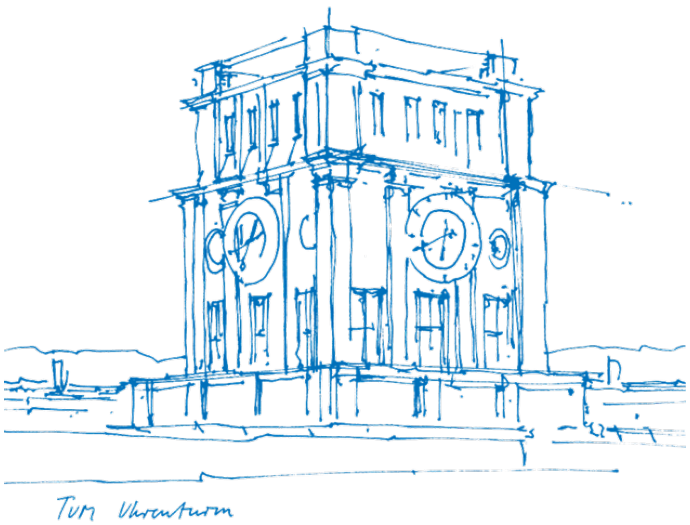
Please adjust group name in tumuser.sty.
School of Computation, Information and Technology
Technical University of Munich

Methods and Tools for Gamespace Engineering: Designing a Web-based Application to Assist the Planning Process of Gamespaces

Methoden und Werkzeuge für Gamespace-Engineering: Design einer webbasierten Anwendung zur Unterstützung des Planungsprozesses von Gamespaces

Viola Stumpf



Methods and Tools for Gamespace Engineering: Designing a Web-based Application to Assist the Planning Process of Gamespaces

Methoden und Werkzeuge für Gamespace-Engineering: Design einer webbasierten Anwendung zur Unterstützung des Planungsprozesses von Gamespaces

Viola Stumpf

Please adjust group name in tumuser.sty.
School of Computation, Information and Technology
Technical University of Munich

Methods and Tools for Gamespace Engineering: Designing a Web-based Application to Assist the Planning Process of Gamespaces

Methoden und Werkzeuge für Gamespace-Engineering: Design einer webbasierten Anwendung zur Unterstützung des Planungsprozesses von Gamespaces

Viola Stumpf

Thesis for the attainment of the academic degree

Master of Science (M.Sc.)

at the School of Computation, Information and Technology of the Technical University of Munich.

Examiner:
Prof. Dr. Gudrun Klinker

Supervisor:
Daniel Dyrda

Submitted:
Munich, 02.11.2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 02.11.2023 Viola Stumpf

Abstract

Digital Game Levels are a powerful tool to create exciting, immersive experiences for the player and as an integral part of the game, they contribute well to the game's overall design, aesthetic and narrative. It is important that they are well-planned and structured. However, as games grow in complexity and challenge, level designers lack the time to apply theoretical insights from research to practical level design. To speed up the process of searching through existing literature, this thesis proposes an integrated knowledge platform that summarizes the tools and methods from scientific research to a single shared vocabulary.

Despite the growing amount of research in the field of level design, there is currently no shared knowledge tool capable of seamlessly integrating individual research topologies, emphasizing interconnections, and fostering knowledge scalability. This thesis provides a solution in form of a digital garden - a non-linear collection of syntactically and semantically connected nodes, enhancing viewer engagement. This digital garden, presented through a user-friendly website, serves as a repository for a diverse set of level design tools and methods. The insights gained from the existing research are categorized in methods, tools, processes, roles, and artifacts, further highlighting the interrelationships and complexity of the garden with a graphical representation.

Rather than competing with existing solutions, this project seeks to complement and extend them. In the future, it aspires to evolve into a collaborative platform, accommodating contributions from multiple collaborators, and further enriching the shared knowledge pool.

Contents

1 Introduction

1.1 Motivation

With the surging popularity of games and recent technology advancements, game design has grown increasingly complex [162] [138]. Comparing the early generation of video games, such as "Tennis for Two" (1958) [163], "Spacewar!" (1962) [145] or "Pong" (1972) [103], with contemporary titles like "Baldur's Gate 3" (2023) [147], "God of War" (2018) [146] or "Detroit: Become Human" (2018) [80], the technical requirements and expectations of video game narratives skyrocketed within a short span of time, forcing game developers to spend more time and resources on game development [108]. For instance, games like "The Last Guardian" (2016) took nine years to develop due to their complexity and lack of scaling with small teams. "Diablo 3" (2012) [85] was published after 11 years of development, battling with the pressure of surpassing the success of its predecessor [6].

Conversely, a growth in research can be observed regarding game development, with renowned books like "Rules of Play: Game Design Fundamentals" [151], "The Art of Game Design: A Book of Lenses" [137] or "Game Design Workshop: Designing, Prototyping, & Playtesting Games" [90], simplifying the process of game design and explaining its theories and best practices. However, there is still a gap between the theoretical research and its practical application to concrete game design [160]. This gap further increases the complexity of the field and challenge for game designers, who often lack the time to read through volumes of information in these books and apply them to their game projects. As the game industry emphasizes efficiency and time-saving [58], it becomes imperative to find a solution that provides game developers with the necessary knowledge they need to create and enhance their game. The knowledge can range from abstract information like viewing the game development process through different lenses [137] to concrete methods like a mood board, a pacing diagram or reward schedule [83] [35] [39].

1.2 Problem Description

This thesis focuses on level design, as it is a part of game design [110]. Regarding the context provided above, it follows the assumption that level designers, equal to game designers, need to save time to focus on the concrete development of the game rather than on researching the different tools and methods relevant for their application. [160] [58] Many research papers and books touch upon different fields of game development and it can be tedious to filter the relevant information gained from these sources, particularly for game designers who lack the time to improve their methodologies and gain inspiration from theoretical findings. It becomes pivotal to filter the large pool of knowledge to pinpoint important research pieces concerning the methodologies and tools relevant to level design. However, given its novelty as a research field [144], are there even enough researches made that focus on concrete methods and tools for level design?

Assuming that there is substantial research in this field, the challenge lies in filtering the collected knowledge. [160] To properly take valuable findings out of the research, the reader has to find similarities connecting the methods and tools. How can the methods and tools introduced by one research be connected to the methods and tools of another research? Are there noticeable similarities, and can certain methods and tools be clustered together? What is the significance of one method if it is only backed up by one research, while other methods appear in multiple papers?

Moreover, authors might employ distinct terminologies to describe identical methods and tools, defending their usage of words. For instance one author might characterize a pacing graph as a tool [153] while another

author labels it as a method [35]. This prompts the question: How can methods be distinguished from tools? Is there a common usage of terminology or is there a need to create a shared terminology?

With the information gained from the research, there is a strong need to make the information available to the peer group of recipients, the level designers. How can the collected knowledge pool be published and made accessible to level designers and other interested readers? What limitations does a publication have regarding its accessibility in different locations around the world, and is it adequate to present the knowledge pool in English alone? While English is the predominant global language, the application might reach more developers if it is additionally translated to other dominant languages like Chinese, Spanish and French.

More and more research regarding game and level design is published, with new insights yet to be discovered [86] [160] [68]. Additionally, there might be smaller research already published but lacking in its popularity, hence not being included in the knowledge collection. To address these researches in the future, it is important for the application to be scalable. Rather than linearly outlining the application and predetermining the flow of reading through the provided information, it may be more interesting to focus on the connection of the knowledge pieces instead. By refraining from a clear beginning and end, the character of the overall research may be displayed, modeling the growing non-linear pool of knowledge into the application that is constantly modified. It is intriguing to learn about the applicability of this approach.

1.2.1 Research Question

This thesis tries to answer the above questions and find a solution that bridges the theoretical, academical aspects of level design with practical application. It aims to relieve level designers by gathering information from existing research and analyzing it for shared and distinctive methods and tools used, providing a shared vocabulary and terminology. Furthermore, it aims to create a shared online platform, accessible to level designers and other individuals interested in the subject matter. Finally, it focuses on the scalability of the information, exploring the concept of depicting the information in a digital garden rather than linear wikipedia-styled entries. With these efforts, the thesis ultimately tries to answer the question:

How can methods and tools for level design be summarized to a shared vocabulary and effectively collected on a shared online platform, highlighting the interconnection of its entries and the scalability of the knowledge pool?

1.3 Content of this thesis

The process of answering the question commences by presenting three existing researches closely related to the proposed solution. The related work section introduces two papers and one web-based application that served as inspiration for this thesis, pointing out similarities and differences between them and the approach of this thesis [160] [58] [21].

The next chapter covers existing research fields that influenced this thesis, introducing the most pivotal concepts to understand the scope of level design that is covered by this thesis and extends beyond it [153] [137] [151]. It is also supposed to underline the motivation behind this thesis, stating that there is still much to be explored and integrated into the methods and tools.

Recognizing the importance to close the gap between the theoretical foundations and the practical implementation [160], the next chapter delves into the driving forces behind this thesis. It argues why it is crucial to create a shared platform for level design despite the already existing efforts made in the related work. It covers the importance of collecting tools and methods from existing research, organizing the information, publishing and sharing it online. Finally, it offers insights into the initial attempts to create a web-based application and argues in favor of discarding these approaches.

The next chapter deals with the implementation of this thesis's project. It explains the concept behind the project - the creation of a digital garden [7] - and thoroughly describes the process of collecting and

organizing information gained from research, fostering the garden and publishing the results. Additionally, it introduces the concept of collaboration, an addition to enhance the project's scalability.

The results of the thesis are explained in the following chapter, stating what the user can do with the project and which features are included.

The discussion in the next chapter revisits the questions prompted above and critically reviews to what extend this thesis provides a solution to the questions. It further includes the significance of the concept in hindsight to the research question and how the project could be improved to tackle the problem more efficiently.

The last chapter concludes the thesis by summarizing the findings regarding the research question, hinting at future work.

At the end of the thesis, the reader finds important attachments and all references used in this thesis.

2 Related Work

Before diving into the content of this master's thesis, it is crucial to acknowledge that there are already existing solutions addressing the primary objective of this project. Specifically, there are three sources that influenced the development of this thesis. [160] [58] [21] Although all three sources aim to provide the reader with the same kind of information, the processes towards achieving the solutions differ significantly. While they research the same field they do not overlap entirely, merely touching upon similar fields and purposes while proposing different methodologies.

First, this chapter will introduce language mapping, a procedure that aims to connect different languages used in game design and find correlations between methods and tools of different sources [160] [58]. The second part of this chapter will deal with the Level Design Book, a wiki that is heavily used among level and game designers [21]. Both approaches are important to mention because they tackle the same goal with different methodologies. This master thesis overlaps in the intentions and some of the methodologies but also differs in important aspects that will be explained further.

2.1 Language Mapping

The field of game design draws inspiration from various domains, such as pattern languages [57], architecture [153], media studies [106] or film & movie making [105]. Among other fields, software development plays a big role, hence it significantly influences the methods and tools used in game development [160, p.4]. Unlike more established disciplines like architecture, software development, or film making, game design is a relatively new field of research [58]. As such, its methods are not only borrowed and inspired by the other fields but they lack in formalization required for effective game design [160] [58]. There is a need to formalize and structure every successful method and tool so far. In particular, two papers that will be introduced in this chapter deal with the question whether it is possible to create formal languages that accurately describe specific methods within game design and map them to other areas of game design. [160] [58] Both papers identify existing methods and tools and look for similarities, relations and connections to construct a formal language capable to support future work and fields. Having the same goal of increasing the quality and the efficiency of game development, they collect a range of design tools and methods from various sources and build separate visual language models to analyze the sources formally, paving the way for future applications. Both researches also recognize their role as bridges rather than as a solution, calling for future research to be made with their language mapping as support.

The subsequent sections will discuss the extend of how these papers serve as a reference to this thesis and in which aspects they differentiate from the goal of this thesis, highlighting why research is still valid and needed in this field.

2.1.1 Languages of Games and Play: A Systematic Mapping Study

The first paper, called "Languages of Games and Play: A Systematic Mapping Study" by Riemer van Rozen, explores the role of languages, notations, patterns, and tools in game development. [160] It underscores that game development is a practice that requires a fast decision process from both designers and great communication between the team which, according to the paper, generally include designers, engineers and artists [160, p.3]. Improvements to the game depend on the insight the designers gain from the players' gameplay experiences while changes to the game become more and more difficult with time progressing fast and decisions becoming fixed over time [160, p.4]. This highlights the designer's importance to the game's quality and the significance of establishing a common language for efficient communication between designers [160, p.4].

Van Rozen states two key research hypotheses [160, 2.2, p.4]:

- Languages, structured notations, patterns and tools can offer designers and developers theoretical foundations, systematic techniques and practical solutions to enhance game quality and play.
- "Software" languages (and specifically domain-specific languages) can help to automate and speed-up game design processes.

By researching and analyzing existing languages in game design, similarities can be found and formalised. This paper specifically picks solutions for domain-specific languages which, originating from software engineering, are defined as follows:

"A Domain-Specific Language is a programming language or executable specification language that offers, through appropriate abstractions and notations, expressive power focussed on, and usually restricted to, a particular problem domain." [157]

Research has demonstrated that domain-specific languages enhance the communication among domain experts [156] [157] [152], making them a promising candidate for boosting the quality of games. The research methodology involved a vast range of scientific sources found on Google Scholar to have varied material that authentically depict game design approaches [160, p.6]. Research areas centered on inquiries such as "where authors have published" [160, p.6], characterizing proposed languages concerning their objectives, structure, applications, and implementation, deployment and availability. Each publication was mapped according to a schema that includes a description of the research topic, the design of the proposed language, the implementation, validation and the availability [Fig. 2.1]:

Facet	Element	Description
Brief description	Problem	Problem statement, game topic
	Objectives	Goals the authors formulate, challenges addressed
	Solution	Solutions proposed, claims on language application and scope, game genre
	Category	Solution category and application area (Table 3)
Design	Pattern	Language design pattern (Table 4)
	Features	Language features (elements shown in Table 5)
	Examples	Snippets of text, code, diagrams or models
Implementation		How is the language implemented, e.g., interpreter, compiler (these details are usually not described)
Validation	Products	Games, prototypes and show cases that are constructed using the language
Availability	Web site	URL of a web site providing information on the language, notation or toolset
	Distribution	URL of a binary distribution or source code repository
	Source license	License under which the source code is available

Figure 2.1 Language Facets to summarize and analyze the collected game design languages. [160, p.8]

Moreover, each discovered language underwent categorization according to its objectives, design patterns and features as shown in three tables 2.2, 2.3 and 2.4.

The primary categories encompassed in the publications are "software engineering, artificial intelligence, humanities, social sciences, education, and game studies" [160, p.11]. Covering these categories enabled Rozen (2020) to filter out fourteen spectres of domain-specific languages that were found and developed in their research. These domains are defined as follows:

1. **Ontologies and Typologies**: Correlated concepts via diagrams and tables [160, p.17]
2. **Pattern languages and Design Patterns**: Describing gameplay goals and best practices in game design [57]. Analyzes gameplay effects and learns from player experiences [160, p.18]. This spectre

Dimension	Category	Description of intent
Scope	Application-specific	Solution is specific for a game or application
	Genre-specific	Solution that is reusable for a specific game genre
	Generic	Generic solution or separated concern
Solution	Framework	Analysis or mental framework for studying, understanding, comparing, categorizing games that does not directly support game development, e.g., ontologies, design patterns, or simulations
	Tool	Authoring tool that facilitates creating a game's parts as models or programs for design or development , e.g., visual environments, programming languages or DSLs
	Engine	Game engine, reusable building block or software library that integrates models fully into game software
Area	Research	Research vehicle primarily intended for performing research in a specific area
	Educative	Platform primarily intended for teaching a subject to a group of people or example meant to illustrate, educate or inform
	Practice	Solution primarily intended for practitioners, supporting game design or game development

Figure 2.2 Language Objectives – solution scope, category and application area [160, p.9]

Dimension	Category	Description
Reuse	Piggyback	Partially uses an existing language, a form of exploitation
	Specialization	Restricts an existing language, a form of exploitation
	Extension	Extends an existing language, a form of exploitation
	Invention	Designs a language from scratch without language reuse
Description	Formal	Formally describes a language using an existing semantics definition method such as attribute grammars, rewrite rules, or abstract state machines
	Informal	Informally explains a language without formal methods

Figure 2.3 Language Design Patterns [118]

subdivides into understanding and analyzing gameplay, such as the gameplay goals, and predicting game changes by informed decision making, measuring the progress and experience in games.

3. **Applied Game Design**: Also known as serious game design, these languages deal with cognitive analysis and how players can be educated through games [160, p.18].

4. **Game Mechanics**: Design patterns that create, analyse and understand game mechanics, such as Petri nets [160, p.19] [127].

5. **Virtual Worlds and Levels**: Tools and methods supporting the creation of worlds, e.g creating 3D worlds with "SketchaWorld" [160, p.20] [140].

6. **Behaviors**: Describing the behavior of Non-Player characters (NPCs) that occurs in AI-related research, e.g. analyzing behavior trees [160, p.20] [71].

7. **Narratives and Storytelling**: Languages that support storytelling in games, e.g. storyboards [160, p.21] [129].

8. **Analytics and Metrics**: Utilizing machine learning to analyze data, using metrics to test hypotheses and evaluate the gameplay quality of the designer's game, e.g. use PlaySpecs that analyzes "sequences of player actions" [160, p.21] [107].

9. **Education**: Focusing on languages specialized in teaching and guiding the players to learn specific content, languages that have an educational purpose [160, p.22].

Dimension	Feature	Description
Notation	Textual	The language has a textual notation
	Visual	The language has a visual notation
Elements	Scopes	Scopes and bounds may be used to separate elements and limit their valid context
	Conditionality	Conditionality features enable or disable other language elements or events
	Recurrence	Recurrence features are elements that can happen again, e.g., in iterations
	Modularity	Modularity features enable composition and/or reuse of language elements
	Domain-specific	Domain-specific language features may be especially created for a purpose that is specific or unique to the subject matter
User Interface	Feedback	Provides a feedback feature enabling understanding
	Mixed-initiative	Provides feedback & feed-forward, alternating between user input and computer generated alternatives
	Live	Provides immediate and continuous feedback, e.g., a live programming environment

Figure 2.4 Language Features, not mutually exclusive [160, p.10]

10. **Gamification**: Languages that use game concepts in other fields like marketing [160, p.22].
11. **General Game Playing**: Languages that study AI algorithms capable of playing a wide scope of games, e.g. AI that plays board games [160, p.23] [133].
12. **Script and Programming**: Programming languages used in game development, e.g. Python [160, p.24] [159].
13. **Model-driven Engineering**: Languages that use models like Unified Modeling Language (UML) diagrams and statecharts to describe generic and abstract concepts that are to be implemented as source code [160, p.25] [132].
14. **Metaprogramming**: Compilers and interpreters that "read and transform the source code" [160, p.25] to build the game.

As an example of one application of the model, the language "Game Mechanics"[160, p.19] is called "Game-o-Matic" [154] falls in the fifth category "Game Mechanics" and is described in the figure below [Fig.2.5]:

Language 28 ***Game-o-Matic*** *generic / tool / practice*

Treanor et al. propose generating arcade-style videogames that represent ideas with so-called *micro-rhetorics*. Micro-rhetorics are parameterized structures with a unique id, a verb and entity roles (parameters). For instance, "`A avoids B`" consists of a subject A, a predicate B and a verb avoids. For each verb, Game-o-Matic randomly selects a representative micro-rhetoric from its library that form partial game descriptions. These are completed with recipes that modify the rules and completes the game's mechanics, adding win and lose conditions and concrete structures for player interaction. Proceduralist Readings [258] is a related framework (see Language 32).

publication	query	publication type	research category	note
[260]	~~language~~	conference paper	proposal of solution	
[259]	~~language~~	workshop paper	proposal of solution	

Figure 2.5 A mapping example of the language "Game-o-Matic" as proposed in the paper "Languages of Games and Play: A systematic Mapping Study" [160, p.50]

As shown above, the language is given a number, a title and a categorization based on the above criteria. The description is the key part of the language. Below the description, the relevant literature is listed.

This structure, applied to each language encountered during Rozen's research (2020), provides a comprehensive overview of the language and facilitates comparisons between them.

Context: Methods and Tools for Gamespace Engineering

This modeling approach shares many similarities with the methodology used in this thesis. Primarily, they share a common goal: Identifying methods and tools relevant in game and level development and accelerating the game design process. Both approaches also start with the collection of material to extract the knowledge from. The schema that aims to analyze the structure of the language aligns with the structure used to describe the methods in this thesis, particularly in the fields of design, implementation and the applicability of the proposed language or method. This demonstrates that both researches pay attention to similar fields, albeit differing in their outcomes.

The most important distinction between these two researches is their approach to specificity and formality. While Rozen's approach (2020) involves collecting a big amount of research to find similarities and categorizing them into domains, this thesis focuses on collecting general solutions applicable in many domains. Methods and tools are not categorized in domains but instead analyzed by their practicability in real-life level design scenarios. There are concrete methods that overlap in both researches, such as state charts, but Rozen (2020) labels the concrete methods in a category and positions them into a bigger context to formulate fields for future research [160, p.86].
It is important to observe that the descriptions of the languages are formulated rather abstract refraining from providing concrete details about the implementation, input, output, applicability and direct usages in game development roles. The languages rather focus on providing the sources from which the language was formulated while Rozen (2020) calls upon future research to delve deeper and concretize these languages [160, p.32]. While the paper successfully identifies formal languages for game design, this thesis also focuses on level design. This calls upon a more direct approach because level design is just a part of game design and therefore covers a smaller spectrum [110].

While this paper serves as related work, it does not fully research the same field and research question. This paper is about mapping languages to game design, whereas this thesis focuses on providing level designers with a database full of concrete methods and tools ready for their practical application in level design.

2.1.2 A Systematic Review of Game Design Methods and Tools

The second research paper, "A Systematic Review of Game Design Methods and Tools" written by Marcos Silvano Orita Almeida and Flávio Soares Corrêa da Silva (2013), presents a systematic exploration of the state of game design methods and tools. [58] By trying to formalize the methods and tools in game design, they define two primary approaches: The development of a shared design vocabulary and the creation of a game design modeling language.
The authors state their motivation for their research, emphasizing that the existing game design methods and tools lack in formalization [58, p.18]. They highlight issues such as the use of prototypes, often handled by developers and engineers rather than game designers, leading to communication gaps between the team and increasing the production cost of games [58, p.19] [121]. Another critique is the use of design documents, which often lack in standardization and formalization [58, p.19], highlighting the need of a shared vocabulary and concrete rules game designers can follow to speed up the design process.
Almeida and Da Silva (2012) systemize the efforts of other researchers, such as Burkart's "Game Design Lexicon" (2005) [67] or Björk et al.'s "Game Design Patterns [102], and map them to other tools and methods for game design. Their mapping structure takes the form of the map [Fig.2.6].

Chapter 4 discusses the approaches taken by other authors, such as Game Design Patterns [102], to find common vocabulary used in game design [58, p.21]. Chapter 5 deals with the approach of researchers to define visual language patterns [58, p.24]. Both approaches subdivide into smaller domains. Shared vocabulary is identified and collected in game concepts [58, p.21], design guidelines, game taxonomies or a dictionary [58, p.23]. Visual languages can be divided into Design Visual Modeling [58, p.24] and Mechanics Modeling [58, p.25]. These two distinctions are summarized by the headline "Towards Game

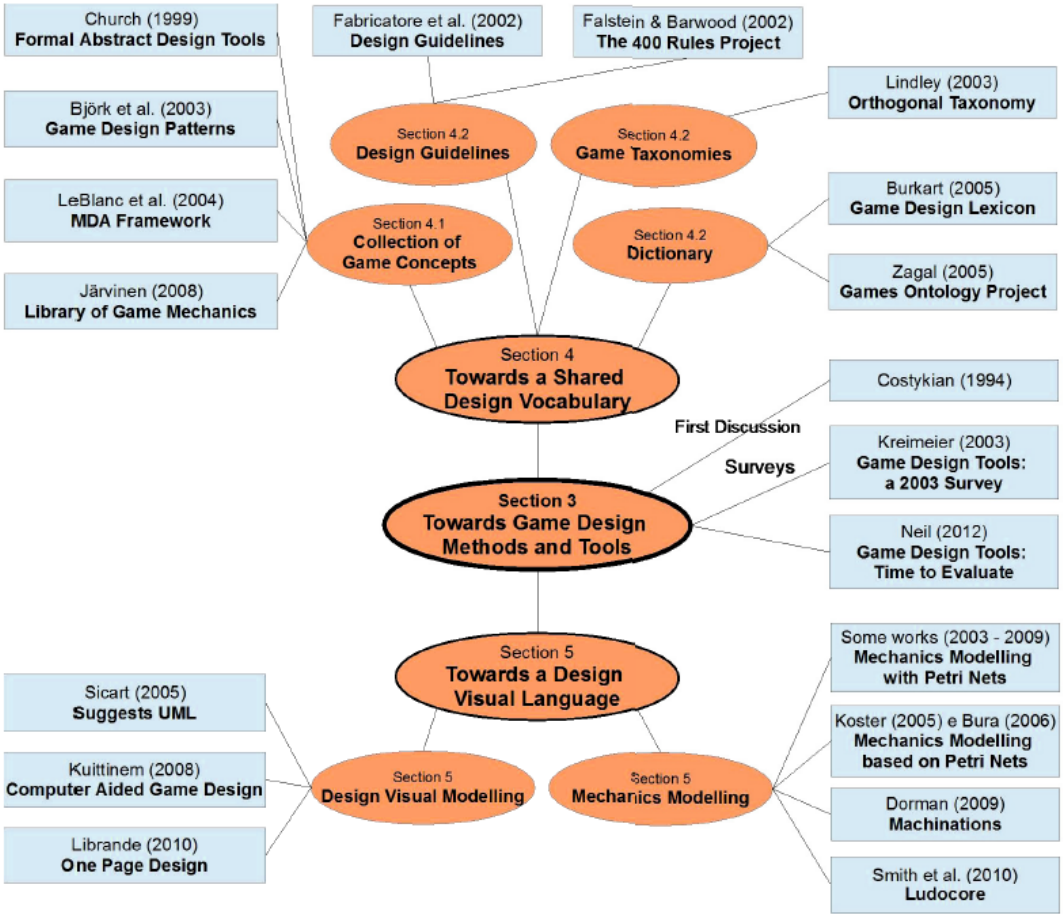


Figure 2.6 State of the Art Map of Game Design Methods and Tools [58, p.20]

Design Methods and Tools" [see Fig.2.6] which also covers first discussions about the lack of formal tool descriptions, surveys that support the findings of the study [77] [13].

In conclusion, this paper primarily underscores the lack of a solution rather than providing one. It aims to draw attention to the importance of more formalized methods that are attractive for game designers backed up by current research. The paper categorizes the research into two sub-categories that need to be solved on their own, providing the reader with critique towards the current forms and patterns in the vocabulary and visual language of game design methods and tools. While it does not provide specific solutions, it plays a valuable role in highlighting the importance of standardized tools and methods for game designers.

In relation to this thesis, both share their vision and motivation to provide a comprehensive overview of the tools and methods in gamespace engineering. Again, this paper focuses on game design and covers a broader spectre compared to this thesis's focus on level design. However, since Almeida and da Silva (2013) only provide a theoretical mapping of already existing critiques and solutions, they lack in their applicability. In contrast, this thesis takes a hands-on approach that can be accessed by level designers directly, addressing the absence of formalization in certain methods such as game design documents. Furthermore, it provides a solution to the lack of formalization even if it is not scientifically formal. It aims to help level designers directly and pragmatically in their work.

2.2 Level Design Book

The "Level Design Book" represents a distinct approach compared to the previously discussed research papers. [21] It aligns with the motivation of this thesis, providing a source that collects level design tech-

niques for designers to read through and apply to their projects. This website-based approach, accessible on https://book.leveldesignbook.com/, is designed to gather and present level design knowledge for 3D video games in a practical, accessible, and up-to-date manner. It targets designers of all experience levels and works with various game engines. Its official description on the website is:

"The Level Design Book gathers level design knowledge for 3D video games in an approachable, up-to-date, and critical way. It is for designers of all experience levels and game engines." [21]

While the Level Design Book is still under construction and not officially published, it is already available for use with its full scope.

Upon visiting the homepage of the level design book as shown in Fig.2.7, readers are presented with five topics that summarize the book's content.

