‘The Hierophant’
Technical Design Document

By: Small Fry Media Ltd
Version 2.0
November 26, 2019

Contents

1.0 Architecture

The architecture section of the TDD describes all the software components of the game. It outlines the program flow and data relationships.

The core gameplay loop revolves around a series of police interviews and tarot readings. While these two scenarios differ slightly in terms of gameplay, both revolve around selecting tarot cards and providing an interpretation through dialogue choices. Dialogue choice is the main player interaction within the game. Because the game has multiple endings, architecture must be implemented to support this. Dialogue choices, therefore, will be assigned a "point value" within an array. Picking specific dialogue will add a point to several variables. These variables include but are not limited to: "Positive Outcome", "Negative Outcome", "NPC affection". When determining an outcome, the game will reference these points and play a specific cutscene or another outcome. Depending on player points, other scenes and side content may be unlocked.

2.0 Narrative System

The narrative is an essential part of *The Hierophant* and will be implemented through an external addon, Pixel Crushers' Dialogue system. The Dialogue system uses variables to track dialogue choices, character relationships, and narrative outcomes. The Dialogue system uses a node-based dialogue tree, with "Actors" that can be attached to game objects that trigger specific conversations. These triggers are attached to most intractable game objects, as most of the narrative is told through dialogue and inner monologue. The Dialogue system also features a sequencer built into each dialogue node, which allows for enabling and disabling game objects, character sprites, and backgrounds. Audio dialogue can also be added to the sequencer, which plays along with the conversation.

Store Link:
https://assetstore.unity.com/packages/tools/ai/dialogue-system-for-unity-11672

Documentation:
https://www.pixelcrushers.com/dialogue_system/manual2x/html

2.1 Dialogue System Breakdown

This is a breakdown of Dialogue System components. If your scene does not contain the essential components, the dialogue system **will not function.**

- Dialogue Actor (Non-essential): Attached to a Gameobject to denote an actor

- Event System: Essential Component that manages events. Default settings should be used
- Selectable: Allows an Object to be selected by the player
- Standalone Input module: Manages Input
- Base Input: Subcomponent of Standalone Input module
- Selector: Assigned to Player Object (Currently an Empty labeled “Celestine”), allows the player to select items with the Selectable script
- Dialogue System Trigger: Assigned to almost all interactable game components. It can be set to call a specific conversation with several variables. For this to work, the object must be assigned a box collider and the selectable component
- Dialogue Database: The base component of the Dialogue system, manages actors, conversations, and variables. Conversations are node-based, with each node featuring a sequencer and a code section.

