



**POTION SHOP: GONE ASTRAY**  
*UNITY PROJECT DOCUMENTATION*

Last Update: 01.16.2024

# Table of Contents

Introduction.....	3
Customer.....	4
First Person Controller.....	5
Game Manager.....	6
Importing Images into the Project.....	7
Inventory Controller.....	8
Item Data.....	9
Item Object.....	10
Item Plane.....	11
Load Potion Shop On Collision.....	12
Materials Use.....	13
Maze AI Controller (and Wander Enemy).....	14 - 15
Maze Spawn Manager.....	16 - 17
Music Box (and Music Enemy).....	18
Order System.....	19
Potion Crafting System.....	20
Potion Data.....	21 - 22
Recipe.....	23
Recipe Book.....	24
Recipe Book Manager.....	25
Stamina System.....	26
Teleportation.....	27

# Introduction:

Hello Reader!

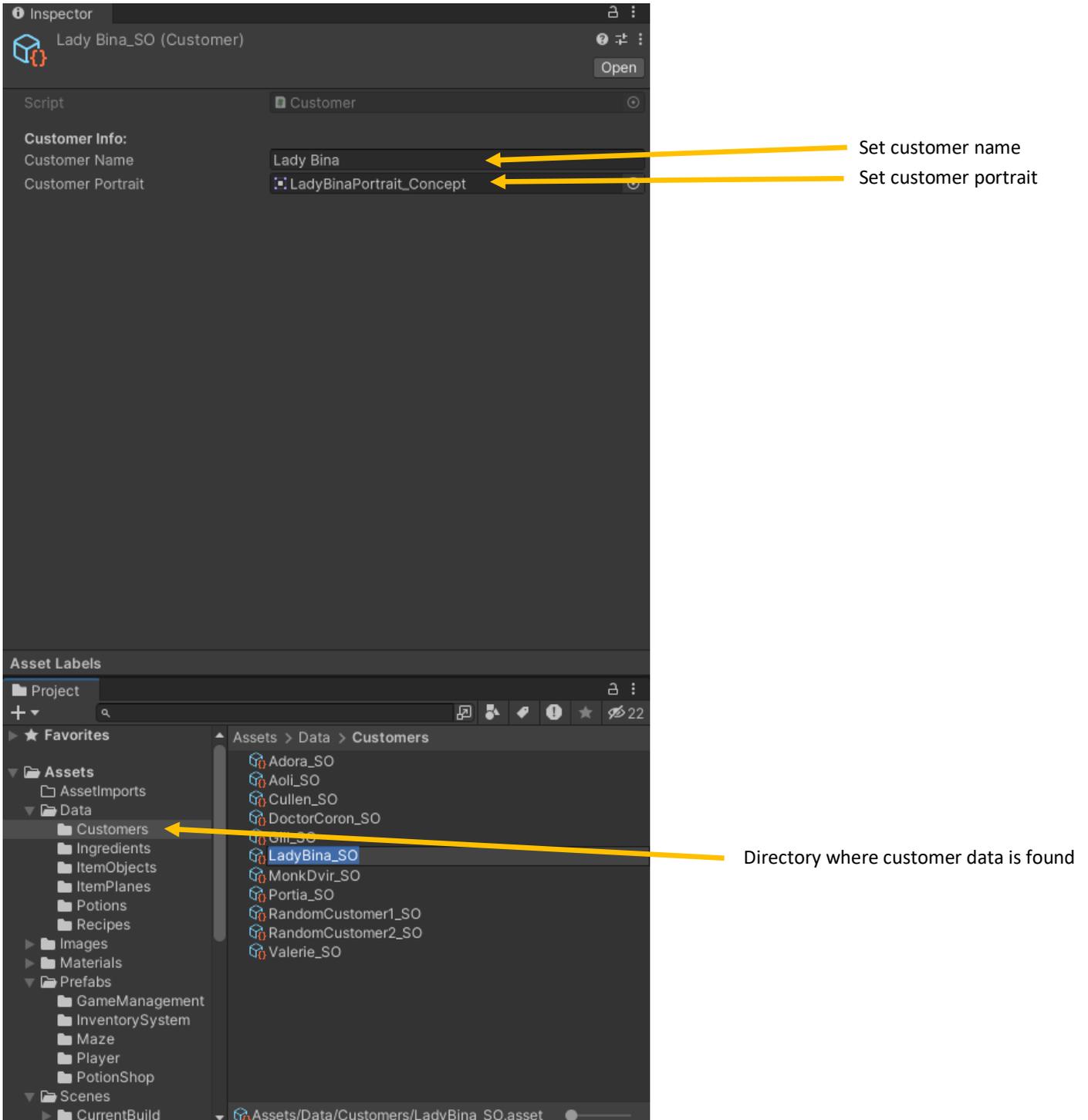
I'm excited that you've decided to learn more about this Unity project! I've created this document for the 2023-2024 UNM Gaming Capstone project titled "Potion Shop: Gone Astray", specifically as a guide to the Unity project itself. The game was designed with many placeholders to implement our finalized media elements later on, including our theme, images, sounds, and animations, to name a few. The purpose of this document is to help provide insight into how the game runs, as well as to highlight the major areas where we will be integrating-in our finalized media elements. Please let me know if you have any feedback, questions, or comments, and I hope this document will be useful to you!

Cheers,

Christopher DeBonis  
*Lead Game Developer*

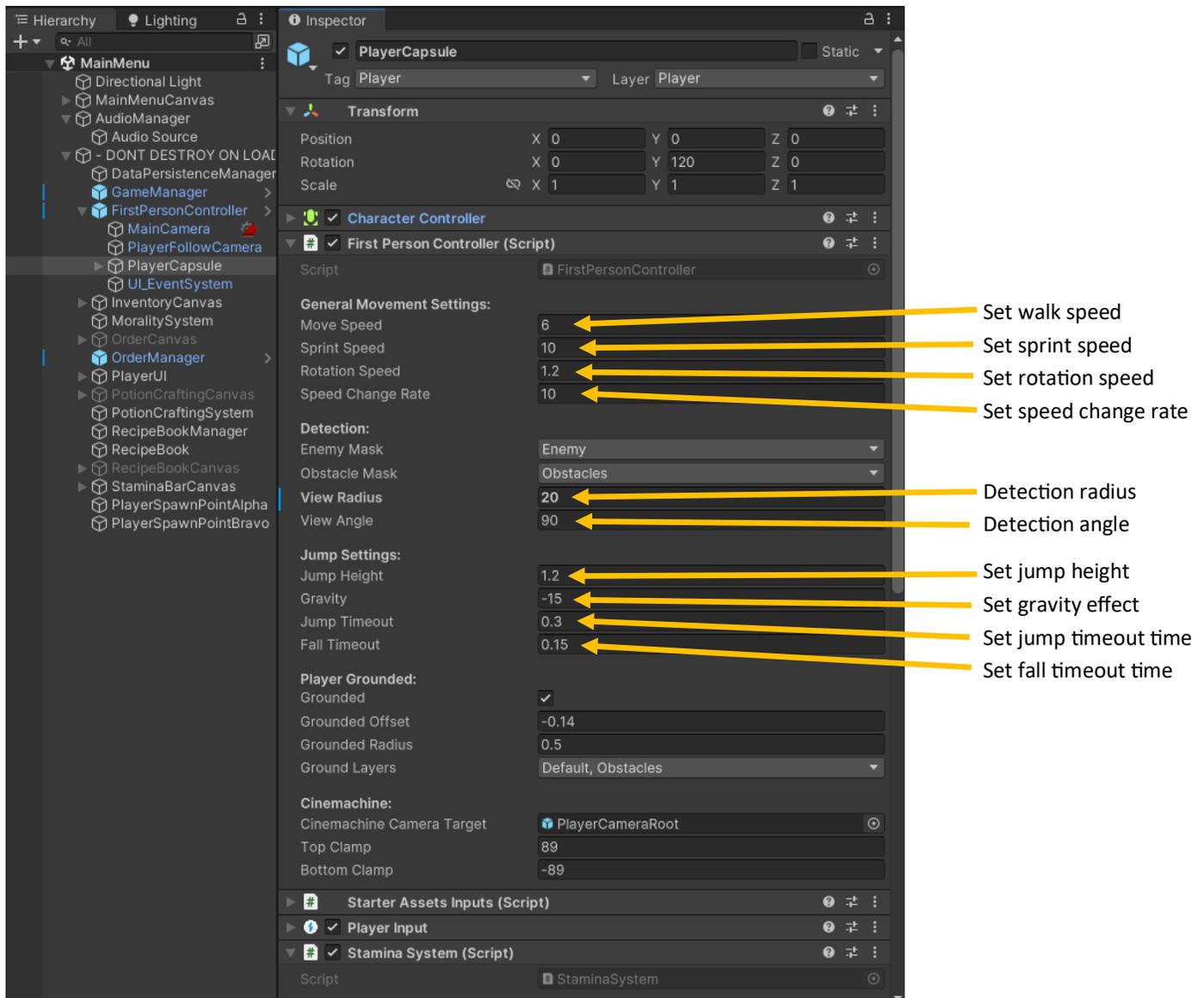
# Customer:

The customer game object holds the name of the customer and their portrait. The customer is used in creating randomly-generated orders, whereby the customer name and portrait is displayed on the order board in the potion shop.