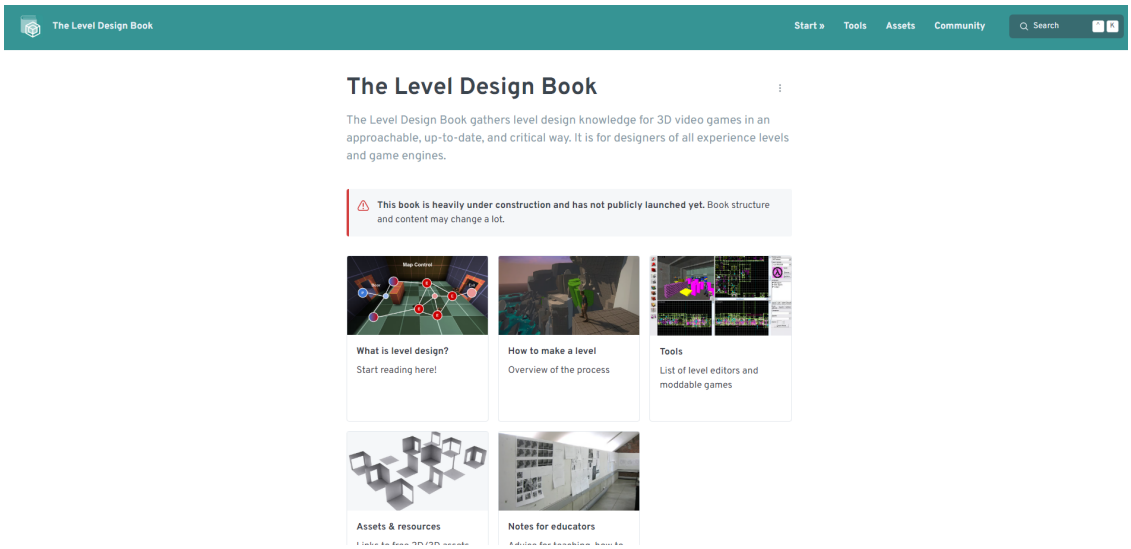


Figure 2.7 The Start Page of the Level Design Book

These contents include an introduction to level design, the process of level design, tools used in level design, assets and resources for level design, and notes for educators interested in teaching level design. Additionally, visitors can use the toolbar on the top of the page to search for specific keywords in the book or access a "Community" section that introduces online communities relevant to level design, ranging from general communities to specific writing or environment art communities. New visitors are encouraged to start with the introduction to level design.

Even though the start page offers several key pages to dig into, the level design book is meant to be read in a linear fashion. This is already hinted at on the start page, as it is recommended to start reading the "What is level design" entry first. When clicking on said page, the user is redirected to the beginning of the book, which provides the authors' perspective on level design. One can read through the whole entry or click the chapter navigation on the right side, similar to the navigation in PDF readers. The content of the level design book is presented with a sidebar that highlights the recommended flow when reading the articles. This supports the linear concept, starting with the page "What is level design" as the very top entry and then continuing with the contents of "Books" that each cover a distinct topic. The first book describes the process of level design with several subtopics. The second book describes the culture of level design including pages that are still in progress. The third book deals with several level studies that the users can contribute to, and the fourth and final book describes the notes for educators which are also linked at the start page. The layout of the navigation bar can be observed in Fig.2.8.

In the appendix, the last section to read through, the book covers tools, assets, communities and information about the authors.

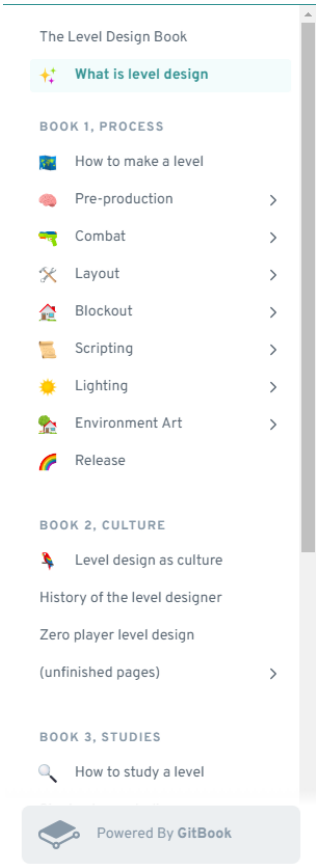


Figure 2.8 The Navigation Bar at the left side of the Level Design Book

After reading the content of the page, readers are provided with further steps they can take to deepen their knowledge or better understand the level design process. This information is written down in the "Now what" [Fig.2.9] section at the bottom of the page that mostly refers to the pages following the current page, though it can also contain advise and further tips on the next steps.

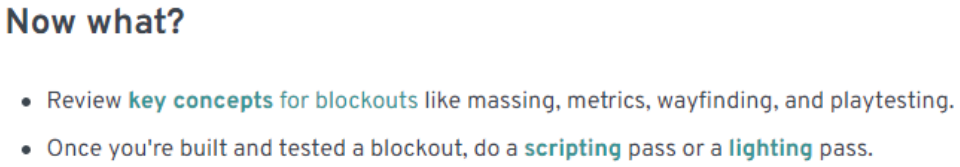


Figure 2.9 A "Now What" section provided on the blog entry about "Blockout", depicted in the Level Design Book

Some pages further contain a block called "Further reading" which provides users with relevant external sources to deepen their knowledge [Fig.2.10].

To aid understanding, the pages also come with images which illustrate the topic [Fig.2.11]. Their sources are named in the footer of every image each.

The level design book covers many processes and methods that are also covered in this thesis, including pacing, combat design, massing and level parti. During the creation of this thesis the level design book was occasionally used as a reference and inspiration because it explains its content in great detail, is easy to understand and provides examples to its processes and methods. In terms of collecting methods used in level design and providing a shared point that can be used in practice, both products share similarities. In contrast to the level mapping approaches evaluated above, neither the Level Design Book nor this thesis try to find formalization in the language for level or game design; instead, they offer a practical solution that

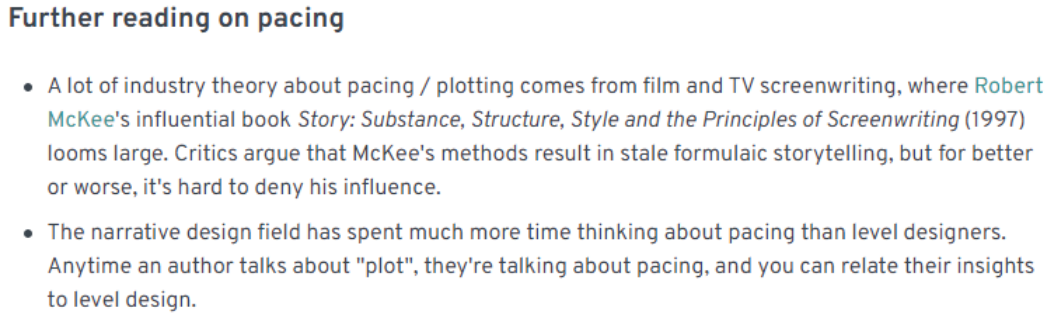


Figure 2.10 Further Reading on "Pacing", depicted in the Level Design Book

can be used by designers, using informal language to enhance readability. Both approaches are accessible online, broadening their impact range because everyone with a link can access the websites.

However, the two approaches do not overlap in every aspect. The most prominent difference is the layout of the solution. The level design book excels in its linearity [Fig.2.8], transforming the website into a book that can be read from start to finish. In contrast, this thesis's project focuses on the bidirectional connections between the separate entries. It creates a net of knowledge that becomes more complex with each node added. There is no specific start and end point; instead, there are central nodes that users can return to after exploring the knowledge pool. Additionally, to maintain its linear character, there is no next slide and previous slide linked on the content pages. Rather than building a Wikipedia-like resource that users can read in one go, this thesis encourages the reader to explore the collected knowledge.

Furthermore, the level design book contains references at the bottom of the page but no direct citations, whereas the project of this thesis includes direct references within the content of the pages. The language used in the level design book is strongly informal, letting the opinions of the authors shine through. Opposed to that, the project of this thesis tries to maintain a certain neutrality and caution against being influenced by a bias. The only biases have been adapted from literature or the level design book itself. Using bullet points instead of full sentences also takes away the wikipedia-like character and feels more like notes that have been taken, enhancing readability and fostering a casual atmosphere.

Rather than picturing both projects as competitors, it is advisable to see them as extensions from each other that can be used simultaneously because they complement each other and add to each other's information. The last update to the level design book was made nine months ago as of today [21], which still makes it an unfinished project with more potential. The gaps that it leaves might be filled by this thesis's project.

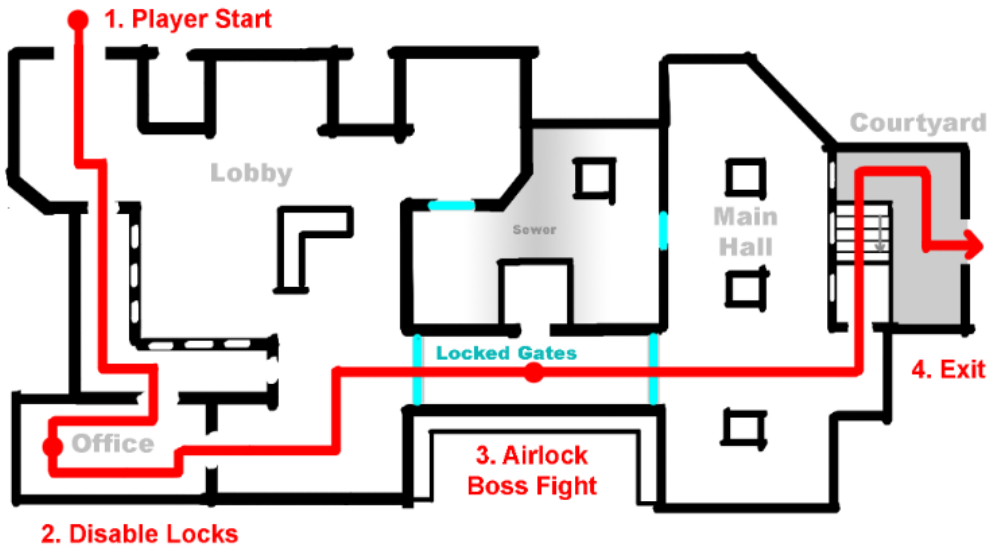
Critical path

The **critical path** (or "golden path") is the minimum / main path to complete a single player level.

More generally, it represents an *idealized player flow* that highlights the most important ("critical") parts of the level and mandatory game content that you want every player to experience.

Single player layout drawings usually need some sort of labeled critical path. Sketch, highlight, or mark the critical path in the drawing -- or if the drawing is too complicated, at least mark the player start, exits, and labeled points of interest.

For more on designing main routes for single player levels, see Critical path.



example level layout drawing with marked critical path and numbered gameplay beats

Figure 2.11 One image illustrating the critical path method in the "Flow" entry with its description, depicted in the Level Design Book

3 Background

To fully understand the importance of this thesis, it is essential to comprehend the contextual landscape of this thesis. There are many fields of researches that influenced the making of this thesis. This chapter provides the requisite knowledge to understand what drives this thesis. While level design serves as a pivotal influence, a foundation in technical background is also necessary. In the end of this chapter, the reader is supposed to not only understand the technical foundations of level design but also extend their understanding to emerging technologies that hold the potential to shape the future of level creation.

3.1 Level Design

Level Design is one of the pillars of Game Design, providing an implementation of the game's fundamental core concepts [134]. There are many approaches defining Level Design, each emphasizing a different focus and role it plays in the gaming experience. For instance, Totten (2019) defines Level Design as "the thoughtful execution of gameplay into gamespace for players to dwell in" [153, p.xxiv]. Salen and Zimmermann (2003) highlight the importance of levels as they coherently strengthen the narrative of a game [151, p.386-87]. Rudolf Kremer (2009) states that levels serve to "interpret the game rules, and to translate them into a construct (a level) that best facilitates play. Another way of expressing this is by stating that 'level design is applied game design'." [110, p.18].

Each researcher brings their own perspective to level design research, but in its core, they agree on the fact that level design aims to communicate the mechanics of the game to players within a confined gamespace as an integral (and narrative) part of the game [110, p.34]The gamespace is limited because the level is a part of the game just like a part of a chain that is connected to the beginning and end points of other levels [151]. Consequently, levels emerge as vital tools for communication between players and designers [153, p.42]. They introduce players to the rules of the world with tutorials, designs and narratives [151]. A game level must be essential to the game by introducing new mechanics, spatial dynamics, storytelling or directing pacing within the game [153].

Given the quantity of level design definitions by various researchers and authors, it becomes evident that level design neither can be defined in one sentence, nor by a couple of rule sets. Instead, it is a topic as complex as Game Design, devoid of a universal formula that fits all scenarios [153, p.xxi]. By exploring the works dedicated to level design, a vast understanding of the authors' different perspectives on level design can be gained.

This thesis partly draws its knowledge from Totten's (2019) book "An Architectural Approach to Level Design". [153] The idea behind this book is to connect architectural principles that have been used over the decades in human architecture to the spatial design principles of level design. The author argues that level designers can take inspiration of the methods and tools used by architects since game levels also encapsulate elements of reality [153, p.xxii]. Non-digital and digital methods and tools used by architects are introduced and referred back to how game designers can create coherent and immersive game worlds using tools and concepts like spatial composition, circulation, scale, rhythm, and lighting [153, p.41 ff.]. Totten (2019) explains how spatial composition can leverage spatial rules and architectural principles, such as playing with open and intimate spaces and using the verticality of a level [153, p.103 ff.]. The book also touches upon the topic of symbols in architecture and the similarity to games as a narrative device and a means of player communication [153, p.169 ff.]. According to Totten (2019), gamespaces do not only have to be used to maximize their spatial potential, but also to communicate the aesthetic idea behind a level, tell emergent stories, play with pacing, risk and reward and enhance the play experience with music and sound [153, p.82,

176, 194, 265, 367]. Levels can also be fitted to a player's individuality and their own style of communication with the game and other players. Ultimately, these concepts shape the player's overall experience, create memorable exposure to the game and invoke emotions that bind the player to the game.

This book sparks many methods that are picked up throughout this thesis and have been incorporated in the thesis's project, the web-based application that collects all available tools and methods, interconnecting them with crucial level design processes and roles.

Another book that sinificantly shaped the findings of this thesis is "The Art of Game Design: A Book of Lenses" by Schell (2008). [137] Although it primarily focuses on game design rather than level design, it provides the reader with a solid and contemporary insight on the various "lenses" a game designer can wear to design their game. In this context, lenses represent the perspectives from which a game can be viewed from, including the perspectives of game designers, players, managers and anyone contributing to the design process of a game [137, p.xxxviii]. There are more than 112 lenses that help the designer to view the game in different perspectives with different attributes. For instance, the book commences with an introduction of the designer, stating that the designer creates an experience and introducing the first "lens of emotion" [137, p.19]. Lenses are built like question prompts, stating the essence of the lens and offering three questions that guide designers toward informed decisions. Returning to the initial example, the "lens of emotion", the following questions are asked [137, p.19]:

- What emotions would I like my player to experience?
- What emotions are players (including me) having when they play now?
- How can I bridge the gap between the emotions players are having and the emotions I'd like them to have?

While many of these lenses are addressing game design rather than level design, specifically, it was stated earlier in this chapter that levels are an essential part of a game [110]. Hence, game designers can use these lenses to develop their levels. Some lenses like the "Lens of Emotion" [137, p.19], the "Lens of the Player" [137, p.131], the "Lens of Time" [137, p.173] or the "Lens of Rules" [137, p.189] are best suited for the early stages of game design, where the focus lies on the overall game concept rather than the specifics of level design [86, p.25]. Other lenses like "The Lens of the Client" [137, p.519] and "The Lens of Profit" [137, p.532] are fit for marketing decisions and strategy [119]. Nonetheless, numerous lenses can be converted to level design, especially those concerning the decision process around the aesthetics of a game [137, p.308], the storytelling of a game and its game levels [137, p.339], and the challenges, rewards and risks within a game [137, p.220 ff.].
This thesis will not explore each lens in detail, but instead connect their ideas with the other tools and methods inspired by other research. As such, lenses are part of the methods collected for level design, fostering a more diverse decision process.

The third book "Rules of Play: Game Design Fundamentals" by Salen and Zimmermann (2003), primarily centered on game design instead of level design, is one of the most notable contributions to the field and influenced many publications in level design such as "An Architectural Approach to Level Design" [153], "Artificial Intelligence and Games" [167] and "Gamification: a systematic review of design frameworks" [120]. [151] It addresses the core concepts of games design and defines games as a set of rules that are established throughout the game [151, p.28 ff., 116 ff.]. This book is also keen to include players as an agent and a recipient, highlighting the importance of a player-centered approach when designing games [151, p.298]. While the book does not look into levels in detail nor introduce specific tools for level design, its insights are valuable for understanding the culture of game design, inspiring some further readings and exploration by other authors [153] [167] [120].

For instance, the concept of the Magic Circle, introduced early in the book, applies to level design as well. [151, p.92 ff.] According to Salen and Zimmerman (2003), a player enters the space of a game, the frame, which is distinct from the real world and offers a unique, separated experience. It is an unconscious and

conscious decision to play the game and be captured by its space. Within the circle, a sense of meaning is created by the game's rules, closing the magic circle and guiding the player through the game. As levels are an integral part of a game, they further contribute to the overall experience, having the responsibility to enforce the rules of the game and create a pleasurable atmosphere as a small fraction of the whole.

The theme of rules is a central focus in the book, dedicating an entire chapter to them. [151, p.116 ff.] The book distinguishes rules from mechanics, as they cover a broader range of applicability. Salen and Zimmerman (2003) state that rules follow these formal characteristics [151, p.125]:

- They limit player action
- They are explicit and unambiguous
- They are shaped by all players
- They are fixed
- They are binding
- They are repeatable

When focusing on digital games, rules can be categorized into three layers or levels [151, p.139 ff.]: The constitutive layer, the operational layer and the implicit layer. The constitutive layer describes "abstract, core mathematical rules of a game" [151, p.139] and defines the overall logic of both digital and non-digital games without explicitly stating how a player should engage with them [151, p.149]. Operational rules, often printed in instructions and at the backside of a game's cover, are the "rules of play that players follow when they are playing a game" [151, p.139]. While constitutive rules concern themselves with internal events of the game, operational rules include the player and their input [151, p.149]. In contrast, implicit rules are the "unwritten rules [...] and behavior that usually go unstated when a game is played" [151, p.139]. In the context of digital games, they may describe the duration of a player's turn, specify that a character attacks upon a right-click action on an enemy, like in the game "League of Legends" (2009) [95], or define that the game can be started, paused and ended [151, p.148]. Implicit rules are often taken for granted [151, p.148].

These rules also apply to levels and level design. Mechanics, a subset of rules as they describe the core functionalities of a game, are considered the core of the game [113]. A game is built on the mechanics as they mark the most important gameplay elements and distinguish one game from another [153]. Game levels contribute to the game as a whole, therefore carrying the burden of transporting a certain meaning [153, p.45 ff.]. They may serve as tutorial levels introducing new mechanics or permitting them to explore the game world and its implicit rules. Alternatively, they may contribute to the game's narrative, establishing a sense of meaning and immersion by shaping the world and adding to the Magic Circle described by Salen and Zimmerman (2003) [151, p.99]. As an aspiring level designer, it is crucial to understand the rules of one's game and how to adept them into their levels, thus giving life to the crafted spaces.

3.1.1 Plan the Space of your Level

Planning the spatial layout of a level is an important aspect of level design and involves many concepts and tools, such as drawing a map and determining the player's starting and ending points. [153] It can be a standalone tool or part of a larger process like prototyping. At times, it may undergo iterations at different stages of the development process to adapt to changes made during development [43]. However, a game designer may find it challenging to efficiently draw a map without further knowledge. One field specifically enhances the spatial planning of levels with technical foundation: Architecture.

According to Totten (2019), spatial composition influenced by architectural principles can significantly enhance a player's experience and engagement [153, p. 103]. A game designer may use architectural spatial design concepts like Figure-Ground, which involves considering the significance of negative space when arranging positive-space objects [153, p. 105 ff.]. Concepts such as "Form Void", which entails removing

parts of an object to create meaningful negative space within the object itself [153, p. 108], and Arrivals, where smaller spaces build anticipation for larger main spaces [153, p. 109], are also applicable. Moreover, **Genius Loci** is a concept important to level design as it describes the atmosphere a level conveys to the player [153, p. 112]. Through the use of "lighting, shadows, spatial organization, and the size of spaces" [153, p. 112] emotions such as dread can be communicated to the player.

Maps often take inspiration from real-world designs, including labyrinths and mazes, increasing their Genius Loci by using the three spatial types of **intimate space**, **narrow space** and **prospect space** [153, p.113 ff.]. While narrow spaces leave the player no choice but to crawl through them or confront an enemy heads-on (see Fig.3.1), thereby creating an uneasy atmosphere [153, p.118], prospect spaces trigger the same emotions in a contradictory concept; they remove spatial limitations of a space and make the player feel small in expansive areas [153, p.124] (see Fig.3.2). On the other side, intimate spaces are often used to create a feeling of safety in the player, using spaces that fit the metrics of the player to prevent feelings of either insignificance or overwhelming magnitude [153, p.120] (see Fig.3.3). These spatial concepts can all be used when drawing a map, allowing for dynamic storytelling and contributing to level pacing by generating spaces of conflict and safety [153, p.125].

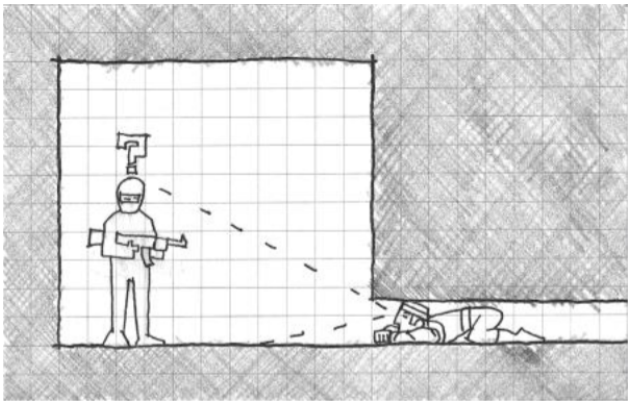


Figure 3.1 An example of narrow space taken from the game "Metal Gear Solid" (1998) [109], handdrawn by Totten (2019) [153]

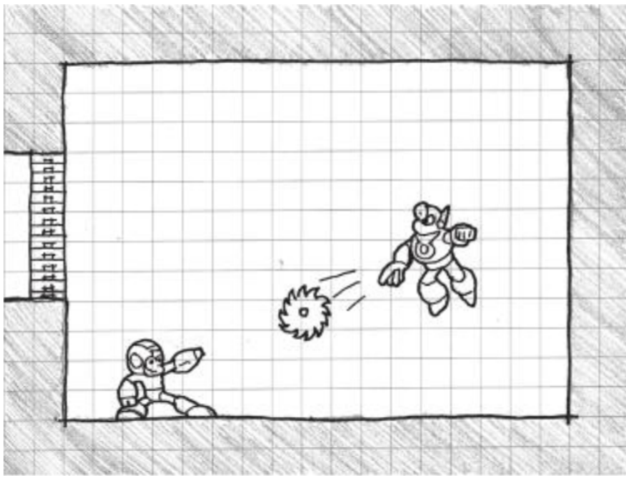


Figure 3.2 An example of prospect space used in the "Megaman" series (1993) [70] where boss rooms are often large, referenced by Totten (2019) [153]

In addition to traditional maps, game designers can create Molecule Diagrams that highlight the relationship between spaces instead of specifying the concrete spatial layout. [153, p.125 ff.] This thesis considers molecule diagrams as a method for spatial level planning. Its bubbles reflect the spaces of a level and the

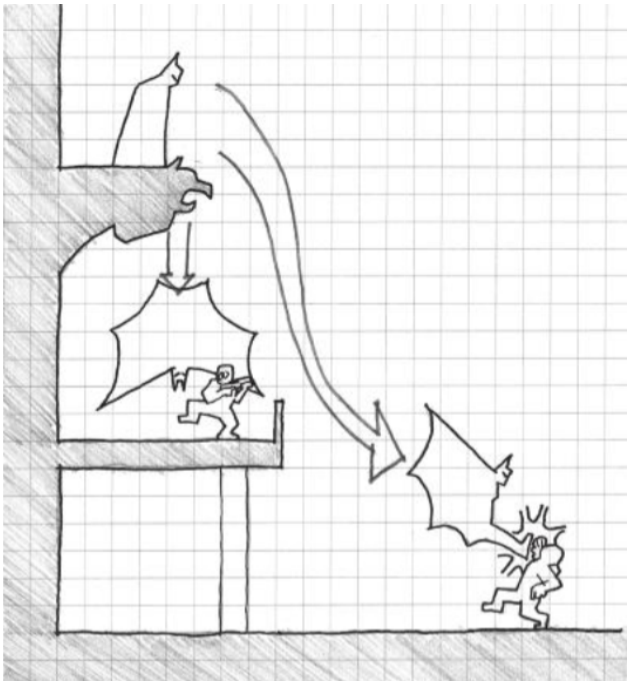


Figure 3.3 An example of intimate space in "Batman: Arkham Asylum" (2009) [148], where players can make use of vantage points to gain a better overview of the fight, handdrawn by Totten (2019) [153]

corridors or lines connecting the bubbles define the relationship between these spaces. They may also include **Steiner Points** which are important gameplay points connecting different spaces of the level, allowing the player to navigate through the level vertically as in Fig.3.4 [26][153, p.128]. Speed runs often integrate steiner points when skipping sections of the level to beat the game fast and efficiently.

Figure 3.4 A map in the game "Borderlands" (2009) [143] with a path and two steiner points that shortcut the two routes from C to B and from E to C, taken from [26]

Should game designers wish to illustrate the importance of a relationship between gamespaces, they can represent corridors as larger or employ different line forms to symbolize various types of relationships, such as dashed lines for spaces which can be seen but not accessed from another space [153, p.132 ff.]. In architecture, these diagrams are called **Proximity Diagrams** (see Fig.3.5) [153, p.132].

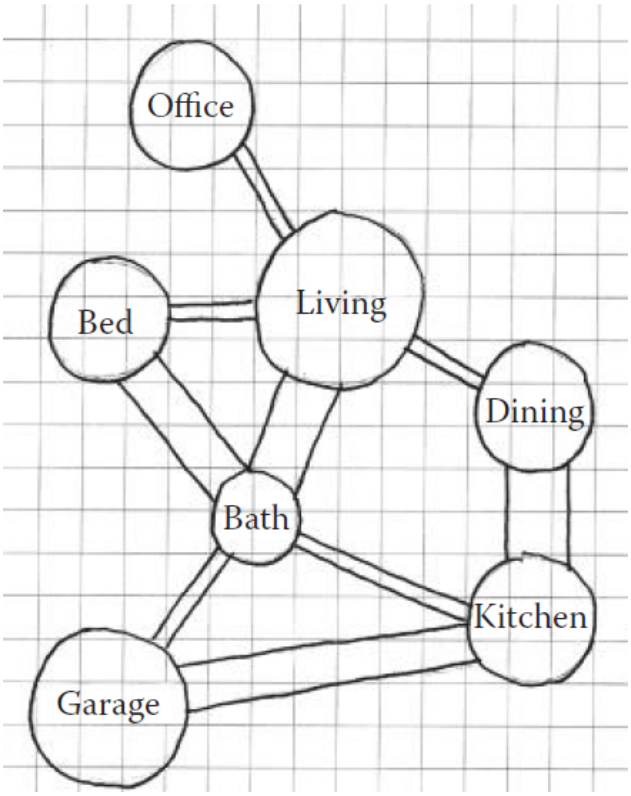


Figure 3.5 A proximity diagram showing the connections of the rooms in an exemplary level, with the significance of the connections depicted by the thickness of the edges. Taken from [153]

There are two types of concrete gamespaces, each serving important purposes in game design. **Hub Spaces** (see Fig.3.6) connect different levels of the game and are often designed with the concept of intimate spaces in mind [153, p.134]. For example, the lake in the game "God of War" (2018) [146] connects different islands and areas of puzzles, exploration and combat. As the lake is gradually drained, more of the game levels are exposed. **Sandbox Gamespaces** emit the feeling of an open world by avoiding hubs and instead allowing the player to explore the world and discover gameplay context spontaneously [153, p.135 ff.]. These spaces also utilize pathfinding and landmarks to simplify the orientation for the player as in Fig.3.7.

All of these methods are collected in this thesis's web-based application, bridging theoretical concepts with real game examples and guiding the reader through the process of applying these methods in the context of game design.

