ESCAPE ROOM GAME DESIGN DOCUMENT

Table of Contents

Overview

Project Fata Mundi (PFM) is an escape room visual novel where the player completes a series of escape room-style puzzle collections in order to progress through a non-linear branching story, akin to similar titles in the space such as Zero Escape and Head as Code.

The gameplay takes place against a backdrop of a story following a viewpoint protagonist, Kal, who travels to an out-of-the-way underground facility in search of her missing partner and is herself kidnapped by an unknown assailant. The story concerns itself primarily with Kal's attempt to recover her partner and abscond, while learning the true nature of the environment she finds herself in.

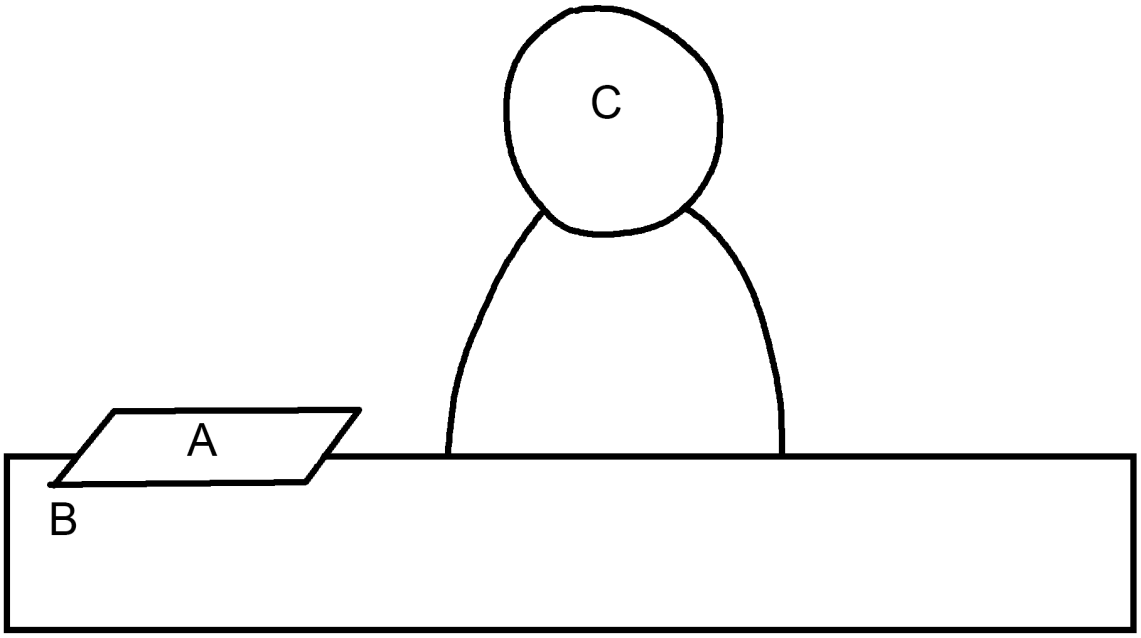
The game itself is composed of two main parts: the escape room segments, during which the player engages in point-and-click style gameplay to collect and combine items, solve puzzles, and escape rooms, and the story segments, where the player observes character dialogue, receives story information, and makes choices that affect the trajectory of the story.

Escape room segments

Story segments

Dialogue manager

The Dialogue Manager is the system responsible for handling dialogue and character sprites in PFM. It ingests dialogue and character data and represents it on the screen as below:



where **A** is a box containing the name of the character, **B** is the dialogue being spoken by the character, and **C** is a sprite denoting the character.

Only one character is ever on screen at a time. Other characters can speak from off-screen, where the character on-screen is still the narrative focus or where the character being off-screen serves some other narrative end, but if they should be displayed on screen, the previous character should disappear.

Ordinarily, characters fade in and out at a linear rate when taking their place in the middle of the screen. This should be controlled by a shader, via shader graph. In certain situations, it would be more expedient for a new character to appear immediately, and the relevant dialogue should include an override which implements this.

When a new string character appears in the dialogue (**B** above), the system should play a bloop through FMOD that is pitched depending on the speaking character's gender.

If the player enters the pause menu while dialogue is active, that dialogue pauses. It resumes playing at the rate it was at when the player paused the game when the player unpauses. If the dialogue was complete, entering the pause menu prevents the player from clicking or inputting to move forward.

The only things the dialogue system controls directly are the sprites, the dialogue box and the contents thereof - the background, music and sound effects are handled by different systems, which the dialogue manager calls when narratively expedient.

