❗ This document has been altered to adhere to NDA guidelines.

Project Black

Dynamic Banter System

Research & Design Document

Feature Owner: Marianne Cassidy

Table of Contents

Research:

Design:

Research Intent

We hope to implement a dynamic and conversational dialogue system in Project Black, and as such, we have researched how other creators and studios have approached similar methods of delivering story. It's important that our "banter" system between the protagonist and his sister is able to adapt fluidly to the player's location, previous actions, and previous conversations, and for their dialogue to feel reactive and malleable.

Sources

The System Behind Hades' Astounding Dialogue (People Make Games, 2020)

The System Behind Hades' Astounding Dialogue

Key takeaways:

- **Sheer number of potential dialogue lines**
    - *Hades* features hundreds of thousands of possible lines for its characters, meaning most players will only encounter a fraction of them. This won't be necessary to such an extreme extent for Project Black, but it might be worth ensuring we at least have enough dialogue that repetition in a single playthrough is unlikely
- **INT priority system for line triggers**
    - Every exchange in *Hades* is given an integer value that dictates how high in the order of priority it is when its conditions are met. For example, if two lines have almost identical conditions and are both available when those conditions are met, but one has an INT priority of 7, and the other is 43, the former will trigger. This could work very well for PB, as we'll be able to ensure players experience conversations in a logical order, and aren't given lines that refer back to dialogue or locations they haven't personally discovered yet
- **Randomizer that removes dialogue once it's been used, and adds everything back/starts over once it's all been triggered**
    - A system like this ensures the player never runs out of dialogue, even in the extreme case where they've been playing the game for several hundreds of hours. Something like this would only make sense for certain exchanges in PB, such as conversations between our protagonist and his sister that don't comment upon their immediate tasks and goals, so the utility of this system might depend on how much casual dialogue we intend to include (which might be relatively little)

- **Reactive/query-based interactions**
    - Dialogue in *Hades* is highly reactive due to the volume of queries it makes for each line. Has the player done X, Y, and Z, and also already heard lines 34, 170, and 2? Then line 60 can trigger, and provide a response to the player's extremely specific experiences so far. This enables players to feel as though the game is watching and taking account of their every move, and can be incredibly satisfying. PB might benefit from such a robust system, as it will make exploration that much more personal and engaging
- **Even extreme edge cases are provided for (e.g. player never encountered an important/hard to miss yet optional character)**
    - It's unlikely that PB will feature as many extreme edge cases as a game like *Hades*, where–for example–talking to almost every NPC is entirely optional, however, what few we end up with will need to be accounted for in our dialogue system

Application to Project Black

Though *Hades* is a very different game to PB and has a very different style of narrative delivery, we can learn a lot from its methods. Using a similar line priority system would ensure smooth narrative delivery during exploration sections, and trying to implement a level of reactive dialogue would provide a more engaging and personal experience for our players.

Narrative Sorcery: Coherent Storytelling in an Open World (Jon Ingold, 2017)

Narrative Sorcery: Coherent Storytelling in an Open World

Key takeaways:

- **Linear vs. circular**
    - Ingold specifies that the opposite of linear in this context is not "non-linear", meaning events happen in a random, possibly non-chronological order, but "circular", with the difference being that in a linear narrative, you never go back on yourself/you're always moving forwards, whereas in a circular narrative, you can infinitely retrace your steps. Project Black will likely be a combination of these two structures, in that the world can be explored in a circular fashion (once it has fully opened up), but the story features linear progression and has a clear and final end point

- **Open world vs. "authored branching narrative"**
    - Branching narratives offer a great deal of choice and personalisation, however, all of these choices are “authored”, meaning they are still tightly designed and intentional in a way open world narratives often avoid in favour of randomised elements and emergent play. Project Black will likely be closer to an “authored” experience, but with an exploration style closer to an open world
- **Quests vs. "encounters”**
    - “Quests” have specific start, middle, and end points, while “encounters” allow for numerous different ways to discover, approach, and resolve stories. All avenues are intentional and offer their own flavour to the narrative, but some might involve more or fewer character interactions, combat, exploration, etc.
    - In Ingold’s example, the player is able to encounter a wolf in the woods regardless of whether or not they have been told about its existence. This differs from many quest designs in games, where the relevant elements of the quest only occur in the world once the player has triggered the start of the questline. Many open world games use a combination of encounters and quests
- **"Defensive logic" - "guard the story", make sure that it works no matter what. The story is able to "cope"**
    - Every “encounter” is built in an ad-hoc way--every scene is built to cope regardless of world state. No assumptions are made regarding what the player knows or doesn’t know. It’s unlikely we’ll need this for Project Black, but the spirit of “guarding the story” might be useful to keep in mind. We want to make sure our dialogue always fits the situation, and that we use robust checks to make sure the player knows what they need to know, - rather than assuming
- **"State trees"/"state machines" that capture complexity**
    - State modelling allows plots to be started, followed, and ended in different ways. The trees in the example are *not player targets*, in that they are not presented to the player in the way a quest might be. They’re not goals for the player, but rather, a way to track progress for the designer. They “depict the causality” for the different states and between-states in a game’s narrative

Application to Project Black

While PB will not feature a branching narrative, Ingold’s use of “state modelling” could be useful for us. Our world will have various states based on a few factors, such as how many Natural Wonders the player has healed, how many regions and abilities they’ve

unlocked, et cetera. By ensuring we track these elements, in tandem with a robust priority system for dialogue, we can ensure the world evolves alongside the player's progress within it.