# First Person Controller:

The first person controller is responsible for translating user input into interaction with the game world. The settings here include the player's movement speed, detection, jump height, effect of gravity, rotational speed (how fast they turn), and speed change rate (inertia).



# Game Manager:

The game manager handles keeping track of global game settings, such as the amount of time in the day, spawn points, player currency, landlord payment amount, and UI areas that get passed data during runtime. The game manager persists throughout the game and is never destroyed.

The screenshot shows the Unity Inspector for the 'Game Manager' script. The settings are as follows:

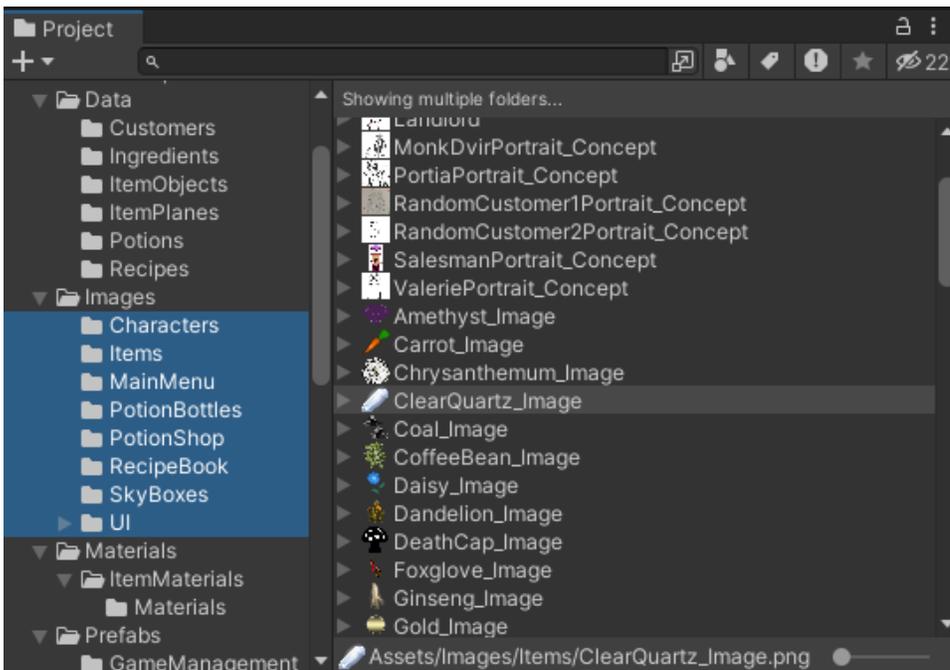
Category	Property	Value
General Settings:	Controller	PlayerCapsule (First Person Controller)
	Landlord Payment	1400
Time and Day Settings:	Time In Day	900
	Is Timer Running	<input type="checkbox"/>
Maze Settings:	Player Capsule	PlayerCapsule
	Drop Spawn Location	DroppedItemSpawnPoint
	Alpha Player Spawn	PlayerSpawnPointAlpha
	Bravo Player Spawn	PlayerSpawnPointBravo
	Charlie Player Spawn	None (Game Object)
Delta Player Spawn	None (Game Object)	
Canvas Settings:	Potion Crafting Canvas	PotionCraftingCanvas
	Order Canvas	OrderCanvas
	Door To Maze	DoorToMaze
	End Of Day Canvas	EndOfDayCanvas
	Win Loss Canvas	WinLossCanvas
Pause Menu Canvas	PauseMenuCanvas	
Text-Related Settings:	Time Remaining Text	TimeRemainingText (Text Mesh Pro UGUI)
	Time Of Day Text	TimeOfDayText (Text Mesh Pro UGUI)
	Current Day Text	CurrentDayText (Text Mesh Pro UGUI)
	Player Currency Text	PlayerCurrencyText (Text Mesh Pro UGUI)
	Landlord Payment Text	LandlordPaymentText (Text Mesh Pro UGUI)
	Win Or Loss Text	WinOrLossText (Text Mesh Pro UGUI)
	End Of Day Text	EndOfDayText (Text Mesh Pro UGUI)
	Day End Player Currency Text	DayEndPlayerCurrencyText (Text Mesh Pro UGUI)
Day End Landlord Payment Text	DayEndLandlordPaymentText (Text Mesh Pro UGUI)	

Annotations with yellow arrows:

- Set landlord payment amount (points to Landlord Payment: 1400)
- Set total time length of day (points to Time In Day: 900)
- Set Alpha maze spawn point (points to Alpha Player Spawn: PlayerSpawnPointAlpha)
- Set Bravo maze spawn point (points to Bravo Player Spawn: PlayerSpawnPointBravo)
- Set Charlie maze spawn point (points to Charlie Player Spawn: None (Game Object))
- Set Delta maze spawn point (points to Delta Player Spawn: None (Game Object))

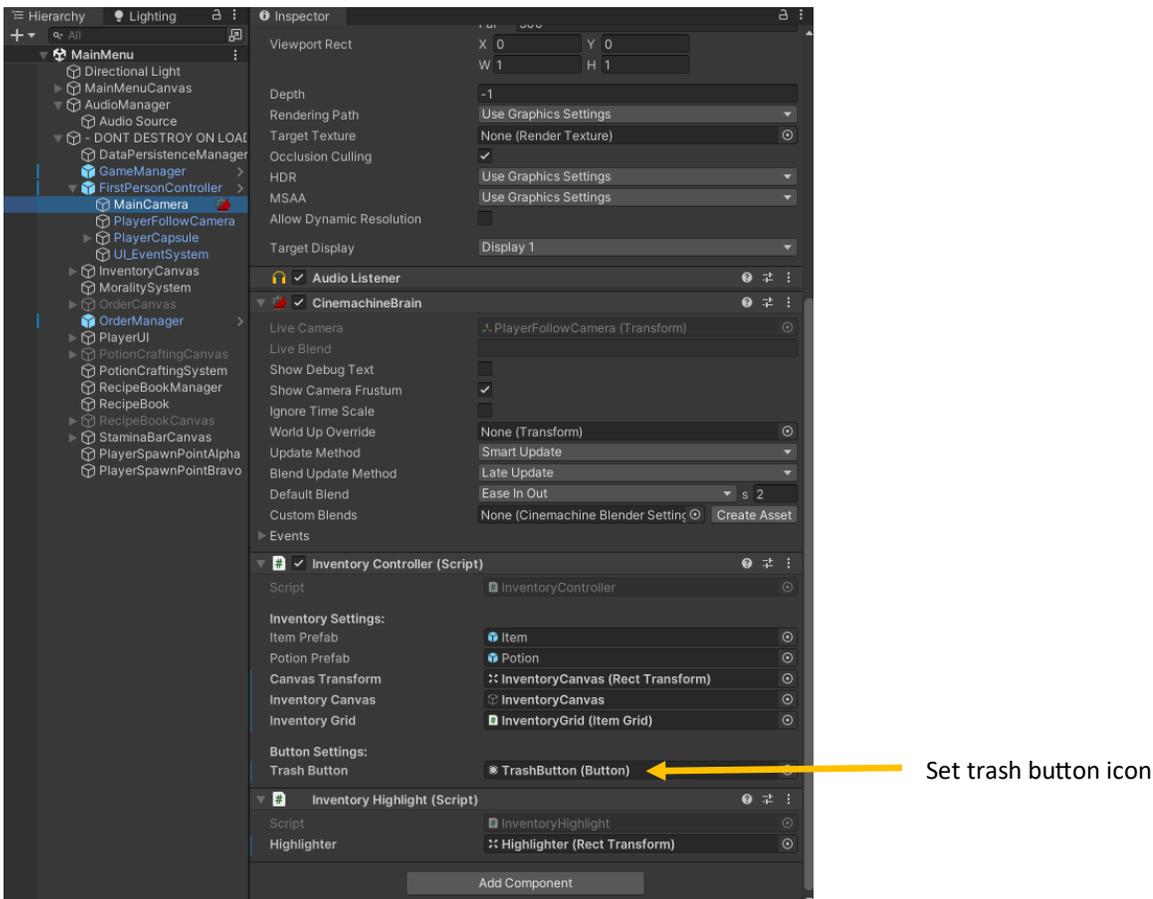
# Importing Images:

Images should be imported into the images folder, and labeled as <name>\_Image. The images should also be converted to 2D/UI sprite objects upon import if they are to be used with the Unity UI system.



