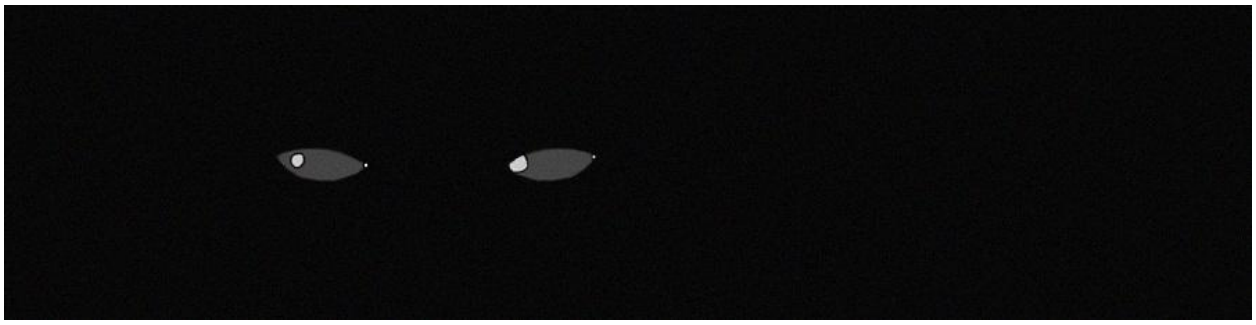


Void: Occultic Syntax (Working Title: Techno-Occult Hangman)

High Level && GDD

Game Designer – Datorien Anderson

Updated: 5/30/2023

4 Pillars:

- Tense
- Engaging
- Occult
- Mystery

Links:
Terminus Briefing

High Level Overview

Game Reference(s):
- Hangman

3Key:
- Puzzle, Thriller
- Orthographic

Engine: Unity

Goal:
- UNKNOWN

Objectives:
- Discover exactly what Vos Is / Play hangman!
- Unlock the story-mode.

(Recurring) Objectives:
- Follow the narrative.
- Play the core/general hangman experience.

Interactions (As of Now, Prototype 1.0):
- The player is able to choose between these inputs to play:
    - Phone-styled Multi-tap Numberpad
    - 0 and 1 button for Binary ASCII
- The player will be able to select word category themes of what they wish to use:
    - Themes are:
        - ALL
        - Demons, Gnosticism, Fiction (Public Domain with the next update), Angels, Cosmic, Matter & Geometry
- The player guesses the letter they want for the hangman game.

Game States/Ideas:
- Story-mode:
- Arcade / General: This is the standard gameplay type–for the proof of concept, all inputs will be available at the start.
    - For version 1.0.0–when it's not a proof of concept, only standard and number pad input (NPI) input will be available and the player will need to get 10 correct words in a row in the NPI to unlock binary input.

Fail States:
- Story-mode: Players' health runs out, The players’ phone system integrity is down and/or they run out of Terminus Favor.
- Player has no more tries left.

TERMINUS INTERACTIVE

GDD

UI Elements & Transitions:

This section will explain the most important part of this game as it's on mobile. It will also likely be available to play via Google Play games beta. (If I decide to go that route.)
Here is the art-brief–will make this better later, but all of my properties use it in some way. Will add some UI references to this later.

Standard / General Gameplay:

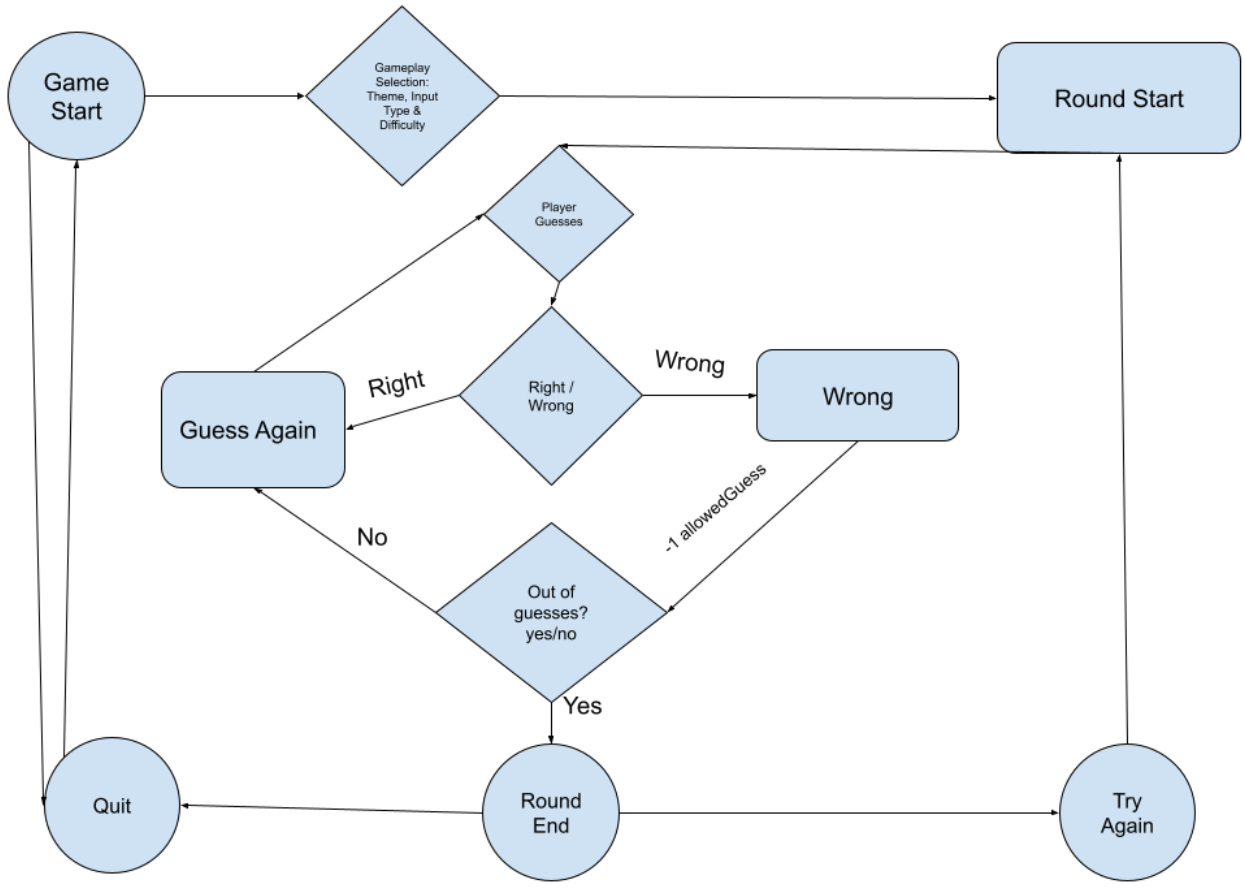
- Timer Function.
- Ironman Mode:
    - When Ironman mode is selected–dashes aren't displayed and there are no more sessionLives. The only way to get an allowedGuess back will be due to 'x' number of consecutive correct guesses.
- Attempted Letter Bank →
    - Standard / Story: This just shows what letters the player already attempted–trying to figure out how to have this in a way that isn't jarring to the screen unless I just make a selectable or make it a minimal thing.
- DisplayedInputString →
    - Standard / Story: Displays the current word as ____-_____ (if Ironman is selected) it will not display the dash.

Storymode Gameplay:

- Conditional Timer Function →
    - The timer can be triggered on and off, which means that the Core.cs script will also need a bool to determine–IF isInStoryMode && isTimerTriggered
    - Not all story beats will be timed–such as with the ASCII/Puzzle Implement.
- ASCII / Implement Puzzle →
    - This might have an enumerator that can be preloaded and randomized between ASCII Art, Symbol or etc., for the player to guess what it is.
    - This will not be timed as it would require more thinking.
    - ASCII / Implement Puzzle →
        - This might have an enumerator that can be preloaded and randomized between ASCII Art, Symbol or etc., for the player to guess what it is.
- Using Inky or Yarnspinner to implement the narrative, which has its own macros.

