Design Decisions for Indie Development of Educational Video Games

A Case Study

Andreas Kuoppa

Computer and Information Systems Science, bachelor's level
2019

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Abstract

Educational video games – especially for the PC market – do not seem to perform as well commercially as games from other genres. We argue that there is room for independent – ’indie’ – developers to break into the marketplace, by identifying certain niches and innovating on the genre. This would generate commercial value for such actors and knowledge value for the players.

Design decisions of high importance made during development of an educational video game demo at the small Swedish company ToLeap Consulting AB were analysed in the pursuit of contributing to effective indie development of such games. Three main problems that arose during development were identified, and three design decisions where implemented to combat these respective problems; (1) Interpreted educational game pattern utilising XML, (2) Function-based game views and (3) Community created assets, open source, and no costly dependencies.

In our case, the formulated design decisions effectively solved our problems, and we argue that they generalise. If a developer creating a similar game (educational video game) in a similar situation (independent development with limited resources) encounters one or more of these problems, the suggested design decisions may help the developer solve the problems, in turn making more educational video games available on the market, generating the aforementioned commercial and knowledge values.

Keywords: Educational video game, indie game, game development, XML, functional programming, open source
Acknowledgements

Thank you supervisor Ingemar Andersson, examiner Svante Edzén and opponent Alexander Tarberg for an interesting and rewarding last semester working on this thesis. I greatly appreciated your support.

Thanks to Toleap Consulting AB and everyone associated with the company, without whom this thesis would not exist. Go dream-team!

Thank you Björn Bergdahl, Andreas Svensk, Robert Kumpulainen, Ola Fornander, Victor Petersson and other friends, for the cloudy days you have brightened and this time spent studying together. I believe our teamwork through the years was a great asset for all of us; it certainly helped me. Also, thanks to all my energetic and inspiring lecturers during my time at Luleå University.

Thank you to all of my aunts and uncles and their respective families and other relatives. You have given me helpful advice through the years. A special thanks to Jasinta Åberg, Svante Åberg, Svea Maria Kuoppa Gunillasson and Nils Kuoppa, who have always shown a high interest in my education.

Thank you mom and dad, for being there all my life without a moment of hesitation. I love you Concis and Jan.

And thank you most of all my beloved Diána Horváth, for always believing in me and giving me the daily energy to keep on doing my best. I greatly look forward to the days yet to come, spent together with you. Love ya, lots!
This page intentionally left blank
**Contents**

1 Introduction  
  1.1 What is an educational video game? ................................. 2  
  1.2 History of educational video games ................................. 2  
  1.3 Educational video game market analysis ............................ 3  
  1.4 Educational video games' effect on learning .......................... 6  
  1.5 Development of educational video games ............................ 7  
  1.6 A note on the meaning of ‘indie development’ ......................... 7  
  1.7 Scope and intent ................................................. 8  
  1.8 Research Question ............................................... 8  
  1.9 Topics covered .................................................. 9  

2 Theory  
  2.1 Indie game development economics .................................. 11  
  2.2 Data management, abstraction and design patterns .................... 11  
  2.3 Functional game development ....................................... 14  
  2.4 Community created assets, open source and licensing ................. 15  

3 Method .............................................................. 16  

4 Results  
  4.1 Game description .................................................. 18  
  4.2 Game views ...................................................... 19  
  4.3 A note on the development of the game ................................ 23  
  4.4 Summary of design decisions ....................................... 23  
  4.5 Design decision 1: Interpreted educational game pattern utilising XML .................................................. 27  
  4.5.1 Problem ...................................................... 27  
  4.5.2 Design decision ............................................... 29  
  4.5.3 Analysis ...................................................... 35  
  4.6 Design decision 2: Function-based game views ......................... 37  
  4.6.1 Problem ...................................................... 37  
  4.6.2 Design decision ............................................... 37  
  4.6.3 Analysis ...................................................... 38  
  4.7 Design decision 3: Community created assets, open source, and no costly dependencies ......................................... 39
1 Introduction

Video games are a relatively young phenomenon. The video game market has steadily been growing and development techniques have improved over the years. Top selling games feature lifelike 3D graphics and are made by teams of hundreds of people at big studios. Video games are a multi-billion-dollar business worldwide. [1] The genres of video games are plentiful, and some genres do better in the market than others. The 'educational' genre and the near relative 'edutainment' genre are seemingly thriving on the mobile and tablet market, which is made clear by the owner and review counts when visiting the 'educational game', 'education', or similar sections at vendors such as Apple Store [2] or Google Play [3]. But how well are such games and applications doing on the PC market? According to statistics leaked from the content delivery system Steam by Valve [4], when looking at the game with the most players in the 'action' category, it had 46.3 million owners in July of 2018. But the game with the most players in the 'educational' category only had 350 thousand owners at the same time (section 1.3).

This raises suspicions that the market might be underdeveloped for the PC, or at least not as fully developed as it could be. There should be potential value left in this market to be gained for society in the form of profit for businesses, but could there also be some value in learning for the customers of these games? As educational and edutainment genres inherently contain some notion of 'teaching' and 'learning', one could argue that such games may be better for an audience to spend their time on, rather than games in other genres. At the very least, some connection between educational video games, learning and motivation has been shown to exist [5] [6] [7]. Hence, pursuing more games – and games of higher quality – in this genre could be a task argued to have some higher moral or ethical value. Would most parents not prefer that their children spent more time playing educational video games rather than violent ones? Maybe there are some reasons to make educational games more entertaining.
1.1 What is an educational video game?

Dondlinger makes an attempt to define educational video games based on observations of other authors:

Characteristics of such games [educational video games] include a system of rewards and goals which motivate players, a narrative context which situates activity and establishes rules of engagement, learning content that is relevant to the narrative plot, and interactive cues that prompt learning and provide feedback. [8, p. 22]

He defines edutainment games as:

Edutainment games, then, are those which follow a skill and drill format in which players either practice repetitive skills or rehearse memorised facts [...] [8, p. 22]

This is one attempt at defining the two terms, although they are in practice often used similarly or interchangeably. We believe that the game to be developed in our study best fits the educational genre, and we will henceforth refer to it as such, although it could be said to contain some elements from the edutainment genre, according to the definitions above.

As with all genres of any medium, there exist sub-genres for educational video games, although the small amount of such video games in the market (section 1.3), does not allow for many clearly defined, established sub genres.

1.2 History of educational video games

The history of educational use of computing and educational video games is essentially as long as the history of home computing [9] [10].

The market for such games often seems to have been very localised. This can be exemplified by looking at the Scandinavian countries over time, for example the PC adventure and travelling game Backpacker in the 1990’s [11], which had the sequels Backpacker 2 and 3. In the 2010’s, popular Scandinavian edutainment games for mobile platforms included titles such as Wordfeud [12] (essentially a variant of the board game Scrabble) and the
The popularity of games such as the latter, and the lack of new follow-up games for the then popular Backpacker 3 in 2004, demonstrate a migration of the medium from the PC platform to mobile platforms.

1.3 Educational video game market analysis

In this section, we will attempt a brief market analysis of the educational game genre with a focus on PC, as it is the target platform for the game developed in this study.

For this section, the term ‘PC’ will mean personal computers with either of the operating systems Windows, Mac OS or Linux; although the term elsewhere is used in both a more general sense as well as a narrower sense.

