Universidad de Valladolid

Escuela de Ingeniería Informática de Segovia

Trabajo de Fin de Grado

Grado en Informática de Servicios y Aplicaciones

Dragon's Memoir

Autor: Iván Nieves Stantcheva
Tutor: Fernando Díaz Gómez
Curso 2024–2025

Dedicated to my parents,
who many a time have asked when this project would be done.

No generative AI models have been
used in the creation of this project.

Contents

Abstract **d**

1 Introduction **1**
- 1.1 Document Structure 1
- 1.2 Overview and History 1
- 1.3 Prior Art 3
  - 1.3.1 *Fire Emblem* series . . 4
  - 1.3.2 *Tactics* series 5
  - 1.3.3 *Wars* series 6

2 Game Design **9**
- 2.1 Gameplay 9
  - 2.1.1 Gameplay Loops . . . 9
  - 2.1.2 Game Mechanics . . . 11
  - 2.1.3 UI and Input 15
  - 2.1.4 Runtime Reqs. 18
- 2.2 Worldbuilding 19

3 Development Plan **20**
- 3.1 Software and Tooling 20
- 3.2 Budget Estimates 22
- 3.3 Dev. Methodology 23

4 Implementation and Testing **25**
- 4.1 Software Architecture 25
- 4.2 Implementation Details . . . 26
- 4.3 Testing and Debugging 28

5 Conclusions **29**

Articles Referenced **30**

Wiki Articles Referenced **32**

Games Referenced **33**

Abstract

Dragon's Memoir is a work-in-progress tactical role-playing game inspired by the *Fire Emblem* video-game series. From a game-play perspective, it tries to expand upon already established mechanics rather than create a completely new genre of its own. From a world-building viewpoint, Dragon's Memoir's story takes place in a world loosely based on that of *Chroma: Bloom and Blight*[26] and on Brandon Sanderson's Cosmere, featuring an original magic system and unique mythology.

This project elaborates on three video-game series that have inspired to varying degrees some features of Dragon's Memoir before going into detail on the game's design and development process.

Key words: fantasy, hard fantasy, role-playing games, turn-based strategy, video games.

Resumen

Dragon's Memoir es un videojuego de rol táctico inspirado en la saga de videojuegos *Fire Emblem*. En lugar de reinventar el género, el juego trata de expandir mecánicas ya asentadas en él. El mundo en el que ocurre la trama de Dragon's Memoir está muy vagamente inspirado en el de *Chroma: Bloom and Blight*[26] y en el Cosmere de Brandon Sanderson, junto con mitología y un sistema de magia originales.

Este trabajo describe tres sagas de videojuegos que han inspirado algunas características de Dragon's Memoir y después detalla el proceso de diseño y desarrollo seguido para la implementación de este.

Palabras clave: estrategia por turnos, fantasía, fantasía dura, juegos de rol, videojuegos.

1 Introduction

1.1 Document Structure

This project report is split into five chapters, roughly corresponding to the five stages of a project: introduction, design, development, implementation, and finally a short conclusion chapter.

This introductory chapter will describe Dragon's Memoir in broad strokes, explain some of its history and delve into prior art in the genre that it belongs to.

The second chapter will go into more detail regarding the game's mechanics and design, concluding with a brief description of the universe Dragon's Memoir plot takes place in.

The third and fourth chapters deal with development and implementation, explaining the tools and methodology used to develop the game, as well as some implementation and testing notes.

Each chapter is split into several sections, as appropriate for each of them; a broad overview paragraph that sums up the content of one or more sections is included before them together with links to the previous, table of contents, and next sections.

1.2 Project Overview and History*

ToC — Next

This is your story. It all begins here.

Path to Exile's flavor text, in *Magic: the Gathering*.

tl;dr — Dragon's Memoir is a tactical role-playing game inspired by *Fire Emblem Awakening*[36] that has been in development for eight years, at some point being written in C++, Xtend, and Java. It doesn't try to blaze a trail as much as expand on well-established tropes in its genre.

*External links in this section have been last accessed on 2025–05–25; dates throughout this document are specified in the format YYYY–MM–DD.

Design and development of Dragon's Memoir began back in 2017 after finishing a playthrough of *Fire Emblem Awakening*.[36] It was originally conceived as a spiritual successor to that game, and at the time it seemed a relatively simple project with which to learn programming.

Originally the game was to be written in C++ using Qt Creator as a front-end library.The project would eventually be rewritten in Java so as to avoid possible licensing issues regarding the usage of Qt Creator's community edition.

While Java was the language Dragon's Memoir was rewritten into, for three years Xtend was used instead as a Java transpiler; IDE performance issues and the eventual conclusion of Xtend's development motivated another rewrite of the game's code back into Java, language Dragon's Memoir has been written in since late 2021.

Concurrently with code development, the game's world and plot was being refined little by little, with features inspired by Dungeons and Dragon's campaigns and Brandon Sanderson's books on the Cosmere.

Dragon's Memoir does not intend to revolutionize the tactical role-playing game genre and instead expands on well-established mechanics without bringing in an overwhelming amount of changes to these well-known systems. Of note among these revised mechanics is Dragon's Memoir's weapon system, which features partially breaking weapons and a unique advantage system expanding upon *Awakening*'s[36] without being as complex as *Dark Deity*'s.[28]

Unlike in the former game, Dragon's Memoir's players don't directly participate in the game's plot and is relegated to a spectator role. This is not to say that the player's choices do not affect the game's plot at all (cf. *Dark Deity*[28]), but that the player lacks an "avatar" representing them in-world.

At the time of writing, Dragon's Memoir is undergoing a relatively large user interface refactor, adding some much needed visual pizzazz and removing some limitations that have plagued the game for years, particularly the inability to resize the game's window to any size other than 800 px by 600 px. That aside, the game currently features ten playable chapters (and a sneak-peek into the eleventh) featuring fifteen playable characters (and a sneak-peek into three more).

1.3 Prior Art

Name three examples.

Epigrams, Gwern Branwen

Prev – ToC – Next

tl;dr — Dragon's Memoir is a turn- and grid-based tactical role-playing game, varyingly related to the *Fire Emblem*, *Final Fantasy Tactics* and *Wars* game series. *Fire Emblem Awakening*[36] is the primary inspiration of the game.

If we were to fully detail the genres Dragon's Memoir belongs to, we would categorize it as a turn- and grid-based tactical role-playing game.

"Grid-based" simply means that the placement of characters and structures are bound to a relatively coarse grid, like the squares of a chessboard. Each character occupies their own square regardless of size, and movement requires an empty square to traverse (like the movement of a rook in chess, for example). There are plenty of games in which this is not the case, where characters and objects aren't restricted to "integer steps" and where a unit's size affects the space they occupy; a notable example of this variation is *Baldur's Gate 3*[23] (which perhaps misleadingly is mechanically closer to the *Divinity: Original Sin* series[29,30] than to the first two *Baldur's Gate* games[21,22]).

A game being grid-based often implies that it is also "turn-based", although exceptions abound, such as *One Step from Eden*[40] and *Crypt of the Necrodancer*.[27] In this context, turn-based (as opposed to "real-time") indicates that gameplay advances in discrete time steps rather than continuously (compare *Brogue*[24] with *Unicorn Overlord*,[43] for example). Within this paradigm, there are two common ways to determine the order in which units act: either all units of a given side (player-controlled, enemy, or neutral) act at the same time (or consecutively, as in the *XCOM* series[46,47]), or turns are interleaved, the exact order being determined by a given unit's abilities and/or with an "initiative roll"; Dragon's Memoir follows the former round structure.

Lastly, a tactical role-playing game combines mechanics from both strategy and role-playing games (henceforth "RPGs"): characters' abilities grow as the game progresses, and those characters must be controlled in large-scale battles rather than only micro-managed in skirmishes. This contrasts both with "pure" role-playing games, where fine-grained unit control is restricted to battles the whole player's party participates in (as in *Ara Fell*[20]), and with "pure" strategy games, where an over-arching story and persistent character growth are often missing (such as *Dota 2*[31]).

Even within this three-step classification there is room for further subgenre distinction that we could descend into, but beyond this point the differences become less significant. Instead, we will detail three sagas whose games also fall into the turn- and grid-based tactical RPG genre, examples of their "offspring", as well as explain their relation to Dragon's Memoir.

1.3.1 *Fire Emblem* series

Dating back to 1990,[39] the *Fire Emblem* series by Intelligent Systems is the saga mechanically closest to Dragon's Memoir. Also related to this series is *Dark Deity*,[28] a recent indie game whose origins are akin to Dragon's Memoir's[3] and which could be considered a far relative in terms of scope.

