UNIVERSITÄT
BAYREUTH

Fachgruppe Medienwissenschaft

Computer Game Sciences

Master's Thesis

Quest Construction in REDengine 3

-

A view into the conditions of

The Witcher 3: Wild Hunt quest modding.

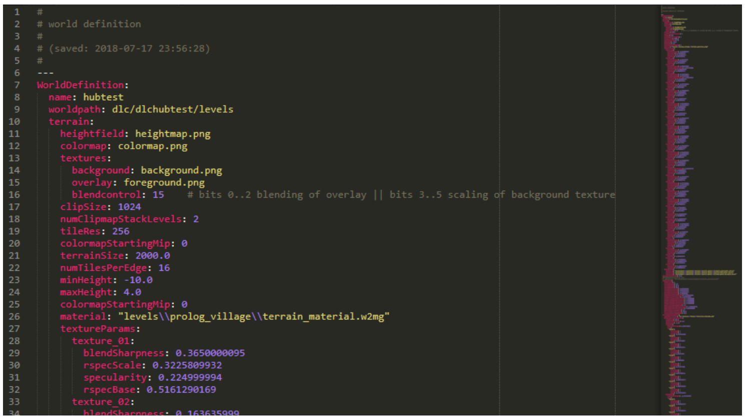
Gregor Hans Christian Sönnichsen

March 24, 2022

Supervisors:

Prof. Dr. Jochen Koubek

Robin Hädicke M.A. Dipl.-Kult.



```
#
# world definition
#
# (saved: 2018-07-17 23:56:28)
#
---
WorldDefinition:
  name: hubtest
  worldpath: dlc/dlchubtest/levels
  terrain:
    heightfield: heightmap.png
    colormap: colormap.png
    textures:
      background: background.png
      overlay: foreground.png
      blendcontrol: 15    # bits 0..2 blending of overlay || bits 3..5 scaling of background texture
    clipSize: 1024
    numClipmapStackLevels: 2
    tileRes: 256
    colormapStartingMip: 0
    terrainSize: 2000.0
    numTilesPerEdge: 16
    minHeight: -10.0
    maxHeight: 4.0
    colormapStartingMip: 0
    material: "levels\\prolog_village\\terrain_material.w2mg"
    textureParams:
      texture_01:
        blendSharpness: 0.3650000095
        rspecScale: 0.3225809932
        specularity: 0.224999994
        rspecBase: 0.5161290169
      texture_02:
        blendSharpness: 0.163635999
```

»The `trial of the radishes` is meant as a guided, self-learning tutorial *without* step-by-step instructions. Instead it focuses on exploratory learning by actively using the tools to solve increasingly challenging tasks.«[1]

»‘trials of the radishes - reportedly only three in ten survived the trials’ 😄«[2]

Note: in order to not assume any gender, I will use “them”/ “them” pronouns for all persons mentioned in this work.

[1] rmemr. (2020). *Trial of the radishes - Trial 1 - Installation at The Witcher 3 Nexus - Mods and community*. Online: https://www.nexusmods.com/witcher3/articles/113 (accessed: 16.03.2023).

[2] rmemr. (2022). [Message]. *w3 radish tools. Discord*. Online: https://discord.com/channels/416336392692695040/417414398987206676/985503608676167740 (accessed: 16.03.2023).

Abstract

In the quest definitions contained in the game files of *The Witcher 3: Wild Hunt* (TW3) a huge amount of knowledge is embedded: Not only varying technical approaches to formalisation of the concept “quest”, but also views concerning how parts of real and fictional worlds are understood and adopted as technical units for design, forming an “engine language”. Analysing and understanding this language as a condition for design is the topic of this thesis.

The theoretical frame is set with conceptions of design, tools and their so-called object languages in design studies, which are integrated with theory about game engines. *REDengine 3*’s[3] language for quest construction is accessible only in a specific context: in the TW3 modding scene, which is dealing not only with the engine’s givens, but also with an ambiguous strategy by *CD Projekt RED*, visible in incomplete tools and cautious support. Analysing the engine and its tools proves them to be engrained with a focus on narratives and specialised sub-languages, and shows that a great deal of designers’ conditions is determined by usability issues.

[3] The engine built for TW3.

Zusammenfassung

In den Quest-Definitionen, die in den Spieldateien von *The Witcher 3: Wild Hunt* (TW3) enthalten sind, liegt eine große Menge Wissen. Dazu gehören nicht nur verschiedene technische Herangehensweisen an die Formalisierung des Konzepts “Quest”, sondern auch Weltansichten bezüglich der Art, wie Teile realer und fiktionaler Welten verstanden und als technische Einheiten für die Gestaltung adaptiert wurden. Die Analyse und das Greifen dieser Sprache als Bedingung für Design sind das Thema dieser Thesis.

Der theoretische Rahmen wird gesetzt mit aus der Designwissenschaft stammenden Konzeptionen von Design-Arbeit, Werkzeugen und deren sogenannten Objekt-Sprachen. Diese Konzepte werden auch mit Theorie über Spiele-Engines integriert. Die Sprache der *REDengine 3*[4] für Quest-Konstruktion ist nur in einem spezifischen Kontext erreichbar: In der Modding-Szene um TW3, die neben den Engine-Gegebenheiten mit der uneindeutigen Strategie *CD Projekt REDs* zurechtkommen muss – sichtbar in unvollständigen Entwicklerwerkzeugen und zurückhaltender Unterstützung vonseiten des Studios. Eine Analyse der Engine und zugehöriger Werkzeuge zeigt, dass in diese ein Fokus auf Narrative sowie spezialisierte Sub-Sprachen eingeschrieben ist. Es zeigt sich zudem, dass ein großer Teil der Design-Konditionen von Fragen der Benutzbarkeit bestimmt wird.

[4] Die Engine, mit der TW3 gemacht wurde.

Table of Contents

Chapter 1. Introduction

The design of quests, missions or stories has become an area of great interest in current game development culture and game studies[5].

I am interested in scholarly analysis of video game technology for narrative content, sicne I am convinced that by performing such intersections of game studies and -industry much knowledge can be gained. The goal of this thesis is to do an investigation of this kind: I will examine how quest construction can be done by modding REDengine 3, using production-historical information and results from narrative game studies to analyse the conditions a designer will face. This master's thesis then poses the following central research question:

> *With what conditions can modders construct quests in REDengine 3 and what are the conditions' consequences?*

The main motivation and estimated contribution of this thesis is to shed light on how designers always work under the influence of technical and cultural conditions, and what very specific consequences, helpful and hindering, this can have. This will show not only whenever I am able to supplement and contextualise engine findings with existing (game) design literature, but also in traces to other narrative forms and in the discussion of modding scenes as a cultural frame. It is from the combination of meanings assigned through technical realities and context that REDengine 3's "world-view" – regarding what quests are, what is associated with them, and how they are easily constructed – as perceived by an observer will become apparent.

On a meta-scientific layer, this thesis forms a crossing point for game studies, design studies and game development. This gives it value as an interdisciplinary project, but also as a project in the respective disciplines.

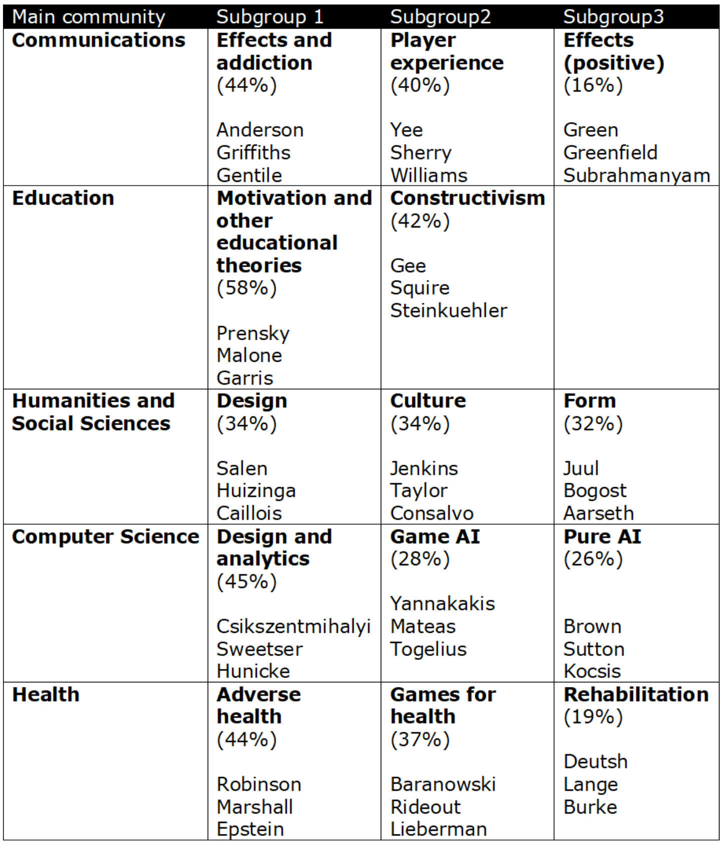
1.1 Methods and Motivations

In the next paragraphs I will present motivations and methods for tackling this thesis' research question, following the outline.

[5] Engström, Henrik. (2019). *GDC vs. DiGRA: Gaps in Game Production Research*. p. 11. "[..] narratives in games have received a massive interest from the research community. The industry has also produced many titles the last decades that have received a lot of attention for their narratives."

1.1.1 Game Studies

First and foremost, I understand this work as situated in the field of game studies, which is based mainly on the fact that this work, indeed, studies a specific game and aspects of video game storytelling in general. This classification is implemented by references to and inclusion of work in game studies and adjacent theory, such as *The Unity Game Engine and the Circuits of Cultural Software* by Nicoll and Keogh or *Quests. Design, Theory, and History in Games and Narratives* by Jeff Howard. Based on Paul Martin's analysis[6], my area of research falls into the **Humanities/ Social Sciences** community – wherein design, culture and form are three major topics (see Table 1). All three are touched upon in this thesis, with design being an important theoretical pillar, modding culture a context, and the form of REDengine 3 and its tools the subject of analysis.



Main community	Subgroup 1	Subgroup2	Subgroup3
Communications	**Effects and addiction** (44%) Anderson Griffiths Gentile	**Player experience** (40%) Yee Sherry Williams	**Effects (positive)** (16%) Green Greenfield Subrahmanyam
Education	**Motivation and other educational theories** (58%) Prensky Malone Garris	**Constructivism** (42%) Gee Squire Steinkuehler	
Humanities and Social Sciences	**Design** (34%) Salen Huizinga Caillois	**Culture** (34%) Jenkins Taylor Consalvo	**Form** (32%) Juul Bogost Aarseth
Computer Science	**Design and analytics** (45%) Csikszentmihalyi Sweetser Hunicke	**Game AI** (28%) Yannakakis Mateas Togelius	**Pure AI** (26%) Brown Sutton Kocsis
Health	**Adverse health** (44%) Robinson Marshall Epstein	**Games for health** (37%) Baranowski Rideout Lieberman	**Rehabilitation** (19%) Deutsh Lange Burke

Table 1: Communities in games research, divided into subgroups based on citations[7].

1.1.2 Adjacent Disciplines

Moving towards the connection to more specific disciplines, other topics and adjacent fields can be easily identified: by integrating *design studies* (Chapter 2) as

[6] Martin, Paul. (2018). *The Intellectual Structure of Game Research.* In: *Game Studies*. vol. 18. no. 1

[7] ibid.

a theoretical foundation for this work, this work potentially opens up a broader perspective on issues revolving around modern human production of cultural artefacts. While it is not the scope of this thesis to approach this branch in the surrounding field of anthropology, it is certainly its intention to point to this bigger picture. Major references in this discipline will be the works of Claudia Mareis, Ken Friedman, Herbert A. Simon and Christopher Alexander. To successfully perform the transition from the relatively abstract design studies to REDengine 3, the book *The Unity Game Engine and the Circuits of Cultural Software*[8] by Benjamin Nicoll and Brendon Keogh will aid, delivering valuable thoughts on how the engine as a non-neutral tool influences design and thus people. In the general structure of this thesis, discussions in design studies will help by allowing for instance narrative in games to relate to some general concepts like what design and design languages are. On the other hand, the concept of language in design studies will function as a conceptual lens to understand the nature of quest building in REDengine 3, and thus form an overall framework for elevating technological details into a scientific discourse. At last, to the authors knowledge there have not been any major couplings between game- and design studies as scientific disciplines, yet doing so provides natural perspectives and bridges to think about games and their production.

The production angle is where this thesis will touch topics of “cultural-historical configurations”[9], contextualising the creative work itself (Chapter 3). Since REDengine 3 is a proprietary tool by CD Projekt RED that was never intended to be particularly accessible from the outside, this thesis uses modding as a major point of entry. The thesis is thereby positioned close to analyses like Tanja Sihvonen’s view[10] into conditions of *The Sims* and its culture of modders utilising the potential of said conditions (or their absence). However, naturally, designing with the proper internal tools is radically different than with just a command line and the tools created by modders. This situation effectuates a whole new set of conditions for anyone who wants to work with the engine – and is thus studied here as well. I will go so far into the culture of modding and CD Projekt RED as is necessary to understand the frame in which a REDengine 3 user will enter when entering via modding. Another part of the cultural context is the history and nature of narratives (in video games). While this thesis does not feature an extensive analysis of how storytelling traditions

[8] Keogh, Brendan; Nicoll, Benjamin. (2019). *The Unity Game Engine and the Circuits of Cultural Software.* Springer International Publishing.

[9] Ruppert, Wolfgang. (1993). *Zur Geschichte der industriellen Massenkultur. Überlegungen zur Begründung eines Forschungsansatzes*. In: Ruppert, Wolfgang. (1993). *Chiffren des Alltags. Erkundungen zur Geschichte der industriellen Massenkultur*. Jonas. pp. 9-22. p. 10

[10] Sihvonen, Tanja. (2011). *Players Unleashed! Modding The Sims and the Culture of Gaming*. Amsterdam University Press. cf. p. 23

map into the engine, it will draw some major traces from e.g. Jeff Howard's work on quest design[11] to quests in REDengine 3.

Chapter 4 will be concerned with an in-depth analysis of selected conditions resulting from the technical givens of REDengine 3 and its tools. As a category for the chapter's contents, one could use the label "video game form", but it also and more specifically fits into the realm of *code and platform studies*, as they were declared by Ian Bogost and Nick Montfort[12]. In their explanation of where platform studies are situated, Bogost and Montfort[13] speak of different *levels* on which "new media" can be analysed:

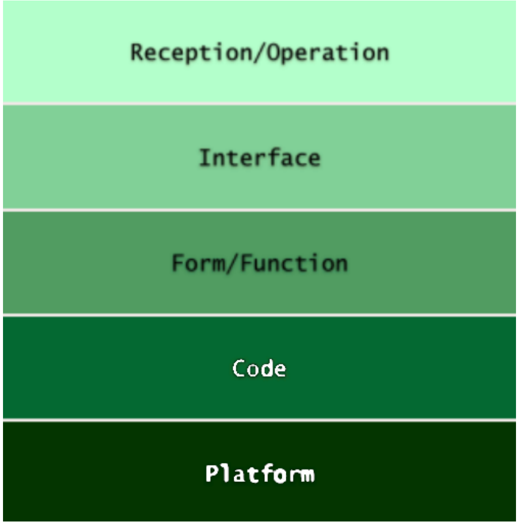


Illustration 1: Levels of analysis for "new media" according to Bogost and Montford (modified for better readability)[14].

Sorting Chapter 4's contents in this hierarchy, the following results:

Reception/ Operation is concerned with the player's experience and can thus be ruled out. **Interface** refers to the boundary between player and game and will be of concern only for illustrative purposes, since many features of the engine translate well to the game's user interface. We can observe the **Form/ Function** lens at work, when speaking about the meaning or consequences of REDengine 3's design capabilities: Form/ Function is "the main concern of cybertext studies", say Bogost and Montfort, implying that aspects like *ergodicity*[15] are discussed on this level,

[11] Howard, Jeff. (2008). *Quests: Design, Theory, and History in Games and Narratives*. 2nd ed. A K Peters/ CRC Press.

[12] Bogost, Ian; Montfort, Nick. (2022). *Levels*. In: *Platform Studies, a book series published by MIT Press, Ian Bogost and Nick Montfort, series editors*. Online: http://www.platformstudies.com/levels.html (accessed: 16.03.2023).

[13] ibid.

[14] ibid.

[15] Aarseth, Espen. (1997). *Cybertext—Perspectives on Ergodic Literature.* JHU Press. pp. 1-2. "In ergodic literature, non-trivial effort is required to allow the reader to traverse the text."

which I will do as well. Regarding the line between code and platform, the categorisation becomes blurry when applied to this project: in another place, Bogost/ Montfort define per citation of Marc Andreessen: “If you can program it, then it’s a platform. If you can’t, then it’s not.”[16]. On the first glance, this would put this project into platform studies, because REDengine 3 is indeed programmable, as it has an internal scripting language. However, I set out to not only inspect what the engine delivers without any game created on top, but *with* a game. So I also analyse the gameplay code and the files in which quest contents are defined, which puts the project into both platform and code studies, concludingly. Studying these lower levels not only caters to the nuisance that the “foundation of new media is still relatively neglected”[17], but also – in exposing technical realities – completes the picture of conditions for quest construction in REDengine 3.

1.1.3 The Witcher 3 Quests

One question might remain: why TW3 and why quests? First of all, doing this work will yield a precise insight into the way a modern, successful, hugely selling narrative-driven game is built.

The specific game that falls into the category of narrative-driven RPGs and which I am going to inspect is *The Witcher 3: Wild Hunt* (2015)[18] with its extensions *Hearts of Stone* (2015), *Blood and Wine* (2016) and every patch up till January 2023, all developed and published by the Polish video game developer CD Projekt RED.

TW3 is a game which thematically is about the relations between father-figures and their (adopted) children and about the consequences of war. Avatar Geralt is a *Witcher*, a professional monster-slayer who struggles with the ethical balance between his proclaimed neutrality and the injustices they find on their path and searches for Ciri, their adoptive daughter. The game is set in the world of Andrzej Sapkowski’s Witcher-series of books and features an extensive 3D game world, explorable in 3rd person on foot and riding. Besides exploration, the game’s two other major game modes are *fights* and *scenes*. Players can encounter a huge variety of monsters, humans and/ or beasts to combat in real-time, and they have a wide array of different swords, armors, potions, skills and more at hand to optimise their chances. The other major game mode are the scenes: some NPCs can be talked to or approached, and then the game switches into per-scene defined cameras and animations with written and spoken text and choices playing on top.

[16] Bogost, Ian; Montfort, Nick. (2009). *Platform Studies: Frequently Questioned Answers*. p. 4

[17] Bogost, Ian; Montfort, Nick. (2022). *Levels*.

[18] CD Projekt RED. (2015). T*he Witcher 3: Wild Hunt*. (4.01).

The game's quests are collections of events such as fights or scenes and goals which are mostly driven by objectives displayed on the screen.

Given that the company had a big, self-funded budget (the game ended up costing $67–81 million[19]), 150-250[20] devs at their proposal and a lot of quests planned (405[21] ended up in the game), the developers as in every such game needed to figure out how to produce so many quests of consistent quality in the given time frame. A common design language is thus inevitable, which motivates the analysis once more from a production studies point of view.

As I will argue, we are able to perceive well-developed languages for various narrative purposes such as quests and scenes in REDengine 3, if such a language is understood as a high-level set of recurring elements for building quests. If we factor in that the analysis will build upon technological details and that TW3's success was to a large part due to its (side) quest design[22][23], concludingly this thesis can provide precise technical reasons for the game's success, which is interesting for (technical) game designers or game scholars, who investigate video game historiography. There is, lastly, a lot of material to be found in the modding context for TW3 quests, as one of the major modding tool sets, the so-called *radish modding tools*[24]*,* explicitly aims at creating new quests for the game.

[19] Wikipedia contributors. (2023). *The Witcher 3: Wild Hunt - Wikipedia*. Online: https://en.wikipedia.org/wiki/The_Witcher_3:_Wild_Hunt#Development (accessed: 16.03.2023).

[20] ibid.

[21] Hopkins, Tom. (2018). *Witcher 3: How Long it Is and How Many Quests There Are.* Twinfinite. Online: https://twinfinite.net/2018/01/witcher-3-how-long-how-many-quests (accessed: 16.03.2023).

[22] Ingenito, Vince. (2015). *The Witcher 3 Review - IGN*. Online: https://www.ign.com/articles/2015/05/12/the-witcher-3-the-wild-hunt-review (accessed: 16.03.2023).

[23] Van Ord, Kevin. (2015). *The Witcher 3: Wild Hunt Review - GameSpot*. Online: https://www.gamespot.com/reviews/the-witcher-3-wild-hunt-review/1900-6416135/ (accessed: 16.03.2023).

[24] In this thesis I use the following version, which is only available on the *w3 radish tools* discord server: rmemr. (2022). *radish modding tools* (preview-v2022-12-26).

Chapter 2. Game Engines as Tools

This chapter will explain the first basic realm of concepts, which is the idea of engines as tools that support and influence execution and design. With these concepts we will get a better grasp of the layer on which (quest) creators work and the bridge that exists between engine and theory.

2.1 To Design for Execution

First of all, I will outline a meaning of the term “design”, which will serve as a basis for further discussions.

In their foundational and summarising work on theories of design, Claudia Mareis names two distinct layers of meaning: on the one hand, design refers to the “preparing, aesthetically formative”[25] process, on the other hand to the “resulting, as a rule intentionally conceived and created, artefacts and services”[26]. This thesis will focus primarily on the first layer – design as an activity – while the latter will appear only as the resulting “work”, “game”, “quest” and so forth from which conditions for design/ execution can be derived. I will now take a closer look at a selection of definition attempts for the term "design".

The Cambridge Dictionary posits that *to design* means “to make or draw plans for something”[27]. If, moreover, *plans* are “sets of decisions about how to do something in the future”[28], then it can be deduced that designing is all about decision-making regarding a potentially not-yet-existent something. This fits with the second understanding I will reproduce here. Political scientist Herbert A. Simon, who did a lot of interdisciplinary work and became also known for their design research[29], describes designers as those who devise “courses of action aimed at changing existing situations into preferred ones”[30].

Both the dictionary’s and Simon’s wordings contain the idea of a designer’s main task being to conceptualise the process for execution in order to achieve a set goal.

[25] Mareis, Claudia. (2014). *Theorien des Designs zur Einführung*. 1st ed. Junius Verlag. p. 38. Translations by the author.

[26] ibid.

[27] Cambridge University Press. (2022). *DESIGN | meaning, definition in Cambridge English Dictionary.* Online: https://dictionary.cambridge.org/dictionary/english/design (accessed: 16.03.2023).

[28] Cambridge University Press. (2022). *PLAN | meaning, definition in Cambridge English Dictionary.* Online: https://dictionary.cambridge.org/dictionary/english/plan (accessed: 16.03.2023).

[29] Mareis, Claudia. *Theorien des Designs zur Einführung*. p. 14

[30] A. Simon, Herbert. (1996). *The sciences of the artificial*. 3rd ed. MIT Press. p. 123

Don Norman however, a designer who is more interested in user-related aspects of design, writes that “design is concerned with how things work. How they are controlled, and the nature of the interaction between people and technology” [31].
While the dictionary and Simon follow the “design as an activity” semantic layer, Norman focuses on the results of the design process: it is important for their understanding that a designed product has certain qualities like the so-called “affordances”, “signifiers”, “feedback” etc.[32], all properties that in general improve usability. Later in their book they introduce an entire agenda geared towards “human-centred design”. In that regard Jesse Schell, author of the well-known *The Art of Game Design*, might agree with them.

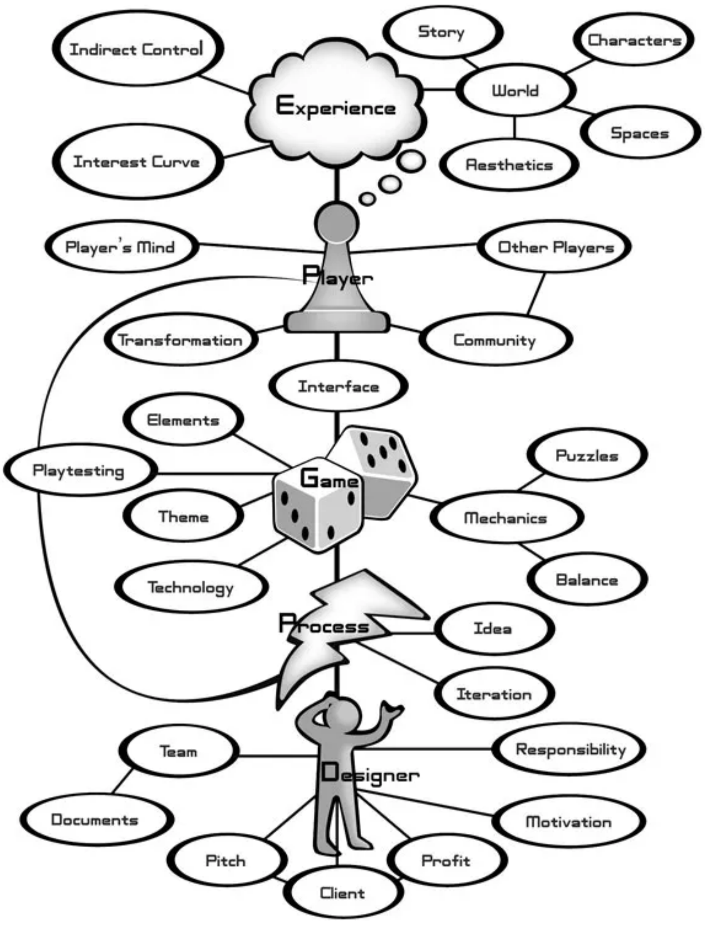


Illustration 2: Jesse Schell’s scheme[33] to illustrate the network of game design concepts presented in The Art of Game Design*.*

Over the course of their book on game design, Schell makes an elaborate argument for the need to grasp the player’s experience in order to make informed decisions

[31] Norman, Don. (2013). *The Design of Everyday Things. Revised and Expanded Edition.* Hachette UK. p. 5

[32] ibid. pp. 10-31

[33] Schell, Jesse. (2019). *The Art of Game Design: A Book of Lenses.* 3rd ed. CRC Press. p. 577

while creating games. The visual accumulation of these arguments results in a graphic which makes an excellent case for why mind-map-ping and searching for a truth by examining connections in a language can indeed be fruitful: see Illustration 2. But more importantly for our discussion, the illustration shows that the concrete design object, a game, is designed for a player and with the thoughts and behaviour of a player in mind. Both Schell's and Norman's conceptions have a strong emphasis on evaluating the design artefact and its consequences, positing the human end-user's experience as the benchmark – as do many works in game and design studies in general.

There are two reasons for why I do not use a result-oriented understanding of design. The first reason is, considering the approaches I presented here, neither Norman nor Schell offer too precise definitions for their result-oriented approaches[34]. Secondly, product-focused design thinking draws away from how the design is influenced by other conditions a designer must face, such as those of a tool or cultural context – both of which are of explicit interest for this thesis.

My understanding will thus be following that of Herbert Simon, who formulated their definition with no specific recipient in mind, and who operated in the tradition of the interdisciplinary thinkers of the 1960s and 70s, who this thesis will meet again later on.

Distinct to the idea of design is *execution*. I got inspired to differentiate by a CD Projekt RED job listing, where it is stated that a quest designer is responsible for "designing quests and implementing them"[35]. In a blog post, former Quest Director Mateusz Tomaszkiewicz goes into more detail, explaining how the team indeed even had a hard separation between designing/ pre-production and implementing/ production[36], which confirms Mareis in their writing that "typically the phase of planning precedes execution and industrial production"[37].

A note on the term "implementation": it is adequate for describing what is happening in a game development studio, at this specific job – namely the transformation of a plan for an in-game quest to a technical manifestation – but considering the wider

[34] While Schell does formulate a clear definition of design in their book, it is not result-, but, again, activity-oriented.

[35] CD Projekt RED. (2022). *Quest Designer | SmartRecruiters*. Online: https://jobs.smartrecruiters.com/CDPROJEKTRED/743999857102501-quest-designer (accessed: 16.03.2023).

[36] Tomaszkiewicz, Mateusz; CD Projekt RED. (2014). *The devil is in the details | CD Projekt RED's Official Blog*. Online: https://web.archive.org/web/20130116043021/http://cdpred.com/the-devil-is-in-the-details/ (accessed: 16.03.2023).

[37] Mareis, Claudia. *Theorien des Designs zur Einführung*. p. 39. Translation by the author.

context of general design, the term “execution”, being less attached to programming -matters, seems more fitting.

2.2 Tools and their “Grain”

In this section the line from design/ executions to tools, industrial production and game engines is drawn.

2.2.1 Engines as Tools

In their essay on the history of industrial mass culture, Wolfgang Ruppert states that “machine work” can be “sharply” distinguished to handicraft due to its structure[38]:

> »While it [manual craft] comprises a largely holistic form of production by means of tools and manual labour, production based on division of labour in the industry split the >mental< processes of construction and design of an object from the production by executing labourers with the help of machines.«

While I would argue that the division is not as sharp as they state – after all CD Projekt quest designers are specialised but nonetheless do both design and execution – Ruppert does draw attention to an important point that will be essential to this research: to facilitate easier design and execution processes, we have got *tools*.
Ernst Kapp proved to have a great influence on the philosophy of technology and provided a well-known theory of tools in their work “Grundlinien einer Philosophie der Technik” (1877):

> »[..] thus tools are a replacement of the organ itself. With its help did the hand craft further tools which in the technical imitation of the organic model - leaving the original approximative form-sameness - barely indicated any form-sameness. But they are nonetheless organic projections.«[39]

For Kapp, tools and metrics are a means to reach “the human’s highest purpose, [..] to be *measuring*, a *measurer* and *thinker!*”[40]. Kapp, who emigrated to the USA where they led a life as a farmer and practical scientist, certainly brought a sense of

[38] Ruppert, Wolfgang. *Zur Geschichte der industriellen Massenkultur.* p. 15. Translations by the author.
[39] Kapp, Ernst. (1877). *Grundlinien einer Philosophie der Technik*. In: Ziemann, Andreas. (ed.). (2019). *Grundlagentexte der Medienkultur*. Springer VS. p. 47. Translation by the author.
[40] ibid. p. 48

"US-american optimism"[41] into their work, which led to a one-sided perspective. Pondering about a new way to do philosophy of technology based on Kapp's thoughts, Karin Harrasser brings the perspective of *faltering* into the discussion. They identify "enhancement and control of self and consciousness"[42] at the core of Kapp's philosophy and, using prostheses as an example, ask why compensating weaknesses is not regarded as a reason for using tools as well. In the end, Harrasser argues to not only categorise those means as a tool, which increase a human's normalised base capabilities, but also those, which improve humans from below that imaginary line[43].

Now how do game engines and, more specifically, REDengine 3, fit into this notion of tools? In *The Unity Game Engine and the Circuits of Cultural Software*, Benjamin Nicoll and Brendan Keogh define a game engine as "a software tool that enables real-time interactive digital content to run on different platforms"[44]. This technical approach points to three important aspects of game engines: that they enable (us to do something), their artificiality and their digital materiality.

Usage of game engines can be said to shrink, modify, extend and altogether enable a human's game making abilities. Not only does this – together with the artificiality aspect – prove that game engines are indeed tools, since they are no organical part of the human body and enable us to make (certain) video games. It also shows an important argument for game engines and tools in general as a way to empower people: Nicoll and Keogh exemplify this with Grace Bruxner, who almost single handedly created a video game with the *Unity Game Engine* next to their studies[45]: access to the engine gave them the freedom to not deal with the implementation of lower-level systems such as a graphics engine. On a larger scale, we can see how CD Projekt RED created its own REDengine to empower themselves, the studio, in designing and implementing their visions.

The artificiality of game engines points to the fact that these pieces of software, while facilitating the creation of designed artefacts, are also themselves designed artefacts. Or, to quote Ernst Kapp: "One tool creates another"[46]. This opens up a perspective onto a hierarchy and history of tools: a product is made with a tool, and

[41] Harrasser, Karin. (2018). *Schwächeln. Technikphilosophie, Techniksubjektivität, Unvermögen.* In: Harrasser, Karin; Timm, Elisabeth (eds.). (2018). *Zeitschrift für Kulturwissenschaften. Homo Faber.* ed. 12. no. 2. pp. 149-159. p. 149. Translation by the author.

