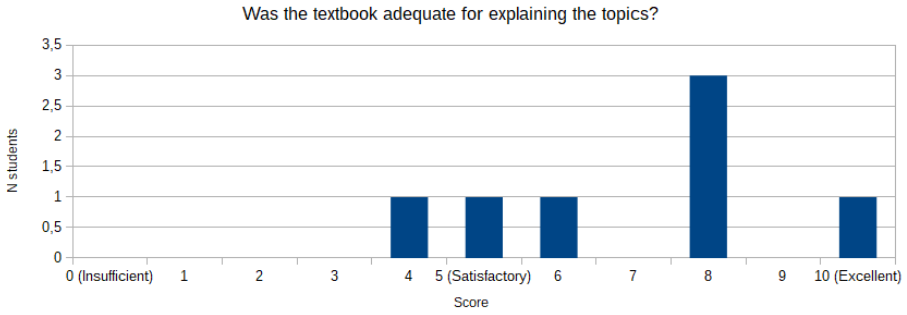

Figure 26: Feedback to the textbook material of week 4.

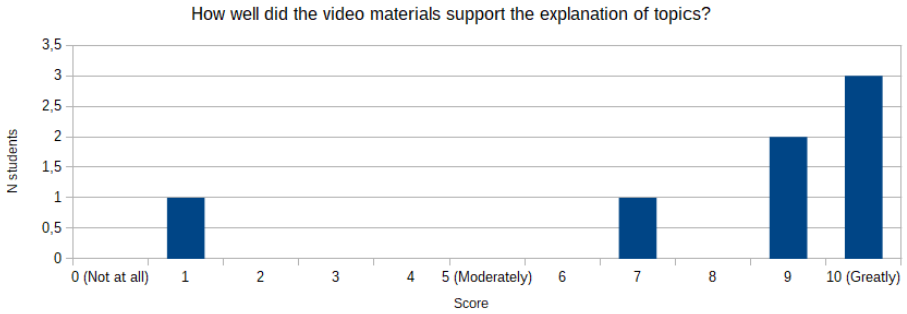

Figure 27: Feedback to the video material of week 4.

Unfortunately, there was an oversight when conducting the questionnaire this week. The questions regarding the quality of the added exercise were left out. Therefore numeric analysis cannot be done. However, there is textual feedback regarding this exercise. This exercise was mentioned to be the best in the week by 6 students.

5.6 Week five

Figures 28 and 29 show the numeric feedback to the textbook and video material. The feedback for this week is positive, excluding a single outlier, which can be observed for both the textual and video materials.

The following was said about the weeks materials:

- Still did not understand the topic - materials covered everything, but many questions still remained. This claim shows, that there are some students, whom the instructors should try to reach and help. This means students, who have questions, but do not communicate of them through the current channels. It would require further investigation to discover ways to reach such students.

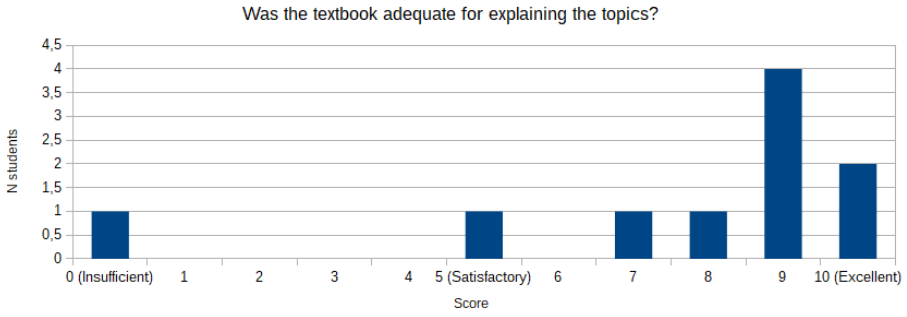

Figure 28: Feedback to the textbook material of week 5.