The games' stories follow the deeds of a noble or, in more recent games, an avatar of the player within the context of a war, the player determining their and other members of their army's actions in battle. Each unit in the game has a class that determines which weapons they can use, which skills they learn as they gain experience in combat, to which other classes they can "promote" to once they reach a high enough experience level, and whether the unit is particularly weak to a certain type of weapon (flying units are weak to bows in most games of the series, for instance).

The three main melee weapons in the series are subject to the "weapon triangle":[17] a rock-paper-scissors relationship between swords, lances and axes. In combat, swords are effective against axes, increasing the chance of successful attacks against, and sometimes also increasing damage dealt to, units wielding axes. The same happens to lancers against units that use swords, and to those that use axes against lances. In some games, this advantage also applies to weapon types beyond those three, such as magic weapons in the games that feature a three-way magic system. Weapons in Dragon's Memoir also abide by this extended weapon triangle, but effectivities (called "colors" in-world) are completely detached from weapon types; this is explored in a optional subplot within the game.

There are two other differences between Dragon's Memoir's and *Fire Emblem*'s weapon systems: First, there are no "weapon levels"[16] in Dragon's Memoir—if a unit's class allows them to use a weapon, they can use all weapons of that type. Second, a weapon's base damage (its "might"[12]) can change as its durability decreases*: a sword becomes dull as it is used, and spellcasters grow frugal as their spells' reagents wane.

Another prevalent mechanic in the *Fire Emblem* series are character supports: some pairs of player-controlled characters build up rapport throughout the game by spending time together—that is, by fighting enemies when they are near each other—which is told as a series of conversations in-between main story chapters once enough support is attained between two characters. The rewards for reaching these support thresholds are stat increases that only apply when the two units are together in combat; Dragon's Memoir extends these rewards by granting unique weapons and skills, on top of intertwining conversation arcs and adding requirements to some support chains to further deepen the support system.

*Only breakable weapons can have these "durability steps", which need not decrease damage as the weapon breaks—the edge of a crystal axe might become sharper as it chips, for example.

1.3.1.1 *Fire Emblem Awakening*

As mentioned before *Fire Emblem Awakening*[36] was both a turning point for the series[6] and Dragon's Memoir's reason for existence. As a matter of fact, it was an issue with single scene that sparked development: discordance between narration and game interface.

In terms of gameplay, *Fire Emblem Awakening* has all the characteristics of a *Fire Emblem* game as described above, with the weapon triangle being restricted to the three main physical weapons. Additionally, the game features a two-generation character system: reaching the maximum possible level of certain support pairs will allow their children to join the player's army; this system was also featured in the series' next game,[37] and there are plans for future plot arcs of Dragon's Memoir to feature a similar system without resorting to time travel[15] in the way that *Awakening* and *Fates* handle it.

Unlike in subsequent games in the *Fire Emblem* series, the player may travel along the world map[18] to purchase weapons and fight skirmishes to increase their units' levels, but not access their base camp proper.[13] In Dragon's Memoir the situation is the opposite, as in *Fire Emblem Fates*:[37] the player cannot leave their camp, and the shop and training grounds are available there instead. This base of operation also serves as the level used for some story chapters and as a backdrop for some support conversations (which in *Fire Emblem Awakening* all use a generic background regardless of where the conversations take place in).

1.3.2 *Final Fantasy Tactics* series

Final Fantasy Tactics is both a 1997 tactical RPG[35] and name of the series that game started. This series is frequently abbreviated to *Tactics*, word that often appears as part of the title of games that feature similar mechanics, as in *Fae Tactics*[33] and *Dream Tactics*.[32] We mention *Fell Seal: Arbiter's Mark*[34] as a recent notable game that fits within the series' paradigm.

Where the *Fire Emblem* series focus combat on the weapon triangle and character supports, relegating abilities to a more passive role, *Tactics* games focus on active skills. The most common action a unit in a *Fire Emblem* game takes is to "just attack" another, whereas in *Tactics* that character would almost always be ordered to use one of the many skills their class taught them. In addition to having an overarching experience level that determines their stats, each unit has their own class levels, which determines which class skills they can use. It is not uncommon for some classes to require reaching a certain level in another class before being able to "reclass" a unit into the former, which, together with a lower maximum class level, builds a class promotion tree more complex than those in the *Fire Emblem* series and facilitates army diversification.

Damage and equipment also work differently in the two series: In the *Fire Emblem* series damage is either "physical" or "magical" with most units locked into one of them, while in *Tactics* skills deal damage of more "concrete" damage types such as "fire" or "slashing" and units have access to several such damage types at any given time. Similarly, items in the *Fire Emblem* series are limited to (almost always breakable) weapons, healing staves, and consumable items; in *Tactics*, all equipment is unbreakable (although consumable items exist), and there are different armor pieces that compliment a unit's innate stats to better defend themselves against certain damage types. Dragon's Memoir follows *Fire Emblem*'s lead and only has physical and magical damage types, and no armor items.

The most noticeable difference between *Fire Emblem* and *Tactics* is in the display of battles: levels in *Fire Emblem* are displayed in a square grid (like the squares of a chessboard), and are vertically flat regardless of the terrain the battle takes place in, whereas in *Tactics* battles are often displayed in isometric perspective and height plays a role in determining movement and skill range.

Somewhat related to isometric perspective, and a mechanic common in *Tactics* games is unit facing: at the end of a unit's turn, they decide on a direction to face along grid lines. Attacks are more effective when done from the sides or the back of the target. This mechanic was once considered for Dragon's Memoir but eventually scrapped during development; some class skills in the game partially implement this system in a simplified manner.

Also a notable difference between *Tactics* and *Fire Emblem* is the reliance on mercenaries: rather than new characters being added to the player's army throughout the whole story, the player is given the option—and heavily encouraged—to hire mercenaries to fill in their ranks. Currently there are no plans to implement unit recruitment in this manner in Dragon's Memoir, although it is not entirely out of the question assuming it can be made to fit within the game's narrative.

Overall, the series' influence on Dragon's Memoir is limited. While skills in Dragon's Memoir are stronger than in the *Fire Emblem* series, they aren't the focus of the game's mechanics, and the class promotion system is yet to be decided on.

1.3.3 *Wars* series

One last notable series within the turn-based tactical RPG genre is the *Wars* series by Intelligent Systems, sometimes called *Advance Wars* after the 2001 game of the same title.[19] Recent games of note within the series' paradigm are *Wargroove*[44] and its sequel.[45]

Unlike in the *Fire Emblem* and *Tactics* series, where the player controls each unit of their army separately, in *Wars* the player gives orders to whole *squads* of their army. Also unlike in the

former series, as a squad's health pool decreases, so does their combat power (similar to how the Red Fog option works in XCOM[14]), meaning damaged units deal less damage and are more likely to be completely defeated.

Chapter objectives in *Wars* vary slightly from those in *Fire Emblem*, as the most common goal is not to defeat all enemies, but to defeat the opposing side's leader and take over the opponents' headquarters. Likewise, the losing condition is not just losing the main character's squad, but also having an enemy reach the player's base.

There is little in terms of character progression or customization in the *Wars* series. Levels are self-contained and the player is incentivized to "purchase" reinforcements during them using resources acquired in the level itself. Squads present at the start of a chapter are of a given, fixed unit type (infantry or ships, for example) that cannot be changed, and similarly, reinforcements cannot change types once summoned.

One notable feature of some *Wars* games are the so called "Powers": powerful active abilities that the player can use once per battle or on a cooldown to help them overcome their opponents. Dragon's Memoir does not feature these abilities as they are unjustified from an in-world perspective and severely warp the game's balance around them.

1.3.3.1 *Symphony of War: The Nephilim Saga*

Symphony of War: The Nephilim Saga[41] is a game borrows features from all three sagas we have described: from *Fire Emblem*, support conversations; from *Tactics*, the class promotion and equipment systems; and from *Wars*, the player controlling squads of units instead of individual units in combat. Unlike in *Wars*, however, the player is allowed more granular control over the units in their squads, being able to freely customize their class and position within the squad, making a broader variety of strategies and army compositions possible.

Rather than bringing in mechanics from *Symphony of War*[41]—many of which are already present by virtue of appearing in the *Fire Emblem* series,—Dragon's Memoir brings in *lessons* on how to implement them:

- The player needs information and control to decide their course of action. In particular, combat forecasts should give enough information to roughly determine their outcome (barring randomness in whether a unit will miss their attack *et cetera*).

- Information displayed should be clear, accurate, not out-of-date, and displayed in a way that does not overwhelm the player. Tooltips sometimes work better than text boxes.

