Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Computer Games and Graphics

A set of educational tools for game development course for videogames

Bachelor Thesis

Štěpán Machovský

Field of study: Computer Games and Graphics
Supervisor: doc. Ing. Jiří Bittner, Ph.D.

August 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Machovský** Jméno: **Štěpán** Osobní číslo: **474661**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Sada výukových nástrojů pro kurz herního vývoje

Název bakalářské práce anglicky:

A set of educational tools for game development course

Pokyny pro vypracování:

Zmapujte způsoby implementace výukových praktik do podpůrných výukových programů. Popište základní principy a metody implementace takového programu a rozhodněte, které z nich využijete ve svém projektu.
Navrhněte metodiku pro implementaci sady jednoduchých výukových programů na podporu výuky základních technických komponent herního vývoje.
Podle navržené metodiky implementujte výukové programy pro 3D transformace, animace, fyzikální simulaci a umělou inteligenci. Pro implementaci využijte herní engine Unity. Vytvořenou implementaci otestujte pomocí uživatelských testů.

Seznam doporučené literatury:

[1] Jesse Schell. The Art of Game Design: A book of lenses. CRC Press, 2008.
[2] Raph Koster. Theory of Fun for Game Design, 2nd edition, O'Reilly Media, 2013.
[3] Simon Egenfeldt-Nielsen, Jonas Heide Smith, Susana Pajares Tosca. Understanding Video Games, 3rd edition. Taylor & Francis, 2016.
[4] Jason Gregory. Game Engine Architecture (3rd edition). CRC Press, 2018.
[5] Michelene T. H. Chi and Ruth Wylie. The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes, Educational Psychologist, 49(4), 219–243, 2014.
[6] De Jong, T., & Van Joolingen, W. R. Scientific Discovery Learning with Computer Simulations of Conceptual Domains. Review of Educational Research, 68(2), 179–201, 1998.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jiří Bittner, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **11.02.2020** Termín odevzdání bakalářské práce: **14.08.2020**

Platnost zadání bakalářské práce: **30.09.2021**

doc. Ing. Jiří Bittner, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Acknowledgements

I would like to express my gratitude to my supervisor Jiří Bittner, associate professor in the Department of Computer Graphics and Interaction, for his useful advice, comments, remarks and guidance. I would also like to thank Magdaléna Oubrechtová for emotional support and Radek Chyla for text correction. Furthermore I would like to thank my parents for all the support, emotional and financial, they provided me with.

Declaration

I hereby declare I have written this thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis.

Prague August 13, 2020

Abstract

When creating an edutainment game or tools, we want to achieve the best learning outcomes. Concurrently we want to minimize the distractions in the game and tools, as it can decrease the teaching value of the learning materials used. Implementing such a tool can quickly become a daunting challenge. This thesis gives an overview of serious games, Edutainment and techniques used for achieving more significant learning outcomes.

We have implemented teaching tools for a game design course in Unity. We have done user and performance test and created hardware requirements for the teaching tools.

Keywords: Edutainment, Serious Games, Unity, Teaching Tools, Teaching Games

Supervisor:
doc. Ing. Jiří Bittner, Ph.D.
DCGI, FEE, CTU in Prague
Praha 2, Karlovo náměstí 13

Při vytváření výukových her, či nástrojů, chceme dosáhnout co největšího nejvyššího výsledku výuky. Zároveň chceme minimalizovat rušivé elementy ve hře a nástrojích, které mohou snížit výukový potenciál použitých materiálů. Implementace takovéhoto nástroje, se záhy může stát složitou výzvou. Tato práce dává čtenáři přehled o serious games, Edutainmentu a technikách používaných k dosažení vyšších výsledků výuky.

Implementovali jsme výukové nástroje pro herní design v Unity. Provedli jsme uživatelské a hardwarové testy a vytvořili pro již zmíněné výukové nástroje hardwarové požadavky.

Klíčová slova: Edutainment, Serious Games, Unity, Výukové Nástroje, Výukové Hry

Překlad názvu: Sada výukových nástrojů pro kurz herního vývoje

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the computer-based entertainment market grows, so does the support for education to be implemented into it. Games usage and reach go far beyond the scope of pure entertainment. With games reflecting reality more and more [19], they get to immerse the players on whole another level.

This immersion not only helps the players relate, but it also creates a place for developers to implement teaching, which will feel almost natural for the player. As a prime example of this, Assasin's Creed: Origins (Figure 1.1) found a peculiar way to implement education into an AAA game. Ubisoft, the creator of Assasin's Creed series, released a mode for the game called Discovery Tour mode [9]. In the mode, players can wander in a scaled-down version of Alexandria, with 75 different "Guided Tours" throughout the city and learn about Egypt and Alexander the Great.

Figure 1.1: Screenshot of Assasin's Creed: Origins by GamePress.

However impressive AAA games with a hint of Edutainment are, the developers of Edutainment titles often work with way smaller budget, than the insane budgets, AAA games have. Therefore the resulting products are often not as well-polished and not as well-propagated as they could be. Nevertheless, recently a few successful edutainment titles have emerged. To name a few, we should name Duolingo, Yousician and Hopscotch-Programming for kids.

Another way how to bolster education is the implementation of teaching tools. Teaching tools do not have to be as elaborate as teaching games, as they do not have to implement a way to support intrinsic motivation for the students. Teaching tools can give the information more straightforwardly, as the students are self-motivated to learn.

In this thesis, we will cover the basics of Serious Games, the basics of Edutainment and different approaches and frameworks for implementing teaching tools and educational games. We will also include niche parts of four fundamental elements of game design. Then we will analyze these approaches and frameworks, integrate them into a creating process of Teaching Tools and implement them in Unity. Finally, we will test the Teaching Tools with users, and we will also check the Teaching Tools on different hardware and set requirements for it.

Chapter 2

Educational techniques and methods

To understand how to implement teaching tools, we will cover various techniques and methods used for implementing education. Firstly, we will talk about ICAP framework. ICAP is an acronym of its four learning modes, *interactive*, *constructive*, *active* and *passive*, although when mentioned the order is often reversed, form the lowest mode of learning to the highest. Then we move onto Bloom's taxonomy. There we discuss its levels, the revision and impact of the internet on it and education in general. Lastly, we will cover discovery learning and its implementation.

2.1 ICAP

ICAP [6] is an educational framework that builds on the idea that students have better results when they are learning actively. It separates learning stages into four modes, through which the student gradually grows in his knowledge of the subject. The modes are *passive*, *active*, *constructive* and *interactive*.

Passive

In the *passive* mode, the student is passively absorbing all the given information, and not creating any outputs of his own. Examples of this mode are when the student is reading a book, watching a video or listening to the lecture, without creating any notes.

The *passive* mode is not worth implementing into a teaching tool. The time invested in such a tool would benefit the student more if invested into making

the sources more approachable. The primary source for the student while in this mode should be videos, lectures and text sources.

Active

In the *active* mode, the student is not only passively absorbing the information, but he actively tries to extract main ideas from the given information. Examples of this behaviour are highlighting in book, replaying the most appealing or most essential parts of a video, or taking notes at a lecture.

The situation for implementation of the *active* mode into teaching tools is the same as with the *passive* mode. The primary source for the *active* mode should be lectures, videos and written text.

Constructive

This mode is the first one with real outputs, even though the student does not need to generate any. In this mode, the student goes more in-depth and tries to make conclusions on his own. Some examples are reflecting out-loud, drawing concept maps, asking questions and taking notes in one's own words comparison and contrasting to prior knowledge or other materials.

The created tool will bridge this part of ICAP mode. Creating such an environment, that student can actively try and experiment with different settings. While also encouraging to do simple tasks to challenge student's understanding of the topic.

Interactive

While the other modes could be done without any peer to interact with, the *interactive* mode depends on a constructive debate. Therefore peer to discuss the matter with is essential. Some examples of this mode are defending and arguing a position in small groups, Asking and answering comprehension questions with a partner or debating with a peer about the justifications.

While implementing this mode into a teaching tool, it is essential to answer and argument all the possible questions. Advanced AI would seem perfect for this daunting task, but it would be very demanding for it to be flawless.

2.2 Bloom's taxonomy

The Bloom's taxonomy framework [30] has six different levels, those are *remembering, understanding, applying, analyzing, evaluating* and *creating.*

In an ideal situation, the student goes through each level, starting from the very basic things moving up to more in-depth knowledge. In Bloom's taxonomy, each level is dependent on the basics of its precedent one.

Roots of Bloom's taxonomy reaches back to the 1940s when Benjamin Bloom with Max Englehart, Edward Furst, Walter Hill and David Krathwohl developed Bloom's taxonomy for classification of educational goals. For the next 16 years, it has been revised and refined at the yearly American Psychological Association convention. It was not until 1956 when the final version was published, lining the path of knowledge acquisition with its six levels of learning. Ever since then, Bloom's taxonomy was vital for many teaching philosophies. It served for nearly 50 years as a tool for many teachers, helping them draw up their curriculums before its revision in 2001.

2.2.1 Revision

(Figure 2.1). In the 1990s a former student of Benjamin Bloom, Lorin Anderson and D. Krathworh reevaluated and revised Bloom's Taxonomy. Their book A Taxonomy for Learning, Teaching and Assessing: a Revision of Bloom's Taxonomy of Educational Objectives [4] was the manifestation of this reevaluation and revision.

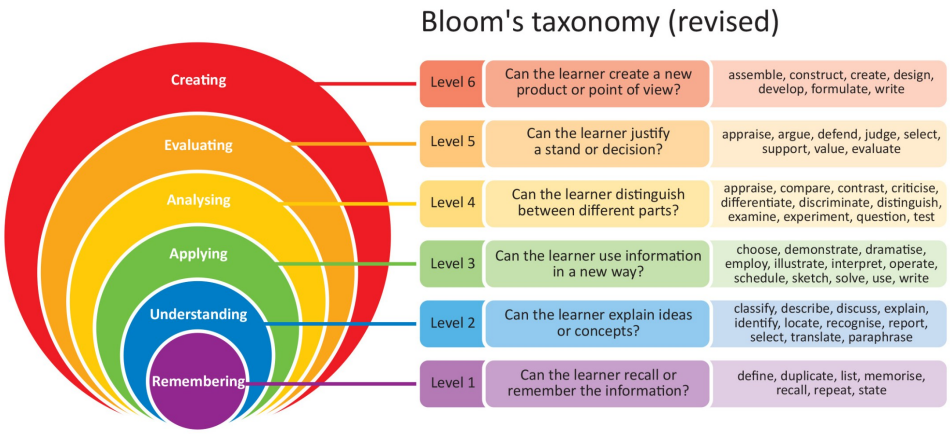

Figure 2.1: Bloom's Taxonomy Revision-visualised created by Niall McNulty

The main difference was transferring the nouns into verbs, as it better reflects what the student should be capable of in each different level and switching the highest order thinking levels *evaluation* and *synthesis.* The revised order of levels, as well as follow up questions and nouns describing the level on (Figure 2.1).

2.2.2 Impact of the Internet on Bloom's Taxonomy

It has never been as easy to share and access information, as in the past two decades. The Internet has become one of the most efficient, available and convenient means of education, always accessible, so it suits students' timelines better than anything before [11]. Students prefer googling over going into libraries for sources of information. With this in mind, Bloom's Taxonomy's suggestions to which steps are appropriate for each level might look outdated without the reflection of new technology in the classroom. So it is only natural, that articles, that update this framework emerged, which present additions to different levels of thinking [7].