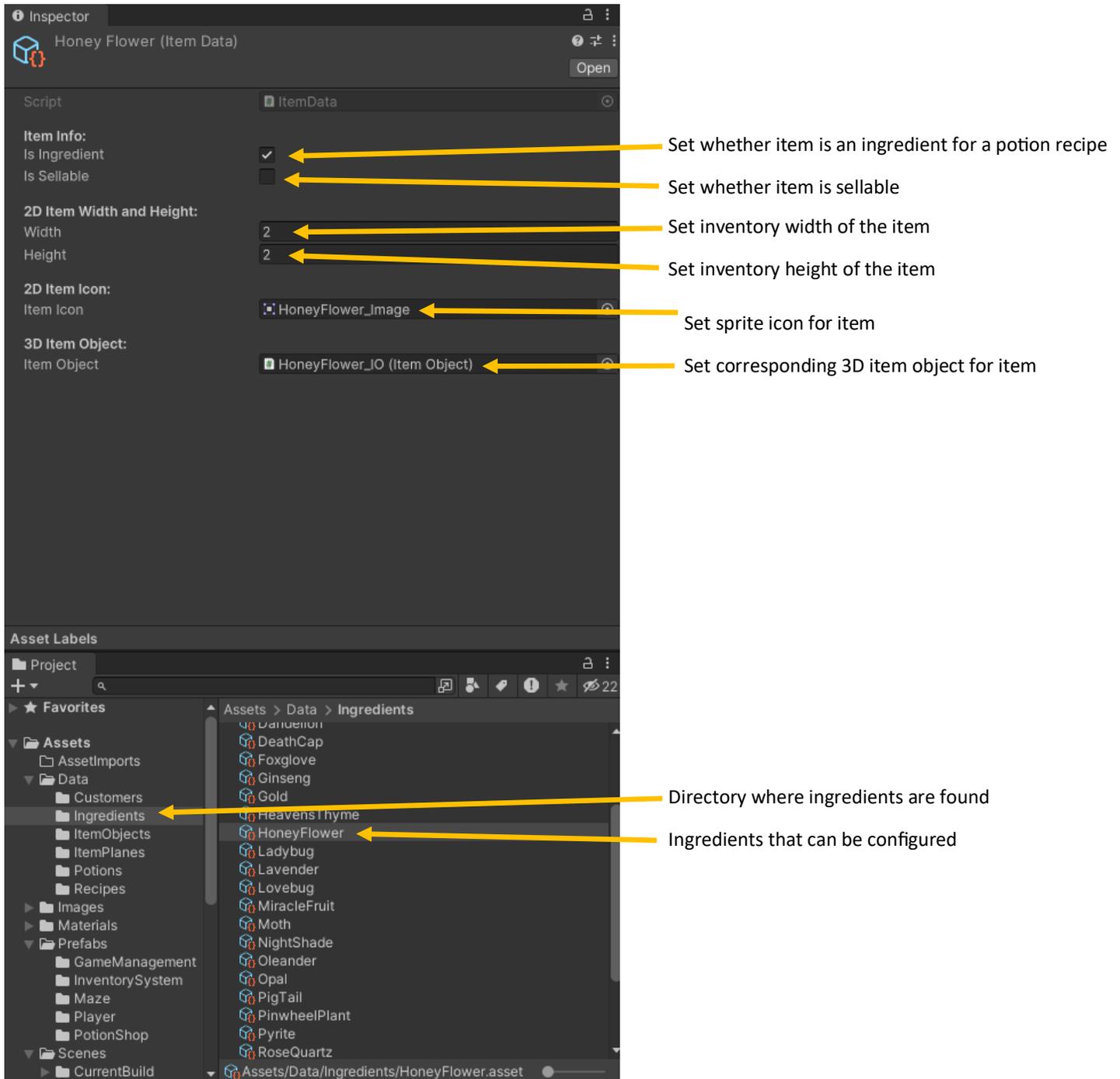
# Inventory Controller:

The inventory controller is where the logic for the inventory system is located. Most settings in here will not need setting or changing. The exception is that the trash button icon can be reset to a different image, when it comes time to remove the placeholder image that's there now.



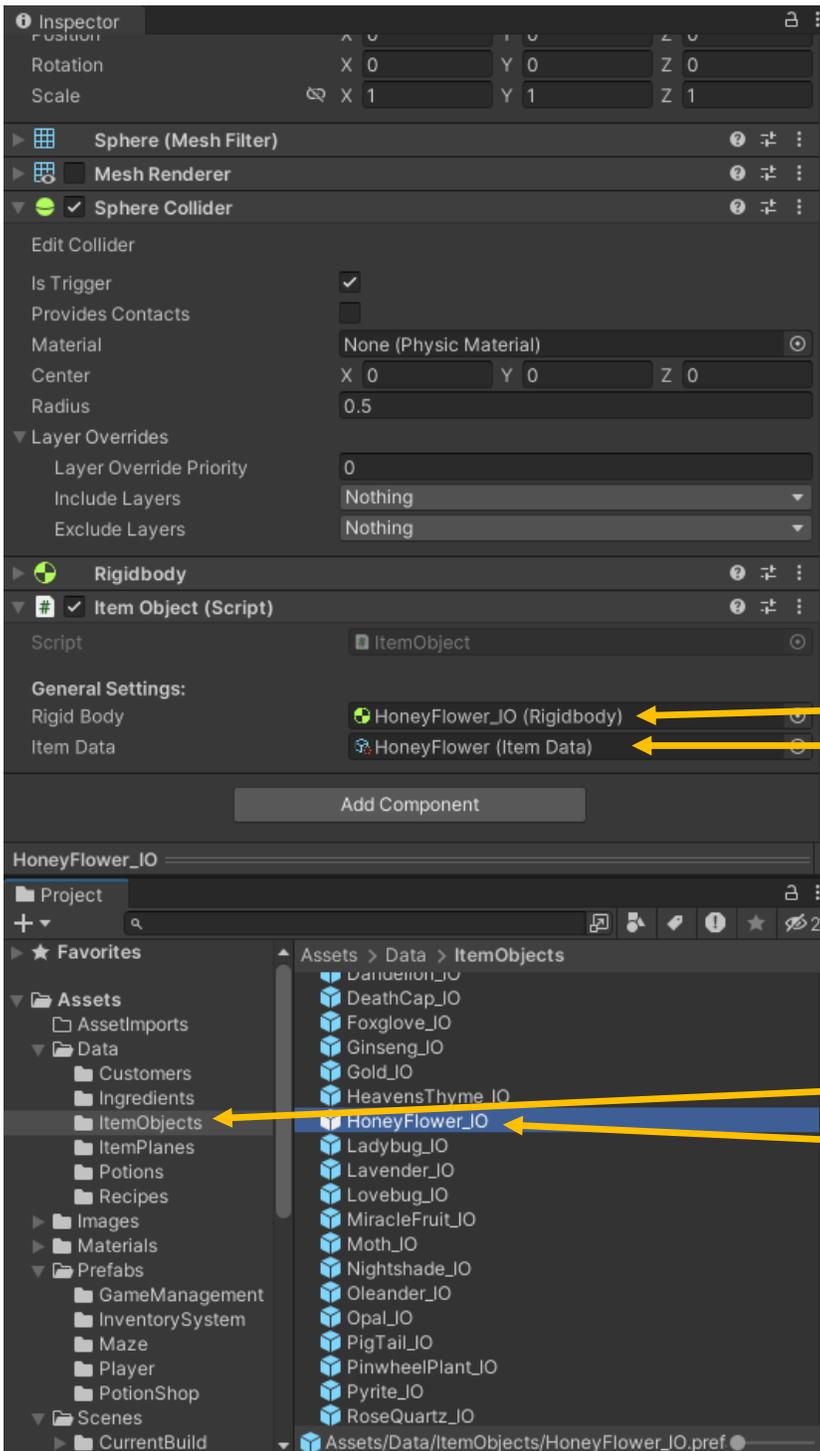
# Item Data:

The item data object represents the data for an individual item in the potion shop game. For ingredients, the checkbox for ingredient should be marked true and the sellable checkbox should be false. The width and height of the item in the inventory can also be set and modified here. The item data game object also contains the image for the 2D icon as well as the 3D model for it.



# Item Object:

The item object represents the 3D instance of the item in the game world. The item object is used to indicate an ingredient item in the 3D maze levels. It's also meant to be collided with, as to be picked up by the player.