For PC, there are several content delivery systems from which video games can be bought; Valve’s Steam, Blizzard’s Battlenet, CD Projekt’s GOG, Electronic Art’s Origin, Epic Games Store, IGN’s Humble Bundle, Microsoft Store and the Apple App Store, just to name a few. For our purposes, focus will be on Steam, as it is one of the most well-known services of this kind available with a significant market share. Also, there are some clever ways to extract sales data from it.

Valve, the company behind Steam, does not usually share video game sales data with the end users. Only Valve and the developer/publisher of a specific game can access data of this kind. As tracking video game sales is of strong interest to the industry, there have been some reactions to this. In 2015, the site Steamspy was launched. The site analyses Steam player data which Valve makes available through an API, and from these data it tries to estimate the owners for the video games available.

Aside from Steamspy, there has not been much insight into the sales data of Steam, except from a few occurrences when such data leaked to the public. One such leak occurred in April of 2018, when a clever exploit was used to generate supposedly accurate sales data of games that offered its users ‘achievements’ (text and image-based rewards given to a player when he or she completes a goal). Valve quickly realised the potential of this exploit, and soon shut down the possibility to perform it. Still, an extensive CSV-file (comma separated values) with the data generated from the leak was made.
available. The file contains ownership data (closely related to sales data) of games with achievements published up until point of extraction. This file is mathematically more probable to portray the sales data to a greater degree of precision than for example the Steamspy data, as it is based on more representative values with higher degree of precision.

Given the CSV file and the Steamspy API – and for the purpose of this simple PC market analysis of educational video games – we have analysed the data available for all games, and also the genres 'Indie' and 'Education' (for this purpose, we included all Steam games which had the tags 'indie' or 'education', based on the Steamspy API). In the following section, we will present some noteworthy findings of this analysis. Once again, please note that the data are a snapshot of all games that offered players achievements via the Steam API in early- to mid-2018. The data are representative to a high degree of precision but probably not entirely correct. Tables 1, 2 and 3 summarises these findings. The terms 'players' and 'owners' are used interchangeably and refer to the number of Steam accounts which owns a certain game.

<table>
<thead>
<tr>
<th></th>
<th>All genres</th>
<th>Indie</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total games</strong></td>
<td>13 281</td>
<td>10 574</td>
<td>127</td>
</tr>
<tr>
<td><strong>Average players</strong></td>
<td>133 952</td>
<td>82 372</td>
<td>171 888</td>
</tr>
<tr>
<td><strong>Median players</strong></td>
<td>4 161</td>
<td>3 696</td>
<td>8 049</td>
</tr>
</tbody>
</table>

Table 1: Some simple statistics for various relevant game genres.

As can be seen in these tables, there are very few educational games on Steam, although those few games are performing relatively well compared to the competition in other genres.

But what are the reasons for this disparity between the educational genre and other genres? Are there inherent problems with educational video games? Are they too boring compared to shoot-em-ups? Are there development reasons they do not get produced in the same quantity? Are the games too expensive to create for too little value? Such issues make one wonder what could be done to improve the situation, although we will refrain from a deeper analysis here.

Mobile platforms seem to be more popular for the educational genre than the PC. Why is that? The educational genre is probably very well suited
Table 2: Number of games and percentages of games with more than a certain number of players, per genre.

<table>
<thead>
<tr>
<th>More than players</th>
<th>All genres</th>
<th>Indie</th>
<th>Education</th>
<th>All genres</th>
<th>Indie</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13 281</td>
<td>10574</td>
<td>127</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>100</td>
<td>12 110</td>
<td>9751</td>
<td>123</td>
<td>91%</td>
<td>92%</td>
<td>97%</td>
</tr>
<tr>
<td>200</td>
<td>11 371</td>
<td>9140</td>
<td>118</td>
<td>86%</td>
<td>86%</td>
<td>93%</td>
</tr>
<tr>
<td>500</td>
<td>10 062</td>
<td>8052</td>
<td>110</td>
<td>76%</td>
<td>76%</td>
<td>87%</td>
</tr>
<tr>
<td>1,000</td>
<td>8 999</td>
<td>7122</td>
<td>105</td>
<td>68%</td>
<td>67%</td>
<td>83%</td>
</tr>
<tr>
<td>2,000</td>
<td>7 881</td>
<td>6149</td>
<td>90</td>
<td>59%</td>
<td>58%</td>
<td>71%</td>
</tr>
<tr>
<td>5,000</td>
<td>6 309</td>
<td>4846</td>
<td>73</td>
<td>48%</td>
<td>46%</td>
<td>57%</td>
</tr>
<tr>
<td>10,000</td>
<td>5 037</td>
<td>3798</td>
<td>59</td>
<td>38%</td>
<td>36%</td>
<td>46%</td>
</tr>
<tr>
<td>20,000</td>
<td>3 963</td>
<td>2910</td>
<td>48</td>
<td>30%</td>
<td>28%</td>
<td>38%</td>
</tr>
<tr>
<td>50,000</td>
<td>2 625</td>
<td>1849</td>
<td>35</td>
<td>20%</td>
<td>17%</td>
<td>28%</td>
</tr>
<tr>
<td>100,000</td>
<td>1 845</td>
<td>1242</td>
<td>24</td>
<td>14%</td>
<td>12%</td>
<td>19%</td>
</tr>
<tr>
<td>200,000</td>
<td>1 141</td>
<td>706</td>
<td>20</td>
<td>9%</td>
<td>7%</td>
<td>16%</td>
</tr>
<tr>
<td>500,000</td>
<td>584</td>
<td>303</td>
<td>10</td>
<td>4%</td>
<td>3%</td>
<td>8%</td>
</tr>
<tr>
<td>1,000,000</td>
<td>335</td>
<td>159</td>
<td>6</td>
<td>3%</td>
<td>2%</td>
<td>5%</td>
</tr>
<tr>
<td>2,000,000</td>
<td>171</td>
<td>76</td>
<td>3</td>
<td>1%</td>
<td>1%</td>
<td>2%</td>
</tr>
<tr>
<td>5,000,000</td>
<td>55</td>
<td>19</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>10,000,000</td>
<td>18</td>
<td>5</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>20,000,000</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>50,000,000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

for portable devices. One could play the game during a short coffee break, whereas the same cannot be said for the PC. Why then cannot the educational genre be better suited for the PC market? Maybe developers should design and develop educational PC games differently to mobile games, to better suit the platform and result in a better player experience.

As can also be seen in tables [1] to [3], the categories 'all genres' and 'indie' seem saturated, but the average 'education' game has double the sales numbers. Maybe there still is a market to be found here for smaller companies or indie developers? If our game in this study could sell 8,000 games, the median sales number from Table [1] the project would be profitable, although it would barely be profitable if it sold 4,000 units, the median value of the other categories. This is according to the more detailed sales estimates and market analysis of our game, which is otherwise mostly confidential. Sales estimates hinting at some chance of success and sustainability are important for indie developers considering beginning game development projects.
1.4 Educational video games’ effect on learning

Studies examining the educational effects of video games include Beyond Nintendo: design and assessment of educational video games for first and second grade students [5], The effects of playing educational video games on kindergarten achievement [6] and The Effects of Game-Based Learning on Mathematical Confidence and Performance: High Ability vs. Low Ability [7].

All these articles stress at least some positive results on learning from utilising educational video games in a school environment, supporting our claims on educational games as learning tools, and the moral value in pursuing more games in the genre.