Changes that affect the first level *remembering* are that students started to bullet-point instead of listing. They now bookmark different websites instead of books. Rather than using libraries as their source of information, students extensively utilise online searches on search engines like Google or Yahoo.

Second level *understanding* also changed. Use of online or computer-based tools for annotating and commenting *.pdf* files or other documents made it easier for an average student to organise his thoughts on the current matter better, with the use of hypertext links and indexing. The digital classification made it a breeze to classify, organise, and structure online data or web page.

Applying changed due to numerous factors. The most noteworthy change is the use of Edutainment. When the student plays or operates the game successfully, he exhibits an understanding of process and task, as well as the application of skill. Also uploading and sharing content online is seen as a simple form of higher-order thinking skill, collaboration.

Analysing received a few notable additions such as reverse-engineering or mashing. The former is analogous with deconstruction, where the student needs to exhibit an understanding of application or system, which he deconstructs. The latter is the integration of multiple information sources into one. Mashing is a complex process, but with more options present today, mashing becomes more straightforward and more accessible mean of analysis increasingly.

Blogging and new means of expressing one's thoughts are adding more and more ways *evaluating* can manifest. Blogs and video blogs create a safe environment for students to post their constructive criticism, reflect upon the subject, discuss the matter with their peers and to collaborate with others. With the emergence of testing, students can exhibit the ability and knowledge needed for analysis of the purpose and function of a process or a tool.

With more means of self-expression, there are more ways to fulfil Creating level of thinking. From posting video-blogs regarding the topic via Youtube through editing page on Wikipedia to programming software about the subject

are all valid aspects of the sixth and most sophisticated level of thinking, *creating*.

2.3 Discovery Learning

Discovery learning [8] is a constructivist, highly self-directed form of learning, also referred to as problem-based learning. This method can be summarised as learning through problem-solving with more than one solution, with minimal guidance, explanations, repetition and memorisation.

Discovery learning also shows signs of longer retention of information, as opposed to other learning techniques, like memorisation, repetition or class-based studying.

The learner's main task in the method is to deduce the underlying basics of an experiment, without any teacher interference. The learner changes inputs in a simulation and observes the resulted changes in the outcome of an experiment.

One of the most common problems with this method is that number of subjects, which tested this method, could not drop a thesis after creating one. Another common problem, described by Van Joolingen and De Jong [8], is called *fear of rejection*. In an analysis of 31 students, Joolingen and Jong found, that most of them avoided hypothesis with a high chance of rejection.

As a means of achieving discovery learning is creating an experiment, and let the learner find his conclusions, onto the matter. Then debrief it afterwards to correct any wrong conclusions. The debrief is essential due to learners tendency to create false conclusions because only a few of them create such experiments, that would disprove their conclusion.

It is also vital, to provide the accompanying text during the experimentation phase, as the students tend to not get any benefits from texts provided before it.

Physics part of the tools lightly hints discovery learning. The student can create an experiment, with three different variables and then observe the results.

Chapter 3

Serious Games and Edutainment

Serious games [13] are those games that do not have entertainment as a primary goal. Serious games can range from simulations to pedagogical use, and serious games have even seen some use in politics. Most serious games use intrinsic motivation brought by the added value of fun and competition. Serious games were born in the late 1950s, but back then it was mostly pen and paper. In this chapter, we will discuss different sub-genres of serious games, differences between gamification and serious games as well as examples of both.

3.1 Edutainment

Edutainment is one of the most influential subgenres of serious games. The term Edutainment also refers to any conjunction of entertainment with education. Edutainment was very popular on television in the '90s. One of the most famous educational TV shows is Bill Nye the Science Guy.

Studies suggest that games are viable alternatives for traditional means of teaching [13]. The studies also indicate that they provide a similar learning outcome, which means that the student has the same chance of learning material using traditional means or when using teaching games. So it is only expectable, that Edutainment represents a large part of serious games.

Games have always been a vital part of the teaching process. While not playful itself, at least the process of teaching in schools is gamified. It has been for a long time. The act of grading with marks, giving points and then summarizing the results at the end, could be taken as a mean of gamification.

Edutainment relies heavily on the premise that teacher or game developer can turn anything into a game, or at least gamify it. This thinking also brings a burden of responsibility for the process of teaching, that it would not turn into playing mindlessly, but rather with the purpose of learning.

3.1.1 History

When we inspect education as a whole, it has been so that for hundreds of thousands of years, children were self-educating through play and exploration [29]. When there was no established institution known as school, way back in hunter-gatherer ages, children still had to learn how to gather food and how to hunt animals. There is evidence that children in hunter-gatherer cultures learnt what was expected of them when they became adults through games and exploration.

Early history

The rise of agriculture and later industry made children become forced labourers and made it, so the playfulness in children was seen as unwanted and became a vice [26]. The work of the children now made labourers was time-intensive and without much innovation while on the other hand, the work of hunters and gatherers was intensive in required knowledge, skill and ingenuity, which children could learn through play. There was not much distinguish between play and work in the ages of hunting and gathering. It became nearly separate in the meantime.

As the industry progressed so did education. The need for children labour became obsolete, and compulsory public education now occupied free-time of children. The switch from labour to education made it, so the education was viewed as children's work which further emphasised on the fact that the public saw playfulness and work as two separate things. The view on playfulness and will to exploration was that it was something that stood between the child and education. Inculcation became vital for the understanding of education.

Value of willingness to learn became forgotten. Repetition and memorisation became the new tedious work for children, who instinctively wanted to explore and learn about the world around them in their own pace. Everyone assumed that this playfulness was a barrier between children and knowledge. playfulness was not seen as the vehicle of learning, rather the enemy.

The Works Of The Reverend John Wesley

This notion could be summarised by quote [40], "At five they are all together with the master. Then till seven they breakfast, and walk or work : for as we have no play days, the school being taught every day in the year but Sundays, so neither do we allow any time for play on any day."

Rod enforced discipline to memorise and to drill the given materials. The school became torture for some children. This created toxic environment, which can be described by the quote [26], "A well-known case of school torture is that of the Swabian teacher who, in fifty-one years of teaching, gave 911,527 blows with a rod, 124,010 blows with a cane, 20,989 taps with a ruler, 136,715 blows with the hand, 10,235 blows on the mouth, 7,905 boxes on the ear, and 1,118,800 blows on the head. He made boys kneel on peas 777 times and on a three-cornered piece of wood 613 times, while he made 3,001 students wear the dunce's cap."

John Amos Comenius

It dates back to John Amos Comenius [27], considered as the father of modern education when the view on education took a turn. His very well known quote "Much can be learned in play that will afterwards be of use when the circumstances demand it." can help us understand his view of teaching, education and game's place among it. One of John Amos Comenius very well known works named "Škola hrou" (School by Play) can further demonstrate his view on this topic. He viewed play as a great tool to teach and that it can help in creating a connection between the taught subject and the real world. Since the sixteenth century, much has changed. However, new generations can still learn from John's fundamental principles.

Recent History

"In the 19th and 20th centuries, public schooling gradually evolved toward what we all recognize today as conventional schooling. The methods of discipline became more humane, or at least less corporal. The lessons became secular. The curriculum expanded, as knowledge expanded, to include an ever-growing list of subjects, and the number of hours, days, and years of compulsory schooling increased continuously.

School gradually replaced fieldwork, factory work, and domestic chores as the child's primary job. Just as adults put in their eight-hour day at their place of employment, children today put in their six-hour day at school, plus another hour or more of homework, and often more hours of lessons outside of school.

Over time, children's lives have become increasingly defined and structured by the school curriculum. Children now are almost universally identified by

grades in school, much like adults are identified by their job or career [17]. Therefore it is not surprising that most people still view studying more like a job, than a self-growing development. Therefore numerous people are still wary when it comes to Edutainment, as they do not believe it is natural, to learn through play, as they think the kids would play all day while learning next to nothing. The early part of the Edutainment history, however, proves this to be a false belief, as kids successfully learnt through play for thousands of years.

The first usage of the word Edutainment dates back to 1954. Famous Walt Disney used it to describe the True Life Adventure series. Edutainment has been mostly on television for the first few decades. As one of the most known examples, we can not omit Sesame Street, which first aired in 1969 and Bill Nye the Science Guy, airing in the years 1993-1998.

In recent times, Edutainment moved to games, while also remaining a video form, but moving on to the YouTube platform. Notable YouTube stars, who teach while entertaining their viewers are Veritasium, Crash Course, SmarterEveryDay, AsapScience and NEZkreslená věda. Most of these channels create a little trick on their audience. They grasp the viewer's attention with a shocking title, as "Which Way Is Down?", which is a title for one of the Vsauce videos, and then they start explaining concepts, that relate to the topic of the video's title, rather than answering the title's question directly.

Nowadays, Edutainment is present in various media, from computer games through television to theme parks. Games like Assasins Creed: Origins do an excellent job of implementing teaching modes into their games, therefore not forcing the Edutainment onto someone, as some critiques of Edutainment suggest.

3.1.2 How to achieve Edutainment in games

To understand how to achieve Edutainment in games, we first have to understand its pitfalls, where Edutainment suffers. We will review the critiques of Edutainment first, as it has a fair share of them.

One frequent commentary is that the Edutainment makes it so, that the adults think that sticking their infants in front of a television, laptop or tablet and putting on an Edutainment game or show, makes them magically brighter and better prepared for school [33]. That, of course, does not happen, as Edutainments purpose is to help with standard means of teaching, not replace them altogether, therefore when implementing Edutainment in games, developers should focus on bolstering already existing paths of education, rather than create new ones.

Another problem Edutainment media faces is the lack of meeting the goal

they outlined, often through distractions. As an example of this, Edutainment media, especially TV shows for infants, use unnecessary exaggerations, which often collide with the infants common sense, as outside of the TV show, the toddlers have no chance of seeing such exaggerated things. To draw an example, we will look at the Mickey Mouse Clubhouse(Figure 3.1).

In the show, there is an abundance of magically appearing and disappearing objects, such as the Clubhouse itself or the flying helper, Toodles, who always helps Mickey Mouse find the right tool. These little distractions add up. Kids often take away the wrong message, and sometimes they even want to solve all their life problems by calling Toodles, to come with some magical help [28]. Not only that, but the lesson taken away from seeing a few episodes of Mickey Mouse Clubhouse is undoubtedly not the lesson about the value of patience and hard work, with Toodles disregarding them.

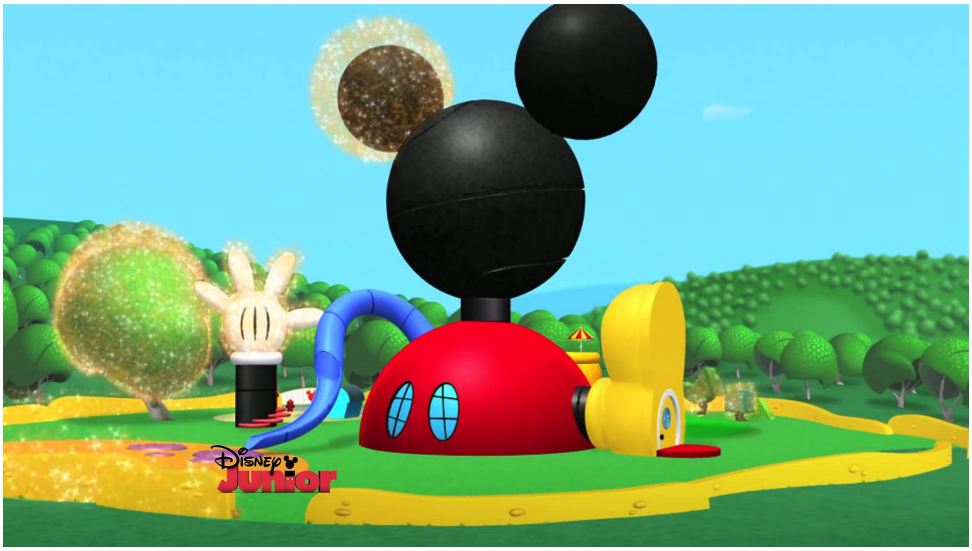

Figure 3.1: Screenshot from the Mickey Mouse Clubhouse - Daisy's Pony Tale episode. Showing, how the house irrationally appears out of nowhere.