Set rigid body for item object

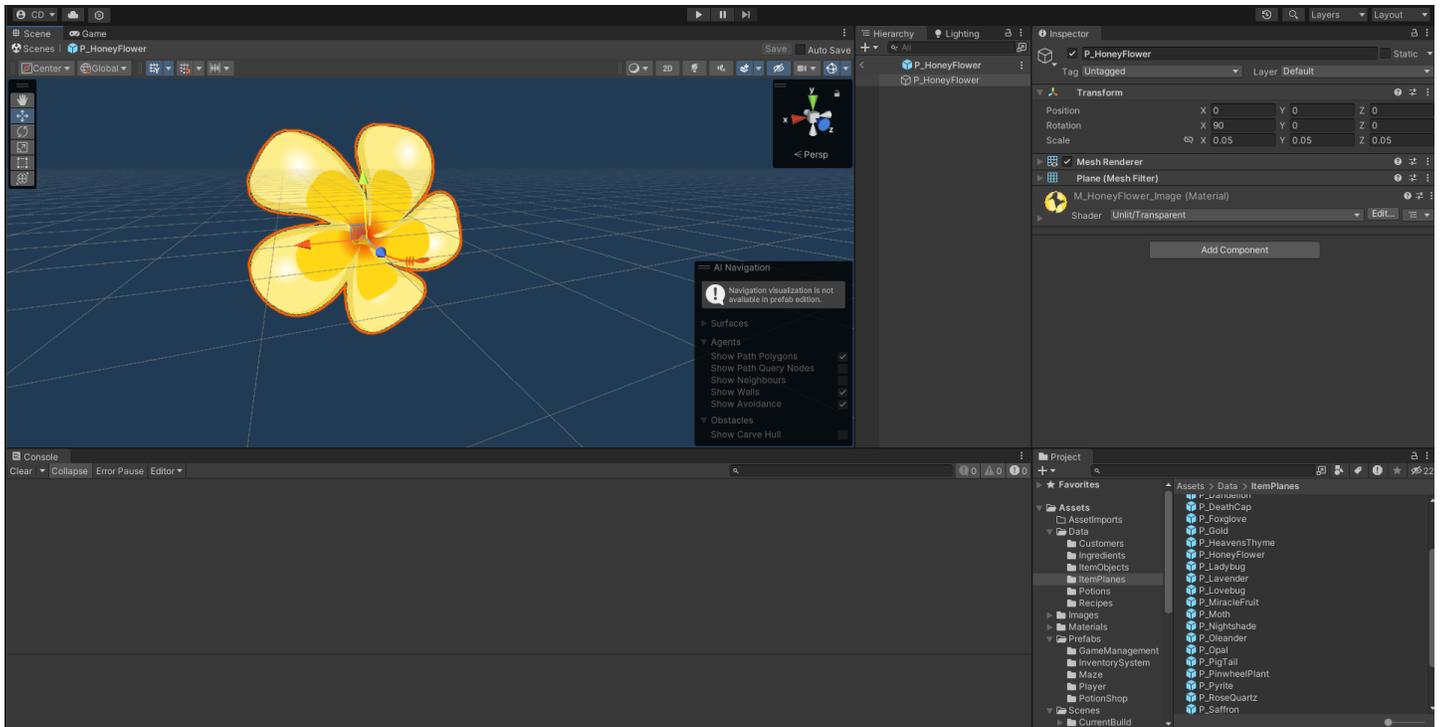
Set corresponding item data

Directory where item objects are found

Item objects which are the 3D models associated with each item (item data)

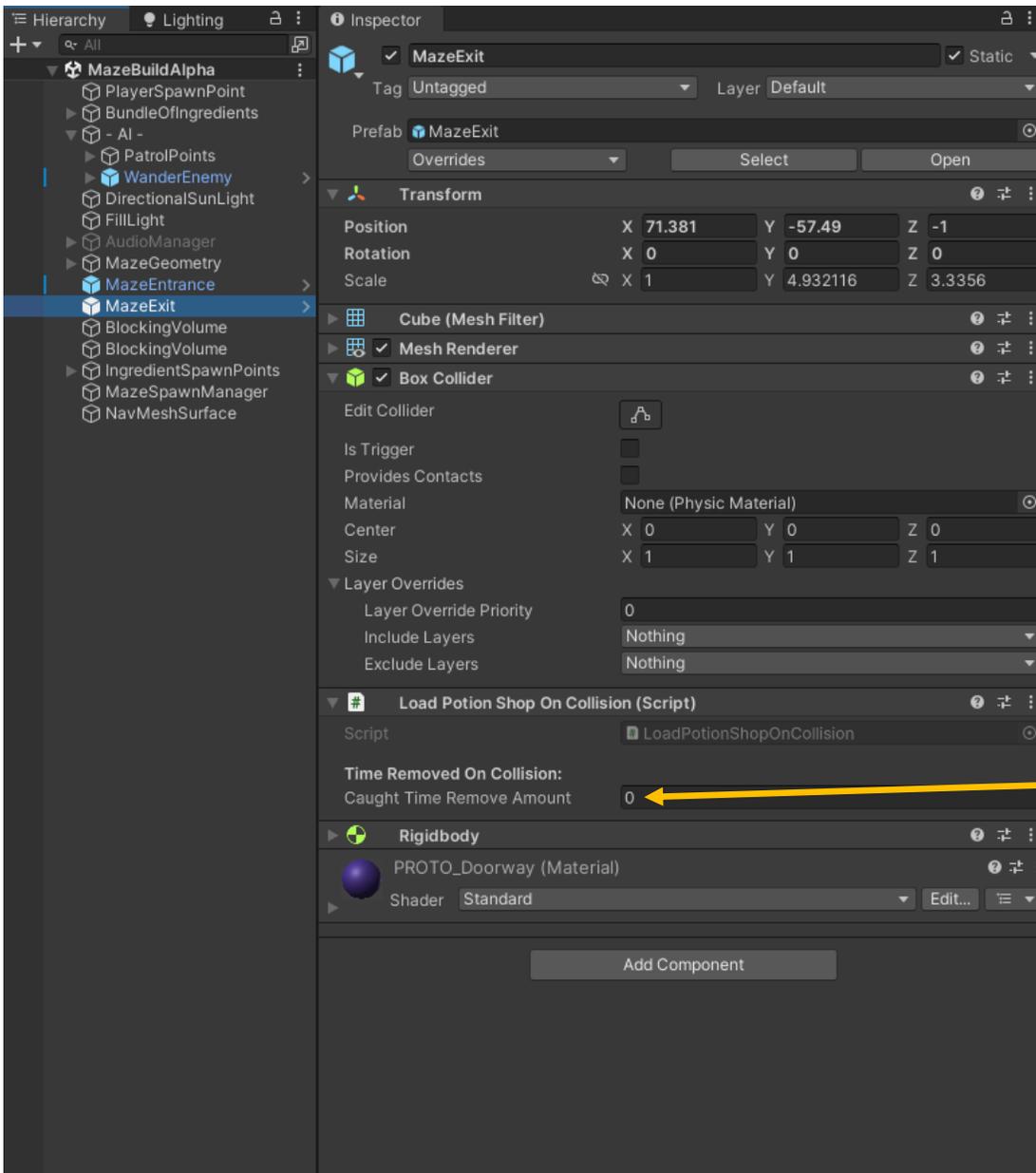
# Item Plane:

The item planes are the three-dimensional 2D planes with the images of the ingredients on them. The planes are scripted to always face the player, like it is in some retro, 3D video games.



# Load Potion Shop On Collision:

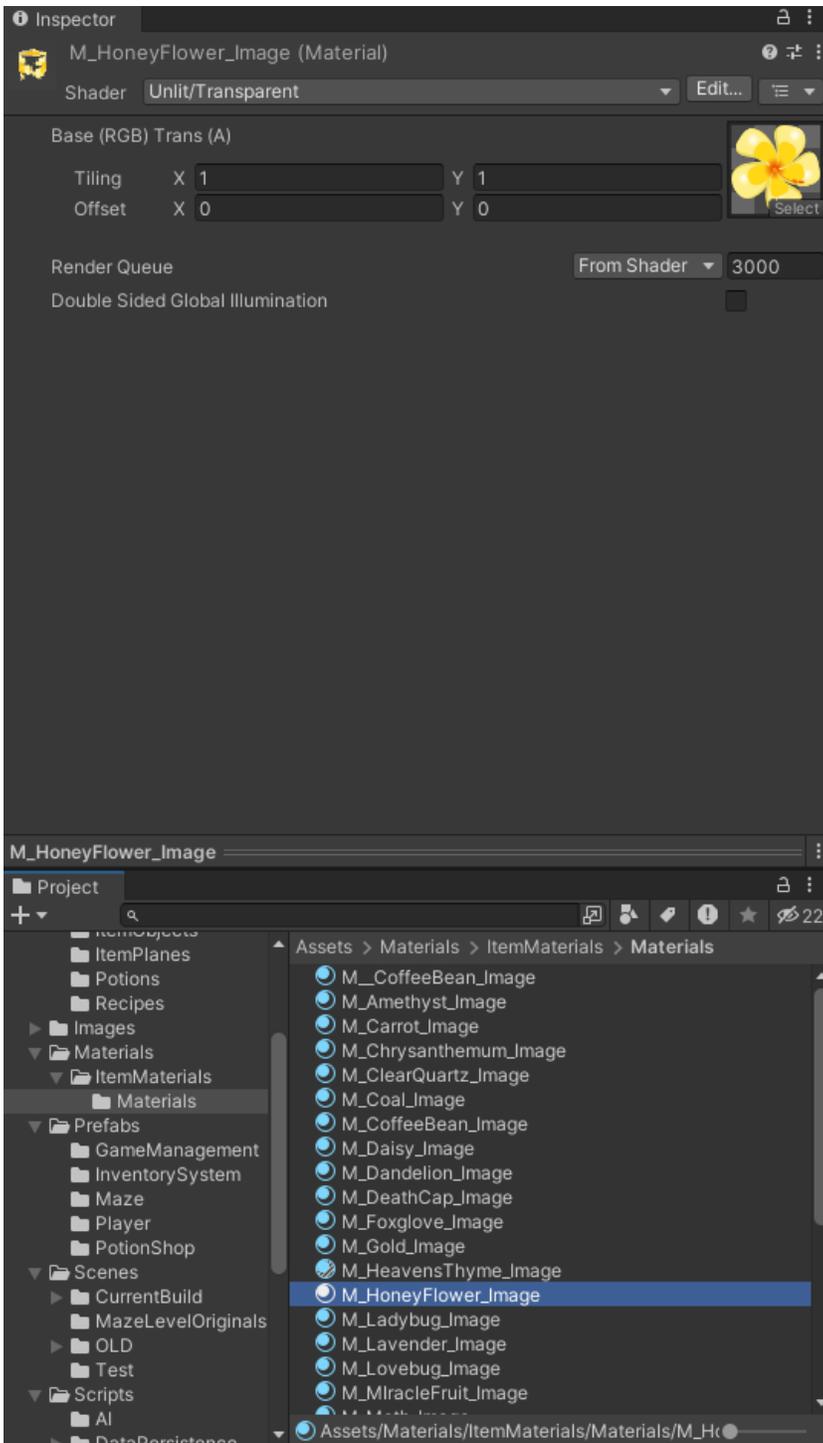
The function of this class is simply to return the player to the potion shop upon collision with it. This class is used both for the maze entrances and exits, as well as the wander AI that catches the player upon collision with them. If desired, time can also be removed from the day when this happens, such as if the player suffered a penalty from getting caught in the maze.