Although we present a few relevant articles above, to our knowledge there have not been very much research on the effects of educational video games. In Beyond Nintendo: design and assessment of educational video games for first and second grade students, the researchers seem to somewhat agree with this observation:

To our knowledge, no other studies have been carried out using video games in the school context with the purpose of teaching regular instructional contents. This is surprising, considering that several authors have indicated the need to bring together the world of video games and education [...] [5, p. 89]
1.5 Development of educational video games

There is not very much material on the effects of educational video games and there does not seem to be much on development aspects of such games either, even though several video game companies have managed to produce successful games in this category. Counterexamples of this are the articles presented in the theory section, which to an extent covers certain development aspects of such games. Another good article on educational game development is *Educational Video Game Design: A Review of the Literature*, which acts as an introduction to the literature of educational video game design, although this article seems to be focused more on educational design elements, rather than a focus on the development process.

We hope that this study can shed some new light on decisions, methodologies and guidelines for effective indie educational video game development, as well as open debate and discussion around the subject.

1.6 A note on the meaning of 'indie development'

The term 'independent' or 'indie' game development is today known to roughly mean small scale, non-enterprise development. Some notable differences between indie development and more professional large-scale studio development include notable differences in budget, man-power, development methods and marketing, indie development is commonly known to be lacking resources for all of those. At least, the methodologies of indie vs enterprise development are very different. Another definition of indie is development independent from entities such as parent companies or publishers – indie development is often crowd-funded (or not fully funded at all) – although such stricter use of the term seems rarer today.

A positive connotation of indie is the relative ability to be very creative and agile, compared to huge corporations which can be slow moving and bogged down in processes and hierarchies. As such, some successful indie games are crowd-funded, and/or depend on some goodwill of the player base in other ways.
1.7 Scope and intent

During this study, a small team of three people – myself included – at the small Swedish company Toleap Consulting AB [15] has worked part time to develop a demo version of an educational video game for the PC market. During development, problems arose, and design decisions were made to combat those problems. This study analyses and evaluates these design decisions. The purpose of this is to pursue improved development techniques for educational games, to in turn make more games – and games of higher quality – of this genre available on the market.

The video game in question being developed in this study, is developed by a small team – three people working part time. The results of this small study might be of value for small to medium sized game studios, rather than for larger ones. Thus, we make use of the term 'indie' to describe our specific development situation.

Although we briefly touch on the economics of developing indie games, we believe that the more intricate economics of such projects are outside the scope of our study. Firstly, we think that were we to focus on such subjects the scope would be too big, and we would lose focus on the development aspects of indie educational games, which is the main area in which we feel most confident contributing to. Secondly, as the game developed in this study has as of time of writing not yet entered the market, we do not have the foundation or the results on which to base such a study.

1.8 Research Question

The study was conducted from the onset with the following research question in mind:

RQ: Which design decisions contribute to effective indie development of educational video games?

Below, all the relevant terms from the RQ are defined:

- **design decision** - Refers to a decision made, trying to solve a specific problem that arose during development.
- **effective development** - In our own terminology in the context of this study: (1) Effectiveness as in quality and performance of the final product, and (2) Effectiveness in the sense that development time is spent as efficiently as possible.

- **indie** - Short for independent. See the discussion in section 1.6

- **educational video game** - A video game with educational elements, see section 1.1. More specifically for this study, see section 1.9

Note that the study of course cannot possibly cover the set of all relevant design decisions for such games, but focus will be on design decisions found to be of importance during development of this specific game. Once again, see section 1.9

### 1.9 Topics covered

Some properties of the game developed in this study which map well to the topics covered are defined in the list below. Other authors have arrived at similar design decisions as we did in this study and are referenced therein. The topics will be further explored in the Theory (section 2) and applied within the context of our development case in the Results (section 4):

The game...

1. is a relatively small-scale, independently developed educational video game for which the development costs need to be kept low \[16\] \[17\] (this is the over-arching problem, from which the other problems in a sense originate)

2. stores, handles and is defined by large amounts of data and needs some kind of question database or storage method and a system to interpret it \[18\] \[19\] \[20\]

3. is centred around a small number of function-based ‘views’ or ‘scenes’ (city, airport, monument, jobs etc), which define the entire game \[21\] \[22\]

4. uses assets such as third-party libraries, maps and photos, which usually are subject to intellectual property (IP) laws and regulation, rights for which can be costly and complicated to purchase \[23\] \[24\]
If a game to be developed is subject to one or more of those properties, the results of this study might be relevant to that specific development situation. If such a game is not subject to any of these properties, the study is not relevant. The list items 2, 3 and 4 maps to design decisions 1, 2 and 3, respectively (sections 4.5, 4.6 and 4.7), guiding the reader to the specific parts of the results he or she might be interested in.
2 Theory

In this section we will discuss the relevant theory for the upcoming sections on the design decisions of the development of the game in this study.

2.1 Indie game development economics

The purpose of this study is improving development strategies for educational video games, to enable more such games on the market, generating monetary and knowledge value. This intent suggests that there might be a need for indie developers to keep costs down and develop games efficiently; in time and on budget. This is certainly a well debated subject and a subject well written about; but more than to mention it, the more intricate economics of developing educational indie games are outside the scope of this study. We will instead refer to Futter, whose books on indie development, business and budgeting are a great introduction to the subject:

- *The Gamedev Business Handbook: How to build the business you’ll build games with!* [16]
- *The Gamedev Budgeting Handbook: How to finish your game in time and on budget!* [17]

2.2 Data management, abstraction and design patterns

A concept that is very central to this study, especially to Design Decision 1 (section 4.5) is *metaprogramming*, which is a well-known concept or design pattern. Summarised, it describes the concept that programs should use *metadata* to describe configuration. This concept is very well described in *The Pragmatic Programmer*:

Configure, Don’t Integrate [18, p. 144]

and

Put Abstractions in Code, Details in Metadata [18, p. 145]
As will be seen in the results section, this is an abstraction well suited for our game development process. More specifically for this study, other authors have applied this concept and taken it a few steps further for educational video games, such as in the articles The <e-game> project: Facilitating the development of educational video games [20] and XML Application for Educative Games [19]. The authors in both articles are describing development of similar games to ours within the educational genre. They draw similar conclusions for the data management of those games.

Both studies decided on using XML files for a number of reasons. The <e-game> developers state that there are two flavours of game development, high-level programming languages for the main development and scripting languages for specific tasks. They stress that it is common for games to have a hard-coded engine and separated script files, which the engine interprets to produce the game. They state that a solution like that is similar to their suggested approach, but they have a focus on modelling content rather than functionality. Furthermore, a certain degree of programming knowledge is a prerequisite in order to understand scripting files; a more simple and intuitive solution is needed for those without programming background. With that context in mind they present XML as a solution:

These needs suggest that XML technologies could be an interesting candidate. XML is focused on describing hierarchically structured content, which is closer to our needs than a full programming language. [...] XML even provides mechanisms such as DTD and XML-Schema that allow the formalisation of the language in a machine-readable way. This facilitates enormously the construction of the <e-game> system. It also facilitates the author’s task by providing means to verify the correctness of the documents before feeding them to the system. Besides, the wide acceptance of XML means that there is an assortment of commercial products supporting this kind of authoring process. [20] p. 4-5

The overall application design of the <e-game> project is presented in Figure [1]
Minović and his team arrives at a similar approach, but also identifies needs which results in the following design decision:

[...] the first step was to separate the process of knowledge creation from the process of creating game environment and game context. We achieved that by creating two separate software modules, thus gaining the ability to resolve the communication conflict between the game designer and the knowledge expert. [19, p. 310]

Thus, they instead used the approach in Figure 2 for their application design. Basically, this process is very similar to the system proposed for <e-game>, with an extra path added, which separates the ruling, scenarios and game logic part from the knowledge repository part of the game. As the <e-game> project is described as an adventure game, it does not seem to have a question or trivia part, thus it is not in need of such separation. But with this in mind the game designer and expert can work independently, since they do not touch the same files.