Conversation, Character and Dialogue objects

The dialogue infrastructure in PFM is composed, in addition to the dialogue manager, of three broad types of object - conversations, characters, and dialogues. Conversations are themselves batches of dialogue strung together with additional context data.

Characters

Characters are objects that contain data on speaking characters. This is distinct from other aggregate narrative data held about character bios, which are maintained by a different system.

Character objects are called by the dialogue system to obtain information about the current character speaking and should therefore be prefabs containing the following data:

- The name of the speaking character (though this can be overridden in the Dialogue object if necessary for narrative purposes);
- The gender of the speaking character, to control the pitch of text bloops that play per character;
- An animator that contains the animation infrastructure required to respond to emotion calls from the dialogue system.

Dialogue

Dialogue are individual lines of dialogue, that should contain the following data:

- A Character object, to pass along information about the speaking character to the system (or, if the character already on screen is still speaking, a bool to indicate the current character is still speaking);
- An emotion enum to dictate what emotion the character speaking should display;
- A string containing the actual dialogue spoken by the character;
- An ID, so that the system can detect whether the dialogue has been read before and permit the player to skip it if they have;
- A boolean indicating that the dialogue is unskippable, even if seen before, defaulting to false;
- A boolean indicating that the dialogue should be voiced rather than blooped, which should reveal a field for an FMOD event to play while the dialogue is running;
- A list of in-text events to play at certain points:
    - UPDATE_SPEED means the rate at which characters are displayed should change from the default. This does not update the global speed, which resumes from the next dialogue;
    - CHANGE_COLOR means the colour of the text enclosed in the initiating # and the following #, should be a certain colour;
    - SCREEN_SHAKE means the background should shake at a variable speed;
    - MUSIC_STOP means the music should stop:

            - IMMEDIATE means the music should stop immediately;
            - FADE_OUT means the music should stop, observing the fade-out time in the relevant FMOD event.
        - BACKGROUND_UPDATE means the background should update in some way:
            - FADE_TO_BLACK means the background should simply fade to black. It should posterise a little as it does so (because I think that's fun);
            - FADE_ACROSS means the background should cross-dissolve to another background;
            - FADE_TO_WHITE means the background should fade to white.

Dialogue strings can contain small commands, indicated by a #, that can instruct the dialogue to perform the above-mentioned in-text events.

Conversations

Conversations are bundles of dialogue and other events that, when played end to end, create character interactions in PFM. Conversations are scriptable objects that can be plugged into the dialogue system and run. Conversations should contain the following data:

- A conversation ID for internal reference;
- A list of event objects, composed of the following:
    - An enum dictating what kind of event this is:
        - DIALOGUE means this event is a dialogue, and the fields of a dialogue object should be visible to set;
        - BACKGROUND_UPDATE means this event is an update to the background:
            - FADE_TO_BLACK means the background should simply fade to black. It should posterise a little as it does so (because I think that's fun);
            - FADE_ACROSS means the background should cross-dissolve to another background;
            - FADE_TO_WHITE means the background should fade to white.
    - When to perform the event:
        - IMMEDIATELY means the event should happen as soon as the event is reached;
        - DELAY means the event should wait for a number of seconds before performing its contents;
        - AT_CHARACTER means the event should play once a certain number of characters in the next dialogue have been displayed.
- An enum dictating what should happen when the conversation ends:
    - START_ESCAPE_ROOM means the player should transition to the escape room system to start the escape room of the given ID (this will also reveal a field to enter the ID of the escape room desired);

    - MAKE_CHOICE means the player should be prompted with a choice, each of which should have its own consequence (this will also reveal fields for the choices desired and their consequences);
    - PLAY_ENDING means the player should see the game over screen for one of the game endings (this will also reveal a field to enter the ID of the ending desired - entering the bad ending ID or entering no ID will prompt the system to display the bad ending).

Character emotions

Characters in PFM have a set collection of ‘emotions’ - animations that display when a character is idling or talking in front of the character, that are modified to display different emotional states for the purpose of narrative communication.

A prefab should be created for every character with an animation graph that travels between different idle and talking states depending on whether the character should currently be talking (i.e. if the dialogue on screen is not complete) and what emotion they should be experiencing.