[42] ibid. p. 152

[43] ibid. p. 155-156

[44] Keogh, Brendan; Nicoll, Benjamin. *The Unity Game Engine and the Circuits of Cultural Software.* p. 9

[45] ibid. pp. 1-3

[46] Kapp, Ernst. *Grundlinien einer Philosophie der Technik*. p. 48. Translation by the author.

every tool can be said to be a product itself, being made with one or multiple "predecessor tools". REDengine 3 was written and constructed with tools like the programming language C++, Nvidia Hairworks and Scaleform GFx[47]. C++ is built on top of the physical processor's instruction set, while the processor was manufactured in industrial factories. Seeing how building factories comes back to manual labour done by construction workers, we can see how game engines might be but one item, a "stepping stone" in a line of an extensive human out-reach to new realms of possibility. We can see as well, how it becomes necessary for the human users and the tools to manage the ever growing, ever farther-from-origin design space. While it is beyond the scope of this thesis to discuss what the next items in the line of tools with game engines as tool-parents are, it is in scope to address the matter of highly abstracted "possibility realms", and how humans manage to deal with(in) them by means of constraints.

2.2.2 Grain and Design Constraints

Keogh and Nicoll adapt the woodworking metaphor of *grain* to describe how the Unity Game Engine

> »in a very abstract sense [..] has 'patterns' and 'fibers' — protocols, standards, and affordances — that orient users towards particular design methodologies.«[48]

We can generalise this statement to arbitrary tools, since it is an inherent property of any tool that it extends upon the base capabilities of a human and thus orients users towards successfully working in the particular fields they are extending to. Reformulating the above quote with the information now gained, it can be said that if a tool is used to execute, then the tool's grain will have predetermined some of the preceding design activity.

The point of a tool orienting us towards a *particular* execution and thus a particular design, is important. Depending on the concrete predeterminations – or: constraints – different actions can be performed (at all, or more efficiently). For example: the existence of a knob on a door allows a user to "grab" the door, an action impossible or significantly harder to do without a dedicated knob. The grain of a knobbed door thus orients a user towards opening by grabbing, whereas a not-knobbed door orients towards opening by pushing.

[47] Witcher Wiki contributors. (2022). *Modding the UI. Witcher 3 Modding.* Online: https://witcher-games.fandom.com/wiki/Witcher_3_Modding#Modding_the_UI (accessed: 16.03.2023).

[48] Keogh, Brendan; Nicoll, Benjamin. *The Unity Game Engine and the Circuits of Cultural Software.* p. 64

According to Don Norman, constraints can be differentiated into four subclasses[49]. First, there are *physical constraints*. These hinder and enable users by means of physical laws, such as a key's unique shape preventing use with other doors and a coded search bar[50] enabling faster finding. Second, *cultural constraints*. As Norman explains, humans are furthermore governed by their conventions: if all major game engines feature a component system with similar workflows, then a designer using REDengine 3 for the first time will be able to draw from that knowledge and understand the engine more quickly. *Semantic constraints* "rely upon our knowledge of the situation and of the world"[51]. In the TW3 game files the following file exists:

```
/r4data/quests/main_npcs/mousesack.w2ent
```

Without any knowledge of video game jargon (e.g. "NPC") and Witcher-lore (the character Mousesack), one might assume that this file is about an important journey as a sack of mice, while presence of aforementioned knowledge constrains the possibilities such that one knows this to be the non-playable side character "Mousesack" from the Skellige-main quest. Lastly, Norman presents *logical constraints*. These work by means of the user's reasoning capabilities: if the user finds a quest block about objective change with four input gates labelled as in Illustration 3, and if they did not know that objectives can be deactivated in the game, then they will know now.

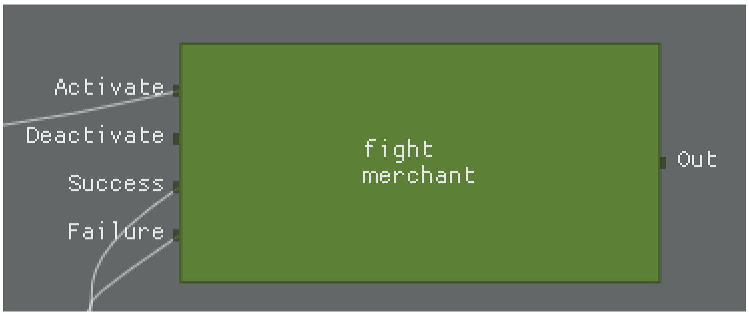


Illustration 3: Screenshot of an objective change node in the quest editor of the radish modding tools.

Having seen now what grain and constraints each mean, how do these terms relate to each other? If a constraint is "something that limits the range of a person's

[49] Norman, Don. *The Design of Everyday Things.* p. 125
[50] Here I assume digital code and data to be physical matter.
[51] ibid. p. 129

actions"[52] and grain is the property of something being "orienting towards"[53], then I conclude: both terms state a fact, namely the action-boundedness, but "grain" explicitly includes the implication that limiting a user's actions changes their stance, approach or relation to the tool.

Working with goals such as deadlines, product vision, financial budget and personal energy (to name four important ones), it is in the interest of any user to work in such a way that design and execution are as easy as possible without departing from their goals. Now *easiness* is relative to a worker's skill in doing something, and thus the value of tools by Kapp's/ Harrasser's definition becomes evident since they can locally increase the effect of a skill.

Concretely, the constraints tools offer manifest as shortcuts during working activities. Instead of having to use our hands, we can use a shovel to move much more earth in the same span of time. Instead of having to think philosophy from the ground up, we can build on and modify an existing theory. Instead of having to decide each feature and parameter of a character controller, we can use the default one with its default settings in Unreal Engine 5 and then improve it and adapt it to our (specific) needs. The omission of steps in between allows workers to focus on or think of other matters, a principle which powered specialisation and modern ideas and phenomena of constant growth by means of technology[54].

The first implication for the designer is that they need to consider the tools at hand during their plans. Depending on the given selection, that is, the problems each tool allows to solve easier, a designer might plan more ambitious. But equally, a designer might need to think about a non-intuitive way to solve a problem, because of a tool's constraints. At this point it becomes clear why practice-oriented design advice often advocates "out-of-the-box thinking" and creative problem-solving. A tool does not only help us solve a problem, it will also solve the problem in a certain, predetermined manner – in ways which exclude other ways.

Tool users are in between the hard facts given by physical constraints, the shifting borders of cultural conventions and concrete designers' ideas for a specific game design problem. Consequently they might create a shared design language based on what the tool allows, pinning the value of some variables of the problem and creating a tighter design that leaves enough space to still let the intended design

[52] Cambridge University Press. (2022). *CONSTRAINT | meaning, definition in Cambridge English Dictionary*. Online: https://dictionary.cambridge.org/dictionary/english/constraint (accessed: 16.03.2023).

[53] See the definition of grain cited above.

[54] Here understood as a term semantically close to "tools".

unfold. It is the origin of such a shared language in a tool's inherent language, which I would like to explore lastly in this chapter in order to then explain what I will formulate in Chapter 4.

2.3 Game Engines' Languages

In 2002, when the field of game studies was forming and literature on designing games was rare, Greg Costikyan published a paper titled: *"I Have No Words & I Must Design: Toward a Critical Vocabulary for Games"*[55]. In it, they discuss the problem of natural language terms like "gameplay" and "interactive", whose semantics are too manifold and unclear for them to be productively used in conversation. Now this is a problem, because scholars and developers by the very nature of their respective work – creating a highly integrated work with other people and evolving the pool of human knowledge – need to be able to communicate about the subject of video games. Building on structuralism as a scientific approach and grain[56] as a decisive property of game engines as tools, I will present a kind of language specialised in the domain of engine technology: object languages and, derived from that game engine languages.

2.3.1 Language as a Design Studies Lens

About 30 years earlier than Costikyan, Herbert A. Simon notes that design manifests in many disciplines. In fact, it is hard to imagine any profession, in which decision-making is irrelevant, posing the question, how any concept of design can be made productive for discussing concrete issues. In their paper on the relation between theory and design, Ken Friedman formulates the following: "Applied research adapts the findings of basic research to classes of problems"[57]. Doing so has advantages: picking a specific class of problems (like *quest* design) automatically decreases the number of variables to consider and thus helps in grounding a theory. Furthermore, by adapting general ideas for specific applications, one preserves the properties the general concepts had. It might be the potential of such overarching theories covering large concrete space, which inspired the scientific movement that is now called *structuralism*.
Structuralism can be regarded as

[55] Costikyan, Greg. (2002). *I Have No Words & I Must Design*. In: Mäyrä, Frans (ed.). (2002). *Proceedings of Computer Games and Digital Cultures Conference*. vol. 1. Tampere University Press.

[56] The metaphor, not the wheat!

[57] Friedman, Ken. (2003). *Theory construction in design research: criteria: approaches, and methods*. In: Cross, Nigel (ed.). (2003). *Design Studies*. vol. 24. no. 6. p. 510

> »the attempt to provide the subject's thought with fixed meaning by seeking to locate this subject in a finite and thus basically overseeable system, in a taxonomic order of elements and oppositions«[58].

This way of thought is interested in the potential of fixed meaning: it generates, in extreme cases, a "finite context for analysis"[59], a frame in which solving problems is but an issue of finding the applying rules of symbolic manipulation in the correct order and manner, not unlike mathematical proofs and algorithms developed in computer science.

In studies of design, structuralism became visible when the studies transitioned away from early 20th-century's "artistic questions on style and form-and-function debates"[60], and in the 1960s/ 70s "experienced an interdisciplinary extension through planning- and problem-solving research and postmodern structuralist influences"[61]. An example of this tradition can be found in Herbert A. Simon's notion of design, which I discussed in Section 2.1. Having chosen a definition close to Simon, I thus grounded this chapter's writing in structuralist thought and in the design discourse of the 1960s and 1970s, which, according to Mareis, "was constitutive for the systematisation of theories and methods of design and its scientification [..]"[62].

"Rightfully linguistics is declared as the origin of structuralism" writes Gilles Deleuze in their handbook on "recognising structuralism"[63]. This is one of many motivations to take a closer look at *language* as a lens for discussing REDengine 3's grain.

As a part of establishing the idea of an engine's language, I will be able to argue why this is a useful conceptual tool for analysing the collective intended effects of the creators, which is useful for both scholars and creators. In other words, it is used as a *lens*, as a selected among many perspectives to perceive a game through and test against[64] – even though language as a lens is more abstract than the lenses in *The Art of Game Design*.

Another key reason for my interest in engine languages is that they connect the study of games to their production and to (human) production itself as an activity -

[58] Pias, Claus. (2010). *Poststrukturalistische Medientheorien*. In: Weber, Stefan (ed.). (2010). *Theorien der Medien.* 2nd ed. UVK Verlagsgesellschaft mbH. p. 252. Translation by the author.
[59] ibid.
[60] Mareis, Claudia. *Theorien des Designs zur Einführung*. p. 31. Translation by the author.
[61] ibid.
[62] ibid.
[63] Deleuze, Gilles. (1992). *Woran erkennt man den Strukturalismus?* Merve Verlag. p. 8 Translation by the author.
[64] cf. Schell, Jesse. *The Art of Game Design: A Book of Lenses*. p. xl

we already saw the introduction of design and tools as important framings - which by means of the language term will once more connect to the engine, contributing to the golden thread of this thesis being formed and theory and practice strung together.

To make a last – and maybe the most important – motivational point: a major purpose of this text is to “let the engine speak”, and see how design arises from the instrument at hand[65]. In game studies, games are often viewed from the perspective of the game experience (while playing it). This has the advantage of directly addressing what many find most interesting about media: the mediated experience, happening at the human player. Following this pathway leads us to identify, infer and differentiate mechanics, systems, scenes, places and so forth based on what we perceive during playthroughs. A thorough analysis however might require more than a reconstruction or deduction of the game’s nature: it might indeed require looking at the construction of the work itself, which happens when game developers design and implement e.g. quests and in the second step when the game executable is running, taking the game files and player input as parameters. This motivates taking a look at the file formats, scripted code, hard-wired reference systems and so on contained in the digital asset that is TW3 and understanding what kind of *language* these elements form.

2.3.2 Boudrillard’s Object Language

Language, in this work, is understood as “a system of communication”[66], or: the amalgamation of any conditions structuring communicative acts in a given context. Typically, we refer to *natural language* when using “language” as a term and mean the speech used by humans to communicate with each other (and animals and machines, in some (edge) cases[67]). In this work, however, I am interested in a different kind of language: the engine’s own language, the language of a (possibly inhuman) object. This notion of an object’s language was developed by Jean Boudrillard in *The System of Objects*:

> »Every transition from a system to another, [..] every commutation within an already structured system, every functional synthesis, precipitates the emergence of a

[65] cf. Engell, Lorenz; Siegert, Bernhard. (2012). *Editorial.* In: Engell, Lorenz; Siegert, Bernhard. (2012). *ZMK Zeitschrift für Medien- und Kulturforschung 3/1/2012: Entwerfen.* Meiner. p.5

[66] cf. Cambridge University Press. (2022). *LANGUAGE | meaning, definition in Cambridge English Dictionary.* Online: https://dictionary.cambridge.org/dictionary/english/language (accessed: 16.03.2023).

[67] I am referring to human-machine communication where the machine accepts and produces natural language-like content, such as ChatGPT.

> meaning, an objective pertinence that is independent of the individuals who are destined to put it into operation; we are in effect at the level of a language here«[68]

Boudrillard thus deducts the language-like quality of structured objects by observing how syntheses into larger structures in such objects generate meanings. Another argument for object languages stems from Walter Benjamin, who posed that "[t]here is no happening or thing neither in animate or inanimate nature, which does not participate in some manner in language, for it is essential to each, to communicate its content"[69].

Benjamin is able to write such a general statement since they have a looser understanding of language than Boudrillard, who finds that some objects are not stable enough in their structure to justify saying they have a language:

> »For technology, unlike language, does not constitute a stable system. Unlike monemes and phonemes, technemes[70] are continually evolving«[71]

Boudrillard furthermore argues that all our "practical objects" are "in perpetual flight from technical structure towards their secondary meanings, from the technological system towards a cultural system"[72], concluding that an object language "cannot be described *scientifically* unless it is treated *in the process*"[73].

I concur with Boudrillard only to some degree at this point. While I agree that the stability of an object should be considered when it comes to its structure – which is exactly that which endures – I would, for one, not say that technology is different to language in that regard. For language changes as well or is at least able to do so; it is never completely stable, complete and set for eternity. On the other hand, certain variants of a language can become stable to the point that people use them to communicate. This becomes most evident when looking at for example the development from older versions of spoken English to its current state or the explicitly labelled *versions* of programming languages like C++. Regarding the language of the technological object TW3/ REDengine 3, the stability-matter came to light for instance during the advent of patch 4.0, which was released during the

[68] Boudrillard, Jean. (1996). *The System of Objects*. 1st ed. Verso. p. 13
[69] Benjamin, Walter. (1916). *Über die Sprache überhaupt und über die Sprache der Menschen*. Online: https://signaturen-magazin.de/walter-benjamin--ueber-sprache-ueberhaupt-und-ueber-die-sprache-des-menschen.html (accessed: 16.03.2023). Translation by the author.
[70] A kind of atomic unit of object languages in Baudrillard's terminology.
[71] Boudrillard, Jean. *The System of Objects.* p. 16
[72] ibid. p. 14
[73] ibid.

development of this thesis. The state of game patch 1.32 was for a long time the latest patch (from 2017-2023) and determined a stable object language. When patch 4.0 was released, the language respectively evolved – but many of TW3's/ REDengine 3's aspects did not change: for example, the way classes with multiple components are saved in a binary format (called *CR2W*) has been the same since TW3's release, thus enabling modders to reverse engineer the technology CR2W. Based on this, stable modding tools could be built, to read and manipulate the respective files.

Boudrillard's mentioning of the "flight towards secondary meanings", imposed by culture and practical usage, marks a core point of their position. They were particularly interested in the (usage-)context of objects and argued that objects being culturalised makes structural analysis alone insufficient, since the object language then becomes intertwined with and changed by the specific practices surrounding the object. I agree that this indeed obfuscates and can additionally destabilise the object's language, but observing the TW3 modding scene, I found that many workflows, tools and routines changed only very slowly or not at all, thus not changing many meanings assigned by this cultural context as well. Nonetheless this context exists and developed its own stance towards how the object of interest is regarded, which is why Chapter 3 will provide a dive into the modding circumstances.

Having discussed what an object's language is, I now want to close in on the contents of such a language. Boudrillard uses the abstraction of a *technological plane* to point towards the "technological reality of objects"[74], such as the code and data forming REDengine 3 and TW3. Elements of this plane are represented by semantic and functional units Boudrillard calls *technemes*, analogous to the *phonemes*[75] of natural language.

> »the elements of a coherent system that is never directly experienced, never apprehended at the practical level. [..] by analogy with linguistic phenomena, those simple technical elements different from real objects -upon [sic] whose interplay technological evolution is founded might well be dubbed 'technemes'«[76]

These technemes, then, are what constitutes the atomic units of an object's language. In one of Boudrillard's examples, an "internal-combustion heat engine" features a cylinder head for cooling, a techneme which, combined with a "ribbing",

[74] ibid. p. 11

[75] Phonemes are meaningful classes of bare sounds used in natural language.

[76] Boudrillard, Jean. *The System of Objects.* p. 13

converges to an undistinguishable structure[77]. In a game engine, mesh and texture as technemes can converge for example to a character model and are no longer visible from the outside as distinct units.

Ian Bogost developed a somewhat similar notion with their *unit operations*, though one can find at least one crucial difference between them: Bogost understands unit operations as “modes of meaning-making that privilege discrete, disconnected actions over deterministic, progressive systems”[78], which on the first glance makes the two concepts clash immediately in the fact that unit operations have a processual nature while technemes are (by tendency) more static, delimited elements of a structure. On the other hand we can find that technemes, like unit operations, focus on the isolation of atomic features in a larger object of interest. Furthermore and lastly, both concepts integrate that there is a semantic aspect to such an atomic feature.

Bogost is interested in performing a “shift away from *system operations*”[79] and concentrates on developing a space where computer games and literature can be analysed with a common lens. This thesis does not have much to add to this call to action; much rather it is building on these thoughts by considering the entire range from processual and inelastic atomic features to complexer integrations and circumstances while analysing its subject.

A conceptual tool to grasp said complexer integrations in (object) languages is the idea of a *pattern*. Patterns were popularised in science and design by architect Christopher Alexander in 1979 mainly through the book *A Pattern Language*, in which they presented 273 patterns to solve certain architectural issues. In the book's prequel, *The Timeless Way of Building*, Alexander draws a line from a “nameless quality” in our lives, to how this quality can be effectuated by means of patterns, to how a collection of interrelated patterns forms a language. While Alexander has been criticised for romanticising spaces where people live, and advocating a regression to older architectural styles, their idea of patterns as building blocks for larger structures and concepts of great generative potential gained great influence: not only architecture, but also in other disciplines such as software development or sociology. In game design literature, Alexander’s ideas are echoed by popular books like Schell’s *The Art of Game Design* or Totten’s *An Architectural Approach to Level Design;* more recently, Chris Barney wrote an entire book (*Pattern Language for Game Design*) solely concerned with applying the idea to game design.

[77] ibid. p. 12

[78] Bogost, Ian. (2006). *Unit Operations. An Approach to Video Game Criticism.* The MIT Press. p. 3

[79] ibid.

The essence of patterns is how they designate repetition. Alexander comes to recognise them by pondering architectural “characters” and concluding that these are determined by those events and architectural features that appear more often than others[80]. They then labels these structural repetitions patterns:

> »Evidently, then, a large part of the “structure” of a building or a town consists of patterns of relationships.«[81]

The success of this term can be attributed to several factors. For one, it is a general term, since structure is one of the fundamental aspects of our perception and repetition as a phenomenon far from scarce. Secondly, patterns are scalable. Unlike technemes or unit operations, which can be regarded as smallest patterns (assuming a level of scale we chose not to come under), patterns function on the entire scale range. This becomes evident in *A Pattern Language*, which contains patterns, for instance, for the positioning of windows and the composition of town districts alike. And lastly, the term “pattern” is relatively easy understood, with it being no academic compound word like “unit operation” or neologism like “techneme”.

The introduction of patterns, unit operations, technemes and their interplay in an object language concludes this excursion into theories of structure and language. The last paragraphs of this subsection return to and at last connect my notions of design, tools and language.

2.3.3 Tool Design Languages

In this thesis, I will use the term of a tool’s *design language*, in order to refer to those technemes, unit operations and patterns of a tool’s language, which drive the designer to choose particular workflows or implementations. In short: a tool’s design language is the grain subset of the tool’s language.

I construct this term to further delimit the focus of the analysis by means of wording, and because what is signified by this term is at the heart of this work: tools influence the planning and execution of content that they were made for, and the collection of a tool’s influencing aspects is a kind of language via which one might understand the tool.

[80] Alexander, Christopher. (1967). *The Timeless Way of Building*. Oxford University Press. pp. 66-68, 87

[81] ibid. p. 87

A tool's design language defines a collection of the tool's paths of least resistance to solving design problems and, more importantly, paths of less resistance than when the developers were constructing solutions without the tool. As discussed in Section 2.2, design languages (the tool's grain) encourage standardisation and thus access to more specialised designs the more inflexible they are. Companies like CD Projekt RED exploit this fact when building their own engines. Writers, cinematic designers and quest designers, for instance, can formulate what kind of grain they require from a quest editor to be oriented towards workflows that are efficient. In conversation with tools programmers and game directors a ressource (time, money, ..)-sensible solution can then be devised and implemented for example in a pre-production stage, to allow for the high-quantity main production phase to work efficiently. For *Cyberpunk 2077*, CD Projekt RED developed a custom scene editor in order to be able to facilitate their vision of a first-person game with seamless cutscenes[82].

At times, video game developers publish (parts of) their tool sets such that modders may modify or extend the game developed with the tool, or develop a new game entirely. In the next chapter I discuss the case of the partly released game engine REDengine 3 as a storytelling tool as well as both chance and headache for modders.

[82] Pierściński, Filip. (2022). *Introduction of the Interactive Cinematics in 'Cyberpunk 2077'*. Online: https://www.youtube.com/watch?v=exqPwGIxryI (accessed: 16.03.2023).

Chapter 3. Modding Stories for *The Witcher 3*

This chapter provides a contextualisation and short explanation for the tools I am inspecting (REDengine 3 and modding tools*)*, and the method (modding) by which the inspection is done. Connecting to the thesis’ goal, I will put a focus on what conditions the context induces. Having closed the rather abstract framing in Chapter 2, the intention behind Chapter 3 is to provide a consecutive, accelerated concretisation of my lens on the object of interest.

3.1 Video Game Storytelling

First of all, I will investigate the narrative game design context.

3.1.1 Short History of Storytelling

Narrative as a mode of human communication[83] has a long tradition, starting with the old practice of oral and gestical storytelling which was and continues to be deeply connected to human culture[84]. One of the side effects of this thesis will be that it demonstrates how some older storytelling traditions are – in a way – kept up and developed in the aesthetics of particular new media such as video games.

Other older, but important narrative forms include for instance theatrical pieces and literature. Theatre, which has its first predecessors in egypt performances dating back to 2000 B.C., tells stories with more focus on the visual arrangement and design of actors or material, as well as their change of locations on the stage. One of theatre's major successors is film, a medium which lost the immediate character of theatre but gained lots of possibilities in return: for example to make hard cuts between one image and the successor or to steer the audience's look by means of the camera: all based on the fact that film, in opposition to theatre, is not *live*, but pre-built and saved on for example a Compact Disc. When the letterpress was invented during the Renaissance, soon written stories gained more momentum as a part of human culture, and through the nature of the medium – distinct symbols fixed on a page – emphasised a more structured experience of the narrated work.

All of these forms pose certain conditions on a storyteller, who is a designer and executor in their own right: they plan a story (e.g. using a storyboard and script in

[83] cf. Crews, Frederick C. (1977). *The Random House Handbook*. 2nd ed. Random House. p. 13

[84] National Geographic Society. (2022). *Storytelling and Cultural Traditions.* Online: https://education.nationalgeographic.org/resource/storytelling-and-cultural-traditions (accessed: 16.03.2023).

film) and execute that plan (the actual shooting), with both not necessarily happening one after another. To produce these forms, various devices exist, and these devices can be considered tools: the stage decoration in theatre allows the storytellers to convey the scenery visually and without the human actors. Different devices imply different design languages. Whether a human uses a stick or their voice to convey the hubris of a character changes not only how listeners experience the story, but also how the storyteller (potentially ad-hoc-) designs their piece in the first place. Thus we can see how the thoughts mapped out in Chapter 2 transfer to storytelling.

Modern manifestations of storytelling include for example narrative movies or video games, with both starting to become important during the 20th century. In this work I am interested in the video game medium and its genre of open world, narrative-driven role playing games (RPGs).

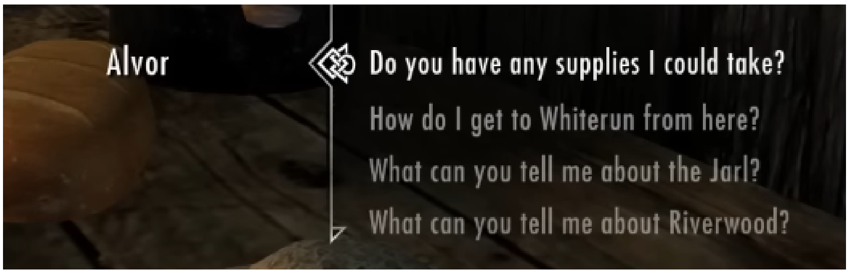
Video games are marked by their *distributed authorship*, which Stephanie C. Jennings understands as

> »the interplay of negotiated capacities of a number of actors (including but not limited to developers, publishers, and players) to create the content, structures, form, and affordances of video game works«[85].

This quote highlights a separation between stakeholders who typically affect the video game experience before the players' involvement, such as developers and publishers, and those who interact with the final product, which are the players. This allows us to imagine a spectrum of influence on the video game experience. On one end of the spectrum reside elements that are mostly influenced by the developers/ publishers, and on the other end are elements mostly determined by players.

Using this spectrum, storytelling methods in video games can be compared and discussed. Pre-rendered cutscenes can usually be said to have a stronger developer-imprint compared to other means. Regular dialogue scenes as seen for instance in *The Elder Scrolls V: Skyrim* offer a bit more control to the player.

[85] Jennings, Stephanie C. (2016). *Co-Creation and the Distributed Authorship of Video Games*. In: Valentine, Keri Duncan; Jensen, Lucas John (eds.). (2016). *Examining the Evolution of Gaming and Its Impact on Social, Cultural, and Political Perspectives*. IGI Global.



Alvor

Do you have any supplies I could take?

How do I get to Whiterun from here?

What can you tell me about the Jarl?

What can you tell me about Riverwood?

Illustration 4: Screenshot of a dialogue choice in The Elder Scrolls V: Skyrim[86]*. Depending on the option chosen by the player, a different chunk of speech is played.*

While locked in movement and to some degree in camera angle, players may, by means of dialogue choices, get the chance to influence the traversal of the speech graph lying underneath such a scene. Environmental storytelling is more collaborative: while developers indeed need to – as Henry Jenkins notes – design the space[87] in order for anything to *be there*, the spatial experience would never occur if the player did not look into the respective location, bring their own decisions on e.g. the exact path or camera angles into the experience and possibly even try to interpret what they find. In the middle of the spectrum, quests can be located. These often provide players with pre-defined objectives and events, but let the player move the avatar themselves in between. The developer-opposite side of the spectrum is closed with stories that, in extreme cases, are told by the player alone: Reddit-user "RazzDaNinja" crafted a character named "Skullhunter" who ignored their fate of being the heroic Dragonborn and chose to excel in fighting ever more powerful enemies. The character required a set of rules for their playstyle:

> »I wasn't allowed to use magic spells or buy weapons. Pure tactics, magic staffs and physical abilities only. I had to either craft or loot weapons and potions. No stealing. No killing defenseless civilians. No stealth archer (I could only melee stealth kill). Always wear a helmet that covers my face, removing it only for 1-on-1 duels.«[88]

The object of study in this thesis is quests in video games, so I will conclude my discussion of general storytelling methods with this area, before finally moving on to modding and the creation of such content within quest systems.

[86] Bethesda Game Studios. (2007). *The Elder Scrolls V: Skyrim* (1.1).

[87] cf. Jenkins, Henry. (2004). *Game Design as Narrative Architecture.* In: Wardrip-Fruin, Noah; Harrington, Pat (eds.). (2004). *First Person. New Media as Story, Performance, and Game*. MIT Press. pp. 118-130. p. 121

[88] RazzDaNinja. (2020). [Comment]. *Reddit*. Online: https://www.reddit.com/r/skyrim/comments/emlue0/player_stories/ (accessed: 16.03.2023).

3.1.2 Video Game Quests

For Jeff Howard, a quest is “a journey across a symbolic, fantastic landscape in which a protagonist or player collects objects and talks to characters in order to overcome challenges and achieve a meaningful goal”[89]. This formulation shows some key aspects of video game quests.

For one, a quest often is a certain journey a player is motivated to do[90]. Howard talks about the player’s experience of a quest or the *actualised* quest[91], which has the seriality that is part of the idea of a journey: in quests like “A Towerful of Mice”[92] from TW3, the player as Geralt can decide many aspects, like from which direction they approach Fyke Isle or whether to trust the potential monster Anabelle. But they will, in any way, always experience one consecutive subset of events in one playthrough if they do not modify their progress to be set to some earlier state (by e.g. reloading): they will always first enter the island and then talk with Anabelle, for example.

Meanwhile, a quest can also be understood as an interrelated collection of events, a template lying in the game files or in the working memory waiting to be instantiated by the game or the player. I will call this concept the *template* quest, which we will find formalised as a graph structure in REDengine 3 in Section 4.1. The concept of pacing, that is, the course of tension over time[93], illustrates the difference: the player usually experiences one pacing since they experience the quest once. But since often quests do not fully, but only partly determine a player’s experience by e.g. letting them decide how to approach a location or whether to complete an optional objective or not, multiple pacings need to be taken into consideration by designers[94]. And these are best analysed by determining which paths may be actualised within the template quests’ events and their relations.

Quests are also characterised by their goal-driven action. In the above definition this is incorporated by the notion that a video game quest’s function is to allow a player to “overcome challenges and achieve a meaningful goal”. Howard brings up three classes of meaningful motivation provided by quests: *initiation* denotes a player experiencing a “gradual movement up through formalised ‘levels’ of achievement”[95].

[89] Howard, Jeff. *Quests. Design, theory, and History in Games and Narratives*. p. xxi

[90] cf. ibid. p. 6: “a ‘journey’ is the spatial movement and temporal duration entailed by a quest”

[91] cf. ibid. p. 1

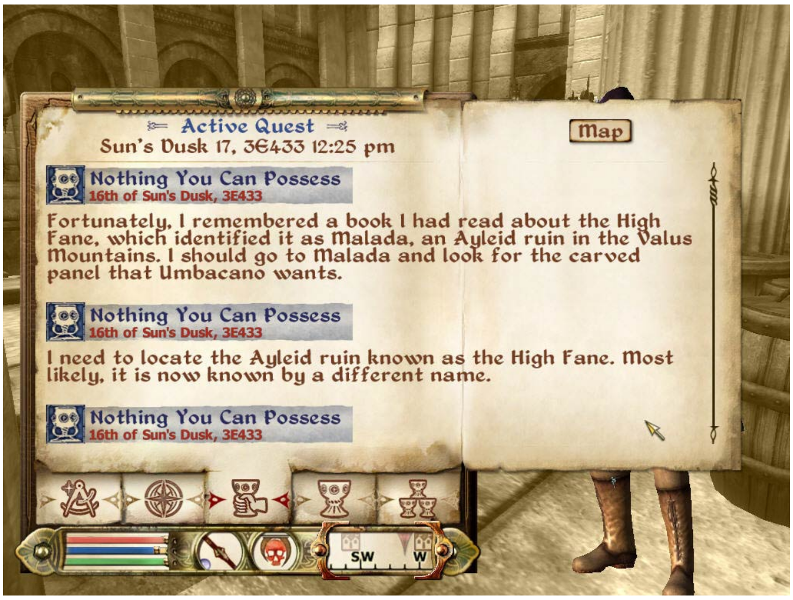
[92] Witcher Wiki contributors. (2023). *A Towerful of Mice*. Online: https://witcher.fandom.com/wiki/A_Towerful_of_Mice (accessed: 24.03.2023).

[93] Sasko, Paweł; Digital Dragons. (2017). *Life, Love and Quest Design. Anatomy of Quests in The Witcher 3: Wild Hunt*. Online: https://www.youtube.com/watch?v=g5TH9KakBDw (accessed: 16.03.2023).

[94] ibid.

[95] Howard, Jeff. *Quests. Design, theory, and History in Games and Narratives*. p. 26

For instance in the form of encounters set up in specific quest moments – boss fights in their extreme – players are given the goal to challenge themselves. For fights, REDengine 3 provides two generic mechanisms, communities and encounters, which I will discuss in Section 4.2. *Narrative* motivation stems from the question “What happened/ happens/ will happen?” and is related to quests insomuch as the sequential nature of actualised quests allows the player to e.g. speculate about upcoming events surrounding a character. We will see how designers can micromanage when which events are happening in Section 4.1. *Thematic* motivation is present when the player “acts out a set of ideas that comment upon”[96]. Howard brings forth the example of “basic conflict between good and evil enacted in the game’s battles”[97], but we will see how REDengine 3/ TW3 (do not) allow for much more interesting morales in Chapter 4. As Howard shows in a later section, such “purposive action” can be supported by *explicit* objectives that are given to the player in a quest log[98]. In Illustration 5 the active quest “Nothing You Can Possess” from *The Elder Scrolls IV: Oblivion* entails several subsequent explicit objectives which are written down as paragraphs of text.



