Unity Versatile Third Person-Controller System Documentation

Table of Contents

- Introduction
- Installation
- Getting Started
- Dependencies
- Scene Setup
- Data Management with Scriptable object
    - Player Configuration
    - Player Movement Configuration
    - Player Jump Fall Configuration
    - Player Ground Configuration
    - Player Collider Configuration
    - Player Cover Configuration
    - Player Climb Configuration
    - Player Vault Configuration
    - Player Roll Configuration
    - Player Ray Configuration
    - Player Health Configuration
    - Player Stamina Configuration
    - Player Aim Lock Configuration
- TP-Controller System Components
    - Player Movement Context
    - Player Movement Base State
        - Idle State
        - Walk State
        - Jog State
        - Run State
        - Jump State
        - Fall State
        - Cover State
        - Climb State
        - Vault State
        - Roll State
        - Aim Lock State
        - Interact State
    - Player Controller Manager
        - Player Controller
        - Player UI Controller
    - Input Action Key Manager
        - Input Action Manager
    - Camera System Manager

        - Free Roam Camera
        - Aim Lock Camera
    - Health Base System
        - Take Damage
        - Heal
    - Stamina Base System
        - Consume Stamina
        - Recharge Stamina
    - Interactable Base System
        - Door
        - Elevator
    - UI Base System
        - Health UI
        - Stamina UI
        - Pause Menu UI
        - Controls Wizard
    - Interfaces
        - I-Health-State
        - I-Stamina-State
        - I-Interactable-State
- Advanced Features
    - Consume Stamina
    - Recharge Stamina
    - Height Fall Damage
    - Recharge Health
- Usage Examples
    - Character Configuration Setup
    - Player Controller Setup
    - Player Health Setup
    - Player Stamina Setup
    - Player Controller UI Setup
    - Door Setup
    - Elevator Setup
- Troubleshooting
- FAQs
- Support

1. Introduction

Welcome to the Unity Versatile Third Person Controller System documentation! This asset provides a flexible and customizable player management solution for your Unity projects. Whether you are creating a game or simulation, this controller simplifies the implementation of main player functionalities.

2. Installation

To install the Unity Versatile Third Person Controller System, follow these steps:

- Open your Unity project.
- Go to the Unity Asset Store.
- Search for "Unity Versatile Third Person Controller System" and click on the asset.
- Click the "Download" button.
- Import the package into your Unity project.

3. Getting Started

After installation, follow these steps to get started with the Unity Versatile Third Person Controller System:

- Add the Player Controller Component to your Player.
- Add the Input Action Manager Component to your player.
- Add Capsule Collider to the player game object.
- Add Rigid body to the player game object.
- Add Animator to the player game object.
- Add Player Health to the game object.
- Add Player Stamina to the player game object.
- Add Player Controller UI to the player game object.
- Add Player Hand IK System to the player game object.

4. Dependencies

The package relies on the following Unity packages:

- Cinemachine
- Input System
- Animation Rigging
- Pro Builder: for the demo scene

5. Scene Setup

To setup the Versatile controller in a new scene, you need to accomplish some simple steps

1. Layers: add the following Layers:
    a. Ground
    b. Interactable
    c. Player
2. Tags: add the following tags:
    a. CinemachineTarget

b. HighCover
c. LowCover
d. Ladder
e. Interactable

6. Data Management with Scriptable Object

Character Config Script

The `Character config' Class is a scriptable object in Unity. This class holds all the required configurations for any character in the game. It encapsulates various aspects of character's behavior. It contains several public properties each of which is an instance of a specific configuration class.

Movement Config Script

The `Movement Config' Class is a scriptable object that holds the configuration settings for character's movement. This class can be created as an asset in the unity project by right-clicking in the project window and selecting `Create > Character > Movement > Config`.

Ground Config Script

The `Ground Config` class is a Scriptable Object that holds the configuration settings for a character's interaction with the ground. This includes settings for determining whether the character is grounded. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Ground > Config`.

Jump Fall Config Script

The `Jump Fall Config` class is a Scriptable Object that holds the configuration settings for a character's jump and fall dynamics. This includes settings for jump height, jump time, terminal velocity, fall time, and gravity. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Jump Fall > Config`.

Aim Lock Config Script

The `Aim Lock Config` class is a Scriptable Object that holds the configuration settings for a character's aiming. This includes settings for the aim look weight speed, active aim look rig weight, aim idle speed, strafe walk speed, and various animation thresholds. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Aim Lock > Config`.

Collider Config Script

The `Collider Config` class is a Scriptable Object that holds the configuration settings for a character's collider. This includes settings for the height and center of the character's collider in different states such as standing, crouching, crawling, jumping, and vaulting. This class can be created as an asset in

the Unity project by right-clicking in the project window and selecting `Create > Character > Collider > Config`.

Climb Config Script

The `Climb Config` class is a Scriptable Object that holds the configuration settings for a character's climbing. This includes settings for the climb up threshold, climb down threshold, climb motion speed, climb foot distance, exit climb up timeout, exit position Y offset, exit position Z offset, enter climb down timeout, enter climb position Y offset, and enter climb position Z offset. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Climb > Config`.

Cover Config Script

The `Cover Config` class is a Scriptable Object that holds the configuration settings for a character's cover. This includes settings for the thresholds for looking right and left while in cover, and for walking and looking right and left while in cover. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Cover > Config`.

Roll Config Script

The `Roll Config` class is a Scriptable Object that holds the configuration settings for a character's rolling. This includes settings for the time it takes to complete a roll and the speed of the roll animation. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Roll > Config`.

Vault Config Script

The `Vault Config` class is a Scriptable Object that holds the configuration settings for a character's vaulting. This includes settings for the time it takes to complete a vault, the speed of the vault animation, and the force applied to the character when vaulting. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Vault > Config`.

Ray Config Script

The `Ray Config` class is a Scriptable Object that holds the configuration settings for a character's ray casting. This includes settings for the distance of the ray used to interact with objects such as ladders, covers, and doors. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Ray > Config`.

Health Config Script

The `Health Config` class is a Scriptable Object that holds the configuration settings for a character's health. This includes settings for the maximum health of the player and the fall damage values. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Health > Config`.

Stamina Config Script

The `Stamina Config` class is a Scriptable Object that holds the configuration settings for a character's stamina. This includes settings for the maximum stamina of the player, the rate at which stamina is drained while walking, jogging, and sprinting, and the rate at which stamina is recovered. This class can be created as an asset in the Unity project by right-clicking in the project window and selecting `Create > Character > Stamina > Config`.

7. Third-Person Controller System Components

Player Movement Context Script

The `Player Movement Context` class provides a context for the player movement. It represents the context of all the components and settings required for the player's movement.

Player Movement Base State Script

The class is an abstract and a comprehensive and extensible class that manages the player's interactions and movements within the game. It uses a state machine design pattern to handle different states of the player, providing a structured and efficient way to manage complex player behaviors.

Player Idle State Script

The class is part of a state machine that controls the player's idle state in the game. It Uses information for the player’s collider (Height & Center). It also handles the transitions to all the other states. It also Recharge the stamina of the player.

Player Walk State Script

The class is part of a state machine that controls the player walking state in the game. It Uses information for the player’s collider (Height & Center). It also handles the transitions to all the other states. It also recharges the stamina of the player after a certain time and if the player is walking for a long time it will start to consume stamina.