TERMINUS INTERACTIVE

Gameplay Loop



Screens:

There are generally seven screens so far:

→ Boot-up screen:

- The most important screen, it's what catches the players attention. Allowing them to get a first impression–this is important! As it will determine initial interest.
    - Boot-up Screen → Main Menu

→ Main Menu

- The player should be able to change the sound and switch between relevant themes. As well as check their profile data.
    - Main Menu → Options
    - Main Menu → Controls
    - Main Menu → Credits
    - Main Menu → Gameplay-Config

→ Options Screen

- The player should be able to change the sound and switch between relevant themes. As well as check their profile data.
    - Options Screen → Main Menu

→ Controls Screen

- This displays the binary ASCII and the numpad information, also should allow for accessibility when playing it.
    - Controls Screen → Main Menu

→ Credits Screen

- Self explanatory–credits all who worked on a feature in the game / on the game.
    - Credits Screen → Main Menu

→ Gameplay-Config Screen

- The player will be able to select the theme, difficulty, input and later select Iron-man.
    - Gameplay-Config Screen → Main Game Screen
    - Gameplay-Config Screen → Main Menu Screen

→ Main Game Screen

- This is where the main logic of the hangman game is found–the narrative will likely be in a different scene for modularity's sake.
    - Main Game Screen → Main Menu Screen
    - Main Game Screen → Main Game Screen

Main App

The main application in this game that's centered around is a hangman game, I'm thinking each dictionary theme should have at least 100 words.

Themes are:

- Demons / Angels
- Cosmic
- Matter
- Geometry

TERMINUS INTERACTIVE

Apps / Mini-games:

At least three mini-games that are playable that need to be made: the two are concept games but CI is an ARG element that I'm thinking of adding in. I still need to add in some more 'mini-game'. Minigames will need to be progressed to unlock progress in the 13 story nodes.

OR – Occult Runeworking- [RUNE.NET]

- The first one is OR – Occult Runeworking. The player is presented with a theme and they must make runes via using antonyms of a word. It should work akin to TES:Online's enchanting system.
- The player is given an 'order' that they must complete for a specific situation. This is a word association game type.
- Will have 27 situations [IN PROGRESS]
    - https://miro.com/app/board/uXjVM3mVn-I=/?share_link_id=269631606483
- Will need to create a set of 'runes' that are in the format of: prefix, root & suffix.
    - Might use my conlang that I created as a basis.

AS – Apollo Sights – [AP SIGHTS]

- A game where you trace out constellations.
- There are 88 Officially Recognized ones by NASA.
- The player should be able to trace via finger or if using an emulator, a mouse.

RA – Ritualistic Ascension – [R.ASC]

- The last minigame. This will just be an endless runner. Likely in the style of pitiful or rush and attack.

CI – Codex Informa

- This is where the hook-up with the ARG element is.

Narrative Design [Ignore for now]:

Backstory Idea

YOU have found a phone that utilizes voidOS–it seems to be hidden by an innate type of hangman game. Upon turning it on, the screen lights up with a word - 'voidOS'. You soon discover that you're unable to access the other functionalities of the device due to an inherent hangman game. The game has an AI - ‘Vos’, it will communicate with you based on the outcome of each hangman game. Vos just stands for VoidOS. Will use Inky and integrate it for this:

Narrative Proposed Gameplay Elements [NOT INTEGRATED AT ALL YET–NEED TO SEE IS POSSIBLE]:

- On certain levels, instead of hangman, you find yourself solving intricate riddles, unlocking hidden ASCII art, or trying to crack coded messages. Successful completion of these tasks might earn you 'Terminus Favor', adding a boost in your game (like revealing a letter in the hangman game or adding back a lost token).
- As you progress in the hangman games, you begin to decode cryptic messages that unravel the story of two mysterious entities - 'The Terminus' and 'TOBI'. With each victory, Vos warms up to you, slowly revealing its affiliation with The Terminus. If you make a mistake, it gets hostile, leading to a reduction in your 'tokens'. Losing all tokens might attract TOBI, posing a potential danger to Vos and its creators, The Terminus.
- As you delve deeper, you start learning about the rivalry between The Terminus and TOBI, and about Vos's dual nature. In some interactions, you might have to make choices that will affect the narrative. For example, deciding whether to keep Vos's affiliation with The Terminus a secret or risking TOBI's attention could branch the storyline.
- As the game approaches its climax, you might find yourself in a position where your decisions have made you an ally of either The Terminus or TOBI. If you have managed to earn enough 'Terminus Favor', Vos might reveal the purpose of The Terminus and why they developed voidOS. Conversely, if you lose all tokens, TOBI could intervene, leading to an entirely different ending where you might have to save Vos from TOBI's tech-priests or even cooperate with them.
- The various ways to communicate with Vos - be it through solving hangman puzzles, figuring out riddles, or unlocking coded messages - can lead to different paths in the narrative, making each player's experience unique.
- For an ARG component→
    - I might create an actual subdomain on my website that might contain hidden clues, coded messages, or other interactive elements that tie into the game's narrative.

This project will use a prototype hangman game that I created as a basis and it won't be on mobile. It will technically be a 'new' experience since I'll be using the prototype of Void: Occultic Syntax and my other prototype Void Installer–which will mitigate some risk.

Where does Inworld come in? Vos, the Void:OS in-lore AI, will be tracking the players–what do: there are two factions TOBI and The Terminus. Which one the player aligns to will dictate how the system responds and I'm wanting to create an Augmented Reality Game (ARG – Component). InWorld AI can help achieve this and create fidelity to the vision of how I wish for the game to be.

I'm hoping to partner with Ultra.io too and this could be an experience on their platform. That'll further my creative vision goals of creating an engaging Transmedia IP.

Other proposed gameplay elements:

- Co-operative mode.
- Versus mode.
- Display ASCII Art – the player must figure what world it represents.

The story progress will be able to be reset.

The story is done in '13' chunks–blocks–it should be:

1. Tutorial
2. Awakening
3. Three
4. Four
5. Five
6. Six
7. Seven
8. Eight
9. Nine
10. Ten
11. Eleven
12. Twelve
13. Thirteen

Player Onboarding

The second single most important part of the user experience is player onboarding.

The player should have immediate access to the General Mode–as the Story mode will not be in creation for this currently–and for now, using Binary on start is allowed.

I will make a Tutorial dictionary set–however, I will have to make sure that it's a one-time thing. This will guide the player into playing the game before the much harder–by default–themes are available to use.

TERMINUS INTERACTIVE

Technicalities & C# Methodologies:

Coding Principles and Technical Writing
How variables should be written in this as well as how assets should be written.

Variables: camelCase

Code Refractory
Github Repo: https://github.com/voidespy/OccultHangman/tree/development

As of recently, GameManager.cs has been gutted and the hangman game logic has been created into a separate script called Core, this has caused some issues and broken functionality:

As on 5/30/2023:

- [x] ~~Binary is not processing the guess and updating the display values.~~
- [x] ~~SelectedTheme in DisplayManager is not updating when the value is selected; I'm sure I fixed this but making the development branch probably screwed with this. **sighs**~~
- [ ] When the word has failed, the timer will continue to decrease.
- [ ] Despite pressing quit, the remainingTimeToSolve float will continue to decrease in the MainMenu.
- [x] ~~NumpadManagers currentInput is not updating as the num blocks are pressed- but the numBlocks are correctly referenced in the inspector.~~

Null References:

As of 5/30/2023:

- [x] ~~HangmanDictionary component not found! This is upon starting the scene in Menu.~~
- [x] ~~NullReferenceException: Object reference not set to instance of an object: Core.Update();~~
- [x] ~~NullReferenceException: Object reference not set to instance of an object: DisplayManager.Update();~~

https://trello.com/b/404VzfW3/techno-occult-hangman
→ Putting changes in here, so the document won't get too crowded. Deciding on whether or not to use Trello or Miro. https://term-al.atlassian.net/jira/software/projects/TOH/boards/1