In our study, we have independently arrived at similar system designs as the authors of the two articles we just mentioned, but modified and improved for
Figure 2: The ‘Educative Game’ application design[19]. Compared to Figure 1, this solution has an added branch for a knowledge repository part of the project.

our needs (section 4.5 and Figure 9).

Another pattern, that is somewhat relevant to our case is the Singleton Pattern, described by Nystrom as:

Ensure a class has one instance, and provide a global point of access to it. [25, p. 73]

This pattern will be used for representing player state, which we found worked well for our purposes as a companion to the functional game view pattern we arrive at in section 4.6.

2.3 Functional game development

Today, the object-oriented paradigm is widespread and well established. It has made way for a plethora of design patterns to select and combine to solve solutions to various problems [18], and solutions such as those are common within the games industry [25].
Functional programming is not as widespread as object-oriented programming within the games industry, although a number of attempts have been made to write frameworks enabling such development methods, such as the Easel Framework for functional game programming [21]. This framework is fully functional, and although within our results in section 4.6 we do not propose to use a fully functional paradigm, we suggest that a functional pattern can be used instead of the object-oriented pattern at certain levels of the application, in a similar spirit to what the authors of the framework suggest.

Another author that has tested to implement a strict functional pattern, in this case a 3D game in Haskell and Yampa, stresses that it is indeed possible to do, and discusses some merits as well as drawbacks [22]. This game is in a sense more similar to ours, as it uses a 3D engine. Although our game does not implement the 3D engine in a fully functional manner, we argue that this article strengthens the case for our ‘functional view’ pattern we present in section 4.6.

2.4 Community created assets, open source and licensing

Today it is common to make use of open source and community created assets. A number of well-established assets that have risen in parallel with the increasing demands of open source have become de facto standards. Examples of such licenses include different Creative Common Licenses [26] among others such as MIT [27] or GNU [28] licenses or similar. There exist web sites that help developers to choose the appropriate licenses for their specific use-case [29] as well as organisations with the intent of raising awareness and adoption of open source, such as the Open Source Initiative [30]. Note that licenses are not limited to source code, most of them can indeed be used for most works of art or products created of creative value, for many different use cases.

Distributed work-flow and versioning systems such as Git [31], along with services such as Github [32], have become well-established by professionals and hobbyists alike. Such systems enable Free and Open Source (FOSS) development and usage.
3 Method

During this case study we have developed an educational video game demo. The case we are interested in studying is the development process of this demo. We will focus on the design decisions that were made during the process, which relate to the topics presented in the introduction (section 1.9).

The problems that arose and the design decisions that were made during development addressing these problems were documented in detail. We kept a journal which we continuously updated during the process, each time we made a seemingly important decision. Later, design decisions made were analysed and evaluated based on the journal and other documentation such as design documents, minutes of meetings and version control system history.

Early on, we looked for relevant theory that previously had tried to solve the set of problems which we had discovered. Although we base our problem-solving process on this theory, the process of developing each design decision has had a slightly different approach, given that the problems have risen in a natural context during development. Each problem was dealt with accordingly, as it appeared. Thus, the results section of this work also contains some method and process description, as the various processes are strongly coupled to the specific solutions and are part of them.

A strategy commonly employed was requirement specifications, listing the needs which a solution in the form of a design decision needed to adhere to. For the first design decision, this is clearly illustrated when we choose our data storage technology. In that section, first we simply present our requirement list and discuss it. Then, we demonstrate a technology that did not fulfil our requirements (SQL), before showing the solution that we decided upon (XML), which did in fact fulfil our requirements. Such requirement specifications or similar requirement descriptions or use case statements were commonly used during our development process, although we do not explicitly show them in all cases in the results sections, as they are not always relevant for understanding the bigger picture on why we decided upon a certain design decision.

The project has a focus on the development process rather than on the pro-
duced IT-artefact – our educational video game demo – and therefore the case study approach seems appropriate. We have not applied any strict methodology or framework for our study, but we have based our approach on relevant theory, to make sure that results can be generalised for building new theory \[33\] \[34\].

In the results section 4, we present each design decision as its own entity, as they are based on separate processes and topics. First, we summarise each of the three design decisions, before going into them in more detail, where we structure each design decision into three sections as follows:

1. **Problem**
   A description and a specification of the problem we have identified.

2. **Design decision**
   We demonstrate the process of coming up with the specific design decision addressing the problem. Then, we describe the design decision and its properties in detail and describe how it has solved the problem.

3. **Analysis**
   We analyse the design decision further and illustrate advantages and disadvantages of it, based on our findings, properties of the subsystems and technologies that make up the design decision, and the theory.

Then, in the conclusion and discussion section 5 of this work, we summarise the important findings of each design decision and what can be learned from each. We close with a discussion on differences and improvements of our design decisions compared to the theory.
4 Results

4.1 Game description

In the introduction we stated that the aim of this study is to contribute to effective indie development of educational video games. The reader needs to know the basics of the game developed in this study to understand the results and which game development situations the design decisions presented in the results might apply to. Such a description follows:

**Game working title**

Voyager

**Game description**

An educational video game where the player travels to various well-known cities. To finance further travels the player must apply for various jobs in the cities so that she can afford plane tickets to another city, or she will go bankrupt and the game is over. At the time of writing, the game does not have any clearly stated goals, but it could be said to have been completed when the player has visited all the cities in the world and played through all the content within each city, including visiting all monuments and working all jobs. The educational part of the game consists of answering questions when visiting monuments and working jobs, but also learning the overall layout of known cities in the world (via a map interface) and seeing different views from photos that are included for the cities.

**Target audience**

Wide, age 7 and up.
4.2 Game views

In this section, the brief game description in in section 4.1 will be somewhat improved upon with a description of the game’s primary views and accompanying screenshots of them, as the project looks at time of writing.

The game is composed of these five primary views:

- **Airport view** - Here the player can purchase a plane ticket to travel to another city.

  ![Figure 3](image)

- **City view** - The 'heart' of the game. A map of the city and an image of it is shown to represent the city which the player is currently in. From here, the player can choose to navigate to any of the other views. That is; visit monuments, visit different workplaces to apply for jobs, view information about the city or visit the airport to travel elsewhere.

  ![Figure 4](image)

- **Monument view** - Here a player sees an image of the monument and is told some trivia about it. Then the player is asked a question related to the monument, if she answers correctly, she is awarded experience points.

  ![Figure 5](image)

- **Job view** - This view represents a fictional company with open job vacancies to be applied for. If the player applies for a job, she must answer enough questions correctly to get the job. A player needs to apply for jobs to receive a salary to finance further travels.

  ![Figure 6](image)

- **Information view** - Here the player gets access to information about the city and the country which the player is in.

  ![Figure 7](image)

These descriptions of the primary game views, along with the screenshots and the brief game description, should guide the reader to make sense of the 'Design decision' parts of this result section.
Figure 3: The airport view. This image displays the airport view in Stockholm and thus displays all plane tickets and routes available from there.