Player Jog State Script

The class is part of a state machine that controls the player jogging state in the game. It Uses information for the player’s collider (Height & Center). It also handles the transitions to all the other states. It also consumes the stamina of the player after a certain time of start jogging.

Player Sprint State Script

The class is part of a state machine that controls the player's sprinting state in the game. It Uses information for the player’s collider (Height & Center). It also handles the transitions to all the other states. It also consumes the stamina of the player after a certain time of start sprinting.

Player Crouch State Script

The class is part of a state machine that controls the player's crouching state in the game. It Uses information for the player’s collider (Height & Center). It also handles the transitions to other states. It also recharges stamina after a certain time from entering the crouch state.

Player Crawl State Script

The class is part of a state machine that controls the player's crawling state in the game. It Uses information for the player’s collider (Height & Center). It also handles the transitions to other states. It also recharges stamina after a certain time from entering the crawl state.

Player Aim Lock State Script

The class is part of a state machine that controls the player's movement in the game. This specific state represents the player aiming and locking onto a target. It handles the animation, position adjustment, and state transition for when the player is aiming and locked onto a target.

A. **Add a new float variable for the jogging threshold:** This variable will determine the speed at which the player will jog. It should be similar to the existing movement thresholds like (walkFwdThreshold, walkBwdThreshold, etc).
B. **Initialize the jogging threshold in the constructor:** Use the movementContext to initialize the jogging threshold. This will allow you to set the jogging speed in the game's configuration.
C. **Add a new entry in the inputSpeedMap dictionary for the jogging input:** The inputSpeedMap dictionary maps input actions to animation speeds. You'll need to add a new entry for the jogging input action and its corresponding animation speed.
D. **Update the HandleMovement method to handle the jogging input actoin:** This could involve checking if the jogging input action is triggered and then setting the targetSpeed to the jogging threshold.
E. **Update the SetAnimationSpeed method to set the animation speed for the jogging input action:** This will ensure that the correct animation is played when the player is jogging.

Player Cover State Script

The class is part of the state machine that controls the player's cover movement in the game. It Detects how the player will enter cover (High – Low). It uses information to determine the player's collider's height and center according to the height of the cover.

Player Climb State Script

The class is part of the state machine that controls the player's climb movement in the game. It detects how the player will climb and it uses information based on the ray cast if the player reaches the top of the object or the bottom. It initializes two rays, one from the eye and the other from the feet. The eye ray is cast in the forward direction to detect more steps while the foot ray is cast in the downwards direction to detect the distance between the player's feet and the ground.

Player Enter Climb State Script

The class is part of the state machine that controls the player when enter climb from above in the game. It determines how the player will enter the object below him, and it handles the position of both Y & Z offsets.

Player Exit Climb State Script

The class is part of the state machine that controls the player exit the climb object from top. It determines how the player will exit from above by adjusting both offsets of Y & Z positions.

Player Roll State Script

The class is part of the state machine that controls the player roll state. It determines how the player will roll based on the input and the camera rotation.

Player Vault State Script

The class is part of the state machine that controls the player vault state. It determines how the player will vault over objects, using vault animation, vault speed & vault force to simulate the vault behavior.

Player Death State Script

The class is part of the state machine that controls the player when his health reaches 0. It disables all the player functionalities.

Player Controller Script

The Player Controller class is the central component of Unity Versatile Third-Person Controller System. It manages the player's movement and transitions between different movement states.

When you add the Player Controller script to the game object, it will automatically add the required components which include:

1. Animator:

This component is responsible for controlling the animations of the player character based on the player's input and the character's state.

2. Capsule Collider:

This component is used for collision detection. It creates an invisible capsule shape around your player game object that can interact with other colliders.

3. Rigid body:

This component allows your player game object to be affected by physics, including forces and collisions.

4. Input Action Manager:

This component is responsible for managing player input. It translates player input into actions that the ***Player Controller*** can understand and respond to.

Player UI Controller Script

The `Player UI Controller` class manages the UI elements related to the player's health and stamina. It updates the UI elements when the player's health or stamina changes.

Input Action Manager Script

The `Input Action Manager` class handles the input from the player and stores the values for the player's movement and actions. It uses the new input system for Unity.

Camera System Manager Script

The `Camera System Manager` class manages the camera system for the player. It handles the camera movement, virtual camera activation, and the position of the mouse cursor in the game. It can be used to add more camera systems to the game.

Health Base System Script

The `Health Base System` class is an abstract class that provides a base for all health systems. It provides methods for initializing health, taking damage, and healing, as well as properties for the maximum and current health values and whether the object is destroyed. This system can be used on any game object in your game.

Player Health Script

The `Player Health` class manages the player's health. It extends the `Health Base System` class and implements the `I-Health-State` interface. It allows the player to take damage and heal.

Stamina Base System Script

The `Stamina Base System` class is an abstract class that provides a base for all stamina systems. It provides methods for initializing stamina, using stamina, and recovering stamina, as well as properties for the maximum and current stamina values and whether the stamina is drained.

Player Stamina Script

The `Player Stamina` class manages the player's stamina. It extends the `Stamina Base System` class and implements the `I-Stamina-State` interface. It allows the player to consume and recover stamina.

Interactable Base System Script

The `InteractableBaseSystem` class is an abstract class that provides a base for all interactable systems. It implements the `I-Interactable` interface, which requires an `Interact` method. It also provides an abstract method for getting the interaction point of the object.

Door Script

The `Door` class manages the behavior of a door, allowing it to be opened and closed. It extends the `InteractableBaseSystem` class and implements the `I-Interactable` interface.

Elevator Script

The `Elevator` class manages the behavior of an elevator, allowing it to move up and down. It extends the `InteractableBaseSystem` class and implements the `I-Interactable` interface.

Player UI Base System Script

The `Player UI Base System` class is an abstract class that provides a base for all player UI systems. It provides methods for showing and hiding UI elements.

Player Health UI Script

The `Player Health UI` class manages the player's health UI. It extends the `Player UI Base System` class and implements its abstract methods. It updates the health bar with the player's current and maximum health values.

Player Stamina UI Script

The `Player Stamina UI` class manages the player's stamina UI. It extends the `Player UI Base System` class and implements its abstract methods. It updates the stamina bar with the player's current and maximum stamina values.

Pause Menu System Script

The 'Pause Menu System' class is responsible for managing the game's pause menu and controls wizard UI elements. It is a part of the 'Player UI Base System' class hierarchy, extending its functionality to handle the specific needs of the pause menu system.

I Health State Interface

The `I-Health-State` interface defines the properties and methods required for a health system in the `True-Tactical-Studio` namespace.

I Stamina State Interface

The `I-Stamina-State` interface defines the properties and methods required for a stamina system in the `True-Tactical-Studio` namespace.

I Interactable Interface

The `I-Interactable` interface defines the methods required for an object to be interactable in the `True-Tactical-Studio` namespace.

8. Advanced Features

Consume Stamina

The section of this code is responsible for the player's stamina consumption. Stamina is typically consumed when the player performs certain actions such as (walking, jogging, sprinting). The rate of consumption can vary depending on the action preformed.

Recharge Stamina

The section of this code is responsible for recharging the player's stamina. After the player has consumed some stamina, it will gradually recharge over time when the player is not performing any stamina-consuming actions.

Fall Damage

This section of this code calculates and applies damage to the player's health when they fall from a significant height. The amount of damage is proportional to the distance fallen.

