



Aleksandar Kojić, BSc

Design and Implementation of an Adaptive Multidisciplinary Educational Mobile Game

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

Assoc. Prof. Dipl.Ing. Dr.techn. Christian Gütl

Dipl.Ing. Dr.techn. Johanna Pirker

Institute for Interactive Systems and Data Science

Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, December 2017



Aleksandar Kojić, BSc

Design und Implementierung eines Mobilen Adaptiven Multidisziplinären Pädagogischen Spiels

Masterarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium: Informatik

eingereicht an der

Technischen Universität Graz

Betreuer

Assoc. Prof. Dipl.Ing. Dr.techn. Christian Gütl

Dipl.Ing. Dr.techn. Johanna Pirker

Institut für Interaktive Systeme und Datenwissenschaft

Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, December 2017

Design and Implementation of an Adaptive Multidisciplinary Educational Mobile Game

Aleksandar Kojić



Advisers

Christian Gütl

Johanna Pirker

Graz University of Technology

Markos Mentzelopoulos

Daphne Economou

University of Westminster

Graz, December 2017

Abstract

This thesis describes the sCool project as a proposed solution for increasing motivation and engagement of high school and first-year university students with science, technology, engineering and mathematics (STEM). The project consists of the two main components which are the web platform and the multidisciplinary educational mobile game. In the web platform, instructors can prepare courses, create educational content and track student progress. Prepared educational content is afterwards loaded into the mobile game where students can select different STEM modules, learn theoretical concepts and practically apply gained knowledge. For the purpose of this thesis, programming is implemented as the first STEM module.

Educational content is presented to students in the form of missions and tasks where students can take either an active or an adaptive learning approach. The benefit for instructors is that they have an insight into the progress of every student and they can focus their time only on concepts that students failed to grasp. On the other hand, students will have fun and be able to grasp complex concepts through the video game without noticing that they are actually learning. In both theoretical and practical modes, dynamic game balancing (DGB) principles are utilized in order to keep players in the *flow channel* where the game is neither too challenging nor too easy. Microsoft Azure cloud platform and Unity game engine are the main technical components of the project.

The prototype of the system was tested by ten study participants from the Graz University of Technology, the University of Westminster, and the RMIT University. The study results show that the system satisfies the System Usability Scale (SUS) criteria with an average score of 84.25 and that happiness was the prevailing feeling on the Computer Emotion Scale (CES).

Kurzzusammenfassung

Diese Arbeit beschreibt das sCool-Projekt als eine vorgeschlagene Lösung zur Erhöhung von Motivation und Engagement von Schülern und Erstsemesterstudierenden für Wissenschaft, Technologie, Ingenieurwesen und Mathematik (STEM). Das Projekt besteht aus den zwei Hauptkomponenten, der Webplattform und dem multidisziplinären pädagogischen mobilen Spiel. In der Webplattform können Pädagogen Kurse vorbereiten, Inhalte erstellen und den Lernfortschritt verfolgen. Vorbereitete Lerninhalte werden anschließend in das mobile Spiel geladen. Dort können die Studierenden verschiedene STEM-Module auswählen, theoretische Konzepte erlernen und praktisch erworbenes Wissen anwenden. Für die Zwecke dieser Arbeit wurde als erstes STEM-Modul Programmierung implementiert.

Lerninhalte werden den Studierenden in Form von Missionen und Aufgaben präsentiert, bei denen die Schüler entweder aktiv oder adaptiv lernen können. Der Vorteil für Pädagogen ist, dass sie einen Einblick in den Fortschritt von jedem Schüler haben. Somit können sie sich auf Konzepte konzentrieren, die Studenten nicht verstanden haben. Auf der anderen Seite haben die Schüler Spaß und können komplexe Konzepte spielerisch erfassen, ohne zu merken dass sie tatsächlich lernen. In Theorie und Praxis, werden dynamische Spielausgleichsprinzipien (DGB) verwendet, um Spieler im „Flow Channel“ zu halten, wo das Spiel weder zu anspruchsvoll noch zu einfach ist. Microsoft Azure Cloud-Plattform und Unity Game Engine sind die wichtigsten technischen Komponenten des Projekts.

Der Prototyp des Systems wurde von zehn Studienteilnehmern von der Technischen Universität Graz, der Universität Westminster und der RMIT Universität getestet. Die Ergebnisse der Studie zeigen, dass das System die Anforderungen der System Usability Scale (SUS) Kriterien mit einer durchschnittlichen Punktzahl von 84,25 erfüllt und dass Freude das vorherrschende Gefühl auf der Computer Emotion Scale (CES) war.

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

Datum

Unterschrift

Acknowledgments

I would like to express my very honest appreciation to my advisors Assoc. Prof. Dipl.Ing. Dr.techn. Christian Gütl and Dipl.Ing. Dr.techn. Johanna Pirker. Their support, professional guidance, and great experience were important for the realization of the project.

Special thanks should also be given to my twin brother and colleague Miloš Kojić, who wrote his Master's Thesis parallel. He was responsible to create 3D models and implement procedural generation for maps, assets and audio. I want to thank him for the moral support and successful collaboration throughout the project.

For the technical and logistic support, provided input and a warm welcome at the University of Westminster, I want to thank Markos Mentzelopoulos and Daphne Economou. Furthermore, many thanks to the stuff and games students of the University of Westminster for the positive energy they shared.

Finally, I want to thank my family and friends for their enormous support throughout my studies at the Graz University of Technology.

Contents

| | |
|---|------------|
| Abstract | vii |
| Kurzzusammenfassung | ix |
| 1. Introduction | 1 |
| 1.1. Aim and Objectives of the Thesis | 2 |
| 1.2. Expected Outcomes | 3 |
| 1.3. Methodology and Structure | 3 |
| 2. Background and Related Work | 6 |
| 2.1. Learning Methodologies | 6 |
| 2.1.1. Behaviorism | 6 |
| 2.1.2. Cognitivism | 7 |
| 2.1.3. Constructivism | 7 |
| 2.1.4. Experiential Learning | 9 |
| 2.1.5. Collaborative Learning | 9 |
| 2.1.6. Mobile Learning | 10 |
| 2.1.7. Adaptive Learning | 10 |
| 2.2. Games in Education | 20 |
| 2.2.1. Play | 20 |
| 2.2.2. Defining Games | 21 |
| 2.2.3. Motivational Aspects of Video Games | 23 |
| 2.2.4. Video Games for Educational and Training Purpose | 25 |
| 2.2.5. Game-Based STEM Education | 26 |
| 2.2.6. Showcase of Programming Games | 30 |
| 2.3. Overview | 34 |
| 2.4. Mobile Trends | 37 |
| 2.5. Dynamic Game Balancing (DGB) | 37 |
| 2.6. Conclusion | 40 |

Contents

| | |
|---|------------|
| 3. Design and Conceptual Model | 43 |
| 3.1. Functional Requirements | 43 |
| 3.1.1. Mobile Video Game | 44 |
| 3.1.2. Web Platform | 45 |
| 3.2. Non Functional Requirements | 46 |
| 3.3. Conceptual Architecture | 47 |
| 3.4. Evaluation of Frameworks/Engines | 49 |
| 3.4.1. Mobile Video Game | 49 |
| 3.4.2. Web Platform | 51 |
| 3.5. Why Python? | 53 |
| 3.6. Game Heuristics | 53 |
| 3.7. Game Design | 54 |
| 3.7.1. Game Overview | 55 |
| 3.7.2. Level Design of Planet Exploration | 60 |
| 3.7.3. Level Design of Robot Missions | 63 |
| 3.8. DGB - Game Parameters | 67 |
| 3.9. Summary | 68 |
| 4. Implementation Details and Showcase Scenario | 71 |
| 4.1. Mobile Game | 71 |
| 4.1.1. Obtaining an Educational Content | 73 |
| 4.1.2. Theoretical Mode | 73 |
| 4.1.3. Practical Mode | 76 |
| 4.1.4. Dynamic Game Balancing (DGB) | 77 |
| 4.1.5. Multiplayer | 79 |
| 4.2. Web Application | 81 |
| 4.2.1. Development with the Entity Framework (EF) code- first approach | 83 |
| 4.2.2. Web Application Structure | 87 |
| 4.2.3. Web APIs | 89 |
| 4.2.4. Content Creation | 90 |
| 4.2.5. Course and Student Analytics | 91 |
| 4.2.6. Practical Implementation of the CAT Algorithm | 102 |
| 4.3. Summary | 105 |
| 5. Evaluation | 110 |
| 5.1. Participants | 110 |

Contents

| | |
|--|------------|
| 5.2. Setup and Procedure | 110 |
| 5.2.1. Computer Emotion Scale | 111 |
| 5.2.2. System Usability Scale | 112 |
| 5.2.3. System Specific Questions for the Web Platform | 112 |
| 5.3. Results | 112 |
| 5.3.1. Participants | 113 |
| 5.3.2. Computer Emotion Scale | 113 |
| 5.3.3. System Usability Scale | 116 |
| 5.3.4. System Specific Questions and Other Feedback by Participants | 117 |
| 5.3.5. Discussion and Limitations | 119 |
| 6. Lessons Learned | 121 |
| 6.1. Literature | 121 |
| 6.2. Implementation | 122 |
| 6.3. Outcome | 123 |
| 7. Conclusion and Future Work | 125 |
| 7.1. Conclusion | 125 |
| 7.2. Future Work | 126 |
| References | 130 |
| A. Questionnaire | 141 |
| A.1. Part 1 - Computer Emotion Scale | 141 |
| A.2. Part 2 - System Usability Scale | 142 |
| A.3. Part 3 - Previous Experience and System Specific Questions | 142 |

List of Figures

| | | |
|-------|---|----|
| 2.1. | Duolingo skill tree. | 12 |
| 2.2. | Different personalized learning pathways. | 13 |
| 2.3. | Graphical representation of different item characteristic curves. | 16 |
| 2.4. | Flowchart of the CAT algorithm (Thompson & Weiss, 2011). | 18 |
| 2.5. | CAT item administration. (Spronck, Sprinkhuizen-Kuyper, & Postma, 2004). | 19 |
| 2.6. | Folded up Streptococcal Protein Puzzle. | 27 |
| 2.7. | Comparison of online game subscriptions to degrees awarded across all STEM disciplines | 28 |
| 2.8. | Screenshot of the Colobot game (Colobot, 2017). | 31 |
| 2.9. | Crafting and testing spells on the surrounding landscape (CodeSpells, 2017). | 32 |
| 2.10. | Screenshot of the coding interface in the Code Combat (Code Combat, 2017). | 33 |
| 2.11. | Screenshot of the Swift Playgrounds (Playgrounds, 2017). | 34 |
| 2.12. | Preset list of commands and an interface adapted to the small-screen devices (SpriteBox, 2017). | 35 |
| 2.13. | Screenshot of smartphone users forecast by Newzoo (2017). | 38 |
| 2.14. | Csikszentmihalyi's flow model (Hunicke & Chapman, 2004). | 39 |
| 3.1. | Conceptual model of the sCool project with its core components. | 50 |
| 3.2. | Highscores table with names and achieved points. | 56 |
| 3.3. | Game story and user registration pop-up. | 57 |
| 3.4. | Screen for joining a course with the field for inserting a token | 57 |
| 3.5. | Skill tree loaded from the server and displayed in the mobile game | 58 |
| 3.6. | Knowledge level for the " <i>Basics(Programming)</i> " in theoretical and practical modes. | 58 |

List of Figures

| | | |
|-------|---|-----|
| 3.7. | Tutorial for the practical part which becomes shown when a player opens this mode for the first time. | 59 |
| 3.8. | Simplified state transition diagram of the theoretical mode. . . | 61 |
| 3.9. | Explorational part of the game with day and night modes. . . | 62 |
| 3.10. | An example of a task with the difficulty level of 50%. | 64 |
| 3.11. | An example of the wrong answer. | 65 |
| 3.12. | Practical part of the game where users can apply gained knowledge. | 66 |
| | | |
| 4.1. | The structure of the sCool project from a technology perspective. | 72 |
| 4.2. | JSON structure of courses and skill-trees. | 75 |
| 4.3. | The front-end implementation and the question structure with the corresponding components. | 76 |
| 4.4. | UI structure of the main components in the coding tool. . . . | 78 |
| 4.5. | Output mismatch in the practical task. | 78 |
| 4.6. | Dynamic map size based on player progress in the theoretical mode. | 80 |
| 4.7. | Disk positioning based on the level difficulty in the practical mode. | 81 |
| 4.8. | Entity Framework code-first approach. | 84 |
| 4.9. | Simplified database model of the sCool system. | 85 |
| 4.10. | Course listing in the web application. | 91 |
| 4.11. | Course listing in the mobile game and the option for joining a new course. | 92 |
| 4.12. | Displaying student details and managing the skill tree. | 93 |
| 4.13. | Displaying skill details and listing created practical and theoretical tasks. | 94 |
| 4.14. | The form used for creation of new theoretical tasks with front-end form validation. | 95 |
| 4.15. | The form for creation of a new practical task with front-end form validation. | 96 |
| 4.16. | Student activities per day and the list of students enrolled in the selected course. | 98 |
| 4.17. | Skill tree overview with the student's progress and activities. | 99 |
| 4.18. | Student statistics and activity details. | 100 |
| 4.19. | Different task difficulties based on student's knowledge level. | 106 |

List of Figures

- 5.1. Average ratings of all users per emotion with corresponding standard deviation values. 115
- 5.2. Results of the SUS per participant P1 – P10. 116

Listings

| | |
|---|-----|
| 4.1. Making web requests with the <i>UnityWebRequest</i> class. | 74 |
| 4.2. Enabling serialization of the course class. | 74 |
| 4.3. Multiplayer data synchronization. | 82 |
| 4.4. Entity Framework Code-First with class attribute annotations. | 86 |
| 4.5. Razor syntax with HTML helpers for the course editing . . . | 88 |
| 4.6. Rasch model function and difficulty scaling | 107 |

List of Tables

| | |
|---|-----|
| 2.1. Comparison of the analyzed programming games. | 36 |
| 5.1. Number of times an answer on the Computer Emotion Scale has been selected | 114 |
| 5.2. Number of times an answer on the System Usability Scale has been selected | 118 |
| A.1. Computer Emotion Scale (Kay & Loverock, 2008) | 141 |
| A.2. System Usability Scale (Brooke et al., 1996) | 142 |

Abbreviations

- AI** artificial intelligence. 30, 38, 39
API application programming interface. 46
AWS Amazon Web Services. 52
- CAT** Computer Adaptive Testing. 12, 14
CBL Computer Based Learning. 12
CBT Computer Based Testing. 14
CES Computer Emotion Scale. vii, 111
CoC Convention over Configuration. 52
CSS Cascading Style Sheets. 83
- DGB** dynamic game balancing. vii, xviii, 37, 40, 67, 107
DRY Don't Repeat Yourself. 52
- EF** Entity Framework. xviii, 83, 89
- HTML** HyperText Markup Language. 83
- IRF** item response function. 14
IRT Item Response Theory. 14
- MIT** Massachusetts Institute of Technology. 26
MVC model-view-controller. 52
MVT model-view-template. 52
- PDAs** Personal Digital Assistants. 10
- REST** Representational State Transfer. 81, 89
RoR Ruby on Rails. 52
RPG role-playing game. 31
- STEM** science, technology, engineering and mathematics. vii
SUS System Usability Scale. vii, 111

1. Introduction

Technology is influencing different areas of people's lives and rapid technological advancements are influencing the way how people learn, run their business and communicate. New technologies are reshaping the landscape of global economy which will require more people with the background in the STEM education. Natural sciences have many complex and abstract concepts which are challenging for students to grasp, so many of them feel discouraged, have poor performances and show little interest in these areas (Mji & Makgato, 2006). Many learning theories have been developed since Confucius until the present day, in order to understand and enhance the learning process, and most of them consider that learning occurs in a classroom with the support of an instructor. However, there are educational thinkers who claim that a closed classroom and an instructor should not be the mandatory requirement for learning to occur, so they developed theories of learning outside of a classroom (Sharples, Taylor, & Vavoula, 2010).

Mobile learning is a new concept created as a result of new technological advancements and availability of mobile devices to broad masses where studies have shown that mobile technologies can increase student engagement and motivation (Tremblay, 2010). New generations are growing up with video games, which became an interesting medium for presenting educational content and students find it appealing and intuitive to use. Digital game-based learning has the positive impact on educational effectiveness and student motivation as well (Papastergiou, 2009). A research by Pew Research Center (2017) states that 88% of high school students (13-17 years old) have a mobile device and 72% of them are playing video games. According to the analysis of the Newzoo (2017), in the period of 2017-2020, a number of mobile users will be increased by billion. Large-screen devices are supported by a number of educational STEM video games, but the need for mobile STEM games might increase as well, following the rapid growth

1. Introduction

of mobile trends. All these facts create a demand for new approaches to increase the motivation of students to engage more with the STEM disciplines and make the entire learning process more easier and fun.

1.1. Aim and Objectives of the Thesis

The purpose of this thesis is design, implementation, and evaluation of the sCool project which is designed for teaching STEM academic disciplines. The theoretical part aims to investigate problems in the STEM education, find potential solutions for increasing students' motivation, and facilitate the learning process. This task is conducted through three research stages: background of learning theories and contemporary learning approaches, motivational influence and effectiveness of teaching high-school and first-year university students (age of 11-19 depending on different educational systems in the world) STEM disciplines through an educational multidisciplinary mobile game, and exploring existing digital systems for STEM education. In relation to findings from the theoretical part, students' motivation and engagement with STEM disciplines should be increased by the pedagogical tool for STEM education based on the two main components:

- A web platform for instructors for creating courses and tracking students' progress
- An educational multidisciplinary mobile video game where prepared learning content will be integrated into and displayed in a form of tasks and missions

The video game is intended to be multi-platform, supporting both active and adaptive learning, with an implementation of dynamic balancing principles of the game difficulty and the difficulty of an educational content in order to keep players engaged. The first STEM module which will be implemented into the game, for the purpose of the thesis, is programming which is a complex and very important discipline in today's world. The video game is supported by the web application where instructors can prepare their own curriculum for STEM courses, create an educational content and track the progress of every individual student. By using statistical tools within the

1. Introduction

web application, instructors can invest more time for students who really need help and focus themselves primarily on students' weaknesses.

1.2. Expected Outcomes

The main expected outcome is that both students and instructors find themselves comfortable with the system and that the system is practically useful by helping students perform better in the STEM disciplines. The other goal is to provide this system, not only to the limited number of schools, but to broad masses, by publishing the game to the leading mobile marketplaces such as Google Play Store¹, Apple App Store² and Windows Phone Store³. As the game is going to be free of charge, students from all over the world, especially students from poorest countries, could have benefits from using it.

1.3. Methodology and Structure

This research is based on the case study, with the focus on the development of the web platform and the multidisciplinary educational mobile game. The project is realized at the University of Westminster (London, UK) in collaboration with (Kojić, 2017) who was researching procedural map, asset and sound generation. Agile software development principles (Awad, 2005) are used in this thesis since issues could be resolved as soon as they were identified with frequent interactions with students, lecturers, and stakeholders. Students are involved in testing the playability and engagement while instructors are testing the usability of the platform to ensure that the final product meets their needs.

The structure of this thesis is composed out of the three main chapters. Chapter 2 is the theoretical part which covers learning methodologies

¹*Google Play (2017)*

²*Apple App Store (2017)*

³*Windows Phone Store (2017)*

1. Introduction

and their characteristics. This chapter discusses the analysis of existing educational video games, their characteristics, and potentials of mobile growing trends. Chapter 3 and Chapter 4 covers the practical part where the conceptual model and the technical implementation of the system are explained. The technical implementation is divided into two parts which are the mobile video game and web application for instructors. Students are playing the game and using a content which is previously prepared by instructors. While playing the game, the game difficulty is dynamically adjusted based on the player's progress. The same thing happens with the educational content which is being adapted to student's needs. Instructors can track students' performances in real time, analyze their progress and help them with concepts which are challenging for them. Finally, Chapter 5, Chapter 6 and Chapter 7 cover the evaluation of the system, lessons learned during different research phases and suggestions for future work.

2. Background and Related Work

The traditional way of teaching is an effective approach to organize a lecture and track the state of the presented material, which is widely used today in schools and organizations, especially when instructors are working with larger groups. Nevertheless, the main drawback of this approach is that student's motivation has to be very high in order to appropriately and effectively follow the lecture. Natural science education is often described by students as a boring field with many abstract concepts which are challenging to grasp. Students have little interest in the STEM classes because of their complexity and students' discouragement when facing complex and abstract matter (Pirker, Riffnaller-Schiefer, & Gütl, 2014).

2.1. Learning Methodologies

This section discusses properties of different educational theories as well as learning approaches which became practically applicable with the increase of computer capabilities.

2.1.1. Behaviorism

Until 1950, behaviorism was the leading educational theory and an approach to understanding human behavior by considering the mind as a "black box", which can be manipulated by controlling external behavior (Cooper, 1993). Its primary focus is on environmental stimuli. The response to stimuli is strengthened, if it results in a reward. Based on an environmental stimuli, applying certain actions through reinforcement and repetition can lead to

2. Background and Related Work

gaining new knowledge where desired behavior is rewarded while undesired is punished. In behaviorism, learning can be defined as acquiring new behavior through social learning and conditioning (Schunk, 1996). Change in behavior defines behavioristic learning process, where the main person in a classroom is an instructor who decides what is right or wrong without providing the opportunity to students for self-evaluation within the learning process. Since external changes in behavior are considered as a basis, there is no space for emotions and internal behavior changes in this educational theory (Hung, 2001; Skinner, 1974).

2.1.2. Cognitivism

After 1950s, cognitivism became increasingly important, and advocates of cognitive theory believed that mind could not be considered as “black box” (Cooper, 1993). They argued that mental processes such as thinking, problem-solving and memory need to be understood first in order to define how exactly learning occurs. For cognitive theorists, the environment is not so important as it was for behaviorists, but individual learners with the focus on mental processes, building intelligence and cognitive development where attention is the first part of cognitive development (Hung, 2001). Cognitivists argued that thinking is so essential and the way people think impacts their behavior. In contrast to behavioristic approach where reinforcement is used to change the behavior, cognitivists use feedback for guiding mental connections (Ertmer & Newby, 2013).

2.1.3. Constructivism

In recent years, constructivism became the prevailing paradigm for instructional designers (Cooper, 1993). Constructivism is a philosophical view of how people learn, understand, know, and it has been built upon cognitivist principles. A French philosopher Jean Piaget was the person who formalized the constructivism learning theory. The emphasis of constructionism is not primarily on interaction with an environment, including other social beings, but on how the mind constructs the knowledge (Dewey, 1906). Even though

2. Background and Related Work

there are many different versions of what constructivism presents, the general opinion is that learning represents an active process of constructing the knowledge, rather than simply acquiring it from an external source. In such an environment, an instructor is a guide for students who are encouraged to learn by increasing their participation and being actively involved so they can build knowledge for themselves rather than simply accepting what they have read or heard from instructors (Laroche, Bednarz, & Garrison, 1999).

The constructivist theory is composed of many suppositions, but the philosophical view can be characterized in terms of three main suppositions. The first supposition represents a core concept and states that what is learned should not be treated separately from how it is learned. The second idea identifies that challenges and puzzles can contribute to increase in motivation and engagement. The final supposition stands that social collaboration plays a very significant role in developing individual understanding. Developing individual understanding, requires social collaboration as a crucial component. Other individuals can provide other views which can challenge thinking and develop knowledge even further, so these collaborative groups are necessary to expand and enrich understanding of different matter (Savery & Duffy, 1995).

In the research conducted by Tynjala (1999), an analysis of the efficiency of the learning environment in universities in comparison to benefits of a constructivist approach was performed. Tynjala argues that, in general, students at universities memorize and reproduce the acquired knowledge in a very similar way which one would do in a behaviourist learning environment. This is a serious issue as the knowledge is not transferable to real working environments and real-world problems, where problems are much more complex, diverse and where social factors can influence outcomes. This passive acquisition of information is completely different from the constructivist approach and real constructivist learning environment. Tynjala also found that students within a well-implemented constructivist learning environment acquired more diversified and transferable knowledge, and students that learned in this environment reported that the knowledge they acquired could be practically applied. Tynjala concludes that findings of this study suggest that educational outcomes much better fulfill requirements of real working situations and one of the reasons for this conclusion could be

2. Background and Related Work

found in better transferability of the skills acquired in constructivist environment (Tynjala, 1999). The social views of constructivism are more recently linked to Vygotsky as he emphasized the influence of cultural and social context on learning (Vygotsky, 1980). Based on his work about the importance of interaction with people for cognitive development, constructivism is called social constructivism.

2.1.4. Experiential Learning

Constructivists argue that students learn more effectively through self-exploration and by deducing conclusions by themselves. Experience plays the central role in the *experiential learning model*. The *experiential learning cycle* (Kolb, 2014) emphasizes the importance of active learning where learning takes place as a sequence of steps. Designers of digital learning environments found experiential learning theory as a very useful in their work (Lainema, 2014). Based on the ideas experiential learning provides, it is possible to integrate video games and pedagogy in a very effective way. Computers are able to provide interaction and feedback which is an important component in the experiential learning cycle. This is very beneficial for computer-based and game-based learning. Gee (2003) also argues that experiential learning cycle is present in video games as students are able to examine virtual worlds, make a reflection on the environments and situations they were in, make assumptions about certain happenings and plan next activities based on previous experiences.

2.1.5. Collaborative Learning

One of the main points of collaborative learning is having students collaborate, learn together, exchange knowledge, discuss ideas and develop communicational skills. Many studies have been conducted about collaborative learning and most of them found that collaboration has positive effects to the learning performance, which means that a group of students have a better chance to perform better than as individuals (Stevens et al., 2005). Through collaboration, students are developing critical thinking skills

2. Background and Related Work

which are crucial in learning. Besides critical thinking, students are developing creativity, clarifying opinions and understanding other individual learning styles and preferences (Bruffee, 1993; McConnell, 2000). Group-based learning is made by intertwining cognitive, motivational and social components where every component can support other component (Strijbos, 2004). One of the examples is that the cognitive component can lead to self-confidence, which can stimulate motivation leading to better collaboration and improvement of social skills.

2.1.6. Mobile Learning

As a result of new technological advancements in the mobile industry and availability of mobile devices to broad masses, mobile learning concept has been created. It relies on existing learning practices but the main differences are that learners are continually in motion and not in a limited physical space. Students are following their own pace and selecting topics by their choice, where learning is based on engaging with mobile devices such as mobile phones, Personal Digital Assistants (PDAs), smartphones and other. Adoption of these technologies is becoming more and more noticeable in further and higher education. Studies have shown that mobile technologies can increase student engagement and motivation (Tremblay, 2010). Mobile learning became very interesting for educators, due to the low cost of mobile devices and possibility to monitor and coordinate activities of users from different locations. However, designing such activities is the complex and challenging task for educators, which requires a lot of time and effort, in order to make effective online and distance learning environments (Kukulaska-Hulme & Traxler, 2005).

2.1.7. Adaptive Learning

A perfect school would be the one where every student could have a dedicated instructor, but that would require more instructors, time for testing and scoring, resources and data. The purpose of an educator is to help students master the most important content but the abilities and the

2. Background and Related Work

learning speed of every individual varies, which makes it very difficult for instructors to create a plan which fits perfectly to all of them. The new technological advancements and learning approaches are utilized to try to solve this issue. Adaptive learning is a contemporary learning approach which attracts more and more interest in the global educational community. It offers a very effective way of enhancing learning by adapting learning content to a user's knowledge level (Chen & Zhang, 2008). Adaptive learning is a branch of personalized learning, where the learning process can be adapted to a learner's needs and interests (Chen & Zhang, 2008). Adaptive learning is focused on the use of technology to personalize learning where the system monitors what users are doing and is capable of analyzing their actions. This results in providing the access to more relevant learning materials. It works in a way that algorithms adapt the process according to the given answers. If the user did not perform well, the system will provide more relevant content to the user or switch to different learning path. There are many systems which are using this educational and technological approach for supporting customized and individual learning.

Duolingo (2017) is currently one of the most popular language-learning platforms with more than 20 languages available to learn and 150 million registered users. The software tracks users' actions, and based on recorded data, plans future lessons, and includes tasks which fit student's knowledge level. A syllabus is presented on the main screen in a gamified skill tree structure that users can progress through (see Figure 2.1). Duolingo is not the only language-learning platform which is based on adaptive learning principles. Other popular platforms such as Rosetta Stone¹ or Babbel² use adaptive learning principles as well Kerr (2015).

Knewton (2017) is a company which created an adaptive learning platform for personalization of an educational content for the STEM education. The platform identifies each student's strength and weaknesses and provides educational content based on their knowledge level. The company presents itself as the world's leading adaptive learning technology provider. The platform offers courseware for fields such as technology and engineering, as well as enabling others to build adaptive learning applications. The

¹*Rosetta Stone* (2017)

²*Babbel* (2017)

2. Background and Related Work

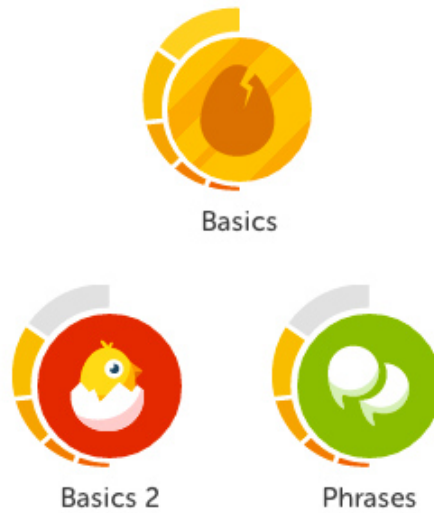


Figure 2.1.: A skill tree in Duolingo - organization of the course's skills with an overview of a course progress (Duolingo, 2017).

platform performs a sophisticated analysis of students' records and makes recommendations based on students' needs. There are different learning paths students can take, and potential student's flow through course material can be visualized with Knewton knowledge graph (see Figure 2.2).