- The game's interface should act in accordance with the player's input method. When using mouse input, clicking on a menu element should perform that element's action, not select the menu.
- The game's mechanics should be clear and consistent, and must be explained properly. If a unit can take a given action, all other units should also be able to take that action; if this is not the case, the player should clearly be told why.
- It is better to not include a mechanic whose effects are unclear than to include it only for the sake of realism: Weather and day-night cycles that only affect some units in unspecified manners should be left out, but fog of war that restricts vision is perfectly acceptable.
- Abilities need to be carefully balanced, and some are just too strong to be included. Allowing a unit to take multiple actions in a turn without any associated cost is one of the latter.

The above laundry list of complaints should not be understood as saying that *Symphony of War*[41] is a bad game, or that it is poorly implemented; it merely identifies things Dragon's Memoir can improve upon.

2 Game Design

2.1 Gameplay

2.1.1 Gameplay Loops

Prev – ToC – Next

> Ever tried. Ever failed. No matter. Try again. Fail Again. Fail Better.
>
> *Worstward Ho*, Samuel Beckett

tl;dr — The high-level loop consists of lead-in conversation, battle, conclusion, and intermission. A turn in battle has the player act first, then their enemies act automatically. Intermissions allow the player to prepare for subsequent battles. See figure 2.1

The game's core gameplay loop can be summarized as alternation between chapters, in which the game's plot advances, and intermissions, in which the player is allowed to do housekeeping in preparation for subsequent chapters.

Most chapters are divided into three parts: a lead-in conversation that advances the plot, a battle where the player is allowed tactical input, and a second conversation that concludes the chapter. Depending on its plot, a chapter may not involve fighting, in which case it might not have a battle and thus only include conversations.

Each battle has a given goal for the player to achieve, and takes place in turns split into two phases. During the first phase of each turn the player is allowed precise control over their units, having them act in whichever order they choose; afterwards, during the second phase, the game automatically controls the remaining units, often with the intent of hindering the player's progress. At the lowest level, a unit acts in two steps: first, they may move to a different position, and second, they may take a concrete action, such as fighting another, or healing them.

Intermissions are unstructured sections that represent downtime in the overarching plot. Not all chapters have an intermission following them, depending on the events that happen during the chapter.

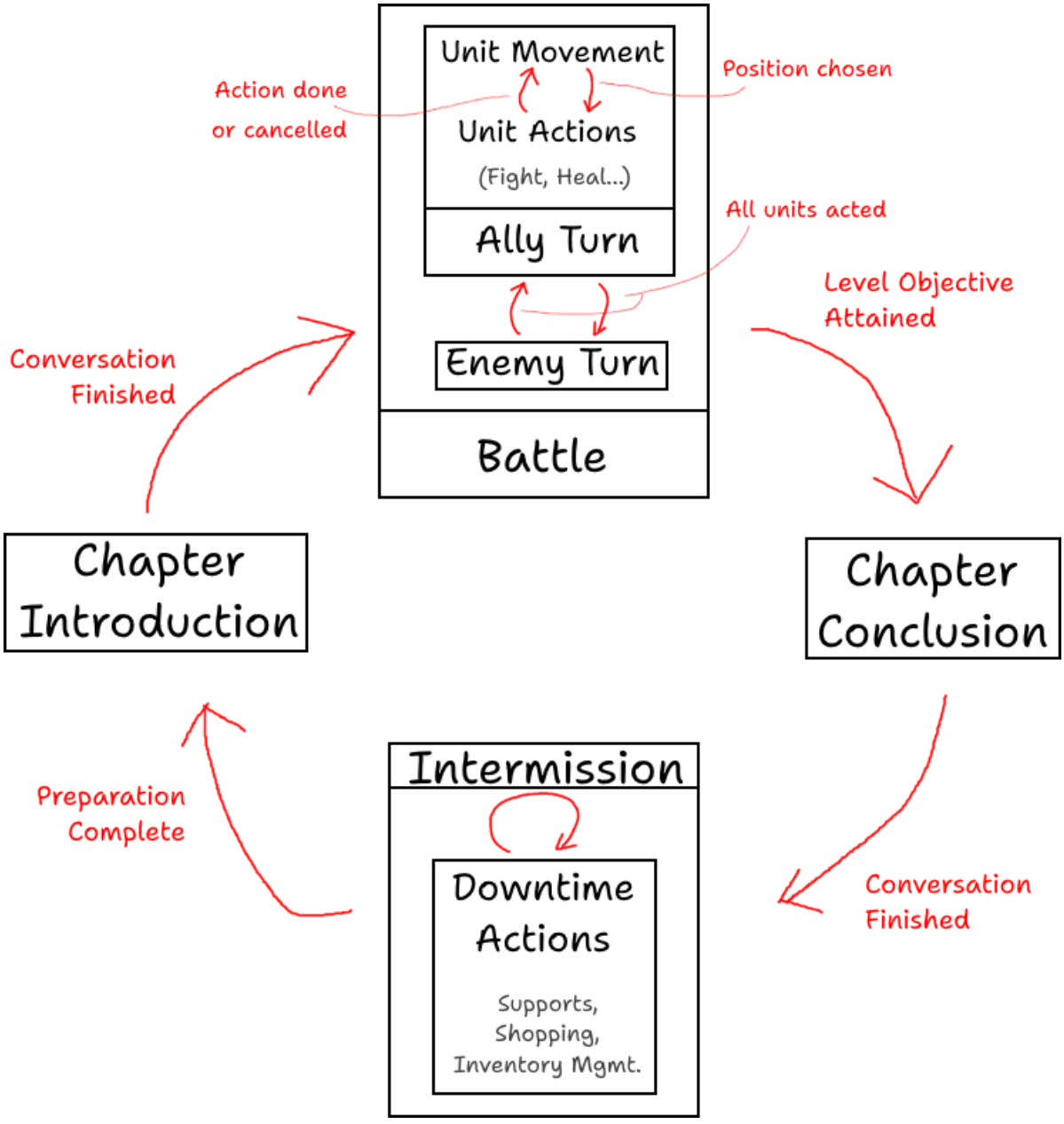

Figure 2.1: Overview of Dragon's Memoir nested gameplay loops. Red arrows represent transitions between states.

2.1.2 Game Mechanics

Prev – ToC – Next

> We do not study war because we love it, but because we hate it.
>
> *Mournful Tutelage*'s flavor text, in *Chroma: Bloom and Blight*

2.1.2.1 Unit-related

Each unit has a class. It determines which kinds of weapons they can use and whether they are weak to any particular kind of weapon. It also determines the base mobility a unit has, as described below.

Units gain experience as they fight in battle, and once enough is accumulated they "level up". These level ups increase the unit's stats and may teach them a new skill depending on their class once a high enough level is reached.. The chance a given stat increases also depends on the class: for instance, mages will have a higher chance of their magic stat increasing than an archer.

Each unit has eleven stats:

- Two offensive stats (one per damage type; "strength" and "magic") that increase the amount of damage the unit's attack deal.
- Two defensive stats (also one per damage type; "defense" and "resistance") that decrease the amount of damage taken.
- One stat ("hit points", often also called "health" or "HP") that determines how much damage the unit can take before falling in battle.
- Three stats ("speed", "skill" and "dexterity") that together determine the unit's hit, critical hit and dodge chances in combat, and whether the unit will try to attack multiple times in combat.
- The three pseudo-stats corresponding to the unit's hit, critical hit and dodge chances; their base values are determined by the previous three stats, but can be subsequently modified by weapons and skills (see below).
- One pseudo-stat ("mobility") that determines how far the unit moves in battle. Its base value is determined by the unit's class and status.

At the time of writing, permanent player-controllable character death is not implemented, although conversations do take into account potential character deaths. Characters "revive" on chapter end; this is sometimes referred to as "Casual Mode" in the *Fire Emblem* series.

A unit can have any amount of "skills", the effects of which can vary. Classes grant a unique skill (a "tactic") to units that belong to it; a unit changing class changes which tactic they have. Skills may also be learnt in other ways, as appropriate to the skill.

Some pairs of player-controlled units can build up "support" during battle; which pairs can depends on the character's personal backstories, attitude and tastes, all of which cannot be changed by the player. Once enough support is built up, the player can "confirm the support level" by watching a conversation unfold between the two units during an intermission. Doing so unlocks certain benefits, the most common of which being temporary stat increases when the two units are near each other in battle. Support levels can have additional conditions needed to unlock them, which can include reaching a given support level with a different pair of units, or *not* having reached that level, effectively making certain support levels exclusive.

2.1.2.2 Item-related

Units can carry an arbitrary amount of items, which contain up to one equipped weapon and staff. A unit will use their equipped weapon when told to fight (or when another unit fights against them), and their equipped staff when told to heal another unit. Not all player-controlled units can equip staves; which ones can depend on the character themself.

Each item has a "color", which modifies the damage their wielders deal and take in combat. Only the equipped weapon, if any, modifies this damage; a generic no-modification color is used in the event a unit without an equipped weapon engages in combat.