3.1.2 Pacing

Another important aspect of level design is Pacing [153, p.82,292 ff.]. Totten (2019) wrote that pacing unfolds as the rhythm and tempo of a level, alternating between intense and calm moments [153, p.82]. Pacing is also referred to as the order of events which builds tension and excitement, often known as narrative pacing [73, p.415]. Ultimately, pacing is a method that game designers use to convey their stories to the audience, balancing out densely packed combat with moments of exploration [35].

One way to design and implement pacing into a game is through the use of a **Pacing Diagram**, a crucial tool for game designers aiming to guide players through their levels while maintaining their attention [35] [21]. The diagram often encapsulates the **three-act structure** as in Fig.3.8, commonly used in storytelling, particularly in film, movies and theater [87]. It divides the story into three parts: A beginning, a middle and an end. Each act serves a specific purpose to advance the plot and character development. The first act sets up the story and the characters, the second act advances the character and story arc with confrontations and obstacles to achieve a leading goal, and the third act resolves the story and leads to resolution [87, p.7]. To

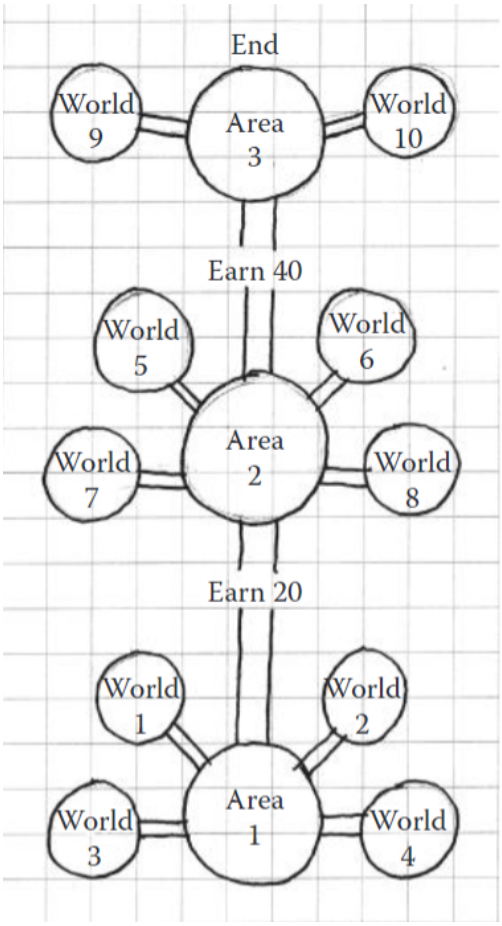


Figure 3.6 A typical hub space structure from an exemplary game, drawn by Totten (2019) [153]

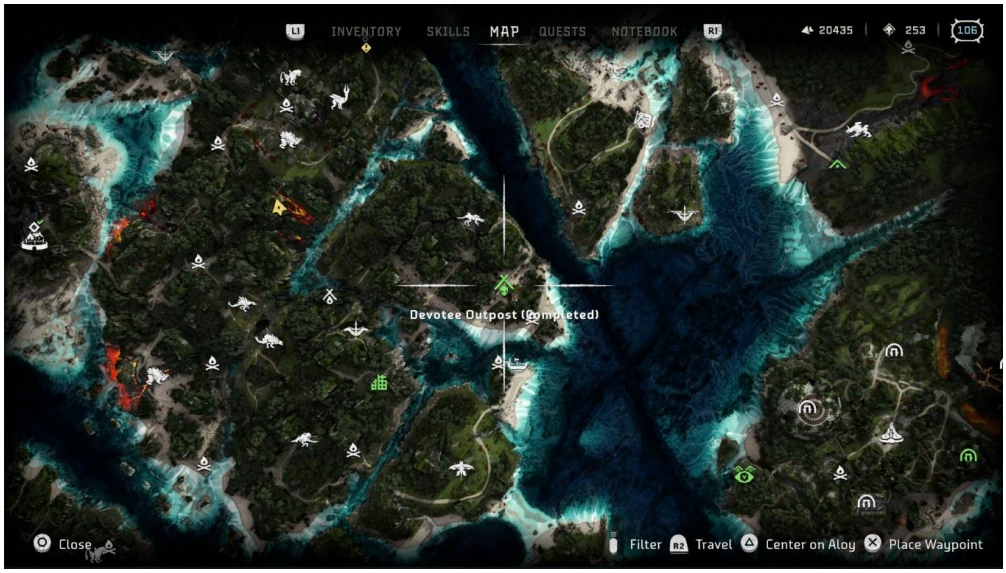


Figure 3.7 The map in the game "Horizon Zero Dawn" (2017) [93] is an example for a sandbox structure as it allows the player to roam freely but still provides some type of overview by showing landmarks on the map.

tell a story within a game and manage pacing, game designers utilize this dramatic structure together with the Hero's Journey, a concept that guides the character through an unknown world to eventually achieve their destined resolution [69] [104]. A narrative eventually peaks in a climax which speeds up the pacing of a game and increases the flow of action [153, p.293]. In "Donkey Kong" (1981), for instance, a narrative can

be divided into multiple levels, where the climax point, such as saving Pauline, is stretched across several levels, each demanding the player to remove barrels under increasingly stressful conditions [124]. To keep the player's attention, game designers can loosen up the tension of high-packed action levels by adding loose-tension and exploration-heavy parts in a level [35].

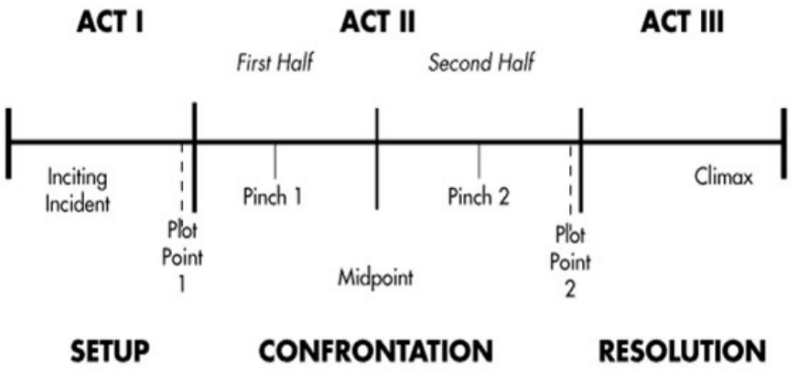


Figure 3.8 The Three-Act-Structure narrative model [35]

Alex K. introduced a couple of methods to include and improve pacing in games [20]: One of these methods involves marking significant **Gameplay Beats** within the story. Gameplay beats are "specific points on the (narrative) chart that indicate what is happening" [20, p.55], highlighting moments when players encounter story-relevant events. The distance of the gameplay beats indicates how densely packed a level is, with a narrow distance indicating that the sequence is filled with high tension and action and a big distance allowing the player to calm down and explore on their own, adding low pacing to the level. This structure often features a constant rise and fall of the level's intensity until it reaches a climax and cools down in the end (Fig.3.9):

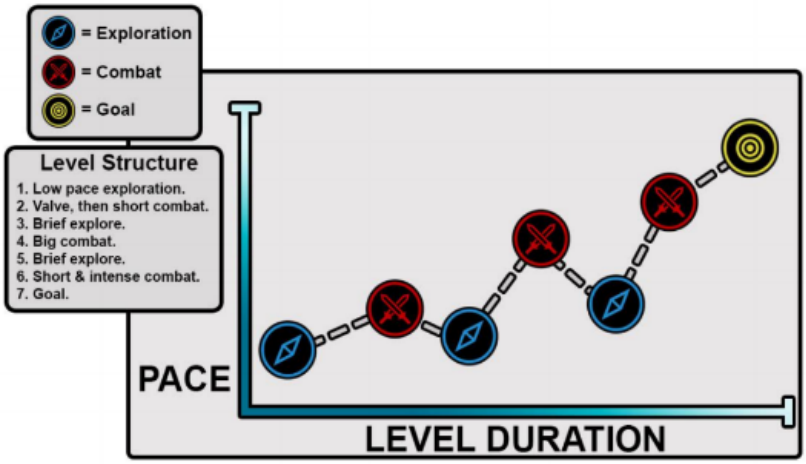


Figure 3.9 An example for the pacing flow during a level [20, p.55]

It can be found in Uncharted 4 as well (Fig.3.10):

Note that the graph is uneven. More ups and downs within the graph increase the dynamic behavior of the story, reducing its predictability and increasing the immersion of the player [20, p.57].

So far, the discussion has primarily focused on linear pacing, following a line with various beats until the story in the game reaches a climax and cools down in the end. By including alternative routes and decision trees which are depending on the player, game levels can also adhere to the concept of **Non-Linear Pacing**

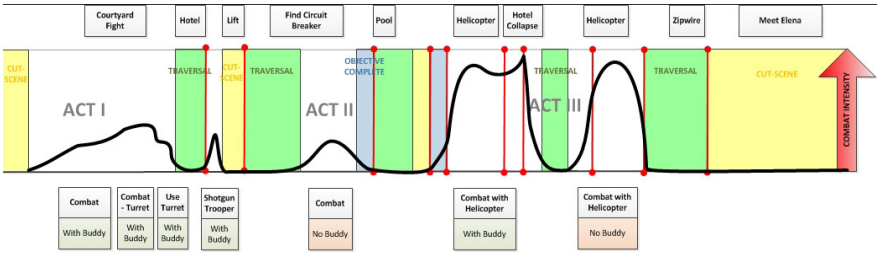


Figure 3.10 The pacing of chapter 6 - 'Desperate Times' from "Uncharted 2" (2009) [79] drawn as a timeline [35]

[20, p.57]. The game "Detroit: Become Human" (2018) uses alternative routes to integrate the player's decision-making in the outcome of their levels, additionally presenting a graph with all possible hidden alternative routes and decisions at the end of each level [80]. This graph is a useful tool to document the opening of the story branch to alternative routes and the merging of different routes to the same ending. These routes can also be grouped into gameplay beats, indicating their parallel timelines (Fig.3.11):

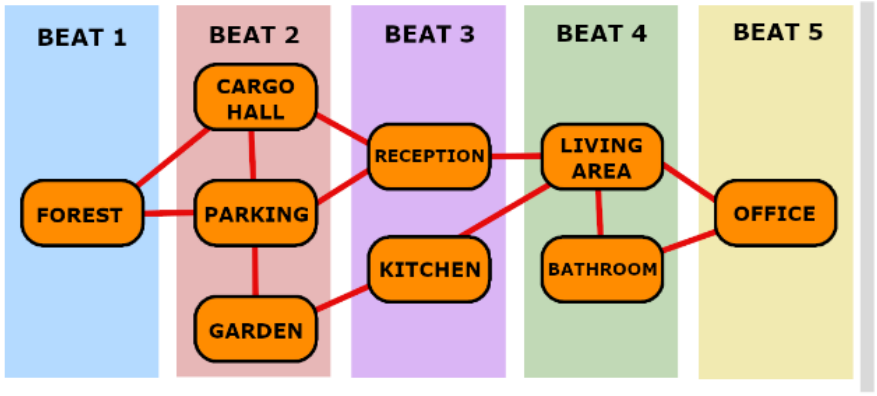


Figure 3.11 A diagram that breaks down the areas of a level into beats and illustrates the non-linear flow of the level pacing [20, p.57]

In general, pacing is a powerful tool to include dynamic storytelling in a level. Creating a pacing diagram is one of the essential methods that have been collected in this master thesis project. The next section will talk about a more abstract concept that cannot easily be explained in a method, yet it is integral enough to be included and thought about.

3.1.3 Create Emotional Experiences

Games guide players through a variety of emotions, increasing their compelling and immersive nature [101]. While spatial layout itself may create feelings of tension, for instance with narrow space, safety as with intimate space, or uncertainty with prospect space, pacing is used to control the intensity and dynamic of emotions the player experiences throughout the game [153]. Many other aspects can increase the intensity of emotions, guide emotions in a certain direction and create new emotions. In fact, the creation of emotional experiences extends beyond games and can be observed in various artistic forms such as plays, literature, movies and even the experience of observing architecture [153] [101].

Totten (2019) describes Delight - a basic human emotion - as a building block of Level Design which is essential to create an immersive level [153, p.8]. The concept of **Genius Loci** describes the "spirit of a place" [153, p.112], binding the emotion of the place to the play experience and the emotions felt through the screen [153, p.190]. Emotion can also be felt through succeeding in a designated task or experiencing defeat [99, p.6], leading us to one of the primary catalysts for fostering emotional experiences in games: Risk and Reward.

Risk and Reward

As Totten (2019) stated, "the play between risk and rewarding players [...] is of utmost importance in creating interesting emotional experiences" [153, p.194]. In fact, it is often a challenge balancing risk with reward, widely recognized as a crucial aspect of game and level design [137] [65, p.86].

Risk describes the uncertainty of a situation or space, known to the player but with limited information available for calculating the extent of that uncertainty [151]. On the other hand, rewards, often in form of success, motivate the player to continue the game [99].Rewards do not have to be linked to success but may allow the player to explore the level further, strengthening their curiosity and incentivizing certain in-game behavior [153, p.242]. Rewards can empower players to take risks, with the scale of the reward determining the magnitude of risk a player is willing to undertake [153, p.243] [65, p.87]. But how are risk and reward connected to emotion?

Playing games is a process that not only strengthens cognitive skills, but promotes motivational development inside and outside of the gaming context [99]. While succeeding in challenges and receiving awards increases the player's level of motivation, losing and being denied of the rewards has a contrary effect; it creates frustration [149]. Risk is superior to plain uncertainty because it provides the player with limited information, offering them the choice to engage with the risk and subsequently experiencing excitement and unease [153, p.194]. If both risk and reward are balanced out, they create a gameplay flow that compels the player to play further [149].

Risk can be integrated into a game, much like uncertainty. For example, there might be a chasm in front of the player, but the player can not see what comes after the chasm in an uncertain context. Including the camera perspective that might give hints on a reward coming after the chasm, the uncertainty is transformed into risk instead. In Baldur's Gate 3" (2023) the camera allows the player to scout the area around the characters and potentially spot treasures, the reward, that might be guarded by foes, the risk [147]. Players can assess the number of adversaries and weigh the potential risk before deciding on their actions.

In their book, Salen and Zimmermann (2003) describe the types of rewards that can be utilized in game and level design [151]. Rewards of glory are an achievement gained by successfully overcoming challenges in the game. Rewards of sustenance are objects that are unlocked to customize the character, for example often used in roleplay-games and games with a detailed character creation such as "Baldur's Gate 3" (2023) or "Horizon Zero Dawn" (2017) [147] [93]. Rewards of access are items gained by the player to unlock previously locked areas, fostering the curiosity of the player and their drive to explore [153, p.245]. Last, rewards of facility are expanding the player's movement set and can also reward them with access to previously inaccessible areas. For example, in "Horizon Forbidden West" (2022) the player gains the ability to deep-dive over time which unlocks treasure hunts in the deep sea [153, p.245] [94].

There are many ways on how to achieve a sustainable balance between risk and award and this thesis's project also includes a method that can be used by level designers to plan their risks and rewards: The risk and rewards schedule. Further details are provided within the project itself.

Lighting and Shadows

Lighting is a powerful tool in level design that can significantly impact the emotional experience of players. As with game design, the careful placement of lighting, saturation and shadow can dominate the atmospheric experience for the player [20, p.64]. Level designers use lighting to enforce the Genius Loci of the place, for example integrating more harsh contrasts and more dark areas to create a suspenseful and uneasy atmosphere [153, p.112]. Lighting can also be used to brighten up places that are supposed to serve as a sanctuary, allowing the light to travel through and create a comfortable atmosphere even in prospect spaces [153, p.221]. Moreover, lighting choices can highlight areas of importance by increasing the contrast on specific objects or within their surroundings [153, p.179].

Shadow can cause an intriguing duality of contradicting emotions. While shadow in corners of prospect spaces can evoke feelings of seclusion, safety and protection from the outer world, an excessive amount of shadow obscuring a significant portion of space can evoke the feeling of being watched [153, p.221]. Shade,

a lighter and softer version of shadow, adds to that ambiguity by defining both sacred and malevolent spaces with light only coming through curtains or other dimming material [153, p.222].

In addition to lighting and shadow, the choice of color can also shift the mood of a space [20, p.64]. Humans are trained to link certain colors to specific emotions, for instance feeling different emotions in a blue colored room compared to a room colored in a bright red [122]. The TV series "Stranger Things" (2016) effectively contrasts the real world and the upside-down by shading the same places with different colors [114]. As in Fig.3.12, a bathroom in the real world can be turned into a creepy, isolated place not only by rotten furniture, but mainly by the absence of light, with primary light sources mostly being replaced by shadows and red foreign objects, transforming the living room in the "upside-down" into a darker and more unsettling setting.

Figure 3.12 A bathroom in the upside-down in "Stranger Things" (2016) [114]

The example of Stranger things hints to another important aspect of conveying emotions: Music and Sound.

Music and Sound

Music and Sound play a vital role in the experience of games and are often overlooked although they are one of the most noticeable elements in a game [153, p.370]. In "Stranger Things" (2016), the upside-down is characterized by muffled, echoing sounds resembling an underwater factory hall. The sounds emitted by the vines are ten-folded and exaggerated compared to the sounds made by real-world vines [44]. By deciding to utilize these sounds, the creators of the show intentionally enforced the unsettling atmosphere.

In games, the same concept can be applied depending on the intended genius loci that the game aims to convey. Music serves to set an atmosphere and can be played at the right times to convey a certain feeling and atmosphere of the setting [153, p.368]. For instance, a hub space that elicits a safe feeling may be compatible with a light, melodic music, while combat sequences typically require an action-packed sound. A study in 1998 proved that distinct types of music affect the listener's emotions, including designer music which is closely related to game music [153, p.370][116]. Games like "Metroid II - The Return of Samus" (1991) utilize music as a medium to convey emotion by integrating specifically designed music compositions to heighten the sense of confinement in narrow spaces [125] [153, p.371].

Moreover, sounds can not only enhance the game's soundtrack, but also guide players through the level as indicators to treasures or usable objects, such as ladders [153, p.368]. A repeating beat may invoke a certain pattern that the player recognises, leading to a strategic approach to the level as in puzzles or rhythm games [153, p.369]. It can also condition the player to feel certain emotions once the sound plays [153, p.373]. In "Tomb Raider" (2013), for instance, a bell chime sound is played every time the player finds a tomb which

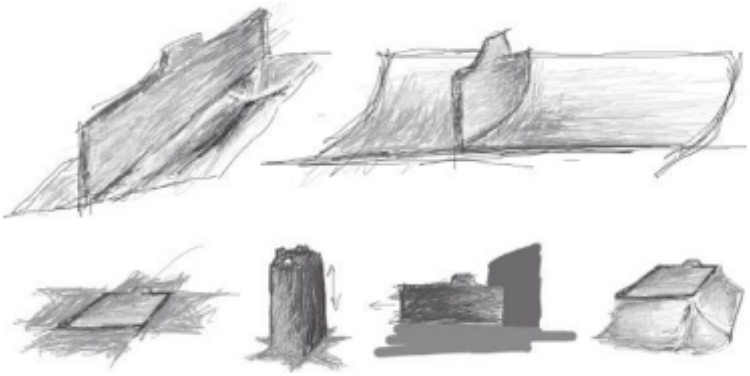


Figure 3.13 A collection of drawn level partis of fence traps used in "SWARM!" (2013) [82], taken from Totten (2019) [153, p.81]

bears rewards and a safe space without enemies [81]. Sounds also contribute to the realism of the game, engaging the player and creating a realistic experience, for instance with the sound of leaves rustling or the wind howling [153, p.380].

Music and Sound Design is not only overlooked but it is also a crucial topic, and delving deeper would require a separate thesis. The information provided here offers only a glimpse and is supposed to give an introduction to the vast spectrum of what music and sound design is able to achieve. Books like "Game Sound: an Introduction to the History, Theory, and Practice of Video Game Music and Sound Design" take a deeper look at how music and sound design can differentiate exceptional games from mediocre ones [74].

3.1.4 Other Fields influencing Level Design

Level Design does not only originate in the science of Game Design as this field itself is a vastly new research field that builds on various other fields like computer science [57], arts [153], psychology [137] and physics [54]. Among other researchers, Totten (2019) emphasizes on the relation between level design and human **architecture**. [153] As already discussed in the previous sections, he argues that many concepts of architecture can directly be translated to the design of a level, especially the spatial design. For example, the same usage of spatial types like narrow space, intimate space and prospect space can be applied to levels as well, supported by architectural building blocks like vantage points, Steiner points and the overall atmosphere created by the Genius Loci. As he guides the reader through the book, he also promotes tools and methods that have been used by architects before [153, p.41 ff.]. Such tools include basic drawing techniques, such as how to draw a line, shading, hierarchichal drawing [153, p.49 ff.], types of architectural drawings like a plan, section, elevation, axonometric, or perspective [153, p.54 ff.], sketching and journal writing [153, p.62 ff.] and even digital tools that support architects and level designers [153, p.65 ff.]. Workflows like a Level Design Parti (see Fig.3.13) originate from architecture as well as they are based on sketching objects and places in different variations until the designer is satisfied with one design [153, p.81 ff.].

As mentioned before, this thesis takes heavy inspiration from Totten's Book since architecture proves to have many practical methods and tools that are useful for game designers as well. Together with other methods, the knowledge gained from this book builds a solid groundwork for any striving level designer to efficiently create their own levels.

Another field that is often compared to game and level design is that of movie making and theatrics, especially when it comes to **storytelling** and narrative in general [105] [128]. "The Hero with a thousand faces" (2008) is a book cited by both sources from game design research and movie making research. [69] Summarizing it briefly, the book describes how many cultures share a common archetype of story: The hero's journey. It typically involves the following stages:

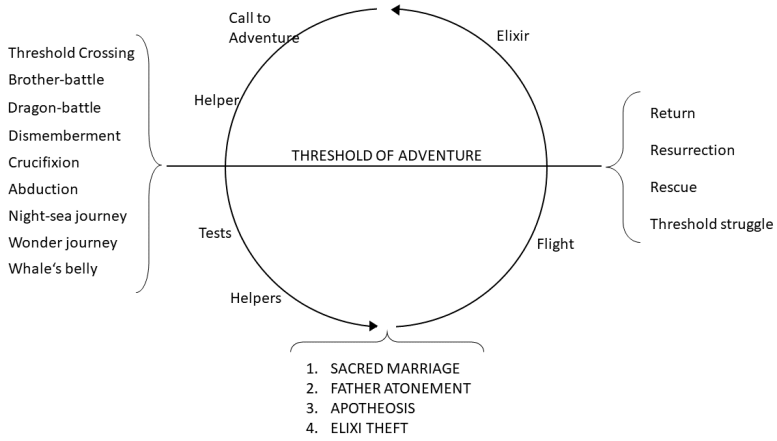


Figure 3.14 The adventure of the hero depicted in a circle. The hero is called for an adventure and ventures to the unknown world (the bottom of the circle), overcomes obstacles with helpers, engages in flight and returns back to their world with an elixir to their initial problem [69, p.227].

- Departure: The hero leaves their ordinary world and embarks on a quest or adventure, often prompted by a call to action or a significant event.
- Initiation: The hero encounters challenges and tests to succeed or fail in trials while confronting adversaries, being guided by mentors and undergoing inner transformation.
- Return: The hero returns to the ordinary world, often bringing newfound wisdom, skills, or a boon to share with their community.

The hero is a symbolic representation of an individual's psychological and spiritual development and a stereotype used in movies, books and games alike [105] [137]. The hero's journey also drives the narrative of said mediums, determining the pacing of games and how relatable a story is [66]. An adventure plot of the hero's journey is depicted in Fig.3.14.

Psychology also plays a big role in game and level design. In cognitive psychology, researchers inspect how humans process information, make decisions and solve problems based on experience which is closely linked to playing games as well since it enables cognitive processes [99]. Behaviorism dives into the actions and habits a human learns from being exposed to stimuli, with players learning how to play with tutorials and by studying the rules of the game [151] [161]. As already mentioned, motivation also plays a vital role in the drive to play further [99].

The social part has so far been dismissed but is not less crucial to game design as the other fields. [99] Not only social psychology is concerned with social dynamics, but the field **Sociology** is just as important [97]. There are many game genres that include social interaction, cooperation or competition with other players such as social games, MMORPGs or casual party games, and the idea of community strives players to loose themselves in the game world even without challenge as a motivational drive [164]. For example, "Animal crossing: New Horizons" (2020) has the player engage in a community with animal NPC-characters to build a village and form meaningful relationships while also integrating a multiplayer-feature to visit another player's island, trade items and participate in activities together [126]. Quest Hubs and spawn points in MMORPGs serve as intimate space in which the player can interact with other players and create unique gameplay experiences [153, p.358]. There is also a different feeling to exploration if players can do it together rather than engaging in activities alone [153, p.360]. Communities have become so big that they extend over the game to other platforms like "Reddit" or "Youtube" in which the players can exchange their experiences and support each other [38] [51].

Especially in the recent years, Artificial Intelligence has advanced at a fast pace and found its way not only into the everyday life with cleaning robots and Chat GPT [4], but also into games [167]. This is a field that

cannot be ignored when talking about the design of levels, especially the design of NPCs, allies, and enemies and the techniques behind path and way finding. But since this is a technical field it will be described in detail in the next section when talking about the Technical Background of Level Design (Chapter 3.2).

3.1.5 Art in Levels

The use of art in level design and game design in general is a critical component that greatly influences the player experience, immersion, and emotional engagement within a game [137] [153]. In the development phase many hours are spent on creating assets, concept art, character art, animation, aesthetics, lighting and textures [137]. Art is a great way to communicate with the player, supporting the message of the game and enhancing the emotional and narrative effect it has on the player [153]. It can also strengthen the pacing by increasing the environmental tension communicated through art [153]. For example, "Final Fantasy XV" (2016) makes use of wide areas and bright lighting that encourages the player to explore the landscape of the world in between the tightly packed story [84]. In "Hollow Knight" (2017), the artwork is what is remembered by fans as a flashy, handdrawn 2D masterpiece greatly contributing to the experience [72] [17]. Integrating art in levels influences many methods mentioned in this thesis, especially the following components of art that specifically impact the level design process:

Aesthetics and Visual Style

Art can define the overall style of the game level, setting the tone and atmosphere. [83] It encompasses choices like color palettes, architectural styles, and thematic elements made in the early design stage. Lighting and particle effects can also enhance the atmosphere depending on the kind of style the designers are going for. Mood Boards and Reference collections are methods that help finding a unique and consistent style to support the overall message of the game and immerse the player in an overall emotional experience.

Environmental Storytelling

Art can also help telling the story, history and context of the game world by carefully placing unique items, flora and fauna in the world. [153] Architecture also plays a big role in environmental storytelling as it helps characterising the game and telling the history of the world's civilizations. Game Designers can use reference collections, especially those referencing real-world architecture examples, to build their own environment that can be distinguished and easily set apart from other worlds or places. Like real history, superficial architecture and nature can tell a story too and make the world the game is playing in more vivid and realistic.

Level Navigation and Flow

Visually guiding the player through the level helps creating an immersive experience. [153] [137] [21] While architectural structures like nodes, hubs and landmarks help the player with orienting within the game space, visual cues can help the player recognize certain patterns over time and react to it accordingly [153, p.130]. Paths can also distinguish themselves from the surrounding environment in their flora or height difference, making them stand out for the player to follow the right directions [21]. The easier it is to navigate within a level, the less frustrated the player is and the better they can concentrate on the gameplay, especially if navigation is not part of the main mechanics.

Symbolism

Symbols are powerful tools in the construction of the level and teaching the player mechanics or patterns. [153, p.171 ff.] Like landmarks, they can be used as distinguishable and unique assets to simplify the navigation throughout a level. They can also be used as guides for players. Especially in the first levels, the player can get used to the visuals of the symbols that teach them gameplay mechanics [153, p.175]. For example, in "Portal" (2011) the early levels are build in geometrical patterns that repeat over the course of the game, hinting to the player that the same strategies from the early levels can be applied later on [155].

Another effective implementation is the visual distinguishment of climbable walls in "Tomb Raider" (2013) that are marked with old white paint [81]. Like visual cues, symbols stand out from other art assets and communicate its significance to the player [153, p.176]. Game designers should define symbols and visual cues and their significance to the mechanics of the game early on. They can get inspired by reference collections.

Emotional Impact

Art plays a vital role in invoking emotion through visual stimuli. [137] Having defined the aesthetics and overall visual style of a game and adding lighting effects, shadows and environmental details, emotional responses can be enhanced. It is important to define which style should be represented in the overall game and if the style changes throughout the game to accompany the narrative, making the levels distinguishable from each other in their art style and emotional impact. Game designers should also take into account how climate changes the overall look of the game and using different flora, fauna and changing the lighting within the level can greatly influence the emotional experience of the player [21].