Figure 4: The city view. Here, the player is in Stockholm.
Figure 5: The monument view. This particular view is about the Stockholm City Hall.

Mao Zedon was a communist leader of China.

Mao Zedon was actually born as a member of the Chinese royal family.

Question

<table>
<thead>
<tr>
<th>True</th>
<th>False</th>
</tr>
</thead>
</table>

Answer
Figure 7: Information view for Stockholm.

Figure 8: View relations, with clickable 3D icons shown by the arrows, which takes the player from the City view to the other views.
4.3 A note on the development of the game

The game demo was developed by three people working part time. Although some definitions of ‘agile’ are very strict, within our team we say that an agile methodology is used during development, more particularly Kanban including the concepts of sprints, product backlog and sprint backlog among other well-known agile concepts [35]. Still, we have not specified the development much more than that, so we will leave it up to the reader to decide whether the project is agile or not. As the project is part-time and is not at this stage planned to be a reliable source of income for the company, it is not subject to any notable company or customer rules or regulations. The progress of the game is largely dependent on the participants will to finish it. The participants have a monetary interest to finish the game – should it be profitable – for two of them in the form of shareholder pay-out and for the third in the form of provision.

Note that the planning and development phases of the game demo were initiated more than six months prior to this case study, so that a few hundred man-hours of progress had already been made to the project, before the start of the case study. The reasoning behind all design decisions presented herein started to take form during those first few hundred hours of development. Early in the process it became clear that we needed a consistent application design that was well thought through and based on relevant theory. We started by formulating our own attempts at solutions to the various problems, presented in the upcoming sections. After those solutions had been identified and clarified somewhat, we searched through what had already been written in the field, and the combination of our preliminary solutions along with the relevant theory landed in what we call design decisions.

4.4 Summary of design decisions

In the problem statement for each design decision presented in the following sections, we will clarify the reasoning of the developers, after which the context behind the proposed design decision to solve the problems will hopefully become clear. Also, we summarise our three design decisions in tables 4-6, linking the design decisions to the relevant topics, theory, problems and processes.
Topic: Data storage method along with an interpreter. Metadata, application design.

Previous theory: A metadata pattern [18] and the application designs of <e-game> [20] and Educative Game [19].

Problem: Educational video games typically need to store and handle large amounts of data. There could be various views with several objects that the user interacts with, and the 'educational' part of the genre often implies the need for a question database. How do we design the game and the data creation and editing processes to enable a streamlined work-flow?

Process: We list the requirements we need from a data storage method or technology. We compare different technologies to this requirement list, and choose the technology that best matches the requirements. Based on our development process and needs along with previous theory, we derive a new application design for our game.

Design decision: We define the interpreted educational game pattern utilising XML, summarised in Figure 9. To produce a working educational game with this pattern, the game designer need only care about writing valid XML files from which the entire game is produced.

Table 4: Summary of Design Decision 1.
Function-based 'views' or 'scenes' (city, airport, monument, jobs etc), which defines the entire game. 3D scenes and programming. Functional game programming, in contrast to object-oriented programming.

Previous theory


Problem

Games with many mechanics and much content can grow to be incredibly complex, as well as can games with 3D graphics. Layer upon layer of dependencies, concepts, and methods are built upon each other, and the finished product risks turning into a complex piece of software. For our game we have very limited resources available. How do we keep our game small and manageable? Also, how do we do all this while simultaneously ensuring good compatibility with the 'interpreted educational game pattern' defined and described for the previous design decision?

Process

We realise that our game is analogous to a slide-show presentation, hyper-linked documents or similar, although stylised in 3D and containing some state. We work backwards from this observation to arrive at a design decision which reflects this approach.

Design decision

We define what we call function-based game views. These are views which are presented to the user, takes the user’s input and produces an output. At code-level, a single function maps to a single view.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Function-based 'views' or 'scenes' (city, airport, monument, jobs etc), which defines the entire game. 3D scenes and programming. Functional game programming, in contrast to object-oriented programming.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>Games with many mechanics and much content can grow to be incredibly complex, as well as can games with 3D graphics. Layer upon layer of dependencies, concepts, and methods are built upon each other, and the finished product risks turning into a complex piece of software. For our game we have very limited resources available. How do we keep our game small and manageable? Also, how do we do all this while simultaneously ensuring good compatibility with the 'interpreted educational game pattern' defined and described for the previous design decision?</td>
</tr>
<tr>
<td>Process</td>
<td>We realise that our game is analogous to a slide-show presentation, hyper-linked documents or similar, although stylised in 3D and containing some state. We work backwards from this observation to arrive at a design decision which reflects this approach.</td>
</tr>
<tr>
<td>Design decision</td>
<td>We define what we call function-based game views. These are views which are presented to the user, takes the user’s input and produces an output. At code-level, a single function maps to a single view.</td>
</tr>
</tbody>
</table>

Table 5: Summary of Design Decision 2.
Use of third-party and open source code and assets.

**Previous theory**
Free and Open Source (FOSS) development practices, methods and results in the game community [23, 24].

**Problem**
The game needs a lot of assets such as photographs and maps. The budget does not allow for much – if any – assets to be purchased. We need to strike a balance between assets created by us – the developers, and assets acquired from elsewhere. The project’s low budget also means that we cannot be reliant on expensive programs and dependencies.

**Process**
Early on in the development process we realise that the budget and the available man-hours in the project are not sufficient for developing our own assets and we explore other solutions. We quickly land in the decision to rely heavily on FOSS.

**Design decision**
We use free and open source libraries and works of art in the development process, with carefully selected licenses appropriate for commercial use.

| **Table 6: Summary of Design Decision 3.** |
|------------------------------------------|---|
| **Topic** | Use of third-party and open source code and assets. |
| **Previous theory** | Free and Open Source (FOSS) development practices, methods and results in the game community [23, 24]. |
| **Problem** | The game needs a lot of assets such as photographs and maps. The budget does not allow for much – if any – assets to be purchased. We need to strike a balance between assets created by us – the developers, and assets acquired from elsewhere. The project’s low budget also means that we cannot be reliant on expensive programs and dependencies. |
| **Process** | Early on in the development process we realise that the budget and the available man-hours in the project are not sufficient for developing our own assets and we explore other solutions. We quickly land in the decision to rely heavily on FOSS. |
| **Design decision** | We use free and open source libraries and works of art in the development process, with carefully selected licenses appropriate for commercial use. |
4.5 Design decision 1: Interpreted educational game pattern utilising XML

4.5.1 Problem

Educational video games typically need to store and handle large amounts of data. There could be various views with several objects that the user interacts with, and the ‘educational’ part of the genre often implies the need for a question database. How do we design the game and the data creation and editing processes to enable a streamlined work-flow?

After some discussion we agreed on some basic properties that a solution for the data format for our game needed to achieve. We list our requirements of such a format below.

Our team decided that we needed to rely on a data format that...

• favours readability and ease of use over performance
  
The solution should be easily interpreted by both humans and machines, although favour human readability over technical performance, should we have to choose. The thinking behind this decision is that we have more computational power than needed for our game at our disposal, but man power is limited. Game designer additions and changes should be quick and easy to implement.

• can describe an entire ’city view’ in a single file of reasonable size
  
During development, it felt natural to talk about the game with the ‘cities’ being the main focus. All the other functionality and views were related to the city view in one way or another. Thus, we decided that if the game for example contains 40 cities, the game should be fully described by 40 files – each one representing a city and all its contents. From one perspective the entire game could simply be viewed as an interpreter of these data.

