Development of Hourglass Echo

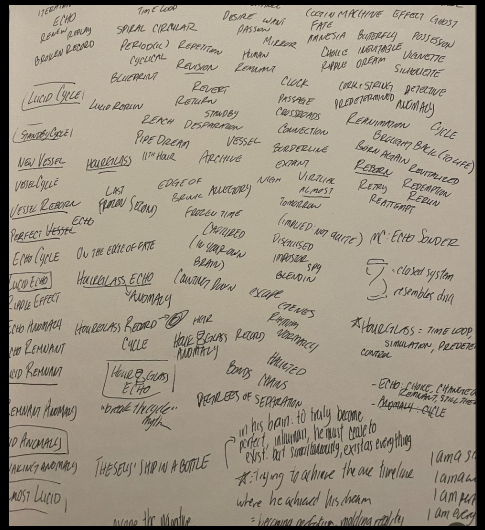
Lee Yoerns

Context: What is Hourglass Echo?

- In-progress personal project; future video game
- Began work in 2019
- Most development work so far has been the story aspect

Draft of the main theme. Non-vocal version.
Made in 2024.

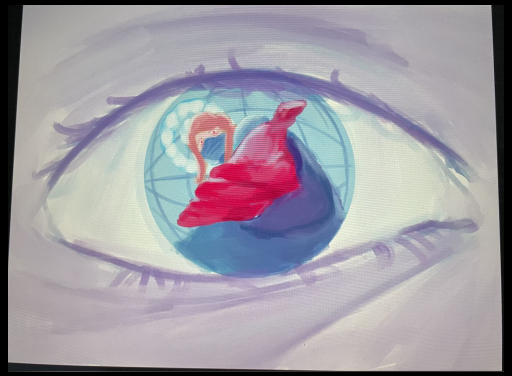
Previous Concept Work Examples



Above: Title Brainstorming. 2023

Center: illustration depicting the overarching plot. 2024

Top Right: concept sketch featuring one of the main antagonists. 2024



Bottom Right: Monste Concept Design. 2021

Problem Identification

The software aspect of Hourglass Echo is wildly underdeveloped (read: undeveloped), both in design and application.

The following features need to be further explored:

- User Interface
- Logic of Game Mechanics
- Sound Design
- Character Responses to Player Choices

Existing Solutions: Phigros' Timing Ranges

- The rhythm game Phigros has ranges of milliseconds that all yield a different "judgement" on a note depending on when player input was registered

- A harder gamemode called "Challenge Mode" decreases the range for "perfect" and "good" inputs by half
- Challenge Mode is designed for more experienced players, as it is only available after unlocking all of the main story albums

Judgement	Combo	Accuracy	Notes	Time window	
				Normal	Challenge Mode
Perfect	Increases by 1	100%	*All notes*	±80ms	±40ms
Good		65%	Tap/Hold	±160ms	±75ms
Bad	Resets to 0	0%	Tap	±180ms	±140ms
Miss		0%	*All notes*	N/A	N/A

Existing Solutions: Pokemon's User Interface

Menu when walking around; typically hidden

- The available options in the UI depend on what is most useful to the player's current task
  - Example: the "run" feature would be arbitrary outside of battle, therefore it is not included in the menu on the left
- The battle menu features button size hierarchy; the "fight" button is significantly larger and centered on the screen, because it is the user's most likely choice

Images shown are from Generation 4 of Pokemon (Diamond/Pearl/Platinum)

Menu when engaged in battle; always visible

Existing Solutions: Don't Starve's Voice Acting

- The game "Don't Starve" uses various musical instruments in order to voice each character
- There are unique sounds for different actions, and there are sometimes multiple different sounds for a single action to create variation
- Each character is designated as a single instrument
- Some characters have special actions, and subsequently additional sound effects

https://www.youtube.com/watch?app=desktop&v=nVzJZZ0qvfY&t=0s

Above: link to a video that compiles voice clips of (almost) all the different characters, with the actions labeled

Application: Battle Mechanics

- Player must make their attack within a certain amount of milliseconds of the song rhythm/note
- Less “on beat” means less damage on the enemy and increased likelihood of a missed attack
- Dodging on beat will prevent any damage from being taken
- The further away the player is from being “on beat” when dodging, the more damage they will take

Left: Mockup of a looping battle theme for an easy-moderate fight. Made in 2023.

Right: Mockup of a moderate boss theme. Made in 2022. One sample loop from Garageband used.

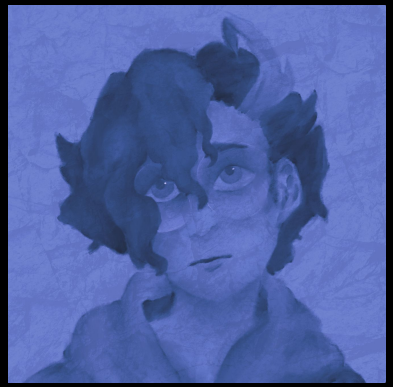
Battle Mechanics (cont.)

- Battle gameplay is not hitbox-based, but instead based on following the rhythm of the music
- Harder fights will have songs with irregular patterns or abnormal techniques
- Boss fights can change tempo, time signature, motif, etc at key points to indicate the next stage of the battle

Mockup of a boss theme with distinct sections. Made in 2022 using sample loops in Garageband.