DreamBox (2017) is a company which is developing mathematics adaptive learning technology. The company's focus is on mathematics education at elementary and middle school levels. Their software consists of a number of challenges, animated adventures and games in order to engage students in learning process.

Computer Adaptive Testing (CAT) Algorithm

Computer Based Learning (CBL) significantly enhanced education science. It can be even further enhanced with Adaptive Learning as an approach which adapts to user's ability level and offers appropriate learning content for every student (Linacre et al., 2000). In order to obtain the student's ability level and provide appropriate learning material, it is necessary to

2. Background and Related Work

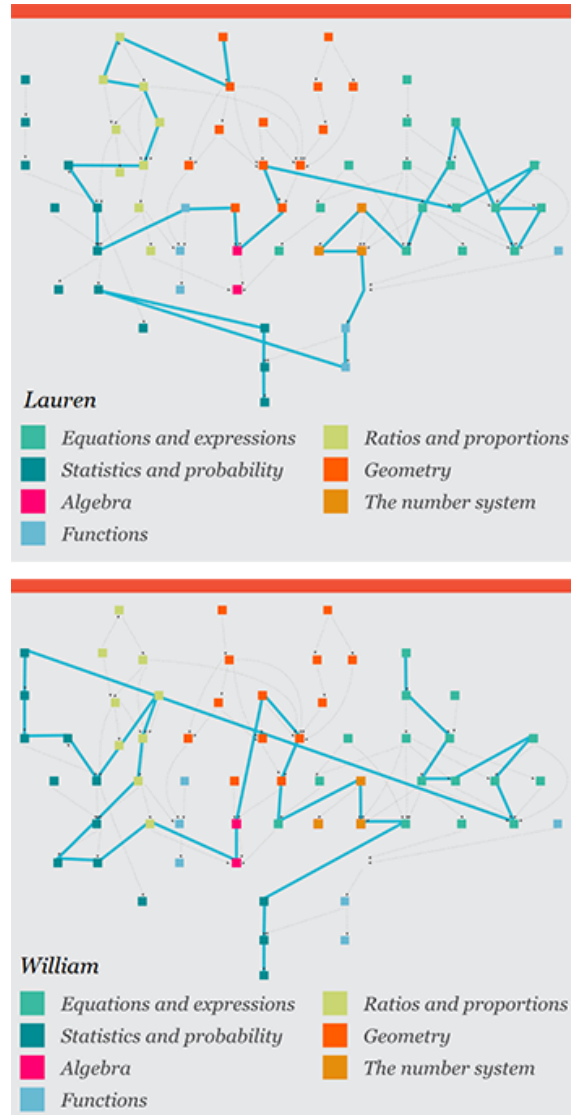


Figure 2.2.: Different personalized learning pathways (Knewton, 2017).

2. Background and Related Work

perform a test. There are different forms of testing, but the focus is placed on Computer Based Testing (CBT). Some CBTs are completely linear (Parshall, 2002) and they are the most similar to paper and pencil tests, because they have fixed size and order of items is predefined. After a user goes through all the question items and completes the test, the ability level can be defined. In contrast to linear tests, CAT has a high level of adaptivity, since they can adapt to each examinee. It can be defined as computer based test which is dynamically created specifically for each examinee based on examinee's ability level and previous responses.

When compared to linear computer-based tests, which are common in classrooms, CAT is better and more sophisticated method for examinations. It attracts attention of technical researchers for nearly 40 years (Lord, 1980; van der Linden & Glas, 2010; Wainer, 2000). In the past, it had many technical limitations, but with the advances of computers those barriers have been removed and it became more widespread. By using conventional tests, the test length is fixed, but the precision is not constant as all the participants have different ability levels. On the other hand, test length in adaptive tests is variable, but the CAT is designed to provide equivalent precision for all test takers with the constraint, that the item bank is properly designed.

CAT is based on Item Response Theory (IRT) which, in contrast to classical test theory, manages to place item characteristics and examinee's ability on the same continuum. As a consequence of IRT properties, item response function (IRF) (see Equation 2.1) can provide probability information of answering an item correctly for any ability level a person has and a difficulty level of the selected item (Lord, 1980). When student's ability matches item difficulty, the probability of obtaining correct answer is 50%. As student's ability becomes less than the item difficulty, the probability of answering correctly tends to 0%. In the opposite case when student's ability becomes greater than the item difficulty, the probability tends to go to 100%.

$$P_i(\theta) = c_i + \frac{1 - c_i}{1 + e^{-1.7a_i(\theta - b_i)}} \quad (2.1)$$

Equation 2.1 is composed of parameters where:

- c is pseudo guessing parameter

2. Background and Related Work

- a is a discrimination value
- θ is ability level
- b the is item difficulty
- e is the base of natural logarithm

Graphical representation of different item characteristic curves is shown in Figure 2.3. IRT models can be categorized based on different criteria. *Dichotomous* and *polytomous* IRT models are based on the number of scored responses. The dichotomous model has a resulting score with two states (correct/incorrect) while the polytomous model has more resulting states. IRT is a family of models which enables flexible use of the parameters. The number of used parameters depends on the type of the assessment and fit of the data. Rasch model takes only one parameter, the ability level of the user, and returns the probability of providing the correct response. Following IRT principles, this ability level is scaled to values from -3 to 3 and then the new value becomes the theta value which is further processed. For some cases the two-parameter model is used, where the first parameter is the ability level and the second one is the discrimination parameter. The discrimination parameter represents an index of how well the item differentiates low examinees from top examinees. Finally, if there are some items where there is reasonable rationale for learners trying to guess the answer, then pseudo-guessing parameter ($1/k$) is included in calculations where k is the number of possible options (De Ayala, 2013).

CAT has an architecture which is composed of five components and is presented in Figure 2.4 (Linacre et al., 2000; Thompson & Weiss, 2011; Weiss & Kingsbury, 1984):

1. **Calibrated item bank**

In this step, questions and content are defined together with their difficulty level. IRT item parameters are pretested using statistical analysis of examinees' responses. For testing purposes, the difficulty level of items can be set empirically but for production purposes, it is necessary to use examinees' responses for statistical analysis.

2. **Starting point**

A perfect scenario would be if user's ability level is already known from some previous tests, but in the case it is not known, the suggested

2. Background and Related Work

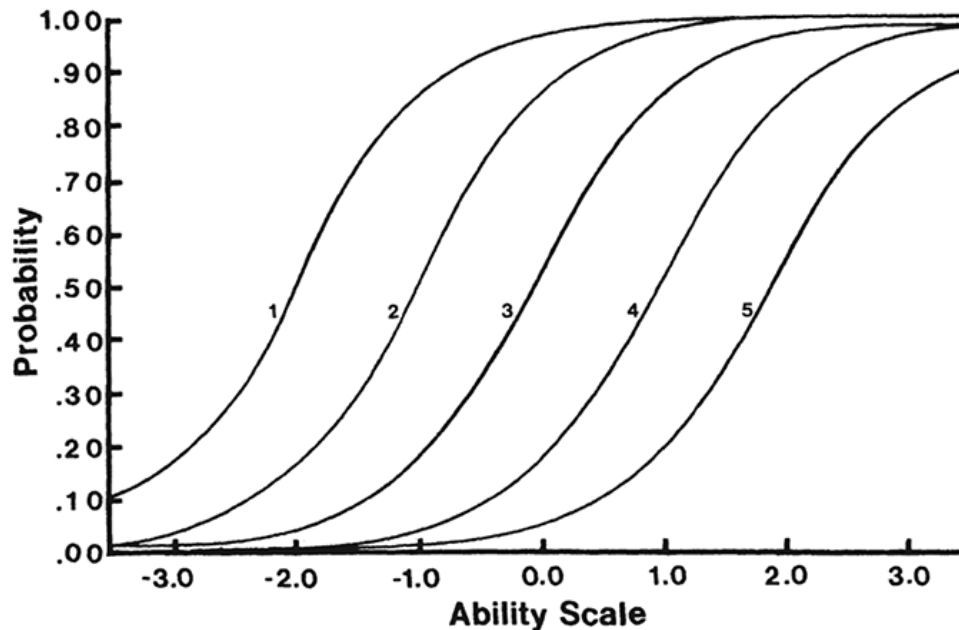


Figure 2.3.: Graphical representation of five item characteristic curves ($b=-2.0, -1.0, 0.0, 1.0, 2.0$; $a=.99, c=.00$) (Hambleton & Swaminathan, 2013).

difficulty level is 50% as it provides the fair and good starting point for testing.

3. Item selection algorithm

Selection algorithm works in a way that it finds the closest item in the item bank for the provided difficulty value. As some of the items might be recently exposed, an algorithm can try to find another item which was less exposed and which is still close to the desired difficulty level.

4. Scoring algorithm

As there are many question types such as multiple choice, single choice, text, numerical, matching or true/false, the corresponding scoring algorithm needs to be selected based on the question type.

5. Termination criterion

As a termination criterion different methods can be used: a fixed number of exposed items, user's ability level, standard error of measurement or considering some specifications of the item bank.

2. Background and Related Work

An example of a process of item administration is shown in Figure 2.5. Technical implementation of every component will be discussed more in details in the next chapter.

Challenges

Before implementing adaptive learning systems, it is necessary to take some technical, business and organizational consideration into account. Thompson and Weiss (2011) made several considerations that need to be clarified:

- Does the organization have the psychometric expertise or is it able to afford it if an external consultant is used?
- Does the organization have the capacity to develop extensive item banks?
- Is an affordable CAT delivery engine available for use, or does the organization have the resources to develop its own?
- Will converting the test to CAT likely bring the expected reduction in test length?
- Does the reduction in test length translate to enough saved examinee seat time, which leads to monetary savings?
- If CAT costs more and does not substantially decrease seat time, is that fact sufficiently offset by the increase in precision and security to make it worthwhile for the organization?

After considering these issues, implementing CAT and properly testing it, positive effects of CAT implementation will start to become visible.

Forgetting Curve and Spaced Repetition

Hermann Ebbinghaus presented the hypothesis of the exponential nature of forgetting which can be described by the curve that shows the decline of memory retention in time (Ebbinghaus, 1913). According to his hypothesis, humans are forgetting newly learned knowledge in a matter of days or weeks. In order to retain the knowledge in memory, it is necessary to review it over and over. Spaced repetition is a powerful learning technique for

2. Background and Related Work

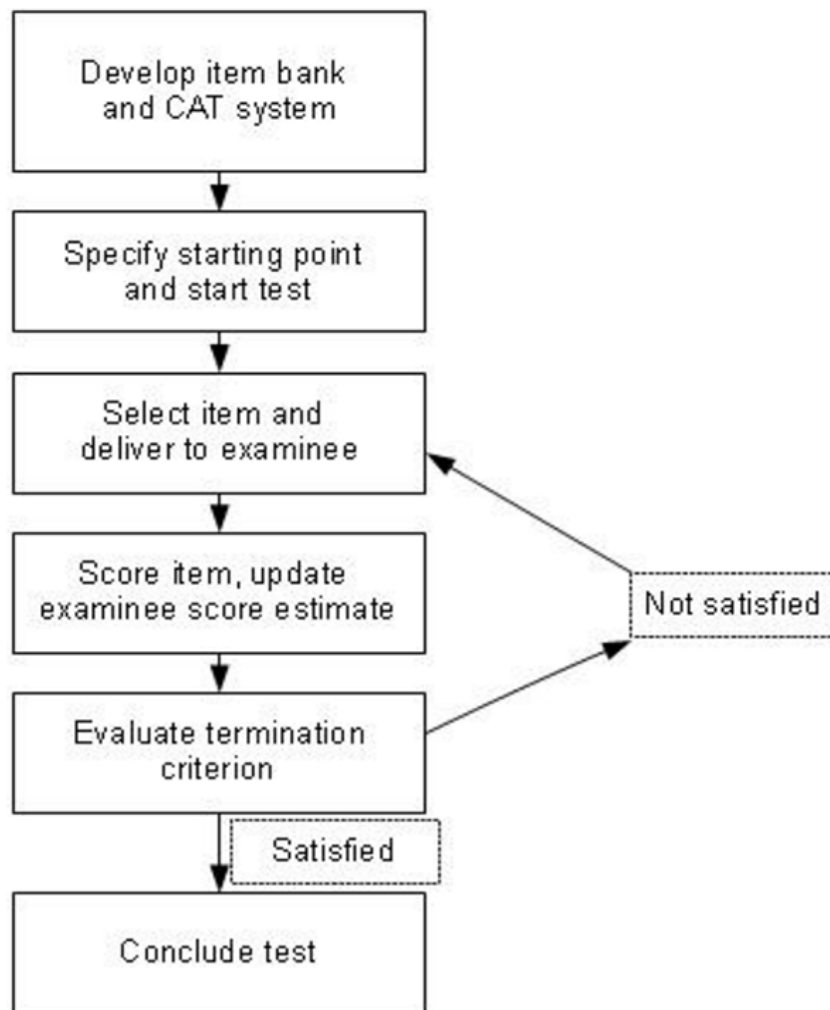


Figure 2.4.: Flowchart of the CAT algorithm (Thompson & Weiss, 2011).

2. Background and Related Work

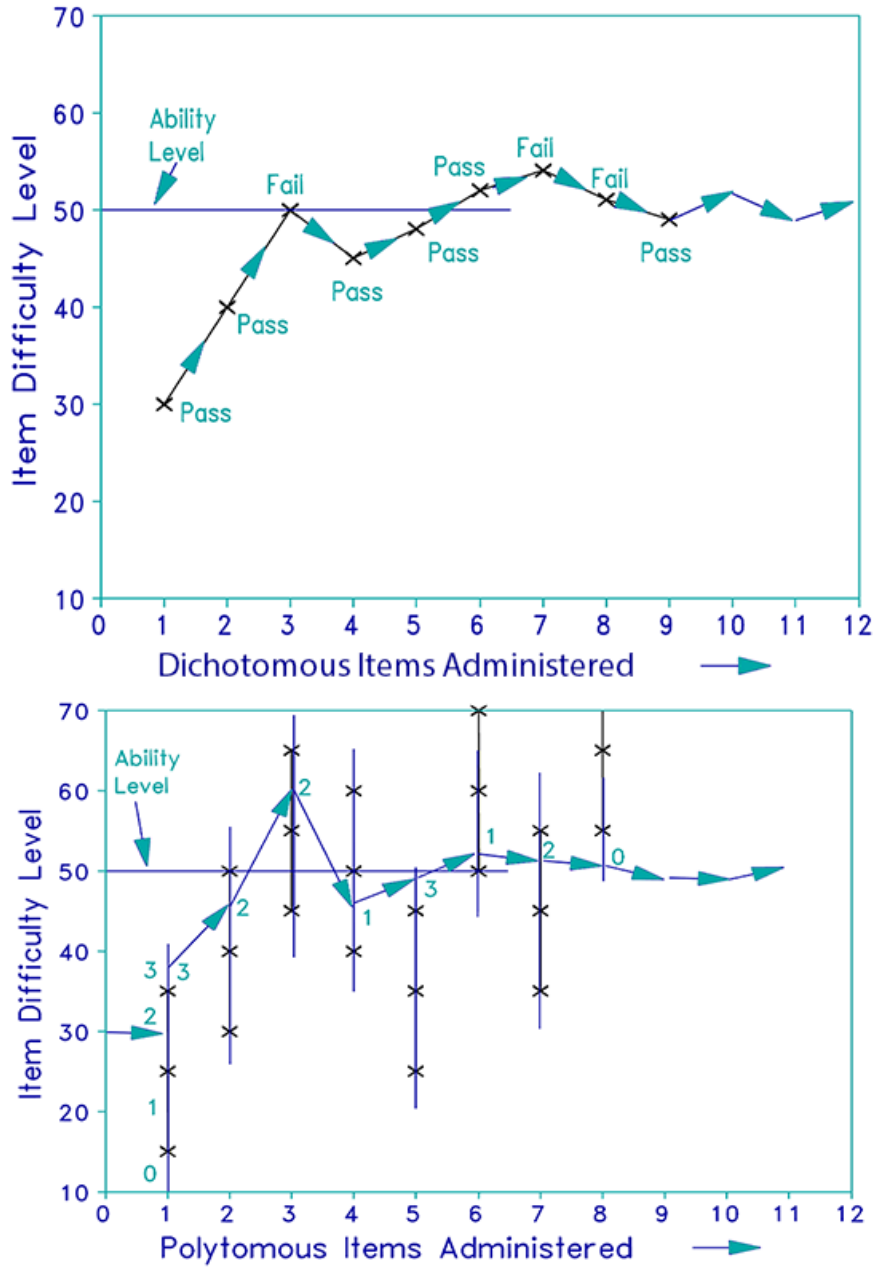


Figure 2.5.: CAT item administration. (Spronck et al., 2004).

2. Background and Related Work

keeping learned knowledge longer in memory where reviews are spaced in gradually increasing intervals (Settles & Meeder, 2016). This means that once students learn new concepts, the sCool system should update knowledge level over time and offer them those concepts to be reviewed again, according to the spaced repetition principles.

2.2. Games in Education

Computer and video games are attracting more and more attention of scientific community because of the influence video game industry have on students' lives in recent years (Squire, 2003). Motivational aspects and pedagogical potential of computer and video games have been studied in order to design enjoyable educational environments and facilitate the learning process. Video games are used not only for the entertaining, but for simulation and training purposes as well as they are capable of visualizing and simulating real-world processes and situations. These games are called serious games and they are used as complementary educational tools in different areas, such as physics simulations, flight simulations, healthcare or military purposes (Michael & Chen, 2005).

2.2.1. Play

Through the process of growing up, playing has a significant importance in young person's development, and this fact is sometimes overlooked. After observing children's behavior, some questions arise, such as why do they play, what can they learn when they play and how this experimentation and exploration of environment influences their further development? Play is strongly connected with early stages of life, but as people grow up, community treats play as something childish and perceive it as immaturity. Contrary to this supposition, it has been shown that play has a powerful influence on learning which makes it fundamental to the development of both children and adults (Rieber, 1996). One of the greatest minds in human history, Albert Einstein, said "*play is the highest form of research*". This scientist

2. Background and Related Work

emphasized the importance of play and its influence on experiments and research.

Play can be defined as a free and voluntary activity without any specific goal, a source of joy and amusement. Within the games, players are devoting themselves to the game spontaneously on their free will (Caillois, 1961). Even though playing video games leads to lack of physical activity or poor face to face social interaction, studies have shown that video games can be a powerful tool in education by developing intellectual and emotional skills in young people (Chang, Kuo, Chen, Hirose, et al., 2009; Crawford, 1984).

Students of age 11 to 19 are pursuing high school depending on a country. By entering high school, students start their professional specialization in certain fields. The high school period for teenagers is when they need additional motivation to overcome the challenges that educational matter in high school brings and this could be supported by video games as students are very familiar and comfortable with this medium. There is a wide variety of video games with the educational purpose which can support science. It is often thought that dealing with science requires many years of intense education, solving unsolvable problems behind the closed doors of laboratory and that it is unapproachable for a regular person. There are many examples where regular people in human history made the huge contribution to the science. Many of these solutions were very simple for some unsolvable problems.

2.2.2. Defining Games

There are many definitions in gaming literature of what term "*game*" means. It is also hard to define what the game exactly is as there are many different types of games. According to the gaming literature (Crawford, 1984; Oxland, 2004; Prensky, 2003; Salen & Zimmerman, 2004), an activity to be considered as a game should have components such as:

- Players
- Rules and constraints
- Quests and challenges
- Outcomes

2. Background and Related Work

- Goals
- Feedback
- Interface to the game world
- Competitions
- Representation of story

De Freitas (2006) defines computer-based learning games as "*applications using the characteristics of video and computer games to create engaging and immersive learning experiences for delivering specified learning goals, outcomes and experiences*". A genre is the French word for a type and games are categorized by various types. Wolf (2001) describes more than 40 game genres. Exploring different game characteristics is beneficial for understanding which game genre provides the best grounding for designing an educational video game. General classification which avoids subcategorization is extracted from gaming literature (Crawford, 1984; Oxland, 2004; Prensky, 2003; Schell, 2014) and is presented below:

- Role-play
- Platform
- Adventure
- Shooter
- Puzzle
- Strategy
- Sports
- Simulation

There is a number of models of game psychology that have been proposed, but the earliest and the most referenced is Bartle player types. This model is very simple one which classifies video game players as *Achievers*, *Explorers*, *Socializers*, and *Killers*. Killers interfere with the functionalities of a game world as well as actions produced by other players. Achievers are very competitive and enjoy beating difficult challenges. Explorers enjoy exploring the world, but not only that, they explore the game mechanics as well trying to explore game shortcuts and other tricks. Socializers are very interested in having relations with other players, instead of only focusing on the game itself. (Bartle, 1996). Bartle also found that players are not constantly in one of the primary categories, but they drift between others depending on their mood and status within the game.

2. Background and Related Work

Bartle player types are also used in *Gamification*. Gamification represents utilizing game mechanics such as points, leaderboards or badges in a non-game context to promote desired behavior (Deterding, Sicart, Nacke, O'Hara, & Dixon, 2011). The main difference between the games and gamification is that gamification uses game elements as a reward for completing existing systems while video games represent the system itself.

2.2.3. Motivational Aspects of Video Games

Video games became one of the most influential forms of entertainment in the world so many educators are interested in the research of what effect video games have on players and how those motivational aspects of video games could be used to facilitate learning. Motivation plays a key role in every aspect of human behavior which requires an action or intention to perform a physical or mental activity. It was challenging for cognitive science to define motivation, but one of the definitions is that learner is willing to make an additional effort to engage in a new age of learning (Gee, 2003). There are studies which examined the source of motivation in video games. Some attributed the compelling nature of games to their narrative context, while others find that motivation is connected to objectives and rewarding system within the game or to the intrinsic source because players enjoy the activity (Dondlinger, 2007).

Main concepts which make a video game interesting are a challenge, fantasy, and curiosity. After observation of some of the most popular video games, it was noticed that players are in control of their action, they have clear goals and objectives, the game difficulty fits players' ability level and they are provided with clear and instant feedback (Squire, 2003). Video games manage to maintain high levels of motivation and engagement in players by keeping them in the flow channel (see Figure 2.14). According to Csikszentmihalyi, players are in the state of optimal experience, where difficulty is neither too challenging nor too easy where frustration and boredom can be avoided.

After comparing video game players who are engaged in states of flow with students in traditional school environments, it was noticeable that students do not have much control over what they learn, they are passive

2. Background and Related Work

recipients of presented educational content, they have to adapt to a pace and an ability level of the whole group and very often they are getting imprecise or unresponsive feedback (Squire, 2003). Squire (2003) also discusses that motivational aspects of video games could be utilized for improving learning environments. With traditional teaching methods, it is challenging to make learning fun. The scientific community is more and more interested in the use of games and simulations to support learning, so enjoyable educational programs should support (Gee, 2003):

- Possibility for players to be in control of their actions
- Clear goals which are meaningful for students
- Clear feedback on their performance
- Defining difficulty levels and adjusting game difficulty to a learner's skills
- Random elements of surprise

After creation of a video game, the next step is to perform an evaluation. Brockmyer proposed the game engagement questionnaire for determining the measure of engagement in a video game-playing (Brockmyer et al., 2009). According to Gee (2003), *"motivation is the most important factor that drives learning. When motivation dies, learning dies and playing stops"*. Video games motivate players to persevere, keep going and step by step teach them how to play the game. As many students lack motivation and give up at the beginning of their courses, exploring key game components might reveal the secret of successful education. Besides engagement, exploring the game mechanics represents one of the main motivational forces.

There are different ways of learning based on different personalities and psychological effects that drive students and they can be categorized as cooperative, competitive, and individualistic learning (Johnson & Johnson, 1994; Vorderer, Hartmann, & Klimmt, 2003). In order to make an effective educational video game, all three components should be incorporated. People who are competition-driven will seek for competitive games where they can compete, try to achieve the best result and see themselves at the top of the high-score list. Individualistic people are not driven by social influence to achieve their personal benefits, while cooperation-oriented individuals try to facilitate learning process with the help of others and at the same time contribute to the benefits of others. As competitive games attract learners

2. Background and Related Work

with competitive preferences, they are not completely suitable for other two types of learners. One of the ways how they can be incorporated as well is that video game could have competitive groups (Burton-Chellew, Ross-Gillespie, & West, 2010). These groups can compete and fulfill competitive requirements but at the same time, individuals can work for themselves in the groups as well as together with others for the benefit of the group.

2.2.4. Video Games for Educational and Training Purpose

Video games can be designed for other purposes rather than for entertainment, enjoyment or fun. They have been used in different areas such as healthcare, education, military or engineering. Serious games represent using video game technologies in the educational game context where learners are placed in entertaining and comfortable environment (Mentzelopoulos, Parrish, Kathrani, & Economou, 2016; Michael & Chen, 2005).

Learning starts with observing the environment, school education, personal development or training and it can be supported by motivation. Learning may occur consciously or without conscious awareness. Concepts of constructivist learning environments are important for this study as these ideas are apparent in certain types of video games. Some researchers have discovered that well-designed video games follow constructivism principles (Dondlinger, 2007). Exploration and puzzle components in video games arouse mental processes in players such as thinking and problem-solving. There is a diversity of opinions which game genre is the best for educational purposes. Adventure and exploration games appear to provide a good foundation for the development of teaching resources (Oblinger, 2004; Quinn, 1994). The characteristics of adventure games are that they are story driven by exploration, solving puzzles and overcoming challenges. Skills required to play adventure games identified by students include logic, memory, visualization, and problem-solving. According to Betz (1995), computer games are enhancing learning through experimentation, visualization, and creativity of play.

A good example of experiencing real-world environments are virtual worlds, where players through immersion get the real notion and experience of the

2. Background and Related Work

environment, so video games can be a good context for problem-solving. Video games can have worlds and environments which users can explore. Once learners practice their skills in video games, they can apply their knowledge in real-world situations (Kalyuga, 2007). By using immersive virtual environments, real-life challenges can be even better simulated so the final result of the practice can be improved. Finally, a basis for the constructivist approach, which is learning from peers and collaboration, can be facilitated through multi-player video games so the overall experience can be brought to a higher level (Whitton, 2007).

There are citizen science games and projects where players play the most important role in solving indecipherable problems. One of them is the game called Foldit (2017) where the crystal structure of a monomeric retroviral protease has been solved by game players after many years of unsuccessful attempts by researchers (Khatib et al., 2011). Foldit (see Figure 2.6) presents the protein folding as an online puzzle game where thousands of people have taken part in the project and managed to make a small revolution in science by solving the problem and proving that capabilities of the unscientific community must not be underestimated. Eyewire (2017) is another example of citizen science game created in order to map the brain which is developed at Massachusetts Institute of Technology (MIT). There are many other gamified projects and platforms such as Zooniverse (2017) which gathers people to utilize their visualization ability for object classification as machines are still not powerful enough as humans to detect and separate very complex visual objects. Games can visualize the problem, provide a toolbox and environment for a citizen scientist to tackle it, where players are not even aware of the complexity of the problem. In fact, science problems and an educational material can be turned into a game where all these elements are carefully presented through a game story.

2.2.5. Game-Based STEM Education

Video games attract an enormous number of users and that number is much higher than a number of students which annually graduate with bachelor's degree in the STEM (see Figure 2.7), which motivated science educators to

2. Background and Related Work

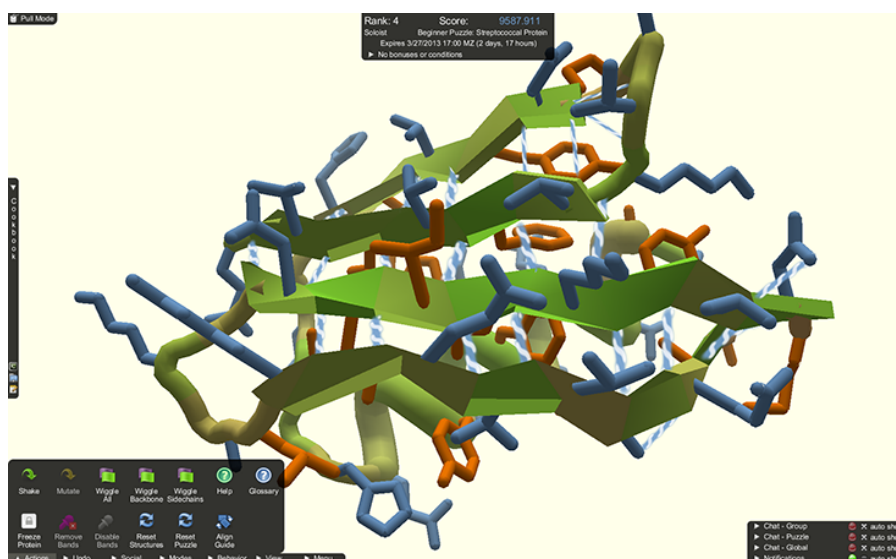


Figure 2.6.: Folded up Streptococcal Protein Puzzle (Foldit, 2017).

try to find solutions for having more students engaged with STEM disciplines (Mayo, 2009). Engaging students in complex and abstract phenomena of STEM disciplines is a very challenging task, but representing simulations, such as physics or chemistry simulations, through video game conventions could increase engagement and support deeper learning (Cordova & Lepper, 1996; Gee, 2003). Many science educators are utilizing video games as a tool for physics learning, because they believe that physics is the best taught through experiments and visualization rather than by mathematical formulae. Supercharged is an electromagnetism 3D simulation game where players are planning and conducting experiments by placing charged particles, observing their effects and exploring electromagnetic mazes (Squire, Barnett, Grant, & Higginbotham, 2004).

