Developing a Visual Novel Framework for Roleplaying Games

Janne Andsten

Haaga-Helia University of Applied Sciences

Bachelor's Thesis

2021

Bachelor of Business Information Technology

12 December 2021

Abstract

Author(s) Janne Andsten	
Degree programme Business Information Technology	
Report/thesis title Developing a Visual Novel Framework for Roleplaying Games	**Number of pages and appendix pages** 40 + 2

This thesis is about designing and developing a visual novel roleplaying game with the Ren’Py game engine, with the goal being designing and creating mechanical functions and components from the ground up to allow the continued development of the video game to focus on the creative writing and story development once the thesis is over.

Ren’Py is a visual novel creation engine which utilizes Python as a coding language. It provides the bare necessities to create a visual novel video game. With customization of its codebase and adding in components and functions in this project, the goal is to create a development environment where the developer can focus solely on the narrative writing aspects of game development, requiring minimal coding experience.

This thesis is a solo project made for potential sales once the product is finished, though neither sales nor finishing the product is within the scope of the thesis. They are, however, considerations that must be made during development. Arts and other custom assets are not focused on, despite being a large part of a visual novel game.

The result of the thesis is the completion of the primary gameplay mechanics and a user interface that shows the information that it needs, even if its design is not finalized. Significant progress in the growth as an independent video game developer is made.

Keywords
Video game, development, design, games, Python

Table of contents

1 Introduction

Video games have for the longest time been a dear pastime for me, and there have been several times when I have daydreamed on the topic of creating one. Insomuch that were I ever to create my own business and begin selling a product, writing my own video game seemed like a natural choice. Over the years I have developed what I like to think a keen eye towards game design, and a penchant for recognizing what works and what does not. Putting them to use in creating rather than just critiquing would be a big step forward for me.

However, writing a video game is a big project and doing so as a solo developer seems extremely daunting. Yet with the proper tools and support, creating a video game from almost scratch should be doable. Focusing on the development process from the ground up from ideas to design to implementation, this thesis is meant to document the entire early process of developing a prototype framework for a visual novel roleplaying game. A finished product is not intended to be in the scope of the thesis, but instead what should be provided by the end of it is a functional codebase for a game that allows for further development into a proper, sellable product.

The thesis will primarily talk about the design and development processes, how features got implemented into the (for now) final presented version and what got left onto the chopping board and why. In addition, discussions on the future of the product will also take place, where the development can go from here and what will have to be done to make the product viable for sale. And finally, the personal growth of myself as a developer will also be touched upon as an important part of the overall process of development.

As the founder of my own business meant to sell the product and as an independent developer, I am myself in charge of defining the minimal viable product as well as to make sure that the features I wish for are implemented in a sufficient manner. Self-control and discipline are keys to success in this instance, but also a willingness to accept that I cannot necessarily complete everything I wish to in the timeframe I am given (and have given to myself). Perfectionism is an easy trap to fall into, but it should not become a deterrent to progress, no matter what happens.

The roleplaying game genre has been chosen because tabletop roleplaying games are something as dear to me as video games if not more so, and my background in reading, analysing, playing, and even sometimes creating my own will serve me well in the process of translating the rules into a video game format. An emphasis in the game will be put on dynamic storytelling, player-facing experiences and unique experiences based on player choice, while adhering to mechanical standards set out by existing games of the same genre, as well as good practices in general as defined by myself and the consensus of developers of modern times.

It is my goal and the product's goal to create a framework for further developing an accessible and modern visual novel game with heavy roleplaying game elements, with mechanics, design, and user interface to support the goal. Functions, data formats and code developed to make it easy to focus on the narrative writing process of the future.

To aid in my progress are several books on developing games, such as The Art of Game Design by Jesse Schell (Schell, 2020), and Game Development with Ren'Py: Introduction to Visual Novel Games Using Ren'Py, TyranoBuilder, and Twine by Robert Ciesla (Ciesla, 2019).

2 Project Scope and Objectives

Before we begin on the chapters about design and development, the scopes of the thesis, tools used and general definitions on what the project is even about have to be laid out.

2.1 What is a visual novel?

Visual novels are a cross hybrid between a video game and a written story, involving heavy use of art and visuals to convey characters, backgrounds, and actions, hence the name "visual novel". They originated in Japan, and are immensely popular, and in 2005 accounted for over seventy percent of video games published for the PC platform in Japan (Anime News Network, 2006). Some of the well-known visual novels are the Ace Attorney series of video games, and the Fate series. The former is most known as a video game, while Fate has had several spinoffs in media, such as the popular animated show Fate/Stay Night.

There exist two different main types of visual novels: visual novels and kinetic novels. Kinetic novels are linear, with no choices or alternative paths. Visual novels involve choices of various kinds, typically through presenting different actions or dialogue the player can choose from, sometimes with extra game mechanics added to them, like tracking how much other characters like you, your character's "health" or money they have.

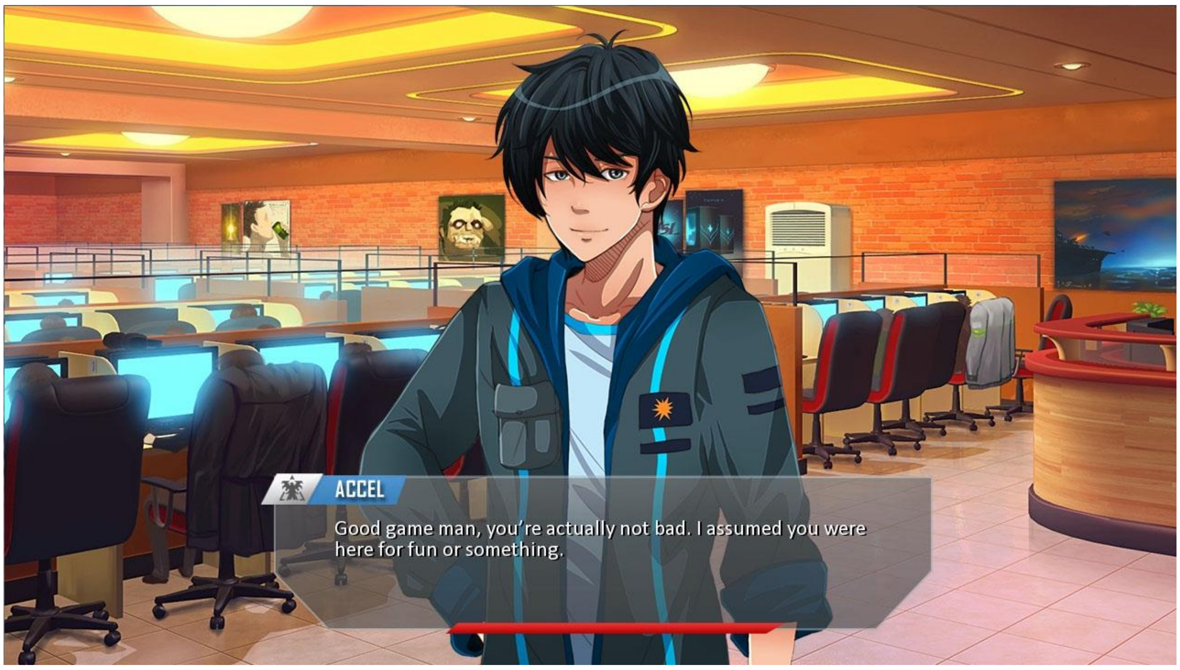

ACCEL
Good game man, you're actually not bad. I assumed you were here for fun or something.

Figure 1: SC2VN (Team Eleven Eleven, 2015), a visual novel taking place in the *Starcraft 2* e-sports scene.

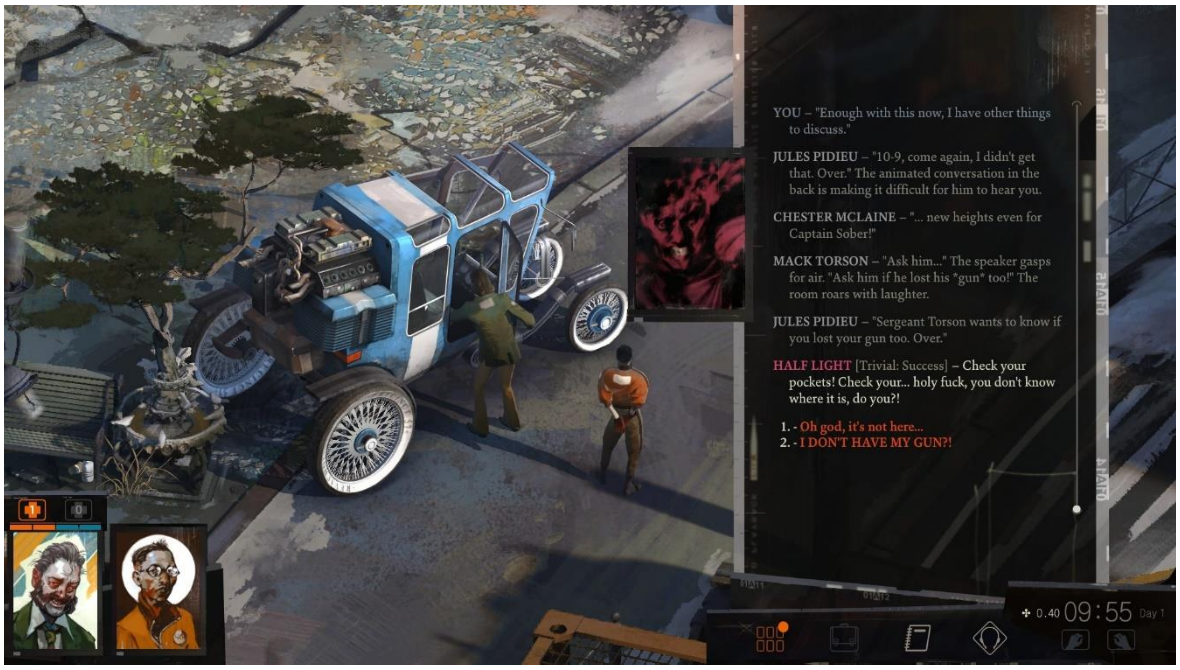
The project for this thesis will be made as a visual novel, with both choices and game mechanics heavily involved in the design from the ground up. Drawing inspiration from both visual novels and roleplaying games, the goal is to provide a storytelling experience that is both dynamic, interesting as well as replayable, bypassing some of the design features that I personally find not to my liking, such as failure states (instances of gameplay where the player's choices lead to game overs and reloading from earlier spots in the game to then take the "correct" choices), and "bad endings" (where your choices result in an end that is less satisfying or not considered to be the "real" ending). Whatever the choices the player makes, it should result in a satisfying story that they can look back at and compare with others, forming something uniquely their own. These objectives stand out throughout the report and are expanded upon later.

2.2 Why a visual novel?

I chose to write a visual novel for several reasons, though the main reason has always remained the same: a passion for storytelling. As a thesis topic, on the other hand, it provides a holistic and fairly comprehensive experience in designing, coding, and writing a product of its kind, in this case a video game, without requiring me to outsource the skills that I do not have: namely, skills in visual arts. For certain, a finished visual novel absolutely requires artwork, but as long as different placeholder images exist for each eventual portrait of a character and their expressions that get shown on the screen, background and splash screen (larger images that take up the entire screen, typically used to emphasize a scene), the core gameplay of the visual novel can be finished, and temporary art assets simply replaced by their final versions.

Visual novels also allow me to bypass the need to build an entirely new engine for my game. Ren’Py is a Python based visual novel engine that still allows for great amounts of customization (Ren'Py, n.d.). It is essentially a canvas that has the main functions of a visual novel built-in: text scrolling, essential UI elements, a menu function, and save/load states. Everything in it can be further customized via Python script, making it an excellent tool to get started on customization while knowing it can still deliver the fundamental experience of a visual novel. It is also free and open source, making it a good choice.

Several other visual novel creation engines were considered, such as Visual Novel Maker, but Ren’Py was decided upon due to its free cost as well as more inherent coding capabilities. It is possible that the engine will be changed at some point upon expansion of the scope of the game, but that is also beyond the scope of the thesis itself. If it is done, much of the code will have to be rewritten, but the design principles will remain the same, as will the narrative.


YOU – "Enough with this now, I have other things to discuss."

JULES PIDIEU – "10-9, come again, I didn't get that. Over." The animated conversation in the back is making it difficult for him to hear you.

CHESTER MCLAINE – "... new heights even for Captain Sober!"

MACK TORSON – "Ask him..." The speaker gasps for air. "Ask him if he lost his *gun* too!" The room roars with laughter.

JULES PIDIEU – "Sergeant Torson wants to know if you lost your gun too. Over."

HALF LIGHT [Trivial: Success] – Check your pockets! Check your... holy fuck, you don't know where it is, do you?!

1. - Oh god, it's not here...
2. - I DON'T HAVE MY GUN?!

Figure 2: Disco Elysium (ZA/UM, 2019) is a game where you play as an alcoholic detective in a strange retro inspired world, from an isometric perspective that is classic to roleplaying video games.

