

# UC Santa Cruz

## UC Santa Cruz Electronic Theses and Dissertations

### Title

Computer-Driven, Yet Human-Controlled: The Runtime-Editable, All-Digital TTRPG

### Permalink

<https://escholarship.org/uc/item/0r9246gd>

### Author

Joslyn, Maxwell

### Publication Date

2023

### Supplemental Material

<https://escholarship.org/uc/item/0r9246gd#supplemental>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA  
SANTA CRUZ

**COMPUTER-DRIVEN, YET HUMAN-CONTROLLED:  
THE RUNTIME-EDITABLE, ALL-DIGITAL TTRPG**

A thesis submitted in partial satisfaction of the  
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTATIONAL MEDIA

by

**Maxwell Joslyn**

December 2023

The Thesis of Maxwell Joslyn  
is approved:

---

Professor Michael Mateas, Chair

---

Professor Noah Wardrip-Fruin

---

Peter Biehl  
Vice Provost and Dean of Graduate Studies

Copyright © by

Maxwell Joslyn

2023

# Table of Contents

<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>ix</b>
<b>Abstract</b>	<b>x</b>
<b>Dedication</b>	<b>xii</b>
<b>Acknowledgments</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background and Motivation</b>	<b>7</b>
2.1 The Basics of RPG Play . . . . .	8
2.2 Two Gameplay Principles . . . . .	10
2.2.1 Principle 1 . . . . .	11
2.2.2 Principle 2 . . . . .	11
2.3 Rules and Facts . . . . .	13
2.3.1 Definitions . . . . .	13
2.3.2 Precursors to Rules and Facts . . . . .	16
2.3.3 Analyzing Chess in Terms of Rules and Facts . . . . .	18
2.4 Trusting the GM: Exceptions to Principle 2 . . . . .	19
2.4.1 Exception #1: Hidden Procedures . . . . .	20
2.4.2 Exception #2: Allowances for Improvisation . . . . .	21
2.4.3 Consequences of the Exceptions . . . . .	22
2.5 Malleability . . . . .	23
2.5.1 An Example of Malleability . . . . .	27
2.5.2 When Should the GM Exercise Malleability? . . . . .	28
2.6 Quantitative Characteristics of RPG Systems, and Their Connections to the Gameplay Principles . . . . .	30
2.6.1 Fundamental Characteristics . . . . .	30
2.6.2 The Relative Importance of the Fundamental Characteristics . . . . .	31

2.6.3	Depth and Granularity Create Coverage . . . . .	33
2.6.4	Depth Creates Breadth . . . . .	34
2.6.5	Connectedness Creates Immersion and Difficulty . . . . .	35
2.6.6	Immersion and Difficulty Create Tension . . . . .	36
2.7	The Perils of High DCG, and the Promise of Computers . . . . .	37
2.7.1	Increases to DCG Run Into Human Limitations . . . . .	37
2.7.2	Computer Augmentation . . . . .	38
2.8	Why Aren't Computer-Augmented RPGs Already Widespread? . . . . .	41
2.8.1	Mainstream Game Design Space is Constrained by Commercial Realities and User Reluctance . . . . .	41
2.8.2	Fixed-Theme Games Trade Principle 1 for Lowered DCG . . . . .	43
2.8.3	Storygames Don't Require High DCG . . . . .	45
2.8.4	A Comparison with Videogame Progress . . . . .	46
2.9	Research Questions Revisited . . . . .	47
<b>3</b>	<b>Related Work</b>	<b>49</b>
3.1	Hybrid and Augmented Games . . . . .	51
3.1.1	Commercial and Academic Examples . . . . .	53
3.1.2	Observations . . . . .	56
3.2	RPG Support . . . . .	57
3.2.1	Commercial and Open-Source Work . . . . .	57
3.2.2	RPG Support in Academia . . . . .	61
3.2.3	Observations . . . . .	65
3.3	Virtual Tabletops . . . . .	65
3.3.1	VTTs in Academia . . . . .	66
3.3.2	VTTs Outside Academia . . . . .	68
3.3.3	Observations . . . . .	75
3.4	Videogames . . . . .	76
3.4.1	CRPGs . . . . .	76
3.4.2	Games with a GM Mode . . . . .	78
3.4.3	A Curious MUD . . . . .	81
3.4.4	Observations . . . . .	86
3.5	Summary . . . . .	87
<b>4</b>	<b>Speculative Design Goals for Computer-Augmented, High-DCG RPGs</b>	<b>89</b>
4.1	Five Goals . . . . .	90
4.2	Summary . . . . .	96
<b>5</b>	<b>The ROLEPLAYINGGAME System</b>	<b>97</b>
5.1	Features Which Were Out of Scope . . . . .	98
5.2	Implementation Details . . . . .	99
5.2.1	Language and Architecture . . . . .	99

5.2.2	Key Implementation-Level Concepts . . . . .	100
5.3	Game World Content . . . . .	105
5.4	Features Supporting Malleability . . . . .	107
5.4.1	Malleability of Substantive Facts . . . . .	107
5.4.2	Malleability of Taxonomic Facts . . . . .	108
5.4.3	Malleability of Rules . . . . .	109
5.5	Functionality and Design of User Interfaces . . . . .	109
5.5.1	Core UI Elements . . . . .	109
5.5.2	Differential Display . . . . .	116
5.5.3	GM’s Inspector . . . . .	118
5.5.4	Game World Tab . . . . .	118
5.5.5	Gallery Tab . . . . .	123
5.5.6	Explore Tab . . . . .	125
5.5.7	Social Inventory Tab . . . . .	125
5.5.8	Market Tab . . . . .	128
5.5.9	Character Tab . . . . .	128
5.6	Summary . . . . .	128
<b>6</b>	<b>Community Feedback</b> . . . . .	<b>131</b>
6.1	Study Design . . . . .	132
6.2	Connections to Game Design and Software Development . . . . .	134
6.3	ROLEPLAYINGGAME Presents a Compelling Vision for Introducing Computer Augmentation to RPGs . . . . .	135
6.3.1	Complexity and Subgames . . . . .	136
6.3.2	Task-Specific Informational Interfaces . . . . .	140
6.3.3	Generating Game Content . . . . .	143
6.3.4	Testing and Refining Game Designs . . . . .	144
6.4	Computer Augmentation as a Learning Aid . . . . .	145
6.5	Criticism and Concerns . . . . .	148
6.5.1	Feasibility and Costs of Using Computer Tools . . . . .	148
6.5.2	Potential for Distraction . . . . .	152
6.6	Summary . . . . .	153
<b>7</b>	<b>Discussion and Conclusion</b> . . . . .	<b>155</b>
7.1	Reflection and Analysis . . . . .	156
7.1.1	Research Question 1 . . . . .	156
7.1.2	Research Question 2 . . . . .	157
7.1.3	Research Question 3 . . . . .	159
7.2	Future Theoretical Work . . . . .	160
7.2.1	Connections to Fields Beyond Game Design . . . . .	160
7.2.2	Diagnosing and Repairing the Causes of Unexpected System Out- comes . . . . .	162
7.3	Future Technical Work . . . . .	164

7.3.1	Formal Content Schemas . . . . .	165
7.3.2	More Malleability, Culminating in Reprogrammability . . . . .	166
7.3.3	Widespread Use of Differential Display . . . . .	167
7.3.4	Rule Design Languages and Story Sifting . . . . .	169
7.4	Conclusion . . . . .	171
7.4.1	A Defense of a Philosophy . . . . .	172
7.4.2	A Classification of Components . . . . .	172
7.4.3	A Method for Quantitative Analysis . . . . .	172
7.4.4	An Account of Malleability . . . . .	173
7.4.5	A Set of Design Criteria . . . . .	174
7.4.6	An Innovative Artifact . . . . .	174
7.4.7	A Shared Vision . . . . .	175
	<b>Bibliography</b>	<b>177</b>
	<b>A Interview Transcript Excerpts</b>	<b>198</b>
A.1	Interview Excerpts from P1 . . . . .	199
A.2	Interview Excerpts from P2 . . . . .	201
A.3	Interview Excerpts from P3 . . . . .	202
A.4	Interview Excerpts from P4 . . . . .	207
A.5	Interview Excerpts from P5 . . . . .	210

## List of Figures

2.1	A level from <i>Baba Is You</i> . In its current state, because one of the active phrases is “FLAG IS WIN” (top right), the level can be won by touching the flag object (center right.) If, for instance, the player moved the word “ROCK” (bottom right) to where the word “FLAG” is now, forming the phrase “ROCK IS WIN”, then the game rule for winning would change such that touching a rock object (center) would win the level. Source: [60].	26
3.1	Planning a token’s movement in Roll20. The path begins on the right, at the dimmed copy of the token which represents its initial position. Source: [119].	73
3.2	A GM creates a vignette within <i>Divinity: Original Sin 2</i> . Source: [88].	80
3.3	The <i>SID</i> Controller (analogous to a GM) works on a level. Source: [136].	81
3.4	The <i>SID</i> Controller creates a new game object. Source: [137].	82
5.1	After picking the <code>move</code> action, this PC can click any of the green hexes to fill the “coord” parameter shown in the sidebar.	110
5.2	The GM uses <code>toggle-rule</code> to see the rules governing each action. From here, he deactivates the one called <code>within-movement-range?</code> (at bottom.)	111
5.3	Once the GM has deactivated <code>within-movement-range?</code> , the PC’s UI will update to show all non-occupied hexes as green (i.e. targetable.)	112
5.4	The GM’s UI in <code>ROLEPLAYINGGAME</code> , with the nav bar and tab selector at top, sidebar at left, and main area at right.	113
5.5	Before choosing a character with whom to <code>equip</code> an item, the <code>slot</code> parameter cannot yet be chosen.	115
5.6	After the GM has chosen a character for the <code>actor</code> parameter, that character’s valid choices for the <code>slot</code> parameter automatically become available in the corresponding dropdown menu.	115
5.7	Alice (as Arvak) is on the Explore tab, viewing the Cavern zone. She sees bonus information (highlighted in yellow) through differential display.	117
5.8	Birimor is also in the Cavern, but his player, Bob, doesn’t see the bonus text that Alice saw as Arvak.	117



5.9	The GM is hovering his mouse pointer (not shown) over Arvak, causing that PC's battle map avatar to glow purple (bottom right.) This indicates interactable content. . . . .	119
5.10	After the GM clicks Arvak, his name appears in the Inspector (at left.)	120
5.11	Clicking Arvak's entry in the Inspector shows the GM every command that can operate on a PC, and what parameter the PC would fill for that command. Each item in this list is interactive, and will highlight in purple when hovered (not shown.) . . . . .	121
5.12	The GM clicks the item for <b>relocate</b> . This sets the GM's sidebar to begin specifying that command, and fills in Arvak as the <b>target-entity</b> parameter. . . . .	122
5.13	The GM views a location, Murkwood, in his Game World tab. Notice the button which allows him to relocate all PCs to this location. . . . .	123
5.14	The GM views a zone, Cavern, in his Game World tab. . . . .	124
5.15	The GM views locations on the Gallery tab. Zones are colored in red. . . . .	124
5.16	A PC, Arvak, views the Social Inventory. His pane is displayed in green, which is used throughout ROLEPLAYINGGAME to mean "owned by or related to the logged-in user." . . . . .	126
5.17	Arvak filters to include just those items whose names contain the string "bo." Matching items are labeled with a checkmark; non-matching items are dimmed, and labeled with an X. . . . .	127
5.18	A PC, Arvak, visits the Market tab for the location Redport. . . . .	129
5.19	Alice checks the Character tab for Arvak, her PC. . . . .	129
6.1	The <i>Skyrim</i> lockpicking UI. Source: [118]. . . . .	139

# List of Tables

5.1	User commands in ROLEPLAYINGGAME. In running text, commands link back to this table, and are highlighted (e.g. <code>toggle-rule</code> ) to distinguish them from actions. . . . .	102
5.2	Character actions in ROLEPLAYINGGAME. In running text, actions link back to this table, and are highlighted (e.g. <code>move</code> ) to distinguish them from commands. . . . .	104
6.1	Interviewees' professional backgrounds; the number of years they've spent participating in RPGs; and the portion of those years they've spent GMing.	132

## Abstract

Computer-Driven, Yet Human-Controlled:  
The Runtime-Editable, All-Digital TTRPG

by

Maxwell Joslyn

Tabletop roleplaying games (TTRPGs) are set apart from other interactive art forms by the core assumption that their game worlds and game systems can change even during play. This quality, which I call malleability, underpins free choice of action by players, and runtime redesign by Game Masters (GMs.) Despite the importance of this unique quality, on the eve of TTRPGs' 50th anniversary, there are still no computer adaptations of the medium which both retain full malleability, and make meaningful use of modern computers' processing power.

Computer roleplaying games (CRPGs) use modern computers' storage and processing capabilities to present large worlds and deep gameplay mechanics, while keeping large amounts of information manageable with slick interfaces and audio-visual feedback. However, because they are completely mediated by the computer, they give up all malleability.

Virtual tabletops (VTTs) offer somewhat more malleability than CRPGs by only partially mediating gameplay. Nevertheless, much of what would be malleable in traditional play is static during VTT play. Furthermore, because VTTs focus on faithfully recreating game systems and accessories which evolved in an analog context,

their gains from existing as software are mainly limited to networked play and improved graphics.

In this thesis, I present a novel software system which supports complex gameplay mechanics and information-rich user interfaces, while also offering malleability of a game's rules and world at runtime. As part of my justification for developing this system, I give the first comprehensive account of malleability and its importance to the design and play of TTRPGs, and I propose and defend a novel method for quantitatively analyzing the complexity of TTRPG systems.

For Alexis D. Smolensk, who showed me a path.

## Acknowledgments

I thank my advisors, Michael Mateas and Noah Wardrip-Fruin, for letting me work on such a niche project, and for giving me endless counsel during both the hacking and writing phases of creating this work. Any flaws herein are my fault, not theirs.

Andrew Fribush, Kyle Littler, Ryan C. Wright, and Orion J. Anderson generously contributed brain-expanding suggestions to the design, implementation, and analysis of ROLEPLAYINGGAME, and Wynn Tranfield, the McHenry Library's STEM Librarian, discovered a trove of related work that I'd overlooked.

Each player I've GMed for has taught me something, and granted me the satisfaction of seeing my work enjoyed. Thanks to Jeremy, Patrick, Nate, Shea, Jon, Cat, Chelsea, Donald, Nic, Mike, Nicole, and Thorsten; Peter, Stephanie, Ieva, Jonas, Mitchell, Nick, and Cecelia; and above all Rachel, Chandler, Brook, Jñani, and Tori.

My parents have always supported my creative efforts. Among other things, they let a dozen friends and I make a tremendous RPG-session racket in the living room until midnight, once or twice a week, throughout two years of high school. The journey that led to this thesis started there. (Thanks, and I love you both.)

Finally, I would not be devoted to reshaping the roleplaying game if I hadn't met Alexis Smolensk ten years ago. In the course of our friendship, his blog, *The Tao of D&D*, has completely transformed my understanding of what an RPG can offer its players, and how a GM can work to provide that offering. I can't thank him enough.

# Chapter 1

## Introduction

Roleplaying games (RPGs) are a form of intellectual entertainment nearly half a century old. As with the war games they descend from, RPGs are traditionally played at a table, for which reason they are commonly called “tabletop RPGs” (TTRPGs.) In this form, they are accessible to all, requiring no equipment beyond pencils, paper, dice, a set of rules, and the participants’ imaginations.

The first RPG, *Dungeons & Dragons* [55] (AKA *D&D*), defined a large and byzantine set of rules to govern gameplay and suggest many possibilities for player action. Early competitors like *Rolemaster* [23] were cut from the same cloth. However, in more recent years, RPG design trends have changed to emphasize games with smaller, more abstract rule systems which primarily value player improvisation and collaborative storytelling, or which deliberately constrain game content to that which fits within specific fictional genres. In other words, the RPG ecosystem has refocused on eliciting the uniquely human behavior of spontaneous creativity. Meanwhile, complex rule

systems reminiscent of earlier RPGs are increasingly found only in videogames, which take advantage of computers' perfect suitability for storing detailed game worlds and consistently applying nuanced rules.

On the other hand, by being implemented inside a computer, a given videogame offers a fixed range of play experiences. Videogames can use procedural generation to create an enormous variety of in-game content, but only within predetermined domains, such as vehicles or guns; furthermore, even theoretically-endless content can start to feel bland over time. A weapon subsystem in a first-person shooter (FPS) which can generate a trillion different guns will hardly improve a player's experience if he or she can't perceive meaningful variation in the generator's last hundred results<sup>1</sup>. Furthermore, games are generally limited to the content and gameplay that they ship with, and while nowadays, while games can be updated after release with patches and expansions, the time required to ship updates to a game is measured in weeks, at best.

By contrast, because most RPGs' core game loop involves players issuing commands to a human Game Master (GM), the scope of an RPG is negotiable and expandable even while it is being played. The GM, being intimately involved in play and tasked with serving the players, is empowered to alter game rules or the game setting as needed to accommodate player desires and actions. These alterations are typically made in a matter of seconds. Under this arrangement, no aspect of the game need be totally immutable, and the game system can sprout countless details and refinements just as they are needed. The only limit to this process of organic growth is the feasibility

---

<sup>1</sup>Kate Compton aptly refers to this as the "10,000 Bowls of Oatmeal" problem [27].



of employing human brains and analog tools to manage game rules and game state.

Feasibility may be the only limit on RPG scope, but it's a tricky limit to overcome; the difficulty of accurately playing by earlier RPGs' rule systems is probably a key reason for the turn toward less complex games. However, fifteen years of devotion to Game Mastering has taught me that complexity engenders gameplay with valuable characteristics which more abstract games lack<sup>2</sup>. Thus, I believe that it is worth preserving and advancing the art of designing complex RPGs, even though they require extra effort to create and play. The time seems ripe to reinvigorate an earlier culture of RPGs with complex rules, by introducing computers as a central, indispensable support system for play. Such integration has the potential to blend the best aspects of RPGs and videogames into something new: a "hybrid" system that takes advantage of what humans do best *and* what computers do best. The successes of projects that hybridize simulation and performance, such as Bad News [140], have shown that such combinations can be uniquely compelling.

This thesis explores answers to several research questions. **For the purposes of this introduction, the research questions can be understood as follows; their canonical forms will be given in more precise terms later<sup>3</sup>.**

1. How can one design and implement software which processes complex rules, like a videogame, but which permits runtime alterations to those rules, like in traditional RPG play? If both of these goals cannot be achieved simultaneously, what trade-

---

<sup>2</sup>I discuss those characteristics and argue for their value in [Chapter 2](#).

<sup>3</sup>See [Section 2.9](#).

offs can be made to achieve one at the expense of the other, and how should we value each quality when making those trade-offs?

2. How can software support the GM in doing game design; support the players in developing plans and goals for their characters; and support all participants in comprehending game rules and game state?
3. What new frontiers in RPG design can be reached when computers, having been made essential for gameplay management, allow game system creators to surpass the complexity ceiling imposed by human limitations?

No small number of hybrid systems have been built in academia and in industry, for RPG purposes and otherwise. But these systems vary along multiple axes, and the multidimensional design space implied by those axes contains countless hypothetical systems which have not yet been built. The central research contribution of this thesis is the design, implementation, and analysis of **ROLEPLAYINGGAME**<sup>4</sup>, a prototype hybrid system from an under-explored area of that space. The results comprise a vertical slice of client and server functionality offering proof-of-concept support for novel features, including on-the-fly, persistent modification of both a game system and a game world governed by that system. **ROLEPLAYINGGAME** enables users to suspend the evaluation of existing game rule; create new rules; expand game-world taxonomies; create, remove, relocate, and equip game entities; expose information to only those players whose characters possess certain attributes; and efficiently search and filter character

---

<sup>4</sup>**RPG Operation, Learning, and Execution Platform Laden with Advantages for Yielding Infinitely Nitty-Gritty Games that Allow Malleable Evolution.**

equipment<sup>5</sup>. It also provides dynamic GUI feedback on a user’s available commands; the targets available for a chosen command; and the current status of character attributes like location, stats, and equipment.

The rest of this thesis is structured as follows:

In [Chapter 2](#), I explain two game-design principles which I follow while developing and GMing my RPG, and which produce a game containing an aesthetic of self-reliance which I want my players to experience. I give a novel account of malleability, a unique and central quality of the RPG medium, and discuss its critical role in supporting my design principles. Thereafter, I propose a method for quantitatively analyzing the complexity of a game system, and argue that increases in a system’s complexity (as measured by that method) will engender qualities of my desired experience in gaming done under that system.

Armed with definitions of complexity and malleability, and arguments for the desirability of each, I discuss difficulties that arise when playing complex and malleable RPGs. I propose integrating computers with traditional RPGs in a novel fashion which surmounts those difficulties, creating a new form of game that inherits both the malleability of traditional RPGs and the complexity of videogames. I finish with revised versions of the research questions from [page 4](#).

In [Chapter 3](#), I review commercial and academic projects which blend computer mediation, human judgment, and sandbox play in varying proportions. Prior

---

<sup>5</sup>An example of “expanding a taxonomy” would be adding a new race of creatures, which can immediately be used to generate player- or non-player characters. The word “taxonomy” has a particular definition here, which will be covered under [Section 2.3](#).

art in this area spans multiple decades, and includes everything from player-editable videogames, to user studies of RPG participants, to academic work involving custom hardware, to the broad categories of virtual tabletops and RPG support software.

In [Chapter 4](#), I analyze the areas of the hybrid-system design space covered by prior work. I show that existing hybrid systems' focus on certain game designs and play styles have left them ill-suited to capturing the particular flavor of high-complexity game designs which I work on. I establish several characteristics which would make a hypothetical hybrid system more suitable for designing or playing high-complexity game systems.

In [Chapter 5](#), I detail the design and implementation of the ROLEPLAYINGGAME software system, explain its affordances and capabilities, and demonstrate that it possesses characteristics established in the previous chapter as necessary for supporting high-complexity RPGs.

In [Chapter 6](#), I describe a pilot study in which I conducted interviews to solicit Game Masters' feedback about ROLEPLAYINGGAME and related topics. I analyze the interview transcripts for recurring topics, surprising insights, and actionable critiques.

In [Chapter 7](#), I assess the successes and shortcomings of ROLEPLAYINGGAME with regard to answering the research questions of the thesis. I then lay out plans for my future development of ROLEPLAYINGGAME, as well as future exploration of this research area by other scholars. Finally, I summarize the thesis's contributions to the academic literature.

## Chapter 2

# Background and Motivation

As one author put it some fifteen years ago, the theory of RPGs “has reached a basic level of academic acceptance, but exists in a state of chaos” ([58], quoted in [128].) My own research experience leads me to believe this is largely still true. Therefore, to make some of this chapter’s points about RPG design and GMing, I cannot help but draw on my own design and play practices, as some of these ideas seem to have few or no antecedents in academic or popular literature. Despite this, please be assured that this is not an artist’s subjective treatise, and that the technical contribution described in [Chapter 5](#) has something to offer even for an RPG practitioner whose game-design style dramatically differs from mine.

In this chapter, I explain the guiding principles which shape how I design and GM my RPG to produce certain experiences for my players. I give a novel account of malleability, a quality of RPGs which allows them to change over time, and which distinguishes them from nearly all other games. I then propose a novel taxonomic framework

for game system elements which permits quantitative analysis of a system’s complexity. I use that method’s terminology while arguing that increases to a game system’s complexity will engender, in the player’s experience of that system, qualities which are desirable for my design principles. Thereafter, I discuss difficulties that arise when playing complex and malleable RPGs, and which seem to have contributed to a turn away from complexity in RPG design. I propose that said difficulties could be overcome by implementing RPGs as programs, in a fashion which retains their malleability (unlike videogames and other digital RPG systems.) I conclude with definitive versions of the research questions posed on [page 4](#).

## 2.1 The Basics of RPG Play

I assume that the reader has some familiarity with RPGs, so I’ll give only a brief description of play<sup>1</sup>. A group of RPG participants consists of multiple players and one Game Master (GM). Each player controls a player character (PC) as her avatar in the game world<sup>2</sup>. The PCs work as a team to establish and pursue goals. The Game Master does not control a PC; instead, he fills multiple roles related to defining the game and keeping it running. I characterize the GM’s roles as follows:

- **Designer** — The GM establishes the rules of the game system and the contents

---

<sup>1</sup>This description of play is rooted in the classic gameplay conventions of *D&D*, which best matches my style of GMing. Many other RPGs have experimented with different core gameplay loops and different distributions of power between players and the GM. Some of these are covered in [Subsection 2.8.2](#) and [Subsection 2.8.3](#).

<sup>2</sup>I use “he” throughout to refer to the GM, and usually use “she” to refer to any individual players. This avoids ambiguities arising from attempting to use “they” for both the party (a group) and particular players or the GM (individuals.) This usage is emphatically not a statement or assumption about the actual genders of RPG enthusiasts.

of the game world. He uses his taste and game-design skills to continually expand and refine the rules and the world<sup>3</sup>. While the GM is the ultimate authority over the game’s system and setting, he is well-advised to consider players’ opinions on potential changes.

- **Coach** — The GM introduces new players to the rules of the game, and continually answers all players’ questions, especially when they apply to the party’s current situation.
- **Executor** — The GM correctly, completely, and impartially applies all rules of the game system, taking into account all relevant facts about the world.
- **Describer** — The GM presents the game world as perceived by the PCs, using a judicious mix of improvised theatrics and straightforward explanations.
- **Moderator** — The GM intervenes to restore order if a conflict or disagreement amongst the players grows too heated.

An RPG’s core game loop proceeds as follows: the GM describes the PCs’ surroundings and what they can perceive. The players deliberate amongst themselves and ask questions of the GM, then declare what actions they want to take as the PCs. The GM describes how the world changes as a result, and the changed world is used as the “next” set of surroundings. If players never deliberated or asked questions, gameplay

---

<sup>3</sup>Some common goals for this expansion and refinement include inciting particular player behaviors, placing foes and obstacles to test the players’ wits and courage, establishing new rules for future resolution of a scenario that unexpectedly arose during play, or offering players the chance to engage in new in-game activities.

could be summarized as an endless cycle through roughly these steps:

1. GM: “You see X. What do you do?”
2. Players: “We do Y.”
3. GM: “Z happens.”

## 2.2 Two Gameplay Principles

As a GM, I provide a game which offers players the chance to earn a particular kind of satisfaction for themselves. The aesthetic, experiential qualities of my game stem from two key principles I follow both while designing my game system, and while making rulings on the fly during live game sessions<sup>4</sup>. The two principles combine into a style I call “zero-ego GMing.” I am willing to let the players pursue any course of action as they see fit. I guarantee that as we game out that course of action, we will follow logical procedures that are, to the greatest extent possible, worked out in advance. And I will never cheat the players by rejecting the results of the dice.

In one sentence, zero-ego GMing places player-driven action above all else, without diminishing the GM’s dual roles of worldbuilding auteur and impartial executor. The player in a zero-ego game must find it within herself to be a builder and a creator. The zero-ego GM must care deeply about the game system and the game world. Yet,

---

<sup>4</sup>Because I’m treating these principles as idiosyncratic design goals, I don’t feel the need to defend them at length. In particular, this thesis contains no discussion of the precise kind of agency that the principles are meant to foster. That said, there’s a substantial body of academic writing about theories of agency in videogames, and much of that might be applicable to defending these principles or making them more nuanced.



in a seeming paradox, he must not care at all what the players do with the system's rules, in the system's world. In its best moments, zero-ego GMing produces gameplay that appeals to something *deeper* than merely having fun. This kind of gaming asks the players to try imposing their will on the GM's world. It encourages them to be clever, confident, and cooperative. And, in stark contrast to real life, it offers everyone a fair playing field, and the promise that victories, once earned, are for keeps.

### 2.2.1 Principle 1

**Players should be totally free to choose what course of action to pursue.**

The GM must not mandate what players are to do in the game. Players set their own goals as a team. They are free to modify or abandon these goals, and attempt to achieve them through any methods that fit within the existing rules and world; they are also free to argue, within reason, that the rules should function differently. The game is “about” whatever the players want it to be about. I will happily run a game where the players try to become corsairs, found an orphanage, murder innocent men and women, construct a stone tower, research magic, travel to distant lands, or dig an enormous hole. They could try all those things, if they wanted.

### 2.2.2 Principle 2

**The game mechanics are predictable,  
but the game world must be discovered.**

The game system's prescriptions must be followed at all times by all participants, including the GM. To confidently set goals or make plans — to strive to reshape the game world as they see fit — the players must trust that the GM, and the world he presents, will follow the established rules. However, the world that operates by those rules can contain countless unforeseen elements, which the players can only gain knowledge of by playing, and playing well.

Consider this analogy. When you drive your car, you want the steering wheel, the tires, the windshield wipers, the blinkers, the headlights, the seatbelts, the accelerator, and all the other parts of the vehicle to function as they should. A part's failure to perform as expected degrades your ability to operate the car within predictable bounds, which may have severe negative consequences — because, even if the car is perfectly predictable, the environment in which you drive the car is not.

You can never know all the road conditions in advance. Will there be traffic? Will a child run into the street? Will construction block your usual route to work? You want your car to be predictable, because roads are unpredictable. You want to eliminate the sources of error or accident that you *can* control, because there are some that you *can't*.

In my style of RPG, the game system is the players' car, and the game world is the road. The players deserve to know exactly how the game mechanics function at all times. However, the GM has free reign to make the game world uncertain and unknown, filling it with obstacles the players must assess and overcome to reach their goals. Finally, although Principle 1 is absolute, Principle 2 must admit of a few practical

exceptions. Describing those exceptions in [Section 2.4](#) will first require classifying the components of an RPG system, which is the subject of the next section.

## 2.3 Rules and Facts

For the purposes of this thesis, I consider each element of an RPG system to be one of three things: a rule, a taxonomic fact, or a substantive fact. This section defines those terms. I use the analytical lens of rules and facts throughout this thesis, especially in this chapter. [Section 2.4](#) relies on the definitions of rules and facts to illustrate a tension between my design principles and the act of GMing (which in turn demonstrates why a player must trust her GM, and how a GM can, in part, build that trust by adhering to my design principles.) My discussion in [Section 2.5](#) of the inherent mutability of RPG worlds and systems also requires having previously established definitions for rules and facts. Most importantly, [Section 2.6](#) proposes rule- and fact-based metrics for quantifying an RPG system's complexity, then uses those metrics to argue that increasing a game system's complexity will improve its potential to structurally endow gameplay with desirable properties, such as emergence and tension. Said metrics are referred to extensively throughout the rest of the thesis.

### 2.3.1 Definitions

**Rules** are procedures delineating how the game world can change over time (such as through player action), and how the effects of such changes are to be calculated.

I refer to the result of following a rule as its **outcome**<sup>5</sup>. In a typical game system, the bulk of the rules govern the actions the PCs can take, and when those actions can be taken. Examples include how the attack action is resolved, how poisons affect a character, methods of manufacturing magic items, how long it takes to put on armor, and how frequently new equipment will be available for purchase at a market.

**Facts**, or declarations about the world, fall into two subcategories. The first of these is **taxonomic facts**. These collectively describe the vocabulary of conceptual objects which are the fundamental, discrete ingredients of the game system. Taxonomic facts also declare key-value attributes possessed by conceptual objects, and the ranges of values which attributes may take on. The above examples of rules made reference to actions, attacks, poisons, characters, magic items, armor, equipment, and markets; these keywords, and the details of each, would all be defined by taxonomic facts. When analyzing a game system, taxonomic facts must be defined first, since they establish the terms used when defining rules and declaring substantive facts. For convenience, I refer to the whole collection of a game system's taxonomic facts as that system's **domain model**.

The second fact subcategory is **substantive facts**. These declare the actual details of the game world content classified by taxonomic facts. Representative substantive facts include the name of the planet on which the game takes place, the existence of a kingdom on that planet, the extent of the kingdom's lands, the name of its monarch,

---

<sup>5</sup>An outcome might be anything, such as a change in game state, a decision about whether a certain action is allowed, a Boolean value, or a number. Since both facts and rules vary so widely between game systems, a precise analysis or taxonomy of rule outcomes is beyond the scope of this thesis.

the monarch's current and maximum hit points, the trade goods the kingdom produces, key events in the kingdom's history, and so on. Note that while a given game session might not lead to the introduction of new rules or taxonomic facts, it is guaranteed to produce new substantive facts, firstly because many rule outcomes will prescribe alterations to substantive facts, and secondly because a GM can hardly avoid either describing new pieces of game world content, or updates to existing content<sup>6</sup>. Substantive facts also represent the current values of all the entity attributes declared by taxonomic facts.

Proposing that game world content is part of the overall game system may seem strange, but I think it's reasonable. First, I'm not the only one to think that a setting should be part of a system, because there are game systems which include the details of a preordained game setting that is assumed for play: *Pendragon* [154] is an especially good example<sup>7</sup>. Second, although there are also game systems<sup>8</sup> intended for generic use with a variety of game worlds — which include lots of rules and taxonomic facts but no substantive facts — the quantity and distribution of substantive facts which a GM uses to flesh out his world will be just as consequential to a player as the rules and taxonomic facts are. If the reader and I were to GM for the same group of players, using the same generic game system with different homebrew game worlds, our

---

<sup>6</sup>For more on how and when such updates take place, see [Section 2.5](#).

<sup>7</sup>On the one hand, published game systems can't possibly address every detail about a game world, so the GM is still free to fill in the gaps as he sees fit, or reject some or all of the substantive facts included in the game system. On the other hand, while it would be easy to customize the premade setting of *Blades in the Dark* [56] (which is deliberately described in broad strokes to allow wide latitude for GM creativity), it would take quite a bit more work to do the same with *Pendragon*, which is narrowly focused on gamifying the experience of being a knight in Arthurian Britain.

<sup>8</sup>Examples include *D&D*, *GURPS* [70], and *Savage Worlds* [61].

players would be justified in making different judgments about what actions might be considered prudent or foolish, and what plans they ought to make. For these reasons, I believe that the term “game system” should be taken to include the game world — and that what I called generic systems above are, in my terms, not complete systems at all, but rather chassis waiting to be fitted with a world.

### 2.3.2 Precursors to Rules and Facts

The terms “rule” and “fact” in this thesis are not wholly my invention. First, the basic sense of how I use those terms is borrowed from their usage in logic programming<sup>9</sup>, in which facts are propositions, and rules define valid inferences that can be drawn on the basis of those propositions. Second, Richard Bartle’s *How to Be a God* [11] — a technical and philosophical treatise on the design of “realities” (MMO game worlds) — includes much discussion of “physics,” or the rules by which a (game) world operates. Bartle proposes a tripartite analysis of the functionality of a world’s physics, and the categories of his analysis resemble my division between game system rules, taxonomic facts, and substantive facts:

A reality isn’t a free-for-all space where anything goes: each one adheres to a set of physical rules individual to it that define its characteristics. Collectively, these characteristics are a reality’s nature. The rules themselves are its physics. ... The physics of a reality affects its nature in three ways:

---

<sup>9</sup>Thanks to Michael Mateas for suggesting I employ these terms.

1. It determines what the components of the reality are. Everything in Reality is either matter, energy or (quite possibly) both. [= taxonomic facts]
2. It manifests these components in an ongoing configuration. The atoms in Reality that comprise your body were doing other things a thousand years ago. [= substantive facts]
3. It determines how the current configuration is transformed to give a new configuration. In Reality, gravity encourages objects to move towards each other, meaning their positions tend to change dynamically. [= rules]

The consequences of a reality's physics are the nature of that reality. Gods of a reality have power over its nature, so that's equivalent to saying they have power over its physics. What, then, in practice, can they do?

Well, a god of a reality has the ability to change any and all aspects of physics for that reality. If you were the god of a reality made up of bottles of soda water, you could decide to allow it also to contain ping-pong balls. If you were the god of a reality made up of sounds, you could spontaneously create (or, if your composition skills aren't great, recreate) a symphony within it. If you were the god of a reality with colours, you

could make their saturation automatically cycle every Sunday.

The non-god inhabitants of a reality can perform none of these activities. They can make changes to the reality's configuration if its physics allows them to do so, but they can't change the physics itself.

— Bartle [11], p. 9–11

### 2.3.3 Analyzing Chess in Terms of Rules and Facts

The following rules-and-facts analysis of chess should be enough to illustrate how to begin doing the same for a game system with much higher complexity, such as an RPG.

Beginning with taxonomic facts, “pieces” and “the chessboard” would be conceptual objects. “Squares” of the chessboard ought to be their own individual conceptual objects as well (albeit ones that can only exist relative to a specific chessboard.) I’ll also include the “turn” as a conceptual object, not least because the *en passant* capturing rule for pawns only applies if a particular board state was reached exactly on the previous turn, through a particular move. Finally, not wanting to assume anything, I will include “player” as a conceptual object, too.

Pieces would have a “color” attribute, with a value of either “white” or “black,” and a “type” attribute, such as “pawn” or “rook.” The chessboard might have one attribute per square, or a “squares” attribute naming a two-dimensional array of squares; the difference is immaterial here. Squares themselves would have a “color” attribute. Turns have the attributes “player” (whose turn it is), “piece,” and “move.”