Websites such as Whyville³ are having millions of users playing science and mathematics games on the voluntary basis. Teenagers spend 5 to 8 hours per week playing games which surpasses the time they spend on their homework. To tackle this problem, B.D. Collier designed a racing car game for teaching numerical methods which resulted in twice the time

³Microsoft Azure (2017a)

2. Background and Related Work

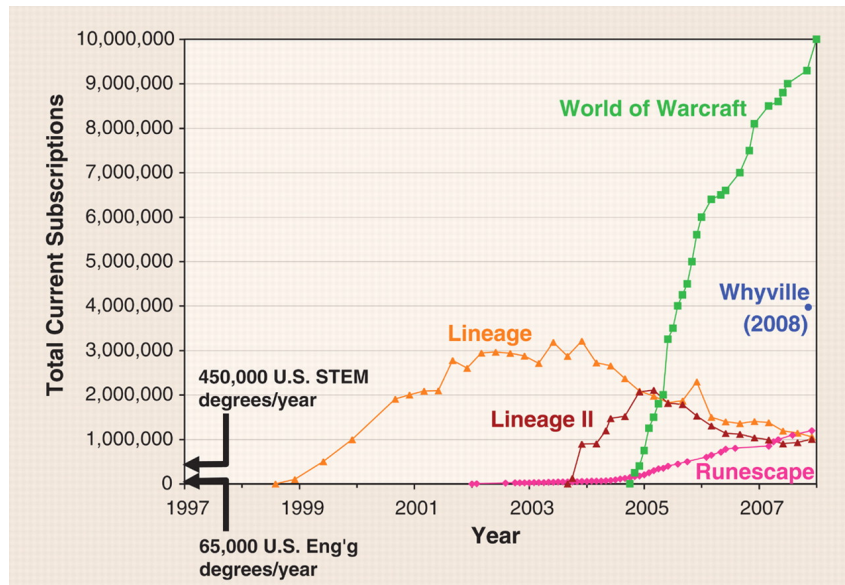


Figure 2.7.: Comparison of online game subscriptions to U.S. bachelor's degrees awarded across all STEM disciplines. Screenshot taken from (Mayo, 2009).

students spent more on homework (Mayo, 2009). There are other math games such as DragonBox⁴ which is a popular mobile game for teaching children algebra, where children are solving linear equations through the game without realizing that they are learning. The Circuit Warz project is using immersive virtual worlds and showing how they can be used to create a game based approach for teaching Electrical Engineering while supporting team-based collaboration. It shows hardware components in 3D world by simulating real laboratory environments (Callaghan, McCusker, Losada, Harkin, & Wilson, 2013).

Students study programming because it is necessary for various disciplines, as understanding of programming concepts requires strong mental activities and knowledge of mathematics and logic. Many students face difficulties grasping many abstract concepts which are present in these courses. Learning programming is a difficult process as it requires a combination of different cognitive activities from problem-solving to creativity. Papastergiou (2009) also argues that supporting students with additional, non-traditional,

⁴DragonBox (2017)

2. Background and Related Work

educational tools is important for reaching educational goals and increasing self-motivation in students. In order to make this process easier, researchers tested the influence of collaboration on this process.

Collaboration for problem-solving has been shown to be very useful as programmers can discuss the problem, exchange ideas and propose solutions based on their previous experience (Flor & Hutchins, 1991). Collaboration is not only useful for experienced programmers but as a good pedagogical approach for introductory programming (Čubranić & Storey, 2005; McDowell, Werner, Bullock, & Fernald, 2002). Programmers working in pairs express a higher level of confidence about their work, the programming process is more interesting and joyful which is very beneficial for problem-solving (Nosek, 1998).

The scientific community is constantly trying to find more effective ways of teaching students programming by moving away from traditional didactic teaching theories to learner-oriented models of learning. As video games are intrinsically motivating, they offer potentially powerful learning environments. Using this powerful fact and incorporating it into the education could lead to solving some important educational problems such as boredom, anxiety and lack of motivation. They increase interest and engagement even after the class, which is a good reason for introducing computer game-based learning (Dickey, 2005; Nawrocki & Winner, 1983; Oblinger, 2004; Squire, 2005).

In order to facilitate the process of teaching and learning of introductory programming, serious games could be used as they showed positive effects and outcomes in learning introductory programming (Kazimoglu, Kiernan, Bacon, & Mackinnon, 2012). Even though serious games show positive effects on teaching and learning, there is a lack of studies evaluating serious games as the highly effective approach which could strongly aid or even replace current teaching and learning methods (Sung et al., 2011). There are many block-based programming languages and environments for games creation, such as Scratch (2017) and Alice (2017). These tools teach students programming by hiding specific language syntax and exposing code elements as blocks through drag-and-drop interaction, but they are not games as such, because they lack some of the core game features such as timely feedback and rewarding system which motivate players to explore more.

2. Background and Related Work

2.2.6. Showcase of Programming Games

A number of studies used serious games to facilitate the learning process of introductory programming and some of these serious games will be discussed with their characteristics, programming topics, and target platforms.

RoboCode

RoboCode⁵ is a multi-platform desktop game built to support learning Java programming language. The goal of the game is to program artificial intelligence (AI) for a tank and have it fight in a battle arena against tanks programmed by other players. It is possible to program a robot to move, shoot and scan for the opponents. RoboCode provides development environment, built-in robot editor, and Java compiler. RoboCode community is large and diverse, and coding challenges are being held worldwide.

Colobot

Colobot⁶ is a real-time strategy 3D game with the goal to find a planet which could be colonized by humans (see Figure 2.8). A player needs to program vehicles, prepare necessary infrastructure and fight aliens which are endangering expedition. The vehicles are controlled in a way that users have to program them. The typed code is a pseudo-code which has similar syntax to C++ and Java.

CodeSpells

CodeSpells⁷ is a desktop game which teaches coding. Players craft magic spells and test them on the surrounding landscape by combining different elements such as fire, water, air, and earth. The game uses intuitive drag-and-drop block language to form different spells which can be later used and shared with friends. In the coding interface, users can prepare each spell and specify what exactly the spell will do. It became widely popular after very successful crowdfunding campaign and raising more than \$160.000 with more than 5000 backers.

⁵Robocode (2017)

⁶Colobot (2017)

⁷CodeSpells (2017)

2. Background and Related Work



Figure 2.8.: Screenshot of the Colobot game (Colobot, 2017).

Code Combat

Code Combat⁸ is a web-based role-playing game (RPG) with fantasy artwork where a player is controlling a character and fighting enemies by typing code (see Figure 2.10). This game offers a variety of programming languages to learn, such as *Python* or *Lua*. Besides general purpose programming languages, the game has web development levels as well. The game shows code errors and provides suggestions for how to fix them. Code Combat is free-to-play for all the core levels, but there is an option to buy additional levels and in-game currency. There are some other popular web-based programming games such as CodeMonkey⁹, CodinGame¹⁰ and Code Hunt¹¹. Code Hunt is a Microsoft's research project where they created serious gaming platform for coding contests and practicing programming skills. Players need to find solutions for puzzles and pass given test cases. It had over 350.000 players since 2016th when it was released (Code Hunt, 2017).

⁸Code Combat (2017)

⁹CodeMonkey (2017)

¹⁰CodinGame (2017)

¹¹Code Hunt (2017)

2. Background and Related Work

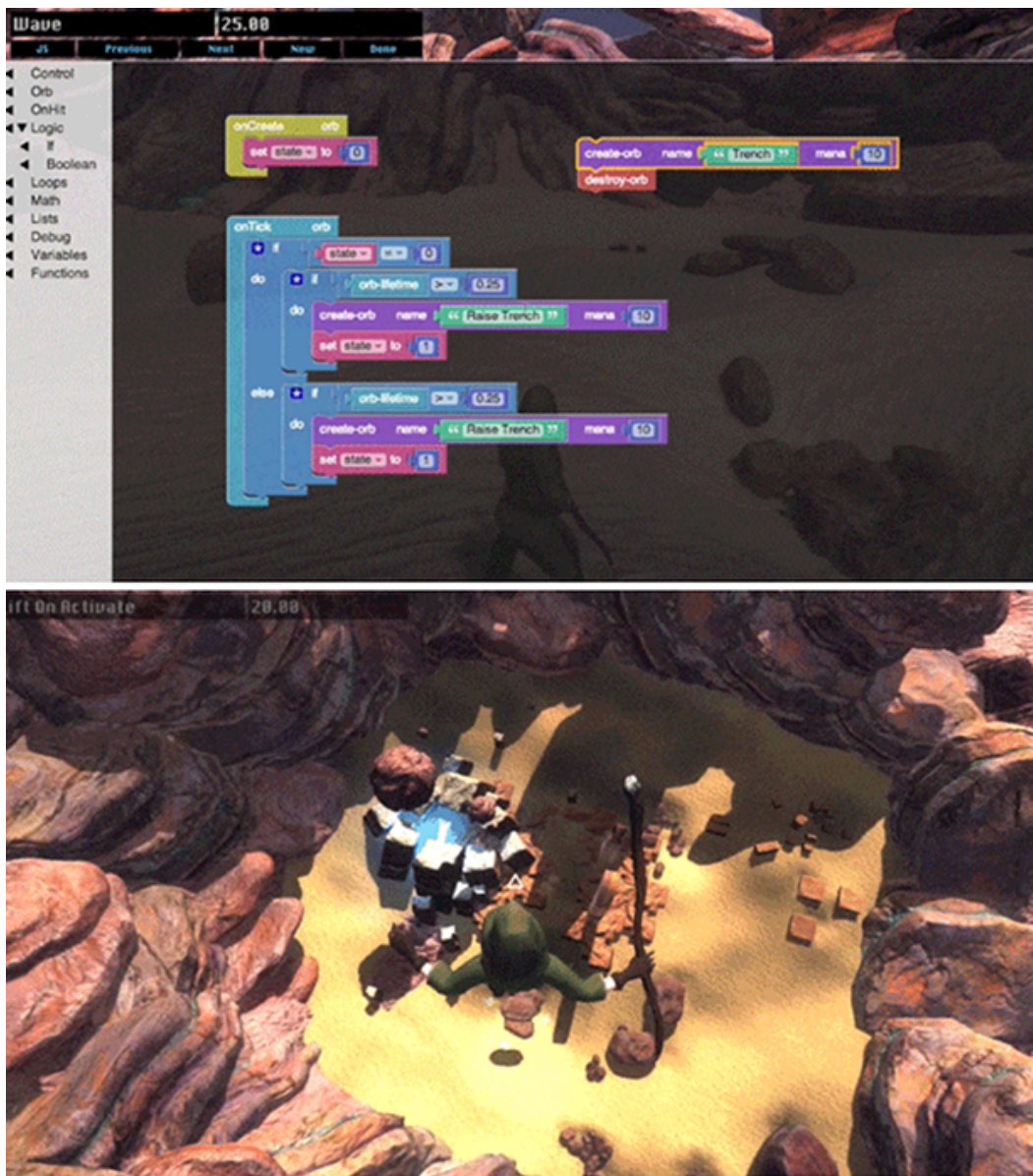


Figure 2.9.: Crafting and testing spells on the surrounding landscape (CodeSpells, 2017).

2. Background and Related Work



Figure 2.10.: Screenshot of the coding interface in the Code Combat (Code Combat, 2017).

Swift Playgrounds

Swift Playgrounds¹² is an iPad game created by Apple which teaches Swift programming language through a video game (see Figure 2.11). Each level represents one programming lesson. Users learn basic programming concepts such as variables, operators, loops, functions, and progress through levels to more complex topics and challenges. Users are provided with code box where they can type the code or drag and drop some predefined commands. The Character is being controlled With the combination of basic commands, loops and conditions. An overall feeling is that the game is visually appealing and easy to use. There are other games for tablets that teach basic programming concepts such as *Tynker* and *Box Island*. Browne from McMaster University developed tablet serious games for teaching introductory computer science concepts (Browne & Anand, 2013).

SpriteBox

SpriteBox¹³ is a puzzle-platformer mobile game with a mixture of exploration and learning kids to code. This game teaches basic principles of programming by allowing users to solve logical puzzles presented on screen and fill in missing parts of the code (see Figure 2.12). The game enables

¹²Playgrounds (2017)

¹³SpriteBox (2017)

2. Background and Related Work

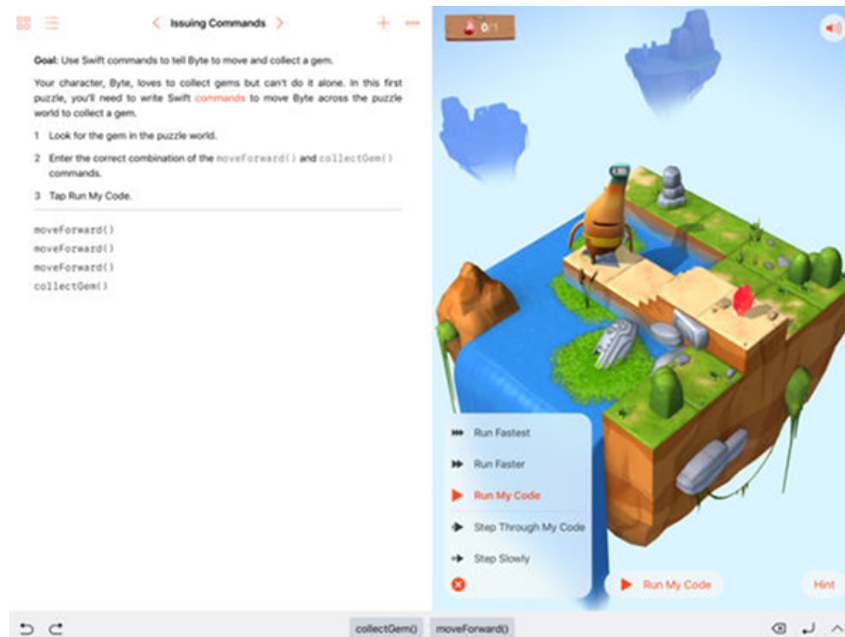


Figure 2.11.: Screenshot of the Swift Playgrounds (Playgrounds, 2017).

users to change parameters and debug faulty logic. There are other mobile games with similar purpose such as Hacked (2017), Lightbot (2017) and Pirates (2017). Hacked is Android game where a player is the main character who is performing hacking activities and saving the world by solving coding challenges in H programming language. Lightbot and Coding Pirates games do not expose code but teach programming logic through some predefined commands for movement.

2.3. Overview

After exploring existing educational tools and video games for teaching students programming, the conclusion is that majority of them are created for desktop devices, because the screen size is large enough and desktop devices represent a more convenient platform for presenting educational content than the screen size of mobile devices. An overview of analyzed

2. Background and Related Work

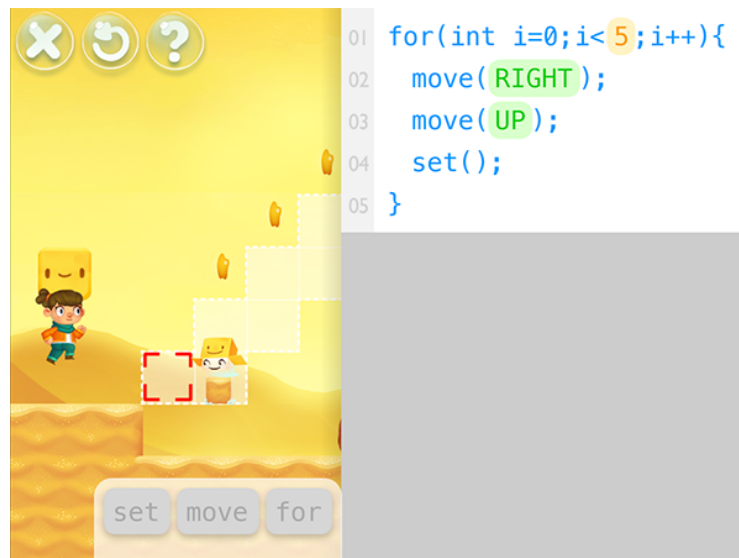


Figure 2.12.: Preset list of commands and an interface adapted to the small-screen devices (SpriteBox, 2017).

games and comparison based on the target device and programming topics is presented in Table 2.1. Desktop games such as *Code Combat* and *CodeSpells* are designed to support both simple and advanced educational concepts as they are attracting a large number of users around the world. On tablet devices, only Apple's *Playground Swift* game was sufficiently good designed and implemented to compete with desktop-based games. The screen size is sufficient for displaying educational content and the game enables users to follow guides, learn programming lessons and practice their skills by typing the code. At the time of writing, there are applications which are using gamification elements, but mobile video games which systematically teach programming and provide environments similar to real coding environments, are not in common practice. The problem might lay in the screen size of mobile devices, so a majority of developers are focused on large-screen devices. Many of them have already predefined curriculum which cannot be updated by instructors and which does not fit student's ability level.

2. Background and Related Work

| Title | Target Device/Platform | Programming Topics |
|---------------------|---------------------------------|--|
| Code Combat | Web | JavaScript, Python, Clojure, Lua, CoffeeScript |
| Code Hunt | Web | C#, Java |
| CodinGame | Web | JavaScript, Python, Ruby, Java, Swift, C#, Scala |
| Robocode | Desktop(Windows, Linux, Mac OS) | Java, .NET |
| Colobot | Desktop(Windows) | Java-like language |
| CodeSpells | Desktop(Windows, Linux, Mac OS) | Block-code, Programming logic |
| Swift Playgrounds | Tablet(iOS) | Swift |
| Tynker | Tablet(iOS, Android) | Programming logic |
| Box Island | Tablet(iOS, Android) | Programming logic |
| SpriteBox | Mobile(Android,iOS) | Programming logic |
| Lightbot | Mobile(Android,iOS) | Programming logic |
| Coding Pirates Game | Mobile(Android,iOS) | Programming logic |
| Hacked | Mobile(Android) | Programming logic, H language |

Table 2.1.: Comparison of the analyzed programming games.

2. Background and Related Work

2.4. Mobile Trends

According to the analysis of a Newzoo (2017), company which is publishing game statistics and market predictions, currently, there are more than a billion worldwide players. Pew Research Center (2017) states that 88% of high school students (13-17 years old) have a mobile device and 72% of them are playing video games. Newzoo presented global mobile market report where there are 2.6 billion smartphone users across the globe in 2017, where almost 50% come from China and India. Until 2020, there are going to be 3.6 billion consumers which means that almost every second person in the world will have a smartphone. According to Newzoo's estimations, there are 3.2 billion smartphones and 260 million tablets with more than 7,000 device models. Smartphone users forecast is presented in Figure 2.13, and in the period of only three years (2017-2020) the number of smartphone users will be increased by a billion.

These facts are the additional contribution to other studies which show that people are intrinsically interested in games, that they feel comfortable in this zone and that this important fact can be used in order to make educational process more interesting and less stressful at the same time (Oblinger, 2004). Placing more effective focus on mobile games research could lead to the overcome of many problems that are identified in present education.

2.5. Dynamic Game Balancing (DGB)

DGB has a significant role as it prevents players becoming bored or frustrated by making sure that the challenge level matches the skills of the human player. Adjusting certain parameters in a game, based on player's abilities, the game becomes more interesting and engaging. Selecting difficulty level at the beginning of the game can lead to frustration, so adapting the difficulty dynamically based on the player's ability is more complex but more beneficial approach. Software system needs to know when and how to intervene in order to maintain a natural game flow by keeping the player challenged and interested, which is a very difficult task. (Hunicke, LeBlanc, & Zubek, 2004). M. Csikszentmihalyi developed a flow model which is

2. Background and Related Work

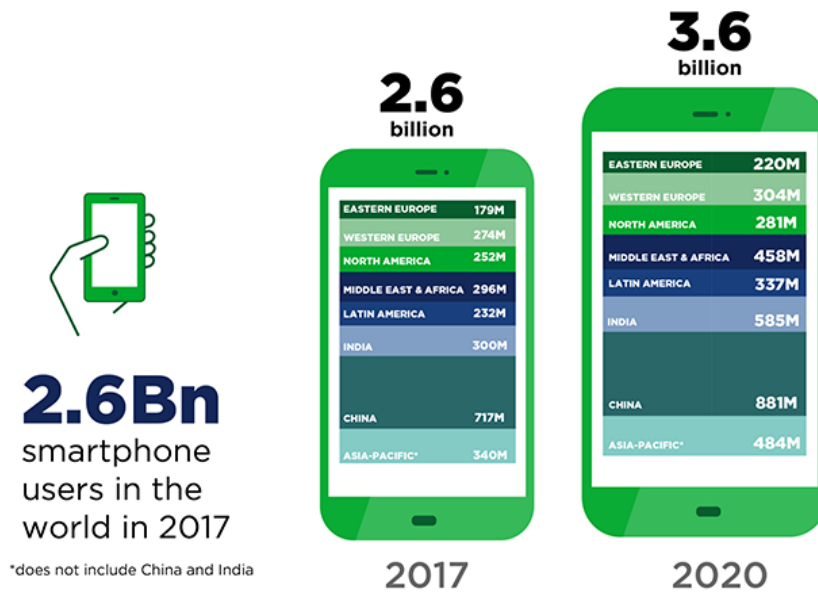


Figure 2.13.: Screenshot of smartphone users forecast by Newzoo (2017).

widely used today in order to keep the player in the flow channel. This means that player is being kept in the states where the game is neither too challenging nor too easy (see Figure 2.14).

There are different approaches in literature for addressing DGB and some of them are:

- Hunicke and Chapman (2004) suggest monitoring game parameters such as current health percentage, an equipment the player has, a number of opponent units surrounding the main player or their damage. Based on these parameters the system can decide should it help the player or make opponents stronger by adjusting the damage, decreasing the number of enemies or recovering health points faster.
- According to (Olesen, Yannakakis, & Hallam, 2008) racing games implement a very commonly used approach called elastic factor, which goal is to make a game more balanced. This means that AI is built in such a way that it keeps a computer car close to the player by keeping a short distance behind the player or overtaking the player and making the moment more challenging for the person who is playing the game.

2. Background and Related Work

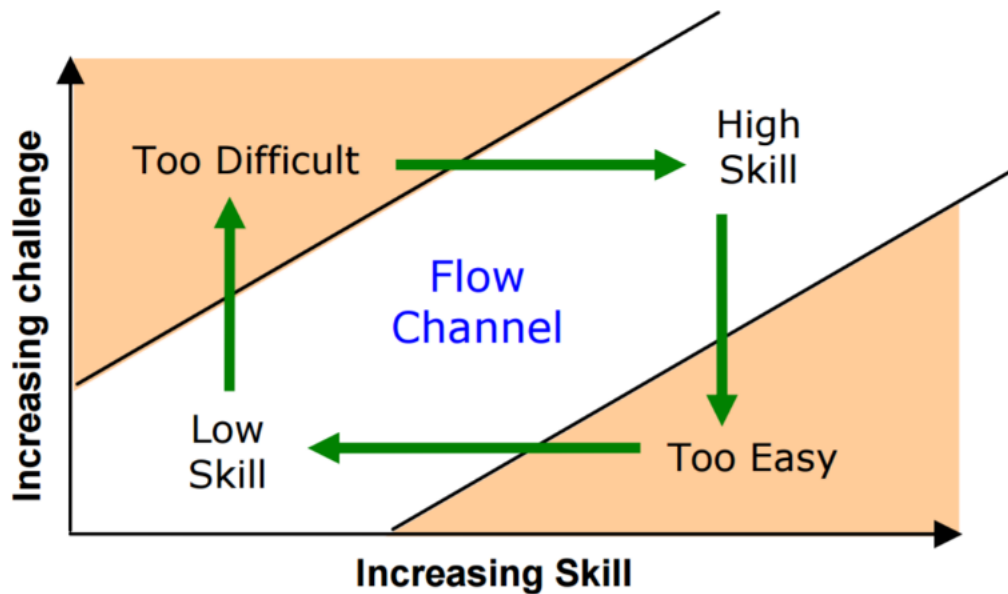


Figure 2.14.: Csikszentmihalyi's flow model (Hunicke & Chapman, 2004).

A key behavior of AI is that it needs to behave in the natural and realistic way, otherwise the players will realize what is happening behind the scene and the AI and game will become predictable.

- Modification of the behavior of the non-player characters. Using dynamic scripting to adapt game AI in order to get an even game with difficulty-scaling enhancements such as high-fitness penalizing, weight clipping and top culling. Machine learning techniques have shown an outstanding success in learning the behavior of human player and adapting difficulty level to them (Spronck et al., 2004). A poor AI for computer-controlled opponent leads to dissatisfaction of game players making them prefer human-controlled opponents, so in this case improving the game AI is a necessary step (Schaeffer, 2001).

2. Background and Related Work

2.6. Conclusion

Video games have shown a great pedagogical potential and this is one of the reasons why there are more and more game-based learning tools. Large-screen devices are already being used by broad masses in educational purposes and these devices have enough space for displaying different types of educational content, so the majority of educational tools for teaching both basic and advanced concepts are being dedicated to desktop and tablet devices. Even though mobile devices have increasing trends, there is an obvious lack of mobile educational tools for teaching more advanced concepts for high school and first-year university students. Some existing mobile games for learning programming are offering concepts of programming logic but the games which simulate real programming environment, have a web platform for content creation, adapt educational content and game difficulty to the player's needs, are not in a common practice. Therefore, there might be an increasing demand for more educational tools of this kind as the number of mobile users is rapidly increasing.

As video games offer more engagement than the traditional way of following a didactic lecture, utilizing mobile growing trends in educational purposes and presenting educational content for students through mobile games could increase motivation and engagement with the STEM disciplines. One of the possible solutions is creation of a multidisciplinary educational mobile game so that players can explore different STEM disciplines and learn following their own pace. In the beginning, it is important to confirm whether a mobile game can engage students in learning and whether it is possible to effectively present educational content on a small-screen device such as mobile phones. Therefore, the programming is selected as a starting discipline for the purpose of the thesis but the project can be easily extended to other disciplines as well. Students are becoming more motivated to learn to code if they are making a game, programming a robot or having similar activity so they can quickly see the output, real purpose of what they are doing and real potentials of this discipline.

All these facts create a demand for new approaches to increase the motivation of students. By utilizing game based learning, DGB, and the adaptive learning, the learning process of this complex discipline might become easier

2. Background and Related Work

and fun. Finding a good solution for some of the above-mentioned educational challenges could be very significant as the knowledge of programming is very demanding for wide variety of jobs in the present and future as well. Throughout smart and clever implementation, the game-based learning has a good potential to facilitate the process of STEM education.

3. Design and Conceptual Model

This chapter describes the design and the conceptual model of this project. Also, it discusses potential concerns which should be taken into consideration during the design process. After exploring existing video games for the STEM education, the conclusion is that most of them are dedicated to desktop devices. These devices are more convenient for presenting educational content because of the screen size. Also, educational video games for mobile devices are mostly teaching very basic concepts. However, there are increasing mobile trends and obvious lack of educational video games for high school and first-year university students, which are interested in learning more advanced topics.

Combining concepts such as the mobile STEM game-based learning, procedural generation, dynamic game balancing or adaptive learning into the project called *sCool*, can create a new value which is not in common practice. The *sCool* project consists of a multi-platform mobile educational game and a web platform where instructors can prepare their own curriculum, track the progress of their students and help students in needs for specific educational content which they had difficulties to grasp. This project follows industry standards and is inspired by some of the above-mentioned projects such as Duolingo (2017), Playgrounds (2017), Code Combat (2017) and Sololearn (2017).

3.1. Functional Requirements

Based on the findings and analysis of already existing systems, this section discusses the main requirements of the project. Some requirement such as educational video games or web platforms for creating educational

3. Design and Conceptual Model

content are already known features. The video game is the main component which students use and it simply has to be implemented together with the web platform where content can be created, updated or deleted and afterwards dynamically loaded into the game. Mobile games which are loading educational content from an external source are not basically a new concept, but there is space for improvements when compared to existing systems.

3.1.1. Mobile Video Game

Mobile video game components which should be integrated to support player engagement and motivation to learn, are given in the list below:

- **Educational Content** - loading educational content by presenting courses and course material in the form of levels. The advantage of loading content from an external source is that it can be constantly updated and improved. Afterwards, it can be cached and used from the local memory.
- **Active Learning** - active learning component relies on the educational content component by providing users the freedom to learn whenever they want, whatever they want and as much as they want.
- **Theoretical Mode** - enabling users to learn theoretical aspects of selected course material.
- **Practical Mode** - enabling users to practically apply gained knowledge of selected course material.
- **Feedback Mechanism** - students are able to see results of every action and progress in the game.
- **Social Feeling** - incorporating components such as friends, teams, messages, and comments in order to support collaboration, engagement, and motivation.
- **Procedural Generation** - generating an endless number of levels by using procedural generation algorithms so instructors do not have to spend their time on level design but on their courses. Every time students run the game, they will see a completely different level.
- **Dynamic Game Balancing** - balancing game difficulty based on player's progress in order to keep the players in the natural flow.