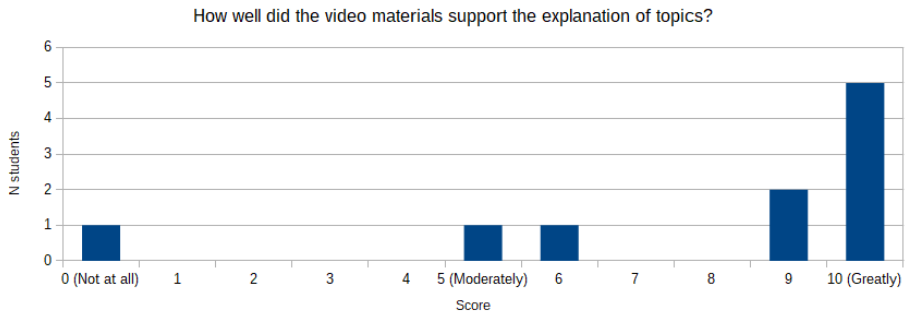

Figure 29: Feedback to the video material of week 5.

5.7 Weeks six to seven: project weeks

Figure 30 shows the numeric feedback for the course material regarding the project. Appropriateness for the purpose of supporting the implementation of projects was graded by the students. The outcome of the questionnaire is positive, peaking at the score of 9.

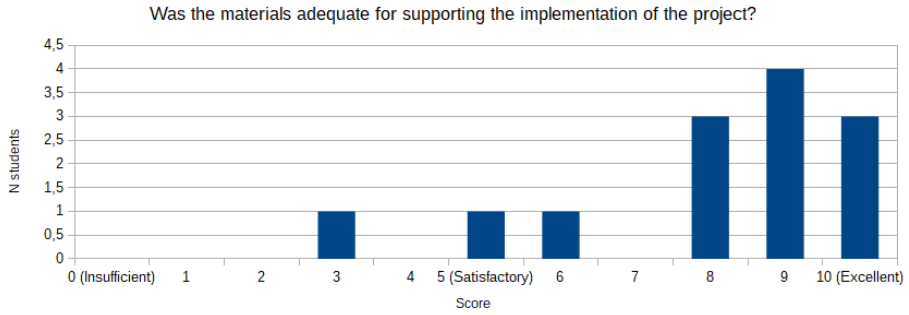

Figure 30: Feedback to the material regarding the project.

Summarily, the following was mentioned by the students:

- Different aspect could be learned.
- Could have used more example projects.

- Jumping from the exercises to the project was a too large endeavour.
- Occasionally had to look for more information.
- The video material helped explain the topics well.

5.8 The course feedback

The questionnaire at the end of the course assesses the entire course as a whole. Figure 31 shows how students grade the course overall. A large majority of students evaluated it as *very good*.

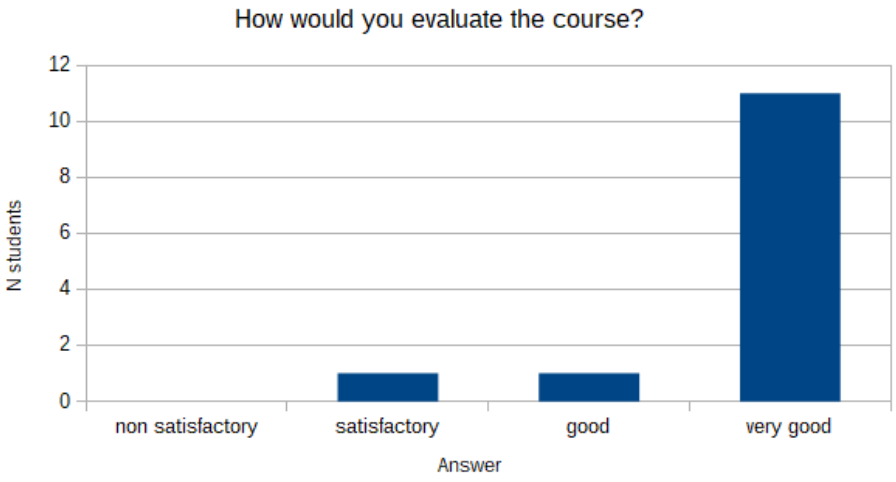

Figure 31: Feedback to the course in the end: overall grade to course.

Figure 32 shows, whether the students would recommend the course to a friend.

Figure 32: Feedback to the course in the end: recommendation to friends.