Students love to find shortcuts, as when they find a way, to shorten their path to victory, they will often choose to do so, even when it is hindering the teaching effect. In other words, if the students find Alexandrine solution, to a Gordian problem, they will choose it over the intended gameplay route [22]. Therefore such loopholes are even more severe, than in an Entertainment-focused game, as the players might find it fun to find such loopholes, but in Edutainment games, it severely hinders the teaching value, such game holds.

The implementation of the game should also consider using the taught subject, as a game element or a mean to achieve victory, as students will not read through an unnecessary text and the teaching outcomes will hinder. Such example is Chefren's Pyramid from 1997 (Figure 3.2), which let the player read a little about pyramids and then lets him play backgammon, thus netting almost zero Educational value.

Figure 3.2: Illustration of Chefren's Pyramid taken from YouTube channel GameGamer

The last problem we will cover, which Edutainment game developers should consider and avoid, is the lack of immersion. The problem is that games often do not invoke the desired feeling in the player, as they should. For example, when we create a game about a snow fight, we might want to add falling snowflakes and the characters might breathe frozen breath, to create the atmosphere of cold, even though in reality, the frozen breath would not make much sense, as it should not be as cold [34]. For example, Assasin's Creed: Odyssey, does the immersion right, as it lets the player play the game before he can go and explore the world, in an to him already known environment and while at it, he can learn about Egypt through guided tours in Discovery Tour mode [10].

3.2 Advertainment

The word Advertainment is a fusion of the words Advertising and Entertainment. Advertainment refers to a wide variety of topics, from product placement in films to product placement in videogames, where players interact with an object from a particular brand. Advertainment is often confused with Advergames, which focuses on creating a game, with the sole purpose of advertising a brand or a product through video game means.

A study done by Professor Michelle Nelson [25] at the University of Wisconsin examined players thoughts of product placement in computer games. The source of information was a blog called Slashdot. The study examined gamers' belief about the topic, as well as the effect it had on them. The realism added to the games via Advertainment was reasonably well-received by posters, who were not against advertising, opposed to the posters, who were cynical about product placement, and viewed advertising in general as a negative. One of the instances of a brand influencing gamers' pur-

chases was Red Bull product placement in the game Wipeout (Figure 3.3). Where some of the posters stated: "I thought Red Bull was this badass futuristic drink (which I could never find at the time)," and another recalls, "my first exposure to Red Bull was while playing Wipeout XL on the Playstation, almost 2 years before I ever saw the product on store shelves. I freaked when I realized that it was a real drink and immediately picked some up (good stuff!)" [25].

Figure 3.3: Screenshot of the ad in Wipeout game.

Although a not negligible part of gaming community condemn Advergames as dull and often lacking proper in-game controls and a plot, some of them have found relative success, for being goofy or showing the brand odd or bizarre light [3]. Examples of this are Chex Quest, Sneak King and Pepsiman.

And one of the most controversial Advergame is Sneak King (Figure 3.4), an Xbox 360 game released by Burger King in 2006, which bundled it with value meals. The player plays as a former Burger King mascot, The King. The player's task is to sneak behind and deliver food to hungry, wandering people. One of the core game elements, the vision cone in front of the wandering people is highly resembling the Metal Gear Solid's vision cone, for which the game is famous.

Figure 3.4: Screenshot of the game Sneak King. The King hiding in the trashcan, waiting for someone to go around.

One of the most well-accepted Advergames is Chex Quest (Figure 3.5). Although released in 1996, its gameplay and graphics were on-par with the standards, mostly because the Chex Quest is a heavily Doom-inspired game, resembling it in most of its aspects. Despite this fact, the game did have

many fans. The game also lacked all the cruelty, which was ever-present in Doom.

Figure 3.5: Screenshot of the game Chex Quest.

And lastly, Pepsiman is a game, released in 1996, where the player plays as a Pepsiman (Figure 3.6). This game made its name as the most over-saturated Advergame ever made. Despite its high saturation of Pepsi logos, the game is remarkably resembling the popular video game Temple Run in terms of mechanics.

Figure 3.6: Screenshot of the Pepsiman game.

3.3 Other Sub-genres

Apart from Edutainment and Advertainment, there are other notable sub-genres of serious games. From these, we will mention political games and games-for-change. While the political games and games-for-change work in a very similar way, we will discuss them together.

With the politics being very controversial, it is only natural for the political games to follow. Although the video games recently became more obviously political [19], it does not make them political games. Political games aim to influence the player for their political profit, or to raise the awareness about a specific political topic, mostly they intend to change a players point of view or to motivate the player for a particular type of behaviour. The US Army has created a prime example of political games, America's Army series.

America's Army (Figure 3.7), is an FPS-type series of games, where the main

focus is on the realism of the used weapons as well as the emphasis on the teamwork. The purpose of America's Army series is to bolster the recruitment for the real US army. The games are the first genuinely political games, as they date back to 2002, with the first game America's Army: Recon. Other examples are Oiligarchy and the Political Machine series.

Figure 3.7: Screenshot of the America's Army: Proving Grounds from the official web site

The Games-for-Change branch of serious games also aims to influence the player' point of view, but rather than for self-profit, the games aim to raise awareness, bring up taboo or controversial topics and to start discussions about those topics. The name games-for-change also refers to a non-profit organisation, which facilitates the creation and distribution of social impact games. The games-for-change organisation yearly hosts G4C festival [15], where the organisation awards various games that fall under the games-for-change branch. Some of the examples of these games are Oiligarchy or Sweatshop [23].

3.4 Gamification

Gamification is the term for the use of game-like elements in a non-game environment. It often manifests as scoreboards, points, badges, progress bars etc. The primary purpose of gamification is to bring motivation to otherwise to some unmotivating tasks like studying, doing fitness or playing an instrument. It also helps reinforce specific behaviour or increases the level of loyalty of customers.

3.4.1 Differences

While both gamification and Serious Games can have the same goal, they have different means. While gamification is the usage of game elements in a non-game environment, Serious Games are games with a purpose other than to entertain.

Serious Games often focus on achieving real-world goals for the user. From teaching him, through making him more familiar with a particular or propagating political view to him to presenting him a world problem.

Unlike Serious Games, gamification is more versatile. It motivates the user to become better at a particular skill through challenges and competition. It reinforces certain behaviours towards a product through rewards. It unveils the progress the user has made via achievements, action tracking and points.

3.4.2 Misuse

Gamification is often seen as a double-edged sword. On the one hand, it can bring motivation, that would otherwise be very difficult to achieve. But on the other hand, it also can hurt the goal of the primary application and steer the user's attention away from it [5].

Example of such misuse can be adding a timer into Teaching Tool. The timer would put pressure on the learner and hence give motivation. But the learner might focus on the timer so much that the learnt subject would become secondary.

There are other difficulties when implementing gamification, such as failure to pinpoint the goal of it. There are two main approaches towards gamification for online tools, collaboration and competition. Developers often just put leaderboards into any product they have, which can result in a counterproductive environment, when the product's focus is on collaboration. According to Kapp [41], an instructional technology professor at Bloomsburg University and author of several books on gamification, the developer should try not to create an environment, where employees or users try to best each other, but to best themselves. He also suggests that people in time will cease to use the word gamification, but its' core values will remain [12].

3.5 Examples

Before laying out the implementation itself, we will present four different serious games to draw an example of various ways to implement them. We have chosen Duolingo, Yousician, Phantomation and Oiligarchy. The first two

represent commercially successful Edutainment titles, while the Phantomation is a representation of a low-budget Edutainment title, with relatively engaging gameplay. As a fourth, we have chosen Oiligarchy, to represent non-Edutainment serious games, as it represents both political games and games-for-change.

3.5.1 Duolingo

Duolingo is an online application for teaching languages, offering 95 different language courses in 38 languages. Students can use both an online website and a mobile app to access it.

Duolingo shows lessons in a tree-like structure, see (Figure 3.8), through which the student progresses. Usually, advised number of lessons a day is 2-3, but people often undergo more. Completed lessons award 10 XP, which then translates to levels.

Figure 3.8: Duolingo's Tree-like structure of lessons.

One of the things that makes the app fun is a little mascot called Duo. He is a male green owl depicting knowledge, wisdom and learning. He serves as a coach and a tutor throughout the course, motivating the users to higher performance.

Another design that helps the users stay motivated is the introduction of an in-game shop. Users obtain lingots throughout the course for the accomplishment of various achievements. Lingots are a currency used in the shop. In the shop, users can use the said lingots to freeze their streaks, buy Power-ups or different bonus skills.

Duolingo's business model follows the *freemium* design. All basic the courses are free to attend, but for some users have to pay in-game currency, obtained throughout the classes, or bought with real money. There are two versions of accounts, free and pro. The difference is that free users have to attend all the lessons online, and have to watch numerous advertisements in between the lessons. Users can not only subscribe to the pro version, but they can also buy different outfits for the mascot Duo, to personalise their experience with the app.

Early Duolingo

Duolingo started as an academic project of consulting professor Luis von Ahn and his graduate student Severin Hacker at Carnegie Mellon University. Louis von Ahn wanted to create a two-purposes-in-one program called "twofer", with the purpose to teach the students while having them translate simple phrases in documents.

Despite its huge success, numerous critiques arose. Mainly about the fact that Duolingo is not capable of teaching beyond the intermediate level, although the CEO Louis von Ahn only promises, that the course will only get the students to an advanced beginner or early intermediate level, depending on their commitment.

3.5.2 Yousician

Yousician is an online Edutainment game, teaching guitar, piano, ukulele, bass and singing. It uses audio signal processing to determine the notes and chords played by the user's instrument, which then provides instant feedback. App offers guidance through different songs or premade courses, crafted for slow progression over different techniques.