3.2 Technical Background

In game design, technology represents one of the four foundational pillars, alongside story, aesthetics and mechanics. [137] Engineers are a critical part in the game development process, responsible for implementing the logic behind the game and mechanics [137, p.475]. However, technology goes beyond the technical implementation of the game's rules; it equips the game with a diverse array of tools, platforms and frameworks that bring the game to live, enable designers to realize their creative visions and communicate these visions to the player [137] [123]. While technology draws inspiration and ideas from the domains of story, aesthetics, and mechanics, it simultaneously supports each of these pillars, too.

Schell (2008) divides technology into foundational and decorational subparts, maintaining an overview and keeping the information organized. [137, p.475 ff.] While the foundational technology serves as the game's foundation and ensures the game's functionality, the decorational technology enhances the already existing work and increases its uniqueness. For instance, detailed 3D art is integrated into the game as decorational technology, enhancing the experience of the player, while sound can be very essential and therefore often belongs to the foundational technology of a game [137, p.500].

Technology manifests in various subcategories of level design. [137, p.497 ff.] It is used in the creation and technical development of a game, such as prototyping, implementation of physics, simulation and mechanics, and in the creation of art and the transfer of the visual style, such as asset creation, concept artwork and landscape design. Furthermore, it influences narrative, storytelling, such as music, sound, cinematics and cutscenes, and the integration of social interaction, such as networking and server infrastructure for Multiplayer.

This section will introduce three key technologies that have a high impact on the technical implementation of the level, benefiting not only designers but also all contributors involved in the level's development.

3.2.1 Game Engines

Game engines are platforms providing developers with crucial tools and functionalities in the creation, implementation and iteration of game levels. [100] Engineers employ these engines for implementing mechanics and the logic of the game while artists may use them when creating the visual scene, assets, terrain and the overall artistic atmosphere. Most game engines also provide inbuilt tools to include lighting, physics and user interface (UI) design. Companies like Ubisoft and Electronic Arts implement and maintain their own engine, as in the use case of "Frostbite", while others make use of publicly available engines like "Unity" [11] [45]. Some game engines were further made public after launching the game, such as the "Unreal Engine" developer kit after the launch of Unreal 3 by Epic Games [8].

In the following, four prominent game engines will be introduced, offering illustrative examples that highlight their significance in the context of level design.

Unity

Unity stands out as one of the most popular game engines in contemporary game development, providing tools and features that empower designers to create immersive and engaging game environments. [45] It excels in its versatility and offers a range of functionalities that facilitate the entire level design process, from conception to implementation.

With Unity, level designers and engineers can create the digital layout of the scene, develop a prototype or a gym scene in which important game mechanics can be tested, use the integrated physics system to define realistic interactions between objects and characters, and use Unity's scripting capabilities to implement gameplay mechanics, AI behaviors, and interactive elements within the level. Dynamic triggers and events allow for a lively, responsive gameplay experience. [45] Unity's lighting tools further enable designers and engineers to create realistic lighting conditions, shadows, and effects that set the mood and atmosphere of the level, while particle effects heighten the visual power of the level.

Artists benefit from the feature to import assets including 3D models, textures, animations, and audio files. [45] Unity's terrain system facilitates terrain sculpting and painting, simplifying the creation of the level particularly for expansive open terrains and open-world games. Additionally, Unity has a built-in prefab system that enables artists to create reusable object templates. This is particularly useful for placing and maintaining consistent elements throughout the level, such as doors, enemies, or collectibles. UI elements such as menus, HUDs, and interactive elements can also easily be created in Unity to enhance the player experience.

As part of the development process, Unity supports designers to quickly test and iterate on their levels within the editor, facilitating the refinement of gameplay, mechanics, and visual aspects. It also has a collaboration mode to enable multiple designers to work on the same project simultaneously, streamlining teamwork and enhancing level design collaboration.

Unreal Engine

Unreal Engine is another popular game engine in the realm of game development. [47] Its first generation was built by the founder of Epic Games to help develop the first-person shooter "Unreal" in 1998 [9]. The engine was particularly acclaimed by its striking graphical innovations related to the time period, notably by the implementation of volumetric fog [46]. With the launch of the game "Unreal Tournament 3" the developers released a free version of Unreal Engine 3 (UE3) called the "Unreal Development Kit" (UDK) for independent developers to use in their projects and games [91] [8]. The latest publicly available version is Unreal Engine 5, with its source code accessible on GitHub [48]. Some prominent titles developed with Unreal Engine include "Fortnite" (2017), "Borderlands" (2009) and "Batman Arkham Knight" (2009) [92] [143] [148].

Unreal Engine plays a significant and multifaceted role in level design, providing a comprehensive toolkit and features that empower designers to create immersive and visually impressive game environments. [47] Notably, it is renowned for its advanced graphics and realistic rendering capabilities to visually engage the player. The engine provides a visual editor for designers and artists to create and arrange game scenes by placing objects, characters, terrain, and other elements within the level to define the overall layout and view changes in real-time. Similar to Unity, it also supports the integration of a wide range of assets, including 3D models, textures, animations, and audio files, further allowing the artists to organize the assets in a file system. It further facilitates terrain sculpting and painting, simplifying the level creation especially for open-worlds and versatile terrain.

Unreal Engine has a visual scripting system called "Blueprints" that empowers engineers to create complex interactions, gameplay mechanics and event triggers without extensive coding. [47] This capability enables the implementation of logic, scripted sequences, and interactive elements that enhance gameplay and storytelling. Realistic physics are handled by the Unreal Engine's physics engine, allowing engineers to create physics-based puzzles, dynamic objects, and interactive elements. The integrated lighting tools en-

able designers to create dynamic lighting conditions, realistic shadows, and atmospheric effects while global illumination, dynamic lighting and post-processing contribute to crafting the desired mood and atmosphere of the level. Much like Unity, Unreal Engine provides the developers with tools to create UI elements like menus and interactive elements to enhance the players' experience.

The level can be optimized by using built-in optimization tools to manage rendering settings, optimize assets, and manage performance to ensure seamless performance on target platforms. Teams can communicate and work on the same project simultaneously. Furthermore, augmented reality (AR) and virtual reality (VR) is supported. [47]

Both game engines, despite their shared role in game development, are distinguished by their unique strengths and what they focus on. [141] Unreal Engine really excels in the visual aspect, offering a broader spectrum of features and rendering quality that leans toward photorealism and visual fidelity. In contrast, Unity adopts C# as its main programming language which is easier to learn and less complex than C++, the programming language used by Unreal Engine. Unity also has an interface that is more user-friendly, while Unreal Engine offers a more extensive array of tools. Both engines maintain their respective asset stores where assets, tools and plugins can be bought and sold. Unreal Engine's store is known for its quality control and focuses on high-end assets, while Unity's asset store is favored among indie developers for its diverse selection, due to its range of simple assets and whole templates [47] [45]. Regarding target platforms, Unreal Engine is well-suited for creating games with high-end graphics, making it a choice for AAA titles and projects targeting powerful gaming platforms compared to Unity whose versatility allows it to target a wide range of platforms, including consoles, PCs, mobile devices, and augmented/virtual reality. Both engines are free-to use, although Unity offers a paid Pro version for games that are expected to create revenue [45]. With Unreal Engine, its priced royalty model only applies to games with significant revenue [47].

RPG Maker

RPG Maker is often referred to as game engine, but its most significant difference to Unity and Unreal Engine is its specialization in creating role-playing games (RPGs). [40] Its particular emphasis lies in storytelling, character interactions, and turn-based combat mechanics, specifically appealing to a niche within the game development community [41]. Games like "Grimm's Hollow" (2019) and "Pokemon Uranium" (2016) were created with RPG Maker [96] [150].

In Level Design, RPG maker is mainly used for creating 2D RPGs because it provides pre-designed tilesets, characters, and assets that are well-suited for classic RPG aesthetics. [40] The tool simplifies level design by offering a grid-based map editor where designers can create scenes, dungeons, towns, and other RPG environments. It also encases dialogue systems, event scripting, and character progression — integral aspects of RPG design that allows designers to create branching narratives, quests, and dialogue interactions contributing to the RPG experience.

RPG Maker is known for its user-friendly, simple interface and focuses on simple scripting without complex visual features like Unreal Engine. [40] Unlike both listed game engines, it is more specialized and therefore does not support any other game genres apart from RPGs limiting its versatility. Even though it provides its own assets, customization and importing of assets are limited compared to the other two game engines. Depending on the genre of the game, developers might still prefer using RPG Maker over the other two game engines.

Frostbite

Frostbite is a game engine developed by DICE, known for its use in creating visually impressive and immersive games. [11] It has originally been used in the development of the "Battlefield" series in 2008 and has since been expanded to other first-person games made by EA [78]. Up to this date, it is solely used by EA. Popular AAA titles like "Mass Effect Andromeda" (2017) and "Dragon Age: Inquisition" (2014) were made

with this engine [64] [63].

This highlights the difference compared to Unity and Unreal: It is not publicly available. Still, it is tailored and perfected to the needs of EA games and known for its high-quality, visually stunning games with a focus on realism and immersion comparable to Unreal Engine and exceeding the latter in terms of photorealism [141]. On the other side, it is often criticized for its complexity, making it challenging to work with for developers and sometimes leading to bugs and visual errors in the games that are recognized by players [23] [12] [10].

Since Unreal Engine was first developed exclusively for the Unreal games but made public after some time, Frostbite could follow the same pattern to allow individual developers to profit from its source code. [141] Even though it lacks versatility compared to Unity and Unreal Engine and might be too complex for seamless use, it is still worth highlighting that big companies like EA usually have their own engine they work with.

This thesis incorporates game engines as tools to use in the process of designing a level, serving as versatile instruments that support various design methodologies. Another important mention is the use of art software specifically used by artists to create individual assets and art for the game, ranging from 2D art to 3D art, concept art and overall landscape design.

3.2.2 Art Software

Art software plays a pivotal role in level design by enabling designers to create, manipulate, and enhance visual assets that contribute to the overall look and feel of game environments. [153, p.70, 283 ff.] These tools empower artists to craft captivating and immersive worlds that engage players and enhance the storytelling experience. It supports the dynamic creation of assets including 2D and 3D models, textures, characters and objects. Art software further helps artists to design the game's environment like landscapes, terrains, architecture, and structures that shape the overall layout of the level [21]. Texture mapping allows the artists to combine their assets with fitting texture that can be imported to the game engine of choice. If the game engine does not provide a sufficient particle system, some art softwares are able to support the creation of custom particle systems [75]. Furthermore, art software can be used to create unique concept art and storyboards that visualize the intended look and style of the level before implementation [137].

Here is a look at a few popular art software options commonly used in level design:

Adobe Photoshop

Adobe Photoshop is a graphic editor developed by Adobe Inc. in 1990. [55] It was initially a tool mainly used to edit the colors of a picture and quickly became the industry standard to manipulate pictures over the years, expanding its features and dominating the market of digital art. Nowadays it significantly contributes to creative processes like level design by enabling designers to create, edit, and manipulate a variety of visual 2D assets that shape the look and feel of game spaces.

Artists use Photoshop to create, edit and paint on textures for game objects, characters, terrain, and environmental elements. [56] By painting lighting and shadow effects directly on the textures, processing power in the game engine can be saved. Furthermore, texture maps can be edited and later applied to 3D objects. Its alpha maps and opacity maps allow for transparent textures and designs.

It is also widely used for concept art with many individual artists using the versatile tools of Photoshop to create detailed, realistic and individual pieces of art [1]. Concept art helps to communicate ideas to the team, guide the design process before the production phase, and ensure a cohesive visual direction by visualizing the appearance and the feeling of the game [153, p.283 ff.].

Photoshop can further help to design UI elements that can be imported to the game engines. [55] With its main focus on image manipulation, pictures and images can be cropped, changed in size in color, rotated on the axes and in perspective, and blended.

Photoshop follows a paid subscription model that is included in Adobe Creative Cloud for 49,54€/month as of today. [55] The app can also be purchased as a standalone program for 23,79€/month as of today,

currently available for Windows and MacOS. It can be used with a graphical tablet.

Autodesk Maya

Among the prominent tools used in level design, Autodesk Maya stands out as a leading choice for 3D modeling and animation. [59] It distinguishes itself from Photoshop by its primary focus on 3D art, modeling, and animation. The software was developed by Alias System Corporations in 1998 and is launchable on Windows, MacOS and Linux. One of the noticeable collaborations was Walt Disney Feature Animation who supported Maya during the development of the production "Dinosaur" [24]. Artists use Maya primarily to create 3D models of terrain, buildings, structures, and environmental elements that make up the game level, but also to sculpt, manipulate, and shape 3D geometry. It is also a popular tool for character creation, allowing for detailed modeling of facial features.

Maya facilitates the seamless transformation of simple shapes into intricate models. It allows artists to incorporate essential elements like UV mappings, texture and rigs. These structures can be implemented with rigging controllers for fluid character animation. The created animations range from idle movements of NPC characters that can be played on repeat to detailed facial reactions used on cutscenes. The 3D models and animations can be exported and imported in game engines.

Blender

Blender is another popular modeling tool that is used in level design. [75] It mainly deals with the creation of 3D models that can be used for the design of environments, structures, objects, and props that populate the game level and shape the overall aesthetics of the level. Specifically, it focuses on character modeling, enabling designers to create 3D characters that fit well into the game's visual style and narrative. Like Autodesk Maya, Blender also includes UV unwrapping tools and texture integration to apply materials onto the model. The software also provides rendering options that enable artists to simulate real-world lighting conditions within the game environment. This assists in viewing models in various stages, from basic polygonal meshes to fully realized, textured models under specific lighting setups. Furthermore, Blender also supports the rigging and animation of the models, the creation of particle effects and dynamic simulations with its integrated physics simulations.

Compared to Autodesk Maya, Blender is a free-to-use open-source program that excels in its variety and the wide range of tools for 3D modeling, animation, and rendering. [75] It is suitable for creating 3D assets, environments, and animations, making it an affordable choice for indie developers and small studios. Autodesk Maya on the other hand is a professional-grade 3D modeling and animation software that focuses on creating complex 3D characters, animations, and environments. It is commonly used in the industry for high-end production. It is not open-source, hence its full potential can only be achieved by paying for it.

Procreate

While not as prevalent in level design as Photoshop, Autodesk Maya, or Blender, shines as a digital painting and illustration app, primarily designed for use on tablets and touchscreens. [37] It finds its niche in specific phases of the design process. Firstly, it proves to be well-suited for creating concept art, sketches, and initial ideas for level design, allowing artists to swiftly outline environmental layouts, character designs, and visual concepts directly on touchscreen devices with a pen, shifting the real-world sketching process to a digital environment. Secondly, Procreate supports the creation of illustrations and storyboards that communicate the mood, atmosphere, and narrative of the level, effectively visualizing key scenes and moments within the game world. Thirdly, Procreate's painting tools come into play when creating textures and detail maps that enhance the visual quality of level assets. Particularly valuable for hand-painted textures, Procreate introduces a distinctive artistic style to the game, a quality cherished by indie developers aiming for a unique visual identity.

Rather than comparing Procreate to the other three tools, it can be seen as enhancing the creation process along these tools. All created assets and artworks can be saved as image files and imported to Photoshop, Maya or Blender for further modifications and integrations.

3.2.3 AI-aided approaches

A subject of increasing importance, not only in general technology but also in game design, is the development and integration of Artificial Intelligence (AI) [167]. Russel and Norwig (2010) define AI as the simulation of human intelligence in machines, capable of performing tasks that typically require human intelligence, such as learning from experience, problem-solving, decision-making, speech recognition, and language understanding [135]. Mathematical algorithms are used to process and interpret data according based on predefined models, thus creating a learning agent that improves in the prediction of future data, adapts to situations and improves their performance over time [123].

Over the past few years, the field of AI has undergone rapid expansion, setting the groundwork for many researches, so defining AI in the context of level design exceeds the scope of this master thesis [123]. However, it is important to at least understand how AI influences different fields and methods of level design and how it can support the game's realism and improve the development process [167]. Therefore, the following will provide an overview of the game development fields where AI is already being used, setting their strengths and weaknesses in relation to their applicability in level design.

Modeling Players

Players are an essential part in the process of game design and understanding them is crucial for developing a good game and level. [137] Schell (2009) dedicated a book to offer different perspectives for designers on the game developing process including different lenses that focus on the needs, motivations and drives of players [137, p.121 ff.]. Another approach to understand players is the usage of models refined by Artificial Intelligence. Proposed by Yannakakis et al. (2018), human player characteristics can be analyzed by studying the behavior of humans while playing a game [167, p.203]. The model can either follow a top-down or a bottom-up approach. In a top-down approach, a model is derived from theoretical foundations and hypotheses borrowed from fields like psychology, neuroscience and existing game research [167, p.208 ff.]. For example, the concept of flow by Ellis (2015) explains when a player feels aroused and is encouraged to focus on the game, describing the conditions under which players are encouraged to concentrate on the gameplay experience [35]. A bottom-up approach, on the other hand, evaluates real data derived by the player like reactions, mimics and gestures. These findings are formulated in a model that predicts future behavior in response to altering stimuli, such as varying level difficulties or shifts between narrative-heavy sequences and action-packed segments [167, p.212 ff.]. AI is used to understand player behavior and improve the game to maximize the desired effects on the player, elevating its status as a highly valuable resource in improving the design of the game. For further research on this topic, the "Modeling Player-like Behavior for Game AI Design" develops an AI-based method strives to accurately depict a human's decision making when playing games [76].

Procedural Content Generation (PCG)

While the concept of prefabs in game engines (Chapter 3.2.1) simplifies the creation and placements of assets in a game level, AI takes content generation to the next level. Procedural content generation (PCG) refers to "methods for generating game content either autonomously or with only limited human input." [167, p.151]. It extends its content beyond individual assets, encompassing levels, rules, maps, characters, objects and plants. With its automated content generation, it helps reducing the workload of artists and designers in automating processes of fabricating assets to the point that it can even generate whole levels including pathways and narrative [167, p.152]. The ethical, philosophical, and creative discussions surrounding PCG and its potential impact on designers and artists are beyond the scope of this thesis. However, those that

wish to know more about the ethics, limits and possibilities of PCG may refer to books such as "Artificial Intelligence and Games" [167] or "Procedural Content Generation in Games" [139].

NPC and Enemy Behavior

In recent years, game research joined the ranks of general AI research, aiming to comprehend, train and mimic human behavior [167]. In their paper "Game AI Revisited", Yannanakis (2012) mentions the game AI research areas that are crucial for including AI in a game [165]. Player experience modeling and procedural content generation, as discussed earlier, have garnered significant attention [166] [139]. Massive-scale data mining - the use of AI to research critical questions towards the game such as *Why do people play this game? Which gameplay parts are particularly fun? Is the game played as intended?* - has been studied by collecting and analyzing player data [165, p.4]. However, NPC AI remains relatively underexplored since it primarily focuses on rudimentary NPC tasks such as navigation and basic NPC control [165, p.5]. To infuse NPCs with human-like behavior, the prevailing approach is to create an NPC model that reacts to changes, enabling NPCs to adapt and act accordingly. Games like "Prom Week" (2011) employ cognitive and affective agent architecture to combine the choice of players with the experience of other characters to create enjoyable stories [115].

Besides the interaction between human and NPCs, extensive research has been devoted to refining NPC and enemy behavior and their corresponding tasks. In one book, the enemy AI of discovering players and NPC characters is explored in "The Last of Us" (2013) with focus on navigation, AI perception and skill handling. [131] The game models character vision with certain limitations; characters possess a field of view modeled by a cone around them, disrupted by obstacles and influenced by distance, that is triggered either when a player steps in the line of sight or combat is initiated [131, p.420 ff.]. Once in combat, it employs complex decision-making and pathfinding algorithms to determine optimal cover positions based on multiple criteria, including the proximity to the player. The skills of the enemies, such as gun fight, panic responses, investigation and hiding, are further prioritized, with the highest-priority skill being executed. Changing between different states like "Investigating", "Search" and "Hostile" allowed for dynamic character behavior that feels close to human behavior, enhancing player immersion and game realism [131, p.425].

The game "Mafia III" (2016), an open-world game with a richly populated environment, designs the above mentioned Enemy AI [52]. In addition to vision cones with raycasting, the player's body explicitly has detection points attached to it that move closer once the player is crouching and behind a cover, increasing the difficulty for the enemies to spot them. [130, Chapter 16] The enemy's reaction, while not transitioning between discrete states, depends on a linear formula influenced by factors such as distance to the event, angle between the event direction and the enemy, illumination, player speed, and the number of visible points on the player. In combat situations, the enemy AI follows a behavior to decide on target selection and firing, creating unpredictable and realistic shooting patterns. The more contextual information is given by the environment, the more sophisticated are the decisions of the enemy AI.

Pathfinding and Navigation

A substantial task of artificial intelligence in games is pathfinding and navigation throughout game levels. [98] As with enemy AI, a significant amount of research, both within the gaming field and beyond, has focused on pathfinding in various contexts [167]. The following information has solely been taken from the book "Pathfinding in Computer Games". [98]

In a static world that does not change, navigation meshes comprised of polygons representing traversable surfaces, serve as the foundation for constructing paths by connecting start and end points via accessible adjacent nodes. However, in dynamic worlds where the environment can change, agents follow a more dynamic pathfinding approach. They initially walk from the starting point to the nearest reachable waypoint, then use algorithms to identify the closest reachable waypoint to the destination, and continue from there [98, p.58]. These pathfinding operations are found in graphs that connect all nodes reachable from the start point. Search algorithms define the effectiveness, time and costs of finding a route to the end point, marking all nodes that will be visited along the way [98, image p.62].

Chatbot Games

Leaving the area of technical implementations of AI into games, the recent trend of AI-boosted chat bots leads the question whether artificial intelligence can create games by itself. So far, the last sections dealt with the capability of AI to enhance the quality of game content, but can it go a step further and generate entire game levels and games?

A chatbot, fundamentally, is a program designed for human interaction. [53] It uses supervised learning, a machine learning approach, to find the best fitting answer to a problem it was asked to solve by analyzing a pool of data and extracting its optimal solution. The first chatbot "Eliza" has been introduced in 1966 and simulated typed conversations with a psychologist. In a contemporary effort, OpenAI's Chatbot GPT (Chat Generative Pre-trained Transformer) has been introduced in 2022, marking a powerful approach to have real typed conversations with a program [4]. While other chatbots are limited in their answers and the users are limited in the size, style and direction of the questions, Chat GPT allows the users to post any discussion in any direction with any length in any style, ranging from simple questions about the weather to complex scientific questions on how to program a Docker Container in the programming language Ruby. Its advanced memory system allows the bot to memorize past questions topics and answer future questions within the same domain. According to France et al. (2023), Chat GPT is further able to copy the songwriting style of the rapper Young Thug, proving its ability to create creative content [88].

So far, it has been experimented to what extend chatbots can be included in games or how games can be made out of chatbots. A serious game called "PrivaCity" (2019) is a serious game that teaches students awareness in smart cities. [61] Its core is a chatbot that engages in a conversation with the player, explaining the safety guidelines for sharing content in social networks. A serious game environment called "EMERGO" (2012) enhances the learning effect of its games by adding a chatbot to the toolset, processing questions of the user and answering appropriately based on its knowledge pool [158].

However, only limited research has been done in regards of Chat GPT as a tool do design games. The best way of truly understanding the capabilities of Chat GPT in this context is to explore the field personally. Inquiries have been made about whether the bot is able to design a whole game, though Chatbot GPT just generates a generic answer, explaining how to design a game, rather than directly addressing the bot's ability to craft an entire game (see Fig.3.15).

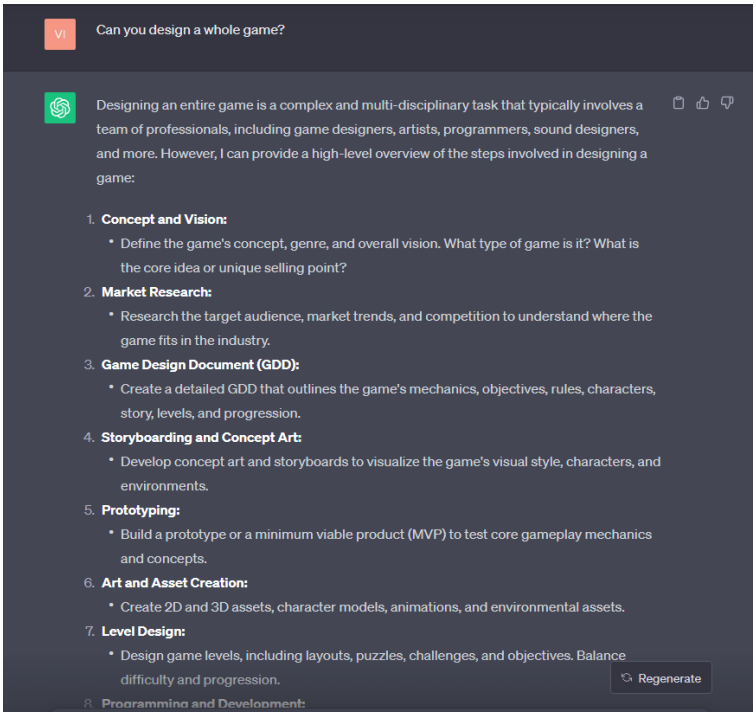


Figure 3.15 A screenshot of GPT's answer to the question whether it can design a whole game [4]

.

However, when followed by a more specific question regarding the chatbot's ability to design a 2D platformer level similar to the Mario Bros. franchise, the bot's response becomes notably more concrete, as seen in Fig.3.16:

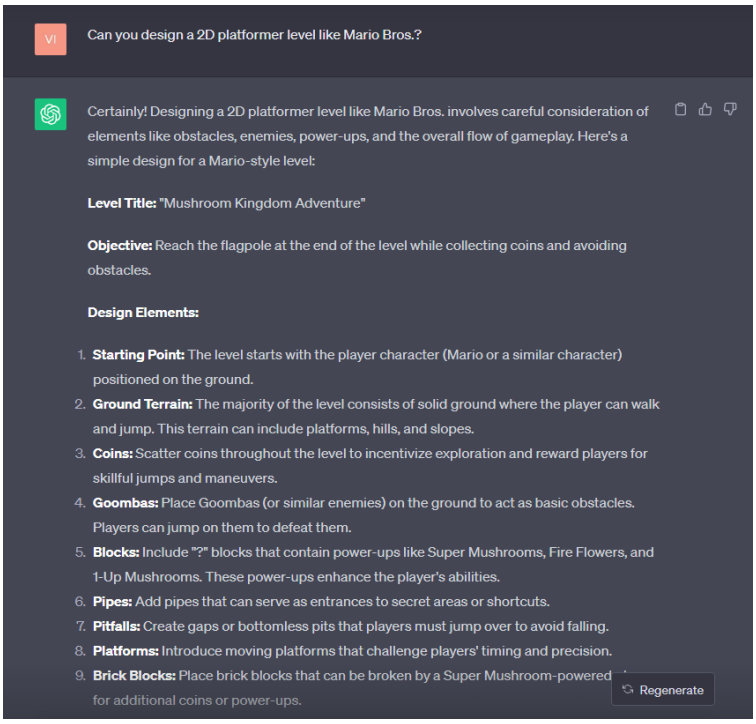


Figure 3.16 A screenshot of GPT's answer to the question whether it can design a 2D platformer like Mario Bros. [4]

.

While the outcome may occasionally appear nonsensical - keeping in mind OpenAI's statement that the bot's responses may not consistently align with coherent logic - this approach provides an intriguing approach for designing a simple level [168]. Yet, it remains an open question whether Chat GPT is also capable of designing complex, reliable levels and games. To explore its possibilities, further research has to be made.

This chapter introduced the essential taxonomy of level design and the concepts behind it. It showed the important aspects that are present in the creation of game levels, including different perspectives that span from rule-based approaches to the practical lens of Schell's (2008) design principles [137]. Additionally, it highlighted the importance of architecture in level design, demonstrating how its guidelines influence the practical methodologies and tools employed by level designers.

Furthermore, readers catch a glimpse of the technology behind level design, ranging from several popular game engines to art software. Given its rising popularity and importance in game development, AI-based methods have been introduced to help understand the implementation of technical innovations like artificial intelligence in games.

For future research, the use of Chat GPT is particularly interesting, a tool that holds considerable promise yet lacks in research in terms of its capabilities and limitations within the field of level design.