There are a few extremely important scripts that drive this:

NumberPadManager	BinaryDictionary	GameManager
HangmanDictionary	Core	DisplayManager

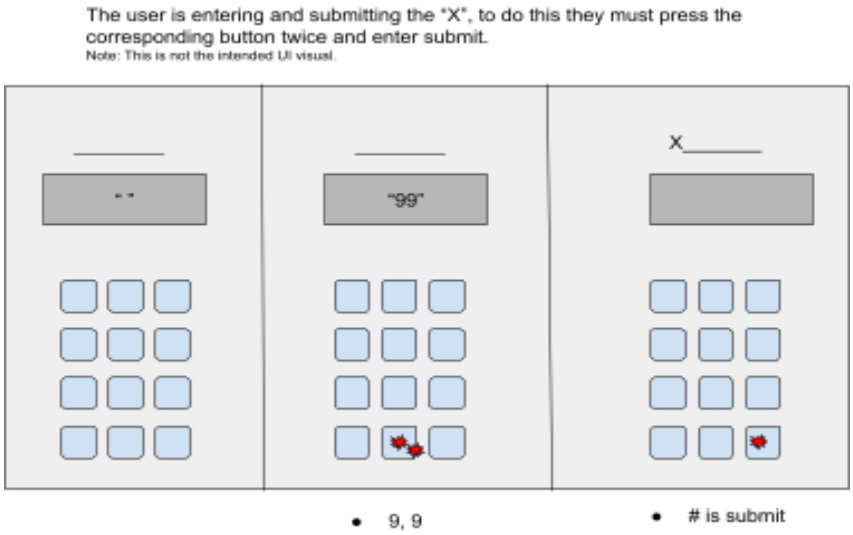
1. NumberPadManager.cs

NumberPadManager holds this:

```
public class NumberPadManager : MonoBehaviour
{
    public Dictionary<string, int> keypadMapping = new Dictionary<string, int>()
    {
        {"1", 0}, {"2", 1}, {"3", 2},
        {"4", 3}, {"5", 4}, {"6", 5},
        {"7", 6}, {"8", 7}, {"9", 8},
        {"*", -1}, {"0", 9}, {"#", -2}
    };

    public Dictionary<string, string> letterMapping = new Dictionary<string, string>()
    {
        {"2", "A"}, {"22", "B"}, {"222", "C"},
        {"3", "D"}, {"33", "E"}, {"333", "F"},
        {"4", "G"}, {"44", "H"}, {"444", "I"},
        {"5", "J"}, {"55", "K"}, {"555", "L"},
        {"6", "M"}, {"66", "N"}, {"666", "O"},
        {"7", "P"}, {"77", "Q"}, {"777", "R"}, {"7777", "S"},
        {"8", "T"}, {"88", "U"}, {"888", "V"},
        {"9", "W"}, {"99", "X"}, {"999", "Y"}, {"9999", "Z"}
    };
```

This is essential for the Numpad Input type, to storyboard this:



NumberPadManager has a maxDigitCount for the string of 4. The method assigned to the submit button passes the string into letterMap and finds the associated ‘char’, if the associated

‘char’ string is not found–the game will deduct an allowed guess and the currentGuess string will clear.

2. BinaryDictionary.cs

The binary dictionary script works the same was as the NumberPadManager script does, except that is has a maxDigitCount of 8 as per required for this binary input. However binary works slightly differently:

```
public static readonly Dictionary<char, string> lowercaseBinaryDict = new Dictionary<char, string>()
{
    { 'a', "01100001" },
    { 'b', "01100010" },
    { 'c', "01100011" },
    { 'd', "01100100" },
    { 'e', "01100101" },
    { 'f', "01100110" },
    { 'g', "01100111" },
    { 'h', "01101000" },
    { 'i', "01101001" },
    { 'j', "01101010" },
    { 'k', "01101011" },
    { 'l', "01101100" },
    { 'm', "01101101" },
    { 'n', "01101110" },
    { 'o', "01101111" },
    { 'p', "01110000" },
    { 'q', "01110001" },
    { 'r', "01110010" },
    { 's', "01110011" },
    { 't', "01110100" },
    { 'u', "01110101" },
    { 'v', "01110110" },
    { 'w', "01110111" },
    { 'x', "01111000" },
    { 'y', "01111001" },
    { 'z', "01111010" }
};
public static readonly Dictionary<char, string> uppercaseBinaryDict = new Dictionary<char, string>()
{
    { 'A', "01000001" },
    { 'B', "01000010" },
    { 'C', "01000011" },
    { 'D', "01000100" },
```