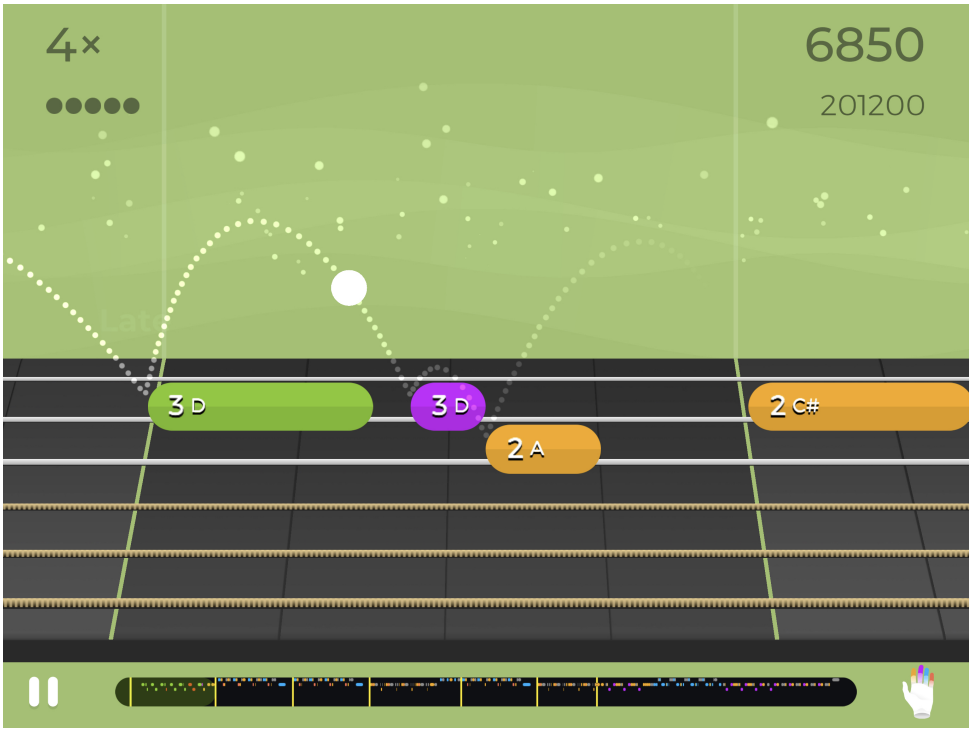

Figure 3.9: Screenshot of the Yousician gameplay.

When playing a song, the user interacts with floating bars, that represent different notes in tablature view (Figure 3.9). Each note has a different colour, based on which finger should press the given thread. Notes also have numbers written on them, and optionally even their name.

Yousician is a prime example of creating an immersive Serious Game with ties to the real-world. The ingenious usage of the instrument as a tool for playing the game on a tablet, mobile or in the web browser made the Yousicians very distinctive. It even became various music teachers tool of choice.

Lawsuit

Ubisoft sued Yousician for infringement of their patent no. 9,839,852 [21]. The patent is for an "interactive guitar game". The patent describes different ways of aiding the learning process via visual clues like notes and chords. It also describes various mini-games and details feedback, to help the player learn to play the guitar. The district court concluded, that the patent was direct toward an abstract idea, nullified the patent for being too broad and dismissed the suit with prejudice, preventing Ubisoft from making another lawsuit on the matter.

As well as Duolingo, Yousician is also a *freemium* type app. Offering unlimited playtime and access to more popular songs for a monthly or yearly subscription fee. Yousician also self-claims the title of the largest music educator.

3.5.3 Phantomation

Phantomation [16] is a Serious Game with the intent of teaching the players to use animation tools in the animation program Play Sketch. GAMBIT Game Lab, the developer of Phantomation and Play Sketch animation program, never finished the animation program. The tool was supposed to aid the user in creating quick, simple animations and had a feature called real-time, which allowed the user to create animation demonstratively, without keyframing.

Figure 3.10: Phantomation game screenshot.

The player gradually progresses through three different game parts, cemetery, village and castle. The Cemetery part is a tutorial that teaches different aspects of animation, namely translation, rotation and scale, and how to use them in the tool via keyframing and real-time animation. In the Village part of the game, the player tackles obstacles in mini-levels. They provide a bridge between the play and the tutorial part. They show how these aspects of animation affect the game environment.

Then in the Caste part, there are sixteen unique levels, where the player plays as a phantom trying to save someone in an abandoned mansion from the death, that Evil spirits have prepared for this unwelcomed visitor. The game plays in a way that the player scares away the living and by that guide them safely through the mansion, evading the spirits lurking behind dangerous doors. In the levels, it is entirely up to the player, if he prefers keyframe or real-time animation. With each level, the difficulty rises, testing the player's animation skills and his ability to think on his feet.

MIT-Singapore Gambit game lab

It was a collaboration between the Massachusetts Institute of Technology and the government of Singapore that lead to the foundation of the Singapore-MIT Gambit game Lab. The purpose of the collaboration was to explore new directions for the development of games as a medium. The GAMBIT's mission was to draw students, developers and technologists together to teach, conduct research and to develop new paths to apply the game design. Although very productive, and having world-class research, most dedicated team and the best-trained students, the Singapore-MIT GAMBIT Game lab ended and took a shift into a new role as the MIT Game Lab.

3.5.4 Oiligarchy

Oiligarchy is a Serious Game that through its plot, criticises the current oil situation around the globe. Although its release in 2008, it still comments on a current topic. Oiligarchy not only comments on the world economy's dependence on oil, but it also criticised oil companies for interfering with elections.

The player is playing as CEO of the oil company. The gameplay consists of searching for new oil reservoirs and creating new wells on them to supply the demand, as can (Figure 3.11). When the supplies in the home country, Texas, are not enough to meet the demand of the society, the player has to expand to Alaska, where the environmentalists create tension, but the player can negate this by financially supporting the political parties. Sometimes it is better to support both, the Democrats and the Republicans. When all the domestic resources start to decline, the player has to expand into foreign countries. Otherwise, he faces the threat of being fired. The overseas operations often

require political and military support from the government, which the player obtains through financially supporting the winning president.

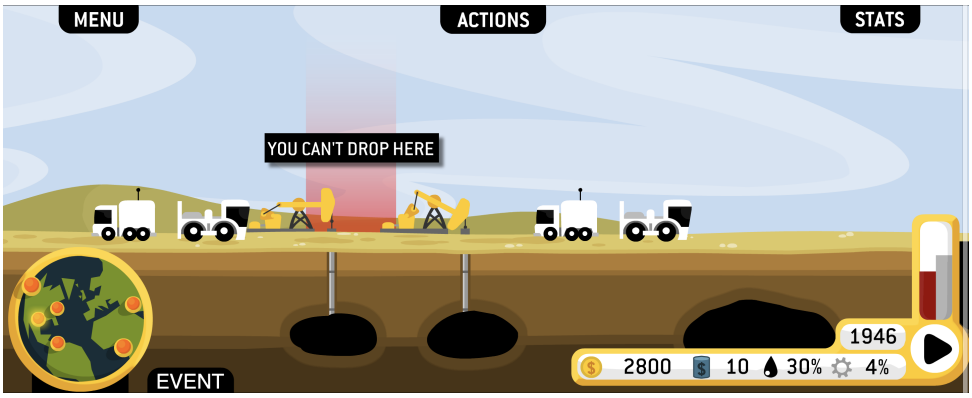

Figure 3.11: Screnshot of the game Oiligarchy.

The game is a free online game, whose sole purpose is to spread awareness on the oil industry practices. The Molleindustria claims their games are varied, from satirical business simulations to reflection on labour and alienation of society. From playable theories to politically incorrect pseudo-games. Molleindustria started producing these games since 2003, intending to challenge the vision, that games should become resemblance to reality. It seeks to show that games can make an impact, and be artsy even with abstraction from the real-world.

Ian Bogost's critique

Bogost, at his talk at the 2013 Games for Change conference in New York City, he criticised Molleindustria's Oiligarchy for failing in delivery of its message without a postmortem [2]. He suggested that Serious Games should be more self-sufficient in their goal. He draws a question, that when a game like Oiligarchy needs postmortem to be fully transparent and deliver its message to a casual player, what is the reason for creating such a game instead of just using the supposedly outmoded text and images medium.

Chapter 4

Implementation of Teaching Tools for game design course

4.1 Covered Topics

Before we describe the tool, we will describe the covered topics. These being physics, transformations, animations and artificial intelligence. We have to state, that the descriptions include more than the tool covers, as the tool was created with the premise, that the student will read through the descriptions of the covered topics to deepen their knowledge of the game design further.

4.1.1 Artificial Intelligence

Another vital part of video games covered by the tool is Artificial Intelligence. From path-finding to self-teaching agent, AI covers a wide variety of different tasks [24]. We will layout two different approaches to implementing Non-Player Character's (NPC) control.

The most basic and also prevalent implementation of AI for any NPC is a decision tree. While having many downsides, it is the easiest way to implement in-game behaviour. It is a flow-chart like structure with Internal nodes representing tests of attributes, branches depicting the outcomes of these tests and leaf nodes represent a class label, the decision made based on all the tested attributes.

Building decision trees often involves a top-down approach, recursively splitting the tree, based on different attributes. Over-fitting is a state where the tree gets too complicated and too specific, which can be solved by pruning. Pruning is a technique, where developer omits unnecessary or too specific sub-nodes, to tackle the trees over-complexity and to make it more readable.

Although more demanding, behaviour trees can create far more complex NPC behaviour, than decision trees. They became popular after Halo 2 used them in 2005 (Figure 4.1) to tackled very complex behaviour of NPC [20]. Despite, the behaviour trees consisting of more than four different node types, the tool will cover only the fundamental four, Sequence, Selector, Inverter, and Action node.

Figure 4.1: Screnshot of the Halo 2 Game.

Sequence node has multiple child nodes and executes them in the given order until one of them returns *failure*. If any child returns *running* or *failure*, the Sequence also returns *running* or *failure*. Otherwise, it evaluates as *success*.

Selector node has multiple child nodes and executes them in the given order until one of them succeeds. If any of them returns *running* or *success*, the Selector also returns *running* or *success*. Otherwise, it evaluates as *failure*.

Inverter Node has a single child node. When the child node evaluates, the Inverter returns *running* for *running*, *failure* for *success* and *success* for *failure*.

Action node is the backbone of every behaviour tree. A different approach towards seeing behaviour trees is seeing them as a structure that chooses, which Action nodes evaluate. Action node has no child node, hence is often referred to as Leaf node. This node executes a given action, based on the circumstances, and evaluates either as *failure*, *success* or as *running*. As an example of this action, we can take a hypothetical node, which we will call the Walk node. The Walk node tries to move a character forward, and if nothing

obstructs the path, it starts the animation of walking and evaluates itself as *running*. If the animation is successfully over, it evaluates itself as *success*. If any circumstance during or before the animation makes the movement impossible, the Walk node evaluates as *failure*.

Other notable nodes, not covered by the tool are Succeeder, Repeater, Repeat Until Fail node and random Selector/Sequence.

Succeeder is a node that has a single child node. After the execution of its child node, it always evaluates as *success*. Succeeder is useful for cases, where the developer wants a branch to execute, despite the fact, that it is very probable, that it will evaluate as *failure*.

Repeater is a node that has a single child node. After execution of its child node, it executes again. Optionally Repeaters can have a set amount of repetitions before they evaluate themselves. Repeaters are often root nodes of a tree so that it would run continuously.

Repeat-until-fail node also continuously reprocesses its only child, with the slight adjustment, that instead of a set amount of repetitions or neverending cycle, Repeat-until-fail node evaluates after the first *failure* of its child.

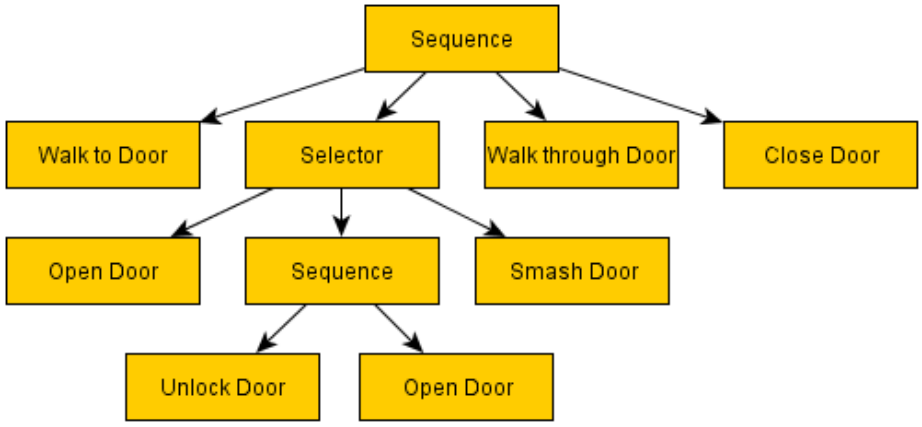

Figure 4.2: Behaviour tree example of AI resolving its way through a possibly locked door. Picture by Gamasutra

To demonstrate how elementary behaviour trees work, we have selected (Figure 4.2). With the evaluation of the tree, NPC will decide, whether to open, unlock and then open, or smash the door.