Many of the methods that are collected in this thesis build on these concepts and are specifically written down for a game development team, especially level designers, to gain inspiration and extract their values in the process of game design.

4 The Importance of a Shared Platform for Level Design

Reflecting on the previous chapter, there is a lot of published research focusing on level design and combining it with other fields of study [153] [151] [167]. For interested readers, this literature serves as a stable resource to deepen their knowledge in the widely researched field of level design and, by extension, game design in general. But for those who seek practicality over theoretical knowledge, a question emerges: Where should they start? How do they sort the knowledge provided by the literature and filter for the important knowledge?

It becomes evident that the knowledge is expansive; what is needed is not more data but a mechanism to present the knowledge cohesively. While collections of research material written down in frameworks and papers already exist, a common tool is missing that unifies all the knowledge in one place [122] [58]. During the research, it became evident that the knowledge is there, it just has to be organized and made available to a wide pool of game designers, with the possibility of extending and sharing the knowledge with fellow people of interest.

This thesis aims to create a platform to collect all tools and methods known in practice, organizing the information locally and sharing it with the world by publishing the knowledge. It further highlights collaboration as an important aspect in teams, as it aims to collect and combine information from various experts.

4.1 Collection of Tools and Methods

There is a wide range of tools and methods proposed in literature and used in practice, some of them already proposed in the previous chapter (Chapter 3). For instance, Salen and Zimmermann (2003) talk about Prototyping as a testing tool for validating the concept of a game [151]. As mentioned before, Totten (2019) combines Level Design with architectural views and introduces spatial level plans like axonometrics that can be transferred to game level plans, as well as stating that intimate space, narrow space and prospect spaces can be used by level designers to create safety, tension or fear [153]. Other methods like Mood Boards are widely used in practice to coordinate between teams and create a sense of freedom during the design process [83].

There are non-digital tools and digital tools described in literature and used in the industry, ranging from simple use of pen and paper to an in-built prototyping tool in the game engine Unity [153, p.48 ff.][45]. Digital Art Programs are instrumental in translating a designer's vision from imagination to digital reality and storage tools facilitate the organisation of the design files and the collaboration between the team members [153, p.70 ff.].

Various sources describe and explain the vast range of tools and methods, so if a game designer is new to the game they have sufficient references to draw from. But game designers are sometimes constrained by time limitations, making it bothersome to research for hours to find papers that are proposing new methods to efficiently draw from. Deadlines push game development teams to limit their resources and prioritize efficient knowledge collection [58].

The intent of this thesis is to build a tool tailored for that exact problem. By creating a knowledge tree that grows over time and can be shared with others, tools and methods can be collected efficiently in a team-based approach. Therefore, the approach builds on the knowledge of various big researchers in this field such as Totten (2019), Shell (2008) and Salen and Zimmermann (2003) [153] [137] [151]. Rather than being defined as a single tool, it can instead be seen as a selection of tools and methods relevant for a team of game developers. Therefore, it does not aim only at game designers, but at the whole game development team.

While it does not focus on an advanced technical implementation it rather aims to give an idea on what is possible, and maybe even give motivation to build an individual collection fitted to the team.

4.2 Thoughts organized locally

Before knowledge can be shared with a broader audience, it must first be catalogued in a predetermined location. People can begin by creating their own pocketbook, writing down what they want to take away from the research and illustrating it with their own pictures. This form of manual organization serves as the foundation of its digital counterpart, where notes are organized locally in a digital folder structure.

Markdown files are one of the most common tools used in the process of digital note-taking. [22] Their versatile nature enables the creation of websites, documents, books, and presentations, as well as simple notes that can be effortlessly converted into online content. Without the necessity of adding code, it provides guidelines on how content should look like on a webpage, therefore simplifying the use for people desiring to collect information through text-based entries that can subsequently be uploaded online. Local applications like Obsidian and SimpleNote process markdown files (.md) and simplify the usage by highlighting the written text, visually distinguishing headers and integrating formatted images for the user to view changes in the file in a visually appealing way [32] [42] (see Fig.4.1). People who wish to avoid these applications can also simply use a text editor like Notepad++ to open the files and work on them.

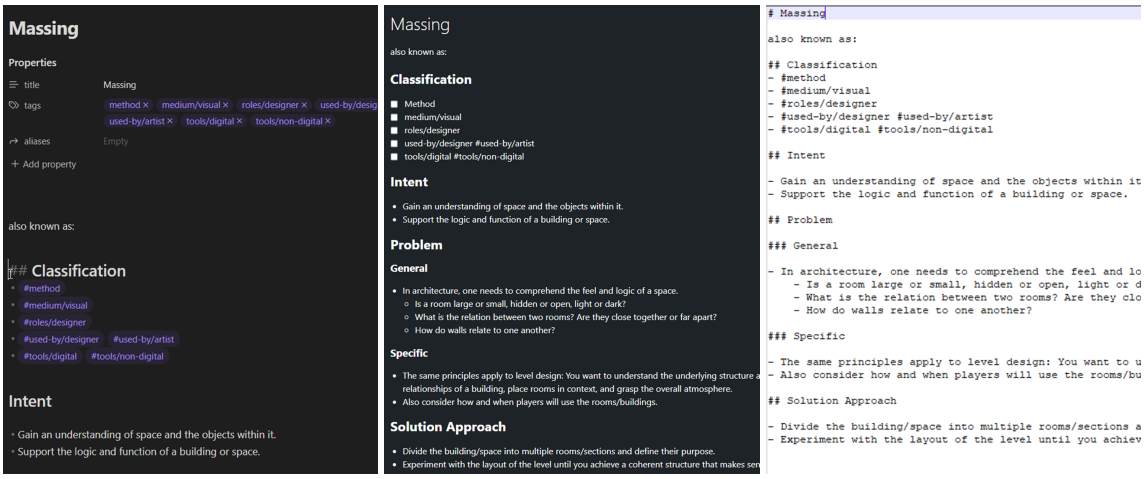


Figure 4.1 Comparison of Markdwon Files in Obsidian (left), SimpleNote (center) and Notepad++ (right)

The Markdown language is easy to use and its accessibility makes it an ideal choice for note taking [22]. By first creating a note locally, the knowledge pool expands and can be organized in folder structures. Hyperlinks further facilitate natural connections between notes, fostering a personalized Wikipedia-esque repository that is stored in its own structured file system. Over time, this repository will grow and take the form of a non-linear knowledge pool for people to use and collaborate on together.

4.3 Publishing and sharing online

After accumulating the knowledge and organizing them in a local structure, it must be shared and made publicly accessible to reach game designers, potential collaborators and other interested people. There are various options to choose from and during the course of this thesis I considered three options in particular.

One of the three options is to build a website from scratch. Markdown files offer an advantage over other languages not only in their simplicity, but also because they can easily be converted to html (Hypertext Markup Language) [25]. Once an `index.md` file has been created, serving as the start of the webpage, a

client named `gh-markdown-cli` can be installed in a local terminal program. The Markdown files can then be converted to html files by using the command `mdown --input "**/*.md" --output html` where "**/*" should be replaced by the location and the name of the file that will be converted. Within the folder structure of the .md files, an html folder will be generated. By clicking on the index.html file, a browser window opens an displays the markdown.

It is important to note that, up to this stage, the html files have been generated but remain viewable only on a local scale since a hosted website with a domain has yet to be established. To share the content with others, consideration must be given to website providers that satisfy the needs and the purpose of the file collection. There are numerous website hosts, for instance "IONOS" or "BlueHost", that each require a different setup [18] [2]. Explaining the setup would exceed the contents of this thesis since I did not further pursue the approach. Subsequently, I will delve into another prevalent hosting alternative that I set up in the beginnings of the project.

4.3.1 GitHub Pages

GitHub stands out as a popular choice for content creators due to its open-source nature, free accessibility and collaboration tools that provide a joint workspace for the contents of the website. [15] In addition to serving as a repository for code and files, its feature "GitHub Pages" allows not only to create a website from scratch, but the tool offers several pre-made templates that beautify the html pages and allow modifications from the user.

In the early stages of this project, I explored the GitHub Pages approach. Its perks of having a repository that can be cloned on any machine, and the collaborative aspect to be accessible irrespective of time and place made the tool popular among developers and convenient to use for those that are already familiar with the usage [60]. However, git repositories cannot be cloned on mobile devices as of today, thus restricting the development process to computers.

Nonetheless, starting off with a familiar tool and combining it with the new concept of creating a web-based application with GitHub Pages seemed like a logical first step at the time. After setting up and cloning a repository, I created the first `index.md` page with a simple heading and textual content to be transformed into html and displayed on the website. However, for the objectives of this project to create an interconnected pool of knowledge, simple html pages were not sufficient enough. They required not only links and connections between each other, but also the creation of a graph to display all the connections between the individual files. Furthermore, backlinks should highlight the references and connections within the folder structure.

In addition to that, the project demanded a way of converting .md files to .html files without manually converting them via the terminal. These requirements drove the search for a framework that would allow for the seamless integration of JavaScript within the project, as well as supporting an automatic conversion of .md files to .html files. Research concluded in the following frameworks that seemed fit for the project's implementation.

Integrating Vue.js and Nuxt.js

Vue, as stated in its own documentation, is "a JavaScript framework for building user interfaces". [49] Its two core features are Declarative Rendering, which offers a template syntax to "declaratively describe HTML output based on a JavaScript state" [49], and Reactivity, a service automatically updating the domain and therefore forfeiting the need to manually update and commit JavaScript changes. This framework is heavily used in frontend development and opts the possibility of easily adding JavaScript code to the website, simplifying the usage of interactive elements and graphs, including network graphs. However, to use markdowns in the project, the Marked Library has to be installed along with other adjustments. The blogpost "How To Parse And Render Markdown In Vue.js" (2012) explains the procedures and steps to set up Vue, as it again exceeds the scope of this thesis because I did not use it myself [50].

Building on Vue.js and simplifying the usage of markdown files, research put another framework into focus. It is named Nuxt.js and emerged as an attractive alternative, ultimately leading to its implementation to kick off the project. [29]

The main advantage of Nuxt.js is its focus on creating a universal app - a Single Page Application (SPA) - that avoids an empty `index.html` page by prerendering the website on the server. This strategy reduces load times and enables SEO, an abbreviation for search engine optimization [30]. As a Vue.js framework, Nuxt.js integrates seamlessly and offers a custom guide on how to integrate the framework into GitHub pages [31]. This guide was instrumental in building the first skeleton of this project.

Combined with the process of creating a GitHub pages application as explained above, the setup of the initial webpage looked as followed:

1. First, I created a repository on GitHub and cloned it on my local machine.

2. I used the starting guide in the Nuxt Manual that describes the process of integrating nuxt to the local project [29].
   In the terminal, I installed the module `npm install --save-dev @nuxt/content` which is mentioned in the guide's section "Add to a project". Depending on the package manager other commands can be used as explained in the guide. Since I already installed the packages `npm` and `yarn`, I only needed to install `pnpm` as a prerequisite.
   Files were added to my project structure. After that, I opened `nuxt.config.ts` and added `@nuxt/content` to the modules section, as explained in the second step of the guide [29].

3. Next, I created a new folder within the local project called "content" and added my first markdown files, including the index.md file which is mandatory. Every created file should have a .md ending in the file explorer, as in Fig.4.2.

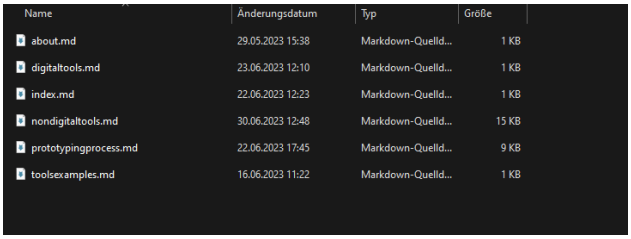


Figure 4.2 The content directory structure of my first approach on the project, including the important index.md file as well as the added "about", "digitaltools", "nondigitaltools", "prototypingprocess" and "toolsexamples" files.

4. Following the guide's "Render pages" section, I added the proposed code (image) to the [...slug].vue file in the pages directory [29]. Because it did not exist yet, I had to manually create the file (see Fig.4.3).

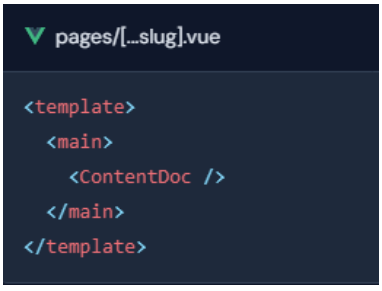


Figure 4.3 The code added to the [...slug].vue file, taken from [29].

5. After finishing with the guide from step 2, I proceed to the guide that explains how to deploy a Nuxt project to Github Pages. [31]

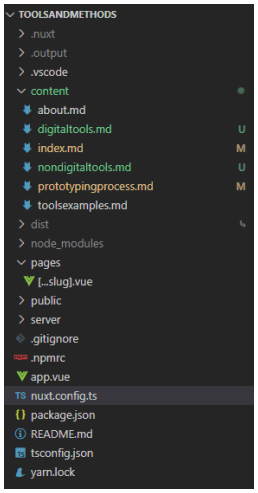


Figure 4.4 The directory structure of my first project in Visual Studio Code. The green files are the ones newly added to the git repository and the yellow ones have been modified compared to the last commit.

6. I opened the project in Visual Studio Code, but any code editor is sufficient. Furthermore, I opened the project folder in the integrated terminal of Visual Studio Code (Fig.4.4).

   I installed the dependency **gh-pages** with the command `yarn install gh-pages`, opened the **package.json** file and added *"deploy": "gh-pages -d dist"*.
   Consequently, I specified the base URL in the file **nuxt.config.ts**. Since the goal is to integrate nuxt with Github Pages and the latter uses the project repository name as domain name, I used my repository name as my URL. The name should not include any spaces or special signs that cannot be processed.

```
export default defineNuxtConfig({
    modules: ['@nuxt/content'],
    app: {
        baseURL: 'toolsandmethods' // name of my git repository
    }
})
```

7. I used the commands `run generate` and `run deploy` whenever I modified my code and markdown files. Because the framework connects GitHub with my project, the repository will be updated automatically without me needing to commit and push changes manually.

8. On the GitHub website, I navigated to Settings and clicked the **Pages** tab on the left side. A configuration page will appear that guides you through the last steps, which can be seen in Fig.4.5.

   I further made sure to enable **Deploy from a branch** as the source and selected **"gh-pages"/"(root)"** as my branch from which content will be deployed. As an addition, I could customize the domain name below but refrained from it as I was still in the early stages of the project. I always made sure to save my edits whenever I modified them.

9. The domain is accessible with the following link provided on Github:
   `https://lethamasahiro.github.io/toolsandmethods/`

   The content may take a few minutes to load once it has been updated with the commands in step 7, but it should be displayed in accordance with my defined and modified files.

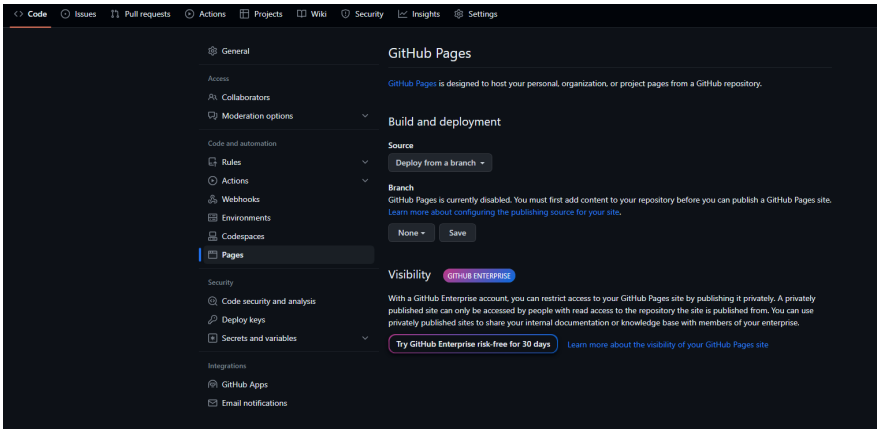


Figure 4.5 The GitHub Pages configuration options [15]

This approach converts the newly created markdown files to html files and deploys them in a relatively straightforward and intuitive manner once, given the intial setup is executed correctly. The link of this initial approach is still accessible through the provided link `https://lethamasahiro.github.io/toolsandmethods/` and shows the index page of the web-based application. Since I abandoned this approach in an early stage in favor of a more effective alternative, the pages are neither styled nor filled with much context. It rather served as a playground and learning platform to explore the linkage of markdown files and include simple style elements like buttons. Some information has already been placed inside the project structure, as evident in the link to *Prototyping Methods*, although the design remains rudimentary, primarily consisting of loosely structured text.

During the course of setting up the Nuxt.js framework, I discovered another approach that soon proved to be fruitful. It promised to facilitate the design of the website by automating any design-related coding, hence allowing the user to concentrate on collecting knowledge in a comprehensible file system. Furthermore, it displays the existent system of markdown files in an automated, visually appealing way. Its greatest feature though proved to be the generation of a visual network graph depicting all links and connections between the files, thus aiding the user understand the relation between each node of knowledge. With the first approach, there was no obvious and intuitive way of implementing a similar graph in the project, concluding in the decision to discard the approach in favor of fostering knowledge with the help of the tool **Obsidian** [32]. This chapter will not discuss the mentioned tool in detail, as it is the main tool used in the overall project and therefore explained sufficiently in the next chapter (5).

4.4 Collaboration

A game development team is dedicated to design and develop a game together by sharing ideas and complementing each others' abilities [137]. By collaborating on a shared digital medium and receiving inspiration from sources outside of the team, the quality of the project can be increased. Therefore, the ultimate goal of publishing this thesis's content online is to foster collaboration and the exchange of knowledge in the game developer community.

This goal relies on many studies focusing on the correlation between digitalization and collaboration. For example, Legnar et al. (2017) pointed out that digitalization offers new opportunities to engage in research investigation and knowledge gathering, thus highlighting the innovative character of online knowledge platforms [111]. Froese (2010) claims that digitalization and collaboration have a significant and direct impact on the overall speed, accuracy, and efficiency of the design of construction projects, concluding that both keywords find acclaim in multiple fields of study as the paper focuses on BIMs - Building Information Models [89]. Since these studies describe the general character of the relationship between teams and digitalization, they can be mapped to the collaboration in and between game development teams as well,

hinting at the increased productivity and work quality when including digital knowledge sources in the work flow.

However, this thesis does not focus on digitalization as a source of increased productivity, but rather as a means to foster collaboration between the game design community. Each game designer and interested individual possesses a pool of knowledge that others may benefit from, with research already proving that sharing the knowledge within game development teams is a powerful tool in game design [137]. The goal of this web-based application is to create a shared pool of knowledge accessible to anyone interested in game design. Following my intrinsic motivations of creating such application, the secrets of game design should not be confined within a limited range usable only by people with access to the important books in this domain, but rather made available for a wide range of people (Chapter 3). Therefore, it should be free-to-use and reachable at any time. Ideally, creators will look at the provided knowledge pool and comment on it, share their opinion and contribute to the knowledge until it grows into a well-structured resource that can be referred to whenever they seek inspiration. This may improve their creative process even to the point of changing the designers view on level design, to the extend that they try out new methods such as mood boards and incorporate them in their creative routine.

Alongside collecting, locally organizing and structuring the knowledge about level design, publication and collaboration are the core motivation of this thesis and the foundation for the next chapter. It will delve into the concept of a Digital Garden as a form of knowledge structure, provide a detailed explanation of Obsidian - the tool hinted at above -, and describe the process of creating a Digital Garden, fostering it and publishing the results for future collaboration to take place.

5 Creating a Digital Garden

The third chapter explored existing data about the process of level design in different sources including methods, tools and the underlying motivations (Chapter 3). Building upon this existing information, the fourth chapter dealt with the importance to have a shared platform for level designers to use and have access to during their design process (Chapter 4). A shared platform does not only provide designers with new methods and tools useful for the level design process; it also fosters collaboration, knowledge sharing and the expansion of the knowledge pool beyond an individual's knowledge accumulation. Although approaches exist that are driven by the same intrinsic motivations and use similar methodologies to push the research towards a unified toolset and shared vocabulary, this thesis explores the research question with a new perspective (Chapter 2). Instead of mapping existing design languages or creating a linear-structured website [160] [58] [21], it rather aims to cultivate a garden; planting seeds of knowledge, nurturing their growth, and extending an invitation to other gardeners to partake in this cultivation. The resulting product is called a **Digital Garden**.

There are no scientific definitions available for a digital garden as the word has been coined and utilized across various digital contents, blog posts and websites. The first use of gardening in a digital context stems from **Hypertext Gardening** in 1998, which describes the similarities between gardening and hypertext, emphasizing the role of hypertext in encouraging viewer engagement [62]. Hypertext is a dynamic structure, connecting data with links in a network of nodes (see Fig.5.1) [142]. Nodes can be created, linked, modified and deleted, therefore dynamically changing the structure. The World Wide Web (www) itself operates on hypertext principles, with webpages linking to other webpages and connecting their content.

According to Bernstein (1998), hypertexts can often become too complex, making it challenging to find a path and navigate through their structure, comparable to big overgrown gardens. [62] Excessive user entries and exits are expensive and can lead to confusion within the structure as they concentrate the attention on center nodes which can astray from important key nodes. As with gardens, the shortest path is not always the best path as it depends on the viewer's intention to visit the garden or hypertext. Visualizations should be used sparingly, enhancing rather than detracting from the experience. It is important to find a balance between allowing the garden to grow and controlling its growth, avoiding excessive growth while infusing sprinkles of uncontrolled wilderness. The size of both gardens and hypertexts can be proportionate to its structural complexity, expanding as complexity increases and shrinking when rigid structures dominate.

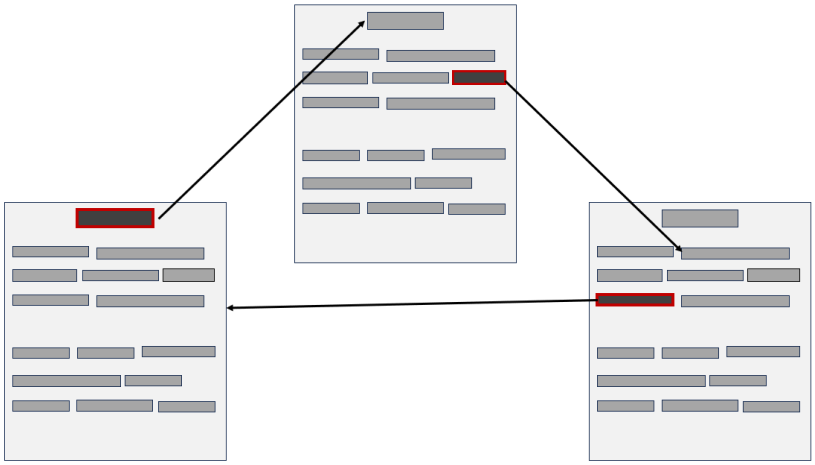


Figure 5.1 Three hypertext nodes that are connected with each other via links.

Hypertext is a powerful tool to interconnect points of knowledge by links. Collecting methods and tools for level design coupled with the identification of connections among them results in a complex structure. As it starts small and grows over time, the need emerges to tend to the structure, avoiding to lose control of the growth - a characteristic that aligns with the concept of a digital garden. Evolving from Bernstein's definition of hypertext gardening in 1998, the expression **Digital Garden** is used on blog posts discussing the clustering, organization, and nurturing of knowledge online [7]. Caulfield (2015) talks about a network of links that correlate with each other, building and fostering associations [3]. He connects the verbs "to link, annotate, change, summarize, copy, and share" [3] to hypertext and digital gardening. Here, hypertext extends beyond the world wide web to thoughts and libraries stored on a local machine.

As mentioned above, there is little scientific research on digital gardens except for blogposts and presentations [7] [3]. However, this thesis's product closely aligns with the concept of a hypertext garden, though with distinctions. The project commenced with one node and built on links to other nodes, creating a graph structure that generates, moderates and manages knowledge. The structure itself is not final, which mirrors the dynamic structure of digital gardens as they themselves grow and transform over time. However, it is not limited to web links; instead, the project connects local files that contain thoughts, ideas and products collected from different authors, leaning towards the definition of digital gardens rather than hypertext gardens. Even though the definitions of hypertext gardens and digital gardens are not scientific, albeit its usage across several applications [136] [117], this thesis's product identifies as a digital garden that is created on a local machine in Obsidian, grown over time and published on the web, offering public access to the acquired knowledge.

The following sections mark an integral part of this thesis as they guide the reader through the process of creating, fostering and publishing a digital garden specified on the tools and methods for level designers. The concept of collaborating on the data structure - an integral aspect of creating a knowledge-sharing platform, as emphasized in the previous chapter - will also be explored in greater detail. The chapter will further examine the capabilities and limitations of publication and collaboration with various note-taking applications, including Obsidian.

5.1 Obsidian

To create a digital garden, one can simply create a markdown file on the local machine and link its content to a second file using the markdown language (see Chapter 4.2). However, there are applications designed to simplify the process of managing and creating these files along with their links. One of these applications is called Obsidian.

Obsidian is a note-storing application that enables users to modify markdown files while providing tools to create, organize and associate notes via links. [32]

The application can be downloaded from its website [34]. The current version is available for Windows, macOS, Linux, iOS and Android. For the purpose of this project, which I developed on Windows, the following description will only contain information eligible for Windows users. An account is not required to use the standard version of the app.

Upon entering the app, the start screen presents three options: Creating a new vault, opening a folder as a vault or opening a vault from Obsidian Sync (see Fig.5.2). A **vault** in Obsidian serves as a container for notes to be saved in and includes additional logic for storing internal templates, themes, and other customization in .json files. Creating a vault results in the automatic creation of a new folder structure with the name and location specified by the user, while choosing to open a folder as a vault merely opens an existing folder structure. The third option, that evolves around opening a vault from Obsidian Sync, requires an account since synchronizing vaults with other users or a team is a paid feature.

Obsidian modifies files as markdown-formatted plain text so they can be opened and changed in other text editors. When a file is edited outside of Obsidian, the changes are automatically reflected within the vault

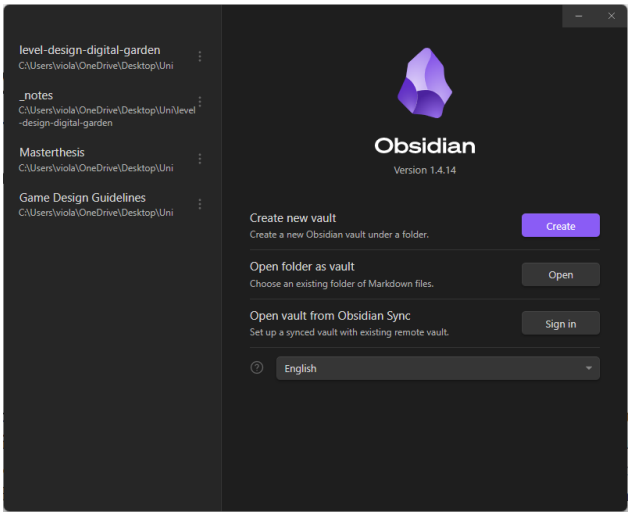


Figure 5.2 The start screen of the Obsidian Desktop application on Windows 11.

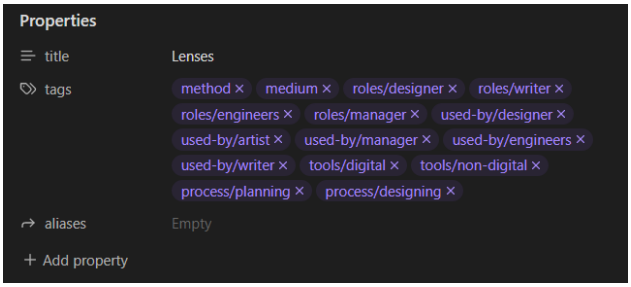


Figure 5.3 Obsidian's hashtag system used as properties in the markdown file "Lenses".

[32]. In Obsidian, the user is presented with an empty canvas and the file directory of their vault. The user can create folders and files in the directory. When opening a file by clicking on it once, the canvas shows the contents of the file. Obsidian provides real-time visualization of commands that are written in markdown. For instance, it automatically renders a title when the user writes one or more hashtags in front of a title, the font size of titles decreasing with more hashtags being added in front of the title. Unordered lists and ordered lists are automatically created when the user either uses a dash or a number followed by a period in front of a note, comment or sentence (see Fig.4.1).