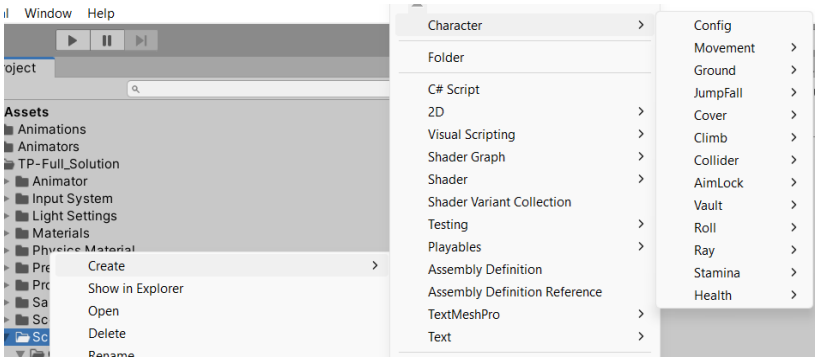
Recharge Health

This section of this code is responsible for recharging the player's health. If the player's health is not full, it will gradually recharge the player's health over time in certain states.

9. Usage Examples

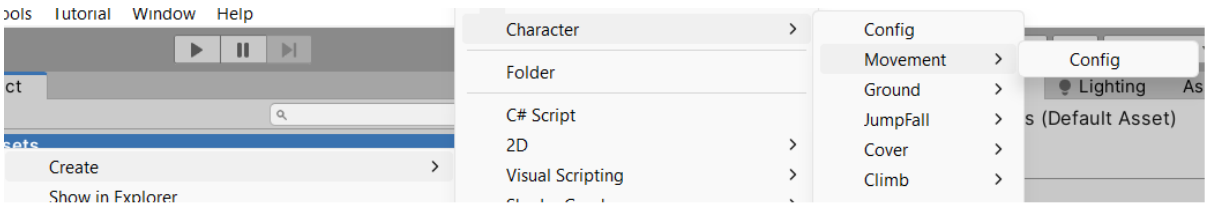
Character Configuration Setup

To setup the character configuration, right click in the project window and selecting create > character > config



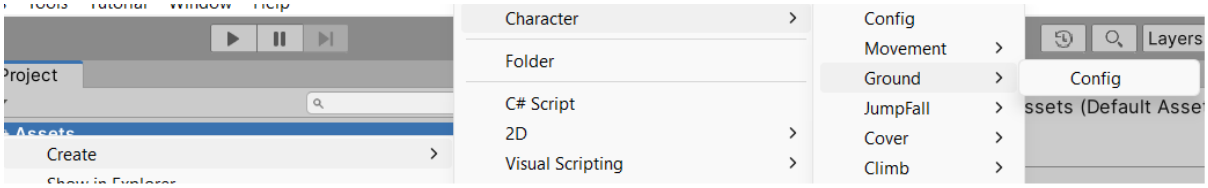
Movement Configuration Setup

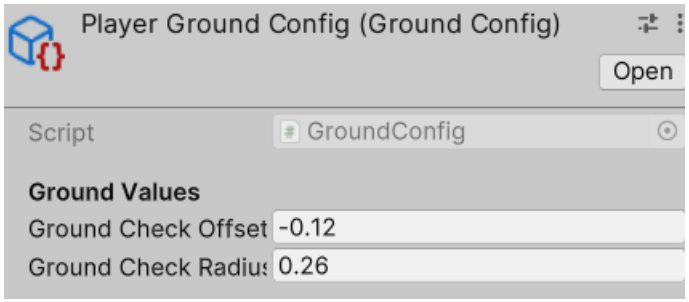
To setup the movement configuration, right click in the project window and selecting create > movement > config



Ground Configuration Setup

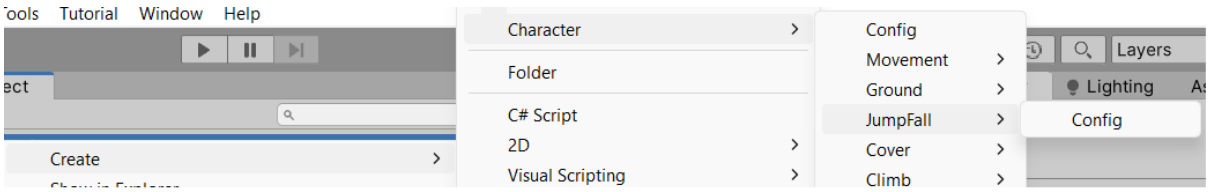
To setup the ground configuration, right click in the project window and selecting create > ground > config





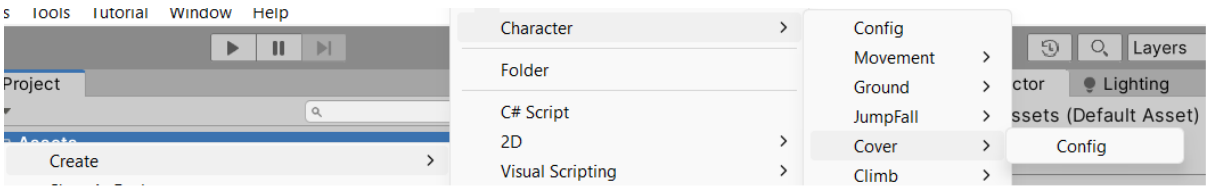
Jump - Fall Configuration Setup

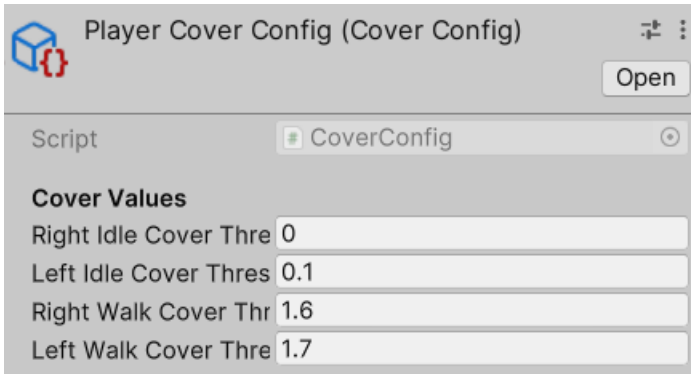
To setup the jump-fall configuration, right click in the project window and selecting create > jump-fall > config



Cover Configuration Setup

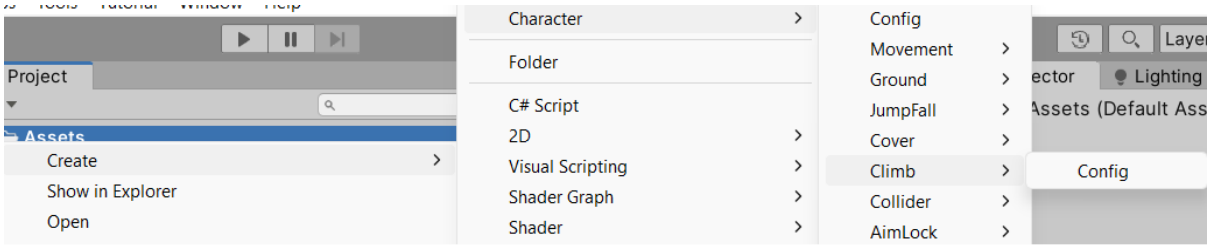
To setup the cover configuration, right click in the project window and selecting create > cover> config