3. Design and Conceptual Model

- **Collaboration** - students are able to collaborate, exchange their ideas and knowledge and learn together.
- **Competition** - students are able to compete with other players and try to achieve more in order to be better positioned on the leaderboard.
- **Player Customization and Upgrades** - user can be more engaged by being able to customize some visual components of the player and, use items from inventory and to upgrade character's capabilities by using virtual currency.
- **Statistics and Logs** - tracking every user's action which includes user's taps, selected options, loaded screen name, player customization and inventory. These data are sent together with game and educational statistics to the web platform for further processing.

3.1.2. Web Platform

The purpose of the web platform is to enable instructors to create an educational content and to use analytical tools in order to help students which are not performing well.

Requirements for the web platform are:

- **Content Creation** - module for creating content which includes courses, categories, lessons and tasks.
- **Inviting Students** - instructors are able to create a course and invite students by sending them invitational tokens over email.
- **Analytics System** - instructors are provided with analytics tools so they can track the progress of the students and identify which concepts students find challenging to grasp. Some functionalities of the analytics system are: an overview of daily active students for the specific course, detailed analytics about player's activities, learning progress and provided answers.
- **Adaptive Learning Module** - every time a student plays a new level, server's adaptive learning mechanism will be triggered and it will retrieve a task which fits the best student's knowledge level.

3. Design and Conceptual Model

- **Web application programming interface (API)** - the mobile game is fetching lessons from web APIs and sending statistical data to server's access point.
- **Databases** - the database system for supporting the web application and storing player's statistics. These statistical data can be later used for further analytics and better understanding user's behavior and preferences.

3.2. Non Functional Requirements

Non-functional requirements play a significant role in software development process. The significance is even higher if the community is the target audience. Some of the most important non-functional requirements are:

- **Usability** - the system has to be user-friendly, simple to understand, and easy to navigate through. Regarding the video game, it definitely needs to be visually appealing with an enjoyable art style and clear user interface. The web application needs to be easy to navigate and use for the content creation. Usability should increase an overall user satisfaction and motivation by enhancing learnability, handling errors in a user-friendly way, and providing expecting results for user actions.
- **Configurability** - users should be able to manage the learning content without the assistance of an expert. The system should be configurable and able to support different functionalities without additional code changes.
- **Modularity** - the system should be able to adapt and support different courses, tasks, and other activities. This is necessary in order to keep users engaged over a longer period of time. Modules should be independent so they can be reusable and easily extendable.
- **Scalability** - the system has to be stable and capable of serving a large number of players.
- **Security** - contemporary security standards should be reached in order to provide secure operations, protect the data privacy, and reject potential attacks.

3. Design and Conceptual Model

- **Tutorials** - as every user needs some time to adapt to a new system, it is recommended to have a quick introduction how to use the system. By doing this, frustration can be avoided and users could be more motivated to continue using the system.
- **Multi-platform** - supporting different mobile platforms such as the iOS, Android or Windows Phone for reaching out more players.

3.3. Conceptual Architecture

Based on the derived requirements, a conceptual architecture model is designed. This conceptual architecture is the simplified representation of the main components and activities in the system. The simplified model is created for easier understanding of the project, where the Figure 3.1 shows an overview of system's most important components and activities.

The mobile game consists of the following components:

- **Player** - a player represents one of the main components in the system. Players are interacting with the mobile game, learning while playing levels, and competing with other players.
- **Educational Modules** - after all the educational content is obtained from the server, a player should be able to select one of the courses and play the game. Theoretical levels should have lessons and each of them should be followed by a quiz. On the other hand, practical levels should come with instructions and tasks. Users have to follow provided instructions and solve the given tasks. Depending on the selected course, corresponding game mechanics and user interface should be shown to the user. Some of the STEM modules that could be incorporated into the game are:
 - Programming
 - Electrical Engineering
 - Physics
 - Mathematics
 - Chemistry

3. Design and Conceptual Model

- **Other Modules** - modules which influence game characteristics and features:
 - **Game Mechanics** - the game should encompass different game mechanics such as moving the character, fighting enemies, controlling the robot, avoiding obstacles, and collecting disks.
 - **Procedural Generation** - students are able to explore different procedurally generated maps where enemies and other collectible items are dynamically positioned on the map.
 - **Dynamic Game Balancing** - component for dynamic game difficulty and content balancing based on student's progress and knowledge level.
 - **Collaboration** - collaboration component where students can communicate, exchange their knowledge and help each other.
 - **Competition** - component which drives and motivates students to perform better by having the best ranking on the highscores list.
 - **Active Learning** - enabling students to learn whenever they want and whatever they want by following their own pace.
- **Data Serialization** - serialization of the game statistics, learning data, and logs and sending these data to the server.
- **Loading Game Content** - obtaining data from the server, performing deserialization, and integrating obtained content into the game.

The web application consists of:

- **Data Processing** - data sent from the video game needs to be processed and stored for further analysis.
 - **Student Answers** - processing student answers and storing them in the database.
 - **Adaptive Learning Component** - every time a student plays a new level, server's adaptive learning mechanism will be triggered and student's knowledge level will be updated.
 - **Storing Logs** - logs about every player's action is stored for further processing and analysis.
- **Web Application & Database** - the web application for instructors with database system for supporting the web application and storing

3. Design and Conceptual Model

educational content and players' statistics.

- **Content Creation** - this module enables instructors to create courses, lessons as well as practical and theoretical tasks.
- **Analytics System** - instructors are provided with analytics tools so they can track the progress of the students and identify which concepts students find challenging to grasp.
- **Highscores** - highscores component for managing highscores list.
- **Web APIs** - the mobile game is fetching lessons from web APIs and sending statistical data to server's access point. These APIs include user registration, obtaining courses and educational content, processing theoretical and practical tasks, storing game statistics, highscores, and joining a course.

3.4. Evaluation of Frameworks/Engines

The first and the most important technical decision is to select a proper environment or framework. Before making this decision, it is necessary to make an analysis of environments and frameworks which can satisfy defined project requirements.

3.4.1. Mobile Video Game

As the goal of this project is to create the multi-platform mobile game and try to reach as many players as possible, an appropriate technology which can support this requirement needs to be chosen. There are several important criteria such as how easy it is to learn and work with, what are the capabilities of certain technology, and finally the pricing as this project is indented for educational purposes. Some of the most popular game engines are shown below.

- **Unity3D¹** is one of the most popular game engines which supports both 2D and 3D game development and exporting a game for different

¹Unity (2017)

3. Design and Conceptual Model

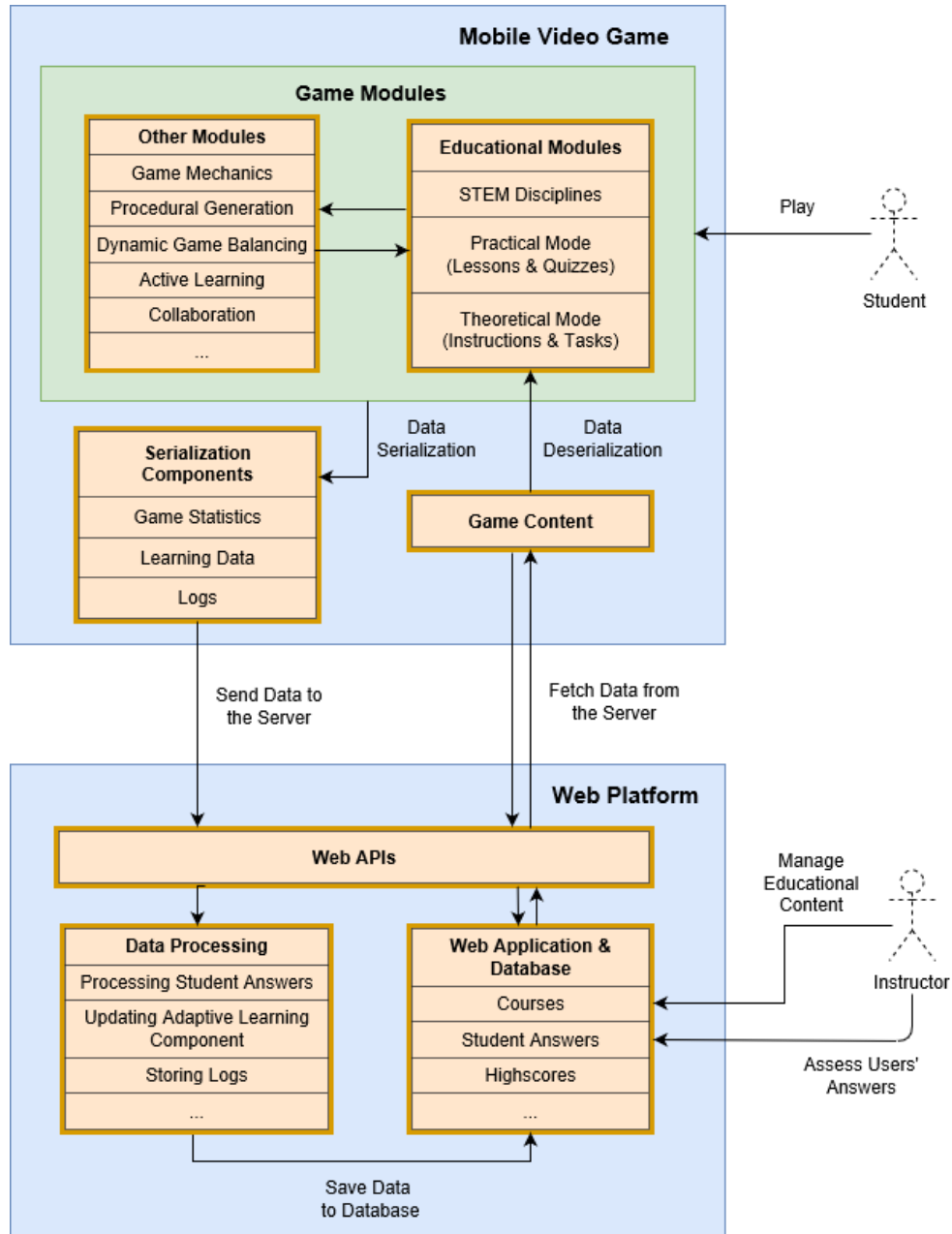


Figure 3.1.: Conceptual model of the sCool project with its core components.

3. Design and Conceptual Model

platforms. It offers a number of features and its interface is easy to work with. Unity3D is the free game engine but there is a *Pro* version with more advanced features. It supports multiple languages like C#, UnityScript, and Boo.

- **Unreal Engine 4 (UE4)**² has very advanced dynamic lighting and particle system which can handle thousands of particles at one time. It can be utilized to create games with photo-realistic graphics. The scripting language in UE4 is C++ and UE4 has a steeper learning curve so it might not be a perfect solution for new game developers. This game engine is free until the game earns a certain amount of money when the developers have to pay for a license.
- **Construct2**³ is a very good solution for developers who do not have much experience and it is not suitable for larger and more complex projects. It is a free tool which is based on the concept of "events" and "behaviors" and enables multi-platform export. It comes with a number of predefined mechanics and features where users can modify parameters such as speed or attack speed based on their needs.

As Unity3D is the leading game engine with a large number of supporting features and the authors of this project are very familiar with the Unity3D game engine, which current version is 5.6 will be used as the main tool for video game creation in this thesis.

3.4.2. Web Platform

There are different technologies and frameworks which can be used to develop the web application with analytics tools. Some of the most used are:

- **ASP.NET**⁴ - ASP.NET is an open source web framework which is used for building web applications and services with .NET platform. Also, it satisfies contemporary industry standards, and by using ASP.NET,

²Unreal Engine 4 (2017)

³Construct 2 (2017)

⁴ASP.NET (2017)

3. Design and Conceptual Model

it is possible to create scalable systems which are capable of serving a large number of users.

- **Django**⁵ - Django is a high-level Python Web framework with the focus on rapid development, reusability of components and clean design. Django is free, open-source and follows model-view-template (MVT) architectural pattern.
- **Ruby on Rails (RoR)**⁶ - Ruby on Rails is a server-side web application framework which is written in Ruby and uses Rails as a model-view-controller (MVC) pattern to organize application development. It offers default structures for different components and a number of plugins created by the community. Ruby on Rails is emphasizing principles such as Convention over Configuration (CoC) and the Don't Repeat Yourself (DRY).
- **Express**⁷ - Express is fast, minimal, and flexible web framework for Node.js. It provides a number of functionalities for web and mobile applications while the process of creating robust APIs is quick and easy. Express is highly scalable and provides good performances.
- **Laravel**⁸ - Laravel is a free, open-source PHP web framework. It is based on MVC architectural pattern and comes with dedicated dependency manager and other utilities that facilitate application deployment. Currently, it is one of the most popular PHP web frameworks.

After the completion of the project, it needs to be hosted on a platform which fulfills above-mentioned requirements. The three main cloud platforms are Amazon Web Services (AWS)⁹, Google Cloud¹⁰, and Microsoft Azure¹¹ and all of them satisfy the needs of the project. As the author of the project is very familiar with the ASP.NET web framework and the Microsoft Azure cloud platform, those have been selected as the main platforms for creation of the web application for instructors. Microsoft Azure satisfies all the project's requirements and offers a free credit and services for students.

⁵Django (2017)

⁶Ruby on Rails (2017)

⁷Express (2017)

⁸Laravel (2017)

⁹Amazon Web Services (2017)

¹⁰Google Cloud (2017)

¹¹Microsoft Azure (2017b)

3.5. Why Python?

The selected programming language to teach in this game is Python because of several advantageous characteristics. First of all, when compared with equivalent C++ or Java code, Python code is typically one-third to one-fifth their size which reduces the typing effort and increases readability on small-screen devices. Second, Python syntax is simple which as a result has more and more schools having Python as a first programming language when teaching students programming (Lutz, 2013). Finally, Python is an interpreting language which does not have to be compiled, which means it can be executed on different platforms, including mobile devices as well. For Python implementation in this project, the authors will use IronPython, an open-source implementation of the Python programming language which is tightly integrated with the .NET Framework (Mueller, 2010). Python code can be executed on mobile devices and users can get an output or an error. This removes the dependency of a dedicated server for executing code and returning a result which could make game playable in the offline mode as well.

3.6. Game Heuristics

Before the game design and the development process start, it is necessary to analyze the experience of previously released video games. Game heuristics are very important because following them, developers can avoid potential issues in final phases of development, invest more time on testing instead of development and make video games better and more engaging to users. Some of game heuristics identified in literature review (Desurvire, Caplan, & Toth, 2004; Desurvire & Wiberg, 2009; Federoff, 2002; Malone, 1980, 1982) are:

- Follow the trends set by the gaming community to shorten the learning curve.
- Do not expect the user to read a manual.
- “A good game should be easy to learn and hard to master” (Nolan Bushnell).

3. Design and Conceptual Model

- The player experiences the user interface as consistent (in controller, color, typographic, dialogue and user interface design).
- A player should always be able to identify their score/status in the game.
- Feedback should be given immediately to display user control.
- Using sound to provide meaningful feedback.
- Should use visual and audio effects to arouse interest.
- Uncertain outcome.
- Artificial intelligence should be reasonable yet unpredictable.
- The game should give rewards.
- The game must maintain an illusion of winnability.
- Pace the game to apply pressure to, but not frustrate the player.
- Any fatigue or boredom was minimized by varying activities and pacing during the gameplay.

The conclusion from above-mentioned heuristics is that game needs to have a clear goal, responsive feedback mechanism, to be challenging but not too frustrating and, to have a clean visually pleasant user interface. These heuristics will be followed during the design and development process of the project.

3.7. Game Design

sCool is a multi-platform mobile game where a player is in the role of a teenager rescuing astronauts who crashed during one of their space missions. This teenager goes on a rescue mission and takes one small robot which will be of great help. The last received message was that astronauts are alive but they need lost disks which are necessary to run their software system and start the engines in order to return back home. This brave teenager takes a robot and goes to collect those missing discs trying to avoid dangerous places and unfriendly species. As some of the disks are damaged, the player takes a challenge and tries to fix the software content.

3. Design and Conceptual Model

3.7.1. Game Overview

This section covers the core game components such as the gameplay, level design, programming concepts which are going to be taught through this game, game screen elements, main controls, and game mechanics. As the goal of the authors of this project is to provide players both theoretical and practical experience, the game is divided into theoretical and practical parts. In theoretically-based levels, players are exploring a mysterious planet, collecting lost disks, and fixing damaged disks by providing the correct answer for given problem. Practically-oriented levels involve applying gained theoretical knowledge where players could be able to code a robot by using programming concepts and control its movement in order to reach lost disk, These two main game components are individually described in the following sections.

Gameplay

In order to play the game, it is necessary to have a mobile device and as the game is designed to be multi-platform, it can be run on Android, iOS or Windows Phone platform. Once the players open the game, they will be prompted to enter their name and email so instructors can enroll them in their courses (see Figure 3.3). Every player is automatically enrolled in courses which are already prepared so they can start playing the game. Students can be grouped into teams so they can collaborate inside their teams and compete against other teams where their scores are presented in the section for highscores (see Figure 3.2).

After running the game, players can play a mission mode or challenge other teams in multi-player mode. If the player selects the mission mode, a new screen will be shown with list of courses that player is currently enrolled in (see Figure 4.11). Every player is automatically enrolled in default courses which are prepared on the platform, but there is an option to join a course that other instructors prepared. Once an instructor enrolls a student into a course, that student will receive an invitation token for joining the course. By clicking "Join New Course" button, a new pop-up window will be displayed with the form where student can insert a token which is provided from

3. Design and Conceptual Model

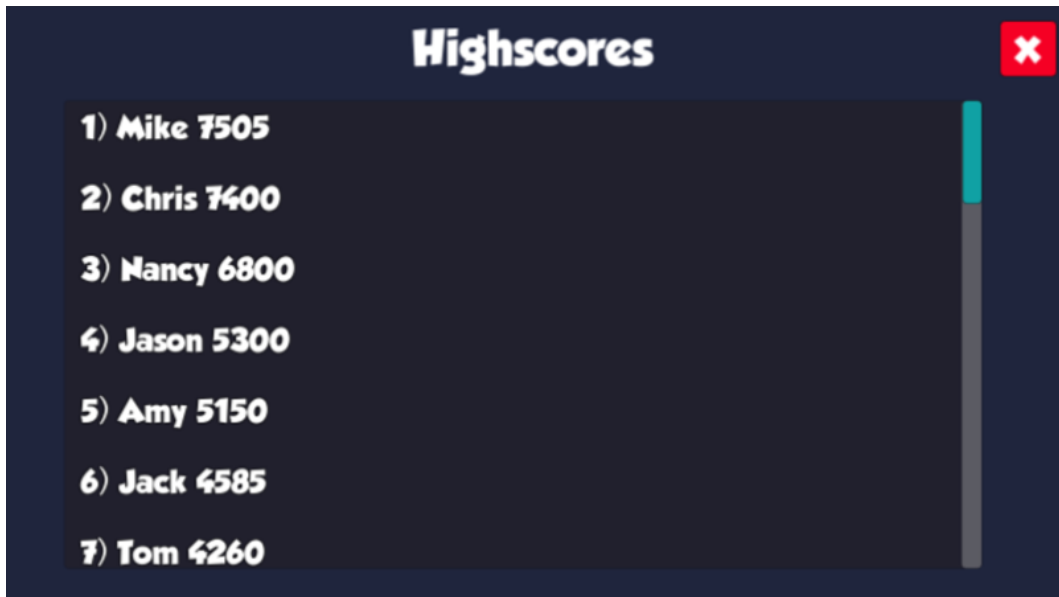


Figure 3.2.: Highscores table with names and achieved points.

the instructor (see Figure 3.4). The next step is to select one of the offered courses. The figure 3.5 displays the skill-tree with different course categories and skills that student has to master within the selected course. Each skill displays current progress level and indicates whether it is unlocked or not with an icon which is next to its name. After selecting one of the skills, players can improve their theoretical or practical knowledge (see Figure 3.6). Also, this screen shows a current progress and knowledge level for both theoretical and practical modes. Both modes come with corresponding tutorials which teach players how to play the game. An example of tutorial for the practical part can be seen in Figure 3.7.

Level Design

sCool game is composed of two types of levels. The first type is theoretically-oriented where students explore the world, fight unfriendly alien creatures, and learn theoretical concepts of programming. The other type is practically-oriented where they work with more practical tasks and type code in an

3. Design and Conceptual Model



Figure 3.3.: Game story and user registration which is shown when students open the game for the first time.

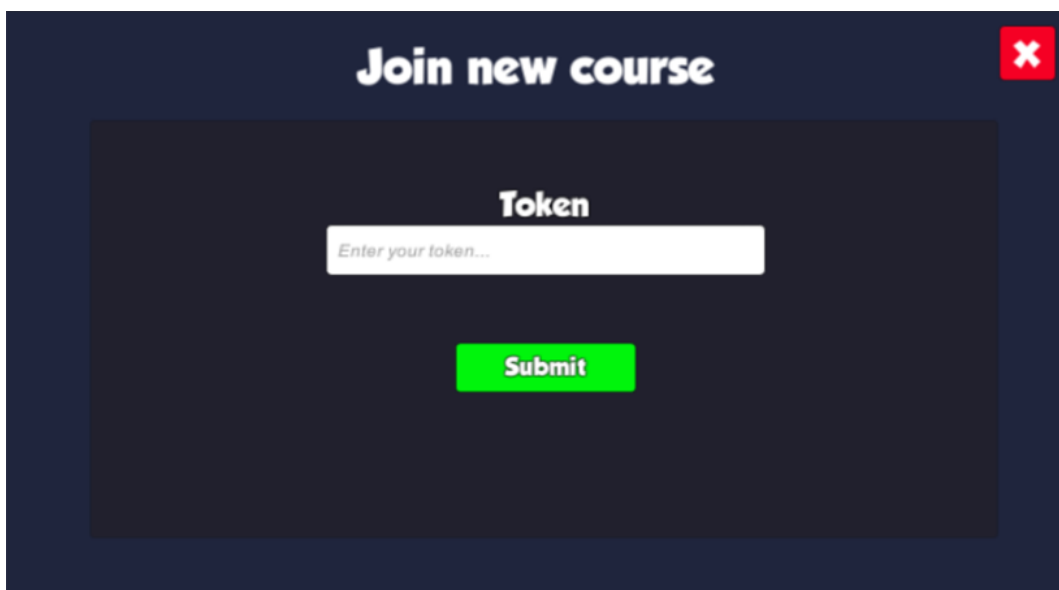


Figure 3.4.: Screen for joining a course with the field for inserting a token.

3. Design and Conceptual Model

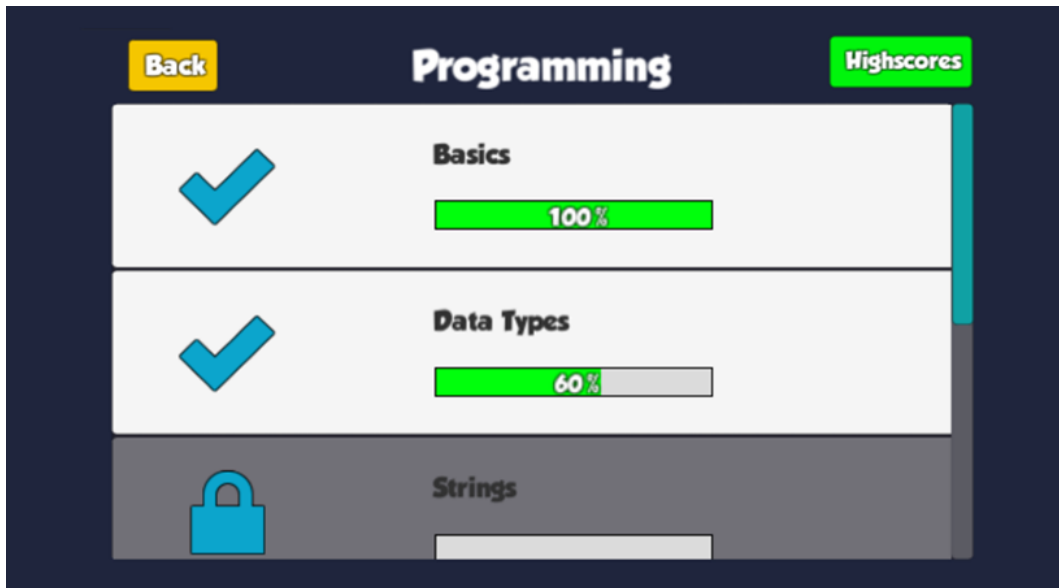


Figure 3.5.: Skill tree loaded from the server and displayed in the mobile game with progress details for every skill.

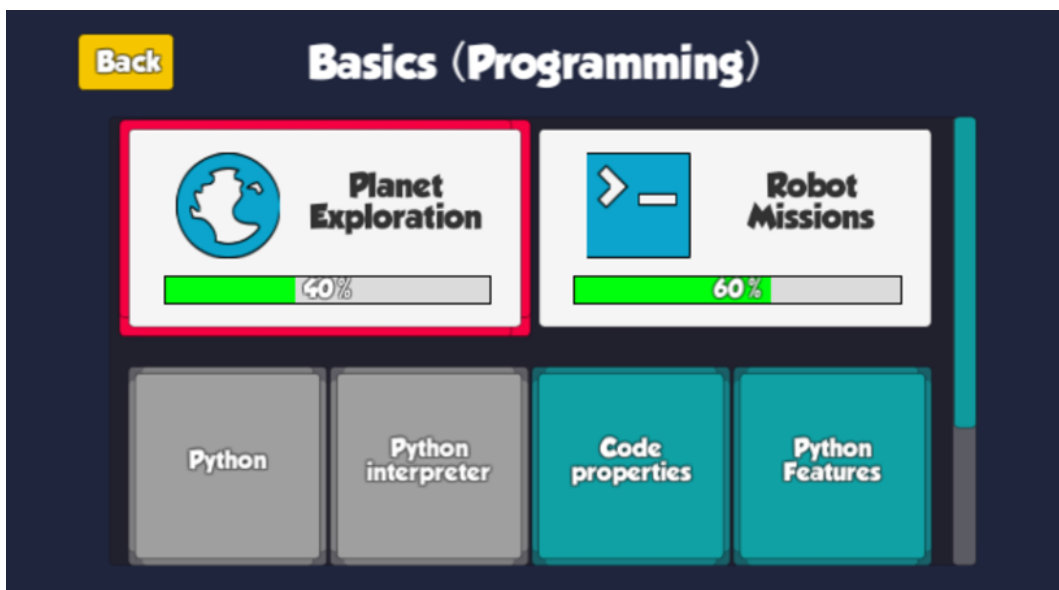


Figure 3.6.: Knowledge level for the "Basics(Programming)" in theoretical and practical modes.

3. Design and Conceptual Model



Figure 3.7.: Tutorial for the practical part which becomes shown when a player opens this mode for the first time.

environment which is designed to be as closest as possible representation of a real programming environment but adapted to the needs of mobile users.

Python Programming Concepts

Programming concepts which are currently used in this game are listed below, but the system can be updated and extended with other concepts as well. For testing purposes and concept evaluation, some lessons used in this game are based on content from Hackerrank (2017); Learnpython (2017); Sololearn (2017); Tutorialspoint (2017).

Programming concepts taught in this game are:

- Basic Concepts (general information about Python programming language and its properties)
- Data Types (data types which are supported by Python, operations with variables)

3. Design and Conceptual Model

- Conditions
- Loops
- Functions

Theoretically-oriented concepts are presented in planet exploration levels so the students can explore and get more information about specific topics while in the levels with robot missions players can practically apply gained knowledge, solve given challenges, and experience real programming environment with code typing, debugging and immediate feedback through robot movement.

3.7.2. Level Design of Planet Exploration

The following sections summarize concept and design of planet exploration level type. It will be described which components are present on the game screen, what kind of game controls are used as well as core game mechanics in this level type. Planet exploration level is designed in a way to emphasize adventure and exploration as good game components which can support education. A simplified state transition diagram of *Planet Exploration* mode is shown in Figure 3.8.

Game Screen

Game screen consists of the main character, unfriendly units, elements which can be collected, progress bars, and other indicators. In the top-left corner of the screen, a health bar is shown with the current state of health points. On the opposite side, in the top-right corner, there is a counter of how many disks player collected out of a total number of disks which are present on the map. Just below it, there is an indicator of earned coins and these coins come with the number of destroyed enemy units.