Items have up to one skill that is active for as long as the item remains in the unit's inventory. Weapons and staves may also have one skill that is active when they are equipped.

Weapons have a few additional characteristics:

- **Each weapon belongs to a given "weapon type"** that determines which classes can equip it. There are ten weapon types, five of which are "physical" and the rest, "magical". This distinction determines which stats are used in combat, as detailed before.

- They are either breakable or unbreakable. Breakable weapons can have multiple "weapon steps", which determine the base damage they deal in combat; unbreakable weapons can only have one.

- A weapon step specifies how many attacks the wielder will attempt each time they make an attack in combat (most commonly only one) and the base damage of those attacks, which need not be deterministic.

- They have a base hit chance, which, as its name implies, determines the chance its wielder's attacks will land in combat. A weapon whose base hit chance is 100% will never miss, regardless of the target's dodge chance.

- They have a fixed attack range that determines how distant the wielder can make attacks. Melee weapons have a range of one, and most ranged weapons have a range of at most two.

Staves follow the same rules as weapons, except that there is only one "staff type", **staves heal instead of dealing damage**, and staves never miss in their healing.

Unused items in a unit's inventory may be stashed away during intermissions to be retrieved later or given to another unit. Some items cannot be put away in this manner, such as those that represent a physical part of a creature.

2.1.2.3 Battle-related

Each battle has a win condition (usually defeating all enemies) **and a lose condition** (usually having a particular unit fall in combat). Achieving the win condition ends the battle and causes the game's plot to advance. Achieving the lose condition results in what could be considered a "game over" and forces the player to replay and win the battle to continue the game. The player is told what the win condition and lose conditions for a battle are. At the time of writing, lose conditions aren't fully implemented; it is expected that the implementation will require modifying the conversation parser to account for defeat.

Movement in battle is square-grid-based. How many squares a unit can cover in a given turn is determined by their stats and the terrain those squares represent; **some terrain can slow certain units down**, or prevent their movement altogether. Units move orthogonally along the grid (that is, along grid lines and not diagonally); effects that measure distance do so using the Manhattan distance (positions that are diagonally adjacent are at distance two of each other).

Only one unit can be in any given square. The player moving a unit onto a square occupied by another of their units allows them to displace the latter to an adjacent square.

After choosing to move a unit (or to not do so), the player may instruct them to change their equipped items, and then to take an action among the following:

- Fight another unit, possibly without intent to fully defeat them. This will have the two units engage in combat, and afterwards may grant experience to the involved units. This will also make supports that involve the attacker and defender progress.

- Heal one of the unit's allies (or themself), recovering their lost health points.
- Talk to a nearby unit, when a conversation is available.
- Activate any skill of theirs that can be activated at this point.

The player is informed of the (potential) **consequences of their actions** (eg. as in figure 2.2), and given a chance to confirm their choice or explore alternative options. Movement can be fully undone before an action is confirmed.

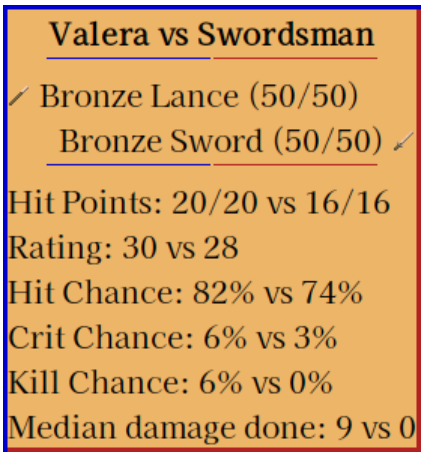

Figure 2.2: Prospective information about a fight between two units. Note that *median* damage is displayed, not *mean* damage, as the former is more representative of the fight's outcome.

The player may also instruct all of their units that are yet to act this turn to do nothing, ending the player's phase this turn.

When two units fight, they attempt to deal damage to each other. First, whoever initiates combat (the "attacker") makes an attack against the other unit (the "defender"). Then, if the defender's weapon's range allows them to attack the attacker, they try to do so. Finally, if the difference in speed stats is large enough, the attacker or the defender will attempt a second attack.

Each attack in a fight has three possible outcomes: miss (the attack deals no damage), hit (the attack lands, and deals damage), and critical hit (the attack lands and deals extra damage). The stats and weapons of the involved units determine the likelihood of each of these outcomes.

Skills can modify a number of aspects of a fight, including who each attack is made against, and adding extra attacks beyond the usual ones.

2.1.2.4 Intermission-related

Between some chapters the player is allowed to explore their base camp and prepare for future battles.

Those preparations include **buying and selling equipment** for their units using money obtained from story events or looted in battle. Which items are available for purchase change from chapter to chapter and shopping may not be available between some chapters.

Intermissions also allow the player to **"cash in" unlocked support levels**. Each support explores small side stories between two characters and once a level is unlocked grants those units some minor boons, notably including temporary stat increases in battle.

Finally, the player is allowed to **play some mini-games** once unlocked. Much like supports, they explore side stories but grant more "tangible" boons in the form of items.

Unlike in *Awakening*[36] and *Symphony of War*,[41] **Dragon's Memoir currently has no way to "train" units out of combat**. This is a downtime action that could be considered, but the problem it solves is arguably better solved by carefully balancing battles.

Once the player is satisfied with their preparation they can "depart", moving on to the next chapter of the story.

2.1.3 User Interface and Input

Prev – ToC – Next

> Novice shopkeeps spend hours deciding how best to display their wares. Veterans focus on portability.
>
> *Batterhorn*'s flavor text, in *Magic: The Gathering*

> **tl;dr** — A window of at least 800×600 pixels is needed. Text uses serif fonts. Tooltips are often used. Both keyboard and mouse can be used for player input.

The game requires a window of 800-by-600 pixels in size in order to properly display its content; this represents a minimum window size, not a maximum size—the window may be enlarged as desired by the player. Currently the option to display the game in full-screen or in a borderless window is not available (a requirement for those options is an in-game exit button, which, just as a proper options menu, is yet to be implemented).

UI elements that the player may interact with are displayed as rectangles with rounded corners when alone or when displayed in a grid. Elsewhere, interactable elements that represent options that are mutually exclusive (for example, when the player is about to choose an action for their units to take, as in figure 2.3) are displayed using a radial menu.

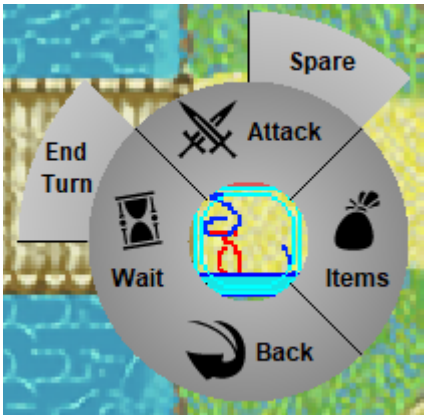

Figure 2.3: Radial menus show exclusive options; here, the possible actions a unit can take.

Text display primarily uses serif fonts so as to evoke a feeling of reading a book. The game's two main fonts are visually quite similar: Libre Baskerville is used to display speech in conversations and Kaisei Tokumin is used to display most UI elements.

Battles are displayed using 2D graphics (see figure 2.4); an image representing terrain serves as the backdrop atop which units are drawn. Alongside each unit a bar is drawn that visually represents the amount of health the unit has; this bar is colored in accordance to the unit's side (blue for allies, green for neutral units and red for enemies) and its color changes as it depletes. Additionally any number of "sprites" is drawn as needed, for example, to indicate skills triggering or support being attained. Combat is currently displayed using sprites, although there are plans to have a separate display system for it.

Most textual information is displayed only when needed using tooltips atop or beside certain UI elements. This includes information about units' stats in battle (figure 2.5), which would otherwise be overwhelming.

Conversations use either a static image or a level as their backdrop, both of which can change throughout it as the conversation's plot advances. The face of up to four characters can be shown at any given time during a conversation representing who is taking part in it; these characters face towards the center of the scene. Conversation text is shown one letter at a time so as to simulate actual speech.

Intermissions use the same display system as battles, using separate scenes for each of the possible "downtime" activities.

In terms of user input, the player may use either keyboard or mouse to interact with the game. All options available can be chosen with either input method. Pressing the space bar and left-clicking are consistently used as the "accept" option where appropriate, and pressing left shift or right-clicking are consistently used as the "cancel" option; more detailed controls are explained in the attached extra documentation.

"Joystick" input is planned but not currently implemented; it will likewise be usable to take any and all possible options, just as keyboard and mouse each on their own are.

