Anna Orlova

EDUCATIONAL WEB GAME FOR LEARNING PROGRAMMING BASICS

Bachelor's thesis
Information Technology

2018

South-Eastern Finland
University of Applied Sciences

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Anna Orlova	Bachelor of Engineering	November 2018
Title		60 pages
Educational web game for learning programming basics		0 pages of appendices
Commissioned by		
Mirum Agency Oy		
Supervisor		
Miia Liukkonen		

Abstract

The basic knowledge of computer science and programming is considered as an essential skill in modern society. The importance and efficiency of modern and unusual ways of studying are acknowledged by educational authorities. The combination of entertaining and challenging experience along with immersion into computer environment can be achieved by playing educational games.

The objective of this thesis work was to create a demo version of an educational web game to be used on a corporate event. The aim of the game was to show and explain the basic concepts of computer programming.

The methods used for designing and implementation of web game included the modern web development technologies like game engines, computer vision and tracking instruments, animation tools, optimization appliances and web application development software. In the process of developing web educational game, the rendering library PixiJS and multiple JavaScript frameworks were used.

The successful outcome of the development was an interactive web browser page displaying a demo version of the educational game for learning programming basic concepts. The web game required players to interact with camera streaming or visual elements on the screen in order to complete exercises. Since the project was focused on the creation of a demo version of a product the potential improvements and additional features were discussed in the thesis.

Keywords

Web development, educational game, programming learning, JavaScript, PixiJS

CONTENTS

1 INTRODUCTION

It is no secret that the best way to teach kids something is simply by playing games related to the educational subject. During the time students are engaged in play-based activities their motivation might be slightly increased, problem-solving skills enhanced and creativity boosted (Bruner 1961, 21-32, 71-72). When it comes to computer science the same playing technique can be implemented with the digital approach introducing the human-computer interaction model to students. However, many schools cannot imagine the curriculum filled with games and hands-on learning activities instead of traditional lectures.

Personal interest for the topic of this thesis was raised after I had come across the inspirational TED talk video by Linda Liukas, a Finnish programmer, author, and illustrator of Hello Ruby, the educational book for children which helps to learn how a computer works and how to write code. Her talk questions the way how the next generation will see the world full of technology. Understanding the responsibility to create a friendly atmosphere in IT education, it is important to consider the code writing process as the way of self-expression which can be as creative as drawing or making music. Learning a programming language can form a vital communicative skill as learning any foreign language, because computers are ‘foreigners’ whose language a person should know in order to build a solid collaborative connection.

The main problem which is analysed and worked with in this thesis project is how to teach children at the age of 7-11 basic programming skills and communication with computer techniques in a web game form. The theory part of this thesis contains the analysis of the educational video game industry and its negative and positive impact on present kids’ learning. The next subjects to focus on are the skills and the way of thinking which should be learned by students in order to simplify their further journey into the computer science world. Finally, technologies used in the design and developing process of the educational game are discussed and explained. The game will be implemented using multiple JavaScript frameworks as the core base of the project and multiple additional features. The practical part of the thesis contains the description of the game and

introduces all the steps of the process of creating the project, shows the result and discusses the problems I faced during the creation process and the ideas for the future development.

The main aim of this thesis work is to make a convenient web game to introduce and teach children programming basics as an out-of-school activity. The theoretical research outcome of this thesis is intended to be useful for anyone interested in implementing game-based educational experiences for students. The practical result can be considered as an example of the educational web game for children out of curriculum hours. Undoubtedly, the process of creating a game will be beneficial for my personal professional growth as a web developer.

2 EDUCATION AND GAMES

This chapter gives an explanation of traditional school and game-based educational approaches defining the benefits and disadvantages of each separately. Similarities and differences between approaches are also discussed. The traditional education system is defined first, followed by the games related approach and the analysis of the two.

2.1 Traditional Education

The educational system, as can be seen today in a form of universal, compulsory and free provision of general various knowledge and vital skills to the people in most countries, was born and gradually developing only from the early 16th to 19th century in Europe. Since the industry become automated in the 19th century the need of human (highly including children) labor declined while the idea of childhood for learning gradually spread leading to the conventional schooling system. (Gray 2008.)

The traditional education method is an old, long-established, socially approved form of interaction between a student and teacher. Subjects, skills and views being valuable and important for specific culture in a certain time period become the focal points of traditional education for the next generations. This kind of

education is usually teacher-centered and aims at forming authority respect and independence (Thompson 2018). Based on multiple traditional education methods, a teacher delivers the subject content to the students in a form of lecture, discussion or demonstration and checks the level of comprehension in a class in a form of discussion, test, homework, laboratory work, etc.

The conventional educational form (another name for “traditional”) is still widely used in most educational institutions worldwide. The best way to understand the image of traditional education is to present it in a form of multiple general concepts. The set of methodical concepts is basically the same in multiple countries and cultures while educational aims and subjects may vary.

The most common characteristics of traditional education include: classroom division, a set of individual subjects, a curriculum, grading system, seatwork, classroom discipline and periodic lessons. Students are divided into groups based on their age and sometimes basic knowledge and the same material presented to the whole group during lessons. A curriculum consists of different subjects from arts and sports to literacy and sciences. In order to calculate the level of education of each student and follow their progress the grading system was invented where each student earns a grade based on the in-class activity, test, homework and practical work results. Many schools have additional sets of rules and restrictions placed on the student’s appearance, behavior during classes or breaks and other aspects of students’ in-school and extracurricular life. Disobeying the rules are usually followed by punishment like isolation from the classroom, reduction of the grade, informing the parents, etc. This kind of methodology is believed to transmit special skills, standards and social or moral models required to be necessary for the next generation’s financial and social stability and success under the consideration of the previous generations. (Erstad 2014.)

As a matter of fact, theory usually diverges from real life, bringing an enormous amount of specific problems and challenges, unpredicted outcomes and statistics. In order to have an effective educational system for a long time, it is

obligatory to analyze the results generation after generation and to make improvements afterward. Undoubtedly, the educational system was slightly changing as the society was evolving in the technological, social, financial and other conditions of life and improving itself gradually from year to year. However, due to the era of technological revolution and further acceleration of technological progress, the way of human thinking has dramatically changed. The international advisor on education Ken Robinson (2010) in his book “The Element: How Finding Your Passion Changes Everything” states:

> *“Public schools were not only created in the interests of industrialism—they were created in the image of industrialism. In many ways, they reflect the factory culture they were designed to support. This is especially true in high schools, where school systems base education on the principles of the assembly line and the efficient division of labor. Schools divide the curriculum into specialist segments: some teachers install math in the students, and others install history. They arrange the day into standard units of time, marked out by the ringing of bells, much like a factory announcing the beginning of the workday and the end of breaks. Students are educated in batches, according to age, as if the most important thing they have in common is their date of manufacture. They are given standardized tests at set points and compared with each other before being sent out onto the market. I realize this isn’t an exact analogy and that it ignores many of the subtleties of the system, but it is close enough.”*

The way of searching, collecting and converting the information into the skills, standards and knowledge is no longer the same as it was before the first computer was invented. Instead of learning how to store huge sets of information inside a human brain the raised importance of the ability to reach qualified and useful information very fast from multiple sources has become the sign of high efficiency. The human brain is no longer needed for storing an enormous amount of information, but the ability to access and apply the information needed in any kind of problem-solving is the guarantee of human success. While digital

methods of receiving, processing and storing the information replace the traditional bit by bit, it is wise to think about creating a comfortable and beneficial digital educational concept. (Panaworld Education 2017.) This tectonic transformation led to the idea of reforming the traditional educational system from its roots, which means revolutionizing the whole system creating a new complex approach called progressive education.

2.2 Progressive Education

By Robert Kennedy (2017) the definition of progressive education is the following:

> "*a reaction against the traditional style of teaching. It's a pedagogical movement which values experience over learning facts at the expense of understanding what is being taught."*

Educational reform and switching to the progressive way of education means that learning is seen from a completely different perspective. It is a pedagogical movement which values the experience and practices over the theoretical memorizing facts and numbers as the understanding of what is being taught lays on the main focus of this methodology.

There are plenty of controversial progressive education programs invented and used nowadays but all of them have a number of values in common (Kohn, 2008). Most of those qualities are listed in a form of statements as follows:

- **Attending to the whole child**
  Making a good learner out of the student is not the only aim of a progressive teacher**.** Teaching children a variety of necessary good person qualities becomes as important as intellectual growth.

- **Community**
  Modern students are taught how communication and collaboration with others can bring better results and ideas rather than competing individually. This helps them to understand how to create a caring and efficient community.

- **Collaboration**

  Group work and participation in collaborative and cooperative learning projects help kids to develop social skills which are a vital aspect of human social life.

- **Social justice**

  Growing in a constant communication within a community help children to raise a sense of community and responsibility for the ones around despite countless diversities and individualities of a human being.

- **Deep understanding**

  Instead of blindly memorizing independent facts and skills children are taught to see the context around them, their purpose and necessity by asking questions and completing the practical projects in a complex of multiple disciplines.

- **Active learning**

  The emphasis is switched onto the completion of practical projects and problems, making decisions, discussing the questions with a teacher instead of passive perceiving theoretical information and blindly following the directions. This approach helps to educate the problem-solving skills and critical thinking among students.

- **Taking kids seriously**

  Learning based on an individual’s goals rather than teaching the same scope of subjects to everyone. The selection of subject content is made by defining the interests and by choosing what kind of skills are needed for the future.

- **Lifelong learning**
  School is no more considered as a preparation stage for future adult life but as a part of an endless learning process happening throughout a person's life (Kennedy 2017).

- **No standard evaluation**
  Not all the kinds of learning and subjects like artistic or physical ones are very easy to measure in a gradual way, but also the whole system of subject knowledge measurement switches student emphasis from learning and understanding to competition with others and getting the best grade. (Meador 2018).

Fundamentally, progressive education is about teaching students how to think instead of what to think. Experimental as a part of progressive learning aims to develop creativity, critical thinking, independent working and collaborative skills which are crucial for the current workplace environment for all professions. (Kennedy 2017.)

2.3 Teaching Methods

The description in this chapter about different teaching methods bases on the section from the educational web source for teachers in the USA Teach.com. with the same name. Teaching method can be described as the summary of "*general principles, pedagogy and management strategies*" to be used by teacher in the classroom.