There is no reason the project needed to be a visual novel, however. It could have been a more “traditional” video game, such as an isometric-camera 3D game like Disco Elysium (ZA/UM, 2019). Visual novels require less artistic coding and developing to make work, so for the confines of this project the fewer assets needed, the better.

2.3 Goals of the project

The goal of the project is to create a mechanical framework for the creation of a visual novel. This includes several game mechanics that need to be created from scratch, and to demonstrate them in a (relatively) short but dynamic and exciting narrative section.

The “framework”, as I call it, is, in essence, a library of all the programming components needed to build a game with the vision I have, allowing future development to continue without having to program more functions unless something highly specific and unexpected is needed. Once finished, it should allow anyone to start or continue development with minimal coding experience, even if it is primarily meant for myself. The functions and features created for the framework that Ren’Py does not provide by default should be easy to use, with as much automation as possible.

The goal also includes a short “proof of concept” video game that shows how a full game will function in a short narrative segment, with several game mechanics included into it. By playing through the segment a reader would get the idea of how the game’s functions work in play, even if the mechanics are invisible to the player at the time. Ideally all mechanics created would be shown in the sequence, and in ways that they would function in a full product as well. The player can then look at the source code of the segment and see how each of the functions work.

The scope of the project may seem very large at first, and that is true. However, parts of the goal of the project as well are to learn proper project management in the building of the video game. As such, maintaining flexibility in what and how features are implemented is important. This mostly applies to the programming and development parts of the project, where realizing that a feature is better left out of scope might happen during the process of creating it. As the product is made to be kept developed after the thesis is finished, this is not an issue.

In addition, a large limiting factor is that the product is being developed by a single person. As such, it is necessary to keep the scope of the thesis manageable by a single person, in both size and topics. There are several aspects of developing a video game that need to be considered that I, as a solo developer, am simply not qualified to talk about. The business side, marketability are two topics that will get a cursory glance at best and remain outside of the main focus of the thesis.

Likewise, while visual novels almost universally have artwork in them, due to the constraints and budget of the project as well as the lack of visual artistic talent from the writer (once again myself), very little “proper” artwork will be used. Instead, placeholder art will be widely used, as well as royalty free art when possible. In the future, more artwork may be acquired and utilized to finish the visual novel in its entirety, but that does not fall under the goals of this project.

What the video game itself is meant to do is to take the strengths of both tabletop roleplaying games and computer roleplaying games to form something that is new and exciting in its combination of strengths, to take the best of both worlds in a new direction for video games. This approach, especially if it were to gain traction among visual novel makers, could bring tremendous value in providing direction and principles to the medium of video games.

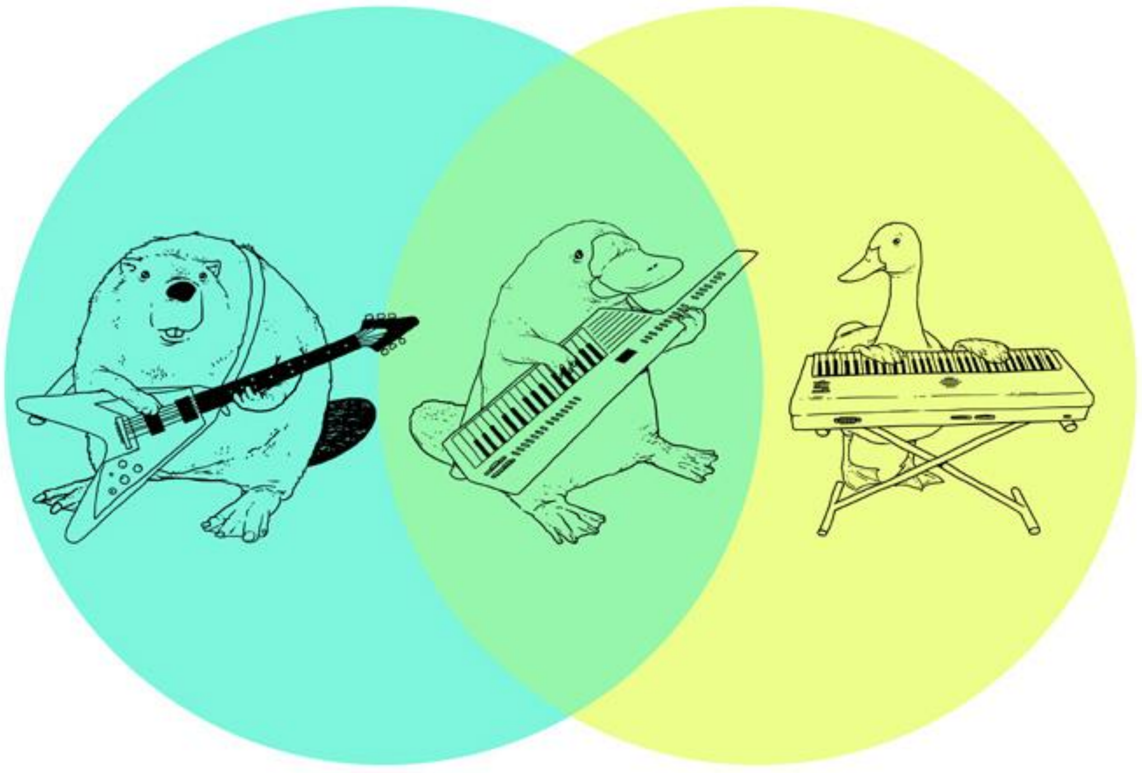


Figure 3: Much like the platypus wielding a keytar, my product would take the best of all kinds of roleplaying games. (Source unknown.)

2.4 Process of Development

Developing a game from just a barebones engine such as Ren'Py is a very iterative process. Before the code parts of the game can be written, design needs to be documented and figured out. However, as an iterative process, development often leads to new design necessities, so even if I were to design the entirety of the game before writing any code, the realities of game development would make me go back and redo design in order to fit what is better for the game, gleaned through experience and testing. As such, the design feeds the development, and the development feeds the design.

Once both development and design are done, though, can the proper implementation phase begin. In that phase, the majority of the code is written already, and the emphasis is on creative writing and putting the pieces together into a whole. There will no doubt exist moments where I need a function or mechanic that does not yet exist, but if all is done well those situations should be few and far between. As it is mostly creative writing, the implementation phase is not in the scope of the thesis much.

For clarity's sake, the thesis divides the process of development into two sections, design and development (the latter of which refers to the code writing part of development). The actual process was never this linear, but the groundwork laid out in the design chapter was set before coding truly began.

3 Design

Before any development can happen, there must be a fundamental layer of design from which to work my way upwards. To accomplish this, I began the entire process of game development with a dedicated design phase, to figure out what I needed, what I wanted and what there was in the market that I felt I could improve upon or incorporate into my own video game. When that was finished, a second design phase could begin where the components of the video game would be outlined, details honed out and put into place, and probabilities calculated. Coding aspects will not be touched upon during design phase, but the capabilities of Python (and my own proficiency in bending it to my will) will of course influence how the various designed aspects make their way into the game.

3.1 Design principles

First things first, Schell writes that game development is more than just developing a game: game development is about creating an *experience* (p.10), and that games are just a means to an end towards creating that experience in the player's mind. To facilitate this approach of thinking and to better understand my project, I need to define what kind of experience the game is meant to create. To do this, I used a technique I have had success with earlier in creating things: a small list of principles that form the basis of what I am doing and that act as a focal point when I am unsure what to do.

I started development by making up these three core principles:

The game's goal is to:

1. To create an exciting, dramatic story for the player to follow.
2. To give the player agency in how the story unfolds.
3. Create replayability through several different options and routes to take.

To clarify these principles and goals even more, I want to look at another source of examining player engagement, through Hunicke et al's paper on the MDA approach: in this paper, eight aesthetics of play are defined:

1. Sensation: Game as sense-pleasure
2. Fantasy: Game as make-believe
3. Narrative: Game as drama
4. Challenge: Game as obstacle course
5. Fellowship: Game as social framework
6. Discovery: Game as uncharted territory
7. Expression: Game as self-discovery
8. Submission: Game as pastime

Two of them strike out as the most prominent ones are Fantasy and Narrative. The game is primarily a story, one set in a land of wonder and fantasy. There are elements of Sensation present from the artwork of the medium of visual novels, as well as some degree of Expression from creating your own character, but those are not the primary focus of the game, with Sensation not being a part of this project at all due to a lack of dedicated artwork to begin with. Expression should, however, be noted as important for the second and third principles of design, for the capability to create your own character (thus expressing yourself) is key to giving the player agency and allowing them access to different routes.

With this in mind, I felt it was necessary to revise the principles to fit something more syncretic instead. So, I came up with the next iteration of principles:

1. Present an exciting, dramatic Narrative.
2. Give the player freedom to Express themselves through their character.
3. Immerse the player in a world of Fantasy, through the story and Sensations of art and music.

As can be seen, introducing the core aesthetics of play into the design principles varies them up and focuses on providing meaningful principles that directly affect the player and their experience. This should help in the long run. Expression remains second, despite not being a primary focus. When writing these new principles I realized it can encompass many different aspects of the player's agency, through things like which skills the player wishes to focus on, or which decisions they make and which characters to befriend. Essentially, any time there is choice involved in the game (even relatively meaningless ones), it contributes to the Expression aesthetic of the game. As stated earlier, it is a goal that when a player finishes the game, they are left with something that is uniquely theirs. Thus, Expression becomes a lot more important than I originally thought.

3.2 The problems and the solutions

Before designing anything, I needed to look at what things specifically I felt were wrong with the visual novel and/or roleplaying game genre and what I needed to do to make them feel better. In order to find the best solutions, I needed to look everywhere I could for inspiration, not just video games. Tabletop roleplaying games in particular were a big source of inspiration. In addition, it was crucial to analyse why exactly certain mechanics and features were in place before tinkering with them. To break the rules, one must first know why those rules are in place.

Fundamentally, roleplaying video games are in the business of storytelling, and there is no overwhelming problem that *needs* fixing per se. However, there are tropes and standards that have been accepted over the years that I personally am not fond of and that I feel I should address if I am to design my own storytelling experiences.

Problem 1: The idea of the "best ending". In visual novels where the player has choices they can make, it is very common for some choices to contribute towards "bad endings", where the game ends in a bad or unsatisfactory ending (like where the protagonist fails to achieve their goals, or certain important characters die). I want to avoid this, and instead provide players with many different satisfying endings that reflect their own progress. While some may be better than others in terms of what got achieved, I want players to feel like it was their story that came to a natural conclusion and that they do not feel left out just because they made a choice that they could not have foreseen to be bad.

Problem 2: Transparency. I also want my mechanics to be transparent, and information easy to reach. If players are attempting to accomplish something, I want them to be informed about it. This allows the players to better make decisions that they feel are satisfying, and that they can gauge the risks for rewards. Clear communication of mechanics, causes and effects of skills, and if there is an important choice, letting the player know they are reaching an important point in the game where their choices will significantly alter their future. Ideally the narrative would take care of this in a way that feels natural, but if and when that fails being explicit about mechanics is an acceptable solution.

Problem 3: Reloading. With saving and loading being an integral feature to almost all games, it can often feel like failure does not matter if reloading is simply a few button presses away. For if a player fails in something they feel important, they can simply reload the game from a prior save and try again until they succeed. To solve this, a way is needed to ensure that players *want* to keep going, regardless of what the results of their choices are. This ties into the first problem with “best” choices and consequences. This is not as easily solved as might be, because no matter how much reloading would be disincentivized there will still be players who want to succeed at critical points. That is fine, the goal for this problem is not perfect elimination of reloading, but to reduce the incentive to do so. It is important to me to *incentivise* continuing, not enforce it. Players should be able to play the game the way they want to, and if that means reloading until they get their desired result, then I cannot judge them on that.

Problem 4: Immersion. It has been my own experience many times that games have felt the need to insert mechanics for the sake of having mechanics, whether it comes from endless combat encounters to artificially lengthen the game, or from clearly marked dialogue options where you click on the button marked with a heart symbol just because that is the one that the game tells you will lead you to romance that character. These sorts of things pull me out of the game, and I would like to be able to prevent this from happening by creating mechanics that feel immersive to the player and do not detract from the narrative at hand. My solution to this is to primarily focus on the narrative and the fiction surrounding it, adding in mechanics where they are needed instead of beginning with the mechanics and building a narrative around it. This might conflict with the ideas of mechanical transparency, and if there is a conflict, I believe that transparency should win out.

Some of these problems can be solved with clever applications of game mechanics. Others need to be dealt with designing narratives that subvert or bypass these problems. What form they take is a big part of the design phase of the project.

3.2.1 Earlier works and inspirations

It is worth documenting where my inspiration of design comes from to be able to better see how earlier design affects my own. Games inspire others, and taking good ideas from earlier sources allows me to credit those that came before.