Figure 33 shows the

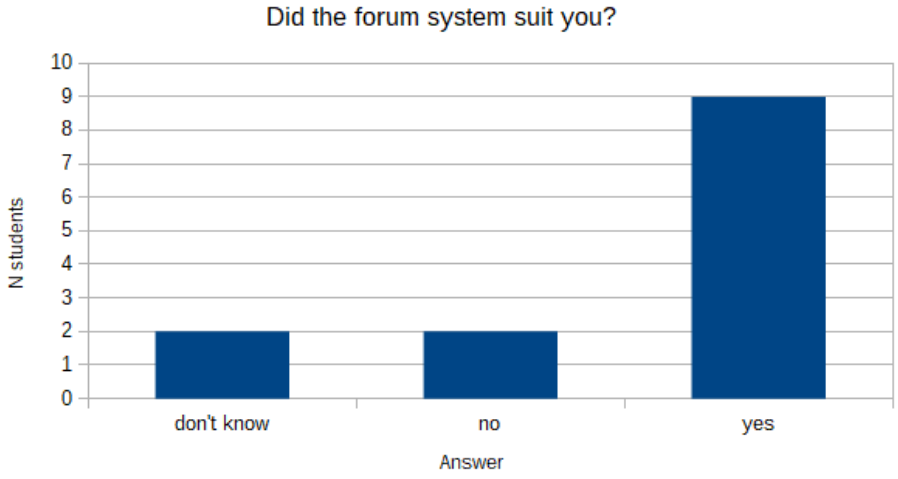

Figure 33: Feedback to the course in the end: forum system suitability

Figure 34 shows the

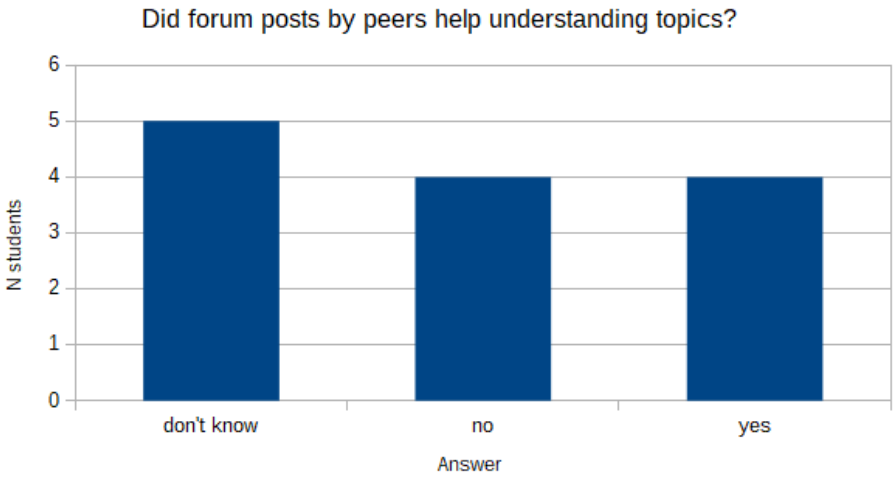

Figure 34: Feedback to the course in the end: support from forum posts by peers.

Figure 35 shows the

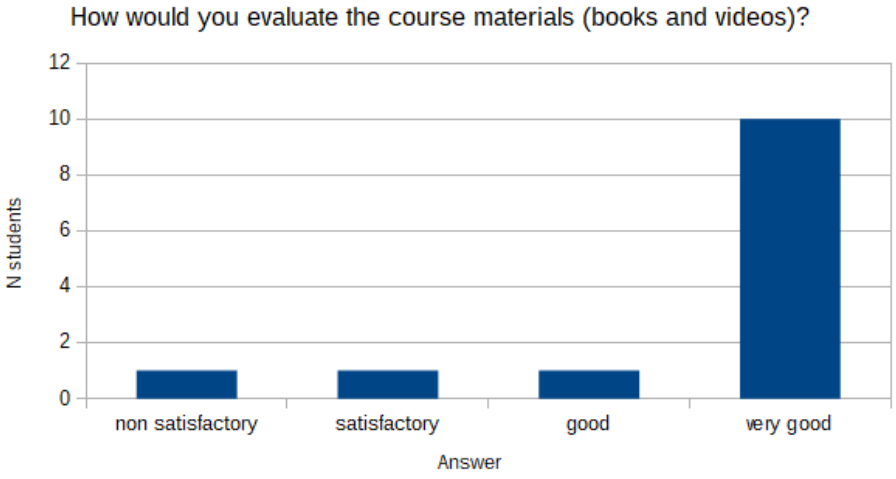

Figure 35: Feedback to the course in the end: evaluation to materials.