The searching methods can be divided in groups by four classifications based on two parameters: approach (student-centered or teacher-centered) and studying material (high-tech or low-tech). The teacher-centered approach is the model where students passively receive knowledge from teachers and this learning is objectively measured by their test and assessment results. The student-centered approach based on the equality of student and teacher roles in the learning process where the teacher is a guide and the measurement objectives become

both formal and informal (group project, participation in the process, etc.) assessments. The high-tech learning approach utilizes modern technologies in the education process like different hardware (tablets, computers), basic software (email, calendar, drive, etc.), gamification educational software, school social-media platforms, accessibility driven technology for students with disabilities. The low-tech is a traditional approach to learning which is useful in some cases where technological presence can slow down and complicate the education process. (Figure 1.)

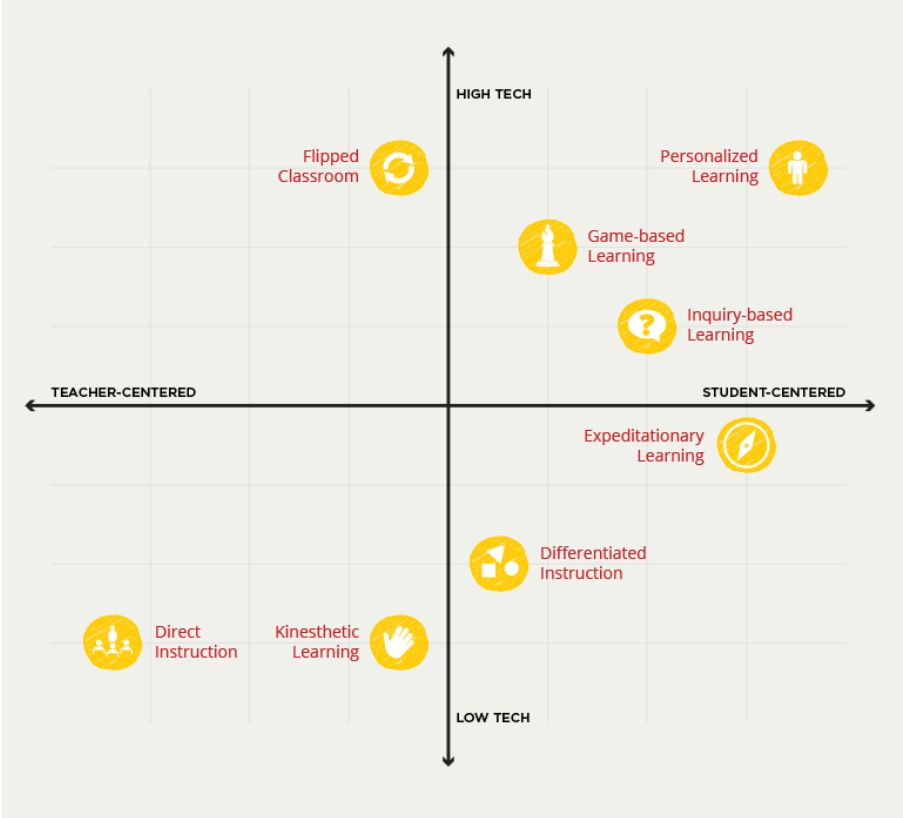

Figure 1. Teaching methods (Teach.com 2018)

Within every category of teaching, there are several suitable teaching methods. There is the direct instruction (low tech), flipped classrooms (high tech) and kinesthetic learning (low tech) in the teacher-centered learning approach group. The direct institution method represents the traditional teaching strategy with passive learning through lectures and presentations of a subject by a teacher. The flipped classrooms approach relies on the opposite idea of listening pre-recorded lectures at home and completing the practical part as an in-class activity. The learning approach where students perform physical activities like

hands-on exercises, dancing, drawing, building, doing sports and others is called kinesthetic. The student-centered group of methods includes differentiated instruction (low tech), inquiry-based learning (high tech), expeditionary learning (high tech), personalized learning (high tech) and game-based learning (high-tech). The aim of differentiated instruction method is to provide the education in the same style but to each student individually based on their personal background level. Students play the active role in the inquiry-based educational method while a teacher takes the supportive role and provide guidance through a student's investigation project. The expeditionary learning method is a project-based education where students go on 'expeditions' to face the environment they have to study and make some practical tasks or research. The curriculum is created individually for each student based on their interests and skills in case personalized learning method applied. The game-based learning method creates an environment where students have to work on quests, chose the actions and earn points and badges in order to achieve a particular goal.

2.4 Learning by Playing Games

Childhood is usually characterized by playing games. While playing a game a child can develop physical, emotional, social, cognitive and other useful skills. Games create a basic educational environment for a child to learn about the world they live in, practice new skills, develop problem-solving and abstract thinking. One of the most exciting ways to learn something experimentally for young students is by playing games. Games are the most interactive, easy and personalized way to learn and solve the most challenging studies or problems. Game-based learning approach is a wide and diverse field of educational practice. (Mackay 2013.)

Games can also be considered as an educational tool. According to the Horizon report (2010), games give students a chance to experience the challenges and success through collaborative work and communication in order to solve a complex problem set within an engaging storyline. There are 3 types of relevant to education games: not digital, digital (not collaborative) and collaborative digital games. According to the subject of this thesis work, the second and third types

are the most applicable to the description, research and comparison. Digital games, including simulations or virtual reality, are capable of helping to teach different subjects, because they are interactive, engaging and immersing activities.

Playing a digital game may look like a dump repetitive activity in isolation, but in reality, it is far more collaborative and challenging than it's thought. Most successful multiplayer games attract millions of individuals to interact, collaborate with groups and compete against each other. Moreover, games allow participants to be active in discovering new ideas or solutions while learning about the environment and engaging in the commitment of succeeding by determining the consequences of each their step. Educational games can teach students to focus while being patient in order to reach the next level; improve self-esteem, while seeing the way they accomplished a particular thing in the game, and memory while collecting milestones needed for further levels. Finally, a game can be helpful in developing a set of multiple skills and abilities at once instead of dividing them by each subject. (Maton 2011.)

One of the greatest examples of how game-based educational methodology works in real life is the public primary and secondary school in Manhattan, New York City, for the 6-12 grades called Quest to Learn. What makes this school so interesting and amazing is the fact that students there learn everything only by playing games. Subjects in this school are usual: calculus, history, geography, but the teaching curriculum here is made by game design principles. The creation of this school concept was a responsibility of a group of teachers in collaboration with game designers. Instead of using the grading system of A, B or C students here get levels like Novice, Apprentice, Senior, and Master. This kind of evaluation system makes more sense for the child and brings more motivation than just three letters of the alphabet. The student can always try new additional 'missions' (tasks in games) to get more points for the level or retake the 'mission' again in case of the previous failure, minimizing the level of stress because of a low score. There are also 'boss missions' which usually take place at the end of semester and are presented as two weeks sessions for the whole group where

each student is responsible for one role in order to solve one complex problem. Quest to Learn also has the social network for students to search for companions for specific missions just like in an ordinary video game. Instead of homework, each student has to teach their virtual companion the subject they just learned. This approach completely changes the way education is seen by students and seems to be more efficient in modern life. (Burak & Parker 2017.)

3 VIDEO GAMES

According to Grant Tavinor (2008), senior lecturer and writer:

> *"X is a videogame if it is an artefact in a digital visual medium, is intended primarily as an object of entertainment, and is intended to provide such entertainment through the employment of one or both of the following modes of engagement: rule-bound gameplay or interactive fiction."*

This definition of video games is fulfilling and determinative, except the part, where entertainment is seen as a main purpose of video games. This is more controversial and questionable statement for gaming industry in 2018 than in 2008, when this work was written. There are currently multiple aims for creating a video game going far beyond simple entertainment.

3.1 Video Game History

Despite the fact that video games reached mainstream popularity only after the 1970s and 1980s, the development process of creating first electronic games on electronic display began in 1947. The invention of cathode-ray tube amusement device happened this year leading to the video games popularity in the early 1950s. A series of games simulating real-world board games were created in the 1950s in order to explore and study programming, computer algorithms and human-computer interaction.

The first video game created for the entire purpose of entertainment was created in 1958 by american physicist William Higinbotham for display at the Brookhaven National Laboratory. It was a sports video game presenting tennis simulation and

called “Tennis for Two”. The game was deployed on an analog computer with an oscilloscope as a device to display graphics. (Ramlagan & Tretkoff 2008.)

Video games played on consoles with graphics on a computer or TV screen and with different controllers like buttons or joysticks, similar to those played today, became widely used by public in terms of entertainment only by the 1970s. The first game which spread outside the research institute and was available for play on different computer installations was ‘Spacewar!’ The game was created by a group of computer scientists at Massachusetts Institute of Technology and was expanded by employees and students in the other universities. There are two spaceships in the game: “the needle” and “the wedge”. Both ships are controlled by human players, have limited amount of fuel for movement and torpedoes for shooting the “enemy” (another player). The game play takes action in the gravity well of a star and the ships follow Newtonian physics constantly remaining in motion. The player loses when the ship is destroyed in case of a torpedo hit or collision with a star. The player also has an option of using hyperspace feature to move the ship to a random location on the screen, which has an increasing chance of destroying the ship. (Graetz 1981.)

According to the article by History.com editors (2017), the prototype multiplayer video game system that could be played on television was invented in 1967 by Sanders Associates developers and was later licensed to Magnavox. That is how the first home video game console ‘Odyssey’ with multiple games and additional devices was globally sold. One of the ‘Odyssey’ games inspired new home computer company Atari to make the first arcade game called ‘Pong’ which led to the successful foundation of arcade video game industry. In 1977 Atari released a home console featuring joysticks and game cartridges with colorful games which initiated the second generation of game consoles. The video game industry started rapidly developing since that time releasing ‘Space Invaders’ arcade game (1978), ‘Pac-Man’ (1980), Nintendo’s portable handheld electronic game toy Game & Watch (1980), launching first only software developing company Activision. (History.com editors 2017.)