Perhaps the primary video game inspiration comes from the game Disco Elysium (ZA/UM, 2019). Unlike most fantasy games (and unlike the one I am creating), Disco Elysium is set in a modern setting with technology and fashion in the 1980s, albeit in a world not like our own. Instead of larger than life heroes slaying dragons, its story is more human and grounded, as well as somber, introspective detective story. Mechanically, Disco Elysium is mostly dialogue, making it not entirely unlike a visual novel, albeit without the same graphical trappings. Its use of skill checks within dialogue and vast variety of ways to solve problems and express yourself through the game's protagonist have given inspiration that cannot be understated in how impactful the game has been.

Another genre of games that has given a lot of inspiration is the ”immersive sim” (Samoylenko, 2018). While vaguely defined and hard to pin down though with common features such as exploration, a first person perspective and a large variety of tools at the player's disposal, a common thread in the genre is the ability to solve problems in the way you want, through stealth, clever use of game mechanics, violence or just talking your way out of things. This kind of player choice is important to me, and is something I wish to emulate, even if it is only through menus and choices.

A flaw in my background that I must admit is that I do not have as much experience in playing visual novels as I probably should. I have played a few over the years, yet as a genre they have never interested me too much. This is not because of the format, however, as much as it is about the lack of interesting visual novels that I have been able to find. I do feel like I have enough experience in reading about them and the design knowledge to make use of them. With some extra reading about how to make the best of them, I expect the end result to be if not great, then at least good enough. The goal has always been to create a hybrid of a more classic roleplaying game within the trappings of a visual novel, so the lack of inherent knowledge of many tropes used by visual novels might give me a fresher perspective on the genre. Still, research has had to be done.

Combining video games with tabletop roleplaying game (such as the venerable Dungeons & Dragons) is a fundamental aspect of the project, and one of the biggest inspirations on tabletop mechanics is the Powered by the Apocalypse design philosophy, created by

Vincent Baker (2010). In the philosophy, colloquially called PbtA, failure at doing something always causes something interesting and fiction moving to happen, not simply a case of ”nothing happens, try again” nor ”you die, game over” (unless under the most dire of situations). This ties in directly with the principle of moving the game forward regardless of how well a skill check goes, resulting in a narrative that does not require the player to reload. In addition, the idea of having three separate results comes directly from the PbtA philosophy, where you have a miss, a weak hit and a full hit as results.

Another tabletop roleplaying game that has given inspiration is Fate Accelerated by Evil Hat Games (2013). In it, characters are not defined by what they are capable of, but by how they solve problems, with ”approaches” of Forceful, Flashy and similar descriptors. This idea is incorporated into the game I am making, combined with an actual skill list, to give more variety to how the player can use certain skills, and to allow players to express the personality and style of their character as much as their proficiency in things.

There are probably dozens of more sources of inspiration that I have gained through osmosis over the years. To me, the sign of talented development is not to take things from other sources wholesale, but to be able to implicitly understand all sorts of different approaches and be able to select what works with the approach that is intended.

3.3 UI design

With the principles and problems laid out, the next step was going into the design aspects of the game. I decided to start with the most visual element, the UI.
For this prototype framework, the design ethos I held for the user interface for the game was “simple but useful”. There were two main challenges that came up when starting the design: how to give all the necessary data to the player while keeping the focus on the art-work and not clutter up the UI with too many windows.

William Nelson wrote in his “Best practices” article about the purposes of UI and what was needed to create an efficient one. The fundamental idea that Edd Coates (the person be-ing interviewed) said was that the developer “should anticipate what the player is going to want to know, and what they’re going to want to see”, and to take inspiration from earlier games and iterate upon them. For this project, finding older roleplaying games to use should be something to design from, and then see how they adapt into the gameplay of a visual novel.

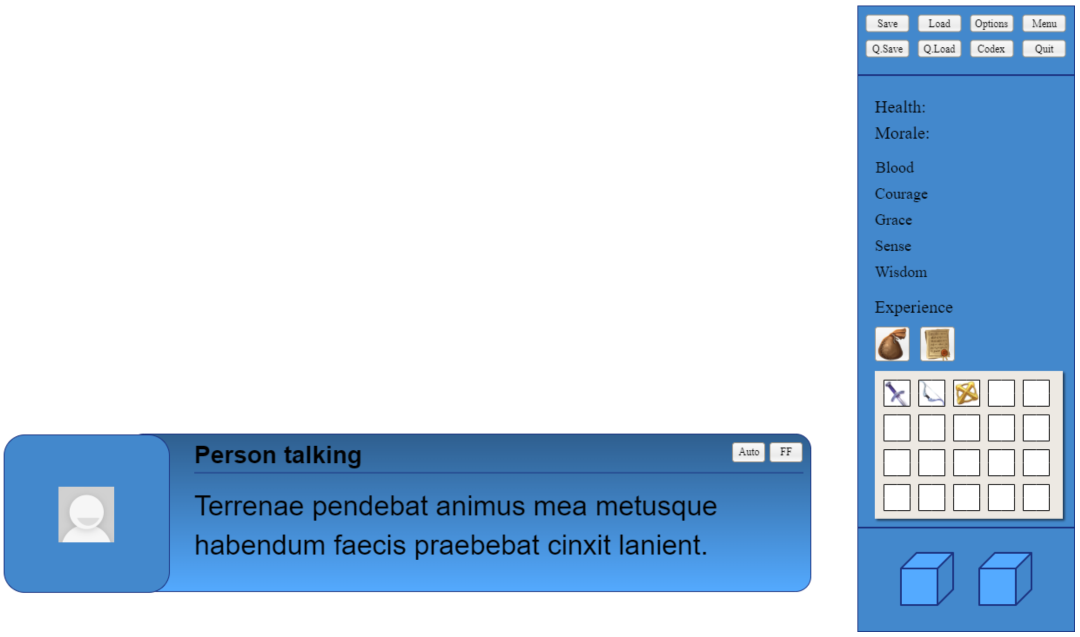
With that in mind, the primary things a player needs to see for a visual novel are the dialogue box where the writing happens, as well as some screen to see the player's statistics and current inventory. Thinking as a player, the most important pieces of information I would like to see in a UI are the information that changes most often. For this game, that would be the Morale bar and the inventory. Furthermore, being able to see who the player's avatar is at a glance and what they are good at (in case the avatar changes during play) is also important. Overloading information is also to be avoided, so any longer lists of things should be in their own views, accessible through buttons on the UI.

Looking at other earlier games, one old first-person roleplaying game, Wizardry 8 (see below), was a direct inspiration. Even though it is not a particularly modern game, its design appealed to me. In Wizardry 8 (see Appendix 2), part of the UI was covered by a bar showing vital statistics such as the player's health bar, skills, and inventory. However, instead of being at the bottom on both sides of the textbox (an integral part of the visual novel experience), I opted to add it to one side, where it could show all the necessary information. As an added touch, to emphasize that tabletop roleplaying feel, a small dice box was added to the bottom to fill in the otherwise rather empty space. In the future, animated dice could be placed there to roll whenever needed, to give a bit of kinetic feedback to the game and to make the randomized roll checks feel more real.

Showing the results of the individual dice rolls (two 6-sided dice are rolled whenever a skill check is called for) as well as showing the rolling process as well was an idea that could theoretically help the player feel like the dice are more impartial and not simply two random numbers created by code. For the time being no 3D animations will be used for the dice, but images of the dice should be made to shuffle around for a few seconds before settling to the result to simulate anticipation and a feel of rolling.

The design is obviously subject to great change during development, as skills and time permit. Visual design is not among my strong skills, and I doubt I will ever be truly satisfied with anything I create myself. As such, the emphasis on how attractive the UI looks is of secondary importance to simple functionality.

In the early design stage, some wireframes were built for the UI to get an idea of how the UI would function, where the various buttons and styles were placed. This allowed me to better visualize what was going on, and whether or not doing what I was a good idea.



Person talking

Terrenae pendebat animus mea metusque habendum faecis praebebat cinxit lanient.

Figure 4: First UI wireframe developed for the project. No background or characters are placed yet.

3.4 Design of mechanics

Games are very little without their mechanics, so no small amount of effort was given to make sure that the game's mechanics felt good, intuitive and that the statistical range of effects would be consistent, and that a sense of progression is gained through advancement. The aim is also to create systems that feel immediately familiar to a player, by using key tropes that players should be familiar with. At the same time, if a player is unfamiliar with roleplaying games, they should not be left wondering what is going on and why. Transparency, again, is key.

The game has only a few core mechanics to it: skill checks, inventory checks and skill unlocks. Inventory checks are the simplest of the three: the player has an inventory of items they carry around, each with a type. During the game, there are options that require you to carry either a specific item (like rope or a pouch of gold) or a specific type of item (like a weapon). If the criteria are fulfilled, that option becomes available, sometimes using up the item in turn. This rewards exploration and replayability, as players who do not have the item with them at the time will still see the option, but have it disabled, letting them know they missed out on something.

Skill unlocks work similarly. Skills have three ranks: untrained, trained and master. Each rank gives a bonus to skill checks (as explained in the next paragraph), but also unlock new options which would not require rolls or allow for rolls not related to that skill. A master of fencing would be able to use that part of the character to, for example, convince a duellist to stand down and not fight them by their sheer reputation alone.

Skill checks are more mechanical, but not aggressively complicated. Whenever the outcome of an action is in question, two 6-sided dice are rolled, and a bonus is added based on the player's rank in the check's assigned skill. In addition, the player also has characteristics that represent their character's proficiency in not what they do, but how they do it (flair, focus, force, guile, haste, and smarts). This number, ranging from 1 to 5, is also added to the result rolled. The result is compared to the difficulty of the check (with several different levels of difficulty), and the result is either a full success, a partial success, or a failure, depending on whether or not they pass the threshold for success and of what kind. Partial successes are easier, and full successes harder. Each result has its own consequences, but all of them move the game forward in some way, as failure might not mean a failure to accomplish the task at hand, but of some other consequences interfering or resulting of it. The idea of partial successes and full successes comes from Apocalypse World (Baker, 2010), one of the big tabletop roleplaying game inspirations.

Apocalypse World does not have specific difficulties for its skill checks, but I wish to have them in the game I am making. As such, the statistics need to be taken into account for each difficulty to make sure that they scale well, and also to have an idea of how large bonuses characters should have by the time such checks become more commonplace. With six different difficulties (easy, medium, hard, heroic, legendary, godlike), the full success thresholds begin from 8 and end at 18, in increments of 2. The most common result from 2 six-sided dice is 7, and as such it was a good place to start for a check where even the least experienced character would have an equal chance of succeeding, as characters have a minimum statistic of 1.

For partial successes the thresholds work somewhat differently. The starting difficulty is 5 and it increases by 2 up to 11 at heroic difficulty, but afterwards increases by 1 for legendary and godlike, to 12 and 13 respectively. The decision for this was made to make those target numbers slightly more accessible to characters who are not extremely specialized. Gaining a full success on a godlike roll requires a bonus of +6 minimum and even then, the player would have to roll two 6s on the dice, making a success very unlikely. Therefore, having a lower bar for a partial success would, ideally, make those options more attractive to players.

The chances and percentages are all to be changed with feedback and more background math. But it is good to have a baseline of probabilities helps give an idea of what is going on and how.

All information about skill checks, available unlocks and consequences are meant to be transparent, and the player should know what is going on and when. If requested in feedback, however, an option to turn off the visibility of this information might be added later, but there are no plans to implement it right away.

Progression systems are paramount in almost any roleplaying game, and this project is no difference. Experience points are gained through skill checks, but as an incentive to not reload after failing at a check, failure gives the most experience points, while succeeding gives the least. Other ways to earn experience should be implemented, like end-of-chapter or milestone rewards when narratively appropriate. This too echoes Apocalypse World in some ways, but with my own spin to take into an environment where there is not a person running the game with a human touch to it.

3.4.1 Roleplaying games, video games and mechanics

Games need mechanics, and in the game I am creating I aim to blend a mix of mechanics from traditional roleplaying games like Dungeons & Dragons with video game conventions. This is nothing new, as there are several video games that already use pen and paper roleplaying game mechanics. However, I am aiming to create them in a way that puts the narrative consequences in the forefront, and to provide an experience that is perhaps closer to the spirit of roleplaying games: combining expression with player agency. The mechanics in this game are not in place to create exciting combat encounters or dungeon crawling sequences where resource management is key, but to create a dynamic and engaging narrative where your choices are directly responsible for the consequences later on. To accomplish this, the mechanics need to match the idea. Fewer but more impactful rolls is key to me here.

In the idea phase, several different kinds of ideas of mechanics come to mind to make this game less of a static visual novel and more into a full-fledged roleplaying game. The most important one is the ability to do skill checks, but there are other mechanics that, to me, feel integral to the roleplaying game experience in a video game. While not all of them might make it into the game due to time constraints, skill ceiling, engine limitations or because they were simply not found to be fun, it is valuable to write them down regardless.

