Inquire from the Taoism

Technical Design Document

Version1.0

Contents

I. Project Overview

1. Game Concept

Inquire from the Taoism is a story-driven Puzzle RPG game. You are a young Taoist who is trapped in an unknown space. Shuttle in different illusions to experience bizarre stories, use your wisdom and spells to overcome difficulties, find out the truth behind the mystery. Be yourselves and choose your own path to escape.

It is built on the Unity3D engine, supported by C\# scripts. The target platform is PC & Mac and the target audience are teenagers and adults who have interests in eastern history/mythology and Puzzle games.

2. System Requirements

PC Windows ? or newer with corresponding latest DirectX.

Mac Version ? or newer.

3. Technical Goals

● Integrated managers and controllers on different aspects of the game (UI, In-Scene, Scene-Transition).

● Reusable modules (e.g., Observing-Interacting Canvas (OB) system, Dialog system, etc).

● Support for Windows and Mac development environments.

● Loose coupling of OB classes.

● File path based on script/component classification.

● Capability for saving the current game state at any time when the player is allowed to save.

- 60 FPS for basic game laptops.
- Simplicity for non-programmers to edit variables and add simple OB modules in Unity.
-

4. Technical Risks

- High correlation among components hierarchy may stop the development process while one bug appears in one component.
- Multilevel hierarchy among components might mislead programmers to repeatedly implement some functions.
- Compatibility between Windows and Mac OS in both development and released packed versions.
-

5. Tools

- Engine : Unity3D v2019.4.newest-versionf1
- IDE : Visual Studio, Visual Studio Code
- Image Editor : Photoshop, Procreate
- Animation Editor : Adobe AE
- DAWs : Logic Pro X

II. Gameplay

1. Interactions

● All interactions are through mouse and keyboard: (see “Core Mechanics Document2.0” in Teambition)

○ Left click to interact with items and Npcs or open UI interfaces.

○ Right click to observe items

○ Press “WASD” to move the avatars

○ Press shortcut keys to open UI interfaces

● Implementation of special interactions for specific puzzles are intended.

2. Mechanics

Contents Hide

—---

III. Architecture

1. Environment: Folder Structure

- Assets
    - Scenes
        - LevelX
    - Audio
        - LevelX or UI
        - SFX
        - Music
    - Materials
        - LevelX
    - Prefabs
        - LevelX
        - UI
        - InScene
    - Scripts
        - SystemSettings
        - Click
        - UI
        - InScene
        - LevelX
    - Sprites
        - AnimationSpritesheet
            - LevelX
            - Characters
        - LevelX
            - InScene
            - OBAsset
        - Placeholder
        - Test
        - BackpackItem
        - UI
    - Textures

2. Environment: Source Control

- Standard Scrum policies:
    - Pull every time before work days. Always pull before pushing.
    - Frequent commits and pushes, to keep repositories up-to-date.
- Origin - Main **should** always be build-able and working.
- Each contributor works on their own scenes, circumventing conflicts of binary files.
- Any intern's contribution **must** be first developed on a branch and then merged if the intern officially becomes a contributor.

3. Unity: Scene

- One Level contains multiple scenes.
- Each scene with OB **must** have a corresponding Canvas, where a gameobject named "OB" is located, containing all OB modules of the current scene at initialization. GameObjectManager will transfer all OB modules to the "OB" gameobject under MainUI.
- Prefabs **must** be unpacked when added into the Hierarchy.
- The coordinate of the main ground (gameobject) in the scene **must** be at (0,0,0).
- The scene **must** be built according to the main character at the unified scale of 0.5.
- Use *A* Main Camera (*A*: name of the scene) for small scenes like 'WLRoom', use the camera tied to the protagonist otherwise.
- GameObjects:
    - Always maintain a hierarchy of gameobjects (parent->child relationships), it gets messy very quickly otherwise, and exponentially harder to debug.
    - Maintain semantic sense: keep objects that need to be grouped together under a single parent (e.g.: all decorational tree objects).

Contents Hide

—---

- Change gameobject's material to 'LightMaterial' if the scene is light-based.

Contents Hide

—---

4. Unity: Code Structure

There are five levels of hierarchy: system-level, UI-level, interaction-level, scene-level, and puzzle-level (OB-level). System-level managers contain references of all GameObjects and generalised utility subroutines so they are accessible to any other level. It also included scripts related to build settings and scene transition. Instances running at the UI-level are responsible for handling continuous processes (e.g. background music). Controllers of this level each control a specific UI component. All classes at this level have to be highly encapsulated and carefully designed (and rarely touched once finalised, except for variable initialization). Managers in the interaction level handle all user inputs, fire off event delegates that other classes (lower level instances) can subscribe to their specific contextual methods to. Scene level instances handle in-scene GameObjects, the main character, camera and lighting of each scene. OB level instances handle all puzzles appearing in OB. They are instantiated and represent independent puzzles, so they are loosely coupled.

System Level:

- *EnforceRatioCamera*:
    - Enforce the build screen ratio of to 16:9.
- *GameObjectManager*:
    - Save all references to the GameObject publicly available.
    - Save the range for zoom degree in different scenes.
    - Transfer OB modules to the current hierarchy position in OnSceneLoad().
- *LevelLoader*:
    - Call events when each level loaded/first loaded.

UI level:

Contents Hide

—---

Interaction level (*Click* folder):

- *ClickInScene.cs*: Include all Click events in the scene.

● *ClickManagement.cs*: Include all Click events in the UI and any keyboard events.

Scene level:

● *InSceneItem.cs*: Contains Collectable & Interactable properties saved here.

● *PlayerCamera.cs*: Include movement related to the camera under the main character gameobject.

● *PlayerMovement.cs*: Include (lock) movement of the main character.

OB level (Under *UI* folder):

● Include some special OB scripts (e.g. BookOB) and General OB functions.

● *OBDisplay.cs*: On each OB module, include properties related to the current state.

● *OBItem.cs*: On each in-scene object, include the reference of its corresponding OB module.

● *OBManager.cs*: Include open, close, update state, transfer OB to backpack item, effect on interaction for general OB modules.

5. Assets: Art and Audio(Sound)

- Art Filename: : A_XxxXxxx (e.g.: A_MainMenu)
- Audio Filename: : S_XxxXxxx (e.g.: S_Birdcall0001)