Application: Sound Design

- Identify each main character with an instrument and/or motif used in the music that plays when they are being prominently featured
- Program will identify which character is present and select one of their themes based on the situation



Character: Miles
(illustration from 2022)
Instrument Motifs: harp, high strings

“Blue Heaven”, mockup of Miles’ character theme. Made in 2021 from Garageband sample loops.

“Gossamer Wings”, unfinished mockup of a secondary theme. Made in 2023 from Garageband sample loops.

Application: Logic

- Flow chart that indicates how different choices can/will lead to different results
- Some of the player's previous choices can be overridden by choices they make in the future

Application: User Interface

- The battle interface should contain different options than the typical interface
  - Additionally, it should be visibly distinct while still maintaining some consistency
- Unlike Pokemon, Hourglass Echo would not be turn based; wanting to use an item should not pull up a menu that covers the battle screen
  - In battle specifically, a button would be designated to toggle back and forth through usable items, and a different button would activate/use the item; faster than searching through menus
  - The buttons would have different (but perhaps similar) functions outside of battle

Design Approach: Battle Mechanics

- “Easy Mode” and “Normal Mode” for battles; Easy Mode has larger ranges of time and less attacks to dodge
- Measure the amount of time (in milliseconds) it takes for the player to input
- Designate set ranges of time and identify which range an input falls into
- Calculate whether an attack hits or misses (when outside of “perfect” range)
- Calculate the amount of damage dealt or received

**Potential for a “Hard Mode” that the player can activate after completing their first “loop” of the game

- Narrows the amount of time within each range and enemy attacks have more erratic patterns

Design Approach: Time Loops

- Hourglass Echo's overarching plot is about both the player and another character being trapped in a time loop
- Because it is not just the player within the time loop, sometimes different outcomes may occur even if the player makes the exact same choices
  - It is intended to make it feel like the other character has free will
- Find a way to code the character to make different actions each time the loop repeats, and make it feel random and unpredictable

Design Approach: Time Loops (cont.)

Solution: import the random library to make dice rolls

- The beginning of the game will have a cutscene where someone rolls a die; the number it lands on determines what events can potentially trigger
  - Some events still require the player to make particular choices before they can occur

Design Approach: Character Responses

- The player will often be able to respond to characters and their environment in a few different ways, ranging from good-natured to mean-spirited
- The characters nearby will act according to the player's choice

- Python's class categorization can be used to set all the choices in a scene to have a "tone" behind them
- The character's sprite will change to match the tone used by the player

Design Proposal: Pseudocode

Damage will be reduced if a player is wearing defensive gear:

```
if decent_gear in equipped_items:
#equipped_items is a list of gear, weapons, etc the player has
chosen to use
     total_dmg += (dmg_taken//1.5)
elif good_gear in equipped_items:
     total_dmg += (dmg_taken//2)
elif great_gear in equipped items:
     total_dmg += (dmg_taken//3)
elif amazing_gear in equipped_items:
     total_dmg += (dmg_taken//4)
```

Determining how much damage a player will take:

```
player_reacttime = user_input
#user_input is the amount of time in milliseconds between the
beat and the registered moment the player reacted
perfect_range = (0, 51)#range max is 2nd number -1
good_range = (51, 101)
ok_range = (101, 151)
bad_range = (151, 201)
#any input delayed more than 200 milliseconds is considered a
miss
if player_reacttime in perfect_range:
     dmg_taken = 0
elif player_reacttime in good_range:
     dmg_taken = user_input
elif player_reacttime in ok_range:
     dmg_taken = user_input + (user_input/2)
elif player_reacttime in bad_range:
     dmg_taken = unser_input * 2
else:
     dmg_taken = 500
```

Pseudocode (cont.)

Determining the good true ending:

```
import random
d6 = [0, 1, 2, 3, 4, 5]
#5 is a symbolically important number in the world of Hourglass
Echo
dice_roll = random.choice(d6)
if dice_roll == 5:
     good_true = dice_roll
```

Potential application of the good true ending:

```
#Within an event (= within a function):
print('Dialogue Option 1')
print('Dialogue Option 2')
if good_true:
     print('Dialogue Option 3')
#The third option only appears for the user if the good true
ending is active, which only occurs if the roll at the beginning
of the game is a five.
```

Pseudocode (cont.)

Determining which character song will play:

```
examplechara_songlist = [song1, song2, song3]
if mood = normal:
     song1
elif mood = sad:
     song2
else:
     song3

#each song variable leads to an audio file playing
#mood is a variable determined at the beginning of each event,
but has the potential to change based on the choices made within
an event. If the event begins as mood = normal, but the player
makes a choice that makes a character upset, the song for the
sad mood would begin playing instead.
```

Open Questions

1. How can I have the program identify when a “beat “ occurs within a sound file, in order to count the milliseconds between when the beat is identified and when the player’s input is registered?
2. How would the application of a music-based fighting system look?
    a. Should there be a combo system?
    b. What are the best controls to attack, dodge, use an item, etc with such game mechanics?

Citations

https://phigros.fandom.com/wiki/Game_Mechanics

https://bulbapedia.bulbagarden.net/wiki/Menu#Gallery

https://www.pokecommunity.com/threads/battle-uis-across-generations.469565/

https://dontstarve.fandom.com/wiki/Category:Audio

https://www.youtube.com/watch?app=desktop&v=nVzJZZ0qvfY&t=0s