Project Misfits Design Bible

This document centralizes documentation for design decisions made in each domain. It mainly documents gameplay design decisions and UI/UX design decisions, with other domains linking to their respective bibles. If you have any questions, comments, or concerns, reach out to Jeremy King.

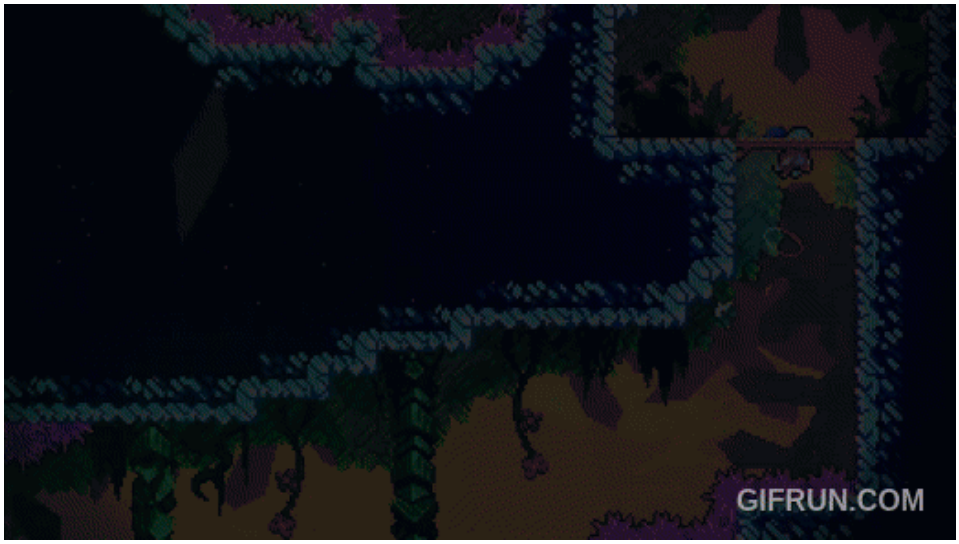
Table of Contents

Player Ability Design

Design documentation

Overall, Player movement will feel responsive, floaty, and fluid. The Leaf Dash should feel fast, evasive, and highly versatile but restricted by resource management; the need to manage the Leaf Meter.

- Overall Movement Design Goals
    - Two modes which chain together that allow for the player to build momentum and use that momentum in any direction they want
    - The player shouldn't be going in one direction at the same speed for a long amount of time.
- Leaf Meter Design Goals:
    - The leaf meter is a meter which drains as the player remains in leaf mode acting as a pressure tool that the player has to preserve while in flight.
    - It returns after a moment of being on the ground in Fenn mode
- Fenn Mode Design Goals
    - Base mode which allows for recharge of the leaf meter and has a more standard style of platforming movement
    - Flow well with the quick movement style of the leaf dash through use of acceleration and deceleration while running and exiting leaf mode.
    - More controlled then the leaf dash allowing for more precise movements
- Leaf Mode Design Goals
    - Fenn transforms into leaves, allowing them to get through small gaps or dash through the air depending on the context.
    - Meant to be fast and harder to control then Fenn Mode but allow for free flight and chaining momentum between Fenn mode and leaf Mode.
    - The player will be launched at the end of the dash allowing the player to reach places the couldn't and chain into more momentum based moments



- Command Companion
    - The current only companion ability is Az's grab in which she uses her magical bangles to grab, push, or pull objects that are marked as grabbable.
    - The purpose of this ability is to create scenarios in which the player needs to exit leaf mode to perform an action emphasizing the feeling of chaining between the two modes
    - Rather than making it just a lever that the hand pushes or pulls we wanted to emphasize Az's strength through this ability.
- Interact
    - Contextual action which "interacts" with a nearby object or NPC.
    - If "interacting" with an NPC, triggers a conversation between the two characters.
    - May also "interact" with levers, keys, and other objects.

Obstacles and Threats Design

Design documentation:

Overall, the obstacles and threats the player will face with Fenn pose a threat to one of three things: platforming progress, Fenn's health, or Fenn's Leaf Meter.

Goals for Enemies:

- Should act more like obstacles
- Player should not be expected to directly fight enemies

Goals for Companion:

- Companion should allow the player to 'combo' with whatever they are doing to effect the environment and obstacles
- Should be made clear the companion is only able to affect things due to who they are (i.e Nothing that's just flipping levers).