As for substantive facts, since a chessboard resets after each round<sup>10</sup>, it seems defensible to call each round of chess a self-contained game world. Each new turn taken by a player can be thought of as appending to a list of turns which is part of the substantive facts. The current state of the chessboard is also a set of substantive facts, with one fact per square indicating whether that square is empty or occupied, and if so, by what.

Finally, there are rules. One rule would describe the procedure for placing a piece of a given color onto a fresh board at the beginning of the round. Another, higher-level rule would stipulate that beginning a round requires following the aforementioned procedure until all pieces are placed onto the board. There would be a rule or rules for how each piece can move, and others about capturing, capturing *en passant*, castling, promotion, being in check, verbally declaring “check,” checkmate, draws, and so on. There would also be rules about how many pieces can be moved per turn, whether or not a player can take back a turn, and whether a turn must be made within a time limit, as in competitive chess.

## 2.4 Trusting the GM: Exceptions to Principle 2

Having established the terminology of rules and facts, I can now explain some exceptions to the first half of Principle 2, which stipulates that the game system should always be fully known to the players. I’ve identified two cases where it is acceptable to

---

<sup>10</sup>I use “round” here to indicate a game of chess because I want to avoid confusion with terms such as “game system.”

keep the players in the dark about certain rules or facts. Both cases illustrate why it is crucial that the players trust the GM if a game under the two principles is to function.

Given that I've previously claimed that following the principles will build the players' trust in their GM, it may seem paradoxical to say that player trust is required for a game using those principles to operate. If the way to "bank" player trust is by following Principle 2 in all situations other than the ones described below, yet trust is required to play under the two principles, then how can a game ever get off the ground? The solution to the paradox is to further stipulate players must, at least at first, trust the GM on faith — without having yet seen whether he will uphold the principles<sup>11</sup>.

#### **2.4.1 Exception #1: Hidden Procedures**

The first exception to Principle 2 arises when rules or facts concern aspects of the world that the players have not yet discovered for themselves. If the players haven't learned of certain world details, it's up to the GM's discretion whether they are entitled to know how the game operates with regard to those details.

For example, suppose that my game world contains a secretive organization called the Phoenix Guardians. Although it is common knowledge that Phoenix Guardians possess esoteric magic formulas of great power, the party has never encountered a Phoenix Guardian, and doesn't know what they do, how their organization operates, how to become one, or even how to find one.

As GM, I am not obligated to tell the players how to become Phoenix Guardians,

---

<sup>11</sup>In my experience, if a player distrusts her GM, it usually stems from having had her initial trust-on-faith broken by the GM's failures to follow the principles.

nor reveal any other knowledge about them, until such time as the players have earned that knowledge through in-game actions. That said, I am obligated to have some idea of how the Phoenix Guardians operate — what their secret magic does, how someone becomes a Phoenix Guardian, etc. — and treat those concepts as facts and rules which I conform to like any others. In other words, I need to pre-commit to rules or facts that cannot yet be revealed to the players. If a verifiable guarantee of that commitment was required, I could mail myself a letter and leave it unopened, or share with the players an encrypted text file containing the pre-committed information.

In practice, I don't need such technologically-backed guarantees, because I have my players' trust — a trust which I continually earn and renew by always respecting established game rules and facts. Exception #1 to Principle 2 shows how the “predictable rules” portion of Principle 2 earns the player trust that is required to operate by the “unknown world” portion. A steadfast adherence to the first half of Principle 2 for all matters of which the players are aware will assure the players that the GM will also adhere to it for matters of which they are not aware.

#### **2.4.2 Exception #2: Allowances for Improvisation**

The second exception to Principle 2 comes up when a rule includes a clause that defers to improvisational GM decisions. I call such a clause an **escape hatch**: a section of a rule that is deliberately designed to require the GM to make a judgment call while resolving it. Allowing this exception makes it permissible to design bounded ranges of unknowability in rules otherwise fully known to players, granting the GM

a measure of flexibility in handling circumstances that defy being detailed ahead of time. Because escape hatches offer an explicit transition from procedural resolution to case-by-case judgment, they exemplify game processes that “only humans can do,” as mentioned on [page 1](#).

For instance, RPG design guru Alexis Smolensk uses a rule which states that if a character takes 15 or more damage in one hit, she also receives an injury [150]. All injuries have certain mechanical penalties in common, such as decreased movement speed and absorbing a certain amount of healing (which would otherwise restore hit points.) However, the injury rule also states that the GM can, and must, invent the precise nature of the injury, because circumstances that can lead to 15 damage at once are so varied that a random table would produce unsatisfying and illogical results. Furthermore, when appropriate for the injury in question, the GM may impose additional mechanical penalties beyond the standard ones. Both the invention of the injury and the discretionary imposition of penalties are escape hatches.

Upholding the established rules and facts, and expanding the rules to cover as many scenarios as possible, is how a GM gets players to trust him when their current situation is altered by facts or rules of which they are unaware. Like Exception #1 to Principle 2, Exception #2 needs player trust to function.

### **2.4.3 Consequences of the Exceptions**

If a GM exercises goodwill, taste, and restraint while operating portions of the game system unknown to players, or when employing the discretion afforded to him by

escape hatches, he will have no trouble in getting players to accept the ultimate escape hatch: the GM’s prerogative to create or change rules and facts on the fly. A GM who wishes to wield that ability to reshape the game on the fly must prove to his players that he can be trusted. That ability, which is the key to RPG’s unparalleled potential to provide endless gameplay in an ever-growing scope, is treated in detail in the next section.

## 2.5 Malleability

As previously mentioned, a GM can extend and change an RPG at “runtime” or “playtime,” i.e. as the game is being played. I call this quality **malleability**. I chose my own term to describe the flexible and fluid nature of RPG systems because I’ve been unable to find much treatment of this topic in the academic literature. The closest antecedent I’ve found is from Richard Bartle [11], in a discussion of strategies for implementing a world’s “physics<sup>12</sup>.” Though Bartle is talking about programming a computer to represent a game world, his description of an “interpreted physics” (i.e. game system), the third item below, bears a strong resemblance to how GMs can wield malleability<sup>13</sup>.

There are three main ways a virtual world’s physics can be so embodied:

1. Hard-coded. The physics of the virtual world’s reality never changes and can be implemented directly in an efficient systems-programming

---

<sup>12</sup>See [Section 2.3](#).

<sup>13</sup>Throughout this thesis, I will use “substantive malleability” as a shorthand for “malleability of/for substantive facts;” the same goes for rules and taxonomic facts.

language. Only the current state of the reality needs to be stored as data.

2. Soft-coded. The physics can't itself be changed, but some of its properties can be. Gravity's strength (represented as a number) could be increased or reduced, or even made negative, but gravity will always have the same functionality. It can't be made to apply only to liquids, for example.

3. Interpreted. The physics can be changed even while being applied. The laws of physics are objects of the virtual world, and could themselves be subject to laws of physics if the designer so decided.

— Bartle [11], p. 70

Malleability is indispensable for upholding Principle 1 in my own game. Since I invite my players to pursue any course of action they can think of, I have to be prepared to create rules and facts that harmonize with existing rules, make sense within the existing world, and satisfy players' desires to know whether, and how, some new plan might be achieved.

Judicious application of this GM ability is also essential for running a game under Principle 2. A game system which is expanded to cover unforeseen circumstances offers its participants increased predictability when those circumstances reoccur<sup>14</sup>. Ev-

---

<sup>14</sup>It's acceptable for rules made on the fly to be minimal, and only treat details of the scenario that gave rise to them. What's important, for a game governed by my Principles 1 and 2, is that the GM makes decisions in a jurisprudential manner — thereby also implicitly promising that, should the same circumstances arise again, the newly-invented rules will be applied again. However, not every GM

ery time I make a new rule or fact, or (with a justification) alter the game system from how it worked up until that moment, I am solidifying the player’s conception of how the game world functions and of their place in it. This increases players’ confidence that I will not accept or reject their chosen actions purely on a whim.

Even GMs who don’t subscribe to my principles constantly make use of malleability. I’ve already covered the unavoidability of making changes to substantive facts while a game is being played (Subsection 2.3.1.) Furthermore, the sentiment of “no matter how much preparation I do, the players always do something I didn’t expect” is so widely shared by GMs as to be a trope. An effective response to the unexpected requires the GM to be able to swiftly decide on the unforeseen actions’ results (or create a rule to determine them), then smoothly incorporate those results into the game world in a logical manner. Both of those GM maneuvers exercise the quality of malleability.

Malleability is nearly unique to RPGs, but I’ve found a few examples of it in other types of games. For instance, if a player of the card-based drinking game *King’s Cup* draws a certain card, she is empowered to establish a new rule that will be followed for the rest of the game by all players. A more elaborate example of malleability outside RPGs can be found in the board game *Nomic*<sup>15</sup>. This game is noteworthy not

---

would agree that judgments about novel scenarios should become part of the ongoing game system. Some GMs prefer to make one-off, situation-specific judgment calls with no promise of reuse, rather than committing to establishing a new rule (however provisional.) GMing in this way is usually called “rulings, not rules.” When a GM is totally uncertain about how a rule ought to be designed, and wants to signal to players that they should not depend on a procedure working the same way in the future, “rulings, not rules” is a reasonable move. However, GMing this way as a matter of course means that GM responses to novel scenarios neither equip the players with more information to exploit, nor improve the depth of the rule system (see Section 2.6) As such, following Principles 1 and 2, I avoid one-off rulings. Rather, I accept that rules created by exercising malleability will always start off limited and incomplete.

<sup>15</sup>The author of *Nomic* published the game’s rules as an appendix to a book [160], but it was first described in the literature eight years earlier by Hofstadter [67], who had read the book in draft form.

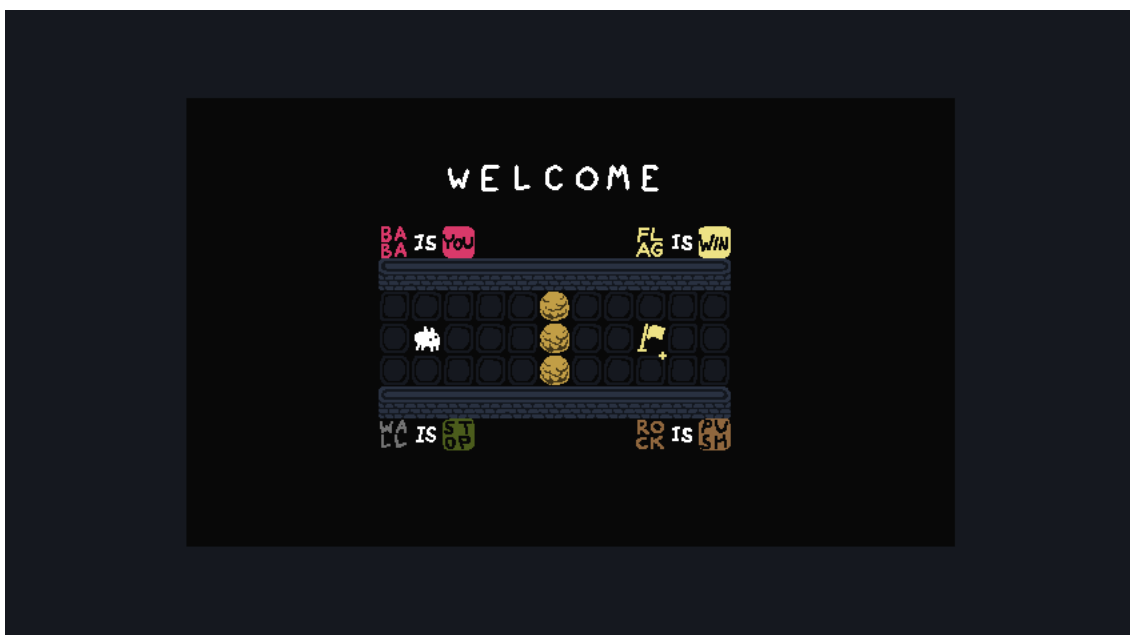


Figure 2.1: A level from *Baba Is You*. In its current state, because one of the active phrases is “FLAG IS WIN” (top right), the level can be won by touching the flag object (center right.) If, for instance, the player moved the word “ROCK” (bottom right) to where the word “FLAG” is now, forming the phrase “ROCK IS WIN”, then the game rule for winning would change such that touching a rock object (center) would win the level. Source: [60].

only because its rules exhibit near-total malleability, but also because *Nomic* gameplay is about malleability and how best to employ it. Its rules lay out procedures for altering the rules by voting, and play itself consists of artful jockeying to approve or deny additional rules through that voting process<sup>16</sup>. Finally, the videogame *Baba Is You* [59] uses a limited form of malleability as a player-facing game mechanic. To solve the game’s puzzles, players activate or deactivate rules and alter facts by pushing objects that represent verbs and game-world nouns into new configurations (Figure 2.1.)

Examples like these are very much the exception to the rule in non-RPG game

<sup>16</sup>An adaptation of *Nomic* is used to power the player-driven democratic legal system of *A Tale in the Desert*, a long-running MMO [158] that is still active as of 2023.



design. On the other hand, in RPGs, malleability is ever-present, allowing the GM to alter how the game functions, adapt to player activity, and increase the number of possible scenarios the game system can address. It can be deployed at any time to address shortcomings or flaws in the structure and function of the system.

The assumption of game system malleability, and the GM's right to exercise it, means that there are no hard limits on the scope or substance of an RPG's gameplay. An RPG system can always be adapted to make its existing gameplay more detailed, or expand it with a new type of gameplay altogether. Virtually all other types of game impose predefined limits on player actions, or the environment in which those actions take place. RPGs need do neither, and one reason I believe strongly in Principle 1 is because it leans into this deep-seated and defining characteristic of the RPG as a distinct medium.

### **2.5.1 An Example of Malleability**

Suppose my game has rules for riding on horseback, but I have not yet gotten around to creating rules for fighting on horseback. Further suppose that a player asks me what would happen if she mounted her horse, rode at an enemy, and attacked with a sword. I may not be able to create fully general or fully satisfying rules to govern that scenario, but I must create some rules. Something as simple as “apply the charging rules as if you were making a charge on foot, then add an attack bonus which scales with your horse's speed” would be enough to start.

By adding this nascent rule for horse combat, I have just expanded the range of

things a player can predict about the operation of the rules. They can strategize about how best to deploy a horse charge, and how best to defend against enemies performing such a charge against them. Most importantly, we didn't have to stop playing to expand the game system with this new content. An RPG's scope of player activity can change dynamically, and those scope changes often take almost no time or effort to enact.

By contrast, if the designers of the videogame *Mount and Blade* [38] had somehow included horseback fighting mechanics for almost every weapon, but had run out of time to do so for swords, they wouldn't be able to fix it without doing some more programming, adding new animations, and possibly altering the AI so that it knows it can use swords on horseback. Finally, all that work can be released to players as a patch.

Those tasks require considerable expenditures of time and effort, as do all aspects of videogame development. Thus, videogames often have mechanics and scopes of possible player activity very similar to previous games. As we will see in our later survey of related work, adaptations of roleplaying games which substitute the computer for a GM must sacrifice most or all malleability. Computer tools/assistants for human GMs to run RPGs fare little better.

### **2.5.2 When Should the GM Exercise Malleability?**

Consider how a car manufacturer designs next year's model, then puts the car through real and simulated stress tests. As the testing reveals defects in the design, the manufacturer must try to understand why the defects have appeared, and then change

the design to remove them. Similarly, the GM must watch how his rules shape player behavior, and adjust the rules if they are not having the desired effect. Determining whether the rules are achieving what is desired is, naturally, only possible if the GM has a philosophy of design that determines what desirability means.

The aforesaid rule adjustments must be done transparently and openly, with good taste as a designer, and with respect for the players — who are voluntarily engaging in a difficult hobby which is information-rich, collaborative, analytical, and imaginative; and who have spent hours of play earning steadily more powerful characters. Thus, if players have committed themselves to a mounted expedition on horseback, for the GM to radically change the horseback riding rules would be a violation of Principle 2. He should assume that the players understood and took into account the existing horseback riding rules while making their plans.

Malleability is not *carte blanche*. The GM should at minimum put forth his rationale for changing the rule, and hear arguments for or against. He is still the final arbiter of the game's design, but once the players have played for a while and invested themselves in the GM's game, they've at least earned the right to speak their minds, with the assurance that the GM will listen.

## 2.6 Quantitative Characteristics of RPG Systems, and Their Connections to the Gameplay Principles

In this section, I establish provisional metrics for measuring, in terms of rules and facts, what I've previously called the “complexity” of an RPG system. To explain why these metrics are meaningful, I then show how altering an RPG system to increase these metrics can induce a desirable increase in the intensity of certain elements of the player experience.

### 2.6.1 Fundamental Characteristics

In developing my design theory, I've identified three simple numeric characteristics that quantify game system complexity: depth, connectedness, and granularity. For convenience, throughout this thesis I refer to this trio of complexity metrics as **DCG**.

**Depth** signifies the grand total of rules and facts which a game system defines. It is the simplest and most important characteristic, as will be discussed in [Subsection 2.6.2](#).

The **connectedness** of a given rule signifies the count of its direct dependencies on facts or other rules, as measured by the number of inputs required to determine the rule's outcome. For example, the rule for resolving the “attack” action in a system like *D&D* depends on the stats, class, level, and current and maximum health of both the attacker and the defender; the terrain on which each of them is standing; the at-

tacker’s armor, and the defender’s weapon; the attacker’s skill with her chosen weapon; and miscellaneous buffs, debuffs, and special characteristics which either directly alter the chance of a successful attack, or alter one of the other characteristics already mentioned. Assuming that each character has three stats relevant to the attack action, I count 19 inputs (before considering buffs, debuffs, and special characteristics, which in most game systems are myriad), giving this rule a connectedness of 19.

Finally, **granularity** signifies the number of distinct values that an element of game state can take on. For instance, if game characters have an optional attribute called “character class,” the granularity of that attribute is  $N + 1$ , with  $N$  being the number of classes defined by the system (plus one for the case of not having a value for that attribute, which is often the case for NPCs.) Taxonomic facts directly influence granularity because they establish the set of acceptable values for each attribute an in-game entity can possess.

### 2.6.2 The Relative Importance of the Fundamental Characteristics

Of the three fundamental characteristics, depth is king. A rule creates relationships between inputs and outcomes, and a fact declares details about a game world. Without a healthy stock of both, there’s nothing to play.

Connectedness has an influence on how players subjectively experience gameplay, because changes to this characteristic alter the difficulty of comprehending the game world and game system (see [Subsection 2.6.5.](#)) However, without rules (i.e. depth), there are no rule inputs with which to measure connectedness. Furthermore, although

a game system with low connectedness will lack its drivers of game difficulty, a game's design can make up for that by requiring players to manage a large number of discrete rule subsystems.

Granularity is, in isolation, the least necessary of the three fundamental characteristics<sup>17</sup>. A GM can run perfectly good RPG sessions with a system in which game-entity attributes have little or no granularity. Suppose that all PCs and NPCs are warriors; the only weapons are sticks (dealing moderate melee damage) and stones (dealing low damage in melee or at range); and the setting consists entirely of a square plain 10 miles on a side, dotted with deposits of power-up crystals that improve a character when consumed.

The limitations of this game world, especially its physical geography, would hamper the longevity of such a game, but I could certainly run at least a year of weekly gameplay sessions with it<sup>18</sup>. Given procedural rules for attacking, sneaking, befriending and angering NPCs, extracting crystals, and building earthworks and other fortifications, the players could create and pursue numerous goals. Further possibilities would arise as I established answers (i.e. rules and facts) for questions that immediately spring to mind (e.g. "How quickly do crystals grow? Do new deposits occasionally form? How long has this bizarre area existed? How many NPCs are here besides the PCs, and how long have they been here? What are those NPCs' personalities?")

---

<sup>17</sup>The next two sections will show how granularity and depth, taken together, allow the derivation of two additional characteristics with vital importance to games run by Principles 1 and 2.

<sup>18</sup>It's no coincidence that this simplistic world resembles the setup for a war game. *D&D* did, after all, grow out of its creators' decision to play iterative war games where units that survived battles increased in power for subsequent ones.

The players' tastes, and their assessments of this world as it evolves, would lead to the emergence of behavioral patterns. They would decide whether to fight, retreat, hide, spy, infiltrate, use guerilla tactics, bargain, broker alliances, betray friends, or launch good old-fashioned frontal assaults. No granularity necessary.

### 2.6.3 Depth and Granularity Create Coverage

**Coverage** denotes the number of possible game scenarios that can be resolved using a particular game system. For the purposes of this thesis, a **scenario** is a tuple  $(S, L)$ , where  $S$  is a game state, and  $L$  is an action that a character wants to perform which is allowed in  $S$ . For a game running under Principles 1 and 2, higher coverage is desirable: it means players can make more predictions about what will happen if they get themselves into a situation with such-and-such characteristics. In an open-ended or “sandbox” game (such as Principle 1 dictates), the total number of possible game scenarios is infinite. No matter how many scenarios you enumerate, you could add another by taking a previously-counted one and tweaking the position of a single character, item, or other element of the game world. Luckily, the infinitude of possible scenarios is not a problem for considering how DCG contributes to coverage.

An increase in depth which expands a game system with additional actions will increase the number of scenarios which are possible under that system. Increases in granularity add additional ways to “tweak” a scenario, as mentioned above<sup>19</sup>. Therefore,

---

<sup>19</sup>For instance, suppose that the taxonomic facts of a game system declare that there are five possible kinds of magic potion. When a sixth kind of potion is added, then, to any set of five imaginable scenarios which differ only in which original potion is present, one can add a sixth scenario containing the new kind of potion. And that's just the simplest case. Consider a scenario in which a character declares

if some system X has higher degrees of depth and granularity than some system Y, then X will generally have higher coverage than Y.

#### 2.6.4 Depth Creates Breadth

**Breadth** is the potential for emergence in gameplay, as game events unfold in unexpected ways. If  $W$  is a game world state, then the breadth of that state equals the branching factor of  $W$ , denoted  $B(W)$ .  $B(W)$  is the number of possible states  $W'_1, W'_2, \dots, W'_N$  that can be reached from  $W$  through the application of game rules. In theory, one could measure the breadth of a whole game system by averaging the branching factors of all possible game states. In practice, as with coverage, the calculation of the true breadth of a game rules system is infeasible — but, also as with coverage, I have no need for a precise accounting of a system's breadth. It's enough for my purposes to demonstrate that increases to DCG will also increase breadth.

Breadth increases with depth: higher depth means more applicable game rules for any given state, and thus more possible transitions out of that state. For a game run according to Principle 2, higher breadth is desirable for a designer because it has the effect of increasing the unpredictability of the world. That unpredictability is itself desirable because it drives the players to seek control over their surroundings — which they can best pursue by exercising organization, clear thinking, tactics, and creativity; cleverly wielding their character strengths and capabilities; and building up mastery of

---

she's drinking a potion while her inventory contains three of them. Choosing each potential potion at random with replacement, and with order not mattering, there are 35 possible combinations before the taxonomy update, and 56 possible combinations after it. The update added 21 new scenarios.



the game system and its implications.

### **2.6.5 Connectedness Creates Immersion and Difficulty**

I've shown how increased coverage improves players' ability to understand how the game system will function for them while engaging with the game world on their own terms, which accords with Principle 1 and the first half of Principle 2. I've also shown how increased breadth accords with the second half of Principle 2, by enlarging the space of unknown or unpredictable situations in which the players can find themselves.

The two derived characteristics above made no mention of connectedness, but the final member of the DCG trio has important effects on the feel of gameplay all on its own. When it matters whether you attack with a sword or a laser, whether your opponent is wearing armor, how fast you can move, where your allies are positioned, or whether your stomach agreed with what you ate for breakfast — then, if you wish to play well, and succeed at your aims, you will pay attention to each of those factors as closely as you are able. This goes hand-in-hand with Principle 2.

If the players want to succeed at the goals they have set, and the game rules are highly connected, then developing a plan to reach some goal will require the players to consider numerous input factors. As the average connectedness of the game system's rules increases, the players will run up against limits on how well they can assess every input factor, forcing them to make hard choices. “Engage with this area of the world, or ignore it? Attempt to influence these strange creatures, or stay out of their way? Doggedly pursue an elusive enemy, or accept that he's escaped?” Debating answers to

questions like these is the essence of planning toward a goal, and high connectedness increases the effort required to carefully consider such questions — while simultaneously eroding the attraction of analysis paralysis by ensuring that exhaustive exploration of all options is too costly in real-world and in-game time to be feasible.

### 2.6.6 Immersion and Difficulty Create Tension

A further consequence of designing an RPG system with high DCG is the infusion of practically every player decision with some potential for emotional tension.

For instance, suppose that the party has been fighting a pitched battle. A PC who primarily uses a bow for combat has been firing steadily. When an enemy charges her, the player realizes she is down to her last arrow. At this moment, emotions will run high for the whole gaming group, because it is easy for all participants to comprehend the stark facts of the character's desperate situation, and imagine the potential consequences of making a poor choice.

I contend that a game which does not require the players to explicitly track pieces of ammunition *cannot* produce the *specific* flavor of tension which the players in this example would experience. A game which abstracts away ammunition management with a rule like “after every shot, roll d20. If you get a 1, next time roll d12 ... once it's a d4, if you get a 1, you're out of arrows,” has far less potential to create a wide spectrum of subtly-varied emotions in the players. Numerous rules and facts (D), high interdependence between rules and facts (C), and high variability of facts (G) combine to give high-DCG RPG systems a richly textured expressive range. Or, from an alternate

perspective, a high-DCG RPG system constitutes an enormously fecund procedural content generator<sup>20</sup>.

## 2.7 The Perils of High DCG, and the Promise of Computers

### 2.7.1 Increases to DCG Run Into Human Limitations

High depth means more rules to remember and apply. High connectedness means more processing-time overhead in the application of a given rule. High granularity means more choices per attribute. Each of us has limits on the size of his working memory, the length of his attention span, his capacity for repeatedly making difficult analytical choices, and his reserves of energy. These human factors must impose some upper bound on what quantities of DCG are feasible.

As a game's DCG increases, it becomes harder for players to play. And a given increase in difficulty for the players corresponds to an even larger increase in difficulty for the GM, given the much wider scope of his roles, and given the fact that any decision that players have to make on the behalf of PCs is a decision that the GM will have to make many more times on the behalf of NPCs. Higher difficulty for players and for the GM also means that the game can take longer to play, necessitating longer play sessions, an acceptance that less ground will be covered per session, or more vigorous

---

<sup>20</sup>For more on the notion of considering RPGs as procedural content generators, see [54]. It might also be profitable to consider the output of RPG rules to be the space of player emotions and behavior they induce, rather than the space of game states they can represent.

efforts (primarily by the GM) to commit the rules to memory.

Finally, there are areas of game rule design space which would be infeasible to include in a game, no matter how quick-witted the GM or how patient the players. A rule with a connectedness of 1,000 input factors is unlikely to be employed at any gaming table, no matter how satisfying its effects on player behavior.

### **2.7.2 Computer Augmentation**

Enter computers. They have functionally infinite storage capacity, and a perfect recall of whatever they store. They can calculate on large numbers of operands at speeds millions of times faster than a human. And they are tirelessly indifferent to the number of inputs or manipulations needed to perform a calculation. With these characteristics, computers could be used to massively raise the upper bound on DCG which human factors would otherwise impose.

Some kinds of general-purpose productivity software are readily applicable to expanding the limits of RPGs. Wikis can be used to create a hypertext rulebook with infinite pages, freeing game designers from the need to limit the depth of a game system just so that it can fit into a hard-copy book. Spreadsheets can be used to implement custom forms for evaluating rules with high connectedness and granularity, with a user-editable field for each input factor, and dropdown lists, data validation procedures, or simple textual reminders to ensure that what the user enters is an appropriate value for the input in question.

But those are just RPG-specific applications of two widespread and flexible

pre-existing types of software. Now imagine a purpose-built RPG app, which allows for creating, editing, and examining all rules and facts. Such a system would offer interactive interfaces that reduce the mental decision-making load imposed by games with high DCG, and would support malleability in a way that typical computer games simply cannot. Such software could do more than improve our RPGs; it could transform our play experiences and the expectations we have for them.

The punchline here is that I and others have already built custom computer tools to enhance our games. For instance, Alexis Smolensk [149] has created a economic simulator system which establishes raw material production quantities for all the areas in his game world, determines a price per unit for each material, and then combines those raw material prices into calculations for hundreds and hundreds player-purchasable trade goods, in such a way that the prices of all items will be different when purchased at different settlements.

I've built my own version of such an system, following in Smolensk's footsteps, and it's made a drastic increase to the quality of my game. The system spurs players to take into account the geography of the game world and the relative local prevalence of certain manufactures and industries when equipping their PCs, or members of the PCs' retinue. This in turn drives them to set goals such as traveling to distant lands with exotic resources, targeting NPC merchant caravans for heist operations, attempting to engage (legally or illegally) in profitable business dealings, and seeking out locations in the world which have yet to be exploited for their material resources. In other words, with this computer program fulfilling the "predictable system" aspect of Principle 2,

the players are emboldened, and more thoroughly embrace the freedoms of Principle 1.

None of this would be feasible if I didn't have a software system capable of doing millions of calculations. Crucially, the system runs so quickly that it can still support malleability: if a player asks to buy an item which doesn't already exist on the equipment table, it's the work of a minute or two to program it in and rebuild the system with the new item available.

My goal in describing this system to the reader is not to claim that computationally-assisted gaming will allow for a more "objective" game with no room for a GM's judgment. As long as RPGs admit of malleability, there will always be a need for GM judgment to create necessary rules and facts on the spot. I bring up the trade system and its effects on my home campaign as a case study. I consider it evidence that if a GM implements a game subsystem as an external artifact, and commits to obeying that artifact's results, his players will come to trust that what the artifact says, goes. This is true whether the artifact is a set of dice or a computer program — but one of those tools can lend far more horsepower to my quest to transcend human limitations.

## 2.8 Why Aren't Computer-Augmented RPGs

### Already Widespread?

#### 2.8.1 Mainstream Game Design Space is Constrained by Commercial Realities and User Reluctance

RPGs have traditional tools for managing certain aspects of gameplay. The most common ones are maps, dice, and pencil and paper. Each game uses these tools differently, if at all, but we can generalize away from individual games by characterizing what each tool offers to the game designer.

A map is a physical tool for understanding a physical environment and tracking precise positioning within that environment. The existence of a map affords the inclusion, in a game's design, of rules which take precise positioning as input. One frequently-used type of map is a battle map, which portrays a location at a scale small enough to support tactical combat rules.

Dice are a tool for random number generation (RNG.) RNG affords the design of rules that give a known range of results, but where the result of a single application of the rule is unknowable until runtime. Other RNG tools include spinners, drawing from a shuffled deck of cards, and drawing from a box full of labeled chits (a particularly old-school method predating widespread availability of plastic dice.)

Pencil and paper together form a tool for storage and recall of arbitrary game state; this affords the design of rules which take various aspects of the current state of the game as input, i.e. which have some measure of connectedness. Even more

importantly, pencil and paper can be used to write<sup>21</sup>, which means that they can be used to “implement” RPG rules. By defining procedures, drawing up tables, or sketching diagrams, the rules of the game can take shape as fast as the GM can write. Obviously, pencil and paper readily support malleability. The two components of this tool are also widely available and extremely cheap.

Since pencil and paper can be used to replace the other two most common tools (by sketching a map, or create paper chits used for random draws), it is the only one of the three which is fundamental. Accordingly, I assume that traditional play can be and is performed only with pencil and paper.

Given the advantages of pencil and paper, it’s only rational that commercial game designers assume that gamers want game systems requiring no other equipment. It would be exceedingly difficult to profit from a rule set which mandated GM and/or player use of custom software.

Assuming for the moment that gamers really are reluctant to use computers for gameplay, that reluctance is also rational, given the current landscape of computer-support tooling for RPGs. Most existing computational interventions on (or augmentations to) RPG play suffer from severe limitations in their rule design language, data modeling, interface affordances for comprehending complex game states, user-friendliness, and malleability (especially this last one.) If gamers perceive computer tools as insuffi-

---

<sup>21</sup>I must assume that all participants are literate. It is difficult to imagine an illiterate person successfully playing in any complex RPG, although she might possibly succeed if her gaming group was extremely patient. This is an instance of a more general observation that physical mechanisms for game input usually assume certain normative standards for players’ physical and mental capabilities. Sadly, addressing this issue in more detail is beyond the scope of my thesis.



ciently user-friendly, powerful, or malleable, they will not accept those tools, regardless of any other advantages they offer.

The consequence of these two groups' rational assumptions is that commercially-available RPGs currently inhabit only that small section of the space of possible rulesets which is realizable using good old-fashioned pencil and paper. The way to break this cycle is to build game-design software which is directly aimed at those GMs who are seriously committed to their role as designers. Once it becomes possible to use such software to redesign an RPG even while it is being played, as defined by malleability, then hopefully more gaming groups will be willing to embrace computer mediation of game rules.

Note that it's not entirely true that gamers are reluctant to introduce computing to their games. [Chapter 3](#) explores a wide range of computational support tools for RPGs, some of which have seen widespread adoption. My analysis of these tools finds them lacking in features needed to support my preferred style of high-DCG, Principles-1-and-2-driven RPG. However, gamers with different tastes have adopted, *en masse*, virtual tabletop (VTT) software, as well as amateur- and professional-made tools which enable users to look up the rules of a game system, roll large numbers of dice, or create maps using a fixed or fluid library of visual assets.

### **2.8.2 Fixed-Theme Games Trade Principle 1 for Lowered DCG**

Some game systems are designed with deliberate limits on DCG which restrict the actions and events of gameplay to fit within an established fictional genre. These

games are commonly said to be “about” playing out (or subverting) the tropes and narrative arcs of that genre. I refer to such games as having a **fixed theme**. Examples of fixed-theme games include *Call of Cthulhu* [122], *Night’s Black Agents* [66], *Blades in the Dark* [56], *Pendragon* [154], and *Dogs in the Vineyard* [7].

In my own game, it’s not a problem that events might go in all manner of directions with no clear theme, but that’s because I have embraced a picaresque, naturalistic unfolding of events which, like real life, doesn’t have a genre. Gamers who are more interested in exploring the tropes, themes, narrative arcs, or character archetypes of a specific genre will rationally prefer a game system with an appropriate fixed theme.

There are other rational reasons to prefer fixed-theme game systems. Some such systems employ a small set of abstract, thematic mechanics to resolve all possible game scenarios<sup>22</sup>. Both the amount of up-front effort needed to design such a system, and the amount of ongoing GM effort required to run one, are much lower than those needed for a high-DCG system. For instance, when abstract mechanics fail to cover a novel scenario, the GM can make do with ad hoc judgments, rather than slow down the game’s tempo with an approach more resembling jurisprudence. That said, even a high-DCG game requires less effort to run when it has a fixed theme, because the theme is ever-present to guide the GM as he designs content or fields player questions.

For players and GMs who are willing to limit the scope of gameplay for the sake of a fixed theme, there is no need for a game system to tend toward ever-higher

---

<sup>22</sup>But by no means all. *Call of Cthulhu* is considerably less abstract than the other three games named above, and *Pendragon*, a game about Arthurian knights, is known for having rules and facts addressing such topics as the thirteen aspects of chivalric conduct, the acceptability of marrying into specific social classes, and the chance of a horse surviving the winter while stabled.

DCG. Without the pressure of running up against human limits on managing DCG, these gamers, in turn, are unlikely to feel a need for computational mediation.

### 2.8.3 Storygames Don't Require High DCG

No discussion of RPG-adjacent design practices would be complete without touching on the **storygame**. Aaron A. Reed notes that the term has multiple conflicting meanings in popular use ([128], p. 18); for this thesis, I use the term to refer to games in which the goal of play is to produce a satisfying story. This usually entails rules that generate or stipulate plot fragments, or which are designed to spur players into improvising elements of a narrative.

Apart from that commonality, storygames are a diverse bunch. They may or may not have a gameplay loop that resembles the one I outlined in [Section 2.1](#). Some storygames contain RPG elements (*Apocalypse World* [8]), and some do not (*Fiasco* [112].) Some storygames have a fixed theme (*My Life With Master* [32]), and some do not (*Microscope* [133].) Even when a storygame has RPG elements, it will usually also embrace a trend in RPG design which transfers, from the GM to the players, some of the power to act as ultimate arbiter of the game system and world. Some storygames don't even have a GM at all, a condition referred to as being "GMless" or "GMful" (in the sense that when all players are empowered to act in a GM-like capacity, the game is "full" of GMs.)

I have nothing against storygames, but every example I'm aware of has relatively low DCG. I believe this is because storytelling (or story generation) is itself the

object of play, which leads storygame designers to see a wealth of rules and facts as reducing the number of opportunities players will have to creatively shape the developing story.

#### 2.8.4 A Comparison with Videogame Progress

The RPGs available today, whether released as commercial products or created by amateur enthusiasts, are not substantially higher in DCG than the first RPG (the original “White Box” edition of *D&D* released in 1974.)

This stands in stark contrast to the evolution of videogames over the same time period. The computers of 2023 have literally billions of times more processing speed, random-access memory (RAM), and hard disk space than the computers of the 1970s. Videogames have consistently been at the forefront of this hardware revolution, as game designers pushed new systems to their limits<sup>23</sup>.