3. Design and Conceptual Model

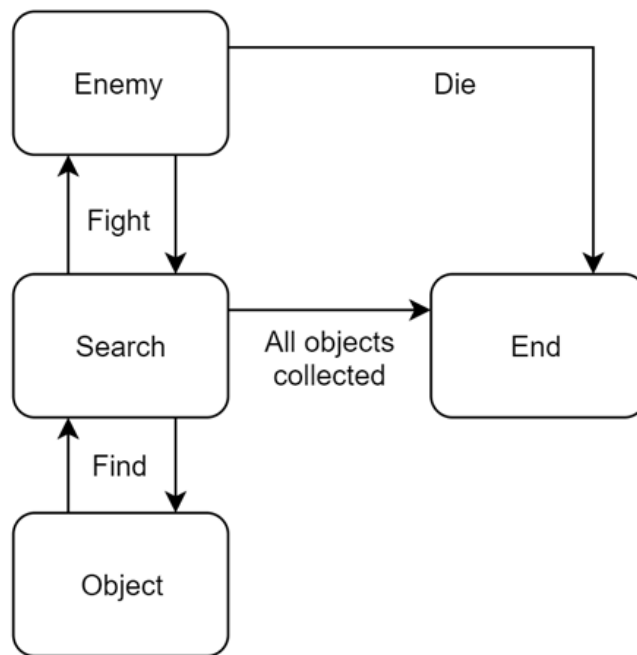
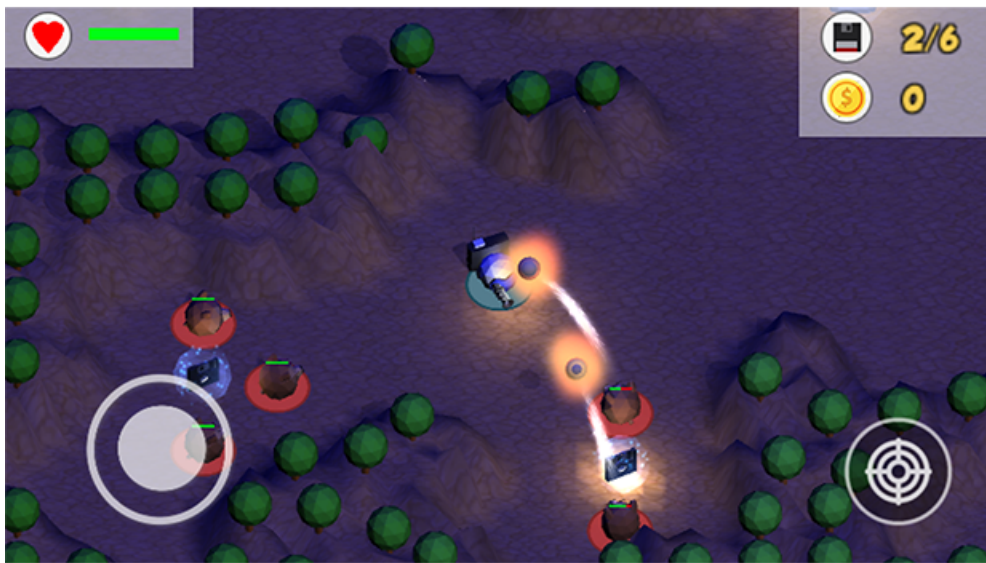


Figure 3.8.: Simplified state transition diagram of the theoretical mode.

3. Design and Conceptual Model



a)



b)

Figure 3.9.: Explorational part of the game with day (a) and night (b) modes.

3. Design and Conceptual Model

Game Controls

Game controls consist of the virtual joystick and the fire button. The controls are created to be suitable and easy to use for mobile devices. Player's movement speed and direction are controlled by using the virtual joystick. Once a player taps on the fire button, missiles are fired and a small charging bar is displayed over the button indicating when this command can be used again.

Game Mechanics

In the exploration part of the game, the main character is exploring procedurally generated maps and collecting lost disks which are spread across a level (see Figure 3.9). These disks are surrounded with different types of unfriendly and dangerous units which can attack the main player. The player needs to be fast and change movement in order to evade attacks of enemy units. For fighting back, the player can use a gun and shoot at enemy units. Pick-up objects which can refill health points are spread over the map so the player can explore different parts of the map and pick-up these objects. Once all the disks are collected, they are combined together and the player has to fix damaged software content by providing correct solutions on the given task. An example of a task with difficulty level of 50% can be seen in figure Figure 3.10. Students are firstly provided with a short lesson and by clicking on the *next* button, the new screen shows up and displays the form for selecting one of the offered answers. If a student provides an incorrect answer, the student is able to continue providing answers as long as there are enough earned coins which can be seen in Figure 3.11.

3.7.3. Level Design of Robot Missions

Similar to the level design of world exploration level type, the following sections summarize design and concept of robot missions level type. It will be described which components are present on the game screen, what kind of game controls are used as well as core game mechanics in this level type. Robot missions level type is designed with a goal to enable

3. Design and Conceptual Model



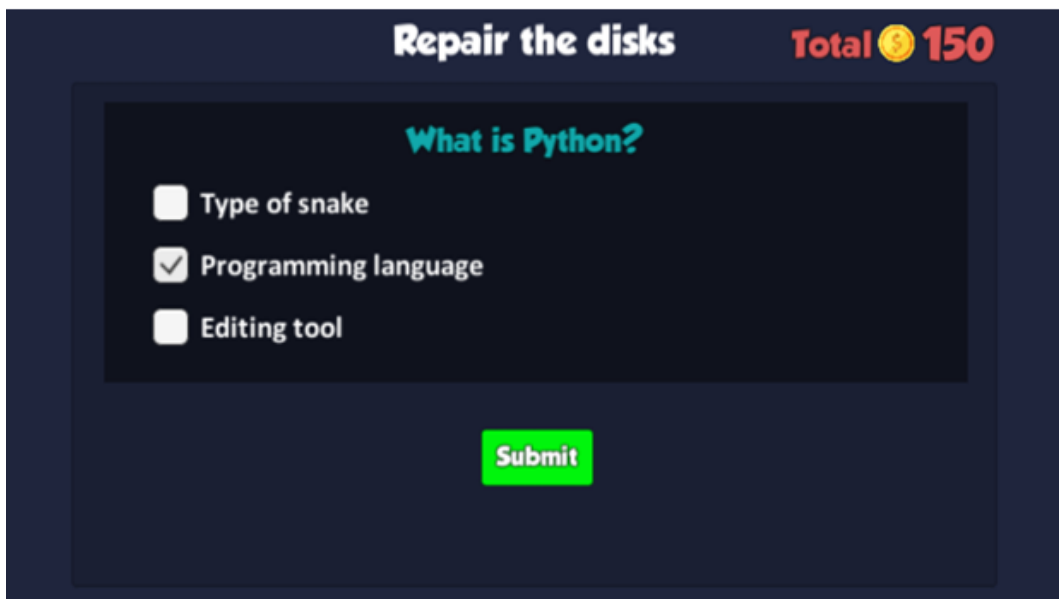
Repair the disks **Total** 🟡 **150**

Python is a high-level programming language, with applications in numerous areas, including web programming, scripting, scientific computing, and artificial intelligence.

It is very popular and used by organizations such as Google, NASA, the CIA, and Disney.

Next

a)



Repair the disks **Total** 🟡 **150**

What is Python?

- Type of snake
- Programming language
- Editing tool

Submit

b)

Figure 3.10.: An example of a task with the difficulty level of 50%. Students are provided with a short lesson (a) and afterwards they need to choose the correct answer (b).

3. Design and Conceptual Model

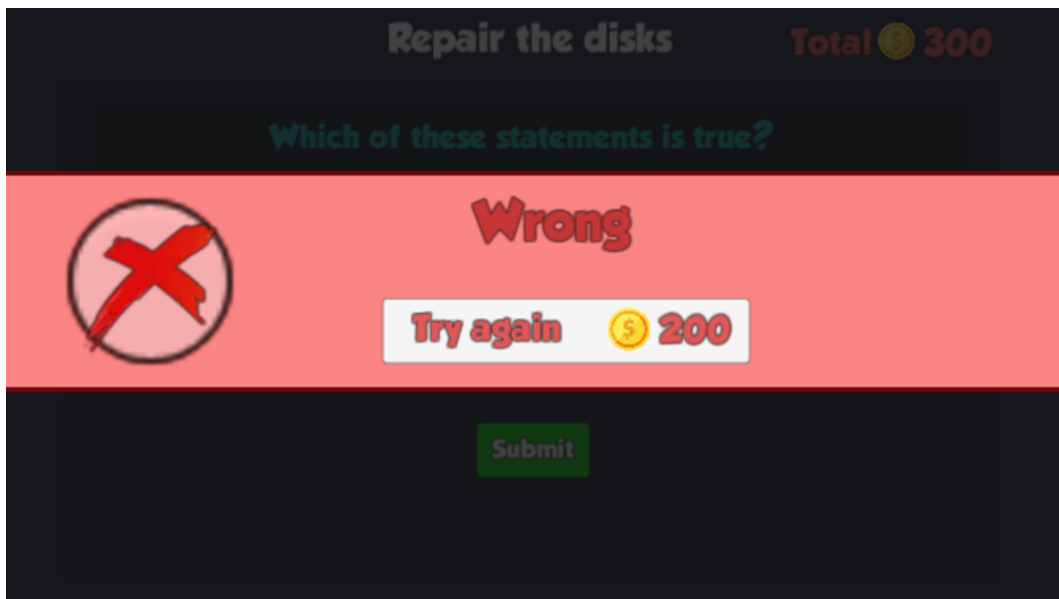


Figure 3.11.: An example of the wrong answer.

students to experience real programming environment on mobile devices. The goal in the robot missions is to use the robot in dangerous areas and areas which player cannot reach, to navigate the robot to the lost disk avoiding obstacles which can destroy the robot. Available commands and coding functionalities depend on obtained task from the server where more advanced functionalities become enabled after the student masters introductory coding concepts.

Game Screen

Main components on the main screen are the predefined commands, place for typing the code and game world as a feedback to given commands and written code. On the left-hand side of the screen, there is a list of the most important commands such as the commands for movement, variable declaration, conditions, and loops. As this is the mobile tool, these code shortcuts are needed because typing code on the mobile screen is slower and more error-prone than writing a code with a real physical keyboard. The

3. Design and Conceptual Model



Figure 3.12.: Practical part of the game where users can apply gained knowledge.

largest part of the screen space is reserved for coding of the robot movement. There are three main tabs. The first tab is for instructions about the level. The second tab is for the code manipulation. Students can either drag-and-drop predefined code blocks or type their own code (see Figure 3.12). Typed blocks of code can be rearranged or dragged to the trash bin which becomes visible once the *drag* event happens. Finally, the code can be interpreted by pressing the *Run* button. Like every programming environment has debugging component where debug or error messages are shown, this game has this functionality as well. Final tab serves to display the debug output or show the errors. A number of remaining attempts is shown in the top-right corner while in the bottom-right corner there is a button for switching between the coding scene and scene for displaying the result in the form of a game scene where the player can see the robot movement.

3. Design and Conceptual Model

Game Controls

In order to control the robot, users type the code in the field for code typing. Depending on the written code, the robot is following loops and branching logic and executing commands for movement.

Game Mechanics

The core game mechanics in this game are based on making correct logic and providing commands which Python will interpret, execute and control the robot in order to reach the lost disk. The world space is tiled so every command for movement moves the player to exactly one tile. The challenge the player has to overcome is to prepare valid code and logic so the robot does not collide with obstacles which are placed over the map. After the robot successfully reaches the lost disk, a pop-up will be shown with a short task which is related to the currently selected topic.

3.8. DGB - Game Parameters

Based on parameters and characteristics which are present in this project, game balancing approach of Hunicke and Chapman (2004) will be followed. Before a level can be balanced it is necessary to measure the difficulty level a user is facing. It can be done by using parameters such as:

- Time required to complete the level
- Remaining health points/lives
- Number of collected items
- Number of killed enemies
- Ratio of successful shoots
- Number of attempts to provide a correct answer
- Result of provided task
- Any other metric necessary to calculate the final score

3. Design and Conceptual Model

As these data are significant for matching learning content with level characteristics, these records can be stored on the server side where statistics and logic for adaptive learning are being processed. After analysis of previous records of the player's progress, the game parameters which can be adjusted are:

- Enemy health points
- Enemy damage value
- Enemy attack speed
- Health points of the main character
- Damage value of the main character
- Attack speed value of the main character
- Number of enemies
- Enemy type
- Map size
- Number of health regeneration objects

Dynamic game balancing can be very useful as it keeps players in the natural flow by adjusting different game parameters.

3.9. Summary

Before the implementation process, it is necessary to define different functional and non-functional requirements for the creation of the sCool project. Functional requirements are divided into two main sections for the mobile game and the web platform. The mobile game needs to be able to support different STEM modules and present both theoretical and practical matter with supporting game mechanics. Dynamic game balancing, based on the player's progress, should be implemented in order to keep players in the natural flow by adjusting different game parameters, such as the damage, health points, number of enemies, and the map size. By incorporating procedural generation techniques, users will be able to play an endless number of levels, interact with wide variety of objects and enjoy in different background music. As students prefer to learn following their own pace, they will be supported with active and adaptive learning approaches. Adaptive

3. Design and Conceptual Model

learning is a contemporary learning methodology which is attracting attention of educational scientists in recent years. However, implementation of adaptive learning brings some challenges, such as the lack of resources, capacity to develop extensive item bank or psychometric expertise in the organization.

The web platform is providing instructors with tools to create courses with theoretical and practical educational content, depending on the selected course and its properties. They will be able to assess students' answers, get an insight into course analytics and help students which are not performing well. The system has some non-functional requirements, such as usability, configurability, and modularity. Systems which are user-friendly, simple to use, visually appealing, have a clear user interface, modular, and configurable are more engaging and enjoyable. Game heuristics are very important when it comes to designing a game and defining its main characteristics. Finally, the technical decision needs to be made for selecting a proper environment. After analysis of existing environments and frameworks which can satisfy defined project requirements, the Unity game engine and the ASP.NET web framework are selected as they fulfill all defined requirements and they are one of the most popular solutions among their competitors.

4. Implementation Details and Showcase Scenario

This chapter covers the technical implementation of the sCool project. The project was built in a client-server architecture. The structure of the sCool project, from a technology perspective, is presented in Figure 4.1. It shows the mobile game and the web platform components which are implemented and described in this chapter. The mobile game is based on the Unity game engine and the game is obtaining an educational content from the server. That educational content is afterwards deserialized and used in the theoretical and the practical modes on which dynamic game balancing principles were applied. The web platform is based on the ASP.NET MVC¹ web framework and the Entity Framework (EF) for facilitating the data access. The mobile game uses the server's web APIs to obtain a content and afterwards sends the results and the statistics back to the server. Also, the web platform provides instructors with the tools for content creation and student analytics which are based on MVC components and UI libraries such as Bootstrap², Morris.js³, Flot⁴, and others.

4.1. Mobile Game

As it is already mentioned, the mobile game was created with the Unity game engine. It is based on Unity's 3D mode and the C# programming language. The game consists of the two main components which are the

¹ASP.NET MVC (2017)

²Bootstrap (2017)

³Morris.js (2017)

⁴Flot (2017)

4. Implementation Details and Showcase Scenario

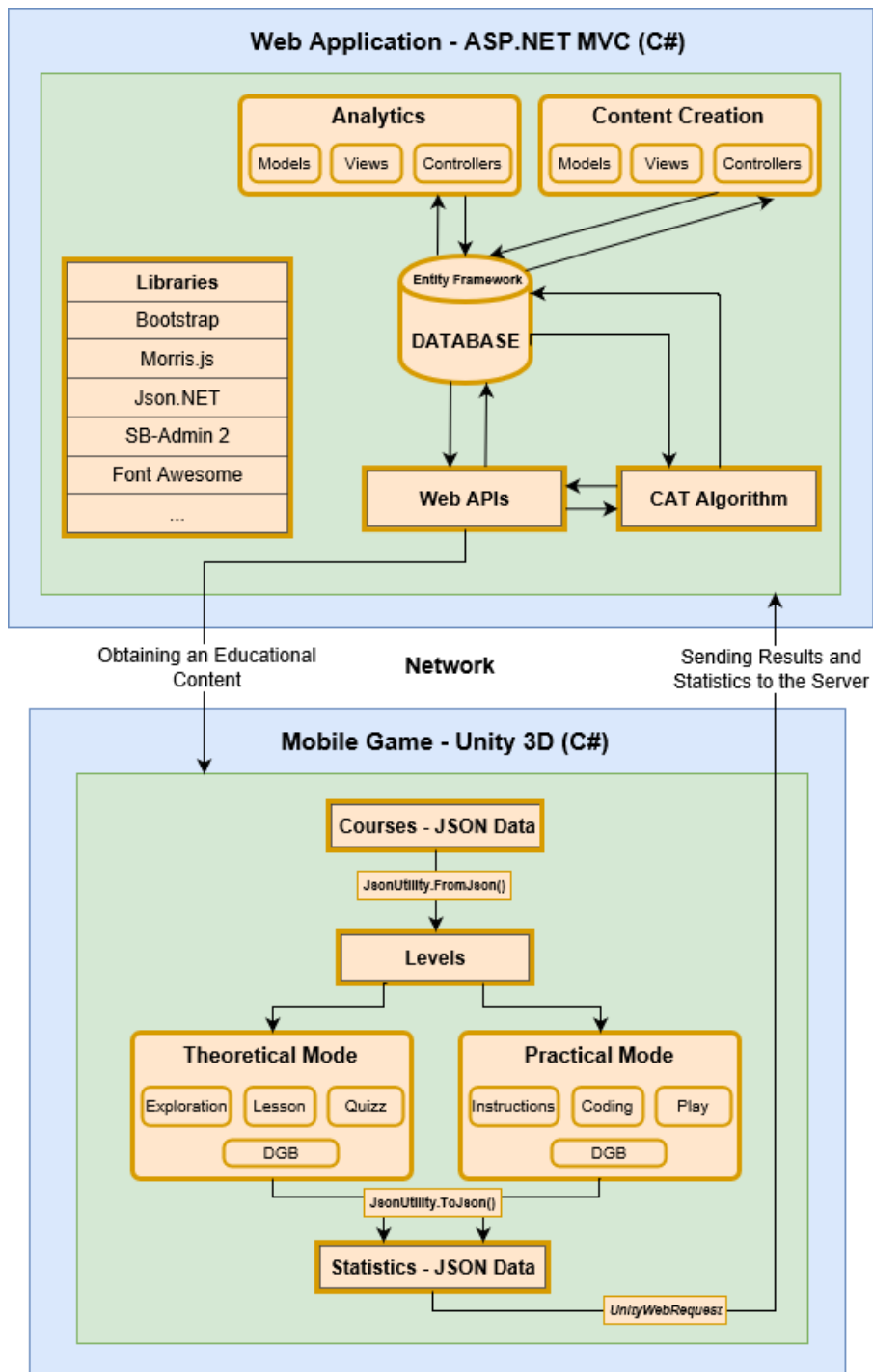


Figure 4.1.: The structure of the sCool project from a technology perspective.

4. Implementation Details and Showcase Scenario

practical and the theoretical modes. Both of these components require an educational content which needs to be retrieved from the server and deserialized before a level can start.

4.1.1. Obtaining an Educational Content

Obtaining an educational content is performed by sending web requests to the server's APIs. The *UnityWebRequest* is the Unity's networking class which is utilized for creating web requests. An instance of this class contains a number of different request properties such as the URL, request type, header parameters or the body parameters. As the game makes different types of requests with different parameters, the code is structured in a way so it can be easily reused (see Listing 4.1). After the request becomes processed by the server, the response is returned in the JSON format. Responses with status codes between 200-300 are indicating that the response message is valid and that error did not occur. The next step is to make data access easier and convert JSON string into the corresponding class instance by using Unity's built-in method *JsonUtility.FromJson()*. An example of the class which instance contains a JSON data (see Figure 4.2) is shown in Listing 4.2. The following step is to use the Unity's UI system to display the student's skill tree for the selected course. The *ScrollRect* component is displaying the UI buttons for every skill. Once the player's data are loaded, courses and skill trees are cached for the further usage. The process of connecting the Unity game and the Microsoft Azure Services is based on ideas and implementation details provided by Douglas (2017).

4.1.2. Theoretical Mode

The theoretical mode is intended to display an educational content. Once a student reads a lesson, the quiz pop-up will be shown in order to check student's knowledge. Educational content is displayed in the *Canvas* component. The Canvas component is an UI holder for other components. The *ScrollRect* component holds a content and enables *Vertical* and *Horizontal*

4. Implementation Details and Showcase Scenario

```
1 public UnityWebRequest request { get; set; }
2
3 public Request(string url, Method method)
4 {
5     request = new UnityWebRequest(url, method.ToString());
6     request.downloadHandler = new DownloadHandlerBuffer();
7 }
8
9 public void AddHeader(string key, string value)
10 {
11     request.SetRequestHeader(key, value);
12 }
13
14 public void AddBody(string bodyJsonString)
15 {
16     if (request.uploadHandler != null)
17     {
18         Debug.LogWarning("Request body can only be set once");
19         return;
20     }
21
22     byte[] bodyRaw = new System.Text.UTF8Encoding().GetBytes(
23         bodyJsonString);
24     request.uploadHandler = (UploadHandler)new UploadHandlerRaw(
25         bodyRaw);
26 }
```

Listing 4.1: Making web requests with the *UnityWebRequest* class.

```
1 [Serializable]
2 public class Course
3 {
4     public string CourseId;
5     public string Title;
6     public string Description;
7     public Skill[] Skills;
8 }
```

Listing 4.2: Enabling serialization of the course class. An example of the JSON object, created as a result of the serialization process, can be seen in Figure 4.2.

4. Implementation Details and Showcase Scenario

```

└─ object {1}
  └─ Courses [5]
    └─ 0 {4}
      CourseId : 1
      Title : Programming
      Description : Programming Course
      Skills [6]
        └─ 0 {7}
          SkillId : 2
          Title : Basics
          Description : Python Basics
          TheoryMeasure : 67.35981384722662
          PracticeMeasure : 55
          SkillUnlocked :  true
          KnowledgeId : 201
          ▶ 1 {7}
          ▶ 2 {7}
          ▶ 3 {7}
          ▶ 4 {7}
          ▶ 5 {7}
        ▶ 1 {4}
        ▶ 2 {4}
        ▶ 3 {4}
        ▶ 4 {4}

```

Figure 4.2.: JSON structure of courses with corresponding skill-trees obtained from the server. The provided JSON object is deserialized into corresponding classes (see Listing 4.2) where class and property names have to be exactly the same like names in the JSON object in order to have a proper deserialization.

4. Implementation Details and Showcase Scenario

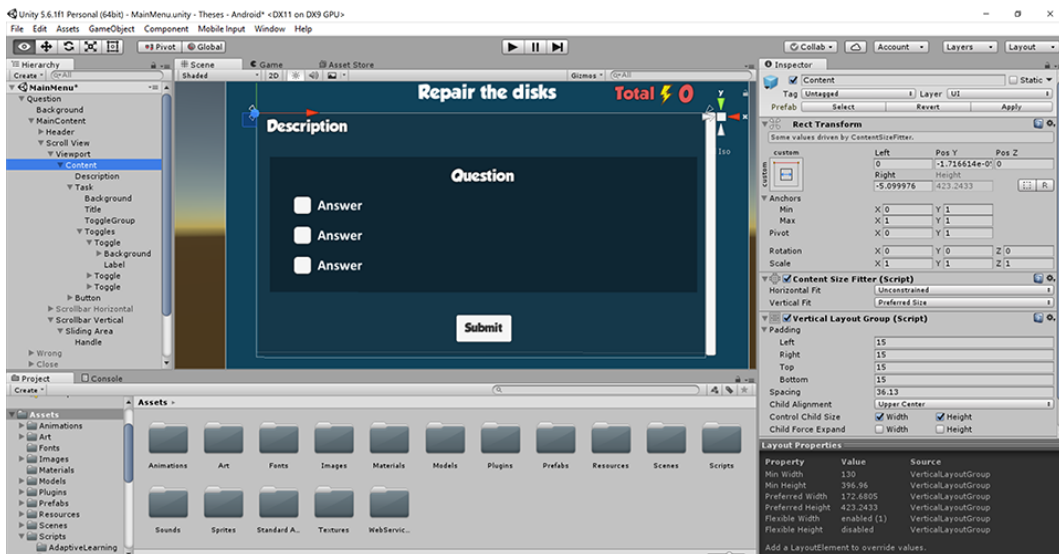


Figure 4.3.: The front-end implementation and the question structure with the corresponding components.

scrollbars. In order to enable dynamic fitting and vertical ordering of elements in the content field, the *ContentSizeFitter* and the *VerticalLayoutGroup* are used. For displaying and selecting offered answers, the *Toggle* elements are included. As each of them works independently, it is necessary to group them and have the functionality where by selecting one, all the other elements become unchecked. For this purpose the *ToggleGroup* component is created and connected with all the toggle elements (see Figure 4.3).

After the student provides an answer, the results are sent to the server. In order to serialize the data, a class requires the *Serializable* attribute. The next step in the serialization process is to pass an instance of an object to the *JsonUtility.ToJson()* method. This method handles object serialization and returns data in the string format.

4.1.3. Practical Mode

The practical mode enables users to practice their skills. Due to technical limitations which are currently present in the Unity's UI system and inability

4. Implementation Details and Showcase Scenario

to use the present UI components to satisfy requirements of the project, it was necessary to develop components which enable functionalities such as:

- Drag-and-drop functionality
- Virtual keyboard for code writing
- Code block reordering
- Tabs system

The UI structure of the above-mentioned components can be seen in Figure 4.4. Some of these components depend on the content received from the server. The more advanced functionalities are becoming enabled as students progress. At the beginning of the level, a new content is obtained from the server. That content contains instructions and objectives which students have to fulfill and afterwards print the result to the *output* panel. By typing the code, they are able to move the robot to reach the disk. As soon as the robot reaches the disk, the output is compared with the solution and if they match the mission is successfully completed, otherwise the pop-up window is shown (see Figure 4.5). As the provided answer can be in a multi-line form, all new-line symbols and space characters are removed. The string is then converted into the lowercase form so the output and the solution can be compared.

4.1.4. Dynamic Game Balancing (DGB)

Players engage in interaction loops of exploring, solving, fighting and collecting. Once an undesirable loop is detected, balancing system needs to be placed in action in order to prevent similar situations in the further attempts. As the goal in this project is to keep the players in the flow channel, some of the game parameters such as the damage, health points, number of enemy units, and the map size are adjusted based on the task difficulty and previous player's statistics. As players progress and develop their skills, the difficulty level increases so the new game items are becoming introduced to the players. If they are not performing well, the game difficulty will decrease and vice versa. The new level difficulty is obtained by using the CAT algorithm. Even though the adaptive learning module is based on this

4. Implementation Details and Showcase Scenario

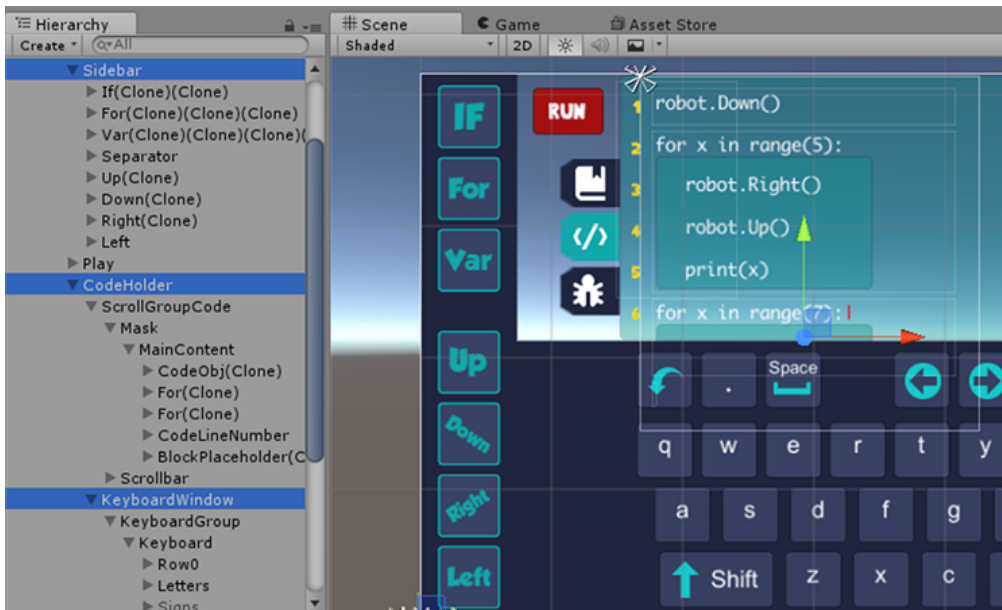


Figure 4.4.: UI structure of the main components in the coding tool.



Figure 4.5.: Output mismatch in the practical task.

4. Implementation Details and Showcase Scenario

algorithm, it showed good results when suggesting new level difficulties based on previous attempts.

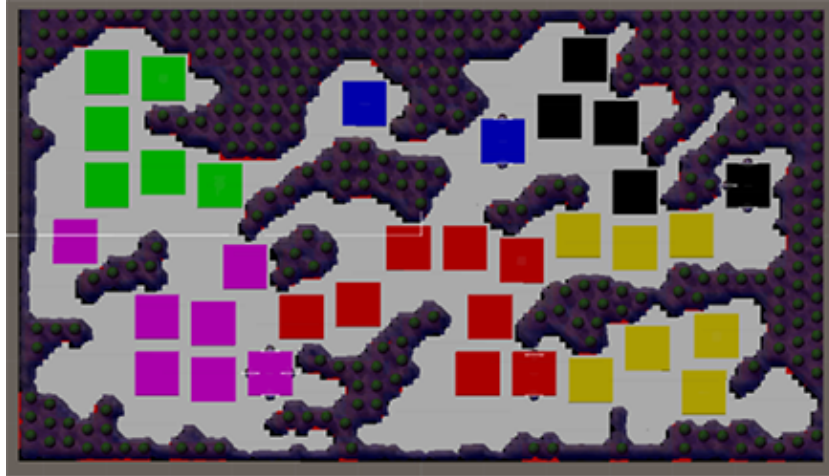
In the theoretical mode, the system tries to generate a map where all prepared items can be placed. If there is no space for all of them, the map will be increased by 10% and this process will be repeated until all the items can be placed on the map. Figure 4.6 shows the different map sizes based on the player's progress. In the practical mode, the adjustment will be performed by positioning the disk closer to the player if the player is performing poorly and vice versa. In the introductory levels, users might not be completely familiar with the environment. As they still have to master their skills, the disk is purposely positioned closer to the robot's initial point where players can reach the disk in several steps. The disk distance from the starting point is increasing following the player's progress. The figure 4.7 shows how disk position depends on the level difficulty. Before the disk is being positioned, all the obstacles are dynamically placed on the map. After that, the matrix representation of the map (15x15) is populated with positions of the obstacles so the disk can be placed on an empty field.