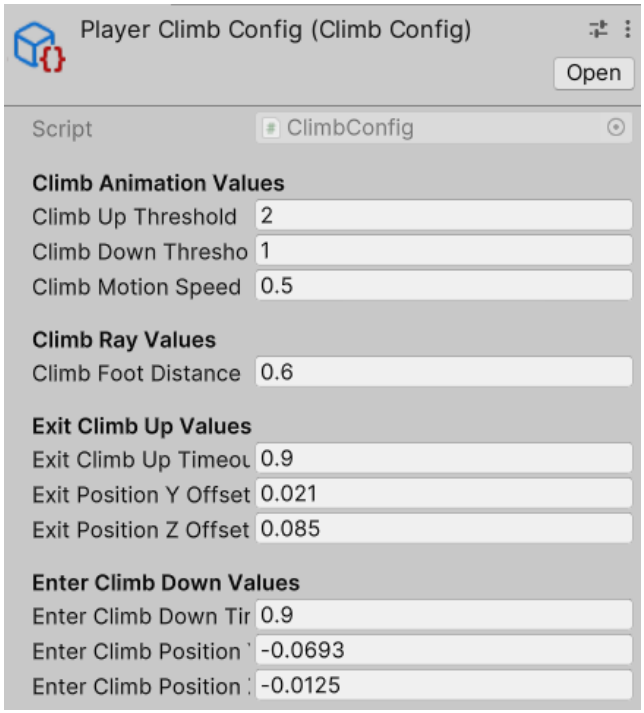




Climb Configuration Setup

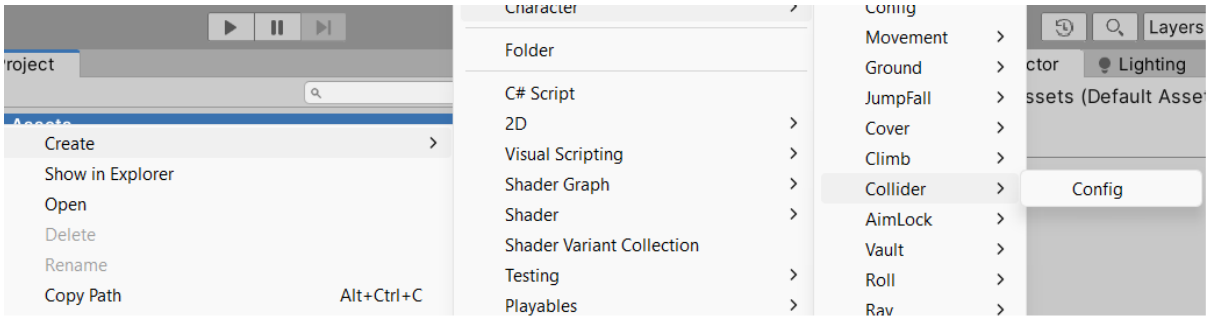
To setup the climb configuration, right click in the project window and selecting create > climb > config

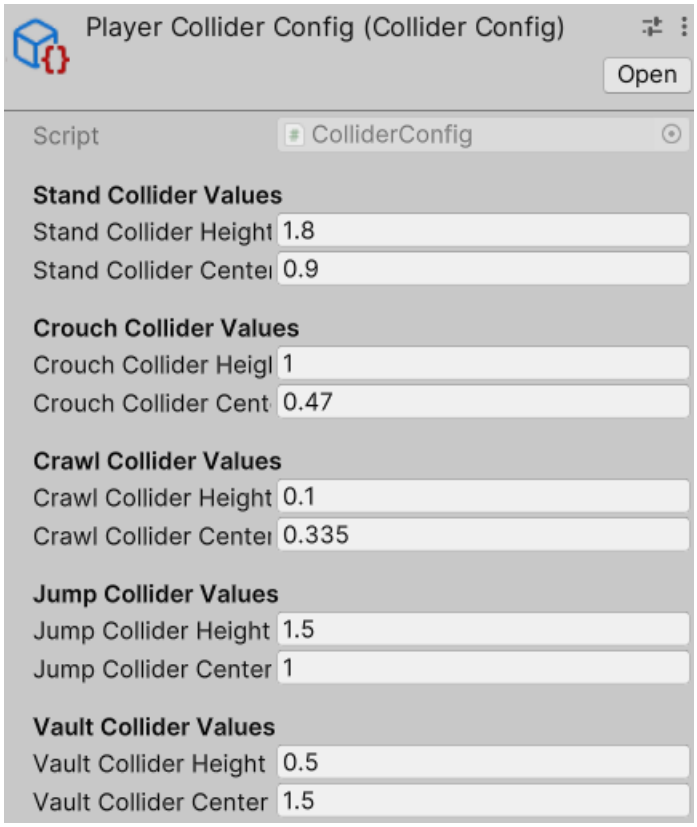




Collider Configuration Setup

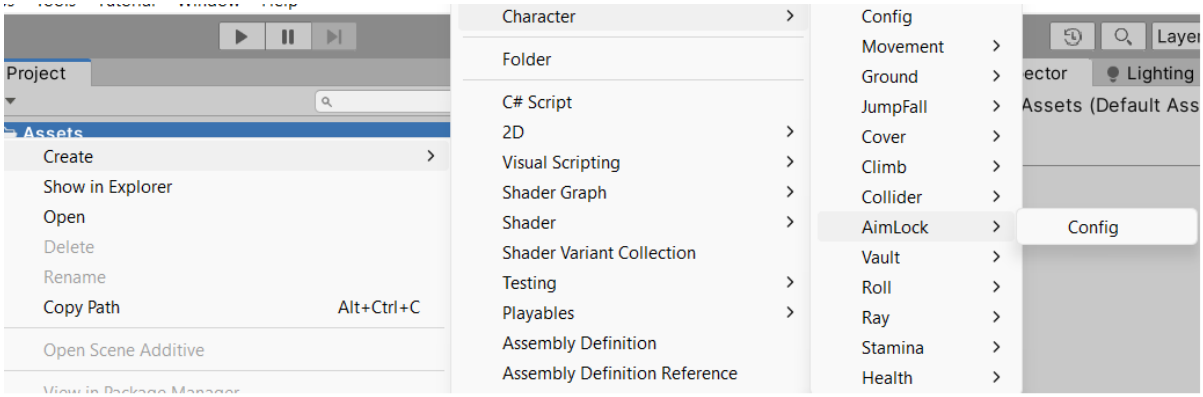
To setup the collider configuration, right click in the project window and selecting create > collider > config