Figure 36 shows the

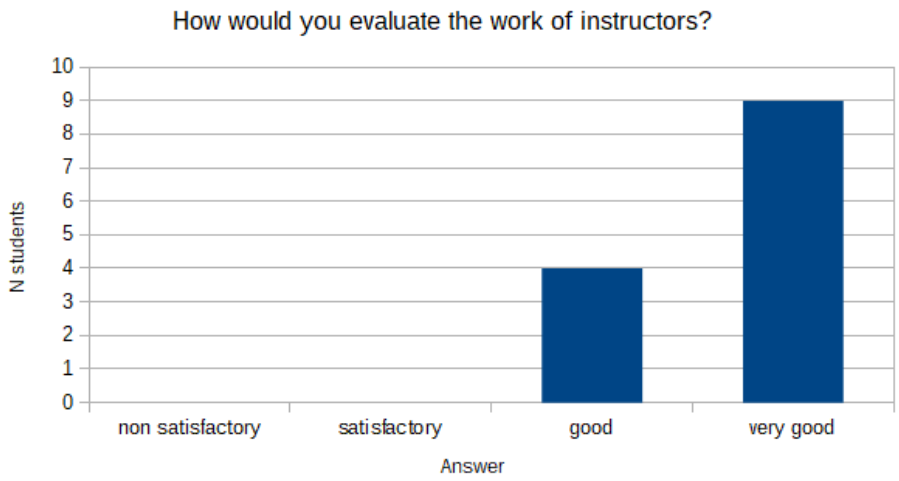

Figure 36: Feedback to the course in the end: work of instructors.

Figure 37 shows the

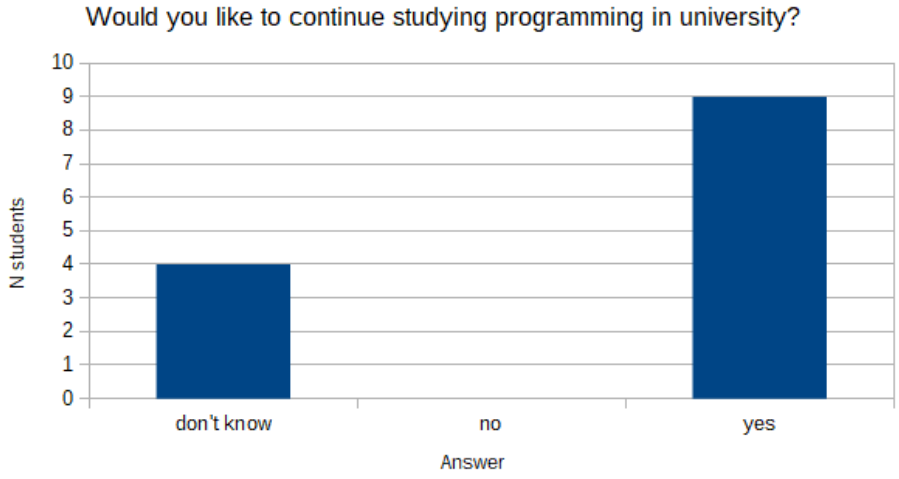

Figure 37: Feedback to the course in the end: student interest in continuing to learn programming.

It can be concluded that the students have a positive opinion about the course.

5.9 Further improvements

5.9.1 The second week

The most problematic part of the course could be the second week. This is supported by the feedback regarding the week. It could be beneficial to split the materials and move some of the topics to the next weeks. The concept of lists is only introduced in week 4. The concept of *for* loops is connected to lists. This means it could be considered to move the *for* loop topic to the fourth week. This could minimize the difficulty when understanding the *for* loops resulting from missing skills of using lists.

5.9.2 Conformity of materials and tasks

One aspect, which could use further work, is possible discrepancies between the exercises and tasks. This is supported by the fact that this was mentioned in the student feedback. As also mentioned earlier, the tasks and materials were simultaneously developed as the work of two theses. A Bachelors's thesis by Mark Muhhin produced a large amount of new tasks. The thesis in hand did all of the work on the materials and some new advanced exercises. This means further work could be beneficial to make sure all the pieces of materials and exercises are compatible.