Skill checks, as mentioned, was an important aspect of the game and one that will not be compromised on. What form it takes, however, is one that will evolve. Currently they are planned to be a classic list of skills, such as fencing, alchemy or stealth, combined with an “approach”, a description of a character’s temperament and how they approach problems instead of what they know. When faced with an issue, the player can make a choice on how they handle a challenge, and the game will tell them what approaches the options use and what skills they have available. Each challenge should ideally have at least two different approaches available, with a variety of skills as well.

The second important feature, and a staple in visual novels, are character relationships. Throughout your adventure, your player character will form bonds with other characters, gaining their trust, friendship, romantic interest or even rivalry. Many visual novels focus heavily on the romance aspect of relationships (“dating sims” are a common genre of visual novels) (Ciesla, 2019), but for this game I wanted to go a bit more in depth. Inspiration is drawn from the video game Tyranny, where the player can form two different kinds of relationships with major characters: loyalty and fear, which are not mutually exclusive. The player can have both high loyalty and fear, one and not the other, or neither, and these combinations create unique situations each. While this game is not as fatalistic as Tyranny, a similar approach is planned: rivalry and friendship. NPCs might view the player character as a rival or comrade, as well as someone they like or dislike. A high rivalry and high friendship will result in friendly competition, while a high rivalry and low friendship will result in an active and bitter enmity. Due to the scope of the planned narrative these effects will likely not be properly seen, but the mechanics should be created regardless.

Dialogue systems are another staple in roleplaying games, but not so much in visual novels. I would like to experiment with the possibility of adding some sort of dialogue system into the game that is separate from the typical choice making process. I do not yet know how much this will compromise the characterization of the main player character, but experimentation with dialogue choices and branching dialogue is an interesting idea.

A fourth mechanic I would like to explore is the idea of an evolving and dynamic player personality matrix, where the choices you make during the game change the attitude of the character, resulting in unique interactions. Alignment choices are one example in existing roleplaying games like Pathfinder: Kingmaker, where your character’s alignment (e.g., Lawful Good, Chaotic Neutral, Lawful Evil, etc.) unlock different dialogue options. Alignment as it exists in tabletop games will not be used (as I personally abhor the system), but instead I was imagining personality types such as aggressive, diplomatic, sarcastic, and so forth. Whenever you pick a choice tagged with one or more of these personality types, it gets added to your character's personality scores. The highest score(s) then determine reactions and possibly even unlock options that would otherwise be unavailable. It would be yet another way to create more replayability, allowing players to pursue different kinds of characters and see different kinds of events within the game in a way that feels consistent with the overall characterization of the player's avatar.

The state of these mechanics can and will change during development, but for now these are some of the things that are planned for the game. The chopping board can be ruthless in video game development, but it is also good to keep options open and entertain possibilities.

3.4.2 The importance of fail-forward mechanics

One of the largest and most important features to design was a way for the game to continue playing even on the failure of a skill check. I have played several visual novels and other kinds of games where a single failure can result in a game over, or in an unwinnable situation, and that has even been a satisfying experience for myself. As such, designing skill checks in a way that keeps the game going is important.

There were a few problems that arose with this method that needed to be addressed. The most important ones I identified in the design process were meaningful consequences, how to handle more severe failure states and the complexity of the narrative resulting from these decisions.

For consequences, there are two categories of consequences: mechanical ones and fictional ones. In addition, there are two severities of them: soft and hard. These form the basis of consequences for the game. To explain, a soft consequence is one where the bad things that happen from the failure are lined up, and the player is given the chance to react to it. In contrast, a hard consequence is where the bad things happen without any way to react. As an example, pulling a gun out would be a soft consequence, and pulling the trigger on a gun is a hard consequence.

The differences between mechanical and fictional consequences are simpler. Mechanical consequences target the player character, by reducing their morale attribute or by removing items. Fictional consequences, on the other hand, change the situation.

Finding a balance between these options is crucial and should be done on a case-by-case basis when writing the branching narrative. There is no one size fits all solution to this, but records should be kept to see how often each consequence is used to balance them out.

3.5 Designing a branching narrative

The consequences of failure and a branching narrative deserve its own heading for discussion. The main problem of designing a branching narrative is the amount of branches a game can have eventually and how to design a scope that the game still can manage without constraining player agency too much.

As the current scope of the project is meant to deliver a small, self-contained narrative section, this is not a concern for the current project, but it is still one that should be considered for the future. There are some ways to narrow down the scope, each with their pros and cons:

1. Self-contained “acts” that limit choices and consequences within themselves.
2. Replacement characters for characters who might die or be absent from the story.
3. Choices between optional events that all lead to the same result.
4. Limit certain sequences to decisions made much before.

There are some problems with each of these options, if they are to align with the overall principles of providing choices and consequences. Using the first option too much can result in a narrative where the acts feel *too* self-contained and where choices made inside each act do not matter outside it. The second option can result in paper-thin characters replacing characters who the player might have grown attached to, and those new characters will have to work a lot harder to make themselves feel impactful. The third option makes all choices feel irrelevant, as if they all lead to the same conclusion then the player’s choices did not matter at all. And finally, the fourth option requires several different and mutually exclusive consequence sequences to be written, taking a lot of time to produce, with an alternative of making certain choices objectively better.

It is a goal of mine to make a narrative where there are fewer ”best” choices to make, and that there are many equally valid choices, consequences and narrative routes that lead to them. The player should get a satisfying ending based on the choices they made that represent those choices, and not punished because of them. Choices should also be transparent enough that the player knows what to expect from a choice, instead of being told it is something different than what it actually does. Unintended consequences are entirely acceptable, but a player feeling like they have been cheated is another and should absolutely be avoided as much as possible.

3.5.1 Interaction between mechanics and narrative

Verisimilitude, or the appearance of truth within the framework of the story, is an important factor for me in order to retain suspension of disbelief for the player. The mechanics and the narrative should both make sense to one another, and situations where a character should be able to do something but cannot for mechanical reasons should be avoided. Of course, due to the constraints of video games and programming eliminating such dissonance entirely is impossible, but it should be minimized.

An important way to minimize the breaking of disbelief is to always provide the players with enough options, even if they are not as different as they could be. The player should be able to express themselves through their character's skills and approaches both, so providing several options for how to act in a situation is important. Done carelessly this can lead to too many options, so a balancing act needs to be performed with enough options for the player to choose versus not too many that the player feels overwhelmed. One way to limit options is to block off decisions based off what skills and items the player has available (for example, you cannot choose to shoot something if you do not have a gun in your inventory).

3.6 The Story

While not the focus of this thesis, I would be remiss if I did not touch upon the story design in general. As the focus of the prototype framework, there will be a short and self-contained scenario to play through with a pre-made player character, Corvo. Corvo is a thief, sent to steal a valuable gem from a royal academy during an important feast. However, his timing could not be worse, as he then gets caught up in a coup by an ambitious and unscrupulous noble, forcing Corvo to manoeuvre between his objective, the chaos of the coup and his own morals. How players approach each situation before them is paramount, and the scenario might end in a vast variety of different ways, with some objectives completed, characters met (or even killed) and decisions made that would affect the story later on (which are not present in the demo section).

Players will be able to customize Corvo's statistics to a certain degree, though as a thief and scoundrel he will always have a certain skill set available to him. Due to the short structure of the scenario, not all skills will be helpful or used, so a baseline of useful skills will be given to the player. Customizing your character would be a proper feature later in development when the full story is implemented.

The player's character would have a voice and personality with the majority of dialogue written by the developer. Significant choices, however, would be implemented to allow the player to express their version of the character, highlight their own morals and priorities, and to promote the idea that the game and story they are playing is truly their own. These choices should always have some sort of effect in the game, be it different consequences to certain characters liking the player more or less because of what they did or said, or simply gaining a benefit that can help the player accomplish tasks later; or more likely, any number of the above. Skill checks and how the player rolls in them are also a significant factor in how things play out. Failing a skill check to sneak past some guards will result in a very different outcome than succeeding at the same task.

When designing the narrative and all its possibilities, in order to properly visualize the branching options, it was helpful to create a flowchart of scenes (see appendix 1) that led to one another and seeing how many different routes there were to each scene, making sure to account for references to the past. After all, it would not make sense for someone to mention something in the past that never happened in the route the player took! The complexity of the narrative is relatively easily handled though, with triggers embedded into decision points that are later referenced in conditional statements. Because the section given is a fairly closed and self-contained environment, there are fewer choices to make in the scenario than there would be in the "proper" version of the game. In this scenario, all routes lead to the same outcome. When developing further, there may well be entire sections of the game that the player never gets to see because the scenes played out in a different manner.

4 Development

Once the design was finished to a sufficient degree, it was time to begin the programming part of the development. But truth be told, the design was never finished and will never be truly finished, just iterated upon. But progress needs to be started regardless, otherwise the project would forever be stuck in the design phase.

Armed with the knowledge and design that I had, the first thing to do was to list everything I needed to create first. Organizing things into lists helped keep the progress in check and in case I ever felt stuck I could always take up another part while figuring out solutions to the others. The core parts of development were creating the UI, creating the player character mechanics, and auxiliary mechanics. The immediate focus was on the first two, as the auxiliary mechanics, while important to making the game feel like a proper roleplaying game, were not required for what I felt to be the core visual novel experience.

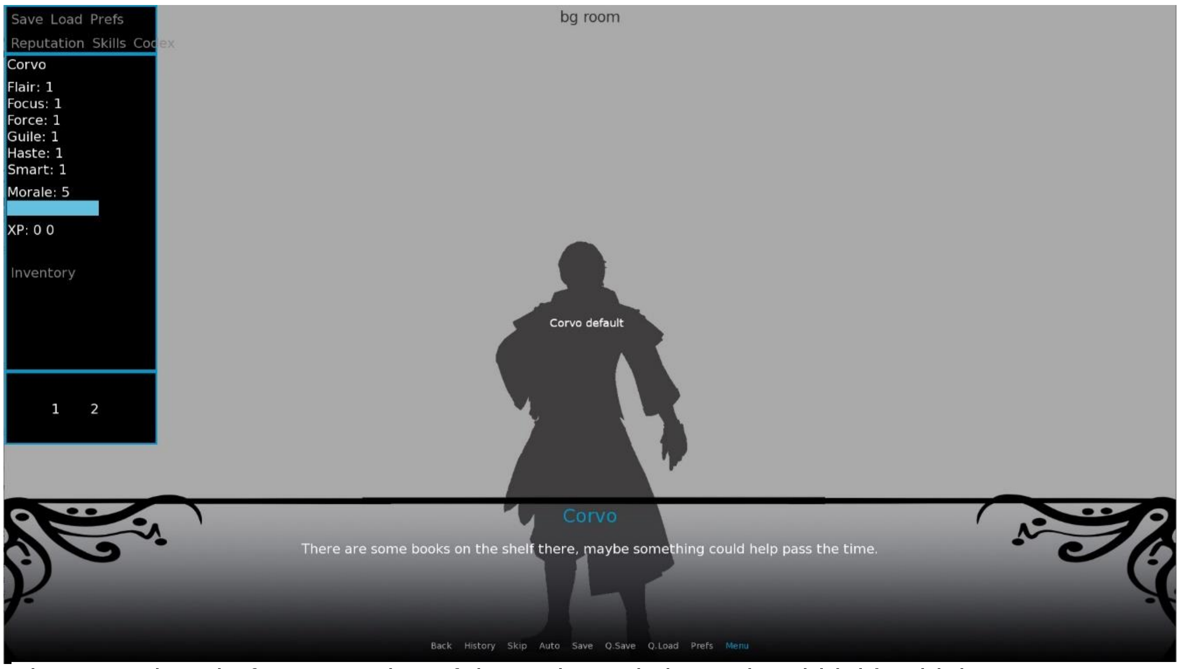
Obviously the most important thing above all of the others would be to make sure that the game can be played, text can be scrolled through, and dialogue shown, with characters appearing and disappearing when necessary. Ren’Py helped me greatly in setting up the fundamentals. As an engine made specifically for visual novels, it provided me with the bare necessities: a main menu, save and load functions, as well as the windows for displaying the speaker and text. More would need to be created, but Ren’Py did much of the heavy lifting to provide the chassis for visual novel development (which is also why I chose the engine). They were, however, entirely bare in terms of art. And while art was never a priority for the project, having something more thematic existing in place of black boxes helped the feel of development greatly, making it feel more like I was tangibly progressing in building something out of the elements I had. The free assets provided by LunaLucid and Fylgjur (online usernames both) were used to fill in the UI spaces and were fantastic in this regard, even if they might not be included in the final version.

The following chapters were not built in the order they are listed, but development spread between them intermittently. To make it easier to talk about each process, however, they are all listed under separate topics.

4.1 User Interface

The UI needed several parts developed. One to co-exist alongside the text window for quick reference of important information, and a few full screen windows to show statistics in more detail, such as the level-up window where you can assign skill points upon acquisition.