Characters can experience the following emotions:

- Neutral: the character’s face is at its default resting state. No indication of any passionate emotion in any direction;
- Happy: the character is smiling, smirking, or grinning (depending on their character; some characters may feel positive emotions more significantly than others);
- Angry: the character is grimacing, has their teeth clenched, eyebrows furrowed, fists clenched, shoulders forward as though to attack, etc.
- Scared: the character has their eyes wide, pupils dilated, mouth slightly agape, shoulders back as though to flee, shivering etc.
- Bashful: the character is blushing, eyes averted, grimacing, etc.
- Sad: the character is downcast, eyes half-closed, teary-eyed, etc.
- Shocked: the character has their eyes wide, pupils dilated - but the expression is of surprise rather than caused by a fear response;
- Impressed: the character has a single eyebrow raised, their mouth slightly raised on one side, head tipped, etc.

Optimally it should be straightforward to add new emotions, though the complications and static nature of the Unity animation graph make this hard to imagine.

Choices

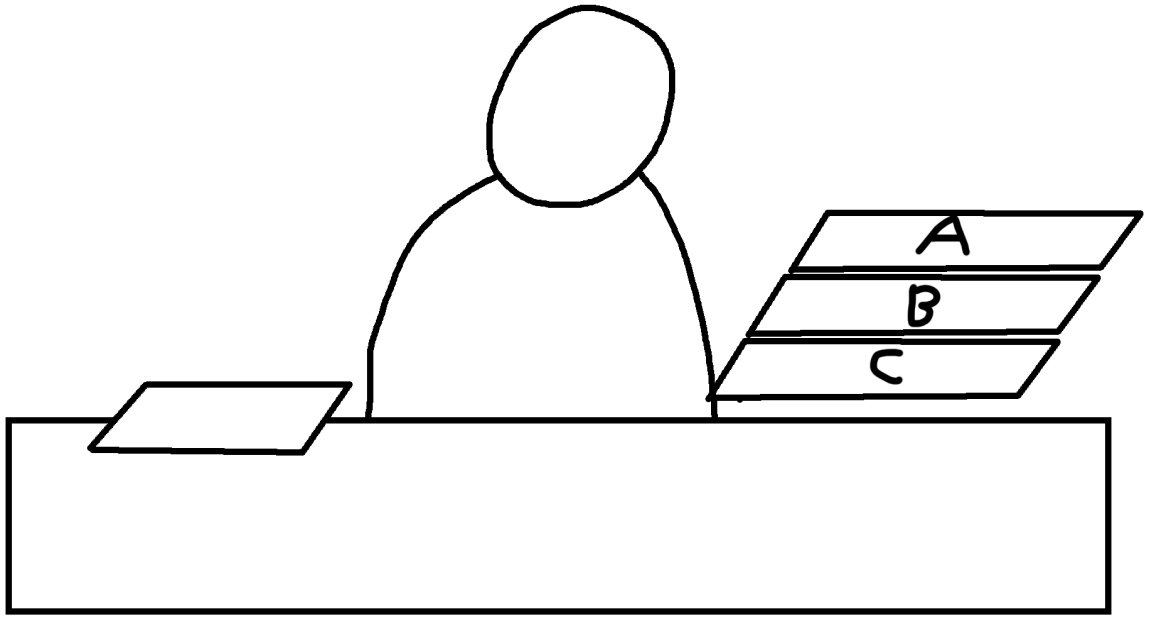
Over the course of the game, the player will be prompted to make certain choices. There are three types, listed in order of how consequential they are:

- White choices: these are choices that do not change the course of the story. They are, for instance, conversation topics when speaking to other characters in escape rooms, or different flavour text for moving along a conversation while in story mode;

- Grey choices: these are choices that change the course of the story mutably - in the sense that the player can go back and make a different choice later. These are, for instance, choices that affect which path down the flowchart the player takes (for now; they can go back and choose the other path(s) later);
- Red choices: these are choices that change the course of the story immutably. Once chosen, the player cannot go back and make a different choice later until the end of the game. These choices do not affect the player's path down the flowchart, but do affect which of the endings the player has access to at the end of the game. If the player returns to this point in the story, the story will immediately progress as though the player already made the choice they made before.
    - Players may remake red choices once they have unlocked the true ending.

While grey or red choices are on screen, the player should not be able to pause the game.