At the end of 1983 North American video game industry experienced severe downtown due to multiple factors like the rising popularity of home computers as gaming platforms, the excess of low-quality games on the market and the commercial failure of several Atari games. The crash led to the bankruptcy of several companies and the end of second generation game consoles. The home video game industry became to emerge in Japan with Sega and Nintendo which Entertainment System (NES) led third-generation of game consoles and third-party game publishing. Nintendo released video game franchises which are still popular today like ‘Super Mario Bros.’ (1985) and ‘The Legend of Zelda’ (1986). In 1989 Nintendo released the 8-bit Game Boy video game device and ‘Tetris’ game which led to the popularity of handheld game consoles.

Sega made the 16-bit Genesis console during the same 1989 which was technologically superior to the NES and released ‘Sonic the Hedgehog’ (1991) game. In 1991 Nintendo released 16-bit Super fourth-generation NES console which led to the ‘console war’ in the early 1990s giving birth to violent video games like ‘Street Fighter II’ (1991) or ‘Mortal Kombat’ (1992). Genesis won the battle in North America because of a large library of games, lower price and clever marketing but Nintendo was still more successful in Japan.

As computer technology was constantly developing the fifth generation of video games ushered in the 3D era of gaming. The first 32-bit Sega Saturn console system which played games on CDs was released in 1995 before Sony’s first PlayStation and Nintendo’s cartridge-based 64-bit system. Sega and Nintendo couldn’t compete with Sony strong third-party 3D game support which led to the Sony’s domination in the next generation. In 2000 PlayStation 2 which first used DVDs released and became best-selling console of all time.

Released at 2005-2006 Microsoft’s Xbox 360, Sony’s PlayStation 3 and Nintendo’s Wii which features include online gaming, motion capture system, motion-sensitive remotes became the symbols of modern age high-definition gaming. Closer to the end of decade video games spread to the social networks and smartphones creating new fields in the industry. In 2012 Nintendo’s Wii U

was released, followed by Xbox One and PlayStation One in 2013. In 2016 Sony released PlayStation 4 Pro which can support 4K video output followed by Xbox One X in late 2017 with the same feature. According to the rising popularity of virtual reality both Microsoft and Sony preparing to compete with each other in the new field and experience of gaming.

3.2 Video Game Platforms

This chapter generally bases on the Georgia Williams (2015) blog about computer game platforms and technologies. The gaming platform is a particular combination of electronic components in computer hardware which creates a device where gaming software can be played in the form of video game. There are multiple platforms currently available specifically for gaming purposes and many other kinds of devices where games can be played as secondary purpose. According to Williams (2015) there are several categories of video game platforms which presented in the list below:

- **Personal computers**
  Personal computers can serve as a good and most common gaming platform. Both desktop PCs and laptops with a specific motherboard, multiple core processor, powerful video/graphics card and memory can be used for the gaming purpose. Personal computers with additional gaming hardware like display or joystick and software can form a useful gaming platform. (Williams 2015.)

- **Video game consoles**
  A console video game is played on a device manufactured especially for gaming purposes by a specific company, video game console. Consoles normally come with an input playing device like a joystick and the main unit box where all the processing work happens and which usually connects to the TV screen to see visual output. Unlike PCs with multiple possibilities of installing gaming software consoles only support their

manufacturer software. The main console platforms are Xbox, PlayStation and Nintendo. (Williams 2015.)

- **Handheld device**

  Handheld console platforms are small, lightweight and feature their own small display, speaker and input game controller like buttons which make them mobile and portable. Handheld consoles share almost the same characteristics with consoles, but they are less powerful. (Williams 2015.)

- **Arcade**

  The arcade game is a big in size device designed and manufactured to be used to play specifically one game. It is usually packed in a cabinet-size coin-operated box with input game controller, speakers and screen. Arcade game machines are usually located in public places like malls or amusement parks. The golden age of arcade games was during the 1980s, but they are quite rare and considered as old-fashioned device now. (Rubin 2014.)

- **Web browser**

  A web browser can also be used as a gaming platform bringing cross-platform environment for video games to be played on different hardware devices. Hundreds of browser games in multiple styles and classes are currently available on the Internet. (Williams 2015.)

- **Smartphone or tablet**

  Mobile games can be played on cell phone devices or tablets. Mobile games are usually designed for a specific operating system the device has, like iOS for iPhone/iPad, Android or Windows Mobile. Games on mobile devices are getting more popular these years. (Williams 2015.)

- **Virtual reality**
  Virtual reality games come with a special head-mounted display device which provides stereoscopic scenes with special optics and uses motion tracking system responsive to the head and body (with hand control units) movements in order to put a player to the virtual environment of a game. VR system requires a separate console or computer to do the processing. The VR game industry is currently a rapidly developing and promising field of development. (O'Boyle & Willings 2018.)

All the different platform types for video game development create a wide miscellaneous environment of gaming where people of any age, interest, financial status and purpose can find a suitable device or platform.

3.3 Video Game Classification

There is a countless number of video games created and in the process of constant development and improvement today. There is also a wide range of features which is used to categorize games in different groups. Video games can be classified by the main purpose of their usage by two wide groups: casual and serious.

Casual games are very easy to access or install, simple to understand the gameplay and rule sets, not difficult to pause or stop and restart from the same place. Casual games usually cover one main purpose – entertainment of a player. Casual games are available to be set on most platforms, can be downloaded from any app store or played online on the web browser on a personal computer.

Serious games are designed for a primary purpose of education rather than just entertainment as in casual games. Serious games can be used as an educational source in scientific, engineering, healthcare, aviation, cosmonautics, building and more other industries. One example of a serious game is a simulator which can

be used as a training practice skills source for juniors or students. (Gamelearn 2017.)

Games can be classified into many different genres according to the gameplay interaction techniques and challenges. Unlike movies or books, video game genres are not defined by their content setting. According to iD Tech blog (2018), there are around 50 genres and subgenres together so they can be linked in groups.

Action games are the most popular type of games where the player is in control and in a center of a constant action. Action games can be a platformer, shooter, fighting, survival and others. Action-adventure games focus on exploration, solving quests, discovering items and using tools. Action adventure games include survival horror genre and Metroidavania genre which name based on two game names combined: ‘Metroid’ and ‘Castlevania’. The main difference of Metroidvania games is that they are not linear and have upgrading character feature. Adventure games are storytelling type where player has to solve quests and interact with environment in order to move forward in a story. There are text or graphic adventures, visual novels, interactive movies or real-time 3D game genres included in the adventure type.

The second popular game genre type after action games is a role-playing game (RPG). There are different types of RPG games like action RPG which include features of action and action-adventure games, massive multiplayer online RPG (MMORPG) which involves hundreds of players interacting with each other, roguelikes genre which name based on the game ‘Rogue’, tactical RPG which is based on tactical role-playing board games, sandbox RPG where player has the ability to roam and change virtual world or select tasks, first-person party-based RPG also known as ‘blobber’ type where player has a first-person perspective.

Simulation type of games is produced to emulate real or fictional reality. Construction and management game simulates construction of buildings and management of the city. Life simulation genre allows a player to manipulate

aspects of character artificial life like appearance, actions and reactions like in a famous game ‘The Sims’. Vehicle simulation games include simple racing or flight simulations.

Strategy games are based on the need for players to develop strategy and tactics to overcome challenges. The 4X strategy game which has four main features: explore, expand, exploit and exterminate. Players have to collect and maintain resources while developing skills in the process of real-time strategy (RTS) game. The multiplayer online battle arena (MOBA) combines the action, RPG and RTS features. In tower defense games player have to protect their base from the computer enemies. Turn-base strategy (TBS) games are mostly similar to the real-time strategy ones but give the player a limited amount of time to take an action. Turn-based tactics (TBT) games provide the most realistic military tactics to the player. Map-based tactical and strategic games are called wargames.

Sport simulating games are categorized to the group sports games. Racing games create competitive vehicle simulation environment between player and computer or other players. Team sports games simulate playing a sport like a football or a hockey in a team with other gamers or computer. Competitive sports games are based on a fictional sport. Games with realistic fighting usually fall into the category of sports-based fighting games.

Puzzle games require a player to solve a logical task to improve the features. Logic games include puzzles and challenges which need to be solved in order to proceed forward. Video trivia games requires player to answer a question in a limited amount of time.

There are many video game genres which are not organized in groups. Idle games are simplified and have a minimalistic design, party programming games have code and programming tricks challenges, board or card games present the simulation of a real board game, advergames are marketing features to promote a brand or sell a product, educational games are used as a teaching tool.

3.4 Game Elements

Every game has multiple features or elements that immerse the player in the process and keep them engaged. There are numerous elements that video game can include based on its type, purpose or platform. Sharon Boller (2013) selects and describes twelve common ones in her blog: conflict, collaboration, competition, strategy, chance, aesthetics, theme, story, resources, time, rewards/scoring, levels. Since the main objective of this thesis is to describe the educational game design, this chapter focuses on the description of five game elements useful in creating effective learning games.

The conflict element makes the game interesting. In order to move forward in the game process, the player has to solve challenges like physical obstacles, fight with another player or puzzle. There are various ways to include conflict in game-based learning. Conflict can be started between players on a competition base or players can collaborate to overcome challenges together.

Strategy games make the player take control over the gameplay by the choice of decisions they can make and the number of consequences they can face moving forward. Otherwise, game event based on a chance stimulate player reactivity and provides unpredicted outcome despite player activity. A game can focus on chance or strategy element alone or combine both simultaneously. In order to create a good educational game, it is a must to exclude a probability of winning by a lucky chance of a specific sequence of events and actions.

In video game industry aesthetics plays a major role in the gaming experience. Visuals are powerful tools of engaging players into the gaming process. The emphasis of aesthetics in educational games is much lower than in entertaining ones but not the importance. It is necessary to create attractive visuals for the positive educational game experience.

The engagement within a learning game can be created with a help of theme but only a complete storyline can make the game interesting because a player can easily remember rules and facts if they are connected to one narrative. There are

four elements which can help to create a strong story: character(s), plot, tension (presence of conflict) and resolution.

Rewards in the game can boost player motivation to move forward and achieve more. Rewards can be given not only for completing the task but also for the performance during it. The score can be a powerful feedback tool which motivates players to act and learn better. (Majumdar 2016.)