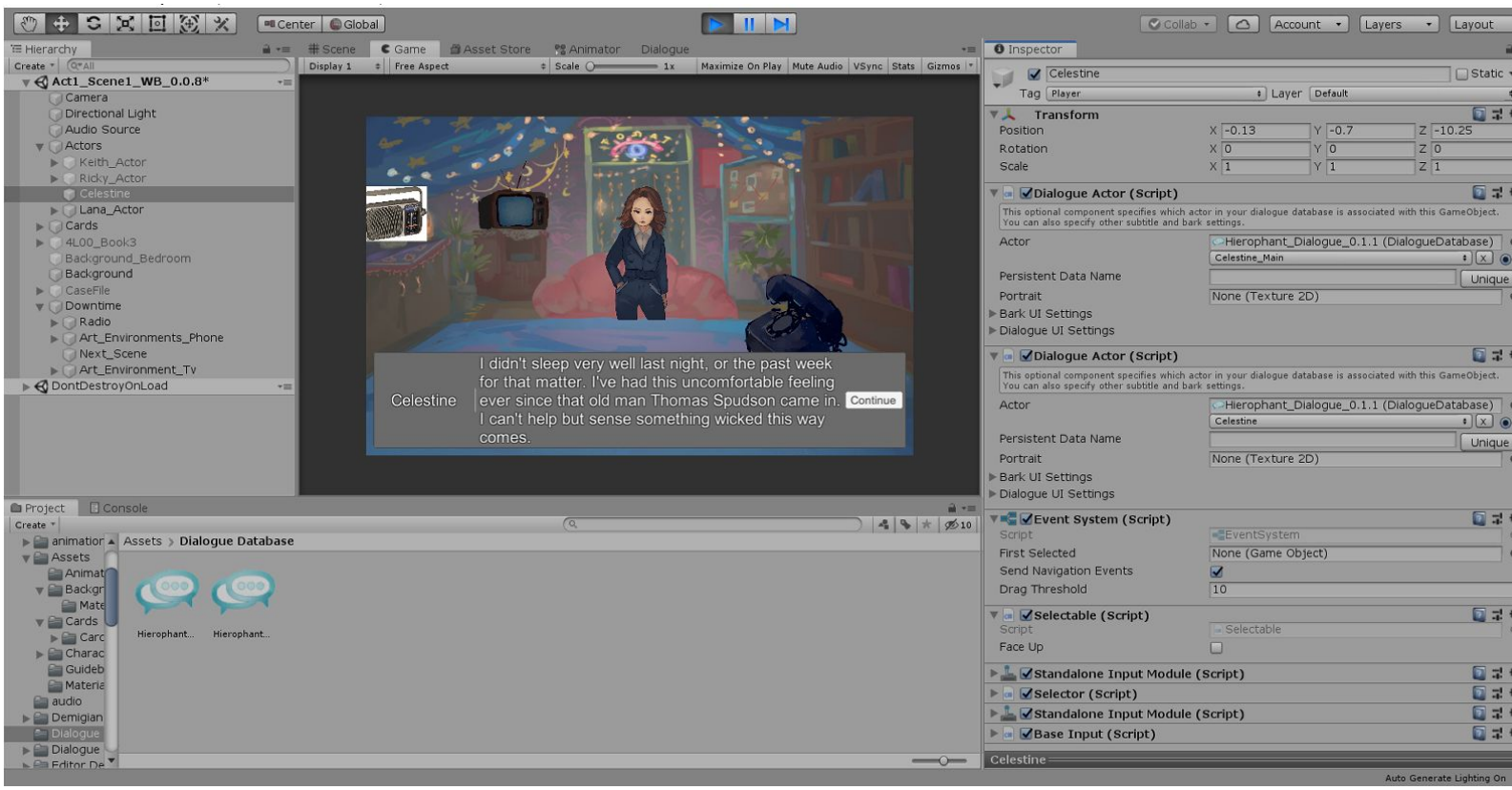


Fig. 1- Breakdown of Scene components and the Dialogue System

3.0 Target Platforms

The target platform for this game is PC. While visual novel-style games do exist on consoles, primarily the Playstation 4, PC is a preferred platform and is easier to target for development.

4.0 Environments

4.1 Play Environment

The play environment will be on a Windows PC, primarily with mouse and keyboards.

4.2 Development Environment

We will be using the Unity Engine version **2019.2.0f1** and will be using its native c# coding language. The development will primarily be accomplished in Brock University provided labs. Unity is also recommended to be installed on the Technical (also known as Tech) team's personal computers.

5.0 User Interface

The Tech team will consult with the Art Team and Design Team to ensure that the interface remains attractive and functional. Much of gameplay interaction is based around the dialogue system and the core gameplay loop detailed above. The Tech team will consult primarily with the Design team to ensure that gameplay systems interact properly. The user interface will consist of a dialogue system that will display the dialogue between characters. The player will come across prompts when they will have a decision to choose between various dialogue options. Other interfaces will be used for interactable objects during downtime which consists of a radio, television and a phone. For a list of each interface component and the components assigned to it, see section **8.0 Unified Modeling Language**

6.0 Role assignment and Inter-team Communication

Roles and responsibilities are managed through a shared Google Sheets list. UI Assets are assigned to individual team members, who are responsible for incorporating the asset into the game engine. To assist with this, each asset has also been given a consultant from each of the other teams. This should create a clear chain of communication and accountability for the implementation of game assets.