Active Quest
Sun’s Dusk 17, 3E433 12:25 pm

Nothing You Can Possess
16th of Sun's Dusk, 3E433

Fortunately, I remembered a book I had read about the High Fane, which identified it as Malada, an Ayleid ruin in the Valus Mountains. I should go to Malada and look for the carved panel that Umbacano wants.

Nothing You Can Possess
16th of Sun's Dusk, 3E433

I need to locate the Ayleid ruin known as the High Fane. Most likely, it is now known by a different name.

Nothing You Can Possess
16th of Sun's Dusk, 3E433

Illustration 5: Screenshot of the quest log in The Elder Scrolls IV: Oblivion[99].

There are also *implicit* objectives, that is, objectives which are heavily suggested, but which are not explicitly stated anywhere. Common forms of this are landmarks or motivations set up in “show, don’t tell”-cutscenes. Throughout Chapter 4 we will

[96] ibid. p. 28
[97] ibid.
[98] ibid. pp. 108-109
[99] Bethesda Game Studios. (2007). *The Elder Scrolls IV: Oblivion Game of the Year Edition* (v1.2.0416).

see how motivation can be enticed by a variety of means with REDengine 3, but, in comparison to the game built on top, it will become evident how TW3 heavily leans to an “explicit objective” style.

One key term for Howard is “symbolism”. Observing the discourse of game theorists and designers at the time, they found them to “increasingly focus on meaning and interpretation”[100]. So, interestingly, a noticeably subset of views about games underwent a similar shift towards semantics around 2004 like design did in the mid 20th century (see the discussion in Section 2.3). This connection is underlined by the fact that Howard brings up Ian Bogost’s unit operations as a prime example for this shift: they use a major point from *Unit Operations* – the argument that “games and literature can meet at the issue of interpretation”[101] – to support their thesis that quests are inherently meaningful and that their connection to quest narratives in literature is indeed substantial. For the analysis of quests, Howard argues that unit operations are not sufficient, since quests are progressive and systemic, which are “two qualities that he [Bogost] denies to units”[102]. This supports that I chose the lens of “language”, including the much more flexible patterns, to analyse the tool in which quests are defined.

Howard writes in their quest definition that a player “collects objects and talks to characters” to reach their goals. While these are not the only means of progressing a quest, they are two major categories of entities which the player might interact with. Objects, in Howard’s understanding, encompass *minor*, *functional* and *quest* items. Minor objects are “largely useless” to the player, but “give a sense of realism”[103]. Functional objects modify the player’s abilities in some way (such as increased movement speed). For quest items, Howard has no final explanation, but they propose that they “play an essential role in the back stories behind a quest” or “possess great magical power within the rules of the game”[104]. The second wording has a seemingly unnecessary focus on magic, but, going for a more progression-logic oriented approach might be reformulated towards more clarity like this: quest objects have special rules defined for them which make them relevant for a quest’s progression. In Sections 4.1 we will see how the modularity of entity, item and reward definitions in REDengine 3 allows for the three kinds to be flexibly mixed, attributed and used, motivating designers to come up with interesting objects for their quests. For characters, a similar principle holds. TW3 Quest designers can profit heavily from several specialised functionalities provided by character entities,

[100] ibid. p. 4
[101] ibid.
[102] ibid. p. 4-5
[103] ibid. p. 77
[104] ibid.

such as the modularity of body parts, sound and effects, clothing, hair and a generic system for poses, gestures and other animations. Much of this is based in REDengine 3's component system, which is similar to that in other engines like the Unity Game Engine or Unreal Engine. I will discuss all such matters in Section 4.3.

Lastly, following Howard's definition, a video game quest happens in a player's traversal of a virtual landscape. Given this, a first question might be how exactly that, which is deemed a quest in a given engine, can influence the world and how the world can in return influence the quest. For some of these use cases, REDengine 3 suggests very rigid patterns like the weather system, while it provides an extremely easy and flexible solution for player-wait-events with the integration of trigger boxes into the quest system. I will discuss relations between landscape and quest in Section 4.2. Landscape has also a much more inherent and concrete effect on quests, since by the mere form of landscape, quest experiences can vary greatly. A small island in the middle of the Sea will have a different effect than an enclosed valley in the mountains. This work is often attributed to level designers and will not be followed up on further here.

This concludes the introduction to the video game and storytelling aspect I am examining.

3.2 Modders under CD Projekt RED Conditions

This section covers developer CD Projekt RED, the concrete game and engine they created, as well as the modding community forming around them.

3.2.1 Work Optimisation at CD Projekt RED

CD Projekt RED is a polish, stock exchange-listed, independent[105] game development studio. The umbrella company, CD Projekt, started as an importer and localiser for the Polish video game market in the 1990s[106] .

In 2003 the predecessor of CD Projekt RED was founded, which started the development of *The Witcher*, based on the novels by Andrzej Sapkowski and built with a heavily modified version of BioWare's *Aurora Engine*. The game was released in 2007 and was followed up on with *The Witcher 2: Assassin's of Kings* (TW2), for which the studio for the first time developed their own internal *REDengine*, whose successor *REDengine 2* for the same game supported multiple platforms.

[105] In the sense that the company has never been bought up.

[106] CD Projekt. (2023). *History - CD PROJEKT.* Online: https://www.cdprojekt.com/en/capital-group/history (accessed: 16.03.2023).

Year	Title	Engine
2007	The Witcher	Aurora Engine
2011	The Witcher 2: Assassin’s of Kings	REDengine 1, 2
2015	The Witcher Battle Arena	Unity Game Engine
2015	The Witcher 3: Wild Hunt	REDengine 3
2018	Gwent: The Witcher Card Game	Unity Game Engine
2018	Thronebreaker: The Witcher Tales	Unity Game Engine
2020	Cyberpunk 2077	REDengine 4
TBA	The Witcher Remake	Unreal Engine 5
TBA	New Witcher Trilogy	Unreal Engine 5+

Table 2: Listing of past and future video games released by CD Projekt RED where the underlying engine is known[107][108][109][110].

For the highly successful (at least commercially) TW3 and *Cyberpunk 2077*, REDengine 3 and *REDengine 4* were developed respectively. In 2022, the studio announced that for future projects they are developing on Epic Games’ Unreal Engine 5 (and future versions)[111].

As a company, CD Projekt RED is, among other things, led by economic interest: to generate revenue and thus, to produce high quality games as fast as possible. In Section 2.2 we saw how the very nature of tools is their ability to shortcut or enable work routines by providing new ways of doing, which motivates any game developer to optimise their selection, creation and usage of tools. This shows quite prominently in CD Projekt RED’s former, current and future video game engines and in the job positions associated with them.

[107] Wikipedia contributors. (2023). *CD Projekt*. Online: https://en.wikipedia.org/wiki/CD_Projekt (accessed: 16.03.2023).

[108] Wikipedia contributors. (2021). *The Witcher Battle Arena*. Online: https://en.wikipedia.org/wiki/The_Witcher_Battle_Arena (accessed: 16.03.2023).

[109] Wikipedia contributors. (2023). *Gwent: The Witcher Card Game*. Online: https://en.wikipedia.org/wiki/Gwent:_The_Witcher_Card_Game (accessed: 16.03.2023).

[110] Wikipedia contributors. (2022). *Thronebreaker: The Witcher Tales*. Online: https://en.wikipedia.org/wiki/Thronebreaker:_The_Witcher_Tales (accessed: 16.03.2023).

[111] CD Projekt RED. (2022). *A New Saga Begins*. Online: https://www.thewitcher.com/en/news/42167/a-new-saga-begins (accessed: 16.03.2023).

Table 2 shows that in addition to a major engine (REDengine/ Unreal Engine) for "story-driven role-playing games", being arguably their main focus[112], CD Projekt RED also uses the Unity Game Engine to produce titles. This is a clear indication that the company exploits engine grain: the Unity Game Engine is apparently better suited for a greater variety of genres, while the solutions offered by REDengine and Unreal Engine for 3D character-based, narrative open world games are fitting for their mainline games.

CD Projekt RED's switch to Unreal Engine demonstrates another interesting point besides putting emphasis on the importance of grain. Nicoll and Keogh write that the Unity Game Engine's grain is *relational*: it is not only shaped by the engine's key stakeholders and engineers, but also a wider community of plugin-developers, asset pack providers, tutorial writers and so on[113]. This also applies to Unreal Engine, having the associated Unreal Marketplace, the Epic Developer Community Forums and countless tutorials on platforms such as YouTube. REDengine 3's grain on the other hand could be said to be *closed*: there is no platform for choosing and automatically integrating custom content packs, its development is intransparent in that no details about new features are distributed and, maybe most importantly, there exists no official, publicly available graphical toolset to build games for it. I will demonstrate further down how TW3 modders and online platforms are making the engine more relational – independently of CD Projekt RED – by developing and adopting content management systems, tools and common workflows.

A second major tool-related field of optimisation for CD Projekt RED consists of tailoring jobs towards tool workflows, in analogy to and together with tailoring these workflows towards (more efficient) solutions of game development problems. This complex, intertwined process stems from the rise of industrial manufacturing in the 19th century, when "mechanical imitation of manual craft"[114] began to happen to a large extent. It is in this time that the position of the designer as a (product) planner gained its modern form: planning and execution were separated in many areas and the designer took over the planning part, while execution was automated[115]. This connects directly to the notion of planning versus executing a plan as developed in Section 2.1 as a current understanding in design theory. A result of this division is that many discussions regarding the value of products crafted by hand in opposition

[112] "Passionate Creators of Story-Driven Role-Playing Games". CD Projekt RED. (2023). *CD PROJEKT RED - Award-winning creators of story-driven role-playing games.* Online: https://cdprojektred.com/en/ (accessed: 13.03.2023).

[113] cf. Keogh, Brendan; Nicoll, Benjamin. *The Unity Game Engine and the Circuits of Cultural Software*. p. 65

[114] Mareis, Claudia. *Theorien des Designs zur Einführung*. p. 49

[115] ibid. p. 48-49

to those generated by machines came up[116]. Those discussions still continue to enter discourses from time to time, as currently seen in the debate on AI-generated imagery and text.

These debates are connected to highly political issues, like the value of human work under the prospect of increasing automation or human working conditions in technological environments. In this context, CD Projekt RED had and has to decide how to negotiate between its economic interest, the care for its workers, relationships with the players and other studios, publishers etc., and how to determine job positions in relations to the only to some extent fluid givens of the engine they develop with. The job positions at CD Projekt RED, being catered towards specific tasks of the development process, can thus not only tell us something about what is understood as a quest internally, but also about how the company chose to enter into dialogue with its technology.

Quest Designers at CD Projekt RED are tasked with "designing quests and implementing them using a proprietary toolset"[117]. Interestingly, design and execution, being separated in many areas during the course of industrialisation to increase efficiency, can be found here joined in a human worker's job, handing not all of the execution to the tool. This is a hint that the execution of quest designs is not fully automatable (yet) or that automation is not desired. Indeed, one major argument for human work in video games and many other fields is that "handcrafted content" has an inherent value in which consumers can trust. TW3, for instance, was often praised for its attention to detail in environment art, cinematic design and writing, with some saying this was only possible only due to the "human touch"[118] by game developers. Whether this is the case or not is not the subject of this text, but it renders the interesting point that CD Projekt RED might have intentionally placed the boundary between human and machine work at the position that it is, to allow for more granular design.

While quest designers "create the storyline on a detailed level"[119], the *writers / story team* is mostly concerned with the narrative's "macro level"[120] and writing dialogue lines[121]. Cinematic, Level, Gameplay, Sound, Character, Lighting, VFX Artists and Designers (to enumerate some of the relevant parties) create further important

[116] ibid. p. 54

[117] CD Projekt RED. *Quest Designer | SmartRecruiters.*

[118] Mello-Klein, Cody. (2017). *The power of handcrafted visual design in video games – storybench*. Online: https://www.storybench.org/the-power-of-handcrafted-visual-design-in-video-games/ (accessed: 16.03.2023).

[119] ibid.

[120] Tomaszkiewicz, Mateusz. *The devil is in the details.*

[121] CD Projekt RED. (2023). Pisarka/Pisarz | SmartRecruiters. Online: https://jobs.smartrecruiters.com/CDPROJEKTRED/743999704525474-writer (accessed: 16.03.2023).

elements needed for a quest, and in the end it is the quest designer who is responsible for “integrating delivered assets”[122]. The number of different assets to integrate and the potential complexity inherent in branching storylines further justifies the quest designer position and motivates quests once more as an interesting subject for analysis.

Having established the economic and creative context from which REDengine 3 stems, I will now turn to the other side: the players, tinkerers and programmers who “work” on the game in their own fashion and with the tools they created.

3.2.2 Modding Scenes

There is a simple reason for why modding is the point of entry to REDengine 3’s object language. The most accurate result could be gained if I had access to the engine and tool developer’s thoughts on the design of REDengine 3. But interviewing studio members would require a whole other level of study where I had to place a lot of my efforts elsewhere. Moreover, since the game’s original release and development started more than seven years ago, finding developers with the information I need might be hard too, since they may have left the company. So given that I do not have access to the studio’s interpretation of the engine’s language, I will use the options that are left to players, more specifically those, who are concerned with the technicality of the game: modders.
Modding, according to Tanja Sihvonen in *Players Unleashed!*, is the following player activity:

> »Modding refers to various ways of extending and altering officially released computer games, their graphics, sounds and characters, with custom- produced content. Modding can also mean creating new game mechanics and new gameplay levels (maps) to the point where the original game transforms into a completely new title.«[123]

For TW3 modding, this statement can be confirmed. We will find that the two main modes of modding declared by Sihvonen, extending and altering, can be mapped directly to two major tool sets developed for interacting with TW3’s game files. The target of these activities is the video game TW3, which moreover always serves as a benchmark for what modders want to achieve. Creators of modding tools have put a lot of effort into making various content types like shaders, foliage (graphics),

[122] CD Projekt RED. *Quest Designer | SmartRecruiters.*
[123] Sihvonen, Tanja. *Players Unleashed! Modding The Sims and the Culture of Gaming*. p. 12

speech (sound) or meshes and entities (characters) accessible, indicating a strive towards an as extensive possibility space for modders as possible. The most advanced of these tools allows users to create new levels, while the freely-editable scripting codebase would indeed allow for new gameplay to some degree.

If we want to use this point of entry, a glimpse into the world of modding communities seems sensible in order to understand what contextual associations come with it. According to Sihvonen, a *modding scene* is the collection of modding efforts manifest in the internet:

> »A 'modding scene' refers to the collaborative internet networks that players use to share the resources for modding (tools, programmes, tutorials, FAQs and general help) and their creations (mods) with other enthusiasts.«[124]

While I concur that the resulting network of data found online on TW3 modding is part of its modding scene, I will also regard the people, their interactions and expressions as constitutive components of modding scenes.

The TW3 modding scene happens largely on two (*Wolven Workshop* and *w3 radish tools*) among several other discord servers, the official TW3 modding forums[125] by CD Projekt RED and the game's Nexus pages[126]. While discord servers and forums are mainly used for the social part of modding as well as sharing progress and asking questions, the tools and creations can be found on Nexus Mods or online repositories and many tutorials or showcases on YouTube.
Black Tree Gaming Ltd. is a company whose main product is the *Nexus Mods* page. The website's main purpose is to provide users the ability to up- and download mods, modding tools and other media for a variety of games. For TW3, ~ 5.800 mod pages exist, and with over 120.0 million downloads the game ranks as the pages game with the 8th most downloads in total[127], which quite probably makes Nexus Mods the biggest host of TW3 mods. The site also hosts a forum, which, quite similar to the CD Projekt RED forums and some of the general channels in the aforementioned discord servers, serve as a starting point for many modders to ask

[124] ibid.
[125] CD Projekt RED. (2023). *MODS (THE WITCHER 3)* [Category]. *CD Projekt RED Forums*. Online: https://forums.cdprojektred.com/index.php?forums/mods-the-witcher-3.69 (accessed: 16.03.2023).
[126] Black Tree Gaming Ltd. (2023). *The Witcher 3 Nexus - Mods and community.* Online: https://www.nexusmods.com/witcher3 (accessed: 16.03.2023).
[127] https://www.nexusmods.com

questions about modding workflows, post tutorials and put in requests for new and old mods.

The official CD Projekt RED forums are not only a starting point for TW3 modders, but also for those of The Witcher and TW2, which are both represented as a category of posts next to TW3 modding. These forums are interesting for one reason in particular: it is there that the official modding tools were first announced and distributed by the studio, and it is also here where some studio members have accounts and from time to time give answers to the community. For example, over the course of the Next-Gen Update for TW3, studio member "Vinthir" released a statement concerning the status of the official mod kit[128], proving that CD Projekt RED regards the forums as their official contact point to the modding community.

The Discord server landscape of TW3 modding is split along the communities two major tool sets: *WolvenKit*, which is mostly discussed on the *Wolven Workshop*, while the *radish modding tools* have a dedicated server "w3 radish tools". Aside from these two (which are probably the most active), a general server for CD Projekt RED game modding exists ("CDPR Modding Community & The Boiz"), mostly occupied with Cyberpunk 2077 nowadays, an official TW3 server from CD Projekt RED, where mainly general fan and mod installation discussion happens, and "The Witcher 3: Modding Community", a very recent split from Wolven Workshop, meant to be focused on tool creation.

Server	**Purpose**
Wolven Workshop	"The Witcher 3 Modmaking Platform", general-purpose TW3 modding and development of the WolvenKit editor.
w3 radish tools	Learning, developing and researching for the TW3 radish modding tools.
CDPR Modding Community & The Boiz	General-purpose modding of CD Projekt RED games.
The Witcher	General The Witcher-related topics.
The Witcher 3: Modding Community	TW3 modding tool creation.

Table 3: Listing of discord servers where TW3 modding discussions happen.

[128] Vinthir; CD Projekt RED. (2022). *Mods and The Witcher 3 next-gen update* [Thread]. *CD Projekt RED Forums*. Online: https://forums.cdprojektred.com/index.php?threads/mods-and-the-witcher-3-next-gen-update.11110486/ (accessed: 16.03.2023).

3.2.3 Asymmetries in Modder-Studio Relations

In these communication platforms and in the modding tools, time and time again the entanglement of TW3 modders with the company CD Projekt RED can be felt. The key point in this relationship centres around the modders trying to tinker and create as much as possible with REDengine 3 and TW3, and on the other hand the company providing to some degree access to and information on their technology. Sihvonen explains how the status of games like TW3 as items that are partially open for modification indicates a power asymmetry:

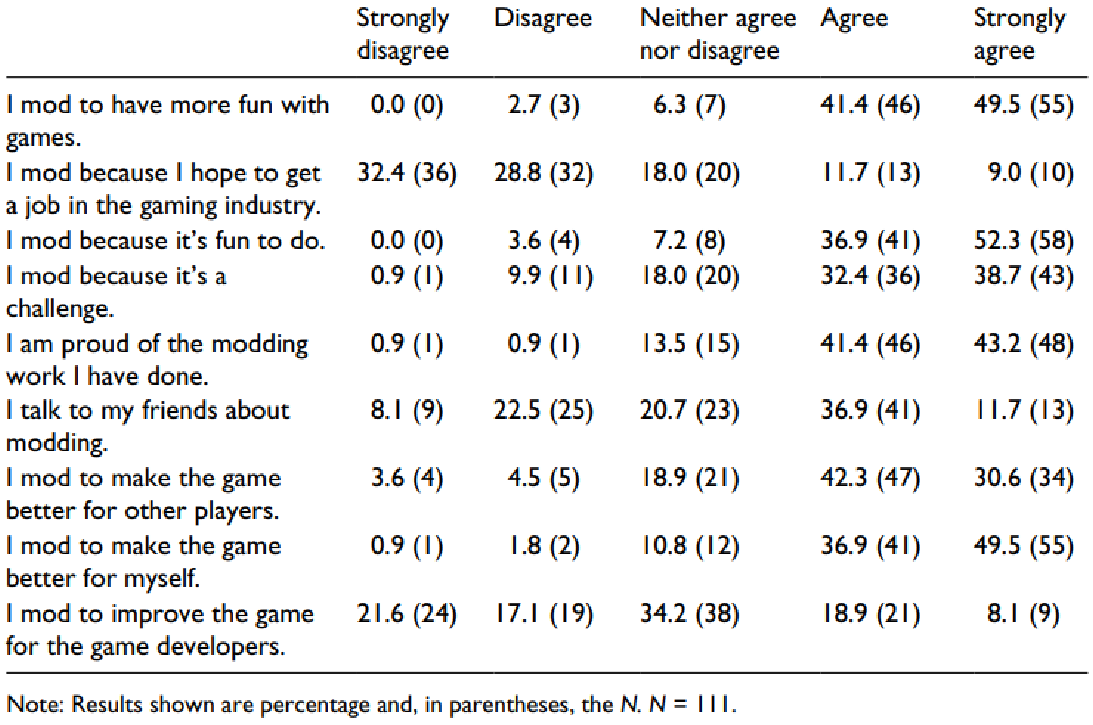
> »the realisation of these propositions depends on the developers' intentions, incorporated in the game code, in comparison to the reconfigurative power allocated to the player«[129]

Sihvonen names propositions and "ideological inclinations"[130] as game aspects whose modability is restricted by the power granted from developers, but we can generalise and say that not only those, but any aspect coupled to a game's technical reality is dependent on "the reconfigurative power allocated to the player"[131]. And since this allocation of power is dependent on "developer's intentions", it makes sense to take a look at the motivations of TW3 modders and CD Projekt RED regarding modding and compare.

There are a multitude of reasons why someone might be motivated to start modding a game. The creator of the radish modding tools, rmemr, lists among else the following motivations[132]: some want to enjoy the game more and thus decide to tweak the game to make it such that it is better for them. On NexusMods, for instance, a variety of so-called "Re-Shade Presets"[133] exist, which are concerned with high-level visual adjustments only. Closely related is the drive to fix discovered bugs, including bugs which may be regarded as bugs only by some people (e.g. the difficulty of an enemy). For some, enjoyment can be found in the ability to express themselves with a given feature-rich tool set and coherent asset collection. For example, the ability to add new sword meshes to TW3 and integrate them

[129] Sihvonen, Tanja. *Players Unleashed! Modding The Sims and the Culture of Gaming*. p. 34
[130] ibid.
[131] ibid.
[132] rmemr. (2015). [Post]. *CD Projekt RED Forums*. Online: https://forums.cdprojektred.com/index.php?threads/im-just-curious-about-mods-newbie-alert.62410/#post-2109367 (accessed: 16.03.2023).
[133] Black Tree Gaming Ltd. (2023). *The Witcher 3 mod categories at The Witcher 3 Nexus - Mods and community*. Online: https://www.nexusmods.com/witcher3/mods/categories (accessed: 16.03.2023).

seamlessly in the in-game economy. Rmemr moreover notes that some simply find it fun to be tinkering with and exploring a game's technology, maybe even being motivated by the inherent challenge to "decrypt things that were not meant to be changed", which might refer to their own work on figuring out some of the low-level formats like W3STRINGS of REDengine 3. This subjective list overlaps to large parts with a scientific survey conducted by Nathaniel Poor[134]:



	Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
I mod to have more fun with games.	0.0 (0)	2.7 (3)	6.3 (7)	41.4 (46)	49.5 (55)
I mod because I hope to get a job in the gaming industry.	32.4 (36)	28.8 (32)	18.0 (20)	11.7 (13)	9.0 (10)
I mod because it's fun to do.	0.0 (0)	3.6 (4)	7.2 (8)	36.9 (41)	52.3 (58)
I mod because it's a challenge.	0.9 (1)	9.9 (11)	18.0 (20)	32.4 (36)	38.7 (43)
I am proud of the modding work I have done.	0.9 (1)	0.9 (1)	13.5 (15)	41.4 (46)	43.2 (48)
I talk to my friends about modding.	8.1 (9)	22.5 (25)	20.7 (23)	36.9 (41)	11.7 (13)
I mod to make the game better for other players.	3.6 (4)	4.5 (5)	18.9 (21)	42.3 (47)	30.6 (34)
I mod to make the game better for myself.	0.9 (1)	1.8 (2)	10.8 (12)	36.9 (41)	49.5 (55)
I mod to improve the game for the game developers.	21.6 (24)	17.1 (19)	34.2 (38)	18.9 (21)	8.1 (9)

Note: Results shown are percentage and, in parentheses, the *N*. *N* = 111.

Table 4: Modders' motivations according to Pool (2013).

The parts rmemr missed out on in comparison to the survey are that modding can be a pathway into the games industry, a motivation reassured by various news about modders (e.g. the developers of WolvenKit) being hired by CD Projekt RED, and that many modders have an inherent motivation to do their work also for the community (although rmemr might have assumed this to be trivially true).

Compared to modders, the motivation for CD Projekt RED to tolerate and to some degree encourage and support modding activities for their games is not entirely mirrored. While game development studios might be motivated by the gains for modders and the player community like those discussed above, they will also be motivated by advantages for themselves, which leads in some points to agendas incompatible to those of modders. In "Precarious Playbour: Modders and the Digital

[134] Poor, Nathaniel. (2013). *Computer game modders' motivations and sense of community: A mixed-methods approach.* In: (2014). *New Media & Society*. vol. 16. no. 8. p. 1257

Games Industry", Julian Kücklich analyses modder-studio relations and brings up five benefits game developers could gain from a large modding community[135].

As a sample point, Kücklich mentions how highly successful mods such as *Counter-Strike* for Valve's *Half Life 2* can provide the developer with new brands or innovation without the studios needing to invest effort into marketing or development. While no mod for a CD Projekt RED game has reached this kind of success, there are projects which have established minor brands and were recognised by the company, like "Rise of the White Wolf", "The Witcher: Farewell of the White Wolf", HalkHogan's "The Witcher 3 HD Reworked Project"[136] or the WolvenKit tool set, whose Cyberpunk 2077 edition was made an officially recommended modding tool[137]. In an unprecedented case for CD Projekt RED, they even included a selection of mods, including HalkHogan's, in the so-called "Next-Gen Update" or patch 4.0 for TW3. This is one angle, where the issue of *playbour* could be raised against the studio, that is, "the commodification of modders' leisure"[138] – if not for the fact that the creators of these integrated mods were actually paid and credited by the company[139]. On the other hand one of the more known TW3 modders named wghost81 criticises in an open letter to CD Projekt RED that said update also includes changes which are similar to various uncredited mods, claiming that some were directly included and rewritten for confusion of observers[140][141]. Without an aspiration to give a final answer to whether or not this is true (as I am neither legal expert nor in knowledge of the relevant mod and game source code) I suspect these claims not to be true. I agree with Aeltoth, another known TW3 modder, that the evidence provided is not tenable and that they should "at least use tools that are made to detect copy/ pasted code"[142]. Regardless

[135] Kücklich, Julian. (2005). *Precarious Playbour: Modders and the Digital Games Industry*. In: Neilson, Brett; Rossiter, Ned (eds.). (2005). *The Fibreculture Journal*. vol. 5.

[136] CD Projekt RED. (2021). *Highlights from the Path: Mods.* Online: https://www.thewitcher.com/en/news/38447/highlights-from-the-path-mods (accessed: 16.03.2023).

[137] CD Projekt RED. (n.d.). *REDmod. Cyberpunk 2077.* Online: https://www.cyberpunk.net/en/modding-support (accessed: 16.03.2023). See section "What is REDmod".

[138] cf. Kücklich, Julian. *Precarious Playbour: Modders and the Digital Games Industry*.

[139] Vinthir; CD Projekt RED. *Mods and The Witcher 3 next-gen update | Forums - CD PROJEKT RED.*

[140] wghost81. (2023). *Time to wake up, samurai – Old Ghost Stories.* Online: https://wghost81.wordpress.com/2023/01/17/time-to-wake-up-samurai/ (accessed: 16.03.2023).

[141] wghost81. (2023). *CDPR correspondence on community mods in TW3 NGE – Old Ghost Stories.* Online: https://wghost81.wordpress.com/2023/01/19/cdpr-correspondence-on-community-mods-in-tw3-nge/ (accessed: 16.03.2023).

[142] Aeltoth. (2023). "[..] with complaints like that, at least use tools that are made to detect copy/pasted code. They're able to detect it even if functions and variables are renamed and moved around. So unprofessional once again, like who compares code in an image with no label (se we don't know which side is which) and with colored boxes smh. [..]" [Message].

of the solution, this incident demonstrates how the motivations and actions of modders and studios can become entangled in both positive and negative ways.

A more widespread issue modders have with CD Projekt RED is the (in their opinion) insufficient degree to which the company provides them with the means to do modding. As we will see later, the TW3 modding scene had to base all their efforts on a single command-line tool, lacking a thorough documentation, any graphical user interface and the ability to decode some of the more interesting formats. The potential for frustration is increased by several other factors.
For one, there are the inherent difficulties of game development/ modding, like the need to direct a project such that the result is aesthetically coherent, or finding ways to circumvent the lack of monetary budget (e.g. for voice acting). Secondly, the level of frustration might be affected by the persisting legal grey zone in which modding communities still operate. The enduring dissonance can be observed in legal documentation. In their official Fan Content Guideline, CD Projekt RED states clearly that they endorse the creation of mods:

> »We're happy for you to make mods for our games (i.e. software that modifies or works with our games – e.g. changing the UI or adding new mechanics) so long as it doesn't breach the relevant game's EULA [..].«[143]

However, in that document it is noted that the actual legally binding rules are, among else, to be found in the User Agreement. In this agreement, the user is denied to

> »modify, merge, distribute, translate, reverse engineer, or attempt to obtain or use source code of, decompile or disassemble the CD PROJEKT RED games [..]«[144]

The dissonance between the de jure prohibition and de facto permission (with only one case of the company taking down mods[145]) constitutes the grey zone, which might lead to modders feeling that all of the company's endorsements are dimmed

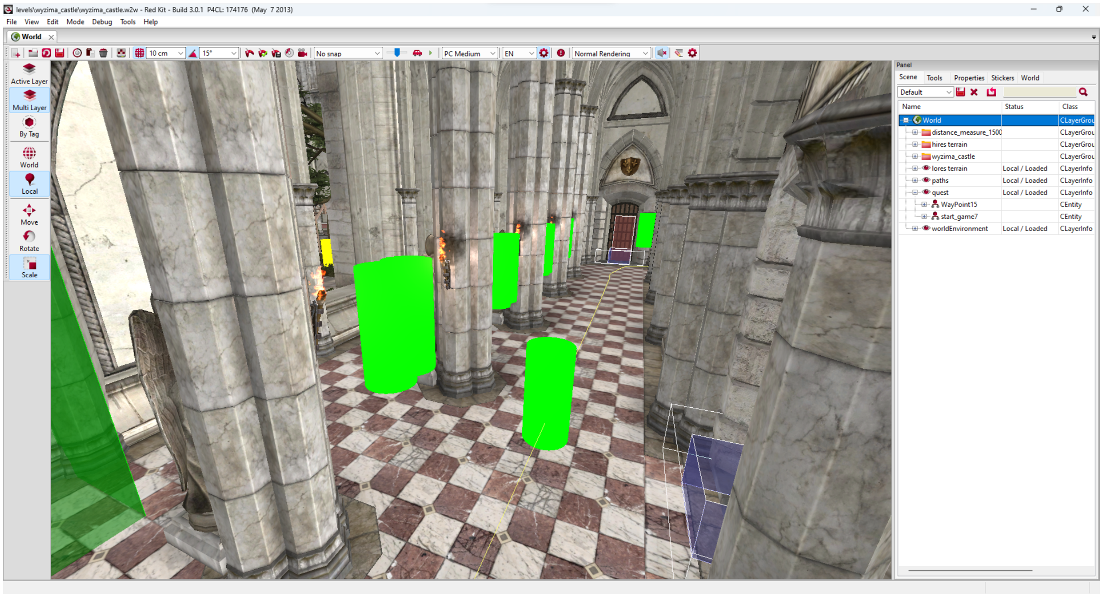
The Witcher. Discord. Online: https://discord.com/channels/597170291021709327/597171985050501150/1071032950826745887 (accessed: 16.03.2023).

[143] CD Projekt RED. (2020). *Fan Content Guideline*. Online: https://www.cdprojektred.com/en/fan-content (accessed: 16.03.2023). Section 3, Paragraph a.c

[144] CD Projekt RED. (2022). *User Agreement*. Online: https://regulations.cdprojektred.com/en/user_agreement (accessed: 16.03.2023). Section 8., Paragraph (d)

[145] Kent, Emma. (2021). *Cyberpunk 2077 Keanu sex mod removed following CD Projekt warning*. Eurogamer. Online: https://www.eurogamer.net/cd-projekt-red-shuts-down-cyberpunk-2077-keanu-sex-mod (accessed: 16.03.2023).