The first iteration of the reference window was three separate blocks on the right side of the screen that were always visible. While functional, upon testing it did not feel satisfactory, from neither a gameplay nor a visual standpoint. The data felt too static, and the sidebar always being there messed with the resolution proportions and screen space the game needed, as the background images should align to the same resolution as the game's window itself – an always visible sidebar would ruin that. The former issue might be solved with proper artwork and "prettifying" the UI, but that I cannot rely on right now. The solution for the latter, and to focus on the idea that the UI needs to show only what the player needs to know, was to make it a toggleable window, easily accessed through the quick menu and maybe even the mouse right-click, a function typically reserved for opening the main menu in visual novels (and one that Ren'Py has automatically assigned to that). This would make it easy to reference when needed, but at the same time not obstructing the art or events going on in the game at any given moment. Some other ideas were thought about, like placing it at the bottom as highly transparent, but upon mouseover have it come into view more. For simplicity's sake, for the time being it remains a toggleable sidebar, however.



Corvo
Flair: 1
Focus: 1
Force: 1
Guile: 1
Haste: 1
Smart: 1
Morale: 5
XP: 0 0

Corvo
There are some books on the shelf there, maybe something could help pass the time.

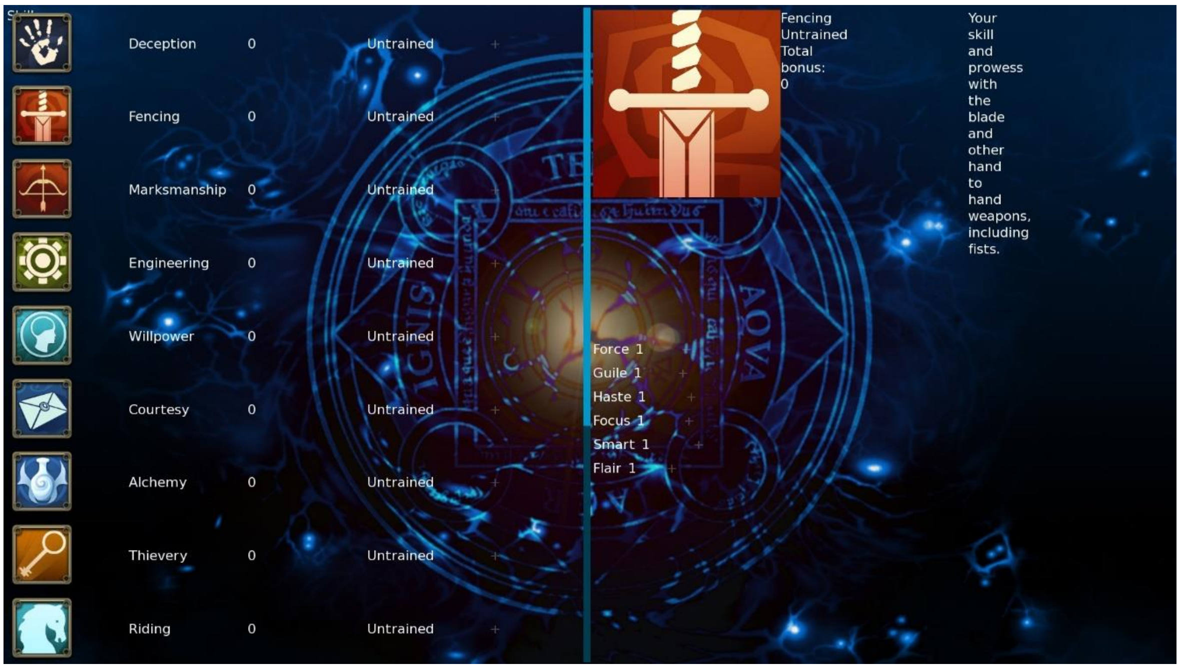
Figure 5: The wireframe version of the main statistics UI in a hidable sidebar

If time would allow it, the sidebar would also affect the positioning of characters on that side of the screen (or that side would never have characters so far as to get hidden by the sidebar. Or to change its position to somewhere where it could always be seen but not get in the way of the action. Due to the constant redesigning of the UI based on freely available assets, the sidebar will for now remain as a hidden object, called forth upon a button click.

One significant issue was encountered while developing the user interface was the Ren’Py mechanics of buttons on screens. For some reason I was unable to decipher, buttons with custom functions would trigger automatically whenever anything was clicked. This made the entire UI essentially non-functional whenever a button did something else than show a screen. Thankfully, I was able to fix it after a week of struggling with it though and found that Ren’Py only had a set number of functions that could be used on screen components. I was unable to figure out how to trigger custom made functions but changing variables and variables in dictionaries was made doable, and that was what I needed. So, while not as optimal as it could have been, I had to deem it good enough to save time and sanity.

With this big hurdle solved I could finally create several of the full screen screens that would allow the player to view and increase their abilities and other character values. Screens for skills, statistics, inventory, and reputation were the four core screens that were needed to make the game function.

For the skills screen, the basic functionalities of the screen were to show each skill, allow the player to increase skills with points earned through experience gain, and to get more detailed descriptions of them. Whenever the player moves their mouse over a skill’s area, whether the button to increase it or just the text and icon associated with it, the right half of the skills screen shows the skill’s description and other functions of it (done by making the entire section for that skill a button that has no action with the ‘hovered’ attribute, since only buttons can have it). The statistics screen eventually made its way into the skill screen as well, though without a hover function for now. Cutting the screen into pieces allowed me to utilize space more effectively, while still keeping it organized and visually distinct.



Deception	0	Untrained
Fencing	0	Untrained
Marksmanship	0	Untrained
Engineering	0	Untrained
Willpower	0	Untrained
Courtesy	0	Untrained
Alchemy	0	Untrained
Thievery	0	Untrained
Riding	0	Untrained

Fencing
Untrained
Total bonus:
0

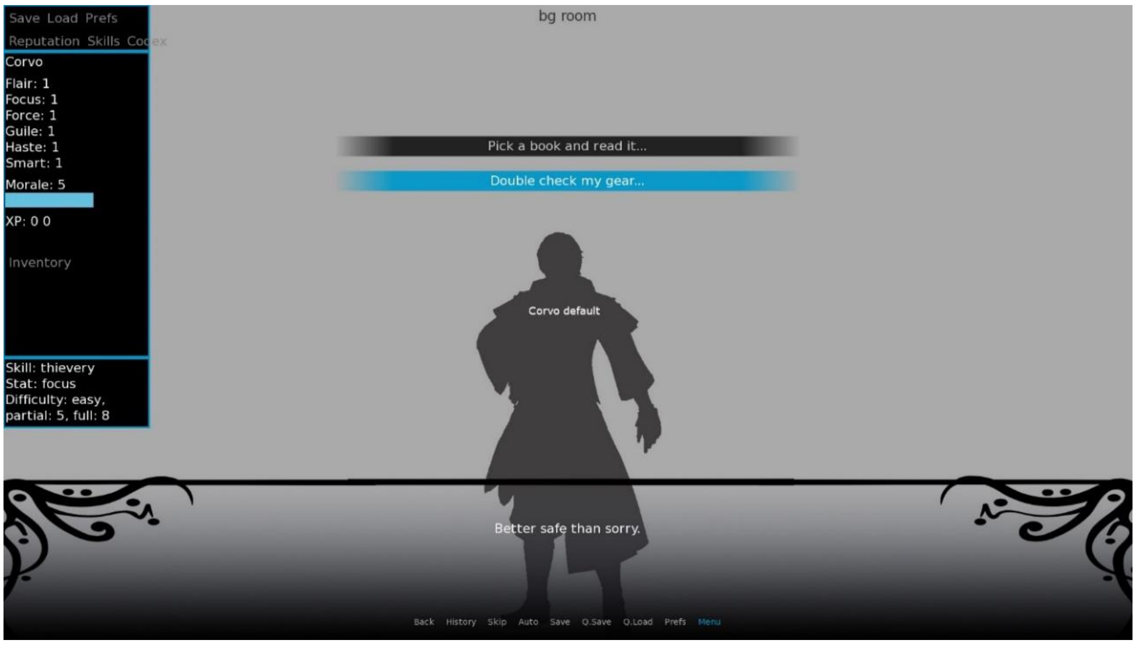
Your skill and prowess with the blade and other hand to hand weapons, including fists.

Force	1
Guile	1
Haste	1
Focus	1
Smart	1
Flair	1

Figure 6: The skills screen. Though hard to see, there are + buttons that can be used to increase skills and statistics. Mousing over each skill also highlights that skill in the area in the top right.

For the reputation screen, the screen was split into two separate parts. One for characters, and another for factions, something new that was created during the development process on a whim, but functionally similar to character relationships. For now, just showing the numerical values for the various reputation tracks and the text rank is fine, but eventually a better sliding position on various axis would be ideal. A small icon is also needed to show each character and faction visually for a nicer and clearer experience. Something to improve upon later, but for now the prototype was in place.

To make communication more transparent when making choices, I wanted to have an extra line of dialogue show up when the player mouses over an option when presented with choices. Initially the plan was to put this into the dialogue box, but engine limitations prevented this, so I was forced to come up with a different solution. Instead of creating a new line of dialogue, I hacked together an identical facsimile of the dialogue box which can be modified and show it only when the player's mouse hovers over a choice. While hardly optimal, it is a hack that works to its intended use, and as such is seems like a good enough solution for a problem that I cannot overcome in another way.



Corvo
Flair: 1
Focus: 1
Force: 1
Guile: 1
Haste: 1
Smart: 1
Morale: 5
XP: 0 0

Skill: thievery
Stat: focus
Difficulty: easy,
partial: 5, full: 8

Better safe than sorry.

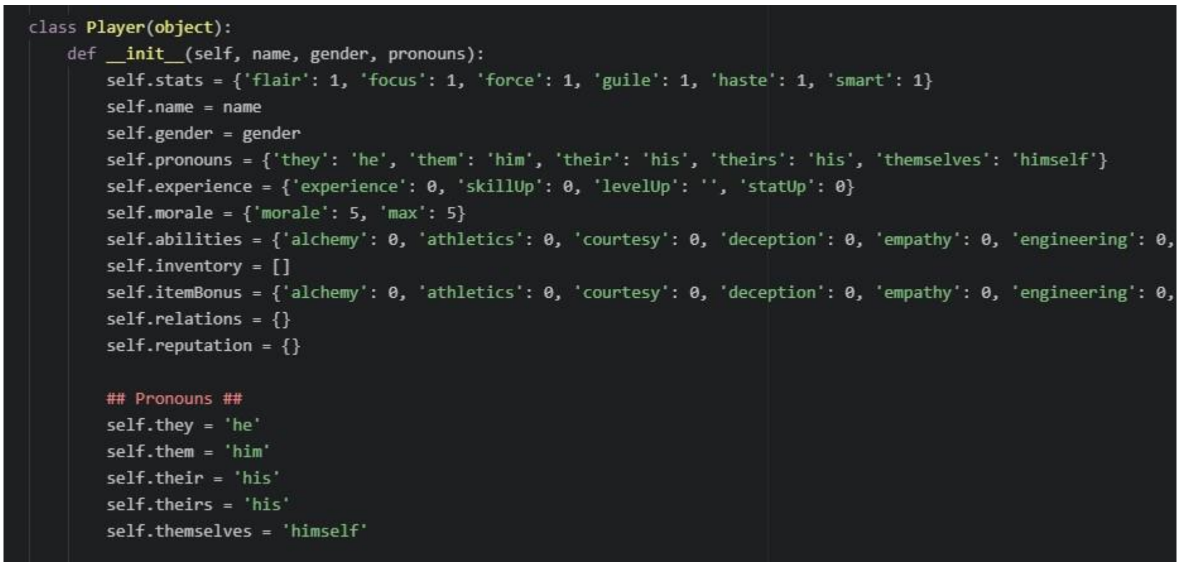
Figure 7: Mousing over a choice. The dialogue box changes when mousing over, as does the information of the skill check in the box underneath the statistics.

4.2 Character mechanics

Character mechanics was a relatively simple process to create. What I needed was a class for a playable character, the variables that go into it, and then finally functions to change those variables in a manner that aligns with the overall design. I could have done without a separate class for the player character, but in case I wished to have several playable characters in the future I wanted to have the option to have several characters to swap between and to modify at once, even if the UI only ever worked for the current playable character (with the variable player, changed every time the playable character changes).

Instead of opting for several dozens of individual variables within the Player() class, dictionaries were created for any statistics groups that could be put together. Eventually, nine variables were created for a playable character, as well as nine functions to change them (some of the variables are utilized elsewhere, like in the inventory which will be detailed later).

If the game is to allow for customizable characters, then some methods to customize it must be given. To this end, functions to change the player character's name, gender, and pronouns (separately from gender, also with a more inclusive 'they' pronoun set) are important, and also simple. For pronouns, each of the possible words (he, his, him, himself, etc) need to be changed individually. A dictionary would have been the most efficient way to organize and list pronouns, but for practical use as the pronouns will need to be referred to very often a dictionary was not viable. Instead, the shortest reference point possible was needed, which ended up being 'self.they' as a reference for he/she/they pronouns, and others separately at the root layer of the Player() class. In order to have several possible characters being referred to at once, assigning them into a global variable was deemed unsuitable and potentially confusing to write. When writing, the phrase "[player.they]" can be used in existing dialogue and text to maximize efficiency and minimize word count, instead of "[player.pronouns['they']]", which is a lot longer and less efficient by comparison. The idea of allowing players to write custom pronouns (the gold standard for inclusive characterization) for their characters did cross my mind but was discarded for now in favour of leanness. If it became something that was widely requested the idea could be revisited.