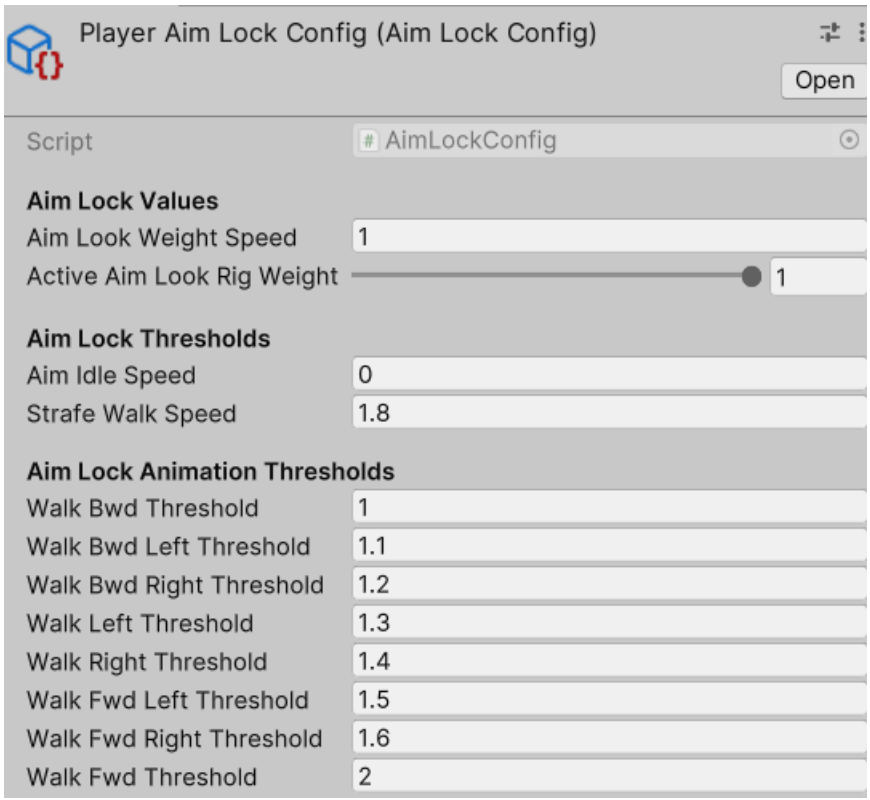




Aim Lock Configuration Setup

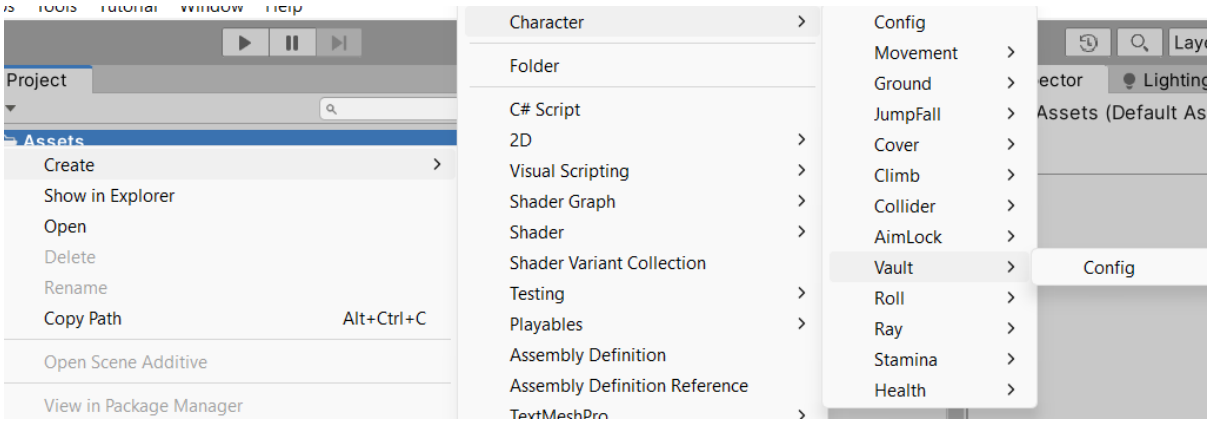
To setup the aim lock configuration, right click in the project window and selecting create > aim lock > config

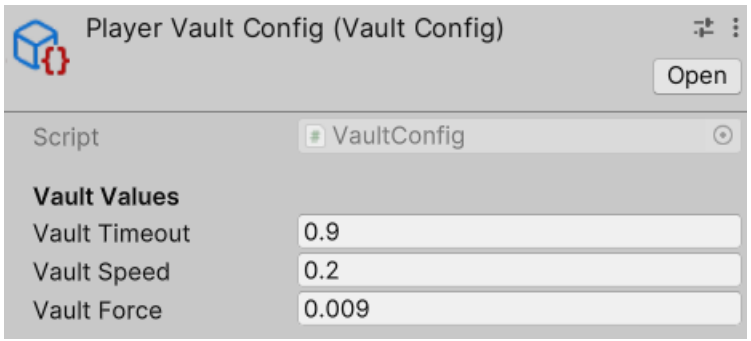




Vault Configuration Setup

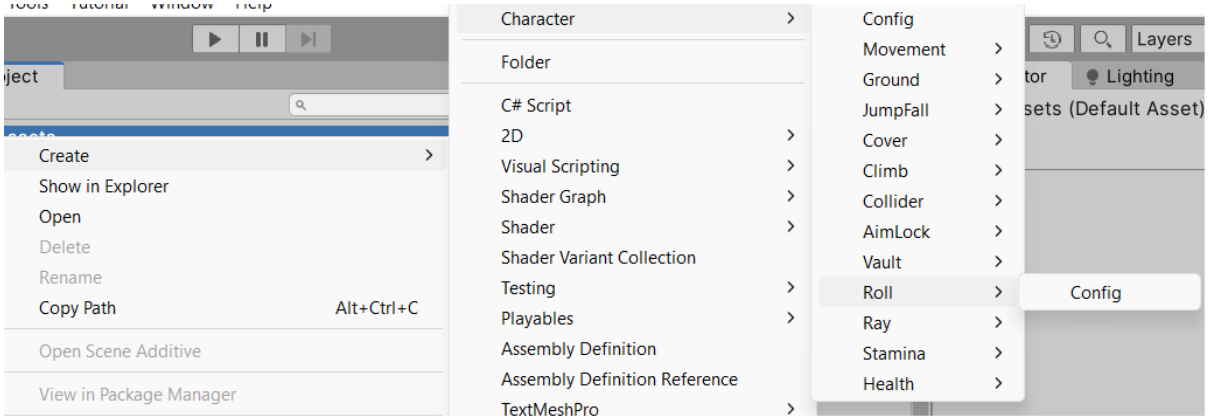
To setup the vault configuration, right click in the project window and selecting create > vault > config

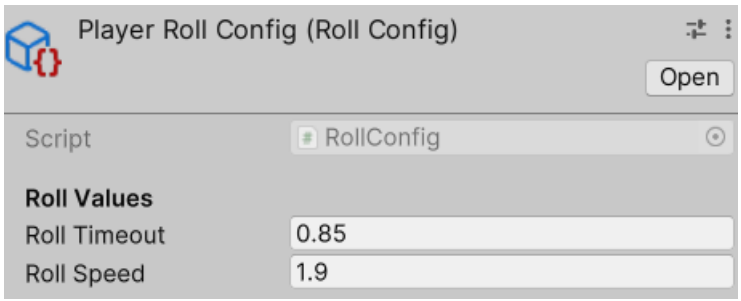




Roll Configuration Setup

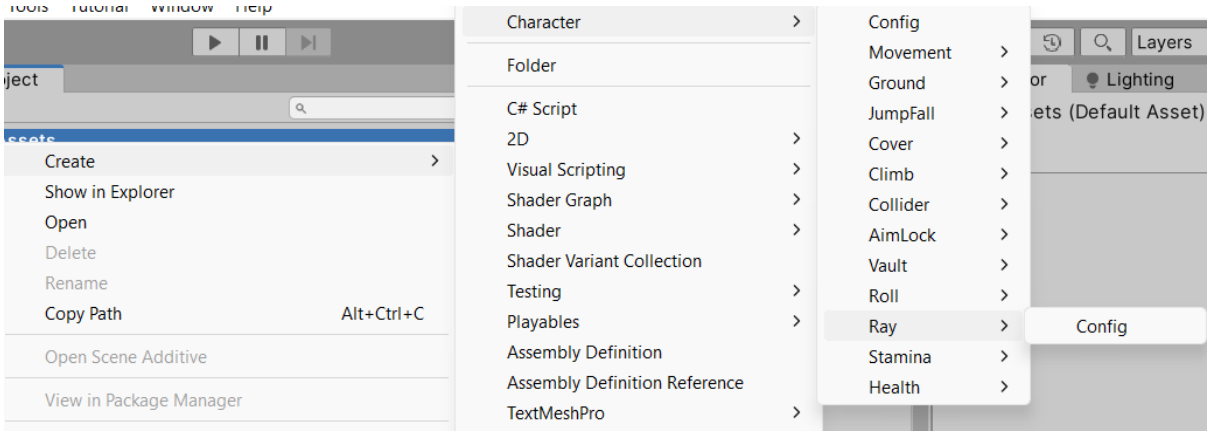
To setup the roll configuration, right click in the project window and selecting create > roll > config