3.5 Educational Games

The main purpose of the educational video game is to train and educate the player. Gamification of education is a significant change in the learning process. Learners can develop technological skills useful for the chosen academic subjects with the help of games. (Mackay 2013.)

It is important to know how to develop an effective educational video game. The main challenge for the game creator is to find the harmony between entertainment and learning in order to provide useful and educational experience. There are seven tips to create an effective experience through playing video game according to the Christopher Pappas writing in his blog (2015).

First of all, it is crucial to design a game layout before the development process starts. Useful layout usually includes the structure of the game, the visual output, interactive components and fields of most attention to focus. Secondly, in order to make the game visually impressive and prevent the overdose of graphics, it is important to stick with one specific theme which includes a set of fitting graphic design elements. The deep research of the target audience and subject of the game can help with the wise choice of game visual theme. Thirdly, all the elements used in the game should serve one main objective and goal to educate. If learner loses interest while playing the game it simply means that the main purpose of the game is not accomplished and there is zero material learned. Moreover, it is wise to provide the player with the feeling of control over the game and a range of choices of different directions they can personally make. Furthermore, players should know what kind of consequences their choices may

have. It is important to show the outcome of the whole game process as well. Finally, educational games should involve both risk and reward elements in order to make them exciting, motivational and informative. Educational games can be fun and training at the same time if those simple tips are carefully followed on each step of game development.

4 PROGRAMMING A LEARNING VIDEO GAME

The times when children and adults had to read a lot of educational literature and online tutorials in order to learn how to code are a long time over. Online education currently becomes more and more popular providing students new ways of getting the knowledge. There are multiple online tutorials, lectures and tests, interactive courses and different educational video games. Most of the educational video games cannot replace the whole course of learning programming, but they can motivate a student to learn more, provide an opportunity to practice new skills, raise curiosity about the subject and give a lot of fun. In order to know how to develop not only entertaining but also an effective code learning game for beginners, it is critical to know what kind of programming basics children have to learn. Moreover, it is a big plus to get acquainted with the most popular and successful programming learning game examples.

4.1 Programming Basics

According to software developer David Bolton (2018) "*programming is a creative process that instructs a computer on how to do a task*". In other words, the programming process includes creating a set of rules and instructions based on logic, transforming this set to the special language computer can understand and implementing it in order to make the computer perform the invented function. Computers do what they are told by humans in a form of computer programs. There is no black magic or rocket science in the programming process, but plenty of nuances and surprises which make coding mysterious. It is no surprise that every modern student with education in a technical field should know at least the very basics of programming.

4.1.1 Programming Basic Elements

Most programming languages present a set of instructions needed for the program to work and have some basic elements in common. There are several common elements of instructions that are going to be listed below as follows:

- **Variable**
  Variables are used in programming languages as special labeled containers holding information which should be stored. The data to be used in the program can be stored in a variable and easily accessed because of a variable label (name) in the memory. There are plenty of processes coming through one program and variables become very helpful in means of holding the outcomes to work with.

- **Loop**
  In programming, a loop is a sequence of instructions which is repeated until a certain condition is achieved. The main purpose of a loop is to maintain an execution of code commands a certain number of times. The infinite loop does not have a functional exit in a number of times to execute a function and it is repeated indefinitely. Loop is a fundamental idea in programming which is commonly used almost in every program. (Study.com 2018.)

- **Conditional**
  Conditional statements are features of any programming language that help a program to decide what function to perform based on the programmer-specified Boolean condition (true or false). There can be multiple conditions in the single one creating a complex choice making system. (Study.com 2018.)

- **Input/Output**
  The input defines a process of getting data and commands into the computer and output defines a process of getting the results out of it. Input

comes from a human by the means of interacting with a keyboard, touchscreen, text file, another program, etc. The output is the result of a program gives to the user based on the input it had in many forms: on screen, print, as sound, etc. (Study.com 2018.)

- **Subroutines and functions**
  The subroutine is a unit in a computer program which performs an exact task. The unit consists of a sequence of program instructions. This unit can be used in different part of programs which means one task can be done at a different time or for different purposes. The function also seems like a unit of code which is usually defined by its purpose in the program. The difference is that a function always returns some value while subroutine does not.

- **Algorithm**
  The sequence of different simple tasks performed in the certain order are called an algorithm. Computer uses different algorithms in order to perform some task or do some calculations. Programmers have to use algorithms so computer can understand the rules and the logic of the task to be performed. (Brogan 2016.)

These elements of programming help to understand how a program works, what it uses as a working material and how a programmer can control it.

4.1.2 Programming Basic Principles

According to the Developer blog (2011), there are several fundamentals which every programmer beginner should know, and they are not directly linked to the coding rules and tutorials, but to the way how a programmer should think and observe things. To begin with, one of the biggest misconceptions about programming is that a good programmer should be good at math which we used to deal with at school. There is plenty of math in programming but not the same as in school. Instead, the skills of thinking ahead, understanding the order in

which things should happen and how to control this are more viable in programming than the knowledge of trigonometry. A person which good logic skills is more likely to be successful in programming.

Secondly, the programing language consists of multiple building blocks or ‘types’. There are ‘string’ type (sequence of characters), number type, array type (set of things in order), ‘hash’ or ‘object’ type (set of keys with their own values). The type concept is constantly used in programming by the means of a data storing structure.

Furthermore, there is one important concept in programming which resembles a ‘russian doll’ toy. Objects are located inside the bigger objects creating a nested structure and a hierarchy within themselves. The situation with commands and functions is the same: there are a lot of small commands and functions used inside the bigger ones.

Moreover, one of the most useful things to know for the beginner programmer is the fact that there always would be situations where the cause of different processes and actions in a program could not be easily defined. The skill of isolating secondary factors of changes in program performance from the significant ones should be developed during the education time.

Finally, the trickiest programming principle is connected to the ‘abstraction’ process. The process of removing some specific characteristics from the set of basic and essential ones, which are often used to define the object, is one of the fundamental ones in computer science. The process of abstraction can be used by a programmer in order to reduce the complexity and increase the efficiency of a program.

4.2 Video Games for Learning Programming

Focusing on all of the elements and rules in creating an educational video game is not the only way to make it efficient and valuable. There is a wide range of educational video games for learning programming at this moment which can be

useful for consideration while creating the game of one's own. Playing and testing educational games made by different companies can help to better understand the risks of creating one, bad and good sides, successful methods and the concepts. For this chapter, I chose famous and successful programming learning web game examples which caught my attention for further research and analysis.

4.2.1 Scratch

Scratch is a programming learning service developed by Lifelong Kindergarden Group at MIT Media Lab. It allows children to create animations, personal games, stories, music using the eponymous object-oriented programming language. Players do not have to write code but to assemble visual component blocks as units of code in order to make a program. The component blocks are made similar to the puzzle pieces, which makes the game visually friendly. Scratch players have countless possibilities to create a combination of blocks to make one's own project or to use someone else's as a base. Scratch became very popular and widely used software development kit as its philosophy is simple for understanding, educating and entertaining. (Rouse 2017.)

Since Scratch is designed for introducing the software development to children it can be used the same way by adults who want to learn programming from the roots. Scratch is also an online community where students can share their own projects, exchange ideas and get feedback and support.

4.2.2 Minecraft

The famous sandbox video game where player can create things out of virtual boxes is called Minecraft. The game was originally created by Markus Persson but afterwards maintained by Mojang AB and Microsoft Studios. Minecraft can be considered not as a game only, but as an engineering tool which can teach logic, geometry and problem-solving. In order to create a custom environment children have to learn how to make changes by using code. Minecraft focuses on learning the concepts of programming rather than syntax and rules. According to the

player skill level more difficult and more related to the actual writing code activities appears on the constantly upgrading path. (Dodge 2016.)

4.2.3 Code Combat

Code Combat is a free browser-based role-playing game. This game allows players to battle with opponents with a help of writing code which instructs their characters where and how to walk and fight. Code Combat helps to learn JavaScript, Python and basic programming concepts. (Thomas 2014.)

5 WEB GAME TECHNOLOGIES' THEORETICAL BACKGROUND

There are multiple technologies used in the process of this game development. In order to understand why and with what purpose they were used it is important to know the basic theoretical background. This chapter describes various game engines and the logic beneath the choice of the suitable one for this specific project. The core of this web game lays on the JavaScript programming language and different JavaScript libraries.

5.1 Web Development Programming Languages

The process of creating a website, in general, is called web development. There are various fields of activities involved in the web development process like web design, programming, network security, server-side development, content creation, etc. In order to perform web development tasks, it is important to understand what are the tools, how to choose and use them correctly. The best tools and platforms for the web are the programming languages like Java, JavaScript, Python, Ruby, C++, etc (Cleverism 2015). There is no best or worst programming language because each of them has a unique set of features which allows them to serve special purposes. The JavaScript language was chosen to be the core of the educational game because it is one of the most popular client-side web development languages today. It also has numerous libraries which make the programming process faster and easier. They also provide a helpful set of tools for creating games, animations, video tracking and optimization for the web.

5.1.1 Web Page Triad

There are three components which can help to build a web page. The HTML (Hyper Text Markup Language) is used to describe the structure of the web page by using the content building blocks called tags. The CSS (Cascading Style Sheets) is style sheet language which is used for creating a layout and style, colors, fonts, animations of a web page. The third component is JavaScript is responsible for functional and dynamic part of a web page. (Sridhar 2017.)

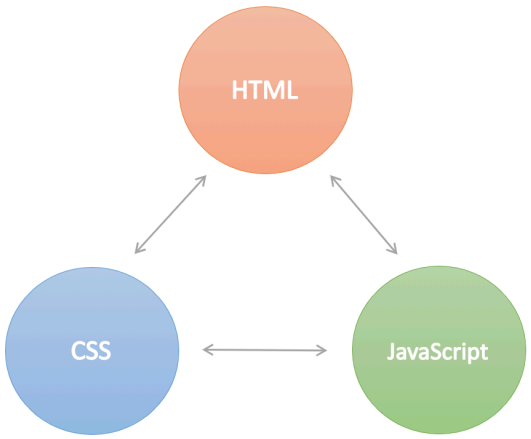

Figure 2. The web page triad

Web page based on three main components (HTML, CSS, JavaScript) as presented in Figure 2 is a very basic concept. However, when it comes to web game development, this kind of structure may affect the performance and optimization in a bad way. Games usually contain a lot of graphical elements and animations which cannot be properly processed using a web page triad method. This is where HTML5 (latest HTML standard) **<canvas>** element may come in use. The canvas element is used as a container in order to draw graphics on a web page. It has several methods when it comes to drawing paths, boxes, circles, text or adding images. There are numerous HTML5 canvas JavaScript libraries which make web games efficient, interactive and lightweight. (Shah 2018.)