5.9.3 Reaching students in difficulty

Further work could be done to make sure pupils get help, when they need it. This could either be done by trying to make use of a new communication platform or by utilizing the current channels of Moodle forum/private messages and e-mails and more effectively. This is most relevant when it comes to the period in the course, when projects are developed. There was a notable amount of students,

who had earned enough points for finishing the course from regular tasks, but did not submit a project. Such new communication platform could be chosen, where instant messaging is supported. This could make the communication less formal in the eyes of the student. This more direct communication approach was suggested in interview 2 as well. Another idea is for the instructor to be more proactive when communicating with students. This means instead of the student contacting the instructor, the instructor would try to detect when someone has been underperforming and would contact the pupil. For regular exercises this is less applicable. This is, because by the time the instructor sees someone having problems on a task, the deadline is already over. However, this is not the case with the projects. The project has two intermediate milestones. This includes choosing the topic and submitting intermediate progress. The right time for the instructor to proactively contact a student would be when work for either of these milestones is late or not submitted. In this case, the instructor can try to offer help before the deadline of the project is approaching.

6 Summary

The process of updating the "Let's make computer games" e-course was described. First, an overview was given about the state of the course before the work of this thesis. This includes the organization of the course, its target group and the contents of the course. The problems in the previous state of the course were identified and analyzed. The thesis then provided solutions for a number of the identified problems.

The solutions included changes in the tools used in the course. The IDE was changed from IDLE to Thonny. The versions of both Python and Pygame used in the course were changed to more recent releases. To avoid discrepancies in the materials and tools used, new videos were recorded to replace the existing videos. In total, 35 videos were recorded. The textual materials were updated regarding the technological changes. Several topics were covered in more detail - these include debugging and the game development concepts. New more complex exercises were added to the course, which introduced graphics at an earlier time starting from the first week. The grading system was changed to be more understandable.

Questionnaires were made during the course to assess the quality of the course. The results of the questionnaires show that the participants consider the quality of the course to be good.

Finally, I would like to thank both of the supervisors, Raimond-Hendrik Tunnel and Ljubov Jaanuska, for their excellent help. Also, a large thank you to all the participants of the course for taking part and giving their feedback!

References

[DP] Christian Dalsgaard and Morten Flate Paulsen. Transparency in cooperative online education. The International Review of Research in Open and Distributed Learning 10.3.

[DT] Marit Dremljuga-Telk. E-learning Quality Assurance System for E-courses in Estonia. Developing Quality Cultures in Teacher Education: Expanding Horizons in Relation to Quality Assurance.

[Kul12a] Tiina Kull. Kursuse "teeme ise arvutimänge - algus" 1. raamat olulised mõisted ja sissejuhatus. Tartu Ülikool, 2012. http://dspace.ut.ee/bitstream/handle/10062/25840/Arvutimangud_raamat1.pdf?sequence=1&isAllowed=y.

[Kul12b] Tiina Kull. Kursuse "teeme ise arvutimänge - algus" 2. raamat otsustused, hargnemine ja tsüklid. Tartu Ülikool, 2012. http://dspace.ut.ee/bitstream/handle/10062/25840/Arvutimangud_raamat2.pdf?sequence=2&isAllowed=y.

[Kul12c] Tiina Kull. Kursuse "teeme ise arvutimänge - algus" 3. raamat tsüklid tsüklites ja listid. Tartu Ülikool, 2012. http://dspace.ut.ee/bitstream/handle/10062/25840/Arvutimangud_raamat3.pdf?sequence=2&isAllowed=y.

[Kul12d] Tiina Kull. Kursuse "teeme ise arvutimänge - algus" 4. raamat funktsioonid ja objektid. Tartu Ülikool, 2012. http://dspace.ut.ee/bitstream/handle/10062/25840/Arvutimangud_raamat4.pdf?sequence=2&isAllowed=y.

[Kul12e] Tiina Kull. Kursuse "teeme ise arvutimänge - algus" 5. raamat graafika ja animatsioon. Tartu Ülikool, 2012. http://dspace.ut.ee/bitstream/handle/10062/25840/Arvutimangud_raamat5.pdf?sequence=2&isAllowed=y.

[Kul12f] Tiina Kull. Kursuse "teeme ise arvutimänge - algus" 6. raamat veel kasulikku ja heli. Tartu Ülikool, 2012. http://dspace.ut.ee/bitstream/handle/10062/25840/Arvutimangud_raamat6.pdf?sequence=2&isAllowed=y.

[Pyga] Pygame. Pygame download page archive. https://web.archive.org/web/20170101045510/http://www.pygame.org/download.shtml.

