⚠️ Use the online documentation for more complete information: https://julhiecio.gitbook.io/ju-tps-documentation/

JU TPS 3 Offline Documentation

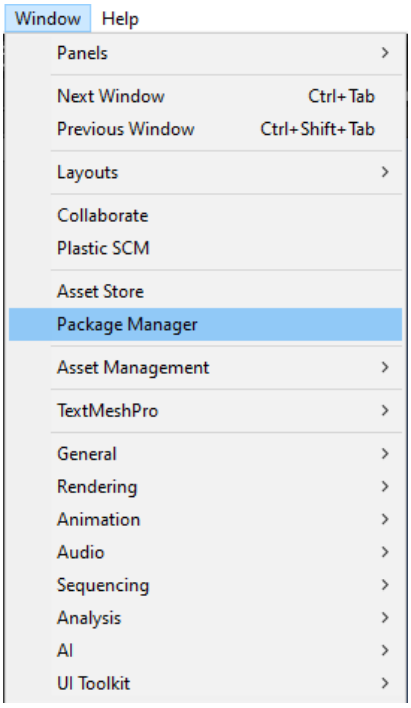
Here you will find everything you need to start your project with JU TPS in the simplest and fastest way possible

Quick Start

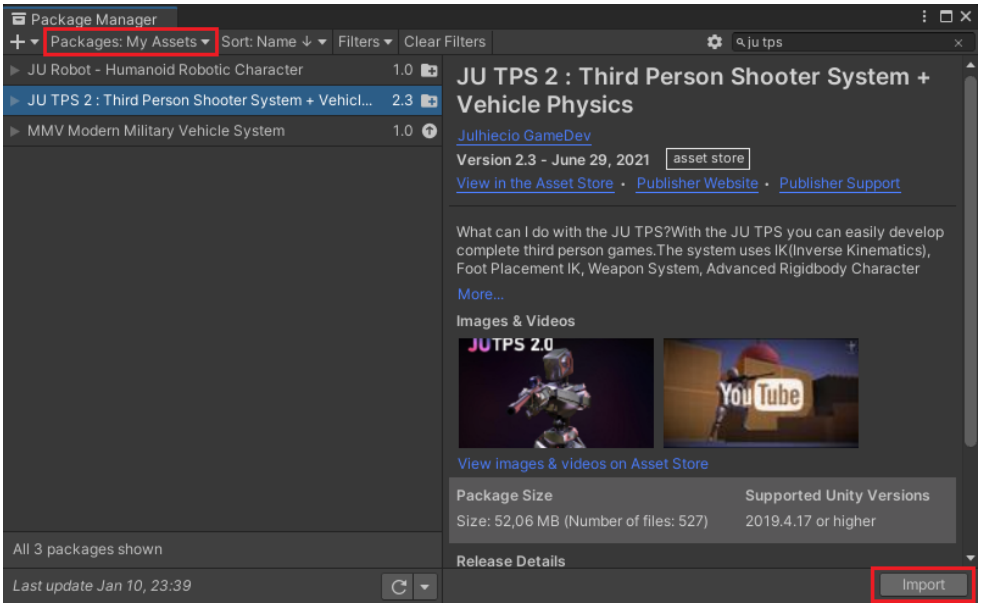
Ready to start using the JU TPS? let's start with the basics.

How to install JU TPS correctly

1 - Open the Package Manager



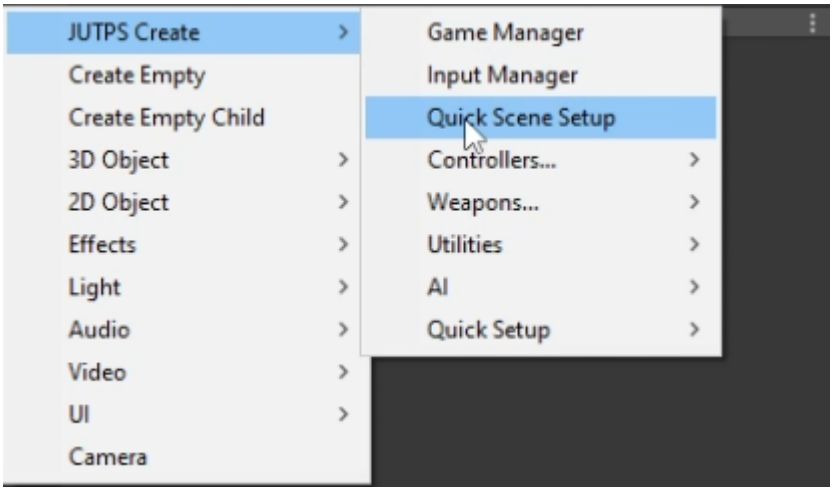
2 - Find the JU TPS Asset, download and press import



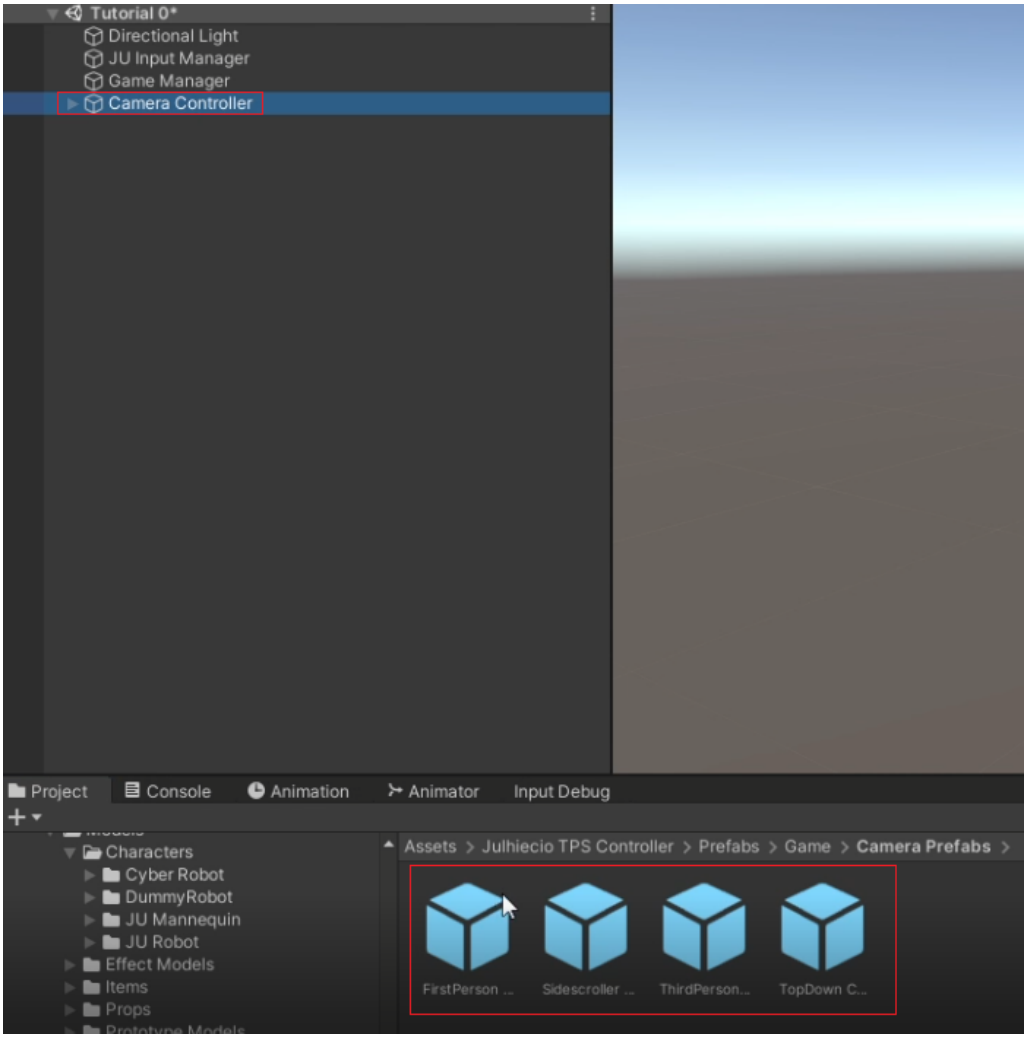
Note that JU TPS needs you to import the Project Settings otherwise it won't work as expected. If you are importing into a non-empty project make a backup of your Project Settings.

How to start a new scene

1 - Create a new scene
2 - Delete the default Main Camera
3 - Click in Create > JU TPS Create > Quick Scene Setup



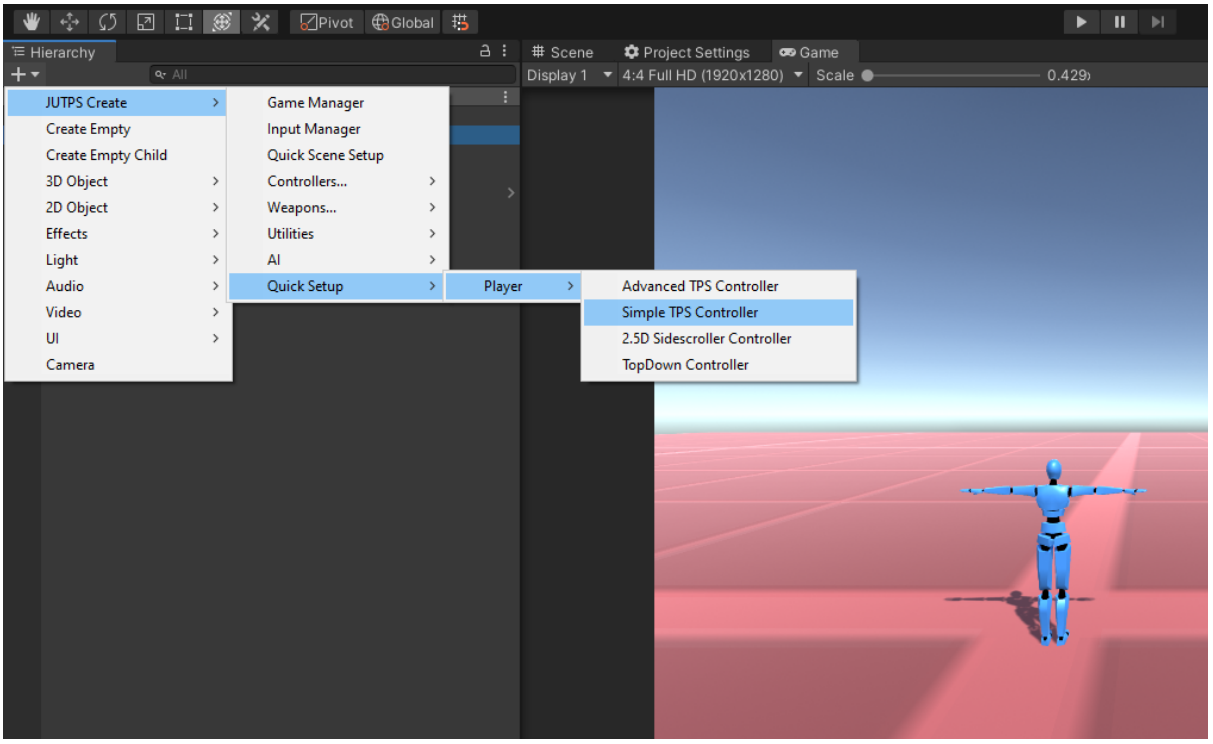
This will create in your scene a JU Input Manager, a Game Manager and a Camera Controller.
You should replace this default camera with camera prefabs according to your game style



How to create a JU Character

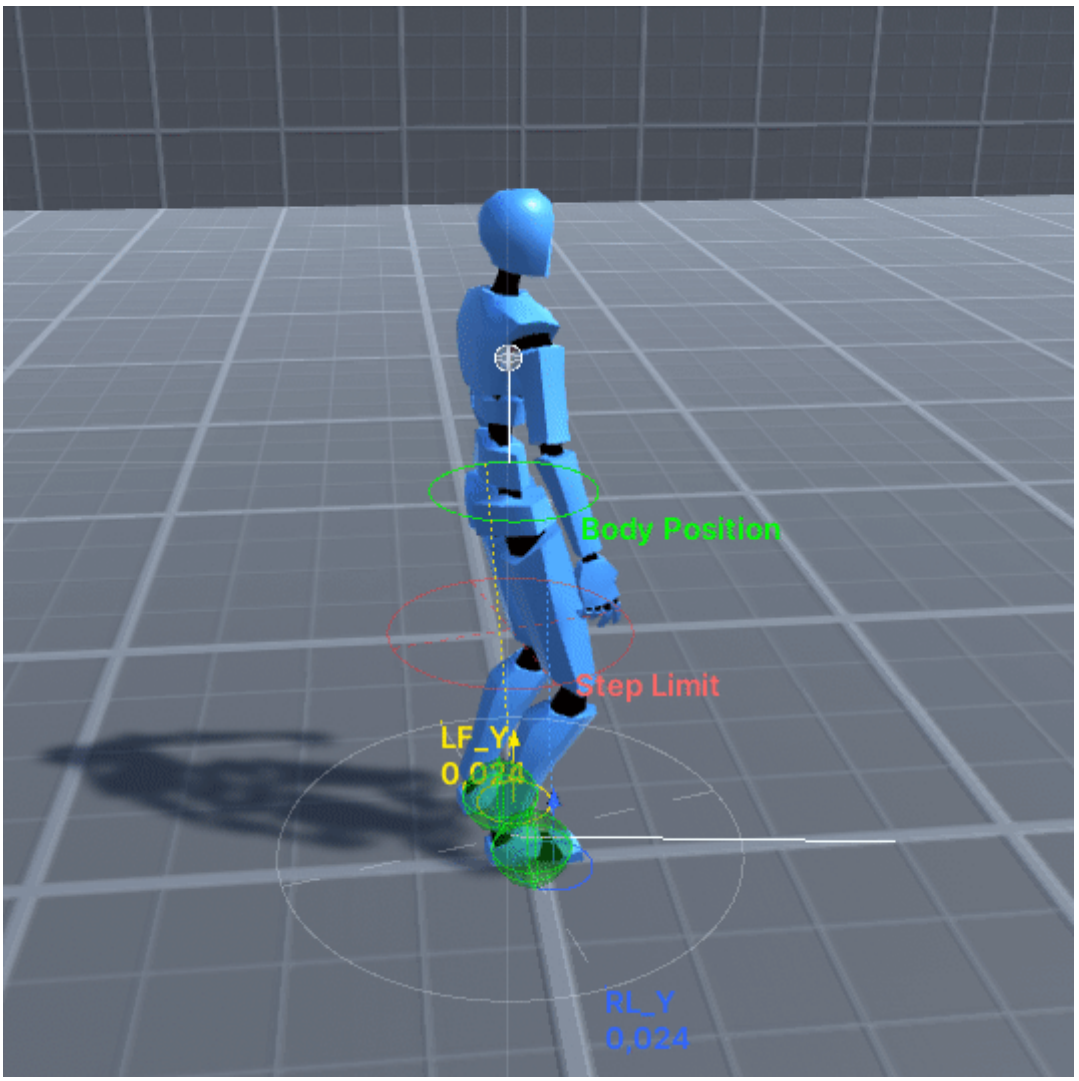
Creating a JU Character

1 - Add the Humanoid Character model in scene
2 - Click on Create > JU TPS Create > Quick Setup > Player



3 - Choose the best option for your game

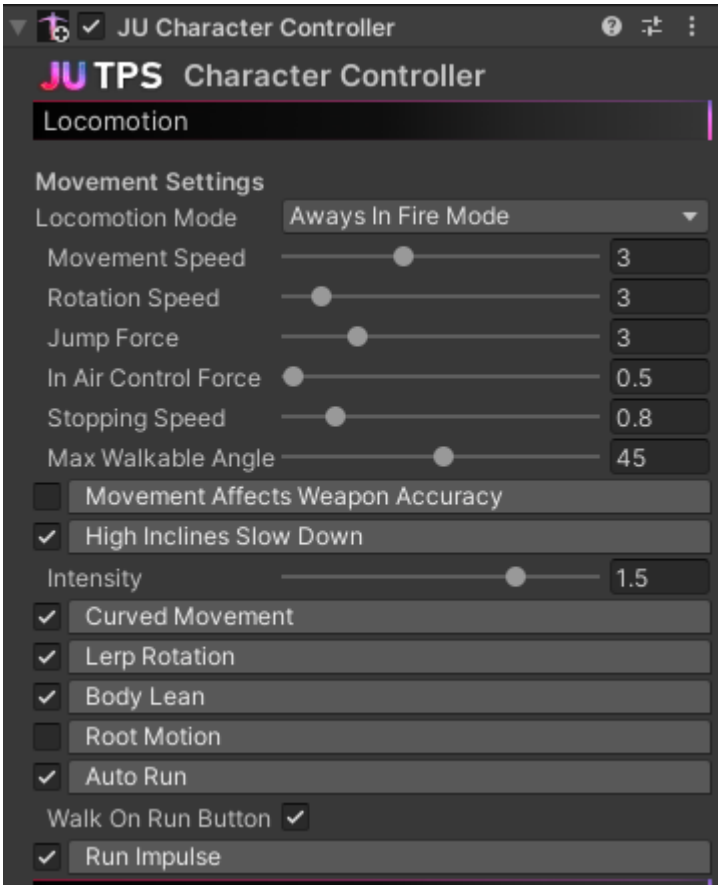
Controller Setup Option	Description
Advanced TPS	Add complementary components: Foot Placement, Footstep, Body Lean, Drive Vehicles, Procedural Driving Animation, Ragdoll Controller
Simple TPS	Simple setup, only JU Controller and Footstep
2.5D Sidescroller	2.5D Sidescroller Game setup (it is necessary to use a Sidescroller Camera to work normally)
TopDown	TopDown games setup (it is necessary to use a Top Down Camera)



You now have a functional JU Character 😎👍
Note that you can add or remove components in any setup and configure it according to your project needs.

Configuring the JU Character Controller

Locomotion Settings



Locomotion Mode has three types of locomotion:
Free
Aways In Fire Mode
JU TPS Classic
Default Locomotion, beautiful.

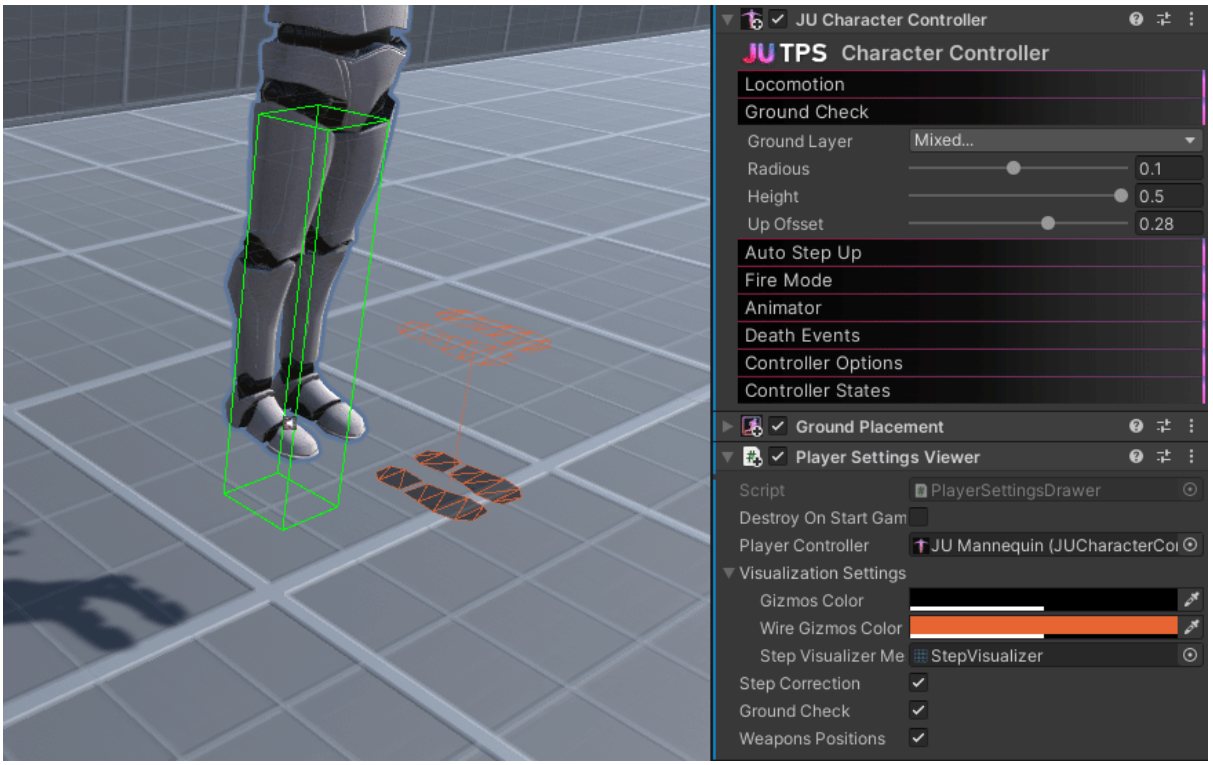
Locomotion Options	Description
Air Control Force	Strength that the character can move in the air
Stopping Speed	Locomotion stopping speed
Max Walkable Angle	Angles greater than this will cause the character slide. ⚠ Leave it at 0 if you use Gravity Switch system
Movement Affects Weapon Accuracy	On moving affects weapon accuracy

High Inclines Slow Down	High inclines slow down speed ⚠ Leave it off if you use Gravity Switch system
Curved Movement	Moves the character according to the current direction and not the desired direction ⚠ Leave it off if you use Gravity Switch system
Lerp Rotation	Smooth rotation
Body Lean	The body leans in curves
Root Motion	Animation controls movement ⚠ Leave it off if you use Gravity Switch system
Auto Run	Automatically run
Run Impulse	Sprinting on start moving

Ground Check

Ground Check Options	Description
Ground Layer	Layers that Ground Checker will detect as ground

The other options are basically this:



Auto Step Up

Auto Step Up is a system that makes the JU Character Controller go up stairs, similar to the "Step Offset" of the default Character Controller, but with more options.