Today’s videogames fully employ the capabilities of today’s computers, but RPGs have barely even begun to incorporate them — and even where they have, such incorporations have been limited by designer and gamer expectations that comport with the limits of pencil and paper, as well as by the sheer difficulty of implementing software which can be altered at runtime as easily as a traditional GM can alter the prose which defines a traditional game system.

Therefore, one reason to pursue the computational mediation of RPG play is

---

<sup>23</sup>It’s beyond the scope of this thesis to offer a full discussion of the link between advances in videogames and advances in computer hardware. Interested readers are referred to the discipline of platform studies, as well as [184] and [104] (neither of these are chiefly concerned with those links, but both touch upon it in numerous ways.)

to drastically expand the space of feasibly-realizable RPGs. I wish to claim for the RPG some of the success of the videogame, by melding the best parts of both art forms. From the videogame, we take full computational mediation of gameplay. From the RPG, we take malleability and the human GM. The following chapter will show that a veritable army of hybrid systems have tackled this problem from different angles, and that for every necessary software component, at least one system has met the bar for fully realizing RPG capabilities in the computer. Even RPG-type total malleability has been achieved by game developers<sup>24</sup>. The problem now is to coherently implement such capabilities in one common system. Then, game designers and academic researchers could experiment with more radical enhancements to RPG design made feasible by an all-digital game.

## 2.9 Research Questions Revisited

To conclude this chapter, I'll revisit the research questions given in [Chapter 1](#), and restate them using the theoretical terms of art which I've introduced throughout this chapter. **The new versions below are the canonical research questions for this thesis.** Phrases which are substantially changed from their original formulations are marked in bold.

1. How can one design and implement software which **processes a high-DCG RPG system**, without sacrificing **the malleability of rules and facts** which is **central** to RPG play? If both of these goals cannot be achieved simultaneously,

---

<sup>24</sup>Once.

what trade-offs can be made to achieve one at the expense of the other, and how should we value each quality when making those trade-offs?

2. How can software support the GM in doing game design, **including the on-the-fly redesign that characterizes malleability**? How can it support the players **of a Principle-1-based game** in developing plans and goals for their characters? How can it support all participants in comprehending **the rules and taxonomic facts of the game system, and the substantive facts of the game world**?
3. What new frontiers in RPG design can be reached when computers, having been made essential for gameplay management, allow game system creators to surpass the **DCG** ceiling imposed by human limitations?

This chapter has described my background in RPGs and my motivation for pursuing computer-driven operation of game systems. In the next chapter, I review works which embody answers to research questions related to mine.

## Chapter 3

### Related Work

There is a staggering number of works which can be said to blend RPGs and computers. Some of these use custom hardware, while some are implemented entirely in software. Some have a GM, some are fully mediated by the computer, and some offer a mix. Some are intended as accessories to an otherwise analog game; still others are intended to replace all analog action. The design space which these works inhabit is multidimensional, which has engendered a remarkable variety and diversity of approaches to combining computing with analog gaming.

The bulk of this chapter is a survey of projects which apply computing specifically to RPGs<sup>1</sup>. Many of these belong to the burgeoning field known as “RPG support,” examples of which include virtual tabletops (VTTs), dice rollers, improvisation aids (e.g. name generators), digital rulebooks, campaign management apps, and map-making programs. I’ll also briefly trace the evolutionary path from RPGs to computer

---

<sup>1</sup>I deliberately refrain from proposing any firm taxonomy for these projects. That would be a valuable direction for future work, but is not necessary for my purpose of reviewing capabilities and limitations.

roleplaying games (CRPGs) to contextualize a discussion of CRPGs which offer playing modes designed to mimic traditional RPG play. Finally, I'll cover academic research studying the relationship between RPGs and computing, such as surveys of GM interest in, and desires for, computational gameplay support. This broad survey will establish the boundaries of the space of previous computational interventions into RPG play and design practices.

I also use this chapter to present no small number of prior works which aren't directly related to RPGs, but nevertheless have relevance to design decisions needed to make RPG-specific software. These projects from further afield include computer-augmented board games, social simulations, drama managers, companion apps for story and board games, interactive digital narratives (IDNs), augmented games, "tangibles," and mixed-initiative design tools. Projects in these areas have features which could enhance RPG-specific software, but which have rarely or never surfaced in that domain. Some of them also offer gaming experiences with higher degrees of DCG than the RPG-specific projects; thus, these are important pieces of prior art for the goal of using computers to present gaming experiences with higher DCG than can be offered by analog games.

In this chapter, I aim to stake out the current dimensions of the design space of RPG support software, and provide evidence for my claim that high-DCG gaming is not well-supported in that space. I also inform later consideration of appropriate software architectures and features for applications that implement both high-DCG RPG systems, and the user interfaces for designing and playing them.



### 3.1 Hybrid and Augmented Games

**Hybrid game** is a catch-all term used in academia to any technological enhancement of an existing analog game, or any game made possible by technology but not fully mediated by it. In [77], Kankainen *et al.* remind the reader that “technology,” in this context, does not always mean computers, for the history of hybrid gaming stretches back further than the history of computing:

There were already games, such as *Code Name: Sector*, using microchips with physical board games released in late the [sic] 1970s, and during the eighties such games as Milton Bradley’s *Dark Tower* gained some popularity. Even prior to microchip technology, since 1910s [sic], there seem to have been board games which had electric devices, such as small lamps, merged with physical game boards, e.g. *Electra*. These games can be considered as ancestors to modern hybrid board games, which utilize the ever increasing computing power of mobile devices.

— Kankainen *et al.* [77], p. 1–2, in-line references omitted

Kankainen *et al.* go on to discuss the fluidity of the terminology in this area. Their claim that “there seems to be no rigid hypernym encasing [all forms of physical-digital games]” (p. 3) matches what I have seen in the literature<sup>2</sup>.

Bergström and Björk [13] present a distinct but related characterization of games which include computation as part of their functioning. These authors argue for a

---

<sup>2</sup>For a thoughtful discussion of the shortcomings of a simple “physical plus digital” or “physical plus electronic” characterization of hybrid games, see the rest of [77].

broader view of how computers can be used to realize what they call augmented games. They say “computers are today mostly used to **encase** and **mediate** games” ([13], p. 3, emphasis mine), rather than supplement human-centered gameplay not focused on computer interactions. They go on to showcase six games which employ computing for processing, display, and storage of game-affecting information, but which do not center the computer itself as the sole interface or tool of gameplay.

Bergström and Björk’s analysis culminates in a typology of features (*ibid.* p. 16–18) which cogently characterize the design space of augmented games. These features are:

- “player-agreed” vs. “artifact-encased” game logic
- “limited” vs. “rich” audiovisual content
- “fluid” vs. “fixed” game content
- “manual” vs. “automatized” excise
- “low-effort” vs. “high-effort” modification of rules
- “low-effort” vs. “high-effort” modification of game state

I include the Bergström and Björk typology here only so the reader can keep the above design space tradeoffs in mind while reading the rest of this chapter. With so many pieces of related work for me to cover, it is beyond the scope of this thesis to thoroughly analyze any of them through this typological lens. However, rigorously assessing some subset of the works in this chapter would be a valuable contribution to

the design and development of augmented games, even (or especially) if that assessment tested the value of this typology for classifying games that could arguably fall outside of Bergström and Björk’s original definition.

### 3.1.1 Commercial and Academic Examples

False Prophets [102] combines custom software, a tabletop equipped with light sensors, a projected map grid, and optical-signal-emitting play pieces to deliver a gaming experience encompassing both computer-mediated and person-to-person interplayer interactions. The system reads game piece positions to supply hidden information to each player: a task that would otherwise be impossible without a GM, who could impartially reveal the correct information. False Prophets exemplifies the application of computers to tasks which would otherwise require a non-player moderator, without de-emphasizing or eliminating desirable a characteristic (face-to-face play) by opting for full “artifact encasement” (in Bergström and Björk’s terms.)

Weathergods [9] describes a board game with a digital screen displaying the game scenario, which gives visual feedback as players move their physical pieces atop it. Most such feedback is purely digital, but in the case of a game piece that marks hidden treasure, a light projector illuminates the piece such that only the player who controls the piece can see the change. A feedback mechanism by which a physical system triggers digital processing, which in turn causes change in the physical system, is so mundane as to be forgettable when driving a car or operating factory equipment — yet, it is unusual in the world of game design, and worth noting.

These key early works in hybrid gaming explore concerns such as how to provide effective game state feedback, and how far to take computational mediation of player action<sup>3</sup>. That said, their physicality or tangibility is relatively minor: they would not change appreciably if they had been implemented as pure videogames. Both systems are also devoid of malleability: each implements a single game in which player actions, and the scope of gameplay possibilities, are strictly limited by preset rules. Thus, they are of limited relevance to the development of systems aimed at capturing the advantages of RPGs.

TVViews [109] is a multi-user interactive digital tabletop which can track RFID-tagged physical objects, and which has novel “puck” rotary controllers which users employ to select options from radial menus. Different game rule systems can be implemented before gameplay time, but malleability is absent.

The ReacTable [76] is a multi-touch input and display surface<sup>4</sup>. Academic publications on the ReacTable focus on using it as a novel method of live music production, and its creators fostered a community around this use case [91]. However, a demo video [181] based on the bachelor’s thesis of a member of the research group [182] also shows its potential as an RPG support tool, with live creation and editing of scenario maps.

TrueSight [124] is a set of 3D plastic pieces for building representations of RPG scenario terrain. The built terrain can be connected to a laptop for use by a GM, and

---

<sup>3</sup>Another work in the same vein as these two projects include Asterocks [187].

<sup>4</sup>Don’t confuse the ReacTable (also spelled “Reactable”) with the “InteracTable,” part of the STARS system (see below).

the pieces will light up in patterns corresponding to GM-selected information, such as a magic spell's area of effect.

Tilt Five [175] is an augmented reality system, sold to consumers for tabletop use, which projects holograms visible through a pair of accompanying glasses<sup>5</sup>. Bryan Versteeg has built a light projector system connected to an ordinary tablet device, which lets him “paint” dynamic lighting onto a physical board-gaming scene [180].

All the systems discussed so far require participants to gather around a table, as in traditional boardgame or RPG play. STARS ([100], [99]) presents a different, more comprehensive notion of augmented gaming. STARS is a multimodal gaming augmentation platform, comprising the “InteracTable” (a touch-sensitive table), headsets worn by the players, a loudspeaker, player PDAs, a projector system, and an overhead object-tracking camera. Available features include dynamic display (such as for fog of war) and RFID-based position tracking. Integration of all these components is particularly impressive given how early this system was developed.

The creators of STARS recognize the importance and potential of digitizing game rules: “[C]omplex game rules are put into the digital domain, so that an accurate simulation of the game world can be realized without slowing down the game flow.” ([100], p. 3) However, while the STARS system can import different code-based rules systems, its capabilities don't extend to design: neither rules nor taxonomic facts are malleable while the system is in use.

---

<sup>5</sup>See the online comment chain at [113] for criticism of Tilt Five, including responses from the system's creators.

### 3.1.2 Observations

Where Weathergods, False Prophets, TViews, and TrueSight primarily differ is in their scope. Weathergods and False Prophets each offer one game only. TViews allows playing out a range of roleplaying scenarios, within the limits of its included software assets.

TrueSight is more general still. Not only could users play out any scenario which is compatible with the pre-programmed rules, the physical pieces could also be put to satisfactory use as RPG props sans software, where the rules proved insufficient. This demonstrates how physical equipment with a computational component can gracefully degrade into ordinary, non-computationally-assisted play equipment<sup>6</sup>. Furthermore, TrueSight could be used outside normal gameplay, as a memory or measuring aid for those learning the game.

These works show a trend of using object tracking (whether through RFID tagging or camera-based detection) as a way to bridge the gap between physical tokens and digital processing. Object tracking affords a degree of malleability for substantive facts: each repositioning of a piece is equivalent to declaring that some game world fact has changed. However, the projects in this section universally lack malleability for rules and taxonomic facts. In some cases, this stems from the system arriving early on the scene of augmented gaming; in other cases, for the creators having spent their “complexity budget” in other areas. For instance, it’s understandable that STARS

---

<sup>6</sup>The term “graceful degradation” goes at least as far back as 1987, where Herlihy and Wing [62] used it in the context of software systems remaining useful even when unable to work optimally (e.g. when under a heavy processing load.)

doesn't incorporate runtime rule redesign, because that work was focused on using custom hardware to integrate multi-modal interactions into a gaming experience.

All in all, prior work in hybrid gaming has focused on systems which combine custom software with tangibly-interactive custom hardware, including RFID tokens, LEDs, push-buttons, and terrain tiles. From my survey, I conclude that when systems require custom hardware, they preclude offering the malleability of a traditional RPG. This conclusion informs the new design criteria I present in [Chapter 4](#).

## 3.2 RPG Support

The term “RPG support” can cover everything from mapping tools to character generators to dice rollers. I've done my best to survey a representative sample of the subcategories which get lumped under this label.

### 3.2.1 Commercial and Open-Source Work

D&D Insider, and its successor D&D Beyond, are official RPG-support tools from Wizard of the Coast, the owners and designers of *D&D* since 1997 [5]. D&D Beyond originated with a different company<sup>7</sup>. After eventually being acquired by Wizards of the Coast, it has received more internal support and development than Insider ever had, which has resulted in Beyond's features being a superset of those found in Insider.

Conclusions about RPG support tooling which might be reached from an analysis of Be-

---

<sup>7</sup>D&D Beyond was originally developed by Curse. The tool was later acquired by Fandom [17], then sold to Wizards of the Coast for USD \$146.3 million [186].

yond are, accordingly, a superset of those which could be drawn from Insider. Therefore, I only discuss the former here.

D&D Beyond offers access to an online compendium of *D&D* game rules; tools for creating PCs, monsters, and encounters (premade collections of monsters and other obstacles); a 3D character designer; a limited form of virtual tabletop; and even a Twitch streaming integration, inspired by the popularity of “real play” RPG podcasts such as Critical Role [165]. Beyond also lets GMs create and upload (private or public) custom content which fits into the predefined categories the software can recognize, including PC races and classes, monsters, magic spells, and special abilities.

Initiative.sh [121] is an open-source web application which supports both consultation of *D&D 5e* game rules, and a form of inspirational PCG similar to that of Imaginarium ([68], discussed below):

To generate a random thing, simply describe it. This can be simple, as in typing the name of the thing you’re looking for, or more complex, describing details of that thing.

```
> a human boy named Roger
```

**Roger** human child, he/him **Species:** human

**Gender:** masculine

**Age:** 2 years

**Size:** 2’9”, 27 lbs (small)



— Initiative.sh `help` command output, showing a sample CLI interaction

[121]

The most unique characteristic of Initiative.sh is that it only offers a command-line (CLI) interface<sup>8</sup>. While its interaction affordances are thus bounded by the limits of CLI design, Initiative.sh embodies a novel answer to this question: “How far we can reduce the amount of user effort needed for a game support tool to return useful information?” Or, as the README for its code repository states:

[I]nitiative.sh’s design philosophy is to minimize the time and effort between the question (“Is there a blacksmith nearby?”) and the answer (“Yes, it’s called Frosthammer & Sons, and Fenrik Frosthammer is at the forge.”).

— Initiative.sh README [120]

**Dice rollers** replace the rolling of physical dice with the results of random number generation (RNG) algorithms. These are most frequently employed when a large number of dice need to be rolled quickly, or when the number of possible die results doesn’t match the number of faces on any of the usual physical dice. Auxiliary functionality includes storing rolls (and descriptions) for reuse, e.g. “Orc Damage Roll: 1d4+1”, or calculating and visualizing probabilities of obtaining certain results on the dice.

A **worldbuilding wiki** (e.g. LegendKeeper [93]) is a piece of software similar to MediaWiki [103] (the software behind Wikipedia), but enhanced with features that

---

<sup>8</sup>More specifically, a web browser simulation of a command-line interface.

allow the user to track family trees, timelines, and other aspects of a fictional world. Use of these tools is not limited to GMs: “Our users have created sci-fi campaigns, planned epic novels, organized LARPs, and scaffolded open-world games” [92]. By contrast, **campaign managers** (e.g. Obsidian Portal [168], World Anvil [172]) are apps specifically aimed at RPG GMs and/or players. They usually incorporate a worldbuilding wiki for the GM’s use, but may also enable users to track the events of each game session, schedule upcoming sessions, or send chat messages during or between games.

If a GM wants to map out part of his game world, he has a wide choice of software to aid him. Besides generic image-manipulation software like Photoshop or Inkscape [79, 26], there is also a whole family of special-purpose **mapping programs** with which a user can create free-form or gridded representations of fictional terrain. Examples include Campaign Cartographer, Wonderdraft, and Inkarnate [96, 171, 166]. These may come with placeable icons which imitate map styles found in classic fantasy works such as *The Lord of the Rings* [176], or which depict naturalistic or fantastic features likely to be found in a fictional world, from caves, to castles, to wizards’ towers, to sea monsters.

Ordinary mapmaking programs are essentially image creation software specialized for the domain of fantasy maps, but there’s also a burgeoning genre of **generative mapmaking tools**, which use PCG to create and render whole game areas from scratch. One example is Dungeon Alchemist [34], which uses a mixed-initiative design paradigm. After Dungeon Alchemist creates an initial 3D map of a location, such as a fantasy house, the user can revise the results by moving or deleting rooms, changing connec-

tions between areas, or even passing control back to Dungeon Alchemist to regenerate a section of the map.

Another example is Neverending Dungeon (NED) [152], which goes further than Dungeon Alchemist by also creating a scenario to accompany the generated map, including the reason the players have come to the generated dungeon and the goals that they should pursue there. NED’s scenarios are simplistic and transparently make use of templated text, so they come off as more of a proof-of-concept than the NED’s map generator does. Still, the concept of whole-scenario generation is suggestive of the topic of quest generation, as well as the wider field of generative narrative. All three of those fields could usefully influence the design of RPG support software which includes game design capabilities for the GM to employ.

### 3.2.2 RPG Support in Academia

Undercurrents [14] is a “tool for providing additional communication channels and media streams during tabletop roleplaying sessions”. It offers secret messaging, audiovisual playback synchronized between all players’ devices, and note-taking capabilities like those of a wiki. Game rules and facts are not part of Undercurrents’ data model.

Imaginarium [68] gives its user an interface for specifying a natural-language ontology of game entities, their qualities, and constraints on both. These ontologies are explicitly inspired by the extensive natural-language world-modeling capabilities of the Inform 7 programming language [117]. A sample ontology looks like this, where “can

be” denotes an optional constraint, and “is/are” denotes a mandatory one:

Cats can be large or small. Persian, tabby, and Siamese are types of cat.  
Cats are longhaired or shorthaired. Cats are grey, white, black, or ginger.  
— Imaginarium [68]

Having entered this ontology, the user can then prompt Imaginarium to “imagine a cat.” Imaginarium will then use a SAT solver to generate constraint-satisfying entities, and return responses such as “the cat is a large Persian” or “the cat is a grey, longhaired Siamese.” Imaginarium is specifically intended for use as an imagination aid that can be expanded with new information on the fly. The ease with which a user can switch between using and editing the ontology is worthy of imitation by future RPG support systems meant for use in live gameplay.

Alexander Mayben’s MS thesis [108] describes both a novel pattern language for creating storygames conforming to the *Powered by the Apocalypse* design framework [8], and a computer tool for designing games within the constraints of said language.

Designing a storygame, Reed writes, is heavily dependent upon the game’s facilitation of one or more of four key activities: generation, negotiation, administration, and what he refers to as “storywrighting;” the process by which ideas are “combined and shaped into an ongoing, coherent, and compelling story”. According to Reed, storygames occur at the midpoint of a “simulative spectrum” between world simulation and storytelling, and a “performative spectrum” between authorship and improvisation. Reed

also characterizes the game master role as unique to storygames in particular, imposing a level of mediation between simulation and performativity.

— Mayben [108], p. 8, discussing work by Reed [128];

in-line citations omitted

Mayben’s work reifies the design space which *Powered by the Apocalypse* offers to game designers. While this reification puts hard limits on the types of games that can be explored by users of the accompanying software, the notion of encoding game design patterns directly into game designers’ tools could lead to valuable UI/UX advances in design-support tools, whether those tools are intended for asset creation between games, or the gameplay-time design activities that define malleability.

Devi Acharya’s MS thesis [3] involved carrying out extensive interviews about GMing practice with experienced GMs, inquiring on such topics as the kinds of improvisation and editing each participant would engage in when adapting prewritten content to his own game. These interviews were conducted to establish an empirical, user-focused basis for the consideration of desirable qualities in future work on computational GM support. Acharya’s results establish two primary directions for said future work, which I summarize here from [3], p. 90–99:

- Tracking, visualizing, and editing game information corresponding to what I call substantive facts, such as relationships between characters. These capabilities could be used to give the GM the context they need to accurately describe a game scenario, spur improvisation while conforming to numerous facts already established about the game world, or update information being tracked by the

software tool to maintain correspondence with events taking place in otherwise analog play.

- Generating game content tailored to the GM and his gaming group. In particular, beyond the usual idea of creating content conforming to the game rules and game world being used, Acharya also suggests (p. 95–96) content creation strategies that take into account the PCs and their capabilities, or choices that the players have previously made.

The question that naturally arises in response to Acharya’s findings is how exactly PC capabilities, player choices, character relationships, and myriad other game facts might feasibly be represented in software, manipulated and inspected by the GM, and deployed for generative purposes. Following that question, one must also ask what the consequences of representing those things in software would be.

For instance, suppose that a hypothetical GM support tool can do game world content generation while taking past PC actions into account. It would not be feasible, during live gameplay, to track PC actions by having the GM do manual data entry; therefore, this tool would require players to declare their character actions through a computer UI. If that computer UI only accepted a pre-determined list of actions, gameplay using this tool would not offer the players the freedom of Principle 1. And if, to counter that, the tool was changed to accept any freeform text as the PC’s action declaration, it would become difficult for the tool to use that freeform text for content generation according to patterns which might be desired by the GM, such as narrative

arcs. My pursuit of a malleable, yet augmented game is an attempt to resolve this conundrum.

Shoelace [4], a GM support application for use during play of the RPG *GUMSHOE One-2-One* [90], is explicitly inspired by Acharya’s conclusions in [3]. It “helps game masters keep track of story events with a graph-based game world visualization, while also providing creative suggestions using Prolog queries over a database of game information (p. 1.)” Said capability for creative suggestion is inspired by the wider research area of story sifting<sup>9</sup>, in which a system pattern-matches sequences of game or simulation events and surfaces them to an observer [139, 83, 82, 81, 141, 86, 84, 25].

### 3.2.3 Observations

## 3.3 Virtual Tabletops

A virtual tabletop (VTT) software system, in its most basic form, digitally replicates key items of tabletop play equipment, such as battlemaps and character sheets. Some VTTs go further by enhancing those items with features unachievable in an analog context (e.g. dynamically revealing portions of a scenario map as the players explore it), or bundling content management features (e.g. allowing the GM to select and play appropriate dramatic music.) The most advanced VTTs also integrate software adaptations of RPG rulesets so that rules can be checked and processed by the computer.

---

<sup>9</sup>See [Subsection 7.3.4](#) for a longer discussion of story sifting in the context of future directions for work on ROLEPLAYINGGAME.

The VTT field has existed since the 1990s, beginning with the release of Virtual Advanced Squad Leader (VASL) in 1996 [179]. At first, VASL could only run a digital adaptation of the war game *Advanced Squad Leader* [52], but by 2002 it had been rebuilt into Vassal, a generic VTT for all war gaming [179].

OpenRPG, the first RPG-specific and generic (system-agnostic) VTT, was released in 2000 [33]. Fantasy Grounds, the earliest VTT still in active development, was released in 2004; its homepage claims it has had over 400,000 users [147]. The former is open-source<sup>10</sup>, and the latter is commercial; other VTTs have come from academic research.

Though there are VTTs intended for war gaming (e.g. Vassal) and board gaming (e.g. Tabletop Simulator [42]), today the term generally implies software specialized for RPG support. Because VTTs are designed for RPGs, are fully implemented in software, and have adapted RPG rules into software without turning them into fully computer-encased videogames, VTTs are closest to realizing the sort of vision I have for hybrid RPGs. Thus, for this thesis, it is vital to accurately explore the current capabilities and limitations of state-of-the-art VTTs.

### 3.3.1 VTTs in Academia

Tisch [57] is “a generic tool capable of enhancing a wide range of analog games, making use of the Microsoft Surface to supplement the analog game with digital feature

---

<sup>10</sup>All free (“libre”) software is open-source software, but not all open-source software (OSS) is free software. The differences have to do with copyright and the permissibility of commercial use. For consistency, I’ll refer to all VTTs with available source code as “open-source software,” without specifying whether or not they are also free software.



sets.” Its primary feature set revolves around sketching on top of prepared visual assets, typically battle maps. Physical objects tagged with RFID emitters enable various user interactions, such as drawing and erasing lines while sketching. Tisch also supports physical play tokens that have been similarly tagged; for instance, the screen can vary what’s displayed with the presence of certain tokens:

[P]laygrounds [are] rectangular screen sections which act the same as windows in most operating systems [and] can easily be created, moved, scaled, and hidden. **They can also be tied to a tagged object, being displayed only when and where the tagged object is put down.** The reasons for using playgrounds are primarily for users who want to take notes or when users venture in split directions or do different things, with each user/group getting a segment of the total screen, possibly interacting with completely separate feature sets or under different conditions.

— Hartelius *et al.* [57], p. 200–201, emphasis mine

Tabula Imaginarium [71] is an iPad application with “the purpose of aiding a player of a tabletop roleplaying game that is spatially separated from the other participants.” It allows for creating battle maps from a predefined tile set; placing and moving virtual character tokens (unlike the physical tokens used in Tisch); and tracking character statistics.

Judging both by year of release and the number of products available, most advances in the VTT field have been made in industry, not academia. Tabula Imaginarium

and Tisch (both released in 2012) are the exceptions. Both of these academic projects predate most commercial VTTs (including industry giants Roll20 and Foundry), and yet, they share similar intended use cases and functionality. The primary difference between them is that Tisch is more concerned with preserving the physical dimension of analog play, while Tabula Imaginarium leans into more computer-mediated play (as befits that system’s goal of supporting remote players.)

Tabula Imaginarium’s authors include a brief survey ([71], p. 21-23) of VTTs which were available at time of publication, including D20 Pro [114], RPG Cartographer<sup>11</sup>, Battle Map [126], Dungeon Mapp<sup>12</sup>[151], Fantasy Grounds, and Tisch. D20 Pro and Fantasy Grounds are still active; furthermore, double-digit numbers of new VTTs have sprung up in the decade since Tabula Imaginarium and Tisch were created (see [Subsection 3.3.2.](#))

### 3.3.2 VTTs Outside Academia

Nearly all available VTTs originate outside academia. An incomplete but representative list of VTTs in development as of 2023, arranged in a rough order of sophistication from large commercial offerings to one-person hobby projects, runs as follows: Fantasy Grounds, Roll20 [169], Foundry VTT [46], MapTool [138], Vassal, Arkenforge [163], Let’s Role [167], Tabletop Simulator, TaleSpire [35], Tabletop [155], Shmeppy [161], Owlbear Rodeo [174], Rolz [143], PyVTT [21], gTove [132], and Mirk-

---

<sup>11</sup>I couldn’t find a citation for this app, but see [40] for a review of it, which also gives a good idea of the feature set which VTTs had achieved by that time.

<sup>12</sup>“Mapp” with a double “p” is correct.

wood Engine [115].

The three most well-established VTTs are Fantasy Grounds, Roll20, and Foundry VTT (hereafter just “Foundry.”) I refer to this trio as the “big three.” Here are some metrics for the size and success of these platforms:

- Fantasy Grounds claims its users played 328,377 game sessions in Q4 of 2020 alone [148].
- “[In] Q3 of 2020, more than 100 million hours were played across Roll20” [170].
- Until recently, Foundry did not collect user or game system statistics, but we can turn to financial measures of success. As of August 2nd, 2023, Foundry’s Patreon page has 1,819 members who donate a total of USD \$7,583 per month (\$90,996 per year) [44] — and that’s on top all the money made to date by selling Foundry itself, at USD \$50 per license<sup>13</sup>.

For this thesis, I’ll primarily draw on the big three for evidence when making claims about the extent of commercial VTT capabilities. The big three have received the most funding, user testing, and development time, while smaller VTTs have far fewer resources. As such, capabilities which the big three possess may well be missing from smaller VTTs, and limitations of the big three are almost certain to also be present in smaller VTTs. Thus, restricting analysis to the big three should not meaningfully affect the validity of any claims about VTTs in general.

---

<sup>13</sup>A single license owned by a GM is sufficient for the rest of his gaming group to use Foundry as players.

All of the big three are generic: a given VTT can be used to play any game system which has been implemented as a package for that VTT. To achieve genericness, the codebase of each big-three VTT defines fundamental conceptual objects which abstract over certain elements of all possible domain models an RPG could have. Recall from [Section 2.3](#) that the taxonomic facts of a game system constitute its domain model; analogously, I provisionally refer to a VTT's restricted set of generic conceptual objects as its **metadomain model**.

Glossing over minor feature-set and terminology differences, the metadomain model of each of the big three contains roughly the following conceptual objects:

- Dice: a variety of dice can be thrown. Some VTTs accompany dice rolls with a simulated display of those dice rolling. Die roll expressions can be written in a language familiar to gamers, e.g. “2d10k1+3” for “roll 2 10-sided dice, keep only the better one, and add 3 to its result.” Those expressions can be saved for reuse.
- Actor: animate, volitional creatures.
- Item: anything not represented by an Actor.
- Scene: a map image with associated contents, usually but not always at battlemap scale.
- Token: a battlemap-compatible representation of an Actor or Item.
- Adventure: a collection of Scenes prepared ahead of time.

- Playlist: a sequence of music or video content, used to add ambience or share information.
- Status Effect: a game-system-defined temporary alteration to Actor or Item characteristics.
- Macro: a player- and GM-authorable sequence of VTT user commands issuable as one unit, e.g. “send this chat message, then perform the ‘attack’ action.”
- Rollable Table: a weighted list of thematically linked results which can be picked from with a die roll or other randomizer.
- Handout: a hypertext document (authorable by the GM, and sometimes by players too) describing some aspect of the game world. Common features include embedded media and triggerable die rolls.
- Package: a collection of assets (code, data, or images) which define a game system for use with a VTT. Said assets often include specializations of VTT metadomain objects, especially the Actor, Item, Dice Roll and Status Effect. It’s also common to include code implementing rules such as leveling up or equipping a character, and UI code for specialized windows or panels in which such activities are performed. Packages also implement custom character sheets for tracking or calculating character attributes, subject to (the package’s realization of) the game system’s domain model. Character attributes can be sent into calculations, including die rolls.

VTT features outside the metadomain model for representing game systems include marketplaces for utility extensions and game system packages; text, voice, and video chat messages; visual effects such as rain and snow (which don't interact with game system rules); and a broadcast log of events, such as die rolls or in-game actions.

All of the big three have sophisticated presentational capabilities for use with battlemaps, including fog of war, which obscures areas from view until characters have entered them, and dynamic lighting, which realistically casts areas of light and shadow based on point sources (such as a carried torch or flashlight.) Battlemaps can also include walls and doors, which the GM can draw out to block movement and visibility while also concretely representing aspects of a game world location. When carrying out combat or other small-scale tactical movement activities, users can move the tokens they control to desired positions. A user can measure point-to-point distances with a virtual ruler, or observe a distance readout while dragging a token ([Figure 3.1](#).) Battlemaps may also display a grid overlay for precise movement and positioning. Finally, to draw others' attention to a battlemap position of interest, users can issue "pings," or audiovisual alerts centered on a chosen point.

All in all, these features demonstrate that the big three VTTs are comprehensive tools for playing existing tabletop RPG systems as written, with some computer-enhanced calculation and visualization added on. They gain an advantage over analog tools by bringing together digital versions of those tools into a one-stop software suite. Their primary shortcomings are a failure to innovate beyond the affordances of analog tabletop-gaming equipment, and a failure to accommodate malleability besides moving

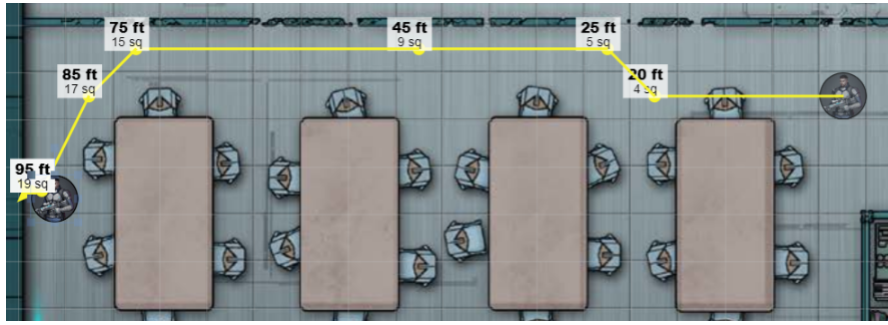


Figure 3.1: Planning a token’s movement in Roll20. The path begins on the right, at the dimmed copy of the token which represents its initial position. Source: [119].

individual character tokens on battlemaps (which is a way of altering substantive facts.)

Regarding malleability support, a GM who wants to make even a tiny change to the rules or domain model of his chosen game system has to jump through numerous hoops. For Foundry, the steps are as follows. First, if the GM is currently running a session for his players, he must shut the server down. Assuming he has access to the package files for his game system, he must open them in an external editor, create or edit code to implement some change, increase the version number of the package, upload the package to the server, and restart the server. When the players reconnect, they download the new package.

What’s more, if the changes require altering the attributes of entities which already exist in the game world (e.g. updating all characters to correct a misspelled attribute), the GM must also write code to correctly perform that task. To my knowledge, Foundry does not give the GM any help with this. This alteration similar to a database migration, and would be a good place for designers of future VTTs or hybrid systems to intervene, perhaps by generating migration code from changes to attributes

that the GM specifies.

As for innovation, with the exception of dynamic lighting, most VTT features could be achieved by supplementing analog play with everyday software<sup>14</sup>. You don't need a VTT to play music or videos, create hyperlinked wiki text, design a rollable table, or move things around on a battle map. What VTTs offer, in all these other areas, is a reduction in the manual toil of tracking the myriad attributes of a typical RPG's domain model, and the bundling together of all those capabilities in one application.

Toil-reduction and bundling are certainly innovations, but they're innovations within a fundamentally un-innovative paradigm: VTTs digitize RPGs almost exactly as they've always been played. This is a rational aim for a commercial software platform which sells to an audience of gamers seeking an easier way to play games they already enjoy, and which presents RPG publishers with a golden opportunity to re-sell gamers VTT-compatible implementations of their favorite systems and supplements. But with VTT developers, gamers, and publishers all seeking to digitally replicate existing RPGs and their manner of play, VTTs hardly seem poised to enable a reimagining of RPG designs to take advantage of the strengths of the computational medium<sup>15</sup>.

As evidence for VTT capabilities failing to drive innovation in RPG design, one need look no further than *Burn Bryte* [53] and *Crucible* [47]. These are game

---

<sup>14</sup>Note that works earlier in this chapter, e.g. TViews, have used software-hardware communication to offer dynamic lighting. Furthermore, at least one VTT is pursuing a more hardware-integrated niche: Arkenforge can display its battlemaps on a touchscreen tablet such as an iPad, and use the touch sensors to track physical tokens as they move across the battlemap [164].

<sup>15</sup>Late in the thesis-writing process, I learned that members of the VTT community have begun experimenting with more radical interface additions and upgrades [20, 1, 127, 110]. I ran out of time to more thoroughly explore this phenomenon; at first blush, while they don't counter my assertions about the deeper problems with VTTs' metadomain models and mimicking of traditional gameplay, they do seem to point to a new wave of innovation in VTT interface affordances.



systems created by the VTT makers themselves: Roll20 and Foundry, respectively. The marketing material for these games emphasizes that they have been designed for the ground up to work with their respective VTT platforms, e.g. “From the ground up, Crucible is designed to leverage the unique capabilities of Foundry VTT [47].”

Despite such claims, both *Burn Bryte* and *Crucible* are game systems that could just as easily have been released as ordinary rule books for traditional analog play. be played analog. None of their subsystems make use of computers’ storage or processing capabilities to achieve a degree of DCG higher than would be feasible in a traditionally-played game. However, there is one way in which *Burn Bryte* explores how computing can enable new forms of interactivity in an RPG. That game has a spaceship creator (somewhat like the house builder in *The Sims* [107]) which allows players to drag and drop ship parts while a total accounting of spaceship capabilities and costs is maintained. This excellent idea, reminiscent of countless videogame interfaces for character or item customization, uses computer interactivity to turn a task that would require tedious analog guess-and-check into an exploratory and creative exercise<sup>16</sup>.