[Pygb] Pygame. Pygame download page archive. http://www.pygame.org/download.shtml.

[She] Esther Shein. Python for beginners. Communications of the ACM 58.3.

[VKDL] Veerasamy, Ashok Kumar, Daryl D'Souza, and Mikko-Jussi Laakso. Identifying Novice Student Programming Misconceptions and Errors From Summative Assessments. Journal of Educational Technology Systems 45.1.

[VKK+] Anne Villems, Ene Koitla, Kerli Kusnets, Lehti Pilt, Marge Kusmin, Marit Dremljuga-Telk, Merle Varendi, and Toomas Plank. Juhend kvaliteetse e-kursuse loomiseks. EITSA.

[W3S] W3Schools. OS Statistics. https://www.w3schools.com/browsers/browsers_os.asp.

[Ül] Tartu Ülikool. Programmeerimise õpik. http://progeopik.cs.ut.ee/.

Appendix

I. First interview transcript

Algus+installeerimisprotsess oli raske? Sh. Pygame. Algus oli päris hea. Materjalid aitasid kaasa. Vaja seletada, kuidas erinevaid asju kokku panna? Raskustase normaalne. Rohkem lahti seletada, mida installeerimine üldse tähendab.

Videomaterjalide kasulikkus? Videomaterjalide eelistatav osakaal? Videod olid väga kasulikud. Kui midagi kohe aru ei saanud, siis see aitas kaasa. Probleemne, et ei saa skippida vajaliku kohani.

Kas oleks põnevam kohe graafilist proovida? Enne tuleks selgitada läbi üldse arvutigraafika toimimismehhanismid. Teisest küljest tahavad mõned kohe tulla ja arvutimänge luua. Materjali osades luua paralleele – kuidas mängus realiseerida teemaga seonduvaid ülesandeid. Samas kohe graafika sisse tuues, võib tekkida õpilastel suurem motivatsioon ja parem ettekujutus, mille nad saavad lõpuks kokku panna.

Kõige raskem teema? Klassi mõistest oli raske aru saada. Mis see on ja mida teeb.

Kas klassi osa üldse välja jätta? Mängu realiseerimisel, on klassi mõte õige. On loogiline näiteks mängutegelased klassidena luua.

Kas oli piisavalt loovust proovile panevaid ülesandeid? Pigem ei meeldi ise asju välja mõelda. Eelistan ülesaindeid. Parem, kui on kindlalt defineeritud, mis peab olema lõpptulemus.

Tehnilise poole ja loovuse proportsioon kursuses? Oli palju asju ette antud. Probleem oli viimase projektiga. Ei tea, kust täpselt alustada. Oleks abi olnud, kui on mõned projektiideed ette antud. Et oleks parem arusaam, mis skoop peab projektil olema. Võiks olla mingi materjal, mis tooks välja vajaminevad etapid projekti tegemisel.

Projekti meeskonnatöö sujumine? Meeskonnakaaslane tegi graafika, ise koodi osa. Tekkisid raskused konkreetsete asjade implementeerimisel. Näiteks oli probleem collision detectioniga. Oleks mõistlik, et ülesanded on jagatud selliselt, et kõik peavd mõistma kõiki osi.

Rõhuasetus – programeerimise vs. mängudisaini õpetamine? Mängudisaini võiks veidi rohkem olla. Osakaal tundus selline, et programmeerimismaterjali vaadati detailselt üle, aga mängu tegemist seletati vähem. Pidi hakkama vähesema õpetusega mängu programmeerima.

Ajakulu Läks päris palju aega. Läks nädala peale ligi 7 tundi. Ülesanded ise ei võtnud väga kaua, kui materjal oli läbi vaadatud.

Võiks olla abstraktsemaid ülesandeid, kus lahenduskäik tuleb mõelda ise.

Lisamaterjalide vajalikkus? Kas oleks aega piisavalt? Võib olla huvilisi, kes tahaks ennast rohkem arendada. Pigem graafilise osa kohta. Graafika osa aitaks saada rohkem ideid lõpuprojektiks. Mõista paremini, kuidas kogu see asi toimib. Programmeerimise osa baas on juba päris hea.