Auto Step Up Options	Description
Step Correction	Enable Step Up System
Up Step Speed	Movement speed
Step Correction Layer	Allowed layers to do step up move
Step Raycast Distance	Length of sensor ray from ground
Forward Offset	Forward sensor ray position adjustment
Step Height	Minimum step height to step up

It is not recommended to leave the Forward Offset value less than the Radius of the Capsule Collider or the Auto Step Up may not work correctly.

Fire Mode

Fire Mode is the way the character is prepared to attack, the camera is closer, the character looks forward and wields an item.

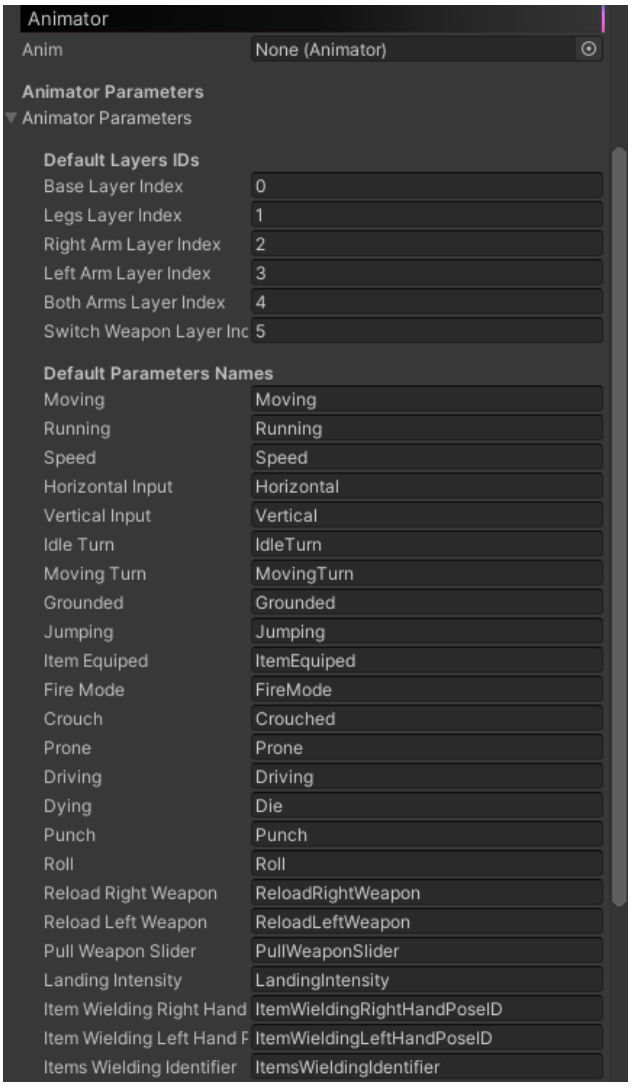
Fire Mode Options	Description
Item Aim Rotation Center	It is Fire Mode's item rotation pivot and also handles wielding positions.
Upper Chest Bone	Upper Chest Bone
Aim Mode	This variable will define if you will have to hold the aim button or just press it once to aim

FireMode Max Time	The time it will stay in Fire Mode without interacting, but this option is only useful in Free Locomotion mode

The "Upper Chest Bone" is assigned automatically when the game starts, if the system cannot find the last bone in your character's spine, assign it manually.

Animator

If you want to create a custom animator or just edit the names of the Animator Controller parameters, this is where you'll edit.



Death Events

Death Event Options	Description
Enable Ragdoll When Die	Enable ragdoll physics when character dies, click here to learn how to setup Ragdoll

Controller Options

These options are very important to define the JU Character

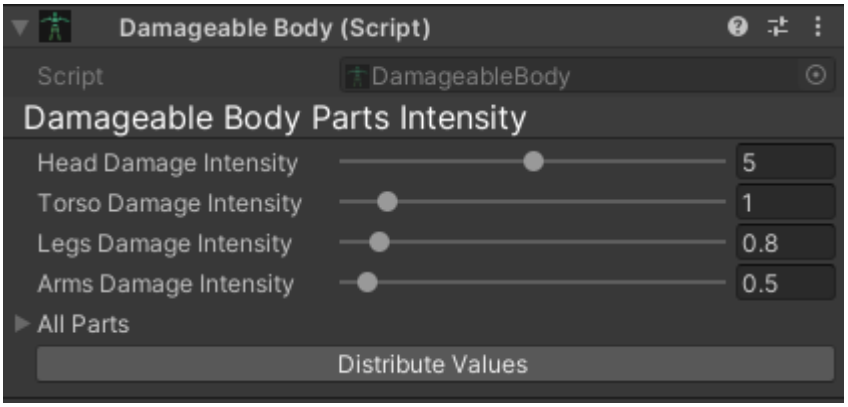
Controller Options	Description
Block Vertical Input	Block Vertical Input from locomotion, useful for sidescroller game type
Block Horizontal Input	Block Horizontal Input from locomotion
Block FireMode On Cursor Visible	Block FireMode when mouse cursor is visible, useful to access menu/inventory and not shoot accidentally, on mobile it makes no difference.
Block FireMode On Punching	Block FireMode when character is punching ⚠ Leave this false if you are making a FPS Game
Enable Punch Attacks	Enable punch attacks
Enable Roll	Enable roll
Is AI	If it's on, the default Inputs won't be used, so you can program an AI/NPC the way you want, or just use the built-in AIs.

Damageable Body

In order for your JU Character to receive damage in the bone colliders(and not in the sphere around it), it is interesting to use the Damageable Body script.

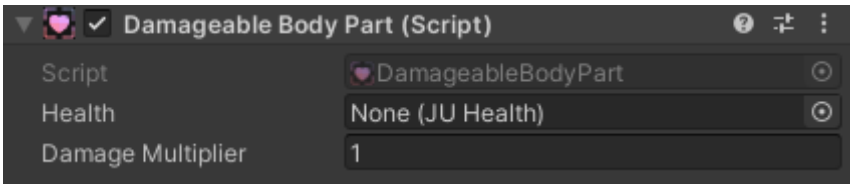
Damageable Body Script

This script automatically adds Damageable Body Parts to a JU Character's bone colliders, or changes the values of existing Damageable Body Parts.



Damageable Body Part

The name can be confusing, but they are different scripts, this script is in the bone collider of the JU Character that will receive the damage, it can increase or decrease the damage, great for a HeadShot. The value of this script is changed to the armor protection system.

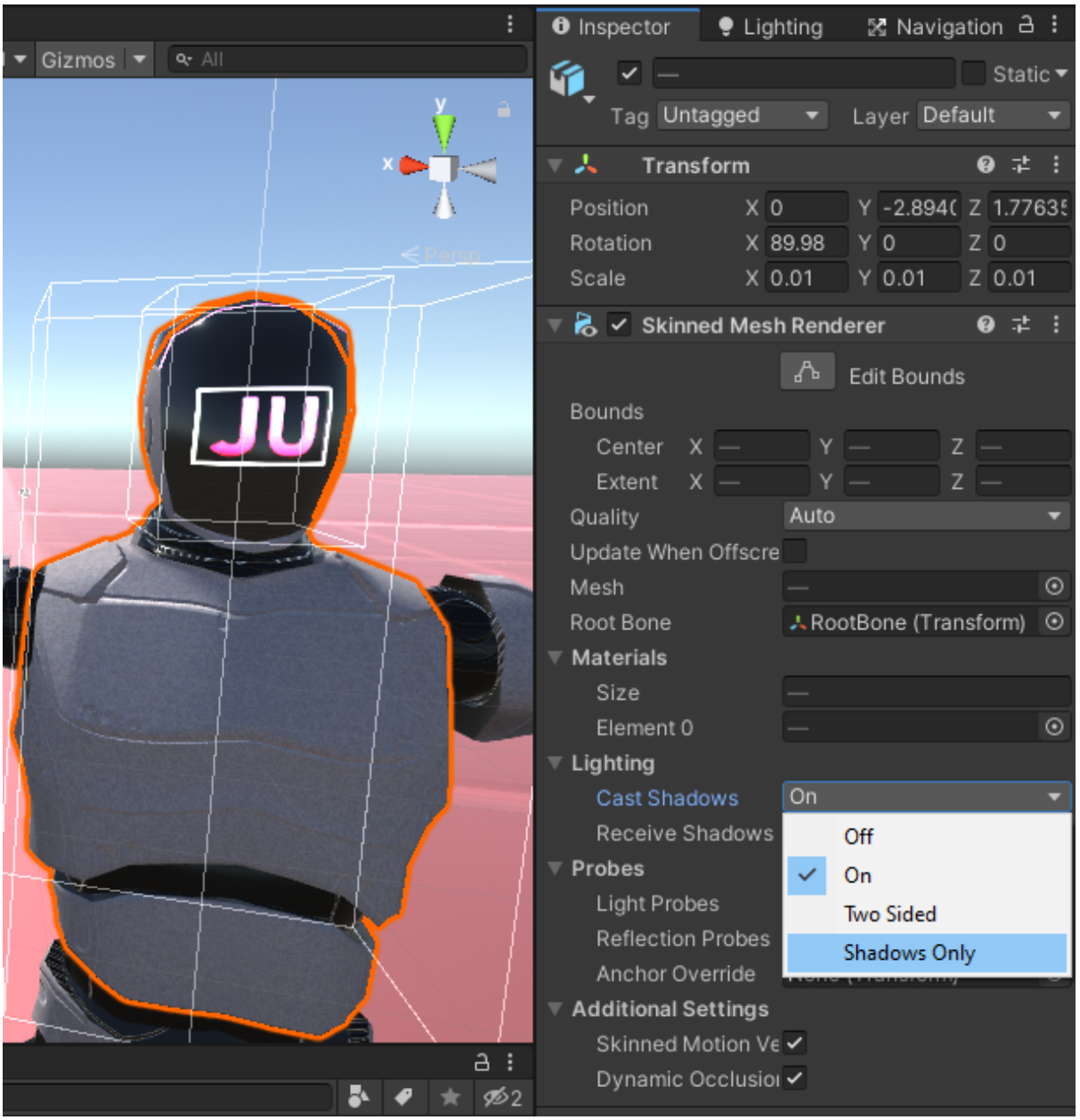


First Person Shooter Style

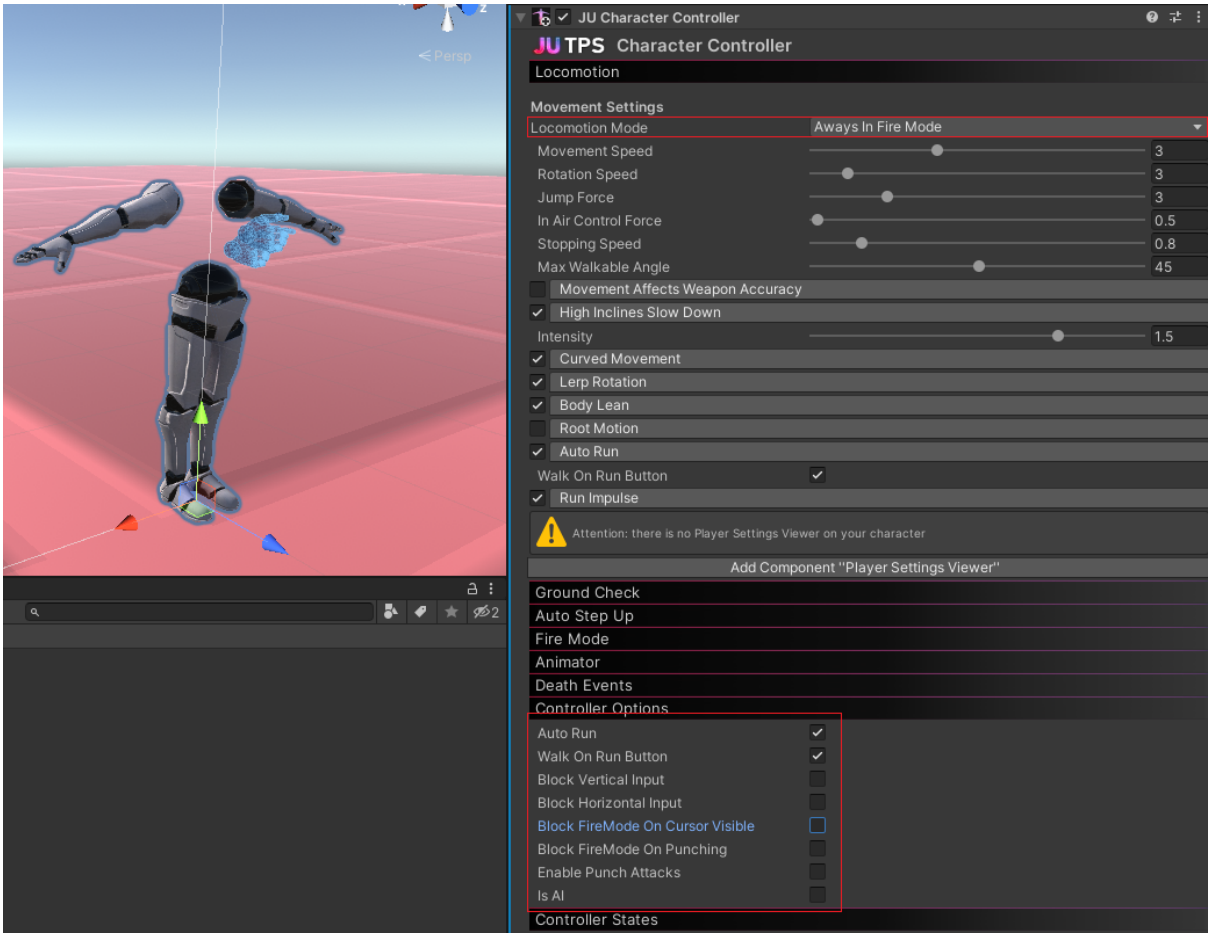
This style uses a Camera FPS and the Player is always locked towards the camera.

How to create a FPS Game?

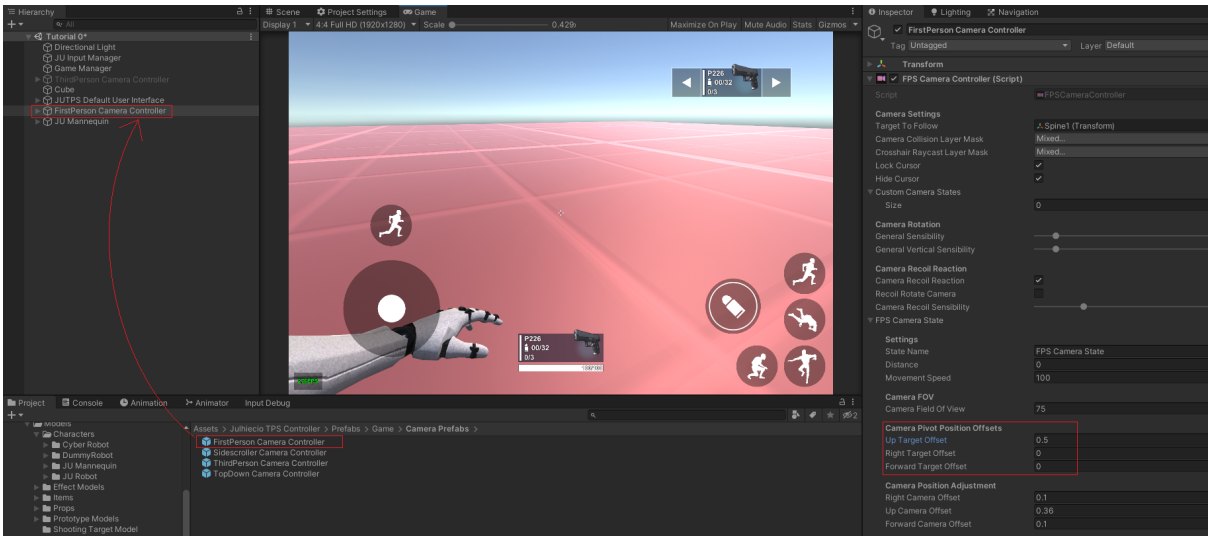
To create this style of controller, you can make a simple or advanced setup on the character and follow the steps below:
1 - You need a character with the torso separated from the arms, you need to check the "Shadows Only" option in the Skinned Mesh Renderer of these parts of the character's body



2 - After doing the character setup you should configure the JU Character Controller like this:

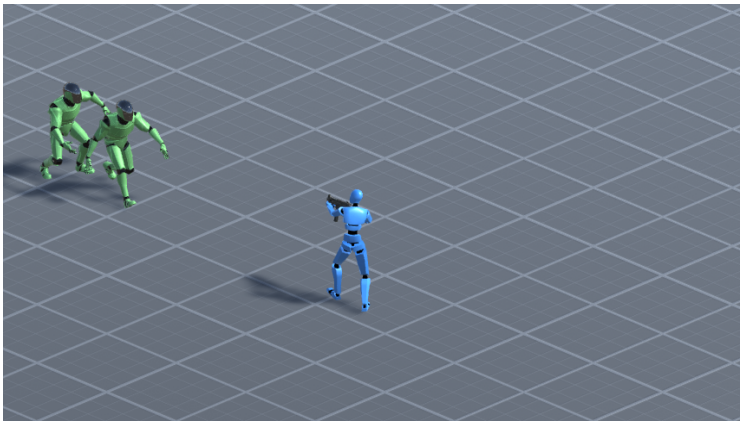


3 - Add the FPS Camera prefab to the scene, adjust the camera position settings and you have a working FPS Controller



Top Down Style

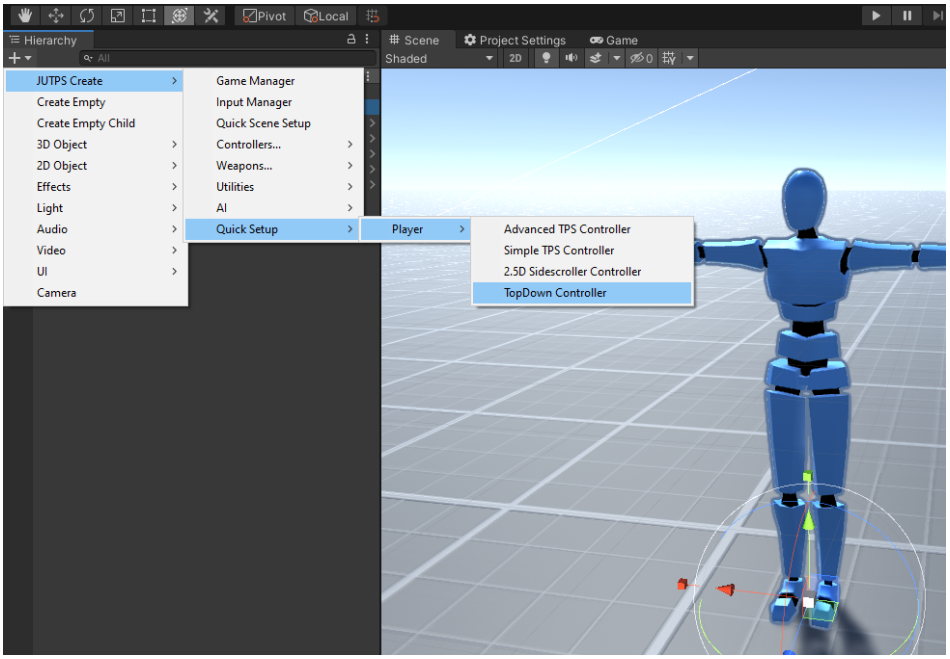
This style of play is characterized by the position and angle of the camera, which is positioned at the top of the scenario.