External links can be generated by the following command:

```
[description of the link](link)
```

While the link is placed in the round brackets, the square brackets can be optionally filled to describe the link with a name other than the URL. Otherwise, the brackets can be left empty.
There is also a possibility to link already existing files in the vault withing the current file. Instead of using round brackets, the following command can be used:

```
[[internal link]]
```

Within the double square brackets, Obsidian provides autocompletion to help users find the file they want to link. As long as the file name does not occur twice, it is sufficient to just write the file name in the brackets. Otherwise, Obsidian will suggest all fitting files with their folder structure for the user tho choose from.

Users can further make use of tags to classify their files. Prefixing a file with a single hashtag without space between the letters creates a hashtag which is highlighted within the file (compare Fig.5.3).

The hashtags can be searched for and accessed next to the file directory. Searching for a hashtag lists all the files containing the hashtag, aiding users in finding connections and similarities in the file system.

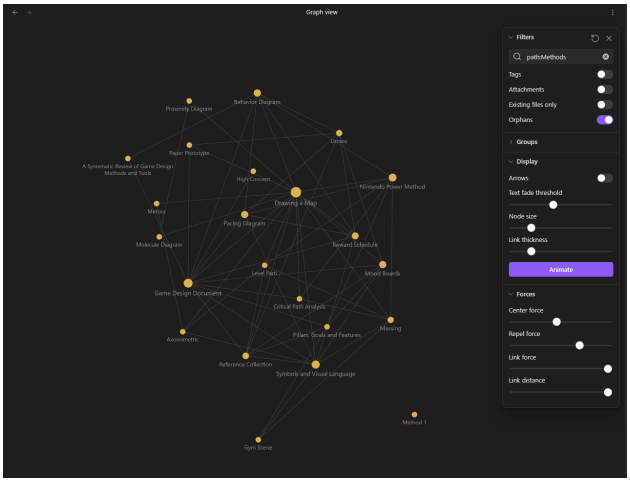


Figure 5.4 Obsidian's graph view which depicts only the methods collected in this project.

Graph view

As mentioned in Chapter 4, Obsidian comes with a graph view, a key feature that visualizes the relationships between files and their links in the form of nodes and edges (see Chapter 4.3). Nodes represent the distinct markdown files, while edges describe the links established between the files (Fig.5.4). Hovering over the nodes highlights them along with all their links and connected nodes, facilitating the identification of semantic connections between the files and the orientation within a complex graph including a multitude of files and links. The links can either be represented by bidirectional connections or by arrows indicating the direction of the link. Users are able to access the graph view on the left side of the window.
The graph view also supports **backlinks**. By clicking on a node on the right side of the screen, all references are opened, depicting links from and to other files, also called "linked mentions" by the program (see Fig.5.5).

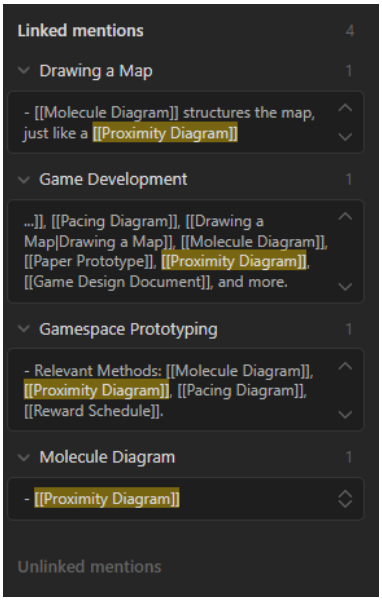


Figure 5.5 The linked mentions of the method "Proximity Diagram", mentioning the files that reference the method.

These snippets additionally provide a glimpse of the files linking to the current file, while highlighting the direct link. Consequently, the backlinks allow the user to identify frequently referenced files, effectively highlighting them as key pieces of the graph and focal points of traffic. In the graph, those center files are represented by bigger nodes compared to the others (see Fig.5.6).

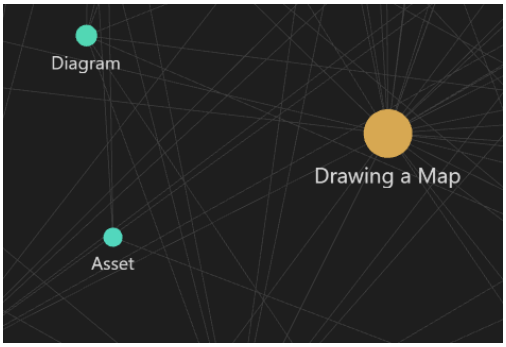


Figure 5.6 A comparison of three nodes in Obsidian's graph view. "Drawing a Map" has significantly more connections to other nodes compared to "Asset" and "Diagram", which is highlighted with a bigger node.

Additionally, customization options are available for the graph by clicking on the settings icon on the right side of the canvas, such as filters, directional and bidirectional arrows, and forces that can be applied to the compactness of the graph. Additionally, the user can choose to color the graph, assigning distinct colors to the folders defined in the vault. For instance, files belonging to the "Tool" directory can be colored in green and files belonging in the "methods" directory in yellow. This feature increases the structure of the graph and allows for a better overview.

By clicking on the wand below the settings icon, Obsidian triggers an animation feature that displays the evolution of the vault's graph over time, showcasing the addition of nodes and links in chronological order to provide users with insights into its transformation.

More plugins

In addition to the graph view, Obsidian provides various other features accessible from the left side of the screen.

Just beneath the graph view icon, there is an option to create a canvas, defined as a space for brainstorming and organizing ideas. By dragging and dropping a file from the directory, a short preview of the file appears. Users can edit and add the content of the file directly in the preview, as well as link it to other files in the canvas. This view presents an alternative method for highlighting connections between files, particularly useful for users who would like to refrain from direct linkages. While this thesis does not operate on a canvas, I encourage readers to find more information about the canvas feature on the Obsidian help page.

Moreover, Obsidian allows the creation of daily notes to document and structure the process of adding and organizing the nodes in the vault. Automatically, it adds a new note for each day. To encourage structured daily tasks, Obsidian's documentation suggests a template that can optionally be employed [33]:

```
#  {{date:YYYY-MM-DD}}

## Tasks

- [ ]
```

Users also have the option to create custom templates. Unlike daily notes, they take the shape of regular files and are saved manually by the user in the file directory. The best practice is to create a folder with the name "Template" and store all intended templates within. In the settings menu located on the bottom-left of the screen, the last option, "Templates", allows users to enable templates within their vault by specifying the location of the templates' folder (Fig.5.7). When creating a normal file, users can then click the "Insert Template" icon on the left, choose the right template which will then automatically be inserted into the file, saving time and supporting a more adjusted and consistent file structure.

Along its many shortcuts, such as pressing `Windows+Shift+N` to divide the canvas into two windows, Obsidian offers a command palette that allows the user to look for and execute commands based on their

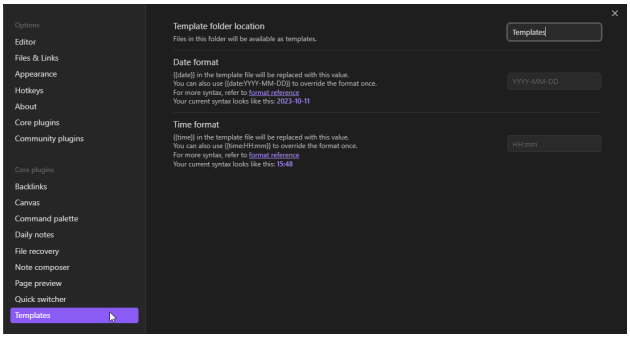


Figure 5.7 The settings to define templates in a vault.

names.
Obsidian comes with further core plugins, including an Audio recorder, Bookmarks, File recovery, Format converter, Page preview, Workspaces and more. Users can also download Community plugins once turning on the option in the settings menu, though the plugins have to be updated manually due to security guidelines [33].

5.2 Creating your Vault/Garden

After gaining an understanding of the tool used to create this thesis's project, Obsidian, the next section will cover my process of creating a vault for the tools and methods used in level design. Since I have already talked about how to create a vault and use Obsidian's built-in core plugins, this section will focus on the process of creating the project's specific structure from scratch, with the information gathered from the sources mentioned in Chapter 3.

Before documenting the information that was gathered during research, I had to establish a consistent structure to maintain across all notes. The research mainly covered the books "An Architectural Approach to Level Design" by Totten (2019), "Rules of Play" by Salen and Zimmerman (2003) and "The Art of Game Design: A Book of Lenses" by Schell (2008) [153] [151] [137], but also various other documentation, books and papers including a Google Docs document named "Level Design - In Pursuit of Better Levels" [20], and various blogs regarding specific methods for level design, such as "Single Player Level Design Pacing and Gameplay Beats" [35]. Furthermore, I took inspiration, knowledge snippets and assets from the "Level Design Book" [21], as well as distinct methods collected by the mapping studies mentioned in the second chapter Related Work (2). As the project progressed, I continued to add additional research whenever it seemed that more information was needed beyond the aforementioned literature. For example, I added findings of the paper "The benefits of playing video games" while researching on reward schedules, elaborating the importance of rewards for players [99]. It has to be noted that not every method has sufficient scientific research, for example Mood Boards and Reference collections are solely explained by one source [83]. However, most methods appear in at least one literature source and are considered useful by either researchers or practitioners of the method in the game developing field.

Once I gathered enough research, a vision began to form regarding the structure of the vault. It seemed necessary to divide the vault into folders that each cover a subsequent aspect of level design. After some elaboration and thinking, I separated the vault into the following structure:

- **Methods**: This folder encompasses the concrete techniques used in practice to help the process of level design. Methods often have a small or limited scope, offering practical solutions rather than abstract processes or ideas. They help in decision-making or produce tangible outputs that can be used in other methods or processes.

- **Tools**: This category includes the tangible instruments used to implement the methods and processes. Tools also include softwares and applications, as long as they provide an environment to help the development team in the process of level design.

- **Processes**: This structure presents broader operations containing multiple steps in a specific order. In the process steps, one or more methods can be referenced to create a tangible output. Compared to methods and tools, the processes' scopes cover more aspects, resembling a recipe to be followed. The processes vary in scale, ranging from broad procedures that aim to make the user understand the big picture - for instance the Software Engineering Workflow and Game Development process - to smaller sub-processes that are part of big processes themselves, such as Prototyping and Combat Design. This distinction creates various layers of abstraction, covering a big spectrum of processes within game design.

- **Artifacts**: This subfolder names the output of the methods as concrete tangible objects. The reason why they are included is to analyze the methods' outputs for similar products. For example, a timeline is produced by the "Pacing Diagram" method and the method "Symbols and Visual Language".

- **Roles**: This folder contains distinct roles in a development team, ranging from designer, artist and engineer to writer, tester and manager. It is important to distinguish between these roles to understand which methods and process steps focus on different aspects of game development.

- **Library**: This section covers all the references and sources that support the research. This information is stored in markdown files that directly access the source files or links to external sources.

- **Assets**: This category contains images and media used within the files.

The structure of the digital garden is independent of the approaches presented in existing literature. As the mapping studies presented in Chapter 2 have pointed out, the game development community lacks a shared vocabulary, resulting in inconsistent naming conventions for methods and tools [160] [58]. For instance, while the "Level Design Book" defines "Metrics" as a tool, the webpage itself also describes the methodology of using metrics in game design [21]. Conversely, the Tools section on the website highlights selected level editors, moddable games, engines, and art tools, assigning the term "Tools" to multiple definitions. These inconsistencies led me to decide on my personal naming conventions, as reflected above.

In the initial stages of defining the structure, there was also a debate over whether the tangible outputs of methods should be placed in a separate folder labeled "Artifacts", given that there are methods that produce similarly defined outputs. For instance, diagrams are produced with a Molecule Diagram, Pacing Diagram and Proximity Diagrams. However, including artifacts raised the issue of overlapping file names between certain methods and artifacts. "Map" originally described the method "Drawing a Map" but was quickly duplicated by the artifact "Map" which is also connected to the method "Nintendo Power Method". To address this concern, I decided to rename the method as "Drawing a Map", while avoiding other duplications in the future as they would complicate the linking process between files.

5.2.1 File Layout

After defining the structure and organization within the folders, I continued with the outline of the files. Since this project was designed to expand over time and incorporate numerous methods and tools, it was crucial to establish a common outline shared among various information snippets in each folder structure. While "Methods" need to deliver information about their inputs and outputs, "Tools" should list popular tools and their features, informing readers about alternatives to choose the one that best suits their needs. Meanwhile, "Processes" present steps for users to follow.

To tackle this issue, I referred back to Obsidian's template feature as described in Chapter 5.1. After reading and collecting information, I defined a general structure for each part of the garden, allowing users to understand and compare different pieces of information within the methods, tools, processes, roles and artifacts. These layouts were saved in a separate "Template" folder and transformed to templates that can be

directly added and modified in any file. This template is fixed and remained consistent over time, although certain methods deviated from it, resulting in the removal or modification of certain items. Some templates were added later in the project as information was progressively incorporated. In the following section, each template will be described, providing insight into their creation in the context of this project's evolution.

Method Template

The method template was defined prior to other templates since methods were the first components added to the project. The layout of the template contains the following sections:

- "Also known as": This section provides alternative names for the methods that appear in multiple literature. This information helps readers to draw connections when encountering the method under different names.
- **Classification**: This sub-header contextualizes the method by associating it with other methods through Obsidian's tag system [see 5.1]. One-word hashtags can be created and reused to briefly define the method based on roles, the type of generated output or the tools it uses.
- **Intent**: Before diving into the practical application of the method, it is important to know why a level developer should use this method.
- **Problem**: This sub-header describes the initial situation that drives the use of this method. It defines the scenario that leads developers to use this method. The problem can be further divided into two parts: The general problem (explaining a state that can be transformed from game development to a more general problem description) and the specific problem (focusing on the problem within the game development environment).
- **Solution Approach**: This section provides a solution to the stated problem without going into detail about the mechanics and process of the method. It often describes which technical or domain-specific knowledge is needed to understand this method (e.g. the solution approach for the "Molecule Diagram" method delves into graph theory). Like the problem section, the solution approach can be divided into general and specific solution approaches.
- **Application**: This section covers the process of applying the method in practice. It is divided into three components: The input (what the method needs in advance), the application (the concrete steps) and the output (the tangible outcome produced by the method and fit for further utilization in the development process). The application is the key artifact of the method description. It further states when to use the method, aligning it with the method's intent, connecting it to the documented processes, and establishing a chronological order within the level development process. Lastly, it identifies the roles directly involved in utilizing the method or the product it produces.
- **Relevancy in the following processes**: This header directly links the method to the processes that utilize it.
- **Applicability**: This section deals with the possibilities and limits of the method. It also lists the pros and cons of the method.
- **Relation to other methods**: Under this sub-header, other methods that are similar, follow or relate to the described method in any way are linked.
- **Examples**: This section provides concrete real-world examples from games or other sources to illustrate what the results may look like.
- **Relevant Tools**: This section links the type of tools that are relevant for using this method.
- **Relevant Literature**: Lastly, this section catalogues and links all the sources used in describing the method, including reference images, books, and external links.

Process Template

The process template shares similarities with the method template but focuses on the process's application and its individual steps. Each step is structured as follows:

- "also known as": Identical to its counterpart in the method template, this section lists alternative names by which the process may be known in different sources.
- **Intent, Classification and Problem**: These sections also do not differ from their counterparts in the method template, including a general and specific problem statement.
- **Process Steps**: This section deals with the distinct steps involved in the process, presented in sequential order. Each step includes its own input, description, output, when to use, relevant roles, related tools and related methods that can be used in this step. This section serves as the heart of the process layout and holds the biggest amount of information.
- **Applicability**: Like in the methods template, this section covers the possibilities and limits of the process as a whole, covering its pros and cons.
- **Related Processes**: This section defines the relation and similarities between this and other processes within the project. For example, it shows how the game development procedure and the software engineering workflow compare in several fields. It also identifies operations that can be added before or after the described process.
- **Relevant Tools**: Similar to the methods layout, this section includes all relevant tools important for the whole process in contrast to the tools relevant for distinct steps.
- **Relevant Literature**: This section is identical to the methods layout.

Tools Template

The tools template differs from the other two templates because it does not describe one process or method in detail and how to use it, but rather lists common utensils that are used in practice. Ideally, the user is able to compare the different tools and select the one best suited to their needs. It is also important to notice that the utensils within their category are not complete, but rather serve as examples to provide an overview of the tools. The layout is described below:

- **Intent**: As with the other two layouts, this section briefly explains the purpose of this type of tool for an individual or a team. Tools are categorized based on their primary function, either enabling the creation of a product or enhancing organization, communication and synchronization within a team, therefore simplifying management tasks.
- **Tool Collection**: This is the main part of the file, summarizing approximately three to five tools falling within the same category. Each tool has the following properties:
  - A name
  - An icon
  - Keywords that link and distinguish the tools across the digital garden
  - A description including its main purpose, the platform availability and a short notice on the pricing
  - The most important features that make the tool unique and help the user find the most suitable tool for their needs
  - Different payment plans (if available)
  - The roles that can benefit from the tool
  - A link to the website (if available)

Roles Template

The template for the roles is less detailed as the other three layouts. The project primarily focuses on the methods and tools of level design rather than the people using it, hence making the roles less important to the project. Nonetheless, every role needs a short description to understand the distribution of roles in the level design process and the interest of the distinct parties towards the methods. In the following, the template of the roles is briefly presented:

- **Description**: This section provides crucial information about the role, including their field of expertise and their responsibilities.
- **Tasks**: This section lists the responsibilities of the respective role within the team.
- **Especially contributing to**: This section links the methods and processes the role is keen to contribute to the most.

Artifacts Template

As mentioned above, I decided to add artifacts towards the end of the project. Artifacts define the tangible outcome of the methods, ready for further processing and storing. Rather than describing one specific outcome, they serve to categorize the type of outcome. For example, a "Document" can be produced by creating a "Design Document" or a "High Concept", noting and storing the decisions that were made in the method. As mentioned above, users can compare methods regarding their artifact, drawing connections between the methods in regards of their produced output. The template of the artifacts is stated briefly below:

- **Description**: This section describes the artifact in context with game development, including its purpose.
- **Concrete Examples**: Below the description, screenshots are placed as concrete examples of the artifact.

5.2.2 Planting Seeds

Following the establishment of the folder structure and the definition of templates, the process of creating the digital garden can commence. Over the course of this project, I did not predetermine when to create which "leaf", representing individual files, nor did I maintain a structure over the creation process. Instead, I assembled thoughts and ideas from various sources while nurturing different "seeds" simultaneously, a metaphor from the context of digital gardens describing new pools of knowledge, unordered and sprayed out [7]. For instance, the method of the pacing graph was among the initial notes written, but its completion occurred gradually, with information added in irregular sequences. Importantly, nothing within the digital garden was considered finalized until every file had been enriched with information — a process that remains ongoing. As time continues, more sources and research will affect this project, similar to the seeds of a garden that grow into a tree, constantly changing in its environment. It is uncertain how this tree will grow or which resources will be required for its cultivation in the future. The following section delves into the modifications and additions that I incorporated into the project as it continued to evolve.

5.3 Fostering the Garden

Throughout the course of this project, the need emerged to make various adjustments, corrections, additions and subtractions to the layout and the content of the digital garden.
Initially, I divided the problem and solution section of each method into a general and specific solution. However, over time I figured that this distinction did not fit every method. Some methods are grounded in theoretical concepts that extend beyond the context of level design. Take, for example, the Molecule Diagram which is used to describe spatial relations not only in the context of games, but in graph theory

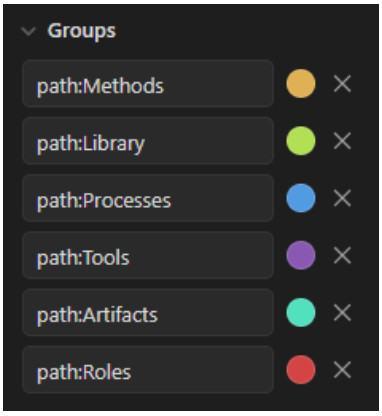


Figure 5.8 These colors distinguish the respective file directories to increase the readability of the graph in the project.

as well. This method serves as a solution for games, specifically, but also outside of the context which is why it makes sense to describe the general and the specific problem separately. However, methods like the Nintendo Power method are specifically designed for the game context and have no usage outside of game development, hence it became superfluous to divide the problem and solution into general and specific sections. During the project's growth, I largely abandoned this distinction into general and specific sections, except for methods with a more theoretical foundation like "Pacing Diagram" and "Lenses".

While collecting the different tools, it became clear that the definition of "Digital Collection Tools" and "Storage Tools" overlapped significantly, mostly listing the same concrete tools that have functionalities covering both definitions. In essence, both Digital Collection Tools and Storage Tools deal with saving thoughts, media and decisions in a virtual repository - a folder, vault or other virtual place -, providing accessibility from different sources while preserving a certain form of order. Since "Digital Collection Tools" include websites like Pinterest, which focuses on collecting references but still permanently stores files on its server, I decided to omit "Storage Tools" and link all files previously associated with them to "Digital Collection Tools" [36].

It was further clear from the beginning that "Non-Digital Design Tools" do not entail specific software, but instead aim to show traditional methods applicable to level design. Therefore, its layout excludes features, payment details, and logos, rather focusing on the tools itself and the roles frequently using these tools.

The pool of methods grew over time. Along with gaining more knowledge from the books and other sources, the comprehension regarding methods, tools, processes, roles and artifacts deepened, clarifying their characteristics and pointing out their distinctions [137] [151]. This comprehension also highlighted the aspects that were still missing in the project. For example, I added the method "Metrics" to the project relatively late, along with the role "Player", as players, though not part of the development team, play a crucial role in methods that require direct player feedback, such as the Nintendo Power Method [153]. The graph also helped finding the files which lacked links to other files, indicating their unimportance to the knowledge pool and urging me to exclude them from the project. So far, though, no files have been excluded as they all provide their own knowledge to the digital garden.

Along with the growing number of files, the complexity of the graph increased, making it challenging to comprehend due to its quantity of interconnected lines and nodes. To enhance readability and comprehensibility, it became vital to make use of the color feature the graph settings provide. I decided to color the dots according to their file category. The methods were colored yellow, the roles red, the processes blue, the tools purple and the artifacts turquoise (Fig.5.8). This was one of the most important changes to the project as it improved the readability and navigability of the graph.

The last change dealt with the literature and source management. It is interesting to see which sources have been gathered specifically for one method and which sources spanned over multiple methods. To increase the visibility of the connections from the literature to multiple methods, a whole new directory of literature was established. This directory contains markdown files that can be linked to, each housing either a direct

link to the source or the source content itself, often in PDF or image format to preserve source validity. In the graph, the nodes of the sources are colored green.

5.4 Publishing

This section regards the process of publishing the vault to enable access and reading on the web. The last chapter already described the first approach of using GitHub pages and Nuxt.js to publish the digital garden (see Chapter 4.3.1). However, the solution shifted towards a method that is compatible with Obsidian. Though said application also offers a solution called "Obsidian Publish", a paid option that serves as an alternative to the currently chosen method, I decided in favor of a publishing method described in the guide "Setting up your own digital garden with Jekyll" [19]. This section will exclusively address the process of publishing with the chosen method, with the pros and cons of Obsidian Publish discussed as an alternative in Chapter 7.

Rather than separating the processes of fostering, growing and publishing the digital garden and executing them sequentially, it comes more natural to tackle them simultaneously. Publishing the vault early in the process increases its testability, analyzing what has been created can also be depicted in the web accordingly. Following some research, a guide introduced a way of publishing a digital garden to the web, independent of the platform or application used to create it. This guide is outlined on this website:

Setting up your own Digital Garden with Jekyll

After creating and setting up accounts for the required tools called Ruby, RubyGems, Git, GitHub and Netlify, I forked the Jekyll template to a private GitHub repository. This defined the building block of this project, its repository named **digital-garden-level-design** [112]. The fork could then be cloned directly to the local machine. As anticipated, this process did not synchronize well with the project stored in Obsidian, rather creating a new empty project on the machine utilizing the Jekyll template's existing files. Sadly, there was no straightforward way to copy and paste the Obsidian project into the new cloned project. Instead, I created a new Obsidian vault with its destination folder set to the git clone, moving the directories manually to the new location into the `\_notes` folder. This created a new folder structure with both git and obsidian directories (Fig.5.9). Obsidian also recognized the change in the project's publication and automatically added a header to each markdown file, labeling them with titles for the future web application to recognize them as webpages.

Subsequently, the guiding website proposed testing the webpage locally by entering the command `$bundle exec jekyll serve` in the terminal after navigating to the directory, additionally using `$bundle$` before since the package has not been installed yet. Unfortunately, the result of this command was just a blank page on the localhost domain, so it was quickly discarded in favor of focusing on achieving a functional online webpage with Netlify [27].

This platform is serverless and accessible for individual use, facilitating the building, deployment, updating and scaling of web applications. In contrast to GitHub pages, it does not build directly on the git project but is capable of integrating it, offering multiple deployment methods [16]. It also allows JS frontend frameworks and provides a drag-and-drop deployment option, enhancing the visual aspect of the deployment process (see Fig.5.10).

In hinsight, this project could have also been deployed with GitHub Pages alone, but at this project stage I decided to continue with Netlify. After logging in, a new webpage can be generated in the "Sites" tab on the left side of the canvas, by clicking the button "Add new site" (see Fig.5.11). Following the steps on the above linked guiding website, GitHub can be selected as the continuous provider on the next page. After clicking "Next", additionally authorizing the access to the git account, the repository for the digital garden can be selected. The site can then be deployed by keeping the default settings.

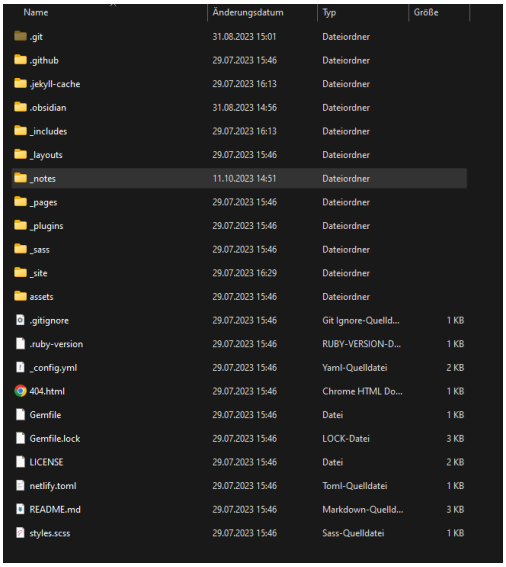


Figure 5.9 The new project directory that combines GitHub with Obsidian files.

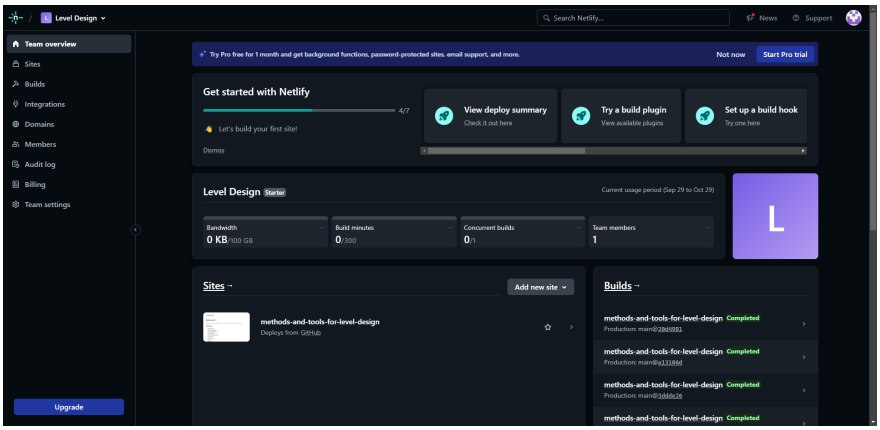


Figure 5.10 The Netlify Start Page once the user is signed in. It already shows a preview of the team "Level Design".

The deployment process typically takes a couple of minutes until the pages show on the link provided for the project [28]. Once the webpage is initialized, Netlify keeps track of any changes done in the repository and updates the webpage automatically (Fig.5.12). Locally, any modifications made in Obsidian must be added, committed, and pushed to the repository.

The initial start page, as defined in the project under `_pages/index.md`, presents a brief welcoming message and lists the most relevant processes, methods, tools, and roles which are determined by their node size in the graph, empowering users to determine their starting point (see Fig.5.13). Like every markdown file, this file can also be modified and edited in Obsidian. When the user hovers over a link, a short preview of the connected page is shown with the title and a preview of its content. Once opening one of the connected webpages, backlinks are displayed on the right side of the editor, linking to the respective webpage entries. The bottom page illustrates the graph view with an automatically generated sentence, from which the other pages can be accessed (see Fig.5.14). However, Obsidian's graph settings, such as the color filter, the forces or the bidirectional links, are not applied.