4.1.5. Multiplayer

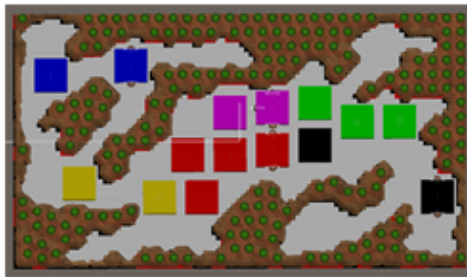
Creating a multiplayer game is a challenging task. In this project, for a proof of concept and for enhancing the development process, the Photon⁵ platform is used. It offers services for connecting end sides and transmitting network packets. As all the levels are procedurally generated, the level with all corresponding items needs to appear on each side. Once the players join the room, the master player generates a seed for a procedural map which is being synchronized along the network. After every participant receives the seed, they are able to display the same level like it is shown on the main player's screen. Then, the master client analyzes the generated map and tries to find some free spots for positioning other players. These positions are sent to other participants together with the generated seed. As the network is the slowest component in the system, it is not practical to send the data at a very high rate. Because of this network property, it is necessary to perform

⁵*Photon Engine* (2017)

4. Implementation Details and Showcase Scenario



100% map size



50% map size



25% map size

Figure 4.6.: Dynamic map size based on player progress in the theoretical mode.

4. Implementation Details and Showcase Scenario



Figure 4.7.: Disk positioning based on the level difficulty in the practical mode.

some optimization techniques and interpolate received data. After testing the game smoothness by using different parameters, it has been noticed that if velocity parameter is passed together with the character's movement, it looks more smooth because the character does not have to wait for the next packet but it can use its own velocity parameter to proceed with the movement (see Listing 4.3).

4.2. Web Application

This section describes the implementation details of the web application for instructors. The web application was built in the ASP.NET MVC web framework using C# programming language. Entity Framework (EF), an object-relational mapper, is utilized for facilitating the data access. By using the EF, the process of code writing was easier as well as the data manipulation. ASP.NET Web API framework offered the Representational State Transfer (REST) interface. The RESTful interface was very important for

4. Implementation Details and Showcase Scenario

```
1 void Update()
2     {
3         if (!photonView.isMine)
4         {
5             transform.position = Vector3.Lerp(transform.
6                 position, this.targetPos, Time.deltaTime * 5);
7             transform.rotation = Quaternion.Lerp(transform.
8                 rotation, this.targetRot, Time.deltaTime * 5);
9         }
10    }
11
12 void OnPhotonSerializeView(PhotonStream stream,
13     PhotonMessageInfo info)
14 {
15     if (stream.isWriting)
16     {
17         // We own this player: send the others our data
18         stream.SendNext(transform.position);
19         stream.SendNext(transform.rotation);
20         stream.SendNext(rigidbody.velocity);
21     }
22     else
23     {
24         // Network player, receive data
25         this.targetPos = (Vector3)stream.ReceiveNext();
26         this.targetRot = (Quaternion)stream.ReceiveNext();
27         this.rigidbody.velocity = (Vector3) stream.
28             ReceiveNext();
29     }
30 }
```

Listing 4.3: Multiplayer data synchronization.

4. Implementation Details and Showcase Scenario

sending an educational content to mobile users and accepting game statistics from mobile devices. For serialization and deserialization purposes, Json.NET⁶ is utilized, which is a high-performance JSON framework for .NET.

The UI of the web application is based on the HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript. An empty ASP.NET MVC project came with already included jQuery and Bootstrap UI libraries. The user registration functionalities were also provided which enhanced the development process. Morris.js and Flot UI libraries were used for displaying different types of charts and supporting analytics functionalities. Font Awesome⁷, the iconic font and CSS toolkit, was utilized in the project as it provides a number of customizable icons. After the implementation process was completed, the web application was hosted on the Microsoft Azure cloud platform.

In the following text it will be discussion about the main project components such as the structure of the web application, data access, web APIs, content creation process, analytics system, and the practical implementation of the CAT algorithm.

4.2.1. Development with the EF code-first approach

ASP.NET web framework is used for the creation of the web application. Entity Framework is utilized in order to enhance data access by using .NET objects. EF supports three different development approaches which are:

- database-first
- model-first
- code-first

The development approach used for this project is the code-first approach where every entity is defined by creation of entity classes. This approach enables creation of migrations which makes the process of tracking the database changes through the version control easy. Also, it enables the input

⁶Json.NET (2017)

⁷Font Awesome (2017)

4. Implementation Details and Showcase Scenario



Figure 4.8.: Entity Framework code-first approach.

validation to be defined on the model level. This means that the validation is automatically performed every time a certain entity is used. This makes the overall system more secure and easier to maintain as validation is defined only at one place by using annotations above class attributes.

Based on the created entity classes and the relationship configuration, the code-first APIs are able to generate the database with tables (see Figure 4.8). An example of the entity class for the statistics entity is shown in the Listing 4.4. Entities defined in the project's database model are:

- **User** - registered instructors
- **Courses** - created courses
- **Students** - registered students
- **Enrolled** - records for courses each student is enrolled in
- **Skill** - course skills
- **Knowledge** - data about students' knowledge level for each skill
- **TheoryTask** - theoretical tasks created by instructors
- **PracticeTask** - practical tasks created by instructors
- **Statistics** - players' statistics

The database model with entity relationships is shown in Figure 4.9. ASP.NET does not provide default functionalities of saving information when a certain property has been created or modified. Since this can be very useful for the situations when an item is created or when a user last time played a certain level, it has been implemented in a way that every entity class extends the *BaseEntity* class which contains *CreatedAt* and *UpdatedAt* attributes. These fields are afterwards automatically updated by the database context.

4. Implementation Details and Showcase Scenario

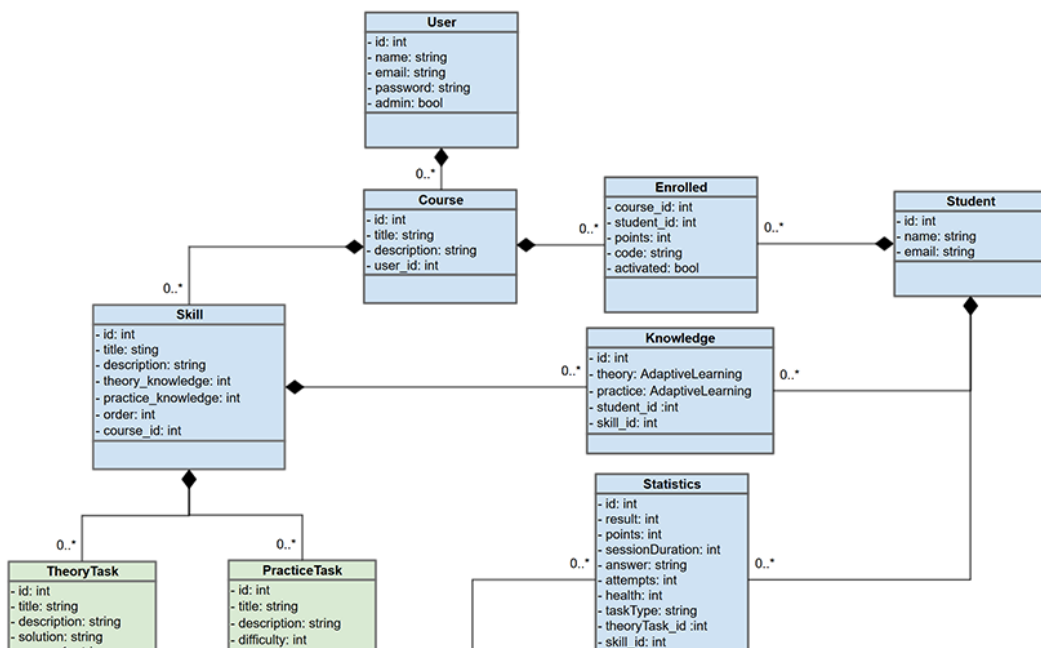


Figure 4.9.: Simplified database model of the sCool system.

4. Implementation Details and Showcase Scenario

```
1 public class BaseEntity
2 {
3     public DateTime? CreatedAt { get; set; }
4     public DateTime? UpdatedAt { get; set; }
5 }
6
7 public class Statistics : BaseEntity
8 {
9     [Key]
10    public int StatisticsId { get; set; }
11
12    [Required]
13    public int Result { get; set; }
14    [Required]
15    public int Points { get; set; }
16    [Required]
17    public int SessionDuration { get; set; }
18    [Required]
19    public string Answer { get; set; }
20    [Required]
21    public int Attempts { get; set; }
22    [Required]
23    public double Health { get; set; }
24
25    public virtual Student Student { get; set; }
26    public virtual TheoryTask TheoryTask { get; set; }
27 }
```

Listing 4.4: Entity Framework Code-First with class attribute annotations.

4. Implementation Details and Showcase Scenario

4.2.2. Web Application Structure

The web application is based on the ASP.NET MVC web framework, which consists of the three interconnected parts. The *Model* is the central component in this software architectural pattern which is responsible for the logic and the data management. Classes for all the models are kept in the *Models* folder.

The *View* component uses the Razor syntax for generating the HTML output. The Razor code example with HTML helpers, for displaying forms, can be seen in the Listing 4.5. These helpers are very useful as they generate anti-forgery tokens, prepare a style format for forms and display error messages if any. The HTML helpers automatically check annotations of class attributes and show error messages below every field if certain criteria are not satisfied (see the Figure 4.14). There exist different default attributes such as *Required*, *Min*, *Max*, and other. It is possible to define a custom attribute annotations as well. An example of the custom attribute annotation is checking if student email already exists. This annotation calls a function which checks if a student with the identical email already exists. The advantage of this approach is that it is reusable for other classes as well.

Finally, the *Controller* component is in charge of accepting input and sending commands to the model and the view components. All the controllers are grouped in the *Controllers* folder. This folder is placed inside of the project's root folder. Similar to the models and the controllers, all the views are kept inside of the *Views* folder. They are grouped by the controller names as one controller can have more actions and thus more views for each controller. Some of the most important controllers used in the project are:

- **Home** - displaying home pages for guest users
- **Account** - login, register and password reset functionalities
- **Courses** - creating, editing, deleting and listing courses
- **TheoryTasks** - managing theoretical tasks
- **PracticeTasks** - managing practical tasks
- **Skills** - creating, editing, deleting and listing skills for the selected course
- **Students** - enrolling students into courses

4. Implementation Details and Showcase Scenario

```
1 @using (Html.BeginForm())
2 {
3     @Html.AntiForgeryToken()
4
5     <div class="form-horizontal">
6         <h4>Edit</h4>
7         <hr />
8         @Html.ValidationSummary(true, "", new { @class = "text-
9             danger" })
10
11        @Html.HiddenFor(model => model.CourseId)
12
13        <div class="form-group">
14            @Html.LabelFor(model => model.Title, htmlAttributes
15                : new { @class = "control-label col-md-2" })
16            <div class="col-md-10">
17                @Html.EditorFor(model => model.Title, new {
18                    htmlAttributes = new { @class = "form-
19                        control" } })
20                @Html.ValidationMessageFor(model => model.Title
21                    , "", new { @class = "text-danger" })
22            </div>
23        </div>
24
25        <div class="form-group">
26            @Html.LabelFor(model => model.Description,
27                htmlAttributes: new { @class = "control-label
28                    col-md-2" })
29            <div class="col-md-10">
30                @Html.EditorFor(model => model.Description, new
31                    { htmlAttributes = new { @class = "form-
32                        control" } })
33                @Html.ValidationMessageFor(model => model.
34                    Description, "", new { @class = "text-danger
35                        " })
36            </div>
37        </div>
38
39        <div class="form-group">
40            <div class="col-md-offset-2 col-md-10">
41                <input type="submit" value="Save" class="btn
42                    btn-default" />
43            </div>
44        </div>
45    </div>
46 }
```

Listing 4.5: Razor syntax with HTML helpers for the course editing

4. Implementation Details and Showcase Scenario

Every information about students' progress needs to be tracked and accordingly updated. These data are stored in the aggregation table between the student and the skill entities. Along the other data, there is the necessity of storing an information for the adaptive learning progress. EF supports a number of primitive data types, however having a complex type for storing and maintaining data can be very practical as it enables the implementation of additional functions within the entity class.

4.2.3. Web APIs

The implemented APIs are based on the ASP.NET Web API framework. This framework offered the REST interface which served as an access point for a data sent to and received from the mobile game. The sCool project contains the following APIs:

- **User registration** - the first time an user opens the game it is necessary to register that user, enroll the user into predefined courses and obtain student's identification number. That ID is required for referencing other records to the user.
- **Skill tree** - providing student's identification number and obtaining a list of courses with corresponding skill trees for every course.
- **Get theoretical task** - providing student's identification number and obtaining theoretical task which will be used in the game. This theoretical task depends on previous answers and current knowledge level.
- **Processing theoretical response** - once results and statistics from the game arrive, the new knowledge measure is calculated. If both the practical and the theoretical measures are above the given threshold, the next available skill becomes unlocked. Parameters for adaptive learning and forgetting curve are accordingly updated.
- **Get practical task** - providing student's identification number and obtaining practical task which will be used in the robot programming module. Similar to the theoretical tasks, this practical task depends on previous answers and current knowledge level.
- **Processing practical response** - processing arrived data and updating ability level in the adaptive learning component. Game statistics from

4. Implementation Details and Showcase Scenario

robot programming mode are received and stored for further analysis. Instructors can even see the code students typed and provide them some suggestions regarding the writing style, logic, and performances.

- **Hightscores** - providing course ID and obtaining the list of students with their details. The results are ordered by the score value.
- **Join a Course** - joining a course based on the token provided by an instructor.
- **Save Student Details** - players are able to change their name, email and other properties within the game and these details are sent back to the server.

These APIs derive the *System.Web.Http.ApiController* class and are placed within the *API* directory, which parent folder is *Controllers* folder where all the project's controllers are located. For defining the properties of APIs, attributes such as the request type, route, and response type are used. A *dynamic* data object, whose operations are resolved at runtime, is used for construction of data responses. As the client-server communication needs to be secured, security mechanisms such as the SSL protocol for encryption and security tokens for authentication are implemented.

4.2.4. Content Creation

After an instructor is logged in, the section for course management is shown. In this section, courses can be listed, created, edited and deleted (see Figure 4.10). Also, every page within the web application has implemented guides and instructions which help instructors to navigate and use the platform. The list of courses is afterwards used and shown in the mobile game (see Figure 4.11). By clicking on one of the courses, a new interface for the skill management is being shown where instructors can prepare a skill tree for the selected course and enroll students into the specific course (see Figure 4.12). Once the skill tree is ready, it can be loaded in the game so, users can see their current progress (see Figure 3.5). Instructors are able to edit, delete and open the page where skill details are displayed with listed tasks. Every skill has two task types which are the theoretical and the practical tasks (see Figure 4.13).

4. Implementation Details and Showcase Scenario

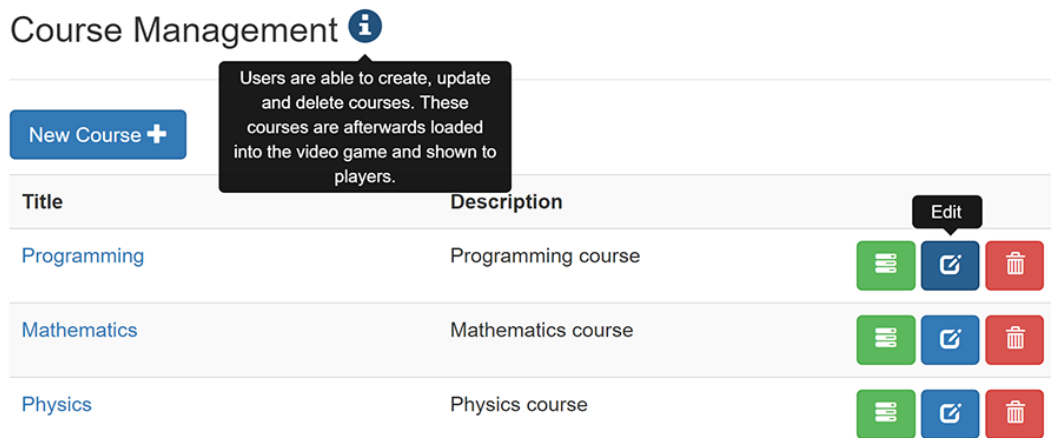


Figure 4.10.: Course listing in the web application with the corresponding instructions.

When creating a theoretical task, an instructor can enter the task title, the task definition and define the possible answers. Only one of the available answers is the correct answer. Tasks are presented in a form of selecting the correct answer among the three displayed answers.

An example of the task creation can be seen in Figure 4.14. The process of the practical task creation is similar to the process of the theoretical task creation. The only difference is that instead of having questions and answers, students will get a task to solve by typing the code and printing results to the output. The figure 4.15 shows the process of creating a new practical task. There are the checkbox options for enabling or disabling specific code shortcuts in the game depending on the task difficulty. These options are enabled per default, but instructor can turn them off if some of them are not suitable for certain lesson.

4.2.5. Course and Student Analytics

The analytics component is supported with the UI libraries such as Bootstrap, Morris.js and Flot. These libraries provided functionalities such as progress bars, activity diagrams, pie charts etc. As soon as students start playing the game, statistics about their progress becomes available. The purpose

4. Implementation Details and Showcase Scenario

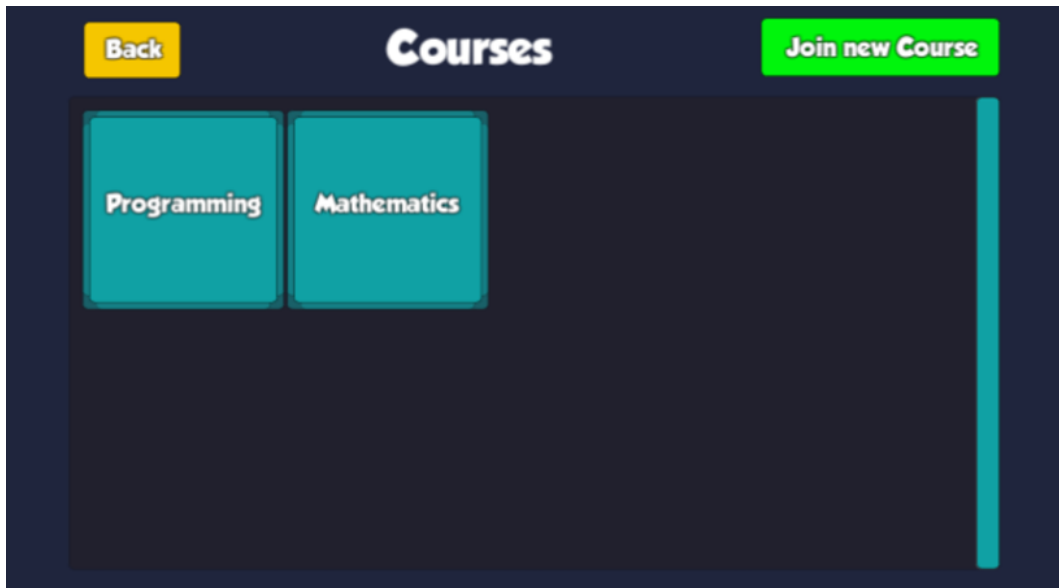





Figure 4.11.: Course listing in the mobile game and the option for joining a new course.

of these statistics is to enable instructors to get an insight of how students are performing, and to detect if students are facing difficulties with some concepts. In the course section (see Figure 4.11), instructors can select the "Students" option to see more details about the students. As it can be seen in Figure 4.16, the list of enrolled students in the selected course is shown, and on the top of the screen there is a graph displaying students' activities per day. By clicking on one of the students, the new page about student's details becomes shown (see Figure 4.17). On this page, an instructor can get a skill tree overview, see student's progress for selected course, and observe how the student is performing. All student's attempts are shown in the graph on top of the screen where those activities are grouped per day.


An instructor can click on one of the skills in the skill tree and get more details about activities and statistics for selected skill. On the statistics page, all student's activities and attempts are displayed, so the instructor can get an insight about student's knowledge and which concepts student failed to grasp (see Figure 4.18). Instructors can get more detailed statistics and see an average percentage of successful answers, earned points per level, an average session duration in minutes, and a number of attempts per


4. Implementation Details and Showcase Scenario

Course Details 

| | |
|--------------------|--------------------|
| Title | Programming |
| Description | Programming course |

 Students

 Course Skills

New Skill +






















| Title | Description | |
|------------|----------------------|---|
| Basics | Python Basics |    |
| Data Types | Data Types in Python |    |
| Strings | Strings |    |
| Conditions | Conditions |    |
| Loops | Loops |    |
| Functions | Functions |    |

Figure 4.12.: Displaying student details and managing the skill tree.

4. Implementation Details and Showcase Scenario































Skill Details 

Title Data Types
Description Data Types in Python

Theory Tasks

[New Theory Task +](#)

| Title | Description | Difficulty (%) | |
|---|--|----------------|---|
| Select a correct output | A variable allows you to store a value by assigning... | 10 |    |
| Select a correct output | A variable allows you to store a value by assigning... | 20 |    |
| Select a correct output | A variable allows you to store a value by assigning... | 30 |    |
| Select a correct output | A variable allows you to store a value by assigning... | 40 |    |
| Select a correct output | A variable allows you to store a value by assigning... | 50 |    |
| Is it possible to reassign a variable in Python? | Variables can be reassigned as many times as you... | 60 |    |
| Which of the following variables contain a floating ... | Python has five standard data types : Numbers, St... | 70 |    |
| Select a correct answer | >>> x = 5 >>> y = ___ >>> print(x + y) 12 | 80 |    |
| Select correct variable assignment | _____ print(myVar) | 85 |    |
| Select an option that will cause a ZeroDivisionErro... | Dividing by zero in Python produces an error, as n... | 90 |    |

Practice Tasks

[New Practice Task +](#)




























| Title | Description | Difficulty (%) | |
|--|------------------------------------|----------------|---|
| Use the print command to print the following calculation | print(2 + 2) | 10 |    |
| Use the print command to print the following calculation | print(5 + 14) | 20 |    |
| Use the print command to print the following calculation | print(52 + 3) | 30 |    |
| Use the print command to print the following calculation | print(5 + 4 - 3) | 40 |    |
| Print the output of calculation (7+2-6) | Python has the capability of c... | 50 |    |
| Create a small program which prints result of the following calculation: >>>print(2 * (3 + 4)) | Python also carries out multipl... | 60 |    |
| Solve the task from description area. | Define variables and print thei... | 70 |    |
| Solve the task from description area. | Define variables and use the "... | 80 |    |
| Python Arithmetic Operators - solve the task from the bottom of the description field. | Assume variable a holds 10 a... | 90 |    |

Figure 4.13.: Displaying skill details and listing created practical and theoretical tasks. These tasks are afterwards loaded and used in the mobile game.

4. Implementation Details and Showcase Scenario

New Theoretical Task



Title ⓘ

Description ⓘ

Correct Answer ⓘ

Incorrect Answer 1 ⓘ
The Incorrect Answer 1 field is required.

Incorrect Answer 2 ⓘ
The Incorrect Answer 2 field is required.

Hint ⓘ

Difficulty ⓘ
The Difficulty field is required.



 Save

Figure 4.14.: The form used for creation of new theoretical tasks with front-end form validation.

4. Implementation Details and Showcase Scenario

New Practical Task



Title ⓘ
The Title field is required.

Description ⓘ
The Description field is required.

Solution ⓘ

Difficulty ⓘ
The Difficulty field is required.

Enable/Disable code shortcuts in the video game

| | |
|-----------------------|-------------------------------------|
| Print ⓘ | <input checked="" type="checkbox"/> |
| Variable ⓘ | <input checked="" type="checkbox"/> |
| If statement ⓘ | <input checked="" type="checkbox"/> |
| For loop ⓘ | <input checked="" type="checkbox"/> |
| Move Left ⓘ | <input checked="" type="checkbox"/> |
| Move Right ⓘ | <input checked="" type="checkbox"/> |
| Move Up ⓘ | <input checked="" type="checkbox"/> |
| Move Down ⓘ | <input checked="" type="checkbox"/> |




Figure 4.15.: The form for creation of a new practical task with front-end form validation.

4. Implementation Details and Showcase Scenario

session. Below the statistics section, the details for every activity are shown. Also, an instructor can see which tasks the student had to solve, what were the provided answers, a number of attempts student have had, and other statistical details regarding how the student performed in the game. This is very important as instructors can organize their classes in a way of focusing themselves primarily on students who really need help and committing more time only on concepts which were challenging for students. By doing this, a class could be more effectively organized and students could get help based on their performances.

Every user's action such as a tap position on the screen, a level name or a provided answer is being stored. These data are stored for further analysis as they can be used for visualizations and understanding of player's behavior and preferences. During the gameplay, all the data are recorded and sent to the server. The complete list of the data which are tracked is shown below:

Theoretical part:

- Task
 - Success/failure
 - Difficulty
 - Provided answer
 - Number of attempts
- Session duration
- Earned points
- Character's health percentage
- Map
 - Seed
 - Width
 - Height
- Enemies
 - Type
 - Position
 - Health
 - Damage

4. Implementation Details and Showcase Scenario

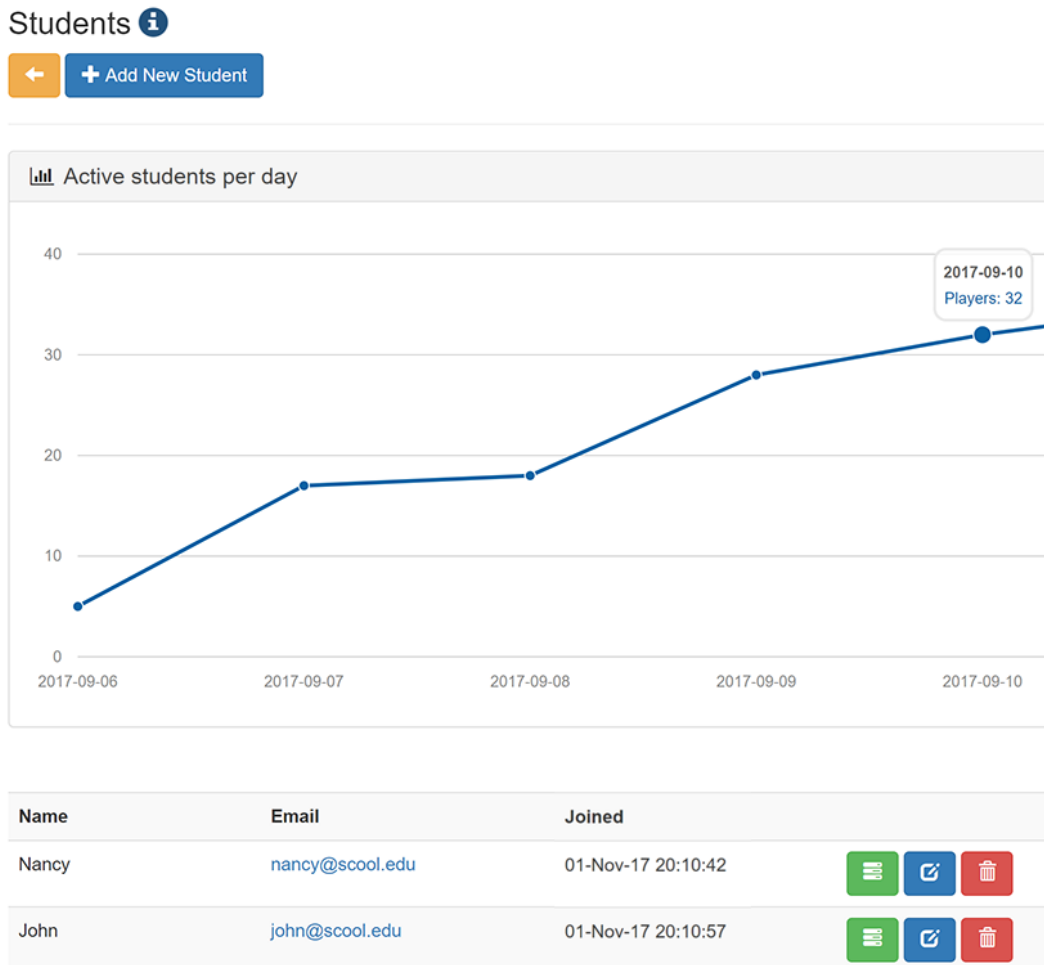


Figure 4.16.: Student activities per day and the list of students enrolled in the selected course.