Number	Game Asset	File Name	Assigned to	Design Consultant	Art Consultant	Narrative Consultant	Sound?	White Box
1	Crime Scene Match/Exam.		Serena/Brian	Serena	Brett	Brian/Diane	No	In Progress
2	Dialogue System		Noah	Noah	Justin	Rachel	Yes	In Progress
3	Newspaper (Secondary)		Noah	Diane	Brett	Diane	No	In Progress
4	Card Selection (Gameplay)		Brian	Noah	Bennett (Back)/Brett (Front)	Rachel	Yes	In Progress
6	Tarot Guide		Noah	Serena	Brett	Rachel	No	Pending
7	Evidence/Case File		Noah	Serena	Khalid	Diane	No	In Progress
8	Television Set (Dialogue-Secondary)		Serena	Noah	Brett	Rachel	Yes	Pending
10	Rotary Phone (Dialogue-Secondary)		Serena	Noah	Hayley	Rachel	No	Pending
12	Game Sound		Serena	Alex	N/A	TBD	Yes- Alex	Pending

7.0 Technical Standards Guide

Our Tech team does not have a great deal of coding experience but will follow the basic coding standards detailed below. There will be a great deal of very simple technical terms in these coding standards. Parts referenced from Microsoft coding standards: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions\

Basic Unity Elements

Saving your Project:

- Saving Unity files does not necessarily mean saving your scene. Make sure that your scene has been saved, and be sure to commit any changes to the SVN.

Scene view:

- Where you construct your game; where you add models, cameras, and other pieces that make up your game
- *Asset Sizes and Scale to be noted here*

Game view:

- The player's perspective of the game. Where you can play your game and how the mechanics work together.
- You can change the aspect ratio to match the screen size of the platform you wish to use *Aspect Ratios TBD*

Hierarchy Window:

- It contains all of the game objects being used in your game.
- Game objects are empty containers that are customizable through adding components
    - Components permit game objects to project geometry, emit light, act as a camera, or create complex behavior via scripts
    - They can also act as folders, containing other game objects, which is useful for organizing scenes
- A scene typically represents a single level of your game, although you could theoretically put the entire game in one scene.

Project window:

- It contains all assets used by your game. You can organize assets by folders. Drag and drop what you need into the hierarchy window.
- Do not make changes to assets directly in the file system because it could break your assets as well as your game.

- Unity contains metadata for assets so moving between folders on the file system breaks the metadata. Any organizational changes to assets should be done in the project window

Inspector window:

- Lets you configure any game object
- You can alter objects while the game is being played
- You can enable and disable any game object with the checkmark at the top of the inspector. **This is important to note due to the high level of enabling and disabling of objects such as character sprites and backgrounds.**

Toolbar:

- Use the toolbar to manipulate the various game objects in the scene view.
- If you lose track of an object, double click to bring it back to center in the scene view
- The hand tool allows you to pan the scene (Q)
- The translate tool lets you select and position a game object in the scene (W)
- The rotate tool lets you rotate objects (E)
- The scale tool lets you scale objects in a single axis or proportionately on all axes (R)
- The rect tool lets you resize, scale, and rotate 2D assets. (T)
- The gizmo display toggles are switches that control how you position gizmos in your scene. The first switch toggles between center and pivot mode. When in center mode if you select two game objects, the gizmo will be placed in the center of the two. In pivot mode, the objects rotate on its own pivot point
    - The second switch toggles between global and local space. In global mode, you're manipulating your object from the point of view of the world around it. Local mode works from the coordinate system of the object itself and changes the axes to match the game object

Play buttons:

- The play button lets you start and stop your game
- The pause button allows you to make changes, though, they have to be experimental since once you click it again the changes disappear
- The step button lets you step through your game one frame at a time
- The layers button is used for things such as preventing the rendering of game objects from physics events like collisions
- The layouts button lets you create and save the layout of views in your editor and switch between them.