### 3.3.3 Observations

VTT systems have proven that traditional game systems which are relatively high in DCG can successfully be converted into computational artifacts. They’ve also shown that gamers are willing to use computer-augmentation systems when they offer

---

<sup>16</sup>For more discussion of how RPGs can learn from videogames, especially when it comes to interface design, see [Subsection 6.3.2](#). For examples of how ROLEPLAYINGGAME incorporates videogame-inspired interfaces, see [Subsection 5.5.2](#) and the figures in [Subsection 5.4.3](#).

sufficient advantages over analog gaming. I owe VTT creators a debt for making it more common to use computer systems while GMing and playing RPGs, because that might mean I'll encounter less resistance when trying to convince potential users to make the leap to full computer mediation. However, the VTT ecosystem seems to have spurred little innovation in the design of new RPGs, and they haven't done much more to support malleability than the hybrid systems covered above, even without the constraints of physical hardware.

## 3.4 Videogames

There are a variety of videogames inspired by the traditional RPG. Full computer mediation tends to limit the range of player actions and the size of the domain model that a videogame can support, meaning most RPG-esque videogames can be considered to have fixed themes<sup>17</sup>. That said, there are several extant videogames that provide a user experience quite similar to that of the VTT. I will also cover a striking and unique work from the world of MUDs which, to the best of my knowledge, is the only computer game which is just as malleable as a traditional RPG.

### 3.4.1 CRPGs

Computer games inspired by the rules and playing experiences of traditional RPGs began to appear within months of the release of *D&D* in 1974 ([130], [16].) The earliest of these computer roleplaying games (CRPGs), such as *The Dungeon* and *dnd*,

---

<sup>17</sup>See [Subsection 2.8.2](#).

were text-based games which imitated a subset of *D&D*'s gameplay [130]. These were soon updated to take advantage of nascent computer networks, evolving into multiplayer virtual worlds so compelling that some people would just hang out inside them and talk, rather than work toward game goals [16]. In 1980, a few years after the breakout success of *Zork* ([69], neé *Dungeon*), Richard Bartle and Roy Trubshaw released a text-based multiplayer CRPG which gained so much popularity that its name, *Multi-User Dungeon* (MUD), became a metonym for this burgeoning game genre<sup>18</sup> [131].

From that time on, advances in computing technology permitted an ever-increasing proportion of CRPGs to incorporate full-fledged graphical interfaces and representations of game content. Today, in 2023, that proportion is nearly 100%, so this section will be concerned only with graphical CRPGs. 1981 saw the releases *Wizardry* [51] and *Ultima* [48]. Jimmy Maher, videogame historian and author of the *Digital Antiquarian* blog, describes the magnitude of their influence:

[*Wizardry* and *Ultima*] stand as the archetypes for two broad approaches to the CRPG that would mark the genre over the next decade and, arguably, even right up to the present. The *Ultima* approach emphasizes the fictional context: exploration, discovery, setting, and, eventually, story. Combat, although never far from center stage, is relatively deemphasized, at least in comparison with the *Wizardry* approach, which focuses on the *process*

---

<sup>18</sup>Note that *Multi-User Dungeon* was not the first MUD; *dnd* and other games were earlier [130, 131].

of adventuring above all else . . .

— Maher [101]

Countless other CRPGs could be considered historic developments, from *Pool of Radiance* to *Final Fantasy* to *Star Wars Galaxies*<sup>19</sup> [146, 153, 37]. Whether turn-based or real-time, whether single-player or massively multiplayer, CRPGs today boast beautiful graphics, enormous worlds, and hundreds of thousands of words of dialogue. Furthermore, their mechanics (even when adapted for real-time play) are almost universally the unmistakable descendants of traditional RPG rulesets.

In almost all cases, CRPGs are unable to completely capture the open-endedness of traditional RPGs. It may seem overly simplistic to sweep, into one bucket, nearly 50 years’ worth of games with dramatically different interfaces and levels of technical sophistication. But lumping them all together is not without justification. As CRPGs are encased in the computer<sup>20</sup>, i.e. fully mediated by computer processing, they almost universally lack malleability.

### 3.4.2 Games with a GM Mode

There are outliers which blur the line between traditional RPGs and videogames.

*Divinity: Original Sin 2* [159], *Sword Coast Legends*<sup>21</sup> [116], and both *Neverwinter*

---

<sup>19</sup>For a full history of CRPGs, see [12].

<sup>20</sup>This apt term is from [13].

<sup>21</sup>A digression: In *Sword Coast Legends*’ GM mode, while the PCs explore a dungeon, the GM spends “thread” — a resource invented for this game with no apparent antecedents in any edition of *D&D* — to perform GM “moves” such as placing monsters and traps, or *remove* such elements which he has already placed. Additionally, the GM gains thread when the PCs succeed, such as by killing monsters; he loses thread when players fail, such as when a PC dies (and on a total party kill [TPK], all thread is lost.) This attempt to gamify the practice of GMing is worthy of further study, perhaps in contrast to how GMing is partially gamified by *Powered by the Apocalypse* game systems. For now, note that the

*Nights* games [15, 36] (henceforth *DOS2*, *SCL*, and *NWN*) can be played as single-player CRPGs, but all of them also contain a multiplayer mode which repurposes the single-player mode’s interfaces, assets, and mechanics for traditional-RPG-like play. In this mode, a GM uses game assets to create an RPG adventure environment, then sets up his copy of the game to work as a server. The players connect to the server from within their own copies of the game, and can then explore whatever the GM has created for them.

In all respects, these videogame GM modes are essentially VTTs with 3D graphics. Their rules and taxonomic facts are not malleable at runtime, and while those elements can be edited by a GM outside of normal gameplay, separate applications are required for that task, and it requires substantial time and effort<sup>22</sup>

All that aside, GM-mode CRPGs are not devoid of innovation. Most notably, *Divinity: Original Sin 2* includes a system of “vignettes,” which are choices that pop up on the player’s screens separately from the rest of the game interface (Figure 3.2.)

GMs can create vignettes on the fly and even link them together. This is a laudable

---

design of thread makes rigid assumptions about what it means to GM well. First, by punishing the GM by taking away thread when a PC dies, *Sword Coast Legends* implies that PC death must be a failure on the GM’s part. Second, making the GM spend thread to remove previously-placed elements implies that the GM should stick with a dungeon design once it has been chosen. While that is commendable as a way of upholding Principle 2, it also patronizes and straitjackets the GM by not allowing him to correct mistakes, nor adapt the dungeon to changing circumstances (i.e. exercise malleability of substantive facts.) The thread cost to remove existing dungeon elements sadly also precludes using a certain total starting amount of thread to characterize the difficulty of a dungeon, which would have been an interesting innovation beyond the entirely ad hoc “challenge rating” measure of individual monster/threat difficulty (which has received little critical examination or revision in official editions of *D&D*.)

<sup>22</sup>*DOS2* comes with Divinity Engine 2, a standalone content-creation program with which GMs can create textures, explorable levels, NPC models, new stat calculations, and even entire mods. Both *NWN* games offer similarly full-fledged mod tools. *SCL*, to my knowledge, offers no modding capabilities whatsoever.

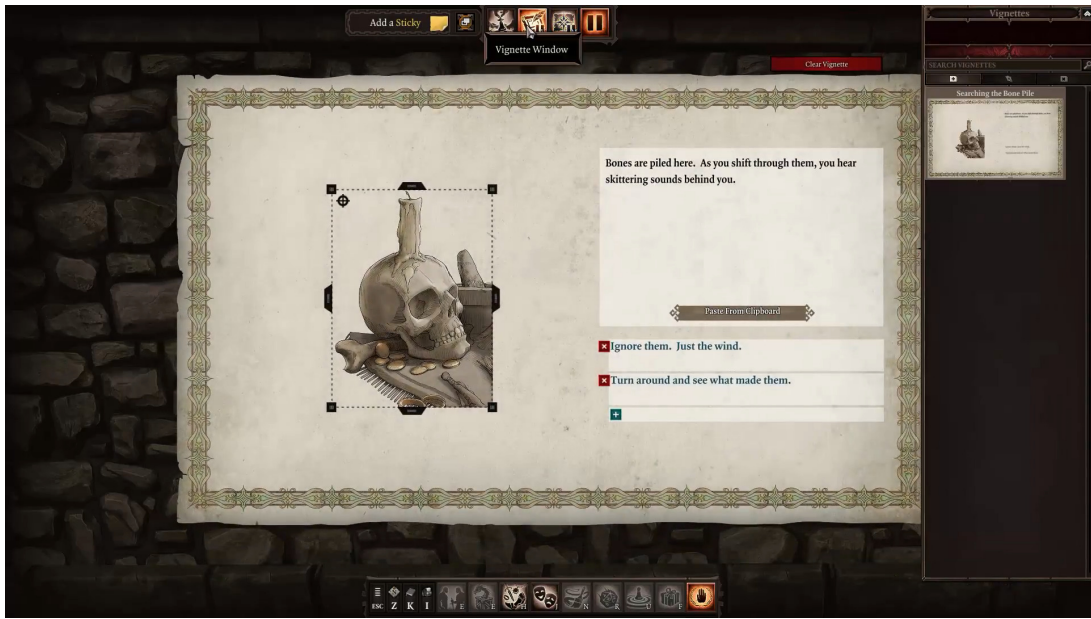


Figure 3.2: A GM creates a vignette within *Divinity: Original Sin 2*. Source: [88].

exploration of a new UX paradigm for computer-augmented RPGs. Unfortunately, vignettes have no interaction whatsoever with the rest of the game system; the GM must manually carry out any effects which he intended the vignette to trigger.

Finally, there’s at least one game which offers a GM mode, but is outside the realm of CRPGs proper: an unusual title called *Sleep is Death* ([135], hereafter *SID*.) *SID* is played by exactly two players. One of the players perceives an explorable graphical environment, with objects, characters, and other interactive features. What’s unique about *SID* is that the game world is controlled by the other player, the “Controller,” who can not only edit and change the world, but also design new game elements for it — all from within *SID* itself, using multiple graphical content-creation interfaces (Figure 3.3, Figure 3.4.) On the one hand, the parallel with players and GMs is obvious,



Figure 3.3: The *SID* Controller (analogous to a GM) works on a level. Source: [136].

and the support for substantive and taxonomic malleability is impressively thorough. On the other hand, *SID* achieves its open-endedness by having no computer-mediated game system whatsoever, instead foisting the creation and interpretation of any system onto the GM. Nevertheless, its integration of game-design tools with a GM interface is worthy of imitation by future RPG support software.

### 3.4.3 A Curious MUD

It is time to discuss an important subject mentioned in this section's opening paragraph. There exists a MUD with malleability rivaling that of a non-computer RPG: 1990's *LambdaMOO* [29]. Reed [129] summarizes its inventor's stroke of genius



Figure 3.4: The *SID* Controller creates a new game object. Source: [137].



as follows:

[Original creator Stephen White] realized that for players to truly be creative in a virtual world, the power to make new rooms and objects wasn't enough. They would need the ability to create new rules and systems, too. But that would require a true programming language capable of **altering the very world its user was immersed in**, and a consistent ontology allowing that world to be changed in a simple and consistent manner. [W]hite decided to build a system where everything in the world—from players, to items, to rooms, to the exits connecting those rooms—was represented as an object that could be created or modified by special commands.

— Reed [129], emphasis mine

Reed continues with a description of the commands available to any *LambdaMOO* player, which could be used to reprogram the game world *from inside itself*:

For instance, a popular in-game coding tutorial would teach you how to create your own pet rock. To program the ability to *pet* your pet rock, you needed to type in three commands at the prompt: **commands no different, from the system's perspective, than any other player input** like *look* or *go north*:

```
@verb rock:pet this none none rxd
@program rock:pet
player:tell("You pet the rock. Nothing happens."); .
```

The words *this none none* in the first line would define the specification of the *pet* verb from the rock’s perspective: it takes a single direct object *this* (the rock) and no preposition or indirect object. *rxd* indicates the verb is readable by others (anyone can pet the rock), callable by other verbs like a function, and will show a traceback if its program crashes. The dot at the end of the third line indicates the program being entered is finished. **A player might weave these instructions into a stream of chatting with friends and interacting with the existing environment: programming the world turned into just another fundamental part of existing within it.**

— Reed [129], bold emphasis mine

While every player in *LambdaMOO* possessed such abilities, it’s easy to imagine how, with some tweaking, this game could have formed the foundation of a digital equivalent to the traditional RPG. Reed’s description of a *LambdaMOO* player seamlessly switching between design (reprogramming) and play (chat and world interaction) is almost exactly analogous to the mix of behaviors exhibited by a GM running an RPG session.

Given that *LambdaMOO* is the high-water mark for malleability in computer games, it’s reasonable to ask whether *LambdaMOO* itself would be a good medium in which to build a malleable roleplaying game. I believe the answer is no.

There are two ways in which one could use *LambdaMOO* to implement a malleable RPG. The first is reshaping *LambdaMOO* itself into an RPG (whether from inside

the game or using an external editor.) This runs into problems almost immediately:

1. As a collaborative MMO of building and socializing, all *LambdaMOO* players have access to the commands that can reshape the game's reality. That might work for a GMless RPG, but not for a GMed one, where player characters typically possess well-defined limits on their ability to alter the game world. Furthermore, as a non-graphical, parser-input MUD, the *LambdaMOO* interface would be considered rather out of date by most gamers today. A textual interface would also be less than ideal for traditional logistical gameplay (e.g. moving characters on a battle map) and for novel user interfaces supporting non-traditional player goals (e.g. choosing where to plant crops on a farm, or designing a PC's house in a Sims-like editor.)
2. *LambdaMOO* and RPGs impose different standards on how the passing of real-world time relates to in-game time. Although player actions in *LambdaMOO* are expressed as discrete commands, gameplay is more aptly called real-time than turn-based, since players can act as fast as they can type. RPGs, in contrast, need specify no fixed relation between real-world and game-world time. Sometimes gameplay is resolved in a turn-based fashion, with characters having limited freedom of action in a given time span; at other times, an action might require skipping forward by hours, weeks, or even larger chunks of in-game time. These latter possibilities would be difficult to reconcile in *LambdaMOO*'s real-time play.
3. It's typical for the GM to confirm that players can actually take the actions

they have declared. When performing this responsibility, the GM is essentially intervening between the declaration and the execution of an action. *LambdaMOO* does not, to my knowledge, have a phase in its interpretation loop where the GM could glance at submitted commands from players and accept or reject them.

There's another approach which could be used to implement an RPG in terms of *LambdaMOO*. Users implemented sophisticated simulation logic inside *LambdaMOO*, as well as turn-based games like a functioning chess set [131]. One could conceivably embed an RPG into *LambdaMOO*, using internal logic separate from *LambdaMOO* to govern play, just as the chess set operates by the rules of chess. A user could log in to *LambdaMOO*, find the room or object granting access to the RPG area, and then “log in” to the RPG. This approach would effectively be using *LambdaMOO* as a user interface to the game of interest, which seems problematic not only for the UI-related reasons given above, but also, for the endless task of reconciling the *LambdaMOO* “host” (however fluid) with the endlessly growing “client” codebase inside it. At some point, it would become more rational to incorporate *LambdaMOO*-style malleability into a different, purpose-built platform.

#### 3.4.4 Observations

In general, videogames which include playstyles inspired by the traditional RPG seem to offer more full-fledged editing tools than VTTs do. While these sometimes suffer, like VTTs, from not integrating tools for malleability directly into the runtime-accessible UI, there does generally seem to be more experimental interface work coming

from videogames than from VTTs.

My chief concern with using 3D action videogames as the basis for a malleable RPG system is that 3D animated assets require substantially more design effort to create than symbolic 2D assets. Without strong user-facing, runtime-accessible generative tools to make up for that deficiency, I don't think that future RPG augmentation work should strive to offer realistic 3D representations of the game world. *ROLEPLAYINGGAME* sticks to symbolic 2D representations of game world content, as VTTs do.

Finally, it's hard to overstate how tantalizing I find the *LambdaMOO* approach to world modeling and user interaction. Its inheritance from the interactive fiction tradition gives it clear, understandable divisions between user commands and PC actions, as well as between what I call rules and facts<sup>23</sup>. With *ROLEPLAYINGGAME*, I designed an interactive-fiction-style command/action distinction to try and achieve the same clarity in my own work. Finally, *LambdaMOO*'s architecture and user-facing affordances give it a level of malleability not seen before or since in any software of which I am aware. This makes it an intensely important object of study for anyone looking to translate RPG malleability into a digital environment.

### 3.5 Summary

My goal with this thesis is to offer an alternative to existing works which have tried to capture aspects of the traditional RPG in a computer-mediated fashion. Having

---

<sup>23</sup>Rules which govern user commands and character actions are indeed attached to in-game items (the existence of which I would call substantive facts.) But that is just a consequence of the *LambdaMOO* implementation model; it doesn't make it hard to tell what's a fact and what's a rule.

reviewed prior art from four areas of the multidimensional design space containing the overlap between software and RPGs, I've identified gaps in all four of those areas with regard to supporting malleable system redesign, as well as assisting players and GMs with comprehending complex game states. In the next chapter, I'll synthesize some of the observations I've made throughout the present chapter into a set of design goals for RPG augmentation software. These design goals stake out a section of design space which has not been heavily explored by the overarching categories of prior art seen in this chapter.

## Chapter 4

# Speculative Design Goals for Computer-Augmented, High-DCG RPGs

The works reviewed in [Chapter 3](#) contain implementation decisions and assumptions about game system content making them unsuitable as a foundation upon which to build towards this goal. For that reason, this thesis’s technological contribution to the literature — the `ROLEPLAYINGGAME` software system detailed in [Chapter 5](#) — is standalone, rather than an extension or plugin for existing RPG-support software.

That’s not to say that those systems are bad or wrong. Furthermore, although those related works can’t provide me with a starting point for software development, they still offer both a background against which to contrast the precise ways in which `ROLEPLAYINGGAME` is novel, and an ecosystem of design decisions which can be

investigated and potentially applied to ROLEPLAYINGGAME. As such, I will ground my discussion of specific system architecture and user interface concerns in the features and characteristics of the work presented in [Chapter 3](#). This also ensures that ROLEPLAYINGGAME can properly be called a response to those works of prior art, rather than just an idiosyncratic invention.

In this chapter, I consider desirable properties of a hypothetical hybrid system for playing, GMing, and designing RPGS that are malleable and high-DCG. After proposing each property, I give examples of related works that possess or lack it. Then, I discuss how the decision to endow a system with that property might constrain the implementation of said system. While developing ROLEPLAYINGGAME, although there were limits on how much I could do as a master’s student working solo on experimental software, I tried to include as many of these properties as I could.

For brevity’s sake, throughout this chapter I use “a new application” to mean “a new computational system for supporting the play and design of malleable, high-DCG RPGs.”

## 4.1 Five Goals

**A. A new application should require a minimal amount of contingent hardware, ideally no more than the computers used to run and access it.**

If a new application requires hardware beyond the computer(s) used to run and access it, the aspects of the new application which use that hardware will be less



amenable to malleability.

A software application can not only exhibit many behaviors, but also alter its behavior over time, whether by modifying its own source code or input data, or by responding to external inputs like user commands. As a medium of expression, software is therefore uniquely suited to representing, or recreating, the malleability of RPGs. The more non-software "material" which is used in the construction of a new application, the more constraints must be imposed on the game designs which can be realized with that application.

For example, consider a system which tracks the movements of game characters by using a scanner to read RFID tags attached to plastic miniatures. Game scenarios played using this system would only be able to contain as many characters as the users have RFID tags, and manufacturing additional RFID tags can hardly be done at the drop of a hat.

The difficulty of manufacturing additional RFID tags is an instance of a more general problem with contingent hardware, which is that every piece of special-purpose hardware used in a new application increases the authoring burden of developing new game content, and therefore limits the ability of a GM to exercise malleability on the fly. For example, even if the STARS system software were changed to offer game-system malleability, a GM designing game content on the fly would have to consider how that content interacted with such hardware peripherals as the object-tracking camera and the player headsets.

Finally, sticking to software, and the familiar interfaces of the personal com-

puter, ensures that a new system can reach the largest possible audience. For commercial systems, the software-only approach also minimizes the system's cost to end users, as a system requiring idiosyncratic hardware would have to include in its sale price both the direct cost of hardware development, and the indirect cost of the longer development period needed to make software and hardware work in unison.

**B. A new application should retain the malleability of traditional RPG play. Therefore, it should computationally model both facts and rules.**

If a new application is to allow for malleability, it must enable GMs to (re)design rules and facts during play. A system with the ability to receive alterations to rules and both kinds of facts would be critical to supporting Principle 1. to permitting players to declare actions that don't yet have code implementations (because the GM could then either implement those actions on the fly, or manually enact the appropriate changes to game world state, i.e. substantive facts.)

Given the unusual degree of user-driven flexibility that such software would possess, attempting to achieve malleable RPG play in computerized form will concomitantly require addressing issues such as how rules and facts should be stored; whether and how rules and facts should be versioned (so that they can be rolled back to previous states); and what UI affordances<sup>1</sup> will best allow GMs to quickly and easily exercise malleability.

I showed in [Chapter 3](#) that non-VTT RPG support tools don't computerize

---

<sup>1</sup>As one of the participants in the user study put it, developing user interfaces for the design and play of malleable, high-DCG malleable RPGs is a juicy HCI problem. I couldn't agree more. A whole thesis could be spent just on characterizing that problem, much less solving it.

rules as executable procedures, and usually don't store facts in a machine-manipulable format, either. While campaign managers and worldbuilding wikis have advantages<sup>2</sup> over pencil-and-paper notes and rulebooks, they are still limited to representing rules and facts as prose.

By contrast, VTTs and GM-mode videogames *do* computerize game rules and game facts. These applications also generally offer malleability for substantive facts, allowing such actions as placing a new NPC or altering the values of a PC's stats. However, even for the smaller population of videogames and VTTs which allow users to add packages of custom rules and taxonomic facts, the GM must design such content outside normal gameplay<sup>3</sup>, and load it before the program launches. As such, even state-of-the-art systems from these two fields cannot be said to support runtime malleability of rules or taxonomic facts.

**C. A new application should make minimal assumptions about, or impose on, the domain model of a game system.**

Taking Foundry as representative of VTTs, apart from bookkeeping activities like managing character inventories or creating new PCs, it can only support game scenarios dealing with precise placements of in-game physical entities (e.g. characters, items, walls, doors) for the purpose of movement, exploration, and combat on a tactical-scale map. A look at the Foundry documentation shows that its notion of a game system consists of character sheet templates, visual assets (e.g. battlemap tokens), and a bundle

---

<sup>2</sup>Such as infinite writing space, full-text search, and easier, faster storing and sharing of game content.

<sup>3</sup>In the case of VTTs, this design work may even take place in an entirely separate program. For instance, a GM who writes code or creates map tokens for a VTT extension will do so in a text editor, like Visual Studio Code, or an image editor, like Photoshop.

of JavaScript code that defines game rules (as well as the functionality of numerous UI components.) There's nowhere to include or define concepts (taxonomic facts) for other types of gameplay which might credibly be included in an RPG.

For instance, some dedicated soul would have to create a truly elaborate extension for Foundry to offer a wargaming interface where masses of characters can be moved at once, or where a single unit on the battlemat represents a number of distinct characters. And the implementation of that extension would be an endless uphill battle against Foundry's presupposition that RPG support can be reduced to character bookkeeping and single-character-scale battlemat scenarios.

All of this shows that a game run under Principle 1 will find Foundry to be of little use outside its core use cases. Even if that game's GM generally prefers to reuse existing taxonomic facts (e.g. character stats, or types of dice roll) when modeling a new aspect of gameplay, Foundry's runtime capabilities at best allow him to create a dice roll macro drawing on certain stats from an input character. He can't declare a new rule which participates in all the other operations of the VTT, just like the rules loaded in at startup time. Nor can he invent and name a new stat which, for each character, should take on a value derived from two existing stats. He can't create a new action for characters to take, and he can't add a new attribute to all existing sword items plus the data template which defines default stats of newly-created swords.

In summary, VTT and videogame domain models are rigid, and rigidity precludes malleability. I understand that the creators of an app must make a firm decision about what domain it operates in, lest development come to a standstill under the

weight of a thousand wild ideas. But, as it stands today, the fact that VTTs (and GM-mode videogames) impose a rigid domain model ensures that they cannot hope to computationally support traditional-RPG malleability.

These domain-model limitations have engendered a state of affairs where, for instance, *all* third-party game systems available for Foundry [45] replicate pre-existing RPGs, such as *D&D*, which were designed under the constraints of traditional tabletop play<sup>4</sup>. Their rules have been ported wholesale into Foundry-compatible JavaScript, but they don't have any game mechanics which could only be executed by a computer. A hypothetical lightweight VTT, or even an image-drawing program, could be used to support many kinds of gameplay: one might declare that a particular battlemat represented the layout of a player's house, or the goods available at a store. But, without explicit modeling of rules and facts, the computer could not be used to make high-DCG gameplay feasible.

**D. A new application should enhance users' thinking about the current game scenario, rather than draw attention away from it.**

Bergström and Björk [13] make good points about players of computer-mediated games attending more to their computer interfaces than to the game. I've noticed this tendency myself ever since I began augmenting my RPG play with computer systems.

Even an ordinary wiki which stores images or rule text can distract players to such a

---

<sup>4</sup>Around February 2023, I remember seeing exactly two Foundry-compatible game systems which were not pre-existing RPG rulesets. One doesn't count because it wasn't actually a game system: it turned Foundry into a tool for designing character sheets. The other, however, was someone's own personal RPG. That would put the lie to my claim of "all," above, but at time of writing, I can't find that personal game system listed at the Foundry Community Extensions link anymore.

degree that other participants must nudge them back into the game session.

However, part of the appeal of computational interfaces for complex games is that the computer can provide information-dense, automatically-updated displays of game state. Furthermore, one of my goals is to enable designers to increase the DCG of their game systems. Thus, accepting a software-mediated player view of the game world (or more generally, a software-mediated RPG) seems unavoidable. Given those considerations, a new application should explore how software-mediated play interfaces can foster immersion and comprehension, rather than distraction.

#### **E. A new application should prioritize co-located play.**

Years of GM experience have trained me to pay attention to players' emotions, facial expressions, tones of voice, and body language. Gauging players' moods and stress levels is critical for, among other things, determining whether to press them with more challenges or let them recover for a while. I don't think it's feasible to meaningfully monitor and manage players over video chat, where each participant is reduced to a low-resolution face, and some amount of lag is unavoidable.

## **4.2 Summary**

This chapter has presented key design criteria that, when taken together, pick out a portion of the design space of RPG augmentation software which has, to date, fostered little or no activity. In the next chapter, I discuss the design, implementation, and features of ROLEPLAYINGGAME, an app within that area of design space.

## Chapter 5

# The ROLEPLAYINGGAME System

In this chapter, I present ROLEPLAYINGGAME, a prototype software system for RPG design and play. It possesses properties from the previous chapter which I argued would be key for supporting high-DCG, malleable gaming operating on Principles 1 and 2. This chapter begins by discussing key ROLEPLAYINGGAME implementation details, as well as important decisions during the development process. Next, I showcase ROLEPLAYINGGAME's support for malleability of rules, taxonomic facts, and substantive facts. After that, I give the reader a tour of ROLEPLAYINGGAME's major user interface components, showing how a fully computer-mediated system can not only enhance traditional aspects of RPGs with affordances not achievable via analog tools, but also expand the set of tasks and activities that can feasibly take place at playtime.

As a master's student working solo<sup>1</sup>, I had to accept limits on what features

---

<sup>1</sup>Of course, throughout my thesis year I received valuable guidance from my thesis advisors, and advice from several other people; however, I was the only person who worked on actually designing and implementing ROLEPLAYINGGAME.

I could develop in the development time available for my thesis project (approximately ten months.) Despite this, I believe `ROLEPLAYINGGAME` represents an advance in the fields of RPG support and human/computer hybrid systems, and offers capabilities which are novel for the field of RPG support software systems.

## 5.1 Features Which Were Out of Scope

A computer-augmented RPG system suitable for use by a real gaming group should certainly include some form of character sheet and tactical battle map. To provide evidence that achieving `ROLEPLAYINGGAME`'s novel features didn't require abandoning the familiar features of existing systems, I implemented minimal versions of a character sheet ([Subsection 5.5.9](#)) and a battle map ([Subsection 5.5.6](#)). Thereafter, I left the character sheet alone, and only added to the battle map when that addition would support novel capabilities.

There are a number of beneficial user experience (UX) or "quality of life" (QOL) features which would be desirable in a fully-fledged system, but which I did not pursue for `ROLEPLAYINGGAME`. For instance, I didn't implement drag-and-drop interactivity for placing items into inventories, or moving characters around. Drag-and-drop support would be an unskippable step in giving a production system an intuitive and familiar UX — but, precisely because it is familiar, implementing it would not be research, only ordinary software engineering. I'll return to the subject of desirable QOL features in [Section 7.3](#).



## 5.2 Implementation Details

### 5.2.1 Language and Architecture

ROLEPLAYINGGAME is written in Clojure, a language in the Lisp family [64]. One reason for choosing Clojure was that, as a hosted language, it has compilers targeting both the Java Virtual Machine (JVM) and JavaScript<sup>2</sup>; this means that much of the source code written for the JVM backend of an application can be reused in a JavaScript frontend<sup>3</sup>. This was vital for avoiding duplication of code used to handle key game concepts like rules. Another key benefit of using a Lisp is that the GM can connect a REPL to a running instance of ROLEPLAYINGGAME, allowing him to execute arbitrary queries and commands against game state when the UI is not sufficient for his needs<sup>4</sup>. This capability came in handy countless times during development; it seems likely to be both useful and necessary for handling the unexpected questions and scenarios which arise so often in play.

ROLEPLAYINGGAME uses a client-server architecture. Given the design goal of co-located play, the GM is expected to run the server on his computer, and both the players and the GM are expected to interact with it using laptop computers.

---

<sup>2</sup>Since the JVM-hosted Clojure differs in some respects from the JavaScript-hosted one, the latter is referred to as ClojureScript (CLJS).

<sup>3</sup>In ROLEPLAYINGGAME's case, the 4,636-line codebase contains only 43 "reader conditionals," which are instructions to evaluate an expression differently (or not at all) depending on whether the JVM or JavaScript compiler is being used. 18 of those are trivial, e.g. loading a library which is named differently in its Clojure and CLJS versions; the other 24 are mostly deployed to prevent the CLJS compiler from loading back-end functionality which the front-end won't need. If each reader conditional guards an average of 15 lines of source code, then  $4,636 - 360 = 4,276$  lines are reusable across both host languages — 92% of the codebase.

<sup>4</sup>Since the frontend client was implemented in ClojureScript, a REPL could be attached to that separately; this would be useful for testing display or visualization subroutines which are only available in the frontend codebase.

ROLEPLAYINGGAME provides a web-based GUI client, which changes its appearance and affordances based on whether the logged-in user is the GM or a player. The web client is implemented using ReactJS [183] to efficiently re-render the UI in response to game state changes which are sent from the server to the client<sup>5</sup>.

This thesis was written with respect to the ROLEPLAYINGGAME codebase as of May 2023. The code is not currently open-source or otherwise publicly available.

### 5.2.2 Key Implementation-Level Concepts

The key abstractions used in ROLEPLAYINGGAME are game state, kinds, commands, actions, rules, and events.

**Game state** is the heart of ROLEPLAYINGGAME. It consists of a large key-value map which contains all the code and data to define both the current status of both the game world (substantive facts) and the game system (rules and taxonomic facts, e.g. “what actions are possible and how do they function?”) The world and the system are themselves deeply nested key-value maps. Outside both those maps, but still inside game state, is player data, consisting of each player’s name, ID, available characters, and active character.

During gameplay, game state is contained in a Clojure “atom,” a wrapper data structure which automatically enforces safe, concurrent multi-threaded read/write access to the underlying primitive or compound type. Manipulating an atom is faster than manipulating a database since atoms are in-memory structures, but game state

---

<sup>5</sup>The ReactJS was not written directly, but was instead compiled from ClojureScript expressions via the Re-frame library [173].

must be stored in a database for persistent storage between play sessions. ROLEPLAYINGGAME automatically performs a database-to-atom loading operation during startup, and does the opposite during shutdown.

**Commands**, listed in [Table 5.1](#), are operations performed by players or the GM<sup>6</sup>. Most commands are “exposed,” i.e. directly usable by an end user; the few outliers exist to simplify implementing other user-facing commands which offer a better UX<sup>7</sup>. At time of writing, all commands alter game state; future work could add ones that don’t, for tasks like setting user preferences (e.g. toggling between light and dark display themes.) Players have access to only one command, `take-action`, with which they issue an action for their active PC; the other parts of the player GUI provide information with which players decide what actions to take next<sup>8</sup>. The GM has access to about a dozen commands, which expose useful manipulations of game state without regard for rules. GM commands disregard rules for several reasons:

1. Malleability requires that the GM be able to create or edit rules at runtime; if commands are able to alter rules, they must exist “outside” the rules.
2. Rules are only intended to apply to character action. How the GM manipulates and extends the world during play is a matter of his taste as a game designer.

---

<sup>6</sup>While this list would be insufficient to run a real game using ROLEPLAYINGGAME, it’s enough to demonstrate the generality and extensibility of this approach.

<sup>7</sup>For instance, if `activate-rule` and `deactivate-rule` were directly exposed to a user, he would have to know whether the rule was currently active before choosing to use the former or the latter. That imposes unnecessary cognitive burden on the user, and addressing that burden would force me to build an interface component outside the sidebar just to visualize rule status. To avoid both issues, I implemented the higher-level command `toggle-rule` in terms of those lower-level ones. Then, in the GM sidebar, I gave `toggle-rule` an idiosyncratic UI which displays the status of all rules at a glance, and allows toggling them on or off with one click.

<sup>8</sup>See [Subsection 5.5.1](#) and onward for user interface screenshots.

Name	Exposed?	Description
activate-rule	no	Activate an inactive rule, imposing an additional constraint on character actions.
add-coord-based-flavortext	yes	Enhance a zone hex with differential display text.
create-npc	yes	Make a new NPC and adds it to the game world.
deactivate-rule	no	Deactivate an active rule, removing a constraint on character actions.
define-race	yes	Add a character race to the game system's domain model (i.e. its taxonomic facts.)
define-rule	yes	Add a rule to the game system.
describe-npc	yes	Generate a 2nd-person description ("you see...") hinting at an NPC's stats, using a custom ChatGPT prompt.
dislocate	yes	Set entity's <code>:location</code> attribute to <code>nil</code> , meaning its precise location is undetermined.
equip	yes	Put an item into a character's equipment slot, e.g. <code>:left-hand</code> .
give-item	yes	Place an item into a character's inventory.
relocate	yes	Move an entity to a location.
take-action	yes	Order a character to take an action. This is the only command available to players. The GM can use it to act as an NPC without violating the active rules.
toggle-rule	yes	Activate a rule if it's currently inactive, and vice versa.
unequip-by-id	yes	If the item with a given ID is in any of a given character's equipment slots, remove it (but leave it in their inventory).
unequip-by-slot	yes	Empty a particular equipment slot of a specified character.
ungive-item	yes	Remove an item from a character's inventory.

Table 5.1: User commands in ROLEPLAYINGGAME. In running text, commands link back to this table, and are highlighted (e.g. `toggle-rule`) to distinguish them from actions.

3. When a GM declares that an NPC is taking action, he uses `take-action` on that NPC, just as a player does for her PC. This ensures that rules apply equally to PCs and NPCs. In cases where the appropriate action is not yet implemented, ideally the GM would exercise malleability by using a command to add that action to the system, then having the NPC take that newly created action. However, there is not yet a command to add an action; furthermore, some actions would be infeasible to code in real time. In those case, the GM can still use commands to edit the world as if that action were available, as long as he takes care to manually make all necessary game-state changes<sup>9</sup>.

**Actions**, listed in [Table 5.2](#), are in-world activities carried out by a PC or NPC. The availability of a given action for a given character depends on the current game state<sup>10</sup>. For instance, the `pick-up` action is only available when the character has a free hand and is positioned next to an item. The connections between action availability and the current game state are established by **rules**. Each rule definition declares a predicate function implementing a concrete implementation of Boolean-valued statements like “has a free hand” or “is positioned next to an item.” Each action specification, in turn, declares which rules govern its own availability, and how the action’s parameters should be passed as arguments to those rules’ predicate functions.

The arguments and effects of each command and action are defined by a corresponding “interpretation function.” When a user issues a command with a chosen set

---

<sup>9</sup>Making it easy for the GM to take care of all the details of these exceptional cases is a UI-design and system-architecture challenge, one which I didn’t have time to try characterizing or solving during the development of ROLEPLAYINGGAME.

<sup>10</sup>See [Subsection 5.5.1](#) for screenshots of the player UI for selecting actions.

Name	Description
buy	Purchase an item from a location that has a market.
move	Move between hex coordinates in a zone.
pick-up	Pick up a nearby item.

Table 5.2: Character actions in ROLEPLAYINGGAME. In running text, actions link back to this table, and are highlighted (e.g. `move`) to distinguish them from commands.

of arguments from his or her GUI, the ROLEPLAYINGGAME server re-validates the command<sup>11</sup>. Assuming the validation check passes, the server executes the command’s interpretation function, which processes the input arguments and returns a sequence of events<sup>12</sup>.

**Events** are data structures specifying minimal meaningful changes which can be made to the world. Just as each command has an interpretation function, each event has an update function, which alters game state when executed.