4. Implementation Details and Showcase Scenario

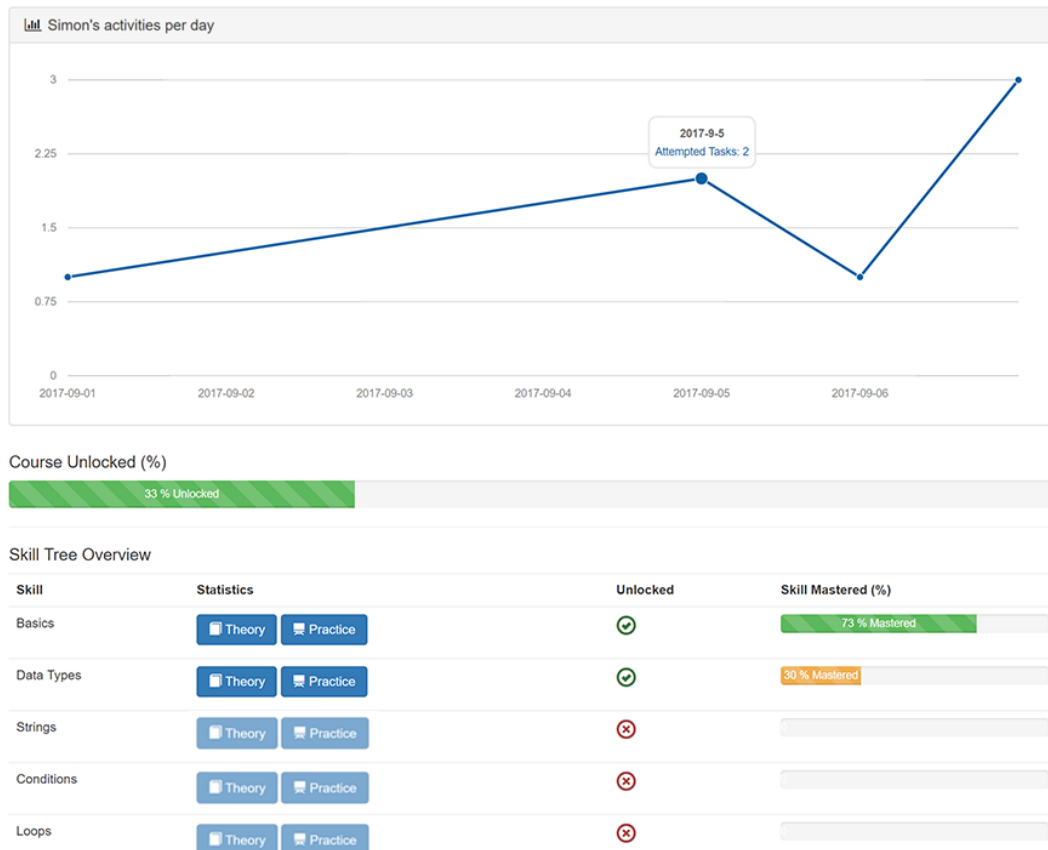


Figure 4.17.: Skill tree overview with the student's progress and activities.

4. Implementation Details and Showcase Scenario

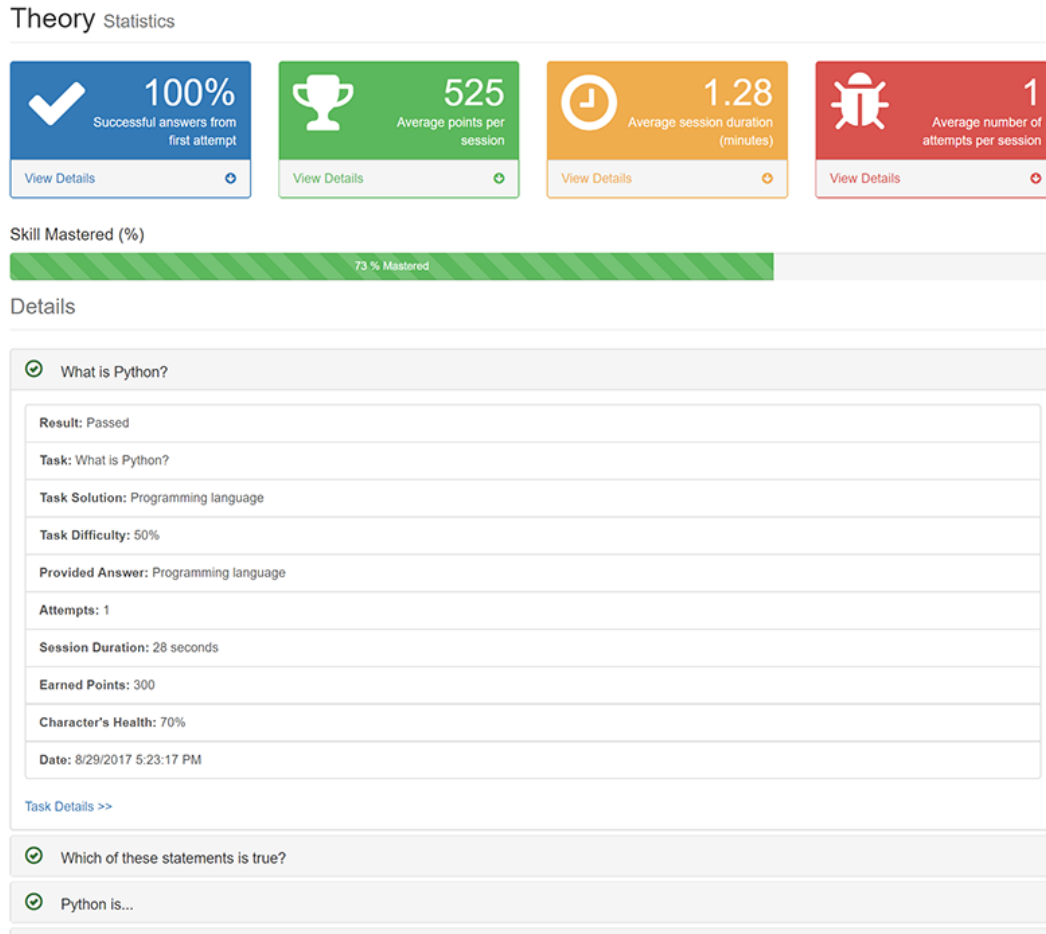


Figure 4.18.: Student statistics and activity details.

4. Implementation Details and Showcase Scenario

- Player's position and time when a player presses the 'Shoot' button
- Information about player's attack
 - Successful or not
 - Enemy type
 - Enemy position
 - Time
 - Damage
- Information about enemies' attacks
 - Successful or not
 - Player's position
 - Time
 - Damage
- Items picked up
 - Item type
 - Position
 - Time

Practical part:

- Tasks
 - Success/failure
 - Difficulty
 - Provided answer
 - Number of attempts
- Session duration
- Earned points
- Provided code
- Selected command to be dragged and dropped
- Run button action
 - No errors: store provided code
 - Errors present: store provided code and errors
 - Produced code for robot movement and sequence of steps that robot takes

4. Implementation Details and Showcase Scenario

- Tab name when one of the tabs is selected
- Deleted code - block of the code when being dragged to the trash can
- Obstacles and their positions
- Disk position
- Robot collided with an obstacle
 - Time
 - Obstacle
- Keyboard
- Show/Hide coding interface

Other:

- Screen Taps
 - Screen position
 - Time
- Name of the new screen
- Sounds enabled/disabled
- Character properties(customization)
- Inventory

4.2.6. Practical Implementation of the CAT Algorithm

The implementation of the CAT algorithm in the sCool project is based on details provided by Linacre et al. (2000), who described core steps for the practical adaptive testing. The original code has been modified and adapted to the .NET standards. The steps and the implementation details of the CAT algorithm are shown below:

1. The process starts and initial values are set where:
 - initial difficulty D is set to the middle value of the difficulty scale.
 - $L=0$, total items taken
 - $H=0$, total difficulties used
 - $R=0$, total number of right answers

4. Implementation Details and Showcase Scenario

2. Finding the closest item for provided difficulty

```
public TheoryTask GetNewTheoryTask(Skill skill, double difficulty)
{
    double minVal = double.MaxValue;
    TheoryTask minItem = null;

    foreach (TheoryTask t in skill.TheoryTasks)
    {
        var difference = System.Math.Abs(difficulty
            - t.Difficulty);
        if (difference < minVal)
        {
            minVal = difference;
            minItem = t;
        }
    }
    return minItem;
}
```

3. Setting the D value to the difficulty of the new item that has been retrieved in the previous step.
4. Exposing the selected item to the user in a certain form depending on the item type.
5. In this step, the response is being obtained from the user.
6. Scoring the response and sending the results back to the server.
7. Total items taken are counted: $L = L + 1$.
8. In the case that response is incorrect, item difficulty is updated: $D = D - 2/L$.
9. If response is correct, item difficulty is updated: $D = D + 2/L$.
10. If response is correct, right answers are counted: $R = R + 1$.
11. If termination criterion is not satisfied, the process is continued from step 2.
12. If termination criterion is satisfied, the following steps are being performed.
13. Wrong answers calculation: $W = L - R$.
14. Measure estimation: $B = H/L + \log_e(R/W)$

4. Implementation Details and Showcase Scenario

15. Error estimation: $S = \sqrt{L/(R*W)}$. As division by 0 and logarithm of 0 are undefined, it is necessary to modify measure and error estimations from this and the previous step.

```
if (W == 0)
{
    B = H / L + System.Math.Log((R - 0.5f) / (W + 0.5f));
    S = L / ((R - 0.5f) * (W + 0.5f));
}
else if (R == 0)
{
    B = H / L + System.Math.Log((R + 0.5f) / (W - 0.5f));
    S = L / ((R + 0.5f) * (W - 0.5f));
}
else
{
    B = H / L + (System.Math.Log(R / W));
    S = L / (R * W);
}
```

16. Comparison of estimated measure B with pass/fail standard T.
17. If B value is between (T - S) and (T + S), then step 2 is performed
18. If $(B - S) > T$, then pass. Additionally, if the knowledge measure is above the threshold value, this means that student has sufficient level of knowledge and the new content can be unlocked.

```
if ((B - S) > T)
{
    //check if new skill should be unlocked
    if(skill.GetMeasure() > unlockLimit){

        //unlock next skill
        UnlockNextSkill();

        //reset variables
        ReInit();
    }
}
```

4. Implementation Details and Showcase Scenario

}

19. If $(B + S) < T$, then fail.
20. After the process is being completed, everything starts from the step 1.

IRT Rasch Model takes one parameter, an ability level value, and produces the measurement in *logits*. The ability level value is firstly scaled from 1-100 down to -3 to 3 logits for the Rasch model (see Listing 4.6). The CAT algorithm starts by loading calibrated item bank and after that, the steps 3-5 are being repeated until a termination criterion is being satisfied. Once measure estimation reaches unlocking limit for the next skill, the next available skill from the skill tree will be unlocked.

Based on the student's answers and the knowledge level, corresponding tasks will be provided to the students. An example of provided tasks with different difficulty levels can be seen in Figure 4.19. In the beginning, the tasks are pretty simple and as students progress, tasks are becoming more and more challenging. In the opposite case, when students are facing difficulties, an easier task will be provided. In these situations, instructors will get an insight into the student's progress through the web platform (see Figure 4.18) and help students with specific tasks which students were not able to solve.

4.3. Summary

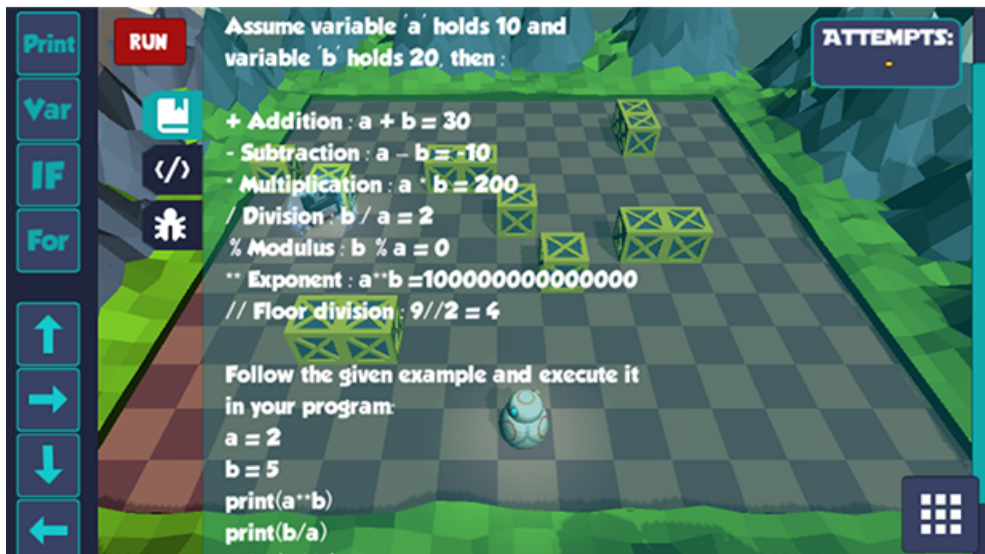
Based on the requirement analysis, the core game and the web platform components have been implemented. The implementation was performed in the form of modules in order to make the system upgradeable and extendable.

For supporting various functionalities such as the course creation and the students' analytics, it was necessary to create the web platform architecture as well as the database scheme. The web platform has been created in the ASP.NET web framework and hosted on the Microsoft Azure cloud platform. The web interfaces for the creation of courses, categories, and tasks have

4. Implementation Details and Showcase Scenario



a)



b)

Figure 4.19.: Task difficulties of 50% (a) and 90% (b) based on student's knowledge level.

4. Implementation Details and Showcase Scenario

```
1 double RaschModel(double itemDifficulty, double abilityLevel)
2 {
3     //Scaling
4     double theta = ScaleDifficulty(abilityLevel);
5     double b = ScaleDifficulty(itemDifficulty);
6
7     double result = (System.Math.Exp(theta - b) / (1 + System.
8         Math.Exp(theta - b)));
9
10    return result;
11 }
12 double ScaleDifficulty(double value)
13 {
14     int[] r1 = { 1, 100 };
15     int[] r2 = { -3, 3 };
16     return (value - r1[0]) * (r2[1] - r2[0]) / (r1[1] - r1[0])
17         + r2[0];
18 }
```

Listing 4.6: Rasch model function and difficulty scaling

been implemented. With these functionalities, the creation of additional courses is a very simple task. Every selected course is offering functionalities for course management and students' analytics where instructors can see more information about students' progress. By using analytics tools, they can observe which concepts were easy or challenging for students to master. One of the main components of the web platform are the web APIs which are processing requests and retrieving data in the JSON format. Every student's answer is stored in the database and processed by the CAT algorithm.

The video game has been created in the Unity game engine. Once users open the game, all the courses are loaded from the server and displayed within the game. Students can select one of the available courses, choose the theoretical or the practical mode and play one of the displayed levels. The levels have been procedurally generated and DGB principles have been utilized in order to keep players in the natural flow. This means that some game parameters such as the health, damage, and the map size are dynamically adjusted based on the player's progress.

4. Implementation Details and Showcase Scenario

Actions and statistics, such as opening a new screen, taps on the screen, ratio of successful shoots, session duration or provided answer are stored in the form of logs. When the internet connection is available, the logs are sent to the server. These data can be further processed and visualized for the better understanding of players and their preferences.

5. Evaluation

There are different factors which make educational tools usable and appealing to users such as ease of use, modularity, configurability, look and feel and other benefits for users. The plan for this study was to find out whether the web application satisfies usability requirements and provides a positive experience when being used in supporting mobile STEM education. Even though the mobile game is inseparable part in the sCool project, only the study for the web application was conducted, as Miloš Kojić performed the study for the mobile game in his work (Kojić, 2017).

5.1. Participants

The target group for the web application are high school and university instructors which are aimed to use this educational software. Therefore, the participants which were involved in this study are instructors from the Graz University of Technology, the University of Westminster and the RMIT University.

5.2. Setup and Procedure

The survey has been conducted by using a software tool *LimeSurvey*¹. This tool enables users to run and analyze online surveys. It provides a wide variety of question types which makes it suitable for use in this study. After preparing the questionnaire, instructors from the Graz University of

¹*LimeSurvey* (2017)

5. Evaluation

Technology, the University of Westminster and the RMIT University were recruited to test the web application and provide their feedback. These participants were asked to use the web application, for which they received the credentials of the web application and afterwards the set of survey questions in English. The process of testing and filling out the survey lasted approximately between 20-25 minutes.

In order to understand how well the prototype of the web platform is accepted by the participants, the two standardized scales CES (Kay & Lovelock, 2008) and SUS (Brooke et al., 1996) were used with the combination of some system-specific questions of the web platform. The questionnaires are shown in the Appendix section.

5.2.1. Computer Emotion Scale

CES is a reliable approach for assessing emotions of the users while learning how to use a new software. Four emotion constructs have been selected for this purpose with the corresponding feelings:

- Anger (Irritable, Frustrated, Angry)
- Anxiety (Anxious, Insecure, Helpless, Nervous)
- Happiness (Satisfied, Excited, Curious)
- Sadness (Disheartened, Dispirited)

Participants are provided with questions about feelings they experienced during the testing process. There are the 12 feelings which can be rated on a Likert scale from 0 to 3, where each of the feelings is a part of one construct. The interpretation of the results can be achieved by summing up result values for each construct and performing a normalization by dividing it by the number of feelings inside of the each construct (Kay & Lovelock, 2008). This scale has been selected for a better understanding of the emotions which users experienced during the testing process of the prototype and this makes it possible to investigate which emotions were the most frequent during the study.

5. Evaluation

5.2.2. System Usability Scale

When specifying the usability of the system it is necessary to define who is going to use the system, what kind of tasks will be performed and what are the characteristics of an environment in which it will be used. With increasing demands for simple and effective evaluation system which will cover usability measures, the System Usability Scale has been designed (Brooke et al., 1996). SUS is a ten-item questionnaire scale which can provide a single number which represents a measure of the overall usability of the system. It is based on the 5 point *Likert scale* with possible answers such as *strongly agree, agree, undecided, disagree* and *strongly disagree*.

The ratings for odd item questions 1,3,5,7, and 9 are subtracted by one and their results are summed up. The similar procedure is performed with even item questions 2,4,6,8, and 10 where their ratings are subtracted from number 5 and summed up. The overall value of the SUS is obtained by multiplying the sum of scores by 2.5 which finally results in the score range of 0-100. According to (Bangor, Kortum, & Miller, 2009), the score of 70 is the lower limit for the system to pass the usability test.

5.2.3. System Specific Questions for the Web Platform

For a better understanding of participants' previous experiences with game-based teaching software, some system specific questions have been prepared. Furthermore, the participants were able to provide a feedback about the software and to suggest some further improvements. The participants were also queried to provide their opinions if they could potentially use the software in their courses in the future. From this part, the specific preferences and users' personal opinions about the web platform should be revealed.

5.3. Results

This chapter discusses the results of the study described in the previous chapter. It outlines the evaluation results of the computer emotion scale,

5. Evaluation

system usability scale, and the system specific questions. Furthermore, the discussion includes some suggestions for improvements. These suggestions are based on the provided feedback by participants who tested the web application and filled out the questionnaire.

5.3.1. Participants

Ten participants took part in the study where 8 (80%) out of 10 were males and 2 (20%) of them were females. The participants were senior lecturers (20%), teaching assistants (60%), and tutors (20%) from the Graz University of Technology, the University of Westminster, and the RMIT University. The average age was 29.3 years with the standard deviation of 4.12. All of the participants stated that they are familiar with e-learning tools, where 60% of them are experienced and 40% are very experienced. However, when it comes to using web platforms with the purpose of the content creation for mobile educational games, the results are opposite. The majority, 60% of participants stated that they are not experienced at all, 30% are not experienced and 10% are somewhat experienced with this kind of tools.

5.3.2. Computer Emotion Scale

Table 5.1 contains the results of the CES which are grouped by the four main emotions. The table shows the number of times an answer on the CES has been selected with corresponding percentages. All the results are on the scale between 0 (none of the time) and 3 (all of the time) as it is explained in the section 5.2.1.

The figure 5.1 summarizes the average ratings of all users per emotion with corresponding standard deviations. According to these results, *sadness* got the lowest score of 0.05 (SD 0.16) where only one participant selected an answer associated with this emotion. *Anxiety* was experienced by the two survey participants. They had feelings associated with this emotion at least some of the time, which led to an average rating of 0.10 (SD 0.24). The similar situation is with the *Anger* emotion. This emotion was experienced by two out of ten participants at least some of the time, which led to an

5. Evaluation

| | None of the time | Some of the time | Most of the time | All of the time |
|------------------|--------------------|-------------------|-------------------|--------------------|
| satisfied | 0 (0.00%) | 2 (20.00%) | 3 (30.00%) | 5 (50.00%) |
| excited | 2 (20.00%) | 3 (30.00%) | 2 (20.00%) | 3 (30.00%) |
| curious | 0 (0.00%) | 2 (20.00%) | 3 (30.00%) | 5 (50.00%) |
| HAPPINESS | 2 (6.67%) | 7 (23.33%) | 8 (26.67%) | 13 (43.33%) |
| disheartened | 9 (90.00%) | 1 (10.00%) | 0 (0.00%) | 0 (0.00%) |
| dispirited | 10 (100.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| SADNESS | 19 (95.00%) | 1 (5.00%) | 0 (0.00%) | 0 (0.00%) |
| anxious | 10 (100.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| insecure | 10 (0.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| helpless | 9 (90.00%) | 0 (0.00%) | 0 (0.00%) | 1 (10.00%) |
| nervous | 9 (90.00%) | 1 (10.00%) | 0 (0.00%) | 0 (0.00%) |
| ANXIETY | 38 (95.00%) | 1 (2.50%) | 0 (0.00%) | 1 (2.50%) |
| irritable | 9 (90.00%) | 0 (0.00%) | 1 (10.00%) | 0 (0.00%) |
| frustrated | 9 (90.00%) | 1 (10.00%) | 0 (0.00%) | 0 (0.00%) |
| angry | 10 (100.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| ANGER | 28 (93.34%) | 1 (3.33%) | 1 (3.33%) | 0 (0.00%) |

Table 5.1.: Number of times an answer on the Computer Emotion Scale has been selected

5. Evaluation

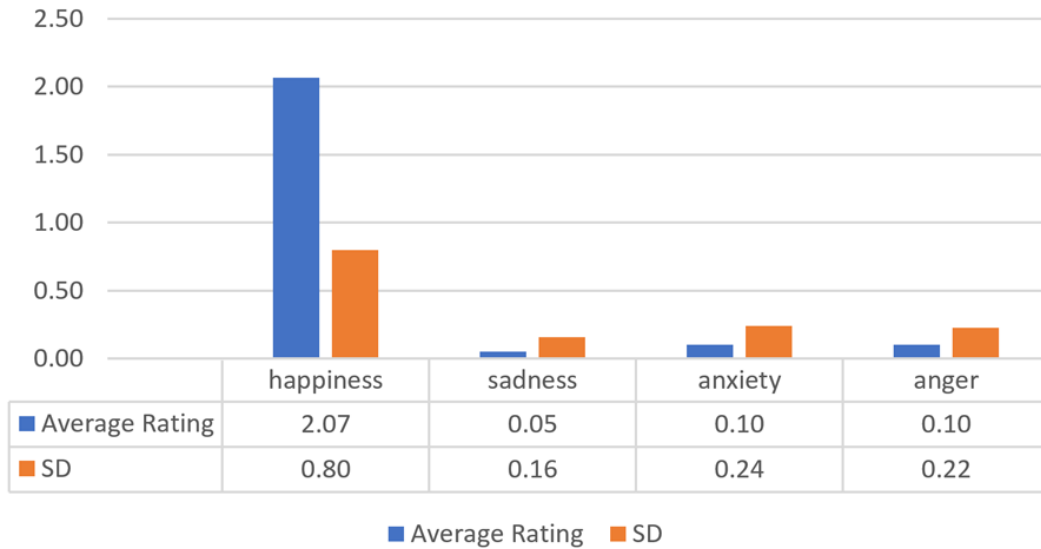


Figure 5.1.: Average ratings of all users per emotion with corresponding standard deviation values.

average rating of 0.10 (SD 0.22). Value of 0.10 for both anxiety and anger shows that study participants did not have very strong feelings associated with these emotions. The strongest emotion was *happiness* which achieved the average score of 2.07 (SD 0.80). The majority of participants experienced feelings with this emotion at least some of the time. However, there are two participants who stated that they did not experience this emotion at all during the testing.

Also, the distribution of the possible answers grouped by the four emotions could be derived from the Table 5.1. The option *none of the time* was selected by the majority of the study participants for emotions *sadness* (95.00%), *anxiety* (95.00%), and *anger* (93.34%) while for the *happiness* emotion it was only 6.67%. The *happiness* emotion was experienced all of the time for 43.33% of the participants while options *some of the time* (23.33%) and *most of the time* (26.67%) had similar distribution share.

5. Evaluation

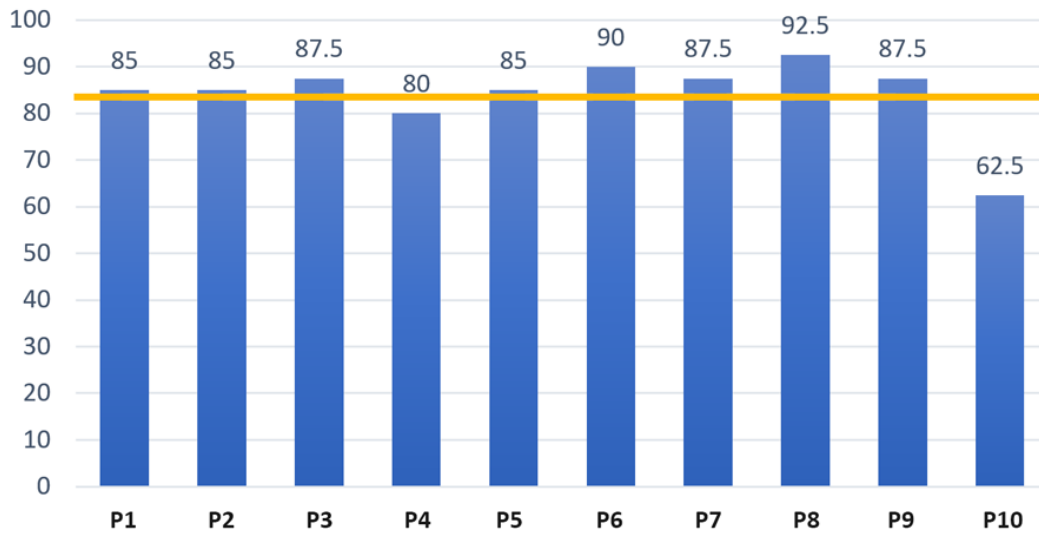


Figure 5.2.: Results of the SUS per participant P1 – P10 with the average score of 84.25 and SD of 8.34.

5.3.3. System Usability Scale

The figure 5.2 shows an overview of the SUS results. It could be seen that participants rated the system as easy to use, however, 10% of them answered that they would not like to use the software in the future. All the participants strongly disagreed or disagreed that the system was unnecessarily complex and that there was too much inconsistency in this system. Eighty percent of participants think that the various functions in this system were well integrated and no one found the system very cumbersome to use. Majority of the users think that they did not need to learn many things before they could get going with the system and that other people would learn to use this system very quickly.

After applying the procedure for calculating the SUS score from the section 5.2.2, on the data from the Table 5.2, the resulting SUS score is 84.25 with the standard deviation of 8.34 (see Figure 5.2). The result of 84.25 is over the limit of 70, which is the minimal result for the usable system Bangor et al. (2009). This means that the overall system satisfies the usability criteria. The results do not vary strongly, and most of the values are pretty

5. Evaluation

close to the average result except for one participant which has the score of 62.5 while the largest score is 92.5.

5.3.4. System Specific Questions and Other Feedback by Participants

The participants generally saw the web platform as a tool which they could potentially use for creating mobile courses for students. The majority of participants would use it in their courses, except the one participant who expressed the negative opinion. Some of the features or improvements they would like to see are more advanced analytics, improved instruction system and a pie-chart diagram which could show how many students successfully completed selected task. Also, there were no major issues reported regarding the content creation and the page navigation.

Many positive comments were given regarding the appearance of the web platform, analytics tools, and the navigation system. One of the study participants wrote: *"It's an easy and straight-forward way for educators who are not technically trained to use serious games."*

Participants answered that they are familiar with using mobile educational games but no one selected an option of being very experienced. Some of the mobile educational games they know and have used are *PocketCode*, *Quantum Moves*, *Kahoot*, *Kodo*, *DragonBox*, and *Lightbot*. All the participants stated that they have experience with web e-learning systems and some of the e-learning systems they used are *Moodle*, *TeachCenter*, *Blackboard*, *Canvas*, and *TTM*. However, 60% of the participants stated that they have no experience at all with educational web platforms for mobile game content creation and adaptive learning. Also, no one selected *experienced* and *very experienced* options. This means that study participants are not very experienced with systems like the sCool system is.