Goals for Environmental Obstacles:

- Should feel very rhythmic
- Should reward players for taking more dangerous paths
- Should be dynamic/not repetitive
- Should play off the players main ability to turn into leaves

Companion Abilities:
Bangles of Reach:

- Az has a pair of magical bangles that she wears with that can as portals to one another. This means that while she still has a bangle the other can act like an extension of her arm.
- She gives Fenn one of her bangles allowing him to summon of her arms to hold something in place
- For example, holding a swinging platform in place to allow the player to use it as a platform

Environmental obstacles or tools that actively interact with the player

- Zones that prevent the player from being able to go into leaf mode

- Zones that give the player infinite leaf meter

- Pipes that when interacted with in leaf mode will send the player to another location and launch the player at the end

- Spikes that hurt the player both in leaf mode and non leafmode

- Ice Cube charges at the player, they will stop charging once they get to where the player was.

Level and Room Design

Level Types

- All levels are areas of the kingdom's castle, the Bay Leaf Palace.
- The first level contains the Council Chamber and the Great Hall. The Council Chamber is where the knights are sealed, so the player cannot actually enter it. The Great Hall is the area where the player first gets control of Fenn, where can speak to NPCs about what's going on, where the player meets Az, where the entrances to gameplay-oriented levels are, and so on. For our purposes, this level is simply referred to as the Great Hall.
- The first gameplay level is called the Cellar.

Room Types

- Zero challenge rooms
    - Connecting hallways/areas
    - Respite rooms (like benches in Hollow Knight)
- Low challenge rooms
    - Minimal platforming and/or enemies that pose a small threat to the player
- Medium challenge rooms
    - The player must use their agility and abilities to overcome platforming and/or enemies that pose a direct threat to the player
- High challenge rooms
    - The player must be comfortable with the various abilities at their disposal to overcome a major platforming, enemy, or other threat
- Special challenge rooms
    - The player must use their abilities to overcome a challenge that furthers the story/quest
    - Difficulty is rated Medium to High
    - For example, using their platforming abilities to scale slippery platforms and pull a lever, turning on a heater
- Boss room
    - The player must use all of their abilities and all of the knowledge gained facing challenges throughout the level to escape a final test of skill
    - Also a challenge that furthers the story/quest (naturally)

Example rooms:

- A long room where the player must build up their Leaf Meter and use it to cross a wide pit of spikes on the floor
- A tall room where the player must build up their Leaf Meter and use it to maneuver vertically and reach a high ledge

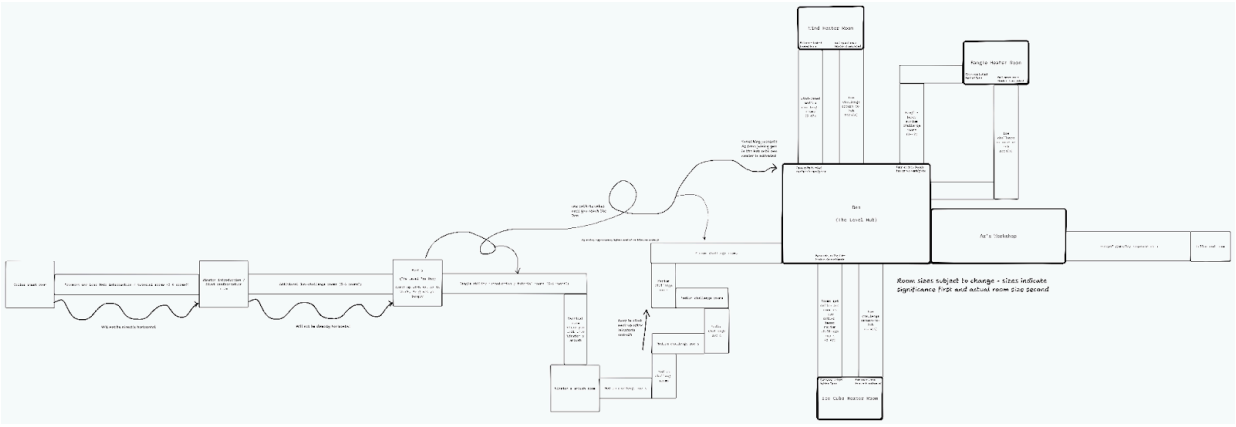
- A closed-in, donut-shaped room where an enemy moves around in a circle and the player has to navigate around the donut shape to avoid it