Set amount of time removed from day if collides with player

# Materials Use:

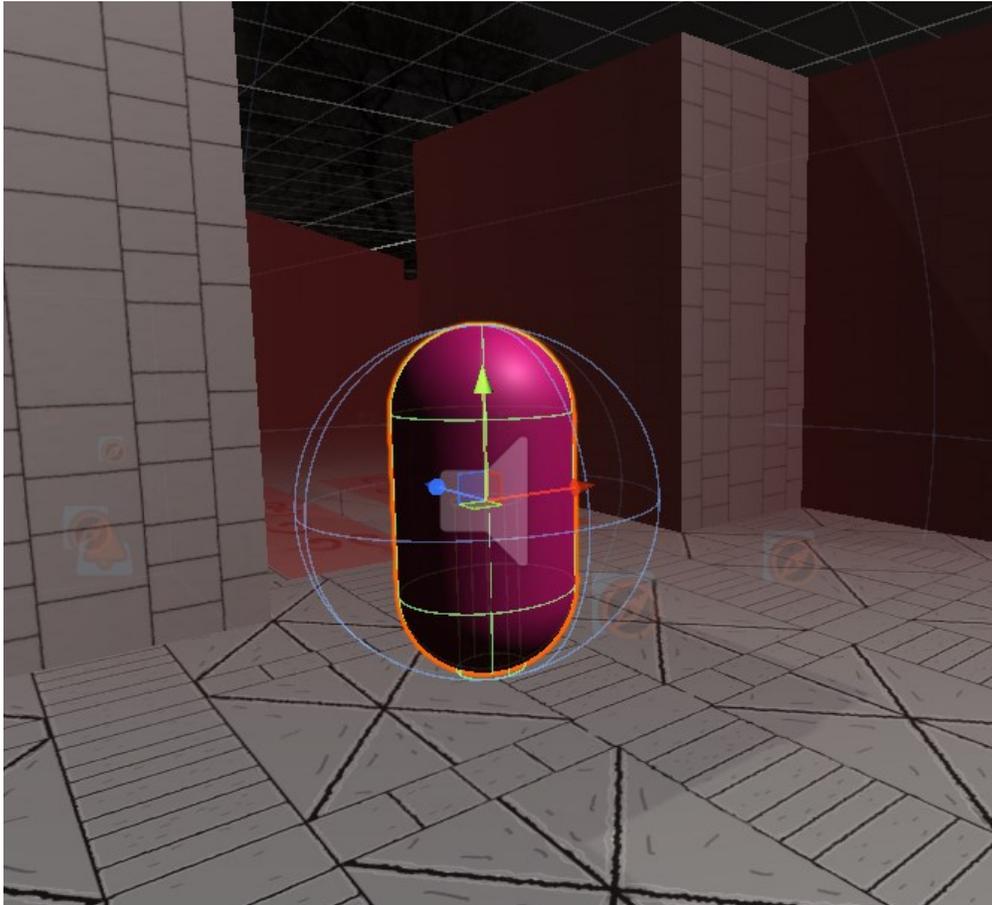
The directory where the materials used in the game can be found. These materials include prototype material and the images for displaying the item objects, made so that they can be wrapped onto a plane.



## Maze AI Controller (and Wander Enemy):

The maze AI controller is the logic for the enemy found there. The AI wanders around on a nav mesh, patrolling a set number of points in random order. If a patrol point can't be reached within a certain amount of time, then it will timeout and attempt to move to a new patrol point. This is done so that the enemy doesn't get stuck due to an instance of bad pathfinding. The maze AI controller is where the settings for the maze enemy can be altered, including patrol speed, chase speed, detection radius and distance, and wait times. (\*\*view inspector window on next page\*\*)

The wandering AI is merely this AI controller with a *load potion shop on collision* component attached to it. The player will be transported back to the potion shop with a time penalty if this collision occurs.



**Inspector**  
#  Maze AI Controller (Script)

Script: MazeAIController

**Movement Settings:**  
Nav Mesh Agent: None (Nav Mesh Agent)  
Walk Speed: 5 → Set AI walk speed  
Sprint Speed: 10 → Set AI sprint speed  
Wait Time: 3 → Set wait time between actions

**Detection Settings:**  
Detection Time: 6 → Set time before ESP detect  
View Radius: 30 → Set detection distance  
View Angle: 90 → Set detection cone of view  
Player Mask: Player  
Obstacle Mask: Obstacles

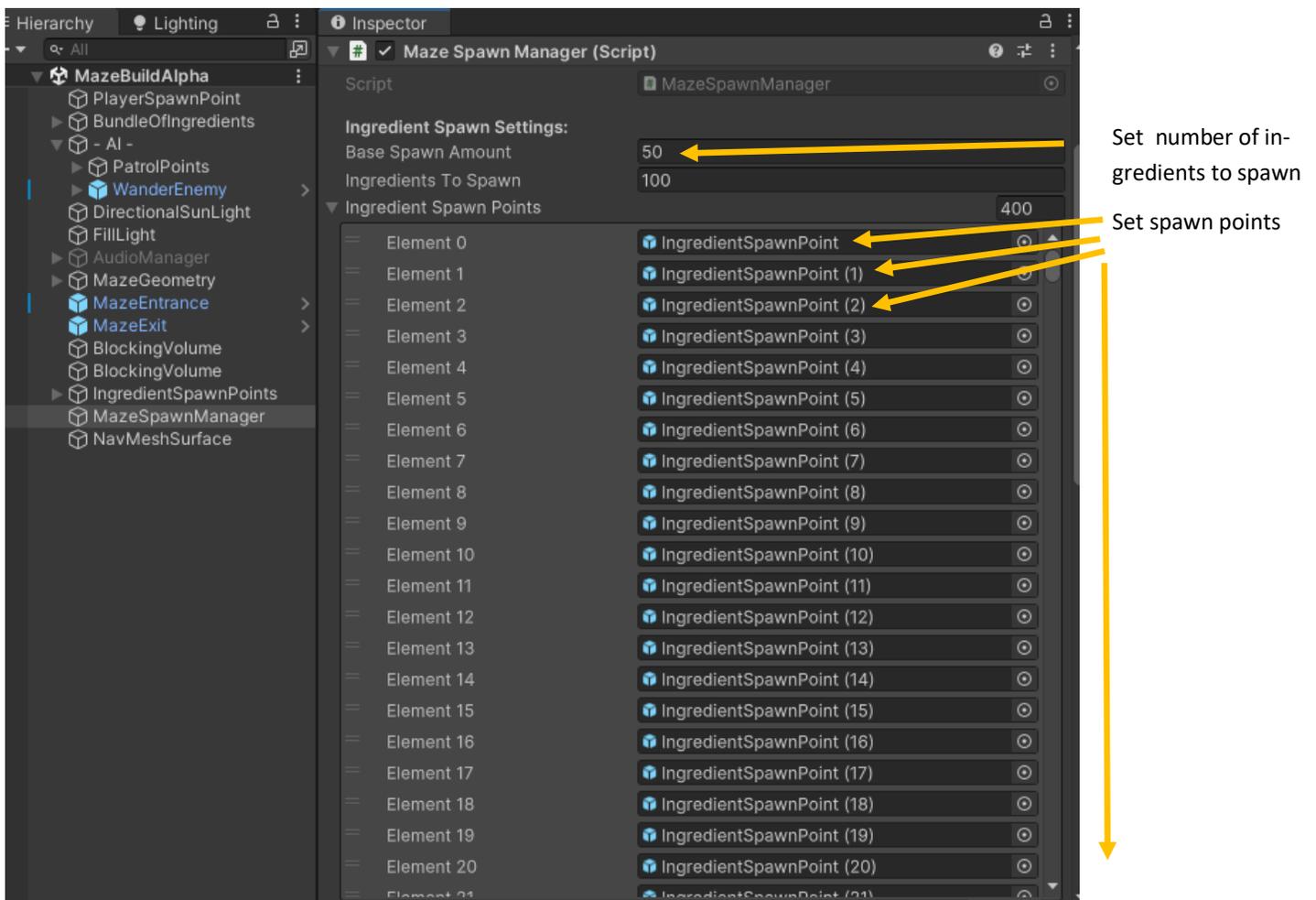
**Patrol Settings:**  
Wait Timeout: 5 → Set time until find new point