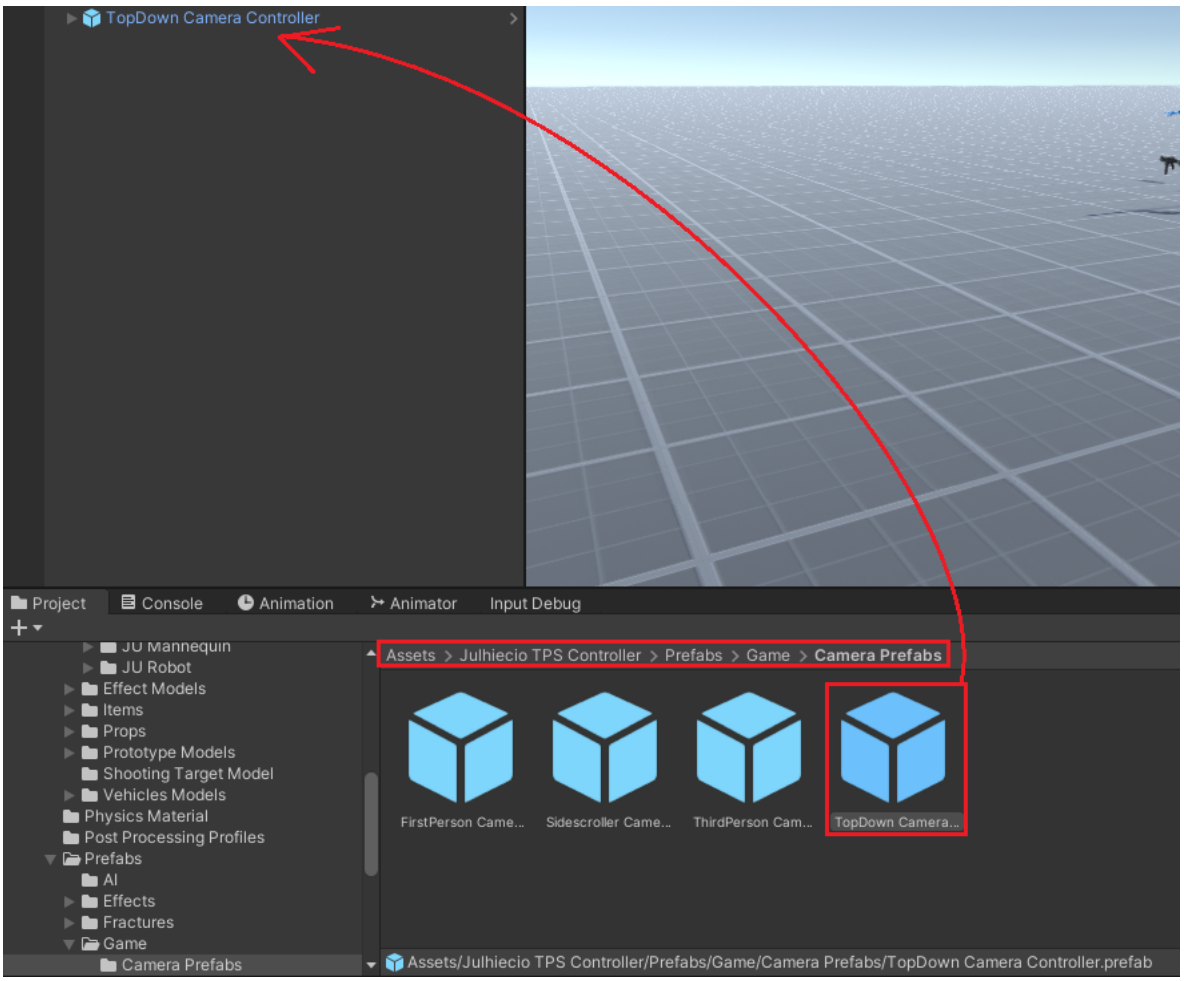
Top Down Example

How to create a Top Down Game?

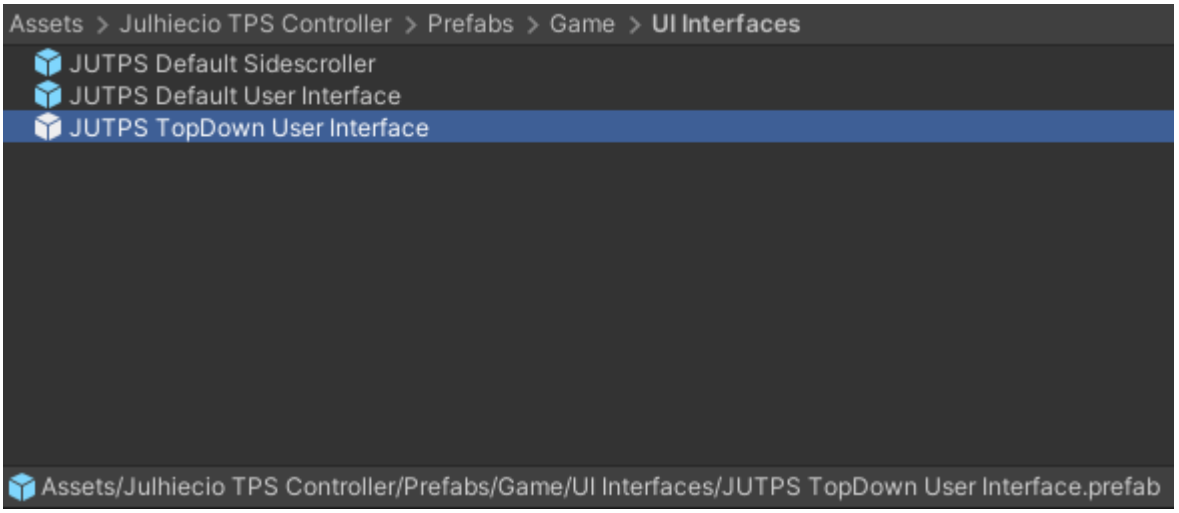
1 - Do Top Down JU Character Setup



2 - Place the camera prefab "TopDown Camera Controller" in the scene



3 - Place the "JUTPS TopDown User Interface" prefab in the scene.



Now you have a Top Down base game

Cross-Plataform Info

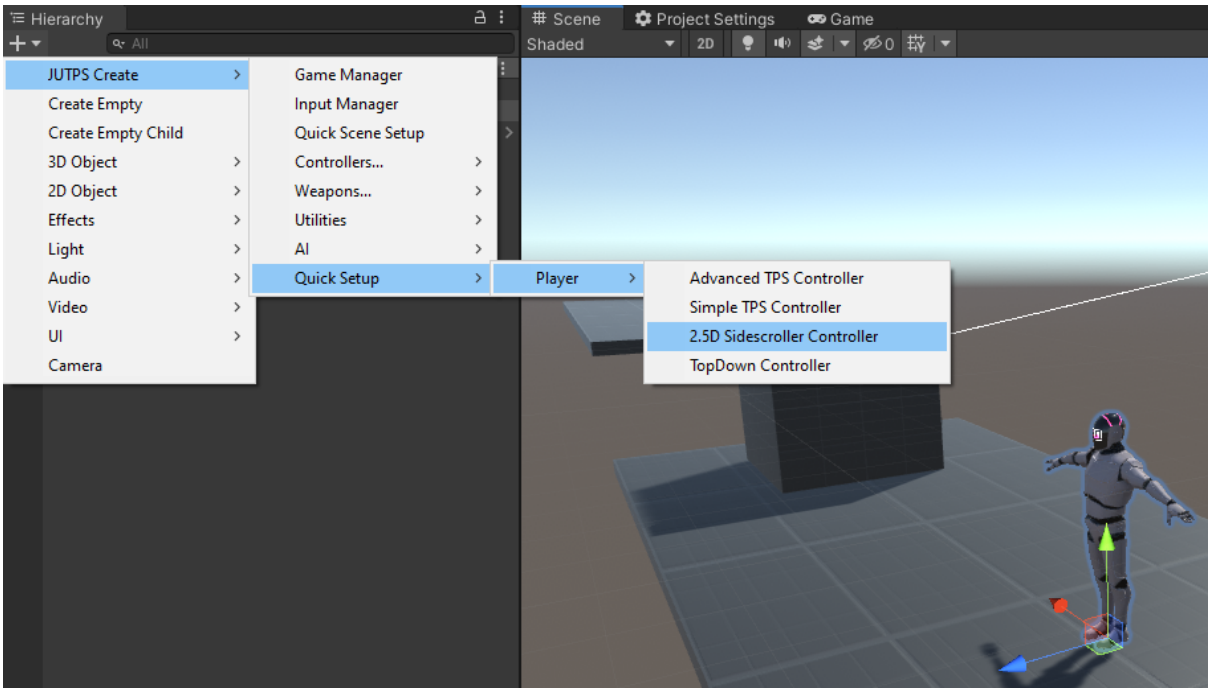
This style by default will only work on Desktop, to adapt to Gamepad and/or Mobile go to the page "Important Info About Cross Plataform

2.5D Sidescroller Style

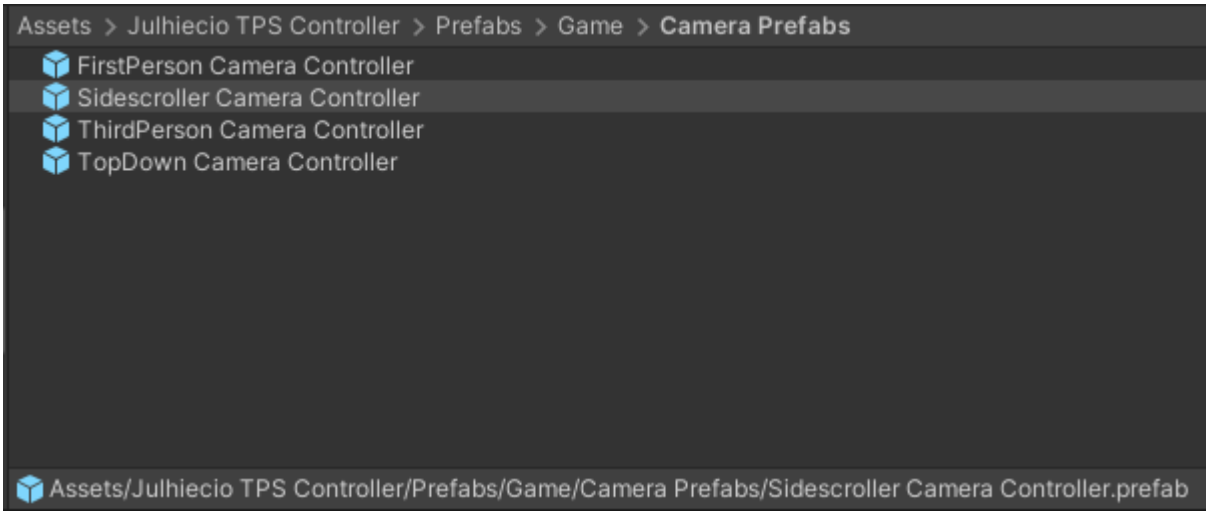
The game style is characterized by the lateral view and two-dimensional control, like a 2D game but with 3D graphics.

How to create a Sidescroller game

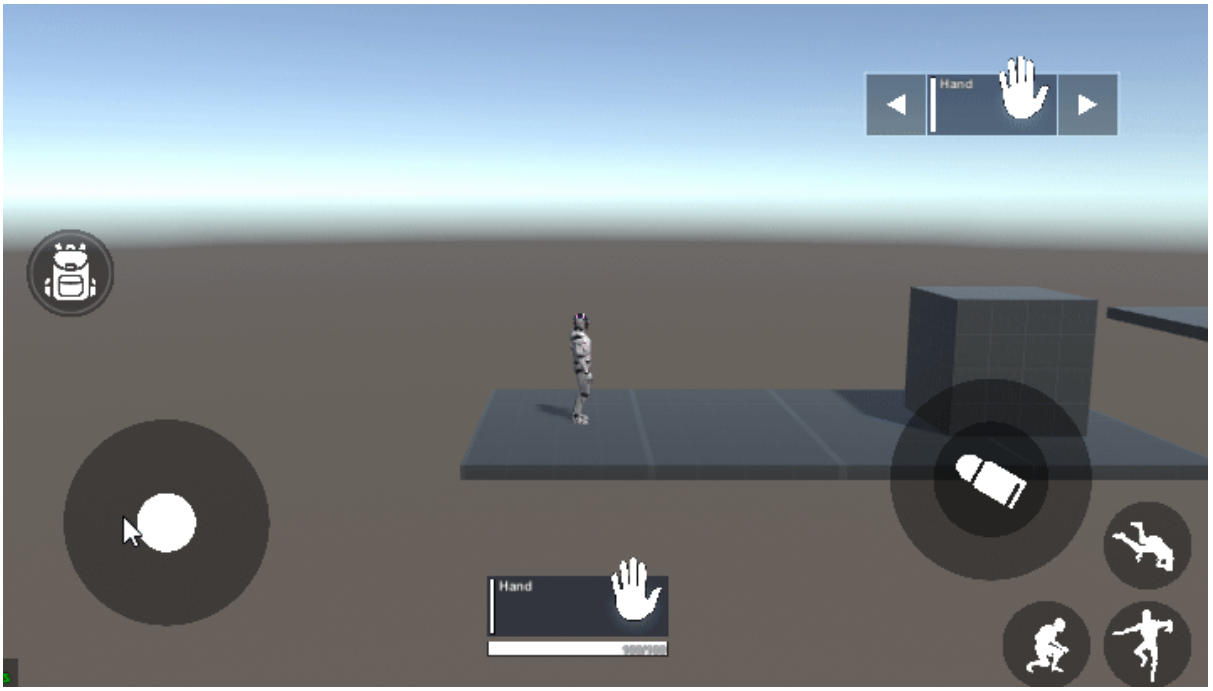
1 - Place character model in scene and make JU Character setup for Sidescroller Style



2 - Add "Sidescroller Camera Controller" prefab in scene



3 - Add "JUTPS Sidescroller User Interface" and its done.



Sidescroller Example with Mobile Inputs

Cross-Plataform Info

This style by default will only work on Desktop, to adapt to Gamepad and/or Mobile go to the page "Important Info About Cross Plataform"

⚠️Important Info About Cross-Plataform

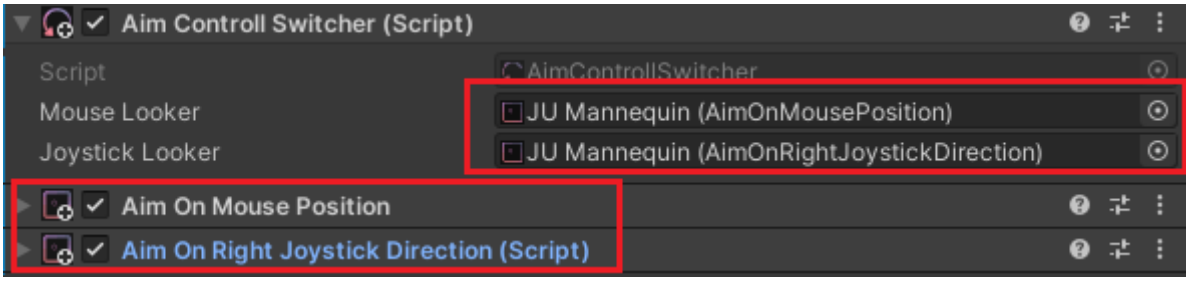
The way the system controls the player's aim is different according to the platform, TopDown and Sidescroller styles that don't use the camera to orient the aim system, so you have to adapt:

Non-TPS Default Setup

The default setup will come with this component, which only works for Desktop with a mouse.

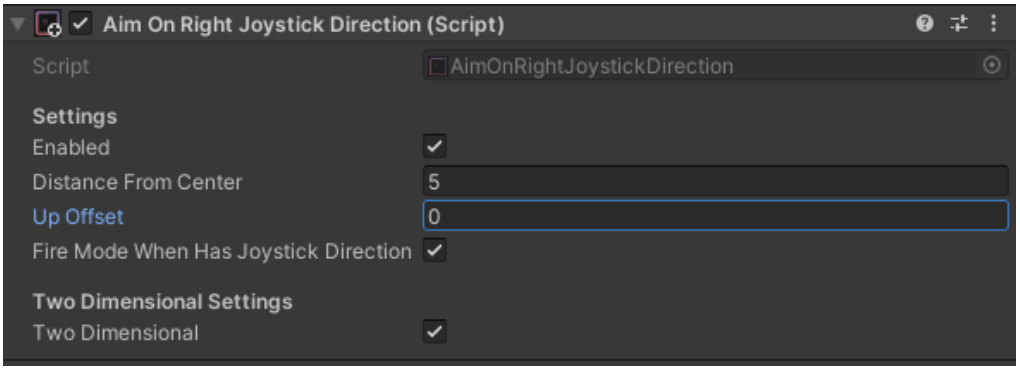
Joystick(GamePad) and Mouse

To use Mouse and Gamapad at the same time you will have to add Aim Controll Switcher, Aim On Mouse Position and Aim on Right Joystick Direction components.



Mobile

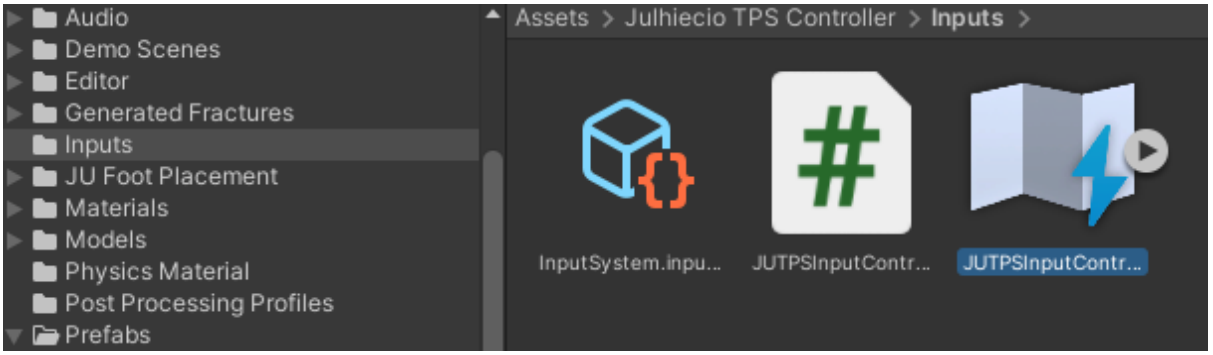
To use aiming correctly on mobile you should just use the Aim On Right Joystick Direction component and configure it.