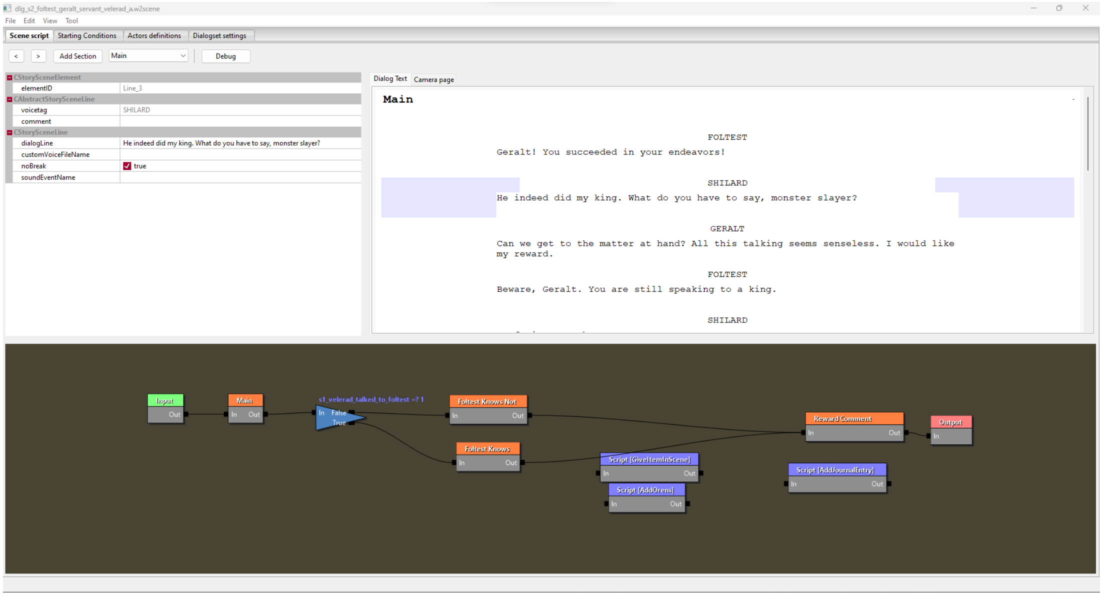
by the conscious choice to not also give modders their legal support. All the while, CD Projekt RED can be counted to have a relatively more modding-friendly EULA, giving modders e.g. full ownership of the mods created with its tools[146], in opposition to e.g. Valve[147].



levels\wyzima_castle\wyzima_castle.w2w - Red Kit - Build 3.0.1 P4CL: 174176 (May 7 2013)

Name	Status	Class
World		CLayerGrou
distance_measure_1500		CLayerGrou
hires terrain		CLayerGrou
wyzima_castle		CLayerGrou
lores terrain	Local / Loaded	CLayerInfo
paths	Local / Loaded	CLayerInfo
quest	Local / Loaded	CLayerInfo
WayPoint15		CEntity
start_game7		CEntity
worldEnvironment	Local / Loaded	CLayerInfo

Illustration 6: Screenshot of the level editor coming with the REDkit.



dlg_s2_foltest_geralt_servant_velerad_a.w2scene

CStoryScenElement	
elementID	Line_3
CAbstractStorySceneLine	
voicetag	SHILARD
comment	
CStorySceneLine	
dialogLine	He indeed did my king. What do you have to say, monster slayer?
customVoiceFileName	
noBreak	☑ true
soundEventName	

Main

FOLTEST
Geralt! You succeeded in your endeavors!

SHILARD
He indeed did my king. What do you have to say, monster slayer?

GERALT
Can we get to the matter at hand? All this talking seems senseless. I would like my reward.

FOLTEST
Beware, Geralt. You are still speaking to a king.

SHILARD

Illustration 7: Screenshot of the scene script editor coming with the REDkit.

[146] Here, for instance, the EULA for the Cyberpunk 2077 modding tool: CD Projekt RED. (2022). *REDmod END USER LICENSE AGREEMENT*. Online: https://cdn-l-cyberpunk.cdprojektred.com/redmod_eula_en.pdf (accessed: 16.03.2023).
[147] Kücklich, Julian. *Precarious Playbour: Modders and the Digital Games Industry.*

As a third and last point, some modders remember the tools CD Projekt RED had given them for previous games. The *REDkit*[148] for TW2 featured a full-scale graphical user interface and specialised editors to create and modify levels, quests, scenes, general entities and more, all of which is not trivially possible with the official TW3 modding tools.

Some might sense a case of double standards here, with the company openly declaring their support of modding and even highlighting modders' works but not providing them with the "proper" tools[149].

Given that the official tools are only light enablers that do not "pre-empt the most common workflows through which users [..] create, edit, and iterate upon content"[150] (or rather: would like to), and given that one of modders' motivations is to tinker with and learn technology, it is no surprise that the TW3 modding community started creating its own tools to expand their scope of possibilities. This can be regarded as a process of self-empowerment, as a strive for more autonomy within the system that was established at first. Eliminating the constraints caused by CD Projekt RED, but also using them to create high-quality content, can moreover be viewed as a move of self-affirmation and self-assertion in the face of a company that is regarded by many to be flawed. The feeling of having the power to do something and empower oneself or one's community is consistent with Nathaniel Poor's finding that modders are, in general, proud of their work: over 80% of the modders they interviewed *agreed* or *strongly agreed* to the statement "I am proud of the modding work I have done" whereas less than 2% *disagreed* or *strongly disagreed*.

We now have an understanding of modding as a way to access REDengine 3's language and as a demonstration of grain at work and reason for its relaxation.

3.3 From Command-Line to Graph Editor

This section looks at technical consequences originating from the modding context, prepares for the examination in Chapter 4 and takes a closer look at the specific workflows and tools of TW3 modding.

[148] CD Projekt RED. (2011). *REDkit* (3.0.1).

[149] Twitter, Inc. (2023). *mod (from:witchergame) - Twitter Search / Twitter*. Online: https://twitter.com/search?q=mod%20(from%3Awitchergame)&src=typed_query (accessed: 16.03.2023).

[150] Keogh, Brendan; Nicoll, Benjamin. *The Unity Game Engine and the Circuits of Cultural Software*. p. 63

3.3.1 Basics of The Witcher 3 Modding

The basis of any modding activity are the game files. TW3 has a lot of file types in its installation and even more internal ones that are packed in compression bundles. One type is interesting for modding and is human-readable from the start: the game's gameplay scripts. On top of C++ for low-level engine work, CD Projekt RED developed a higher-level scripting language named *REDscript*, which uses features and systems provided by the C++ - code and allows gameplay programmers to work with the advantages of scripting languages, such as: runtime interpretation, relatively simple syntax and semantics and automated memory management. The code written in this language can be found in "[W3]\content\content0\scripts" and accessed freely. I will come back to this topic in Section 4.3.

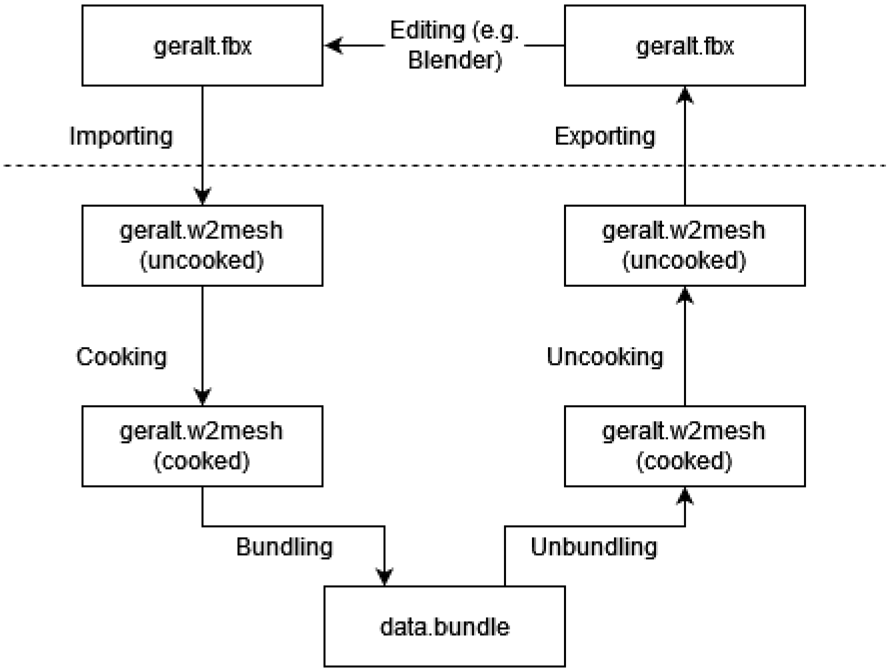


Illustration 8: Graphic demonstrating the import/ cook/ bundle process for the character mesh of Geralt. For more information, see[151].

Any more advanced TW3 modding starts with a command line tool that is named *wcc_lite.exe*[152]. This application, released on 14. August 2015[153], allows modders to (un)bundle, (un)cook as well as im- and export a great variety of game files (compare Illustration 8). *Bundles* are the REDengine 3 equivalents of ZIP files (or

[151] https://wiki.redmodding.org/legacy-wolvenkit/guides/github-guides/project-structure
[152] CD Projekt RED. (2015). *Witcher 3 Mod Tools* (3.0).
[153] Momot, Marcin; CD Projekt RED. (2018). *Getting Started / FAQ (UPDATED 21.11.2018)*. [Thread]. *CD Projekt RED Forums*. Online: https://forums.cdprojektred.com/index.php?threads/getting-started-faq-updated-21-11-2018.56132/ (accessed: 16.03.2023).

anything similar): containers for compressing multiple files and directories to smaller size[154]. Bundles are also the files which end up in DLC- and mod packages that are placed in the game's `dlc` and `mods` directories. *Cooking* refers to the process of compressing single files (e.g. textures with raw pixel data) to a smaller size. *Im*- and *export* refer to transforming data from external formats such as FBX for 3D models to their internal (uncooked) counterpart such as W2MESH and back.
In addition to scripts, wcc_lite allowed the modding community to unlock editing lots of XML and CSV files (containing mostly gameplay data), Textures and Meshes.

However, the TW3 modding scene still had an initial major problem which was very much related to this process: while all files could be (un)bundled and (un)cooked, many of them could not be im-/ exported and thus not edited with any external program. Even worse, most contents of the game's main format, CR2W (Witcher 2 Resource Class) were part of this subset. CR2W's for instance encode quests, characters, journal entries, interactive entities, NPC behaviours, (cut-)scenes and many more. It makes sense that there is no import/ export for them, since there exist no standardised formats for entities like quests anyway, but this made life a lot more difficult for those wanting to tinker with these files. Compared to the extensive graphical editors for quests, communities, entities etc. in The Witcher's *D'jinni Adventure Editor* or TW2's REDkit, many modders were disappointed that the TW3 Modding Tools only entailed a command line tool.

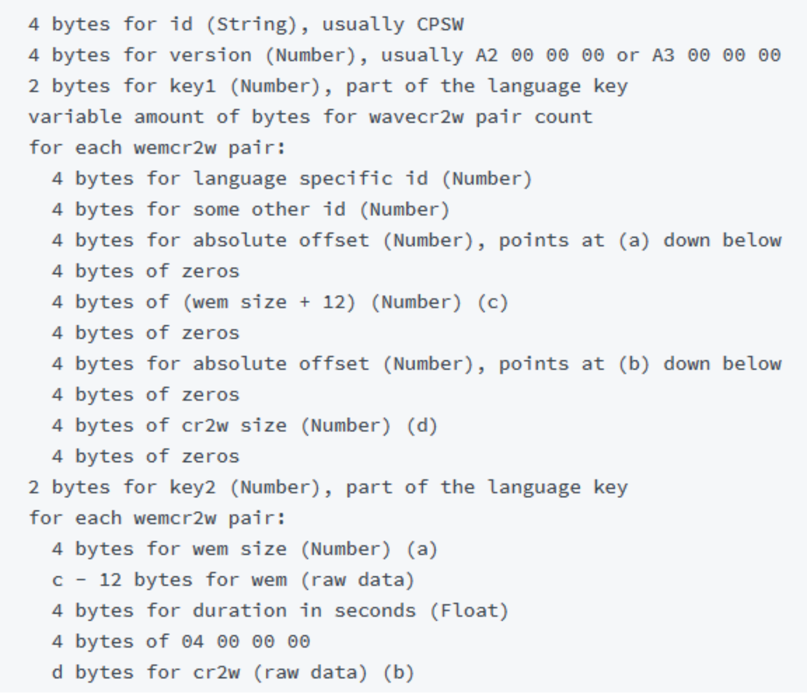
3.3.2 Community-Made Tools

As mentioned, consequently some began working on their own tools. These range from a simple graphical user interface for wcc_lite (Mod Kitchen[155]) to a first encoder for strings (w3strings encoder[156]) to the first attempts at a multi-tool for reading and editing CR2W's (Sarcen's Mod Editor[157]). Both strings encoder and Mod Editor were not able to rely on the im- and export functions and thus needed new code that would transform data that could be handled by human users (e.g. in a human-readable format like CSV or in a graphical front-end) to the custom uncooked formats of REDengine 3. Since initially no one knew how those formats

[154] Bundles can be found in `[W3]\content\content0\bundles`, if `[W3]` is the TW3 installation directory.
[155] https://www.nexusmods.com/witcher3/mods/389/
[156] https://forums.cdprojektred.com/index.php?threads/utility-strings-encoder-for-adding-new-strings-new-ids-and-keys-as-standalone-w3strings-file.62959/
[157] https://forums.cdprojektred.com/index.php?threads/mod-editor.58758/

were structured, modders needed to reverse engineer encoding/ decoding protocols from scratch.

As an example, audio data for dialogue lines was saved in an unknown format called W3SPEECH and wcc_lite did not provide any import function, meaning that in the end adding new dialogue was limited to text only. It was only when it was figured out that W3SPEECH files had the following structure, that someone could write a program (an *encoder*) to transform WEM audio to uncooked REDengine 3 data:



```
4 bytes for id (String), usually CPSW
4 bytes for version (Number), usually A2 00 00 00 or A3 00 00 00
2 bytes for key1 (Number), part of the language key
variable amount of bytes for wavecr2w pair count
for each wemcr2w pair:
  4 bytes for language specific id (Number)
  4 bytes for some other id (Number)
  4 bytes for absolute offset (Number), points at (a) down below
  4 bytes of zeros
  4 bytes of (wem size + 12) (Number) (c)
  4 bytes of zeros
  4 bytes for absolute offset (Number), points at (b) down below
  4 bytes of zeros
  4 bytes of cr2w size (Number) (d)
  4 bytes of zeros
2 bytes for key2 (Number), part of the language key
for each wemcr2w pair:
  4 bytes for wem size (Number) (a)
  c - 12 bytes for wem (raw data)
  4 bytes for duration in seconds (Float)
  4 bytes of 04 00 00 00
  d bytes for cr2w (raw data) (b)
```

Illustration 9: Graphic demonstrating how WEM audio data is encoded in the W3SPEECH format[158]*.*

Figuring that out solves just one part of the problem, though. In order for users to utilise the added power that is given by such an encoder (in this case: adding new speech audio to the game), some kind of user interface for inputting the data is still required.

And this is where the two major tool sets developed by the community come into play: besides adding encoders/ decoders for all major formats where none exists in wcc_lite, they add a more or less human-readable representation of that data. The first tool, *WolvenKit*[159], is built upon Sarcen's Mod Editor and provides functionality to read, display and edit uncooked files from the engine. The second and much later

[158] REDengine 3 research contributors. (2022). *W3Speech file. Generic - REDengine3 research.* Online: https://wiki.redmodding.org/redengine3-research/formats/generic#w3speech-file (accessed: 16.03.2023).

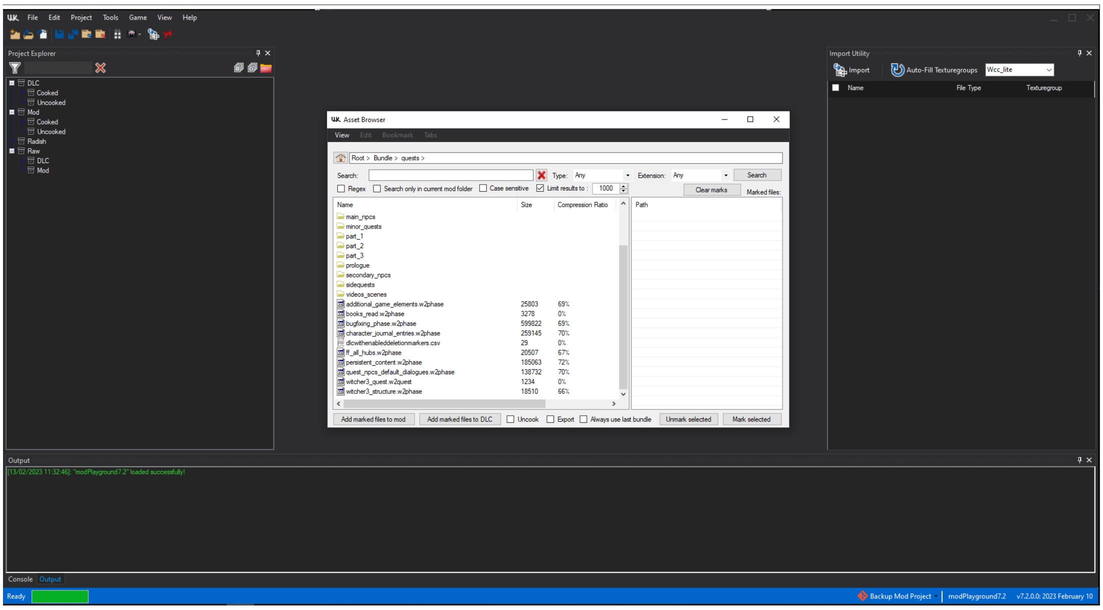
[159] WolvenKit contributors. (2023). *WolvenKit/WolvenKit-7: WolvenKit for Witcher 3* (7.2.0). https://github.com/WolvenKit/WolvenKit-7.

released tool set is named *radish modding tools* and allows users to create new uncooked files with a range of textual and some graphical input representations. Both toolsets entail various scripts to automate their respective workflow, that is:

WolvenKit: decoding -> editing (UI) -> encoding -> cooking -> bundling
radish tools: editing (UI, YAML) -> encoding -> cooking -> bundling

The basis of both tools is that the majority of game files have been unbundled and uncooked.

I will now provide a short introduction to the interface of these created tools. WolvenKit can be characterised as an asset editor. After creating a project, users are able to open up the Asset Browser or the general CR2W file-picker to select an uncooked asset they want to edit (Illustration 10).



Root > Bundle > quests >

Search: Type: Any Extension: Any Search

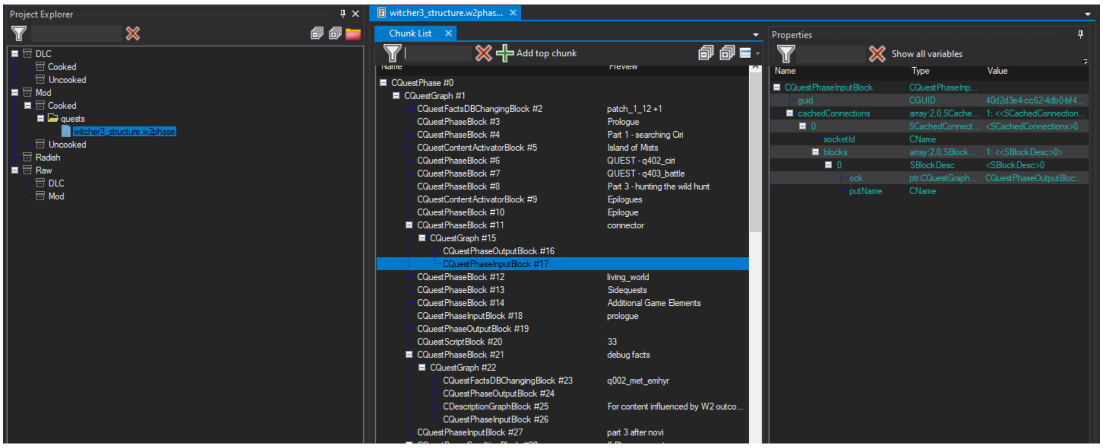
□ Regex □ Search only in current mod folder □ Case sensitive ☑ Limit results to : 1000

Name	Size	Compression Ratio
main_npcs		
minor_quests		
part_1		
part_2		
part_3		
prologue		
secondary_npcs		
sidequests		
videos_scenes		
additional_game_elements.w2phase	25803	69%
books_read.w2phase	3278	0%
bugfixing_phase.w2phase	599822	69%
character_journal_entries.w2phase	259145	70%
dlcwithenableddeletionmarkers.csv	29	0%
ff_all_hubs.w2phase	20507	67%
persistent_content.w2phase	185063	72%
quest_npcs_default_dialogues.w2phase	138732	70%
witcher3_quest.w2quest	1234	0%
witcher3_structure.w2phase	18510	66%

□ Uncook □ Export □ Always use last bundle

```
[13/02/2023 11:32:46]: "modPlayground7.2" loaded successfully!
```

Illustration 10: Screenshot showing WolvenKit *and its asset browser, which allows searching through all the game files that have been unbundled and uncooked to a certain location.*

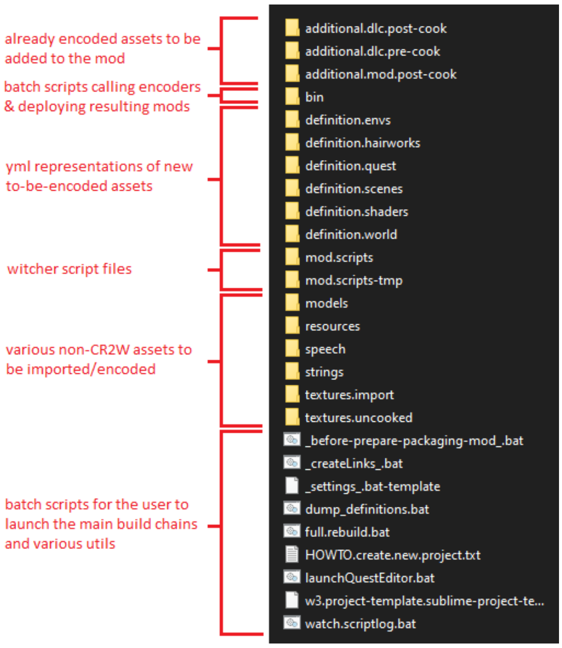


Name	Preview
CQuestPhase #0	
CQuestGraph #1	
CQuestFactsDBChangingBlock #2	patch_1_12 +1
CQuestPhaseBlock #3	Prologue
CQuestPhaseBlock #4	Part 1 - searching Ciri
CQuestContentActivatorBlock #5	Island of Mists
CQuestPhaseBlock #6	QUEST - q402_ciri
CQuestPhaseBlock #7	QUEST - q403_battle
CQuestPhaseBlock #8	Part 3 - hunting the wild hunt
CQuestContentActivatorBlock #9	Epilogues
CQuestPhaseBlock #10	Epilogue
CQuestPhaseBlock #11	connector
CQuestGraph #15	
CQuestPhaseOutputBlock #16	
CQuestPhaseInputBlock #17	
CQuestPhaseBlock #12	living_world
CQuestPhaseBlock #13	Sidequests
CQuestPhaseBlock #14	Additional Game Elements
CQuestPhaseInputBlock #18	prologue
CQuestPhaseOutputBlock #19	
CQuestScriptBlock #20	33
CQuestPhaseBlock #21	debug facts
CQuestGraph #22	
CQuestFactsDBChangingBlock #23	q002_met_emhyr
CQuestPhaseOutputBlock #24	
CDescriptionGraphBlock #25	For content influenced by W2 outco...
CQuestPhaseInputBlock #26	
CQuestPhaseInputBlock #27	part 3 after novi

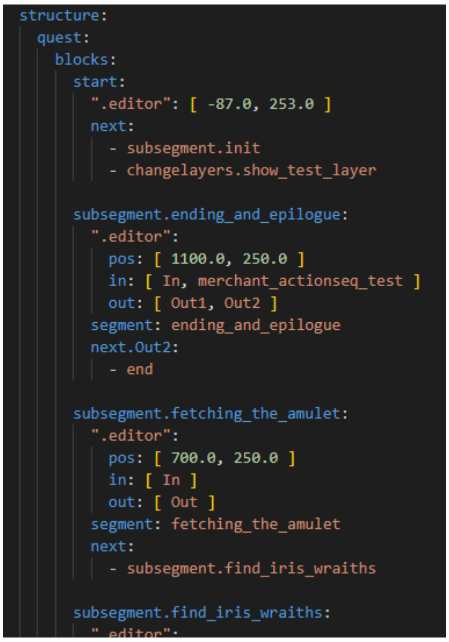
Name	Type	Value
CQuestPhaseInputBlock	CQuestPhaseInp...	
guid	CGUID	40d2d3e4-cc62-4db0-bf4...
cachedConnections	array:2,0,SCache...	1: <<SCachedConnection...
0	SCachedConnect...	<SCachedConnections>0
socketId	CName	
blocks	array:2,0,SBlock...	1: <<SBlockDesc>0>
0	SBlockDesc	<SBlockDesc>0
ock	ptr:CQuestGraph...	CQuestPhaseOutputBloc...
putName	CName	

Illustration 11: Screenshot showing the project hierarchy as well as the decoded chunks and properties of a CR2W file (in this case the top-level main quest structure) in WolvenKit.

Selecting an asset for modification/ inspection then adds the asset to the project hierarchy, where it can be opened – it is in this step that the asset is decoded to a hierarchy of *chunks*. Each chunk has various *properties* (see Illustration 11).
WolvenKit, by means of this generic CR2W window, thus allows users to inspect and modify any quest, behaviour, scene or world content arrangement. However, doing so can become quite complex – especially for content that allows for heavily interconnected chunks, such as quests, where a block of functionality can have any number of successors and predecessors. The biggest drawback of WolvenKit, then, is that there are no specialised windows for working e.g. with the connections within a quest graph, the arrangement of elements on a scene timeline or placing entities in the game world.
In this regard the radish modding tools can shine. While they do not enable users to conveniently view/ edit existing assets, they do facilitate some much user-friendlier workflows for content creation. Instead of an editor integrating all other sub-editors and commands, the radish modding tools' main access point is a set of batch scripts and YAML files contained in a standard project template, referencing the encoder executables lying in their installation directory (see Illustration 12). Unlike WolvenKit, in order to inspect build chain output (which delivers valuable information in case of errors) it is sensible to use a command line for starting the batch scripts which highlight errors and warnings there. Akin to WolvenKit, the radish modding tools allow for the import of various assets which wcc_lite supports by default.


already encoded assets to be added to the mod
batch scripts calling encoders & deploying resulting mods
yml representations of new to-be-encoded assets
witcher script files
various non-CR2W assets to be imported/encoded
batch scripts for the user to launch the main build chains and various utils

```
additional.dlc.post-cook
additional.dlc.pre-cook
additional.mod.post-cook
bin
definition.envs
definition.hairworks
definition.quest
definition.scenes
definition.shaders
definition.world
mod.scripts
mod.scripts-tmp
models
resources
speech
strings
textures.import
textures.uncooked
_before-prepare-packaging-mod_.bat
_createLinks_.bat
_settings_.bat-template
dump_definitions.bat
full.rebuild.bat
HOWTO.create.new.project.txt
launchQuestEditor.bat
w3.project-template.sublime-project-te...
watch.scriptlog.bat
```



```
structure:
  quest:
    blocks:
      start:
        ".editor": [ -87.0, 253.0 ]
        next:
          - subsegment.init
          - changelayers.show_test_layer

      subsegment.ending_and_epilogue:
        ".editor":
          pos: [ 1100.0, 250.0 ]
          in: [ In, merchant_actionseq_test ]
          out: [ Out1, Out2 ]
        segment: ending_and_epilogue
        next.Out2:
          - end

      subsegment.fetching_the_amulet:
        ".editor":
          pos: [ 700.0, 250.0 ]
          in: [ In ]
          out: [ Out ]
        segment: fetching_the_amulet
        next:
          - subsegment.find_iris_wraiths

      subsegment.find_iris_wraiths:
        ".editor":
```

Illustration 12 (left): Screenshot showing the project template of the radish modding tools and elaborating on how various kinds of data and commands have their designated places. Illustration 13 (right): Screenshot showing a sample YAML quest definition.

The `definition.*` directories are reserved for the YAML representations of CR2W-based assets, such as scenes or environments. YAML is a file format standard for human- and machine-readable data. These files are the input for the encoders which translate them to CR2W-assets.

There are two graphical editors in the tool set: a quest editor, allowing to change the YAML quest definition more easily (compare Illustration 13 and 14), and a phoneme editor to map dialog strings and speech to phonemes from which lip sync animations are generated (Illustration 15). These two editors are one of the major advantages to the radish modding tools.