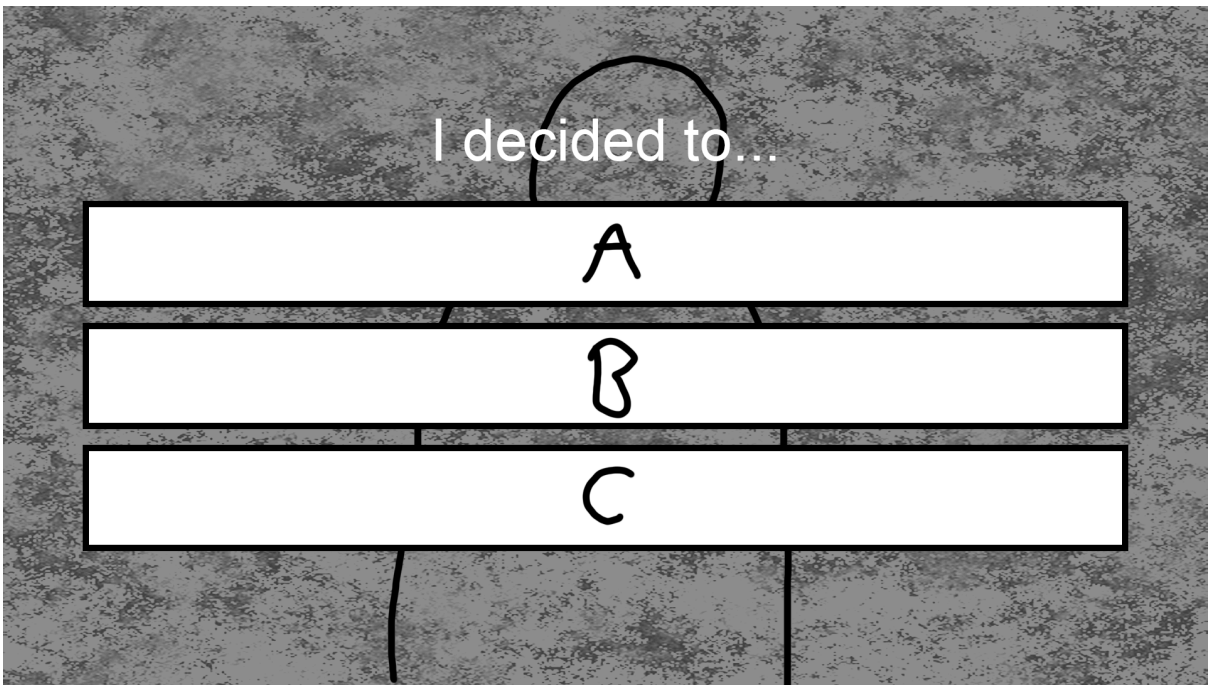
Choices should be displayed differently based on how consequential they are. White choices should appear as follows:



where **A**, **B**, and **C** represent the available choices. The UI should scale to accommodate the number of choices requested in design automatically.

Where white choices are used to represent topics of discussion in an escape room, the last choice should always be some form of "Back" or "[End conversation]" to enable the player to get out of the conversation and return to gameplay. The conversation should return to the list of discussion topics at the end of every conversation thread.

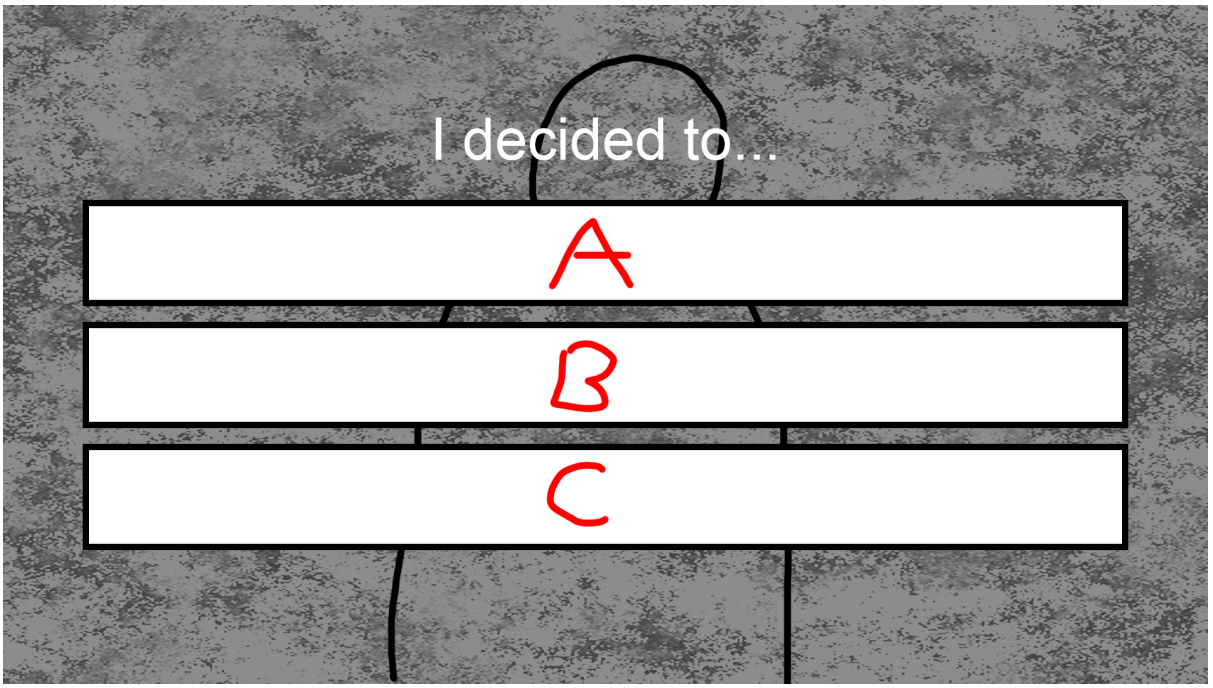
Grey choices, which affect the course of the story, should be displayed as follows:



where **A**, **B**, and **C** represent the choices available to the player. There are not always three; sometimes the choice is binary (e.g. choosing which of two paths to go down to make progress).

Grey choices should only be used where the choice causes a branching path on the flowchart, or where the choice is otherwise of sufficient consequence to warrant being taken more seriously.

Red choices should be displayed similarly to grey choices, but as follows:



Red choices should be distinguished from grey choices in the following ways:

- The text of the choices should be red;
- The shimmering noise background should be harsher, a different colour, or otherwise more sinister looking;
- The musical or sound accompaniment should be darker, scarier or otherwise dramatic in such a way as to communicate the significance of the moment.

In addition, the player should be prompted when making a choice that once the choice is made, they are committed to it and cannot go back to change it. The system of red choices should also be explained the first time they appear in the game.

Once a red choice is made, the player cannot go back and make that choice differently until they have completed the true ending.

True ending decision

The decision the player makes at the end of the game is a red choice of a kind, in the sense that once the player makes it, they are on some level committed to it.

The player must decide how to remake the world, and once they have chosen how they want the world to be, their save file is permanently marked with that decision.

The decision should have a unique UI or other display for how the player decides, and should not be communicated as an ordinary red choice.

On the initial decision, the player is told that their decision is permanent and if they return, any subsequent decision is not canon, but rather a glimpse at how the world could have been if they'd chosen differently.

Once the player has committed to a true ending, red choices become available to remake again.

Game state machine

The GameStateMachine (GSM) keeps track of the state the current game is in, and transmits that state to other systems to keep things consistent depending on where the player is and what they are doing.

The GSM can be in one of the following states:

- MAIN_MENU: The player is in this state when in the main menu. Pausing is not available in this state;
- DIALOGUE: The player is reading dialogue in the dialogue system. Pausing is available in this state;
- PAUSED: The player has paused the game in any of the states where pausing is available. When entering the pause state the GSM records the previous state, so that gameplay can resume after the player unpauses;
- ESCAPING: The player is currently in an escape room. Pausing is available in this state;
- CUTSCENE: The player is observing a video cutscene. Pausing is not available in this state;
- CHOOSING: The player is making a grey or more significant choice. Pausing is not available in this state;
- TRUE_ENDING: The player is making the true ending choice. Pausing is not available in this state.

Escape rooms list

Vintage room

Items

- Knife (found in the drawer of the cupboard)
- Lockpicking book (found in the cardboard box on top of the printer)
- Printed sheet (found in the printer)
- Letter (found under lounger)

