Steam Game Selector V0.1 TDD

Goal:

The goal of this project is to make a game selector that will launch the executables of steam game files. This is a very broad definition, but hopefully the rest of this document will be suffice in documenting what is needed by this product.

Overview/Summary:

The viewer experience is as follows, the viewer can have a playable character to be able to move around a room for the starting screen. Then be able to go to several objects (a computer or exit for now) and select them to select a different screen. The main two to be mentioned is the door object and a computer object.

The door object will lead to a menu asking if the viewer would like to exit (either yes or no). If yes, then the viewer will leave the application, if no, then the menu will stay.

The computer object is in there to show the game selection menu, where a game can be selected and played.

Tasks for a minimum viable product:

- Controllable character to control (done)
- A menu to select and launch a selected game (done, needs work with controller selecting the games in the dropdown)
- A way to exit the application (done)
- Capable of being used with either a mouse or a Controller(Done)

//TODO: Create a Beta to see what is a good way to structure the whole thing

Steam Game Selector V0.2

Goal:

The goal is to make a game selector that is easier to code with and more structured/clean code. Most is done through V0.1 to see what is needed.

Overview:

The purpose of V0.2 is to improve documentation and make it easier to add stuff later on. Taking a scene by scene approach that doesn't involve much overlap between scenes is what seems like it is needed. Strict use of scripts being limited by what scene they are used in. If there is a script that is global, it would have to be considered "global."

Minimum Viable Product for V0.2:

- Create a menu for adding and deleting paths to be selected to be able to start an application
- Create a menu for being able to play apps
- Create a player selection screen to select the menu
- Make aesthetics for the computer menus (creating sprites and animations for computer)
- Make sizing for menus work smoothly and consistently for a 16:10 grid (at least looking well while keeping that aspect ratio)

Naming Conventions:

Versions:

The version doesn't signify much except the order that the steam game selectors are created, so it will just ascend until there is something that it does change with.

Namespaces:

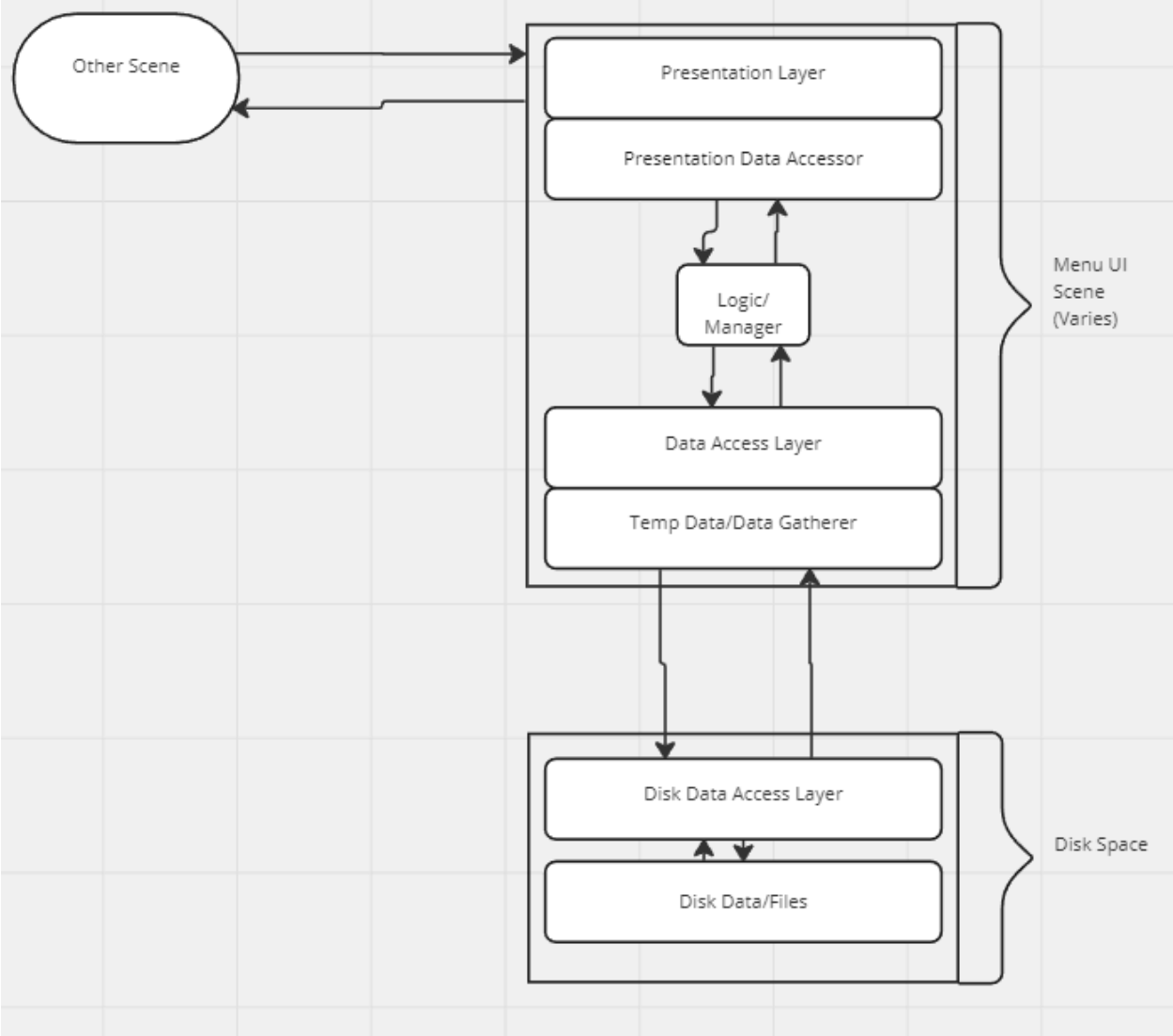
- [Scene]Scripts: Something that belongs to a specific scene and only that one scene.
- GlobalScripts: Something that is extended to multiple scenes.