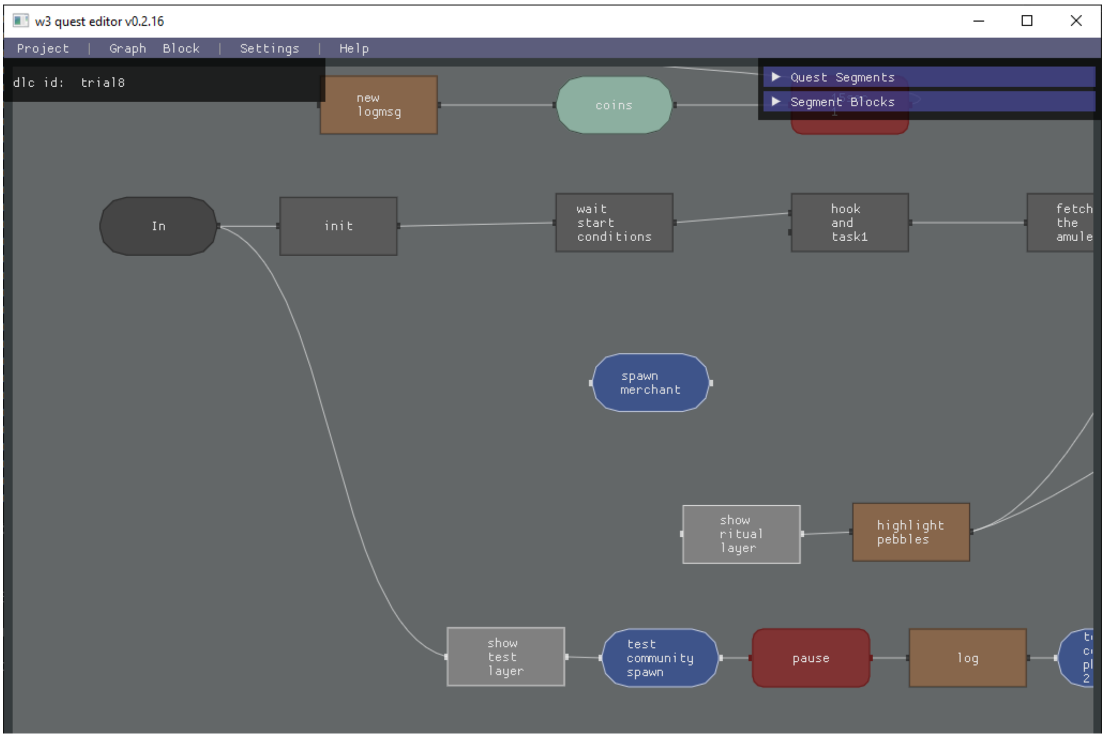
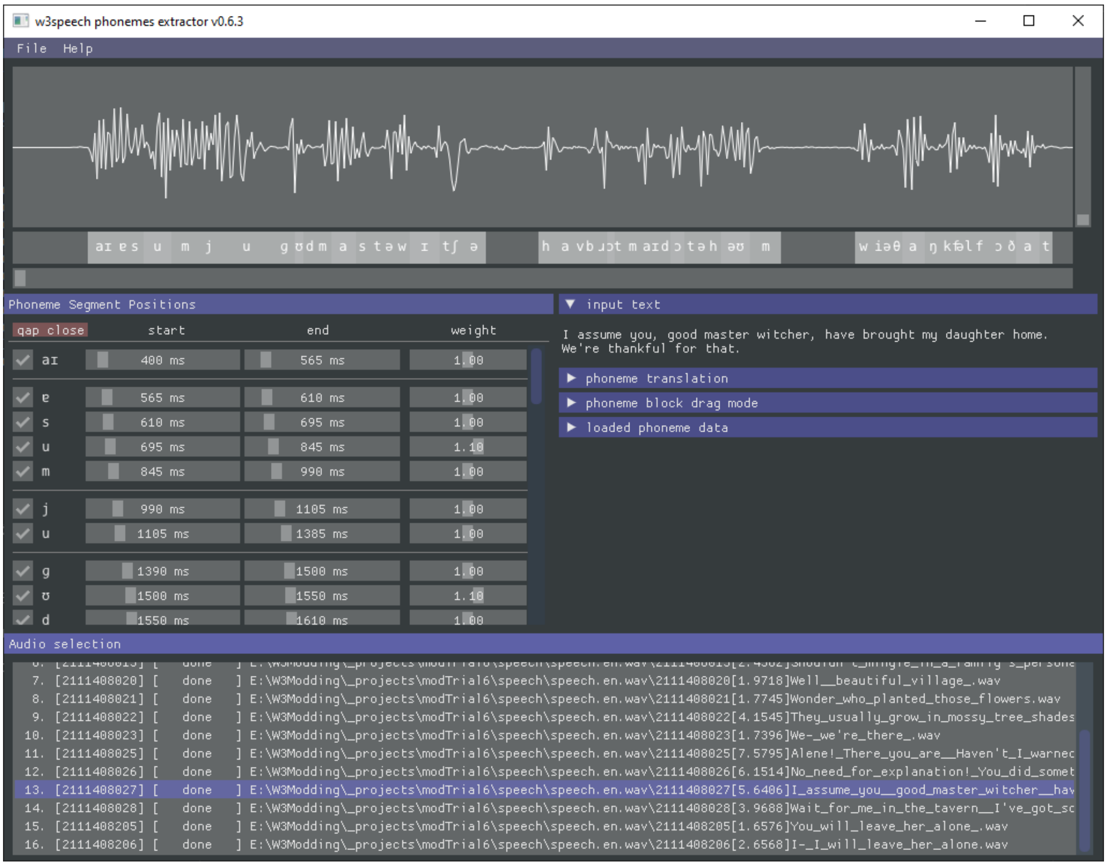


Illustration 14: Screenshot showing the quest editor of the radish modding tools with a loaded YAML quest definition (the one displayed in Illustration 13).


w3speech phonemes extractor v0.6.3
File Help
aɪ e s u m j u g ʊ d m a s t ə w ɪ tʃ ə
h a v b ɹ ɔ t m a ɪ d ɔ t ə h əʊ m
w i ə θ a ŋ k f ə l f ɔ ð a t
Phoneme Segment Positions
gap close
input text
phoneme translation
phoneme block drag mode
loaded phoneme data
Audio selection

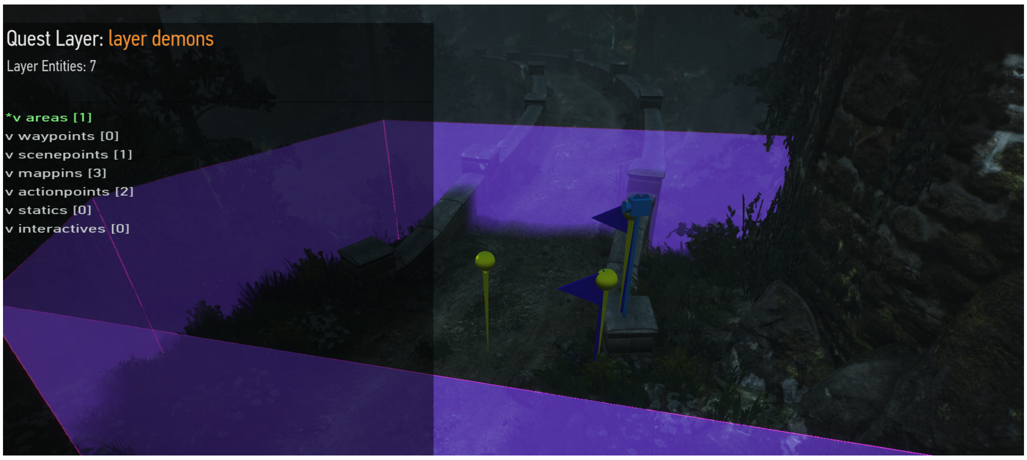
	start	end	weight
✓ aɪ	400 ms	565 ms	1.00
✓ e	565 ms	610 ms	1.00
✓ s	610 ms	695 ms	1.00
✓ u	695 ms	845 ms	1.10
✓ m	845 ms	990 ms	1.00
✓ j	990 ms	1105 ms	1.00
✓ u	1105 ms	1385 ms	1.00
✓ g	1390 ms	1500 ms	1.00
✓ ʊ	1500 ms	1550 ms	1.10
✓ d	1550 ms	1610 ms	1.00

I assume you, good master witcher, have brought my daughter home.
We're thankful for that.

```
 7. [2111408020] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408020[1.9718]Well__beautiful_village_.wav
 8. [2111408021] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408021[1.7745]Wonder_who_planted_those_flowers.wav
 9. [2111408022] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408022[4.1545]They_usually_grow_in_mossy_tree_shades
10. [2111408023] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408023[1.7396]We-_we're_there_.wav
11. [2111408025] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408025[7.5795]Alene!_There_you_are__Haven't_I_warnec
12. [2111408026] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408026[6.1514]No_need_for_explanation!_You_did_somet
13. [2111408027] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408027[5.6406]I_assume_you__good_master_witcher__hav
14. [2111408028] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408028[3.9688]Wait_for_me_in_the_tavern__I've_got_sc
15. [2111408205] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408205[1.6576]You_will_leave_her_alone_.wav
16. [2111408206] [   done   ] E:\W3Modding\_projects\modTrial6\speech\speech.en.wav\2111408206[2.6568]I-_I_will_leave_her_alone.wav
```

Illustration 15: Screenshot showing the phoneme extractor of the radish modding tools, into which dialog speech strings and audio files and mapped to phonemes for lip sync generation.

Another is the fact that the tool’s encoders are able to link between some assets automatically. For instance, the quest editor automatically detects all community definitions and provides e.g. the option to spawn them without the user needing to type a path to the asset. Thirdly, the tools feature an extensive in-game UI to define, preview, debug, and output various kinds of data, including scene data (animations, cameras, ..) as well as placement of entities in the world, foliage and more (Illustration 16).



Quest Layer: layer demons
Layer Entities: 7

*v areas [1]
v waypoints [0]
v scenepoints [1]
v mappins [3]
v actionpoints [2]
v statics [0]
v interactives [0]

Illustration 16: Screenshot showing the in-game UI of the radish modding tools for placing entities in the game world.

Chapter 4. REDengine 3’s Quest Design Language

Having established a theoretical framing and exposed cultural conditions, the next and last major matter in this thesis is consecutive: the actual exposition of technical givens in REDengine 3 quest modding and their implications for usage.

The form of this analysis centres on the quest graph as a manifested pattern for quest design. Starting from there, I will explore the language of the various aspects that are element to or adjacent to REDengine 3 quests. When discussing each (sub-)aspect, I will follow the differentiation between constraints and grain in Section 2.2. Firstly, I will demonstrate how the aspect is technically determined. Secondly, I will explain how this constraint shapes the user's workflow and their understanding of what is feasible and what is not, thereby defining the design language of the engine.

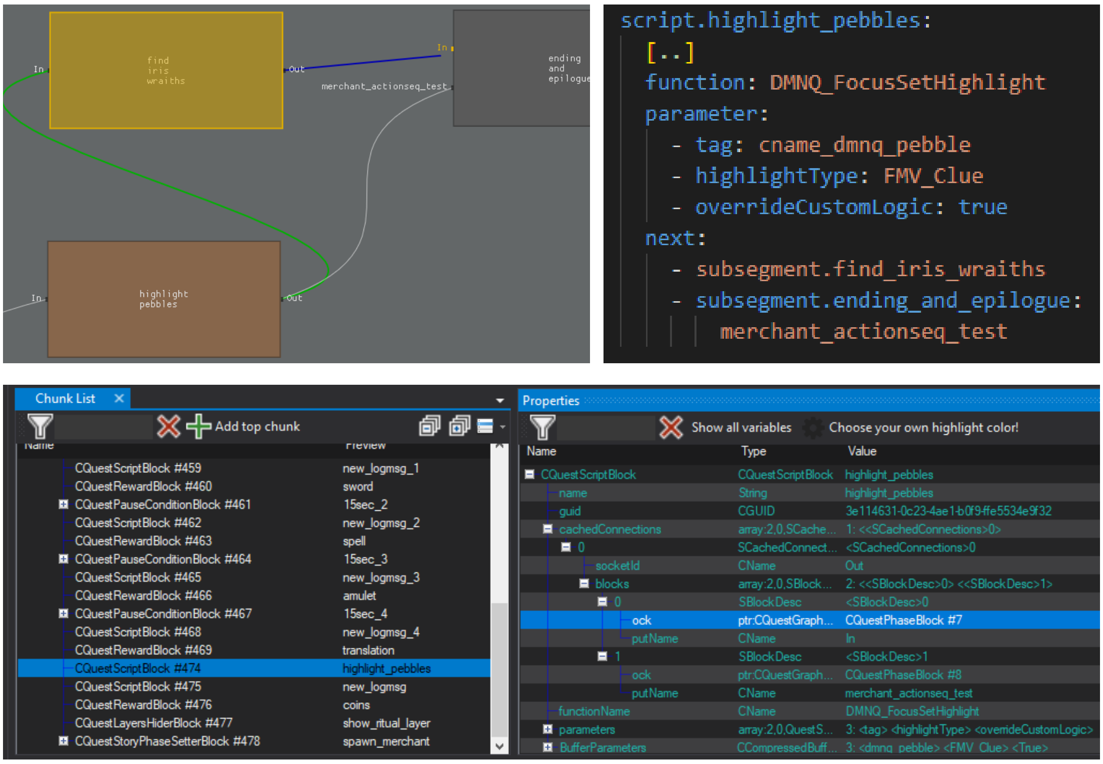
4.1 Graph-like Progression

The main structures for quest designers to work with in REDengine 3 are what I call *quest graphs*. A quest graph is understood here as a technical asset in video game engines which allows developers to execute the design of template quests’ high-level collection and connection of events. As a consequence, the quest graph is also the place where the integration of the majority of quest-related assets is happening.

4.1.1 Graph Editing

The radish modding tools contain a quest editor (cf. Illustration 14), which allows to create new *quest blocks*, configure them and set *connections*. A running quest graph receives a *signal* through its start block(s) and this signal will progress through the other blocks and connections whenever possible. I differentiate two major block types, besides some minor exceptions: *event* blocks, which cause some happening (e.g. a character walking to a location), and *condition* blocks, which halt the signal until some condition (e.g. time of day is evening) has been fulfilled. Any block can have multiple inputs and/ or outputs, to accommodate e.g. for different outcomes in a scene block. The blocks and connections form a mathematical *graph*: a set of arbitrary nodes combined with a set of edges (node pairs); in the quest graph, the nodes are the blocks and the edges are the connections.

The visual graph editor is a major advantage for quest creation, since it communicates the state of the graph's set of edges much clearer than other means. The alternatives are the YAML representation of the graph in the radish tools (to which the editor saves), and editing in WolvenKit, which both have the problem that due to them presenting the graph as an one-dimensional list of blocks, one cannot intuitively see which block connects with which (see Illustration 17).



```yaml
script.highlight_pebbles:
  [..]
  function: DMNQ_FocusSetHighlight
  parameter:
    - tag: cname_dmnq_pebble
    - highlightType: FMV_Clue
    - overrideCustomLogic: true
  next:
    - subsegment.find_iris_wraiths
    - subsegment.ending_and_epilogue:
        merchant_actionseq_test
```

Name	Preview
CQuestScriptBlock #459	new_logmsg_1
CQuestRewardBlock #460	sword
⊞ CQuestPauseConditionBlock #461	15sec_2
CQuestScriptBlock #462	new_logmsg_2
CQuestRewardBlock #463	spell
⊞ CQuestPauseConditionBlock #464	15sec_3
CQuestScriptBlock #465	new_logmsg_3
CQuestRewardBlock #466	amulet
⊞ CQuestPauseConditionBlock #467	15sec_4
CQuestScriptBlock #468	new_logmsg_4
CQuestRewardBlock #469	translation
CQuestScriptBlock #474	highlight_pebbles
CQuestScriptBlock #475	new_logmsg
CQuestRewardBlock #476	coins
CQuestLayersHiderBlock #477	show_ritual_layer
⊞ CQuestStoryPhaseSetterBlock #478	spawn_merchant

Name	Type	Value
⊟ CQuestScriptBlock	CQuestScriptBlock	highlight_pebbles
name	String	highlight_pebbles
guid	CGUID	3e114631-0c23-4ae1-b0f9-ffe5534e9f32
⊟ cachedConnections	array:2,0,SCache...	1: <<SCachedConnections>0>
⊟ 0	SCachedConnect...	<SCachedConnections>0
socketId	CName	Out
⊟ blocks	array:2,0,SBlock...	2: <<SBlockDesc>0> <<SBlockDesc>1>
⊟ 0	SBlockDesc	<SBlockDesc>0
ock	ptr:CQuestGraph...	CQuestPhaseBlock #7
putName	CName	In
⊟ 1	SBlockDesc	<SBlockDesc>1
ock	ptr:CQuestGraph...	CQuestPhaseBlock #8
putName	CName	merchant_actionseq_test
functionName	CName	DMNQ_FocusSetHighlight
⊞ parameters	array:2,0,QuestS...	3: <tag> <highlightType> <overrideCustomLogic>
⊞ BufferParameters	CCompressedBuff...	3: <dmnq_pebble> <FMV_Clue> <True>

Illustration 17: Screenshots comparing quest graph edge editing with the means of the radish tools' quest editor (top left), its underlying YAML representation (top right) and access to encoded CR2W quest assets in WolvenKit (bottom).

On the other hand the radish quest editor has a major drawback, which is based on the fact that it is so much more user-friendly. This draws modders away from some additional quest graph modding options possible only outside the tool: REDengine 3 defines 82 `CQuestGraphBlocks`[160] and blocks buildable from `IQuestConditions`, whereas the editor implements only 41 of them, missing blocks like the `CQuestCutControlBlock`, which was deemed to be "very important" by a modder and quest designer at CD Projekt RED[161]. This block allows

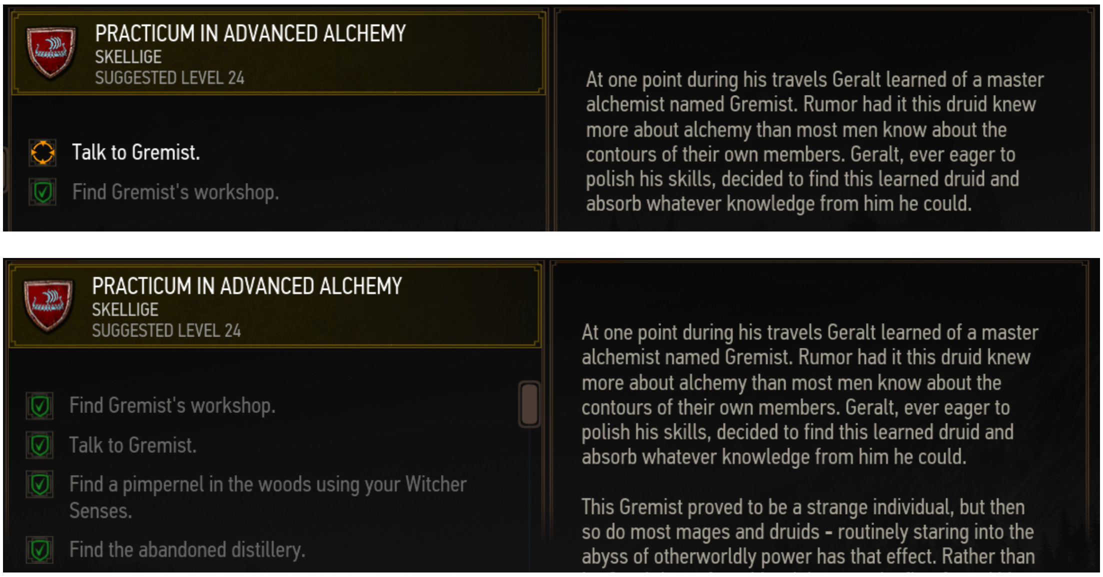
[160] These are defined in REDengine 3's *RTTI* (run-time-type-information) index.

[161] erx. (2018). "there is a good variety of different blocks to choose from, however two VERY important blocks that are missing in my opinion are CQuestHiResRealtimeDelayCondition and CQuestCutControlBlock (and of course scripted actions would be super cool to have in the future)" [Message]. *w3 radish tools. Discord*. Online:

users to eliminate a pending signal in another branch of the quest graph and was likely not implemented due to the need for a special UI feature. While cutting control can be managed with a workaround utilising the fact system, the `CQuestEntityMotionBlock` can not be simulated by other blocks and was omitted in the editor's preprogrammed blocks for unknown reasons. There are also some blocks, like `CQuestEncounter*`, which would require a significant amount of additional effort to implement in the radish tool set: modders have not come to figure out how encounters work in REDengine 3 and thus the creation of new encounters and their control with quest graphs is not possible as well.

4.1.2 Interactive Fiction with Questgraph

The bare quest graph itself is not exposed to the player in its entirety. Instead, as in many video games, a questgraph has some associated elements to communicate to the player where they are within the quest progression. Quests in TW3 are organised into *stages*, markers of important points to be reached in the graph. A stage could be that the player has cleared a dungeon and found its secret, which is then communicated to the player with a specific sound, a HUD pop-up and a new text appearing in the quest journal (see Illustrations 18 & 19).



PRACTICUM IN ADVANCED ALCHEMY
SKELLIGE
SUGGESTED LEVEL 24

Talk to Gremist.

✓ Find Gremist's workshop.

At one point during his travels Geralt learned of a master alchemist named Gremist. Rumor had it this druid knew more about alchemy than most men know about the contours of their own members. Geralt, ever eager to polish his skills, decided to find this learned druid and absorb whatever knowledge from him he could.

PRACTICUM IN ADVANCED ALCHEMY
SKELLIGE
SUGGESTED LEVEL 24

✓ Find Gremist's workshop.

✓ Talk to Gremist.

✓ Find a pimpernel in the woods using your Witcher Senses.

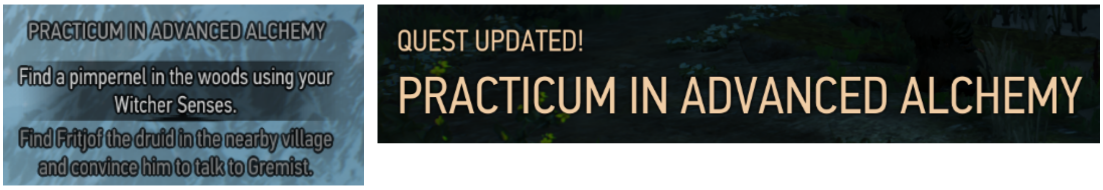
✓ Find the abandoned distillery.

At one point during his travels Geralt learned of a master alchemist named Gremist. Rumor had it this druid knew more about alchemy than most men know about the contours of their own members. Geralt, ever eager to polish his skills, decided to find this learned druid and absorb whatever knowledge from him he could.

This Gremist proved to be a strange individual, but then so do most mages and druids - routinely staring into the abyss of otherworldly power has that effect. Rather than

Illustration 18: Screenshots of a quest entry progression in the quest journal of TW3. The second image shows how various new objectives and a new description appeared, and that some objectives were marked as completed (green checkmark).

https://discord.com/channels/416336392692695040/416343412191789087/448227521268547585 (accessed: 17.03.2023).



PRACTICUM IN ADVANCED ALCHEMY

Find a pimpernel in the woods using your Witcher Senses.

Find Fritjof the druid in the nearby village and convince him to talk to Gremist.

QUEST UPDATED!

PRACTICUM IN ADVANCED ALCHEMY

Illustration 19: Screenshots showing how the questgraph state is communicated via the in-game HUD. The left screenshot shows the current quest's objectives and the focused objective, the right shows a pop-up when a description or objective has been updated.

The series of quest descriptions appearing for a specific actualised quest reveals the strong connection between TW3 quests and interactive fiction: a TW3 questgraph could be reduced to a series of scenes with choices only and quest description updates as consequences – the basis of any *ergodic literature*, that is, text whose reading requires non-trivial effort[162]. The generalised form of description updates are event blocks that trigger an update for the user. Choices on the other hand can be represented in two ways: the first way is to use blocks with multiple outcomes, like a branch on whether a given predicate is true or false. The second possibility is to connect a block's output to at least two condition blocks which wait for a player action.

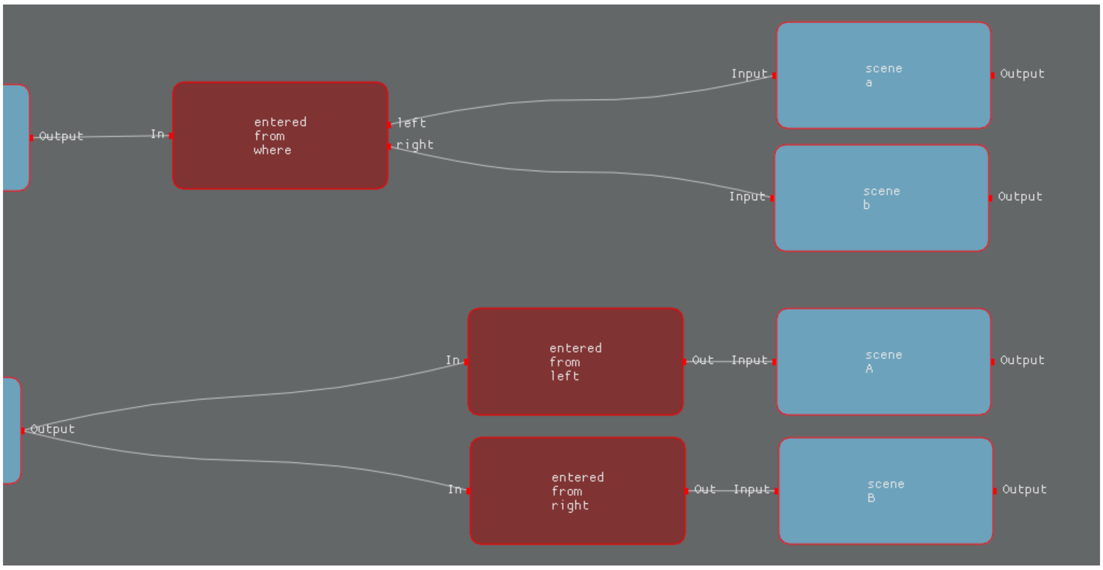


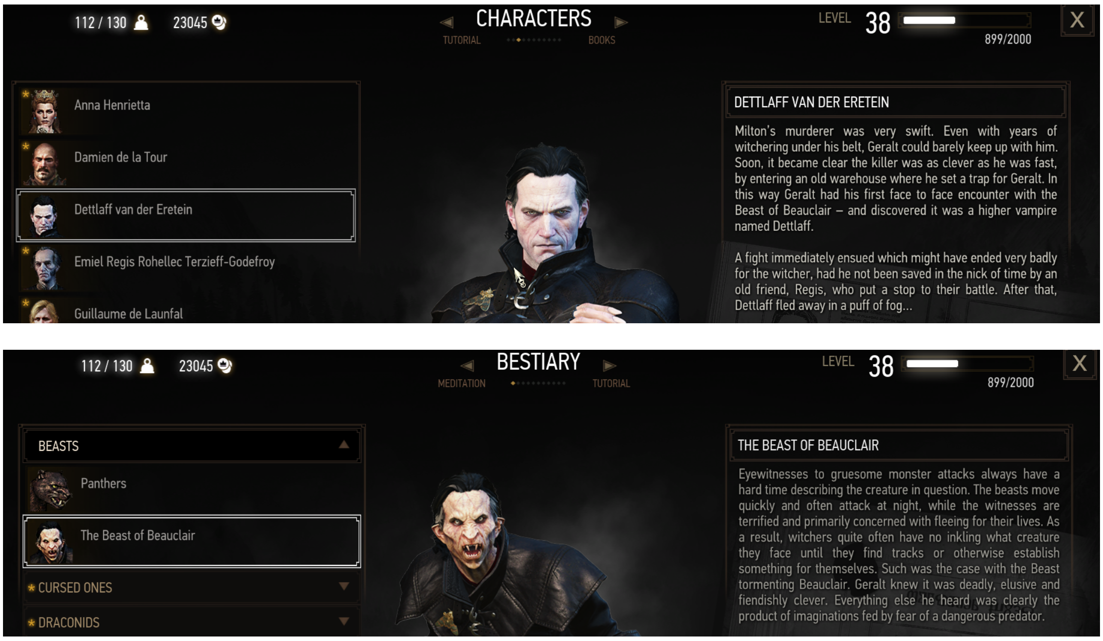
Illustration 20: Screenshot of the two ways to represent player choice: either the branch is embedded in the code that is called upon the signal reaching the condition block (top), or it is implicitly coded by two condition blocks representing the happening of either branch (bottom).

[162] Aarseth, Espen. *Cybertext—Perspectives on Ergodic Literature*. pp. 1–2.

Interactive fiction often not only encompasses the delivery of text, but also e.g. changing background images – this principle is abstracted and scaled up by the general event and condition blocks of REDengine 3 and its effects/ queries to the game (sub)states. Nonetheless it can be said that the emphasis on narrative text within the game provides developers with a cultural nudge towards seeking inspiration from written stories. This effect is naturally amplified when the team includes a dedicated writer, such as the ones at CD Projekt RED.

4.1.3 Narrative Glossary Categories

The subtle flair of written fiction is supported by the game's extensive glossary, containing entries with additional information on, for instance, characters and beasts. This design choice not only encourages the provision of backstories to these, but also underlines the recurring theme of text as a storytelling/ worldbuilding device in TW3, catering to players who seek to immerse themselves further. Moreover, the separation makes for an interesting example of narrative told through UI: The separation suggests two mutually exclusive categories, implying that humans and beasts do not overlap and forming a grain which nudges designers to produce this distinction in their work.



DETTLAFF VAN DER ERETEIN

Milton's murderer was very swift. Even with years of witchering under his belt, Geralt could barely keep up with him. Soon, it became clear the killer was as clever as he was fast, by entering an old warehouse where he set a trap for Geralt. In this way Geralt had his first face to face encounter with the Beast of Beauclair – and discovered it was a higher vampire named Dettlaff.

A fight immediately ensued which might have ended very badly for the witcher, had he not been saved in the nick of time by an old friend, Regis, who put a stop to their battle. After that, Dettlaff fled away in a puff of fog...

THE BEAST OF BEAUCLAIR

Eyewitnesses to gruesome monster attacks always have a hard time describing the creature in question. The beasts move quickly and often attack at night, while the witnesses are terrified and primarily concerned with fleeing for their lives. As a result, witchers quite often have no inkling what creature they face until they find tracks or otherwise establish something for themselves. Such was the case with the Beast tormenting Beauclair. Geralt knew it was deadly, elusive and fiendishly clever. Everything else he heard was clearly the product of imaginations fed by fear of a dangerous predator.

Illustration 21: Screenshots of glossary entries regarding Detlaff in the character glossary and bestiary, respectively. The bestiary screenshot furthermore exemplifies how the engine codifies some animals as beasts.

Consumers of marketing material surrounding TW3's release might be confused at this point: one of the core selling points of the game was its questioning of whether beasts are the only monsters in the world and whether humans, indeed, might be monsters as well[163].

The dissonance resolves when inspecting the actual usage of the journal's categories: these, contrary to their appearance, can each gain an entry concerning the same character/ beast, such as one of the main characters from Blood and Wine, Detlaff. Illustration 21 demonstrates how they are sorted in both "characters" and "bestiary".

Nonetheless: Detlaff's double categorisation is a rarity (to the author's knowledge the only case in the game), which shows that even the developers at CD Projekt RED, who had the explicit goal to blur beast-human boundaries, did not take up this chance more often and use the subversive potential of this grain.

Another aspect that comes to light when inspecting the journal categories is that there is a range of animals, which are sorted in as "beasts". And while for a lot of monsters the distinction to characters and even humans is blurred in the game's narratives (as in the Detlaff example), the same work was not done for wolves, bears, panthers and others. I will return to this theme of ambiguity towards animals when discussing how animals are placed in the game world in Section 4.2.

4.1.4 Objectives incentivising Heavy Guidance

Next to quest stage descriptions, further ways to telegraph a quest's state to the player include short, written objectives (Illustration 18) whose state (activated/ deactivated/ failed/ succeeded) can be controlled with a dedicated block, as well as a quest outcome (completed/ failed, compare Illustration 22). The existence and regular use of the quest journal and HUD in the base game strongly incentivises designers and modders to use these features. While objective lists and descriptions can be left empty, this will result in empty panels in the journal and HUD view, breaking with the expectations of players that the dedicated spaces should be filled with instructions and commentary. Consequently, designers will be drawn to filling those structural spaces and create heavily structured task lists for the player, providing short objectives for every situation and thus giving them heavy guidance as to what their next goal should be.

The quest graph editor then resembles a kind of control panel, where an operator provides a test subject with instructions and triggers events based on their circumstances in the test chamber (the world). In the videogame *Portal*, this rhetoric

[163] cf. CD Projekt RED. (2013). *The Witcher 3: Wild Hunt - Killing Monsters Cinematic Trailer*. Online: https://youtu.be/c0i88t0Kacs (accessed: 17.03.2023).

of the player being a subject of someone’s productivity management is addressed explicitly. The avatar *Chell*, a test subject, is led through various test chambers by the artificial intelligence supervisor GLaDOS, who instructs them to do various tasks, like carrying cubes, reaching a place or pressing a button in dedicated environments set up for Chell’s traversal. The main twist of the game is based on Chell’s decision to exit the loop of recurring tests and externally imposed tasks by finding a way out.

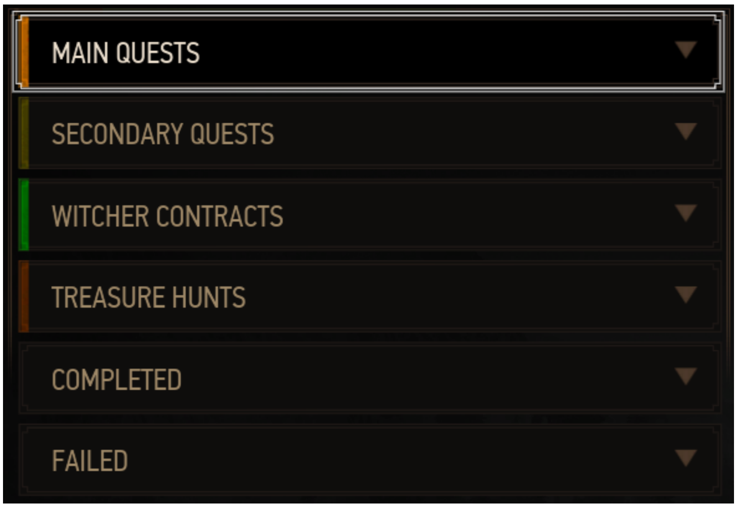
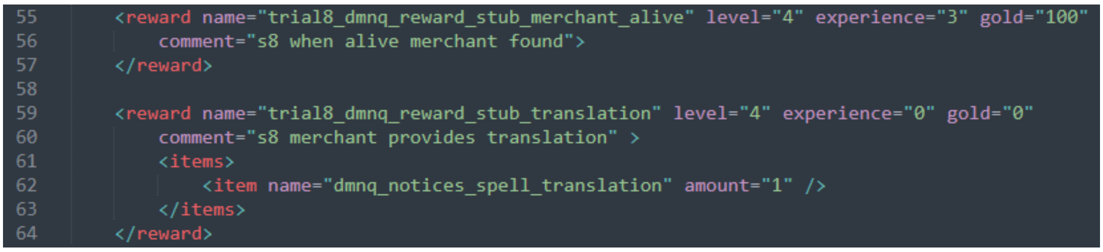


Illustration 22: Screenshot of quest categorisation in TW3 quest journal.

REDengine 3, in its basic features, does not incentivise such a meta-commentary on player choice and developer control. Instead, the existence of the journal in its given form as well as some other features like automated rewards (see farther below), which are completely absent in *Portal*, indicate that CD Projekt RED embraced quests as a game element with a heavy focus on productivity. This is a recurring feature in big open world role-playing games, with many games in popular series such as *The Elder Scrolls*, *Grand Theft Auto* or *Assassin’s Creed* featuring a HUD display of objectives and extensive quest/ mission journals. While some titles, such as *The Legend of Zelda: Breath of the Wild* or *Elden Ring* show that open world RPGs can work without such heavy guidance, CD Projekt RED seems to have decided to stick with it: not only do all titles in the Witcher series of video games have these features, but also their newest game Cyberpunk 2077.