5.1.2 JavaScript

JavaScript is a dynamic programming language created in order to make the elements on the web page interactive and provide a user-friendly experience. Most of the interactive web page elements like menus, search bars, dialogue boxes, video players, 2D and 3D graphics, animations, video and audio players, interactive maps are made with the help of JavaScript. This technology is supported by most modern web and mobile browsers. (Sridhar 2017.)

The JavaScript language was created in 1995 by Netscape Communications employee Brendan Eich in order to make the Netscape Navigator web browser more interactive. Because of the competition environment, the new language should have been created very quickly. Some features were borrowed from other programming languages in the process of JavaScript development: syntax from Java, a prototypal inheritance from Self, strings, arrays and regular expressions from Python and Pearl. In the process of development after the first release the language had different names: Mocha, LiveScript and finally JavaScript. The language was submitted to the ECMA International specification in 1996. (Rauschmayer 2014.)

JavaScript is known as a client-side script which allows to make the interaction between user and device (computer, smartphone, etc.) happen without the connection to the server. One of the most powerful features of JavaScript language is the ability to communicate with the remote server not causing the interruptions to the user interaction which is called asynchronous interaction. When a web browser loads the web page, HTML parser creates a Document Object Model (DOM) from the page content. JavaScript can make updates to the DOM in order to add different interactive features. After the page structure and styles are loaded, JavaScript is executed and the page is constantly rendered and updated to perform interactions. (Sridhar 2017.)

Since JavaScript is constantly developing language there are plenty of versions available today like ES6 (ES2015), ES7 (ES2016), ES8 (ES2018). There are multiple differences coming with each release starting from the syntax and ending

additional features and changes in the functional. It is important for a successful developer to follow the changes and improve the understanding of the constantly changing technology.

5.2 Game Engines

A game engine is software which provides useful features in order to create a video game. Game engines are basically responsible for rendering graphics, memory management, artificial intelligence features, collision detection, animation and many other technologies suitable for game development (Baker 2016). Basically, the game engine is the same as a framework or a bunch of multiple libraries together used in order to create a game. Since the chosen video game type is web, it is still more efficient to use a game engine instead of web development libraries and frameworks together, because a game engine provides the scene graph, level editor and other essential for games features.

There are several supporting technologies which have to be explained before the game engine discussion starts. First of all, API (application user interface) is a technology which allows accessing the data or features from different sources like operation systems or other in order to be used in the following application. Secondly, WebGL (Web Graphics Library) is a JavaScript API which renders interactive 2D/3D graphics to be used in a compatible web browser.

There are numerous game engines in the modern game development world, and it is sometimes not easy to find a suitable one without getting lost. There are several valuable, useful and famous game engines described in the following list. (Ourcodeworld.com 2016.)

- **PixiJS**
  It is an easy-to-use super-fast 2D renderer which can create both HTML5 games but also various lightweight interactive digital content. PixiJS supports both WebGL and HTML5 Canvas and allows developers without prior WebGL knowledge easily use the tool.

- **Phaser**
  This is a JavaScript engine which allows creating 2D HTML5 web games for both desktop and mobile. Phaser is very popular among the programming beginners, because it has a wide open-source database and a huge community of developers along with easiness of use and small codebase in order to start using it. Phaser supports WebGL and Canvas rendering along with PixiJS. The last version of Phaser does not support video stream.

- **Babylon**
  It is a powerful JavaScript library supporting web 3D game creation based on WebGL, HTML5 and Web Audio. It does not require any installation on the computer to use because it works within a computer browser.

- **Crafty**
  The Crafty library works on pure JavaScript. There are no DOM manipulation or custom drawing routines required. It also has a wide community with various tutorials and fast support system.

5.3 JavaScript Libraries

JavaScript is a very powerful programing language in the means of the constantly evolving environment of libraries, frameworks, tools and package managers. It has become more and more difficult for web developers to follow its constant fast-pace growth and make predictions about the future relevance of currently popular technology. There are numerous JavaScript tools and libraries available for the use today and it is important to be able to find the useful and suitable for one exact project ones. In this chapter, a number of libraries described below along with their purposes and roles in the educational web game development.

According to Linux.com statistics, **npm** package manager for JavaScript runtime environment **Node.js** becomes the largest software registry in the world with over

than 350 000 packages (Brown 2017). It has the command line client (**npm**) and online database of packages (**npm** registry) which is accessible through **npm** website. This package manager helps to easily install or update multiple tools, run packages without installing, share code with other **npm** users, manage multiple versions of code or code dependencies and many other things. The package manager is a great time-saving solution for managing the project. (npmjs.com 2018.)

Another useful tool for web development is the one called **Webpack**. **Webpack** is a module bundler which purpose is to optimize web performance for users to load and interact with the website more quickly. The main concept of this tool is to make the dependency graph of each module the project uses and to generate bundle or multiple chunks. They can be asynchronously loaded in a single runtime reducing the initial loading time of a page. (webpack.js.org 2018.)

There are plenty of JavaScript versions available today each of them having the differences in the syntax and functional components. The newest version allows a developer to include more features to the project avoiding a lot of code writing. The main problem of new JavaScript versions is that they are no longer supported by old browser environments. This is where Babel JavaScript library becomes very useful while converting the ECMAScript 2015+ code backward to previous versions. This tool allows a web page to be compatible with most web browsers and their versions. (babeljs.io 2018.)

There is a very useful tool for macOS (operation system for Apple devices) users which is called Homebrew (or brew). It is free and open-source package management tool which helps to install missing software with a help of a command line. (brew.sh 2018.)

Tracking.js is a very useful library for web development based on real-time video tracking features in a browser environment like face detection, color recognition, and others. This library can be very effective in web game environment in order to make it more interactive and entertaining for the player. **Tracking.js** uses

different computer vision algorithms and techniques. This powerful and lightweight open source JavaScript library uses modern HTML5 specifications. (trackingjs.com 2018.)

In order to make the gaming environment entertaining and bring interactive elements to the gaming process, it is necessary to use various animations. There are different various JavaScript animation libraries available but the one called **Anime.js** was chosen for the web game. It is a lightweight animation library which works with CSS properties, SVG (Scalable Vector Graphics) image format, different DOM attributes, and JavaScript objects. The library includes several useful features like keyframes (chaining multiple animation properties), timeline (synchronization of multiple instances together), playback controls (play, pause restart features), function-based values (individual values for multiple animated targets) and many others. It also has very clear and understandable documentation along with community support. (Github community 2018.)

6 WEB GAME DEVELOPMENT

As was mentioned before, during the process of this thesis I started to work on an educational web game. The main goal of the game is to introduce the programming basics to the children outside the school environment. This chapter discusses the prototyping aspects of web game project development. This includes the concept and the idea of the game with a focus on audience description, style and design methods choice.

6.1 Concept

The idea of this project lies in the creation of an educational game in a web browser for primary school children as an extra curriculum activity. Children can play this game as a part of participation at Mirum Agency Helsinki office event. The ‘bring your kids to the office’ day is quite famous corporate world event where children are introduced to their mom’s and dad’s daily office activities. It is important and exciting for the child to know what their parents do at work, but it is more important to explain to the child the basics of each profession activities so

that they can not only understand but find it challenging and fun. That is how the idea of introducing the different professions in a form of a game came up.

There are multiple professions in the office which require programming skills like software engineer, web developer, front-end developer, back-end developer, etc. In order to explain to a child what programmers basically do at work. This game demo version is created. I personally do not find the idea of instructing an activity out of its environment (for example, learning programming with a board game or a book) attractive. Most modern children learned about the existence of multiple digital devices like computers, mobile phones and tablets since early childhood. Digital game is the best way to learn about communicating with a computer.

There are a lot of essential requirements for creating such a game in a specific environment. First of all, the game should be fast and easily accessible. The web format of a game fully covers these requirements, because it can be easily launched from the browser window without any installation and configuration steps. Next, the game should be short and not difficult to understand, because there is a limited time for each child to play it during the event. There is no multiple level structure, strong storyline or countless additional features, because the main objective is speed and easiness in the gaming process instead of keeping the player in the game for as long as possible. In addition, the colorful and child friendly design is a must have for an educative game. Fortunately, the final design solution for this game will be provided by the professional creative team of a supervising company. Finally, the game should be more entertaining than educational. The purpose of the game is to show what skills are required for a person to be a programmer and make the whole process enjoyable. Serious education can only start only after the independently made decision of a child to try it or not after playing this game.

6.1.1 Game Idea

The game follows the aim of introducing the programming world to children in a short, understandable and entertaining form. The gameplay is based on the idea of communication between a human and computer. The most necessary thing

needed for playing this game is a computer or laptop camera and a browser window. The whole gameplay bases on the camera color detection system and player decisions.

The gameplay process is divided into two different but connected to each other tasks the player has to perform in order to understand how the algorithm concept in programming work. The idea of the game relies on the concept of dividing a simple activity process to the sequence algorithm of smaller actions a human have to perform in order to have it done properly. For example, making a pizza process can be divided to rolling out the dough for the pizza base, adding a tomato sauce, adding toppings, baking and cutting activities. These activities have to be performed in a strict sequence of actions, otherwise, the desired result (a pizza) would not be achieved.

During the first part of a game the player has to show the computer in what order to perform the activities for the positive outcome. There is a number of different actions and human movements player can perform in front of the camera. They can be read by the computer system and translated to various animations which later performed on the screen. The first part of the game is based on the ability of recognition systems to define the color on a video stream. Player has can see a set of available colors together with a set of pizza cooking actions. Each color is connected to a certain activity. Player has to find out the right order of the actions in order to make pizza and show the colors in the corresponding order. The player also has a set of physical color cards to show to the camera. In case the command was correct and the color detection system reads the color card correctly the animation dependent to that exact color will appear, otherwise there is an error pop up on the screen.