Level Structure, Layout, and other Details

- The structure and layout of the level strongly correlates with the level's narrative structure, which is described in more detail in the Narrative Outline here**.** A level design-oriented overview is also provided below.
- There will be no branching paths in the level. The only time the level will branch is when the player reaches the level hub and must enable each of the three heaters.
- Rooms will be navigated between via doors the player presses a button to interact with, as opposed to screen transitions you see in games like Hollow Knight.
- Rooms will only ever have one entrance and one exit. Specific level layout related rooms (like the GreatHall Hub and the Cellar Hub) will have more than two doors. Speak with Jeremy King if you are working with a room that needs more than two doors.
- There will be no hidden/destructible walls in the level. All traversable areas should not be obscured. That isn't to say that secrets can't be included; they just need to be in corners of existing rooms and make use of the game's existing obstacles and threats rather than requiring the player Leaf Dash into a random wall that's actually fake.
- Rooms should gradually increase in challenge. The first rooms will be zero challenge rooms to get the player accustomed to the movement and abilities, with the latest rooms primarily being medium and high challenge rooms.
- The level will follow kishotenketsu, where mechanics are presented to the player in the following stages: introduce, develop, twist on concept, and master.
    - The rooms between the level hub and the heater rooms will function as the twist on concept challenges and the heater rooms will function as master challenges.
    - Mechanics can follow the kishotenketsu method at different paces as needed. For example, enemies will not be introduced to the player immediately. They will show up once the player has completed a couple rooms that introduce and develop the movement mechanics of platforming and Leaf Mode. Same goes for the Claw companion ability, which will be introduced and developed roughly halfway between the start of the level and reaching the level hub room.
- Heater rooms and the boss room will be built out of existing obstacles and threats rather than custom designed/coded sequences.
    - Building the special challenge and boss challenge rooms out of our basic obstacles and threats simplifies the development and design process.

Jacobs Level Design Findings:

Level Overview:



Because the image is large, it is also uploaded to the Drive. You can zoom in and actually read the text and other details here:

LevelLayout.png

Room Pipeline

Room files (each being an individual Godot scene) will use the following naming convention:
01_LevelName_a_RoomName_room.tscn

- '01' refers to the internal level number. GreatHall is 01 and Cellar is 02.
- 'LevelName' refers to the name of the level itself. This will be one of the names listed above in the Level Types section. This section must use CamelCase for legibility.
- 'a' refers to the portion of the level that the room appears in. The starting tutorial might be 'a', the companion ability tutorial section might be 'b', the heater portions might be 'c', 'd', and 'e'. These are to help us see the general progression of the level with just a glance at the filenames.
- 'RoomName' refers to the name of the room itself. 'Entrance', 'Boss', and 'IciclePlatforming' are all acceptable names. This section must use CamelCase for legibility. Try to stick to one-word names, but it's fine if more than one word is needed to describe the room at a high level.
- All rooms must have the suffix '_room.tscn'. This makes it easier for us to filter out what scene files are rooms. The door entity has a variable that points to what room they go to, which uses the file path to the room scene file itself. Ending the files with '_room.tscn' means it will show up in the file picker when changing this parameter in the inspector tab for each door.
- Grayboxed levels instead must follow the format 'GB_RoomName_room.tscn' and go in the folder 'res://src/rooms/graybox/'.
- Testing levels instead must follow the format 'TEST_RoomName_room.tscn' and go in the folder 'res://src/rooms/test/'.
- For some room name examples, see below:
    - 01_GreatHall_a_Entrance_room.tscn
    - 01_GreatHall_b_CellarDoor_room.tscn

- 02_Cellar_a_AzEncounter_room.tscn
- 02_Cellar_c_AzWorkshop_room.tscn

Creating a new room:

- In the project files, there is a scene at the path ‘res://src/rooms/room.tscn’. Right-click it in the FileSystem panel and select ‘New Inherited Scene’. It will open a new scene in the main editor and scene panel.
- Before making any changes, save the scene, and put the file in the appropriate level folder with the appropriate filename as outlined above.
- Next, double-click the ‘Room’ node in the Scene panel and rename it. Name it exactly the same way you named the file, but without the ‘_room.tscn’ at the end.

Setting up room transitions:

- Each Room scene has a child node called ‘Doors’ by default. This node will have two child Door nodes, one for the room entrance and one for the room exit. If you ever need to add a Door to your room (say you accidentally delete one of the existing ones), click the ‘Instantiate Child Scene’ button at the top of the Scene panel and select ‘door.tscn’. Make sure that every door you add is a direct child of ‘Doors’!
- Every door you add will need three variables set in its inspector panel before it works correctly. Each inherited Room scene starts with two doors to make this part easier.
    - Path to Target Room: This is the file path of the Room scene you want the door to lead to. Click the file icon on the right of the box and navigate to ‘res://src/rooms/<your level>/’. Select the file for the room you want the door to go to.
    - Door Name: This is the name of this door. This should be ‘enter’ or ‘exit’. Doors at the logical “start” of the room are ‘enter’ and doors at the logical “end” of the room are ‘exit’. Name the Door nodes in the scene tree ‘DoorEnter’ and ‘DoorExit’ to match for convenience.
    - Target Door Name: This is the name of the door you want the player to spawn at in the target room. If this door is ‘enter’, this variable should be ‘exit’. If this door is ‘exit’, this variable should be ‘enter’.

