POTION SHOP: GONE ASTRAY

UNITY PROJECT DOCUMENTATION

Last Update: 01.16.2024

Table of Contents

Introduction:

Hello Reader!

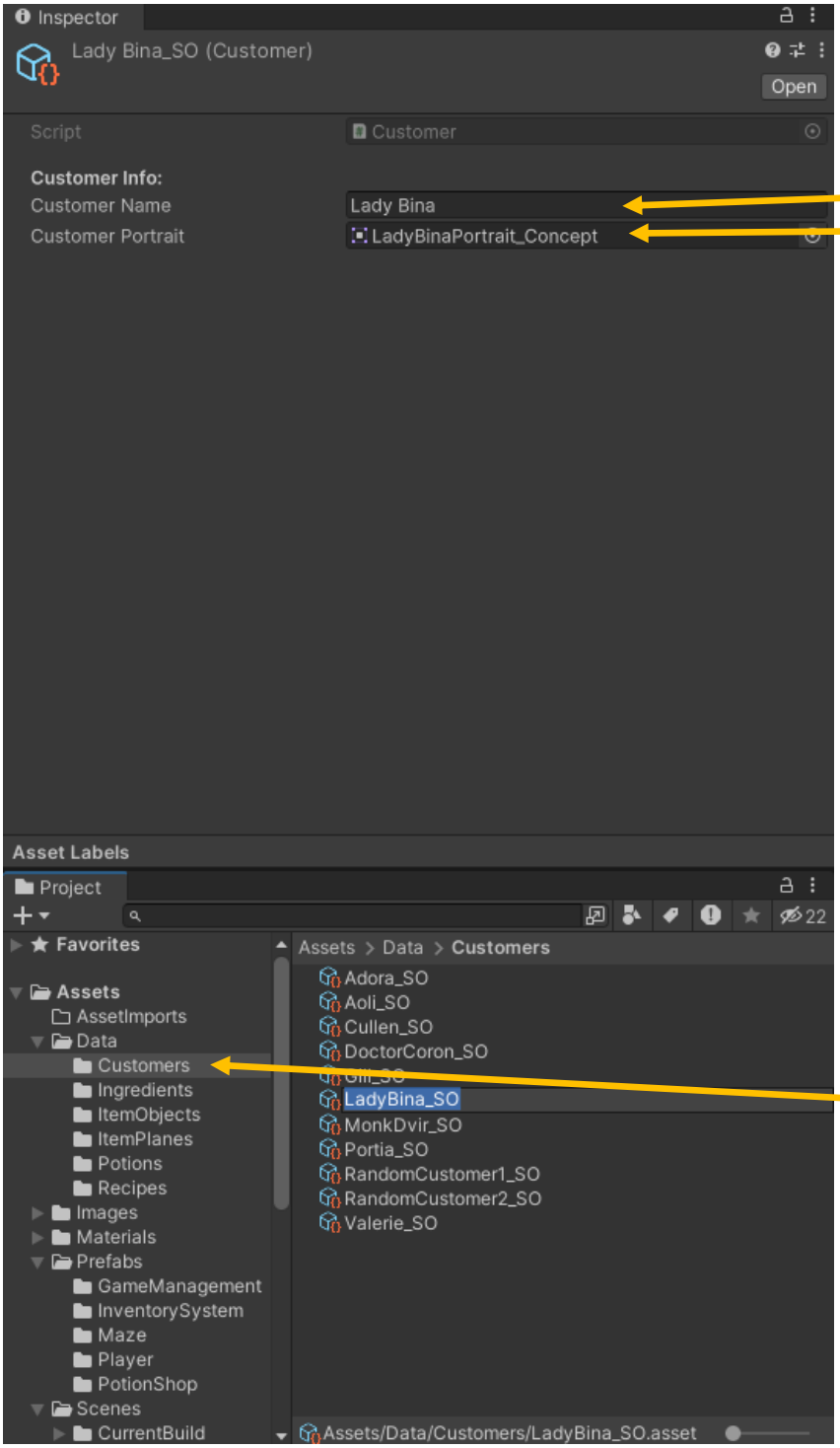
I'm excited that you've decided to learn more about this Unity project! I've created this document for the 2023-2024 UNM Gaming Capstone project titled "Potion Shop: Gone Astray", specifically as a guide to the Unity project itself. The game was designed with many placeholders to implement our finalized 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.



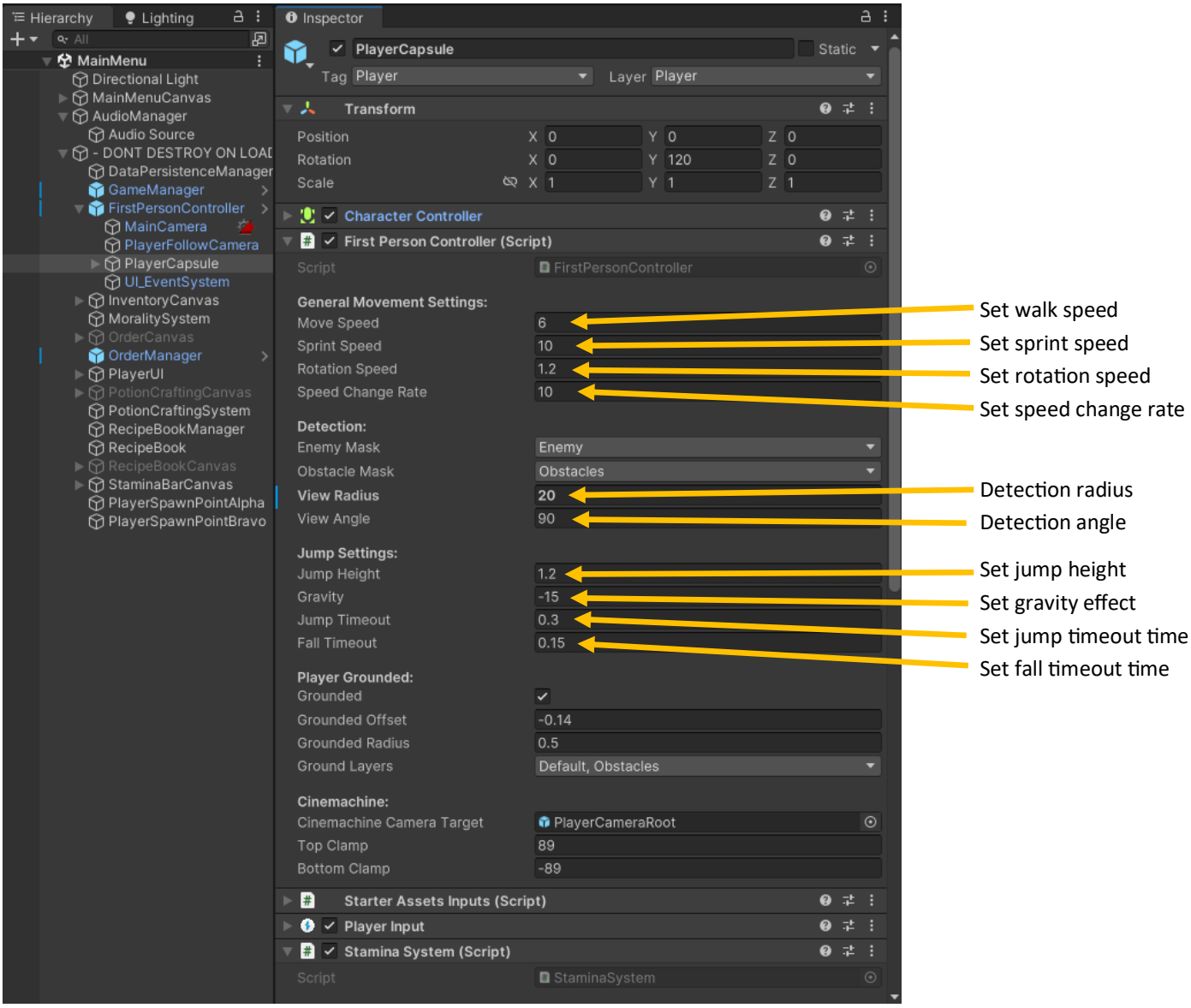
Set customer name

Set customer portrait

Directory where customer data is found

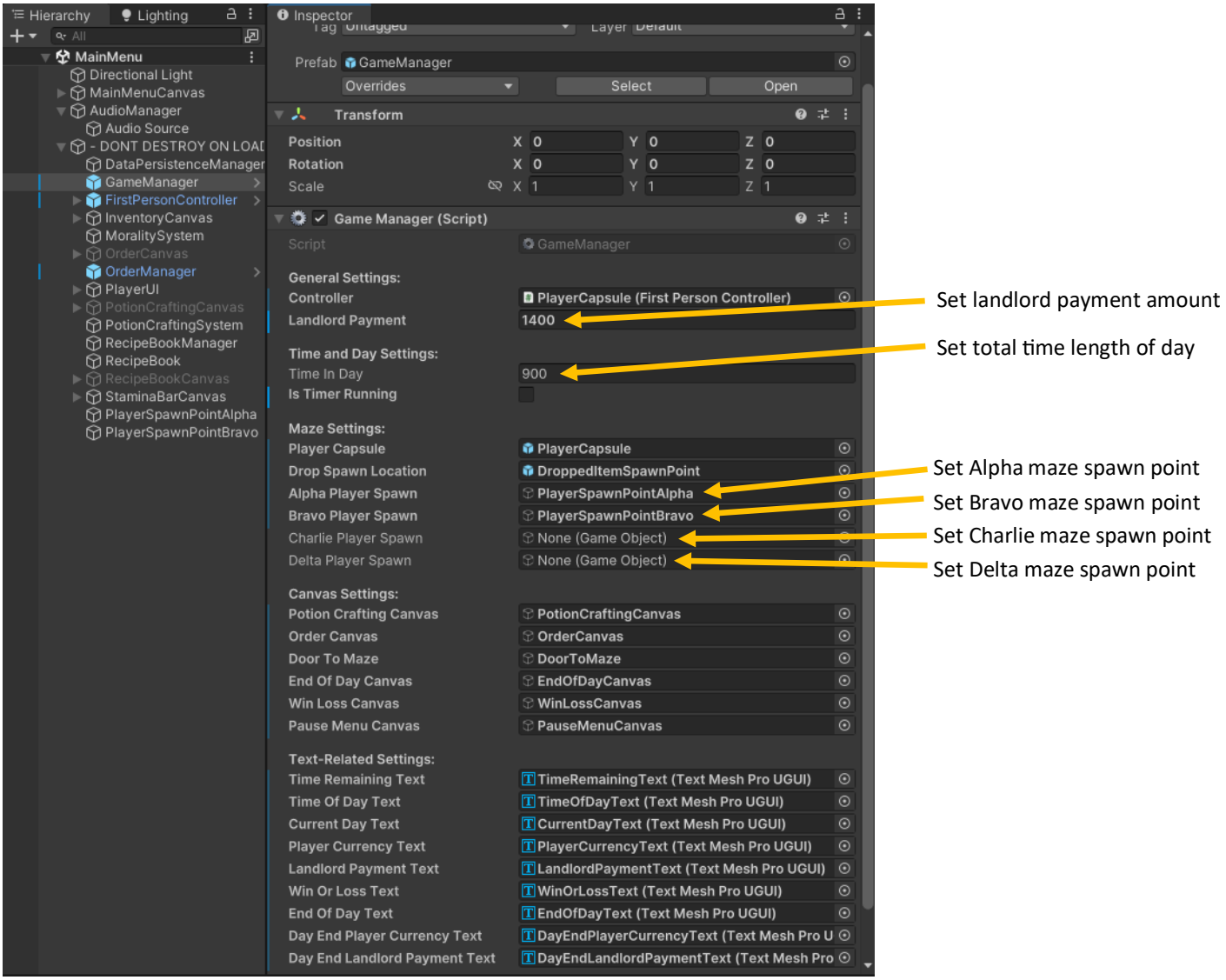
First Person Controller:

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).



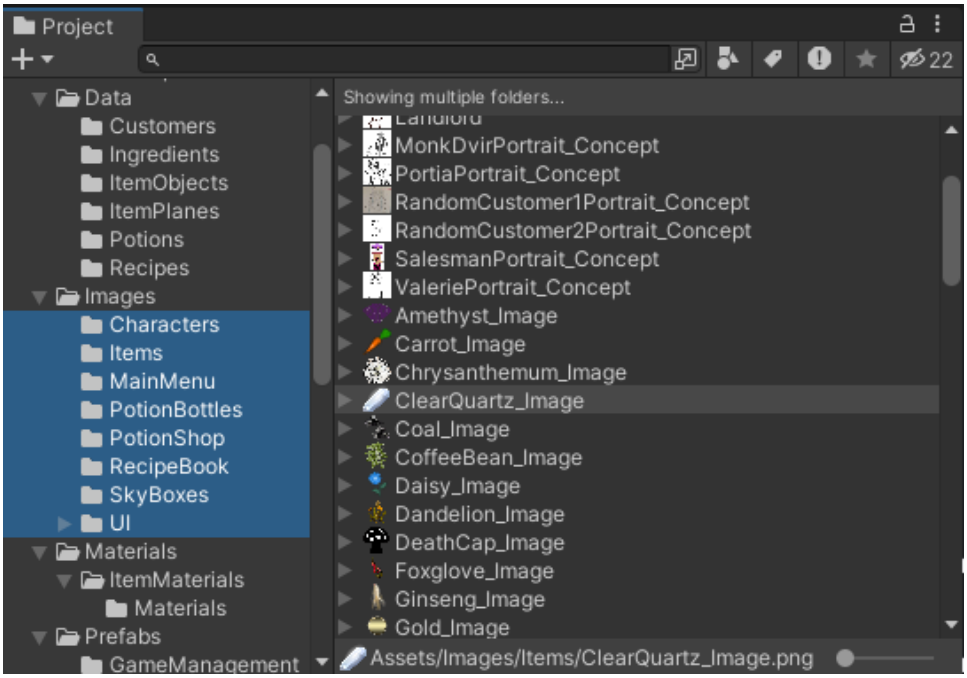
Game Manager:

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.



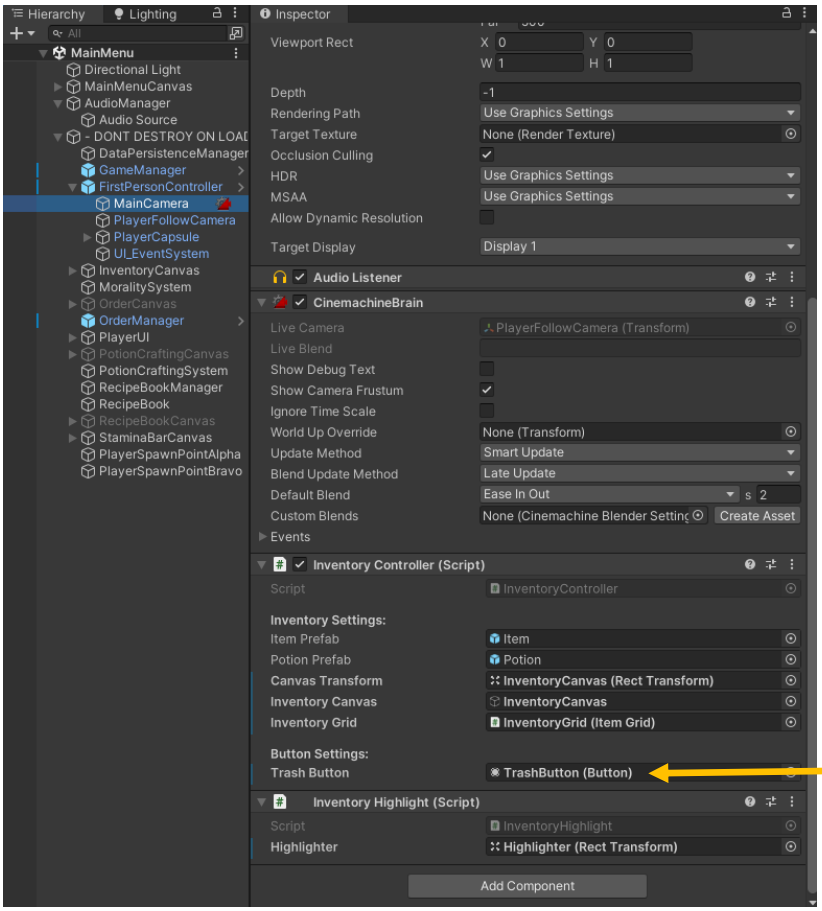
Importing Images:

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:

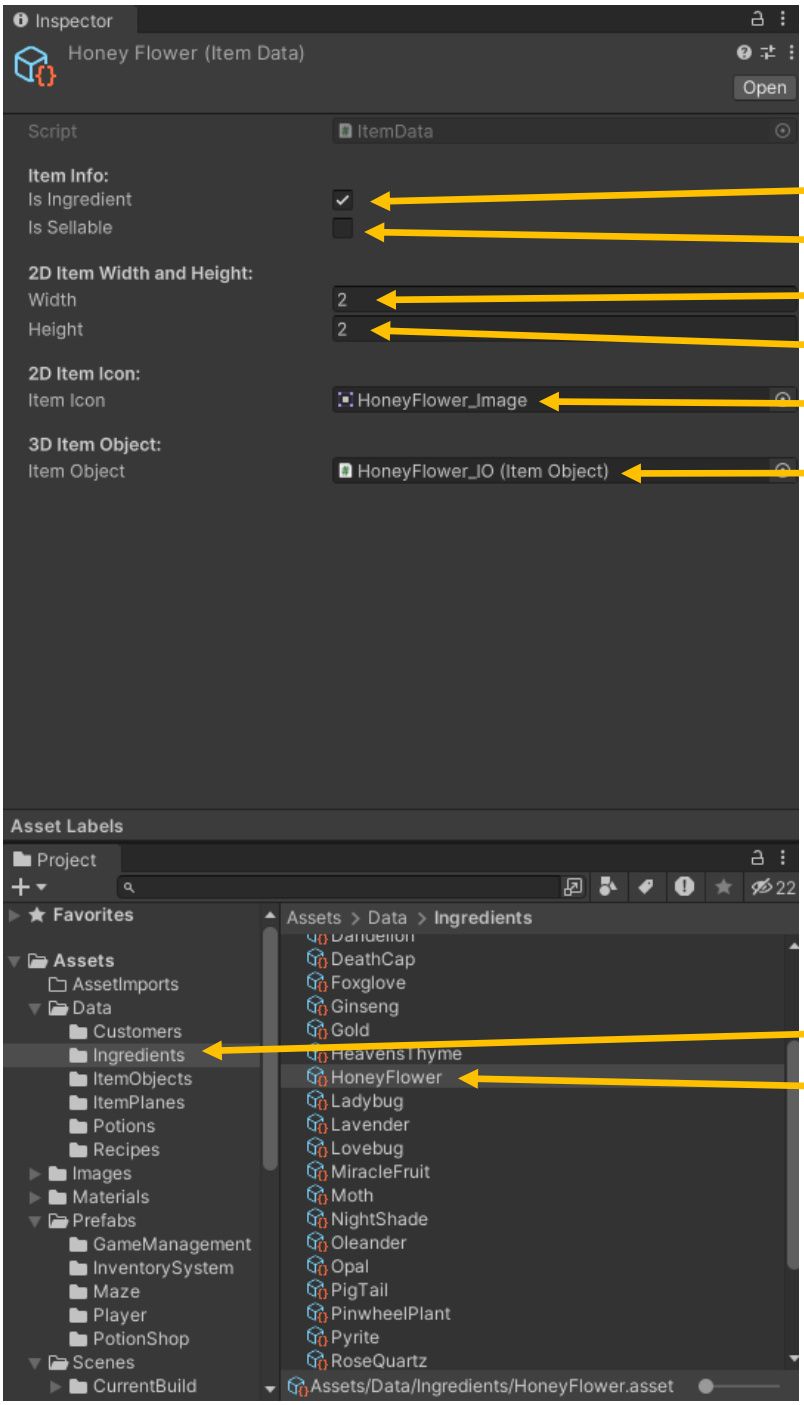
The inventory controller is where the logic for the inventory system is located. Most settings in here will not need setting or changing. The exception is that the trash button icon can be reset to a different image, when it comes time to remove the placeholder image that's there now.