**Patrol Points** 46 → Set number of patrol points  
Set individual patrol points

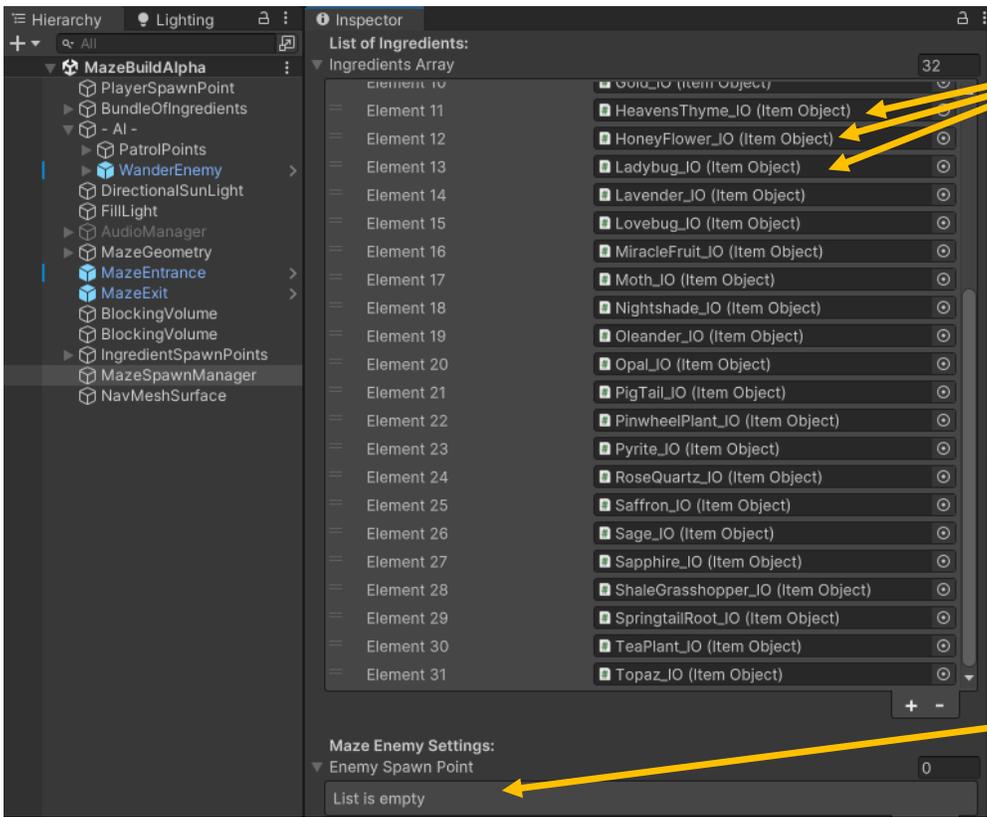
- Element 0: PatrolPoint0 (Transform)
- Element 1: PatrolPoint1 (Transform)
- Element 2: PatrolPoint2 (Transform)
- Element 3: PatrolPoint3 (Transform)
- Element 4: PatrolPoint4 (Transform)
- Element 5: PatrolPoint5 (Transform)
- Element 6: PatrolPoint6 (Transform)
- Element 7: PatrolPoint7 (Transform)
- Element 8: PatrolPoint8 (Transform)
- Element 9: PatrolPoint9 (Transform)
- Element 10: PatrolPoint10 (Transform)
- Element 11: PatrolPoint11 (Transform)
- Element 12: PatrolPoint12 (Transform)
- Element 13: PatrolPoint13 (Transform)
- Element 14: PatrolPoint14 (Transform)
- Element 15: PatrolPoint15 (Transform)
- Element 16: PatrolPoint16 (Transform)
- Element 17: PatrolPoint17 (Transform)
- Element 18: PatrolPoint18 (Transform)
- Element 19: PatrolPoint19 (Transform)
- Element 20: PatrolPoint20 (Transform)
- Element 21: PatrolPoint21 (Transform)

# Maze Spawn Manager:

The maze spawn manager is used to set how many ingredients spawn into the maze without the morality modifier (base spawn amount). The *ingredients to spawn* variable is merely the amount of items that will end up spawning during runtime when the morality modifier is applied. The maze spawn manager is also used to assign possible ingredient spawn points in each maze. The way maze ingredient spawning probability works at this time is that higher amounts/concentrations of ingredient spawn locations are placed in areas that are higher risk/danger. The ingredients that will spawn randomly in the maze at runtime must also be added to a list on the maze spawn manager. (\*\*see next page\*\*)



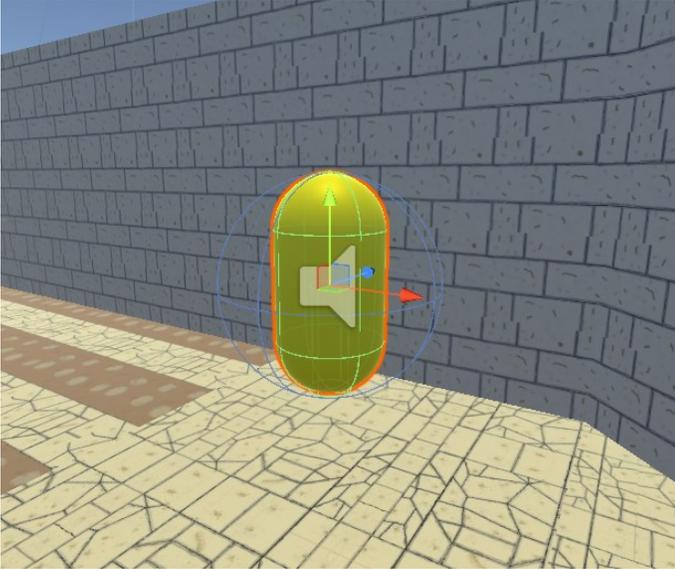
# Maze Spawn Manager (cont...):



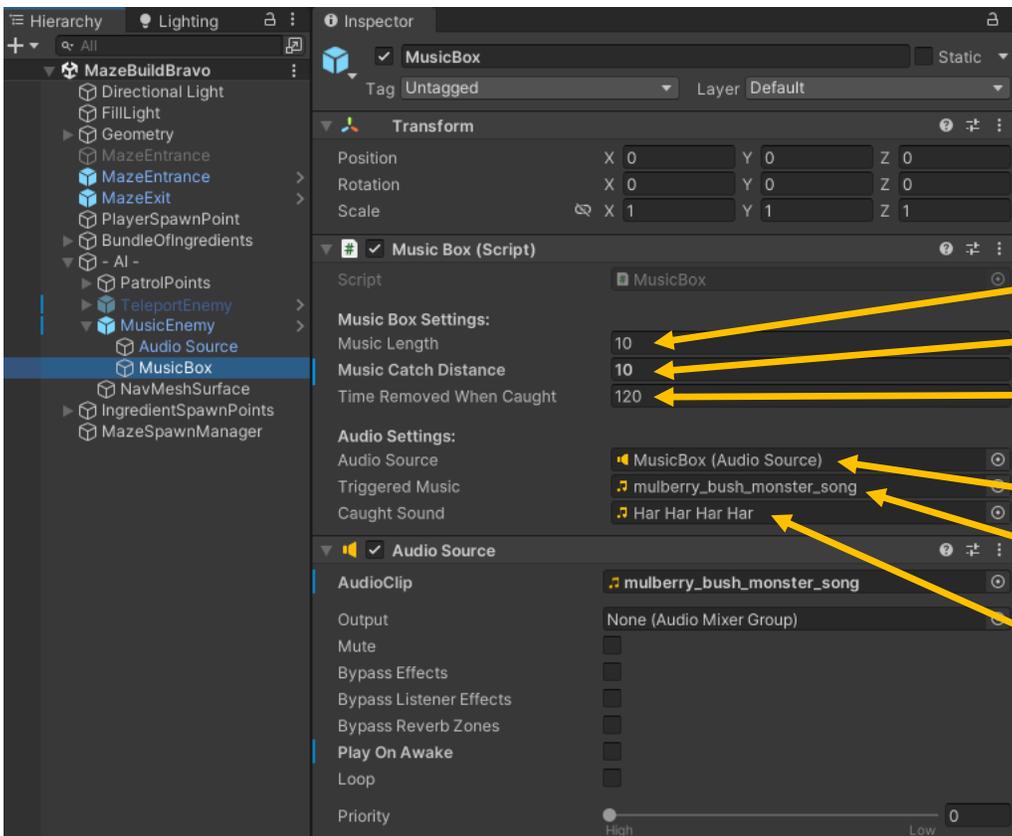
Set items to choose from to spawn

Set enemy spawn points (not yet implemented)

# Music Box (and Music Enemy):



The music box is a component that gets attached to a basic, wander AI, which functionally turns it into the music enemy. The music enemy wanders around the maze looking for the player, and when they spot them, they begin chasing them. A song also starts playing upon the player being detected, and if the player is within a certain radius from the music enemy when the song stops, the player is teleported back to the potion shop with a time penalty. The music box contains the settings for the music enemy's song length, catch distance, and more.