Scripts:

- Private Variables should have _scriptName;
- Protected and public variables should be named scriptName;
- Static scripts should be named FunctionStatic

Architecture:

Menu UI Architecture (generalized):



<u>MenuUI Scene</u>

Presentation Layer: Used to show and select (in cases of user input) pieces of data to do events in UI.

Presentation Data Accessor: Used to allow global use of data the user inputs and selected. This is generally dropdowns and input game objects (buttons can use direct connections)

Logic/Manager: Used to set the logic used by the program to access and use the data grabbed (left generalized because different menus will different computations, therefore needing different managers)

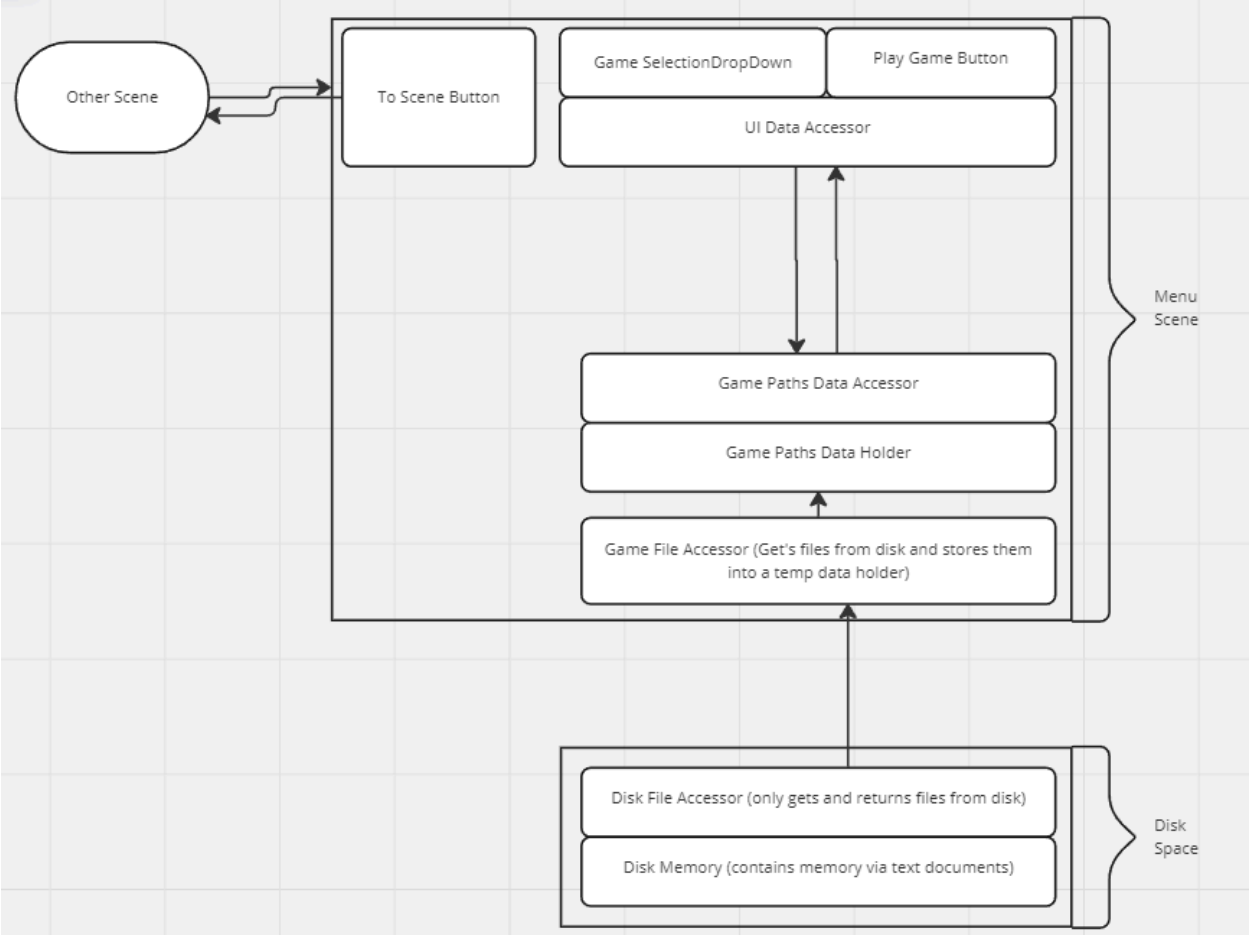
Data Access Layer: A global static script used to grab data set in the temp data so that there are no direct calls to the gameobject data holders.