The top of the page links back to the start page.
Unfortunately, Netlfy lacks to support the hashtag feature from Obsidian, rendering them less useful in the web context. Nonetheless, they remain functional within Obsidian, to organize the working process in the

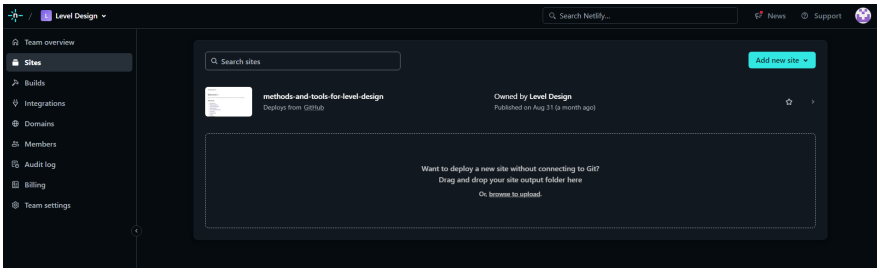


Figure 5.11 By clicking on the tab "Sites", a new project can be added on Netlify. In this case, the project "methods-and-tools-for-level-design" has already been added.

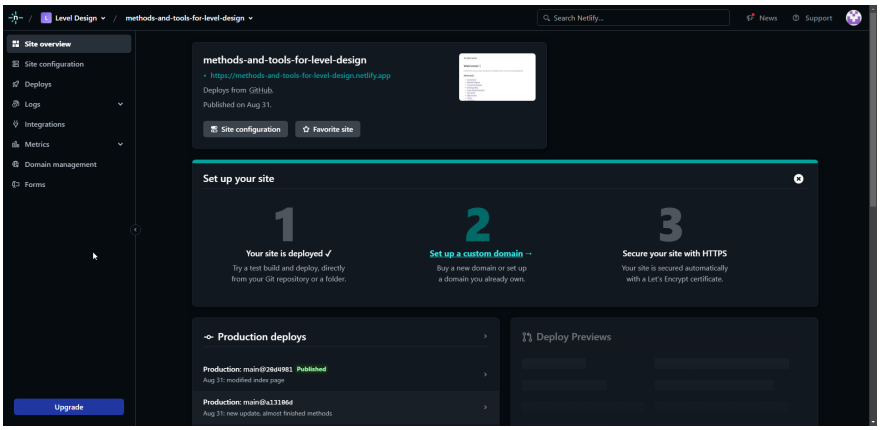


Figure 5.12 The Netlify control page of my project with a preview of the rendered page.

vault. This is also an argument for potentially using Obsidian Publish in the future, as it preserves Obsidian's features.

5.5 Collaboration

The goal of this project extends beyond creating a digital garden and collecting important methods for level design; it also aims to collaborate and share information with each other. Achieving this goal involves enabling multiple users to work on the project simultaneously.

Up to this point, I have been the only contributor to the project, working on the vault locally and publishing the results to Netlify for the application to deploy the vault. With this workflow, it is only possible to collaborate on the project with GitHub. Every person that gains access to the project and clone it to their local machine can edit the files provided in the project. Instead of creating a new vault, they download Obsidian, sign in and open an already existing vault by selecting the cloned GitHub project in their local directory. When collaborating on a shared project, the collaborators have to follow the git commit rules of adding the files that have been modified, committing them with a meaningful message, then pulling possible changes from other users, eventually solving merge conflicts and pushing the changes to the repository. Effective communication among collaborators is vital, particularly in case of two individuals seeking to modify the same files. This communication ensures information sharing and the collaborative growth of the garden, as opposed to a single individual filling the vault with information.

There are further collaboration options, one of them included in the Obsidian Publish option that has to be paid to gain access to. In the future, exploring the option of incorporating the project into my mentor's vault may be worthwhile. However, for the scope of this master thesis, Obsidian Publish is not a part of the project.

The last sections covered the process of creating the digital garden, fostering it, and publishing it to the web for level designers to access. This project does not aim to be complete. Instead, it serves as a dynamic knowledge base that continually grows, also by collaborating with other experts who may offer diverse

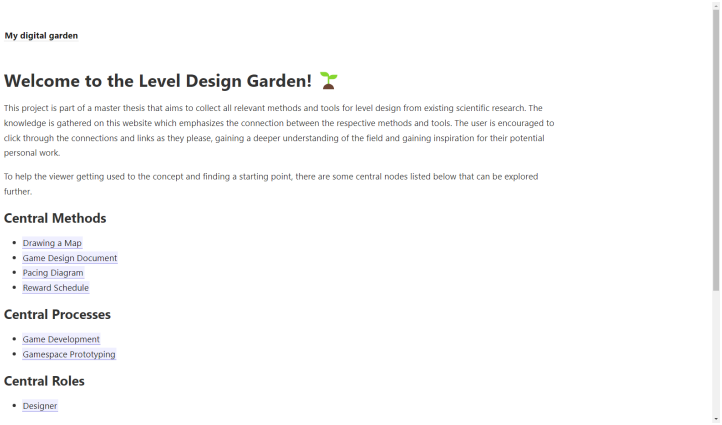


Figure 5.13 The start page of the digital garden, hosted by Netlify.

Figure 5.14 The graph shown on the website, as opposed to Fig.5.4 without colors. Upon clicking on the nodes, the respective page opens to read through.

perspectives on methods, tools, and their distinctions. The next chapter will cover the results of the process, outlining its limitations, possibilities, and the goals achieved.

6 Results

Regardless of what worked well and what could have been improved, this chapter outlines the end product, its defining characteristics and the qualities it brings to the level design community.

After dedicating four months to the vault, numerous methods and tools have been collected and integrated into the digital garden. There is a total of 21 methods, 7 tool domains, 5 processes, 7 roles, 14 artifacts and around 27 literature nodes with the tendency of expanding with more information being added in the future. In the graph, every node is connected to at least one other node, except for the templates that are stored separately but remain part of the graph. The graph itself is complex and the nodes are connected by many edges, creating a singular, coherent entity resembling a blop, or cookie as my supervisor calls it (compare Fig.6.1).

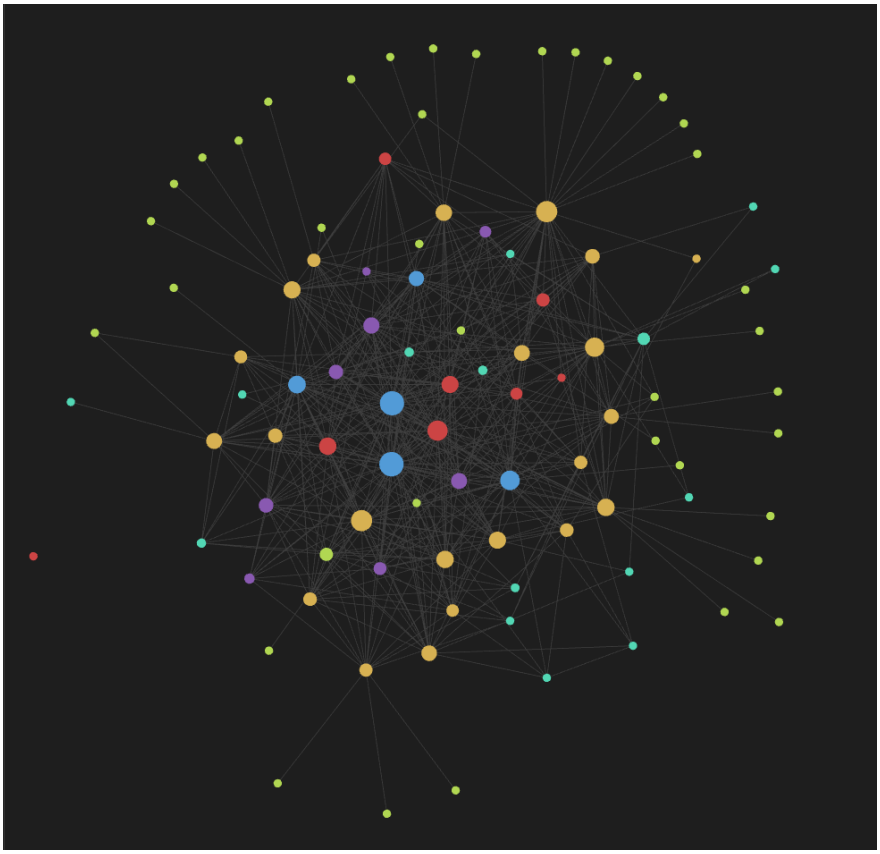


Figure 6.1 The Obsidian graph view at the end of this project. The different colors of the nodes depict the different categories, the size of the nodes highlights their centrality and importance.

In this web, the most central and biggest process node is "Game Development" with more than 30 connections to the other nodes, closely followed by the process "Gamespace Prototyping". The "Designer" role emerges as the most interconnected role, again with more than 30 links. The most referenced methods include the "Game Design Document" (about 22 links), "Drawing a Map" (about 27 links) and "Pacing Diagram" (about 18 links). Among the tools, "Non-digital Design Tools" have the most connections with

about 17 links. The most connected artifact is the "Diagram" with around 10 links . In terms of literature reference, "Totten 2019 - An Architectural Approach of Level Design" is referenced the most.

Each file within the vault contains a minimum of 100 words, with some files extending up to 1500 words, such as the "Game Development" process. I opted to write the content in a bullet-point format since it is easy and fast to read through, providing compact information in a structured and organized way. Depending on the content's scope and the accessibility of relevant literature, crafting each file took from half an hour to three hours.

With this digital garden, beginners, experts and experienced level designers alike can explore the methods, tools and processes used in level design. They can access the project's website with its link and are greeted by a start page that links to the most central material in the respective categories (see Fig.5.13).

When reading a selected page, users can navigate to connected pages, either through backlinks or by clicking on the desired node in the graph at the bottom of the page. Additionally, users can return to the start page at any time by clicking the corresponding link at the top of each page. At the bottom of each page, they have an overview of all the relevant literature concerning the respective knowledge snippet, sometimes linked to external sources.

Regarding the goals specified in the beginning of this thesis, the digital garden comprehensively covers all identified methods gathered from background knowledge on level design, primarily sourced from books such as "An Architectural Approach to Level Design," "Rules of Play," and "The Art of Game Design: A Book of Lenses" [153] [151] [137]. The connections are made both directly and indirectly, either by referring back to the source material or by drawing connections based on personal experience and knowledge derived from these sources. The importance of the distinct pages developed over time with more and more connections being found between the pages.

It was clear in the beginning that methods, processes, and roles could be drawn from existing literature, but a similar approach for tools was less apparent, as the term "tools" is often used interchangeably with methods in existing literature. To distinguish tools from methods, I actively sought out concrete tools utilized in level design and organized them in the "Tools" sub-category. Consequently, this project presents the information informally to the reader, given that I provide the information through a subjective lens.

Artifacts were not initially part of the project but emerged as a natural consequence of ongoing work. The inclusion of artifacts in the project simplifies the demonstration of similarities between methods that are not apparent when merely reading about them. Artifacts represent the direct, tangible output from the methods, clarifying said semantic connections between the methods as they might produce the same technical output, although their purposes and applications differ. While this connection may be informal and not supported by existing literature, it encourages readers to think about their preference regarding the type of outputs, such as textual or visual outputs.

In addition to just reading the pages, the digital garden supports collaboration. To participate, individuals must gain access to the git repository and follow the steps outlined in the previous chapter (5.5). Those wishing to add information to the garden can contact me via email to discuss potential collaboration on the project.

Again, this project is not intended to be final or finished, but instead it is an ongoing work in progress. Similar to the "Level Design Book" which is still under construction, the garden is designed in a way to simplify the processes of adding and modifying of information [21]. Thus, in the future, it is expected to scale and grow with more methods and tools, further enhancing its utility to level designers in streamlining their processes and workflows.

7 Discussion

This chapter will comprehensively discuss the project's relevance concerning its research question and its alignment with existing work in the field. It will further interpret the results and address the project's quality, encompassing implementation challenges, limitations, opportunities, and stating alternatives. The last section will hint towards additional methods that could further support or refute this thesis's contribution to its research field.

7.1 Concept

As stated in the introduction of this thesis, a pivotal objective towards a shared vocabulary is to collect important methodologies and terminologies from different sources, summarizing them into a singular accessible resource for interested parties to draw inspiration from. The core motivation and underlying concept of this thesis has not changed, though it remains to be answered whether this approach adequately addresses the issue of level designers lacking time to filter relevant methodologies from existent literature. Having dedicated over five months to this project, I can confidently affirm that it indeed confronts a contemporary challenge, one that is currently under-researched. This digital garden was created to let level designers and other parties have access on important methods and tools used in level design, supporting their creative process, providing new sources and alternatives for them to explore. To further prove its raison d'être, I will revisit the research question stated in the introduction and align the project's outcomes with the related work.

7.1.1 Research Question

The primary research question of this thesis sought to answer whether methods and tools for level design can be summarized to a shared vocabulary and effectively collected on a shared online platform, highlighting the interconnection of its entries and the scalability of the knowledge pool.

The initial component of the research question, the shared vocabulary, has been effectively addressed through the inclusion of numerous methods, processes, artifacts, and tools within the project. The mere act of dividing this knowledge into sub-folders represents a new organisational approach. Taking the approach's validity and scientific rigor out for a second until discussed in further sections, it proves that a shared vocabulary can be found.

Whether the approach has been efficient or not, the project offers a solution that collects the knowledge on a shared web-based platform. The unique feature of the graph view, characterized by its non-linear structure, highlights the interconnectedness of the nodes rather than providing conventional hierarchical framework. This allows the reader to fully decide which specific fragments of knowledge they want to focus on. Moreover, this structure supports the project's scalability by simplifying the process of adding new nodes with markdown files and connecting them through links to the existing knowledge pool. The hashtags further hint to connections between nodes across subfolders. Additionally, the creation of artifacts as outputs of methods enables readers to discover similarities between each other through the comparison of their respective outcomes.

7.1.2 Context: Related Work

Throughout the project's development, the central focus always circled back to the core research question, emphasizing the importance of the different aspects defining the research question, such as the shared vocabulary, the shared online platform, the interconnection of the entries and the project's scalability. The project not only needs to address these specific aspects but also deal with the more abstract essence of the question:

Whether any type of knowledge pool derived from literature can be derived to support the process of level design. Therefore, the mapping studies and the level design book presented in the related work proved to be very helpful, but they did not fully encompass the multi-dimensional scope addressed by this project [21] [160] [58].

The mapping study "Languages of games and play: A systematic mapping study" primarily provides a solution for categorizing languages into formalized domains, sharing a common motivation with this thesis to establish a shared vocabulary but diverging in its approach [160]. While the second mapping study "A systematic review of game design methods and tools" also shares common motivation, it abstains from practical implementation entirely [58]. Consequently, while both mapping studies make significant advances in terms of a shared vocabulary and its importance, they do not offer a holistic solution for interconnecting different level design methodologies collected from literature. They further refrain from summarizing similar languages and methodologies into one single file, but instead only categorize them in domains, if they categorize at all. This thesis offers a more fitting response for the specific research question as it introduces a practical, tangible platform that extends the mapping studies. It not only extracts knowledge from these studies but also references them in methodologies that make use of their particular findings, such as the "RogueGame Language" mentioned in the method "Drawing a Map" [160, p.25].

On the other hand, the Level Design Book approaches the idea of the research question more closely. [21] It is very practicable and detailed, providing numerous explanations of commonly used level design concepts in a linear fashion. Many references were taken from the book and its full sentences might prove more fitting to the knowledge-sharing character compared to the bullet points used in this project. However, compared to this project, its linear structure does not align with the scalability requirement, which emphasizes interconnections over a linear narrative. It also lacks the collaborative and sharing aspects crucial to this project. While the Level Design Book may serve as a valuable resource for level designers, particularly due to its detailed content and user-friendly format, it fails to address important insights from relevant books such as "An Architectural Approach to Level Design" and "The Art of Game Design: A Book of Lenses" [153] [137]. It also has to be noted that the Level Design Book has not been updated since 2022. Conversely, this project provides a more flexible design and incorporates sharing and collaborating aspects with an up-to-date knowledge base. On the other side, it is not clear whether readers prefer a linear or non-linear approach to the thematic. To test the project's significance relative to the Level Design Book, user evaluations of both websites, potentially involving surveys for data interpretation, would be necessary.

7.2 Implementation

In consideration of the above-interpreted related options, there are still questions to be answered with solely this project. This section introduces the challenges that arose during the process of creating the digital garden, states the limitations that could not be overcome, and spotlights opportunities that, while proposed by this thesis, remain unexplored. Additionally, as explored in Chapter 5, this section presents two other publishing alternatives beyond Netlify.

It is important to mention that this section does not aim to make the project seem perfect. Instead, its purpose is to engage in a critical examination of the project's evolution, highlighting areas that could have been executed more effectively and aspects that went well. This review extends beyond the mere implementation, including the achievement of the overall concepts, ideas and rationale behind the decisions made in this project.

7.2.1 Challenges

During the development of this project, several challenges emerged, spanning in their variety and scale.

Arguably, one of the biggest challenges was to find a shared vocabulary that spanned all the collected researches, defining the vocabulary from scratch if none was found. It is evident from the related work that

similar or same methodologies are often characterized by numerous titles, such as "Design Patterns in FPS Games" [160, p.41] and "Combat Design" [20] that both describe methods to design combat. Even within research, there is no universally accepted definition of tools and methods. For instance, Totten (2019) regards tools as equivalent to methods and techniques [153], while Van Rozen (2020) defines tools as "theoretical foundations, systematic techniques and practical solutions" [58, p.4] that increase the productivity of level designers. These variations in terminology increased the difficulty of summarizing the numerous amount of research into a coherent structure beneficial for level designers. The proposed division into methods, tools, artifacts, roles, and processes, while not without flaws, was considered the most practicable solution to underscore the distinctions presented by the research. However, the distinction poses potential confusion, particularly concerning tools. Given the context, tools can refer to concrete software or programs that aid in the development of games in different categories, but they can also pose as aliases to methods since research sometimes uses the two words interchangeably. Similarly, the term "artifacts" may define in-game collectibles or tangible outcomes of the methods. To reduce confusion for the reader, each method and process is equipped with aliases that encompass synonyms and similar terminologies encountered during the knowledge compilation process. Regarding the naming of the categories, such as "Methods", "Artifacts", and "Tools", suggestions for refinement are welcome.

Over the course of collecting and reformulating knowledge, the issue of lacking sources became evident. While some methods, like the "Game Design Document", have numerous sources to draw information from [20] [14], others, such as "Mood Board", possess only one reference [83]. This issue raised questions about the significance of a method that lacks substantial coverage in relevant literature. Is a method less significant to the research field if it is lacking the sources to support this method? Unfortunately, the graph does not hint at the lack of significance in the literature, since the size of the nodes depends on its interconnection to the other nodes and not only to the literature references. Presently, all methods are gathered in the project regardless of its support by underlying literature, acknowledging the novelty of level design as a scientific research field. Future research may possibly discover additional methods and tools crucial to level design.

Furthermore, the project encountered a challenge in addressing the scientific validity of its sources. While certain methods like the "Nintendo Power Method" are presented in scientifically acclaimed books [153], others, such as "Pillars, Goals, and Features," were solely derived from internet blog posts [5]. To maintain this project's scientific credibility, I sourced out scientific publications whenever possible. However, exceptional cases of methods without the necessary scientific proof are still included in the digital garden. This poses the question of whether non-scientific resources suffice to justify a method's inclusion in the digital garden. For now, I kept the methods that I deemed significant to the research field, but future work may seek to locate sufficient scientific research to support the method's existence in the digital garden.

7.2.2 Limitations and Opportunities

As this project nears its conclusion, its results paint a clear picture of its opportunities and limitations. However, it is important to consider that the end of this thesis does not mark the end of the project, nor does it mark its finality. The non-linear structure allows nodes to be added easily, interconnecting them with existing knowledge. Furthermore, the project's flexibility allows for modifications and corrections to be made to the single files, should there be any inaccuracies.

Nevertheless, certain aspects are bound by the project's structure, hence making it impossible to achieve absolute flexibility. For example, the project's primary concentration on addressing the research question prevents it from transforming into a supplement for the related work. The "Level Design Book", a valuable resource for the level design community, summarizes common knowledge and presents information in a structured, comprehensive manner [21]. The first mapping study is very detailed and subtracted numerous languages from a variety of sources, introducing a unique approach of categorizing specific methodologies in domains [160]. Both results mark a distinctive approach to address parts of the research question, which

cannot be supplemented by the project, as they draw inspiration from other directions and sources. This project rather extends the related work by referencing them in the proposed methods, tools and processes.

As a solo project, this digital garden further reflects the limitations of individual capacity, rather showing a glimpse of what can be achieved with its manageable amount of 21 methods. On the other side, the project's source graph is already complex, with many connections made and many lines drawn. Balancing the increasing complexity of the graph with the need to preserve order over all entries is a present concern. Reducing the number of connections and discussing their validity with multiple experts could structure the graph in a more sustainable manner, facilitating the addition of more nodes without multiplying its complexity. Nonetheless, the fundamental problem remains: As the knowledge pool grows over time, the complexity of the graph will increase. It is an intricate problem for future research to find a balance between maintaining structure and accommodating growth in a rapidly evolving knowledge pool.

While finding this balance, the project is allowed to grow not only by the hand of the creator, but also by the hand of collaborators. The possibility to share the project with other people holds the potential for communal knowledge creation and sharing. In the future, the project aspires to become a collaborative platform, inviting experts to contribute their knowledge to the existing digital garden, increasing the knowledge's breath, depth and value. The initial step in said direction involves combining the vault of this thesis with the vault of my supervisor Daniel Dyrda, expanding his knowledge pool with my own interpretation of the methods, tools and their interconnection. However, it might be a challenge to combine two separately built structures, refraining from duplicates and structural incompatibilities. Despite these hurdles, both vaults could significantly benefit from each other by extending and complementing each other's work.

Yet, the project's web-based nature introduces its own set of considerations. Presently, the website's accessibility has only been tested on two browsers of the same computer. It remains unexplored whether the website is equally accessible from other geographical regions or operating systems besides Windows. The possibility of offering the vault in multiple languages remains for future research to be analyzed, too. While introducing languages like Spanish could broaden accessibility, the manual translation of each markdown might encompass too much work to be considered valuable. Moreover, the problem emerges whether two different languages can be integrated into one single vault, potentially complicating source graph interaction. Ultimately, finding a pragmatic solution that balances multilingual capabilities with the preservation of a coherent structure is a research opportunity that can further be explored in the future.

Another limitation is posed by the website's UI design. The layout of the website is shaped by the inherited structure of markdown files, with simple structural characteristics used to define titles, highlight important words and add bullet points, images and hashtags. The title page stands out as a basic page with only one short welcome notice and five lists that contain the most important methods, tools, processes, roles and artifacts available to read through. This characteristic leaves room for improvement, as the main page, in particular, could have been modified to create an inviting, visually engaging interface. Providing improved navigation options, a better structural layout, and enhanced search functionality would enhance the reader's experience. Presently, the main page serves primarily as a basic structure, allowing direct access to any of the knowledge entries and initiating exploration the user's selected point.

It further has to be noted that the website does not provide the user with the same experience as in the Obsidian application. The absence of hashtags is noticeable and the knowledge graph does not apply the same organized structure as Obsidian's graph (compare Fig.6.1 and Fig.5.14). Although images can be displayed after relocating them in the publish directory of the website, I could not figure out how to display pdfs, hence the user is occasionally presented with empty links (see Fig.7.1). This could be solved by using another publishing provider.

7.2.3 Publishing Alternatives

When considering the project's limitations and opportunities, it is worth commenting that this project is currently published by the provider Netflify. However, there are two other publishing alternatives that are worth being considered.

Figure 7.1 The Netlify page cannot find the link to the pdfs even though they are linked correctly in Obsidian.

Obsidian Publish

Obsidian Publish is an inherent feature provided by Obsidian itself. 5.1 It allows users to publish their vaults online without further set ups. Since it is a sub-product of Obsidian, it subsequently shows all hashtags, links and the source graph as intended by the application, with more options to be explored if the project uses additional features. However, this alternative is a paid option and costs 8$ per month. Given that my current approach does not incur any costs, I refrained from integrating Obsidian Publish into my project in the beginning. However, as my supervisor's vault is already published using Obsidian Publish, this option will likely be adopted when this project's vault will be integrated in his. This alternative might also prove more valuable once the project grows in volume, as Obsidian automatically updates the provided website with any changes made to the vault.

GitHub

Another publishing option is Github Pages. [15] In hindsight, this option could prove more valuable than Netlify, as I had initially experimented with publishing during my initial approach (Chapter 4). Rather than importing the GitHub repository to Netlify, I could have simply enabled the Pages settings discussed in Chapter 4.3, decreasing the amount of configuration made to publish this project. However, it is important to note that the Netlify pages include the graph view at the bottom of each page, a feature not supported by GitHub Pages. This discovery supports me in my decision, as the graph view is integral to the project, even if it does not display with the precise forces applied or the colors assigned to the structures. GitHub Pages also does not format the markdown files with additional beautification, resulting in a plainer appearance in contrast to the pages created by Netlify. Fig.7.2 shows the visual difference between these tow approaches.

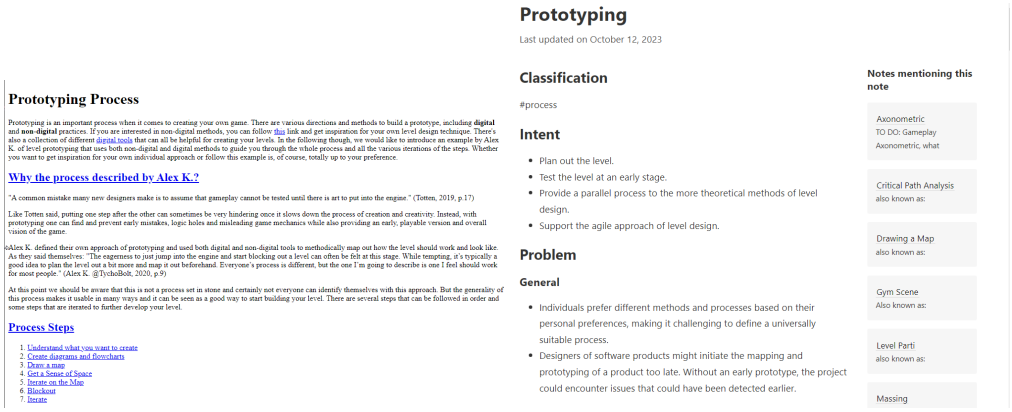


Figure 7.2 Comparison between a webpage published on "Github Pages" (left) and on "Netlify"(right), respectively.

7.3 Validity

An analysis of the thesis's validity remains ambiguous, as it lacks an attached survey or formal evaluation. The project was predominantly created and modified by one person, with the exception of periodic reviews by the supervisor. Except these reviews, only friends and family have been introduced to the project but not asked to participate in an evaluation. Without a proper evaluation or survey, there is no proof of the project's significance and justification alongside existing resources like the "Level Design Book" and the two mapping studies [21] [160] [58].

On the other hand, determining the precise questions to prove the project's significance is a complex process. Simple questions, such as whether users find the website or vault enjoyable, do not provide sufficiently critical evaluation material. Comparing the website with the Level Design Book can lead to a popularity bias and stray away from the research question, potentially setting up a competition between the two resources rather than encouraging their coexistence. Research on the most frequently accessed areas of the website might prove interesting, but it, too, might fall short of justifying the project's existence.

Validating the project remains an open quest for the future. Future steps involve considering the metrics and methods needed to evaluate the project's importance in the field of level design. Until then, this thesis maintains subjective validity, as it answers the initial research question proposed in the beginning of this master thesis.

7.4 Next steps

This particular project is designed to evolve over time, presenting an opportunity to include additional research and methods in the vault. Simultaneously, this work will contribute to my supervisor's vault, transforming it into a valuable teaching resource. This might prove its validity, as there will be students accessing the knowledge. I welcome thoughts and input from interested individuals stumbling upon my project, given it is backed up with adequate scientific research.

8 Conclusion

Concluding this thesis, it achieved most of the goals it tried to implement. At its core, it addressed the fundamental question: Can methods and tools for level design be summarized to a shared vocabulary and effectively collected on a shared online platform, highlighting the interconnection of its entries and the scalability of the knowledge pool? The urge for a solution to this question becomes adamant in the context of level design, a growing field with much research yet to be made, and with level designers lacking the time to properly study the research to enhance their game development process.