Music play length

Radius for catch distance

Amount of time removed if player is caught

Audio source attached to AI

Music to play when alerted by detecting player

Sound to play if player is caught

# Order System:

The order system is what randomly generates, keeps track of, and performs data operations for orders in the game. The order list is the order board that is displayed in the potion shop, where the player can view and fulfill customer's orders. There is also an area to add new Customer game objects and Potion types/icons to the order system, which will then be used for random order generation in the game.

**Order and Customer Lists:**

Element	Order
Element 0	Order1 (Order)
Element 1	Order2 (Order)
Element 2	Order3 (Order)
Element 3	Order4 (Order)
Element 4	Order5 (Order)
Element 5	Order6 (Order)
Element 6	Order7 (Order)
Element 7	Order8 (Order)

**Customer List:** 11

Element	Customer
Element 0	Adora_SO (Customer)
Element 1	Aoi_SO (Customer)
Element 2	Cullen_SO (Customer)
Element 3	DoctorCoron_SO (Customer)
Element 4	Gill_SO (Customer)
Element 5	LadyBina_SO (Customer)
Element 6	MonkDvir_SO (Customer)
Element 7	Portia_SO (Customer)
Element 8	RandomCustomer1_SO (Customer)
Element 9	RandomCustomer2_SO (Customer)
Element 10	Valerie_SO (Customer)

**Potion Icons:**

Icon	Potion
Antidote Icon	Antidote_Potion
Benefit Icon	Benefit_Potion
Crippling Icon	Crippling_Potion
Death Icon	Death_Potion
Hatred Icon	Hatred_Potion
Health Icon	Health_Potion
Love Icon	Love_Potion
Lucky Icon	Luck_Potion
Poison Icon	Poison_Potion

**Text Settings:**  
Potion Type Text:

**Annotations:**

- List of current orders which are displayed in the potion shop (randomized each day)
- List of the available customers that can be chosen at random to order a potion
- List of the available potion icons/types that can be chosen at random for an order

# Potion Crafting System:

The potion crafting system contains the logic for making ingredients into potions. The way that it functions is that the player adds between 1-4 ingredients to the cauldron, and then if they have a valid recipe that contains exactly those ingredients, they can then press brew to begin crafting a potion. The settings for the potion crafting system can be greatly customized, including display color, potion quality ranges, and brew time (how long a potion needs to be brewed in seconds before it is complete).

**Brew Time Settings:**  
Cook Time: 60

**Display Color Settings:**  
Freezing Temp Display Color: [Blue]  
Low Temp Display Color: [Blue]  
Medium Temp Display Color: [Green]  
Hot Temp Display Color: [Orange]  
Boiling Temp Display Color: [Red]

**Quality Settings:**  
Ultra Quality Time Percentage: 0.9  
High Quality Time Percentage: 0.7  
Medium Quality Time Percentage: 0.5  
Low Quality Time Percentage: 0.3

**Potion Quality Color Settings:**  
Ultra Quality Color: [Pink]  
High Quality Color: [Blue]  
Medium Quality Color: [Green]  
Low Quality Color: [Yellow]  
Failed Color: [Red]

**Display Image Settings:**  
Temperature Display Image: TemperatureDisplayBackground (Image)  
Empty Slot Image: UIMask

**Button Settings:**  
Ingredient 1 Button: Ingredient1Button (Button)  
Ingredient 2 Button: Ingredient2Button (Button)  
Ingredient 3 Button: Ingredient3Button (Button)  
Ingredient 4 Button: Ingredient4Button (Button)  
Potion Retrieval Button: PotionRetrieveButton (Button)  
Increase Temp Button: IncreaseTempButton (Button)  
Decrease Temp Button: DecreaseTempButton (Button)  
Put On Lid Button: PutLidDownButton (Button)  
Remove Lid Button: PutLidUpButton (Button)  
Stir Button: StirringSpoonButton (Button)  
Brew Button: BrewButton (Button)

**Text Settings:**  
Temperature Display Text: TemperatureDisplayText (Text Mesh Pro UGUI)  
Time Remaining Text: TimeRemainingText (Text Mesh Pro UGUI)

Annotations:  
- Set length of time that potions will take to brew (points to Cook Time)  
- Choose display colors for the temperature setting background (points to color swatches)  
- Percentages needed to determine the quality of potion when being brewed, based on lid placement, stirring, and temp (points to Quality Settings values)  
- Choose display colors for the potion quality background (points to color swatches)  
- Set temperature display image (points to Temperature Display Image)  
- Set empty slot sprite/mask (points to Empty Slot Image)  
- Set the Button game objects for the potion crafting system (points to Button Settings list)  
- Set the TMP Text game objects for the potion crafting system (points to Text Settings list)

# Potion Data:

The potion data game object is a child of the item data class, which gives it the same attributes as the item data class, including inventory size, 2D and 3D representations of the item, and whether it's an ingredient or sellable. Make sure not to set the potion as an ingredient and instead mark that it is sellable as true, because these will be used to complete orders. In addition to the attributes that came with the item data class, the potion data also requires one potion type to be chosen for it. (\*\*\*)see inspector window view on next page\*\*\*)



# Potion Data (cont...):

The image shows the Unity Inspector and Hierarchy panels. The Inspector panel is for the 'Remis Detriment\_PD (Potion Data)' object. The Hierarchy panel shows the 'Assets > Data > Potions' directory with a list of potion data objects, including 'RemisDetriment\_PD' which is highlighted.

**Inspector Panel:**

- Script: PotionData
- Item Info:
  - Is Ingredient:  (Do not set as ingredient for a potion)
  - Is Sellable:  (Set as sellable for a potion)
- 2D Item Width and Height:
  - Width: 2 (Set width of potion in inventory)
  - Height: 2 (Set height of potion in inventory)
- 2D Item Icon: Poison\_Potion (Set the 2D icon for this potion)
- 3D Item Object: None (Item Object) (Set the planar 3D representation of the potion)
- Potion Type: (Only Select One)
  - Is Health:
  - Is Love:
  - Is Hatred:
  - Is Antidote:
  - Is Death:
  - Is Lucky:
  - Is Poison:  (Select the type of potion (only one))
  - Is Benefit:
  - Is Crippling:

**Hierarchy Panel:**

- Assets > Data > Potions
  - HealthPotion\_PD
  - JestersLie\_PD
  - JyransEffect\_PD
  - LastBreath\_PD
  - LillithsCurse\_PD
  - LionsHeart\_PD
  - MaunsStep\_PD
  - NaturesBlessing\_PD
  - OwlsSight\_PD
  - PeakAchuEffect\_PD
  - RemisAid\_PD
  - RemisCure\_PD
  - RemisDetriment\_PD (highlighted)
  - RemisLoss\_PD
  - RemisMiracle\_PD
  - SketesStep\_PD
  - SolaceRest\_PD
  - TaliasAid\_PD
  - TavorasVisit\_PD

# Recipe:

The recipe game object is where data for each individual recipe in the Unity project is stored. This data includes the potion that is made from this recipe (potion data), the ingredients with which it needs to be made (min = 1, max = 4), and the specific brew settings for that recipe (temp, stirring, and lid settings).

The image shows the Unity Inspector and Hierarchy panels. The Inspector panel displays the configuration for a **Recipe** game object. The Hierarchy panel shows the project structure, with the **Assets > Data > Recipes** directory selected.