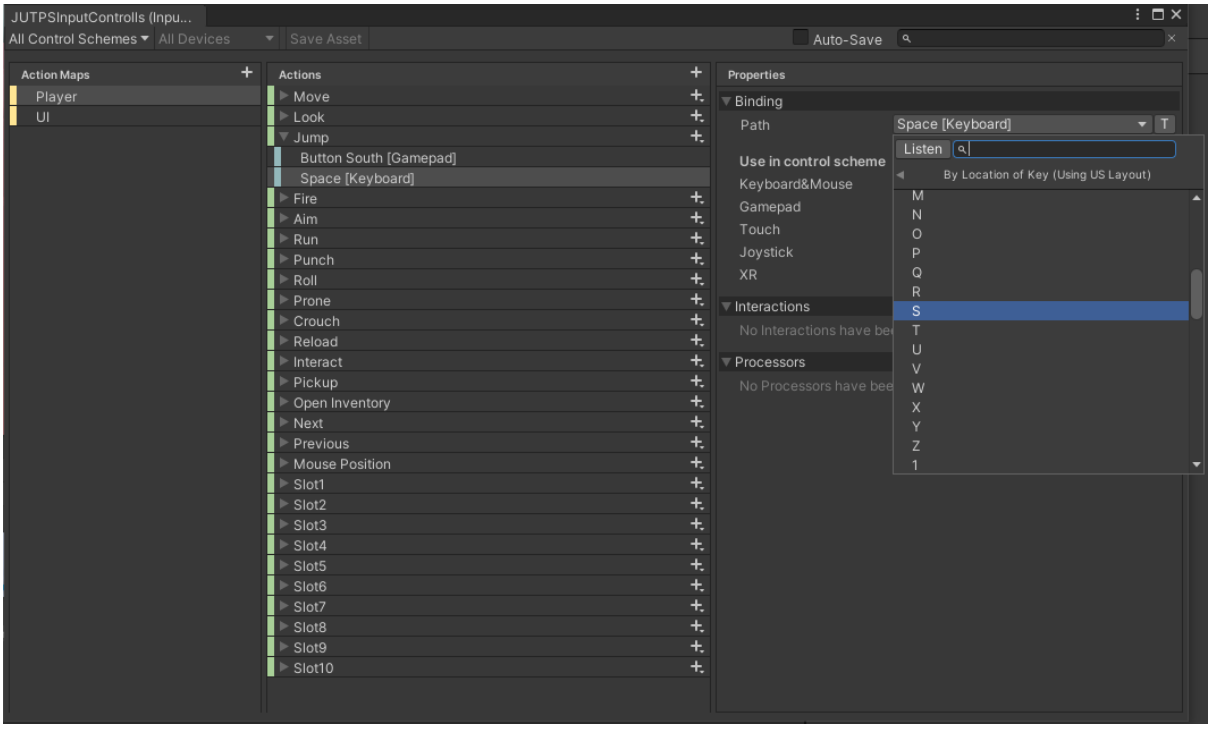
Controls / Inputs

New Input System

To change the control in the new input system you will have to go to the Inputs folder, double click on the file "JUTPSInputControlls"

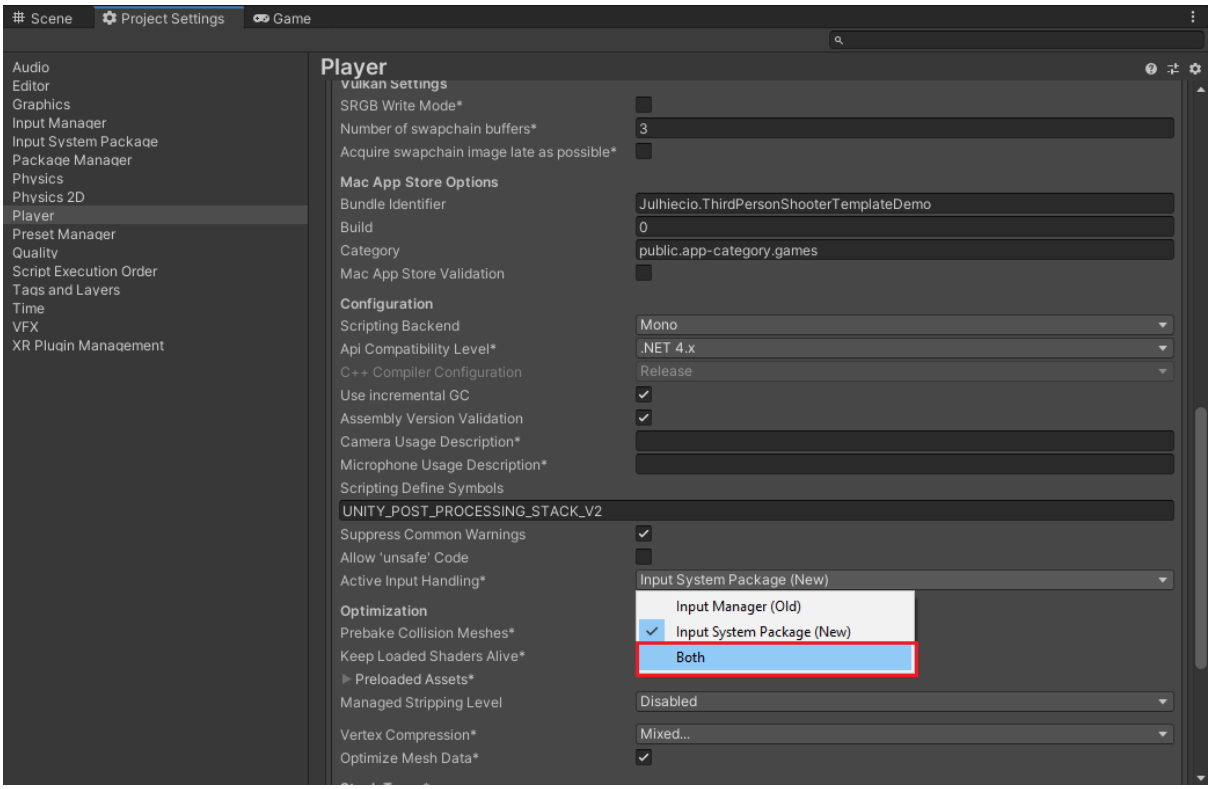


The window below will appear, and here you can edit the controls as you like.



Old Input System

The JU TPS currently works completely on the new Input System and the old one is disabled in the project settings, but if you want to use both, it's possible. Click on Edit>Project Settings>Player, and in "Active Input Handling" mark it as Both.



Now to edit the inputs of the Old Input System

JU Input Manager

The JU Input Manager helps with input handling and cross-platform operation.

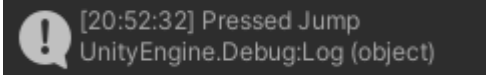
Simple Input Detection

This page will be useful when you want to expand the system, add actions or just detect inputs.
Using the JU Input Manager to detect inputs is simple, inspired by the old input system:

```
1
using UnityEngine;
2


3
// 1 Step -> Import JUInputSystem
4
using JUInputSystem;
5


6
public class JUInputHandlingTest : MonoBehaviour
7
{
8
8  void Update()
9
   {
10
       // 2 Step -> Check if Jump Button(Space Default) was pressed
11
       if (JUInput.GetButtonDown(JUInput.Buttons.JumpButton))
12
       {
13
           Debug.Log("Pressed Jump");
14
       }
15
   }
16
}
```



<- With this code this message should appear in the console
In line 4 the JUInputSystem is imported, in line 11 the check is made if the jump button was pressed, and in line 13 it is an example of action that can be replaced by any other code.

Functions and Description

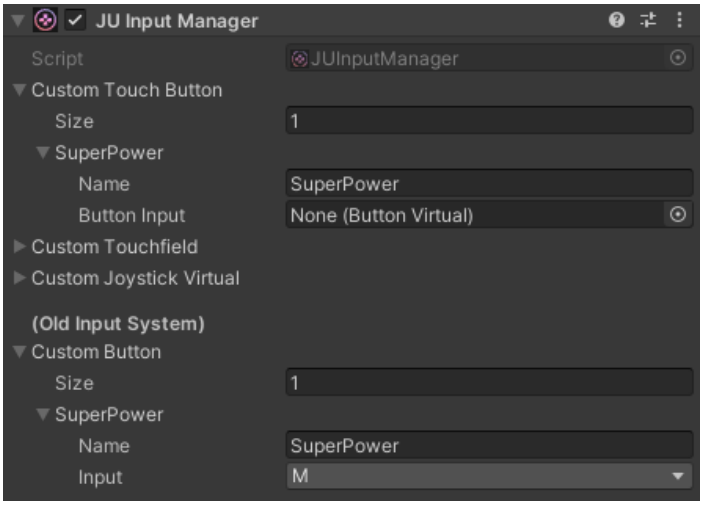
Function	Summary
float GetAxis(Axis axis)	Returns the value of the deafult axis
bool GetButtonDown(Buttons Button)	Returns the value of the deafult buttons when pressed down
bool GetButton(Buttons Button)	Returns the value of the deafult buttons when pressed
GetButtonUp(Buttons Button)	Returns the value of the deafult buttons when pressed up
bool GetCustomButton(string CustomButtonName)	Returns the value of the custom buttons when pressed
bool GetCustomButtonDown(string CustomButtonName)	Returns the value of the custom buttons when pressed down
bool GetCustomButtonUp(string CustomButtonName)	Returns the value of the custom buttons when pressed up
bool GetCustomTouchButton(string CustomButtonName)	Returns the value of the custom touch buttons when pressed
bool GetCustomTouchButtonDown(string CustomButtonName)	Returns the value of the custom touch buttons when pressed down
bool GetCustomTouchButtonUp(string CustomButtonName)	Returns the value of the custom touch buttons when pressed up
Vector2 GetCustomTouchfieldAxis(string CustomTouchfield)	Returns the value of the custom Touchfield
Vector2 GetCustomVirtualJoystickAxis(string CustomJoystickName)	Returns the value of the custom Joystick
Vector2 GetMousePosition()	Return current mouse position
int GetTouchsLengh()	Returns the number of touches on the screen
InputSystem.Controls.TouchControl[] GetTouches()	

Enums	Content
Axis	MoveHorizontal, MoveVertical, RotateHorizontal, RotateVertical
Buttons	ShotButton, AimingButton, JumpButton, RunButton, PunchButton, RollButton, CrouchButton, ProneButton, ReloadButton, PickupButton, EnterVehicleButton, PreviousWeaponButton, NextWeaponButton, OpenInventory

The Enums above are all default JU TPS Inputs that you can easily access with the above functions.
Example: float moveHorizontal = JUInput.GetAxis(JUInput.Axis.MoveHorizontal);

Custom Inputs

In the JU Input Manager you can create custom inputs and call it normally, first create a Custom Input:



Note that the "Custom Button" only works in the Old Input System, if you want to check inputs in the new input system you can use Input Events Components
And you can call that input like this:

```
//Check if Custom Button "SuperPower" was pressed
if (JUInput.GetCustomButtonDown("SuperPower") ||
JUInput.GetCustomTouchButtonDown("SuperPower"))
{
  Debug.Log("Pressed M");
}
```

See below all Custom Input Functions:

Functions	Summary
bool GetCustomButton(string CustomButtonName)	Returns the value of the custom Input Button when pressed
bool GetCustomButtonDown(string CustomButtonName)	Returns the value of the custom Input Button when pressed Down
bool GetCustomButtonUp(string CustomButtonName)	Returns the value of the custom Input Button when pressed up
GetCustomTouchButton(string CustomButtonName)	Returns the value of the custom Virtual Button when pressed
GetCustomTouchButtonDown(string CustomButtonName)	Returns the value of the custom Virtual Button when pressed down
bool GetCustomTouchButtonUp(string CustomButtonName)	Returns the value of the custom Virtual Button when pressed up
Vector2 GetCustomTouchfieldAxis(string CustomTouchfield)	Returns the value of the custom Touchfield
Vector2 GetCustomVirtualJoystickAxis(string CustomJoystickName)	Returns the value of the custom Joystick

Rewrite Functions

These functions rewrite the Inputs, it is used by Mobile Rig to make the virtual buttons work.

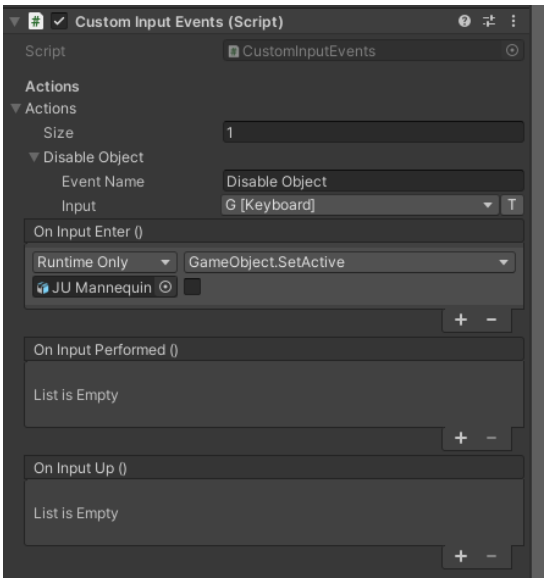
Functions	Summary
RewriteInputAxis(Axis axis, float AxisValue)	Rewrite the value of a default JU input axis
RewriteInputButtonPressed(Buttons button, bool ButtonValue)	Rewrite the value of a default JU input button
RewriteInputButtonPressedDown(Buttons button, bool ButtonValue)	Rewrite the value of a default JU input button
RewriteInputButtonPressedUp(Buttons button, bool ButtonValue)	Rewrite the value of a default JU input button

Input Events

JU TPS has scripts that help detect Inputs in a simple way

Custom Input Event

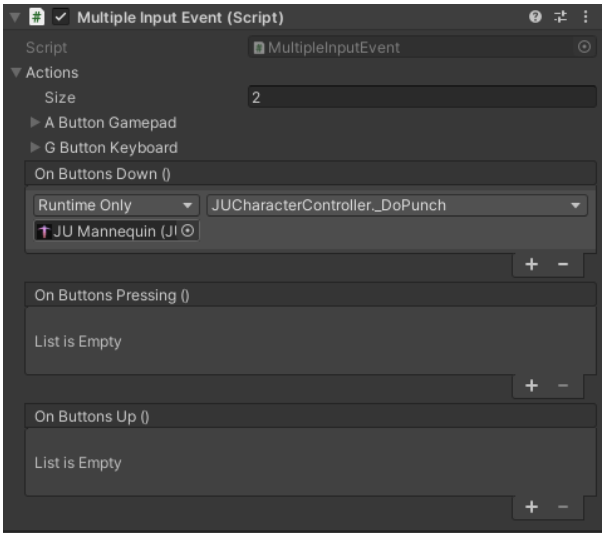
This component allows you to check an Input and perform simple actions like deactivating an object or calling functions from code.



Example: Custom Input Event will disable a Gameobject when pressing [G]

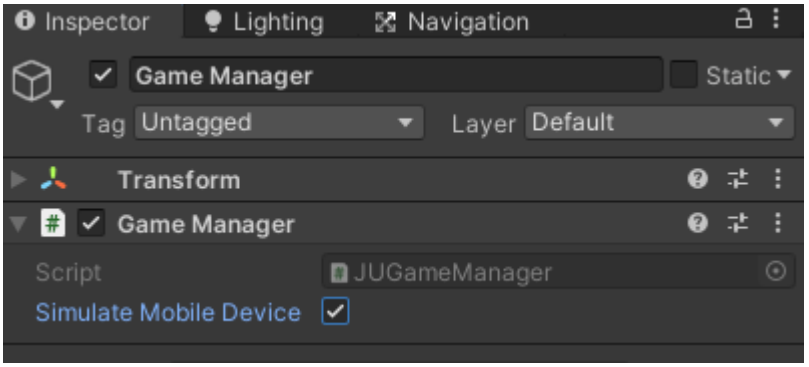
Multiple Input Event

This component is the same as the previous component, with the difference that it supports Multiple Inputs, perfect for a cross-platform action.



How to create a Mobile Game with JU TPS

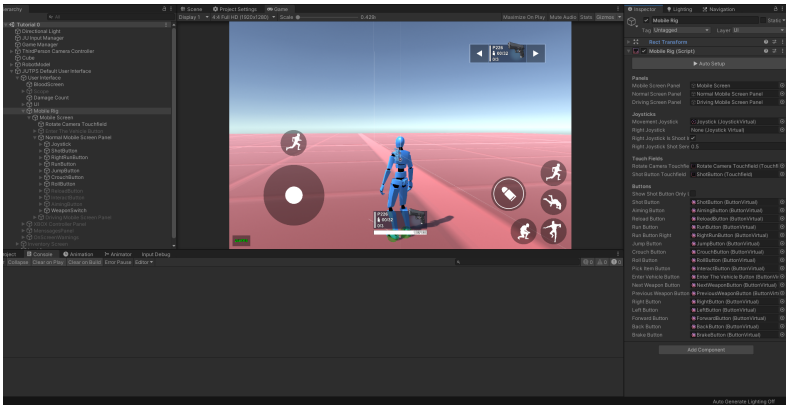
It is simple to create a mobile game in JU TPS, you only need to use the standard interface with the Mobile Rig already configured and in the Game Manager mark the toggle "Simulate Mobile Device"



Mobile controls will be enabled anyway if the app is running on a handheld device

Mobile Inputs

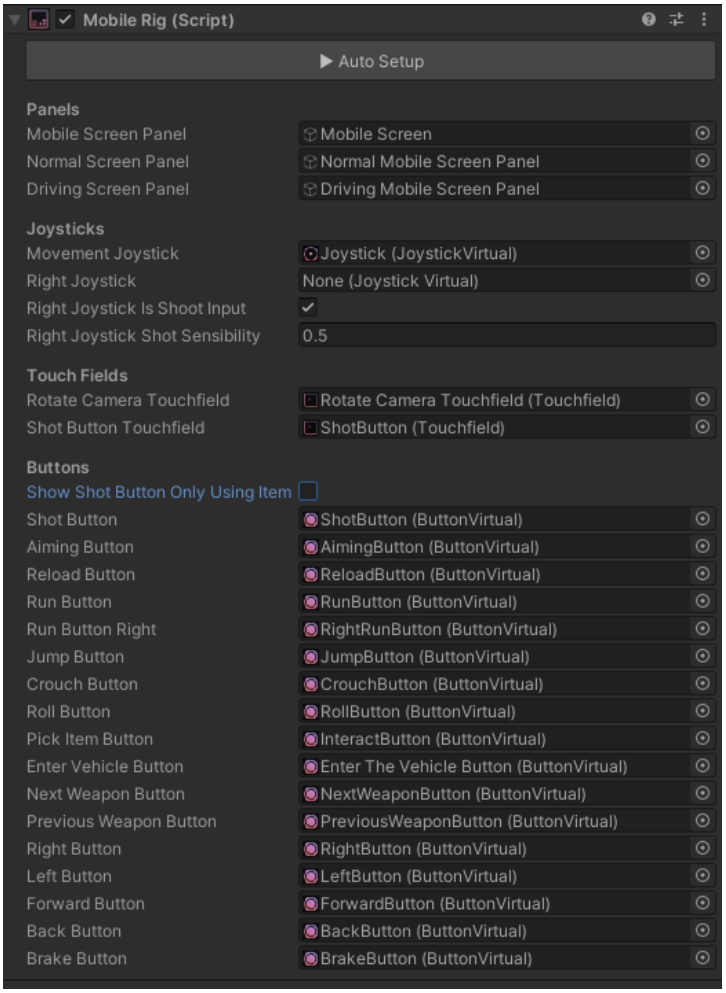
The Mobile Inputs are a set of Virtual Button, Touchfield, Virtual Joystick and mainly the Mobile Rig, which rewrites inputs of the JU Input Manager



Mobile Rig Component in JU TPS Default User Interface

The Mobile Rig

The Mobile Rig is this script that rewrites the inputs.
Auto Setup Button finds all virtual buttons, joysticks and touchfields by name. You don't need to press it unless you're creating an own Mobile UI and want Mobile Rig to find the buttons, but it will only find them if the components are named exactly the same as the image below.



The PANELS settings are the default screens, they are two control screens(Normal, Driving) and a parent of the previous screens that is enable/disable if the game is running on a mobile device.