Set trash button icon

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.



Set whether item is an ingredient for a potion recipe

Set whether item is sellable

Set inventory width of the item

Set inventory height of the item

Set sprite icon for item

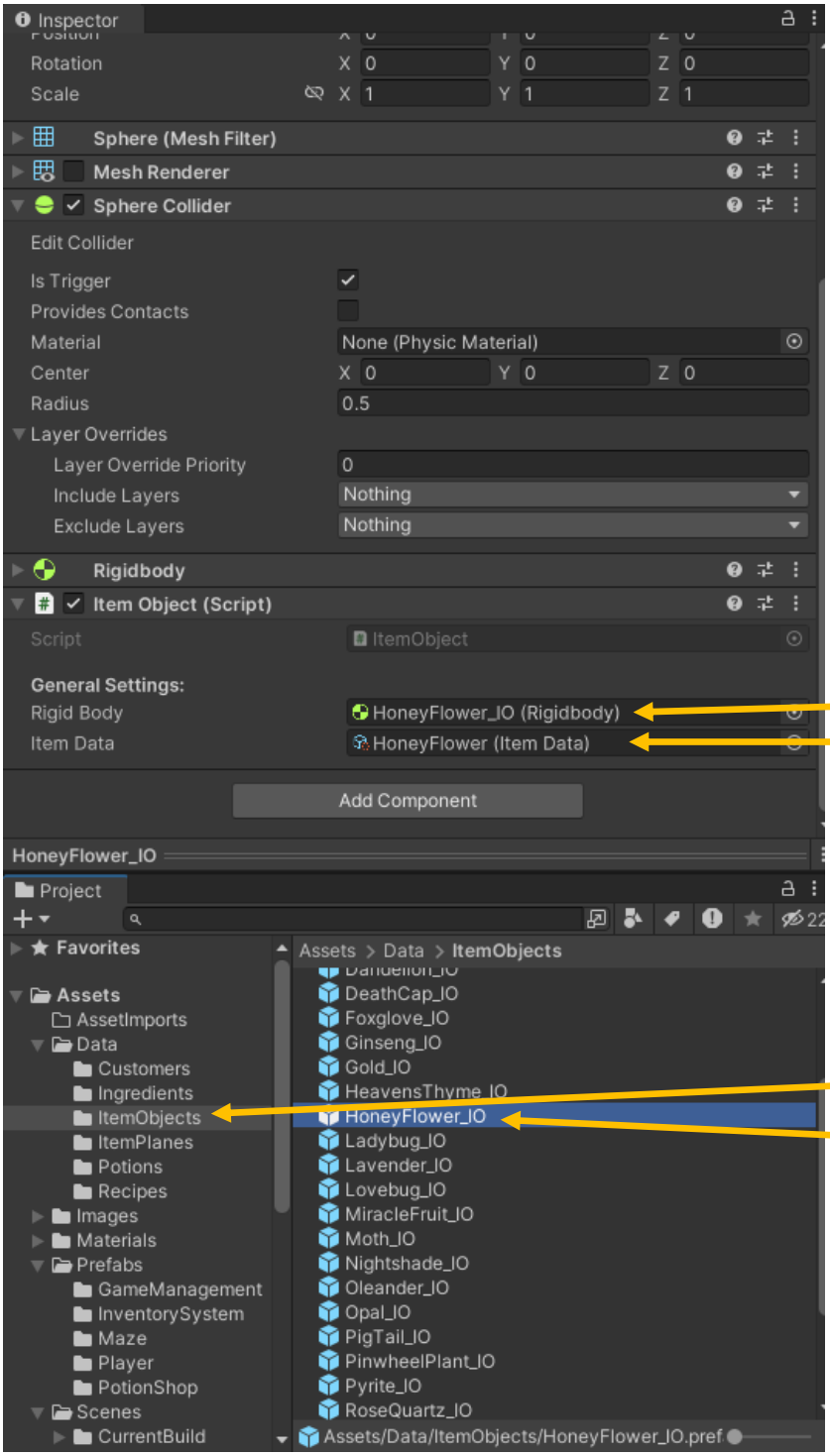
Set corresponding 3D item object for item

Directory where ingredients are found

Ingredients that can be configured

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

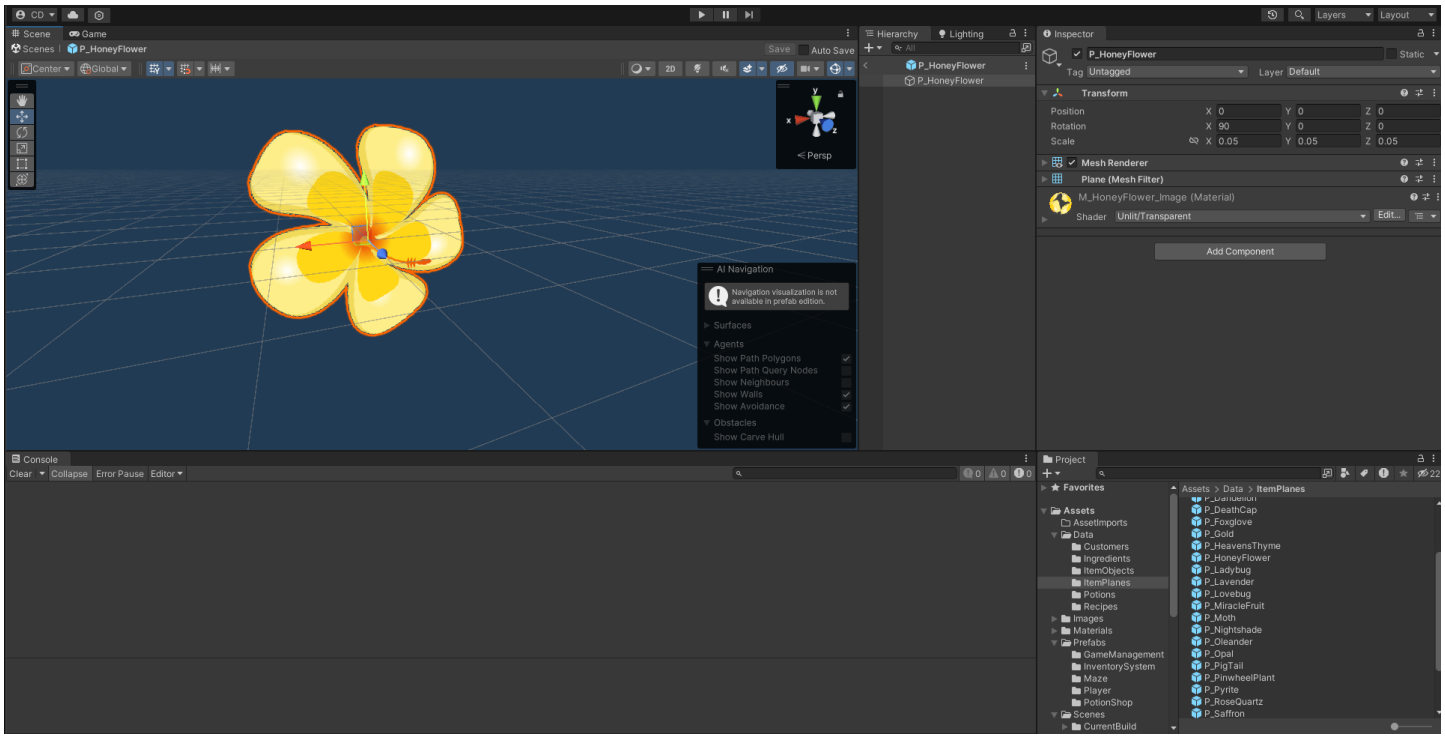
Set corresponding item data

Directory where item objects are found

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

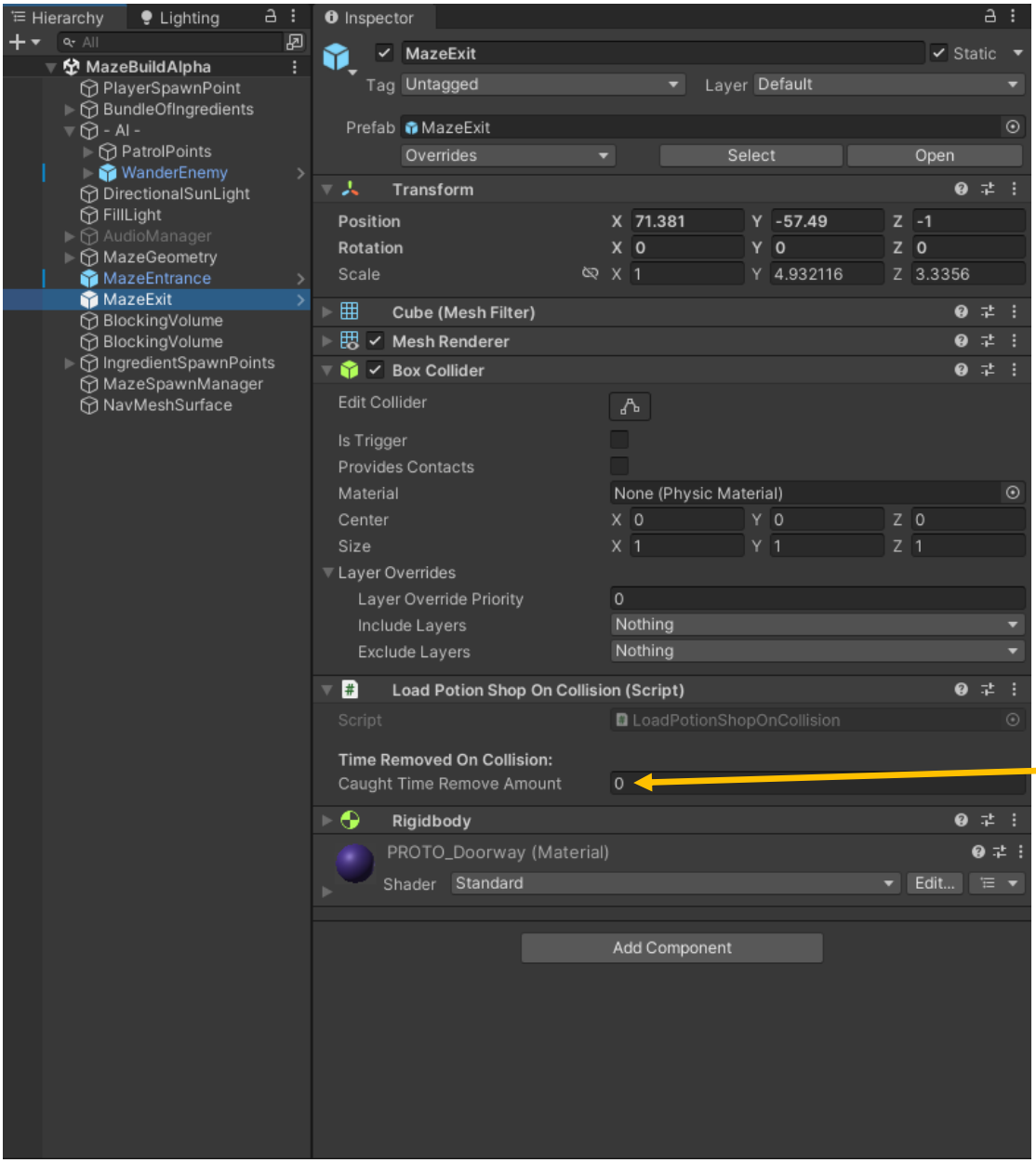
Item Plane:

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:

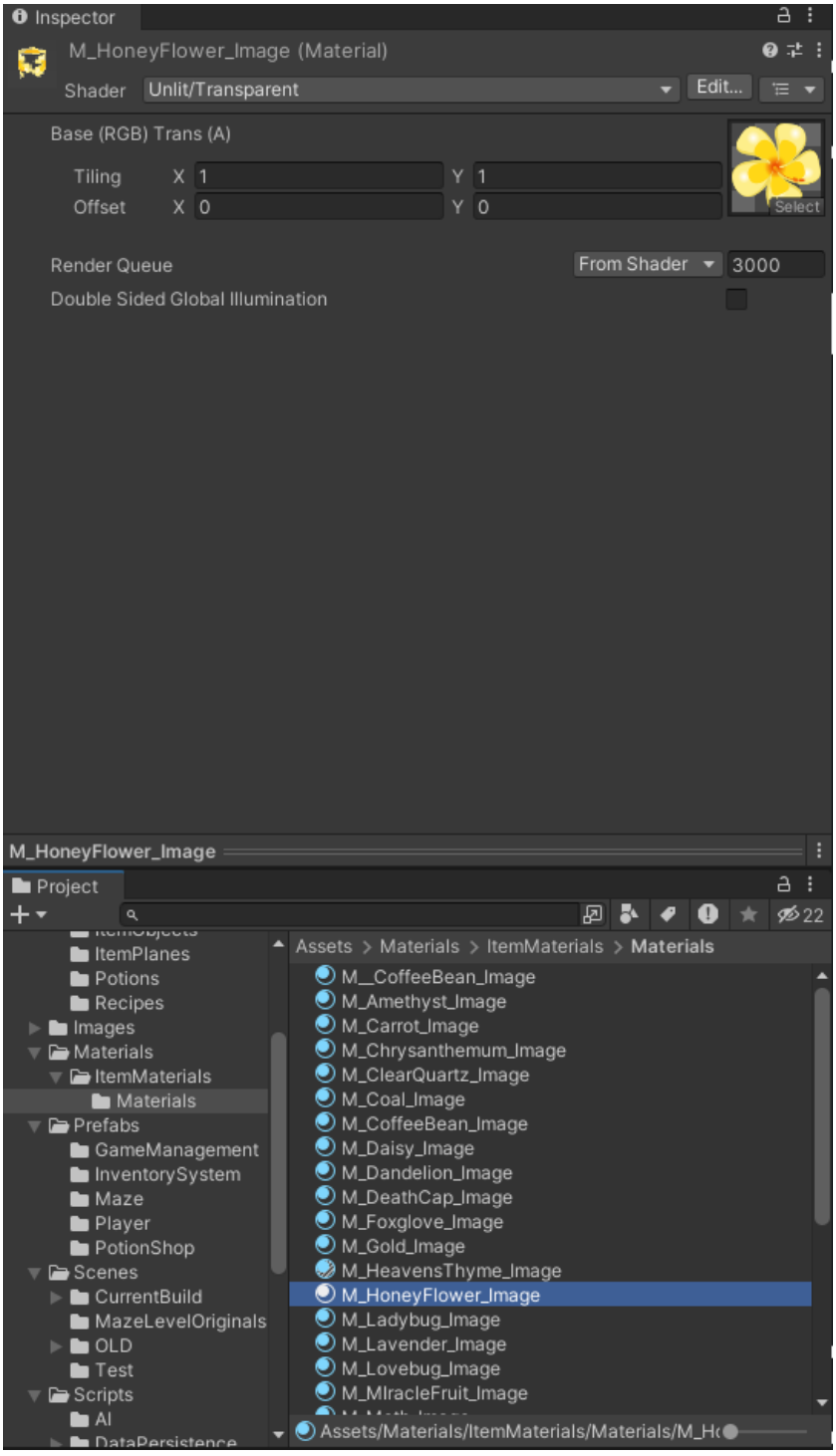
The function of this class is simply to return the player to the potion shop upon collision with it. This class is used both for the maze entrances and exits, as well as the wander AI that catches the player upon collision with them. If desired, time can also be removed from the day when this happens,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

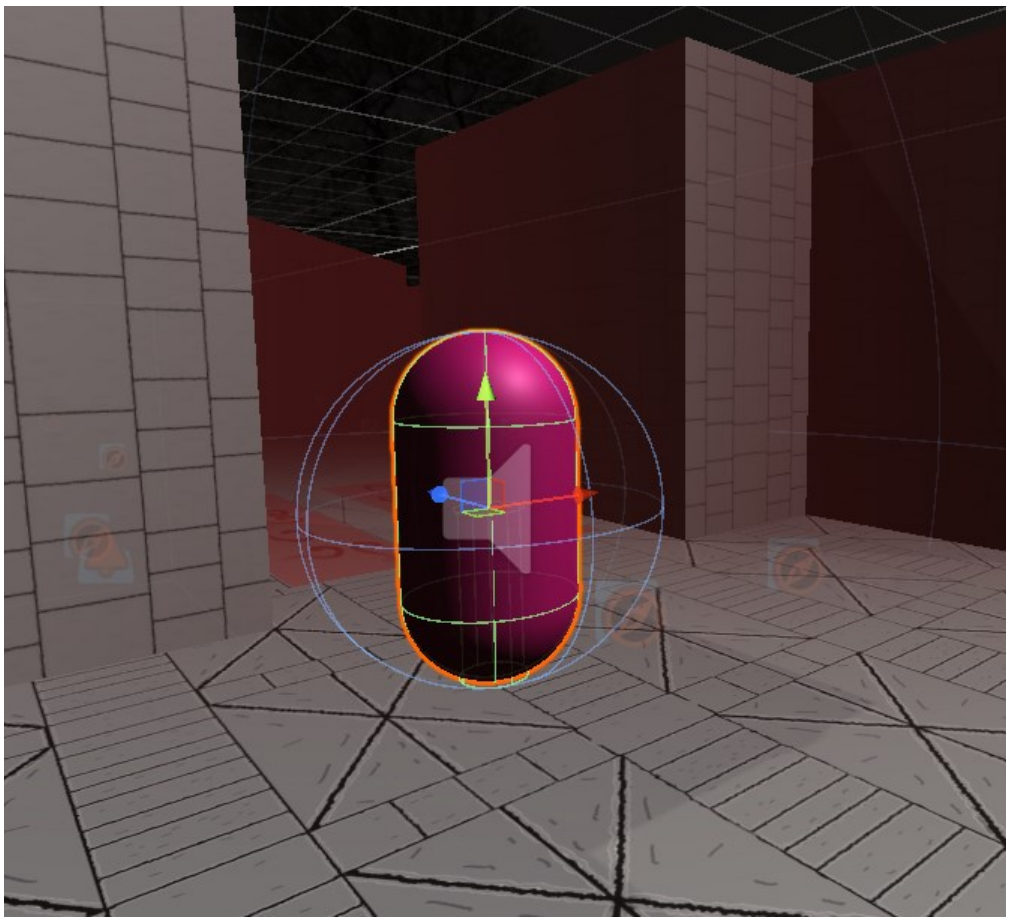
The directory where the materials used in the game can be found. These materials include prototype material and the images for displaying the item objects, made so that they can be wrapped onto a plane.