Kas juhendajad saaks veel midagi teha, et õppija elu lihtsamaks teha? Koolis kipub olema selline olukord, kus õpetaja annab/räägib materjali ette, aga õpilased väga midagi ei küsi. On palju inimesi, kes ei julge küsimusi küsida ja jäävad hätta. Võiks kasuks olla iganädalane küsimustik, kus raporteerida probleeme.

II. Second interview transcript

Algus+installeerimisprotsess oli raske? Sh. Pygame. Pythoni seadistamne oli üsna lihtne. Teisi Java kursusega võrreldes oli lihtne. Ei olnud ka Pygame'ga probleeme. Oli hästi seletatud, kuidas asjad töötavad.

Videomaterjalide kasulikkus? Videomaterjalide eelistatav osakaal? Vaatasin enamasti siis, kui kirjalikust midagi segaseks jäi. Oli piisavalt videosid,

ei jaksanud alati kõiki ära vaadata.

Kas oleks põnevam kohe graafilist proovida? Ei oleks üldiselt eelistanud. Pigem meeldis mitte-graafiline osa.

Kõige raskem teema? Mängu tegemisel tsükli tegemine. Oli pikalt pusimist. Ussimängu näide oli üsna suur – ei viitsinud läbi teha.

Kas oli piisavalt loovust proovile panevaid ülesandeid? Jah, meeldisid ülesanded, kus sai midagi omalt poolt sisse panna.

Tehnilise poole ja loovuse proportsioon kursuses? Oli piisavalt loovaid ülesandeid. Seiklusmängus oli see eriti hea.

Projekti meeskonnatöö sujumine? Projekt oli koos oma kooli inimesega. Oli väga hea, kui sai otse arutada. Interneti kaudu oleks raskem olnud.

Rõhuasetus – programeerimise vs. mängudisaini õpetamine? Tasakaal oli üsna paigas. Mängude osa oli piisav. Olemasolevate materjalidega oli võimalik asi endale selgeks teha. Juurde oli vaja otsida selle kohta, et kuidas siduda erinevad programmeerimise valdkonnad ja ka mängudisaini sidumine.

Ajakulu Alguses läks vähem aega, kuna ülesanded olid lihtsamad. Kui see graafiline osa juurde tuli, siis võttis rohkem aega. Seiklusmängu välja mõtlemine võttis ka rohkem aega.

Kursuse alguses 2-4 tundi nädalas. Alguses pooleks, hiljem rohkem ülesannete peale. Hilejm kuni 6 tundi.

Lisamaterjalide vajalikkus? Kas oleks aega piisavalt? Pigem ei ole tarvis. Aega ei oleks tõenäoliselt ka piisavalt.

Kas juhendajad saaks veel midagi teha, et õppija elu lihtsamaks teha? Mingit sorti otsene suhtlus juhendaja ja õpilase vahel. Meili teel suhtlemise protsess võttis omajagu aega.

III. Student feedback forms

The feedback forms and answers from the course are in files attached to this thesis, in the "feedback" folder. The files contain the questions and answers without changes. These files include:

- Week one feedback form: feedback/w1-feedback.xlsx
- Week two feedback form: feedback/w2-feedback.xlsx
- Week three feedback form: feedback/w3-feedback.xlsx
- Week four feedback form: feedback/w4-feedback.xlsx
- Week five feedback form: feedback/w5-feedback.xlsx
- Project feedback form: feedback/project-feedback.xlsx
- Course feedback form: feedback/course-feedback.xlsx

IV. Updated materials

The updated materials used in the course are in files attached to this thesis, in the "textbooks" folder. These files include:

- Week one textbook: textbooks/1. ÕPPEMATERJAL - olulised mõisted ja sissejuhatus.html
- Week two textbook: textbooks/2. ÕPPEMATERJAL - otsustused, hargnemine ja tsüklid.html
- Week three textbook: textbooks/3. ÕPPEMATERJAL - tsüklid tsüklites ja listid.html

- Week four textbook: textbooks/4. ÕPPEMATERJAL - funktsioonid ja objektid.html
- Week five textbook: textbooks/5. ÕPPEMATERJAL - graafika ja animatsioonid.html
- Project example: textbooks/ProjectExample.pdf

Non-exclusive licence to reproduce thesis and make thesis public

I, Jaan Janno (date of birth: 30th of December 1991),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Introductory Game Development and Programming Course Materials

supervised by Raimond-Hendrik Tunnel and Ljubov Jaanuska

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 16.05.2017