5. Evaluation

| | Strongly disagree | Disagree | Undecided | Agree | Strongly agree |
|--|-------------------|------------|------------|------------|----------------|
| I think that I would like to use this system frequently | 1 (10.00%) | 0 (0.00%) | 1 (10.00%) | 7 (70.00%) | 1 (10.00%) |
| I found the system unnecessarily complex | 5 (50.00%) | 5 (50.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| I thought the system was easy to use. | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) | 5 (50.00%) | 5 (50.00%) |
| I think that I would need the support of a technical person to be able to use this system. | 6 (60.00%) | 3 (30.00%) | 1 (10.00%) | 0 (0.00%) | 0 (0.00%) |
| I found the various functions in this system were well integrated. | 0 (0.00%) | 0 (0.00%) | 2 (20.00%) | 5 (50.00%) | 3 (30.00%) |
| I thought there was too much inconsistency in this system. | 8 (80.00%) | 2 (20.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| I would imagine that most people would learn to use this system very quickly. | 0 (0.00%) | 0 (0.00%) | 1 (10.00%) | 5 (50.00%) | 4 (40.00%) |
| I found the system very cumbersome to use. | 7 (70.00%) | 3 (30.00%) | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) |
| I felt very confident using the system. | 0 (0.00%) | 0 (0.00%) | 0 (0.00%) | 6 (60.00%) | 4 (40.00%) |
| I needed to learn a lot of things before I could get going with this system. | 4 (40.00%) | 5 (50.00%) | 1 (10.00%) | 0 (0.00%) | 0 (0.00%) |

Table 5.2.: Number of times an answer on the System Usability Scale has been selected

5. Evaluation

5.3.5. Discussion and Limitations

Based on the feedback of the study participants, the prevailing feeling was happiness and the system satisfies usability criteria. The study participants were in general satisfied with the functionalities and the visual appearance of the software. There were no reported missing features nor some major issues which might prevent them from using the software.

However, the number of ten study participants is not enough to draw highly relevant conclusions. Also, only university instructors participated in the study and there were no high school instructors due to organizational and time constraints, but in some further studies it is necessary to have high school instructors involved as well. Therefore, the provided results can be used to detect problems that occurred during the study and this study can serve as a foundation for further research.

As the majority of the participants are not the native English speakers and most of them are not very familiar with this kind of educational tools, these circumstances might have influenced their overall experience with the software and the final study results.

Even though the system contains instructions which explain the main functionalities, the conclusion from the user feedback is that the instruction system needs to be improved and study participants need more time to get familiar with this kind of educational tools. Also, additional studies about educational effectiveness of the software are necessary to be performed in the future.

6. Lessons Learned

This chapter summarizes the experiences and the issues which were encountered during different project phases. It includes conclusions about the literature, project implementation, and evaluation phases.

6.1. Literature

Based on the project's goals to create a multidisciplinary educational mobile game, different theoretical aspects and application domains have been researched. First of all, the analysis of already existing game-based learning systems and their characteristics has been performed in order to better understand the current state in this field. The second step was to analyze learning methodologies, as understanding of different learning methodologies was important for planing and designing an educational game.

Concepts such as the dynamic game balancing, adaptive learning, and procedural generation have been analyzed as well. Dynamic game balancing is important for keeping players in the flow model while adaptive learning adapts the educational content to student's knowledge level which results in increasing motivation and engagement in students. Procedural generation was very useful as this approach enables instructors to focus themselves primarily on the course creation.

Video games have shown a great pedagogical potential and they are already used by broad masses for learning various STEM disciplines on large-screen devices. Even though mobile devices have rapidly growing trends, there is an obvious lack of mobile educational tools for teaching more advanced concepts for high school and first-year university students.

6. Lessons Learned

Therefore, the proposed solution was the creation of a multidisciplinary educational mobile game so the players can explore different STEM disciplines and learn following their own pace. The programming is selected as a starting discipline for the purpose of the thesis, but the project can be extended to other disciplines as well.

6.2. Implementation

After the analysis of already existing educational games and aspects which could facilitate the learning process, it was decided to make the project with two main components.

The first one is the multidisciplinary educational mobile game where students can learn different STEM disciplines. Some of the initial challenging questions were: "How to effectively present educational content on small-screen devices?", "How to dynamically integrate different courses?" and "How to make an interesting video game so users do not get bored?". Literature review and early-stage testing with users helped to get answer on these questions. As one of the goals is to reach as many as possible users, the necessary decision is to select an environment which could make the game be a multi-platform for supporting different platforms such as the iOS, Android, and Windows Phone. The Unity game engine has been used as it showed to be capable of satisfying defined requirements. Another important decision was about the way how to execute the code that students have to type. There were two possibilities, to use some external resource or to try to make the execution locally on the phone. Fortunately, the authors of the thesis managed to integrate IronPython and to have Python code execution on mobile devices.

The second component of the project is the web platform where instructors can create courses and track student's progress. The first challenges were to select the proper technology and to design the database scheme. ASP.NET web framework comes with a number of components which speed up the development process such as the registration module and user interface libraries. EF code-first approach was a good choice as it enabled the database version control and the validation on the model side. This means that there

6. Lessons Learned

was no need to perform the content validation on every single place while working with entities. Microsoft Azure cloud platform provided all the necessary resources for easy-deploy, scalability, and web APIs. The web platform has a clean and modern design. It is responsive and comes with instructions which helped users to use the system.

6.3. Outcome

Different aspects have been covered in the evaluation process. For the evaluation process, it was necessary to make the web platform publicly available so participants from other institutions could be involved as well. The web platform was hosted on the Microsoft Azure cloud platform which provided all the necessary infrastructure including a sub-domain. This has been achieved with an initial credit for students which was very practical during the implementation and testing phases. The mobile game was published to the Google Play Store for the purpose of the beta testing and by doing this, many potential issues regarding the manual software installation were avoided.

The theoretical part of the evaluation was focused on previous experience with game-based learning software. The practical part rated the web application itself according to usability of the platform, emotions participants experienced during the testing process, functionality and general aspects. The results showed that the web platform is well received by the instructors and they are willing to use it in the future. It is positive to see that game-based teaching approach brought new experiences to users. According to the testing results, the software satisfies the usability criteria. Also, prevailing feelings that participants experienced were positive.

Finally, it is important that all suggestions for the improvement are regarding some details and that no one expressed missing features which might be of very high importance nor some unnecessary functionalities.

7. Conclusion and Future Work

This chapter provides the summary and the outlook of the project. Ideas for future work and further improvements of the project will be discussed as well.

7.1. Conclusion

Technology became an important part of almost every aspect of human lives. The knowledge of the STEM disciplines is becoming necessary for more and more jobs. However, the process of the STEM education is for many students very challenging which leads to demotivation and poor performances. Therefore, educators were exploring different approaches in order to find a solution which could increase motivation and engagement with the STEM disciplines. Digital game-based learning is an approach which has shown the positive impact on educational effectiveness and student motivation.

As a result of the new technological advancements, mobile and adaptive learning approaches have been developed. Adaptive learning is attracting more and more attention in educational community as it is capable of adapting educational content to a student's knowledge level. Large-screen devices are supported by a number of educational mobile STEM games, but the need for mobile STEM games might increase following the rapid growth of mobile trends. Therefore, this thesis proposes the sCool system as a solution for increasing motivation and engagement of high school and first-year university students with the STEM disciplines. It consists of the multidisciplinary educational mobile game and the web platform. The web platform is intended to be used by educators for course creation and

7. Conclusion and Future Work

tracking students' progress. Once the educational content is ready, it can be used in the video game where students can select one of the listed courses, learn theoretical concepts, and practice their skills. The first implemented STEM module is programming where users can write and execute Python code on mobile devices. The video game is created to be multi-platform and capable of supporting both active and adaptive learning. The game comes with the implementation of dynamic balancing principles of the game difficulty and the difficulty of an educational content in order to keep players engaged. Procedural content generation algorithms are utilized for a map, content and sound generation so instructors can focus themselves only on educational content creation. Another benefit for instructors is that they can get an insight into the progress of every student and can focus their time only on concepts that students failed to grasp. The main technical components of the project are the Microsoft Azure cloud platform and the Unity game engine as they are capable of supporting a wide variety of project requirements.

The prototype of the system was tested by ten study participants from the Graz University of Technology, the University of Westminster, and the RMIT University. The study results show that the system satisfies the criteria of the System Usability Scale with an average score of 84.25. Also, happiness was the prevailing feeling on the Computer Emotion Scale while negative feelings such as the sadness, anxiety, and anger got very low score values. The study participants generally saw the web platform as an interesting tool which they could use up to some extent in their courses.

7.2. Future Work

There are some additional functionalities which should be improved and features that could be implemented in the project. The web platform should have more created courses which can be incorporated into the game and already existing ones should be extended and revised. Also, it should support a creation of different question types and the functionality for reordering of course, skill and task items. Based on the feedback provided by questionnaire participants, each task item should display a graphical

7. Conclusion and Future Work

representation of how many students have taken the selected task and what were the results. The other suggestion derived from the user feedback is that the activity diagram should display not only the number of active students for the selected day, but the complete list of their names, so educators can see who exactly used the software for the selected day. The technical implementation of the adaptive learning module already exist but the missing part is a calibrated item set.

The video game should be able to support additional modules for other disciplines. Some of the necessary components which should be implemented are a collaboration tool where students can chat and discuss given tasks among the group and a note system where they can write and save the information that they find important. Also, additional improvements could be performed on the enemy AI, dynamic game balancing and spaced repetition components. The completion of the multilayer module should be performed as well. It could be interesting to create a functionality of restricting a minimal and maximal number of condition, loop, variable and movement controls in the code. The server-side implementation already exists, so the only missing part is a client-side functionality.

One of the necessary future tasks is to perform a more detailed testing of learnability and educational effectiveness of the project which, unfortunately, was not performed due to the time and organizational constraints.

Appendix

References

- Alice. (2017). *alice.org*. Retrieved 2017-03-02, from <https://www.alice.org/>
- Amazon web services. (2017). Retrieved 2017-04-08, from <https://aws.amazon.com/>
- Apple app store. (2017). Retrieved 2017-03-15, from <https://itunes.apple.com>
- Asp.net. (2017). Retrieved 2017-04-06, from <https://www.asp.net/>
- ASP.NET MVC. (2017). *Asp.net mvc*. Retrieved 2017-05-23, from <https://www.asp.net/mvc>
- Awad, M. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*.
- Babbel. (2017). Retrieved 2017-03-11, from <http://www.babbel.com>
- Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3), 114–123.
- Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1), 19.
- Betz, J. A. (1995). Computer games: Increase learning in an interactive multidisciplinary environment. *Journal of Educational Technology Systems*, 24(2), 195–205.
- Bootstrap. (2017). *Bootstrap*. Retrieved 2017-07-15, from <https://getbootstrap.com/>
- Brockmyer, J. H., Fox, C. M., Curtiss, K. A., McBroom, E., Burkhart, K. M., & Pidruzny, J. N. (2009). The development of the game engagement questionnaire: A measure of engagement in video game-playing. *Journal of Experimental Social Psychology*, 45(4), 624–634.
- Brooke, J., et al. (1996). Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194), 4–7.
- Browne, K., & Anand, C. (2013). Gamification and serious game approaches for introductory computer science tablet software. In *Proceedings of the*

References

- first international conference on gameful design, research, and applications* (pp. 50–57).
- Bruffee, K. A. (1993). *Collaborative learning: Higher education, interdependence, and the authority of knowledge*. ERIC.
- Burton-Chellew, M. N., Ross-Gillespie, A., & West, S. A. (2010). Cooperation in humans: competition between groups and proximate emotions. *Evolution and Human behavior*, 31(2), 104–108.
- Caillois, R. (1961). *Man, play, and games*. University of Illinois Press.
- Callaghan, M. J., McCusker, K., Losada, J. L., Harkin, J., & Wilson, S. (2013). Using game-based learning in virtual worlds to teach electronic and electrical engineering. *IEEE Transactions on Industrial Informatics*, 9(1), 575–584.
- Chang, M., Kuo, R., Chen, G.-D., Hirose, M., et al. (2009). *Learning by playing. game-based education system design and development: 4th international conference on e-learning, edutainment 2009, banff, canada, august 9-11, 2009, proceedings* (Vol. 5670). Springer Science & Business Media.
- Chen, S., & Zhang, J. (2008). The adaptive learning system based on learning style and cognitive state. In *Knowledge acquisition and modeling, 2008. kam'08. international symposium on* (pp. 302–306).
- Code Combat. (2017). *codecombat.com/*. Retrieved 2017-03-03, from <https://codecombat.com/>
- Code Hunt. (2017). *microsoft.com/en-us/research/project/code-hunt/*. Retrieved 2017-03-03, from <https://www.microsoft.com/en-us/research/project/code-hunt/>
- Code hunt. (2017). Retrieved 2004-05-27, from <https://www.codehunt.com/>
- CodeMonkey. (2017). *playcodemonkey.com*. Retrieved 2017-03-04, from <https://www.playcodemonkey.com/>
- CodeSpells. (2017). *codespells.org/*. Retrieved 2017-03-03, from <https://codespells.org/>
- CodinGame. (2017). *codingame.com/*. Retrieved 2017-03-04, from <https://www.codingame.com>
- Colobot. (2017). *ceebot.com/colobot*. Retrieved 2017-03-02, from <http://www.ceebot.com/colobot>
- Construct 2. (2017). Retrieved 2017-04-05, from <https://www.scirra.com/construct2>
- Cooper, P. A. (1993). Paradigm shifts in designed instruction: From behaviorism to cognitivism to constructivism. *Educational technology*, 33(5),

References

- 12–19.
- Cordova, D. I., & Lepper, M. R. (1996). Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of educational psychology, 88*(4), 715.
- Crawford, C. (1984). The art of computer game design.
- Čubranić, D., & Storey, M. A. D. (2005). Collaboration support for novice team programming. In *Proceedings of the 2005 international acm siggroup conference on supporting group work* (pp. 136–139).
- De Ayala, R. J. (2013). *The theory and practice of item response theory*. Guilford Publications.
- De Freitas, S. I. (2006). Using games and simulations for supporting learning. *Learning, media and technology, 31*(4), 343–358.
- Desurvire, H., Caplan, M., & Toth, J. A. (2004). Using heuristics to evaluate the playability of games. In *Chi'04 extended abstracts on human factors in computing systems* (pp. 1509–1512).
- Desurvire, H., & Wiberg, C. (2009). Game usability heuristics (play) for evaluating and designing better games: The next iteration. In *International conference on online communities and social computing* (pp. 557–566).
- Deterding, S., Sicart, M., Nacke, L., O'Hara, K., & Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In *Chi'11 extended abstracts on human factors in computing systems* (pp. 2425–2428).
- Dewey, J. (1906). The experimental theory of knowledge. *Mind, 15*(59), 293–307.
- Dickey, M. D. (2005). Engaging by design: How engagement strategies in popular computer and video games can inform instructional design. *Educational Technology Research and Development, 53*(2), 67–83.
- Django. (2017). Retrieved 2017-04-07, from <https://www.djangoproject.com/>
- Dondlinger, M. J. (2007). Educational video game design: A review of the literature. *Journal of applied educational technology, 4*(1), 21–31.
- Douglas, D. (2017). *David douglas*. Retrieved 2017-05-28, from <https://github.com/deadlyfingers>
- Dragonbox. (2017). Retrieved 2017-03-15, from <http://dragonbox.com/>
- DreamBox. (2017). *dreambox.com*. Retrieved 2017-03-08, from <http://www.dreambox.com>
- Duolingo. (2017). *duolingo.com*. Retrieved 2017-03-27, from <https://www.duolingo.com/>

References

- Ebbinghaus, H. (1913). *Memory: A contribution to experimental psychology* (No. 3). University Microfilms.
- Ertmer, P. A., & Newby, T. J. (2013). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quarterly*, 26(2), 43–71.
- Express. (2017). Retrieved 2017-04-07, from <https://expressjs.com/>
- Eyewire. (2017). *eyewire.org*. Retrieved 2017-03-13, from <http://eyewire.org>
- Federoff, M. A. (2002). *Heuristics and usability guidelines for the creation and evaluation of fun in video games* (Unpublished doctoral dissertation). Indiana University Bloomington.
- Flor, N. V., & Hutchins, E. L. (1991). A case study of team programming during perfective software maintenance. In *Empirical studies of programmers: Fourth workshop* (p. 36).
- Flot. (2017). *flotcharts.org/*. Retrieved 2017-07-16, from <http://www.flotcharts.org/>
- Foldit. (2017). *fold.it*. Retrieved 2017-03-13, from <https://fold.it>
- Font Awesome. (2017). *fontawesome.io*. Retrieved 2017-07-17, from <http://fontawesome.io/>
- Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment (CIE)*, 1(1), 20–20.
- Google cloud. (2017). Retrieved 2017-04-08, from <https://cloud.google.com/>
- Google play. (2017). Retrieved 2017-03-15, from <https://play.google.com>
- Hacked. (2017). *Hacked*. Retrieved 2017-03-05, from <https://play.google.com/store/apps/details?id=com.hackedapp>
- Hackerrank. (2017). *hackerrank.com*. Retrieved 2017-03-27, from <https://www.hackerrank.com/>
- Hambleton, R. K., & Swaminathan, H. (2013). *Item response theory: Principles and applications*. Springer Science & Business Media.
- Hung, D. (2001). Theories of learning and computer-mediated instructional technologies. *Educational Media International*, 38(4), 281–287.
- Hunicke, R., & Chapman, V. (2004). Ai for dynamic difficulty adjustment in games. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2–5.
- Hunicke, R., LeBlanc, M., & Zubek, R. (2004). Mda: A formal approach to game design and game research. In *Proceedings of the aaii workshop on*

References

- challenges in game ai* (pp. 04–04).
- Johnson, D. W., & Johnson, R. T. (1994). *Learning together and alone. cooperative, competitive, and individualistic learning*. ERIC.
- Json.NET. (2017). *Json.net*. Retrieved 2017-05-23, from <https://www.newtonsoft.com/json>
- Kalyuga, S. (2007). Enhancing instructional efficiency of interactive e-learning environments: A cognitive load perspective. *Educational Psychology Review, 19*(3), 387–399.
- Kay, R. H., & Loverock, S. (2008). Assessing emotions related to learning new software: The computer emotion scale. *Computers in Human Behavior, 24*(4), 1605–1623.
- Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences, 47*, 1991–1999.
- Kerr, P. (2015). Adaptive learning. *Elt Journal, 70*(1), 88–93.
- Khatib, F., DiMaio, F., Cooper, S., Kazmierczyk, M., Gilski, M., Krzywda, S., ... others (2011). Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology, 18*(10), 1175–1177.
- Knewton. (2017). *knewton.com*. Retrieved 2017-03-08, from <https://www.knewton.com>
- Kojić, M. (2017). *Procedural Generation in Multidisciplinary Educational Video Game* (Unpublished master's thesis). Graz University of Technology, Austria.
- Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development*. FT press.
- Kukulka-Hulme, A., & Traxler, J. (2005). *Mobile learning: A handbook for educators and trainers*. Psychology Press.
- Lainema, T. (2014). Enhancing organizational business process perception: Experiences from constructing and applying a dynamic business simulation game.
- Laravel*. (2017). Retrieved 2017-04-07, from <https://laravel.com/>
- Larochelle, M., Bednarz, N., & Garrison, J. (1999). Constructivism and education.
- Learnpython. (2017). *learnpython.org*. Retrieved 2017-03-27, from <https://www.learnpython.org/>

References

- Lightbot. (2017). *lightbot.com*. Retrieved 2017-03-03, from <https://lightbot.com>
- Limesurvey. (2017). Retrieved 2017-08-21, from <https://www.limesurvey.org/>
- Linacre, J. M., et al. (2000). Computer-adaptive testing: A methodology whose time has come. *Chae, S.-Kang, U.-Jeon, E.-Linacre, JM (eds.): Development of Computerised Middle School Achievement Tests, MESA Research Memorandum(69)*.
- Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Routledge.
- Lutz, M. (2013). *Learning python: Powerful object-oriented programming*. "O'Reilly Media, Inc."
- Malone, T. W. (1980). What makes things fun to learn? heuristics for designing instructional computer games. In *Proceedings of the 3rd acm sigsmall symposium and the first sigpc symposium on small systems* (pp. 162–169).
- Malone, T. W. (1982). Heuristics for designing enjoyable user interfaces: Lessons from computer games. In *Proceedings of the 1982 conference on human factors in computing systems* (pp. 63–68).
- Mayo, M. J. (2009). Video games: A route to large-scale stem education? *Science*, 323(5910), 79–82.
- McConnell, D. (2000). *Implementing computer supported cooperative learning*. Psychology Press.
- McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin*, 34(1), 38–42.
- Mentzelopoulos, M., Parrish, J., Kathrani, P., & Economou, D. (2016). Revr-law: An immersive way for teaching criminal law using virtual reality. In *International conference on immersive learning* (pp. 73–84).
- Michael, D. R., & Chen, S. L. (2005). *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade.
- Microsoft azure. (2017a). Retrieved 2017-03-15, from <http://whyville.net/>
- Microsoft azure. (2017b). Retrieved 2017-04-08, from <https://azure.microsoft.com/>
- Mji, A., & Makgato, M. (2006). Factors associated with high school learners' poor performance: a spotlight on mathematics and physical science. *South African journal of education*, 26(2), 253–266.

References

- Morris.js. (2017). *Morris.js*. Retrieved 2017-07-16, from <http://morrisjs.github.io/morris.js/>
- Mueller, J. P. (2010). *Professional ironpython*. John Wiley & Sons.
- Nawrocki, L. H., & Winner, J. L. (1983). Video games: Instructional potential and classification. *Journal of Computer-Based Instruction*.
- Newzoo. (2017). *newzoo.com*. Retrieved 2017-03-13, from <http://www.newzoo.com>
- Nosek, J. T. (1998). The case for collaborative programming. *Communications of the ACM*, 41(3), 105–108.
- Oblinger, D. (2004). The next generation of educational engagement. *Journal of interactive media in education*, 2004(1).
- Olesen, J. K., Yannakakis, G. N., & Hallam, J. (2008). Real-time challenge balance in an rts game using rtneat. In *Computational intelligence and games, 2008. cig'08. ieee symposium on* (pp. 87–94).
- Oxland, K. (2004). *Gameplay and design*. Pearson Education.
- Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education*, 52(1), 1–12.
- Parshall, C. G. (2002). *Practical considerations in computer-based testing*. Springer Science & Business Media.
- Pew Research Center. (2017). *pewresearch.org*. Retrieved 2017-04-03, from <http://www.pewresearch.org/>
- Photon engine. (2017). Retrieved 2017-06-27, from <https://www.photonengine.com/>
- Pirates, C. (2017). *codingpiratesgame.com/*. Retrieved 2017-03-03, from <https://www.codingpiratesgame.com/>
- Pirker, J., Riffnaller-Schiefer, M., & Gütl, C. (2014). Motivational active learning: engaging university students in computer science education. In *Proceedings of the 2014 conference on innovation & technology in computer science education* (pp. 297–302).
- Playgrounds, S. (2017). *apple.com/uk/swift/playgrounds/*. Retrieved 2017-03-02, from <https://www.apple.com/uk/swift/playgrounds/>
- Prensky, M. (2003). Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1), 21–21.
- Quinn, C. N. (1994). Designing educational computer games. In *Proceedings of the ifip tc3/wg3.2 working conference on the seign, implementation and evaluation of interactive multimedia in university settings: Designing for*

References

- change in teaching and learning* (pp. 45–57). New York, NY, USA: Elsevier Science Inc. Retrieved from <http://dl.acm.org/citation.cfm?id=647111.716300>
- Rieber, L. P. (1996). Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational technology research and development*, 44(2), 43–58.
- Robocode. (2017). *robocode.sourceforge.net*. Retrieved 2017-03-02, from <http://robocode.sourceforge.net/>
- Rosetta stone. (2017). Retrieved 2017-03-11, from <http://www.rosettastone.de/>
- Ruby on rails. (2017). Retrieved 2017-04-07, from <http://rubyonrails.org/>
- Salen, K., & Zimmerman, E. (2004). *Rules of play: Game design fundamentals*. MIT press.
- Savery, J. R., & Duffy, T. M. (1995). Problem based learning: An instructional model and its constructivist framework. *Educational technology*, 35(5), 31–38.
- Schaeffer, J. (2001). A gamut of games. *AI Magazine*, 22(3), 29.
- Schell, J. (2014). *The art of game design: A book of lenses*. CRC Press.
- Schunk, D. H. (1996). Learning theories. *Printice Hall Inc., New Jersey*, 1–576.
- Scratch. (2017). *scratch.mit.edu*. Retrieved 2017-03-02, from <https://scratch.mit.edu>
- Settles, B., & Meeder, B. (2016). A trainable spaced repetition model for language learning. In *Acl (1)*.
- Sharples, M., Taylor, J., & Vavoula, G. (2010). A theory of learning for the mobile age. In *Medienbildung in neuen kulturräumen* (pp. 87–99). Springer.
- Skinner, B. F. (1974). *About behaviorism*. Vintage.
- Sololearn. (2017). *sololearn.com*. Retrieved 2017-03-27, from <https://www.sololearn.com/>
- SpriteBox. (2017). *spritebox.com*. Retrieved 2017-03-02, from <http://spritebox.com/>
- Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E. (2004). Difficulty scaling of game ai. In *Proceedings of the 5th international conference on intelligent games and simulation (game-on 2004)* (pp. 33–37).
- Squire, K. (2003). Video games in education. In *International journal of intelligent simulations and gaming*.
- Squire, K. (2005). *Game-based learning: Present and future state of the field*.

References

- Masie Center e-Learning Consortium.
- Squire, K., Barnett, M., Grant, J. M., & Higginbotham, T. (2004). Electromagnetism supercharged!: learning physics with digital simulation games. In *Proceedings of the 6th international conference on learning sciences* (pp. 513–520).
- Stevens, R., Soller, A., Giordani, A., Gerosa, L., Cooper, M., & Cox, C. (2005). Developing a framework for integrating prior problem solving and knowledge sharing histories of a group to predict future group performance. In *Collaborative computing: Networking, applications and worksharing, 2005 international conference on* (pp. 9–pp).
- Strijbos, J. W. (2004). The effect of roles on computer-supported collaborative learning.
- Sung, K., Hillyard, C., Angotti, R. L., Panitz, M. W., Goldstein, D. S., & Nordlinger, J. (2011). Game-themed programming assignment modules: A pathway for gradual integration of gaming context into existing introductory programming courses. *IEEE Transactions on Education*, 54(3), 416–427.
- Thompson, N. A., & Weiss, D. J. (2011). A framework for the development of computerized adaptive tests. *Practical Assessment, Research & Evaluation*, 16.
- Tremblay, E. (2010). Educating the mobile generation—using personal cell phones as audience response systems in post-secondary science teaching. *Journal of Computers in Mathematics and Science Teaching*, 29(2), 217–227.
- Tutorialspoint. (2017). *tutorialspoint.com*. Retrieved 2017-03-27, from <https://www.tutorialspoint.com/>
- Tynjala, P. (1999). Towards expert knowledge? a comparison between a constructivist and a traditional learning environment in the university. *International journal of educational research*, 31(5), 357–442.
- Unity. (2017). Retrieved 2017-04-05, from <https://unity3d.com/>
- Unreal engine 4. (2017). Retrieved 2017-04-05, from <https://www.unrealengine.com/>
- van der Linden, W. J., & Glas, C. A. (2010). *Elements of adaptive testing*. Springer.
- Vorderer, P., Hartmann, T., & Klimmt, C. (2003). Explaining the enjoyment of playing video games: the role of competition. In *Proceedings of the second international conference on entertainment computing* (pp. 1–9).

References

- Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Harvard university press.
- Wainer, H. (2000). Cats: Whither and whence. *ETS Research Report Series*, 2000(2).
- Weiss, D. J., & Kingsbury, G. (1984). Application of computerized adaptive testing to educational problems. *Journal of Educational Measurement*, 21(4), 361–375.
- Whitton, N. J. (2007). *An investigation into the potential of collaborative computer game-based learning in higher education* (Unpublished doctoral dissertation). Edinburgh Napier University.
- Windows phone store. (2017). Retrieved 2017-03-15, from <https://microsoft.com/en-us/store/apps/windows-phone>
- Wolf, M. J. (2001). *The medium of the video game*. University of Texas Press.
- Zooniverse. (2017). zooniverse.org. Retrieved 2017-03-13, from <https://www.zooniverse.org>

Appendix A.

Questionnaire

A.1. Part 1 - Computer Emotion Scale

While using the sCool web application, I felt:

| | None of the time | Some of the time | Most of the time | All of the time |
|--------------|-----------------------|-----------------------|-----------------------|-----------------------|
| satisfied | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| disheartened | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| anxious | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| irritable | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| excited | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| dispirited | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| insecure | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| frustrated | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| curious | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| helpless | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| nervous | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| angry | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Table A.1.: Computer Emotion Scale (Kay & Loverock, 2008)

Appendix A. Questionnaire

A.2. Part 2 - System Usability Scale

| | Strongly disagree | Disagree | Undecided | Agree | Strongly agree |
|--|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| I think that I would like to use this system frequently | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I found the system unnecessarily complex | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I thought the system was easy to use. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I think that I would need the support of a technical person to be able to use this system. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I found the various functions in this system were well integrated. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I thought there was too much inconsistency in this system. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I would imagine that most people would learn to use this system very quickly. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I found the system very cumbersome to use. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I felt very confident using the system. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I needed to learn a lot of things before I could get going with this system. | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Table A.2.: System Usability Scale (Brooke et al., 1996)

A.3. Part 3 - Previous Experience and System Specific Questions

Previous experience and system specific questions:

- How experienced are you with using mobile educational games? (5 very .. 1 not at all)
- Name mobile educational games you know and have used.

Appendix A. Questionnaire

- How experienced are you with web e-learning tools? (5 very .. 1 not at all)
- Name web e-learning tools you know and have used.
- How experienced are you with educational web platforms for mobile game content creation and adaptive learning? (5 very .. 1 not at all)
- Name web platforms of this kind that you know and have used.
- What did you like about the application?
- Was it easy for you to create courses and educational content?
- Were there any problems completing the tasks?
- Do you find the page / the navigation well structured?
- Did the application miss any important features?
- Would you use the this tool for creating mobile courses for students?