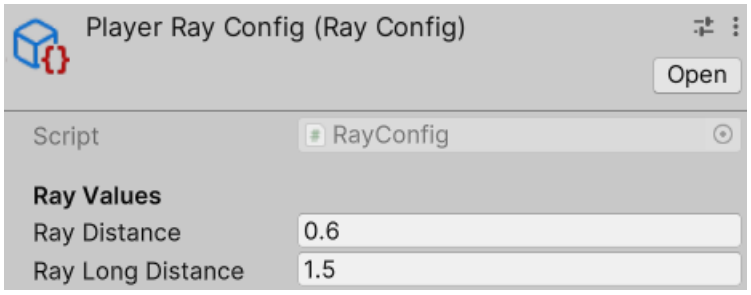




Ray Configuration Setup

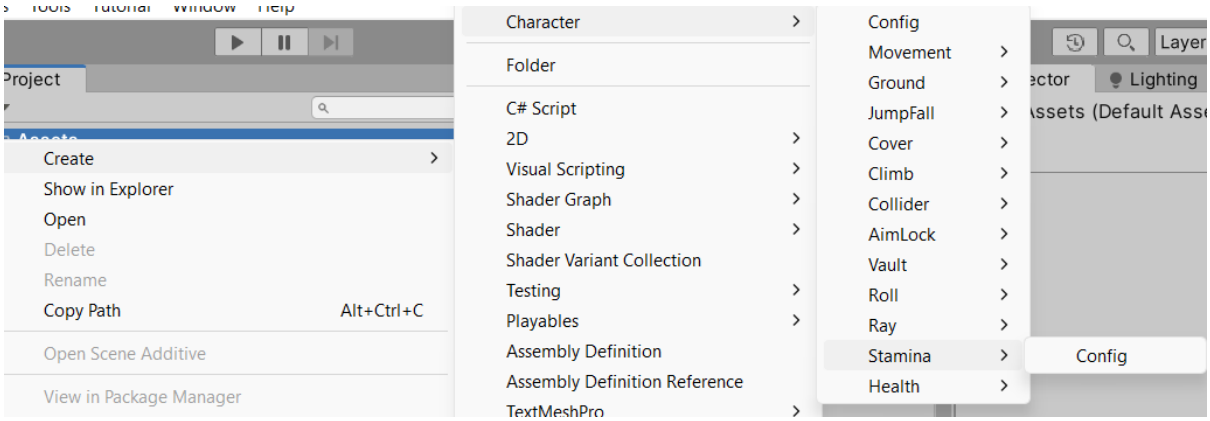
To setup the ray configuration, right click in the project window and selecting create > ray > config

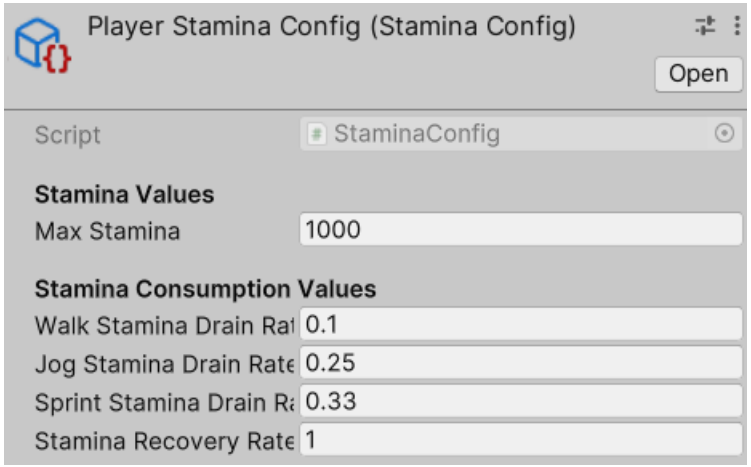




Stamina Configuration Setup

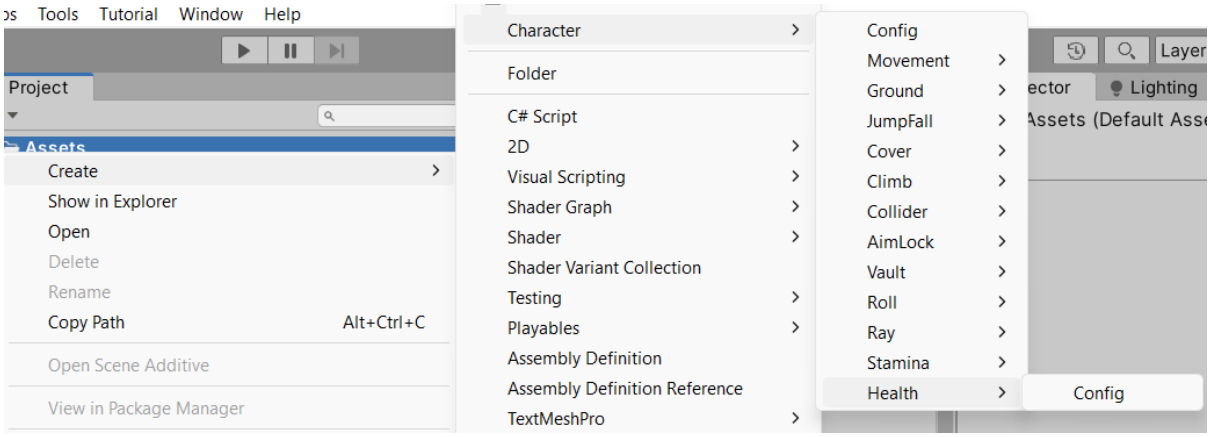
To setup the stamina configuration, right click in the project window and selecting create > stamina> config

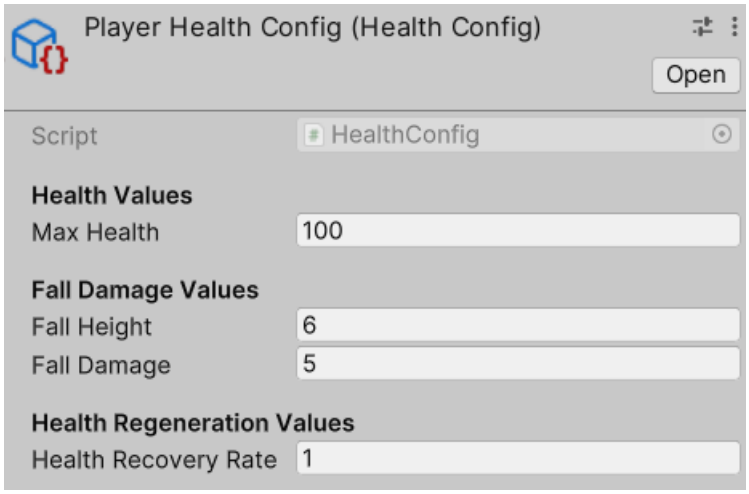




Health Configuration Setup

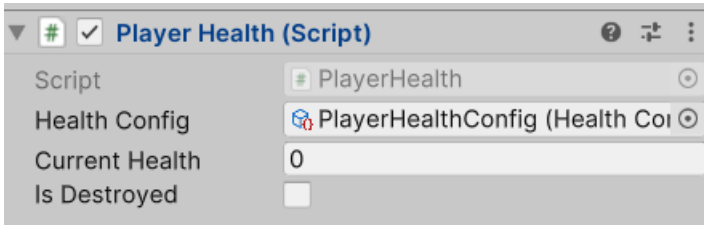
to setup the health configuration, right click in the project window and selecting create > health > config