• encourages the game designer to work directly in the file
  
As we wanted development to be rapid and with fast iterations, we wanted the game designer to be able to work directly in the file format.
Indeed, this should not only be possible to do, but the main work flow. We did not want any extra UI or software to be produced for file entry or manipulation, as this is both costly, time consuming and also likely to be error-prone, as such a UI system introduces the possibility of more errors/bugs. The preferable alternative would be to be dependent on a file format that can easily be validated. This requirement is closely related to the readability and ease of use requirement.

- **is relatively lightweight and can be bundled with the game when shipped**

  We want the finished game to be entirely self-contained, and not be subject to any heavyweight dependencies or programs. One reason for this is that it makes it faster and easier to release updates, for example when releasing the game to several content delivery systems or marketplaces. Lightweight solutions could be argued to be better both performance-wise and development-wise. This is a case of ‘use the right tool for the job’, we do not need the whole toolbox when a hammer is sufficient.

- **can easily describe composition/aggregation**

  Data in the game are often relational. Using terminology from RDBMS and OOP the ‘has a’ relationship is very common. For example, a city ‘has a’ job. The ability to describe these kinds of relationships – while still conforming to the other requirements listed here – is of high importance to us.

- **is not subject to vendor agendas/quirks such as versioning, bugs, up-sells and security problems**

  We do not want to be dependent on ‘other entities’. We feel like using a well-known standard or well-established technology which is well-defined and self-contained, to as high a degree as possible, is preferable to being dependent on a technology from a vendor that requires regular updates and maintenance.

- **can be extended to form a ’custom’ file format which can easily be validated**

  As we decided early on that the game should consist of a small number of ‘city’ files that together define the whole game, and that the game
could simply be viewed as an interpreter of those files, we wanted a file format which we could extend to define our own format for valid city files. We also wanted to use a checker for these files, to easily see if they are valid against our own defined format.

4.5.2 Design decision

Some distinct technology solutions quickly came to mind which are usually used in situations like these, such as, but not limited to:

- SQL-based relational databases
- NoSQL or NewSQL databases
- Object-oriented databases
- File formats such as JSON and XML

We decided that the first three of these list items did not fulfil our requirements. Instead of expanding on all our reasoning for disregarding these technologies, we will select only one of the technologies – SQL-based relational databases – to show which requirements the technology did not fulfil. Further discussion on other technologies, and why they did or did not fulfil the other requirements, is left as an exercise for the reader. Instead we will do the reverse: show the solution that we ended up using – XML files – and explain why it fulfilled all the requirements.

We admit that there are exceptions and counter-arguments to be had to the following analysis:

**SQL-based relational databases**...

- do NOT favour readability and ease of use over performance

Relational databases are meant to be easily understood, usable and highly logical, as well as very optimised from a performance point of view. Yet, we argue that they are not very ‘readable’ in the sense that one cannot simply look at a large database schema and at a quick glance get a sense of what is going on within it, without already understanding the thinking behind that specific database schema. Relational database
concepts and quirks such as normalisation and de-normalisation do not add to the ease of use of such a technology, either.

- **can NOT describe an entire ‘city view’ in a single file of reasonable size**

  There probably need not exist many database tables to describe a ‘city view’ as we use them in the game. Still, database schemes are often highly coupled to their respective RDBMS. Most relational database solutions do not simply allow for easy file transfer of a data model, in fact, one purpose of using a RDBMS is to eliminate the need for file transfer.

- **do NOT encourage the game designer to work directly in the file**

  RDBMS systems are highly specialised and dependent on specific software and methodologies. Changing or adding to a database schema is mostly not as simple as opening a file, making the changes, saving, and then closing it.

- **are NOT relatively lightweight and can NOT be bundled with the game when shipped**

  Relational databases require the use of a query language such as SQL, this in itself immediately introduces complexity. Also, RDBMS’s are often very heavy weight.

- **ARE subject to vendor agendas/quirks such as versioning, bugs, up-sells and security problems**

  This technology is strongly dependant on the chosen RDBMS.

As can be seen above, SQL-based relational databases do not fulfil five out of our seven requirements we have of a data model technology, according to our own analysis.

After careful consideration, the team decided to use a file format based on XML, validated against XSD, as it fulfils all the listed requirements.
• **favours readability and ease of use over performance**

  XML is – when well-formed and well-written – very readable. Using XML with an intent for humans to read it, rather than computers, can be very rewarding. XML does not perform badly, but there certainly are better file formats to use when optimising for performance, hinting that XML may be a good choice if readability is the goal.

• **can describe an entire ’city view’ in a single file of reasonable size**

  At time of writing, two cities are complete. Both these city XML files contain just over 500 lines of data. We suspect, worst-case scenario, that each of the city files will be 1 000 lines long. If the finished game contains 40 cities, this amounts to a total of 40 000 lines of XML, which is by our measures a good count. We are also impressed by how just a few lines of XML maps to quite a large amount of playable content.

• **encourages the game designer to work directly in the file**

  Both of the completed city files in the game contain easy-to-read data, like well-written modern HTML. As they are also very short, and the XSD validation warns the game designer if his XML is not well-formed, he is encouraged to add and change assets and data in the game by working directly in the XML city files. Also, using a good editor, full XML code blocks can be pre-filled, according to the XSD definition.

• **is relatively lightweight and can be bundled with the game when shipped**

  The finished XML city files are around 20 kB in size, considering our worst-case scenario of double the file length, they would be 40 kB each. For a total of 40 cities, this would amount to roughly 1.6 MB, which is incredibly small sized considering that these files makes up the heart of the game. Another plus of using stand-alone data files such as these is that if we want to change assets and data – for example due to a bug or if we want to add more content – we can easily just release an update with changed data files.
• can easily describe composition/aggregation

XML defines the concepts of elements and attributes, both of which supports composition and aggregation (the 'has a' relation). For example, elements can be nested, so that an element 'has' another element. An element also 'has' an attribute.

• is not subject to vendor agendas/quirks such as versioning, bugs, up-sells and security problems

The XML 1.0 standard which we use for our game is very well established and well used, and is not property of a third party.

• can be extended to form a 'custom' file format which can easily be validated

XML supplies the Document Type Definition (DTD) and XML Schema Definition (XSD) file types for exactly this purpose, 'out of the box'.

The solution of using XML and XSD files for this game proved to be very successful for our purposes. The development process and application design we derived is summarised in Figure 9. Examples of the XML and XSD implementation as used in-game can be found in Figures 10 and 11.
Figure 9: The application design for Voyager. The game designer and the expert work with the same set of XML files, validated against XML Schemas (XSD’s) defined by the developer. The game is defined in its entirety by an arbitrary number of valid XML files, the component repository and the interpreter. With this system, the game designers and experts only need to care about writing valid XML files, to produce a working game. Note the similarities as well as the differences to the proposed system designs in the theory (Figures 1 and 2). We have streamlined the process so that the game designer and the expert work with the same set of files. We found that if they are validated against an XSD, there was not any need to separate the two processes. To clarify, no conflict situations can arise if both the designer and experts write valid files. Also, we have explicitly drawn the XSD validation as part of the process. Although the authors of the other system designs hints at the usage of schemas, they have not drawn them in their development processes, whereas we feel that they are so essential to the process that they should be explicitly shown. With this solution, the interpreter does not need to handle erroneous input files. We will refer to the pattern presented in this figure as the ‘interpreted educational game pattern’
Figure 10: Example of the XML city file describing Stockholm. The file is abbreviated and changed from the one used in-game for presentation purposes here but is fully valid according to our XSD. Using this file, a map of Stockholm will be shown on the screen, with an airport, info center, workplace, monument and so on, each of which is clickable and will generate the corresponding view, as described in Figure 8. Note the relative ease in which one can read and understand the file. It is easy to deduce what the elements and attributes mean, without having a deeper knowledge about developer design choices.
4.5.3 Analysis

Since we were very happy with the results of using XML and XSD, an in-depth discussion on potential drawbacks of the method is not very interesting; we find it sufficient to list just a few of the commonly known disadvantages of XML:

- **It needs an application processing system**

  XML needs a system to parse it. Although there are common simple parsing tools available today for XML in all most known languages, the program for which the XML is used still needs to understand the written XML to a higher level as defined by the developer, this process could be argued to be less straight-forward than when using file formats such as JSON.