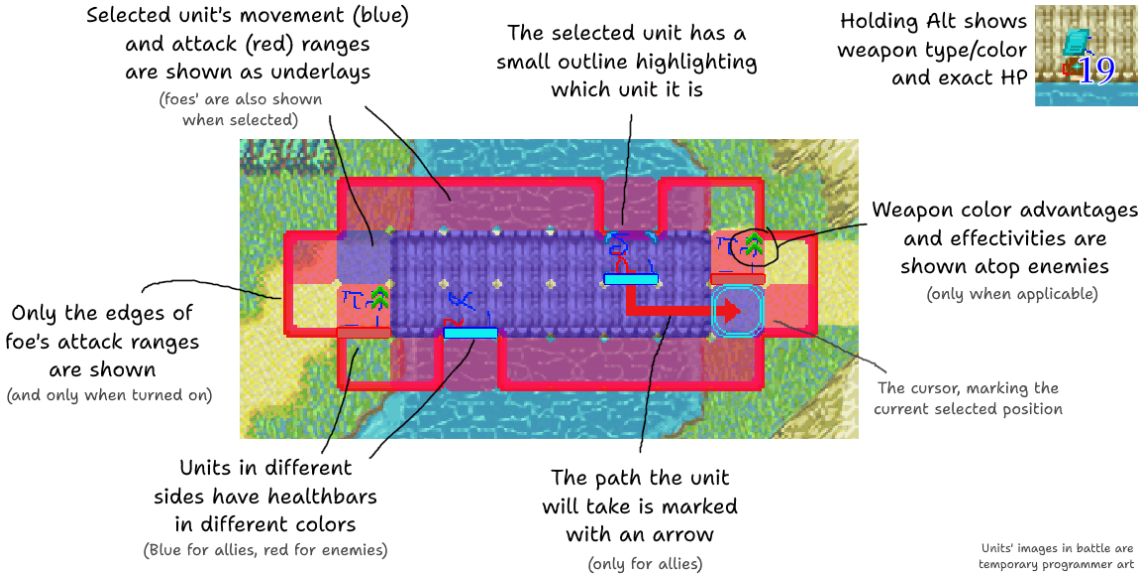

Figure 2.4: Some information displayed in battle. Most information is only shown when needed; healthbars are always shown in battle and enemy range is turned off by default.

Figure 2.5: Overhead labels display basic information about hovered units. Holding Alt shows *all* information instead.

2.1.4 Runtime Requirements

Everything is HUUUUGE

Software Disenchantment, Niki Tonsky

Prev – ToC – Next

tl;dr — Java 24 or newer. 512 MB of RAM. 35 MB of storage.

A Java runtime environment ("JRE") of version 24 or newer* is required to run the game. The game requires no internet connection once it and the JRE are installed; Dragon's Memoir on its own requires no installation beyond being downloaded since it is distributed as a runnable .jar file.

At runtime the game may use up to 200 MB of RAM, thus at least 512 MB of system RAM is required. Dragon's Memoir tries to load resources only as needed as opposed to keeping in memory all assets at all times; most of the RAM used by the game is taken up by the JRE.

In terms of persistent storage, the game requires around 14 MB for the .jar file itself, and about 20 MB more to store save data and logs. This latter space is used to preserve *all* past save points (to allow retrying chapters as desired) and *all* previous runtime logs (which are stored in a compressed format; this is to aid in debugging). Dragon's Memoir will try to not "litter": saves and logs are stored in subfolders of the "installation" folder.

It should be noted that the game has only been tested under Windows 10, but in accordance to Java's philosophy of "write once, run anywhere", the game is expected to be able to run anywhere its required JRE can run.

Dragon's Memoir intended frame rate is 60 frames per second. The game achieves this by offloading long-running tasks, notably disk access, outside of the main UI thread.

*Development usually targets the latest released version of Java, regardless of long-term support. To preserve forward-compatibility, preview features are not used in code.

2.2 Worldbuilding

Prev – ToC – Next

> "I'm a storyteller," Wit said, with a flip of his fingers. "I have the right to redefine words."
>
> *Wind and Truth*, Brandon Sanderson

tl;dr — The game takes place in "Enceladus", where technology is lacking. Religion is based on the Dragon and the Drakainas. Magic is limited and all spells have verbal and material components.

The world Dragon's Memoir takes place in, called Enceladus, is loosely based on that of *Fire Emblem Awakening*[36] and *Chroma: Bloom and Blight*.[26] It features five countries spanning two continents and its history spans five and a half millennia.

In-world technology has hardly progressed at all throughout its history and several discoveries and attempts at innovation have been met with strong resistance: steel was only discovered recently and was met with significant backlash, chemistry has barely been explored, and biology studies are uncommon and unrefined.

Religion in Enceladus is limited to adoration of the Dragon, who is almost universally believed to have created the planet and life within it; the Disciple, who some consider to be the creator of magic; and the five Drakainas*†, who are considered priestesses of the Dragon, and their aides, called Respites. Only one or two Drakainas have lived in Enceladus at the same time, leading some people to believe that "consecutive" Drakainas follow a mother-daughter relationship, a statement that the Drakainas and Respites have refused to confirm.

In terms of design, the magic system used in Dragon's Memoir is created following Brandon Sanderson's laws of magic.[10] Magic in Enceladus is limited in scope (primarily by its practitioners' lack of understanding of it) but has clear usage guidelines: an incantation is needed to "ignite" a spell's "fuel", and each spell requires a different pair of incantation and fuel. This means that, unlike in the *Fire Emblem* series, spellcasters do not use spell books in order to cast their spells; this also helps explain why magic "weapons" break—their users simply run out of materials to use them. Item colors are also related to this magic system in a way that Enceladus' inhabitants don't fully understand; a subplot within the game explores this further.

Separately from the above, a constructed language has also been designed for the game, although it is only used in-game in proper nouns since using it for all in-game text is obviously unreasonable (but do see *Tunic*[42] and *Chants of Sennaar*[25] for games that do this).

*Drakainas are somewhat related to manaketes[11] from the *Fire Emblem* series: both have long lifespans and feature pointed, elf-like ears, but only manaketes can transform into dragons.

†The word "Drakaina" comes from the Greek mythological figure of the same name; in-world, it also means "small female dragon".

3 Development Plan

3.1 Software and Tooling*

A bad workman always blames his tools.

Proverb

Prev – ToC – Next

tl;dr — Dependencies: Guava, Lombok, Tinylog, and YamlBeans.
Tools: Eclipse, Paint.net, Vim, Tiddlywiki, and Google Docs

Dragon's Memoir is written in Java and uses Swing as its front-end framework. Its software dependencies are the following external libraries:

- Guava (Apache License 2.0): Contains generic utilities, notably new collection-like types including multimaps, bidirectional maps, and networks.
- Project Lombok (MIT License and others): Code generation, particularly auto-generation of null checks and accessors. Only required during compilation, not at runtime.
- Tinylog (Apache License 2.0): As its name implies, it's used for logging.
- YamlBeans (MIT License): Serialization to and from YAML, the format used for save data.

Additionally, Eclipse's nullity and resource-owning annotations are used at compile time, and two "home-brew" libraries are used, one containing miscellaneous utilities and another providing a more fluent API to Swing components.

The following tools are or have been used for development and asset creation:

- IDE: Eclipse, together with SonarQube for IDE (formerly called SonarLint) and Spotbugs for linting.
- Image editor: Paint.net.
- Text editor: Vim through Cygwin. Formerly Notepad++.

*External links in this section have been last accessed on 2025–04–28

- Version control: Sourcetree as a Git for windows front-end.
- Language construction: PolyGlot and Lexique Pro.
- Documentation: Tiddlywiki and the Google Docs suite.

3.1.1 Use of Generative Artificial Intelligence

Generative AI is a parasitic cancer.

FREYA HOLMÉR
in the video of the same title.

Prev – ToC – Next

tl;dr — No.

Dragon's Memoir is a project developed by humans, to be consumed by humans. The use of generative artificial "intelligence" is not contemplated neither for conversation writing ("if you can't be bothered to write it, I can't be bothered to read it") nor for coding (one of the project's main goals is learning to program and "vibe coding" defeats the point).

The only aspect of the game where using generative AI would be somewhat reasonable is in creating art. Drawing conversation backgrounds would not be difficult, and with some manual editing it would also be possible to generate level backdrops. Some neural networks are already capable of generating anime-like faces,[8,9] which unfortunately is not enough for Dragon's Memoir since it requires almost-full-body images like in *Dark Deity*.[28] Training an *ad hoc* model for this purpose is *almost* feasible, if not for its significant drawbacks:

- A training set would have to be elaborated; an option for the training images would be *Fire Emblem Heroes*[38]-like drawings which could be cleaned versions of other existing art. Despite certain AI companies having no qualms using copyrighted materials[1] to train their models, for this project this is a nonstarter.
- Training itself requires resources that are better spent elsewhere or not consumed altogether, notably electricity.
- Legislation of the copyright of generated images is still undecided, which could harm or prevent copyright enforcement in the future.