**Kinds** are the categories of game world content which an instance of ROLEPLAYINGGAME can process<sup>13</sup>. Every piece of game world content in ROLEPLAYINGGAME is represented as a key-value map. Each such map must at minimum contain the `:kind` attribute, the value of which must be one of the kinds defined by the game system<sup>14</sup>.

ROLEPLAYINGGAME’s kinds are arranged in a hierarchy (concretely, a key-

---

<sup>11</sup>This is standard practice in web development to guard against malicious or broken clients that send incorrect data.

<sup>12</sup>The interpretation function for the `take-action` command treats its first argument as the name of an action, and passes remaining command arguments down to that action’s interpretation function — which processes the arguments and returns a sequence of events. The parallels between action processing and command processing are deliberate.

<sup>13</sup>The term “kind” here has no relation to the concept of “kinds” in programming language theory’s study of type systems.

<sup>14</sup>Using maps where other languages might use objects or structures, and including a key-value pair identifying what domain-specific content the map holds, are both standard Clojure practices.

value map) which can be used to determine whether one kind is considered a child of another<sup>15</sup>. Both the set of allowable kinds, and the relations between them, are instances of taxonomic facts in a game’s design; accordingly, these are included in the game system portion of game state<sup>16</sup>.

### 5.3 Game World Content

Having discussed how kinds are implemented, I can now explain the kind hierarchy, the members of which equate to the types of game world content that ROLEPLAYINGGAME can process and represent. Kinds can be *abstract* (never directly instantiated) or *concrete* (directly instantiated.) The most fundamental kinds supported by ROLEPLAYINGGAME are entities, trade goods, and locations.

**Entity** is an *abstract* kind. It has one concrete subkind, **item**, and one abstract subkind, **character**. The intended distinction between items and characters is that items are generally inanimate and characters are generally animate<sup>17</sup>. The character kind is divided into two concrete subkinds, **PC** and **NPC**.

A **trade good** is *not* an individual item existing in the world. Rather, it is a template of the attributes, including the price, of one type of ideal, brand-new item being sold at an in-game market, such as a sword, a cake, a glass jar, or a bucket of blue

---

<sup>15</sup>Clojure’s notions of hierarchy and inheritance are similar to those of OOP, save that (a) hierarchical relations can be defined between heterogeneous values (e.g. keywords or strings), not just between OOP classes; and (b) programmers can freely create new hierarchies, with each embodying a different set of child-parent relations.

<sup>16</sup>At present, no ROLEPLAYINGGAME features exercise the malleability of the kind hierarchy. I spent a little time on it anyway in the hopes of laying a foundation for future work.

<sup>17</sup>I say “generally” because these divisions might not hold up in a fantasy-game context.

paint. When a character takes the **buy** action on a trade good, a new piece of content (i.e. key-value map) is created to represent the actual thing existing in the world. That key-value map has `:kind` set to `:item`, but otherwise inherits many of the trade good's attributes.

ROLEPLAYINGGAME distinguishes between an item template (i.e. trade good) and a particular in-game item created from that template because I have done extensive prior work on a generative economy system for my personal game system (see [Subsection 2.7.2.](#)) ROLEPLAYINGGAME partially integrates with that economy system by reading its output database during startup; for more on that, see [Subsection 5.5.8.](#)

Finally, virtually all game systems need a way to play out precise character-scale movement, but otherwise games vary widely in the level of detail assigned areas of the game world not used for that purpose. As such, ROLEPLAYINGGAME provides two concrete types that a GM can mix together to represent broad and specific areas of his game world. These are **locations** and their subtype, **zones**.

All of the locations and zones in the game world together form the location tree (see [Subsection 5.5.4.](#)) Zones are leaves in that tree. They represent areas of the game world with a known extent, and which are suitable for character-scale movement and positioning. As such, zones are ROLEPLAYINGGAME's equivalent to battlemaps.

Locations are non-leaf nodes in the location tree. They represent areas of the game world which are large enough that the game system is insensitive to their precise extent. Zones can serve as battlemaps for precise positioning because they are



subdivided into hexes: cells representing character-sized chunks of 2D terrain. Thus, when a character is in a zone, or when the GM is viewing a zone, that zone can be rendered in a battlemap-style display.

ROLEPLAYINGGAME distinguishes locations and zones to demonstrate how a platform could support game systems with radically different approaches to modeling game-world space. A “globetrotting spies” game system might start with a “Planet Earth” location, with the GM then adding locations for countries, provinces, cities, and buildings, and zones for individual floors in a building. A storytelling system about Chinese rural migrants might only need “Beijing” and “The Countryside” as two distinct locations, with no zones whatsoever because precise character movement is not part of the game system. A game setting consisting of a single apartment building might have one top-level “Building” location for bookkeeping purposes, with zones for each apartment — or it might have locations for each floor of the building, with the central hallways and the apartments on each floor being zone children of the floor location. ROLEPLAYINGGAME can handle all these cases.

## **5.4 Features Supporting Malleability**

### **5.4.1 Malleability of Substantive Facts**

Malleability of substantive facts is fairly common across VTTs and RPG support software. I included this feature in ROLEPLAYINGGAME to show that implementing novel features did not preclude the addition of standard ones.

The GM has several commands at his disposal for manipulating substantive facts. For instance, he can `relocate` entities to different locations in the game world. He can also `dislocate` them, which keeps them in the game world, but without a specified location. This is useful for preparing entities to be used at a future time, and for modeling the fact that an entity is no longer present in a location without forcing the GM to make an up-front decision about exactly where the entity went.

With the `create-npc` command, the GM can insert new non-player characters into the world, i.e. he can add one kind of substantive fact. He can also put items into characters' inventories with `give-item`, and place items into characters' equipment slots with `equip`; both of these have counterparts that do the opposite. Finally, the GM can use `add-coord-based-flavortext` add descriptions which become visible to players if they move to certain parts of the game world (see [Subsection 5.5.2](#), below.)

#### 5.4.2 Malleability of Taxonomic Facts

As a proof of concept, I implemented a command which allows the GM to expand the game system's domain model at runtime. With `define-race`, the GM can add a new race (i.e. species) of characters to the game world. In terms of DCG, this increases the game system's granularity, because it expands the range of legal values for the `:race` attribute of characters. The design of `define-race` could be reused for other commands, each of which would make a new aspect of a `ROLEPLAYINGGAME`-based game system malleable. See [Section 7.3](#) for more.

### 5.4.3 Malleability of Rules

The GM can suspend or resume the automated checking of particular game rules, with corresponding changes in action availability being immediately apparent to the players. For instance, [Figure 5.1](#) shows a player’s UI after choosing the `move` action. She can now click on the specific hex to which she wishes to move her PC; green hexes are valid choices, and gray ones are not, as determined by the evaluation of one of the rules governing the `move` action. If the GM uses his `toggle-rule` command to switch off the rule-governing PC movement speed ([Figure 5.2](#)), the player’s UI immediately changes to show every non-occupied hex as green ([Figure 5.3](#).)

It is also possible for the GM to add new rules at runtime, using the `define-rule` command. The current implementation requires the GM to type in Clojure code, which will be converted into a string, sent to the server, and evaluated to create the body of the new rule’s predicate function<sup>18</sup>.

## 5.5 Functionality and Design of User Interfaces

### 5.5.1 Core UI Elements

ROLEPLAYINGGAME’s GM and player UIs are divided into three elements: the sidebar, the main area, and the navbar, the most important part of which is the tab selector ([Figure 5.4](#).) Each tab surfaces a different set of facts from the current game state. Because this information is presented or visualized in the UI’s main area, what

---

<sup>18</sup>See [Section 7.3](#) for discussion of potential improvements to this paradigm.

ARVAK [Log Out](#)

---

### What do you do?

move

Param location

Cavern

Param coord

### Cavern

*It is far colder here than outside. You get a whiff of the sulfurous stench of troll blood.*

You can see Birimor, Cuthbert and Derendil here.

Figure 5.1: After picking the **move** action, this PC can click any of the green hexes to fill the “coord” parameter shown in the sidebar.

DM Log Out

Reset Client State

### Change World and Rules

toggle-rule ▾

**Pick Up**

slot-empty?  Active  
A character needs an empty equipment slot (left hand or right hand) to equip or pick up an item.

not-held?  Active  
A character can only pick up an item if nobody else is holding it.

close-enough?  Active  
A character can only pick up items within N hexes, where N = the reach stat.

**Buy**

*No rules for this action yet.*

**Move**

not-occupied?  Active  
A character can only move to a coord if no other character occupies it.

within-movement-range?  Inactive  
A character can only move to a coord within N hexes, where N = the speed stat.

Figure 5.2: The GM uses `toggle-rule` to see the rules governing each action. From here, he deactivates the one called `within-movement-range?` (at bottom.)

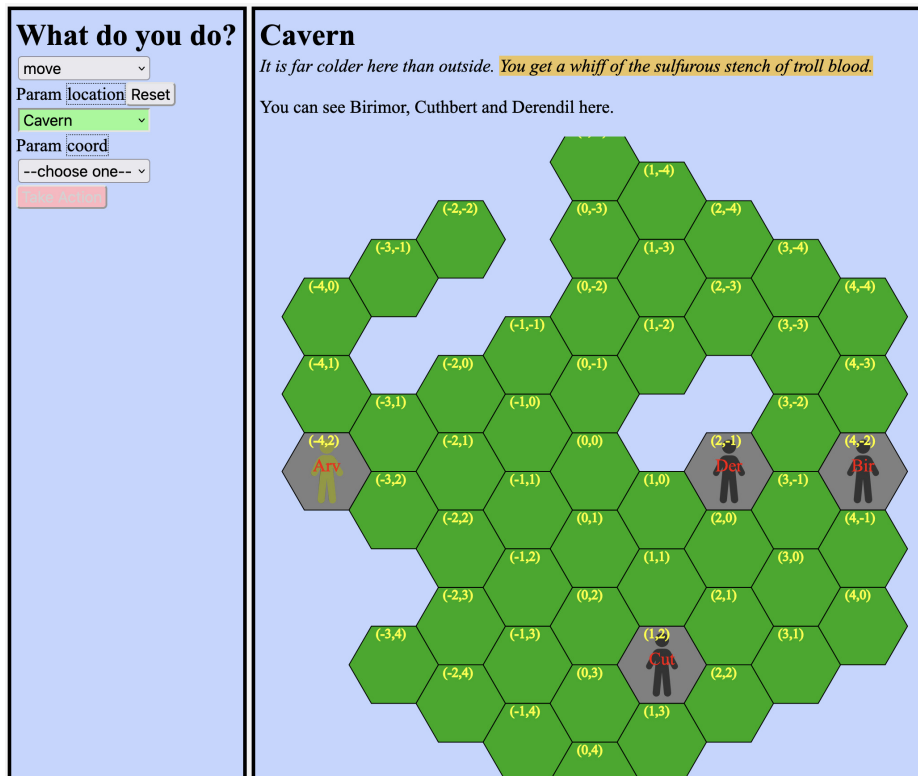


Figure 5.3: Once the GM has deactivated `within-movement-range?`, the PC's UI will update to show all non-occupied hexes as green (i.e. targetable.)

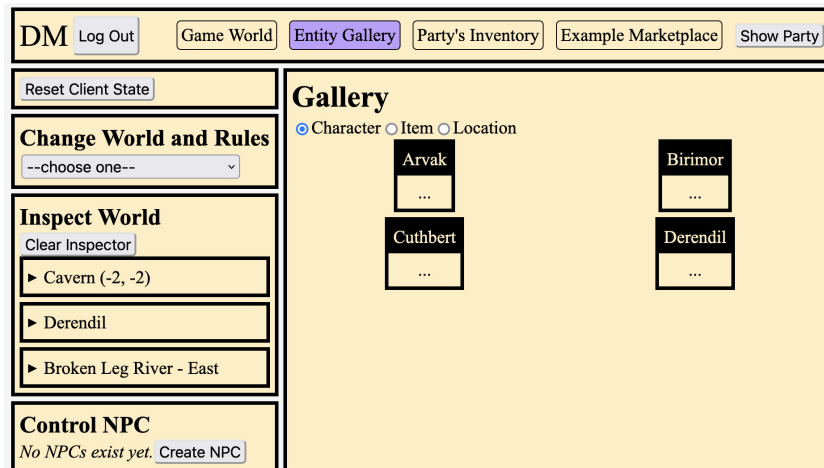


Figure 5.4: The GM’s UI in ROLEPLAYINGGAME, with the nav bar and tab selector at top, sidebar at left, and main area at right.

is seen in the main area changes from tab to tab. The sidebar, however, stays the same regardless of what tab is open. The sidebar is used while deciding what command<sup>19</sup> to take next, and what parameters to pass to that command. Because the sidebar retains state across tabs, a user can gather information from multiple tabs while deciding what to do next. She can also change her mind about what command to issue without losing the work she put into picking parameters. Parameters specified for command X will be retained even if she switches to picking parameters for command Y, so if she switches back to command X, her previous parameter choices will appear.

While the sidebar offers drop-down menus to choose commands and their parameter values, one of the key functions of the main area for each tab is to visualize clickable elements that can be used to populate the the sidebar’s menus. Choosing a parameter in this way, or changing to a different parameter, is faster than selecting

<sup>19</sup>The sidebar works exactly the same for actions as for commands; for simplicity, throughout this subsection I say “command” rather than “command or action.”

from the dropdown menus. It's also more intuitive for the user because he or she can directly interact with the visualized `ROLEPLAYINGGAME` entities; in a dropdown, by contrast, a richly-detailed entity object must be reduced to a textual name or description<sup>20</sup>. The dropdown menus serve as a fallback option for selecting entities not being visualized by any of the tabs. For instance, the GM can use the dropdowns to select and operate on a character which can't be interacted with in a visualized location because the character has been `dislocated`, i.e. had its `:position` attribute set to `nil`.

The information display in a tab's main area is sensitive to parameters that have already been chosen for a particular command or action. For instance, as covered in [Subsection 5.4.3](#), when a PC or NPC has selected the `move` action while in a zone, the hexes which are valid movement targets will light up with a gently-pulsing green, while the others dim to gray. The valid values for a command or action's yet-unchosen parameters are also dependent (where appropriate) on the already-chosen parameters. For instance, when the GM selects the `equip` command ([Figure 5.5](#)), the possible choices for the `slot` parameter will depend on which character is to receive the item ([Figure 5.6](#)) because characters don't necessarily possess the same equipment slots: perhaps one character has lost a hand, for instance.

---

<sup>20</sup>This might be improved with extensive customization of the HTML `<select>` tag that creates the dropdowns, or a JavaScript replacement of the same.



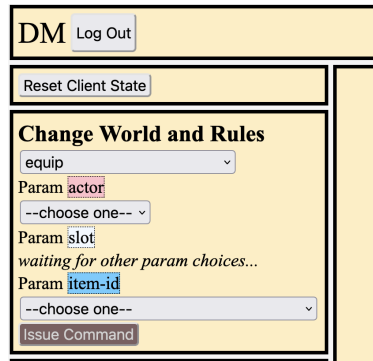


Figure 5.5: Before choosing a character with whom to `equip` an item, the `slot` parameter cannot yet be chosen.

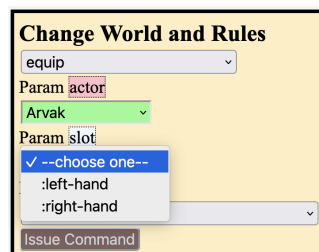


Figure 5.6: After the GM has chosen a character for the `actor` parameter, that character's valid choices for the `slot` parameter automatically become available in the corresponding dropdown menu.

### 5.5.2 Differential Display

**Differential display** is a means of displaying information only to players whose characters have particular stats, skills, or sensory capabilities<sup>21</sup>. It is intended to immerse the player in the experience of being a character that’s quantitatively and qualitatively different from the other characters; foster cooperation between players; and encourage players to attend to the game UI instead of getting distracted by other computer apps. The GM can add information (and predicates which must be satisfied to display it) to items, locations, and specific coordinate hexes within zones. At present, differential display only works by adding information; it can’t take anything away.

Figure 5.7 and Figure 5.8 demonstrate the appearance of differential display text in a player’s UI. The former shows a section of the main area of the explore tab, as seen by a player named Alice, whose PC is Arvak (top left.) The italicized text is the GM’s pre-written description of Arvak’s location, the Cavern. Some of the italicized text is also highlighted, and this highlighting indicates the presence of differential display. Alice can only see that portion of the Cavern’s description because her PC has some particular attribute or characteristic<sup>22</sup>. Compare this with Figure 5.8, in which a second player’s UI does not show the information which was displayed (and highlighted) for the first player. It’s up to Alice to communicate this information to the rest of her party.

---

<sup>21</sup>The term I originally used instead of differential display was “quality-based rendering,” after one of its inspirations, “quality-based narrative.” Quality-based narrative is a mechanism for interactive narrative systems which allows player access to chunks of a story only if the game state or character state matches certain predicates. Emily Short says [144] that the term was coined by Failbetter Games (the developers of *Fallen London* [43].) It has since been adopted by industry and academia for discussions of interactive narrative systems — though [144] also notes that Failbetter has ceased use of the term, since it seems to imply that other kinds of narrative structure are low-quality.

<sup>22</sup>In this case, it’s because Arvak has points in the Beast Hunter skill.

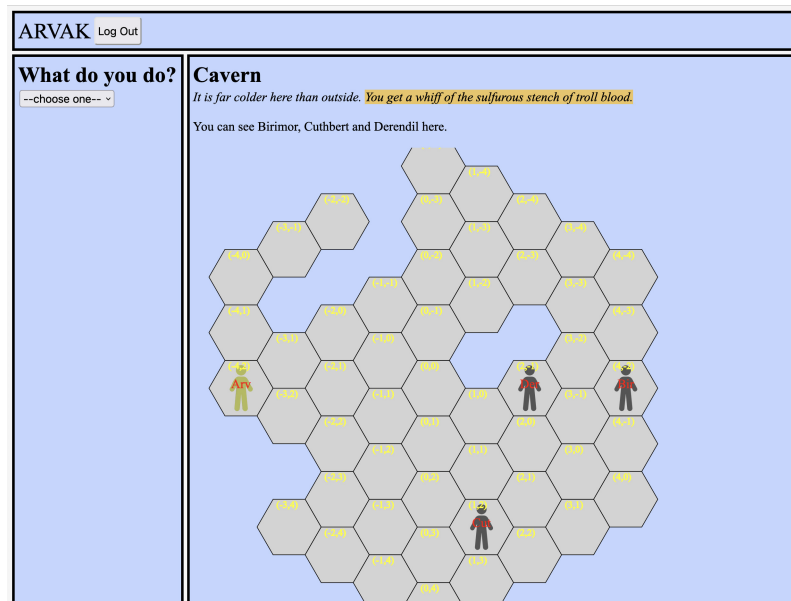


Figure 5.7: Alice (as Arvak) is on the Explore tab, viewing the Cavern zone. She sees bonus information (highlighted in yellow) through differential display.

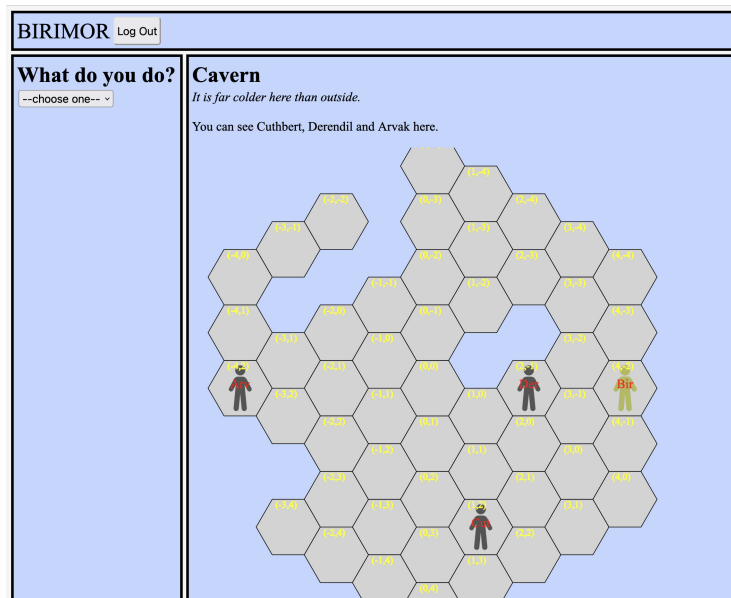


Figure 5.8: Birimor is also in the Cavern, but his player, Bob, doesn't see the bonus text that Alice saw as Arvak.

### 5.5.3 GM's Inspector

The GM's sidebar contains an area called the **Inspector**. This is a quick-access menu to which the GM can “pin” entities which he plans to interact with soon. With the Inspector, the GM can rapidly switch up the parameters he wants to use for a given command. Throughout the ROLEPLAYINGGAME interface, on-hover highlighting (Figure 5.9) is used to indicate UI elements that are interactive<sup>23</sup>. When the GM clicks a UI element representing a game object, it gets pinned to the Inspector (Figure 5.10.) From there, the GM can view all the command parameters which accept that object's kind; parameter names here and in the upper area of the GM sidebar are colored with the same hashing algorithm to aid in visual identification. (Figure 5.11.) Clicking a parameter listing will both make that command active in the GM's sidebar, and set that parameter to the object in question (Figure 5.12.).

### 5.5.4 Game World Tab

The GM's Game World tab shows him the containment relations between larger and smaller locations, and enables him to view any of the game world's locations and zones. Of particular importance is the fact that the code which renders a zone on this tab is almost identical to the code that renders a zone for players on their explore tab. This is useful to the GM in several ways: it ensures that he can easily add PCs and other zone contents to his Inspector; it allows him to observe the PCs as they take actions within the zone; and it makes it convenient to pick a coordinate

---

<sup>23</sup>This convention is not used when interactivity is obvious from the element type alone, e.g. buttons and hyperlinks.

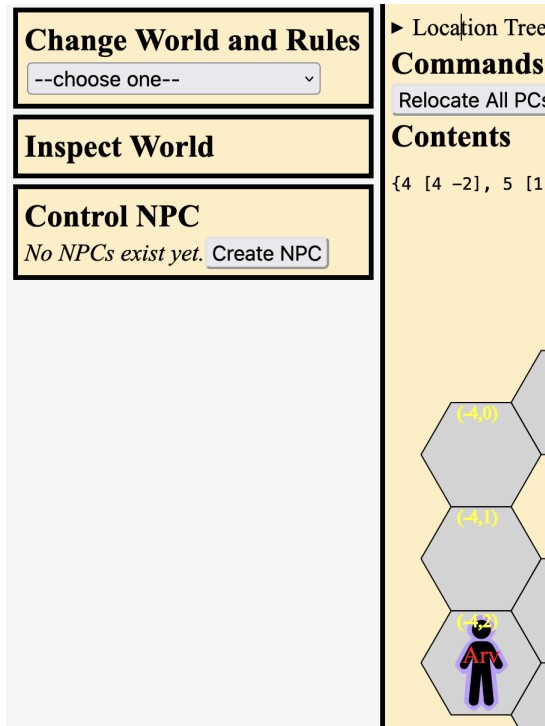


Figure 5.9: The GM is hovering his mouse pointer (not shown) over Arvak, causing that PC's battle map avatar to glow purple (bottom right.) This indicates interactable content.

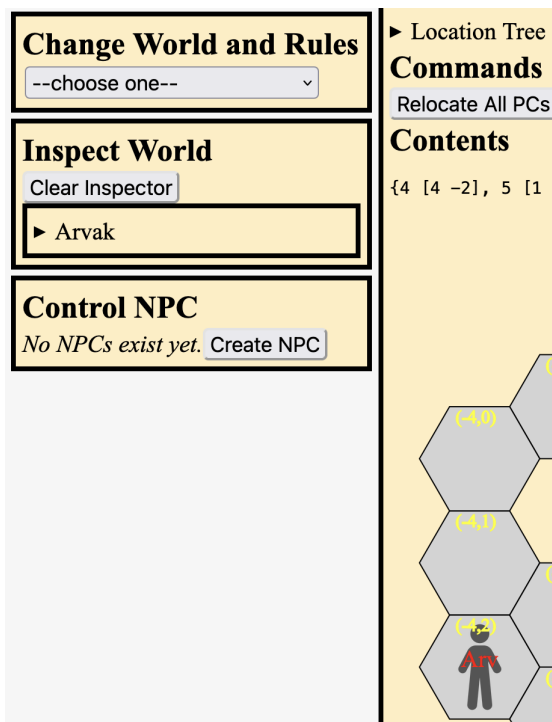


Figure 5.10: After the GM clicks Arvak, his name appears in the Inspector (at left.)

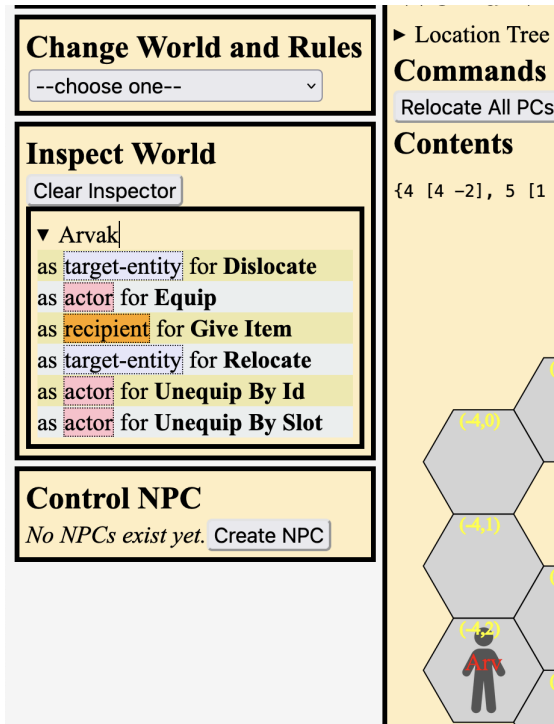


Figure 5.11: Clicking Arvak's entry in the Inspector shows the GM every command that can operate on a PC, and what parameter the PC would fill for that command. Each item in this list is interactive, and will highlight in purple when hovered (not shown.)

### Change World and Rules

relocate

Param **target-entity**

Arvak

Param **location**

--choose one--

Param **coord** (optional)

*waiting for other param choices...*

Issue Command

### Inspect World

Clear Inspector

▼ Arvak

- as **target-entity** for **Dislocate**
- as **actor** for **Equip**
- as **recipient** for **Give Item**
- as **target-entity** for **Relocate**
- as **actor** for **Unequip By Id**
- as **actor** for **Unequip By Slot**

### Control NPC

*No NPCs exist yet.* Create NPC

► Location Tree (

### Commands

Relocate All PCs

### Contents

{4 [4 -2], 5 [1 2

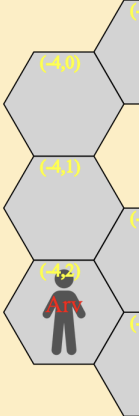


Figure 5.12: The GM clicks the item for **relocate**. This sets the GM's sidebar to begin specifying that command, and fills in Arvak as the **target-entity** parameter.



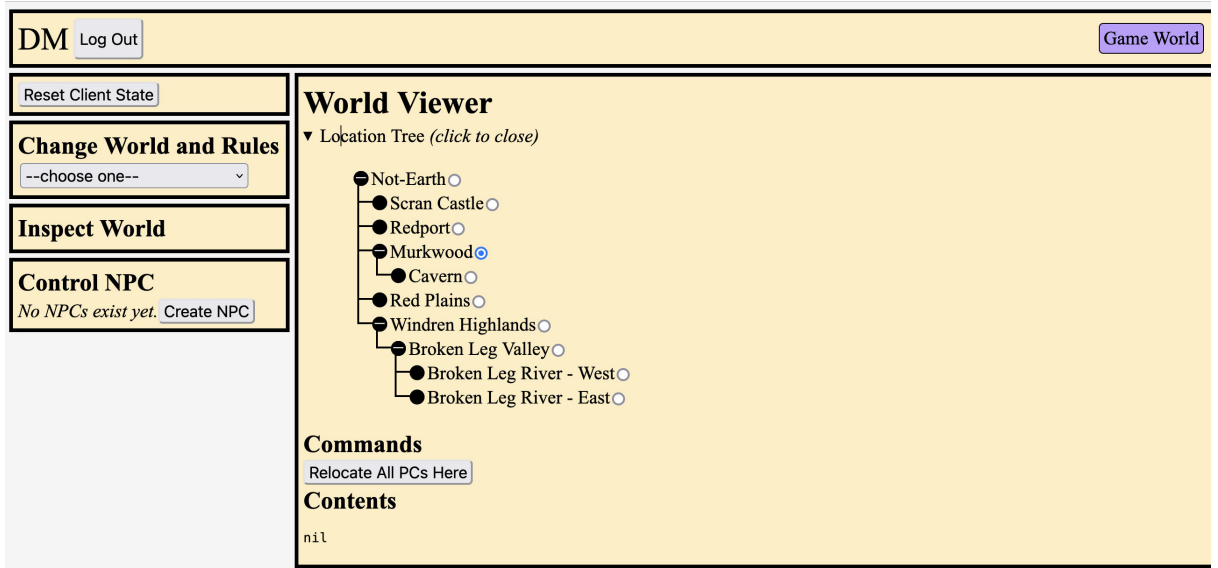


Figure 5.13: The GM views a location, Murkwood, in his Game World tab. Notice the button which allows him to relocate all PCs to this location.

for the `add-coord-based-flavortext` command, with which the GM can author new differential-display text on the fly.

### 5.5.5 Gallery Tab

The Gallery is a tab on which the GM can view substantive facts, i.e. entities or locations (and subtypes of both) that have been declared to exist in the game world. The visualization in this tab is merely a placeholder: at-a-glance display of entity or location attributes would be desirable. However, at present the Gallery does allow the GM to switch between different kinds of substantive facts, and clicking on one will add it to his Inspector.

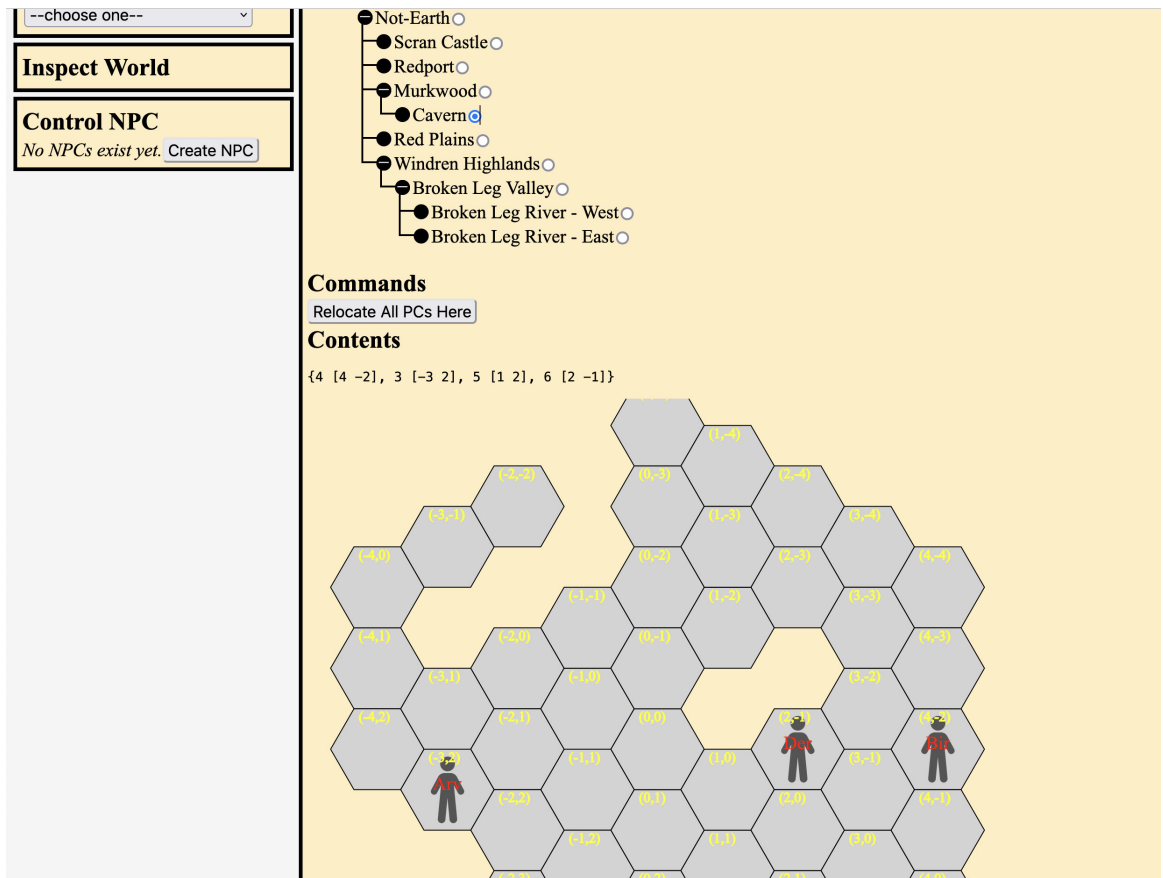


Figure 5.14: The GM views a zone, Cavern, in his Game World tab.

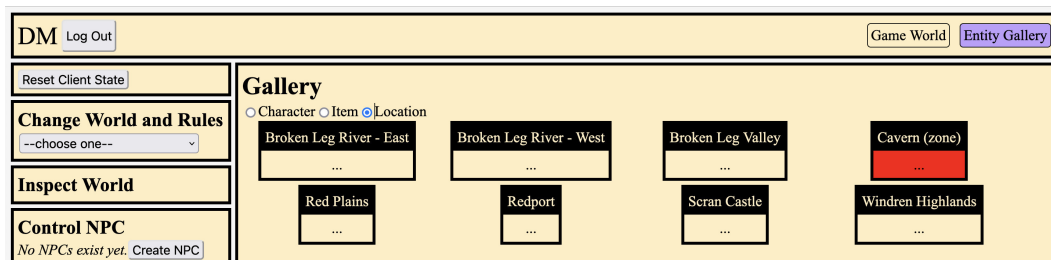


Figure 5.15: The GM views locations on the Gallery tab. Zones are colored in red.

### 5.5.6 Explore Tab

The Explore tab (see [Figure 5.7–Figure 5.8](#)) represents a PC’s immediate surroundings, i.e. the location or zone named by her `:position` attribute. PCs, items, and NPCs in the same location are visible to her on this tab, as are hex coordinates if the location is a zone. The Explore tab is most useful when players wish to move around inside a zone, traveling from hex to hex with the `move` action.

### 5.5.7 Social Inventory Tab

The Social Inventory tab ([Figure 5.16](#)) displays what each PC is carrying. It’s called “social” because each PC (and the GM) can see the whole party’s possessions. This design is intended to model the familiarity that each PC would have with each other, and with each other’s possessions, after much time spent adventuring together<sup>24</sup>.

This tab does more than just visualize possessions, however. It also offers information-retrieval functionality which addresses a complaint raised by players in my home campaign. My players use Excel spreadsheets to track their PCs’ inventories, but they felt that searching those spreadsheets was error-prone and time-consuming. For instance, finding out whether someone in the party had a particular item required getting everyone’s attention to do a party-wide search through PC inventories.

Therefore, as shown in [Figure 5.17](#), the Social Inventory allows users to filter the party’s inventories, making matches more visually salient than non-matches. Filters can select for items having a certain category tag, items which match a text string, or

---

<sup>24</sup>Thanks to Alex Bakker for originally suggesting, in private correspondence, the idea of using a shared inventory to represent intra-party familiarity.

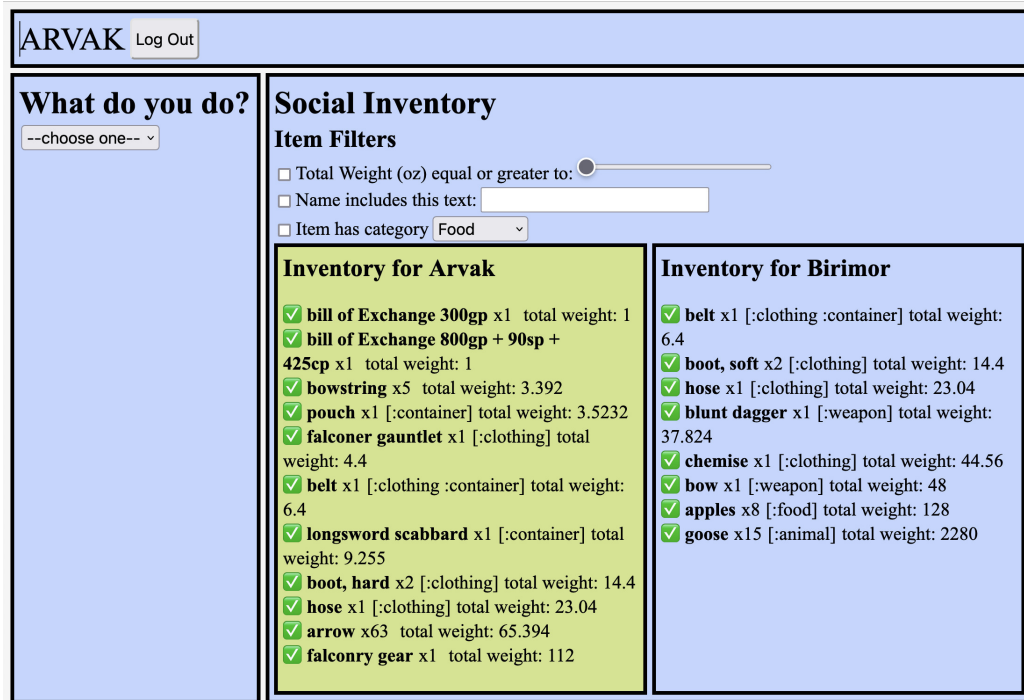


Figure 5.16: A PC, Arvak, views the Social Inventory. His pane is displayed in green, which is used throughout ROLEPLAYINGGAME to mean “owned by or related to the logged-in user.”