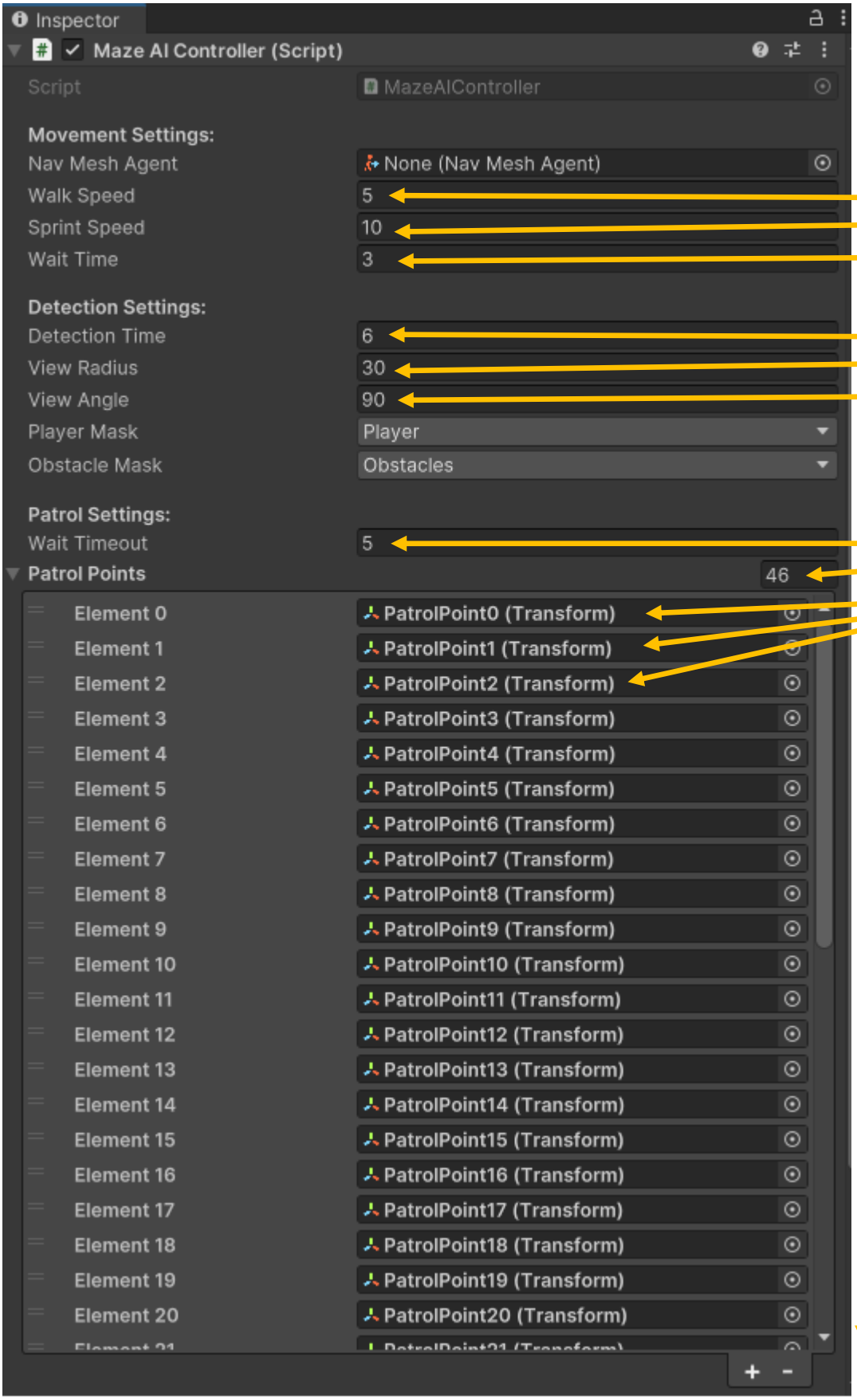


Maze AI Controller (and Wander Enemy):

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.





Set AI walk speed

Set AI sprint speed

Set wait time between actions

Set time before ESP detect

Set detection distance

Set detection cone of view

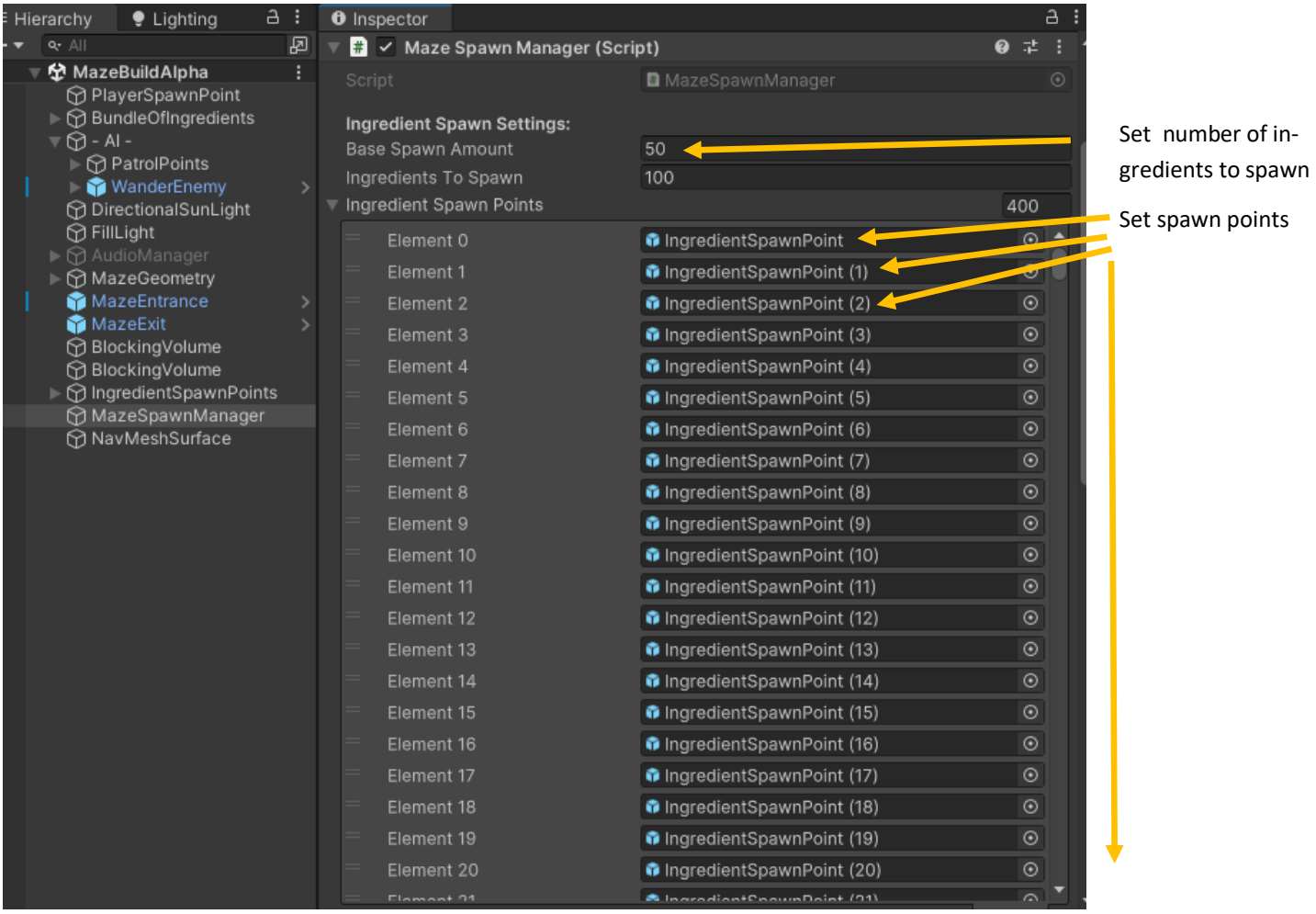
Set time until find new point

Set number of patrol points

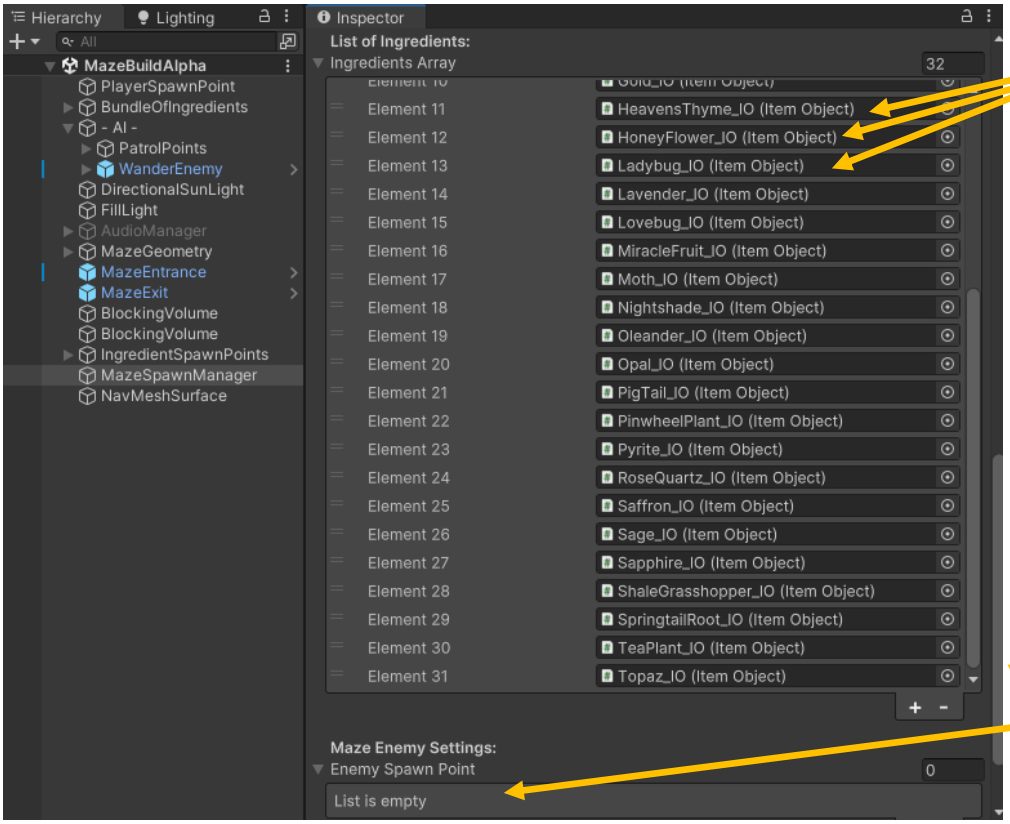
Set individual patrol points

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***)



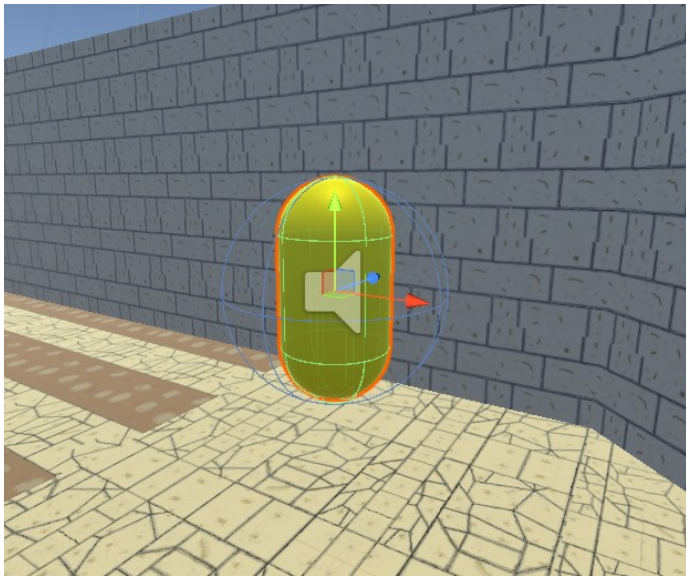
Maze Spawn Manager (cont…):