All of this means that generative AI is not and will not be used in Dragon's Memoir.

3.2 Budget and Timing Estimates*

Prev – ToC – Next

> It always takes longer than you expect, even when you take into account Hofstadter's Law.
>
> Hofstadter's Law

tl;dr — 100 000 $\$_{90\,000\,€}$ and 12–15 months.

In terms of timing, Dragon's Memoir could be reasonably finished in twelve to fifteen months of full-time work, expecting most of that time to be taken by art production rather than coding.

Estimating the amount of money required to finish the project is a more difficult task. Many "indie" games rely on Kickstarter to obtain the funding they need; as that could be way to acquire funding for Dragon's Memoir, an estimation of its budget could be based on other games' Kickstarter projects.

It might seem reasonable to base Dragon's Memoir estimated budget on *Dark Deity*'s[28] Kickstarter project, as that game is one of the most similar to Dragon's Memoir both in terms of development and scope. However, its initial goal of 12 000 $\$_{10\,800\,€}$ is undeniably too low and would be insufficient for Dragon's Memoir.

A more realistic picture is painted by *Fell Seal*'s[34] Kickstarter project. By extrapolating the figures contained therein, assuming the 40 000 $\$_{36\,000\,€}$ goal corresponds to 40% of its total budget, we obtain the figures contained in table 3.1.

Expense	Amount	
Art	54 000 $	48 600 €
Music etc.	28 000 $	25 200 €
Marketing	10 000 $	9000 €
Miscellaneous	8000 $	7200 €
Total	100 000 $	90 000 €

Table 3.1: Estimated budget. The "Miscellaneous" category encompasses potential publishing fees, any taxes and serves as a small emergency buffer.

The budget in that table does not include code development (programming et cetera) costs as it assumes coding is done in-house. While that model would also be appropriate for Dragon's Memoir, it might be worth estimating how much would be needed to hire someone to program the game.

If just one person is required to work full-time on coding for twelve months, and assuming a salary of 2000 $\$_{1800\,€}$ per month, about 24 000 $\$_{21\,600\,€}$ more would be needed, for a new total budget of 124 000 $\$_{111\,600\,€}$.

*External links in this section have been last accessed on 2025–06–04. USD-EUR conversion rates are current on 2025–05–13. Converted figures are approximated.

3.3 Development Methodology

Prev – ToC – Next

> **tl;dr** — Primarily feature- and chapter-driven. Refactors and redesigns are interleaved with feature and chapter additions. A Tiddlywiki wiki serves as game design document.

Dragon's Memoir's development has been haphazard and unstructured, but in broad strokes, it could be said that it has been primarily feature-driven, with these features being added as required by the chapters being implemented. More concretely, development is most often split pursuant the implementation of a story chapter each "prototype"*. Bug fixes are integrated as soon as they are checked for correctness, following "git-flow"[2] to some extent.

Besides "forward-progress" versions, refactors are also made from time to time in order to simplify the implementation so as to reduce code complexity and ease the introduction of new features. Similarly, the implementation of design-wise reworked features (eg. skills being reassigned between unit classes, or changed entirely) and the formalization of previously *ad hoc* code (eg. the turn cycle) are also integrated as though they were normal code refactors.

The steps followed to implement a chapter from scratch are:

1. *Design* — Decide the main chapter plot and setting.
2. *Level Art* — Create the level backdrop, if needed. This includes both art and level data (size and the tile at each location).
3. *Character Design* — Design the playable characters introduced this chapter: decide their name, write a short blurb explaining their backstory (this backstory may be referenced in support conversations with this unit), choose a class for them, and draw the images they'll use in conversations and in battle.
4. *Class Design* — If the new characters have new classes, create them: decide what kinds of weapons they can use, what weakness they have (eg. "infantry" or "cavalry"), and what skills the class teaches (including the tactic). Also draw the art used to display "generic" units of this class in battle as well as any skill icons needed.
5. *Writing* — Write the chapter's conversations; their text can be refined later and/or in parallel if needed.

*No real prototypes have been made, as the game is not ready for public testing. Instead, a "prototype" is considered done once a particular feature set is implemented, typically once a new chapter has been implemented and tested.

6. *Conversation Parser Implementation* — Implement and expose to the conversation parser any newly required actions to be taken as the conversation takes place. This includes moving units along the backdrop and preparing battles.

7. *Other Implementation* — If needed, implement any required features needed in battle or otherwise that can't or shouldn't be exposed to the conversation parser (eg. actions that trigger at a particular turn in battle, or aspects related to user interface or unit AI).

8. *Integration* — "Inform" the "game engine" of the existence of the new chapter and its content and "hook them up" to the current implementation. This usually requires updating `module-info.java` and other files in the `META-INF/` folder to register new skills, and telling implementation classes to expose the new items, characters and classes to the engine.

9. *Gameplay testing* — Play through the chapter to test if everything works as intended.

10. *Debugging* — Make changes (to the conversations, the implementation, or even the design) as needed to address the issues found during testing, if any.

Newly designed content (chapters, characters, classes, skills...) is also kept track of in a Tiddlywiki wiki (Dragon's Memoir *de facto* game design document) for ease of (cross-)reference. This wiki also stores a more detailed explanation of the mechanics described in Section 2.1.2.

4 Implementation and Testing

> "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?"
>
> —
>
> *Passages from the Life of a Philosopher*,
> Charles Babbage

tl;dr — Model-view-controller for battle UI; factory methods, service locators, and lazy initialization elsewhere. See table 4.1. Only behavioral tests are done; reliance on logging and other custom debugging tools.

4.1 Software Architecture

The main software architectural pattern used for player interaction in battle is a variant of model-view-controller in which the view and the controller are fused together. This union stems from the fact that both player input and game display are handled by event listeners in `JPanels`. Conversation and intermission display and UI handling is likewise combined into one class per "task".

Other auxiliary design patters are used as needed to implement the game's features:

- Whenever possible, classes are turned into singletons, or made uninstantiable at all (cf. the zero-one-infinity rule).
- Factory methods are often preferred for public APIs instead of exposing constructors. For some complex entities, particularly items and unit classes, builders are used.
- A variant of the prototype design pattern is used to separate items and item *stacks*: a unit may carry any amount of item stacks, each of which references a given prototype item.
- Most state-changing events in the game are implemented as commands over the battle handler. This also includes some actions during conversations, such as moving units and "summoning" units into battle.

- To a minor extent, proxy objects are used to avoid eager initialization of certain objects whose whole representation is not needed. Generally lazy initialization is preferred over the use of these objects.

Most of the game's code runs in a single thread (the AWT event queue), with some lazily loaded objects being loaded ahead of time in separate threads. For these concurrent accesses neither `volatile` fields nor double-checked locking are used*; atomic objects and locks are used instead.

Service locators (using Java's own `ServiceLoader` class) handle the registration of the game's content. This avoids hard-coding the game's characters, classes, items etc., possibly allowing for the creation of community modifications ("mods") in the future, at the cost of introducing a centralized registry to acquire instances of them.

4.2 Implementation Details

Dragon's Memoir's source code proper is split into separate packages contained in a single module,[7] as described in table 4.1.

Non-code assets (conversations, level data, images, and fonts) are stored in separate folders as appropriate for each asset. Similarly, localized text other than conversations is split into several `.properties` files each with the same name and location as the class that uses them.

Where reasonable, disk access is done in separate threads so as to not block the UI thread. This includes reading images and saved data, and writing saved data to disk.

Performance measurements have shown that rendering translucent images (that is, images with pixels that are neither fully opaque nor fully transparent) in Swing can take up a significant portion of frame rendering time (upwards of 10 milliseconds) since they are drawn in software†. In order to not incur in this performance penalty, enemy attack ranges in Dragon's Memoir are indicated by only highlighting the edges of the threatened area; this contrasts with several other games (*Fire Emblem Awakening*,[36] *Dark Deity*,[28] and *Symphony of War*,[41] to name a few) in which attack ranges are shown using translucent underlaid grids.

Dragon's Memoir also differs from *Dark Deity*[28] and *Symphony of War*[41] in its attack range calculation. Whereas in the latter two games attack ranges are calculated exclusively on demand, Dragon's Memoir recalculates attack ranges eagerly, ensuring that the ranges shown to the player are accurate; this allows skills and other game mechanics to refer to these ranges

*Those idioms don't work[5] and should not be used.[4]

†This is a Swing/AWT limitation. A possible solution would be to switch to OpenGL or similar back-end, at the cost of a huge refactor of all UI and input handling.