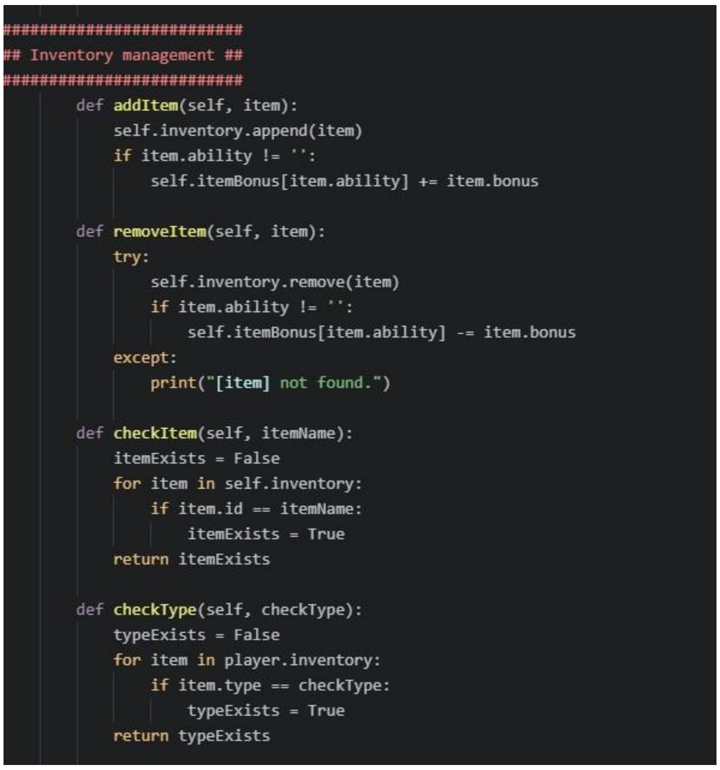
```
class Player(object):
    def __init__(self, name, gender, pronouns):
        self.stats = {'flair': 1, 'focus': 1, 'force': 1, 'guile': 1, 'haste': 1, 'smart': 1}
        self.name = name
        self.gender = gender
        self.pronouns = {'they': 'he', 'them': 'him', 'their': 'his', 'theirs': 'his', 'themselves': 'himself'}
        self.experience = {'experience': 0, 'skillUp': 0, 'levelUp': '', 'statUp': 0}
        self.morale = {'morale': 5, 'max': 5}
        self.abilities = {'alchemy': 0, 'athletics': 0, 'courtesy': 0, 'deception': 0, 'empathy': 0, 'engineering': 0,
        self.inventory = []
        self.itemBonus = {'alchemy': 0, 'athletics': 0, 'courtesy': 0, 'deception': 0, 'empathy': 0, 'engineering': 0,
        self.relations = {}
        self.reputation = {}

        ## Pronouns ##
        self.they = 'he'
        self.them = 'him'
        self.their = 'his'
        self.theirs = 'his'
        self.themselves = 'himself'
```

Figure 8: The Player() object. Full skill list gets cut off.

Several other mechanics were needed to introduce to the Player() class as well, such as the inventory system. In roleplaying games, gathering gear is an important part of progression, so each character had their own inventory to draw from. Functions were created to add items, remove them as well as check if items already existed. Items were created and compiled in their own file and were easy to add to a character by simply using the addItem() function which would take all the data and put it into the inventory array, which in turn appeared on the UI. Likewise, removing an item (and making sure that it does not crash if the item to be removed does not exist) is easily done with just the name of the item.

When an item is received, the game checks if that item exists already to prevent duplicates. If it does not, the item is added into the player's inventory. Some items have a skill bonus attached to them, giving the player extra competency in a specific skill, for example a finely crafted sword would give a +1 to the player's Fencing ability. When such an item is received, the game checks if there is already an existing item bonus (managed in their own dictionary as part of the Player() class) and if the new bonus is higher than the existing bonus to that skill, the new bonus overrides the old bonus. This is to prevent weird situations where a character has three swords and an axe and is somehow able to wield them all to great effect. As a compromise, however, some rare items give a flat bonus to skills as well but are a lot less common. These flat bonuses are handled in their own dictionary and have the item's modifier value added or removed from that dictionary upon acquisition and removal.



```
##########################
## Inventory management ##
##########################
        def addItem(self, item):
            self.inventory.append(item)
            if item.ability != '':
                self.itemBonus[item.ability] += item.bonus

        def removeItem(self, item):
            try:
                self.inventory.remove(item)
                if item.ability != '':
                    self.itemBonus[item.ability] -= item.bonus
            except:
                print("[item] not found.")

        def checkItem(self, itemName):
            itemExists = False
            for item in self.inventory:
                if item.id == itemName:
                    itemExists = True
            return itemExists

        def checkType(self, checkType):
            typeExists = False
            for item in player.inventory:
                if item.type == checkType:
                    typeExists = True
            return typeExists
```

Figure 9: Inventory management

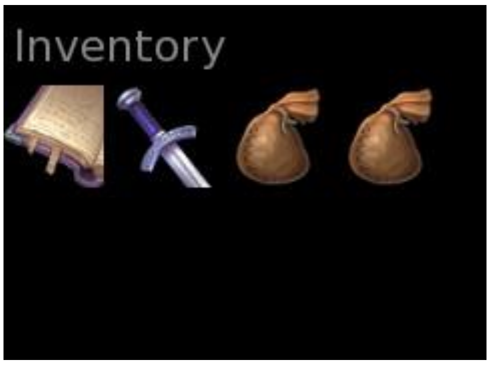


Figure 10: Inventory UI with items

Another core function of the game, skill checks, were also made as part of the player class. Mostly because they rely on the class's statistics and other variables in them. Utilizing the Difficulty() class's definitions (which are initiated upon loading the game to provide the target numbers for skill checks), a long if-else statement is created to get the numbers needed, and then two random numbers are called in a way that makes it impossible to redo the randomization by going back in dialogue (backtracking is a common feature in visual novels). Then the function simply adds in all the bonuses that exist for that skill and statistic combination and returns the result. In addition, experience points are awarded by calling the gainExperience() function and passing in the result (so the amount of experience the player gains depends on the result of the skill). The design allows for a chain reaction of functions that allows the developer to write only one function when needed and to take let the game take care of the rest, while still allowing individual functions to be called if necessary (for example, to reward extra experience upon accomplishing a task).

4.3 Auxiliary mechanics

Many of the auxiliary mechanics are part of other categories, being linked to one or the other. However, putting them in their own chapter makes it easier to talk about them as their own systems, as most of them work fairly independently and are not core to the overall experience, but rather additions and things that are nice to have to provide for the experience the product is aiming for.

The first auxiliary mechanic would be the reputation system. For player choices to be measured in the long term, it is easier to have an abstract scoring system which increases or decreases based on the player's choices in addition to separate choice triggers. This allows for more consequences on how characters react to the player. For character relationships, three parallel score systems were created: Friendship and Respect, each describing how the character feels in that particular way towards the player character. It is important that they are not mutually exclusive. A character might have high Respect but low Friendship, meaning that while they do not see the player as a friend (maybe even seeing them as an enemy), there is a grudging respect towards them for their accomplishments. A high Friendship and high Respect might lead to staunch camaraderie between the characters, one that can survive the strongest of trials.

Some characters can also be romanced, with a Romance Boolean and value attached to a reputation. Certain choices will flip the Boolean to True, and the player can then pursue a romantic relationship with that character. Romantic options in roleplaying games are very common these days, and the idea of dating sims is very common for visual novels as well, so it felt like a natural thing to have on the side, even if it is not something that will have a lot of emphasis put into.

A similar system of reputation (sans romances, naturally) is extended to factions as well, denoting how well a particular faction responds to the player and what their standing is.

Instead of Friendship and Respect, factions have the values Favor and Wrath. Favor represents how well the faction sees the player, and Wrath is how much the player has angered them. Like with characters, the faction reputations are not exclusive, and one can theoretically have high Favor and high Wrath with a faction, leading to a tumultuous reputation of both good and bad.

Regardless of characters or factions, each reputation tracker has also a rank attached to it. This rank is an amalgamation of all the reputation values put together and will be shown as a value to the player to summarize the reputation into one concrete line. It might also be used in the future for significant consequences for characters, though just as often individual values may be used. Ranks are created by having four tiers of relationship: low (0-2), moderate (3-5), high (6-8) and extreme (9-10), with values going no higher than 10.

Reputation and relationships are all character-specific, so in a game with several characters these can all be tracked independently. To make things easier to manage, every time a value is changed the function checks whether or not the target's relationship or reputation tracker exists already, and if it does not a new one is created. This allows for easier management of characters the player might not meet depending on what choices they take, so that the creation of the value happens immediately.

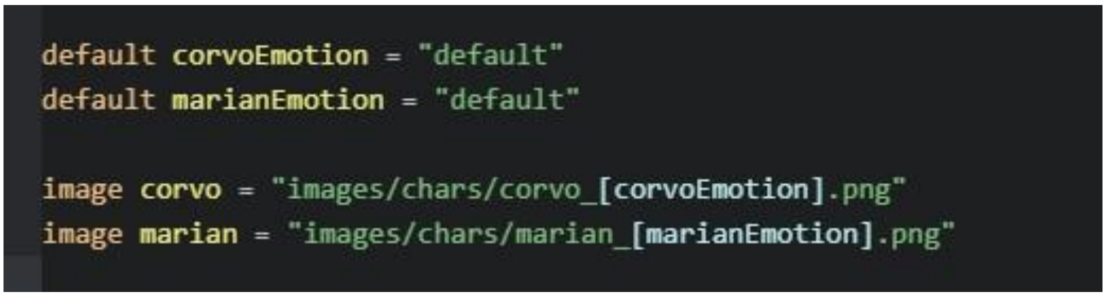
4.4 Writing the narrative

Writing the narrative began with an outline of what the setting is, what happens first and how the player character can react. In this instance the player begins in a room inside a palace, and without anything pressing going on. To get the player to understand the character they are playing and the situation they are in, a brief introductory narration is established where the player's avatar talks about his mission in the palace, why he accepted and how he feels about it. Once this is done, the player can also name the character, with a default name given: Corvo, a reference to the Dishonored games (and the name this character was given when the story was played out as part of the tabletop roleplaying game that eventually spawned the idea of this visual novel adaptation of the story).

Once the player has been eased into the setting and character can the inciting incident of the story begin. Before the player can take any steps towards reaching their goal, a fire breaks out, and the player is given a choice: to help with rescuing people trapped in the fire, or to use it as a distraction to accomplish their goals faster. The latter option might seem callous, but if a player wishes to play such a character, I should give them the opportunity and provide reasonable consequences.

In order to start writing the narrative section for the demo, I also needed some ready-made placeholder assets to use. I opted to use character renders from the Dynasty Warriors series and cover them with grey. The character renders have distinct silhouettes and poses that make them excellent for distinguishing between characters easily, and the grey overlay on top of them is for me to then write in pure words what the emotion of that portrait is. In visual novels it is common for characters to have several portraits with different facial expressions, and in absence of that art I substitute it with simply writing the character and emotion onto the sprite to distinguish between what is going on. For now, imagination will have to do the rest.

One trick I had learned of before for managing several character sprites was to define the image that shows the character with a variable in the image's string. Thus, instead of having several images that all need to be stored and replaced manually, I had one image called (for example) 'corvo_[corvoEmotion]' defined per character, which allowed me to simply change the corvoEmotion variable into whatever is needed, and thus changing what is on the screen immediately. Learning tricks like this to optimize the writing process was key in figuring out how to become a better visual novel developer and learning them early will obviously require less backtracking through legacy code to fix everything into something more streamlined and efficient.



```
default corvoEmotion = "default"
default marianEmotion = "default"

image corvo = "images/chars/corvo_[corvoEmotion].png"
image marian = "images/chars/marian_[marianEmotion].png"
```

Figure 11: Changing emotions this way is a simple but ingenious trick that helps write the story faster and more efficiently.

With ideas for the first scene in place and the art assets secured, it was time to properly start writing it. First off, a refresher of how Ren'Py handles scene commands, showing and moving characters and transitions. After that was done, I tested adding my new character assets into the scene. And realized that they were far too large. Cons of taking high resolution assets from a video game, it seemed. After downscaling all the character art I had to a more reasonable scale (from 2000 pixels to 700) and adding transparency layers to those that had it missing, I was ready to begin.