- **It is verbose**

  XML is by definition more verbose than other file formats such as JSON. It could be argued that this verbosity introduces unnecessary overhead. Also, the quality and readability of XML as such is very
dependent on the author and the common rules for writing the XML documents.

• **It is legacy technology**

There seems to be an overall feeling with XML that it is old technology, and this might be somewhat true, since it is a relatively old standard and a relative of the older, similar, standard SGML. Surely, there are newer technologies more suitable for similar use-cases.

Although there is some solid reasoning about these drawbacks of XML, we would argue that reality is not clear-cut, and that XML can be used in more situations than is often given credit for by common jargon. We want to stress the mantra ‘the right tool for the job’. This is well illustrated by our use case, which made it clear that XML was the best candidate for our purposes. XML can be really powerful, especially when validated against DTD or XSD, and when written in a ‘human-friendly’ style, akin to the style of modern HTML. Also, XML being old is a viewpoint that partly might originate from systems which use XML and have grown to become monolithic and non-developer friendly. This could be a chimera; we believe that the simple fact that XML has been used in the field for such a long time makes it more likely to be used in questionable ways. Technologies that have only been around for a few years may not yet have had the same opportunities to be misused.

Another positive of using the XML and XSD solution dawned on us after implementation. It is relatively simple to extend the game data and asset creation with a GUI editor which generates and edits the XML city files. In our requirements we stated that we did not want to be dependent on such a solution, but it is nice to have the ability for such an extension, should the need arise. One such imagined scenario – should the game be successful – is to release such a program as a ‘modding’ tool for the player base to create their own custom cities in the game. This would be preferable as the players cannot be expected to make changes directly in the source files.
4.6 Design decision 2: Function-based game views

4.6.1 Problem

Games with many mechanics and much content can grow to be incredibly complex, as can games with 3D graphics. Layer upon layer of dependencies, concepts, and methods are built upon each other, and the finished product risks turning into a complex piece of software. For our game we have very limited resources available. How do we keep our game small and manageable? Also, how do we do all this while simultaneously ensuring good compatibility with the 'interpreted educational game pattern' defined and described for the previous design decision (section 4.5 and Figure 9)?

4.6.2 Design decision

After exploring different solutions to how the game views should be structured, we finally decided on something very simple. We came to realise that our game was not much more complicated than a slide-show presentation, although within this analogy some of the slides are very complex, stylised in 3D and containing some state. With this thought in mind, we encourage the reader to think about figure 8, showing the game's views. If one does not take game state into account, such as the amount of money the player is currently carrying, or whether jobs have already been applied to, or which cities and monuments have been visited and so on, one could envision the game just as a number of views that are linked together, similar to a PowerPoint presentation, hyper-linked HTML documents, or a bunch of paper sheets laid out on a table (with some rules on how to get to the next paper). By elaborating on this idea, we decided on the following conceptualisation:

The purpose of a "view" should be to gather a limited amount of data from the user. Treat each view as a single function. Give the functions the least possible data they need. A view is rendered when the function is called. Within the function, draw the user interface and process player input. Return from the function only the data which the view is supposed to collect from the user. Just before the function returns a value the render should shut down. After a view closes, the main game loop decides which view should be called next.
Take the airport view as an example. To use it when developing the game, the following code is all that is needed:

\[
\text{destination\_city\_id, ticket\_price} = \text{airport(city\_id)}
\]

The airport function takes an ID of a city (integer) as an argument, for example 14 for Stockholm. This represents the city in which the player is in right now. When called, the 3D interface shows and lets the player examine which cities he can travel to. The player can interact with the view by purchasing a ticket or leaving the airport (as shown in Figure 3). When the player has either purchased a ticket or left the airport the ID of the city he travelled to and the price of the ticket is returned from the function, and the airport view closes. In the code example above, these returned values are saved in appropriately named variables. Those values would then be used appropriately by the main loop, for example by calling other view functions. In this case they would edit the player state and set the new current city to the destination city, and the ticket price would be deducted from the money the player carries.

But how does the airport view know which cities are available for travel? And how does it know if a player can afford a ticket? The answers to those questions are simple; all views have reading access to the game data (the XML files as described in earlier sections) and the player state, although only certain parts of the main loop has writing access to them. Player state is in our game handled by using the singleton pattern.

4.6.3 Analysis

This design decision maps well to the 'interpreted educational game pattern' presented previously. Given information from the XML parser, one can just feed the relevant parameters from those files into the relevant functions, each time a new view is loaded.

We believe that this is a much simpler solution for handling 3D views than is the case in many games. For a small- to medium-sized game with a limited amount of views we think that this solution is sufficient. For larger sized games, one could discuss if game views should be treated in other ways.

Although using function-based views does not require the use of a singleton
pattern, this proved successful for our game in regards to the player state. For more general purposes, one should consider the strengths and drawbacks of this pattern\(^\text{25}\); we do not necessarily endorse it.

### 4.7 Design decision 3: Community created assets, open source, and no costly dependencies

#### 4.7.1 Problem

The game needs a lot of assets such as photographs and maps. The budget does not allow for much – if any – assets to be purchased. We need to strike a balance between assets created by us – the developers, and assets acquired from elsewhere. The project’s low budget also means that we cannot be reliant on expensive programs and dependencies.

#### 4.7.2 Design decision

We decided to not purchase any assets at all, and instead we made the decision to use assets freely available for commercial use online, provided one gives appropriate credit to the original creators.

For the city map views (Figure\(^\text{4}\)), we decided to use the open wiki world map OpenStreetMap\(^\text{36}\), with appropriate credit given to the creators, according to their terms of use, guidelines and license\(^\text{37}\). The library does not cost anything to use even for commercial projects, if one follows these terms.

We decided to use photos found in various online databases, licensed under Attribution 4.0 International (CC BY 4.0)\(^\text{38}\). This licence is free for commercial use, if appropriate credit is given to the original author. There are other versions of this licence, and several similar licenses appropriate for our use cases, but we decided to primarily use only one licence if possible. In cases where we could not find any photos appropriate for our needs under this primary license, only then would we deviate from this plan and use another license with similar properties. Metadata on the license we use for each respective photo and appropriate credit to the author, is saved in the XML knowledge repository (section\(^\text{4.5}\)) and is visibly displayed in-game along with the photograph, according to the conditions and terms of use.
Whenever possible – instead of using licensed images – we would use assets created by us, the developers. For example, the developers have previously visited cities such as Stockholm, Budapest and Paris. These cities are included in the game, thus photographs taken there by us could be used for those cities.