Naming Conventions

To help separate team-created variables, they should be denoted by both camel case, and by including an underscore in front of the namespace.

Example:

```
public class EnemyMove : MonoBehaviour
{
    private UnityEngine.AI.NavMeshAgent _nav;
    private Transform _player;
    public int _currentRandomPoint;
    public float Distance;
```

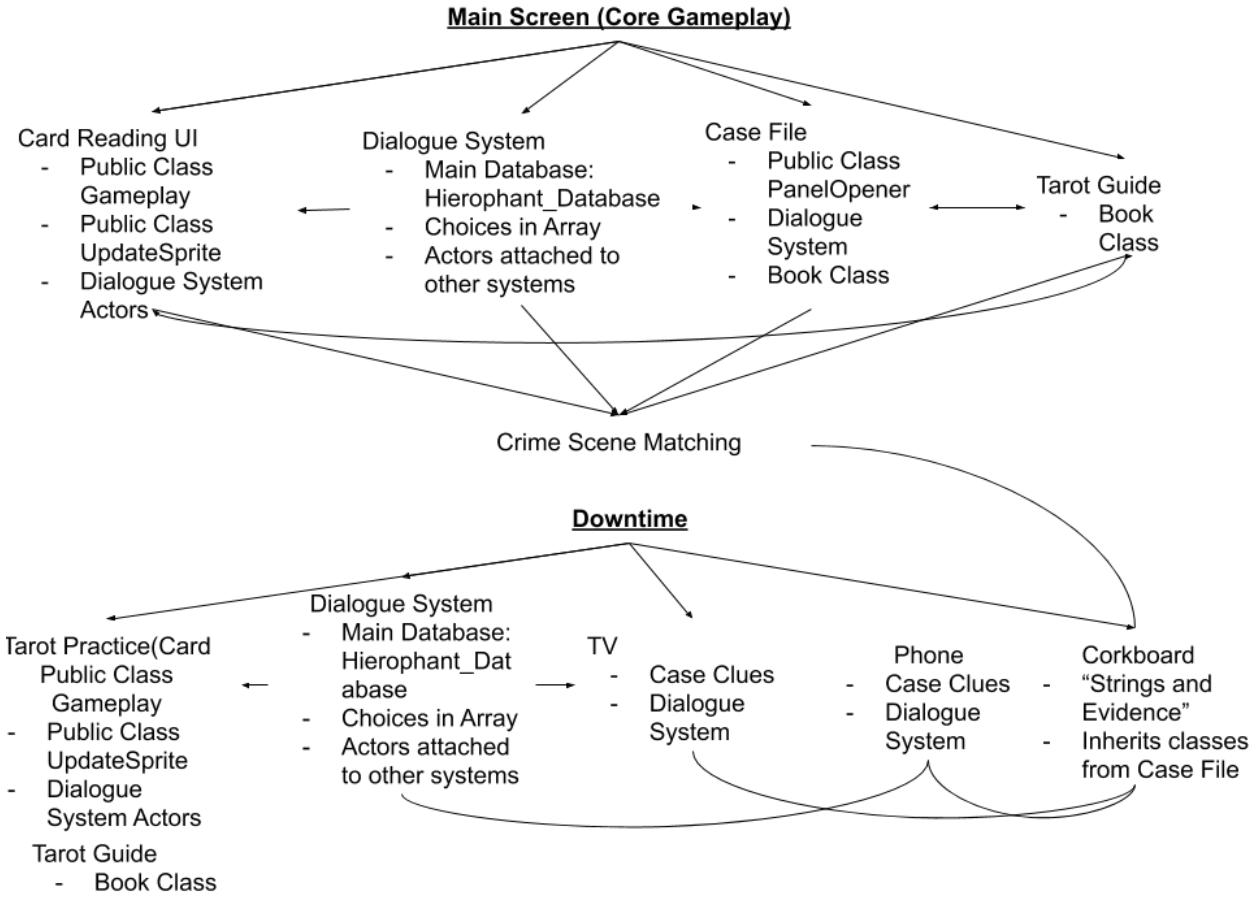
Layout Conventions

- Write only one statement per line.
- Write only one declaration per line.
- If continuation lines are not indented automatically, indent them one tab stop (four spaces).
- Add at least one blank line between method definitions and property definitions.
- Use parentheses to make clauses in an expression apparent, as shown in the following code.