Creative writing has been a hobby of mine for a long time, and while the medium of a visual novel is new, it is not quite insurmountable. Certain differences need to be made in order to fit the medium, and those are likely to be truly internalized only once more of it is written and experience gained. For now, a small sample narrative with descriptions, a few choices and showcasing of systems is necessary to prototype the product, trying to incorporate all the available mechanics into the system to get a good idea of how they function in the game, though not necessarily quite as a story.

4.5 The cutting board

It is very common for features and content of video games to end up not being released in the end product, as exemplified by an entire website – The Cutting Room Floor – documenting what has been cut in various games. As such, I went into the development process of my own video game knowing full well that not all ideas I had would end up being viable for the game, doable for a single developer, or ultimately all that enjoyable.

However, as this is simply a prototype and proof of concept for a further, larger release, I feel it is important to document future possibilities and what they might bring to the game.

One of the first things to end up on the cutting board was the idea to have a separate dialogue screen where the player could pick their responses in most conversations. The scope of the project and timetable simply did not allow for it, but it is something that I would like to look into later down the line, as the idea itself is fairly interesting and allows for greater player expression, even if these dialogue choices would not be tremendously consequential all the time. It would have made the game closer to feeling like a computer roleplaying game, but for now it should be cut and re-evaluated only once more important things are finished.

5 Discussion

If there is anything I have learned throughout this, it is this: as a solo project, game development is all about momentum. I already knew it is about determination and persistence, and not at all about motivation; this I had learned during other creative processes like when I was writing a novel (a regular one, not visual). But for game development, it is far easier to get stuck in something crucial and there the process grinds to a halt as you spend hours upon hours trying to figure out a solution to a function or component that is absolutely vital to the rest of the game. It is a frustrating feeling, and I know it all too well from several points in this development. But it is important to keep trucking on, despite the frustration. Moving onto different areas, writing down documentation, anything to keep the development process going. All very simple theoretical instructions, but not all that easy to keep in mind when developing something in practice.

As important as simply keeping on with development as an independent developer is also setting the definition of done for the project. It is easy to become a perfectionist when developing on your own, but it is necessary for at least the timeline of the thesis project to be stricter on what is the focus and what is not. I have nobody to please but myself, and I need to learn to be happy with what I have in the capacity I am able to achieve. There will always be time later to go back and fix things where my first (or second, or third) attempts were lacklustre, but for a thesis I had to know when to quit. Anything and everything I do, even (and especially) failure, is helpful for my own personal development and to that of the project, even if it later has to be iterated upon. As an independent developer, my growth is also my project's growth; and as personal growth is a lifelong pursuit, it would be foolish to attempt to strive for absolute perfection in the timescale I have had. Never let perfect be the enemy of good.

Time management was obviously a big thing with this project. The timescale I had to write the thesis and the game was rather tight, and I needed to make the most of the time I spent, especially as I had a fulltime job as a software developer. I cannot say I was always successful in managing my time nor keeping momentum going with the project. However, without anything but self-appointed deadlines and with means to support myself by working at the same time, I never felt like the option of extending deadlines for this project was a massive impediment to its overall quality. As a thesis project, on the other hand, perhaps I might have chosen a too ambitious timeline, and I will admit that the pressure did get to me several times. From a personal perspective, on the other hand, I place higher value on keeping myself free from undue stress and to not overburden myself with tasks. Our working culture is in many ways broken, and video game development especially so (Semuels, 2019), with too much crunch time, overworked employees and a culture that results in burnt out developers. As such, developing products on my own terms and on my own schedules is, to me, a superior option over burning myself out and hating the very product I have been wanting to develop for years now.

Schell writes at the beginning of his book that game designers need to know many disciplines to be successful (pp. 3-7) and having now myself gone through much of the groundwork of creating a video game it is becoming more apparent the longer I go. Even in this short amount of time, I have had to consider many different elements, from statistics and probability to user experience to art design to even how to draw proper flowcharts. And this is all without even touching on the topics of marketing and actually producing artwork for the project and everything that goes into those topics (do these two art styles work together? Are these characters wearing consistently themed outfits? What is and is not historically appropriate for the rough level of technology there is in the setting? When were windmill invented and can I have one in the background? These are all questions I have had to face when designing settings before). Thankfully, creating the game in a fantastical setting instead of a historical one allows me some leeway with historical facts, but having a consistent setting is still useful. Regardless, there are several disciplines where my skills are not sufficient, so if this game were to be finished, I would need a team to fill in the gaps of my own ability.

Perhaps above all else, above momentum and above multidisciplinary approaches, the process of game development is fundamentally at its core about problem solving. There were countless problems faced during development, from UI functions to engine limitations to simply where to get icons and placeholder artwork. Some of the problems came from learning Python and Ren'Py and how they function. Other problems came from the limitations of Ren'Py and solutions from how to circumvent the limitations by use of clever hacks and shortcuts, though those often resulted in even more problems that needed to be solved. As a whole the problems often seemed insurmountable, but the key part was to take them one at a time, work through one problem and then the next. Eventually, things began falling into place and what looked like a game began to appear. Looking back, there are not many problems that I was unable to solve on some level at least, even if I was forced to change the requirements somewhat.

5.1 Further development

The plan was always to continue developing the game into a larger and longer story, and that will no doubt continue now that a lot of the development of the code has been finished and I can focus more on the narrative, which was always the part that truly inspired me more. Without a deadline, however, work on the game will likely be sporadic and subject to inspiration more than anything. Artwork is not going to be made until there is a reasonable amount of the rest of the game to showcase, so as a product itself the game will take a lot longer to be deemed viable. But all this is okay, because my primary goal with the visual novel was creation for the sake of creation and to satisfy my own desires to bring the game into reality, not primarily to sell it. Though if I were able to sell it, I would not be averse to the idea.

With more time, more of the features might be implemented as well. It is hard to say as of yet how many of them will make it in and in what capacity, but more features would be nice to implement down the line, alongside revisions to how the game works. Mechanics should ideally be created as soon as possible, in order to prevent having to go through all the written dialogue and narrative to insert the new mechanics seamlessly.

The principle of writing code in a way that makes it easy to modify or to introduce new things will no doubt pay off once further development is in process, as changing things around is simple and does not require a lot of refactoring code (although if I do end up changing skills or statistics, there will be a lot to go through in the script – but there is little I can do about that).

While the product itself so far has been written in Python and using the Ren'Py engine, the design principles of the mechanics can be easily adapted to any engine or format, be it a classic isometric RPG or a simple visual novel. This is useful as the reason the visual novel format was chosen was not necessarily only for its pros, but the cons of the other formats (namely, difficulty and other artistry needed). But were the circumstances to align themselves to allow for a different engine or style, the design would need only small changes to accommodate for the change.

5.2 Selling the product

Being the sole owner of the product, and as it is being made as if it were a product to be sold, in addition to creating the product I also need to consider how the product will be sold. It is not a topic of huge concern for me at the moment, as the product will not be in a finished state at the end of the thesis, but it never hurts to think ahead anyway.

Before we begin to discuss how it could be sold, we need to list all the features and components needed to make it into a sellable state. The main features are:

- A proper story
- Artwork
- Lack of critical bugs
- Functional mechanics

Having a proper story is one of the two fundamental requirements of a visual novel. Primarily written through dialogue and descriptions rather than gameplay, the story should deliver an experience that the audience wants. In this case, the story is aimed towards a broad demographic of people who enjoy fantasy stories and roleplaying games, both tabletop (like Dungeons & Dragons) and computer roleplaying games (like The Elder Scrolls VI: Skyrim). In addition, many of the themes and design choices are aimed towards an uplifting, progressive worldview, which might broaden the appeal to some people who would otherwise leave it. While there are some groups of people to whom this sort of political leaning would be a negative, I will not lose any sleep over those people not buying my product.

Artwork is the second critical component of a visual novel, as can be surmised by the two words that take up the name of the genre. Because there will be no real artwork for characters or background in the game, external help would have to be acquired to draw up all the missing artwork. For an artist, one could either be hired to work on the project by splitting the revenue (probably 50-50, just to keep things fair and even), or commission pieces from someone for an upfront payment and licensing fee. Several asset stores exist online from which for example background art could be acquired, which would warrant further investigation. To keep the artwork thematically consistent, the number of artists should be kept as low as possible, although more artists might be hired to work on different areas (character art versus backgrounds, splash screens and interface assets).

It should be noted though, that the lack of artwork does not hinder the work in progress in any real way. Placeholder art will be used for all assets that can then easily be replaced when the new art is acquired. This also makes it easy to track what is needed by simply looking at the images folder in the work directory and crossing off the assets that have their final art in place.

When coding it is always helpful to keep the number of bugs to as low as possible. The lack of tremendously complicated code helps limit the existence of critical bugs, but QA throughout the development process should still be handled. As a one-person team, QA would be handled by myself for now, but other beta-readers could be asked to play through the game and see if they find something that does not work.

Having functional mechanics is important, as they are one of the ways the product differentiates itself from other, “simpler” visual novels. Having a fully-fledged set of mechanics that meaningfully affect the gameplay experience is very important to me, and as such care should be put into expanding the mechanics into something that makes the game stand out.

5.2.1 Distribution

The question of where and how to distribute the game warrants a mention as well. If the game is to be distributed non-commercially, the main platform to sell it on would likely be itch.io. It has good support for independent developers and takes a smaller amount of effort and money to have it listed there.

However, if the game is going to be sold for money, Steam is the ideal storefront. While it might drown in the thousands of other games on the store, the client base is over twenty million daily (Steam, 2021), so simply putting it there and paying the marginal fee should be worth it. Besides, making a game and having it sold on Steam is *the* sign of having successfully shipped a game.

For a price point, a modest price would be the go-to option. Depending on the investment into artwork, if the game were to have a price it would fall under roughly 10€ or 15€. The length of the story would affect the end price, as would the general amount of content in it. It is not impossible for me to envision even 25€ for the game, but any higher and I feel like it would not gain traction with players. To sell the game more and to get players interested in the story, a free demo should be made available so they can sample the writing style, artwork and general story and mechanics.

No real marketing budget exists for the game, so buying advertisement space is almost impossible. More grounded advertisement, making posts of various kinds of social media pages, is more possible and likely to happen, finding a core audience in online hobby spaces dedicated to roleplaying games and visual novels.

It could be possible, if highly unlikely, to start a Kickstarter fundraiser for the project to fund artwork. This idea, however, has not been thought about much, and I would need to make a high number of calculations to determine how much funding the game would need and what it would be spent on before it could go any further.

6 Conclusion

As the project owner, it is up to me to determine when a project is finished. This might sound difficult, but with the experience gained in development and project management I have a lot more confidence at the end of the development cycle to be able to say what is good and what still needs work. Everything I have done needs to be taken not as a finished product, but as a steppingstone towards further development. Still, it is important for me to evaluate what I have done from an impartial viewpoint.

Overall, I am satisfied with the results of the mechanics and design, but the UI has unsurprisingly left with some things to be desired. Ren'Py was more difficult to work with in terms of customizing UI than I imagined at first, but with perseverance I managed to create the screens I wished in a manner that makes their further refinement easy. Most importantly, each screen shows the data that I wish it to, even if it is not in a manner that is most optimal. More design is, however, needed to make the UI look and feel better, even if they show everything they need to.

The game mechanics work, and they conform to the design I created for them. Overall, I am very happy with the mechanics, and the way they are tracked, improved, and presented to the player. I feel like this is the most important part of the product, as things like UI and layout can easily be changed at any point during development, while game mechanics require a lot of redoing of items, skill checks existing in the game, and other complications. For game mechanics, I am highly satisfied. Very little iteration on the core mechanics should be needed going forward, and everything that is in the game feels like it belongs, which is precisely what I set out to do.

For a prototype project where art is not the focus, the artwork, and assets I have work for their purposes. I am particularly happy with the temporary portraits from Dynasty Warriors, as they allow me to distinguish between characters and their expressions with ease. The other free assets, especially for the UI, are good and give the impression that the game is shaping up to be something more than just a prototype, which was very helpful for development morale. Finally, the icons I have for skills and items are probably going to make it all the way to the final product (and are also the only commercially licensed assets I have, bought from a bundle sale).

All things considered, it is important to note that I have created the framework I intended to, even if some parts are still incomplete compared to what their final design might look like. Art and UI are not as important right now as functionality takes precedence, and the key takeaway to all of this is simple: I am a developer, and with enough time and effort, I can accomplish this and a whole lot more.

7 References

7.1 Works cited

Anime News Network, 2006. *AMN and Anime Advanced Announce Anime Game Demo Downloads.* [Online]
Available at: http://www.animenewsnetwork.com/pressrelease.php?id=1510

Baker, V., 2010. *Apocalypse World.* s.l.:Lumpley Games.

Ciesla, R., 2019. *Game Development with Ren'Py: Introduction to Visual Novel Games Using Ren'Py, TyranoBuilder, and Twine.* 1 ed. Berkeley(CA): Apress.

Evil Hat Productions, 2013. *Fate Accelerated Edition.* Silver Spring: Evil Hat Productions.