The second part of the game is based on the understanding how computers can understand what action to perform accordingly to the human commands. The player has a set of the same pizza cooking actions presented on the screen in form of icons. Using drag and drop feature the player have to place each action icon to the certain placeholder to make the correct sequence of actions. After that

the animation can be started by pressing the ‘run’ button. In case the action sequence was not guessed, the error window will appear on the screen.

6.1.2 Target Group

The educational interactive web game is primarily aimed at children aged seven to ten but can also be fun for adults without programming or technological background. The target group average age was calculated based on the ages of employee’s children who agreed to participate in the event and the target audience age survey.

6.2 Game Design and User Interface

For the whole process of developing a game, the design side was mostly avoided while the whole attention focused on the functional side of production. Most of the objects during the implementation of the game core structure and interactive functional were presented in the form of primitive placeholders. The rough estimation of the layout and object design was a result of the separation of duties made by the project manager of the company. The main focus of web developer lays on providing the functional game core architecture while the designer’s center of attraction is a creation of visual side of game style, scenes and objects. Unfortunately, according to the corporate laws of protecting the intellectual property of each employee, the final design side of the web game project cannot be described and presented inside this thesis work.

Despite the fact that most of the visual materials for the game can be provided by design professionals, the core and basic design of game layout and elements can help the game developer to navigate inside the game elements more easily and find mistakes faster. In order not to spend a lot of the time on designing elements of the game for the personal use of developer, the minimalistic approach may be used. This approach includes the basic color palette including different colors for each game scene on the screen, the key style set of design for each element on the screen (buttons, animation elements, tabs, etc.), use of open-source lightweight icons.

The paper cards made for the camera color tracking task done in five colors: cyan, magenta, yellow, blue and green. The cyan, magenta and yellow colors are supported by the **tracking.js** while blue and green have to be manually defined in the program. That particular colors were chosen because they are opposite in the RGB (red green blue) color model. Their rgb color channel representation varies from minimum (0) to maximum (255), so they are more easily defined by the **tracking.js**.

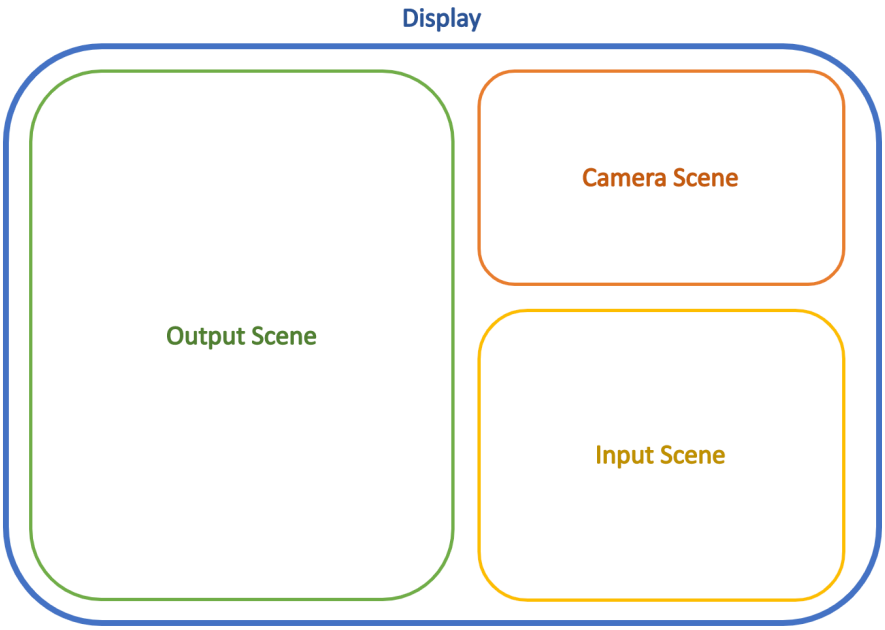

Figure 3. Game layout

The game layout (Figure 3) is divided by three visually independent areas each of them performing a certain role in the gameplay. The whole screen of the game is vertically divided by two parts in similar size. The left half of the screen is an output scene where all of the animations presented as a result based on the player actions. The left half of the screen is also horizontally divided by two parts in the approximate 1: 2 proportion. The top right part is a camera scene where the video output of the computer camera translation is showed. The bigger lower right part of the layout is shown as an input scene with two tabs each presenting different concepts. The first tab shows the gameplay instructions for the first task

of the game while the second tab presents an interactive drag and drop area for the second task of the gameplay.

7 GAME IMPLEMENTATION

This section describes in detail each step of the game development process along with implementation technologies and features.

7.1 Predevelopment

The first phase of each game development process requires the installation of the core tools, libraries, and technologies among with a directory structure creation. The process of educational web game project development is described for macOS.

7.1.1 Packages Installation

First of all, the text editor has to be chosen in order to provide a production environment for a programmer. There is always a constant battle on choosing the best code editor among developers, but the truth is there is no number one for everyone. Still, there are several popular ones like Sublime Text, Atom or Visual Studio Code. Since Sublime Text editor is old-fashioned, expensive and lacking multiple useful features and Atom is free for use and very comfortable in means of customization in my opinion, I decided to try the new for me editor Visual Studio Code. This editor is made by Microsoft is a free newcomer in the field of programming tools (it was released in 2015). The impression from VS Code is surprisingly positive because of GitHub (web-based software development platform) integration, customization, in-editor debugger and many other features it has. It also has a build-in command line interface which is a perfect tool for packages installation and project managing through simple and short commands.

The project creating process started with the installation of all the needed JavaScript libraries and tools. To begin with the project setup, the Homebrew, package manager for macOs, is installed on the first place. This tool makes the process of installing multiple packages and additional software very easy and fast

with a help of command line. Secondly, in order to make use of the simple interface to work with packages and tools the **Node.js** and package manager **npm** have to be installed first with a help of brew installation commands. All the next packages can be easily installed with the help of **npm**. After that, **Webpack** module bundler may come in use. It can be set up through the command line with the help of **npm** installation commands.

The most challenging part of the installation process is related to the choice of the game engine to use in the web game. The reason for the first mistake of game engine choice is related to the lack of practical skills in this field. The most popular and suitable for the needs of educational web game engines are **Phaser.js** and **PixiJS**. On the first sight, both sound and animation engines and keyboard input feature are not supported by **PixiJS** when **Phaser.js** is a wide library with endless features. Unfortunately, the latest release of **Phaser.js** has major changes in the core structure and lacks proper documentation which makes the education process very slow and hard. Moreover, the last version of **Phaser.js** excludes the video streaming feature which is essential for the web game development project. As a result, the **PixiJS** was chosen for the game engine of this project and the experience was very positive.

7.1.2 Directory Structure

The project should have the clear directory structure in order to create clear hierarchical environment and allow the packet managers and module bundler work properly. The directory structure for the web game project can be seen on the Figure 4 below.

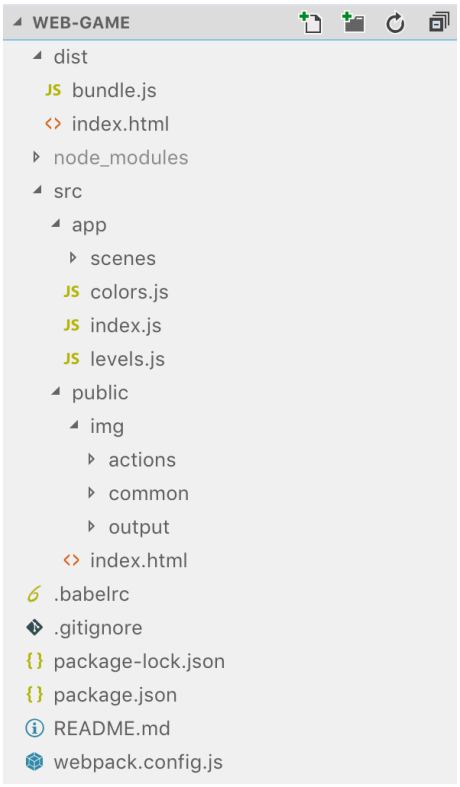

Figure 4. Game directory structure

There is *node_modules* directory containing all of the build tools and dependencies can be found inside the main directory along with the **package.json** file defining the libraries installed. The *src* directory is the main project container including the app and public directories. All JavaScript files should be stored in *src/app* directory and the *src/app/index.js* is the entry point of the whole project. The *src/public* directory is used for holding different assets and static files used in the project and *src/public/index.html* file is the main project template. There is another directory inside the main one called *dist*. It contains the **index.html** and **bundle.js** files. These files are created as the result of the **Webpack** building process of entry point */src/app/index.js*. The **bundle.js** file is a result of compilation all files from *src/app* directory with the help of **Webpack**. In order to use development server instance connecting to localhost with live reloading feature the *webpack-dev-server* was installed with **npm** scripts. Finally, with the help of **npm** the **babel.js** and **tracking.js** libraries are set up for the project.

7.2 Game Layout

According to the concept of the game, the browser page window should be divided into several different areas each performing the unique function. In the game design, the independent parts of the game executing different actions are called scenes. The game should have three scenes: video streaming scene, input scene (where a player can solve a problem), output scene (where a player can see the result of actions). The scene hierarchy structure is more complex internally. The process of implementation and configuration of scenes and their dependencies is described in details in this chapter.

All the scenes are located in the *scenes* folder in *src/app* directory. Each of the scenes is represented by a JavaScript file. The main scene which contains video, input and output scenes and covers the whole game playing area is called a root scene (**root.js**). Basic parameters of the scene are defined in the BaseScene class (**baseScene.js**). Each of the scenes is extending it and all the shared code is defined there. The scene which is showing the video output of the computer camera is called camera scene (**camera.js**). The input and output ones are more complex, so the components related to each scene are contained in corresponding folders. The output scene (**outputScene.js**) and its components are located in the *output* folder. The input scene (**inputScene.js**) which is located in the *input* folder contains two independent scenes based on two tasks the player have to perform. The independent input scenes (**colorInputScene.js** and **sequenceInputScene.js**) are presented as tabs which means only one of them can be visible to the player. (Figure 5.)