Temp Data/Data Gatherer: Used as a gameobject to store data to be able to see in the editor but also gathers the data from the disk.

Disk Space (Can switch with Unity Player Prefs):
Disk Access Layer: A global script to access files that are in disk space while being interchangeable to what is being hooked up to.

Disk Data/Files: Where disk data is stored.

Game Selector Architecture:



Menu Scene:

UI Data Accessor: A static script used to access and set the data of the UI elements

Game Paths Data Accessor: A static script used to get the paths of the games defined in the disk space

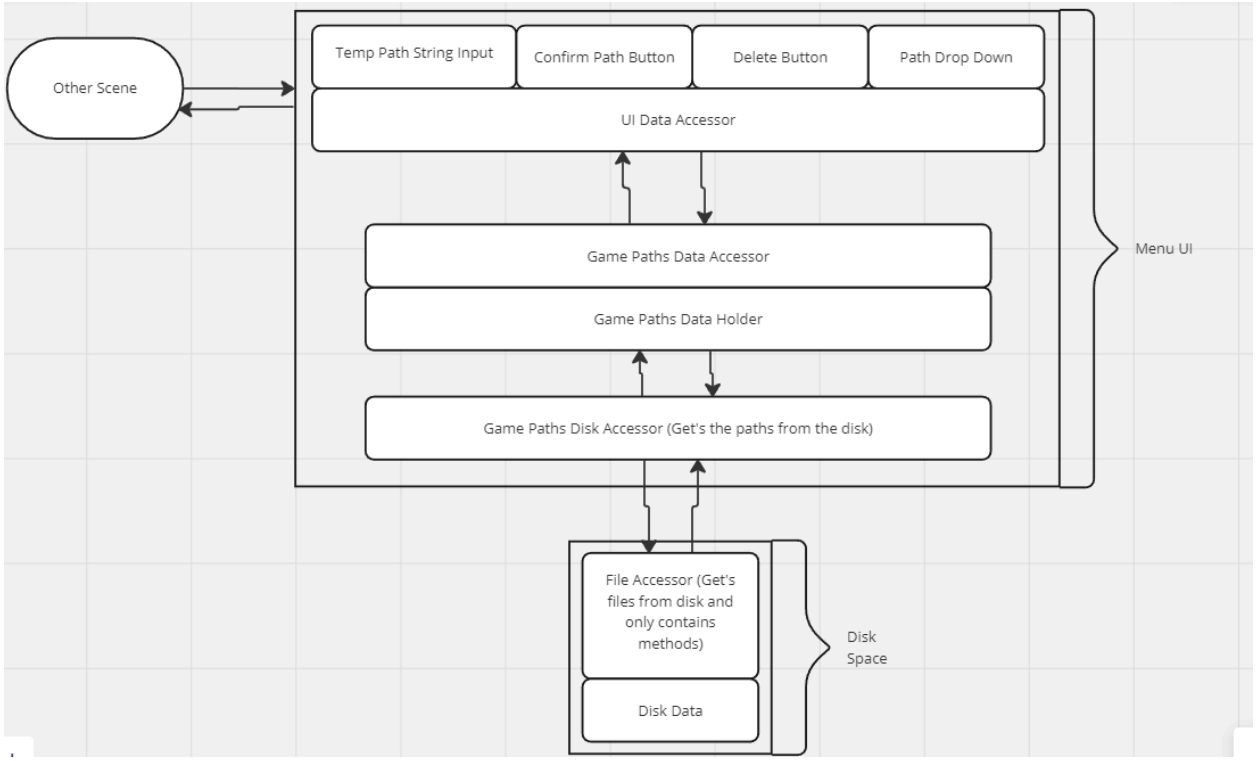
Game Paths Data Holder: The data holder that set get's the data from the file accessor and holds them in a game object (this is a monobehavior script).