(root)	Main class, component registries, global game data
`ai`	AI interface and path-finding utilities.
`chapters`	Chapter interface.
`combat`	Combat simulations, damage data, and experience calculations
`convos`	Conversation handling.
`display`	UI implementations.
`….level`	Battle UI.
`….units`	Unit-handling UI.
`impl`	Item, character, chapter etc. implementations.
`….skill`	Skill implementations.
`items`	Item and item stack interfaces and implementations.
`level`	Battle system implementation.
`minigames`[a]	Minigames during intermissions.
`pers`	Unit interfaces and implementations. Unit supports.
`….cpers`	Unit classes and skills interfaces.
`save`	Serialization and save handling.
`utils`	Miscellaneous utilities.

Table 4.1: Dragon's Memoir code packages. Package names are relative to `ivaniesta14.pfe`[b] or to the previous package when beginning with an ellipsis.

[a] This package and children thereof.
[b] "PFE" was Dragon's Memoir in-development name. Its origins are unclear: "FE" does not stand for "Fire Emblem", and original design documents use "FI" in place of "FE".

without calculating them themselves. Performance-wise this only takes between two and four milliseconds per recalculation, not including the time required to draw the overlay (which is done asynchronously in separate threads).

The calculation of unit movement ranges in Dragon's Memoir relies on Dijkstra's algorithm to determine all locations a unit can reach from their position in battle given their current mobility: all positions that are at a distance equal or lower to their mobility, taking into account tile costs, are considered part of the unit's movement range. Once a unit's movement range is calculated determining their attack range is easy: simply determine all positions that are at a distance equal to the weapon's attack range (or equal to any of the weapon's attack ranges, if it has more than one); this calculation is done in linear time on each of the amount of positions the unit can move to, the number of weapon ranges, and the weapon range itself (larger weapon ranges take linearly longer to explore).

Unit AI and certain minor features also make use of the A* ("A-star") path-finding algorithm to compute the least-expensive path between two positions. The heuristic used for these com-

putations uses a "shifted down" Manhattan distance* (two is subtracted from the distance, to a minimum of zero); directly using the Manhattan distance results in sub-optimal paths in the presence of zero-cost edges that don't immediately decrease distance.

4.3 Testing and Debugging

Dragon's Memoir's implementation doesn't lend itself well to unit testing since most of the content is registered via `ServiceLoader` and mocking it would require significant architectural changes to the code. Instead, testing relies on integration (behavioral) testing, checking whether the game does what it's supposed to from the player's perspective rather than whether each component of the game works in isolation. More thorough testing will be performed once the game is ready for an alpha or beta release.

Runtime log analysis is the technique most commonly used during debugging; the game's frequent usage of logging is another reason why unit-testing Dragon's Memoir is not doable—mocking or disabling Tinylog in a test environment is not feasible. The game's logs come in two "flavors": standard output console, serving mostly as sanity checks, and log files, which store plenty of information about what happens in the game.

Contained within these latter logs are also timing measurements of code paths that have been identified as potential performance bottlenecks. These timed spots include conversation and level parsing, in-battle unit movement and attack range calculations and drawing, and overall frame drawing time (which is only logged when the duration of a frame exceeds a certain threshold).

Measuring frame drawing time has identified two surprisingly expensive features: first, as mentioned before, Swing handles semi-transparent image drawing in software; and second, text outlining requires drawing raw shapes and is also fairly expensive. These features are used sparingly so as to not reduce frame rate.

As a last measure to ease debugging, the game exposes some of its internal state to be queried when needed via the standard input console[†] and certain key combinations.

*Both the Manhattan distance and the shifted down Manhattan distance are inadmissible heuristics (they overestimate path costs) in the presence of many zero-cost edges. Using the latter is a compromise between being able to handle a few zero-cost edges and having a reasonable heuristic when there are none.

[†]Since all console input in Java is blocking, this "console listener" can cause the game's process to not terminate when instructed, which is why it is disabled by default.

5 Conclusions

> Every story, even a faerie tale, comes to an end.
>
> *Declaration of Naught*'s flavor text, in *Magic: The Gathering*

Despite more getting close to a decade of development time Dragon's Memoir is quite far from being done. Currently only ten out of the possibly fifteen chapters of the first major arc are fully implemented and most of the support conversations are yet to be written. Most of the game's art is "programmer art" not suitable for public release, and there's no sound at all.

That said, Dragon's Memoir's engine is mostly complete, meaning that future development can be streamlined into creating more of the game's content.

All in all, Dragon's Memoir has succeeded at its original goals and shows promise in what it can grow into.

Articles Referenced

[1] *AI Art and its Impact on Artists*, Harry H. Jiang, Lawren Brown, Jessica Cheng et al., Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society, pp. 363–374, 2023–10–08.

https://doi.org/10.1145/3600211.3604681 LA 2025–04–28

[Referenced in page 21]

[2] *A successfull Git branching model*, Vincent Driesen, 2010–01–05.

https://nvie.com/posts/a-successful-git-branching-model LA 2025–05–14

[Referenced in page 23]

[3] *Dark Deity is an ambitious strategy-RPG from a rookie team that's out now*, Jason Wilson, GamesBeat, 2021–06–15.

https://venturebeat.com/games/dark-deity-is-an-ambitious-strategy-rpg-from-a-rookie-team-thats-out-now LA 2025–02–28

[Referenced in page 4]

[4] *Double-checked locking should not be used*, Sonar Rules.

https://rules.sonarsource.com/java/RSPEC-2168 LA 2025–05–18

[Referenced in page 26]

[5] *Java double checked locking*, answer 1625180 by Yishai to question 1625118 by Jim, Stack Overflow.

https://ao.ngn.tf/questions/1625118/#1625180 LA 2025–05–18 Stack Overflow original

[Referenced in page 26]

[6] *Fire Emblem Awakening podría haber sido el último*, Fernando Mateus, Hobby Consolas, 2013–05–26.

https://www.hobbyconsolas.com/noticias/fire-emblem-awakening-podria-haber-sido-ultimo-52742 LA 2025–02–28

[Referenced in page 5]

[7] *JEP 261: Module System*, Alan Bateman, Alex Buckley, Jonathan Gibbons, Mark Reinhold, OpenJDK.

https://openjdk.org/jeps/261 LA 2025–05–18

[Referenced in page 26]

[8] *Making Anime Faces With StyleGAN*, Gwern Branwen, 2022–10–19.

https://gwern.net/face LA 2025–04–28

[Referenced in page 21]

[9] *This Waifu Does Not Exist*, Gwern Branwen, 2020–01–20.

https://gwern.net/twdne LA 2025–04–28

[Referenced in page 21]

[10] *What Are Sanderson's Laws Of Magic?*, Brandon Sanderson[?], ca. 2018[?].

https://faq.brandonsanderson.com/knowledge-base/what-are-sandersons-laws-of-magic/ LA 2025–04–26

[Referenced in page 19]

Wiki Articles Referenced

[11] *Manakete*, Fire Emblem Wiki.

https://fireemblemwiki.org/wiki/Manakete LA 2025–04–26 Rev 2024–09–28 at 00:52

[Referenced in page 19]

[12] *Might*, Fire Emblem Wiki.

https://fireemblemwiki.org/wiki/Might LA 2025–03–04 Rev 2025–09–01 at 18:45

[Referenced in page 4]

[13] *My Castle*, Fire Emblem Wiki.

https://fireemblemwiki.org/wiki/My_Castle LA 2025–03–01 Rev 2024–10–10 at 19:37

[Referenced in page 5]

[14] *Second Wave*, XCOM Wiki

https://antifandom.com/xcom/wiki/Second_Wave LA 2025–02–28 Fandom original Rev 2023–08–20 at 04:14

[Referenced in page 7]

[15] *Time Travel*, Fire Emblem Fandom Wiki

https://antifandom.com/fireemblem/wiki/Time_Travel LA 2025–02–28 Fandom original Rev 2024–11–03 at 16:35

[Referenced in page 5]

[16] *Weapon level*, Fire Emblem Wiki.

https://fireemblemwiki.org/wiki/Weapon_level LA 2025–03–04 Rev 2025–02–21 at 17:58

[Referenced in page 4]

[17] *Weapon triangle*, Fire Emblem Wiki.

https://fireemblemwiki.org/wiki/Weapon_triangle LA 2025–02–24 Rev 2025–02–13 at 04:04

[Referenced in page 4]

[18] *World Map*, Fire Emblem Wiki.

https://fireemblemwiki.org/wiki/World_map LA 2025–03–01 Rev 2025–02–08 at 05:16

[Referenced in page 5]

Games Referenced

[19] *Advance Wars* (2001), developed by Intelligent Systems, published by Nintendo.