Example:

```
void OnCollisionEnter(Collision other)
{
    if (other.gameObject == _player)
    {
        _player.GetComponent<playerValues>().TakeDamage(1.0f);
    }
```

8.0 Unified Modeling Language



9.0 Tools

9.1 Hardware and Software Use

- Hansoft
- Adobe Suite
    - Illustrator, Photoshop, After Effects
- Unity Game Engine
- Logic Pro
- Wordpress

9.2 Game Engine

Unity Game engine is a free, easy to learn platform, with relatively simple architecture and framework. The team will use **Unity version 2019.2.0f1** on Windows. No other versions or operating systems will be used to avoid data issues.

A backup engine would likely be a dedicated visual novel creator. Tyranobuilder is an inexpensive and powerful VN builder, but there are many free options.

9.3 Application Programming Interfaces

Project Hierophant uses Unity and therefore uses the native C# script API. For example:

```
using System;
using UnityEngine;
using System.Collections.Generic;
```

9.4 Art Tools

The Art team uses a variety of tools based on their skills and competencies. These include: Adobe Illustrator, Photoshop, After Effects, Procreate, Clip Studio Paint, Paint Tool Sai. Art team animations are made in the Unity engine, since running multiple .Mp4 instances caused performance issues, and in-engine animations proved less taxing.

Because of the wide variety of art tools used, the Art team and Tech team have set up a system to ensure that art remains at a consistent quality. See section 6.0.

Link to the art document for further information.

9.5 Graphics

Design and Art consultants will assist with ensuring that graphics remain at a consistent level of fidelity. The best way to ensure this is to follow the guidelines detailed in the Media formats and Inter-team communication sections.

9.6 Hardware/Software

Unity executables are designed to be run on most Windows environments. We do not expect this game to require a high-end computer to run. From the Unity website:
Desktop:

OS: Windows 7 SP1+, macOS 10.12+, Ubuntu 16.04+

Graphics card with DX10 (shader model 4.0) capabilities.

CPU: SSE2 instruction set support.

(https://unity3d.com/unity/system-requirements)

9.7 Third-Party Technology Dependencies

Unity provides a suite of extensions and add-ons to assist with game design. We will likely be depending on add-ons for a number of systems. Currently, the Tech team is planning to use addons to manage dialogue, with others being added as needed.

10.0 Risks and Contingencies

The greatest risk to the Tech team is ultimately a failure to onboard the relevant skills and experience to properly implement game systems and ideas. By using tutorials, department resources, attending code-a-thons, game jams, and other events, we can likely mitigate this, but it is ultimately the responsibility of the Tech team to manage this risk.