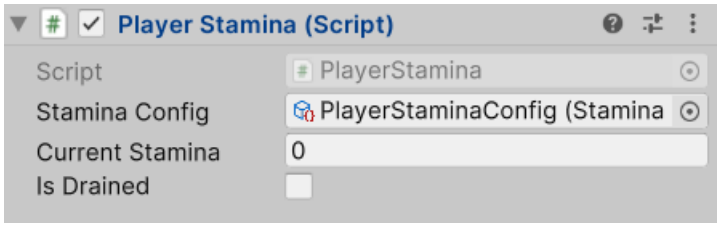
Player Health Setup

To set up the player health script, drag the health configuration you've created and assign it.



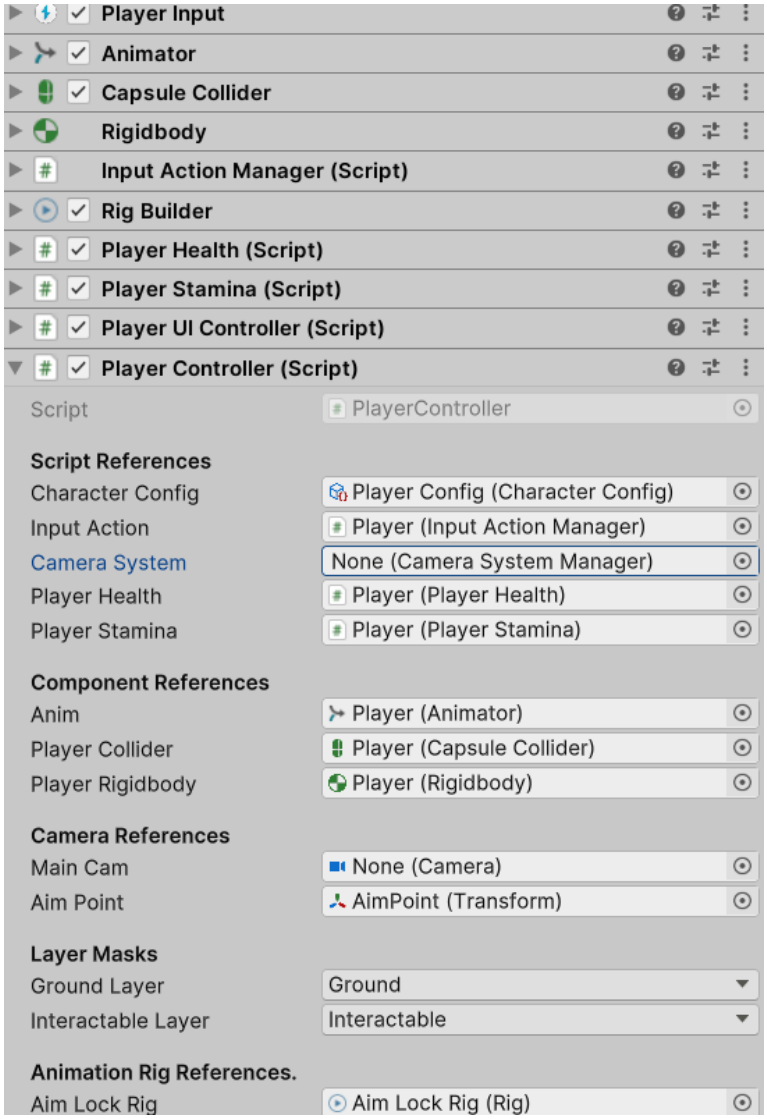
Player Stamina Setup

To set up the player stamina script, drag the stamina configuration you've created and assign it.



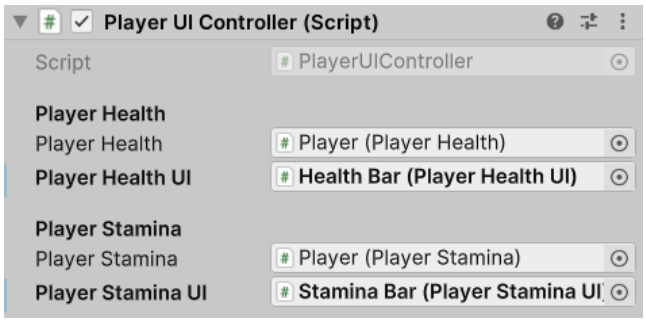
Player Controller Setup

To set up the player controller script, add player controller script to the player game object into your scene and it will add all the necessary components and scripts all at once except for the Rig builder you'll need to setup alone



Player UI Controller Setup

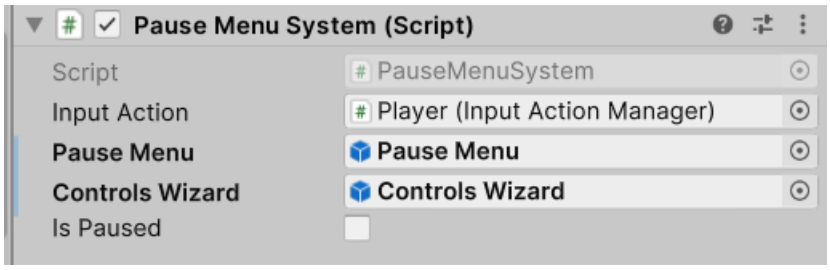
to set up the player UI controller script, assign the script components to each field, and drag the UI of each one and assign to the appropriate field.



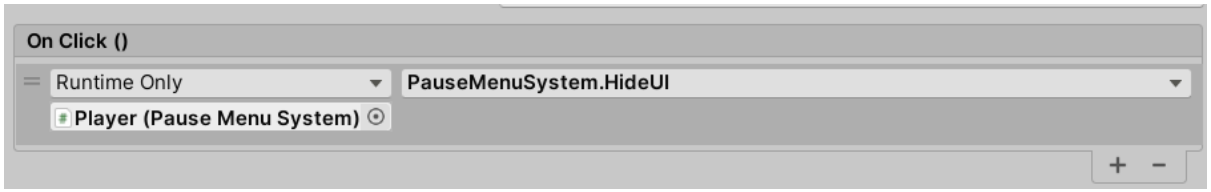
Pause Menu System Setup

To set up the pause menu system script component, follow the below instructions for each game object