Random Selector/Sequence nodes work identically to their namesakes, except it evaluates in a random order rather than a given one.

4.1.2 Transformations

The Transformations in games are vital for 3D and 2D game to exist. Without them, the camera would not be able to move, the world would remain motionless, and the game objects would not be able to change via scaling, skewing, reflection and more.

In 3D applications, 4x4 matrices represent various transformations (Table 4.1), with points represented as 1x4 vectors with one as the fourth coordinate. The vectors are represented as 1x4 vectors as well, with the difference of their fourth coordinate being zero. The reason behind these representations is that there is no representation for translation as a 3x3 matrix, as it is an affine transformation [14].

X-rotation	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Y-rotation	$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Z-rotation	$\begin{bmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Scale	$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Translation	$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 4.1: Different 3D transformations represented as 4x4 matrices.

Points have one as their fourth coordinate, so the translation would apply to them, while the vectors have zero, so it would not change their magnitude and direction, as translation should not affect vectors.

Another representation for transformations is representing them as quaternion [36]. Unity represents all of its rotations as a quaternion, as it does not suffer from gimbal lock, they are more compact, and their interpolation is elementary. Quaternions extend complex numbers. Therefore they are not intuitive, and students often have problems getting the grasp of them.

In Unity, physics does most of the translations and rotations. However, some objects need to move via a script, for example, doors, which open, a ghost passing through windows, and more. For this use, the ideal option of implementation in Unity is the isKinematic [37] attribute. The *isKinematic*, however, will still affect other *RigidBodies*, so for a ghost, this would mean he would knock over everything, he would go through. Therefore omitting *RigidBody* from the ghosts is the next logical step. Yet this step leads to another problem. Whenever a *GameObject* without *RigidBody* enters a trigger, it does not register the collision via *OnCollisionEnter* [35] method, as at least one of the colliding colliders need to have non-kinematic *RigidBody* attached. To handle this issue, developers can use scripts to trigger the trigger manually.

4.1.3 Physics

Physics in games are the laws of physics applied to the simulation or game engine. The purpose is that the game would resemble the real-world, but this is often not the case, for example, some games employ silly ragdoll physics as their main selling point, like Human: Fall Flat as seen on (Figure 4.3), downloadable from Steam. Ragdoll physics refer to physics, in which the joint and skeletal muscle stiffness in characters is either zero or very low in value. The lack of stiffness makes the character collapse much like a rag doll toy. The field of physics is vast, and many topics hide behind this term, but an accurate description of what most game call physics is *Rigid Body Dynamics* simulation.

Figure 4.3: Screnshot of the Human: Fall Flat

Rigid body refers to an idealised solid object, which is infinitely hard and non-deformable, which translates to the fact, that any distance between two points on the object is constant. *Rigid body* is also often regarded as

homogenous in its weight distribution, which is not valid for some game engines.

Dynamics is the process of resolving the influence of forces on the movement and interaction of *rigid bodies*. Simulation of these dynamics gives the objects in the game freedom to move in a genuinely interactive and naturally chaotic manner. Animation clips are less eligible for this task, while the simulation can create unique experiences for every playthrough.

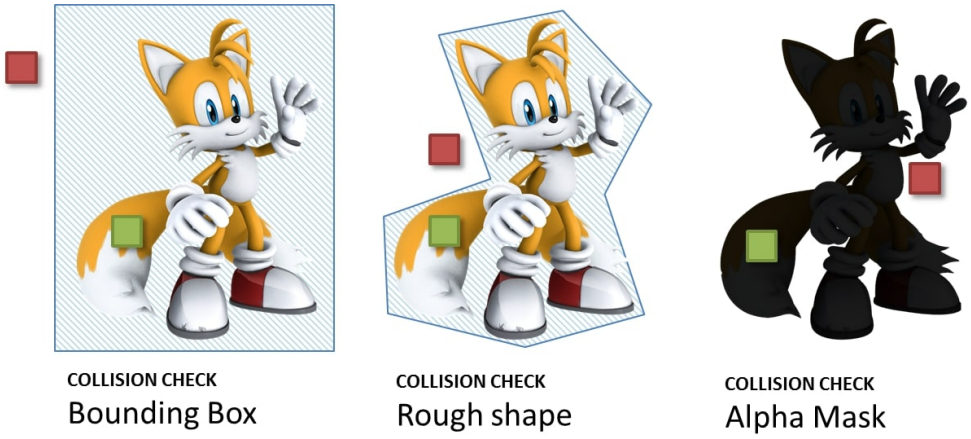

Figure 4.4: Example of collision detection. The red squares represent object resolving as negative, in the current step, where the green one resolves as positive.

For accurate simulations of various physical behaviour, like a ball bouncing on a floor, ice cube sliding down the hill, rolling of a ball or a car coming to a rest, a *rigid body dynamics* simulation is used in tandem with a *collision detection system*. However, accurately computing collisions is a demanding task. Therefore collisions are resolved via chaining in *collision detection strategies*. To demonstrate how this works, we will draw an example in 2D (Figure 4.4). The first strategy is to resolve all the collisions via *bounding boxes* (Figure 4.5), and all the positive collisions continue down the chain. Next step is to describe the shape via polygons and repeat the collision check. The chain then continues with positive detections to the alpha mask check, which is the last step. Any collision that tested negatively in any step resolves as not colliding.

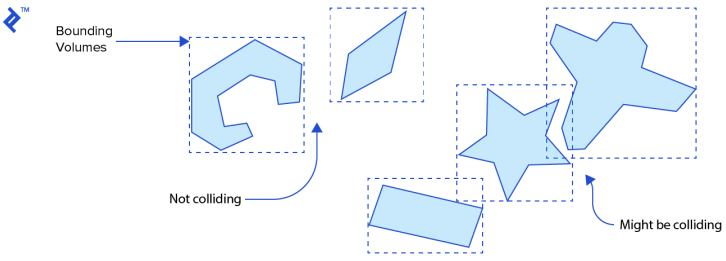

Figure 4.5: Example of resolution of bounding boxes collisions.Picture by Nilson Suoto.

Collision detection system is one of the backbones of any game engines. It is the component responsible for the objects not passing through one another or the environment. Many games do not implement a "physics system", *collision*

detection system can be used without it. All games have to implement *collision detection system*, if they involve a movement of objects and their interaction, in both 3D and 2D space.

Game physics are not limited to only detecting collisions of game objects with other game objects or world geometry and applying forces to them. It also simulates spring-mass systems, destructible buildings and structures, traps such as avalanches, cloths, hair, water surface, upthrust and the list goes on.

Although most simulations are run-time based, complementary offline pre-processing simulation tools help alleviate some of the computing difficulty. Examples of these tools are Maya, Blender and Cinema 4D.

4.1.4 Animation

Animation in video games is a crucial element of the game engine when it comes to making life-like characters and environment. There are many ways how to achieve animation, like 2D and 3D digital animation, *stop-motion*, *puppetry*, *claymation*, and many more.

Figure 4.6: Screnshot of the Neverhood game taken by MobyGames.

The animation is the creation of an illusion of movement. The illusion is created by displaying images in rapid succession, thus resembling the movement. Unlike movies, video games are interactive, and unpredictableness of their environment makes the use of traditional animation techniques challenging [31]. Therefore most video games use 2D and 3D digital animation,

but some games use unusual methods, which makes them unique, like the Neverhood from DreamWorks Interactive, which used *claymation* as seen on (Figure 4.6).

With the games having to deal with different situations, animations often become part of gameplay, like reloading a gun in FPS games or auto-attack animation in MOBA games.

The fundamental 3D animation technique is a *rigid hierarchical animation*. The animation of in-game objects via their rigid parts with simple transformations [18]. This approach, however, leads to many difficulties, such as artefacts near the joints, where joint-cracking occurs with humanoid characters. The usage of this technique has targeted to the animation of robots and machinery, which genuinely consists of solid parts.

Figure 4.7: Extreme poses of the *morph target animation* for Daz characters from REALLUSION.

To avoid the silliness of the joint-cracking on the humanoid characters, animators use brute-force technique, known as per-vertex animation. The method consists of the animators determining how the vertices move, then exporting the motion data, so the game engine knows, how to move the vertices. However, this technique is data-intensive and hence not suitable for real-time games. There is a variant of this technique, called *morph target animation*, which has some implications in real-time games. In the method, the animator creates a small set of fixed, extreme poses (Figure 4.7), and the engine interpolates between them. The morph target animation technique found its usage in the animation of human facial expressions, for its complex nature.

The most used technique for humanoid character animation is *skinned* animation. *Skinned* animation, also knows as *skeletal* animation, is animation technique, which binds vertexes to skeletons constructed from rigid bones. The first usage of this technique in video games dates back to Super Mario 64. In the *skeletal* animation technique, the bones do not render, and smooth continuous triangle mesh is bound to the joints, meaning the vertices of the mesh will follow the movement of the joints. The bones are animated as in the *rigid hierarchical animation*, which then affect the vertices of the

mesh (Figure 4.8). The mesh, to which the vertices are attached is called skin. The vertices can bind to multiple joints with different weights, to make the skin stretch naturally. The terms bones and joints are interchangeable, but technically speaking, the bones are the empty spaces between the joints, which move. Game engines take only the joints into accounts as they ignore the empty spaces between them. So whenever someone is referring to the bone in the context of *skinned* animation, it most certainly is referring to joint.

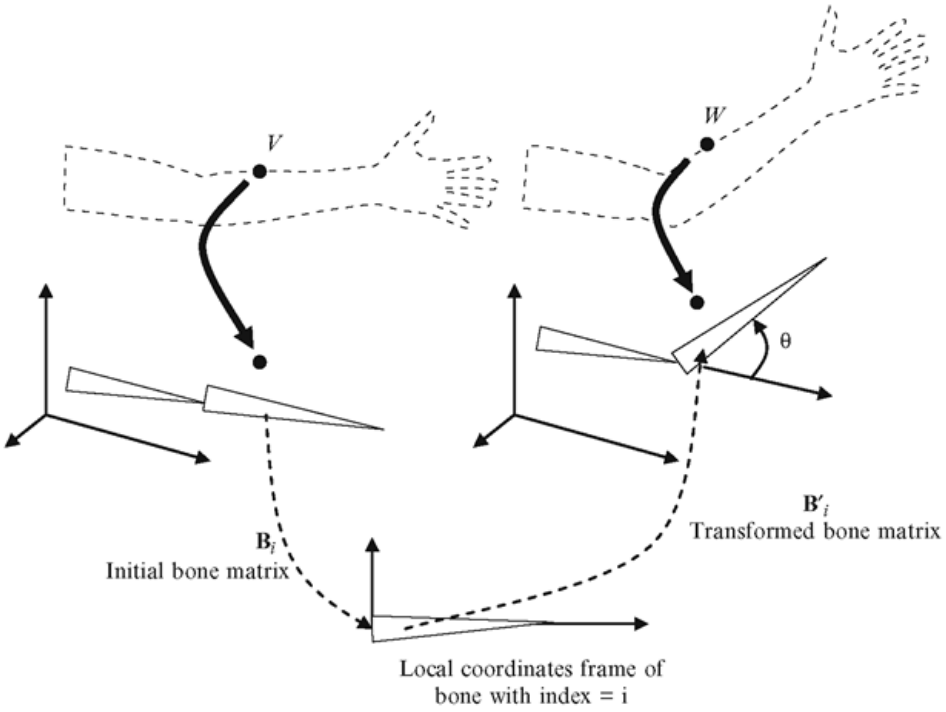

Figure 4.8: Example of the *skeletal* animation by what-when-how.