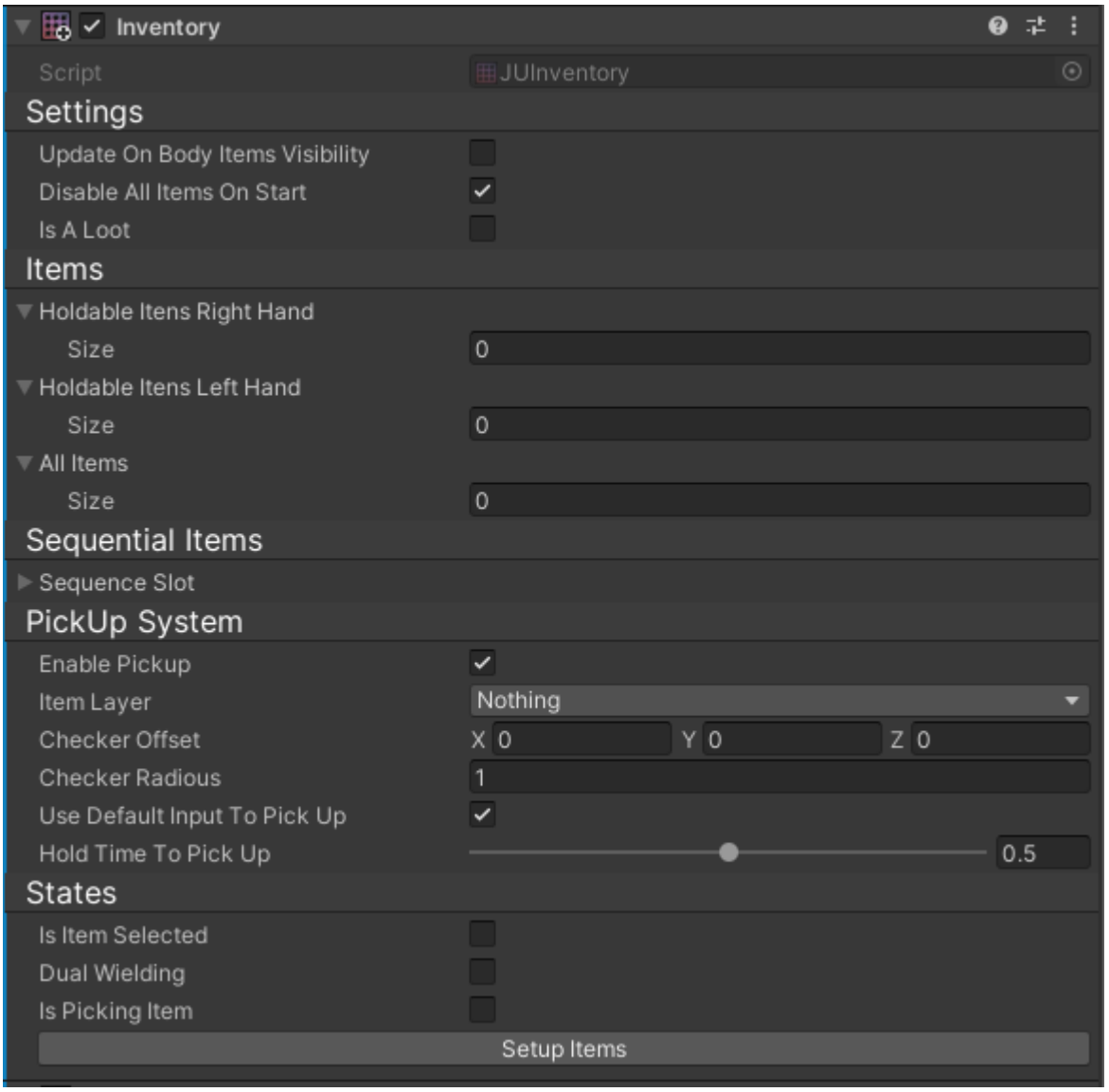
The JOYSTICK settings allow you to use up to two joysticks by default, one for movement and one for aiming and shooting, however this second joystick will only be useful if you are making a Sidescroller or TopDown

How to use inventory and add items

To use items you need an inventory.

How to use Inventory

Simple, just add the "Inventory" component to your JU Character and you are ready to add and use items.

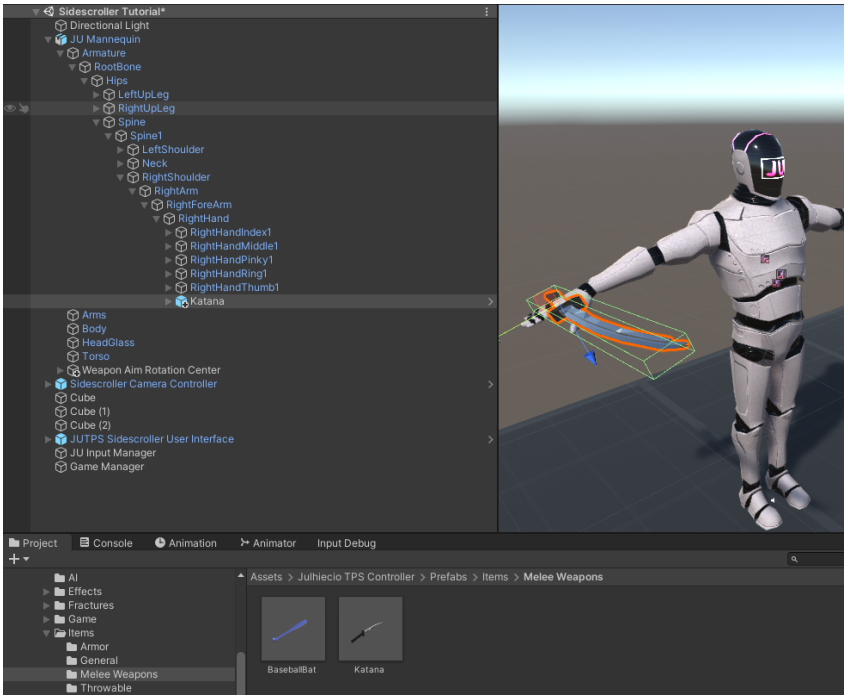


Options	Description
Update On Body Visibility	Updates the visibility of items by body
Is A Loot	This option in an external inventory makes it possible to access items from another inventory (Collecting items from a dead enemy for example)

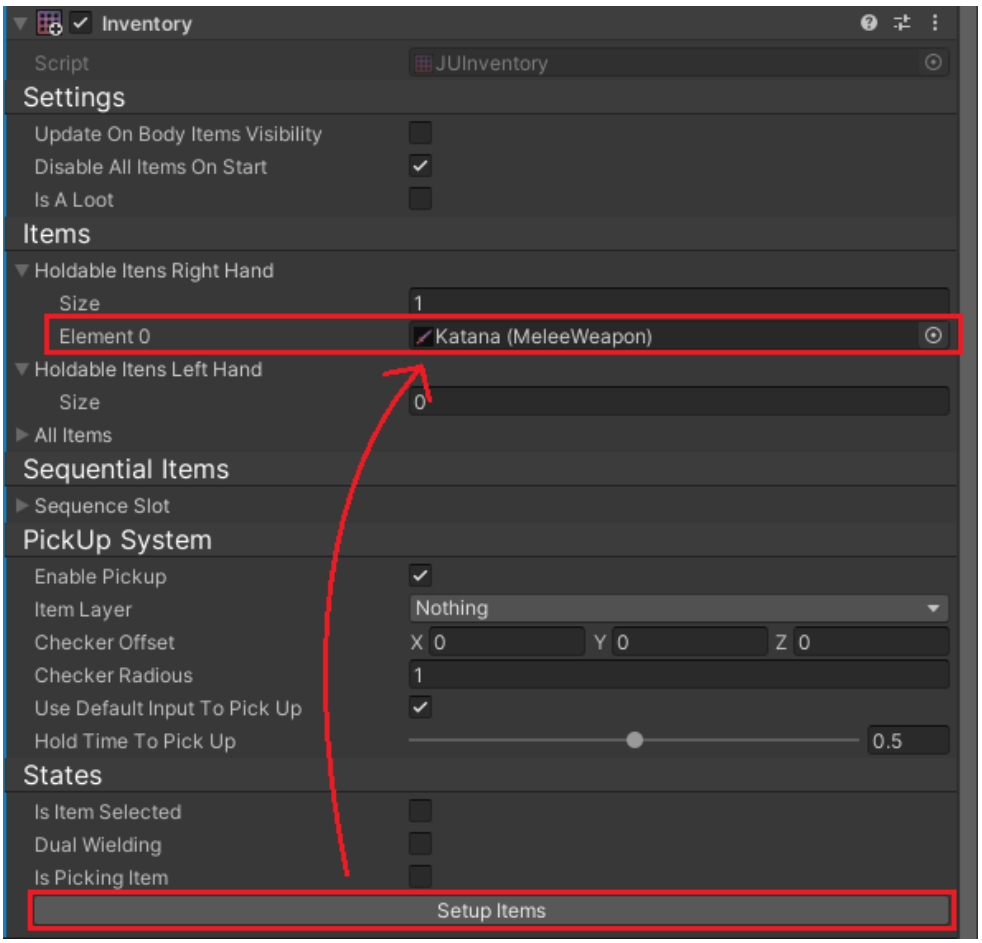
Holdable Items Right Hand	Items that are possible to wield in the right hand
Holdable Items Left Hand	Items that are possible to wield in the left hand
Sequence Slot	Sequence for changing holdable items by alphanumeric key numbers. Can be set by Inventory Slots present in the UI.
Enable PickUp	Enable Item Pick Up System
Item Layer	Pick Up Checker Item Layer
Checker Offset	Pick Up Checker Position Offset
Checker Radius	Pick Up Checker Radius
Use Default Input To Pick Up	Uses standard input control to collect items from environment
Hold Time To Pick Up	Time to pick up a item

How to add items

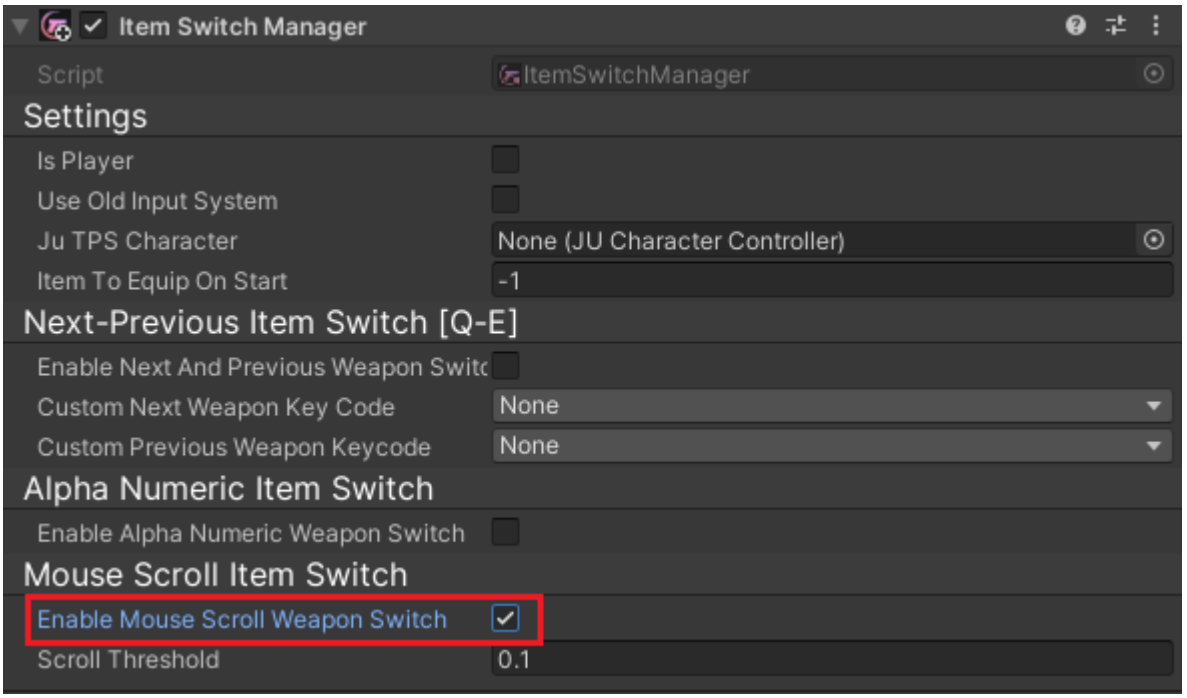
1 - Find the right hand bone and place the item in hand



2 - Click on "Setup Items" button

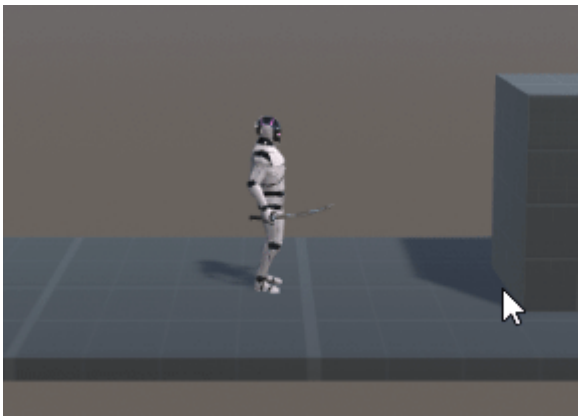


The item is now ready to use, but to be able to change items during the game, let's add the component Item Switch Manager:



Choose the most convenient item switch options for your game.

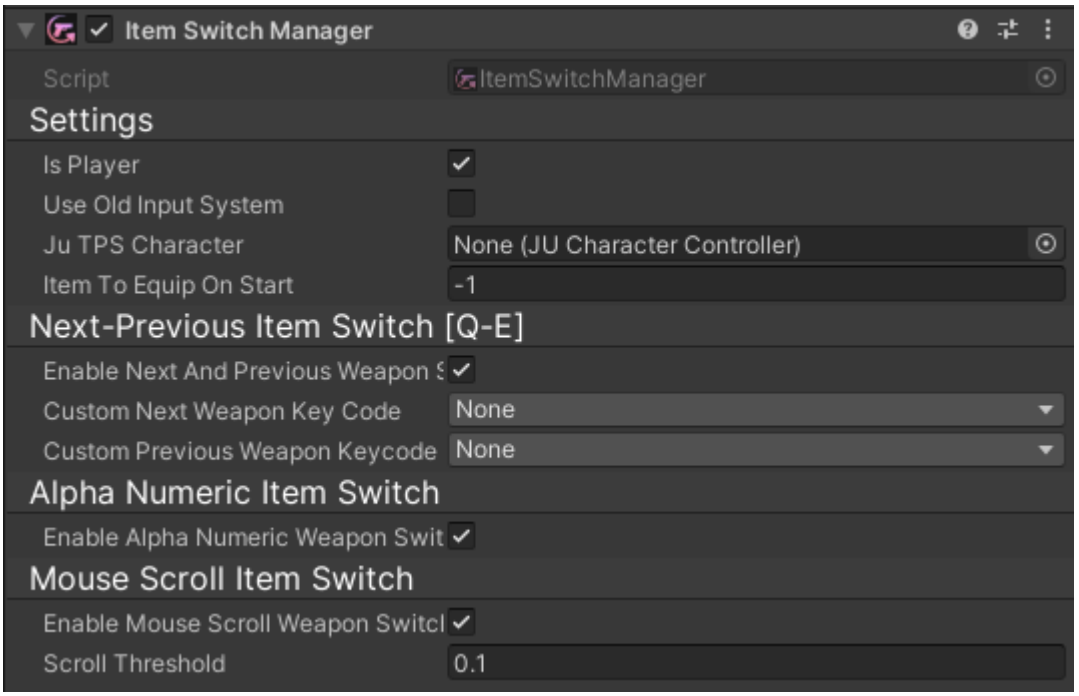
✅ Now you have a item working



How to switch between items

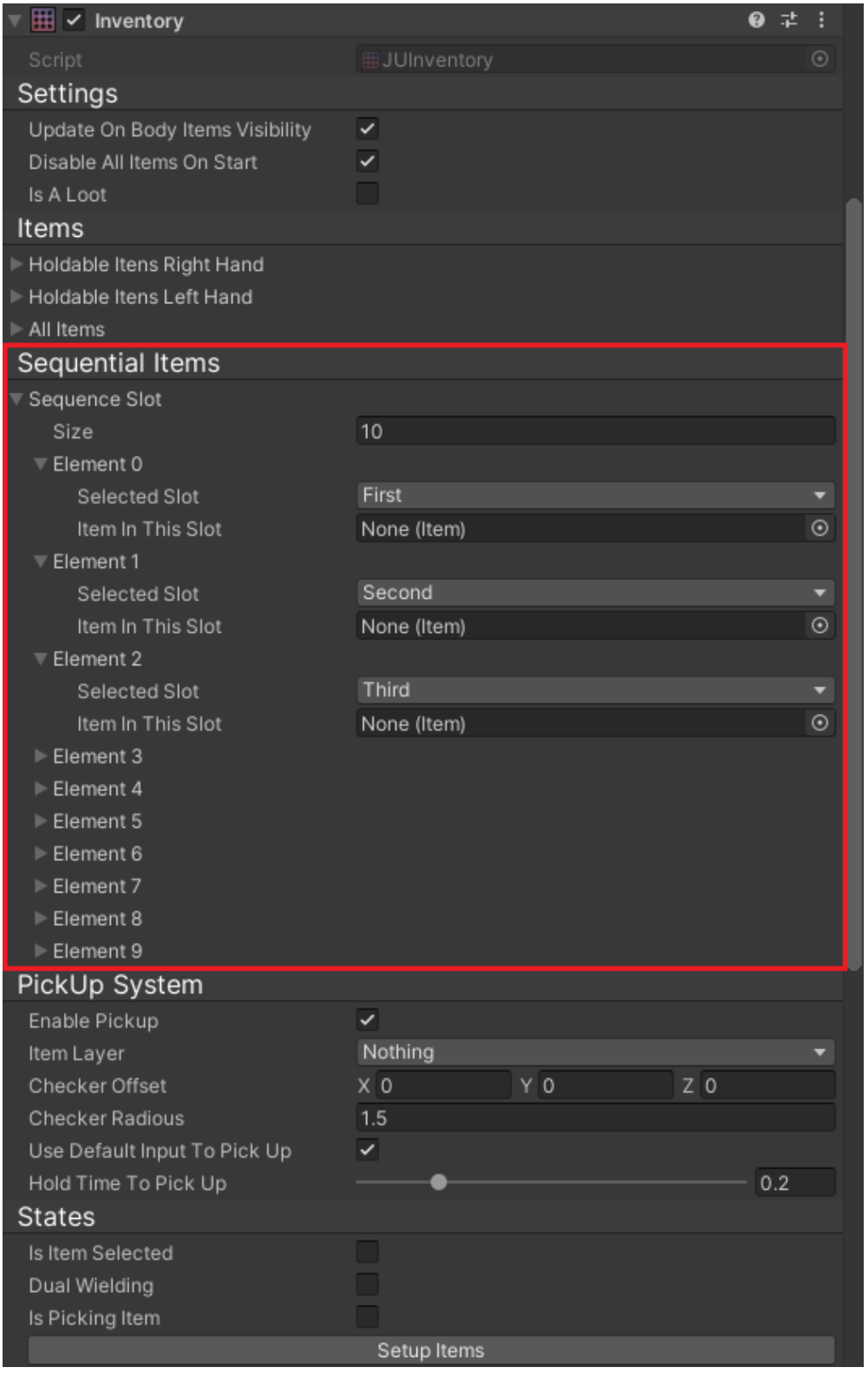
Item Switch Manager

This script controls the item switching system of the characters. It is possible to switch items by scrolling the mouse, sequentially with numbers on the alphanumeric keyboard, and by pressing the next/previous button.

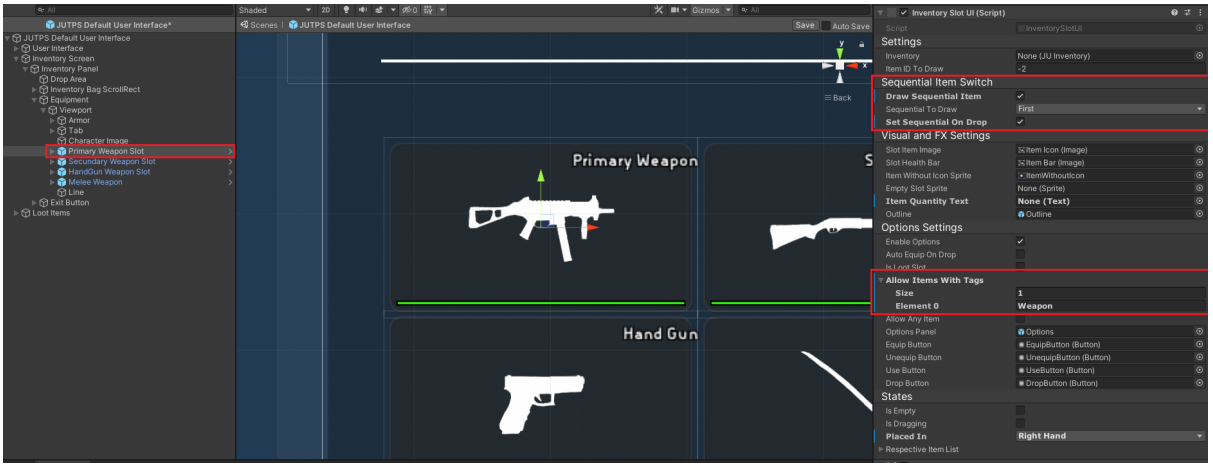


How to use Sequential Item Switching

The item switching sequence is set in the Sequence Slot in the Inventory Component. You can drag and drop items into each slot and it will work normally.