Game File Accessor: A static script used to get the files from the data accessor script.

Disk Space:

Disk File Accessor: The disk file accessor that is stored in unity, and only gives methods that allows the editor to get access to text files and their contents

Creating Paths Scene:



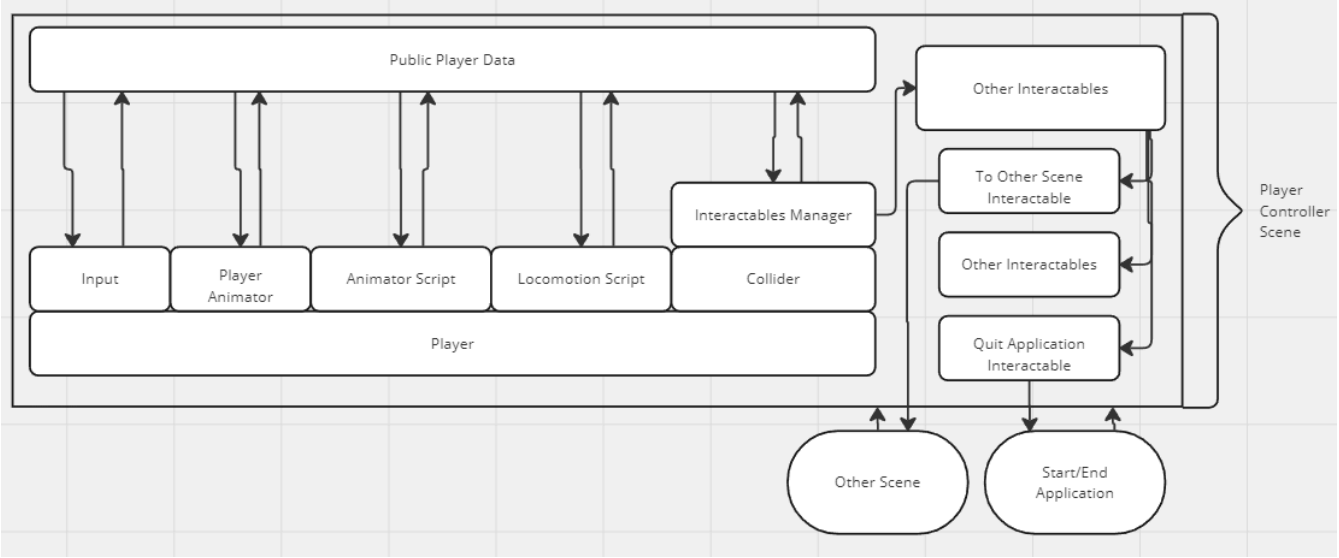
Menu UI:

Similar to game selector architecture. Only difference is the purpose of the UI, where someone types in a path, and pressing confirm puts it in the game paths accessor if it exists.

Disk Space:

Again, see Game Selector Architecture for same information.

Player Scene



Player Controller Scene: The scene where the player can choose what menu to show up. This will show which scripts that is important to pull up.

Public Player Data: A global script used to make the player scripts accessible to other scripts (primarily to other player scripts).

Player: The player gameobject used to control what is being moved and what is being selected.

Interactables Manager: A Interactables detector used to detect objects that are interactable that are being collided with with a IInteractable interface.

Input: The input manager containing the inputs of the player character, as well as holding the methods to give out considering what button is pressed.

File Architecture:

Overview:

Files would be sorted based on which scene they are being applied to, for example, there would be one main file per scene such as the player scene and menu scene. Each would have their own script folder (which sub folders will vary), prefabs folders, etc. Plus there would be an extra folder to get universal stuff that would be used by multiple scenes (such as a “ToScene” button.

Example:

Menu File Architecture:

The menu architecture doesn’t have to be complicated and will vary based on the type of menu that is being developed. But each menu file set will contain the following at minimum:

- Scripts
- Scenes
- Prefabs

Player Scene Architecture:

The player scene architecture is more akin to what is contain in normal top down games. It should contain at minimum, the following:

- Scripts
- Scenes
- Prefabs
- Animations
- Sprites

Packages:

- New Input System
- Everything related to 2D sprites
- Text Mesh Pro

Assets needed to design:

Player Scene

- Player Sprite
    - Idle
        - North
        - South
        - East
        - West
    - Walking
        - North
        - South
        - East
        - West
- Wall Sprite
- Floor Sprite
- Accessories (optional)
- Desk With Computer Sprite (Animation Optional)
- Bed Sprite
- Door Entrance Sprite (Just some indication where to leave the application)
- Interactable indication sprice

Game Selector

- Computer Screen Sprite
- Drop Down Sprite
- Button Sprite
- Background Sprite
- Blocky Text Font

Game Path Editor

- Computer Screen Sprite
- Input String Sprite
- Background Sprite