4.2 Tools description

The teaching tools are aimed at bolstering the education of game development. The tools reflect the earlier described methods and practices, as well as exhibits elements of Edutainment.

Firstly, the framework, upon which the plan for the tools stands, is ICAP. As mentioned in chapter two, the idea behind the design is that the tools would work in a way which helps the student with *Constructive* stage of game-design learning, and will achieve it through experiments and demonstrations, although the transformation and animation part exhibit sigs of *Active* stage as well. Secondly, the tools, use text during the exhibits and experiments, while studies described earlier suggested, that giving the students complicated text before them would yield lesser results. In terms of revised Bloom's Taxonomy, the students will be undergoing fourth, fifth and sixth levels, these being

Analyzing, *Evaluating* and *Creating*. The experiments in the physics part, transformation part and animation part cover the *Analyzing* and *Evaluating* level. The AI part is more about being able to understand the structure of behaviour tree and then creating their own AI, thus covering the *Evaluating* and *Creating* levels. The physics part as an only part contains Discovery Learning.

Tools part	ICAP	Bloom's Taxonomy	Discovery Learning
Physics	*Constructive*	*Evaluating, Analysing*	Yes
Transformations	*Constructive, Active*	*Evaluating, Analysing*	No
AI	*Constructive*	*Evaluating, Creating*	No
Animation	*Constructive, Active*	*Evaluating, Analysing*	No

Table 4.2: Used Methods and Techniques.

The tools exhibit Edutainment elements mostly in AI part of the tools, as the students get to play Pong, and are encouraged to create their own AI so that it would be more competitive. Other elements are in the physics part of the tools, where the student can experiment with different settings and then conduct small experiments with play/pause options. Other two parts of the tools do not exhibit Edutainment elements, as they are mostly instructable.

The assumptions made about the students were that the students would have basic knowledge of Unity, and how it works. The students would be able to understand C# programming language, as it is the language for creating scripts for Unity. Other assumptions were that the students would have intrinsic motivation to use the tools, so the tools do not display any signs of gamification.

The main element to communicate the information to the students is a text window, which is prevalent in every scene of the tools. It instructs the students on controls, as well as tell them basics about the scene's topic.

Other than the text window, each of the scenes, but main menu, contain options menu, which works as a means to control the exhibitions and experiments. It also functions a way to communicate the values used in them.

We chose a minimalistic approach towards usage of assets, as the early game development does not rely on them. This approach, as well as the assumptions and other design choices, will be evaluated in the discussion. With the minimalistic approach, we also chose not to go too deep on any of the topics, and instead to show the students pitfalls they might stumble upon during game development.

The build, as well as the Unity project can be found in (Appendix A).

4.2.1 Artificial Intelligence

Artificial Intelligence part of the tool introduces the student to behaviour trees. The student gets to play Pong, one of the oldest video games ever, made by Allan Alcorn back in 1972 for Atari.

We chose Pong as a model game because it does not have complicated rules, and nearly everybody knows about it. Pong is a two-player game, with two paddles, a ball and an area for the game to be held. Each players goal is to deflect the ball, which travels predictably under a certain angle, with regards to the angle of incidence being equal to the angle of reflection, with his paddle, so the ball does not end behind it. A player scores, when he manages to deflect long enough for the other player to make a mistake and let the ball go past his paddle.

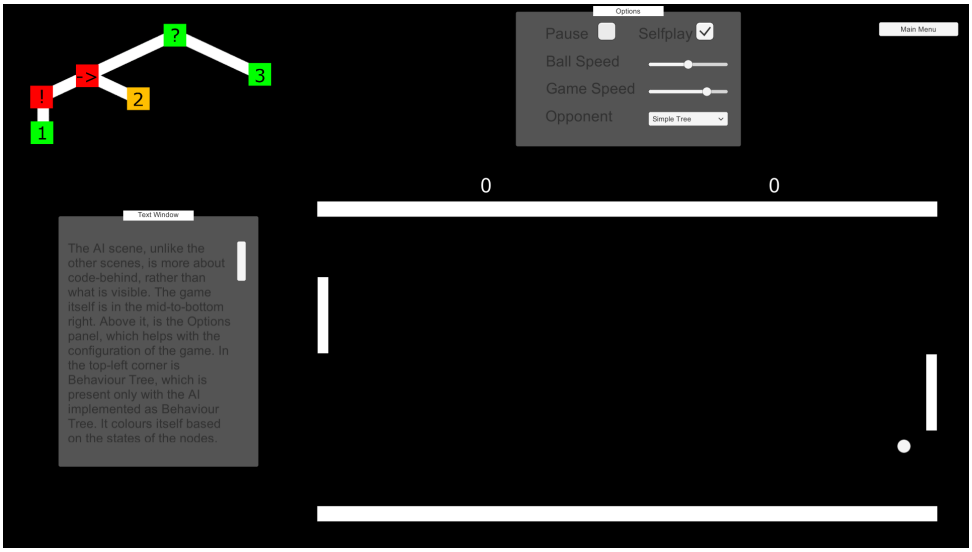

Figure 4.9: Screnshot of Artificial Intelligence part of the Tools

The student plays against an AI, represented by the behaviour tree, as seen on (Figure 4.9), or a simple AI. Student can examine the behaviour tree during the gameplay, through an infographic visualising the tree. The tree structure is rather trivial. It consists of three Action nodes, a Sequence, a Selector and an Inverter. The nodes can never achieve a state of *Running*, while they have been created in a way so that they can be either *Failure* or *Success*. This implementation enables the student to see which nodes, the execution of the tree evaluated and how. The game also supports game pause for this cause. The student controls the paddle on the left, while the AI takes control of the right one. The gameplay itself can seem distracting to the student, so the toll supports a self-play button, which lets the student observe the AI and controls the paddle for him.

The game also has adjustable game-speed as well as ball-speed, for the ease of examination. The student can also choose various AI, which controls the right paddle. Moreover, the student is encouraged to implement his own AI,

which would control the paddle, to enhance his knowledge. The code-behind presents the student with existing classes and already implemented behaviour tree, as well as basic AI if the student chooses not to take the behaviour tree path of implementation. The code used uses the structures taken from "Unity 2017 Game AI Programming - Third Edition: Leverage the power of Artificial Intelligence to program smart entities for your games" [32]. These structures implement four elementary nodes, Sequence, Selector, Inverter and finally Action node.

4.2.2 Animation

The Animation part of the tool shows different approaches to animation blending. There are two rows, each showing different aspects of animation blending, as can be seen in (Figure 4.10). The upper row shows five characters of Xbot, which was with all the animations downloaded from Mixamo.com [1], each with different animator controller, the state machine responsible for blending and mixing of animations. It shows how vital baking into position and rotation is. All of the characters, but the right-most one have rotation and position baked into their animation. The right-most one, however, has only the position baked into the animation. Therefore it rotates ever-so-slightly, which becomes noticeable after a few loops of the jumping animation. The student is encouraged to notice this fact and encouraged to try and remember this fact for future use, for it can create artefacts during the gameplay.

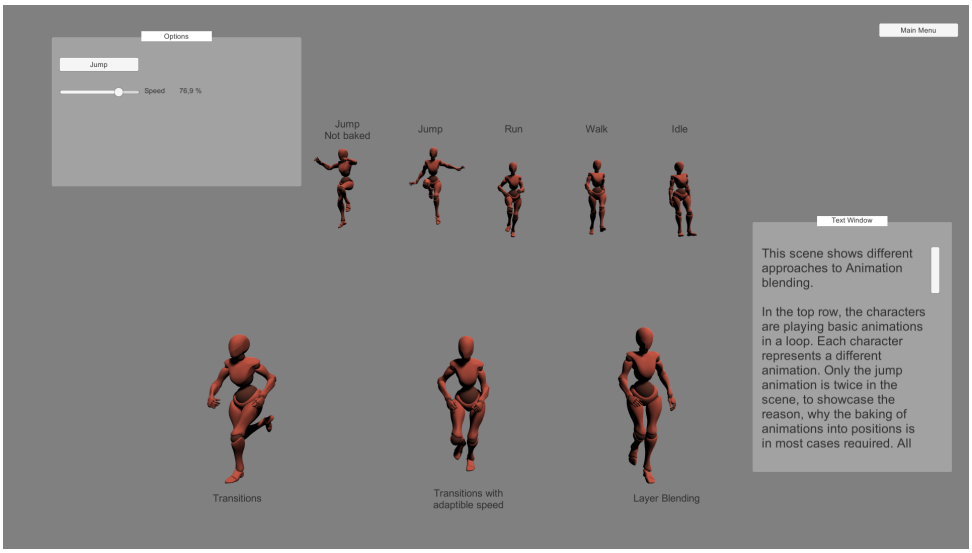

Figure 4.10: Screenshot of Animations part of the Tools

The second row exhibits various animation blending approaches. The left-most character represents the most basic approach, the transitions. The animations in the animator controller for the character have been baked into position and rotation. They play at a constant speed, and the transitions happen without fixed exit time. The only exception is the transition from

the jump state to the others, with the exit time at the end of the animation clip. The first transition is between the *idle* and *walking* state when the condition of value of speed to be higher than zero. The reverse transition has a condition of value of speed being zero. Second, the third and fourth transitions are all the same, as the condition for the transition between any state and the *jump* state, is simply the trigger of a jump button. At the end of the *jump* state, the state always transitions to another one. The condition of the *jump* state applies to all the different animator controllers. The condition for the *idle* state is the speed being equal to zero, for *walking* is the speed value being between, but not equal to zero and zero point seven. For *running* state, the condition is the speed value being higher than zero point seven. The last transition, from the *walking* to the *running* state, is conditioned by the speed value being higher, than zero-point-seven, and the reverse is when this condition is not met.

The second one works the same, with the difference of the *walking* and the *running* animations clips being slowed down based on the speed parameter. This approach should be the ideal one, however in the demonstration, it has been overemphasised, so the transition does not look as smooth, as it should, but this way it helps the student see the difference better.

The third one is the blending of two animation layers, which correspond to each other. The first layer is *idle* to *walking* state, with the *jump* state conditions as mentioned above. The *idle* state transitions to the *walking* state whenever the speed value is non-zero and vice-versa. The second layer is the same, just with the *running* state instead of *walking*. The layers are blent based on the speed value. The lower the speed value, the higher is the impact of the *walking* state and likewise for the *running* state with a higher value of speed.

After the completion of the animation part of the teaching tools, the student should be aware of animation baking and different types of animation blending.

4.2.3 Physics

The Physics part of the Teaching Tools deals with mass, forces and bounciness, which is the ability of a game object to bounce on a surface. The higher the number, the less energy is lost during the impact, with the bounciness of one there is no loss of energy. It has two parts, bounciness demonstration and Newton's cradle.

The bounciness demonstration, (Figure 4.11) shows how the bounciness is unaffected by the mass of the object. It also shows the effect of rounding errors and the lack of implementation of conservation of energy in Unity physics. The student gets to observe five balls with different bounciness, which can change via the options menu, which also shows each respective

bounciness of the balls in the scene. There is also a slider for change of mass.