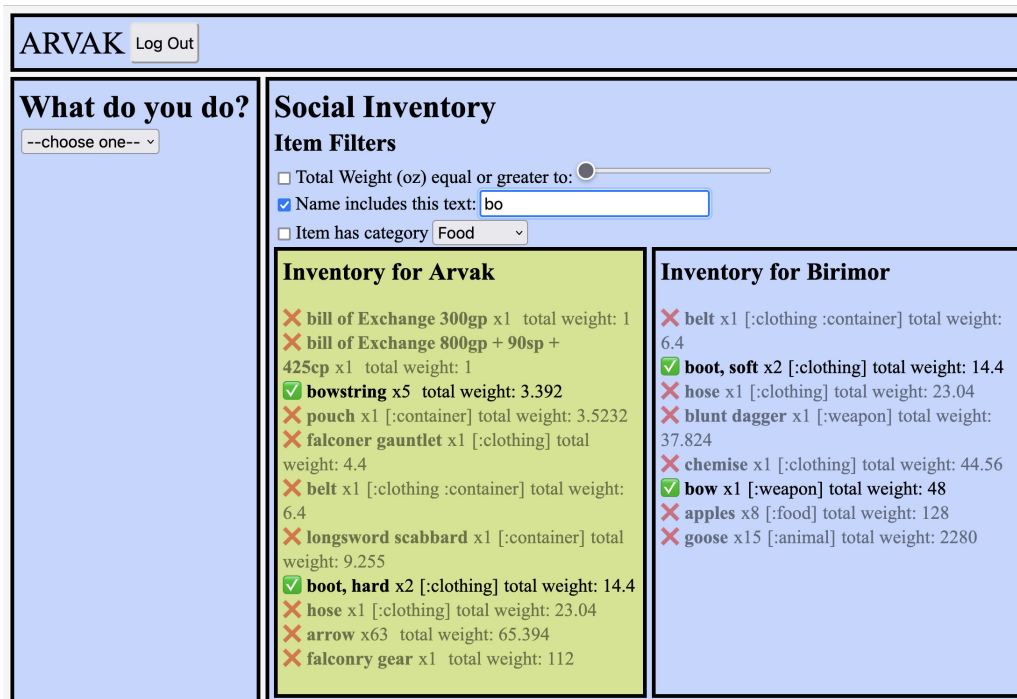


Figure 5.17: Arvak filters to include just those items whose names contain the string “bo.” Matching items are labeled with a checkmark; non-matching items are dimmed, and labeled with an X.

items which have a weight equal or greater than a chosen value. This feature, though simple, is nevertheless beyond the reach of a pencil-and-paper game, and would be tedious to mimic even in a game where players used computers but didn’t have a central platform for info management.

Informal user testing with two players from my home campaign revealed that they were excited about the possibilities presented by this interface. Those two players said that, aside from lessening the burden of manual inventory review, this unfettered view of the whole party’s possessions would allow them to more easily make suggestions to other players. As one player put it, “It’s easy to know what my own character has,

but I rarely remember what everyone else is carrying. This would be super helpful for thinking of plans which could make use of several characters' equipment.”

### 5.5.8 Market Tab

The Market tab (Figure 5.18) is where a player can **buy** new items for her PC (as long as the PC's current `:position` is a location with a market.) The Market tab demonstrates that `ROLEPLAYINGGAME` can readily integrate with external information sources. As mentioned in Subsection 2.7.2 and Section 5.3, the data for this tab comes from an economic simulation project which I developed for my home campaign before writing this thesis. The output of that simulation gets stored in a SQLite database file; when `ROLEPLAYINGGAME` starts up, a Clojure function loads trade good information from that database into game state.

### 5.5.9 Character Tab

The Character tab displays a PC's attributes. Both players and the GM can access this tab for any of the PCs in the party. I created this tab for completeness' sake, but it's little more than a placeholder, because displaying a character sheet is hardly novel.


## 5.6 Summary

With this chapter, I've given a comprehensive overview of the user-facing features of `ROLEPLAYINGGAME`, and shown that it has both user interface affordances

ARVAK [Log Out](#) [Visit Market](#) [Your Location](#) [Your Character](#)

**What do you do?**  
--choose one--

**Marketplace for Redport**  
Armorer



Name	Price	Unit	Gross Weight	Description	
coat of plates	31 gold	1 item	14.74 pound	AC +3; as leather armor, but reinforced with metal plates between the leather layers	<a href="#">Buy</a>
coat of plates, small	20 gold	1 item	9.458 pound	AC +3; as leather armor, but reinforced with metal plates between the leather layers	<a href="#">Buy</a>
full-plate	132 gold	1 item	48.29 pound	AC +7; steel armor covering the wearer from neck to ankle, with full metal sleeves	<a href="#">Buy</a>
full-plate, small	84 gold	1 item	30.99 pound	AC +7; steel armor covering the wearer from neck to ankle, with full metal sleeves	<a href="#">Buy</a>
gambeson	17 silver	1 item	9.010 pound	AC +1; thickly-layered cloth armor with full sleeves, covering wearer to the waist	<a href="#">Buy</a>
gambeson, embroidered	132 gold	1 item	9.088 pound	gambeson with embroidered designs as desired, up to 64.62 sq_in total	<a href="#">Buy</a>
gambeson, embroidered, small	85 gold	1 item	5.832 pound	gambeson with embroidered designs as desired, up to 64.62 sq_in total	<a href="#">Buy</a>
gambeson, padded	19 gold	1 item	15.02 pound	gambeson with extra padding; layers of flannel increases interval for chafing damage from 2 hours to 4	<a href="#">Buy</a>
				gambeson with extra padding; layers of flannel increases interval for chafing damage	

Figure 5.18: A PC, Arvak, visits the Market tab for the location Redport.

ARVAK [Log Out](#)

**What do you do?**  
--choose one--

You are playing  
**Arvak**

Ability Stat	Score
Strength	15
Dexterity	8
Constitution	13
Intelligence	9
Wisdom	13
Charisma	12




**Money**  
 GP  100  
 SP  50  
 CP  13  
**Avatar**

Figure 5.19: Alice checks the Character tab for Arvak, her PC.

and a level of malleability support which are unprecedented in fields which combine elements of computer gaming and traditional RPGs. In the next chapter, I present the results of a user study with RPG GMs and players, in which I get their feedback on ROLEPLAYINGGAME, and discuss topics including computer augmentation, user interface woes, opinions of VTTs, and the desirability of high-DCG game systems.



## Chapter 6

# Community Feedback

To receive feedback on ROLEPLAYINGGAME, I conducted freeform interviews with RPG enthusiasts of my acquaintance. This preliminary user study was intended to determine whether ROLEPLAYINGGAME's features were comprehensible, useful, and intriguing for GMs, players, and/or game designers whose tastes, backgrounds, and experience levels differed from mine. In this chapter, after discussing the design of the study and the interviews, I identify key themes that arose while the interviewees and I discussed ROLEPLAYINGGAME and related topics.

Interviewees' reactions to ROLEPLAYINGGAME and its capabilities were generally positive, ranging from partial acceptance to genuine exuberance; each of them found multiple aspects of ROLEPLAYINGGAME to be exciting and intriguing. The interviewees felt that ROLEPLAYINGGAME offered a compelling vision for RPG support software, and showed unambiguous advantages over existing tools for assisting with such tasks as forming plans; understanding the current game state and changes made

Interviewee ID	Background	Years RPGing	Years GMing	# Familiar Game Systems
P1	math, CS, data viz (PhD)	14	11 (79%)	>5
P2	astrophysics (PhD)	<2	none	1
P3	history, library science (MA)	20	15 (75%)	>5
P4	anthropology, theater (BA)	20	10 (50%)	>5
P5	mechanical engineering (PhD)	9	7 (78%)	2–3

Table 6.1: Interviewees’ professional backgrounds; the number of years they’ve spent participating in RPGs; and the portion of those years they’ve spent GMing.

thereto; employing game rules with high connectedness; and sharing secret information with select players. Interviewees also felt that `ROLEPLAYINGGAME`, or a future version of it, could help them reach their own RPG design goals. Interviewees also gave frank critiques: they said the UI needed more work to be table-ready for players and GMs, and one was particularly concerned about the feasibility of the GM keeping the in-system game state updated during full-speed live gameplay.

## 6.1 Study Design

There were 5 interviewees in total. All but one of them (P2) had substantial experience with roleplaying games in general, and as GMs specifically<sup>1</sup>. Two of them (P1 and P5) had also spent time developing personal or academic software projects in the realms of RPG support and game data visualization. A summary of interviewees’ RPG experiences are given in [Table 6.1](#).

Each interview was conducted over Zoom, and lasted about 90 minutes. Key

---

<sup>1</sup>P2 not only had no GMing experience, but was also relatively new to being an RPG player. That was my mistake: I misunderstood P2’s background during recruitment. However, P2’s feedback still proved valuable as a gauge of whether inexperienced RPG players (not GMs) might see potential in `ROLEPLAYINGGAME`.

quotes from those interviews, and an explanation of how transcripts were created, are in [Appendix A](#); full transcripts are available in the supplement to this thesis. After getting informed consent for the interview to be recorded and transcribed, each interview began with showing the interviewee two videos, both about five minutes in length. I named these videos **Motivation** and **Demo**.

Motivation’s content echoed [Chapter 2](#) of this thesis: it briefly introduced my game design philosophy (with example rules and images from my home game), then segued into describing RPG characteristics which have been difficult to capture in a computer. The goal of showing interviewees the Motivation video was to help them understand the kinds of games for which ROLEPLAYINGGAME was originally envisioned as a support tool.

Demo was a pre-recorded demonstration of numerous ROLEPLAYINGGAME features within a frame narrative (delivered by an accompanying voiceover) of a GM using ROLEPLAYINGGAME to run a game scenario for four players. The goal of showing interviewees the Demo video was to showcase ROLEPLAYINGGAME’s capabilities, demonstrate ROLEPLAYINGGAME’s applicability to common gameplay tasks, and illustrate that ROLEPLAYINGGAME’s novel architecture and design had permitted the development of features not found in similar programs, especially VTTs.

In the [Appendix A](#) interview transcripts, I have annotated key points from each interview with bracketed numbers. Hereafter, whenever I characterize specific interviewees’ responses, I cite those key points for support. The citation convention is that [\[P1:12\]](#) indicates the 12th key point in P1’s interview, to which I’ve appended

a “[12]” annotation. In the PDF copy of this thesis, those citations are hyperlinked to their corresponding transcript locations; alternatively, that citation can be found manually by visiting the P1’s transcript, then searching for a “[12]” note at the end of a line.

## 6.2 Connections to Game Design and Software Development

One of the largest perspective changes which I got from these interviews came from P1’s proposal that ROLEPLAYINGGAME’s actions and events, taken together, form something like a storylet system [P1:17]. A **storylet** comprises a piece of game world content, prerequisites for presenting that content, and updates that will be applied to the game world state after the content is presented<sup>2</sup>. In the case of ROLEPLAYINGGAME, the piece of content is the action itself; the prerequisites are the rules (predicate functions) about game world state which must be satisfied for an action to be presented in a player’s UI sidebar; and the state updates are calculated by the action’s interpretation function after the player has used `take-action` to submit the action (and any arguments) to the server.

P1, being steeped in the academic literature on game design, had other suggestions for how to view ROLEPLAYINGGAME in the context of certain areas of study, and how to frame future publications about ROLEPLAYINGGAME:

---

<sup>2</sup>See [144] for an introduction to storylets and a discussion of how storylets can be used to structure interactive narratives. See [85] for a comparison of storylet-based design systems.

1. They suggested that the DSL-like aspects of ROLEPLAYINGGAME (e.g. command and action definitions) could pave the way for publishing ROLEPLAYINGGAME-related material as an HCI study in making DSL-based design accessible to non-programmer GMs [P1:12], or as an application of PL theory to game design [P1:13].
2. The flip side of the prior proposal would be to publish in the game design literature, showing its practitioners some uses of DSLs and other aspects of PL theory [P1:14].
3. They pointed out a direct correspondence between the design of ROLEPLAYINGGAME's actions, and the verb-definition facilities of the Inform 7 language [P1:18].
4. After pointing out commonalities between *LambdaMOO*, Inform 7, and assorted ongoing efforts to develop the virtual-reality metaverse, P1 suggested that this thesis could support arguments for including malleability as a foundational component in a metaverse ([P1:19], [P1:20].)
5. They observed that the concept of malleability helps identify and define an underexplored, or even unexplored, aspect of what GMs do and say while running RPGs [P1:22].

### **6.3 ROLEPLAYINGGAME Presents a Compelling Vision for Introducing Computer Augmentation to RPGs**

Between interviewees' responses to ROLEPLAYINGGAME's features, and their explanations of what tasks they would like to delegate to computers during play,

it seems clear that the interviewees considered `ROLEPLAYINGGAME`'s conception of the future of RPGS to be at once fascinating and believable. Most notably, everyone loved the notion of differential display, and it was frequently the first thing an interviewee would comment on (e.g. [P3:1], [P2:1].) P3, P4, and P5 also found my examples of “more widespread differential display” desirable and intriguing for their potential to guide the act of roleplaying with hard systemic constraints ([P3:17], [P5:21], [P5:22].) Our discussions left me convinced that computer augmentation would serve needs, voiced by ordinary GMs, that span all roles a GM fills in an RPG session<sup>3</sup>. Although there is still much work to be done in this field, the interviewees felt that `ROLEPLAYINGGAME` represents a step in the direction of meeting their needs as designers and GMs.

### 6.3.1 Complexity and Subgames

One topic I brought up with my interviewees was whether they had ever designed or found RPG systems which they wanted to use, but which seemed too complex to feasibly play. I raised this topic to assess whether real-world GMs felt unable to achieve their game-design goals, and to start a conversation about how software could aid them in reaching those goals. I found out that interviewees had indeed come across rules which seemed desirable, yet impractical. Furthermore, they had also encountered the converse situation, finding themselves struggling with the task of rule design even in cases where the finished rule would be no trouble at all to run at the table.

P4 had much to say on this subject, starting with a desire for a live-updating

---

<sup>3</sup>See [Section 2.1](#).

world economy for which even attempting an implementation seemed futile: “that would never have gotten started [P4:11].” After that, P4 explained that they and P3 both like the game *Traveller* [111], but that the two of them have never run it, due to the difficulty of making science-fiction settings that the players find both realistic and compelling [P4:13]. P4 also suggested that *Traveller*’s complex rule subsystems would be a perfect use case for components of a software app that each deal with one aspect of gameplay which is so deep that it feels like a game within the game [P4:14]. They felt that “having many games in a game ... expands a world or makes it seem more real [P4:15].”

I intend to discuss P4’s comments at length. Their opinion — that computer augmentation would benefit RPGs by providing interfaces and tools specialized for certain portions of gameplay — has connections to points raised by other interviewees, and direct implications for the relevance of my research questions to RPG designers. Before continuing, however, I will first ground that discussion in a more formal (albeit provisional) definition of P4’s “game [with]in a game,” which I’ll refer to as a **subgame**.

Hereafter, I use “subgame” to mean part of the overall game’s gameplay which, though connected<sup>4</sup> the rest of the game, feels like a distinct mode of player interaction<sup>5</sup>. Subsystems used within a subgame may or may not be specific to that subgame alone, but the moves, actions, or operations available to players during the subgame almost

---

<sup>4</sup>In the DCG sense: see [Subsection 2.6.1](#).

<sup>5</sup>At least within this thesis, I distinguish subgames from several other concepts that may seem similar. Minigames are the subset of subgames which are completely divorced from the rest of the game: they have zero connectedness to any game state outside themselves. A subsystem (of a game system) is a collective term for rules that jointly govern one aspect of gameplay, but a subsystem is also a subgame only if the player experience of using it feels like an activity distinct from ordinary gameplay. Finally, a subgame differs from a scenario (see [Subsection 2.6.3](#)) because there can be many scenarios within a given subgame: in a spaceship-building subgame where players decide how to outfit a ship chassis, new scenarios can be created by varying the type of chassis or the collection of available parts.

certainly will be. In particular, a subgame within a videogame typically has its own idiosyncratic user interface.

In my own experience, most RPGs lack subgames, possibly because, lacking user interfaces like those of videogames, they can't use sensory feedback to help make a subgame feel distinct. Even relatively high-DCG RPGs with complex subsystems don't usually change the core assumption that players control the actions of individual characters — but there are exceptions. In my experience, the most common type of RPG subgame is a mass-combat subsystem, which is used to quickly resolve battles between large armies by suspending or simplifying some of the DCG complexity normally found in combat rules<sup>6</sup>. Fighting at the scale of individual characters has a tactical feel: each individual choice matters, and could mean life or death for a cherished character. Fighting in a mass-combat subgame, on the other hand, will feel “zoomed-out” and strategic, with players directing companies, regiments, or entire armies in maneuvers.

With a working definition of subgames in hand, I can return to P4's points. Their example, the *Elder Scrolls V: Skyrim* ([157], hereafter *Skyrim*) lockpicking subgame, was aptly chosen. A *Skyrim* player who initiates lockpicking is transferred to a UI showing a tumbler lock, as seen in [Figure 6.1](#). The player's challenge is to suss out the angle at which to place a lockpick, and the degree to which to twist the lock's body, with success meaning an open lock and failure meaning a broken pick.

The lockpicking UI sits at a remove from the rest of *Skyrim*. The normal flow of game time pauses while it is displayed; it occludes part of the default first-person

---

<sup>6</sup>Game systems which offer a mass-combat subgame include *Pendragon* and the *Adventurer, Conqueror, King System*, or *ACKS* [97], to name just two examples.





Figure 6.1: The *Skyrim* lockpicking UI. Source: [118].

view of the game world; and the visuals are always the same, no matter whether the lock protects a peasant’s house or an ancient dungeon. These abstractions isolate and distinguish the lockpicking activity from all other gameplay — yet lockpicking is not totally divorced from the rest of *Skyrim*. For one, the lockpicking UI is overlaid on the player’s first-person view, but does not totally hide it; for another, character stats and player skill<sup>7</sup> both influence the chance of successfully picking a particular lock.

One way that P4’s comments contributed to this thesis is by uncovering a shortcoming in [Chapter 2](#)’s theorizing. In my terms, the existence of a “pick lock” action is a taxonomic fact, the action itself is a rule (with further rules possibly influencing it), and the presence of a certain lock in the game world is a substantive fact. However, neither breaking down a game system into rules and facts, nor quantifying the effects

---

<sup>7</sup>That is, skill at manipulating the pick angle and the lock twist.

of rules and facts with DCG, seems to account for the concept of a subgame, leaving no way to formally reason about them, nor a way to quantify their contribution to game feel. Even more importantly, P4's comments open up a discussion about what real GMs and players want from computer-augmented RPGs, which is my next topic for analysis.

### 6.3.2 Task-Specific Informational Interfaces

The game-within-a-game feeling of subgames makes them natural candidates for purpose-built computational interfaces, but a computer-augmented RPG platform like `ROLEPLAYINGGAME` could offer a specialized UI for almost any part of gameplay. Such UIs could present particular subsets of game-state information, and allow GMs or players to perform gameplay operations appropriate to just those pieces of information. My home-campaign experiences have shown me that specialized tools and/or interfaces are critical to GMing or playing in a high-DCG game<sup>8</sup>. Those experiences guided key design decisions for `ROLEPLAYINGGAME`, such as splitting the UI into distinct, task-specific tabs; ensuring the sidebar dropdown menu for choosing a given command parameter would only be populated by entities of the appropriate kinds; and designing the GM's Inspector specifically to surface entity-appropriate commands in a quick-access menu.

I was gratified to hear multiple interviewees not only agree with those design decisions, but also express the opinion that specialized informational interfaces would

---

<sup>8</sup>In addition to the complex item-pricing system mentioned in [Chapter 2](#), tools I've made include a Python app for generating extensive character backgrounds, a spreadsheet for generating believable daily weather patterns at real-world locations, and a calculator for character drunkenness which incorporates character body weight (relative to the average for her race and sex) and the ABV and quantity of the chosen beverages.

be a central benefit of computer-augmented RPGs. P4 asserted that game-content generators<sup>9</sup>, on their own, wouldn't be useful without an interface serving, to the GM, generated content which was directly relevant to the game scenario at hand:

P4: I think the GM is like the bridge between the world — all of the generation — and the players. **It's about enabling the GM.** You could generate all the characters with all details and have it as a giant list. But that's not easy for the GM to use, to suddenly pull someone out of that pack and be this character. So while you could potentially generate that stuff ahead of time, what the GM needs in the moment is what gets pulled up [on screen]. ... So, it's like, **I don't want to see everyone in the town, I want to see who's in this room. What about this character do I need to know? What are his personality traits, or how does he talk?**

— [P4:22], [P4:23]

Similarly, P5 said, “Sometimes you just want to look at information. You don't need a map. You need a list of characters [P5:44].” They felt that ROLEPLAYINGGAME's implementation of this style of interface was a success [P5:13]. They also pointed out that complete isolation of different subsets of information would be undesirable, given participants' need to maintain awareness of game state on multiple levels [P5:41].

P3's gaming group only recently began playing an RPG system with suffi-

---

<sup>9</sup>See [Subsection 6.3.3](#).

ciently high DCG to require computers at the gaming table; despite this, P3 (like P5) felt that ROLEPLAYINGGAME successfully demonstrated the utility of computer augmentation<sup>10</sup>, and offered much commentary on the value of special-purpose computational interfaces ([P5:13], [P3:6].) They expressed their appreciation for the feature of the ROLEPLAYINGGAME Explore tab that shows players valid positions for the `move` action, then explained that existing VTTs require players to manually plan out their movements, just like in analog play ([P3:9], [P3:10], [P3:11].) When I asked what other videogame-like features would be a boon for RPGs, P3 said that it's easy, in traditional RPGs, for players to get bogged down in decision paralysis, whereas videogame interfaces use feedback and visualization to bolster high-level thinking and planning ([P3:13], [P3:14], [P3:15], [P3:16].) They felt that computer-augmented RPGs should adopt those elements of videogame interfaces<sup>11</sup>, and that my home campaign's market-pricing system<sup>12</sup> exemplified the value of such interfaces [P3:30]. Finally, P3 expressed a desire for an interface that would help their group understand the probable results of different dice pools, and saw utility in interfaces which present narrative patterns that match against recorded game events, i.e. story sifting ([P3:25], [P3:27], [P3:39].)

Even P2 saw value and potential in specialized interfaces, despite generally being skeptical<sup>13</sup> of ROLEPLAYINGGAME's suitability for general-purpose augmentation of in-person play. They felt that ROLEPLAYINGGAME's Explore and Gallery

---

<sup>10</sup>That said, they also had reservations about introducing computers to the table because of their potential to distract players [P3:6]. I return to that topic in [Subsection 6.5.2](#).

<sup>11</sup>This line of thinking was echoed by P5 [P5:45].

<sup>12</sup>I ported some of this pricing system to ROLEPLAYINGGAME. See [Subsection 5.5.8](#).

<sup>13</sup>See [Subsection 6.5.1](#).

tabs would be useful for tactical combat and looking up an NPC’s details while role-playing as him, respectively; they also liked the idea of adding a rules editor ([P2:14], [P2:15], [P2:21].)

### 6.3.3 Generating Game Content

Recall that P4 felt it difficult to create even one *Traveller* planet that felt realistic and compelling to their players; what’s worse, *Traveller*’s intergalactic setting implies a need for many such planets. Furthermore, the technology available to science-fiction PCs would probably permit rapid travel between multiple areas of a planet, or multiple planets in a galaxy. These aspects of *Traveller*’s setting massively increase the scale of the content-creation burden a GM would face while running that system.

Procedural-generation tools which respect rules and taxonomic facts could be a fertile area for future work in computer-augmented RPGs. A program that allowed P4 to generate a planet, populate it with inhabitants, and simulate its history — all while respecting *Traveller*’s taxonomic facts outlining, say, planet types, or acceptable stat maximums for GM-designed alien species — would eliminate the content-authorship problem posed by *Traveller*’s setting. As P4 put it when I asked what they would want from computational worldbuilding tools: “Especially for world building, it’s about filling in the gaps of what you don’t want to create ... You don’t want to limit yourself. So those tools [should] come in to fill in all those gaps you don’t have time to do [P4:18].”

Furthermore, interviewees seemed to specifically desire the ability to rapidly add game world content (i.e. substantive facts) to the game world while the game is

being played. P3, P4, and P5 all liked the demonstration of using `define-race` to add a new character race on the fly, i.e. exercise taxonomic malleability ([P3:2], [P4:4], [P5:4].) P3 added that not only did this address a GM’s real need to react to the unexpected, it also improved on shortcomings of existing VTTs [P3:3]. P3 said it “can be really hard” to represent what they want in the game world just by manipulating character tokens and data within a VTT: “more often than not [they] will write down what [they] want it to be on paper notes and just play from that instead ([P3:2], [P3:3], [P3:4].)” For P3, at least, VTT interfaces are too rigid: they lack the affordances a GM needs to rapidly and effortlessly manipulate the game system and world, i.e. exercise malleability.

#### 6.3.4 Testing and Refining Game Designs

When I posed the question of “rules too complex to feasibly employ,” which I discussed at the beginning of this section, P3’s responses shed light on how computer augmentation could also assist game design outside of live gameplay. Some time ago, P3 and a friend began work on *Mecha World*, a *Powered by the Apocalypse* game system inspired by giant robot anime. Although they had some initial successes with designing character-creation rules that drew on tropes of the mecha genre, the two designers found it difficult to develop rules for the gameplay proper, including rules of key subsystems like combat and NPC interaction [P3:18].

To some degree, all GMs must commit themselves to the task of creating rules, testing them through play, and refining them over time. That said, this “playtest-and-see” feedback loop could be improved upon by employing computer simulations

of individual subsystems, which P3 and I discussed, and which P5 has constructed to test his own game designs ([P3:21], [P3:22], [P5:45].) Furthermore, development of fixed-theme games like P3's *Mecha World* might be supported by the following process:

1. Construct a corpus of reviews or discussions of theme-relevant source material, most likely various works of art.
2. Perform topic and clustering analysis to create a dataset extract genre conventions and tropes, plus relations thereof.
3. Offer those results to the designer-GM, either as a data dump for sifting at will, or behind a user interface offering features such as listing all the source materials containing a given trope.

The information and/or tool resulting from such an undertaking would help a designer explore dense quantities of inspirational material at his or leisure. There are even pre-existing corpora which could be easily converted into a dataset to simplify the first two steps; example corpora include the user-created content tags on the fanfiction aggregator Archive of Our Own [39] and the TVTropes collection of categorized artwork tropes [177].

## 6.4 Computer Augmentation as a Learning Aid

One of my research questions asks how computer-augmented RPGs can help players comprehend the rules and facts of a game system while planning their next

moves in the game world. More than one interviewee believed that computer systems like ROLEPLAYINGGAME would be especially useful for new players still learning the ropes of how to play an RPG, let alone how to play skillfully. For instance, P4 felt that ROLEPLAYINGGAME would be useful for helping newcomers adjust to the informational burdens of playing an RPG:

So I could see it being either something ... a little more UI-friendly [so the players could use it directly]. So you can never have played a roleplaying game before, but get in there and be like, “Oh, yeah, this is my character. I know where the menu is [and] I can get to that quicker.”  
— [P4:3]

P5 expressed more fleshed-out ideas along the same lines. They felt that using computer augmentation to flatten the learning curve of a Principle-1-based game would help players break out of the habitual passivity instilled by game systems which don’t expect players to set their own goals. To P5, evaluating and seizing the opportunities presented by a Principle-1 game world are habits that can be trained, and only through training could there come to be a larger mass of RPG enthusiasts to appreciate such games:

A 5e player can do nothing, essentially. Just kind of drift along. If you’re trained to do that, it’s very difficult to break out of it. Even as a DM, every word is planned out for you in advance [P5:11].

... [And] **that’s why I like your system, is that you can**



**see all this data. The data is right there in the app, and I don't have to go to another screen or another app.** If it's all right there, for these complicated inventory things, and I don't have to add it up on paper — I mean, there's a certain kind of player that will do that. I do that on paper when I play. But I've been in math disciplines my whole life. That's normal for me. **If we're gonna reach the normal person, it has to be easy to do this stuff, so they can free up that brain space to go and make these decisions that they haven't ever had to make before in a roleplaying game** [P5:13].

... [T]he barrier to entry does weed out a lot of people who are gonna be too lazy to play a game like this, but I think a lot of those people can be trained. I think they're lazy because it's easy to be lazy. It's so much easier to be lazy that there's no benefit to making the effort to go up. And so if we can shave off some of the rough edges ... we'll have more of a chance with the person who can be convinced ([P5:15], [P5:17].)

In addition to the possibility of using computer augmentation to assist new players, several interviewees were interested in my proposal that a computer-augmented RPG could enable more seasoned players to review past gameplay, the same way athletes will review footage of past games ([P2:6], [P3:25], [P5:32].) If ROLEPLAYINGGAME were upgraded to record all in-game events<sup>14</sup>, users would be able to play them back

---

<sup>14</sup>At present, ROLEPLAYINGGAME records all past game states in a database, but it doesn't record the user commands and resulting events that actually *caused* those state changes.

while visualizing the accompanying changes to game state. This would permit players to study battle tactics, practice using complex character abilities, or relive past moments of glory in perfect fidelity. Event-level data recording would also help a developer working on ROLEPLAYINGGAME (or a ROLEPLAYINGGAME-supported game system) to debug his or her code.

## 6.5 Criticism and Concerns

### 6.5.1 Feasibility and Costs of Using Computer Tools

One argument in favor of computer-augmented RPGs is the possibility of automating away manual operations that don't directly deliver a benefit to users. For instance, the driving force behind the design and implementation of the social inventory tab ([Subsection 5.5.7](#)) was my desire to relieve players from having to carry out the manual task of searching through and comparing inventory spreadsheets. However, sometimes it's difficult to avoid creating a situations where the formal interfaces of a computer system impose a higher cost to perform some task than was previously the case in analog play.

For instance, in my home campaign, players use Google Sheets for recordkeeping<sup>15</sup>. Sometimes they have to paste semicolon-separated strings from the market table into their character sheets, then use a dropdown menu to split them into the required column format. Each such copy-and-paste operation consumes a little of the time and

---

<sup>15</sup>My home campaign does not yet make use of ROLEPLAYINGGAME, which is still very much a prototype.

mental energy which is so precious for playing high-DCG games. P4 was right when they said that even a savings of a few minutes over the course of a game session can be meaningful for adult players with real-world responsibilities [P4:10]. And, as P5 said, knowing how to split strings in Google Sheets isn't a core skill for engaging with the RPG itself; rather, it's an incidental chore imposed by my game making use of Google Sheets [P5:38].

Section 6.4 explains why the study interviewees believed that ROLEPLAYINGGAME, or something like it, could enable more widespread adoption and enjoyment of high-DCG, player-driven games. On the other hand, the interviewees also leveled criticism against several elements of ROLEPLAYINGGAME's user interfaces, revealing a need to redesign those for improved end-user accessibility.

P2 had the most critical reaction to ROLEPLAYINGGAME. They acknowledged that the tactical battlemat interface available to the players and the GM<sup>16</sup> would be valuable for precise combat movement [P2:12]. However, they weren't convinced that players or GMs would be able to successfully keep ROLEPLAYINGGAME's internal state abreast of game world changes resulting from the back-and-forth conversation that drives gameplay<sup>17</sup>:

MJ: Could you see you and your group using a virtual tabletop, or my thing — or even just Photoshop on someone's computer with all of you looking at it — to carry out more detailed game rule operations, and then

---

<sup>16</sup>On the explore and game world tabs, respectively. See Subsection 5.5.6 and Subsection 5.5.4.

<sup>17</sup>See Section 2.1, especially the conversational game loop described at the end.

[doing] the rest of the gaming [without that computational aid]?

P2: Not really, honestly, because that seems like too much overhead. I think like I said, [for] the more rules-based, tactical parts of the game, your tool makes perfect sense and seems like it's a huge help there. But then everything you do the rest of the time — having to make sure that the computer's state is consistent with all of the stuff you guys did just by talking ... That seems like it might be real overhead, from a user perspective: the player's interaction.

— [P2:12], [P2:13]

That said, despite being unsure of the value `ROLEPLAYINGGAME` could provide for playing with low-DCG game systems, P2 was interested in the possibility of modifications to `ROLEPLAYINGGAME` that would allow a gaming group to pick and choose what features they wanted to use ([P2:19]), e.g. opting into precise movement tracking, and opting out of computational support for inventory management. Together, P2 and I concluded that *a la carte* feature activation would expand the potential audience for `ROLEPLAYINGGAME`, by making its functionality useful even for players of relatively low-DCG games [P2:20].

As a final note about P2's comments, recall that my error in recruitment meant that P2 had much more limited RPG experience than the other interviewees (see Table 6.1.) They had only ever played in one campaign, and weren't sure what edition of *D&D* was being used in it [P2:5]. It would be unfair for me to expect a casual player

with no GMing experience not to struggle to connect with a tool aimed at supporting high-DCG games, and which I probably presented with an emphasis on how it enables a GM to do the runtime game design required for malleability. I absolutely am not belittling P2 here. I value their opinions, and while they might not be the audience for my kind of game, other aspects of computer augmentation seemed to appeal to them. That puts us on the same “team.”

The other interviewee who gave substantial interface feedback was P1. Overall, though they were more enthused than P2 overall, they were also open with their concerns about UI shortcomings, which made sense given their background in designing interactive tools for data visualization and exploration. In particular, P1 identified a flaw with the current implementation of differential display, which is that it doesn't inform a player why a given piece of text is being displayed [P1:1]. They also said that it makes sense for ROLEPLAYINGGAME to eventually offer malleability even for its UI design, as a spreadsheet allows users to reinterpret what cells mean, “but it can't actually promise that at the same level of fidelity for any arbitrary field you you might add [P1:5]”.

P1 was also quick to identify troubles that might arise during future development of ROLEPLAYINGGAME because it occupies a strange middle ground between an administrative interface for complex tasks and a transaction processor for those tasks — and that's without getting into malleability.

P1: Yeah. The tension I'm seeing here is you are saying to the GM, "Look, you can operationalize everything you're doing. Here is a beautiful language." But code is scary. So you are going to have to build this involved visual editor to round trip you from your [RPG design done on] paper, to the database, through this ontology of transactions that you've beautifully laid out here. And it's very difficult to make progress on, just because you have to spend all this time iterating on the representation, and the ontology suffers. You end up with a toy ontology because the representation is very hard to create.

— [P1:10]

### 6.5.2 Potential for Distraction

Recall that P3 has newly introduced computers at their gaming table. They raised concerns that this could cause interruptions to gameplay if players got distracted by non-game uses of the computer [P3:6]. On the other hand, P3 also suggested that using an iPad or other tablet-form-factor computer, given its focus on performing one task at a time, could help limit distractions without unduly impacting the usability of a computer interface [P3:7].

## 6.6 Summary

Overall, the community feedback I received for ROLEPLAYINGGAME was primarily positive, with individual reactions ranging from partial acceptance to total excitement. The interviewees not only agreed that I had identified a real gap in the design space of RPG-support software systems, but also felt that ROLEPLAYINGGAME was a real first step towards filling that gap. In the course of our conversations, I learned that I'm not the only one disappointed with VTT affordances, nor the only one thinking about how computers could support complex, crunchy games by learning from videogame development and human-computer interaction. At minimum, I now know for sure that there's an audience for the new class of software that I believe ROLEPLAYINGGAME has founded — and I'm more confident that, though ROLEPLAYINGGAME is the first entry in this category, it won't be the last.

P3 and P4 were strongly encouraging about the direction ROLEPLAYINGGAME has taken, with the former concluding their interview by saying, “I think this is a really cool project, and I think you're asking the right questions [P3:40].” But of all interviewees, P5 exhibited the most unbridled enthusiasm for ROLEPLAYINGGAME. After the Demo video concluded, their first words were, “I'm not going to lie. I would play this right now [P5:1].” As we wrapped up, they not only asked to read this thesis once it was finished, but also offered to spread the word about ROLEPLAYINGGAME when it was more ready for open sharing [P5:50]. Not everybody will agree, like P5, with my motivations for wanting to explore this design space ([P5:24], [P5:25], [P5:26], [P5:27]), nor

with my characterization of the RPG ecosystem as lacking a “design culture [P5:28].” But it’s reassuring to know that there are others out there who are willing to embrace the computer as the perfect tool for reshaping this subculture’s decades-old assumptions about what RPGs should contain, how they should played, and how they should be run.