Wiki: `https://warswiki.org/wiki/Advance_Wars` LA 2025–03–11

[Referenced in page 6]

[20] *Ara Fell* (2016), developed by Stegosoft Games, published by Dangen Entertainment.

Official page: `https://stegosoftgames.com/Games/AraFell` LA 2025–02–21

[Referenced in page 3]

[21] *Baldur's Gate* (1998), developed by BioWare, published by Black Isle Studios, Interplay Entertainment, and Sega.

Wiki: `https://antifandom.com/baldursgate` LA 2025–02–19 Fandom Original

[Referenced in page 3]

[22] *Baldur's Gate II: Shadows of Amn* (2000), developed by BioWare, published by Black Isle Studios, and Interplay Entertainment.

Wiki: `https://antifandom.com/baldursgate` LA 2025–02–19 Fandom Original

[Referenced in page 3]

[23] *Baldur's Gate 3* (2023), developed and published by Larian Studios.

Official page: `https://baldursgate3.game` LA 2025–02–19

Wiki: `https://bg3.wiki` LA 2025–02–19

[Referenced in page 3]

[24] *Brogue: Community Edition* (2023), forked from the original by tmewett.

Official repository: `https://github.com/tmewett/BrogueCE` LA 2025–02–20

Wiki: `https://antifandom.com/brogue` LA 2025–02–20 Fandom Original

[Referenced in page 3]

[25] *Chants of Sennaar* (2023), developed by Rundisc, published by Focus Entertainment.

Official page: `https://www.rundisc.io/chants-of-sennaar/` LA 2025–04–26

Wiki: `https://chantsofsennaar.miraheze.org/` LA 2025–04–26

[Referenced in page 19]

[26] *Chroma: Bloom and Blight* (2021), developed by Clarity Games, published by Philipp Baumgart, and WhisperGames Interplay Entertainment.

Wiki: `https://antifandom.com/chroma-bloom-and-blight` LA 2025–04–26 Fandom Original

[Referenced in pages d and 19]

[27] *Crypt of the Necrodancer* (2015), developed by Brace Yourself Games, published by Brace Yourself Games, Klei Entertainment, and Spike Chunsoft.

Official page: `https://braceyourselfgames.com/crypt-of-the-necrodancer` LA 2025–02–20

Wiki: `https://necrodancer.miraheze.org` LA 2025–02–20

[Referenced in page 3]

[28] *Dark Deity* (2021), developed by Sword & Axe LLC, published by indie.io.

Official page: `https://darkdeitygame.com` LA 2025–02–24

Wiki: `https://darkdeity.wiki.gg` LA 2025–02–24

[Referenced in pages 2, 4, 21, 22, and 26]

[29] *Divinity: Original Sin* (2014), developed by Larian Studios, published by Larian Studios, Focus Home Interactive, and Spike Chunsoft.

Official page: `http://www.divinityoriginalsin.com` LA 2025–02–19 (**HTTP only**)

Wiki: `https://divinityoriginalsin.wiki.fextralife.com` LA 2025–02–19

[Referenced in page 3]

[30] *Divinity: Original Sin 2* (2017), developed by Larian Studios, published by Larian Studios, and Bandai Namco Entertainment.

Official page: `https://divinity.game` LA 2025–02–19

Wiki: `https://divinityoriginalsin2.wiki.fextralife.com` LA 2025–02–19

[Referenced in page 3]

[31] *Dota 2* (2013), developed and published by Valve.

Official page: http://www.dota2.com LA 2025–02–21

Wiki: https://liquipedia.net/dota2 LA 2025–02–21

[Referenced in page 3]

[32] *Dream Tactics* (2024), developed by Spectra Entertainment Inc., published by indie.io.

Official page: https://www.playdreamtactics.com LA 2025–03–01

Wiki: https://dreamtactics.wiki.gg LA 2025–03–01

[Referenced in page 5]

[33] *Fae Tactics* (2020), developed by Endless Fluff Games, published by Humble Games.

Official page (from the publisher): https://www.humblegames.com/games/faetactics LA 2025–03–01

Wiki: https://antifandom.com/fae-tactics LA 2025–03–01 Fandom Original

[Referenced in page 5]

[34] *Fell Seal: Arbiter's Mark* (2019), developed by 6 Eyes Studio, published by 1C Entertainment.

Snapshot of the official page: https://web.archive.org/web/20221206120714/https://www.fellseal.com LA 2025–03–11 Arch 2022–12–06

Wiki: https://antifandom.com/fellseal LA 2025–03–11 Fandom Original

[Referenced in pages 5 and 22]

[35] *Final Fantasy Tactics* (1997), developed by Square, published by Square and Sony Computer Entertainment.

Snapshot of the official page: https://web.archive.org/web/20090617052440/http://www.square-enix-usa.com:80/games/fft/fft-index2.html LA 2025–03–01 Arch 2019–06–17

Wiki: https://antifandom.com/finalfantasy/wiki/Final_Fantasy_Tactics LA 2025–03–01 Fandom Original

[Referenced in page 5]

[36] *Fire Emblem Awakening* (2012), developed by Intelligent Systems, published by Nintendo.

Snapshot of the official page: `https://web.archive.org/web/20190430083426/http://www.fireemblemawakening.com` LA 2025–02–28 Arch 2019–04–30

Wiki: `https://fireemblemwiki.org/wiki/Fire_Emblem_Awakening` LA 2025–02–28

[Referenced in pages 1, 2, 3, 5, 15, 19, and 26]

[37] *Fire Emblem Fates* (2016), developed by Intelligent Systems, published by Nintendo.

Snapshot of the official page: `https://web.archive.org/web/20210117181056/https://fireemblem.nintendo.com/fates` LA 2025–02–28 Arch 2021–01–17

Wiki: `https://fireemblemwiki.org/wiki/Fire_Emblem_Fates` LA 2025–02–28

[Referenced in page 5]

[38] *Fire Emblem Heroes* (2017), developed by Intelligent Systems, published by Nintendo.

Official page: `https://fire-emblem-heroes.com` LA 2025–02–24

Wikis: `https://fireemblemwiki.org/wiki/Fire_Emblem_Heroes` LA 2025–02–24 and `https://antifandom.com/feheroes` LA 2025–02–24 Fandom Original

[Referenced in page 21]

[39] *Fire Emblem: Shadow Dragon & the Blade of Light* (1990), developed by Intelligent Systems, published by Nintendo.

Wiki: `https://fireemblemwiki.org/wiki/Fire_Emblem:_Shadow_Dragon_%26_the_Blade_of_Light` LA 2025–02–24

[Referenced in page 4]

[40] *One Step From Eden* (2020), developed by Thomas Moon Kang, published by Humble Bundle.

Official page: `https://www.onestepfromeden.com` LA 2025–02–20

Wiki: `https://antifandom.com/onestepfromeden` LA 2025–02–20 Fandom Original

[Referenced in page 3]

[41] *Symphony of War: The Nephilim Saga* (2022), developed by Dancing Dragon Games, published by indie.io.

Official page: https://www.dancingdragongames.com/symphony-of-war
LA 2025–03–11

Wiki: https://symphonyofwar.wiki.gg LA 2025–03–11

[Referenced in pages 7, 8, 15, and 26]

[42] *Tunic* (2022), developed by Isometricorp Games, published by Finji.

Official page: https://tunicgame.com/ LA 2025–04–26

Wiki: https://antifandom.com/tunic LA 2025–04–26 Fandom Original

[Referenced in page 19]

[43] *Unicorn Overlord* (2024), developed by Vanillaware, published by Sega and Atlus.

Official page: https://unicornoverlord.atlus.com LA 2025–02–19

Wiki: https://antifandom.com/unicornoverlord LA 2025–02–19 Fandom Original

[Referenced in page 3]

[44] *Wargroove* (2019), developed and published by Chucklefish.

Official page: https://wargroove.com LA 2025–03–11

Wiki: https://wargroovewiki.com LA 2025–03–11

[Referenced in page 6]

[45] *Wargroove 2* (2023), developed and published by Chucklefish.

Official page: https://wargroove.com LA 2025–03–11

Wiki: https://wargroovewiki.com LA 2025–03–11

[Referenced in page 6]

[46] *XCOM: Enemy Unknown* (2012), developed by Firaxis Games, published by 2K.

Official page: https://www.xcom.com/xcom-enemy-unknown LA 2025–02–21

Wiki: https://antifandom.com/xcom LA 2025–02–21 Fandom Original

[Referenced in page 3]

[47] *XCOM 2* (2016), developed by Firaxis Games, published by 2K.

Official page: https://xcom.com LA 2025–02–21

Wiki: https://antifandom.com/xcom LA 2025–02–21 Fandom Original

[Referenced in page 3]