In a similar manner to what is describe above, we are only reliant on open source programs with very liberal licenses. We are not reliant on any costly dependencies.

4.7.3 Analysis

When developing a game such as this with assets and dependencies of this kind, it can be well worth the effort to consider using such material that is available online, with a license that meets the requirements of the product.

Which licenses that are best for a particular product varies from product to product, and one must be careful when deciding on which license to use. Care must be taken to follow all the terms and guidelines for that particular license. Preferably, the entity using these licenses should develop a thorough guideline document for how such material should be used internally. Fair use benefits all parties, and the costs can be incredibly high both monetarily and reputation wise, if the original authors/creators starts a legal dispute over the use of their original work.

Should the game be decently profitable, so that development costs are covered and some means can be invested in upcoming projects, we are considering to financially contribute a small percentage of the profits to a number of the dependencies and assets that the project were most reliant on. It is very important to give back to the community if possible, to allow open source and similar concepts to thrive.

One of the original ideas for this project was that we wanted to try to use as much material already available as possible. We knew that we did not have the means to develop much dependencies, or buy costly programs, or create a large number of assets. If it had not been for the amount of high-quality material like this created by other developers publicly available for free, this project would not have a high probability of completion. Indie projects such
as ours are highly dependent on such solutions. A key idea for this project was to plan to use such solutions to as high a degree as possible. As always, it all boils down to a debate on how resources should be spent.

The game’s 3D engine is developed by us. We also created all 3D models used in game. How well does this rhyme with the philosophy described above; to not re-invent the wheel, and focus resources on the parts that makes your product unique? Creating an entire 3D engine surely must be more expensive and time-consuming than using one of the numerous engines available? We would argue that this is not necessarily the case. Our 3D engine is very specialised and streamlined for our specific purposes. Also, our game is more similar to a fancy slide show presentation, rather than an AAA game with photo realistic graphics (see section 4.6). In fact, getting the 3D engine up and running for our specific game was not a very resource-consuming part of development. System design, patterns, work-flows and strategies were all more time-consuming tasks. The most time-consuming developments task of all thus far, but also in the future according to our time estimations, is asset gathering and creation.

This whole section on this design decision probably does not add much to the debate; it is not new information that many benefits can be had using programs, assets and libraries available online when creating a new product or work of art. Still, we felt that this design decision was one of the most important ones for our game, and thus worthy to be included, evaluated and discussed in this case study.
5 Conclusions and discussion

5.1 Conclusions

The Research Question for this study is:

**RQ:** Which design decisions contribute to effective indie development of educational video games?

From the development process of our game, we could identify the following three design decisions with this purpose. It is our belief that the results generalise to other educational video game indie development situations:

1. Use of XML files validated against XSD files for the data repository part of the game. The game is fully defined by such XML files, a component repository and an interpreter. The interpreter interprets valid XML files and produces a fully functioning game from them and the component repository. All of this is according to the 'interpreted educational game pattern' we define which is summarised in Figure 9.

2. We propose that educational video games can be structured by conceptualising the game into a few different 'views', a view being a tangible part of the game from the user’s point of view which takes an input from the user and produces an output. Such views can preferably be organised in such a way that a single view is represented by a single function at code level (for emphasis; one view, one function). This conceptualisation works well with the XML and 'interpreted educational game' pattern we proposed, as the various functions can simply be given the relevant values from the XML as input.

3. Finally, we solidify the case for utilising Free and Open Source (FOSS) libraries and works of art in the development process, with carefully selected licenses appropriate for the specific development situation at hand.

One of the aims of this study is for the reader to be encouraged to use one or more of these three strategies presented, in his or her own educational video game development projects. Although the design decisions presented here
may or may not be of importance to such projects, we have established some concrete guidelines to discuss when developing games of this kind.

We also hope that the study contributes to facilitation of educational video games. Maybe more independent actors and small companies are encouraged to invest in projects like these, by learning from the general strategies and concrete solutions presented herein.

5.2 Modifications and improvements on previous research

The ’interpreted educational game pattern’ shown in Figure 9 is a modified design from those which other authors proposes and we illustrate in figures 1 and 2 [19] [20]. We believe that we have improved on those designs in two senses; one being that we have introduced the validation part explicitly in the process and made sure that the developer who writes the interpreter is responsible for defining valid file formats which the interpreter understands. This way, if valid files are given as input to the interpreter, the interpreter need not be concerned with error handling and can produce a working game without exception. This both speeds up the process and makes it less error-prone. Also, it enables the other improvement; we have made sure that the game designer and the expert (and other roles) can work with the same set of files. Then, different wills, misunderstandings or disagreements between the roles should not result in a broken or erroneous game, as long as only valid files are fed to the interpreter.

Regarding the functional game view pattern we introduce, we feel that we have taken some of the better parts of the functional paradigm and combined them with the object-oriented paradigm. Nelson’s functional game framework Easel [21] and the fully functional 3D game by Cheong [22] hints that this is indeed possible to do, and it may have some advantages over the object-oriented pattern in some cases. Although we were inspired by decisions such as those made by these authors, we did not make our game fully functional. Instead, we think that combining the functional pattern with the object-oriented pattern can make way for faster development of games, if the decisions are well thought through and can be conceptualised in a nice way, such as we think is the case by our functional game view pattern.
For the final design decision – to use FOSS libraries and works of art in the development process with appropriate licenses – we can be self-critical and state that we feel like it does not bring many new insights to the development processes of indie games. Decisions such as these, aiming to keep costs of such projects low [16][17], have long been used in practice [23][24]. If anything, we feel that including this design decision in this study is important as it was indeed one of the most important decisions made during development for our game to have a higher chance of success. We hope to have had solidified that this is indeed likely to be the case for many indie development processes.

5.3 Further development

The game is as of time of writing at a pre-alpha stage. The goal is for a full release to have been finished and released on several content delivery systems, Steam included, sometime in 2020. The working title 'Voyager' is then likely to have been changed to a permanent title.

5.4 Further research and other potential approaches

The method chosen for examining which design decisions contribute to effective development of educational video games has striven to answer the question satisfactorily, even if there certainly are other ways to explore the same area [39].

This study is a case study and by design quite wide in scope, to enable the possibility of finding design decisions of importance. Our approach features elements of both qualitative as well as declarative research. A narrower study, focusing on just one of these methods and with a more specific research question, may be able to find results of another kind. One of the primary purposes of this study is to identify several design decisions of importance contributing to effective development of educational video games. Another, more specific, study could explore individual design decisions and the reasons to why these are of importance, which could yield results of a more explanatory nature.
5.5 Ethical considerations

Below, we briefly introduce some ethical considerations and questions of potential importance for this study. Exploring these questions further is a task left for the reader:

- As the game to be developed belongs in the educational genre, an ethical question that comes to mind is the quality and content of the educational material that is intended to be taught. Who decides which content goes into the game? In this case it is the people developing the game at Toleap Consulting AB. An ethical consideration is what responsibility we have for this educational material.

- How much could we morally strive to steer the landscape of video games towards education and edutainment? Is it worth the effort?

- Is it ethical to encourage screen time and playing a video game which may be addictive? Is a game being ‘educational’ enough for this to be justifiable? Is there an ethical difference between pushing an educational agenda rather than an entertainment agenda?
6 References


[26] “About the Licenses - Creative Commons.” https://creativecommons.org/licenses/


[38] “Creative Commons - Attribution 4.0 International.” https://creativecommons.org/licenses/by/4.0/