It goes through both BinaryDictionaries and the player is given a timer until the string clears–this does not deduct allowedGuesses by design. Due to the inherent difficulty of binary this idea is chosen to reinforce quick learning of binary.

Since binary doesn’t have a submit, the player presses so far a green button and if it's a match it will pass it through and display it if it's correct but of course, if it's not– it will deduct from allowedGuesses.

The only way deduction happens (probably unless it's on Ironman mode: as in, if the player allows the timer to go to zero for the input instead of pressing clear but ONLY in Ironman.) is if the player presses the green submit button and they don’t know.

3. Core.cs

This contains all the relevant logic and information to that instance of hangman. Specific data should be serialized and json'd for player statistics. You can say this is the digestive system of the program. This includes but is not limited to:

- PlayerID
- If the player completes the correct amount of words on numberPad Input to unlock the story mode.
- The amount of errors received.
- The streak of correct guesses.
- The streak of consecutive correct word guesses.
- The frequency of which theme is selected.
- Most frequent word occurrence.
- Most failed word, then words.
- Shortest time for word completion.
- Longest time for word completion.
- If the player completed an entire theme dictionary in one game.

The other things that core currently will do are:

- If a word has been used it will add it to usedWords,
    - if this was correctly guessed it will be added to solvedWords.
    - If this was not correctly guessed it will stay in usedWords, and may not be used as the next 'x' (3) words UNLESS there is less than four words left in that chosen theme to use.
- Sets the number of guesses based on which mode is picked IRONMAN_MODE is triggered on– will likely change the variable to just IRONMAN_MODE as its more simple.
- Get NewWord() and GetRandomWord() are technically the same thing– I need this just needs to be consolidated into one thing.
- ProcessGuess(string guess) method is what the BinaryDictionary and NumpadManager as well as normal input uses to process the guess to and see its a correct guess or not.

4. DisplayManager.cs

This is where all the visual information is contained–as in this is all the user's true facing properties that will be displayed. Such as:

- Updating the time when a Timer is used.
- DisplayDefeat → Pop up, when the player runs out of allowedGuesses.
- DisplayVictor → Pop up, when the player correctly guesses a word.

- DisplaySucessWord → This is what allows DisplayDefeat and DisplayVictor to even show up on the screen. This method should change to DisplayEndState
- internal void UpdateWordDisplay(string currentWord, bool isIronOn, SessionDifficulty sessionDifficulty) → The heart of how the string is updated.

5. HangmanDictionary.cs & GameManager.cs

This holds all the information relating to what words this is the heart of the program. Without it–it can't choose the word. Will need to add a serialization and likely an .xml part, using the C# I/O stream and json to make it easier to make new string themes and values. The following below are not final– Fiction should use public domain–or will likely be edited out.

Such as:

- Current Themes[Working] are: Demons, Gnosticism, Cosmic, Matter, Geometry, Angels, Fiction and Geometry
- Proposed Themes [To Be Added] are: Alchemy & Herbology

While not a part of HangmanDictionary.cs, an adjacent script to that and GameManager.cs is the ThemeDropdown script. This is what allows the player to select a chosen theme from the dropdown–as well as is how GameManager is able to even access the WordsByTheme and get the name of the themes from HangmanDictionary.

Other Technical Stuff:

The easiest way I'm thinking to do the ASCII Art, is to make a gif and convert it to this: https://www.adobe.com/express/feature/video/convert/gif-to-mp4

However; the scriptable object might need to hold the video then when a specific ASCII gets picked it will set the video clip to the one on the scriptable object.