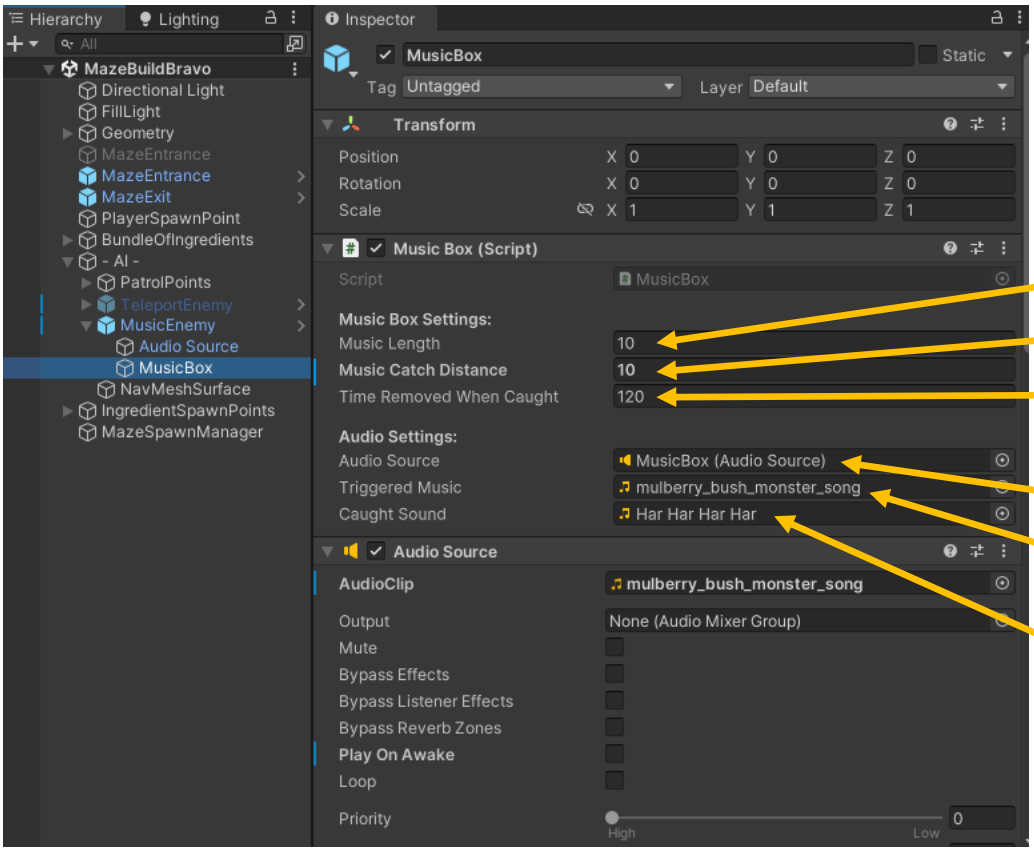
Set items to choose from to spawn

Set enemy spawn points (not yet implemented)

Music Box (and Music Enemy):

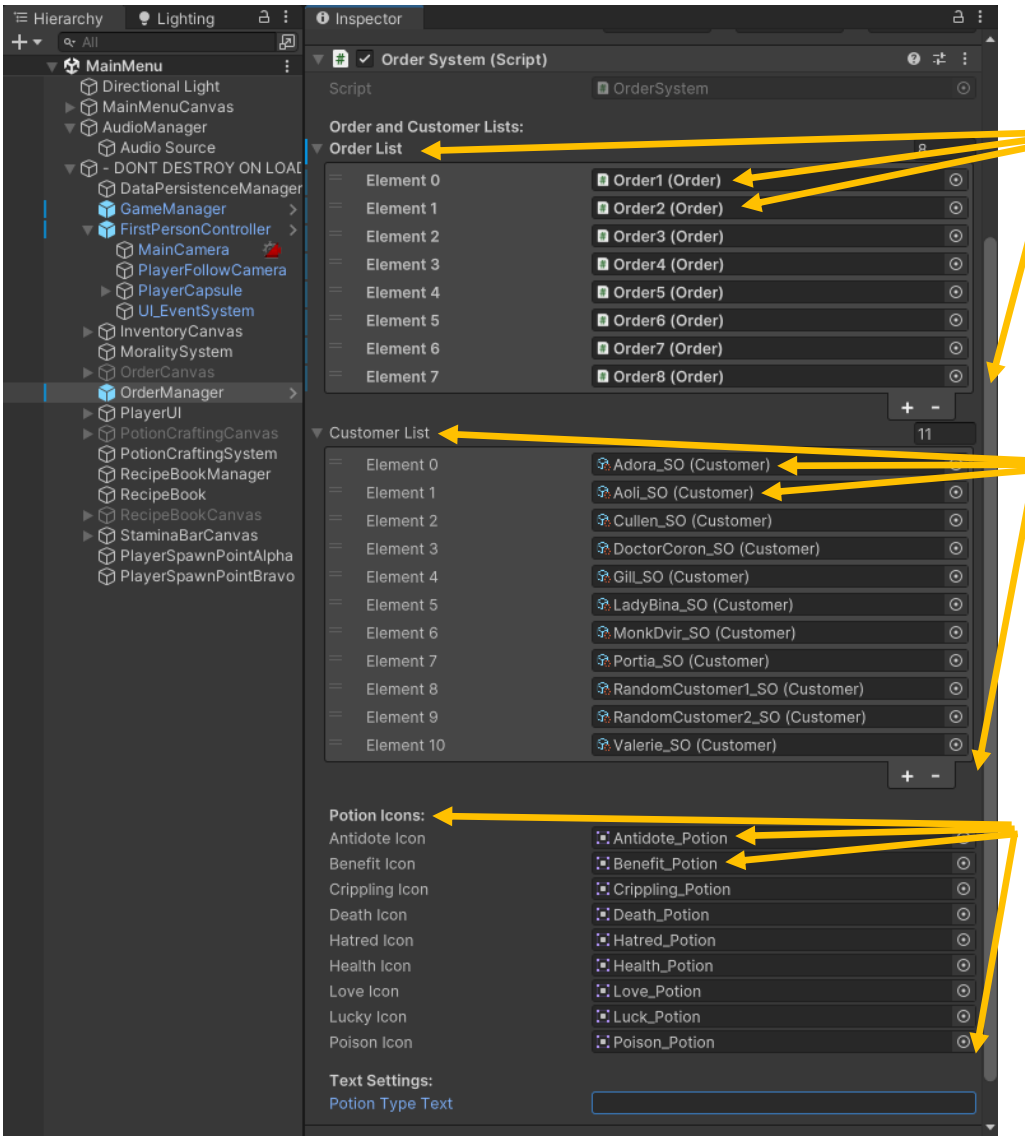


The music box is a component that gets attached to a basic, wander AI, which functionally turns it into the music enemy. The music enemy wanders around the maze looking for the player, and when they spot them, they begin chasing them. A song also starts playing upon the player being detected, and if 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.



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.



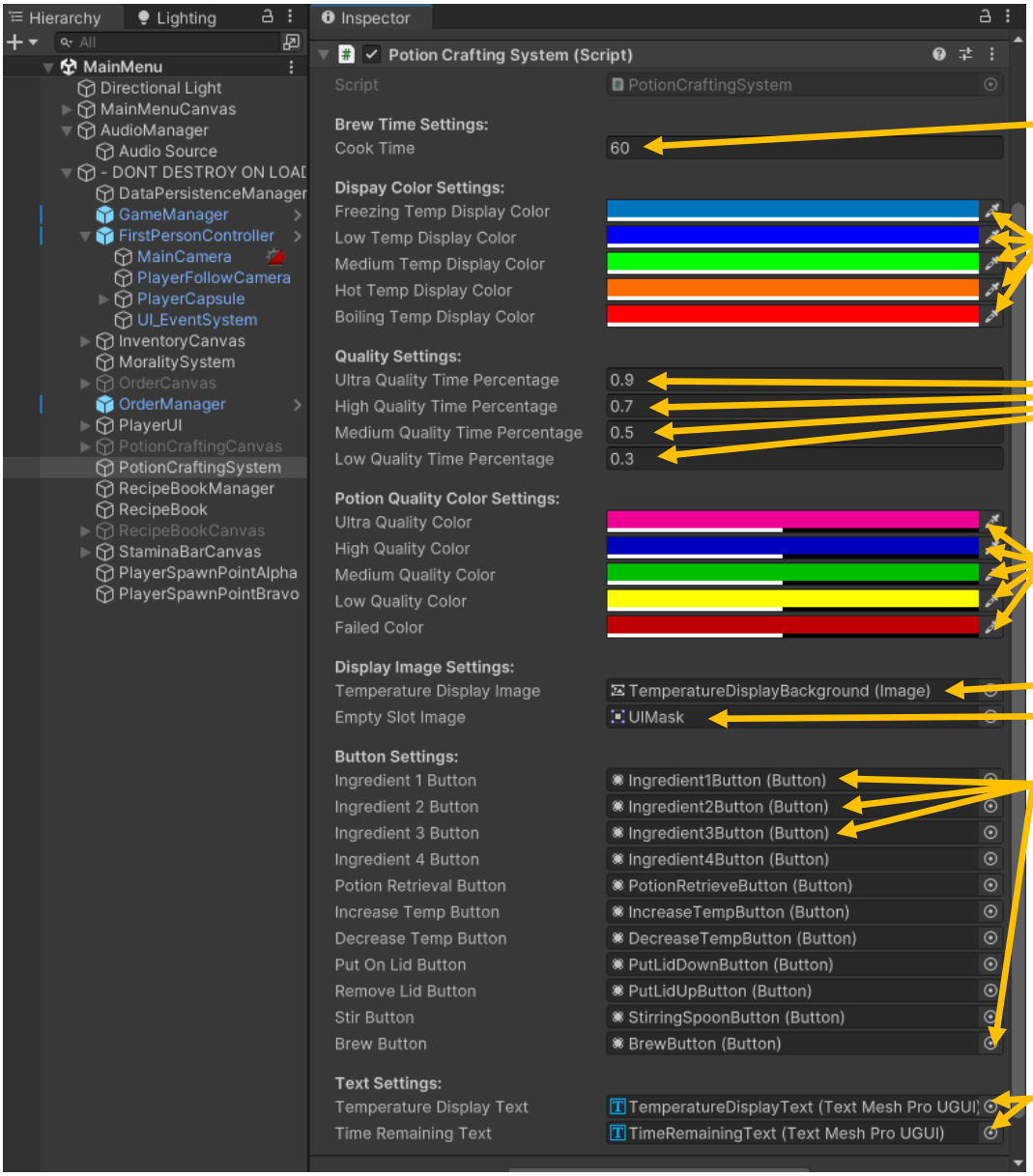
List of current orders which are displayed in the potion shop (randomized each day)

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).