4.1.5 Quest Rewards and the Path of Least Resistance

An important component of many quests in both literature and video games are quest rewards. In a most general understanding, quest rewards are any positive event for the player generated by the quest system for achieving a certain progress. Classical quest rewards are (cut-)scenes, special items, improvement of a stat, level-ups, savepoints or new views or sections of a level. REDengine 3 has a dedicated reward system implemented for the combined delivery of *reward packages*, which are defined in a special XML format:



```
<reward name="trial8_dmnq_reward_stub_merchant_alive" level="4" experience="3" gold="100"
    comment="s8 when alive merchant found">
</reward>

<reward name="trial8_dmnq_reward_stub_translation" level="4" experience="0" gold="0"
    comment="s8 merchant provides translation" >
    <items>
        <item name="dmnq_notices_spell_translation" amount="1" />
    </items>
</reward>
```

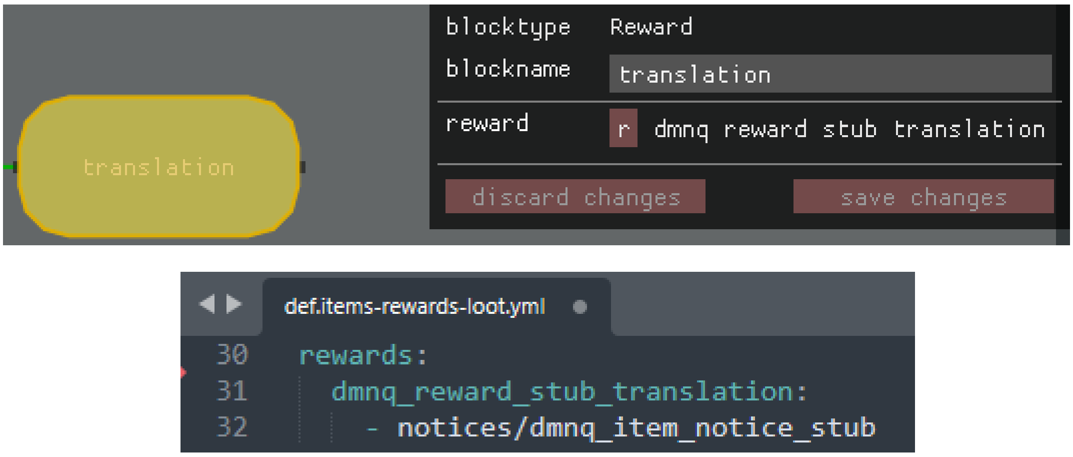
Illustration 23: Screenshot of two modded TW3 reward definitions in XML.

In Illustration 23 we can see two rewards, each delimited by `<reward></reward>` tags. Such a reward contains the following parameters: a name for identification in other parts of the game, the level for which the reward is defined (used for scaling based on the player's actual level), experience gain, gold gain and lastly a comment field for developers. A reward can provide multiple items, and each `<item/>` has a name field to reference the externally defined item and a specification of the amount.

Within the radish modding tools, not only reading and editing the XML format can pose a challenge, but also the integration of reward packages into quests and scenes – the game's most important high-level event-systems.

Rewards in REDengine 3 seem to be very rigidly defined at first glance, but investigating what an item definition can be shows that item rewards are actually highly customisable: an item is a CR2W entity, which, due to the chunk-/component-based approach of CR2W allows creators to add a huge variety of features to the item: sound effects, visual effects and even further meshes. Item entities can also be derived from other definitions in the TW3 data files. In general, a lot of constellations are possible, for example the merging of two completely different character models, since limbs are separate sub-entities and can be built together in arbitrary combinations – but one caveat remains: there is no documentation on how all the CR2W chunks which are definable and addable are intended to relate to each other. This, effectively, makes entity creation a complex field of researching, testing and iterating for modders.

Similarly, the definition of rewards itself is largely undocumented for WolvenKit and the more content-creation-focused radish modding tools alike and riddled with challenges to overcome. To illustrate the point, I will describe the process of reward creation in the radish toolset here:



blocktype Reward
blockname translation
reward r dmnq reward stub translation

```
rewards:
  dmnq_reward_stub_translation:
    - notices/dmnq_item_notice_stub
```

Illustration 24: Screenshots showing how rewards are integrated in a quest using the radish modding tools. Top: reward block in the quest editor. Bottom: stub reward in YAML.

The quest editor provides a reward block, wherein one can select a radish reward definition – which is different from the REDengine 3 XML definition noted above and written in YAML. The YAML reward is then transformed by the encoders to its XML counterpart. However: the radish reward definitions do not allow to set the experience and gold parameters. That is why users of the tools are falling back to defining the XMLs themselves. But leads to another problem: the quest editor will not find the rewards since it scans for YAML-defined rewards only. Consequently, the solution pattern is to create a YAML reward, assign it in the quest editor and replace the encoder-generated XML with the custom one, where the experience and gold-values are set.

Undocumented fallacies like these or the fact that the XMLs need to be based on UTF-16 instead of the much more common UTF-8 encoding are major reasons why many modders do not continue in this learning path, or why it takes them a long time to finish a project. It is such problems which proof the value of (great) tools: while the radish modding tools enable a lot of opportunities, they do not enable as many efficient patterns for people to use them.

4.2 Layers on the Game World

In this section I will take a closer look at how the quest graph can be connected to the 3D game world with modding tools.

4.2.1 Guiding with Mini-Map Pins

As a first entry point, the matter of quest-induced navigation with the (mini) map will be investigated. Maps are a tool to assist in navigation through a topography, which, in the case of the open-world video game TW3, is a 3D rendered world filling the entire screen. The entity traversing this world is the avatar Geralt, steered by the player.

Since the game world is *open*, providing multiple paths to reach a given target, several issues may arise for the player as noted by video game essayist Razbuten. These issues include getting lost, missing an event or a location, or being detected by an enemy, all of which motivate the use of a map[164]. TW3 provides a map in the menus and a mini map in the HUD[165]:

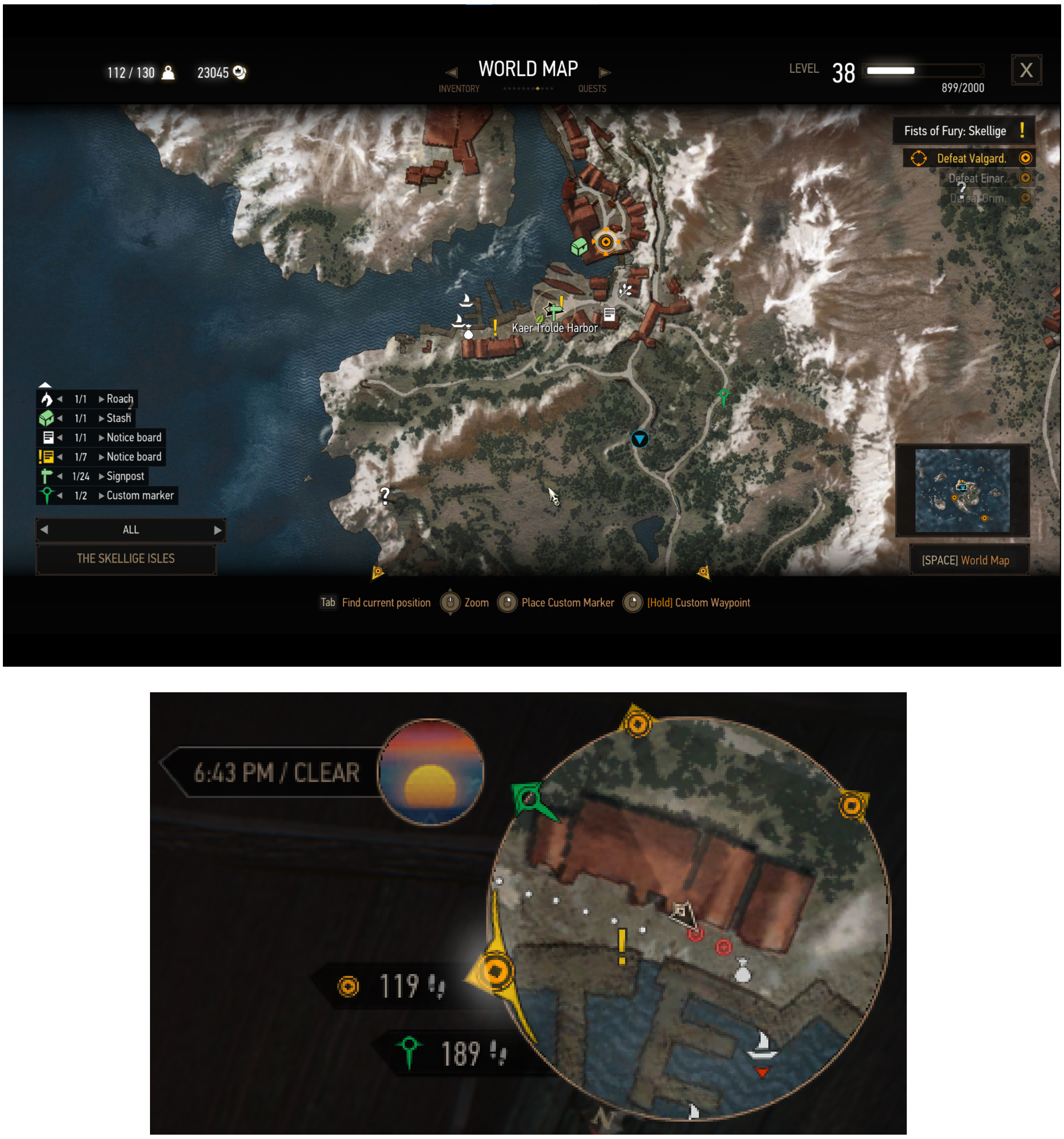


Illustration 25: Screenshots of TW3's map (top) and mini map (bottom).

[164] Razbuten. (2022). *The Problem With Mini-Maps*. Online: https://youtu.be/nmzYRT7LBQs (accessed: 17.03.2023).

[165] HUD = heads up display: UI that is rendered on top of the game world.

For the issue of getting lost, TW3's maps automatically display the player's location (Illustration 25 the cursor-like shapes). Moreover, quest locations are automatically displayed and highlighted (Illustration 25: yellow markers), with the world map also allowing players to set their own custom waypoints (Illustration 25: blue and green markers). For custom markers and the quest locations the direction is shown on the compass, with active quest objective locations even showing a way to the goal. Regarding the issue of missing events, collectable herbs or roaming enemies, we can see that for example the world map displays question marks to lead players to points of interest, while the mini map prominently shows yellow question marks to depict a new quest.

In their video "The Problem With Mini-Maps", Razbuten argues that mini maps are "too useful"[166]. While they do increase the player's efficiency, they do it so much that some players begin to rely on the mini map as their sole means for traversal and shifting attention oftentimes away from the activity of moving through the game world itself. According to Razbuten, this is especially tragic in the case of open-world games like TW3, since

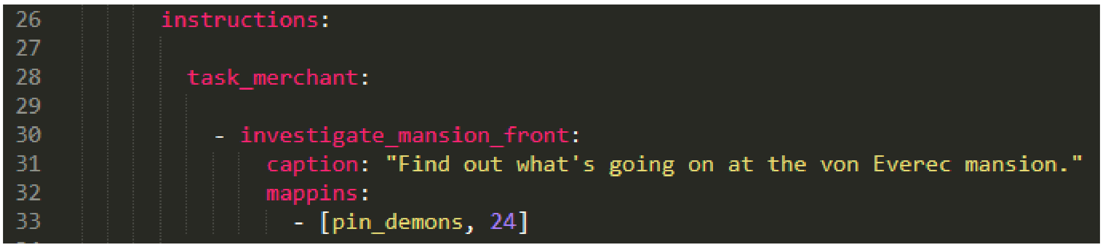
> »the quality of an open-world title isn't determined by how quickly the player can get through it. In fact, most would say that a good open-world game is one you can get lost in.«[167]

REDengine 3's cultural grain is such that many designers will be directed towards providing a good setup for player overreliance on the mini map. Modders or designers working with the engine most likely have in mind how TW3 uses the aforementioned features and moreover most probably strive to achieve the same aesthetic patterns in their work, if they use REDengine 3 as a tool. Were it different, then much easier to operate and less specific tools could be used – for example the Story Creator[168] for *Assassin's Creed Odyssey* or any RPGMaker quest system plugin. Consequently, if a modder has committed to using TW3, then it would be possible for them to construct quests as in *Elden Ring* – without quest journal entries, objectives and quest markers – but the motivation to do so is unlikely to be there in the first place if they intend to stay faithful to the game. All the while, actually omitting a map pin is similarly easy as it is to leave a quest description blank. One simply needs to not add the "map-pin" element (lines 32-33 in Illustration 26) under the respective objective definition:

[166] ibid.

[167] ibid.

[168] https://assassinscreed.ubisoft.com/story-creator-mode/en-us/



```
instructions:

  task_merchant:

    - investigate_mansion_front:
        caption: "Find out what's going on at the von Everec mansion."
        mappins:
          - [pin_demons, 24]
```

Illustration 26: Screenshots of a map-pin definition for an objective in the radish modding tools.

Now, as discussed in Section 3.2, there are a lot of modders interested in improving or tinkering with games per se. Would they not be interested in removing such features that are regarded by some as a flaw? First of all, there are in fact several mods modifying the maps' functionality, e.g. disabling the display of the player's location[169] or of objectives on the main map[170]. The problem is that disabling all automated location guidance causes a major soft lock for players, since the writing and quest design in TW3 frequently lacks sufficient clues in character dialogue or journal descriptions to deduce the next location to go to for a quest. That is why there is a strong convention in TW3 that every quest has location markers for each of its objectives, and this convention transfers to those who intend to create new quests that are integrated in the existing world. The case might be different in a new world hub, where the introduction of new conventions would be more feasible, but the publishing of such a mod is not foreseeable for the near future.

4.2.2 Beast Ambiguities in TW3 Communities

To grasp the most immediate connection between quest graphs and the game world, we need to look at three types of engine content respectively called entities, layers and communities. *Entities* encompass most game data that can be placed into the game world, most notably static meshes (architecture, props, ..), areas (e.g. for use in a quest wait block, or to locally change a world aspect) or characters. *Layers* are collections of entity placements in the game world. For instance, one layer could contain all entity placements forming a small village, while another layer contains all areas triggering quest progression in that village, and yet another layer containing all the waypoints for NPCs. *Community* definitions contain one or more *phases* with each defining how a given set of NPCs behaves by default when they are not specifically scripted to do something else.

Both layers and communities can be (de-)activated via the questgraph, and communities can be triggered to change to a different phase. Thus the game world

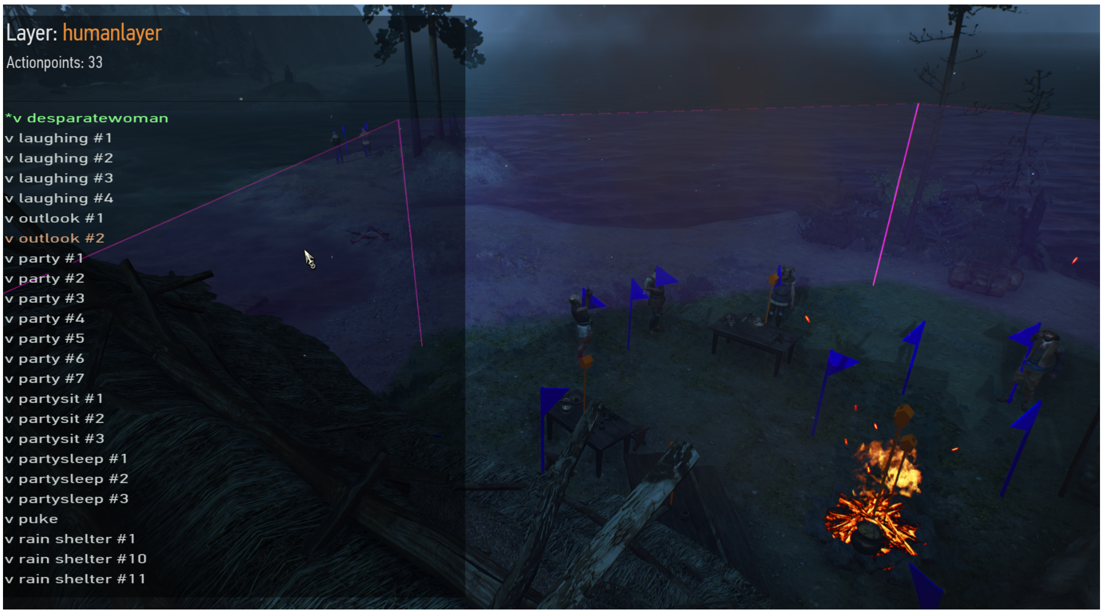
[169] https://www.nexusmods.com/witcher3/mods/1045
[170] https://www.nexusmods.com/witcher3/mods/2796

of TW3 becomes *layered* with set pieces. This is in principle quite similar to theatre, where the base stage is decorated with props, background canvases or painted plates that can be slid onto the stage from left and right. The set designer for theatrical plays then corresponds to the environment artist in video games, and the stage manager to the executive side of the quest designer's job.

In Illustration 27.1, the house, bonfire and pirates are entities collected in the layer named "humanlayer". A community definition, which is not part of the layer but rather references it, defines that the pirate entities are assigned to various activities (e.g. `work_woman/stand_mw_upset__jt`) defined in the layer – called *actionpoints*.

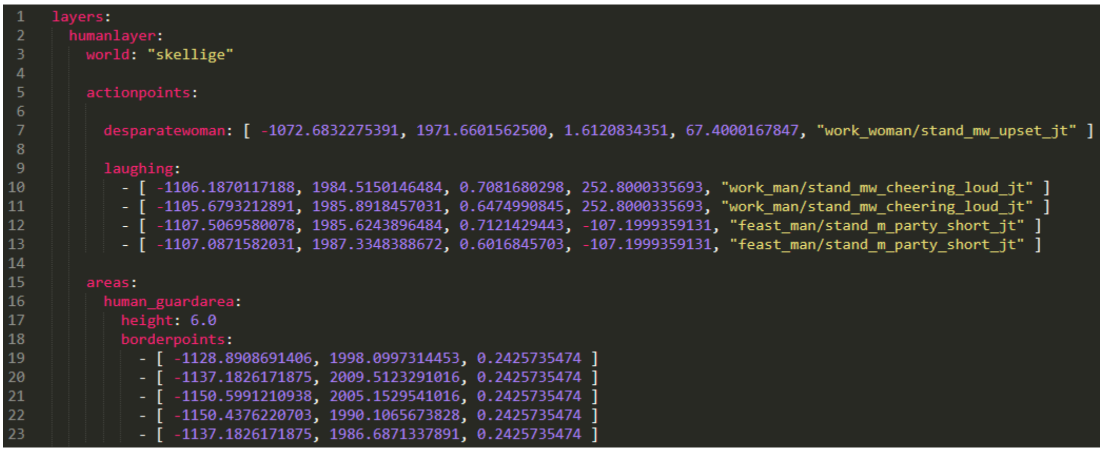
This technical structure suggests and at times almost enforces patterns which range from interesting, over simply useful to questionable. An interesting pattern is that by the logic of the game files, humans, animals and monsters alike form communities. The ambiguous categorisation of entities is further reflected when inspecting the file paths of each kind. It shows that all three are subsumed under the "characters" directory, while in the subdirectory models a distinction between humans, animals and monsters is made.

But this distinction is once again disturbed by the fact that the `\animals` directory also contains animals which are typically hostile in TW3 and sorted in in the "bestiary", like berserker bears or wolves. Using entities in a community, one can set the entity's *attitude* to the player, that is, whether they will attack them or how quickly. It shows that all monsters and typically hostile animals can be set to be friendly to the player, which might be the final sign that the game files and the engine, while in general ambiguous, in the end literally and metaphorically support blurring the line between monsters, animals and humans.

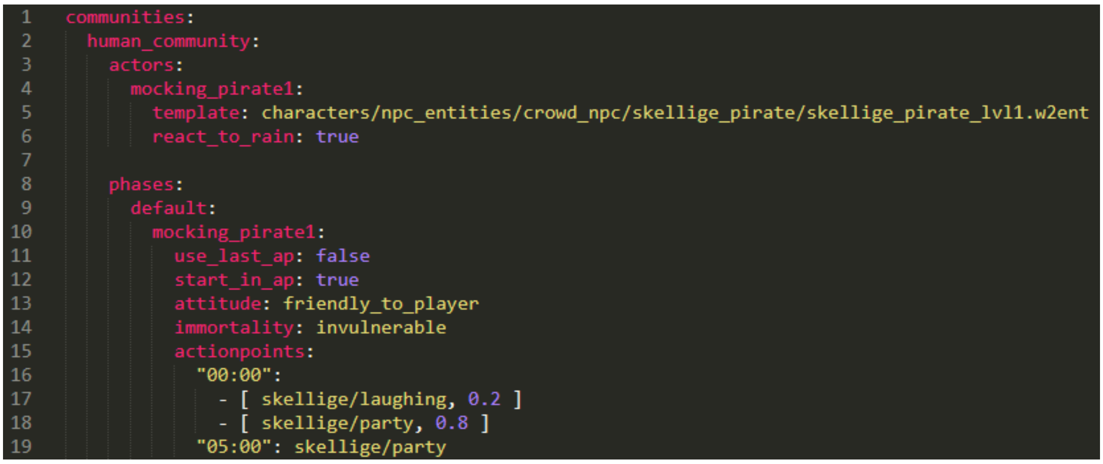


Layer: humanlayer
Actionpoints: 33

*v desparatewoman
v laughing #1
v laughing #2
v laughing #3
v laughing #4
v outlook #1
v outlook #2
v party #1
v party #2
v party #3
v party #4
v party #5
v party #6
v party #7
v partysit #1
v partysit #2
v partysit #3
v partysleep #1
v partysleep #2
v partysleep #3
v puke
v rain shelter #1
v rain shelter #10
v rain shelter #11



```yaml
layers:
  humanlayer:
    world: "skellige"

    actionpoints:

      desparatewoman: [ -1072.6832275391, 1971.6601562500, 1.6120834351, 67.4000167847, "work_woman/stand_mw_upset_jt" ]

      laughing:
        - [ -1106.1870117188, 1984.5150146484, 0.7081680298, 252.8000335693, "work_man/stand_mw_cheering_loud_jt" ]
        - [ -1105.6793212891, 1985.8918457031, 0.6474990845, 252.8000335693, "work_man/stand_mw_cheering_loud_jt" ]
        - [ -1107.5069580078, 1985.6243896484, 0.7121429443, -107.1999359131, "feast_man/stand_m_party_short_jt" ]
        - [ -1107.0871582031, 1987.3348388672, 0.6016845703, -107.1999359131, "feast_man/stand_m_party_short_jt" ]

    areas:
      human_guardarea:
        height: 6.0
        borderpoints:
          - [ -1128.8908691406, 1998.0997314453, 0.2425735474 ]
          - [ -1137.1826171875, 2009.5123291016, 0.2425735474 ]
          - [ -1150.5991210938, 2005.1529541016, 0.2425735474 ]
          - [ -1150.4376220703, 1990.1065673828, 0.2425735474 ]
          - [ -1137.1826171875, 1986.6871337891, 0.2425735474 ]
```



```yaml
communities:
  human_community:
    actors:
      mocking_pirate1:
        template: characters/npc_entities/crowd_npc/skellige_pirate/skellige_pirate_lvl1.w2ent
        react_to_rain: true

    phases:
      default:
        mocking_pirate1:
          use_last_ap: false
          start_in_ap: true
          attitude: friendly_to_player
          immortality: invulnerable
          actionpoints:
            "00:00":
              - [ skellige/laughing, 0.2 ]
              - [ skellige/party, 0.8 ]
            "05:00": skellige/party
```

Illustration 27: Screenshots of layer and community definitions using the radish modding tools.

Top (27.1) visualises a layer using the in-game quest UI.

Middle (27.2) shows a layer that is defined in YAML.

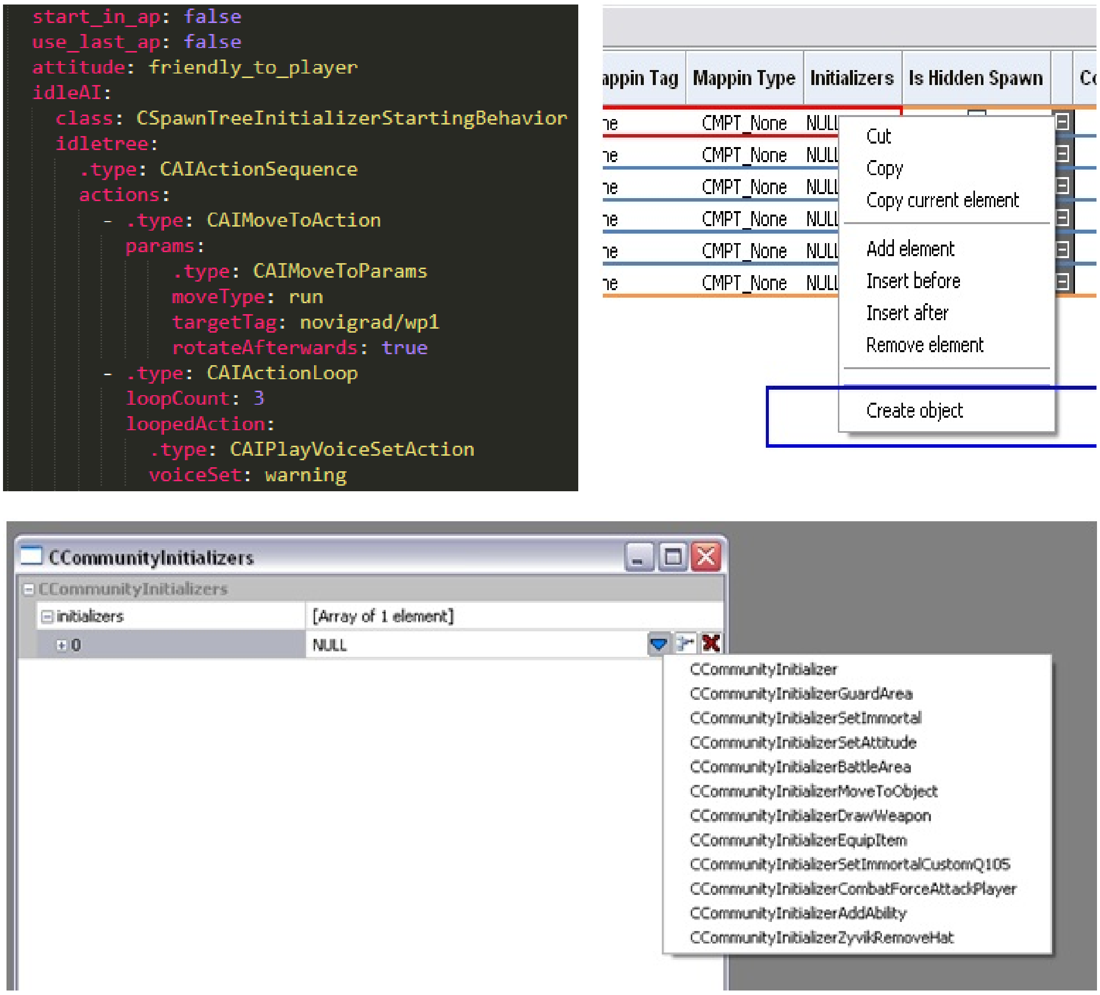
Bottom (27.3) shows how one NPC's behaviour is defined via a YAML community definition.

4.2.3 Encounters and The Time Issue of Modding

A useful aspect of this layer-entity-community system is that it enforces the definition of a community in order for an actionpoint to work. And by enforcing a community definition, designers are naturally guided to the many NPC behaviour features that REDengine 3 provides, such as different action points per day time, reacting to rain, initialisers, roaming in areas or large-scale assignment of entities to classes of action points (which was heavily utilised e.g. in the in-game city of Novigrad). In the radish modish tools, these are all features which are quite accessible, given the human-readable format of community YAML-definitions: in Illustration 27.2/ 3 it can be seen how defining the pirate's behaviour is a matter of 19 lines of simple attribute setting and list creation. Even though the radish NPC behaviour definitions support a lot of features and have relatively good usability, they do not support every engine feature and most probably are not as user-friendly. This matter proves to be an excellent example to make a point about the time and scale aspect of designing/ executing with tools, tool creation and the production gap between industry and modding scenes.

CD Projekt RED likely began developing the REDengine, which was used for all subsequent games, after the release of The Witcher in 2008. The TW2 version of the engine was subsequently released in 2013. Therein we can already find a smaller version of the community definitions I described above, editable by the user with a small UI tool, as depicted in Illustration 28. The illustration shows the big advantage of UI tools over text tools: all possible properties that a community can have are predefined editable fields, with some, like the "initializer", even having dedicated submenus to search and select from all initializers scripted so far. The radish modding tools definitions support the same initializer feature, but it took creator rmemr 8 years, starting from the TW3 release, to implement the full capacity of that feature[171], and the usability is significantly lower: the option to add these initializers is "hidden" in the sense that there is no visible "Initializers"-field in a new, empty YAML file, and there are no in-place lists of available initializers with automatic generation of initializer parameters – everything needs to be manually researched in the Witcher Script code base and transferred to the YAML file. This is a process taking much more time, and even though quite useful, it is unlikely it will actually be used often, given that after 8 years, the TW3 modding scene has shrunk significantly.

[171] A basic version was already available with the tool release in 2019/2020.



```
start_in_ap: false
use_last_ap: false
attitude: friendly_to_player
idleAI:
  class: CSpawnTreeInitializerStartingBehavior
  idletree:
    .type: CAIActionSequence
    actions:
      - .type: CAIMoveToAction
        params:
            .type: CAIMoveToParams
            moveType: run
            targetTag: novigrad/wp1
            rotateAfterwards: true
      - .type: CAIActionLoop
        loopCount: 3
        loopedAction:
          .type: CAIPlayVoiceSetAction
          voiceSet: warning
```

initializers [Array of 1 element]
0 NULL

CCommunityInitializer
CCommunityInitializerGuardArea
CCommunityInitializerSetImmortal
CCommunityInitializerSetAttitude
CCommunityInitializerBattleArea
CCommunityInitializerMoveToObject
CCommunityInitializerDrawWeapon
CCommunityInitializerEquipItem
CCommunityInitializerSetImmortalCustomQ105
CCommunityInitializerCombatForceAttackPlayer
CCommunityInitializerAddAbility
CCommunityInitializerZyvikRemoveHat

Illustration 28: Screenshots of the TW2 REDkit community (initialisation) editor and an analog definition in the radish modding tools.

Playing a modded quest like “A Night to Remember”[172] takes about 1-3 hours. Creating such a quest takes a year or more. Building the tools to create those quests took 4 years and is still ongoing – with some problems not being solved even after all this time.

Communities are not the only way to steer NPC behaviour in REDengine 3. They are only a variant of the more powerful *encounters*, a class which enables flying monsters and generic spawn behaviours. The encounter feature has been discussed and investigated several times on the w3 radish tools server but no one found out how to arrange chunks in a CR2W such that a working encounter is produced. This was not only complicated by the complexity of the subject, but also by the long-standing inability to copy-paste CR2W chunks in WolvenKit, which prevented researching modders to replicate encounters from existing ones.

The issue of encounters is solved out of the box in the TW2 REDkit, providing, as with communities, a handy UI. Given the fact that there is a ready-to-use solution for

[172] https://www.nexusmods.com/witcher3/mods/4670

so many problems at hand, I believe that there is more to the motivation of some modders than what Nathaniel Pool found in their analysis (see Section 3.1). The work by some of the reverse engineers, researchers, tool creators and deep-diving quest modders can also be interpreted as an expression of demarcation, resistance and striving towards a feeling of superiority against CD Projekt RED. The basis for this would be a community that – as discussed in Section 3.2 – also materialised in the TW3 modding scene. The fuel for contra-stances can be found in the frustration about the power asymmetry, the time modders spent and spend due to it, the playbour issue[173] and the many engine flaws uncovered by modders. A we-versus-the-company stance can not only be observed in the letters by wghost81, but also details like the blatantly *non*-corporate-styled discord server cultures e.g. on "CDPR Modding Community & The Boiz", which makes it reasonable to add "proving the modding scene to stand en par with the company" as a possible motivation for modders.

4.3 Scenes and the Power of Domain Languages

In this section I provide a perspective on how domain-specific languages and the CR2W format contribute to REDengine 3's overall language and success.