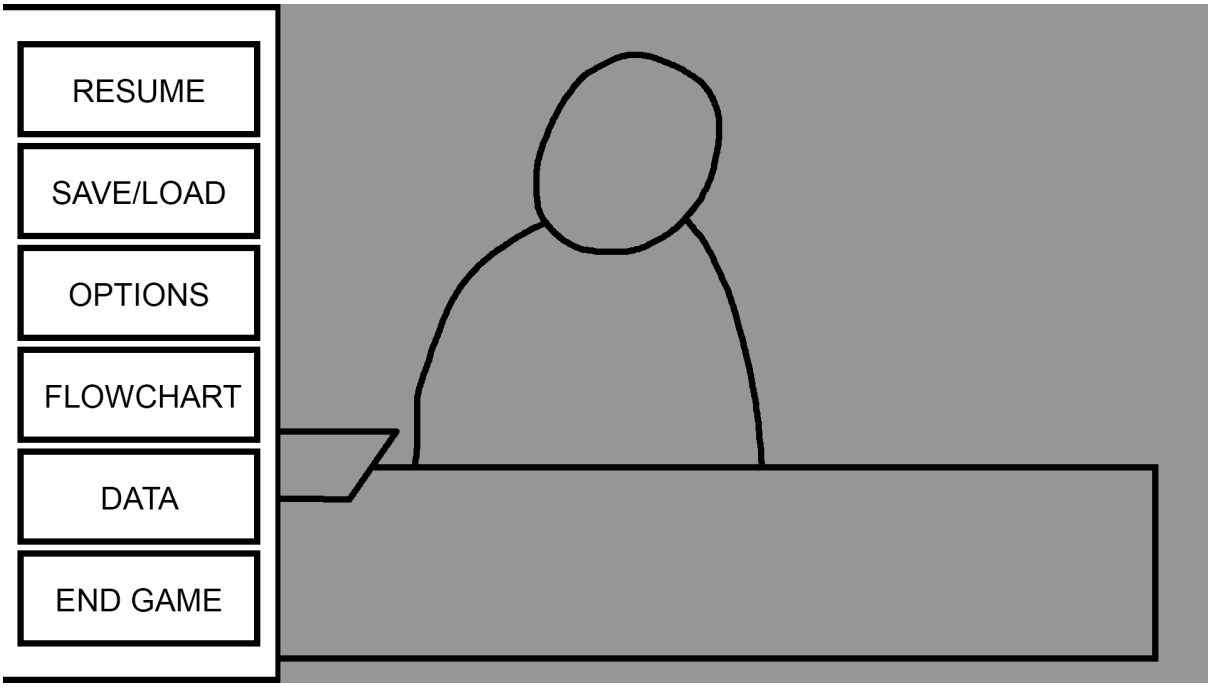
Steps

- Get knife from cupboard
- Get lockpicking book from cardboard box
- Get letter from under lounger
- Complete star battle puzzle on desk to get key
- Use key on globe table to unlock globe
- Use knife on globe to get token
- Use token on wall slot to unlock Corner 1
- Check TV to get TV manual
- Use lockpicking book on door to access lockpicking minigame
- Use TV audio and TV manual to stab the map in the right places to get token
- Use token on wall slot to unlock Corner 2
- Click light switch to get glass ball with key
- Use key to escape room

Pause Menu

Overview

The pause menu appears when the player presses the Escape key (or alternative key binding on other input types). The pause menu should look roughly like this, with the menu and its contents arranged vertically on the left, and a translucent black covering the gameplay:



The pause menu should be accessible:

- During story segments, while in dialogue only (i.e. not during video cutscenes, scene transitions, fades, etc.);
- During escape room sequences, either while talking to a character or while outside of the inventory screen / a puzzle.

The pause menu should not be accessible in the main menu or otherwise outside of the ordinary flow of gameplay.

While the pause menu is active, other gameplay systems should move into the paused state, retaining the state they were in previously so they can return to it when the pause is over.