Ideas and dynamic conversation in Heaven's Vault (Jon Ingold, 2018)

Ideas and dynamic conversation in Heaven's Vault

> ***“ The Knowledge Model***
>
> *The knowledge model we use is designed to be strong but efficient to author; so at any given moment in the script, the writer can quickly test if the game-state is appropriate for a line or action (or location!) to become available; with the test usually having two parts:*
>
> *1. Are all necessary conditions met for this action or dialogue line to make sense?*
>
> *2. Are any redundancy conditions met? Has anything happened that would invalidate the action?*
>
> *The first covers things like "unlock this door if you have the right key", but also "don't ask someone a question if you don't think they might know the answer". The second check covers things like "don't try to open a door that's already open", and "don't ask someone a question to which you already know the answer." All requirements are required; but a single redundancy is enough to fail.*
>
> *(These tests are, in many ways, the same! But we find it's useful to have both from an author's perspective because the two define the "space" in which an action can happen: when the player is between one set of discoveries and another. To put that another way, What does this action require? usually includes the triggers for an event or action; and What makes this action redundant? usually includes a list of the consequences of the action - the door is now unlocked, the fact in question is now learned, and so forth.)*
>
> *For most of the game, this test is just right: if the player approaches a door, or an NPC, we can use it to figure out what options we should be providing.*

Line Relevance

In order to tackle the problem of relevance, we've started to think of those "required" states not just as requirements, but also as ideas. We've begun recording not just which states have been set, but what was set recently. What was the last state to be set? That's what the protagonist is thinking about, right now. What were the last ten states to be set? That's what she's been mulling over in the background.

Then, as we run through the prioritised dialogue list, we ask a slightly modified question: is this line valid, and were any of its trigger states set recently? If they were, then the topic is "closer to surface" of our character's mind, and should be pushed up to the player.

\- Jon Ingold, 2018

Application to Project Black

Checking for redundancies alongside our priority system and state checks will help ensure the cohesion and relevance of our dialogue. The idea of prioritising lines based on recent state changes could also be useful to us. For example, if Hafeez and Neisha have just healed a Natural Wonder, we could reprioritise dialogue to take this into account, bumping reactive lines further up in the hierarchy than they might have been otherwise.

Dynamic Banter System (DBS)

Design Documentation

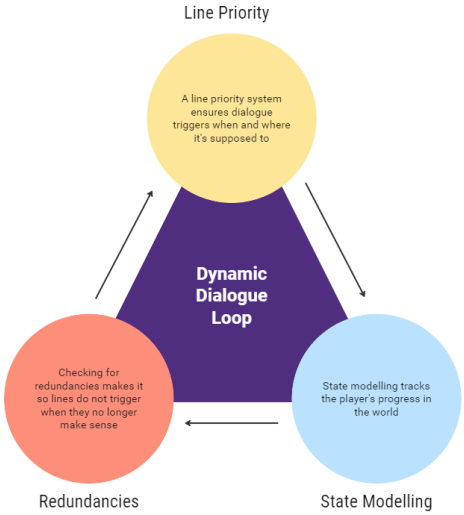
Design Intent

The Dynamic Banter System (DBS) aims to deliver story in a fluid and adaptable way, while enabling us to seamlessly incorporate narrative into exploration. By assigning a priority value to each line of banter, tracking the world's state and the player's progress within the narrative, and making continual checks for when dialogue is no longer appropriate, we'll be able to create a system that moulds itself to each player's individual experience.

Design Goals

- We want the player to feel as though the world and protagonists are reactive to their input
- We want the player to experience the story in a fluid but cohesive order
- We want to avoid giving the player information that is no longer relevant to their situation

Feature Loop



State Modelling

State modelling will allow us to set and monitor conditions for banter, using a robust series of checks to make sure specific conversations trigger exactly when they're needed. All dialogue will have several conditions that must **all** be met.

A potential feature of our state checks could be to keep track of how recently certain conditions were met, and have line priorities shift accordingly, to keep dialogue as relevant as possible.

Redundancies

Redundancy checks will ensure dialogue is removed from the banter pool when it is no longer needed. Lines will need to meet all their conditions in order to play out, but **even one redundancy will remove them from the available dialogue permanently**.

Conditions vs Redundancies
For dialogue to trigger, all its conditions must be met. These conditions are repeatedly checked for. However, once a single redundancy is found, dialogue can be permanently removed from these checks.

Line Priority

All dialogue will be given an INT value that determines its position in a priority system. This way, if the conditions for multiple lines are met simultaneously, dialogue will trigger in an **intentional and predetermined order**.

In the case where two lines of dialogue mistakenly have the same priority, dialogue will be chosen arbitrarily by random number, and an error will throw to tell us to fix the inconsistency.

Example Game Flow

The player is exploring a region after completing its storyline, but has not yet visited or healed the region's Feature. They encounter a vista that shows off the Feature, so we go looking in the banter pool for the appropriate dialogue:

Step 1: Check conditions
All these choices' first conditions (that the player is at this region's Feature vista) are met. But Banter B's second condition is not met, and therefore, it cannot trigger at this time.

Step 2: Check redundancies
Only Banter D has a redundancy, and in this case, it has become redundant since we have completed the region's story. Banter D can now be safely, permanently removed from the pool of available dialogue.

Step 3: Check priority
We now have two valid choices, A and C. Banter C has the higher priority number, so it is selected and executed in game.