Set length of time that potions will take to brew

Choose display colors for the temperature setting background

Percentages needed to determine the quality of potion when being brewed, based on lid placement, stirring, and temp

Choose display colors for the potion quality background

Set temperature display image

Set empty slot sprite/mask

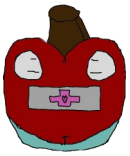
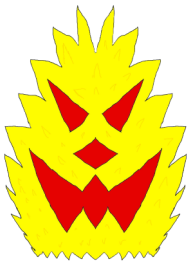
Set the Button game objects for the potion crafting system

Set the TMP Text game objects for the potion crafting system

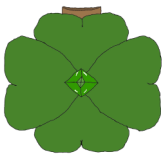
Potion Data:

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***)

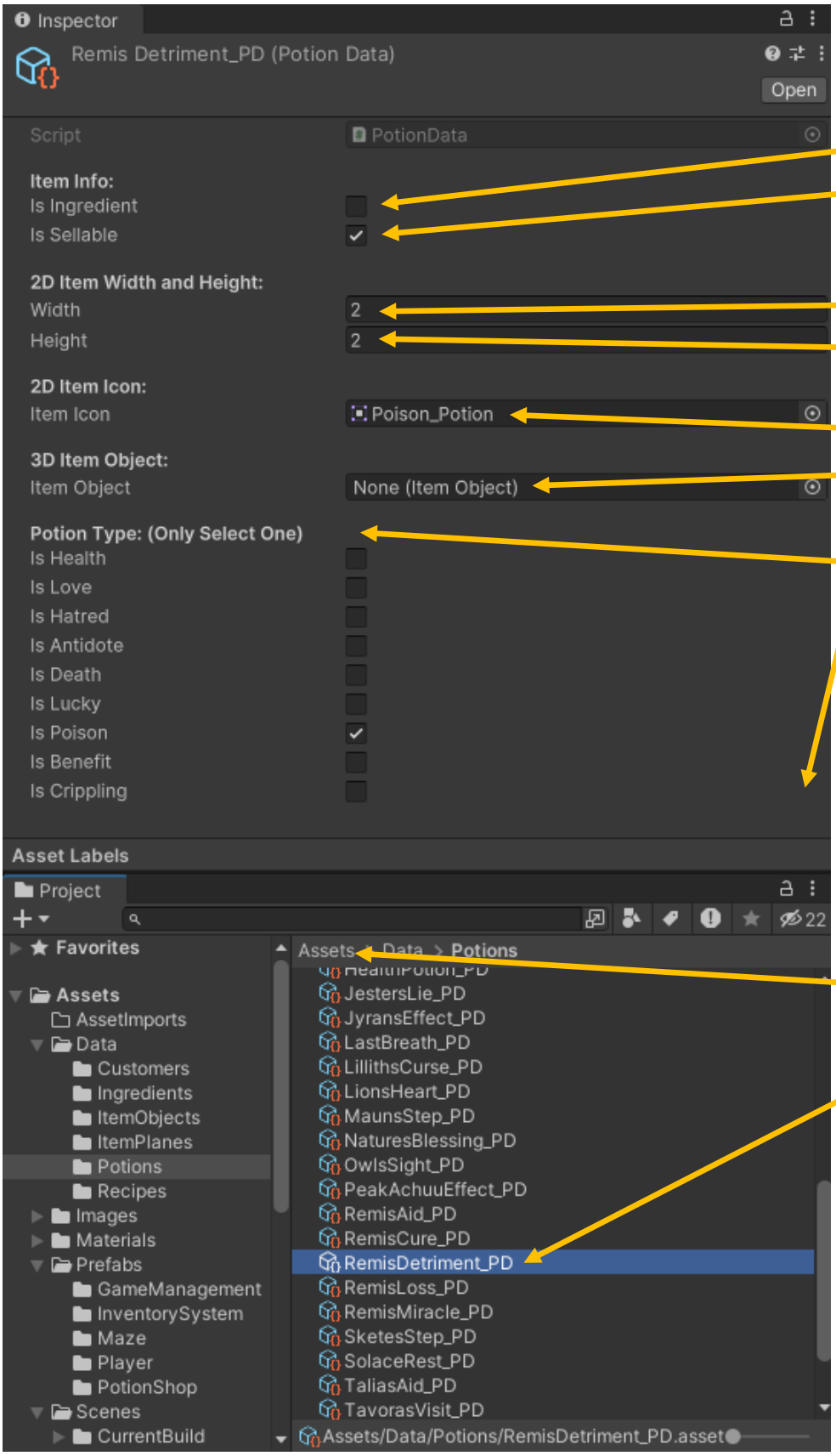




Potion Data (cont…):



Do not set as ingredient for a potion

Set as sellable for a potion

Set width of potion in inventory

Set height of potion in inventory

Set the 2D icon for this potion

Set the planar 3D representation of the potion

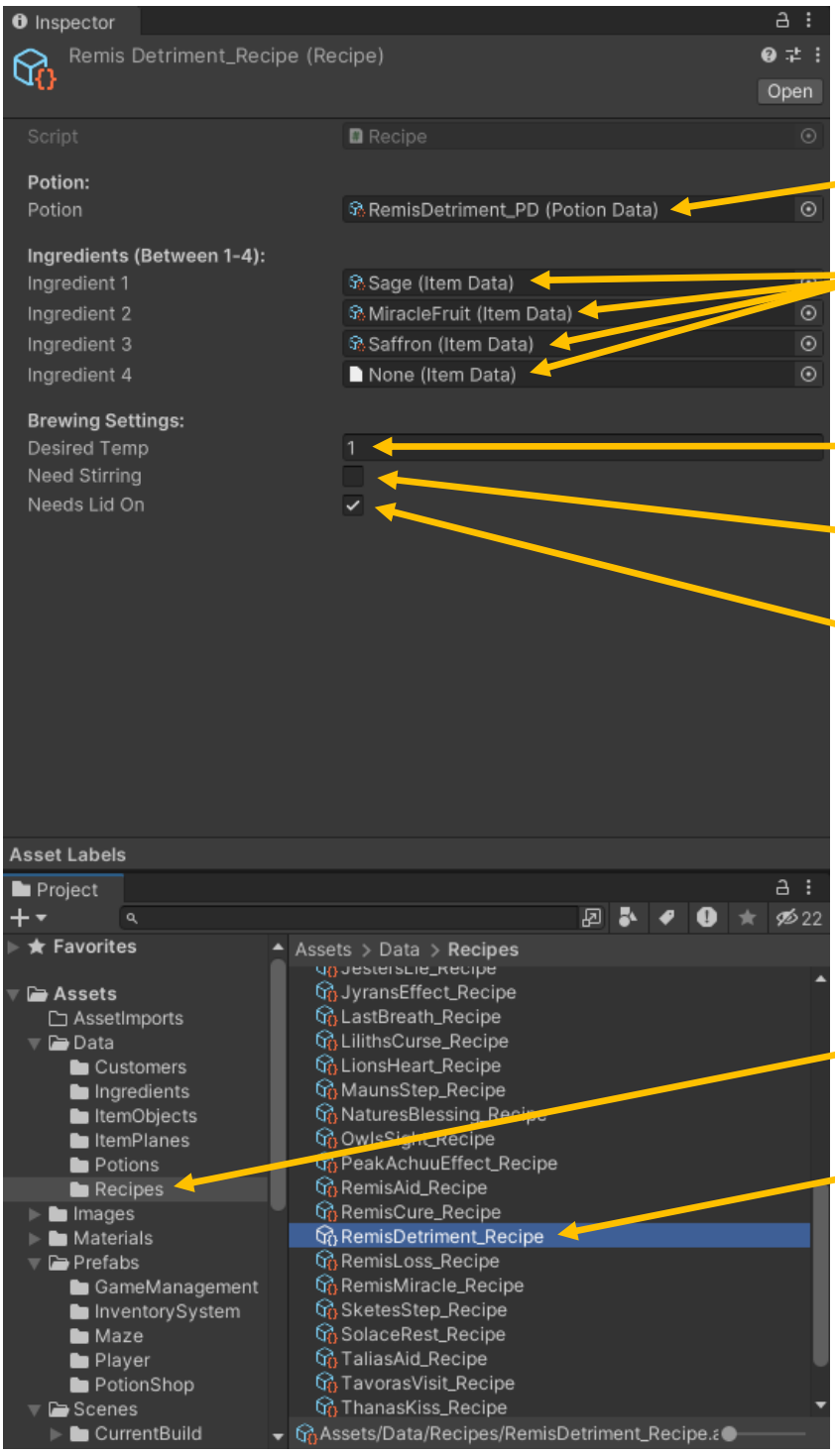
Select the type of potion (only one)

Directory where the potions (potion data game objects) are located

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

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).