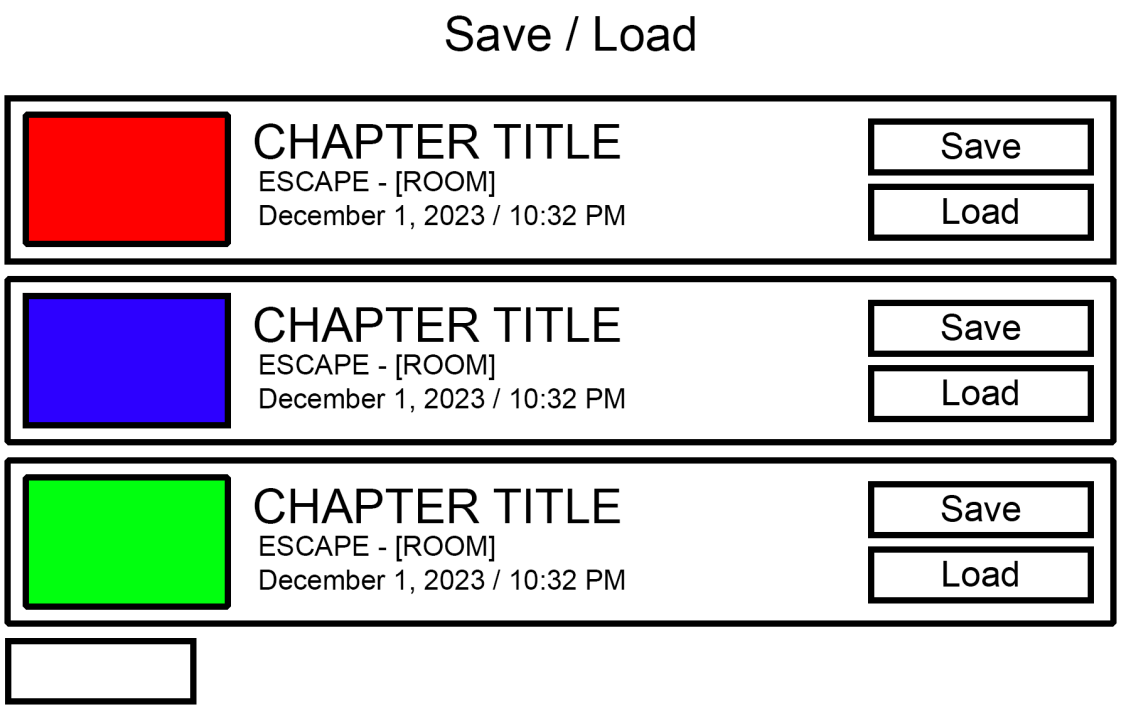
While in the pause menu, the player can view the following options:

- RESUME: This resumes the game from the point it was paused, creating no new UI;
- SAVE/LOAD: This accesses the save/load screen, allowing the player to save their progress or reload their progress from a different file. This is distinct from the flowchart, in that flowchart progress is also contained in the save file;

- FLOWCHART: This accesses the flowchart menu, which enables the player to move to a different point in the story that they have already unlocked;
- OPTIONS: This accesses the option menu, which enables the player to change options such as audio volume, accessibility measures, etc.;
- DATA: This accesses the data menu, which contains information about characters the player has learned over the course of the game. This data updates over the course of the game;
- END GAME: This opens the end game menu, which gives the player the option to quit either to the menu or to desktop.

Save/Load

The Save/Load menu allows the player to save their progress in one of three save slots, arranged as below:



where **CHAPTER TITLE** contains the name of the current chapter, the line reading **ESCAPE - [ROOM]** contains the name of the current event on the flowchart the player is in (containing either ESCAPE - X, where X is the name of the room they are escaping, or otherwise the name of the narrative event the flowchart point is concerned with), and the red, blue and green rectangles to the side contain an image representative of the point in the story the player is in currently. (This image is representative of the point in the flowchart the player has travelled to, rather than representative of where the player may have reached in the story as a whole.)

The rectangle in the bottom left should contain the word “Back”, and returns the player either to the pause menu, if they travelled to the Save/Load menu from pause, or to the main menu, if the player is loading a save. In the latter case, the option to save the game will be greyed out.

Pressing Save on any slot saves the progress of the player in that slot, recording:

- unlocked points on the flowchart;
- any endings the player has achieved;
- the current conversation and dialogue ID the player is in, if any;
- the current state of the escape room the player is in, if they are in one, including:
    - what items the player is carrying;
    - what puzzles have already been solved;
    - what conversation options the player has exhausted with people in the room;
    - what items have already been interacted with, and how many times.
- the current state of the background;
- the current music playing, if any;
- any character data the player has unlocked in the data menu.

If the player presses save on a slot that already contains a save file, the player will be prompted to make sure they intended to overwrite that save. Responding affirmatively to that prompt will delete the existing save file, and replace it with the new one.

Pressing Load on any slot will load the progress of the player in that slot, resuming either a conversation, if the player was having one, or an escape room, if the player was in the process of completing one.