## Chapter 7

### Discussion and Conclusion

This chapter begins with a reflection on the research questions posed in [Section 2.9](#). I consider how the ROLEPLAYINGGAME prototype, its development process, and extensive feedback from members of the RPG community have all contributed to answering those questions. After that, I propose directions for future research which could take inspiration from this thesis, such as ROLEPLAYINGGAME enhancements based on insights gleaned from the user study, and ways to connect my idiosyncratic concepts of rules-and-facts analysis, DCG, and malleability with other areas of scholarly research and popular discourse. Finally, I conclude by summarizing what this thesis contributes to the academic literature.

## 7.1 Reflection and Analysis

### 7.1.1 Research Question 1

My first research question was concerned with the feasibility of designing and implementing a computational RPG platform which could support malleability without restricting game system DCG. With the `define-race`, `define-rule`, and `toggle-rule` commands, `ROLEPLAYINGGAME` offers malleability of rules and taxonomic facts for a computer-augmented RPG. These are firsts in the field of RPG support.

Furthermore, although the toy game system which I wrote for `ROLEPLAYINGGAME` doesn't approach the DCG of videogames or typical traditional RPGs, it contains sufficiently many elements of typical RPGs (e.g. a domain model containing races, weapons, equipment slots, skills, and character actions) that there appear to be no obstacles preventing `ROLEPLAYINGGAME` from being upgraded to support the further features<sup>1</sup> needed to implement a game system at least as high-DCG as real, published RPGs.

The `ROLEPLAYINGGAME` implementation also demonstrates the feasibility of programming a malleability-supporting system using non-exotic technology. While I did choose to implement `ROLEPLAYINGGAME` in a Lisp-family language, I didn't directly use any unique Lisp features such as general-purpose metaprogramming or treat-

---

<sup>1</sup>Such features include rules with outcomes other than Boolean values, and tracking the passing of in-game time as character actions take place (or when the GM wills it.) For more on this subject, see [Section 7.3](#).

ing code as data. That said, such language features might be critical for expanding malleability deeper into every corner of ROLEPLAYINGGAME itself: in [Subsection 7.3.2](#), I discuss how increasing the malleability of both a supported game system and ROLEPLAYINGGAME itself might culminate in exposing full-blown reprogrammability to end users.

Of the five RPG enthusiasts who I interviewed for the user study, four had both a long history of involvement with the art form and plentiful GMing experience<sup>2</sup>. All four of these seasoned GMs agreed that malleability was crucial to RPG play, but existing RPG support software lacked meaningful and effective affordances for exercising it — to the point that one interviewee considered it easier to fall back on pen and paper than fight the interface of a VTT<sup>3</sup>. As such, interviewees reacted positively to ROLEPLAYINGGAME’s malleability support, and my contention that it should be the chief design concern for creators of future RPG support software.

### 7.1.2 Research Question 2

My second research question was concerned with the affordances that a new system in this area could present to its users. ROLEPLAYINGGAME demonstrates the value of offering an interface which (for a particular task, subgame, or subsystem) groups together a display of relevant game state information, and affordances for users to operate on that state. It also demonstrates the value of having these interfaces co-located in one application with one game-wide data store. Finally, it shows the benefits of using

---

<sup>2</sup>See [Table 6.1](#).

<sup>3</sup>See P3’s comments in [Subsection 6.3.3](#).

videogame-style visual design to compartmentalize secret information, help users comprehend the current game state, and ease the cognitive load of decisionmaking. Multiple user study participants said that a critical factor that made ROLEPLAYINGGAME appealing was its stance toward incorporating the whole of both game system and game state into one application, as videogames do.

Feedback from the user-study interviewees indicates that GMs and players find these offerings highly desirable, even for use with games that are relatively low-DCG. Differential display, in particular, elicited strong positive reactions from all interviewees (Section 6.3.) The interviewees also readily offered suggestions (Subsection 6.3.2) for additional computer interfaces that would provide just-in-time assistance with comparing dice-roll probabilities, refreshing memories of NPCs and other scenario-relevant details, charting patterns of potential character actions that could further a narrative, and designing and evaluating in-game objects with significant opportunities for customization (such as spaceships.) Finally, there was a great deal of discussion (Section 6.4) around using computer augmentation to help new players get their start in RPGs, or improve existing players' skill and confidence for player-driven, Principle-1-based gameplay.

The user-study interviews make it clear that applications of computing to RPG design and play have placed insufficient emphasis on counteracting the information overload that afflicts players and especially GMs. What's more, the cognitive challenges of GMing for relatively high-DCG RPG systems, and adding personal designs atop them, were cited by interviewees as the reason they had not made meaningful use of game systems which they otherwise found compelling. I hope that systems like

ROLEPLAYINGGAME can change this situation for the better.

### 7.1.3 Research Question 3

My third research question was concerned with degrees of DCG that could be achieved only by an RPG system which, though run and designed by a human, was fully implemented in a computer. I don't believe ROLEPLAYINGGAME itself was responsive to this question, but other systems I've already created<sup>4</sup> have shown that using RPG subsystems which require a computer to operate can deepen the complexity of challenges facing the players, and, in turn, the potential satisfaction which players will feel after besting those challenges. ROLEPLAYINGGAME, being extensible and written in a general-purpose language, is not only compatible with my earlier computer-driven subsystems, but could (in the future) combine all of their advantages under the improved UX of a single monolithic application.

As I mentioned while reflecting on the second research question, it turns out that even RPGs which were designed for analog play can still be too high-DCG for experienced GMs to feel comfortable running them. It seems that, in addition to asking whether apps like ROLEPLAYINGGAME can let designers achieve greater heights of DCG, researchers and software developers should also consider how such apps can prevent an otherwise-promising RPG system from being something that gamers merely aspire to playing, rather than an approachable, functional game system.

---

<sup>4</sup>See [Subsection 2.7.2](#).

## 7.2 Future Theoretical Work

Because I wanted to argue as thoroughly as possible that `ROLEPLAYINGGAME`'s features were novel, this thesis's literature review was focused on software-based contributions to RPG support and related fields. I needed a long time to gather and assess that body of work, leaving me less time than I wanted to explore other areas relevant to designing software for playing and designing high-DCG games, and to theorizing about RPG design in general<sup>5</sup> This section proposes a few ways in which the DCG framework and computer augmentation could be connected out to other fields of study.

### 7.2.1 Connections to Fields Beyond Game Design

Dozens of areas of academic research could be brought to bear on developing empirically-motivated theories of how to GM and design RPGs that will induce certain emotions in the players. Academics in literature, theater, and game design have transcribed and studied RPG play sessions, which could be mined for clues as to how digital tools might preempt conflict and miscommunication. Literary theorists have put forward proposals for viewing RPG play as a form of meaning-making (e.g. [142]), which could suggest how game-rule descriptions might best be written to emphasize the Principle 1 notion that the players control their own destinies. Psychological research on how people form plans, act under stress, and respond to failure could be leveraged to propose ways for the GM to fruitfully manage players at the table.

---

<sup>5</sup>For instance, there is surely prior art which somewhat prefigures my proposed concepts of DCG and malleability — even though not much has turned up so far — and I regret not being able to properly couch my formulation of Principle 1, and arguments for it, in the voluminous literature on player agency in games.

For instance, here's how empirical results from the psychology of creativity could inform RPG design. Psychologists investigating whether the time-worn saying "constraints breed creativity" has any basis in reality (e.g. [2], [156].) have attempted to ascertain how many constraints can be imposed before they stop fostering creativity and start stifling it. For the sake of argument, suppose the findings show that having zero constraints means you get no work gets done because of the "fear of a blank page," and having too many constraints means you get no work done because you can't figure out how to satisfy all of them at once. Further suppose that, at a sweet spot somewhere between those two extremes, the constraints are sufficiently numerous to guide your work, but not so numerous as to otherwise inhibit your creativity. If that were true, then that sweet-spot number could indicate a desirable value for the average or maximum connectedness<sup>6</sup> of a typical rule.

To see why that's useful, consider the case of a highly-connected rule which has 300 inputs. Not only will the GM be hard-pressed to apply that rule accurately (which defeats the first half of Principle 2), but also, the players will probably not feel like they have agency in scenarios governed by that rule, because it would take an inordinate or impossible amount of time to maneuver their characters so as to predictably control a significant fraction of those inputs' values. On the other hand, if that rule is revised to take only a dozen inputs, the players will feel rewarded if they expend the time and effort to control six or 10 of them, then participate in a game scenario where their preparation causes that rule to frequently deliver favorable outcomes.. There's no better

---

<sup>6</sup>See [Section 2.6](#).

sign of players embracing Principles 1 and 2 than seeing them deliberately take actions in the game world to get some rule to function the way they want, with the larger goal of setting up a scenario where they can achieve success with the help of that rule. Those hypothetical players have taken their fate in their hands, and have understood the rewards of playing in a game strictly governed by rules.

### **7.2.2 Diagnosing and Repairing the Causes of Unexpected System Outcomes**

Personal correspondence with Orion J. Anderson, a graduate student in psychology at the University of Virginia, led him to offer me a tentative taxonomy of “anomalies,” or moments in RPG play where interactions between game system components create an emergent outcome<sup>7</sup> which takes the GM by surprise. Accompanying this taxonomy was a lucid comparison of how those anomalies might be diagnosed and repaired by the GM of a traditional, analog game and the GM of a ROLEPLAYINGGAME-esque, computer-encased one. With Anderson’s permission, I paraphrase his taxonomy and proposal below.

1. Serendipity: The relevant rules, whether written or programmed, are working as intended, and no data input errors have occurred. The surprising outcome was always implicit in that combination of rules, but that outcome is desirable and the GM need make no changes.
2. Revelation: As serendipitous, but the outcome is undesirable to the GM. The

---

<sup>7</sup>This specifically means the result of executing a rule, as defined in [Section 2.3](#).



rules which led to the outcome must be adjusted so that they have the desired effect in the future.

3. Bug: One or more rules are incorrectly implemented, such that their operation deviates from the designer's intentions. The surprising outcome reveals this flaw, and spurs the GM to correct the implementation.
4. Mistake: The surprising outcome came from user error while recording data or operating a computer system, such as erroneously transcribing a monster's hit points, or fat-fingering the wrong answer to a software prompt.

Anderson suspects that a traditional RPG and a computer-based, malleable RPG are likely to differ with respect to how often they generate anomalies, the distribution of anomalies they generate, the GM's ability to diagnose anomalies, and the GM's ability to either cope with or repair the anomaly.

His hypothesis is that traditional RPGs are typically not sophisticated (i.e. high enough in DCG) enough to generate much serendipity or revelation, but in exchange, bugs are rare. A computer-based RPG offers vastly increased sophistication, with concomitantly higher chances of serendipity, revelation, and bugs. He thinks that mistakes will be more common in the traditional paradigm, because humans are less effective at dealing with DCG than computers are; furthermore, given fallible human memory and low-tech recording instruments, it may be difficult to remember what the correct pre-mistake value should have been or when the mistake was made. On the computational side, without innovations in HCI specific to RPG support, even comparatively

few mistakes could have more severe consequences: automatic processes will happily churn away on legal-but-incorrect values, propagating bad information to far corners of the system and making it hard to recover from the mistake. On the other hand, logging and other instrumentation may make it easier to narrow down the point at which the mistake happened, and various strategies are available for rolling back data to a known-good state.

I think an empirical investigation into the correctness of Anderson’s taxonomy, and his conjectures about relative frequency severity and recoverability of different anomalies, would be an excellent research program for a games researcher. To compare traditional and computational methods of gaming, perhaps an RPG could be implemented both as a set of written rules and a malleable computer program. Then, each of a group of GMs (who are the study’s subjects) could run a preset scenario, with the players being research collaborators who do their best to take the same in-game actions each time.

### **7.3 Future Technical Work**

In this section, I present several meaningful goals for additional development work on ROLEPLAYINGGAME’s capabilities. Where possible, these goals are grounded in suggestions or realizations from the user study.

### 7.3.1 Formal Content Schemas

Following established Clojure practices, each instance of a game-world kind is represented by a key-value map defining its attributes. One could create schemas for these maps which delineate required keys, optional keys, and the potential datatypes or values permissible at each key. For instance, a partial schema for a game character which constrains the `:strength` attribute might be expressed in English as “Characters have a required key called `:strength`. The value at `:strength` must be an integer greater than or equal to 0.” Such schemas would be (collections of) taxonomic facts, so they ought to be made malleable by including them in the game state, as with the taxonomic facts that constitute the hierarchy of kinds.

Implementation-wise, “working with schemas for program data” is exactly the use case for Clojure’s `spec` library [65], which can not only define such schemas, but also validate a key-value map against a schema, and even generate maps which are guaranteed to be schema-compliant. With `spec`-style schemas defined, `ROLEPLAYINGGAME` could offer procedural content generation (PCG) for anything in the kind hierarchy, including characters, items, and locations; this would permit the implementation of high-level GM commands that both create and insert new content. Even more promising would be subsequent investigation into mixed-initiative user interfaces for `ROLEPLAYINGGAME`, with the computer system generating candidate content, and the GM user selecting or rejecting candidates to determine what actually enters the game world.

Finally, it might also be feasible to define schemas for implementation concepts such as commands, actions, events, and rules, which would allow PCG to add to both the game system and ROLEPLAYINGGAME itself. That steps somewhat outside traditional PCG, falling more closely under the header of automated game design (AGD), an active research area studying computer systems that can generate entire games (or subsystems thereof.) To my mind, the most successful and intriguing AGD project is Ludi<sup>8</sup> ([19], [18]) . The Ludi system applies numerous human-defined evaluation criteria for game rules (operationalized as programmed functions) to evolve an initial rule set into one that better satisfies the criteria. Perhaps it would be possible to operationalize DCG in a manner amenable to Ludi-style evolutionary techniques; if so, ROLEPLAYINGGAME could then assist GMs with the act of game design itself.

### 7.3.2 More Malleability, Culminating in Reprogrammability

The GM command `define-race` suggests a family of other commands which add to or alter the taxonomic facts of the game system, such as `define-weapon` or `define-tradegood` . In general, any taxonomic fact which enumerates a set of allowable values for an attribute, such as a weapon list or a catalog of magical spells, could have a `define-X` command implemented for it. It's also easy to imagine a `define-attribute` command which stipulates an entirely new attribute, rather than extending the range of an existing one. For instance, the GM might give all current and future characters a `:melee-attack-range` attribute, complete with some calculation

---

<sup>8</sup>Not to be confused with its successor, the similarly-named Ludii [123]

for finding a default value for a given character. This `define-attribute` command would allow the GM to act like a database administrator who migrates to a new schema, giving a preexisting table a new column<sup>9</sup>.

Neither events nor commands have their definitions included in game state. If that were fixed, one could implement `define-event` and `define-command` commands to alter or extend those vocabularies as well. The latter, in particular, would greatly extend a GM's control over ROLEPLAYINGGAME by enabling him to reify and easily trigger event sequences that change game state in a certain way (akin to the macro system seen in certain VTTs, or even some MMOs.)

The above may sound like adding full-fledged programming capabilities to ROLEPLAYINGGAME's UI. Indeed, I think that making ROLEPLAYINGGAME into a tool supporting on-the-fly malleable game design must eventually cash out in LambdaMOO's master stroke, i.e. the ability for a GM to (almost) completely reprogram ROLEPLAYINGGAME from inside itself. Anything less would mean some sacrifice of analog RPGs' malleability.

### 7.3.3 Widespread Use of Differential Display

Differential display's potential for fostering immersion extends far beyond controlling the presence or absence of descriptive text. More radical alterations of players' user interfaces would enshrine in software the requirement for players to work together

---

<sup>9</sup>One thing Foundry gets right is support for exactly this kind of "migration." A Foundry game-system designer, having changed how a game system works and bumped it from Version X to Version X+1, can also include code that defines how to change entities to suit. While this is a wise design decision, it's not sufficient to establish taxonomic malleability: a game world using Version X cannot be kept running during the migration to Version X+1.

if they want the best chance of succeeding at their goal. In addition, games such as *Artemis: Spaceship Bridge Simulator* (in which each player performs a different role to collectively operate a *Star Trek*-esque spaceship) [134] have shown that computer-mediated doling out of drastically different information to each player can produce a thrilling and satisfying cooperative play experience. Here are some use cases for differential display which go beyond descriptive text:

1. A character with a skill that grants knowledge of smuggling can see a bonus “Black Market” merchant available on the market tab, where she can purchase items of an unsavory nature that aren’t available elsewhere.
2. A character with knowledge of shipping and trade looks at the market tab, and not only sees item prices for her current location, but also sees a readout of what those prices would be at nearby markets.
3. A character with an eye for hidden passages notices a secret door connecting two zones.
4. A character with sufficiently low intelligence is unable to accurately track the amount of money she carries. Instead of getting a numeric count of the coins she possesses, her character sheet simply says “a few” or “lots.” When she buys items, she will overspend by a random amount. Barring a permanent change in her intelligence score, the only way to counter this is for a smarter character to either accompany her to market, or use the “count money” action on the unintelligent character’s coins, which will temporarily allow the unintelligent PC to keep track.

Using differential display in this manner raises the question of whether the software should permit a player to share differential-display information that she alone possesses, or whether the players should have to pass that information between themselves. I think this would have to be handled on a case-by-case basis, depending on the type of information being revealed through this mechanism. The aforementioned character who sees the concealed pathway should be able to point it out to a second character, at which point either of them might use the route. On the other hand, if the differential information is the position of an invisible object, and only one character is under a spell allowing her to see invisible things, then at best she should only be able to mark, on other players' UIs, the current position of the item. That information should not get updated even if the object moves around, forcing the character with the spell to spend time directing the other characters if she wants to give them the benefit of her enhanced vision.

#### 7.3.4 Rule Design Languages and Story Sifting

If ROLEPLAYINGGAME is to become accessible to GMs who are not programmers, it must at some point incorporate a domain-specific language (DSL) for rule design, one which requires less skill to use than a full programming language. Statements in such a language could then be compiled into Clojure code equivalent to what the GM must currently write for himself while using the `define-rule` command. One good starting point for best practices in rule design might be game description languages (GDLs). A GDL is a formal language which describes a class of games; I believe

that files written in some GDLs can be directly executed by an appropriate interpreter. These come from the field of automated game design (AGD), which, as an evolution of PCG, seeks to generate entire games from scratch.

Another worthwhile starting point comes from the field of interactive digital narrative (IDN.) **Story sifting** is the emerging practice of using logic programming to extract partial or complete narrative patterns from a record of events created by a game or simulation<sup>10</sup>. All story sifting systems possess some method for specifying (or generating) patterns. Roughly speaking, a pattern consists of a set of logic variables, a set of relations and predicates which the variables must singly or jointly satisfy, and an outcome that will be reported by the sifter — if, within its event database, it can find a set of events which, when bound to the logic variables, satisfies the relations and predicates.

One way to conceive of a rule input is as a filter that selects the correct piece of game state to feed into the rule. For instance, if a rule for lifting a heavy object takes as inputs a particular character, that character’s strength score, an object, and the weight of that object, then it’s possible to execute that rule for any pair of entities where one is a character, one is an object, the character has a strength score, and the object has a weight property. If we call “ability to execute the rule” an outcome, then this formulation of a rule’s inputs is strikingly similar to the previously given formulation of a story-sifting pattern.

---

<sup>10</sup>The term originates with James Ryan [139]. Max Kreminski and their collaborators have done pioneering work in this area ([83], [82], [81], [86], [84]). Other work worth reviewing includes [72], [141], and [25].



Therefore, it might be possible to equip `ROLEPLAYINGGAME` with a rule design language and story-sifting capabilities at the same time, with one code module. While it might be more prudent to try working on them separately, then fuse the two modules if they prove successful and useful, I'm supremely curious about whether a computer-augmented RPG could gain both of these features at one fell swoop.

Regardless of whether a rule DSL and sifting patterns can profitably be combined, I wish I had had the foresight to design `ROLEPLAYINGGAME` to be compatible with sifting from the get-go. When combined with a computer-augmented RPG that stores the history of its game events and world states — especially if there is also some simulation of NPCs' daily routines, weather patterns, or other autonomous processes that regularly change substantive facts — sifting has the potential to enable entirely new kinds of GM assistance. One example is displaying suggestions for how a GM should portray an NPC, based on sifter patterns for “revenge,” “gratitude,” or “double-crossing” which match up with actions that the PCs have taken toward this NPC in the past. Another is visualizing relationships between the major NPCs (based on a simulated history) that dwell in a location the players have just entered. Several of the story sifting works cited here are intended to support use cases like these.

## 7.4 Conclusion

In this thesis, I have made the following contributions:

### **7.4.1 A Defense of a Philosophy**

In [Chapter 2](#), I presented my concept of “zero-ego GMing,” a perspective which seems inadequately explored by either academic or commercial discourse. I distilled zero-ego GMing into Principles 1 and 2, then argued that a game run by those principles is worth pursuing because it cultivates, in its players, such virtuous behaviors as creativity, cleverness, cooperation, and self-sacrifice. I explained how this philosophy required me to pursue a certain style of system design, which in turn led to augmenting my game with custom software. My goal for supplying this background information was not to showcase the benefits I’ve reaped from my software, but rather to demonstrate why existing RPG support software is inadequate for my needs.

### **7.4.2 A Classification of Components**

In [Chapter 2](#), I classified RPG components into rules, taxonomic facts, and substantive facts. I asserted that elements of the game world (substantive facts) ought to be on equal footing with the mechanisms governing their structure and evolution (taxonomic facts and rules) because substantive facts also contribute to setting players’ expectations for gameplay. In effect, this classification erases the traditional distinction between “game system” and “game world.”

### **7.4.3 A Method for Quantitative Analysis**

In [Chapter 2](#), I built on the aforementioned classification to propose a method for quantifying the complexity of an RPG system. This method stipulated three funda-

mental system characteristics: depth, (average) connectedness, and (average) granularity, collectively called DCG. I described how the DCG characteristics combine with each other to produce tension, difficulty, and potential for emergence — qualities of gameplay which I claimed were desirable because they increased the range of scenarios a system could represent, and thus the range of subtly-different, scenario-specific emotions that players could find themselves experiencing. I concluded that players' reactions to those scenarios and emotions would lead them to develop the virtuous behaviors mentioned in [Subsection 7.4.1](#).

#### **7.4.4 An Account of Malleability**

In [Chapter 2](#), I gave what I believe to be the first academic definition of malleability, a characteristic which is almost unique to traditional RPGs. I identified malleability as the crucial feature of RPGs which frees them from having any limits on the scope of their gameplay, and I connected this benefit of malleability with my own belief in Principle 1. I also showed that GMs must constantly exercise malleability, since they can't avoid creating or alter substantive facts during a game session even if rules or taxonomic facts don't change.

In [Chapter 3](#), I showed that work in and around the field of RPG support has rarely taken malleability of all three parts of my system taxonomy as be a design concern. Work in this area which has allowed malleability typically only did so for substantive facts. Furthermore, even when malleability for rules and taxonomic facts was an option, it was reduced to a design task that can't take place during live gameplay,

and didn't offer tooling beyond standard code and 3D-asset editors to assist GMs with the art of game design.

#### 7.4.5 A Set of Design Criteria

In [Chapter 4](#), I laid out five design criteria that collectively characterize a sector of the RPG support design space which [Chapter 3](#) showed had had little prior exploration. To recap, these criteria select for computer-augmented RPGs that do the following:

1. Minimize or eliminate any hardware other than computers themselves.
2. Enable malleability of all rules and facts.
3. Impose minimal or no limitations on a game system's domain model.
4. Enhance player and GM comprehension of rules and facts.
5. Prioritize co-located, in-person play.

#### 7.4.6 An Innovative Artifact

In [Chapter 5](#), I described `ROLEPLAYINGGAME`, the first representative of a new class of software within the field of RPG support. `ROLEPLAYINGGAME` at least partially fulfills all five design criteria from [Chapter 4](#), and it possesses capabilities which are novel for this field, including:

1. Malleability of rules.

2. Malleability of taxonomic facts.
3. Videogame-style interactive visuals.
4. Task-oriented interfaces which group game data together with relevant commands.
5. Cross-task interfaces (e.g. the Inspector) that permit changing one's desired command without losing current context.
6. A general UI design which doesn't privilege the battlemat over other methods of rendering game state.
7. Integration of differential display of information into the user interface.

#### 7.4.7 A Shared Vision

In [Chapter 6](#), I described and analyzed the contents of interviews I conducted with five RPG community members. I gauged their receptiveness to ROLEPLAYINGGAME specifically, and both high-DCG gaming and computer augmentation in general.

These interviewees sensed that much of the potential of this art form had yet to be explored. Those who yearned for more complex RPGs felt unable to overcome the accompanying design and GMing challenges. Even those who preferred fixed-theme or storygame systems readily identified aspects of play that would benefit from technological intervention. All the interviewees expressed coherent, realistic proposals for valuable gameplay enhancements which computer augmentation could provide.

Happily, reactions to ROLEPLAYINGGAME were generally positive. The interviewee with the least RPG experience was skeptical about the feasibility of using a computer system during live play, but still recognized its value for certain use cases, and proposed a couple more. Meanwhile, the other four interviewees, all having much RPG knowledge, were enthusiastic about the novel features of ROLEPLAYINGGAME, and felt that it represented a step into the future of how we learn, play, GM, and design RPGs.

The computing systems which will bring us further into that future are not beyond the ability of programmers to provide: proofs-by-existence can be found in academia and especially in the world of videogames. Programming is not the only kind of work needed to bring about the computational future for which I advocate, but it is the most important kind. That said, if the traditional, malleable RPG is going to survive its third attempt at being computerized (without collapsing into the CRPG or the VTT), then ease of development must be subordinate to correctly and fully embodying the RPG's nature. Even after ROLEPLAYINGGAME is nothing but a relic, I hope my definitions of rules, facts, DCG, and above all malleability will continue to inspire new work which shares that goal.

# Bibliography

- [1] 1000Nettles. Introducing “hey wait!”: A module to give gms a bit of breathing room. [https://www.reddit.com/r/FoundryVTT/comments/kizbt1/introducing\\_hey\\_wait\\_a\\_module\\_to\\_give\\_gms\\_a\\_bit](https://www.reddit.com/r/FoundryVTT/comments/kizbt1/introducing_hey_wait_a_module_to_give_gms_a_bit).
- [2] Oguz A Acar, Murat Tarakci, and Daan Van Knippenberg. Creativity and innovation under constraints: A cross-disciplinary integrative review. *Journal of Management*, 45(1):96–121, 2019.
- [3] Devi Acharya. Computational support for game masters of tabletop roleplaying games. Master’s thesis, University of California, Santa Cruz, 2021.
- [4] Devi Acharya, Jack Kelly, William Tate, Maxwell Joslyn, Michael Mateas, and Noah Wardrip-Fruin. Shoelace: A storytelling assistant for GUMSHOE One-2-One. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, page 1–9, Lisbon, Portugal, Apr 2023. ACM.
- [5] Shannon Applecline. A brief history of game #1: Wizards of the coast: 1990-present, Aug 2006. <http://www.rpg.net/columns/briefhistory/briefhistory1.phtml>.

- [6] Various Authors. Vassal 3.6.19, 2023. <https://vassalengine.org>.
- [7] D. Vincent Baker. *Dogs in the Vineyard*. Lumpley Games, 2004.
- [8] D. Vincent Baker and Meguey Baker. *Apocalypse World*. Lumpley Games, 2010.
- [9] Saskia Bakker, Debby Vorstenbosch, Elise van den Hoven, Gerard Hollemans, and Tom Bergman. Weathergods: Tangible interaction in a digital tabletop game. In *Proceedings of the 1st international conference on Tangible and embedded interaction - TEI '07*, page 151, Baton Rouge, Louisiana, 2007. ACM Press.
- [10] Lionel Barret, Claudia Vance, and G. Michael Youngblood. Lessons in user interface design in the procedural city generation for games tool *Ürban pad*. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, Bordeaux, France, Jun 2011. ACM.
- [11] Richard A. Bartle. *How to Be a God: A Guide for Would-Be Deities*. NotByUs, West Bergholt, Essex, 2022.
- [12] Matt Barton and Shane Stacks. *Dungeons and Desktops: The History of Computer Role-Playing Games*. A K Peters/CRC Press, [2nd edition]. — Boca Raton: Taylor & Francis, [2018], 2 edition, Apr 2019.
- [13] Karl Bergström and Staffan Björk. The case for computer-augmented games. *Transactions of the Digital Games Research Association*, 2014.
- [14] Karl Bergström, Staffan Jonsson, and Staffan Björk. *Undercurrents: A computer-*



- based gameplay tool to support tabletop roleplaying. In *Nordic DiGRA 2010*, 2010.
- [15] BioWare. *Neverwinter nights*, 2002.
- [16] Chester Bolingbroke. Game 123: Orthanc (1975), Nov 2013. <https://crpgaddict.blogspot.com/2013/11/game-123-orthanc-1977.html>.
- [17] Adam “Badeye” Bradford. Curse media and fandom are joining forces!, Dec 2018. <https://www.dndbeyond.com/forums/d-d-beyond-general/news-announcements/29976-curse-media-and-fandom-are-joining-forces>.
- [18] Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.
- [19] Cameron Bolitho Browne. *Automatic generation and evaluation of recombination games*. PhD thesis, Queensland University of Technology, 2008.
- [20] buttonpushertv. Meet my roll20 macro pad. [https://www.reddit.com/r/Roll20/comments/gk8e9w/meet\\_my\\_roll20\\_macro\\_pad](https://www.reddit.com/r/Roll20/comments/gk8e9w/meet_my_roll20_macro_pad).
- [21] cgloeckner. Pyvtt, 2021. <https://github.com/cgloeckner/pyvtt>.
- [22] Y.-L. Betty Chang, Stacey D. Scott, and Mark Hancock. Supporting situation awareness in collaborative tabletop systems with automation. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces - ITS '14*, page 185–194, Dresden, Germany, 2014. ACM Press.

- [23] Coleman Charlton, John Curtis, Pete Fenlon, and Steve Marvin. *Rolemaster*. Iron Crown Enterprises, 1980.
- [24] Adrian David Cheok. *Art and Technology of Entertainment Computing and Communication*. Springer London, London, 2010.
- [25] Ben Clothier and David E. Millard. Awash: Prospective story sifting intervention for emergent narrative. In Lissa Holloway-Attaway and John T. Murray, editors, *Interactive Storytelling*, pages 187–207, Cham, 2023. Springer Nature Switzerland.
- [26] Inkscape Developer Community. Inkscape, 2003. <https://inkscape.org/>.
- [27] Kate Compton. So you want to build a generator, Feb 2016. <http://www.galaxykate.com/buildagenerator-kcompton.pdf>.
- [28] Rui Craveirinha and Licinio Roque. Exploring the design-space: The authorial game evolution tool case-study. In *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology*, page 1–10, Osaka, Japan, Nov 2016. ACM.
- [29] Pavel Curtis. *Lambdamoo*, 1990.
- [30] M. Cutumisu, D. Szafron, J. Schaeffer, M. McNaughton, T. Roy, C. Onuczko, and M. Carbonaro. Generating ambient behaviors in computer role-playing games. *IEEE Intelligent Systems*, 21(5):19–27, Sep 2006.
- [31] Maria Cutumisu, Curtis Onuczko, Matthew McNaughton, Thomas Roy, Jonathan Schaeffer, Allan Schumacher, Jeff Siegel, Duane Szafron, Kevin Waugh, Mike

- Carbonaro, Harvey Duff, and Stephanie Gillis. Scriptease: A generative/adaptive programming paradigm for game scripting. *Science of Computer Programming*, 67(1):32–58, Jun 2007.
- [32] Paul Czege. *My Life with Master*. Half Meme Press, 2003.
- [33] Chris Davis. OpenRPG: Online virtual tabletop. <http://www.rpgobjects.com/orpg>.
- [34] Wim De Hert and Karel Crombecq. Dungeon alchemist, 2021. <https://www.dungeonalchemist.com>.
- [35] Bouncycastle Entertainment. Talespire, 2015. <https://www.talespire.com/>.
- [36] Obsidian Entertainment. Neverwinter nights 2, 2006.
- [37] Sony Online Entertainment. Star wars galaxies, 2003.
- [38] TaleWorlds Entertainment. Mount & blade, 2008.
- [39] Organization for Transformative Works. Archive of our own. <https://archiveofourown.org/>.
- [40] John Four. iPad app review: Rpg cartographer, 2010. <http://www.campaignmastery.com/blog/ipad-app-review-rpg-cartographer/>.
- [41] Fulvio Frapolli, Béat Hirsbrunner, and Denis Lalanne. Dynamic rules: Towards interactive games intelligence. *Proceedings of the 12th International Conference on Intelligent User Interfaces*, 2007.

- [42] Berserk Games. Tabletop simulator, 2015. <https://www.tabletopsimulator.com/>.
- [43] Failbetter Games. Fallen london, 2010. <https://www.fallenlondon.com>.
- [44] Foundry Gaming. Foundry gaming — creating foundry virtual tabletop — patreon. <https://www.patreon.com/foundryvtt>.
- [45] Foundry Gaming. Packages for foundry virtual tabletop. <https://foundryvtt.com/packages/systems>.
- [46] Foundry Gaming. Foundry virtual tabletop, 2020. <https://foundryvtt.com>.
- [47] Foundry Gaming. Crucible, 2023. <https://foundryvtt.com/packages/crucible>.
- [48] Richard Garriot. Ultima, 1981.
- [49] Felix Gillette and Thomas Buckley. Breaking the curse. *Bloomberg Businessweek*, Apr 2023.
- [50] Marcello A. Gómez-Maureira, Giulio Barbero, Maria Freese, and Mike Preuss. Towards a taxonomy of ai in hybrid board games. In *International Conference on the Foundations of Digital Games*, page 1–6, Bugibba, Malta, Sep 2020. ACM.
- [51] Andrew C. Greenberg and Robert Woodhead. Wizardry, 1981.
- [52] Don Greenwood. *Advanced Squad Leader*. Avalon Hill Games, 1985.
- [53] The Orr Group. Burnbryte, 2020. <https://burnbryte.com>.

- [54] Matthew Guzdial, Devi Acharya, Max Kreminski, Michael Cook, Mirjam Eladhari, Antonios Liapis, and Anne Sullivan. Tabletop roleplaying games as procedural content generators. In *International Conference on the Foundations of Digital Games*, page 1–9, Bugibba, Malta, Sep 2020. ACM.
- [55] Gary Gygax and Dave Arneson. *Dungeons & Dragons*. TSR, Inc., 1974.
- [56] John Harper. *Blades in the Dark*. Evil Hat Productions, 2017.
- [57] Ulf Hartelius, Johan Fröhlander, and Staffan Björk. Tisch: Digital tools supporting board games. In *Proceedings of the International Conference on the Foundations of Digital Games*, page 196–203, 2012.
- [58] JT Harviainen. A hermeneutical approach to role-playing analysis. *International Journal of Role-Playing*, 1:66–78, 2009.
- [59] Hempuli. Baba is you, 2019. <https://www.hempuli.com/baba>.
- [60] Hempuli. Baba is you press kit image 01, 2019. [https://www.hempuli.com/press/Baba\\_Is\\_You/images/image1.png](https://www.hempuli.com/press/Baba_Is_You/images/image1.png).
- [61] Shane Lacy Hensley. *Savage Worlds*. Pinnacle Entertainment Group, 2003.
- [62] Maurice P Herlihy and Jeannette M Wing. Specifying graceful degradation in distributed systems. *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, pages 167–177, 1987.
- [63] Emily Hery and Glenda Drew. Tool for map creation and map interaction during

- tabletop game sessions. In *Proceedings of the 2019 on Creativity and Cognition*, page 531–535, San Diego, CA, USA, Jun 2019. ACM.
- [64] Rich Hickey. Clojure, 2007. <https://clojure.org>.
- [65] Rich Hickey. `clojure.spec` - rationale and overview, 2016. <https://clojure.org/about/spec>.
- [66] Kenneth Hite. *Night's Black Agents*. Pelgrane Press, 2012.
- [67] Douglas R. Hofstadter. Metamagical themas: About nomic: a heroic game that explores the reflexivity of the law. *Scientific American*, 246(6):16–33, 1982.
- [68] Ian Horswill. Imaginarium: A tool for casual constraint-based pcg. *Proceedings of the AIIDE Workshop on Experimental AI and Games*, 2019.
- [69] Infocom. *Zork*, 1977.
- [70] Steve Jackson. *GURPS*. Steve Jackson Games, 1986.
- [71] Jonathan Johansson and Peter Lundberg. Tabula imaginarium - an ipad application for aiding a spatially separated tabletop role-playing game. Master's thesis, Chalmers University of Technology, 2012.
- [72] Shi Johnson-Bey and Michael Mateas. Centrifuge: A visual tool for authoring sifting patterns for character-based simulationist story worlds. In *The 17th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment: Programming Languages and Interactive Entertainment Workshop*, 2021.