**Inspector Panel:**

- Potion:** RemisDetriment\_PD (Potion Data)
- Ingredients (Between 1-4):**
  - Ingredient 1: Sage (Item Data)
  - Ingredient 2: MiracleFruit (Item Data)
  - Ingredient 3: Saffron (Item Data)
  - Ingredient 4: None (Item Data)
- Brewing Settings:**
  - Desired Temp: 1
  - Need Stirring:
  - Needs Lid On:

**Hierarchy Panel:**

- Assets > Data > Recipes
  - JestersLie\_Recipe
  - JyransEffect\_Recipe
  - LastBreath\_Recipe
  - LilithsCurse\_Recipe
  - LionsHeart\_Recipe
  - MaunsStep\_Recipe
  - NaturesBlessing\_Recipe
  - OwlsSight\_Recipe
  - PeakAchuuEffect\_Recipe
  - RemisAid\_Recipe
  - RemisCure\_Recipe
  - RemisDetriment\_Recipe
  - RemisLoss\_Recipe
  - RemisMiracle\_Recipe
  - SketesStep\_Recipe
  - SolaceRest\_Recipe
  - TaliasAid\_Recipe
  - TavorasVisit\_Recipe
  - ThanasKiss\_Recipe

Set the potion (potion data game object) that is created with this recipe

Set the ingredients (at least one and up to four)

Set the desired temperature for this recipe  
1=freezing, 2=low, 3=medium, 4=hot, 5=boiling

Set whether the recipe requires stirring halfway through brewing

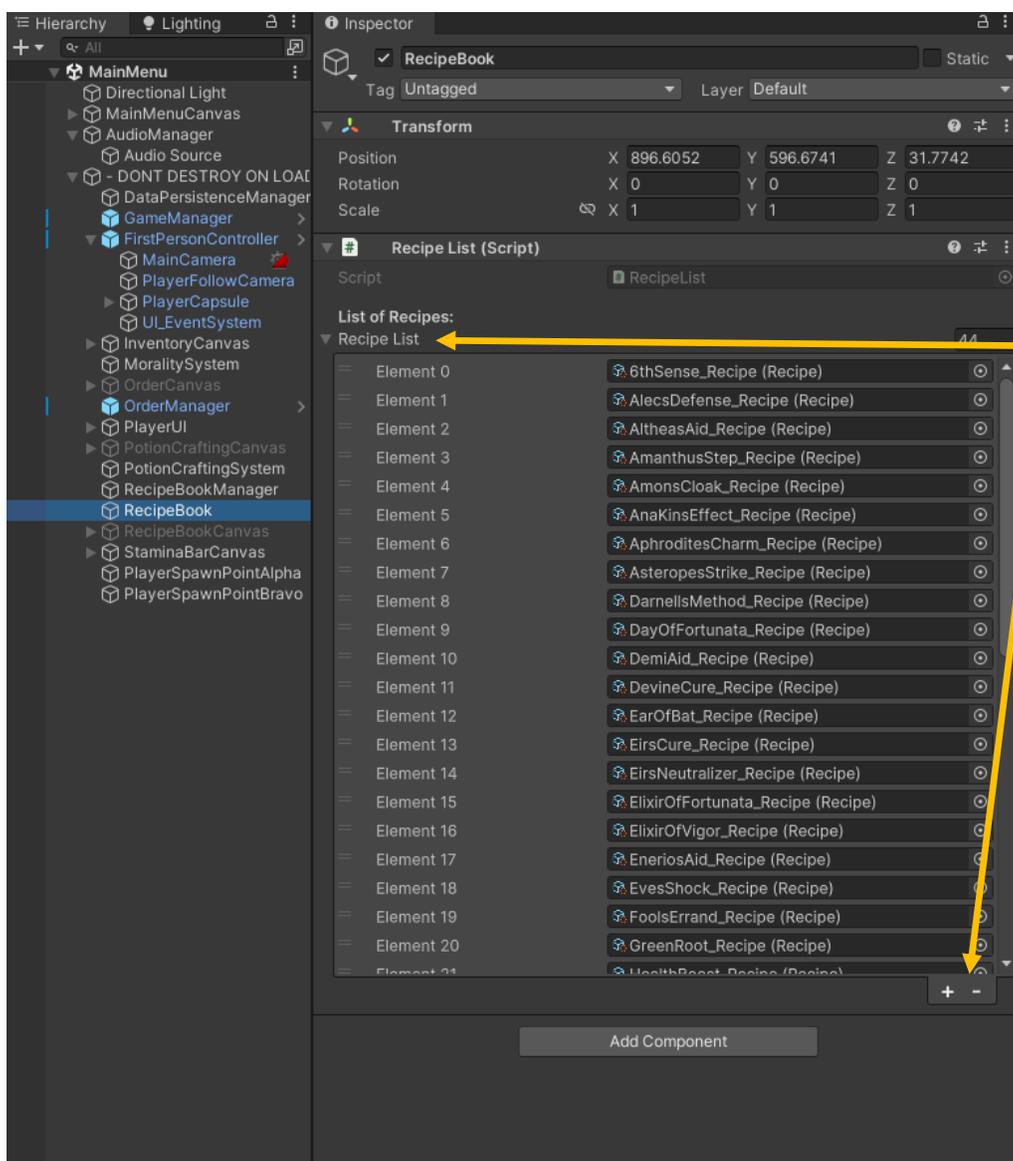
Set whether the recipe desires the lid to be on the cauldron

Directory where the recipes (recipe game objects) are located

The list of recipes in the Unity project as found in the directory

# Recipe Book:

The recipe book is the game object that holds all of the different recipes (combinations of ingredients) that can be made in the game. New recipes are easily made by creating new Recipe scriptable objects, which also requires creation of a new Potion (potion data) scriptable object that the recipe is used to produce.



Set the list of recipes that can be made in the game

# Recipe Book Manager:

The recipe book manager contains the settings for holding and displaying the recipe book pages. The recipe book/manual is displayed as an icon in the player's inventory, and its settings can be configured here as needed. The individual pages for the recipe book will be assigned to this game object, as will the buttons that'll be used for navigating the recipe book. This class is designed with customization in mind, including the ability to include the hand-drawn recipes for the game, once they're created and available.

The screenshot shows the Unity Inspector for the **RecipeBookManager** script. The settings are organized into several sections:

- Transform:** Position (X: 711.0211, Y: 464.8622, Z: -1.830809), Rotation (X: 0, Y: 0, Z: 0), Scale (X: 1, Y: 1, Z: 1).
- Recipe Book General Settings:**
  - Apothecary Manual Canvas: **RecipeBookCanvas (Canvas)** (Set the canvas for the recipe book UI)
  - Inventory Grid: **None (Game Object)** (Set the player's inventory grid)
- Page Settings:**
  - Book Page Index: **0** (Set the index of the current book page (debugging in editor))
- Book Pages:** A list of 14 elements with assigned objects:
  - Element 0: **PotionManual Title**
  - Element 1: **PotionShopOperationTitle**
  - Element 2: **PotionShopOperation1**
  - Element 3: **BrewingInstructionsTitle**
  - Element 4: **BrewingInstructions1**
  - Element 5: **PotionRecipesTitle**
  - Element 6: **PotionRecipes1**
  - Element 7: **PotionRecipes2**
  - Element 8: **PotionRecipes3**
  - Element 9: **PotionRecipes4**
  - Element 10: **IngredientsTitle**
  - Element 11: **Ingredients1**
  - Element 12: **Ingredients2**
  - Element 13: **LastPagesBlank**(Assign the pages in the recipe book in desired order)
- Button Settings:**
  - Page Forward Button: **PageForwardButton (Button)** (Set the go to next page Button)
  - Page Back Button: **PageBackButton (Button)** (Set the go to previous page Button)
  - Toggle Book Canvas Button: **PotionShopManualButton (Button)** (Set the toggle recipe book Button)

# Stamina System:

The stamina system is what keeps track of the player's energy level (ability to run). There are many variables that can be customized here, especially during our own playtesting, so that we can eventually figure out which settings provide the best player experience.

Set reference to the first person controller

Set the player's max stamina

Set the player's stamina cost when they jump

Set the rate at which stamina decreases when sprinting

Set the rate at which stamina replenishes when not sprinting

Set the run speed of the player when out of stamina

Set the player's normal run speed

Assign the image for the UI slider bar which displays stamina progress

Assign the canvas group that contains the stamina bar

# Teleportation:

The teleportation component is what will be used for the third and final maze enemy type: the teleporting enemy. The teleporting enemy, who is invisible, will wander and then *phase in* to the map upon seeing the player (simulating teleportation). The player must then look at the enemy and see them within a certain amount of time, or else the player gets teleported back to the potion shop with a time penalty applied to the day if they don't by the time the enemy *phases out*. If the player does manage to look at the teleporting enemy before it phases out, then the player will remain in the maze untouched and the teleportation enemy will return to wandering the maze invisibly, until once again seeing the player (after a cooldown so it's not immediately phasing back in).

The screenshot shows the Unity Inspector for a **Teleportation** component attached to a **TeleportEnemy** object. The settings are as follows:

- Teleportation/Phase Settings:**
  - Phase Length: 10
  - Phase Catch Distance: 40
  - Time Removed When Caught: 120
  - Maze AI Controller: TeleportEnemy (Maze AI Controller)
- Audio Settings:**
  - Audio Source: Audio Source
  - Triggered Music: haur haur haur haur
  - Caught Sound: Har Har Har Har

Yellow arrows point from the following text labels to the corresponding settings in the Inspector:

- Set teleport time length (points to Phase Length)
- Set radius for catch distance (points to Phase Catch Distance)
- Set amount of time removed if player is caught (points to Time Removed When Caught)
- Set the maze AI controller (points to Maze AI Controller)
- Set audio source attached to AI (points to Audio Source)
- Assign music to play when alerted by detecting player (points to Triggered Music)
- Assign sound to play if player is caught (points to Caught Sound)