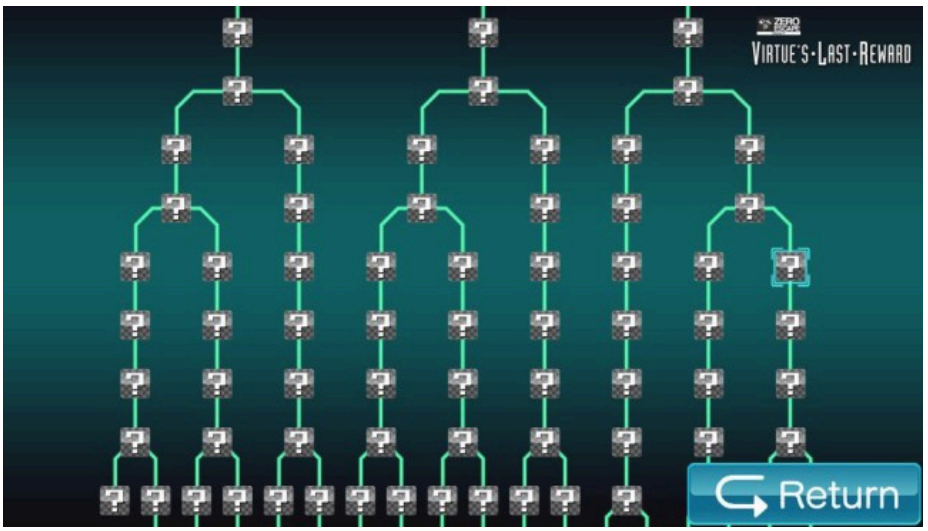
Regardless of if the player is loading from gameplay or the main menu, the game will prompt the player to confirm that they are resuming the correct save file (and optionally will include a message that unsaved data will be lost, if the player is loading from gameplay). Responding affirmatively to that prompt will discard the existing gameplay session, and load either the escape room or dialogue relevant to the new save file.

If a save file is for a completed game - i.e. the player has reached one of the game's true endings - it will be distinguished in some way with the icon of the player's chosen ending. This icon will not change, even if the player goes back and chooses a different ending later.

It is not possible to delete save files from inside the game. The player can delete save files by accessing them directly in Explorer, if they so choose.

Flowchart

At any time during the story, the player can open the Flowchart menu. As the name implies, the Flowchart menu displays a flowchart, starting from the first playable point in the story, and flowing down through the consequent events of the story. See below an example of a similar flowchart from Virtue's Last Reward, with the antecedent events of the story at the top and the consequent ones flowing down as a result of player choice:



Not all events are significant enough to warrant being checkpointed; checkpoints should be placed at:

- The beginning of escape room sequences;
- The beginning of story segments that immediately succeed the completion of an escape room sequence;
- The beginning of story segments that immediately succeed making a choice that affects the trajectory of the story (not all choices meet this standard);
- Points in story segments that are narratively significant enough (i.e., reveal important plot information or critical character or plot revelations) to warrant being broken into their own point in the flowchart.

When in the Flowchart menu, players can use the points in the displayed flowchart to travel to the beginning of any escape room or story event they have unlocked. These events will appear as boxes filled in with simple art denoting what kind of story event it is - escape or narrative - and will have a title and brief description to describe the situation at that moment in the story to remind the player of the context for where they are about to go.

Players unlock boxes on the flowchart by beginning the sequence the box contains - i.e. starting a new story segment or starting a new escape room. Once they have reached this point, they can return to it at any time.

Players cannot use or otherwise interact with boxes on the flowchart that have not been unlocked. The title and brief description will instead read as “???” and the art will be nondescript to indicate this is a part of the story the player has not unlocked yet.

Audio

Overview

Audio in PFM is controlled by an Audio Manager, which interfaces with FMOD to work. All FMOD audio events are broken down into three categories: music, SFX and voice. They are piped through separate lines in the audio mixer to enable the player to set the volume of each category of audio independently from each other.

Music

When started, music will play on loop until stopped. Music events should have loop regions in FMOD, setting how the music should loop, and should have ADHSR controls to fade out when instructed.

Sometimes, it is narratively expedient for music to stop immediately - FMOD has support for this.

If the player enters the pause menu while music is playing, the music should be filtered to sound distant, like it is being heard underwater. This effect ends when the player leaves the pause menu. This effect does not occur in gameplay menus, such as the flowchart.

SFX

When played, SFX will play once and then release. There are no circumstances where looping SFX is desired.

If the player enters the pause menu, any active SFX event should be filtered to sound distant, like it is being heard underwater. The SFX still ends as normal when the sound is completed.

Voice

When played, voice lines will play once and then release. If possible, metadata in the FMOD events for the voice lines would be interesting to control when certain blocks of text appear in dialogue to line up with the actor speaking - this is a stretch goal.

If the player enters the pause menu, any active voice event is paused. The voice resumes playing when the pause menu is exited.