The thesis further looked into related work that proposed a solution for parts of the problem (chapter 2). These ranged from mapping game design languages to formalized domains to an established, popular website called the "Level Design Book", a widely recognized resource that depicts many of the important methods in a linear manner [21]. Rather than viewing these works as competitors to the project, they collectively contribute to different aspects of the research field, each complementing this thesis's work.

The next chapter (3) outlined the multifaceted definition of level design taken from multiple sources, illustrating the most important researches that influenced this thesis such as Totten's "An Architectural Approach to Level Design", Salen and Zimmermann's "Rules of Play" and Schell's "The Art of Game Design: A Book of Lenses" [153] [151] [137], as well as other fields defined by numerous papers and researches [20] [101]. Additionally, it provided essential insights into the technology behind level design at the end of the chapter, providing the reader with sufficient knowledge to understand the collected methods and tools on a deeper level.

The fourth chapter (4) marked the beginning of the project phase, describing the urge towards a shared platform that collects relevant resources and grant interested parties accessibility to the knowledge. Collaboration emerged as a central drive for the project, along with the organization of the knowledge within a digital garden. This chapter also marked the commencement of addressing the research question, aiming to forge a common understanding with unified vocabulary, collect important information from other research and collaborate on the established platform.

After abandoning the initial approach, the fifth chapter (5) explained the process of the project from its creation to its completion. It introduced the concept of a digital garden, a symbolism taken from early HTML approaches and running as a recurring motif throughout the thesis. Obsidian proved to be a fitting tool to create a folder structure, called "Vault", facilitating the growth of knowledge. Diverging from the linear format of the Level Design Book, the project rather adopted a nonlinear approach, simplifying the addition of new knowledge entries. In addition, the source graph became a valuable feature from Obsidian, illustrating the connections between each entry and further structuring the knowledge with colors, node sizes and applied forces. The resulting website was constructed with Netlify, a website provider that imports an established GitHub repository and visually illustrates its markdown files, facilitated universal accessibility, permitting collaboration via the shared repository in the future.

The sixth chapter (6) summarized the results of the project, establishing the current state-of-the-art of a vault or digital garden. Its complex knowledge graph features numerous entries including methods, tools, processes, roles, artifacts, and literature notes. The vault can be accessed over the website at all times, offering uninterrupted entry to anyone possessing the link.

The last chapter (7) revisited the research question, concluding that this thesis, for the most part, addressed the research question, though with some challenges and limitations that arose over the development of the digital garden. As such, its most notable challenge was the creation of a structure, given the absence of a unified, shared terminology within the literature and research field. Consequently, the project is designed to evolve over time along the growing amount of research in the field. It serves as a testament to the ever-evolving thoughts and ideas within the field, symbolized by the non-linearly connected nodes, each contributing to the design of a level. Some nodes have more impact than others; the digital garden is capable of showing the

relation and significance of each method with its source graph. The project is still considered non-final and it is uncertain whether its solution is significant enough to be empirically validated within the research field, as it lacks formal studies or evaluations to substantiate its worth. In the future, this specific project will be added to my supervisor's vault, integrating the achieved knowledge in the existent structure and enhancing the pool of already accumulated knowledge to be accessed by the students of TUM. A prospective evaluation involving students over several months may serve to prove the project's scientific validity.

8.1 Future Work

Regarding the field of level design, future research could further explore the possibility of collecting valuable knowledge from scientific research and sharing them with the community of level designers. For instance, it could delve into the automation of knowledge collection from scientific research, facilitating the accumulation of knowledge and integration into an existing structure. The mapping study "Languages of games and play: A systematic mapping study" (2020) already proposed a structure that organizes design languages, paving the way to transfer this thought over to level design methods and tools [160].

It can further be researched on the terminology itself. As underscored by various research endeavors, the urge for a shared vocabulary is strong [160] [58]. While this project has made an effort in unifying existing research and developing a shared terminology, it is but one approach among many. Future research could explore alternative terminologies and justifications for their usage, thereby enhancing the unified vocabulary and terminology.

This project is based on theoretical knowledge, although it tries to bridge scientific studies with its practical use in the level design field. Therefore, an investigation into the practicability of the knowledge might be interesting. The option of executing the collected knowledge in a game for users to explore and play with marks an interesting approach of deepening the understanding of the provided material, further supporting level designers to utilize the knowledge.

Exploring language diversity offers another promising dimension for future investigation. This project solely made use of the English language. Expanding the project's reach by translating its knowledge into other languages could unlock its potential for a global audience. However, it still has to be explored whether an automatic language translator can be applied to facilitate file creation and management.

Obsidian comes with many more features and this project uses only a fraction of it. Future work upon this foundation, enhancing the project or similar works with additional Obsidian features and creating a comprehensive knowledge network. This might also simplify the use of the knowledge network in various applications.

Moreover, the study of alternative software or applications capable of hosting markdown files holds potential for future research. Delving deeper into the concept of using markdown files as the foundation for knowledge networks can further enhance the research field, with possible alternatives to be more fitting to answer the research question.

In conclusion, finishing this thesis marks not an endpoint but rather a waypoint in the ever-expanding landscape of level design research. It serves as a stepping stone for future endeavors to cultivate and organize knowledge within the field, bridging the gap between research and practice, and fostering a shared understanding among level designers worldwide.

9 Acknowledgements

Throughout the course of this project, I can call myself fortunate to have the support and guidance of many individuals who played a vital role in helping me shape my work and stay motivated.

My first and most important thanks goes to Daniel Dyrda, who offered to supervise me even though I betrayed the ranks of the Games Engineers and transferred to Information Systems. He is busy all the time and constantly overworking, but he still sat down with me to develop a master thesis concept tailored to my strengths, met with me almost every week to provide invaluable support, and will soon have to read through roughly 80 pages until he can have a little laugh over these words. He is also the one who inspired me to invest in my second most important stress reliever in the last three months: Baldur's Gate 3. For that, too, I am very grateful.

I would like to express my gratitude to Prof. Dr. Gudrun Klinker, who has been an invaluable part of the TUM Games Engineering community. Her contributions to the field of AR are truly remarkable, and she has added a unique charm to the chair. I appreciate her for making me, as a woman in a predominantly male community, feel welcome.

Four of my friends patiently reviewed my thesis, offering suggestions and critique. I am immensely thankful to them, not only for proofreading, but also for supporting me throughout my whole university journey, too. On this note, I also want to thank my friend group, the Mangohunters, for making me laugh in the most inappropriate moments and create innovative and fun games with me when we still participated in the game jams.

I would also like to thank my dance community, who are not just friends and colleagues, but also family. Dance has become an integral part of my life and a primary stress reliever, providing a safe place in uncertain times.

Last but not least, I am deeply grateful to my family, who reach out to me even though I often forget to check up on them. They are the safety net I can always come back to, and for that, I am truly thankful.

Bibliography

[1] Artstation. https://www.artstation.com/. Accessed: 2023-10-09.

[2] Bluehost website. https://www.bluehost.com/. Accessed: 2023-10-10.

[3] The garden and the stream: A technopastoral. https://hapgood.us/2015/10/17/the-garden-and-the-stream-a-technopastoral/. Accessed: 2023-10-10.

[4] Chat gpt. https://chat.openai.com/. Accessed: 2023-10-08.

[5] Design pillars – the core of your game. https://www.gamedeveloper.com/design/design-pillars-the-core-of-your-game. Accessed: 2023-10-12.

[6] Diablo 3 official website. https://eu.diablo3.blizzard.com/de-de/. Accessed: 2023-10-07.

[7] A brief history & ethos of the digital garden. https://maggieappleton.com/garden-history. Accessed: 2023-10-07.

[8] Epic games announces unreal development kit, powered by unreal engine 3. https://www.ign.com/articles/2009/11/05/epic-games-announces-unreal-development-kit-powered-by-unreal-engine-3, . Accessed: 2023-10-09.

[9] Better with age: A history of epic games. https://www.polygon.com/2012/10/1/3438196/better-with-age-a-history-of-epic-games, . Accessed: 2023-10-09.

[10] Former bioware gm opens up about difficulties of frostbite engine. https://www.gamesindustry.biz/former-bioware-gm-opens-up-about-difficulties-of-frostbite-engine, . Accessed: 2023-10-09.

[11] Frostbite engine. https://www.ea.com/frostbite, . Accessed: 2023-10-09.

[12] Anthem development suffered from the usage of frostbite engine – report. https://gamingbolt.com/anthem-development-suffered-from-the-usage-of-frostbite-engine-report, . Accessed: 2023-10-09.

[13] Game design methods: A 2003 survey. https://www.gamedeveloper.com/design/game-design-methods-a-2003-survey. Accessed: 2023-10-07.

[14] A game design document (gdd) tutorial that will save you time and energy. https://dev.to/garrett/a-game-design-document-gdd-tutorial-that-will-save-you-time-and-energy-238n. Accessed: 2023-10-12.

[15] Github website. https://github.com/, . Accessed: 2023-10-10.

[16] Github pages vs netlify. https://blog.back4app.com/github-pages-vs-netlify/, . Accessed: 2023-10-11.

[17] How was hollow knight art made. https://osgamers.com/faq/how-was-hollow-knight-art-made. Accessed: 2023-10-08.

[18] Ionos website. https://www.ionos.de/. Accessed: 2023-10-10.

[19] Setting up your own digital garden with jekyll. https://maximevaillancourt.com/blog/setting-up-your-own-digital-garden-with-jekyll. Accessed: 2023-10-11.

[20] Level design - in pursuit of better levels. https://docs.google.com/document/d/1fAlf2MwEFTwePwzbP3try1H0aYa9kpVBHPBkyIq-caY/edit. Accessed: 2023-10-08.

[21] Level design book. https://book.leveldesignbook.com/. Accessed: 2023-10-07.

[22] Markdown guide. https://www.markdownguide.org/getting-started/. Accessed: 2023-10-09.

[23] The frostbite engine nearly tanked mass effect andromeda. https://www.vg247.com/the-frostbite-engine-nearly-tanked-mass-effect-andromeda. Accessed: 2023-10-09.

[24] Autodesk maya collaboration with disney. https://web.archive.org/web/20110720201920/http://www.microfilmmaker.com/reviews/Issue56/Maya11_1.html. Accessed: 2023-10-09.

[25] How to create a simple web page using markdown. https://medium.com/craftycode/how-to-create-a-simple-web-page-using-markdown-95e462e43e01. Accessed: 2023-10-10.

[26] The metrics of space: Molecule design. https://www.gamedeveloper.com/design/the-metrics-of-space-molecule-design. Accessed: 2023-10-08.

[27] Netlify hosting website. https://www.netlify.com/, . Accessed: 2023-10-11.

[28] My digital garden (netlify publication of my project). https://methods-and-tools-for-level-design.netlify.app/, . Accessed: 2023-10-11.

[29] Nuxt.js framework manual. https://nuxt.com/docs, . Accessed: 2023-10-10.

[30] Nuxt.js over vue.js: When should you use it and why. https://www.bornfight.com/blog/nuxt-js-over-vue-js-when-should-you-use-it-and-why/, . Accessed: 2023-10-10.

[31] nuxt-github-pages. https://github.com/lucpotage/nuxt-github-pages, . Accessed: 2023-10-10.

[32] Obsidian developer documentation. https://docs.obsidian.md/Home, . Accessed: 2023-10-09.

[33] Obsidian help website. https://help.obsidian.md/, . Accessed: 2023-10-11.

[34] Obsidian homepage. https://obsidian.md/, . Accessed: 2023-10-11.

[35] Single player level design pacing and gameplay beats. https://www.worldofleveldesign.com/categories/wold-members-tutorials/peteellis/level-design-pacing-gameplay-beats-part1.php. Accessed: 2023-10-07.

[36] Pinterest. https://www.pinterest.com. Accessed: 2023-10-11.

[37] Procreate software. https://procreate.com/. Accessed: 2023-10-09.

[38] Reddit. https://www.reddit.com/. Accessed: 2023-10-08.

[39] Reward schedules and when to use them. https://www.gamedeveloper.com/business/reward-schedules-and-when-to-use-them. Accessed: 2023-10-07.

[40] Rpg maker. https://www.rpgmakerweb.com/, . Accessed: 2023-10-09.

[41] Rpg maker: How a niche game maker created a vibrant community of developers. https://uk.pcmag.com/gaming-gear/128516/rpg-maker-how-a-niche-game-maker-created-a-vibrant-community-of-developers, . Accessed: 2023-10-09.

[42] Simplenote. https://simplenote.com/. Accessed: 2023-10-09.

[43] The 7 stages of game development. https://www.g2.com/articles/stages-of-game-development. Accessed: 2023-10-08.

[44] How the "stranger things" sound team creeps you out. https://www.motionpictures.org/2022/06/how-the-stranger-things-sound-team-creeps-you-out/. Accessed: 2023-10-08.

[45] unity. https://unity.com/. Accessed: 2023-10-09.

[46] Blinded by reality - the real story behind the creation of unreal. https://web.archive.org/web/20010519154729/http://www.gamespot.com/features/makeunreal/, . Accessed: 2023-10-09.

[47] Unreal engine. https://www.unrealengine.com/, . Accessed: 2023-10-09.

[48] Unreal engine 5. https://github.com/topics/unreal-engine-5, . Accessed: 2023-10-09.

[49] Vue framework manual. https://vuejs.org/guide/introduction.html, . Accessed: 2023-10-10.

[50] How to parse and render markdown in vuejs. https://blog.openreplay.com/how-to-parse-and-render-markdown-in-vuejs/, . Accessed: 2023-10-10.

[51] Youtube. https://www.youtube.com/. Accessed: 2023-10-08.

[52] Hangar 13. Mafia iii. [PlayStation 4, Xbox One, Microsoft Windows, macOS, Google Stadia, Mac OS], 2016.

[53] Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. In *IFIP international conference on artificial intelligence applications and innovations*, pages 373–383. Springer, 2020.

[54] Ernest Adams and Joris Dormans. *Game mechanics: advanced game design*. New Riders, 2012.

[55] Adobe Inc. Adobe photoshop. URL https://www.adobe.com/products/photoshop.html.

[56] Luke Ahearn. *3D game textures: create professional game art using photoshop*. CRC Press, 2014.

[57] Christopher Alexander. *A pattern language: towns, buildings, construction*. Oxford university press, 1977.

[58] Marcos Silvano Orita Almeida and Flavio Soares Correa da Silva. A systematic review of game design methods and tools. In *Entertainment Computing–ICEC 2013: 12th International Conference, ICEC 2013, São Paulo, Brazil, October 16-18, 2013. Proceedings 12*, pages 17–29. Springer, 2013.

[59] Autodesk, INC. Maya. URL https:/autodesk.com/maya.

[60] Andrew Begel, Jan Bosch, and Margaret-Anne Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *IEEE software*, 30(1):52–66, 2013.

[61] Erlend Berger, Torjus H Sæthre, and Monica Divitini. Privacity: A chatbot game to raise privacy awareness among teenagers. In *Informatics in Schools. New Ideas in School Informatics: 12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2019, Larnaca, Cyprus, November 18–20, 2019, Proceedings 12*, pages 293–304. Springer, 2019.

[62] Mark Bernstein. Hypertext gardens: Delightful vistas. *Eastgate Systems Web site, accessed at www. eastgate. com/garden, February*, 1998.

[63] BioWare. Dragon age: Inquisition. [PlayStation 4, Xbox One, PlayStation 3, Xbox 360, xCloud, Microsoft Windows], 2014.

[64] BioWare. Mass effect: Andromeda. [PlayStation 4, Xbox One, xCloud, Microsoft Windows], 2017.

[65] Brenda Brathwaite and Ian Schreiber. *Challenges for game designers*. Course Technology/Cengage Learning Boston, Massachusetts, 2009.

[66] Margo Buchanan-Oliver and Yuri Seo. Play as co-created narrative in computer game consumption: The hero's journey in warcraft iii. *Journal of Consumer Behaviour*, 11(6):423–431, 2012.

[67] P Burkart. Discovering a lexicon for video games: New research on structured vocabu-laries. *International Digital Media and Arts Association Journal*, 2(1):18–24, 2005.

[68] Edward Byrne. *Game level design*, volume 6. Charles River Media Boston, 2005.

[69] Joseph Campbell. *The hero with a thousand faces*, volume 17. New World Library, 2008.

[70] Capcom. Megaman. [NES, Xbox, PlayStation], 1993.

[71] Alex J Champandard. Understanding the second-generation of behavior trees. *Tillgänglig på internet: http://aigamedev. com/insider/tutorial/second-generation-bt.[Hämtad: 14.04. 06]*, 2012.

[72] Team Cherry. Hollow knight. [Microsoft Windows], 2017.

[73] Neil Cohn. Visual narrative structure. *Cognitive science*, 37(3):413–452, 2013.

[74] Karen Collins. *Game sound: an introduction to the history, theory, and practice of video game music and sound design*. Mit Press, 2008.

[75] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL `http://www.blender.org`.

[76] David Conroy, Peta Wyeth, and Daniel Johnson. Modeling player-like behavior for game ai design. In *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, pages 1–8, 2011.

[77] Greg Costikyan. I have no words and i must design. interactive fantasy# 2. *British roleplaying journal*, 1994.

[78] DICE. Battlefield: Bad company. [PlayStation 3, Xbox 360], 2008.

[79] Naughty Dog. Uncharted 2. [PlayStation 3, PlayStation 4], 2009.

[80] Quantic Dream. Detroit: Become human. [Playstation 4, Microsoft Windows], 2018.

[81] Crystal Dynamics. Tomb raider. [PlayStation 4, Xbox One, Microsoft Windows, ...], 2013.

[82] e4 Software. Swarm! [Mobile Game], 2013.

[83] Nada Endrissat, Gazi Islam, and Claus Noppeney. Visual organizing: Balancing coordination and creative freedom via mood boards. *Journal of Business Research*, 69(7):2353–2362, 2016.

[84] Square Enix. Final fantasy xv. [PlayStation 4, Microsoft Windows, Xbox One, Android, Google Stadia], 2016.

[85] Blizzard Entertainment. Diablo 3. [Windows, macOS, PlayStation 3, PlayStation 4, Xbox 360, Xbox One, Nintendo Switch], 2012.

[86] John Feil and Marc Scattergood. *Beginning game level design*. Thomson Course Technology, 2005.

[87] Syd Field. *Screenplay: The foundations of screenwriting*. Delta, 2005.

[88] Eli France. Can chat-gpt rap: Analyzing the ability of chat-gpt to replicate and analyze the lyrical style of young thug. 2023.

[89] Thomas M Froese. The impact of emerging information technology on project management for construction. *Automation in construction*, 19(5):531–538, 2010.

[90] Tracy Fullerton, Chris Swain, and Steven Hoffman. *Game design workshop: Designing, prototyping, & playtesting games*. CRC Press, 2004.

[91] Epic Games. Unreal tournament 3. [Microsoft Windows, PlayStation 3, Xbox 360], 2007.

[92] Epic Games. Fortnite. [GeForce Now, xCloud, Xbox One, PlayStation 5], 2017.

[93] Guerilla Games. Horizon zero dawn. [PlayStation 4, Microsoft Windows], 2017.

[94] Guerilla Games. Horizon forbidden west. [PlayStation 5, PlayStation 4, Microsoft Windows], 2022.

[95] Riot Games. League of legends. [GeForce Now, macOS, Microsoft Windows, PlayStation 4, Xbox One, Mac OS], 2009.

[96] ghosthunter. Grimm's hollow. [Microsoft Windows], 2019.

[97] Anthony Giddens and Simon Griffiths. *Sociology*. Polity, 2006.

[98] Ross Graham, Hugh McCabe, and Stephen Sheridan. Pathfinding in computer games. *The ITB Journal*, 4(2):6, 2003.

[99] Isabela Granic, Adam Lobel, and Rutger CME Engels. The benefits of playing video games. *American psychologist*, 69(1):66, 2014.

[100] Jason Gregory. *Game engine architecture*. crc Press, 2018.

[101] Celia Hodent. *The psychology of video games*. Routledge, 2020.

[102] Jussi Holopainen and Staffan Björk. Game design patterns. *Lecture Notes for GDC*, 2003.

[103] Atari Inc. Pong. [Arcade Game], 1972.

[104] Barry Ip. Narrative structures in computer and video games: Part 1: Context, definitions, and initial findings. *Games and Culture*, 6(2):103–134, 2011.

[105] Henry Jenkins. Game design as narrative architecture. *Computer*, 44(3):118–130, 2004.

[106] Henry Jenkins and Mark Deuze. Convergence culture, 2008.

[107] JoeOsborn. playspec-js. `https://github.com/JoeOsborn/playspecs-js`, 2018.

[108] Christopher M Kanode and Hisham M Haddad. Software engineering challenges in game development. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 260–265. IEEE, 2009.

[109] Konami. Metal gear solid. [Nintendo Switch, PlayStation, PlayStation 3, Microsoft Windows], 1998.

[110] Rudolf Kremers. *Level design: concept, theory, and practice*. CRC Press, 2009.

[111] Christine Legner, Torsten Eymann, Thomas Hess, Christian Matt, Tilo Böhmann, Paul Drews, Alexander Mädche, Nils Urbach, and Frederik Ahlemann. Digitalization: opportunity and challenge for the business and information systems engineering community. *Business & information systems engineering*, 59:301–308, 2017.

[112] LethaMasahiro. digital-garden-level-design. `https://github.com/LethaMasahiro/digital-garden-level-design`, 2023.

[113] Sus Lundgren and Staffan Bjork. Game mechanics: Describing computer-augmented games in terms of interaction. In *Proceedings of TIDSE*, volume 3. Citeseer, 2003.

[114] Ross Duffer Matt Duffer. Stranger things. [Netflix], 2016.

[115] Josh McCoy, Mike Treanor, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. Prom week: social physics as gameplay. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 319–321, 2011.

[116] Rollin McCraty, Bob Barrios-Choplin, Michael Atkinson, and Dana Tomasino. The effects of different types of music on mood, tension, and mental clarity. *Alternative therapies in health and medicine*, 4 (1):75–84, 1998.

[117] Hilary McLellan. Digital storytelling in higher education. *Journal of Computing in Higher Education*, 19:65–79, 2007.

[118] Marjan Mernik, Jan Heering, and Anthony M Sloane. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344, 2005.

[119] Briar Lee Mitchell. *Game design essentials*. John Wiley & Sons, 2012.

[120] Alberto Mora, Daniel Riera, Carina González, and Joan Arnedo-Moreno. Gamification: a systematic review of design frameworks. *Journal of Computing in Higher Education*, 29:516–548, 2017.

[121] Katharine Neil. Game design tools: Time to evaluate. *Proceedings of 2012 DiGRA Nordic*, 2012.

[122] Niels A Nijdam. Mapping emotion to color. *Book Mapping emotion to color*, pages 2–9, 2009.

[123] Nils J Nilsson. *Artificial intelligence: a new synthesis*. Morgan Kaufmann, 1998.

[124] Nintendo. Donkey kong. [Atari, Nintendo], 1981.

[125] Nintendo. Metroid 2 - return of samus. [Nintendo Switch, Game Boy, Nintendo 3DS], 1991.

[126] Nintendo. Animal crossings: New horizons. [Nintendo Switch], 2020.

[127] Francisco R Ortega, Frank Hernandez, Armando Barreto, Naphtali D Rishe, Malek Adjouadi, and Su Liu. Exploring modeling language for multi-touch systems using petri nets. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*, pages 361–364, 2013.

[128] Roger E Pedersen. *Game design foundations*. Wordware Publishing, Inc., 2003.

[129] David Pizzi, Jean-Luc Lugrin, Alex Whittaker, and Marc Cavazza. Automatic generation of game level solutions as storyboards. *IEEE Transactions on Computational Intelligence and AI in Games*, 2 (3):149–161, 2010.

[130] Steven Rabin. *Game AI pro: collected wisdom of game AI professionals*. CRC Press, 2013.

[131] Steven Rabin. *Game AI pro 2: collected wisdom of game AI professionals*. CRC Press, 2015.

[132] Emanuel Montero Reyno and José Á Carsí Cubel. A platform-independent model for videogame gameplay specification. In *DiGRA Conference*. Citeseer, 2009.

[133] Johannes Willem Romein. Multigame-an environment for distributed game-tree search. 2001.

[134] Richard Rouse. Game design. In *The Routledge Companion to Video Game Studies*, pages 83–90. Routledge, 2014.

[135] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[136] Marie-Laure Ryan. Beyond myth and metaphor: Narrative in digital media. *Poetics Today*, 23(4): 581–609, 2002.

[137] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC press, 2008.

[138] Karen Schrier. *Learning, education and games. Volume one: Curricular and design considerations*. Carnegie Mellon University, 2014.

[139] Noor Shaker, Julian Togelius, and Mark J Nelson. Procedural content generation in games. 2016.

[140] Ruben Smelik, Tim Tutenel, Klaas Jan De Kraker, and Rafael Bidarra. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–8, 2010.

[141] Antonín Šmíd. Comparison of unity and unreal engine. *Czech Technical University in Prague*, pages 41–61, 2017.

[142] B Smith, John and F Weiss, Stephen. Hypertext. *Communications of the ACM*, 31(7):816–819, 1988.

[143] Gearbox Software. Borderlands. [Microsoft Windows, PlayStation 3, Xbox 360], 2009.

[144] Olli Sotamaa. Perceptions of player in game design literature. In *DiGRA Conference*, 2007.

[145] Peter Samson Steve Russell, Martin Graetz. Spacewar! [Analog Computer], 1962.

[146] SCE Santa Monica Studio. God of war. [PlayStation 5, Mac OS, Microsoft Windows], 2018.

[147] Larian Studios. Baldur's gate 3. [Digital and Physical Copies], 2023.

[148] Rocksteady Studios. Batman: Arkham asylum. [PlayStation 4, PlayStation 3, Nintendo Switch, Xbox One, Xbox 360, Microsoft Windows, macOS, Ouya, Mac OS], 2009.

[149] Penelope Sweetser and Peta Wyeth. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3):3–3, 2005.

[150] Uranium Team. Pokemon uranium. [Microsoft Windows], 2016.

[151] Katie Salen Tekinbas and Eric Zimmerman. *Rules of play: Game design fundamentals*. MIT press, 2003.

[152] Ulyana Tikhonova, MW Manders, MGJ van den Brand, Suzana Andova, and Tom Verhoeff. Applying model transformation and event-b for specifying an industrial dsl. In *10th International Workshop on Model Driven Engineering, Verification and Validation*, pages 41–50. CEUR-WS. org, 2013.

[153] Christopher W Totten. *Architectural Approach to Level Design*. CRC Press, 2019.

[154] Mike Treanor, Bryan Blackford, Michael Mateas, and Ian Bogost. Game-o-matic: Generating videogames that represent ideas. In *Proceedings of the The third workshop on Procedural Content Generation in Games*, pages 1–8, 2012.

[155] Valve. Portal. [Microsoft Windows; Xbox 360; PlayStation 3; macOS; Linux; Android; Nintendo Switch], 2007.

[156] Jeroen van den Bos and Tijs van der Storm. Bringing domain-specific languages to digital forensics. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 671–680, 2011.

[157] Arie Van Deursen, Paul Klint, and Joost Visser. Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000.

[158] Peter Van Rosmalen, Johan Eikelboom, Erik Bloemers, Kees Van Winzum, and Pieter Spronck. Towards a game-chatbot: Extending the interaction in serious games. In *The 6th European Conference on Games Based Learning*, 2012.

[159] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[160] Riemer Van Rozen. Languages of games and play: A systematic mapping study. *ACM Computing Surveys (CSUR)*, 53(6):1–37, 2020.

[161] John B Watson. *Behaviorism*. Routledge, 2017.

[162] Wim Westera, Rob J Nadolski, Hans GK Hummel, and Iwan GJH Wopereis. Serious games for higher education: a framework for reducing design complexity. *Journal of Computer Assisted Learning*, 24 (5):420–432, 2008.

[163] Robert Dvorak William Higinbotham. Tennis for two. [Analog Computer], 1958.

[164] J Patrick Williams, David Kirschner, Nicholas Mizer, and Sebastian Deterding. Sociology and role-playing games. *Role-playing game studies: A transmedia approach*, pages 227–244, 2018.

[165] Geogios N Yannakakis. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292, 2012.

[166] Georgios N Yannakakis and John Hallam. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence*, 21(10):933–971, 2007.

[167] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*, volume 2. Springer, 2018.

[168] Min Zhang and Juntao Li. A commentary of gpt-3 in mit technology review 2021. *Fundamental Research*, 1(6):831–833, 2021.

List of Figures