Quarantine Framework- Help

1. Gneral Tips

1.1. Subtopic 1

ShowStatsPanel

Setp new input

2. Basic Scene Setup

2.1. Create a new Blank Scene

2.2. Delete the existing MainCamera from the hierarchy

This will be superseded and replaced by the playerControlManager and Player cameras.

2.3. Add a PlayerControllerManager prefab to the scene

This Prefab contains everything required to spawn and track the Player, their stats and communicate with the Hud. The Player will spawn at this Gameobjects Location when each level begins.

Position the PlayerController at the location you wish the player to Spawn.

2.4. Add a PlayerHudCanvas prefab to the scene

This component handles the display, animation and update of the players Hud. It can be used to display Messages, Tutorial Prompts, Objective instructions, Completion Messages and includes a Damage Overlay which triggers when the player's DamageHandler registers damage.

3. Damage and Kill Zones

This leverages the DamageHandler's flexibility. By adding a Damage Handler with sufficiently high Damage and "Applies Damage" set, this object can serve to 'catch' the player or server as a danger volume the player must not come into contact with. The Damagehandler takes care of registering the player and communicating the Respawn functionality to the PlayerControllerManager, ensuring the player respawns and can continue.

3.1. Create a Box Gameobject (or any object with a Collider Component)

3.2. Add a DamageHandler component to the object

Add Component>DamageHandler

3.3. tick Applies Damage. this will ensure the player takes damage when they overlap with the Volume

3.4. Set the desired Damage To Apply Amount

The player begins with 100 health by default, so if you want to instantly kill the player, set this number to 100 or higher.

3.5. Set the size and position of the Kill zone

3.6. The player will now die when they contact the Kill Zone, and respawn at the PlayerController's location.

4. Teleporter

The Teleporter allows you to move the player instantly from one part of your level to another. It is simplistic but effective NOTE: This facilitates one-way Teleporter travel, to set up bi-directional teleporting place a separate and independent Teleporter at the Destination location (NOT overlapping with the Destination from the first teleporter!) and this will aloow the player to go back to the original location. If you put these too close together you will find the player ping-pongs between them, as it will immediately register travel in the other direction once the player is teleported.

4.1. Place a Teleporter Prefab in the scene

Or add a C_Teleporter component to any gameobject with a Collider set to IsTrigger.

4.2. Specify the Target Gameobject to use as a Destination

This can be an Empty Gameobject, we only care about the Transform of this Destination Object.

4.3. The player will instantly relocate to the Target Gameobject Position

Take care to position your Target object above the ground, as the player may fall through the floor if they spawn too low to the ground.

5. Grabbables

5.1. To make any object Grabbable, set the Object's Tag (Top left of the Inspector) to Grabbable (you can create this tag if necessary)

5.2. Ensure the object has a Collider component (it will not respond to the Grab Trace if this is missing)

5.3. If the object tagged as Grabbable contains a Rigidbody, the object will be picked up using a Physics Constraint.

5.4. If the object is not using a Rigidbody it will snap to the player's 'hand' or grab position.

6. Level Switching

6.1. Place a Level Load prefab into the scene

6.2. Specify the Scene Name you wish to transition to

6.3. Ensure the Level name is spelled correctly, including Capitalisation.

6.4. Ensure the desired scene has been added to the List of Maps you Build in the Build Settings panel. File>Build Settings

6.5. To create a custom Level Loading trigger

Create Gameobject with a Collider Component

Set the Component to is Trigger

Add the Level Load component to your gameobject

Specify the desired level as described previously.

6.6. The player will now trigger a new scene to load when they overlap with this volume.

6.7. This actor can be disabled for specific activation later, for example when the player completes an objective or places the desired Gameobject on a spearate Slot.

7. MultiDoor System

7.1. Basic Setup

The Door system uses a Root gameobject as a root point.

The actual Door objects are Children of this Root object

These door objects shoul dhave their pivots set appropriately from the Art Package (maya) so that they are centered around the right point

The Door Objects should have Colliders and Door Movement scripts applied to them, such as Swing Door or Sliding Door.

The MultiDoor script on the Root will find any Door Components in its children, and cause them to open or close when it is triggered.

7.2. Locking/Unlocking Doors

Doors have a Locked variable which can be modified through the use of a Door key Pickup, Door Unlock Interaction script (for instance on a button or slot), or through your own script.

Button Example

Slot Example

7.3. Triggered Vs Interactable Doors

The Doors can be set to respond only to direct Interaction, or use a Collider as Trigger on the Root Object to detect the Player and open and close accordingly

The doors will automatically open away from the player based upon which side of the door they are on when they trigger or interact with the door.

8. Interactables

The Interactable System is intended to allow for the user directly clicking on objects within their level, and affecting other objects, triggering behaviours or changing states.

It operates by placing an Interactable script component on the desired object, and specifying the Behaviour type desired.

In some cases, custom Interactables have been created to allow for oft-used functions such as locking or unlocking doors, turning Lights on or off, or entirely Activating and Deactivating targeted GameObjects in the scene.

The Interactor is a script applied to the PlayerController, which will perform a trace into the world when the user presses the Fire input, and signal any Interactables that it hits that they should Start Interaction.

8.1. Buttons

8.2. Interactable Types

9. CheckPoints

The Checkpoint system is primitive, using a Trigger Overlap to register the Player, and moving the PlayerControllerManager Gameobject in the scene to the Checkpoint's Position. This means that if the player dies, they will respawn at the PlayerControllerManager Position as intended, but that has now been relocated to the Checkpoint's position. This give the appearance of a checkpoint without needing to store additional Data and allowing for levels to include multiple segments the player can try and fail in without need to restart the entire level.

9.1. Place Checkkpoint Prefab in scene

9.2. Adjust Collider size to desired Volume

9.3. Custom Checkpoint:

Create any Gameobject with a Collider Component, or an empty one to which you add a Collider

Ensure the Collider is set to IsTrigger

Add the Checkpoint script to the Checkpoint Gameobject.

The player will now respawn at the Checkpoint Location after triggering it.

10. Pickups

10.1. Apply commodities to PlayerController

11. Slot System

11.1. Will slot the specified desired Target Object when that object overlaps with the Slot Collider