The Tech Bible describes “glue code” that will string together all of our game systems. The SceneManager is one such system, as it handles swapping rooms in and out, switching from menus to gameplay, and the like. It will not handle anything beyond that; any initialization those scenes need to do must be conducted by themselves.

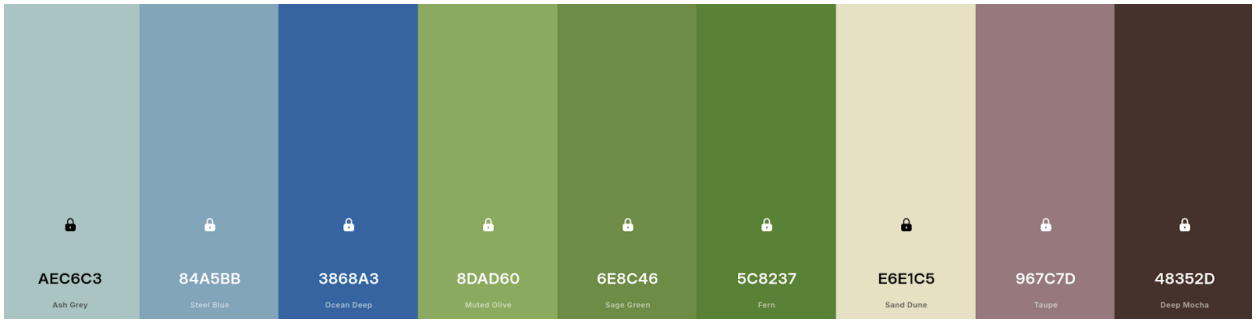
Level Layout

UI/UX and Interface Design

Design documentation

Standards

- Fonts
    - Beholden (https://amorphous.itch.io/beholden?download)
        - Used for all text
        - All capitals for titles
    - ZT Mota (https://www.1001fonts.com/zt-mota-font.html)
        - Used for the game Logo and main menu title
- Initial Color Palette



- Specific Colors
    - UI Base Color: #E6E1C5
    - UI Base Text Color: #352620
    - Leaf Color (unless otherwise indicated): #5C8237
    - Title/Main Screen
        - Game Title Color: #4F8F0E
        - Return Address + Stamp Base Color: #AEC6C3
        - Return Address + Stamp Accent Color: #617875
    - Menu/Settings/Control Screens
        - Envelope Color: #BCB692
        - Title Text Color: #576F35
    - Dialogue Boxes
        - Fenn Name + Accent Color: #576f35
        - Az Name + Accent Color: #A86A19
        - Winston Name + Accent Color: #35639C
        - Wizard of Doors + Accent Color: #6B2D76
        - Plain Color: #352620
        - Progress Dialogue Indicator: #352620

    - HUD Health
        - Background Color: #967C7D at 65% transparency
            - NOTE: This may change once things get into engine because of it being overlaid onto the game but I wanted to have something that I could try to start.
    - HUD Wind Meter
        - Background Color When Not Full: Background Color: #967C7D at 65% transparency
            - NOTE: This may change once things get into engine because of it being overlaid onto the game but I wanted to have something that I could try to start.
        - Background Color When Full: #8DAD60 at 65% transparency
        - Swirl + Leaf Color: #425422

Critical Elements

- Title/main menu screen
    - Envelope
        - Stamp Takes you to credits
            - Cute icon in the middle with the “Credits” curved around on the top and the bottom
        - Return Address takes you to how to play
            - “KINGDOM’s How to Play”
        - Project title and subsequent selections underneath are centered
- Settings screen/ Controls screen
    - Will have letter theming
    - Will be an overlay that covers game play
        - Game play in background will be paused and blurred and muted in color
- Dialogue box
    - Inspired by Moorish architecture and shapes
    - Dialogue loads left to right
    - Blinking indication ability to continue on
- Player HUD (health, wind meter primarily)
    - Health
        - Discrete lives displayed by leaves that disappear to indicate life/health lost
    - Wind Meter
        - Progress bar
        - Long wind symbol will slide in from left to right with leaves that appear to show progress

Pipeline

- Art Creation
    - Base assets (no text) are created and exported as PNGs to be added in engine
    - All assets are created in a way that each piece can be extracted from each other
        - Ex: envelope base and stamp are created separately from each other and are exported as two assets
- Engine Implementation
    - Assets are imported into the UI assets folder and into subsequent folders based on use
    - All assets are placed together in their respective Godot scene allowing for reuse of assets
        - Background is placed into scene then button art is placed on top of the background
    - Any text is added last using Godot's built in text and button features.
    - Once all assets are placed, code and functionality can be added to the scene and each element when needed