Figure 5. Game scenes structure

All elements which are rendered to the screen are extending **PIXI.DisplayObject** class, whereas scenes use **PIXI.Container** as a base. It was designed to represent a collection of display objects and acts as a container for them. This class is very useful for displaying different scenes or collection of the same objects inside one scene. The **PixiJS** library describes this relationship as “parent-child”, so each container has **addChild**/**removeChild** methods.

7.3 Camera Scene

The camera scene displays an output of the video stream captured by a camera. Under the hood it is also responsible for tracking colors using **tracking.js** library.

The video stream is captured using native browser mechanism. The following snippet is requesting camera input with resolution of 640x320 and 30 frames per second. This functionality requires explicit user approval, which is automatically requested by browser.

```
const streamPromise = navigator.mediaDevices.getUserMedia({
  video: {
    width: 640,
```

```
    height: 360,
    frameRate: 30
  }
})
```

Code 1. Camera input request

To display the received stream, it should be attached to a **<video>** element which will be rendered to the scene later. This is also important for integration with **tracking.js**, which processes output of that element. The **getUserMedia** method returns a promise with an actual stream, which is resolved in Code 2.

```
// Video HTML5 element
this.video = document.getElementById('cameraVideo')
streamPromise
  .then(stream => {
    // Assign video stream to video element
    this.video.srcObject = stream
  })
  .catch(err => {
    // Display dialog with error description
    alert(err)
  })
```

Code 2. Displaying video stream

The same **<video>** element is converted to **PIXI.Sprite** and displayed inside the scene.

Tracking.js provides minimalistic API to extract coordinates of objects with specific color from camera input. Supported colors are defined in the *color.js* file alongside with tracking configuration for each of them (Code 3). They should be registered in **tracking.ColorTracker** to be able to distinguish them.

```
'red': {
    hex: 0xFF0000,
    trackingConfig: function(r, g, b) {
        return r > 200 && g < 50 && b < 50
    }
}
```

Code 3. Color definition example

The results of color tracking process are passed to the input scene.

7.4 Input Scene

The input scene contains two independent tabs, each for the corresponding task. It passes information about current actions to the output. In case the first tab is open, the instructions for the camera color detection task are shown. In the opposite case, the input field with drag and drop game is displayed. The input scene controls what game is played at the moment and what kind of animations inside the output scene should be shown to the player.

7.4.1 Tab Controls

Tab functionality is controlled by the input scene. It responds to events from a header which is extracted as a separate component. Tab header checks for an active tab after each click and emits an event with corresponding index if tab was changed. The code snippet below shows how this component is attached to the scene.

```
this.tabHeader = new TabHeader(
  this.tabParams,
  clickedIndex => this.setTabActive(clickedIndex)
)
this.addChild(this.tabHeader)
```

Code 4. Tab header creation

The input scene reacts to the tab header events by calling **setTabActive** method. There it finds new contents based on event parameters and replaces the previous ones (Code 5).

```
setTabActive(index) {
    // Remove previous scene
    if (this.currentTabIndex != null) {
      const previousScene = this.sceneParams[this.currentTabIndex].scene
      this.removeChild(previousScene)
    }
```

```
    // Add new one
    this.currentTabIndex = index
    const newScene = this.sceneParams[this.currentTabIndex].scene
    this.addChild(newScene)
}
```

Code 5. Change contents of the tab

7.4.2 Camera Input Scene

Camera input scene describes relation between colors and actions, displays it to the player and passes that information to a parent.

```
colorToAction: [
    { color: 'cyan', action: 'actions/dough' },
    { color: 'magenta', action: 'actions/oven' }
]
```

Code 6. Relation between color and action

Each action is defined as a string, which corresponds to its icon and animation for an output.

7.4.3 Sequence Input Scene

The sequence input is much more complex as it includes drag and drop algorithm builder, which requires player interaction with scene elements. It consists of two containers: a pool of initial actions and an execution sequence. They are extracted as separate components to handle prevent bloating the scene. Therefore, it only handles interaction between its parts as well as communication with parent.

The drag and drop logic relies on three events: **mousedown** to capture initial mouse button press, **mousemove** for tracking cursor position and **mouseup** to end drag (drop). These interactions are handled inside the scene and both child components are checked when item is dropped. Then, it is either accepted to an algorithm cell or returned back to an action pool.

The **mousedown** listener is attached to each individual action element. It triggers **startDrag** method in the scene, where the initial preparation for event handling are done: event listeners for actual drag and drop are attached and the element is saved for further interactions. (Code 7)

```
startDrag(event) {
    const element = event.currentTarget
    window.addEventListener('mousemove', this.dragListener)
    window.addEventListener('mouseup', this.dragStopListener)

    this.addChild(element)
    this.dragElement = element
}
```

Code 7. Start drag process

Afterwards, that element follows movement of the cursor. The drop container checks for collision and highlights a suitable cell. (Code 8)

```
drag(event) {
    const element = this.dragElement
    const eventPosition = new PIXI.Point(event.clientX, event.clientY)

    element.position = eventPosition
    this.dropContainer.onDrag(element)
}
```

Code 8. Continue drag process

When the mouse button is released the listeners are removed and element is either accepted by drop container or moved back to the action pool. Moreover, it is important to check whether the cell is not empty, so the player will not put two actions in one place. (Code 9)

```
stopDrag(event) {
    const element = this.dragElement

    window.removeEventListener('mousemove', this.dragListener)
    window.removeEventListener('mouseup', this.dragStopListener)

    const [accepted, elementToReturn] = this.dropContainer.onStopDrag(element)
    if (accepted) {
        if (elementToReturn) this.dragContainer.onStopDrag(elementToReturn)
    } else {
```

```
        this.dragContainer.onStopDrag(element)
    }
}
```

Code 9. Drop element

After player had placed all the actions, the button click will pass the sequence to the parent and then to the output scene for the validation and execution.

7.5 Output Scene

The output scene displays action animations based on player input from camera or sequence builder. It behaves in these two modes differently:

- When input is fed by showing colors, it is awaiting next action and displaying hints.

- When the task is to create a sequence, it is restarting all the animations from the beginning every time.

The mode depends on the active tab in the input scene.

When action identifiers are passed to the output scene, they are converted to sprites with attached animation. Each of them is created using factory parameterized by action identifier. In the following snippet, the duration is controlled with external parameter, which makes it more flexible.

```
const outputAnimations = {
    'actions/dough': doughAnimation,
    'actions/sauce': sauceAnimation
}

function doughAnimation(duration) {
    const dough = new PIXI.Graphics()
    dough.beginFill(0xeddbc5)
        .drawCircle(50, 50, 50)
        .endFill()

    const animation = anime({
        targets: dough.scale,
```

```
        duration: duration,
        x: 4,
        y: 4,
        easing: 'easeInOutBack'
    })
    dough.on('added', () => animation.restart())
    dough.pivot = new PIXI.Point(50, 50)

    return dough
}
```

Code 10. Animation definition example

In the example above, the **dough** is displayed in a form of circle which is created with **PixiJS**. Its **scale** property is updated using **anime.js** library. Animation is triggered only when the element is displayed on the screen.

The sequence mode requires action elements to be applied one after another. This is achieved with **setTimeout**. It is built-in JavaScript function which allows to delay execution of the code block.

The created action sprites with identifier are contained inside **this.sprites** array, whereas correct action order is inside **this.actions** array. On each step, the next sprite is displayed and the following action is determined. In case all of the actions were shown in correct order, the task is successfully accomplished. When the amount or order of actions is mistaken, the error screen is displayed. Otherwise, the next operation is recursively scheduled after animation duration. (Code 11)

```
scheduleSequence(millis) {
  this.sequenceTimeout = setTimeout(() => {

    const [id, sprite, duration] = this.sprites.shift()
    const correctId = this.actions.shift()

    this.addChild(sprite)

    const isSuccess = this.actions.length == 0 && this.sprites.length == 0
    const isError = this.sprites.length == 0 || correctId != id
    const hasNext = this.sprites.length > 0

    if (isSuccess) {
```

```
      this.showSuccess()
    } else if (isError) {
      this.showError()
    } else if (hasNext) {
      this.scheduleSequence(duration)
    }
  }, millis)
}
```

Code 11. Sequence presentation logic

The camera mode follows the similar logic, although it is showing the action straight away and is reverted to a previous step in case of error.

7.6 Summary

Overall, the web game implementation result was successful. The game demo version presents two relative exercises. Each exercise requires the player to interact with different gameplay features. The pattern algorithm learned in the first task should be implemented during the second in a different way. The animation response of the game shows player if their decisions were correct. The web game takes a short amount of time to participate and provides the same level of entertainment and education.

There were several challenging details of the web game implementation process which required more effort on practice than have been expected in theory. First of all, the game engine choosing process failed on a first try. Based on this experience, more time and concentration were taken for consideration of new technologies for practical implementation. Moreover, the color tracking technology is very sensitive to the lightning and surroundings around the player, so the color detection properties should be changed accordingly to each environment. This requires the time for preparation before the game is presented, which should be taken into account.

8 CONCLUSION

The knowledge of computer science and programming basics plays a very important role in modern children education. It is as critical and essential skill as

the understanding of math, literacy or foreign languages. The most efficient way to teach children computer literacy is to provide the appropriate environment and make the process entertaining. In other words, in a form of educational video game. The global intent of this thesis work was to break the misconception that game is a form of distraction and entertainment.

The aim of this thesis was to study the role of video games in the modern children education along with the possible ways of teaching programming basics and how that can be implemented in practice in a form of web game. The theory part described the differences between traditional and modern teaching methods, video game concept, the impact of video game appearance on modern education. The basics of programming education and the combination of programming teaching and video games were also discussed in the theory part of this thesis.

The combination of education and games with a high possibility of a positive outcome became a focal point of the practical part of this thesis. The main idea for practical part was to develop an educational web game which describes the basics of programming for the company employee's children during the corporate event.

To summarize, the project was a success as the goals defined for the study achieved and the practical implementation resulted in a functional product outcome. Throughout the development process of this thesis, I gained valuable experience for my future career growth. Unfortunately, according to the corporate restrictions the game is not visible for the public except the basic demo version available during the thesis presentation for evaluation.

8.1 Ideas for Future Development

The project development performed in this project was a demo version, based on which a team of programmers, designers and testers can develop a real product. There are several improvements for future related to the practical implementation of the game which are stated below.