- [73] Sergi Jordà. The reactable: tangible and tabletop music performance. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, page 2989–2994, Atlanta, GA, USA, Apr 2010. ACM.
- [74] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reacTable: Exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, page 139–146, Baton Rouge, Louisiana, Feb 2007. ACM.
- [75] Jesper Juul. *Half-Real: Video Games Between Real Rules and Fictional Worlds*. MIT Press, Cambridge, Mass, 2005.
- [76] Martin Kaltenbrunner, Sergi Jordà, Gunter Geiger, and Marcos Alonso. The reacTable\*: A collaborative musical instrument. In *15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06)*, page 406–411, Manchester, UK, 2006. IEEE.
- [77] Ville Kankainen, Jonne Arjoranta, and Timo Nummenmaa. Games as blends: Understanding hybrid games. *Journal of Virtual Reality and Broadcasting*, 14(4), 2017.
- [78] Ville Kankainen and Janne Paavilainen. Hybrid board game design guidelines. *Proceedings of the Digital Games Research Association*, 2019.
- [79] Thomas Knoll and John Knoll. Photoshop, 1990.  
<https://www.adobe.com/products/photoshop.html>.

- [80] Mehmet Kosa and Pieter Spronck. What tabletop players think about augmented tabletop games: A content analysis. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, page 1–8, Malmö, Sweden, Aug 2018. ACM.
- [81] Max Kreminski, Melanie Dickinson, and Michael Mateas. Winnow: A domain-specific language for incremental story sifting. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2021.
- [82] Max Kreminski, Melanie Dickinson, Michael Mateas, and Noah Wardrip-Fruin. Why are we like this?: The ai architecture of a co-creative storytelling game. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, FDG '20, New York, NY, USA, 2020. Association for Computing Machinery.
- [83] Max Kreminski, Melanie Dickinson, and Noah Wardrip-Fruin. Felt: a simple story sifter. In *Interactive Storytelling: 12th International Conference on Interactive Digital Storytelling, ICIDS 2019, Little Cottonwood Canyon, UT, USA, November 19–22, 2019, Proceedings 12*, pages 267–281. Springer, 2019.
- [84] Max Kreminski, Melanie Dickinson, Noah Wardrip-Fruin, and Michael Mateas. Select the unexpected: A statistical heuristic for story sifting. In *Interactive Storytelling*, pages 292–308. Springer International Publishing, 2022.
- [85] Max Kreminski and Noah Wardrip-Fruin. Sketching a map of the storylets design



- space. In *11th International Conference on Interactive Digital Storytelling, ICIDS 2018*, pages 160–164. Springer, 2018.
- [86] Max Kreminski, Noah Wardrip-Fruin, and Michael Mateas. *Authoring for Story Sifters*, pages 207–220. Springer International Publishing, 2022.
- [87] Christina Köffel and Michael Haller. Heuristics for the evaluation of tabletop games. *Conference on Human Factors in Computing Systems: Evaluating User Experiences in Games Workshop*, 2008.
- [88] Languard. Divinity: Original sin 2 - game master mode. <https://fextralife.com/wp-content/uploads/2017/09/divinity-original-sin-2-video-screencap-vignette.png>.
- [89] Andreas Larsson, Jonas Ekblad, Alberto Alvarez, and Jose Font. A comparative ux analysis between tabletop games and their digital counterparts. In *Extended Abstracts of the 2020 Annual Symposium on Computer-Human Interaction in Play*, page 301–305, Canada, Nov 2020. ACM.
- [90] Robin Laws. *GUMSHOE One-2-One*. Pelgrane Press, 2016.
- [91] Reactable Legacy. Reactable community. <http://community.reactable.com>.
- [92] LegendKeeper. Frequently asked questions, 2022. <https://www.legendkeeper.com/faq/>.
- [93] LegendKeeper. Legendkeeper, 2022. <https://www.legendkeeper.com>.

- [94] Yannis Lilis and Anthony Savidis. An integrated development framework for tabletop computer games. *Computers in Entertainment*, 12(3):1–34, Sep 2014.
- [95] Craig A Lindley and Mirjam Eladhari. Narrative structure in trans-reality role-playing games: Integrating story construction from live action, table top and computer-based role-playing games. *Proceedings of the Digital Games Research Association*, 2005.
- [96] ProFantasy Software Ltd. Campaign cartographer, 1993. <https://www.profantasy.com>.
- [97] Alexander Macris, Greg Tito, and Tavis Allison. *Adventurer, Conqueror, King System*. Autarch, 2011.
- [98] Carsten Magerkurth, Adrian David Cheok, Regan L. Mandryk, and Trond Nilsen. Pervasive games: bringing computer entertainment back to the real world. *Computers in Entertainment*, 3(3), Jul 2005.
- [99] Carsten Magerkurth, Maral Memisoglu, Timo Engelke, and Norbert Streitz. Towards the next generation of tabletop gaming experiences. *Graphics Interface*, 2004.
- [100] Carsten Magerkurth, Richard Stenzel, and Thorsten Prante. Stars - a ubiquitous computing platform for computer augmented tabletop games. *Video Track and Adjunct Proceedings of the Fifth International Conference on Ubiquitous Computing*, 2003.

- [101] Jimmy Maher. Playing wizardry, 2012. <https://www.filfre.net/2012/03/playing-wizardry/>.
- [102] Regan L. Mandryk and Diego S. Maranan. False prophets: Exploring hybrid board/video games. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, page 640–641, 2002.
- [103] Magnus Manske and Lee Daniel Crocker. Mediawiki, 2002. <https://mediawiki.org>.
- [104] John Markoff. *What the Dormouse Said: How the 60s Counterculture Shaped the Personal Computer Industry*. Penguin, 2005.
- [105] Chris Martens. Ceptre: A language for modeling generative interactive systems. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 11, pages 51–57, 2015.
- [106] Stacey Mason. *Responsiveness in Narrative Systems*. PhD thesis, University of California, Santa Cruz, 2021.
- [107] Maxis. The sims, 2000. <https://www.ea.com/games/the-sims>.
- [108] Alexander Mayben. Tabletop role-playing game design through a pattern language software model. Master’s thesis, University of California, Santa Cruz, 2020.
- [109] Ali Mazalek, Basil Mironer, Elijah O’Rear, and Dana Van Devender. The tvIEWS table role-playing game. *JVRB-Journal of Virtual Reality and Broadcasting*, 5(8), 2008.

- [110] mcglintlock. Dungeon draw: A new module to draw maps within foundry. [https://www.reddit.com/r/FoundryVTT/comments/qtatzi/dungeon\\_draw\\_a\\_new\\_module\\_to\\_draw\\_maps\\_within](https://www.reddit.com/r/FoundryVTT/comments/qtatzi/dungeon_draw_a_new_module_to_draw_maps_within).
- [111] Marc Miller, Frank Chadwick, John Harshman, and Loren Wiseman. *Traveller*. Game Designer's Workshop, 1977.
- [112] Jason Morningstar. *Fiasco*. Bully Pulpit Games, 2009.
- [113] moron4hire, Jun 2023. <https://news.ycombinator.com/item?id=36345851#36348918>.
- [114] Mesa Mundi. D20 pro, 2011. <https://d20pro.com>.
- [115] Myu-Unix and dolanor. Mirkwood engine, 2021. [https://github.com/Myu-Unix/mirkwood\\_engine](https://github.com/Myu-Unix/mirkwood_engine).
- [116] n Space and Digital Extremes. Sword coast legends, 2015.
- [117] Graham Nelson. Inform 7 programming language. <https://ganelson.github.io/inform-website/>.
- [118] NimsTV. [skyrim] how to's - the secret of lockpicking, Jun 2023. <https://www.youtube.com/watch?v=R5fZobDqQbU>.
- [119] Noppaseppele. Tips on playing ttrpg in roll20. <https://noppaseppele.fi/playing-ttrpg-online-roll20-tips/>.
- [120] Mikkel Paulsen. Introducing initiative.sh. <https://github.com/initiative-sh/initiative.sh/blob/main/README.md>.

- [121] Mikkel Paulsen. Initiative.sh, 2021. <https://initiative.sh/>.
- [122] Sandy Petersen. *Call of Cthulhu RPG*. Chaosium, 1981.
- [123] Éric Piette, Dennis J. N. J. Soemers Soemers, Matthew Stephenson, Chiara F. Sironi, Mark H. M. Winands, and Cameron Browne. Ludii - the ludemic general game system. *CoRR*, abs/1905.05013, 2019.
- [124] Bas A. Plijaer, Daisy O’Neill, Eloisa Kompier, Günter Wallner, and Regina Bernhaupt. Truesight battle grid - enhancing the game experience of tabletop role-playing through tangible data visualization. In *Extended Abstracts of the 2020 Annual Symposium on Computer-Human Interaction in Play*, page 103–107, 2020.
- [125] Jan Pokorný. A framework for modelling tabletop game rules. Bachelor’s thesis, Masaryk University, 2019.
- [126] Razeware. Battle map 2. <https://appadvice.com/app/battle-map-2/384800918>.
- [127] Realistic-Ad4965. Evolution of the kitchen table rpg. [https://www.reddit.com/r/FoundryVTT/comments/o8bpq6/evolution\\_of\\_the\\_kitchen\\_table\\_rpg](https://www.reddit.com/r/FoundryVTT/comments/o8bpq6/evolution_of_the_kitchen_table_rpg).
- [128] Aaron A. Reed. *Changeful Tales: Design-Driven Approaches Toward More Expressive Storygames* *E EXPRESSIVE STORYGAMES*. PhD thesis, University of California, Santa Cruz, 2017.
- [129] Aaron A. Reed. 1975: dnd, Feb 2021. <https://if50.substack.com/p/1975-dnd>.

- [130] Aaron A. Reed. 1980: Mud, Mar 2021. <https://if50.substack.com/p/1980-mud>.
- [131] Aaron A. Reed. 1990: Lambdamoo, May 2021. <https://if50.substack.com/p/1990-lambdamoo>.
- [132] Rob Rendell. *gtove*, 2017. <https://github.com/RobRendell/gTove>.
- [133] Ben Robbins. *Microscope RPG*. Lame Mage Productions, 2011.
- [134] Thom Robertson. Artemis: Spaceship bridge simulator, 2010.
- [135] Jason Rohrer. Sleep is death, Nov 2010. <http://sleepisdeath.net/>.
- [136] Jason Rohrer. Sleep is death tutorial 1: Controller basics, 2010. <https://www.youtube.com/watch?v=BYdVDmfbwjI>.
- [137] Jason Rohrer. Sleep is death tutorial 5: Advanced object editor usage, 2010. <https://www.youtube.com/watch?v=ygUbKF1qKcU>.
- [138] RPTools. Maptool 1.13, 2023. <https://github.com/RPTools/maptool>.
- [139] James Ryan. *Curating Simulated Storyworlds*. PhD thesis, University of California, Santa Cruz, 2018.
- [140] James Ryan, Ben Samuel, Adam J. Summerville, and Michael Mateas. Bad news: An experiment in computationally assisted performance. *Lecture Notes in Computer Science*, Nov 2016.
- [141] Ben Samuel, Adam Summerville, James Ryan, and Liz England. A quantified analysis of bad news for story sifting interfaces. In *Interactive Storytelling: 14th*

*International Conference on Interactive Digital Storytelling, ICIDS 2021, Tallinn, Estonia, December 7–10, 2021, Proceedings 14*, pages 142–156. Springer, 2021.

- [142] René Reinhold Schalleger. *The Postmodern Joy of Role-Playing Games: Agency, Ritual and Meaning in the Medium*. McFarland, 2018.
- [143] Udo Schroeter. Rolz, 2011. <https://rolz.org>.
- [144] Emily Short. Beyond branching: Quality-based, salience-based, and waypoint narrative structures, 2016. <https://emshort.blog/2016/04/12/beyond-branching-quality-based-and-salience-based-narrative-structures>.
- [145] Emily Short. Storylets: You want them, 2019. <https://emshort.blog/2019/11/29/storylets-you-want-them/>.
- [146] Strategic Simulations. Pool of radiance, 1988.
- [147] SmiteWorks. Fantasy grounds. <https://www.fantasygrounds.com/>.
- [148] SmiteWorks. Fantasy grounds game system usage through 2020Q4, 2021. <https://www.fantasygrounds.com/reports/2020Q4/>.
- [149] Alexis D. Smolensk. Trade system, Apr 2018. <https://tao-dndwiki.blogspot.com/2018/04/trade-system.html>.
- [150] Alexis D. Smolensk. Injury, 2023. <https://wiki.alexissmolensk.com/index.php/Injury>.
- [151] Ambitious Software. Dungeon mapp, 2011. <http://www.ambitioussoftware.com>.
- [152] Spellarena. Never ending dungeon, 2021. <https://neverendingdungeon.ai>.

- [153] Square. Final fantasy, 1987.
- [154] Greg Stafford. *Pendragon*. Chaosium, 1985.
- [155] Alex Stavrinou. Tableplop, 2020. <https://www.tableplop.com/>.
- [156] Patricia D Stokes. *Creativity from constraints: The psychology of breakthrough*. Springer Publishing Company, 2005.
- [157] Bethesda Game Studios. The elder scrolls v: Skyrim, 2011.
- [158] Desert Nomad Studios. A tale in the desert, 2003. <https://www.desert-nomad.com/>.
- [159] Larian Studios. Divinity: Original sin 2, 2017.
- [160] Peter Suber. *The Paradox of Self-Amendment: A Study of Law, Logic, Omnipotence, and Change*. Peter Lang International Academic Publishers, Bern, Switzerland, 1990.
- [161] John Sullivan. Shmeppy, 2019. <https://shmeppy.com>.
- [162] Philip Tchernavskij, Andrew M. Webb, Hayden Gemeinhardt, and Wendy E. Mackay. Readymades & repertoires: Artifact-mediated improvisation in tabletop role-playing games. In *Creativity and Cognition*, page 298–311, Venice, Italy, Jun 2022. ACM.
- [163] Arkenforge Team. Arkenforge, 2017. <https://arkenforge.com>.



- [164] Arkenforge Team. Arkenforge: Touch screen table support, 2017. <https://arkenforge.com/touch-screen-table-info>.
- [165] Critical Role Team. Critical role. <https://critrole.com>.
- [166] Inkarnate Team. Inkarnate, 2018. <https://inkarnate.com>.
- [167] Let's Role Team. Let's role, 2023. <https://lets-role.com>.
- [168] Obsidian Portal Team. Obsidian portal, 2007. <https://www.obsidianportal.com>.
- [169] Roll20 Team. Roll20, 2012. <https://roll20.net>.
- [170] Roll20 Team. The orr group industry report Q4 2020: 8 million users edition!, Feb 2021. <https://blog.roll20.net/posts/the-orr-group-industry-report-q4-2020-8-million-users-edition/>.
- [171] Wonderdraft Team. Wonderdraft, 2018. <https://www.wonderdraft.net>.
- [172] World Anvil Team. World anvil, 2017. <https://www.worldanvil.com/>.
- [173] Day 8 Technology. Re-frame, 2015. <https://day8.github.io/re-frame>.
- [174] Nicola Thouliss and Mitch McCaffrey. Owlbear rodeo, 2020. <https://www.owlbear.rodeo>.
- [175] Inc. Tilt Five. Tilt five, 2022. <https://www.tiltfive.com/>.
- [176] J. R. R. Tolkien. *The Lord of the Rings*. Allen & Unwin, 1954–1955.
- [177] TVTropes. Tvtropes. <https://tvtropes.org/>.

- [178] Elise van den Hoven and Ali Mazalek. Tangible play: Research and design for tangible and tabletop games. In *Proceedings of the 12th international conference on Intelligent user interfaces - IUI '07*, Honolulu, Hawaii, USA, 2007. ACM Press.
- [179] Vassal. What is vassal?, 2023. <https://vassalengine.org/about.html>.
- [180] Bryan “spacehabs” Versteeg. <https://twitter.com/spacehabs/status/1577314692544233473?s=20>.
- [181] Ramon Viladomat Arró. *ReactTable Role Gaming - RRG*. 2008. <https://www.youtube.com/watch?v=QfIrIK-m4Ts>.
- [182] Ramon Viladomat Arró. Reactable role gaming (rrg): Disseny, implementació i avaluació d’un entorn pel desenvolupament de jocs de rol per a taules interactives. Bachelor’s thesis, Universitat Pompeu Fabra, 2009.
- [183] Jordan Walke and Meta Platforms Inc. Reactjs, 2013. <https://react.dev>.
- [184] Noah Wardrip-Fruin and Nick Montfort, editors. *The New Media Reader*. MIT Press, 2003.
- [185] Andrew M. Webb and Pablo Cesar. Uncovering seams in distributed play of tabletop role-playing games. In *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, page 773–780, Barcelona, Spain, Oct 2019. ACM.
- [186] Business Wire. Hasbro to acquire D&D beyond from fandom, Apr

2022. <https://www.businesswire.com/news/home/20220412006151/en/Hasbro-to-Acquire-DD-Beyond-from-Fandom>.

- [187] Christopher Wolfe, J. David Smith, and T. C. Nicholas Graham. A low-cost infrastructure for tabletop games. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, page 145–151, Toronto, Ontario, Canada, Nov 2008. ACM.
- [188] Chris Zimmerer, Philipp Krop, Martin Fischbach, and Marc Erich Latoschik. Reducing the cognitive load of playing a digital tabletop game with a multimodal interface. In *CHI Conference on Human Factors in Computing Systems*, page 1–13, New Orleans, LA, USA, Apr 2022. ACM.

# Appendix A

## Interview Transcript Excerpts

After each interview, I drafted a transcript while listening to a recording of the interview. I removed occasional sections where discussion veered away from ROLE-PLAYINGGAME, game design, or other relevant topics. I edited for clarity by adding speaker labels to each line, and inserting editorial comments in brackets. I edited for concision by removing filler words (e.g. “um,” “like”) and false-start sentences (e.g. “What I think is that — My point is that RPGs are interesting.” became “My point is that RPGs are interesting.”) Finally, for ease of reference in the main document, I marked passages of particular interest with numbers in square brackets.

The full transcripts of these interviews are available in the supplement to this thesis, which can be found on ProQuest or eScholarship. For brevity, here I have only reproduced the key points which I specifically refer to in this thesis.

## A.1 Interview Excerpts from P1

P1: So I don't have a legend for this orange text [i.e. differential display text] ... but then how am I going to tell which character has access to which information? Do they each get a color? [1]

P1: Which is pointing me at — this wants to be the spreadsheet interface, but it can't actually promise that at the same level of fidelity for any arbitrary field you you might add. [5]

P1: Yeah. The tension I'm seeing here is you are saying to the GM, "Look, you can operationalize everything you're doing. Here is a beautiful language." But code is scary. So you are going to have to build this involved visual editor to round trip you from your [RPG design done on] paper, to the database, through this ontology of transactions that you've beautifully laid out here. And it's very difficult to make progress on, just because you have to spend all this time iterating on the representation, and the ontology suffers. You end up with a toy ontology because the representation is very hard to create. [10]

P1: And you can create an interface for — you have the proof of concept interface for [those GMs to] use programming languages. This is very HCI and PL [programming languages] kind of work, where you're trying to basically improve on the IDE experience somewhat, so that non-programmers can access the power of this transaction language [ed: i.e. the collection of DSLs implied by ROLEPLAYINGGAME, one each for defining actions, rules, items, etc.]. [12]

P1: Yeah, you're formalizing the foundations of the game design and bringing it to

people who want to see game designers using their [PL theory] stuff in the wild someday. [13]

P1: The converse of that would be to go to game studies and say, “Here’s a DSL. You want to know all of these aspects of programming language theory. I want to articulate why a DSL is something valuable and interesting for designers [to] use.” So those are two, you see, those are closely related papers, but they’re not the same one because you have to assume different knowledge on either side. [14]

P1: Okay. Fantastic. So between the interpret function and the preconditions [of an action to take place], you’ve essentially created storylets but for database transactions. [17]

P1: There’s a precondition. We don’t care about manually overriding the precondition since that’s an interface consideration — wild! And the postcondition is decomposed into these three units [i.e. events] that the database understands. So this is analogous [to] what Inform 7 wants to do within its language for verbs. [18]

P1: But I think what you’ve done here is you’ve articulated some connections that weren’t clear to me between Inform and LambdaMOO and your system. And that to me is a compelling survey of the history of, frankly, the metaverse. The thing that the metaverse promises, which is a perfectly extensible language of updates to a world model. [19]

P1: It’s going to keep being done, which is why it’s more relevant than ever to create a survey of: “Here is the history of how people have tried to kind of break databases over their knee and make them into into frames that can be extended so that you can solve your frame problem from good old symbolic AI.” [20]

P1: But you’ve identified an orthogonal element of what the GM does. [He] says things [to the players] like, “I see you have 15 geese. Maybe it’s time to talk about how you’re feeding 15 geese.” Even though the goose object did not previously have any special relationship to seed objects [i.e. something the goose might eat]. [22]

## A.2 Interview Excerpts from P2

P2: [time 19:30] The highlighted [i.e. differentially displayed] information only this character knows: that’s pretty cool. The way I’m used to playing D&D, the DM sitting right there will be like, “Oh, by the way, your character sees this thing in their darkvision,” but everyone hears that, and has to pretend that their characters don’t know. Which can be successful, but it’s cool that this way, it’s truly up to just that one character whether or not they inform the other players. [1]

P2: [After MJ asked which D&D editions he’d played:] I don’t know. [5]

P2: Study the tapes. [ed: referring to what sports teams do when they watch their past games to understand and improve their performance.] [6]

P2: Not really, honestly, because that seems like too much overhead. I think like I said, [for] the more rules-based, tactical parts of the game, your tool makes perfect sense and seems like it’s a huge help there. But then everything you do the rest of the time — having to make sure that the computer’s state is consistent with all of the stuff you guys did just by talking... [12]

P2: That seems like it might be real overhead, from a user perspective: the player’s

interaction. [13]

P2: This is kind of a different question, but is there any encyclopedia of NPCs or something like that? Because that seems useful. [14]

P2: That seems useful that if some player mentions, like, “Oh yeah, where’s that guy Bob we talked to?” and you [the GM] are like, “Who’s Bob? I forget.” Being able to check there and see, “Bob is a 35-year-old elf who’s the mayor of this town and we got in a fight with him about this.” [15]

P2: We’ve been talking about how different your game is from my game and that it seems like this tool is being tuned for your style of game. Certainly. And having the DM be able to decide what parts of the rules in logic and logistics are managed by the computer, and what parts they’ll do analog, seems like a good way for them to be able to accommodate their style of play with your tool. [19]

MJ: That’s a good point. It’s almost like it provides an adoption path. [20]

P2: Having a rules editor would be great. [21]

### **A.3 Interview Excerpts from P3**

P3: I don’t have any questions yet, but I’m sure some will emerge. The first thing I found really cool was the context-sensitive information distribution. For instance, if a character has good skills, they’ll get info that others don’t. I thought that was cool, and a good use of everyone having their computer. It allows for sharing information, not secretly, but privately, and in a way predetermined by skills, which is more efficient than



writing it on a slip of paper and handing it over, making everyone else feel weird. [1]

P3: But yeah, I thought that that was really cool. I also liked — actually, there were a lot of things that I liked. I did like adding the troll, Gluzznub, or whatever the troll's name was. [2]

P3: Yeah, I thought that was cool. I thought that it looked pretty straightforward. And like you said, it's a problem with prep for RPGs in general, where you can prepare forever, and then the first thing that your players do will be something you didn't prepare for, even if it's super minor. And, well, the decision-making process after they throw whatever wrench at you is, I guess, the same [ed: between VTTs and analog gaming]. [But] trying to represent it in Roll20 or something can be really hard. [3]

P3: Because it's like, I have to get a token and then assign, but I can't [pick a guy and assign him a token, i.e. go the other direction]. And I feel like more often than not, I just will write down what I want it to be on paper notes and just play from that instead. [4]

P3: Watching the video, I think it definitely has some interesting utility that I hadn't really thought of. I feel like a professor who says, "No computers in the classroom, because you won't be taking notes, you'll be looking at whatever." I kind of feel that way. Yeah, if I step back and think about it, it does have huge utility. There are a lot of ways it can improve gameplay. You can do quick reference for things, take better notes, have more information readily available and searchable. You're not leafing through your rulebook to find rules when you need them. But, as someone who's GMed for a long time, I know it can be easy for people to lose focus. And so, part of me having the

computer is worried, like, are you dialed in? [6]

P3: It's harder to multitask on, yeah. [7]

P3: One good example from your video was clicking on move, and it just shows where my guy can move. In Dungeons and Dragons, I have to remember I have a 30-foot movement range, and if I'm wearing heavy armor, it's restricted. Even in Roll20, you have to use the measurement tool. [9]

P3: No, they don't. [10]

P3: You have to measure to 30 feet. So having it pop up showing all the spaces you could move to is a really great tool. [11]

P3: Right now, I think one of the big things that causes a lot of drag in tabletop roleplaying games, and there's not as much of in videogames, is decision paralysis and possibility space. I get decision paralysis in videogames too, but with roleplaying games, even in Baldur's Gate 3, which I just started playing, and 5e D&D — same rule set basically — you have a 30-foot movement range and can do one attack and one other action on your turn. [13]

P3: Looking at a videogame, it often concisely shows where you can go, which guys you can attack, which allies are in spell range. In contrast, playing on the table, it's more abstracted. You still have that 30-foot movement range, but it's not visually represented in the same way. It's not doing that back-end [logic]. [14]

[ed: by back-end, P3 means the calculations performed by a videogame performs in order to show the aforementioned visualizations, and other concise expressions of game rules, which he was talking about prior.]

P3: I feel like the videogame lets you focus on higher-order thinking, deciding the best course, not the back-end processing of calculating movement and actions: “If I can go six squares here, I can do this.” [15]

P3: I think it’s a different kind, but similar. A big thing in planning is the info gathering. It’s like not having all the information, which is okay, but deciding whether to buy a bunch of bags of grain or one cool magic sword. And then figuring out, can we carry all the grain? Can we buy the grain? Can we haggle the grain down? Can we even do that? That exploration process can take a really long time. [16]

P3: [Regarding expanded differential display] Oh wow, that’s cool. That’s really cool. [17]

P3: But we really ran into a wall with things beyond that, like rules for NPCs, how much equipment people can have, and the length of an encounter. Deciding what’s fun but not too much is where we hit our wall a bit. [18]

P3: Right. Another struggle is harm or damage, like how much people are taking, dealing, healing in an encounter or session. That might be easier to automate or have a tool help with. [21]

P3: We also have complexities, like a class that takes damage to do more damage. Introducing that and accounting for how often players might use that is challenging, especially with certain player styles. If we play with [a mutual friend], you know he’s going to be doing that “take damage to do damage” move all the time, so how do you account for that [in a simulation]? [22]

P3: [Regarding story sifting] That would be really cool. [25]

P3: Yeah, really cool. [27]

P3: I think you already have part of that informational side with your big trade table, like, “Computer, tell me how much they could get for selling a million bags of rice right now.” [30]

P3: And one cool thing about the trade table, because I’m a freak, is the “distance from production” aspect. Like, if I’m Rogar the fighter and we’re near an iron mine, and we’re going somewhere that doesn’t have a lot of iron, I might think of buying some iron and selling it there for a profit. [31]

P3: And that can help you as a GM, too, because if your players are like, “What we’re doing in this adventure is we are going to build the best farm in the world. Step one, we’ve got to paint our barn red, because duh.” And then you [the GM] know where they’re going to go, or you know where they might go to get the best deal on red paint. You could provide them with that information. They can use it. [32]

P3: And that’s like a moderately complicated, conditional statistics problem. The numbers are small, but when you add the condition “if the numbers come out this way, then I want to do this,” it suddenly becomes something you can’t just do by multiplying 1 over 6 times itself a couple of times. We can figure out what adding another die to the roll does probability-wise. Well, I can’t, because I’m not smart, someone at our table can, or probability.com, or whatever. But I have nine of these resources I can spend. What’s the best way to spend them? It would be cool if there was a reference tool for that. [39]

P3: I don’t think so. I think this is a really cool project, and I think you’re asking the

right questions. [40]

## A.4 Interview Excerpts from P4

P4: Fundamentally, [my reaction] is “useful.” I think, like you were just saying, obviously this wouldn’t be the final layout [of] the UI ... like, a GM could maybe get familiar with it, but I think that sometimes the GM’s role has to be making the game easier to understand for the players. So I could see it being either something that the GM would just have and communicate to the players, or it being a little more UI-friendly [so the players could use it directly]. So you can never have played a roleplaying game before, but get in there and be like, “Oh, yeah, this is my character. I know where the menu is [and] I can get to that quicker.” [3]

P4: I was just talking about coding on the fly, right? And the troll was a great example of that. Suddenly there’s a gap in your universe of something, and you need to fill it in. How can you use this to quickly fill it in in the moment? [4]

P4: [time 44:05] Definitely. I would also say, coming at it from a theater perspective, and also filmmaking, if you’re going to be putting a movie [or play] together, time management is one of the most important things along with the story. It’s like, how do you tell a story in 90 minutes? The first time you write out your screenplay, it’s going to be like three hours long. Yo’ve got to make it as efficient as possible, which is also why writing short stories can be harder than writing a full-length novel. And when you’re an adult, [there’s also] time management: how much time do you actually have

with your group of players? There's probably someone missing from the game, [or] who needs to go pick their kid up from something. So five or ten minutes of looking in a book for something [is] cutting into [more] valuable use of time. [10]

P4: We played around with, at least in high school, a different type of health system that felt a little more realistic than having a certain number of points [before] you're gone ... And then I think P3 might have tossed around the idea of having a living economy where, depending on what happens in one town, the value of wheat or something might increase or decrease in another one. But if you create a new town, how do you factor all of that in? And that would never have gotten started. [11]

P4: Sure. So I would say that one of the great things about Traveller is — like in a lot of sci-fi games or sci-fi stories, you're trying to create a world that seems real and lived in, that it has rules that you can relate to. So it feels like it could be the future from now. So the choices and stuff that you make, or how the world is like at that time feels more impactful because it's almost like you're creating a story in the future within your actual timeline of your life, but you'll be too old to actually see that happen in the future. So you need something that feels robust. If you go to a new planet, you want it to feel like there's stuff that's going on there, whether or not you ever landed there. [13]

P4: Some of the cool stuff that I like about Traveller [are the] many systems. I would love for it to be on a virtual platform that I could just pick up. Like if I want to build a spaceship from scratch, I almost [want] a side game to pop up. [14]

P4: Exactly, this idea of having many games in a game is something I think also really expands a world or makes it seem more real. I'm thinking of Skyrim, right? You have

crafting, you can build your house, and there's so many other things that have their own little rule systems, even lockpicking. It's not like some games where lockpicking is literally you press a button and based on your skill points, you might get in or not. [Instead] you're actually trying to break in. Or oh my god, like GTA 4 when, what's his name? The guy calls you up and wants to play darts or go bowling. [15]

P4: Sure. Especially for world building, it's about filling in the gaps of what you don't want to create. Like, if you want a big city, you might have some ideas of its biggest resources, such as mining — something to add flair — but then you have to fill in the gaps. You don't want to limit yourself. So those tools come in to fill in all those gaps you don't have time to do. [18]

P4: Yeah, I think a contrast, if we're talking about player experience, would be if you have a system that pre-generates all the NPCs in a city, versus generating and affecting the NPC generating system as it goes when you go into a new shop. So no NPCs were created until you go to this new shop. That shop has these NPCs generated that are curated for the specific role, and then that affects that system so that the next time you generate an NPC, they won't need another blacksmith. So they'll take that out of the equation. To the players, they don't really notice the difference. It's like as they're coming across these NPCs, they [begin to] exist. But you [ed: the player] can assume that the rest of the world is filled with people because, when you go to this town or you go here, there are people. Beforehand, there aren't, but only the GM really has to deal with that. [19]

P4: I think the GM is like the bridge between the world — all of the generation — and

the players. It's about enabling the GM. You could generate all the characters with all details and have it as a giant list. But that's not easy for the GM to use, to suddenly pull someone out of that pack and be this character. So while you could potentially generate that stuff ahead of time, what the GM needs in the moment is what gets pulled up [on screen]. [22]

P4: So, it's like, I don't want to see everyone in the town, I want to see who's in this room. What about this character do I need to know? What are his personality traits, or how does he talk? [23]

## **A.5 Interview Excerpts from P5**

P5: I'm not going to lie. I would play this right now. I know as the creator, you're always looking at something like, "Ah, man, this really sucks. I hate this. I wish I had this feature." But I think this is, at the very least, off to an excellent start. [1]

P5: OK, very nice. Yeah, I like this a lot, particularly the GM, or DM, being able to fill in the gaps of the code. [4]

P5: As both, but particularly as the player. A 5e player can do nothing, essentially. Just kind of drift along. If you're trained to do that, it's very difficult to break out of it. Even as a DM, every word is planned out for you in advance. And those who have deviated and try to make their own stuff are still kind of locked into those ways of thinking. [11]

P5: And that's why I like your system, is that you can see all this data. The data is



right there in the app, and I don't have to go to another screen or another app. If it's all right there, for these complicated inventory things, and I don't have to add it up on paper — I mean, there's a certain kind of player that will do that. I do that on paper when I play. But I've been in math disciplines my whole life. That's normal for me. If we're gonna reach the normal person, it has to be easy to do this stuff, so they can free up that brain space to go and make these decisions that they haven't ever had to make before in a roleplaying game. [13]

P5: The barrier to entry does weed out a lot of people who are gonna be too lazy to play a game like this, but I think a lot of those people can be trained. I think they're lazy because it's easy to be lazy. It's so much easier to be lazy that there's no benefit to making the effort to go up. And so if we can shave off some of the rough edges — [15]

P5: Yeah. Giving people a ladder, and eventually [some people will] try it and like it. Not everyone will, but we'll have more of a chance with the person who can be convinced, if you don't have to sit there and hold their hand with all of these, essentially, hacks that you've had to cobble together to play the game. [17]

P5: Yeah, that's simple. I like that a lot. So you're actively manipulating the way the character views the world, which is important because it's hard to be anything other than you. This has been a roleplaying challenge forever. [21]

P5: I've got a really dumb character, and I'm really smart. Or my character is really smart, but I'm kind of a dumbass. So how do you work that? And if you have this layer over everything where the computer, or the model, is filtering everything based on your attribute — You know, I like that a lot. How things react. How the world around you

reacts to you, the party and things like that. [22]

P5: I think I can see what you're describing, but I also don't have the language. I don't think anyone does. But I think it's a great goal to figure out the terminology. And, to go back to adoption, maybe that is part of our job now, at this particular time, while we're building these tools: to figure out the terminology. How do we talk about these things? [24]

P5: You know, the ontology of this space. And then, once that's kind of worked out in broad terms, that can kind of pick up other people to grab onto those tools that we've created. Even though we didn't push it to its edge, we got it to where other people could grab it. Right now, you can feel it in your hands, but it's ephemeral. We don't quite know what we have yet. We know it's there, but we just have the vaguest contours of what it is. And if we can describe it in enough detail, others can come along behind us and figure it out. [25]

P5: Yeah, it's not about what goes on in the games. When we have a rule and we want to test how it works, that's still subjective at some level. The games we play are more objective than 5e or Dungeon World. But at some level, it still has to come down to, "How do people react to the rule, and does it make their game experience better?" [26]

P5: Exactly. Exactly. That's a very tough question. It's easy sometimes when you have some objectivity to be able to point to those effects, but I don't think you can ever escape fully the subjective nature of the player receiving the rule, or conversely, how easy it is for the GM to implement [the rule]. [27]

MJ: Nobody's actually trying to investigate those dissonances, which is why I like Alexis

so much. He cares about that stuff. If I pulled any five papers from my bibliography right now and showed them to you, they would all open with phrases like, “RPGs are collaborative storytelling exercise.” No one is talking about design. There’s no design culture around RPGs! [28]

P5: That would be fantastic. [32]

P5: The same goes for anything you can make easier for the player to do. Knowing how to type semicolon-separated values is not an essential part of the game, and there comes a point at which that is just an obstacle. [38]

P5: A lot of things have to be happening at the same time, and I think this ties back into something you said a half hour ago about different levels of detail. When you’re in the battle map, the market doesn’t matter so much, but it doesn’t matter zero. [41]

P5: When you’re talking about other things going on in the background ... when you open up Roll20, you see the battle map. And you can put other pictures in there on different screens, but the whole thing is based on that view. The advantage of a web-based app, or even one locally running, is that you can have all these different things and toggle between them quickly and define whatever views you. Sometimes you just want to look at information. You don’t need a map. You need a list of the characters. Things like that. [44]

P5: So there’s a full distance. A maximum movement area. I should look at an individual hex and say, okay, you’re getting there somehow, and instead of trying to figure out the valid combinations of AP to reach some hex, just [work backward from each destination hex to the source]. I don’t remember why exactly I found that unsatisfactory,

but it was a reasonable comment. And both of us were trying to think in kind of this way, which is we need to apply videogame techniques. How much movement does your character have, and how can we apply it to the game world? And we're going to use pathfinding to do it. I actually was using a modified pathfinding algorithm just with step lengths of various sizes. But you're right, none of these programs do this, and it would not be that complicated to build, even if they had to build out a little battle map maker. It'd be trivial. [45]

P5: Sure, this was great. I'm glad I could be of help. Let me know when your thesis gets submitted. I'd love to take a look at it and read through it. And I think you're on a great track, so when the app comes out, drop me a link and I'll get it and plug it. [50]