Hunicke, R., LeBlanc, M. & Zubek, R., 2004. *MDA: A formal approach to game design and game research..* s.l.:s.n.

Nelson, W., 2021. *Best practices for designing an effective user interface.* [Online]
Available at: https://www.gamesindustry.biz/articles/2021-03-04-best-practices-for-designing-an-effective-video-game-ui

Owlcat Studios, 2018. *Pathfinder: Kingmaker,* s.l.: Deep Silver.

Ren'py, n.d. *Why Ren'Py.* [Online]
Available at: https://www.renpy.org/why.html

Samoylenko, M., 2018. *Five pillars of Immersive Sims.* [Online]
Available at: https://maximsamoylenko.medium.com/five-pillars-of-immersive-sims-7263167e7258
[Accessed 9 December 2021].

Schell, J., 2020. *The Art of Game Design: a Book of Lenses.* 3rd edition ed. Boca Raton: CRC Press.

Semuels, A., 2019. *Video Game Creators Are Burned Out and Desperate for Change.* [Online]
Available at: https://time.com/5603329/e3-video-game-creators-union/

Sir-Tech Canada, 2001. *Wizardry 8,* s.l.: Night Dive Studios.

Steam, 2021. *Game and Player Statistics.* [Online]
Available at: https://store.steampowered.com/stats/Steam-Game-and-Player-Statistics
[Accessed 8 November 2021].

Team Eleven Eleven, 2015. *SC2VN,* s.l.: Team Eleven Eleven.

Various contributors, 2021. *Cutting Room Floor.* [Online]
Available at: https://tcrf.net/The_Cutting_Room_Floor

Visual Novel Maker, n.d. *Visual Novel Maker.* [Online]
Available at: https://visualnovelmaker.com/

ZA/UM, 2019. *Disco Elysium,* s.l.: ZA/UM.

7.2 Assets used

LunaLucid, 2020. *Magical Template*. Available at: https://lunalucid.itch.io/

Fylgjur, 2020. *Magic GUI Elements.* Available at: http://fylgjur.deviantart.com/

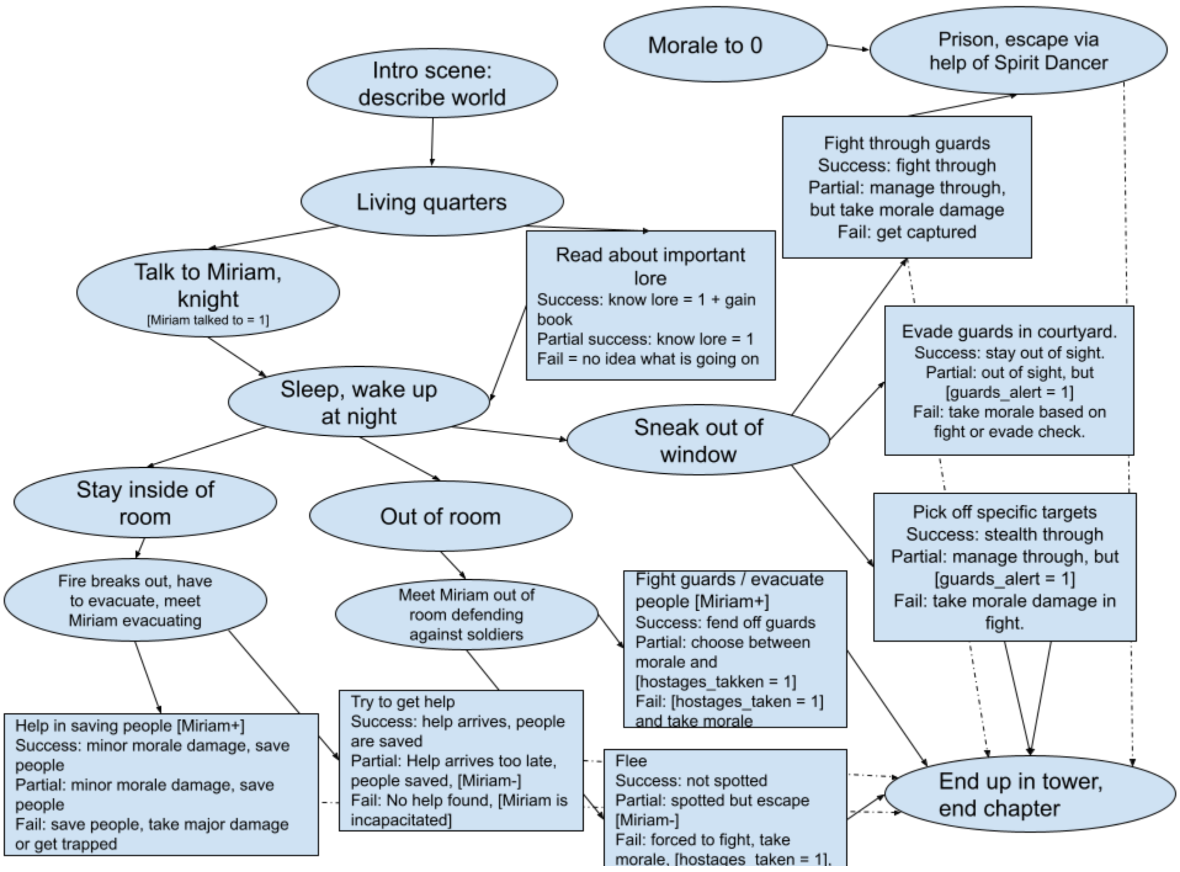
Omega Force, 2014. *Dynasty Warriors 8*. Tecmo Koei.

REXARD, 2020. *Fantasy Weapons Icons.*

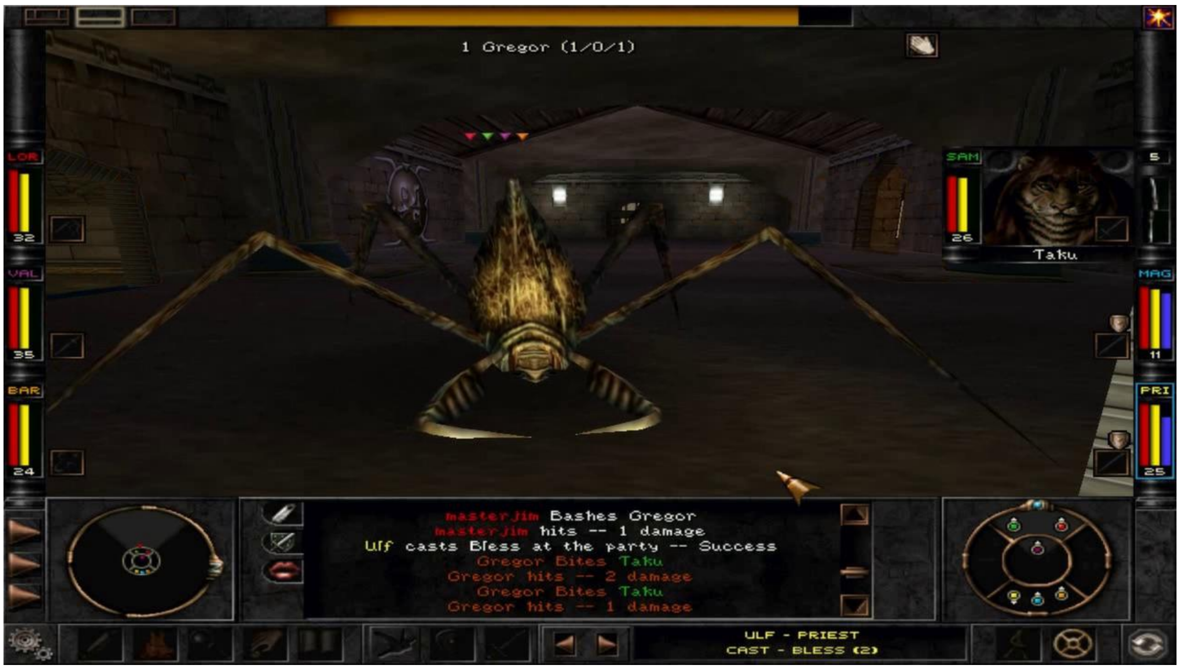
REXARD, 2020*. Flat Skills Icons.*

Appendices

Appendix 1. Narrative design chart

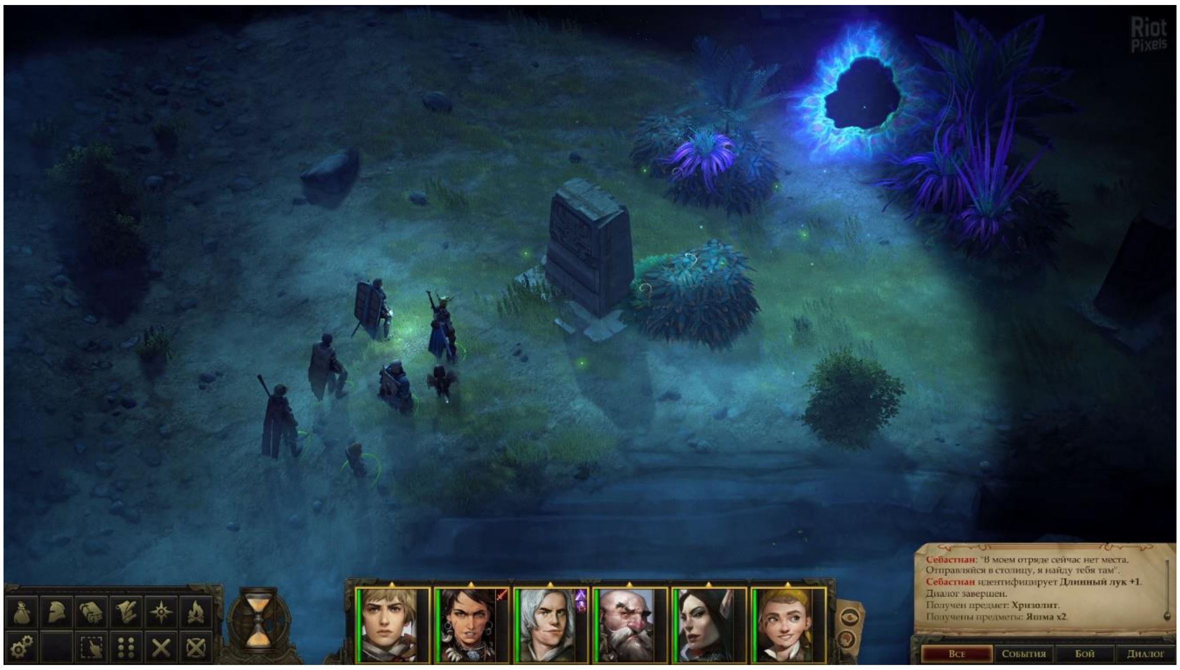

Morale to 0
Prison, escape via help of Spirit Dancer
Intro scene: describe world
Fight through guards Success: fight through Partial: manage through, but take morale damage Fail: get captured
Living quarters
Read about important lore Success: know lore = 1 + gain book Partial success: know lore = 1 Fail = no idea what is going on
Talk to Miriam, knight [Miriam talked to = 1]
Evade guards in courtyard. Success: stay out of sight. Partial: out of sight, but [guards_alert = 1] Fail: take morale based on fight or evade check.
Sleep, wake up at night
Sneak out of window
Stay inside of room
Out of room
Pick off specific targets Success: stealth through Partial: manage through, but [guards_alert = 1] Fail: take morale damage in fight.
Fire breaks out, have to evacuate, meet Miriam evacuating
Meet Miriam out of room defending against soldiers
Fight guards / evacuate people [Miriam+] Success: fend off guards Partial: choose between morale and [hostages_takken = 1] Fail: [hostages_taken = 1] and take morale
Help in saving people [Miriam+] Success: minor morale damage, save people Partial: minor morale damage, save people Fail: save people, take major damage or get trapped
Try to get help Success: help arrives, people are saved Partial: Help arrives too late, people saved, [Miriam-] Fail: No help found, [Miriam is incapacitated]
Flee Success: not spotted Partial: spotted but escape [Miriam-] Fail: forced to fight, take morale, [hostages_taken = 1],
End up in tower, end chapter

Appendix 2. Games that provided UI inspiration



```
masterjim Bashes Gregor
masterjim hits -- 1 damage
Ulf casts Bless at the party -- Success
Gregor Bites Taku
Gregor hits -- 2 damage
Gregor Bites Taku
Gregor hits -- 1 damage
```

Figure 12: Wizardry 8 (Sir-Tech Canada, 2001) has a blocky style that might be old, but has an old-school charm to it



Себастиан: "В моем отряде сейчас нет места. Отправляйся в столицу, я найду тебя там".
Себастиан идентифицирует **Длинный лук +1**.
Диалог завершен.
Получен предмет: **Хризолит**.
Получены предметы: **Яшма x2**.

Figure 13: Pathfinder: Kingmaker (Owlcat Studios, 2018) has a minimalistic menu buttons by default, easily put into the corner out of the way.