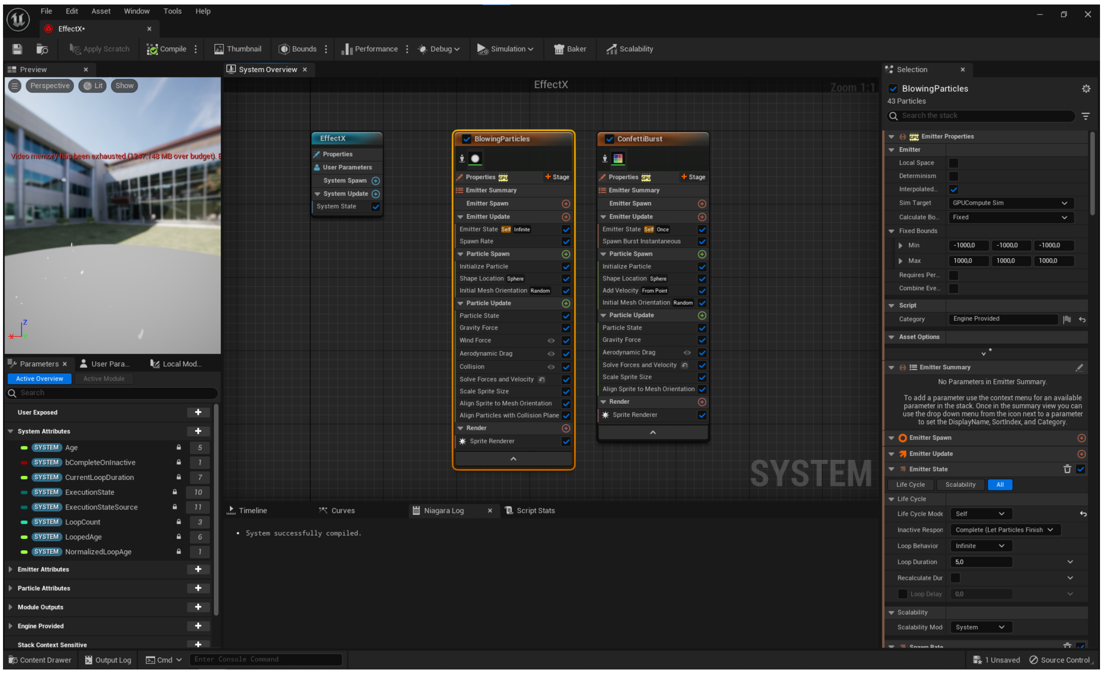
4.3.1 Abstraction Hierarchies with Specialised Ends

The architecture of many game engines, together with the editing interfaces made for them, have a structural quality which is essential for them being a tool for solving not only one, but multiple classes of problems. I call this quality an *abstraction hierarchy with specialised ends*. In the most general sense, an abstraction hierarchy is composed of sets of structures and patterns, with some sets symbolising and encapsulating other sets. As an example, the Unreal Engine Niagara Editor[174] abstracts VFX building in the engine: upon opening a VFX asset with it, the editor (cf. Illustration 29) provides a structure consisting of (among else) a space representing the assets VFX components, a preview-window representing the components in a visual manner and a details-panel exposing various smaller settings of a given selected element. Patterns can be found in recurring principles like VFX features being plug-in-able or data values being composable to complexer calculations (see Illustration 30) in place. The Niagara Editor is just one of many specialised front-end editors, upon which no other editor or system is built, which

[173] As discussed in Section 3.2 and in Kücklich, Julian. *Precarious Playbour: Modders and the Digital Games Industry*.

[174] Epic Games, Inc. (2023). *Unreal Engine 5* (5.1).

shows it to be a *specialised end* of Unreal's abstraction hierarchy. The file format of Niagara VFX assets on the other hand, while being an abstraction on the engine's internal VFX composer, is no specialised end, since the abstraction does not end here, as evident with the Niagara Editor.



Video memory has been exhausted (1247.148 MB over budget).

System successfully compiled.

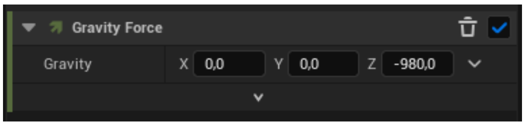
BlowingParticles
43 Particles

Emitter Properties
Emitter
Local Space ☐
Determinism ☐
Interpolated... ☑
Sim Target: GPUCompute Sim
Calculate Bo...: Fixed
Fixed Bounds
Min: -1000,0 -1000,0 -1000,0
Max: 1000,0 1000,0 1000,0
Requires Per... ☐
Combine Eve... ☐
Script
Category: Engine Provided
Asset Options

Emitter Summary
No Parameters in Emitter Summary.
To add a parameter use the context menu for an available parameter in the stack. Once in the summary view you can use the drop down menu from the icon next to a parameter to set the DisplayName, SortIndex, and Category.

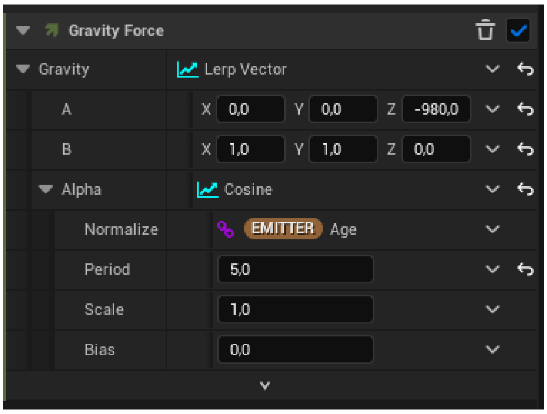
Emitter State
Life Cycle
Life Cycle Mode: Self
Inactive Respon: Complete (Let Particles Finish
Loop Behavior: Infinite
Loop Duration: 5,0
Recalculate Dur ☐
Loop Delay: 0,0
Scalability
Scalability Mod: System

Illustration 29: Screenshot of a Niagara VFX asset opened in the Unreal Engine Niagara tool. In the middle the VFX components can be edited, in the top left a rendering is located, and to the right details of the currently selected component are editable.



Gravity X 0,0 Y 0,0 Z -980,0

Illustration 30: Screenshot of a pattern engrained in the NIagara Editor: parameters of a given VFX element can be complexified by various transformations. In this case the gravity parameter has been transformed to periodically swing to zero.



Gravity: Lerp Vector
A X 0,0 Y 0,0 Z -980,0
B X 1,0 Y 1,0 Z 0,0
Alpha: Cosine
Normalize: EMITTER Age
Period: 5,0
Scale: 1,0
Bias: 0,0

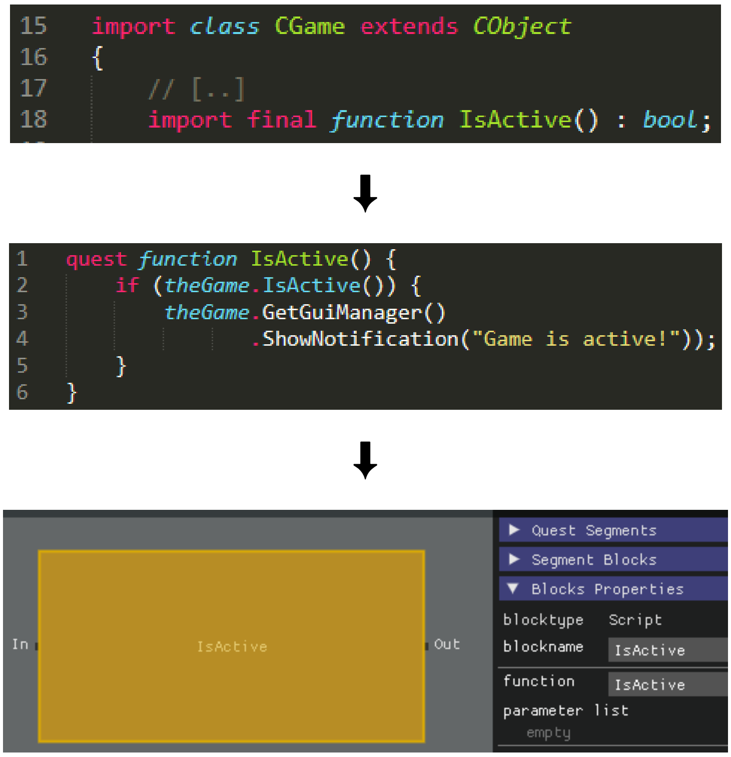
Many users who might not understand the technical details of VFX creation are able to reach out to that realm by means of this sub-language as presented by the Niagara Editor.

The interesting lens arising from this is the following: since the tool engineers from a company or modding scene have limited time only, not all engine features can get a specialised end point at the highest abstraction level a toolset provides. Thus, in analysing what features were exposed at what abstraction level of an engine tool sets language, we can gain information on what kind of domain-specific languages the developers wanted the users to speak.

In Section 3.3 and over the course of Chapter 4 I already touched upon the various major levels of abstraction present in REDengine 3 and the radish modding tools. These are the CR2W file formats, XML, REDscript, YAML file formats and the two visual editors, one for quest graphs, one for phonemes (cf. Illustrations 14/ 15). Sorting in problem classes like AI, scene and lastly overall quest design in this hierarchy will give us a good closing overview of what the grain is leading to.

4.3.2 REDscript as Mediating Language

Starting from the quest editor, the two abstraction levels referenced by it are YAML format files and REDscript – and I will now take a closer look at REDscript as a programming domain language for high-level game scripting. REDscript, as discussed in Section 3.3, is the second major language that was used to program REDengine 3 and TW3 next to C++, and compared to C++ it is faster to use.



```
import class CGame extends CObject
{
    // [..]
    import final function IsActive() : bool;
```

```
quest function IsActive() {
    if (theGame.IsActive()) {
        theGame.GetGuiManager()
            .ShowNotification("Game is active!");
    }
}
```

blocktype Script
blockname IsActive
function IsActive
parameter list
empty

Illustration 31: Screenshots of one way how REDscript blends into the abstraction hierarchy: by exposing C++ engine/ game features and building on them for use in the next higher level.

Illustration 31 demonstrates how REDscript and gameplay coders function as mediators between C++ (programmers) and quest (designers). The base for REDscript is C++, which is relatively hard to write and to which the job of an engine programmer is dedicated at CD Projekt RED. REDscript helps expose and build on code written in C++. In the designer friendly quest editor quest designers implement the high-level sequential flows of a game. REDscript allows designers to code some features for use in the editor themselves, and lets gameplay coders relieve the engine programmers from implementing higher-level functionality. This division of work follows directly from the engine's architectural design and establishes gameplay programmers as coders who have to deal with a lot of the shifting requirements of a game: those requirements, which can not be easily implemented within the stricter visual editors and which are not too critical for performance to let a programmer implement it in C++. Examples for this are setting UI contents, the crafting system or all the quest functions and conditions for the quest editor. The trichotomy introduced by this separation most importantly allows for quest designers to focus more on their high-level storytelling, supporting CD Projekt RED's goal to create narrative-driven games. The trichotomy also imposes a hierarchy of dependencies and task formulation from designers to programmers, where the engine programmers are the first to be able to do something and the designers the last.

For modders, the C++ - REDscript - editor choice defaulted to REDscript until the quest editor was released. Consequently, the mods made for REDengine 3 have been dominated by REDscript and simple XML modifications/ mesh imports for a long time. And while there are very few quest mods as of the day of writing, there are none which actually modify the engine's C++ source code, since in contrast to REDscript, C++ code is translated to optimised machine code which is practically impossible to reverse engineer. Thus, the decisions by CD Projekt RED engine architects as to what is written in REDscript or C++ over long time spans influenced what modders can actually change or not.

Some other architectural decisions of consequence can be found in REDscript's interface to quests and scenes as well as entities and communities. In order for quest and scene editors to automatically find functions for usage in the "script" blocks of the visual editors, REDscript features the `quest` and `storyscene` keywords to be put in front of function definitions (cf. the second screenshot of Illustration 31). This feature at the latest characterises REDscript as a *domain specific language*. In computer science, domain specific languages are programming languages which are not neutral in the sense that they "are tailored to a specific application domain"[175]. The inclusion of these keywords furthermore underlines that REDengine 3 is geared towards having a focus on being a storytelling tool and, more interestingly, that it hardcodes the semantics of terms such as "scene" and "quest" to a large degree. We have already seen plenty of evidence for how quest content is standardised in certain formats such as the quest graph or the pattern of NPC community building, and we will now see that the very same is true for scenes as elements of a quest.

4.3.3 A Language for Scene Building

Scenes in REDengine 3 are integrated as many other quest elements via simple quest blocks into the quest graph – direct their own quests and determine when and where a certain scene should play. A scene asset itself is a CR2W file containing a *scene graph*, a structure very similar to quest graphs, with the difference that nodes can be sections of speech and cinematics, choices or script calls. Illustration 32 demonstrates how such a scene graph looks visually – while the example is taken from TW2 and opened in REDkit, the actual tool from CD Projekt RED used for TW3 looks basically the same[176].

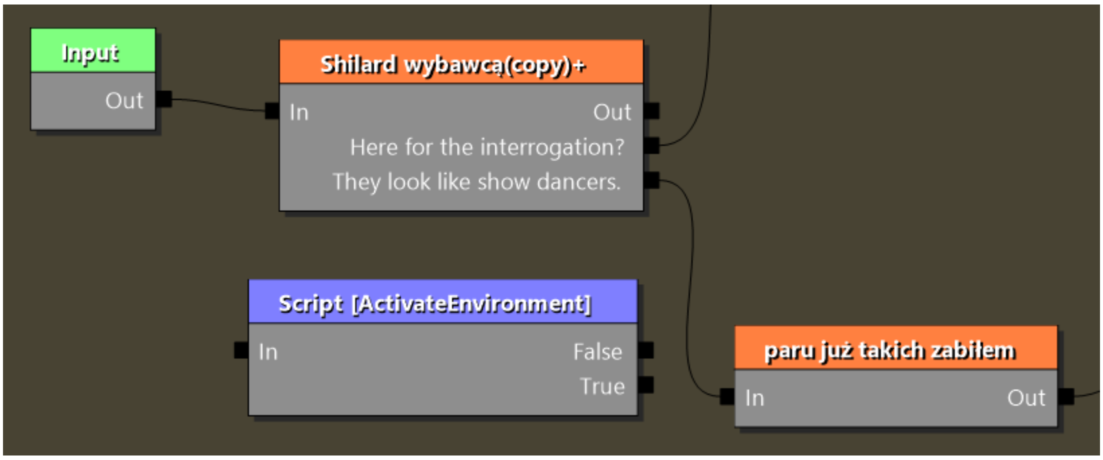

Input
Out
Shilard wybawca(copy)+
In
Out
Here for the interrogation?
They look like show dancers.
Script [ActivateEnvironment]
In
False
True
paru już takich zabiłem
In
Out

Illustration 32: Screenshots of a sample scene graph extract from TW2, showing input, section with choice, script (added by the author) and regular section nodes.

Already at this point we can make some interesting notes. First of all, much like the quest graph, the scene graph provides incentives for the making of complex and non-sequential narrative experiences: as discussed in Section 4.1, non-sequentiality is supported by the graph structure itself and by the possibility to give nodes and

[175] Mernik, Marjan; Heering, Jan; Sloane, Anthony M. (2005). *When and how to develop domain-specific languages*. In: *ACM Computing Surveys*. vol. 37. no. 4. pp. 316-344. p. 316.
[176] Tomsinski, Piotr. (2017). *Behind the Scenes of the Cinematic Dialogues in The Witcher 3: Wild Hunt*. Online: https://youtu.be/chf3REzAjgI?t=479 (accessed: 17.03.2023).

graphs different outcomes. But there is another system in REDengine 3 assisting with more complex story structures, and that is called the *fact system*. Facts in REDengine are simple named integer values, which usually assume the values 0 and 1 (false and true) only. While the player plays the game, the engine upholds a database of all facts that were added at some point. The special point about the fact system is that facts can be added, edited and read from any level, be it the quest graph, a scene, a script and due to their scriptability, also any entity, community, etc. This makes facts a kind of "glue" between the different levels and allows for complex interactions, for example death counters for a group of hostile NPCs triggering a progression of quest objectives from "Kill 0/X" to "Kill X/X" – a common pattern which can easily be built using the fact system.

We can conclude that the interconnected graph and code systems on multiple levels in REDengine 3 focus a specific kind of systemic thinking: content architects are forced to think in terms of the subsystems at their disposal, the content(-patterns) definable within and the content definable by means of interaction patterns like facts.

4.3.4 Modularity and Abstraction as Enabling Base Principles

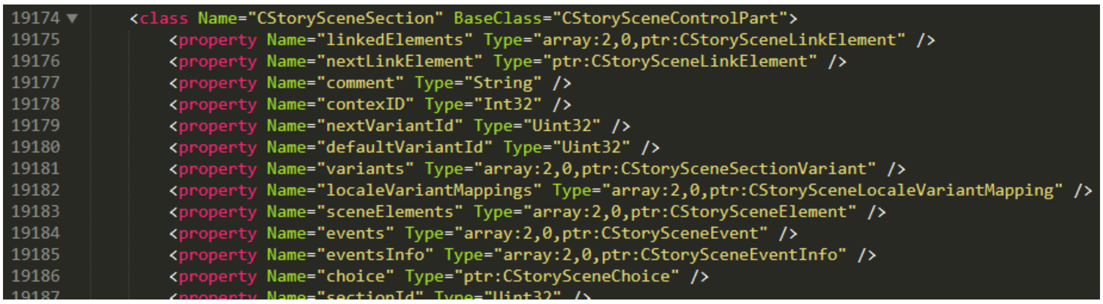
The scene definition tools in REDengine 3 and in the radish modding tools also expose a major reason for the successful production of the game's numerous scenes: user-friendly modularity. At the time of release, the "nuanced [..] dialogue system"[177] supporting high-quality narratives in not only main but also side quests, was one of the aspects praised. The reasons for CD Projekt RED cinematic designers being able to go into such detail that the scenes feel nuanced are, first of all, the modularity of the CR2W scene definitions and, secondly, the way in which this very technical format is made accessible to the user through the abstraction hierarchy.

The base format CR2W allows arbitrary lists and nestings of chunks, where chunks are essentially nodes of data with sets of properties – the technemes of the CR2W language of REDengine 3. Scenes are one type of data saved in this modular format, with elements of a scene mapped by means of standardised protocols to CR2W chunks. The screenshot in Illustration 34.1 shows how a `CStorySceneSection` element is mapped to a chunk with various properties, among which are definitions of a camera, additive animations for some characters, look-ats[178] and scene props. The interrelation between these elements is defined in REDengine 3's *RTTI* (*runtime type interface*), a collection of definitions for data

[177] Prescott, Shaun. (2015). *The Witcher 3 PC review*. PC Gamer. Online: https://www.pcgamer.com/the-witcher-3-review/ (accessed: 17.03.2023).

[178] "look-ats" define where a scene actor is looking at in a given moment.

types in the engine (the structure of e.g. communities and quest graphs is defined here as well).

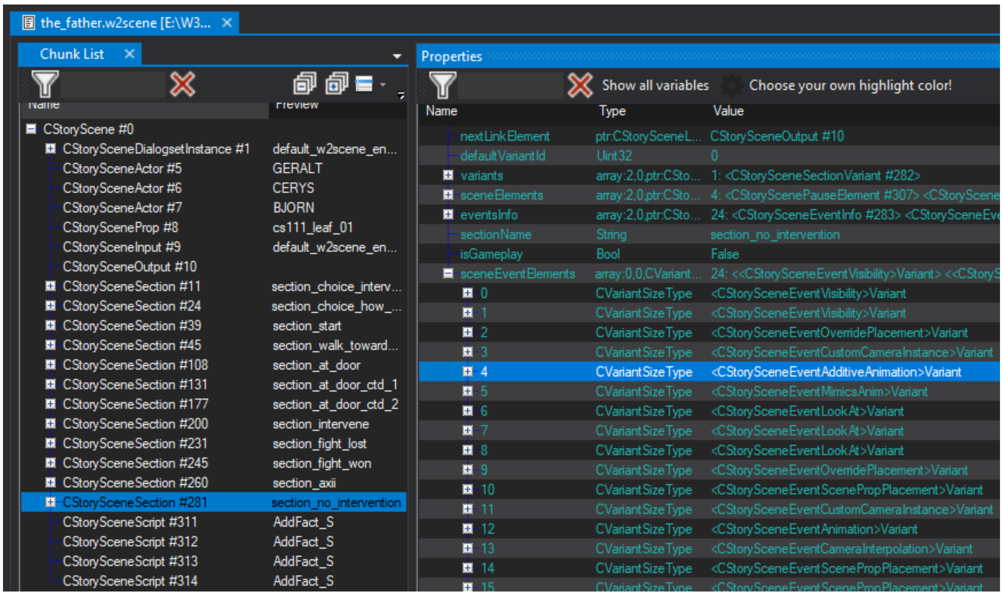


```
<class Name="CStorySceneSection" BaseClass="CStorySceneControlPart">
    <property Name="linkedElements" Type="array:2,0,ptr:CStorySceneLinkElement" />
    <property Name="nextLinkElement" Type="ptr:CStorySceneLinkElement" />
    <property Name="comment" Type="String" />
    <property Name="contexID" Type="Int32" />
    <property Name="nextVariantId" Type="Uint32" />
    <property Name="defaultVariantId" Type="Uint32" />
    <property Name="variants" Type="array:2,0,ptr:CStorySceneSectionVariant" />
    <property Name="localeVariantMappings" Type="array:2,0,ptr:CStorySceneLocaleVariantMapping" />
    <property Name="sceneElements" Type="array:2,0,ptr:CStorySceneElement" />
    <property Name="events" Type="array:2,0,ptr:CStorySceneEvent" />
    <property Name="eventsInfo" Type="array:2,0,ptr:CStorySceneEventInfo" />
    <property Name="choice" Type="ptr:CStorySceneChoice" />
    <property Name="sectionId" Type="Uint32" />
```

Illustration 33: Screenshot of an excerpt of the RTTI definition of `CStorySceneSection`*, defining for instance that a story scene section has a field* `nextLinkElement` *to define its successor.*

The definitions in the RTTI are modular in the sense that for example a section is allowed to contain any event inheriting the `CStorySceneEvent` class. This modularity of the RTTI definitions combined with its implementation in the CR2W format drives not only the scene system but the majority of assets in the entire game and thus also forms the first major point of abstraction in the engine's abstraction hierarchy.

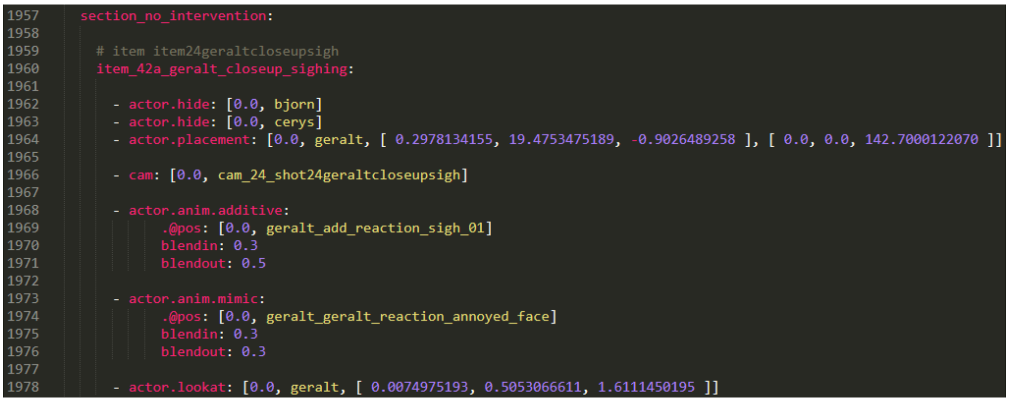
The YAML scene definitions in the radish modding tools are a pretty direct translation of the engine's asset format. A radish scene file contains three building blocks. First of all, a *repository* of all the camera definitions (e.g. position, rotation, field of view) as well as any asset references from the depot (to reuse e.g. existent entities and animations). Secondly, the *storyboard*, consisting of sections of items with each item applying to a line or pause from the dialogscript. An item defines, relative to the line or pause, that a scene event like a character animation, prop attachment, sound effect or camera change is played (see Illustration 34.2). Thirdly, a *dialogscript*, where all lines/ pauses with preceding cues to storyboard items, choices and script sections are defined. The dialogscript defines the scene graph.



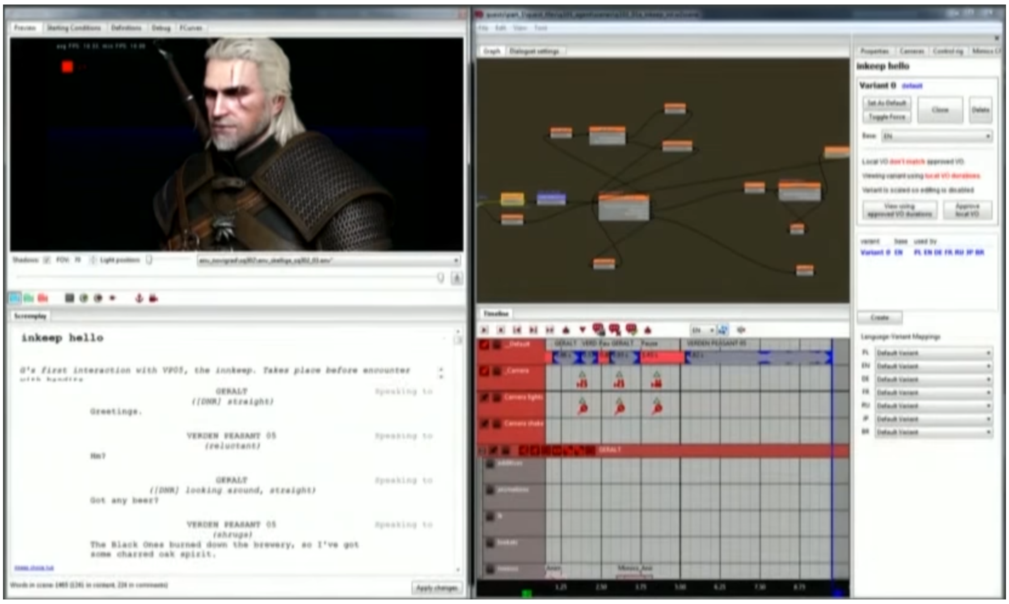
the_father.w2scene [E:\W3...
Chunk List
Properties
Show all variables
Choose your own highlight color!

Name	Preview
CStoryScene #0	
CStorySceneDialogsetInstance #1	default_w2scene_en...
CStorySceneActor #5	GERALT
CStorySceneActor #6	CERYS
CStorySceneActor #7	BJORN
CStorySceneProp #8	cs111_leaf_01
CStorySceneInput #9	default_w2scene_en...
CStorySceneOutput #10	
CStorySceneSection #11	section_choice_interv...
CStorySceneSection #24	section_choice_how_...
CStorySceneSection #39	section_start
CStorySceneSection #45	section_walk_toward...
CStorySceneSection #108	section_at_door
CStorySceneSection #131	section_at_door_ctd_1
CStorySceneSection #177	section_at_door_ctd_2
CStorySceneSection #200	section_intervene
CStorySceneSection #231	section_fight_lost
CStorySceneSection #245	section_fight_won
CStorySceneSection #260	section_axii
CStorySceneSection #281	section_no_intervention
CStorySceneScript #311	AddFact_S
CStorySceneScript #312	AddFact_S
CStorySceneScript #313	AddFact_S
CStorySceneScript #314	AddFact_S

Name	Type	Value
nextLinkElement	ptr:CStorySceneL...	CStorySceneOutput #10
defaultVariantId	Uint32	0
variants	array:2,0,ptr:CSto...	1: <CStorySceneSectionVariant #282>
sceneElements	array:2,0,ptr:CSto...	4: <CStoryScenePauseElement #307> <CStoryScene
eventsInfo	array:2,0,ptr:CSto...	24: <CStorySceneEventInfo #283> <CStorySceneEve
sectionName	String	section_no_intervention
isGameplay	Bool	False
sceneEventElements	array:0,0,CVariant...	24: <<CStorySceneEventVisibility>Variant> <<CStoryS
0	CVariantSizeType	<CStorySceneEventVisibility>Variant
1	CVariantSizeType	<CStorySceneEventVisibility>Variant
2	CVariantSizeType	<CStorySceneEventOverridePlacement>Variant
3	CVariantSizeType	<CStorySceneEventCustomCameraInstance>Variant
4	CVariantSizeType	<CStorySceneEventAdditiveAnimation>Variant
5	CVariantSizeType	<CStorySceneEventMimicsAnim>Variant
6	CVariantSizeType	<CStorySceneEventLookAt>Variant
7	CVariantSizeType	<CStorySceneEventLookAt>Variant
8	CVariantSizeType	<CStorySceneEventLookAt>Variant
9	CVariantSizeType	<CStorySceneEventOverridePlacement>Variant
10	CVariantSizeType	<CStorySceneEventScenePropPlacement>Variant
11	CVariantSizeType	<CStorySceneEventCustomCameraInstance>Variant
12	CVariantSizeType	<CStorySceneEventAnimation>Variant
13	CVariantSizeType	<CStorySceneEventCameraInterpolation>Variant
14	CVariantSizeType	<CStorySceneEventScenePropPlacement>Variant
15	CVariantSizeType	<CStorySceneEventScenePropPlacement>Variant



```yaml
section_no_intervention:

  # item item24geraltcloseupsigh
  item_42a_geralt_closeup_sighing:

    - actor.hide: [0.0, bjorn]
    - actor.hide: [0.0, cerys]
    - actor.placement: [0.0, geralt, [ 0.2978134155, 19.4753475189, -0.9026489258 ], [ 0.0, 0.0, 142.7000122070 ]]

    - cam: [0.0, cam_24_shot24geraltcloseupsigh]

    - actor.anim.additive:
        .@pos: [0.0, geralt_add_reaction_sigh_01]
        blendin: 0.3
        blendout: 0.5

    - actor.anim.mimic:
        .@pos: [0.0, geralt_geralt_reaction_annoyed_face]
        blendin: 0.3
        blendout: 0.3

    - actor.lookat: [0.0, geralt, [ 0.0074975193, 0.5053066611, 1.6111450195 ]]
```



```
inkeep hello

G's first interaction with VP05, the innkeep. Takes place before encounter
with bandits
                         GERALT                              Speaking to
                    ([DNR] straight)
          Greetings.

                   VERDEN PEASANT 05                         Speaking to
                      (reluctant)
          Hm?

                         GERALT                              Speaking to
              ([DNR] looking around, straight)
          Got any beer?

                   VERDEN PEASANT 05                         Speaking to
                        (shrugs)
          The Black Ones burned down the brewery, so I've got
          some charred oak spirit.
```

Illustration 34: Screenshots of scene definitions in the abstraction hierarchy.

Top (34.1): in the CR2W format, shown in WolvenKit.

Middle (34.2): with YAML in the radish modding tools.

Bottom (34.3): In CD Projekt RED's internal scene editor.[179]

[179] Tomsinski, Piotr. (2016). *Behind the Scenes of the Cinematic Dialogues in The Witcher 3: Wild Hunt*.https://youtu.be/chf3REzAjgI?t=604

Words like “storyboard” or “prop” and the visual appearance of the dialogscript editor in Illustration 24.3 hint towards a major inspiration for TW3 scenes: film. Not only do TW3 scenes[180] cut between different cameras, they are also built using several common principles from the filming industry. Among these are for example *continuity editing* (shot/ countershot, continuity of movement and position)[181] or standard camera framings such as *medium shots*[182]. The radish in-game UI provides grain for this language, with it providing for example a *rule-of-thirds* overlay (see Illustration 35) for camera position/ rotation selection and predefined actor setups. Lastly, the job of cinematic designer at CD Projekt RED shows that the studio modelled not only their engine, but also their job positions around the focus on the filmic scene mode of gameplay as a major tool for storytelling.

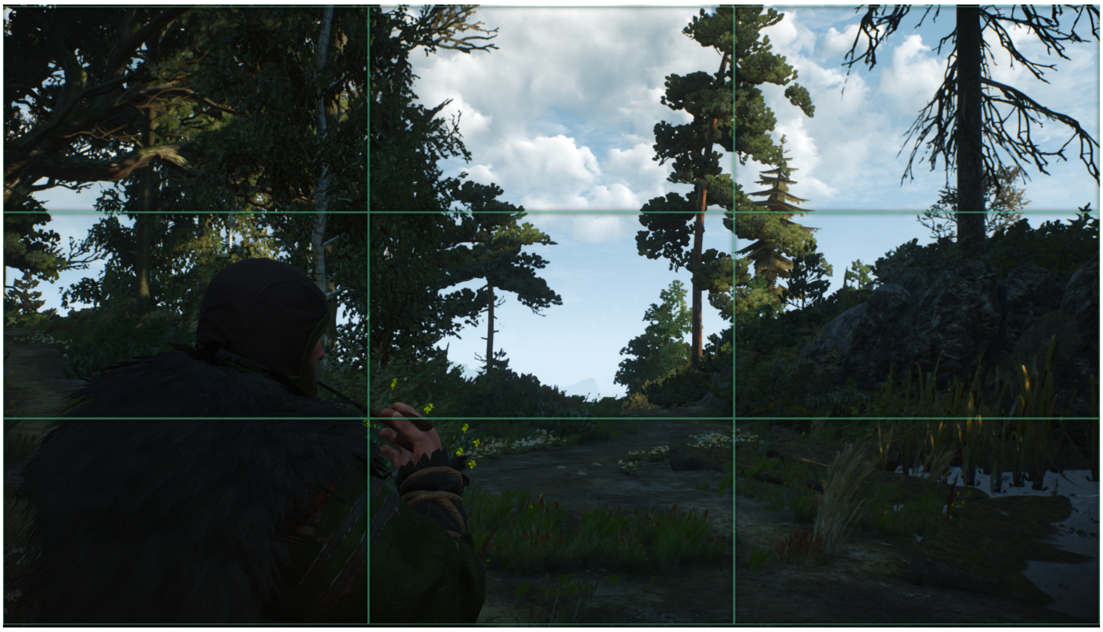


Illustration 35: Screenshot of the radish modding tools in-game camera position/ rotation selection tool with a rule-of-thirds overlay.