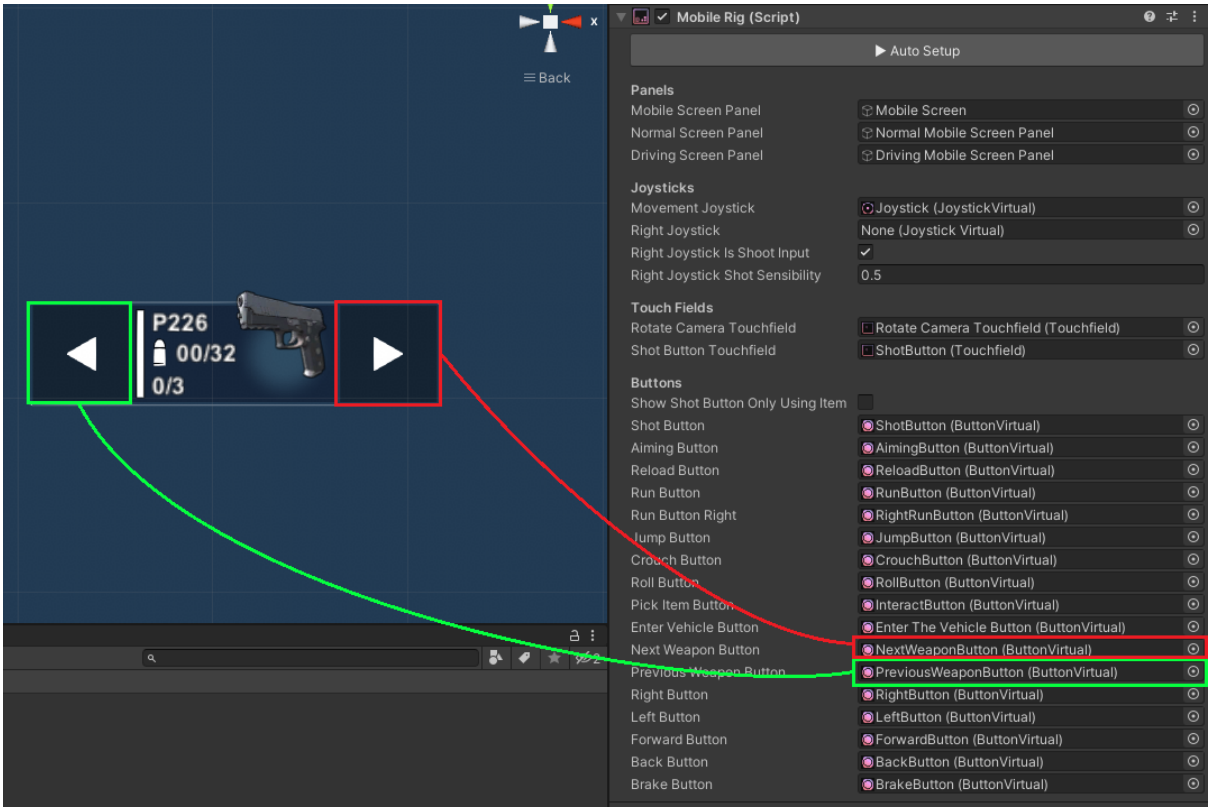
Or you can use Slots in the UI to set items in sequence:



It is important to note that unlike Sequential Item Switching, the item switching order is in the order of position in the Hierarchy(inside player bone hand) by default.

Mobile Item Switching

Item Switching on mobile is done by [Mobile Rig]



Inventory Item Switching

It is possible to equip items from the inventory by pressing the item slot and then the equip button.

Switching by code

To switching an item for a code is simple, the below example code checks when you press the G key and switch items to the next one.

```
using UnityEngine;
public class SwitchOnPressGKeyExample : MonoBehaviour
{
   void Update()
   {
        if (Input.GetKeyDown(KeyCode.G))
        {
            ItemSwitchManager.NextPlayerItem();
        }
   }
}
```

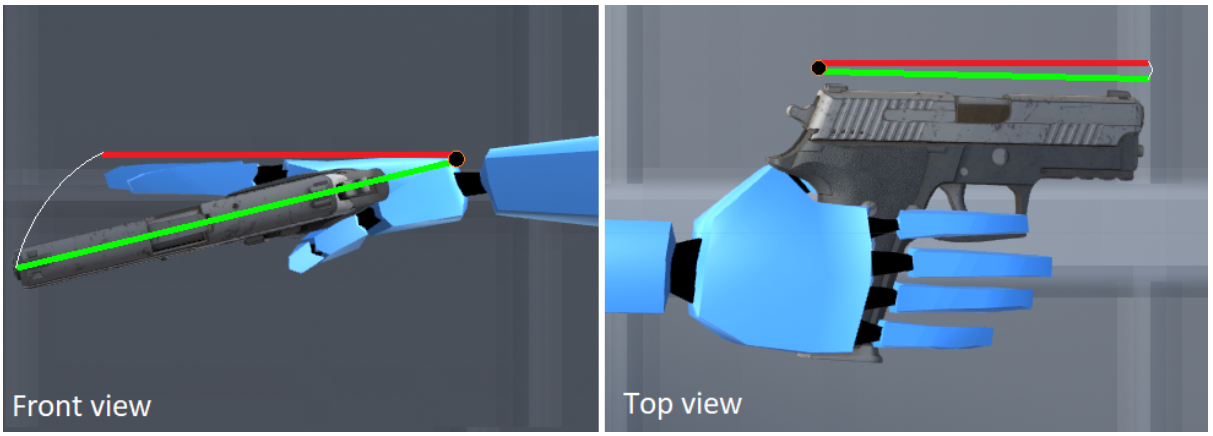
Some static methods you can use in code for item switching:

Static Methods	Description
SwitchCharacterItem(JUCharacterController character, int SwitchID)	Switch a JU Character item to a specific one in the list
SwitchPlayerItem(int SwitchID)	Switch player(JUCharacter with "Player" tag) item to a specific one in the list
NextPlayerItem()	Switch player(JUCharacter with "Player" tag) to next item in sequence.
PreviousPlayerItem()	Switch player(JUCharacter with "Player" tag) to previous item in sequence.

How to align weapon rotation

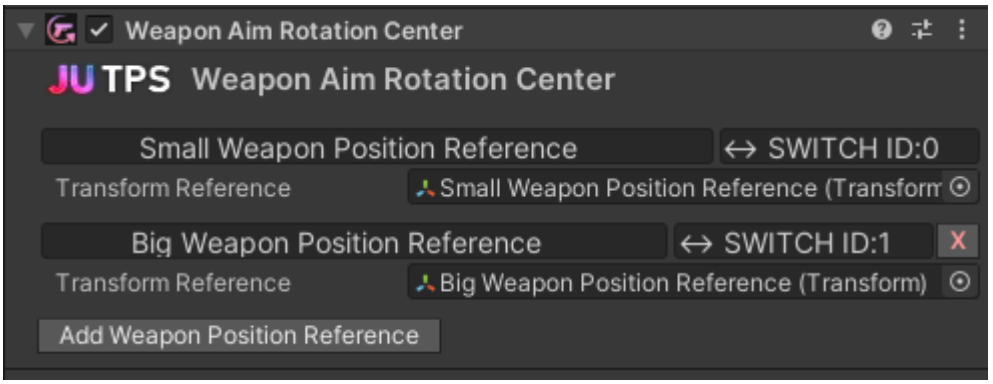
Align in hand weapon direction

If you align like this by default you will get a good result:

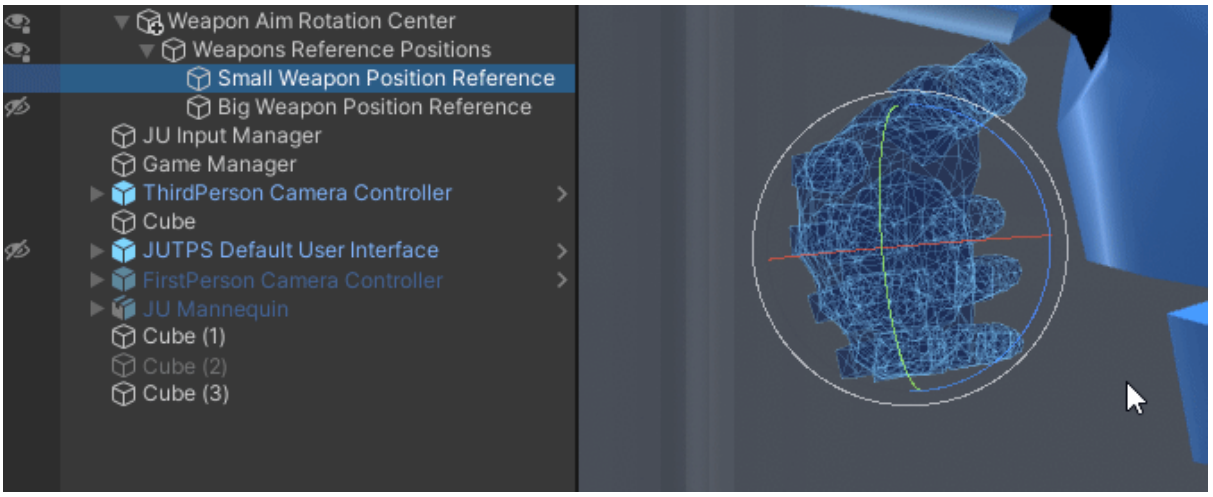


Item Aim Rotation Center Align

Is an essential component to assist with hand position and rotation during Fire Mode, it is automatically created in the JU Character Controller setup.



The Item Aim Rotation Center has the item wielding positions, and this will affect the direction of the weapon.

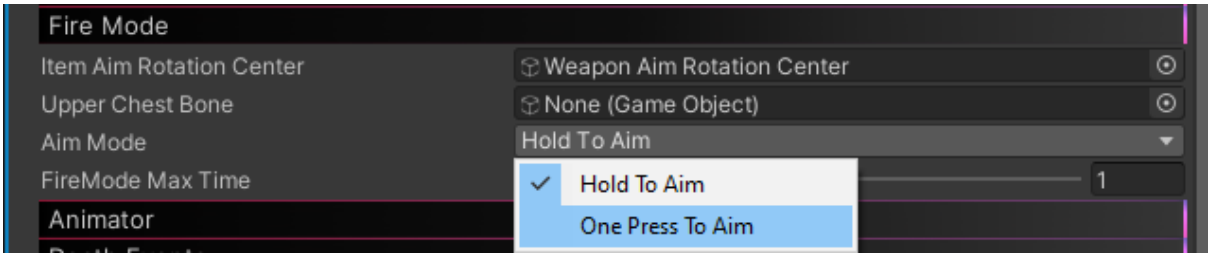


This rotation will not affect the trajectory of the bullet, adjust it so that the rotation is closer to aiming at the center of the camera. You may need to adjust the weapon rotation as well to get it to look good.

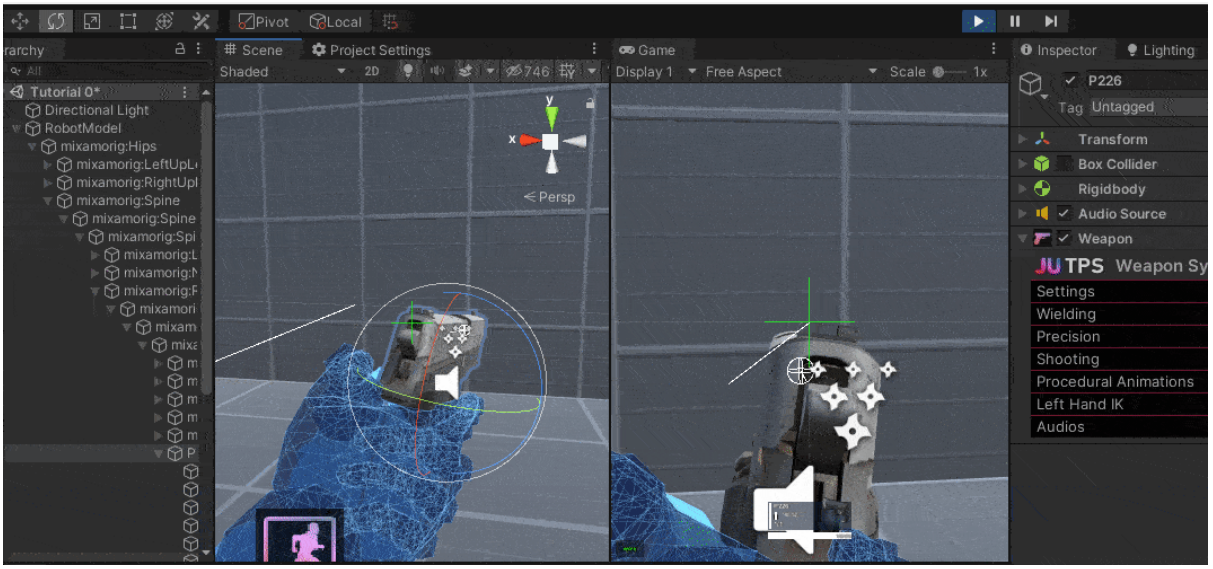
Aim Mode Align

If you want to use "Aim Mode" you will have to adjust the rotation of the weapon in this mode.

First, change your JU Character Controller's setting to "One Press To Aim" mode to adjust properly:



Place the Game and Scene windows side by side, press play and aim, press the Esc key and click on the weapon in the Hierarchy. Then just adjust the rotation so that the weapon's crosshairs align correctly with the center of the screen crosshair:

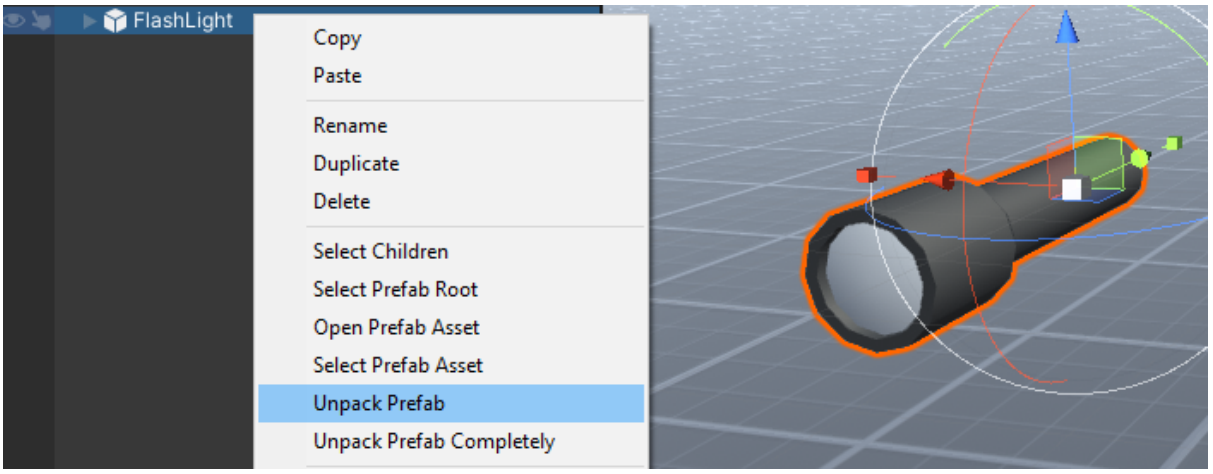


Note: if you are not seeing this green crosshair on the screen activate Gizmos
After aligning, just copy the rotation of the weapon, exit Play Mode and paste the rotation into Transform Component.

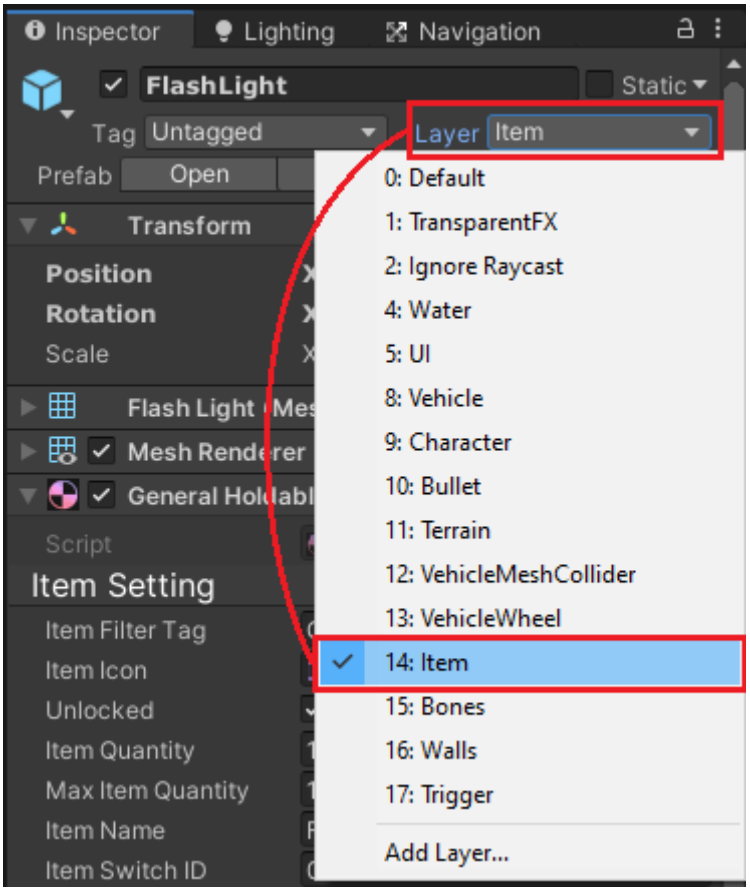
How to create weapons and items

First Steps

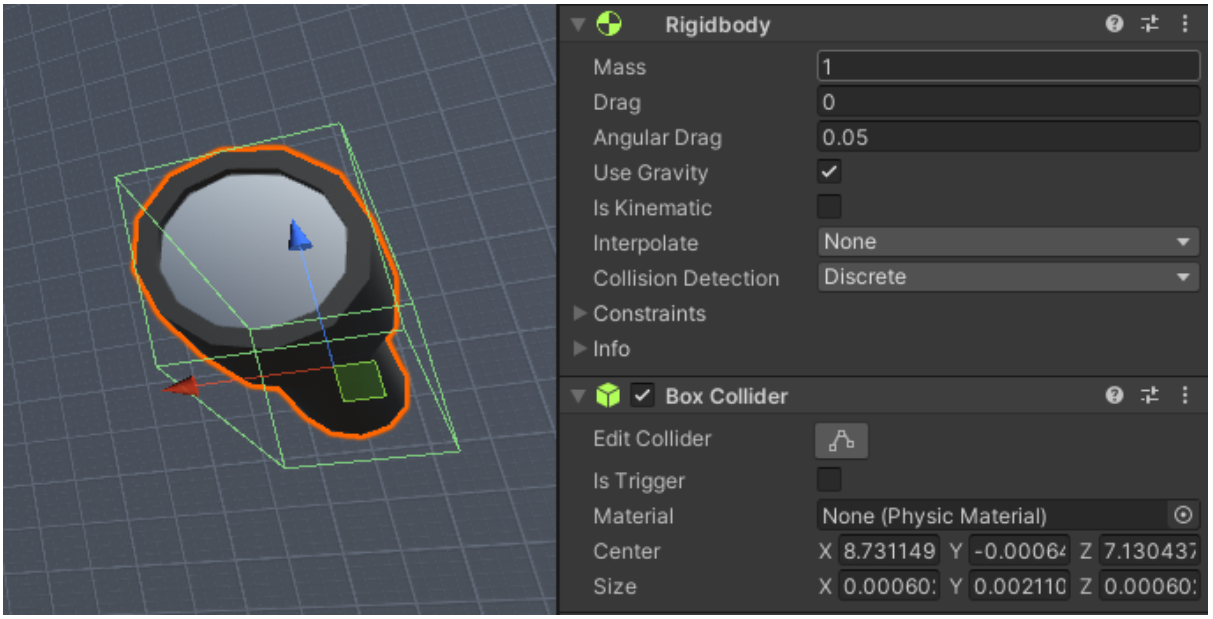
1- Place Item Model in Scene and Unpack Prefab