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

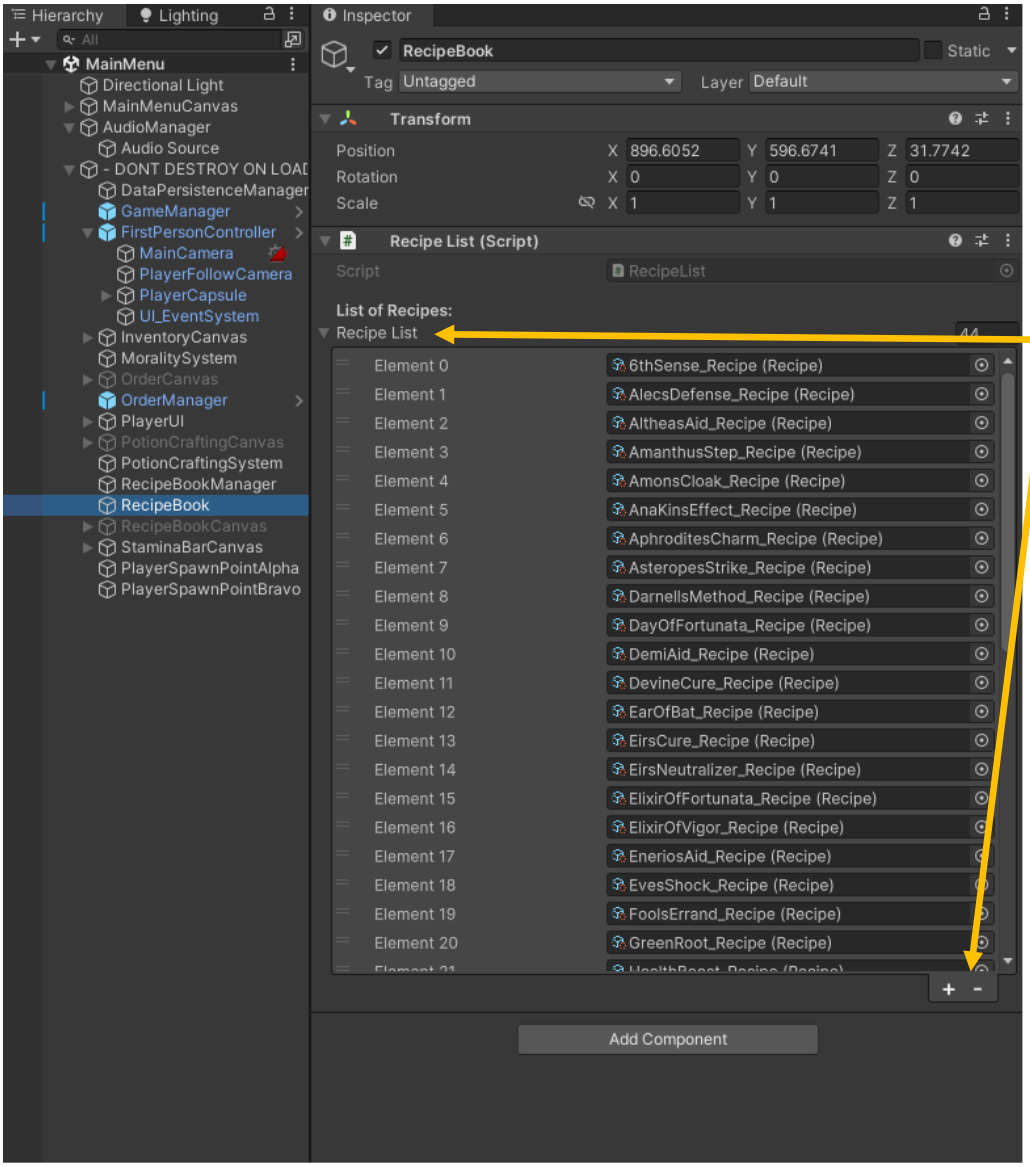
Set whether the recipe desires the lid to be on the cauldron

Directory where the recipes (recipe game objects) are located

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

Recipe Book:

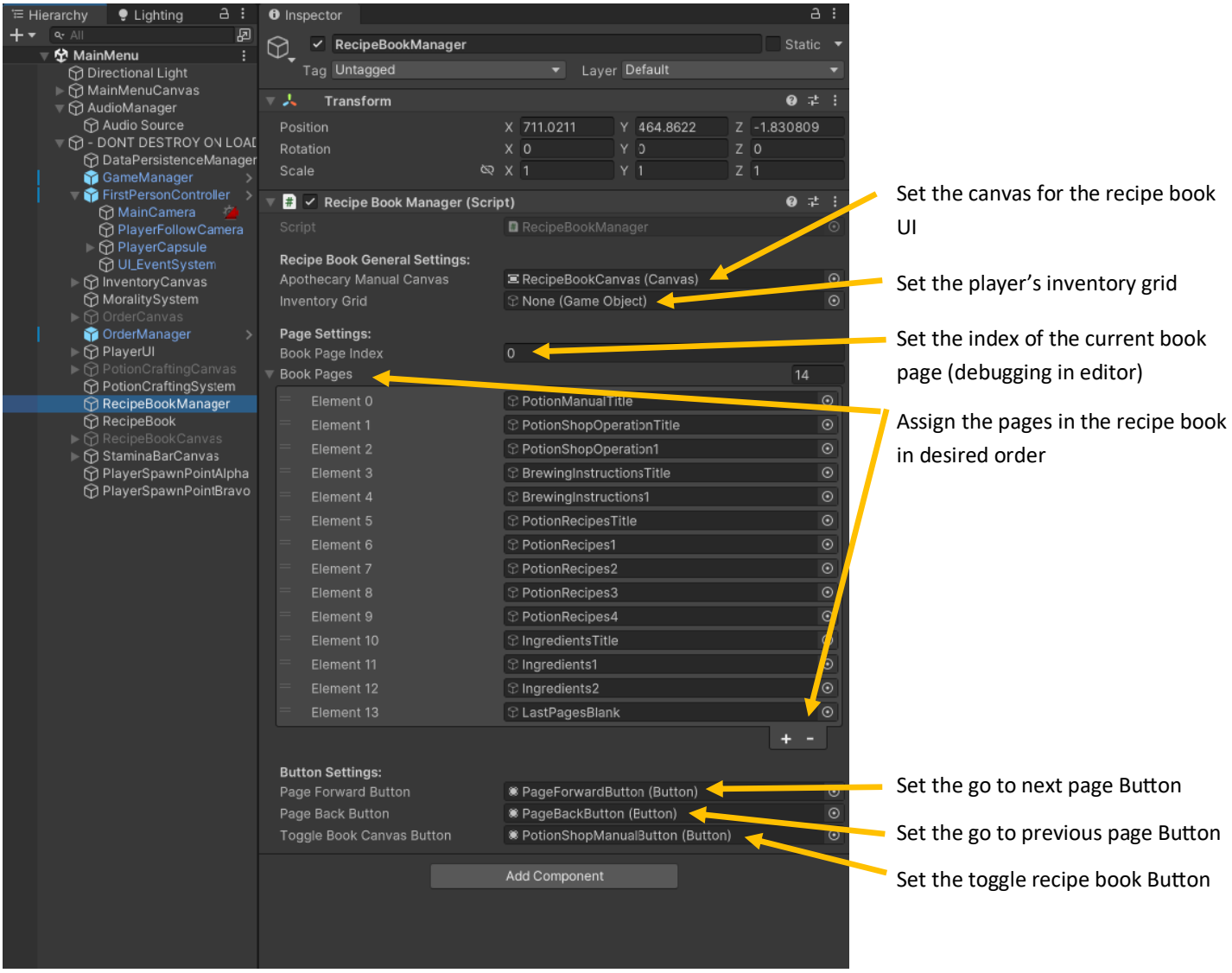
The recipe book is the game object that holds all of the different recipes (combinations of ingredients) that can be made in the game. New recipes are easily made by creating new Recipe scriptable objects, which also requires creation of a new Potion (potion data) scriptable object that the recipe is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



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

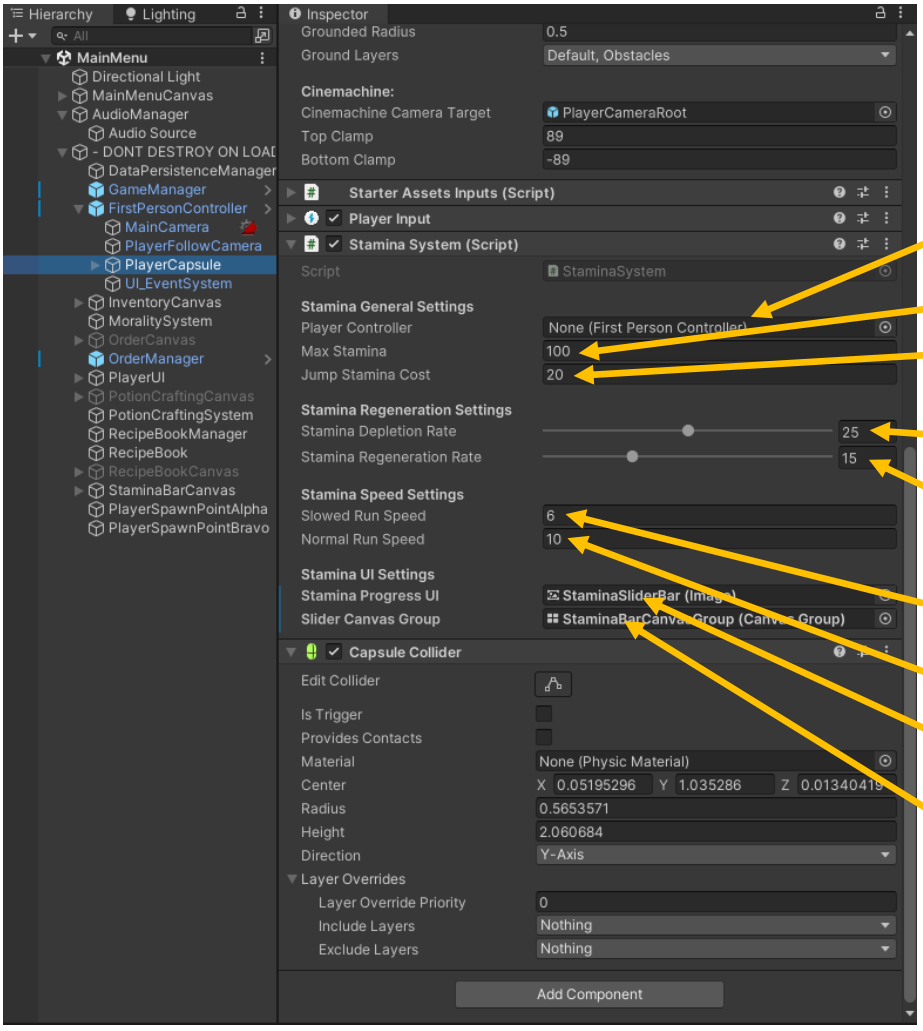
Recipe Book Manager:

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.



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).