- **Game Test**

  One of the crucial steps in the process of creation each web development product is testing. The game tests help to provide quality control, optimization and performance boost for the final product. Web game testing methods are based on identifying the bugs (failures or faults in the product performance). There are different areas of game performance where tests have to be done in order to provide positive user experience. For example, the game has to be tested for cross browser compatibility (positive performance on different browsers) in order to find out what elements or features may fail and fix them.

- **Multiple game tasks**

  Since the thesis project outcome is a demo version, the final game can be expanded for multiple tasks educating different programming aspects with different interactive elements so the player has the ability to make a choice. The face or motion detection task can be applied to the game along with loop or function concept educational exercise.

- **Difficulty levels**

  Providing the various difficulty levels for each task can provide better user experience. Each player can choose the level of each task according to the personal experience. There can be at least two levels in the web game. The easy level includes the hint feature, while difficult level does not.

- **Accessibility**

  The computer science and programming education among with entertainment of playing video games can be available for every child. Sometimes, disability can be a serious barrier for participating in the interaction with computer process but modern technologies can help to solve that issue. There are multiple guidelines how to improve the game design and mechanics in order to be accessible for children with cognitive,

physical, intellectual, sensory or mental disabilities. For example, the game can be played in different form for children with vision difficulties. The camera tracking interaction task can be changed for voice recognition and the basic elements and principles can be described with voice over text feature applied.

Overall, the field of additional features for game development is endless as for every web development project. It is limited only by the programmer imagination and set of skills both of which can be expanded by more practice.

REFERENCES

Babeljs.io. 2018. What is babel? WWW document. Available at: https://babeljs.io/docs/en/index.html [Accessed 22 November 2018].

Baker, M. 2016. How Do Game Engines Work? WWW document. Available at: https://interestingengineering.com/how-game-engines-work [Accessed 22 November 2018].

Boller, S. 2013. Learning Game Design: Game Elements. Blog. Available at: http://www.theknowledgeguru.com/learning-game-design-game-elements/ [Accessed 22 November 2018].

Bolton, D. 2018. What Is Computer Programming? Blog. Updated October 5, 2018. Available at: https://www.thoughtco.com/what-is-programming-958331 [Accessed 22 November 2018].

Brew.sh. 2018. Homebrew. WWW document. Available at: https://brew.sh/ [Accessed 22 November 2018].

Brogan, J. 2016. What's the Deal With Algorithms? WWW document. Available at: http://www.slate.com/articles/technology/future_tense/2016/02/what_is_an_algorithm_an_explainer.html?via=gdpr-consent [Accessed 22 November 2018].

Brown, P. 2017. State of the Union: npm. WWW document. Available at: https://www.linux.com/news/event/Nodejs/2016/state-union-npm [Accessed 22 November 2018].

Bruner, J. S. 1961. The act of discovery. *Harvard Educational Review* 31, 21–32, 71–72.

Burak, A., Parker, L. 2017. Power Play. How Video Games Can Save the World. New York City: St. Martin's Press.

Cleverism Blog. 2015. Top Programming Languages Used in Web Development. Blog. Available at: https://www.cleverism.com/programming-languages-web-development/ [Accessed 22 November 2018].

Developer blog. 2011. Seven things you should know if you're starting out programming. Blog. Available at: https://www.theguardian.com/info/developer-blog/2011/oct/07/programming-developer-journalist [Accessed 22 November 2018].

Dodge, D. 2016. 4 Reasons Why Minecraft Is Great for Teaching Kids Coding. WWW document. Available at: https://tech.co/minecraft-great-teaching-kids-coding-2016-12 [Accessed 22 November 2018].

Erstad, W. 2014. Online vs. Traditional Education: What You Need to Know. WWW document. Updated August 16, 2017. Available at: https://www.rasmussen.edu/student-life/blogs/college-life/online-vs-traditional-education-answer-never-expected/ [Accessed 22 November 2018].

Gamelearn. 2017. Eight examples that explain all you need to know about serious games and game-based learning. Blog. Available at: https://www.game-learn.com/all-you-need-to-know-serious-games-game-based-learning-examples/ [Accessed 22 November 2018].

Github community. 2018. Anime.js. WWW document. Available at: https://github.com/juliangarnier/anime [Accessed 22 November 2018].

Graetz, J., M. 1981. The origin of Spacewar. *Creative Computing* 18.

Gray, P. 2008. A Brief History of Education. WWW document. Available at: https://www.psychologytoday.com/gb/blog/freedom-learn/200808/brief-history-education [Accessed 22 November 2018].

iD Tech. 2018. The Many Different Types of Video Games & Their Subgenres. Blog. Available at: https://www.idtech.com/blog/different-types-of-video-game-genres [Accessed 22 November 2018].

Johnson, L., Levine, A., Smith, R., & Stone, S. 2010. The 2010 Horizon Report. PDF document. Available at: https://files.eric.ed.gov/fulltext/ED510220.pdf [Accessed 22 November 2018].

Kennedy, R. 2017. Progressive Education: How Children Learn. WWW document. Updated March 18, 2017. Available at: https://www.thoughtco.com/progressive-education-how-children-learn-today-2774713 [Accessed 22 November 2018].

Kohn, A. 2008. Progressive Education. WWW document. Available at: https://www.nais.org/magazine/independent-school/spring-2008/progressive-education/ [Accessed 22 November 2018].

Kyrnin, J. 2018. HTML5 Canvas Uses. This element has benefits over other technology. WWW document. Available at: https://www.lifewire.com/why-use-html5-canvas-3467995 [Accessed 22 November 2018].

Liukas, L. 2016. A delightful way to teach kids about computers. TEDxCERN event in Meyrin, Switzerland 9 October 2015. Video Clip. Available at: https://www.ted.com/talks/linda_liukas_a_delightful_way_to_teach_kids_about_computers [Accessed 22 November 2018].

Mackay, R. 2013. Playing to learn: Panelists at Stanford discussion say using games as an educational tool provides opportunities for deeper learning. WWW document. Available at: https://news.stanford.edu/2013/03/01/games-education-tool-030113/ [Accessed 22 November 2018].

Majumdar, A. 2016. 5 Game Elements That Create Effective Learning Games. WWW document. Available at: https://elearningindustry.com/5-game-elements-create-effective-learning-games [Accessed 22 November 2018].

Maton, N. 2011. How Games Can Influence Learning. WWW document. Available at: https://www.kqed.org/mindshift/16098/how-games-can-influence-learning [Accessed 22 November 2018].

Meador, D. 2018. The Pros and Cons of Using a Traditional Grading Scale. WWW document. Updated July 10, 2018. Available at: https://www.thoughtco.com/the-pros-and-cons-of-utilizing-a-traditional-grading-scale-3194752 [Accessed 22 November 2018].

Npmjs.com. 2018. About npm. WWW document. Available at: https://docs.npmjs.com/about-npm/ [Accessed 22 November 2018].

O'Boyle, B., Willings, A. 2018. What is VR? Virtual reality explained. WWW document. Available at: https://www.pocket-lint.com/ar-vr/news/136540-what-is-vr-virtual-reality-explained [Accessed 22 November 2018].

Ourcodeworld. 2016. Top 15: Best open source javascript game engines. WWW document. Available at: https://ourcodeworld.com/articles/read/308/top-15-best-open-source-javascript-game-engines [Accessed 22 November 2018].

Panworld Education. 2017. Benefits of Digital Learning Over Traditional Education Methods. Blog. Updated March 23, 2017. Available at: http://www.panworldeducation.com/2017/03/23/benefits-of-digital-learning-over-traditional-education-methods/ [Accessed 22 November 2018].

Pappas, C. 2015. 7 Tips To Create Effective eLearning Games. Blog. Available at: https://elearningindustry.com/7-tips-create-effective-elearning-games [Accessed 22 November 2018].

Ramlagan, N., Tretkoff, E. 2008. October 1958: Physicist Invents First Video Game. WWW document. Available at: https://www.aps.org/publications/apsnews/200810/physicshistory.cfm [Accessed 22 November 2018].

Rauschmayer, A. 2014. Speaking JavaScript: An In-Depth Guide for Programmers. Sebastopol: O'Reilly Media.

Rouse, M. 2017. Scratch. WWW document. Available at: https://whatis.techtarget.com/definition/Scratch [Accessed 22 November 2018].

Rubin, P. 2014. Check Out This Glorious, Colorful History of Arcade Games. WWW document. Available at: https://www.wired.com/2014/05/arcade-history/ [Accessed 22 November 2018].

Sridhar, J. 2017. What Is JavaScript and How Does It Work? WWW document. Available at: https://www.makeuseof.com/tag/what-is-javascript/ [Accessed 22 November 2018].

Study.com, 2018. 5 Basic Elements Of Programming. WWW document. Available at: https://study.com/academy/lesson/5-basic-elements-of-programming.html#/lesson [Accessed 22 November 2018].

Tavinor, G. 2008. Definition of Videogames. PDF document. Available at: https://digitalcommons.risd.edu/cgi/viewcontent.cgi?article=1140&context=liberalarts_contempaesthetics [Accessed 22 November 2018].

Teach.com. 2018. Teaching Methods. WWW document. Available at: https://teach.com/what/teachers-know/teaching-methods/ [Accessed 22 November 2018].

Thomas, D. 2014. CodeCombat Game Review. WWW document. Available at: https://www.commonsensemedia.org/game-reviews/codecombat [Accessed 22 November 2018].

Thompson, V. 2018. The Goals of a Traditional Education. WWW document. Updated August 10, 2018. Available at: https://classroom.synonym.com/goals-traditional-education-8023.html [Accessed 22 November 2018].

Trackingjs.com. 2018. Tracking.js. WWW document. Available at: https://trackingjs.com/docs.html#introduction [Accessed 22 November 2018].

Video Game History. 2017. WWW document. Updated August 21 2018. Available at: https://www.history.com/topics/inventions/history-of-video-games [Accessed 22 November 2018]

Webpack.js.org. 2018. Concepts. WWW document. Available at: https://webpack.js.org/concepts/ [Accessed 22 November 2018].

Williams, G. 2015. Understand game platform types. Blog. Available at: https://georgiawilliams5.wordpress.com/2015/01/12/understand-game-platform-types/ [Accessed 22 November 2018].