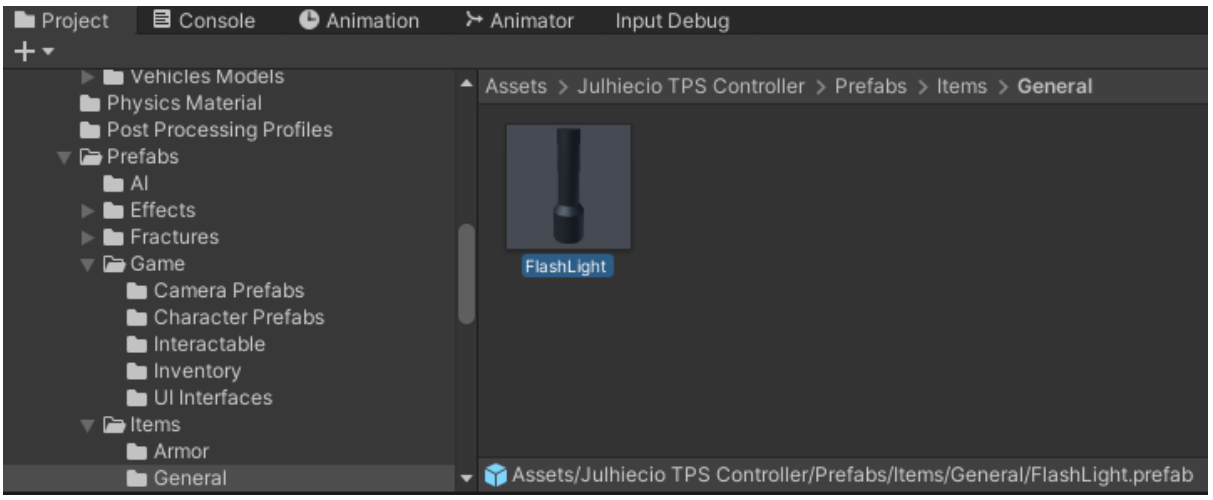
2- Set layer to "Item" layer



3- If you want use Drop System and Pick Up System, add a Collider and RigidBody

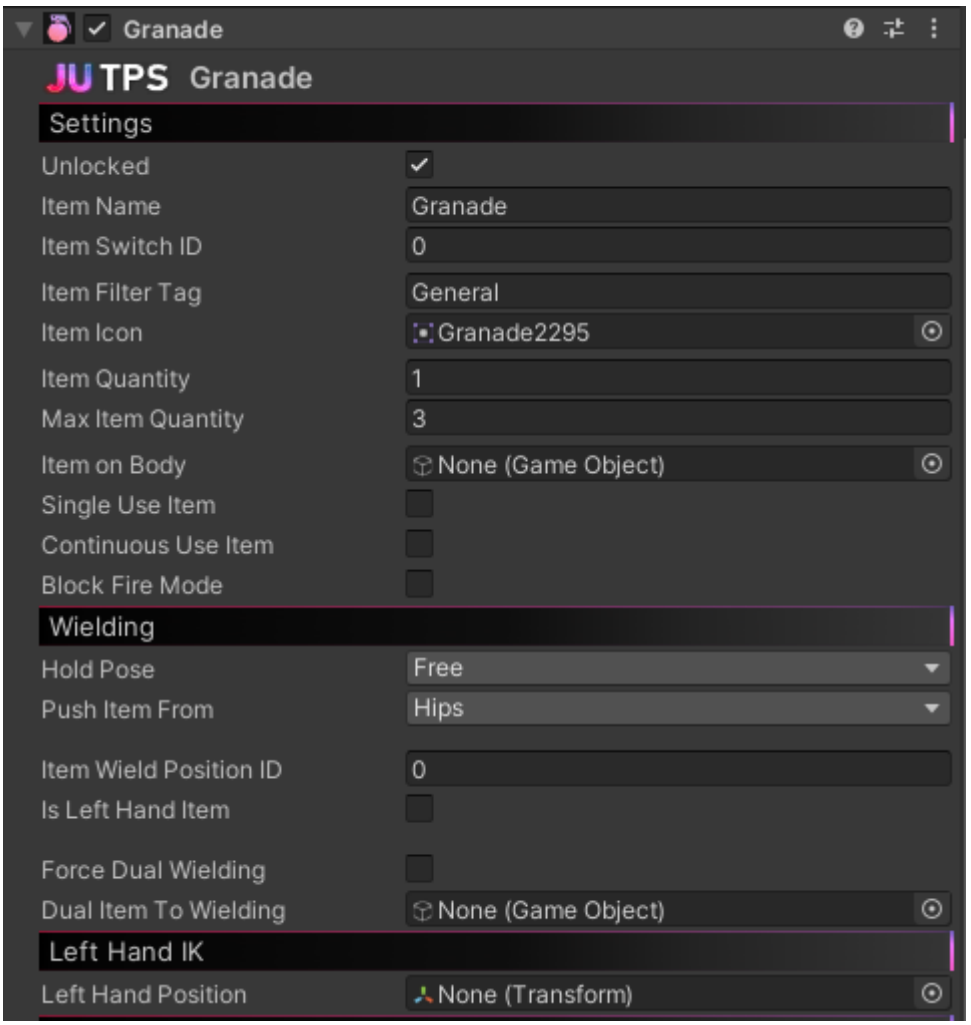


4- Add any of the components below, configure it and create the prefab by dragging it to a folder in the Project window.



Basic Holdable Item Setup

In JU TPS there are some types of ready-made holdable items, they are: General Holdable Item, Weapon, Granade and Melee Weapon
All holdable items will have the essential parameters below, this means you can repeat the first steps for all holdable items



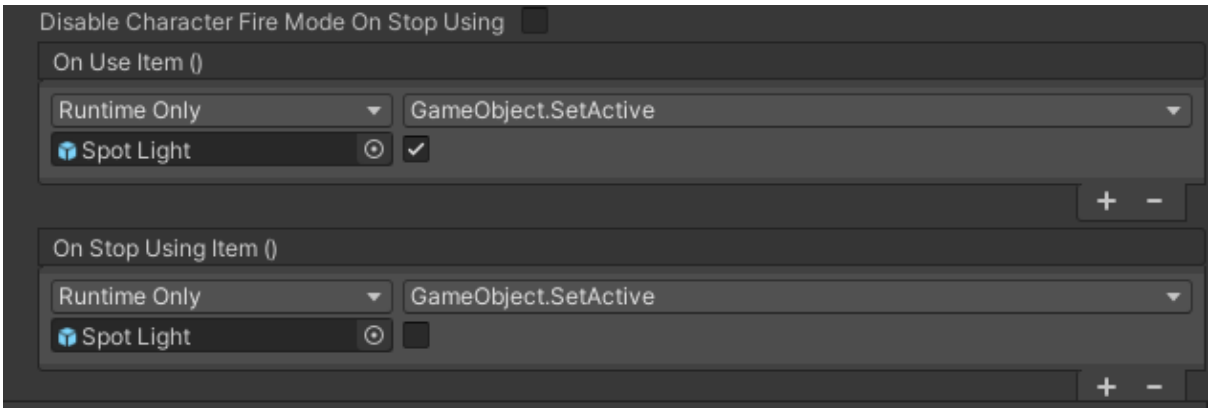
Grenade component as an example of a holdable item

Parameters	Description
Unlocked	When checked, the item is ready to be equipped and used
Item Name	Name of how the item will appear in the inventory, it is also required for other actions such as Pick Up
Item Switch ID	Index for the Item Switch System, it is automatically set when the game starts, as the value must be according to the position in the hierarchy
Item Filter Tag	You should use this when you want to differentiate between types of items, to filter what can be equipped in an Inventory Slot. ⚠ Example: Primary Weapon, Secundary Weapon[...]

Item Icon	Icon of the Item that will appear in the inventory and in the "Current Item Information" present in the default UI templates
Item Quantity	Quantity of items, if it is 0 the item will be blocked
Item On Body	Visual 3D model of the Item on the body, it is used to distribute the items on the body when not in use, not necessary if you don't want to
Single Use Item	When used the item will be locked.
Continuous Use Item	With this option, while the use button is being pressed, the item's use function will be called at each Frame
Block Fire Mode	For items that don't need Fire Mode to work, like grenades and melee weapons, which can block Fire Mode to work properly
Hold Pose	List of poses to hold item while not in Fire Mode
Push Item From	Basically the animation of equipping the item
Item Wielding Position ID	Wielding position index during FireMode (You can see IDs in Item Aim Rotation Center)
Is Left Hand Item	This is to differentiate if the item is left hand item
Force Dual Wielding	The Item when equipped will activate the "Dual Item to Wielding" in the left hand and they will work together.
Dual Item To Wielding	Item in left hand to be activated when Item in right hand is equipped for Dual Wielding System to work.
Left Hand Position	A GameObject that stores the position and rotation of the left OR right hand wielding the item ⚠ Example: the left hand supporting the right hand on the pistol

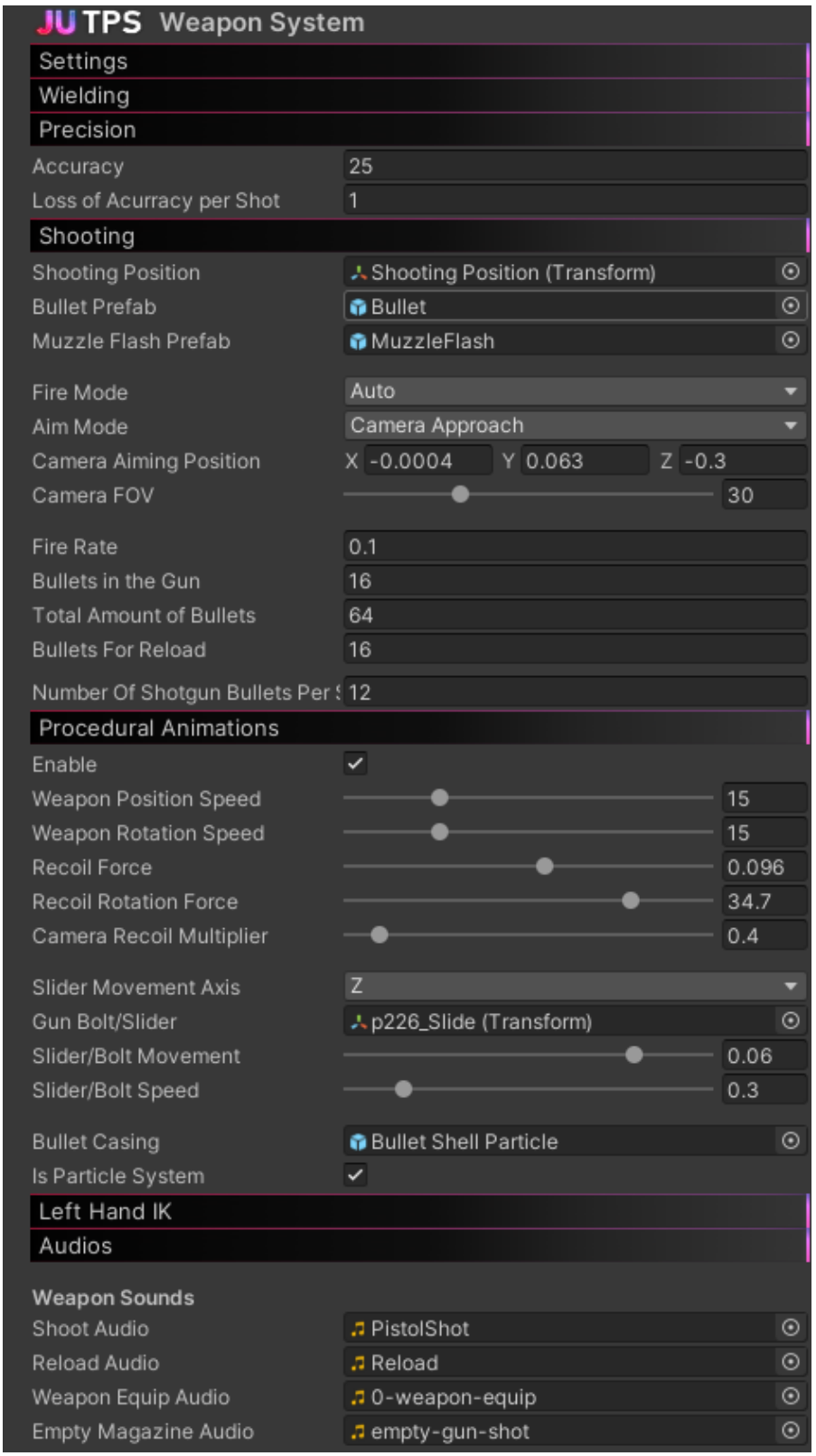
General Holdable Item Setup

It has no differences from the default holdable items settings other than events that you can use to call functions from code or simple actions like turning a light on and off:



Weapon Item Setup

The Weapon Component has some unique parameters that you will need to configure:



Precision System Parameters	Description
Accuracy	It's the accuracy of the weapon 👍

Loss of Accuracy per Shot	The larger it is, the less accurate the gun will be with each shot. It is recommended to leave it at a low value.

Shooting System Parameters	Description
Shooting Position	GameObject that keep the position where the bullet is instantiated, is also used to calculate the direction of the bullet.
Bullet Prefab	Prefab of Bullet
Muzzle Flash Prefab	Prefab of Muzzle Flash Effect
Fire Mode [Auto, Semi Auto, Bolt Action, Shotgun]	Auto: Machinge Guns Semi Auto: Pistols Bolt Action: Snipers Shotgun: Shotguns
Aim Mode [Camera Aproach, Scope Texture]	Weapon sight type
Camera Aiming Position	Camera position while aiming.
Camera FOV	Camera Aiming FOV
Fire Rate	Shooting fire rate
Bullets in the Gun	Bullets loaded in the magazine
Total Amount of Bullets	Total bullets stored of the weapon
Bullets for Reload	Number of bullets per magazine
Number of Shotgun Bullets per Shot	Number of bullet instances per shot in a shotgun

Procedural Animation Parameters	Description
Enable	Enable procedural weapon animations.
Weapon Position Speed	Speed at which the weapon returns to its original position after a shot.

Weapon Rotation Speed	Speed at which the weapon returns to its original rotation after a shot.
Recoil Force	Force at which the gun pushes the hand back after a shot.
Recoil Rotation Force	Force at which the gun rotates the hand after a shot.
Camera Recoil Multiplier	Force that the recoil affects the direction of the camera.
Slider Movement Axis	Axis of movement of the weapon slider.
Gun Bolt/Slider	Weapon Slider GameObject
Bolt/Slider Movement	The distance the Slider will be pulled back after the shot.
Bolt/Slider Speed	Speed at which the Slider will return to its original position.
Bullet Casing	Bullet casing, can be a GameObject to instantiate or a Particle System(you have to check the option below to use Particle System)
Is Particle System	Bullet casing is a Particle System

Weapon Audio System Parameters	Description
Shot Audio	Shot Audio Clip
Reload Audio	Reload Audio Clip
Weapon Equip Audio	Equip Audio Clip
Empty Magazine Audio	No Ammo Audio Clip

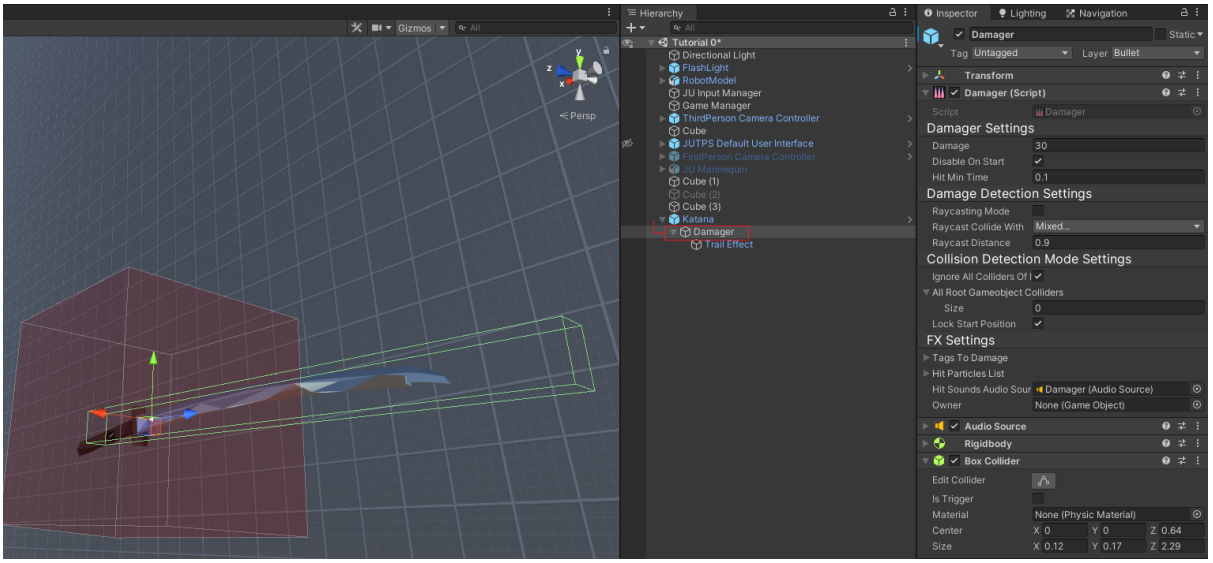
Melee Weapon Setup

Melee weapons are simple:

Melee Weapon System Parameters	Description
Attack Animator Parameter	Parameter that triggers the sequence of attacks by the JU Character animator. So far by default there are three parameters in Animator: OneHandMeleeAttack, TwoHandMeleeAttack and DualHandMeleeAttack
Damager To Enable	Damager Gameobject to enable when using melee weapon.
Enable Health Loss	Enable health to item
Health	Health value
Damage Per Use	When the Damager collides, the weapon takes damage

Damager

Damager is a script that basically works to damage any GameObject with JU Health Component.



Needs a collider and a rigidbody to detect collisions.