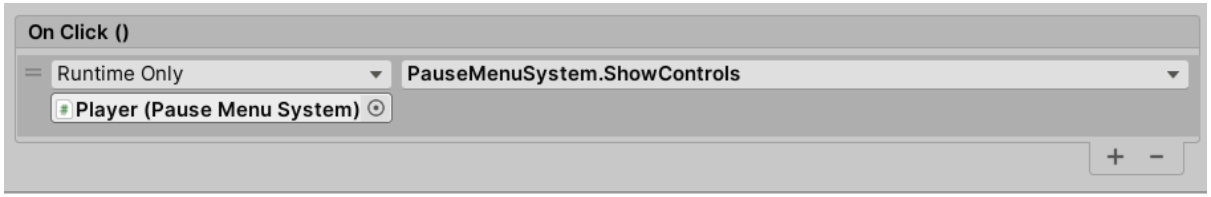
Add the Pause Menu System script to the player game object and assign the fields as shown below



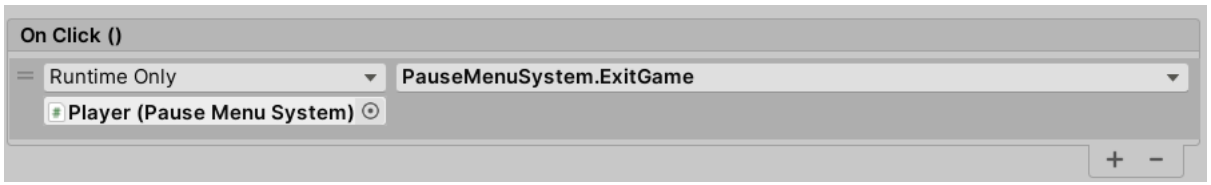
- **Resume Game:** This Game Object is associated with the button that resumes the game when clicked. When the game is paused, clicking this button will hide the pause menu and resume the game.



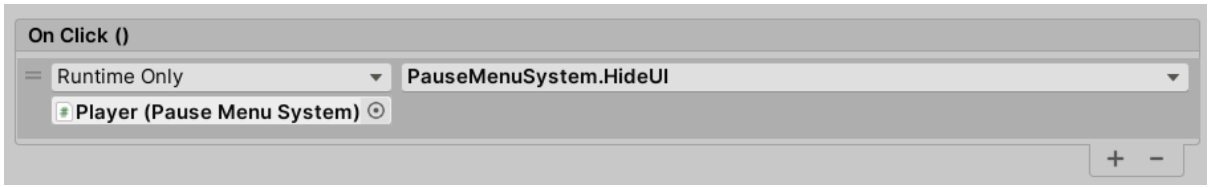
- **Controls Menu:** This Game Object is linked to the button that opens the controls wizard. When clicked, it hides the pause menu and displays the controls wizard, allowing the player to view and modify the game controls.



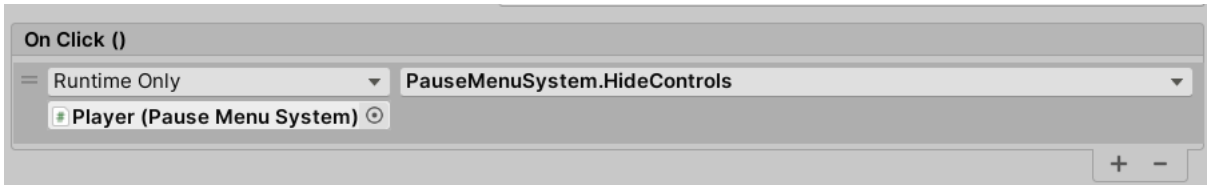
- **Exit Game:** This Game Object is tied to the button that exits the game. Clicking this button will close the game. If the game is running in the Unity editor, it will stop the play mode. If the game is running in a build, it will quit the application.



- **Close Window:** This Game Object is connected to the button that closes the currently open window. It can be used to close the pause menu or the controls wizard, returning the player to the game.

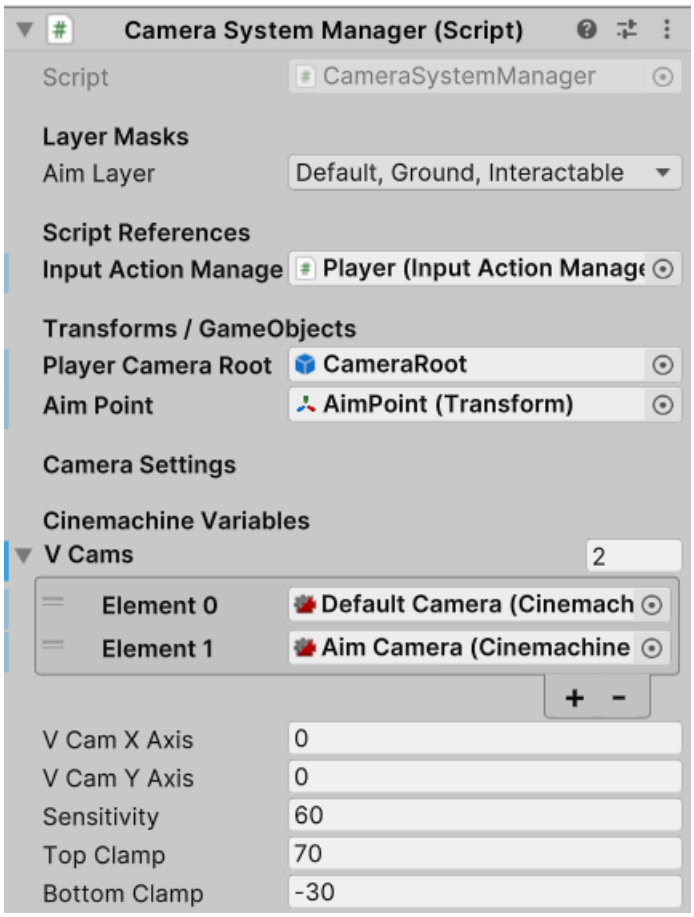


- **Controls Wizard**: This Game Object represents the controls wizard itself. It is a UI element that displays the game controls to the player. It can be shown or hidden using the ‘Show Controls’ and ‘Hide Controls’ methods, inside this game object there is a close window button.



Camera System Manager Setup

To set up the camera system manager script, assign the required fields as shown below



10. Troubleshooting

If you encounter issues, refer to the Usage Examples section in the documentation or visit our Support page for assistance.

11. FAQs

1. **Q: Can I customize player movement states or add new states?**

A: Yes, the system is designed to be extensible. You can customize existing states or add new states by modifying the provided scripts and implementing your desired functionality.

2. **Q: What should I do if my player's character animations are not playing correctly?**

A: Ensure you've unchecked the "Apply Root Motion" in the animator component, check for any warnings or errors in the console and verify the animations are properly triggered based on the player input and state transitions.

3. **Q: Can I add custom UI elements for the player's health and stamina?**

A: Yes, you can customize the player UI element by modifying the provided prefabs that handle the UI under the canvas.

4. **Q: How Can I implement additional camera systems or controls?**

A: The camera system manager script allows you to manage camera systems for the player, you can extend this functionality to add more camera systems based on your game requirements.

5. **Q: Can I use this asset with my existing AI solution?**

A: Yes, the core Health, Stamina and Interaction Base Systems are designed to be modular and can be easily integrated with your AI, providing a unified framework for character behavior.

6. **Q: What should I do if I encounter issues during setup or usage?**

A: If you encounter any issues, first refer to the "**Troubleshooting**" section in the documentation. If the problem persists, visit our Support page or contact us at support@truetacticalstudio.com for assistance.

12. Support

For technical support or inquiries, please feel free to reach out to us:

- **Email:** support@truetacticalstudio.com
- **Visit our website:** www.truetacticalstudio.com