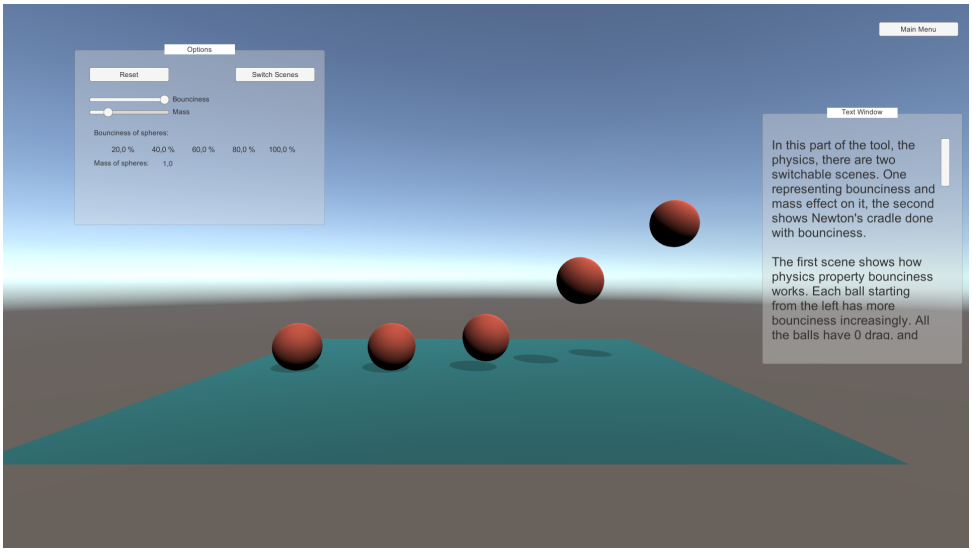

Figure 4.11: Screnshot of Bounciness Demonstration in the physics part of the Tools

The student is also encouraged to observe the ball, which has the maximum possible bounciness of one. The ball jumps higher and higher with each additional bounce, which the text window, prevalent in every scene, points out to the student. The bounciness demonstration shows how the bounciness works for the student to put in contrast with the second scene.

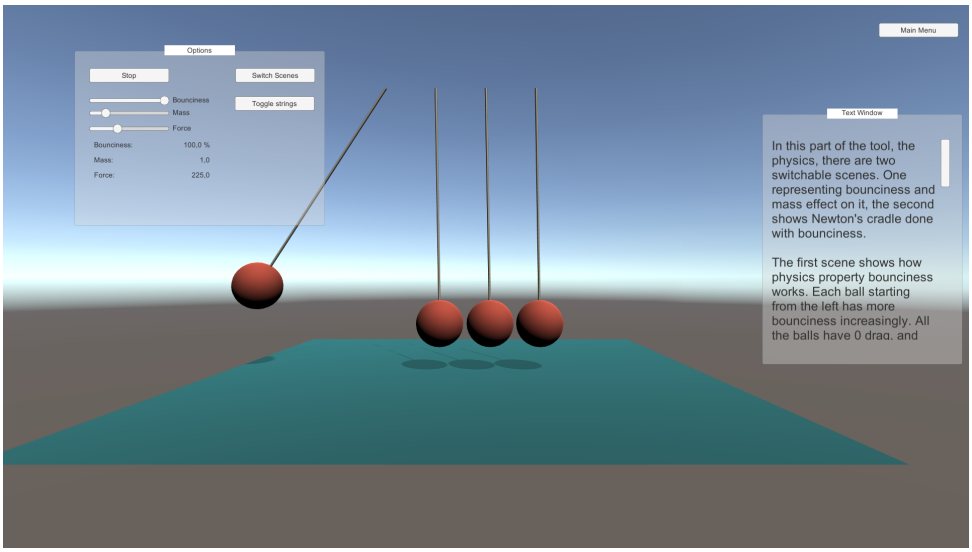

Figure 4.12: Screnshot of Newton's Cradle in Physics part of the Tools

The second scene contains Newton's cradle and shows how the bounciness does not always work the way it does in the real world. The Newton's cradle consists of four balls, which all have the maximum bounciness of one, which is unusual because Newton's cradle's balls are supposed to be metal, which does not have bounciness anywhere close to one. The student is encouraged to try and change the bounciness to observe the different results of the simulation with various settings. The student can toggle the strings, which seem to hold

the balls attached to an anchor above the balls, as can be seen on (Figure 4.12).

There is also an extra added feature of the force and mass, where the student can see that the only difference the mass makes is the effect on the forces applied to the balls. All the various settings apply in real-time so that the student can experiment with the scene in a more creative way, other than just setting up an experiment and then observing the outcomes. The student is also encouraged to reflect on the fact that the physics of the game engine often does not work the way the physics of real-world do and to consider it in their future game-design.

After the completion of physics part of the teaching tools, the student should be aware of the fact, that physics in the game engines contain rounding errors, do not respect the laws of the real-world physics to their whole extent and should consider these facts, when designing a game.

4.2.4 Transformation

The transformation part of the teaching tools teaches about the translation. It shows it in 3D and 2D environment.

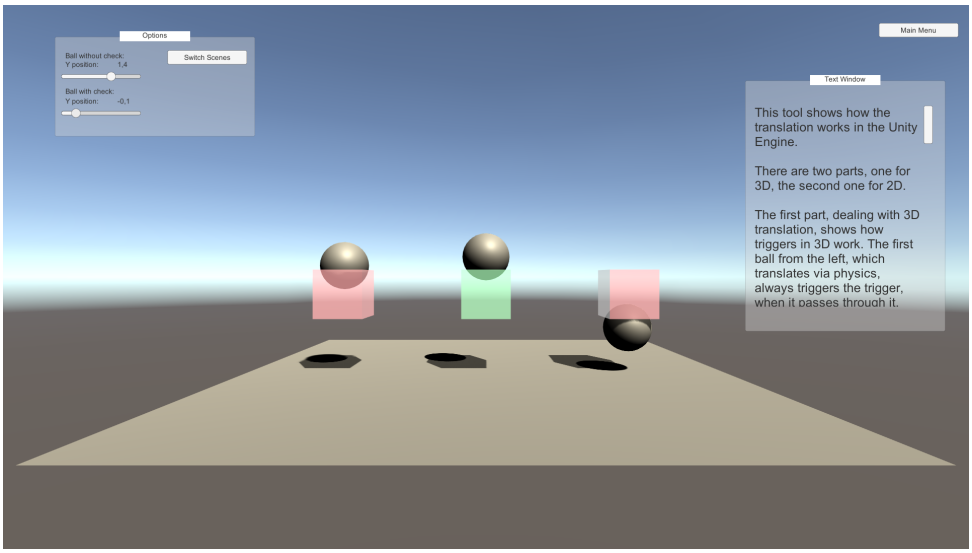

Figure 4.13: Screnshot of 3D part of transformation part of the tools.

In the 3D environment (Figure 4.13), there are three balls and three green triggers, which turn red on the OnCollisionEnter, and back to green on OnCollisionExit. The physics engine controls the movement of the left-most ball, while the other two are code-controlled. The physics controlled ball shows the student, how the trigger triggers, by the ball going through it as expected. Contrary to that, there is the middle ball, which shows how the ball does not trigger the trigger because the *RigidBody* component is not present. The *RigidBody* is omitted from the object, for it not to

create artefacts when interacting with physics-driven objects. The right-most ball also does not have a *RigidBody* component. With each translation, it checks whether it collides with the trigger and if so, it triggers the trigger artificially. The student can change all the positions of the balls separately by using the options window, through the respective sliders. The text also tells the student, to not use *RigidBody* component on an object, that is code-driven via *RigidBody*.transform.position [39], and instead to use the *RigidBody*.MovePosition [38], so it moves there continuously, to lessen the number of artefacts.

The 2D environment (Figure 4.14) demonstrates how the 2D translations are not limited to the two axes, X and Y, but also shows, how the Z-axis affects the rendering of different sprites. The scene contains a ball and five red obstacles, each with different Z position. The option menu sliders control the X and Z position of the ball. The student is encouraged to experiment with different positions of the ball to observe how the rendering is affected. All the obstacles are transparent so that the ball is observable, when behind them. In Unity, the default position of the camera is that it faces the X-Y plane, with Z-axis growing with the view-direction. This placement means that objects with lower Z position overlap objects with higher. The text also reminds the student to use *orthographic* view for 2D games and perspective for 3D, so the objects in 2D would not shrink with incrementing Z position.

Figure 4.14: Screnshot of 2D part of transformation part of the Tools

After the transformation part, the student should be able to understand the translation impact on objects in 2D and 3D, and the implications it brings. The student should also know why to use *RigidBody.position* instead of *GameObject.transformation.position* and the importance of cameras *orthographic* view in the 2D games.

Chapter 5

Evaluation and Discussion

5.1 Performance testing and Hardware Requirements

Hardware testing was required, to ensure, that even with slower hardware, the teaching tools would function optimally and possibly without the compromise of lowering the quality. To test the FPS, we used a script, which is present in the scripts folder. In the prefabs folder, there is prefab for the FPS counter so that anyone can try out their hardware.

OS	**CPU**	**GPU**	**RAM**	**FPS**	**Note**
Windows	AMD Ryzen 9 3900x @3.79 Ghz	GeForce GTX 2080 Super	32 Gb	60	High-end Desktop
Windows	Intel i5 4460 @3.20 Ghz	GeForce GTX 960	8 Gb	60	Mid-end Desktop
Windows	Intel i5 4460 @3.20 Ghz	GeForce GTX 750 Ti	8 Gb	60	Low-end Desktop
Windows	Intel i7-7700HQ @2.80 Ghz	GeForce GTX 1060	16 Gb	120	Mid-end Laptop
Mac Os	Intel i5-7267 @3.1 Ghz	Intel Iris Plus Graphics 650	16 Gb	20	13-inch MB Pro 2017

Table 5.1: Hardware Testing Results, FPS represents minimal FPS during the testing.

As can be seen on (Table 5.1), the tests included five different computers. Three of them were desktops, and two were notebooks. While the different tested PC sets do not cover the whole variety market has to offer, we made presumption, that a game design student would want to have at least GeForce GTX 750 Ti. From the table can be seen that any of the Windows operated computers worked. As a reference display, we used a generic FullHD 60Hz display for the desktops, for Windows notebook we used its FullHD 120Hz display, and for the Macbook Pro its 2560-by-1600 60Hz display.

We presumed that the worst of the desktop GPUs, the GeForce GTX 750 Ti paired with a good CPU would be enough for the ultra settings. Since this was proven, the quality settings options are not required and we even found them distracting, so we omit them from the build.

	Minimum	Recommended
GPU	GeForce GTX 750 Ti	GeForce GTX 960
CPU	Intel i5 4460 @3.20 Ghz	Intel i7-7700HQ @2.80 Ghz
RAM	8 GB	16 GB
Storage	100 Mb	100 Mb
OS	Windows 7	Windows 10
Display	1920 x 1080 60Hz	1920 x 1080 120 Hz

Table 5.2: Minimal Hardware Requirements

Although the GeForce GTX 750 Ti is sufficient, the Macbook Pro's Intel Iris Plus Graphics 650 is not powerful enough for the ultra settings and 2560-by-1600 display. Not only that, but to distribute the software for macOS users, we would either have to be proved software developer, so the macOS would not reject to open the app, or we would have to instruct every macOS user, how to open the application. Therefore the application's support for macOS is nonexistent, and there is no build for it.