The biggest bottleneck of quest creation with the radish modding tools are scene definitions: scenes are, as explained above, a highly interconnected structure with references to other game assets and a graph-like structure. But, in contrast to quests, the tool set does not provide a graphical editor, which is why a lot of work has to be done manually. The next problem lies in the fact that this YAML language for scene definitions is complex: presumed technemes like animations, props,

[180] Compare this scene from the quest “Towerful of Mice”: IGN. (2015). *The Witcher 3 Guide - Side Quest: Towerful of Mice pt 1*. Online: https://youtu.be/SDnnJ4s-SmI?t=706 (accessed: 24.03.2023).

[181] Bordwell, David; Thompson, Kristin; Smith, Jeff. (2017). *Film Art: An Introduction.* 11th ed. McGraw Hill. pp. 230f.

[182] ibid. p. 189

effects, placements, cameras or dialog text need to be defined in different sections of the file.

Patterns such as the division into text definition in the dialogscript and event definition in the storyboard complicate the focus on a given timespan in the scene. The toolset provides only rudimentary help in the form of partial generation of the repository. The generation is based on a limited in-game scene construction and the generation of a timeline of events and dialogscript elements.

Compared to this the official scene editor (see Illustration 34.3) developed by CD Projekt RED replaces the entire management of text-based graph-, timeline- and dialogscript editing and moreover gives the designer instant feedback with a preview window, while modders need to recook the mod and restart the game to inspect changes. The complexity of the systems such as scenes explains why the level of abstraction of the specialised end points is so important for the resulting work environment. Cinematic designers at CD Projekt RED were able to create detailed scenes on a massive scale due to REDengine 3 scenes being highly modular and the modularity being exposed to the user with very convenient tools.

Chapter 5. Conclusion

In this final chapter I will sum up the results of this thesis, reflect on the work and the process and discuss ideas for future research.

5.1 Evaluation

This text has examined the conditions that enable and delimit quest modding in REDengine 3. I will now evaluate the approach, which involved an examination of texts theories from different fields, such as linguistics, media studies and cultural studies, and the practical examination of the REDengine 3 and its modding context.

5.1.1 Integrating Design- with Game Studies

As a first step, an overarching framework for further discussion was searched in the field of design studies. Settling on definitions and differentiations like Herbert A. Simon's understanding of "design" and the corresponding idea of "execution" helped setting this frame: they established a vocabulary to describe what this thesis is about, with an at times almost mathematical derivation of consecutive vocables, as in the case of design/ execution and tools. The value of these concepts for answering the question about the conditions was more in helping to describe and understand the question than in actually answering. In that regard, the case of the quest designer job at CD Projekt RED and their handling of both design and execution was an exception. Starting with Herbert A. Simon's design definition also established that this thesis uses structuralist design studies as a baseline, which allowed it to work interdisciplinary in Section 2.2, where design and game studies met at the issue of tool-aided work.

Starting from a division between machine work and handicraft and using organ -projection theory, tools were characterised to improve some aspect of a user's abilities and allow for reaching out to specific ability realms. I generalised Keogh and Nicoll's conception of game engine "grain" to a property of arbitrary tools – since any tool has particular constraints orienting its user towards particular actions and away from others. This set the stage for one of the core themes: how tools both limit and free people. It also gave a precise conceptual point from which further research, for example investigating affections towards particularly grained tools, could start.

After discussing the value of constraints for users, I moved on to language as a lens for analysing technology. Taking up the structuralist movement in design studies from the 1960s/ 70s and relating it to this thesis' understanding of design, I could identify structuralism as the broadest theoretical influence for this work. My main argument for searching a language for REDengine 3 was to regard the engine as an entity which has something to “say” about the game which it is running – delivering a potentially unusual but valuable perspective on the game and pointing to approaches like Actor-Network Theory. Therein entities like REDengine 3 and a designer using it would be considered equal *actants*, interconnected in a network and communicating with one another. Given that there even already exist derivations of Actor-Network Theory for design studies[183], a study building on this thesis might consider how the theory integrates with REDengine 3 modding.

As a theoretical access to the “language” term, I deployed Jean Boudrillard’s idea of object languages, which is motivated by the observation that the moving parts and synthesises of a system have language-like qualities. I then addressed Boudrillard’s stance that object instability is in conflict with language–likeness, arguing that many languages experience change to some degree and how REDengine 3 is indeed rather stable as a system. Boudrillard also argues for language instability caused by those who engage with the object in question and by means of varying workflows, patterns and other changes in the language – I agreed with this and thus motivated the discussion in Chapter 3. This more holistic lens of grasping an object language proved to be another positive effect originating from the engagement with design studies. Lastly, I identified various ways to grasp elements of a language and concluded with characterising a tool’s design language as the tool language’s subset defining the tool’s grain. The idea of an engine’s language or dialogue with an engine, rendered a productive discussion and added to hints towards an Actor-Network Theory-like perspective on tools, but did not produce an explicit formulation of a “vocabulary” of technical units and grammar. This might be resolved by investigating concrete pattern languages for games, as written by Robert Nystrom for architecture[184], Marcus Trenton for quest designs[185] or Christopher Barney for game designs[186]. On the other hand the less structured analysis in Chapter 4 allowed me to focus on the more interesting aspects and not burden it with the grains about which not much of interest could be said.

5.1.2 Connecting to the Cultural Context

Chapter 3 was designed to provide an insight into the cultural context surrounding work with REDengine 3 if the (working) person in question has access to the engine

[183] Yaneva, Albena. (2009). *Making the Social Hold: Towards an Actor-Network Theory of Design*. In: (2009). *Design and Culture*. vol. 1. no. 3. pp. 273-288

[184] Nystrom, Robert. (2014). *Game Programming Patterns*. Genever Benning.

[185] Trenton, Marcus Alexander. (2009). *Quest Patterns for Story-Based Video Games* [Thesis].

[186] Barney, Christopher. (2020). *Pattern Language for Game Design*. CRC Press.

via modding only. It starts with drawing a line from other storytelling media like theatre, books or cinema to video game storytelling. I could later on show that REDengine 3 and TW3 provide the grain and style to create filmic scenes and a theatre-like "stage" layered with hideable set decoration. In writing about video game storytelling, Jenning's idea of distributed authorship helped to identify quests as artefacts standing right in the middle between player- and developer influence. Then, based on a definition by Jeff Howard, different elements of quests in video games are discussed to be applied in Chapter 4. This connection to (video game) storytelling showed that the story tech designers at CD Projekt RED did not create their means from a vacuum, but built on known means which they scaled to a large production.

In Section 3.1 I differentiated between a template quest, which is the possibly non-serial structure to be instantiated, and the actualised quest, which is the instantiated variant, completed by the player's actions. Moreover, discussing quest goals and motivations and objectives as a way for designers to explicitly prescribe goals were introduced. In Section 3.2 I presented CD Projekt RED as a video game studio which, based on publicly available information, seems to be very aware of their engine and workflow choices. REDengine 3 was characterised as a closed engine in opposition to Nicoll and Keogh's description of the Unity Engine as relational: REDengine 3 is not meant to be a general-purpose, communal game engine and has rather restrictive usage conditions. Chapter 3 additionally showed how CD Projekt RED engages in optimisation of workflows and job positions, tailoring jobs towards specific aspects they want to be hand-made and providing automation for all else. Consecutive work could take up this "field research" investigating the work conditions in CD Projekt RED, for example via analysing further publicly available information such as the strategy update on the "RED 2.0" work transformation[187], or actually interviewing studio members. In general, interviews or surveys could enhance the objective value of my thoughts in Chapter 3.

The thesis then turned to evaluate modding as a necessary context for exploring REDengine 3. Modding was introduced as an activity focused on and motivated by extending, latering and learning about games. The TW3 modding scene as amalgamation of the "collaborative internet network"[188] concerned with modding was shown to have built or adapted a lot of (infra-)structure, including communication

[187] CD Projekt. (2021). *CD Projekt Group Strategy Update*. p. 9. Online: https://www.cdprojekt.com/en/wp-content/uploads-en/2021/03/cd-projekt-group-strategy-update-1.pdf (accessed: 19.03.2023)

[188] cf. Sihvonen, Tanja. Players Unleashed! Modding The Sims and the Culture of Gaming. p. 12

platforms, mod distribution sites and custom tools. Similarly to how investigating the (work) culture at CD Projekt RED could be extended, the TW3 modding scene could be researched more, for example with a focus on how natural language jargon might relate to an object language.

I discussed the power asymmetry resulting from CD Projekt RED's control over REDengine 3 and the modders' wish to use it. In a comparison of motivations regarding modding it showed that modders were mostly motivated intrinsically (e.g. exploration of technology or wanting to enhance and implement their own ideas) while the company is driven by using modding also as a means to proliferate themselves, gain free content and a new talent pool. The case of wghost81's claims shows how the resulting company strategy of bare-minimum technical modding support but endorsement can lead to frustration and frontiers. At last Chapter 3 showed how the missing support from the official side for "proper" modding led to a small group of modders engaging in tool, format and workflow development themselves. This political issue of power asymmetry between modders and engine owner could be compared to other games and engines: companies like Nintendo are even more restrictive in that regard[189], in contrast to, for example, Bethesda Softworks.

5.1.3 Inspecting REDengine 3

With Chapter 4 I dived into an analysis of REDengine 3's grain. I started with an explanation of the questgraph as an architectural structure collecting quest flow, events and conditions in one data asset type. Combined with a quest editor as given in the radish modding tools, this artifice allows for quest designers to work directly and centralised on their formalised object of interest. The matter of the radish quest editor also brought to light consequences from the minimal official modding support: some functionality like the `CQuestCutControlBlock` are not implemented in the community editor and thus disallow the full usage range possible in REDengine 3.

Quest graphs together with the quest description journal feature proved to have a structural analogy to interactive fiction and form a kind of ergodic literature. More generally, the customisation options for quest designers showed that quests in REDengine 3 very blatantly include storytelling in text form, with the entire glossary being a collection of short stories and character descriptions or -histories. The glossary categories, together with how entities are sorted and integrated in game

[189] Brookes, Tim. (2020). *Everything you need to know about Nintendo Switch modding.* How-To Geek. Online: https://www.howtogeek.com/670631/everything-you-need-to-know-about-nintendo-switch-modding/ (accessed: 19.03.2023)

world communities, showed an ambiguous stance as to how beasts, humans and animals relate to each other, incentivising blurry boundaries between them, as promised in the game's marketing.

Section 4.1 and 4.2 exposed moreover how UI spaces and parameters dedicated to specific purposes – like objectives – by their mere existence nudge designers to use them and thereby, in the case of TW3's objective system, also nudge towards a quest design marked by heavy guidance and productivity-measurement. Comparing this aspect with other games showed that CD Projekt RED in general prefers this style of games. Discussing whether modders might be able to change this aspect of the game, allowing for a less-guided experience of Witcher games, rendered the result that this is practically impossible for existent content, but might be possible for mods playing in a new world.

The case of defining rewards demonstrated that the length of the path of least resistance to solving a problem can hinder execution and thus influence design as well. Users of the radish modding tools will avoid giving coins or experience as rewards, since YAML-defined rewards do not support these parameters and XML rewards are not easily integrated with the quest editor. Examples like these show how the grain of REDengine 3 is much more influenced by usability advantages, than by features being present or not. This begs the question, whether future analysis of engine grain should indeed be conducted with a background in user interface/ experience design. The concept behind the path of least resistance also came into play when I observed time scales for various happenings in the TW3 modding scene. REDengine 3 features like initializers or encounters have been inaccessible for quest mod creators, simply because they took or still take a long time to research and implement for the tool creators. The path of least resistance to them is too long and thus hinders them from being reachable. This could be connected to the issues surrounding the power asymmetry, since controlling access to tools also means controlling what modders spend their time on, which can be regarded as another fuel for frontiers between modders and the company, and in the end motivating this frontier in itself.

In the last section of Chapter 4 I introduced the "abstraction hierarchy with end points" as a concept to explain why the overall architecture of REDengine 3 was successful. Identifying various intermediate and smaller domain languages (e.g. REDscript), it became clear that one of the main strengths of REDengine 3 lies in it providing specific languages catered to its end users, allowing them to reach out to certain levels of quantity and quality. In turn, most of these could be created and made accessible because most of them rely on one highly modular and efficient file format, and because their inner complexity was mapped to convenient graphical user interfaces. With the case of the engine's quest and scene design languages I showed how the absence of an editor can indeed be the reason why an extremely powerful technical language is not used as much as it could be.

5.2 Résumé

The conditions of quest modding with REDengine 3 showed to be deeply intertwined to the modding context: the creators' intentions as expressed in the engine's architecture and narrative focus played a role, but were overshadowed by the lack of proper tools to mod TW3, be they by CD Projekt RED or ambitious modders. Design and language as theoretical frame provided a suiting context, but yielded only partial success in providing concrete lenses for analysis. Most notably did neither cultural nor technical conditions really fit into a strongly structured, overarching language – much rather did natural language descriptions lend themselves to capture what is most notable. This leads to the conclusion that structuralism in its more mathematical form can be indeed insufficient to subsume even highly structured objects such as REDengine 3.

In exposing low-level technical details of REDengine 3 and TW3 and interpreting them for instance with Aarseth's ergodicity term, this thesis successfully brought information into a game studies context. However, researching the engine showed that the "foundations" of a game like TW3 are not all interesting for game studies – CR2W as a format is mostly interesting for its language-like character and modularity, but how exactly it is implemented in machine code does not yield much information for the study of games.

Notwithstanding, this information can be interesting for game developers, the second target audience of this work: one could also call this thesis a work from the field of *games engineering*. Understood as a sub-discipline of *software engineering*, games engineering might be explored as a scientific discipline as well: the case of domain languages has already shown how software engineering theory could be applied to game technology. Other concepts from this field such as *software product lines* also have potential and could be pursued further.

Bibliography

Literature, Websites and Posts

Aarseth, Espen. (1997). *Cybertext—Perspectives on Ergodic Literature.* JHU Press.

Aeltoth. (2023). “She should just stop now, the only people that will trust what she says are gullible people who don't understand [..]” [Message]*. The Witcher. Discord.* Online: https://discord.com/channels/597170291021709327/597171985050501150/1071032950826745887 (accessed: 16.03.2023).

Alexander, Christopher. (1967). *The Timeless Way of Building*. Oxford University Press.

A. Simon, Herbert. (1996). *The sciences of the artificial.* 3rd ed. MIT Press.

Barney, Christopher. (2020). *Pattern Language for Game Design*. CRC Press.

Benjamin, Walter. (1916). *Über die Sprache überhaupt und über die Sprache der Menschen*. Online: https://signaturen-magazin.de/walter-benjamin--ueber-sprache-ueberhaupt-und-ueber-die-sprache-des-menschen.html (accessed: 16.03.2023).

Black Tree Gaming Ltd. (2023). *The Witcher 3 mod categories at The Witcher 3 Nexus - Mods and community*. Online: https://www.nexusmods.com/witcher3/mods/categories (accessed: 16.03.2023).

Black Tree Gaming Ltd. (2023). *The Witcher 3 Nexus - Mods and community.* Online: https://www.nexusmods.com/witcher3 (accessed: 16.03.2023).

Bogost, Ian. (2006). *Unit Operations. An Approach to Video Game Criticism.* The MIT Press.

Bogost, Ian; Montfort, Nick. (2009). *Platform Studies: Frequently Questioned Answers*.

Bogost, Ian; Montfort, Nick. (2022). *Levels*. *Platform Studies, a book series published by MIT Press, Ian Bogost and Nick Montfort, series editors*. Online: http://www.platformstudies.com/levels.html (accessed: 16.03.2023).

Bordwell, David; Thompson, Kristin; Smith, Jeff. (2017). *Film Art: An Introduction*. 11th ed. McGraw Hill.

Boudrillard, Jean. (1996). *The System of Objects*. 1st ed. Verso.

Brookes, Tim. (2020). *Everything you need to know about Nintendo Switch modding*. How-To Geek. Online: https://www.howtogeek.com/670631/everything-you-need-to-know-about-nintendo-switch-modding/ (accessed: 19.03.2023)

Cambridge University Press. (2022). *CONSTRAINT | meaning, definition in Cambridge English Dictionary*. Online: https://dictionary.cambridge.org/dictionary/english/constraint (accessed: 16.03.2023).

Cambridge University Press. (2022). DESIGN | meaning, definition in Cambridge English Dictionary. Online: https://dictionary.cambridge.org/dictionary/english/design (accessed: 16.03.2023).

Cambridge University Press. (2022). *LANGUAGE | meaning, definition in Cambridge English Dictionary*. Online: https://dictionary.cambridge.org/dictionary/english/language (accessed: 16.03.2023).

Cambridge University Press. (2022). *PLAN | meaning, definition in Cambridge English Dictionary*. Online: https://dictionary.cambridge.org/dictionary/english/plan (accessed: 16.03.2023).

CD Projekt. (2021). *CD Projekt Group Strategy Update*. Online: https://www.cdprojekt.com/en/wp-content/uploads-en/2021/03/cd-projekt-group-strategy-update-1.pdf (accessed: 19.03.2023)

CD Projekt. (2023). *History - CD PROJEKT.* Online: https://www.cdprojekt.com/en/capital-group/history (accessed: 16.03.2023).

CD Projekt RED. (2022). *A New Saga Begins*. Online: https://www.thewitcher.com/en/news/42167/a-new-saga-begins (accessed: 16.03.2023).

CD Projekt RED. (2023). *CD PROJEKT RED - Award-winning creators of story-driven role-playing games.* Online: https://cdprojektred.com/en/ (accessed: 13.03.2023).

CD Projekt RED. (2020). *Fan Content Guideline*. Online: https://www.cdprojektred.com/en/fan-content (accessed: 16.03.2023). Section 3, Paragraph a.c

CD Projekt RED. (2021). *Highlights from the Path: Mods*. Online: https://www.thewitcher.com/en/news/38447/highlights-from-the-path-mods (accessed: 16.03.2023).

CD Projekt RED. (2023). *MODS (THE WITCHER 3)* [Category]. *CD Projekt RED Forums*. Online: https://forums.cdprojektred.com/index.php?forums/mods-the-witcher-3.69 (accessed: 16.03.2023).

CD Projekt RED. (2023). *Pisarka/Pisarz | SmartRecruiters.* Online: https://jobs.smartrecruiters.com/CDPROJEKTRED/743999704525474-writer (accessed: 16.03.2023).

CD Projekt RED. (2022). *Quest Designer | SmartRecruiters*. Online: https://jobs.smartrecruiters.com/CDPROJEKTRED/743999857102501-quest-designer (accessed: 16.03.2023).

CD Projekt RED. (2022). *REDmod END USER LICENSE AGREEMENT*. Online: https://cdn-l-cyberpunk.cdprojektred.com/redmod_eula_en.pdf (accessed: 16.03.2023).

CD Projekt RED. (2013). *The Witcher 3: Wild Hunt - Killing Monsters Cinematic Trailer*. Online: https://youtu.be/c0i88t0Kacs (accessed: 17.03.2023).

CD Projekt RED. (2022). *User Agreement*. Online: https://regulations.cdprojektred.com/en/user_agreement (accessed: 16.03.2023). Section 8., Paragraph (d)

Costikyan, Greg. (2002). *I Have No Words & I Must Design*. In: Mäyrä, Frans (ed.). (2002). *Proceedings of Computer Games and Digital Cultures Conference*. vol. 1. Tampere University Press.

Crews, Frederick C. (1977). *The Random House Handbook*. 2nd ed. Random House.

Deleuze, Gilles. (1992). *Woran erkennt man den Strukturalismus?* Merve Verlag.

Engell, Lorenz; Siegert, Bernhard. (2012). *Editorial*. In: Engell, Lorenz; Siegert, Bernhard. (2012). *ZMK Zeitschrift für Medien- und Kulturforschung 3/1/2012: Entwerfen*. Meiner. pp. 5-9

Engström, Henrik. (2019). *GDC vs. DiGRA: Gaps in Game Production Research*.

erx. (2018). “there is a good variety of different blocks to choose from, however two VERY important blocks that are missing in [..]” [Message]. *w3 radish tools. Discord*. Online: https://discord.com/channels/416336392692695040/416343412191789087/448227521268547585 (accessed: 17.03.2023).

Friedman, Ken. (2003). *Theory construction in design research: criteria: approaches, and methods.* In: Cross, Nigel (ed.). (2003). *Design Studies.* vol. 24. no. 6. pp. 507-522

Harrasser, Karin. (2018). *Schwächeln. Technikphilosophie, Techniksubjektivität, Unvermögen.* In: Harrasser, Karin; Timm, Elisabeth (eds.). (2018). *Zeitschrift für Kulturwissenschaften. Homo Faber.* ed. 12. no. 2. pp. 149–159.

Hopkins, Tom. (2018). *Witcher 3: How Long it Is and How Many Quests There Are.* Twinfinite. Online: https://twinfinite.net/2018/01/witcher-3-how-long-how-many-quests (accessed: 16.03.2023).

Howard, Jeff. (2008). *Quests: Design, Theory, and History in Games and Narratives.* 2nd ed. A K Peters/ CRC Press.

IGN. (2015). *The Witcher 3 Guide - Side Quest: Towerful of Mice pt 1*. Online: https://youtu.be/SDnnJ4s-SmI (accessed: 24.03.2023).

Ingenito, Vince. (2015). *The Witcher 3 Review - IGN*. Online: https://www.ign.com/articles/2015/05/12/the-witcher-3-the-wild-hunt-review (accessed: 16.03.2023).

Jenkins, Henry. (2004). *Game Design as Narrative Architecture.* In: Wardrip-Fruin, Noah; Harrington, Pat (eds.). (2004). *First Person. New Media as Story, Performance, and Game*. MIT Press. pp. 118-130

Jennings, Stephanie C. (2016). *Co-Creation and the Distributed Authorship of Video Games*. In: Valentine, Keri Duncan; Jensen, Lucas John (eds.). (2016). *Examining the Evolution of Gaming and Its Impact on Social, Cultural, and Political Perspectives*. IGI Global.

Kapp, Ernst. (1877). *Grundlinien einer Philosophie der Technik*. In: Ziemann, Andreas. (ed.). (2019). *Grundlagentexte der Medienkultur*. Springer VS. pp. 45–53

Kent, Emma. (2021). *Cyberpunk 2077 Keanu sex mod removed following CD Projekt warning*. Eurogamer. Online: https://www.eurogamer.net/cd-projekt-red-shuts-down-cyberpunk-2077-keanu-sex-mod (accessed: 16.03.2023).

Keogh, Brendan; Nicoll, Benjamin. (2019). *The Unity Game Engine and the Circuits of Cultural Software.* Springer International Publishing.

Kücklich, Julian. (2005). *Precarious Playbour: Modders and the Digital Games Industry*. In: Neilson, Brett; Rossiter, Ned (eds.). (2005). *The Fibreculture Journal*. vol. 5.

Mareis, Claudia. (2014). *Theorien des Designs zur Einführung*. 1st ed. Junius Verlag.

Martin, Paul. (2018). *The Intellectual Structure of Game Research.* In: *Game Studies.* vol. 18. no. 1

Mello-Klein, Cody. (2017). *The power of handcrafted visual design in video games – storybench*. Online: https://www.storybench.org/the-power-of-handcrafted-visual-design-in-video-games/ (accessed: 16.03.2023).

Mernik, Marjan; Heering, Jan; Sloane, Anthony M. (2005). *When and how to develop domain-specific languages*. In: *ACM Computing Surveys*. vol. 37. no. 4. pp. 316-344.

Momot, Marcin; CD Projekt RED. (2018). *Getting Started / FAQ (UPDATED 21.11.2018).* [Thread]. *CD Projekt RED Forums*. Online: https://forums.cdprojektred.com/index.php?threads/getting-started-faq-updated-21-11-2018.56132/ (accessed: 16.03.2023).

National Geographic Society. (2022). *Storytelling and Cultural Traditions.* Online: https://education.nationalgeographic.org/resource/storytelling-and-cultural-traditions (accessed: 16.03.2023).

Norman, Don. (2013). *The Design of Everyday Things. Revised and Expanded Edition.* Hachette UK.

Nystrom, Robert. (2014). *Game Programming Patterns*. Genever Benning.

Pias, Claus. (2010). *Poststrukturalistische Medientheorien*. In: Weber, Stefan (ed.). (2010). *Theorien der Medien.* 2nd ed. UVK Verlagsgesellschaft mbH. pp. 277-296

Pierściński, Filip. (2022). *Introduction of the Interactive Cinematics in 'Cyberpunk 2077'*. Online: https://www.youtube.com/watch?v=exqPwGIxryI (accessed: 16.03.2023).

Poor, Nathaniel. (2013). *Computer game modders' motivations and sense of community: A mixed-methods approach.* In: (2014). *New Media & Society*. vol. 16. no. 8. pp. 1249–1267.

Prescott, Shaun. (2015). *The Witcher 3 PC review*. PC Gamer. Online: https://www.pcgamer.com/the-witcher-3-review/ (accessed: 17.03.2023).

Razbuten. (2022). *The Problem With Mini-Maps*. Online: https://youtu.be/nmzYRT7LBQs (accessed: 17.03.2023).

RazzDaNinja. (2020). "My longest playthrough was with an Argonian named Skullhunter. I played him in a way that he couldn't care less [..]" [Comment]. *Reddit*. Online: https://www.reddit.com/r/skyrim/comments/emlue0/player_stories/ (accessed: 16.03.2023).

REDengine 3 research contributors. (2022). *W3Speech file. Generic - REDengine3 research.* Online: https://wiki.redmodding.org/redengine3-research/formats/generic#w3speech-file (accessed: 16.03.2023).

rmemr. (2015). "There are many reasons one can think of… Some do this because they are great fans of the previous games [..]" [Post]. *CD Projekt RED Forums*. Online:

https://forums.cdprojektred.com/index.php?threads/im-just-curious-about-mods-newbie-alert.62410/#post-2109367 (accessed: 16.03.2023).

rmemr. (2020). *Trial of the radishes - Trial 1 - Installation at The Witcher 3 Nexus - Mods and community*. Online: https://www.nexusmods.com/witcher3/articles/113 (accessed: 16.03.2023).

rmemr. (2022). "'trials of the radishes - reportedly only three in ten survived the trials' 😄" [Message]. *w3 radish tools*. *Discord.* Online: https://discord.com/channels/416336392692695040/417414398987206676/985503608676167740 (accessed: 16.03.2023).

Ruppert, Wolfgang. (1993). *Zur Geschichte der industriellen Massenkultur. Überlegungen zur Begründung eines Forschungsansatzes*. In: Ruppert, Wolfgang. (1993). *Chiffren des Alltags. Erkundungen zur Geschichte der industriellen Massenkultur.* Jonas. pp. 9-22

Sasko, Paweł; Digital Dragons. (2017). *Life, Love and Quest Design. Anatomy of Quests in The Witcher 3: Wild Hunt*. Online: https://www.youtube.com/watch?v=g5TH9KakBDw (accessed: 16.03.2023).

Schell, Jesse. (2019). *The Art of Game Design: A Book of Lenses*. 3rd ed. CRC Press.

Sihvonen, Tanja. (2011). *Players Unleashed! Modding The Sims and the Culture of Gaming*. Amsterdam University Press.

Tomaszkiewicz, Mateusz; CD Projekt RED. (2014). *The devil is in the details* | *CD Projekt RED's Official Blog*. Online: https://web.archive.org/web/20130116043021/http://cdpred.com/the-devil-is-in-the-details/ (accessed: 16.03.2023).

Tomsinski, Piotr. (2016). *Behind the Scenes of the Cinematic Dialogues in The Witcher 3: Wild Hunt*. Online: https://youtu.be/chf3REzAjgI (accessed: 17.03.2023).

Trenton, Marcus Alexander. (2009). *Quest Patterns for Story-Based Video Games* [Thesis].

Twitter, Inc. (2023). *mod (from:witchergame) - Twitter Search / Twitter*. Online: https://twitter.com/search?q=mod%20(from%3Awitchergame)&src=typed_query (accessed: 16.03.2023).

Van Ord, Kevin. (2015). *The Witcher 3: Wild Hunt Review - GameSpot*. Online: https://www.gamespot.com/reviews/the-witcher-3-wild-hunt-review/1900-6416135/ (accessed: 16.03.2023).

Vinthir; CD Projekt RED. (2022). *Mods and The Witcher 3 next-gen update* [Thread]. *CD Projekt RED Forums*. Online:

https://forums.cdprojektred.com/index.php?threads/mods-and-the-witcher-3-next-gen-update.11110486/ (accessed: 16.03.2023).

wghost81. (2023). *CDPR correspondence on community mods in TW3 NGE – Old Ghost Stories.* Online: https://wghost81.wordpress.com/2023/01/19/cdpr-correspondence-on-community-mods-in-tw3-nge/ (accessed: 16.03.2023).

wghost81. (2023). *Time to wake up, samurai – Old Ghost Stories.* Online: https://wghost81.wordpress.com/2023/01/17/time-to-wake-up-samurai/ (accessed: 16.03.2023).

Witcher Wiki contributors. (2023). *A Towerful of Mice*. Online: https://witcher.fandom.com/wiki/A_Towerful_of_Mice (accessed: 24.03.2023).

Witcher Wiki contributors. (2022). *Modding the UI. Witcher 3 Modding*. Online: https://witcher-games.fandom.com/wiki/Witcher_3_Modding#Modding_the_UI (accessed: 16.03.2023).

Wikipedia contributors. (2023). *CD Projekt*. Online: https://en.wikipedia.org/wiki/CD_Projekt (accessed: 16.03.2023).

Wikipedia contributors. (2023). *Gwent: The Witcher Card Game*. Online: https://en.wikipedia.org/wiki/Gwent:_The_Witcher_Card_Game (accessed: 16.03.2023).

Wikipedia contributors. (2023). *The Witcher 3: Wild Hunt - Wikipedia*. Online: https://en.wikipedia.org/wiki/The_Witcher_3:_Wild_Hunt#Development (accessed: 16.03.2023).

Wikipedia contributors. (2021). *The Witcher Battle Arena*. Online: https://en.wikipedia.org/wiki/The_Witcher_Battle_Arena (accessed: 16.03.2023).

Wikipedia contributors. (2022). *Thronebreaker: The Witcher Tales*. Online: https://en.wikipedia.org/wiki/Thronebreaker:_The_Witcher_Tales (accessed: 16.03.2023).

Yaneva, Albena. (2009). Making the Social Hold: Towards an Actor-Network Theory of Design. In: (2009). Design and Culture. vol. 1. no. 3. pp. 273-288

Video Games

Bethesda Game Studios. (2007). *The Elder Scrolls IV: Oblivion Game of the Year Edition* (v1.2.0416).

Bethesda Game Studios. (2007). *The Elder Scrolls V: Skyrim* (1.1).

CD Projekt RED. (2015). *The Witcher 3: Wild Hunt.* (4.01).

CD Projekt RED. (2015). *The Witcher 3: Wild Hunt - Hearts of Stone.*

CD Projekt RED. (2016). *The Witcher 3: Wild Hunt - Blood and Wine.*

Valve. (2007). *Portal*[190].

Software

CD Projekt RED. (2011). *REDkit* (3.0.1).

CD Projekt RED. (2015). *Witcher 3 Mod Tools* (3.0).

Epic Games, Inc. (2023). *Unreal Engine 5* (5.1).

rmemr. (2022). *radish modding tools* (preview-v2022-12-26).

WolvenKit contributors. (2023). *WolvenKit* (7.2.0).

Images

All images in this work, unless otherwise noted in-place, are screenshots created by the author.

[190] The version of Portal inspected for this thesis is unknown. I accessed the game on the Steam distribution platform in January of 2023.

Declaration

Ich versichere, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen in schriftlicher oder elektronischer Form entnommen sind, habe ich als solche unter Angabe der Quelle kenntlich gemacht.
Ich habe diese Arbeit nicht bereits zur Erlangung eines akademischen Grades eingereicht.
Mir ist bekannt, dass im Falle einer falschen Versicherung die Arbeit mit „nicht ausreichend" bewertet wird. Ich bin ferner damit einverstanden, dass meine Arbeit zum Zwecke eines Plagiatsabgleichs in elektronischer Form versendet und gespeichert werden kann.

Gregor Sönnichsen

Bayreuth, 24.03.2023