10.1 Loss of Data

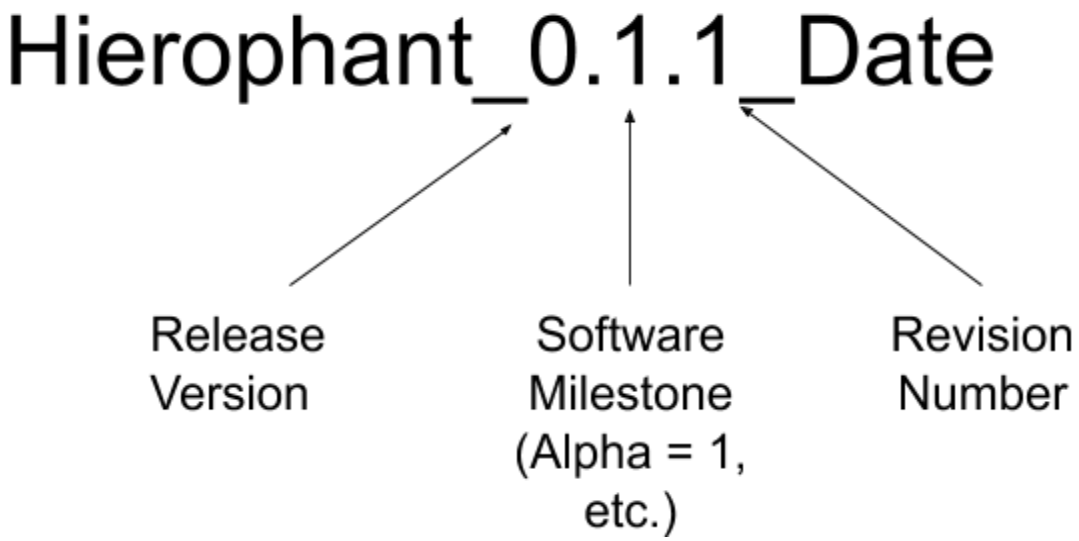
Frequent SVN backups, version controls, and backups on local machines should mitigate any data loss. To avoid accidental loss of data, non-tech team members will need to ask for the Tech team permission before accessing or editing the main unity scene. Consistent tagging of builds and versions allows the Tech team to remain current. Data loss or SVN issues should be reported to the Tech lead, where a previous version will be pulled. One of the best ways to restore data is to revert a file to a previous version, then copy and rename the previous version. Team members should declare what files and scenes they are working on in the “SVN Checkout” discord channel.

10.2 Schedule Slips

The Tech team will strive to set goals that are reasonable for the team to accomplish. Personal events or incidents will be taken into account, and the rest of the team will have to pick up any work leftover from the missing team member. One likely scenario is SVN downtime, possible causing conflict as tech members work with older versions of the build. The main way to avoid such scenarios is to have tech members update their SVN as much as possible.

10.3 Revision Control

Revisions will be handed through an SVN repository. Revisions will be handled through tagging in the SVN, and a numerical revision system will be established. We will be using the system detailed below.



11.0 Weekly Builds

A weekly build will be tagged in the SVN as a starting point for any tasks for the week. The version number will be tagged along with the date.

12.0 Input/Outputs

Input is to be handled by the mouse cursor and left click for selecting Tarot Cards and Dialogue options. During other sequences of the game, the player will also input actions with the mouse. When reviewing the evidence, the player will be able to move them around with mouse control. When selecting items that call a dialogue system conversation, a prompt will appear for the player to press a button to start the conversation (currently Space). The output will be displayed by showing the player how objects react when they are interacted with. This can be achieved by highlighting objects when they are interacted or displaying a new interface when an object is meant to be displayed.

13.0 Media Formats

Unity is relatively flexible with file types, but a few general guidelines should be established. All media should be exported in lossless file types. All assets should be titled clearly, using underscores. Names should follow the category-specific name.

General Asset Example:
team_category_itemversion

Sound Team Example (.WAV files):
audio_SFX_cardShuffle03.wav

Narrative Team Example (.docx files):
narrative_case01_theFoolsErrand.docx

Art Team Example (.psd and .png files):
Art_tarot_lover_01.psd
art_tarot_lover_01.png

Tech Example (.asset files):
act1_scene01

14.0 Performance Goals

To be decided

15.0 Sound

The soundtrack for this game will be composed using Logic Pro X and FL Studio. Other software(s) will be used by sound team members to create samples or soundscapes to assist the Audio Lead, this software(s) includes GarageBand, Adobe Audition. Logic Pro X will be used for editing and the final mixdown. There will be other assets recorded using dedicated hardware specific to music composition. External hardware includes a midi keyboard, electronic drum set, and Scarlett 2i2 audio interface. Audio will be mixed on Mackie MR624 studio monitors and referenced on Mackie Cr3 monitors.

16.0 Pipeline