When testing on macOS, we also found out that the non-full HD resolution displaces the UI. Therefore we require full HD monitor for the usage of the tools as seen on (Table 5.2).

5.2 User Test

For user testing, we created a questionnaire on Google Forms, containing 31 questions regarding the quality of the tools, and whether the participants would recommend the tools for usage in class, the answers can be found in (Appendix A).

Out of ten participants, only seven were fit to fill the questionnaire, as the remaining three did not want to finish the task without oral guidance. This lack of motivation for the three, lead us to the conclusion that the presumption about intrinsic motivation might have been too optimistic.

When reviewing the answers to our questionnaire, we found out that participants who filled the answers on their phones made many typos. Therefore whenever we quote a reply from the survey, it is not the original text, but a version with fixed typos. The attachment, however, contains the original versions.

The questionnaire proved to be useful, as we adjusted UI a little, based on the feedback given. The survey, for example, uncovered design flaw of wrongly placed exit button. One of the answers to the question "If so, what did you find confusing about the Tool?" : "Quit button under the main menu button. I clicked on it on accident and had to restart the application", showed us,

that locating the exit button in the same place as the main menu button, which is clicked repeatedly in the tools, can cause stress onto the student. The feedback made us move the exit button.

The feedback also confirmed the presumption, that the students who have already some experience with Unity and understand basic terminology would benefit from the tools the most. Some of the respondents told us that the tools used were very instructable, as it taught things, they have googled for hours, while they designed games. Critique part of the questionnaire, however, did not cover these comments fully. Thus they are not recorded, as they added them as a commentary to the survey in oral form.

Question	Yes	No
Do you have any experience with Game Design?	4	3
Did you find the tool confusing?	2	5
Was the usage of the Text Window clear?	7	0
Was the AI part of the tool useful or interesting to you?	7	0
Did you learn anything new from the AI part?	5	2
Did you try to implement your own AI?	3	4
Have you coded before?	4	3
Was the physics part useful or interesting to you?	7	0
Did you learn anything new from the Physics part?	6	1
Was the Animation part useful or interesting to you?	7	0
Did you learn anything new from the Animation part?	5	2
Was the Transformation part useful or interesting for you?	6	1
Did the Transformation part teach you anything new?	4	3

Table 5.3: Yes/No Questions and their answers

As some of the questions were Yes/No questions, we have captured them in a (Table 5.3). Other than Yes/No questions, the questionnaire also included linear scale questions, to evaluate the different parts of the tools and the tools as a whole, we have captured them in a (Table 5.4), with one representing the maximum satisfaction and five being the minimum.

Part of the Tool	1	2	3	4	5
Animation part	4	1	2	0	0
Physics Part	4	3	0	0	0
AI part	5	2	0	0	0
Transformation part	3	3	0	1	0
Tools as a whole	2	5	0	0	0

Table 5.4: Linear Scale Questions and their Answers

As can be seen in the presented tables, the teaching tools as a whole, have been well received. As for the particular parts, the AI part has been the best received, while transformations scored even four on one of the answers.

The respondent, who evaluated the transformations part with four, has never coded before, had no experience with game design and had problems understanding some of the technical terms. Complementary to that, the respondents, which scored lower marks, had a lower chance of having experience with game design and coding.

As can be seen in (Table 5.3), the respondents liked some parts of the tools, even when it did not teach them anything. This notion contradicts the assumption that the motivation would be intrinsic, rather than external.

5.3 Possible Improvements

Possible improvements for the teaching tools derive from user test, performance tests and the unimplemented things, as they were time-consuming.

From user test, we found numerous things, which the tools can improve on. For example, some of the respondents suggested that the teaching tools could show the usage of the topic in games, so the student can more closely relate to the subject, rather than just learning about it abstractly. To tackle this, we suggest implementing a few exhibits from AAA games for each covered topic. As an example, we could show how animation transitions affect autoattacking in MOBA games, as they are sometimes uncancelable.

Others suggested that the sliders used were small. As a possible solution for this, we recommend yet another test, to see which UI is most comfortable for the average student of game design, because we believe that only enlarging the sliders would not be enough.

Another way to help improve the teaching tools would be to optimize the build so that it can be easily accessible on macOS. This improvement includes the optimization of workload put on the computer, as well as a way to make the macOS not gatekeep the application for not having identified developer. And another technical improvement could be having the UI scale with different resolutions, as it only supports full HD now.

Lastly, we suggest an improvement in the transformation scene could be made by showing the importance of 4x4 matrices and Quaternions, rather than only mentioning them in the text.

Chapter 6

Conclusion

In the first part of the thesis, we described different approaches to the implementation of education and shown how the internet affected teaching. Then, we have described serious games, and their impact on the players, we drew examples of them on games like Phantomation, Oiligarchy and Duolingo. We explained what Edutainment means and discussed its positives and negatives. Then we have discussed different approaches to creating such a game or simulations and planned out the implementation of the teaching tools. After that, we implemented the teaching tools in Unity.

In the teaching tools, we describe to the student the four fundamental elements of game design. In the physics part, the student gets to know bounciness and how game-engine physics do not mimic the real-world perfectly. In the animation part, we introduce the student to the basics of layer-blending and transitions in Unity, done through the animator controller. In the AI part, we let the student see how to implement behaviour trees and guide him through the process of such implementation. In the transformation part, we show how triggers require *RigidBody* component on the colliding *GameObject* and how the rendering in 2D is affected by Z-position.

Users have then tested the created teaching tools and gave feedback, which we laid out in graphs and compared it to our presumptions made during the planning of implementation.

We have tested the teaching tools on five different setups to ensure a smooth experience for an average game design student, and we laid out requirements for such a seamless experience.

Future improvements on the teaching tools should focus on transferring the knowledge better, with examples from popular games like Fortnite or League of Legends so that the students can see better how to implement the game-design elements into games, rather than only learn about abstract ideas.

Bibliography

[1] Adobe Systems Incorporated, *Mixamo*. Online Tool, 2020.

[2] L. Alexander, *Bogost: Let's make 'earnest' games, not 'serious games'*. Gamasutra Blog, June 2013.

[3] J. Amirhkani, *10 Company Branded Video Games That Didn't Suck*. Complex Blog, 2013.

[4] Anderson, L.W. and D. Krathwohl, *A Taxonomy for Learning, Teaching and Assessing: a Revision of Bloom's Taxonomy of Educational Objectives*, Longman, New York., 2001.

[5] T. Barber, *Why gamification doesn't mean serious games*. Dashe & Thompson blog, 2019.

[6] M. T. Chi and R. Wylie, *The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes*, EDUCATIONAL PSYCHOLOGIST, 4 (2014), pp. 219–243.

[7] A. Churches, *Bloom's digital taxonomy*. Edorigami Wikispaces, 2008.

[8] T. De Jong and W. R. Van Joolingen, *Scientific discovery learning with computer simulations of conceptual domains.*, Review of Educational Research, 2 (1998), pp. 179–201.

[9] H. Dingman, *Assassin's Creed: Origins's Discovery Tour mode shows how great educational games could be*. PCWorld Blog, February 2018.

[10] ——, *Assassin's Creed: Odyssey's educational Discovery Tour mode is here at last, and better than before*. PCWorld Blog, September 2019.

[11] N. Dogruer, R. Eyyam, and I. Menevis, *The use of the internet for educational purposes*, in Procedia - Social and Behavioral Sciences, New York: Oxford University Press, 2011, pp. 606–611.

[12] E. Duffy, *When gamification goes wrong (and it often does)*. Pathgather Blog, January 2016.

[13] S. Egenfeldt-Nielsen, J. H. Smith, and S. P. Tosca., *Understanding Video Games,3rd edition*, Taylor & Francis, 2016.

[14] Frank D. Luna, *Introduction to 3D Game Programming with DirectX 10*, Jones & Bartlett Learning, 2009. ISBN: 9780763782801.

[15] G4C, *Festival Web Page*, August 2020. festival.gamesforchange.org.

[16] GAMBIT, *Phantomation.* Prototype Document, 2012.

[17] P. Gray, *A Brief History of Education.* Psychology Today, 2008.

[18] J. Greggory, *Game Engine Architecture, 3rd edition*, CRC Press, 2018.

[19] M. Hetfeld, *Why we now talk about politics in games so much.* Eurogamer Blog, 2019.

[20] D. Isla, *GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI.* Gamasutra Blog, 2005.

[21] J. Korenchan, *Ubisoft Entertainment, S.A. v. Yousician Oy (E.D.N.C.).* Patent Document, August 2019.

[22] R. Koster, *Theory of Fun for Game Design,2nd edition*, O'Reilly Media, 2013.

[23] Litteloud, *Sweatshop.* Online Game, 2011. www.gamesforchange.org/game/sweatshop/.

[24] H. Lou, *AI in Video Games: Toward a More Intelligent Game.* Blog, Special Edition on Artificial Intelligence, August 2017.

[25] N. Michelle, K. Heejo, and Y. Ronald, *Advertainment or adcreep game players' attitudes toward advertising and product placements in computer games*, Journal of Interactive Advertising, 5 (2004).

[26] J. Mulhern, *A history of education: A social interpretation, 2nd edition*, University of Oklahoma, 1959.

[27] J. NĚMEC and J. TRNA, *Edutainment or entertainment. education possibilities of didactic games in science education*, in THE EVOLUTION OF CHILDREN PLAY - 24. ICCP Word Play Conference., Pedagogická fakulta, Masarykova univerzita Brno, 2007, pp. 55–64. 10 pp. ISBN 978-80-210-4666-5.

[28] Oh, Honestly, *4 Life Lessons You Won't Learn from Mickey Mouse Clubhouse.* Blog, 2015.

[29] P. Gray, *The play theory of hunter-gatherer egalitarianism*, in Ancestral Landscapes in Human Evolution: Culture, Childrearing and Social Wellbeing, D. Narvaez, K. Valentino, A. Fuentes, J. McKenna, and P. Gray, eds., New York: Oxford University Press, 2014, pp. 190–213.

[30] C. Persaud, *Bloom's taxonomy: The ultimate guide.* blog, August 2018.

[31] Pluralsight, *How Animation for Games is Different from Animation for Movies.* blog, April 2014.

[32] Raymundo Barrera, *Unity 2017 Game AI Programming - Third Edition: Leverage the power of Artificial Intelligence to program smart entities for your games*, Packt, 2017.

[33] J. Rozen, *That's Edutainment.* The Atlantic Magazine, July 2006.

[34] J. Schell, *The Art of Game Design:A book of lenses*, CRC Press, 2008.

[35] Unity Engine, *Collider.OnCollisionEnter Documentation.* Scripting API Documentation, 2020.

[36] ——, *Quaternion.* Scripting API Documentation, 2020.

[37] ——, *Rigidbody.isKinematic.* Scripting API Documentation, 2020.

[38] ——, *RigidBody.MovePosition.* Scripting API Documentation, 2020.

[39] ——, *Transform.Position.* Scripting API Documentation, 2020.

[40] J. Wesley, *The Works Of The Reverend John Wesley, A. M.; Volume 4*, B. Waugh and T. Mason, for the Methodist Episcopal Church, 1835.

[41] B. Zomick, *Where gamification went wrong — and how to do it right.* Pathgather Blog, December 2017.

Appendix A

DVD contents

The enclosed DVD contains the text of the thesis as well as built teaching tools and their Unity project. The build requires graphics card with DX10 (shader model 4.0) capabilities and 100Mb of disk space. The DVD also contains the questions and answers of the user test.