Damager Parameters	Description
Damage	Damage
Disable On Start	Disable GameObject on game start
Hit Min Time	Delay to hit again after a hit
Raycasting Mode	This mode does not need a collider or a non-kinematic rigidbody.
Raycast Collide With	Raycast Layer Mask
Raycast Distance	Raycast Distance
Ignore All Colliders of Root GameObject	Ignores all colliders of the root GameObject (this is to ignore the player's body colliders for example and not damage itself)
Lock Start Position	As the Damager needs a non-kinematic rigidbody to detect collisions, it is normal for it to physically move, this option locks the local position of the Damager
Tags To Damage	All tags listed here will receive damage on collision with the Damager (the object being collided needs a JU Health)
Hit Particles List	Particle and audio effects for each tag

Hit Sounds Audio Source	Audio Source to play audio clips

Throwable Item / Granade Setup

The Grenade Component is basically the Throwable Item Component with an explosion option:

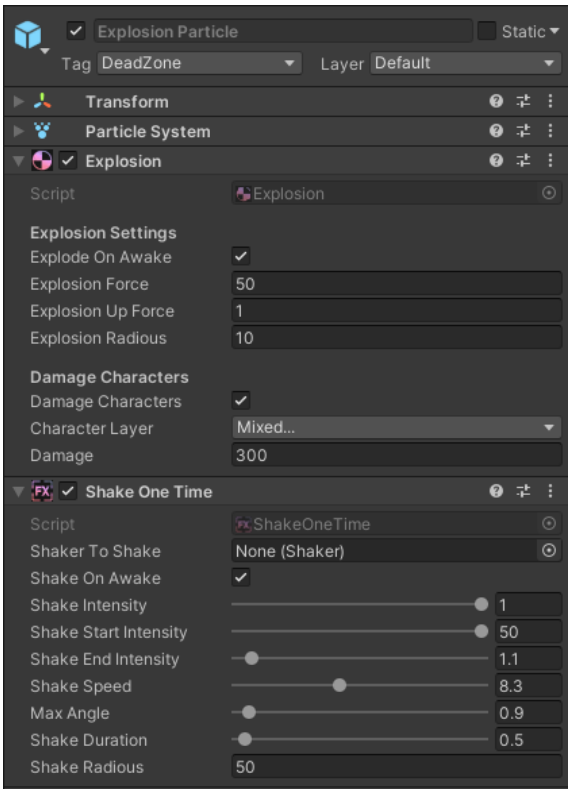
Throw Parameters	Description
Animation Trigger Parameter	Parameter that will be called in the JU Character animator to throw the object. ⚠ Note that throw function is called by a JU Animation Event
Throw Force	Throw Force
Throw Up Force	Throw Up Force
Rotation Force	Throwed Object Rotation Force
Seconds To Destroy	Seconds to destroy after being thrown. ⚠ Set the value to 0 if you don't want the object not to be destroyed

Position To Throw	Throw Position (Pay attention to Gizmos)
Direction To Throw	Throw Direction (Pay attention to Gizmos)

Explosion Parameters	Description
Explosion Prefab	Prefab with the explosion effect
Time To Explode	Time for the grenade to explode
Time To Destroy Explosion	Time that the explosion effect will visually last

Explosion Effect

The explosion effect can be equipped with the Explosion Component which can physically simulate an explosion and deal damage to characters. It's also interesting to put the Shake One Time component in the explosion, it will do the Camera Shake effect (Make sure you have a Shaker

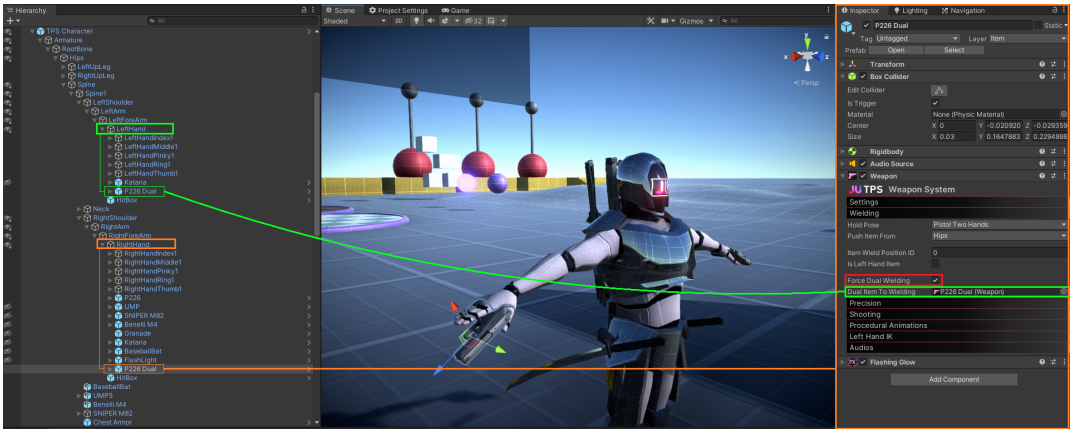


How to use Dual Wielding System

In JU TPS it is possible to use items in both hands at the same time in a simple way

Simple Setup

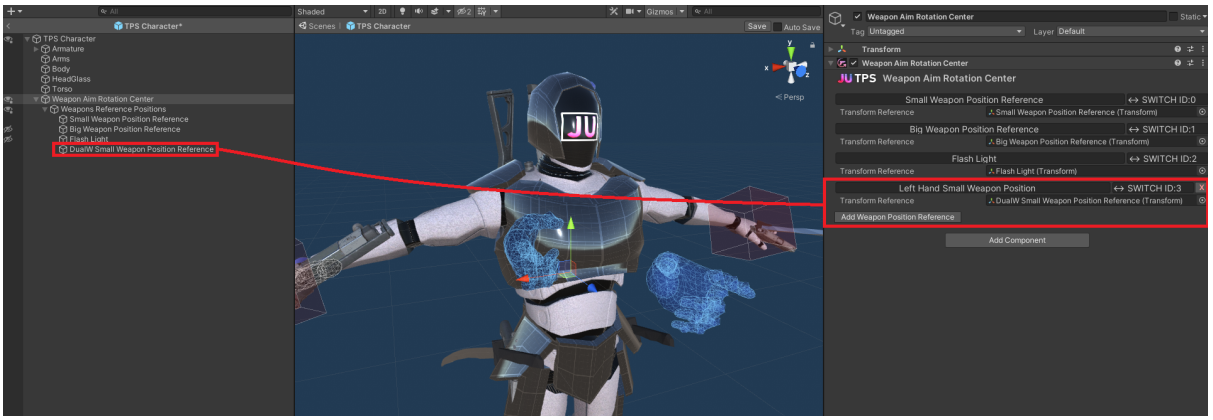
To use Dual Wielding is simple, just have two items, one in each hand, and in the right hand item, check the "Force Dual Wielding" option and assign the left hand item in the "Dual Item To Wielding" parameter of the right hand item.



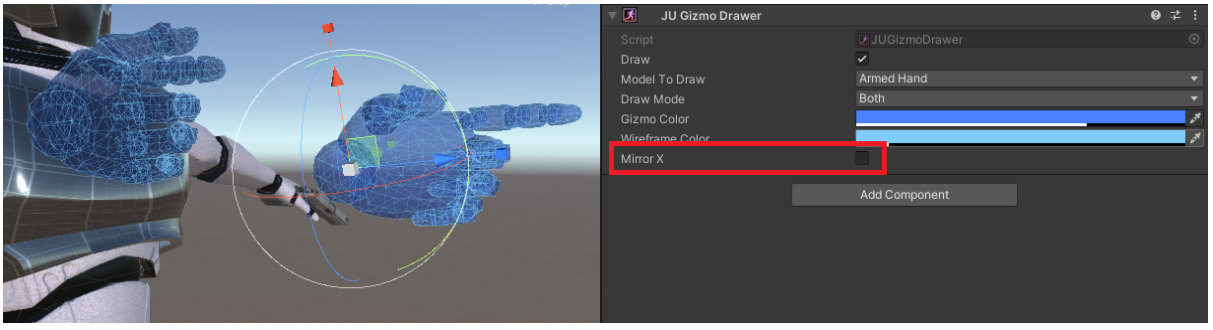
Open image for a better visualization

Left Hand Wielding

To get the left hand wielding correctly, create a new position for the left hand only:

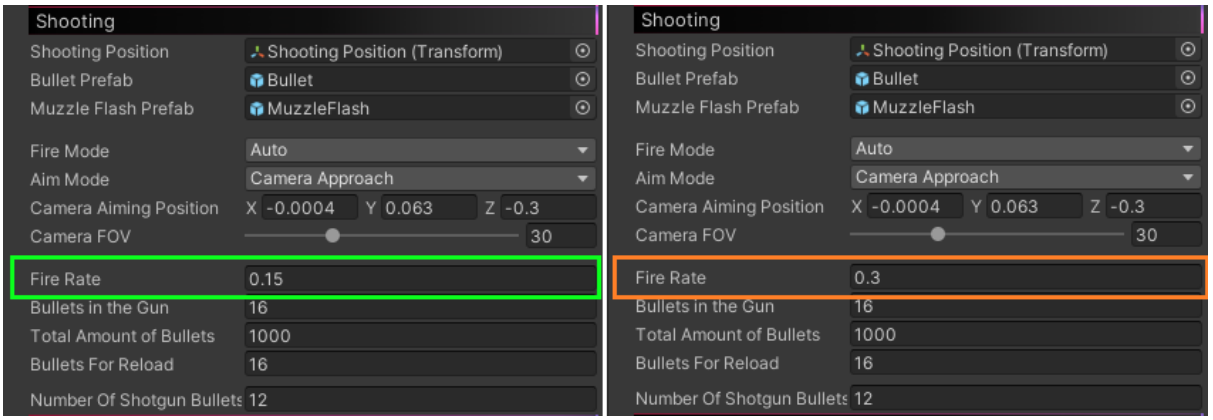


You can select the new wielding position and disable the "Mirror X" parameter, so the left hand will be drawn.



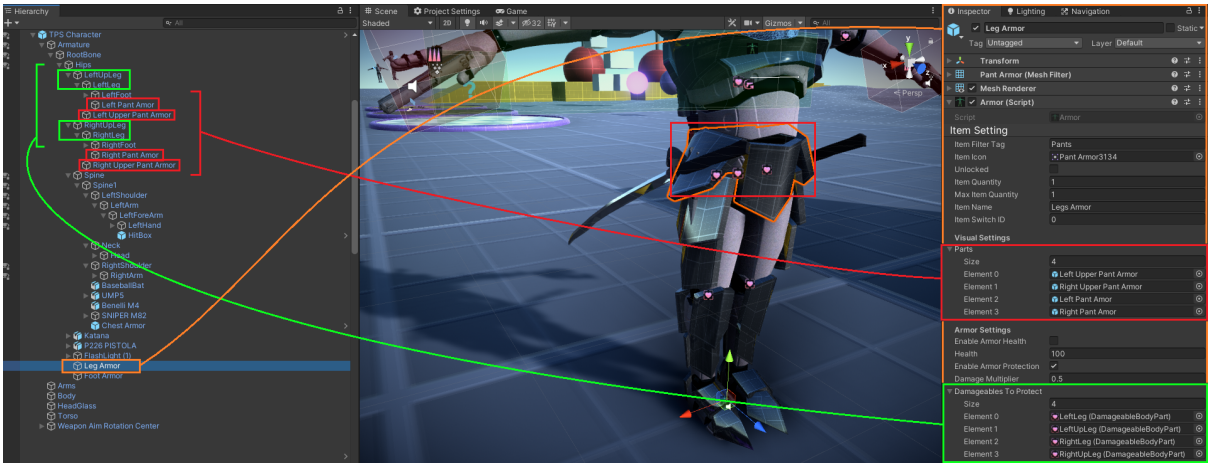
Weapon and Fire Rating

In this mode it is interesting to use the Auto FireMode and the "Continuous Use Item" option and vary the Fire Rate of the weapons in left hand and right hand:



How to use Armor/Cloth System

See how the armor system works below in a simple way
🟧 The armor works in a simple way, I'll use Leg Armor as an example.
🟥 As this armor has several parts, the models are distributed by the body separately so that they move correctly, and then assigned in the "Parts" parameter of the Armor.
🟩 And to use the protection system activate the "Enable Armor Protection" and assign the Damageable Body Parts in the parameter "Damageables To Protect".



Armor is a Non-Holdable Item, so it can be placed anywhere on the body, it can also be equipped, dropped and picked up like other items.

You don't need to use the protection system, so you can use this component as a clothing system only.

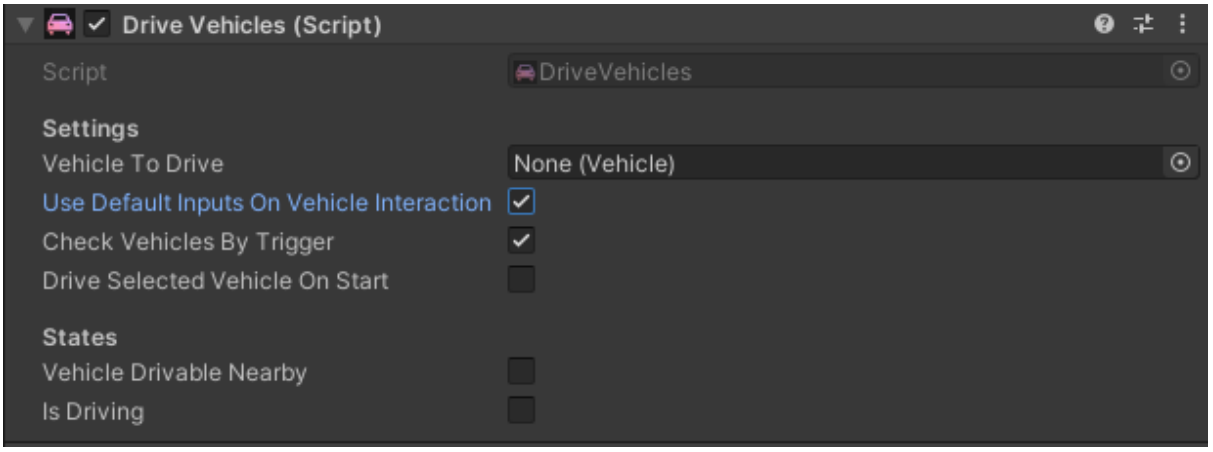
It is also not necessary to have several parts of an armor, you can separate each part as you wish, the example is to show that it is possible to use several models for only one Armor component.

How to use Vehicle System

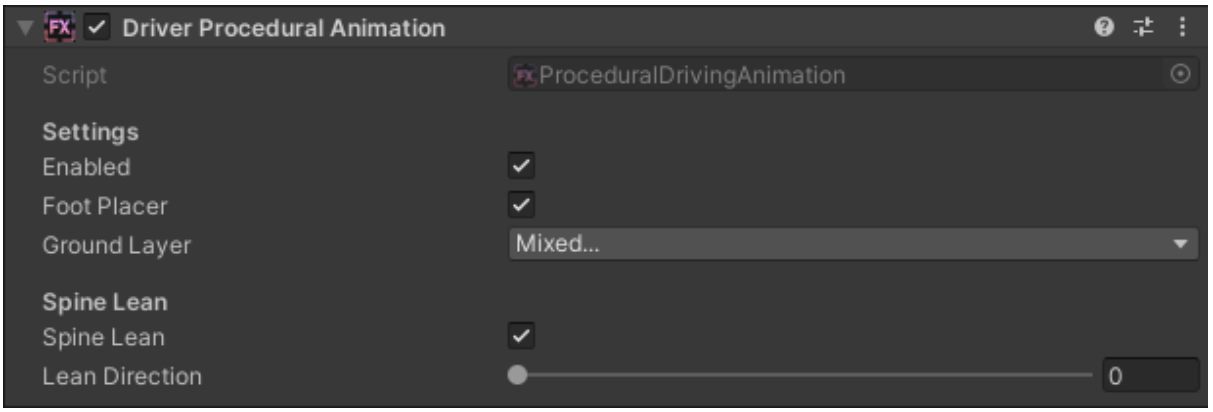
On this page you will learn how to make your JU Character drive vehicles

Drive Hability

For the JU Character to be able to drive, you need to add the component Drive Vehicles:



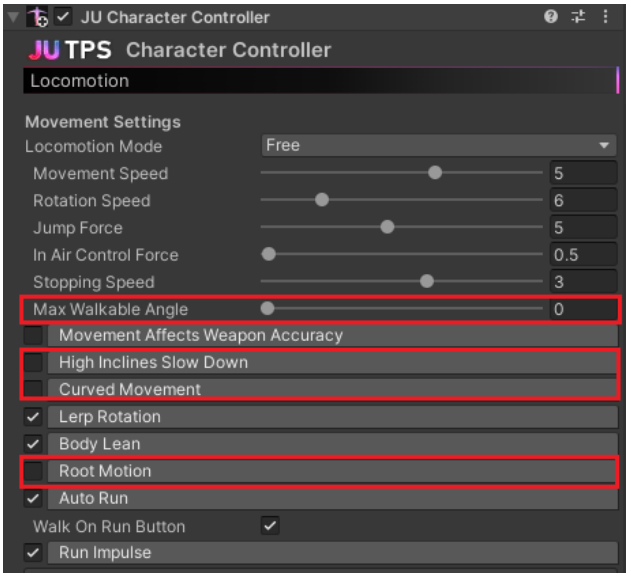
Procedural Driving Animation



How to use Gravity Switching

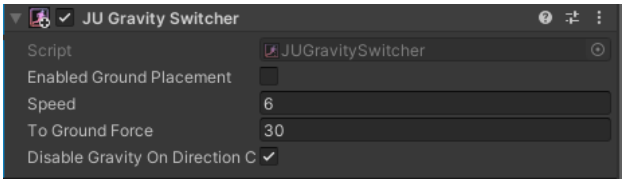
Controller Configuration

To use the Gravity Switching system your JU Character needs some essential changes, make sure the highlighted options are set like this:



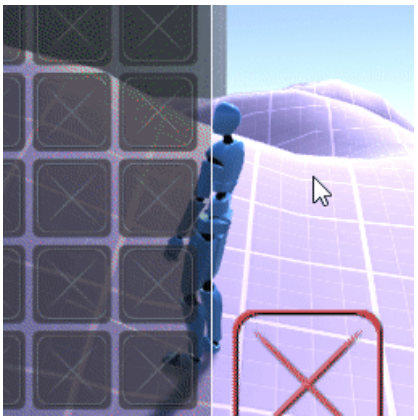
JU Gravity Switcher

Add this component so that the JU Character can adapt correctly to surfaces.



JU Gravity Switcher Parameters	Description
Enable Ground Placement	It will adapt the JU Character's rotation to any surface without needing any external components, it will be like a sonic game
Speed	Aligning Speed
To Ground Force	(Optional) Add force in Ground Placement to fix the character on the ground
Disable Gravity On Direction Change	If gravity direction is different than usual, "Use Gravity" on rigidbody will be disabled

See Ground Placement working:

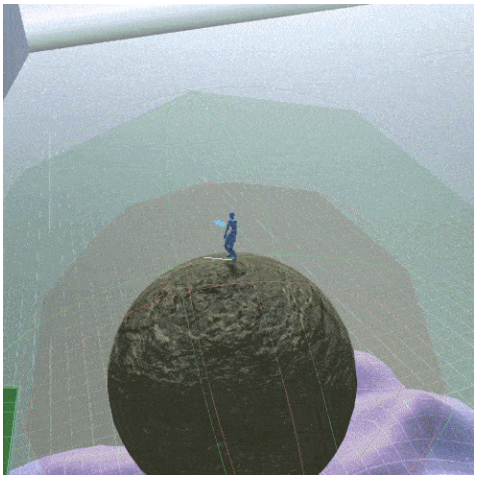


Gravity Box

An invisible box that switch gravity direction

Gravity Sphere

Like a planet 👍



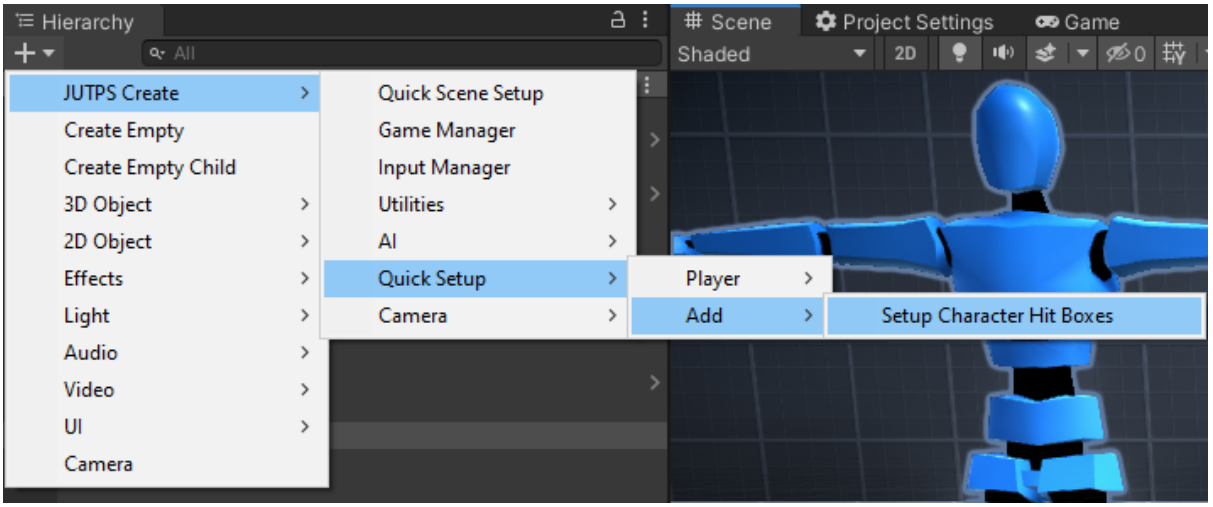
How to use Enemies AI

Standard first steps to use Character AI

1- Create a JU Character in Simple Setup
2- Turn on IsAI option and set tag to Enemy(or any other tag that is not "Player")
3- Add a JU Health component and JU Inventory
4- Bake a NavMesh
Your JU Character is now an NPC who can die and also keep items.

Hit Box

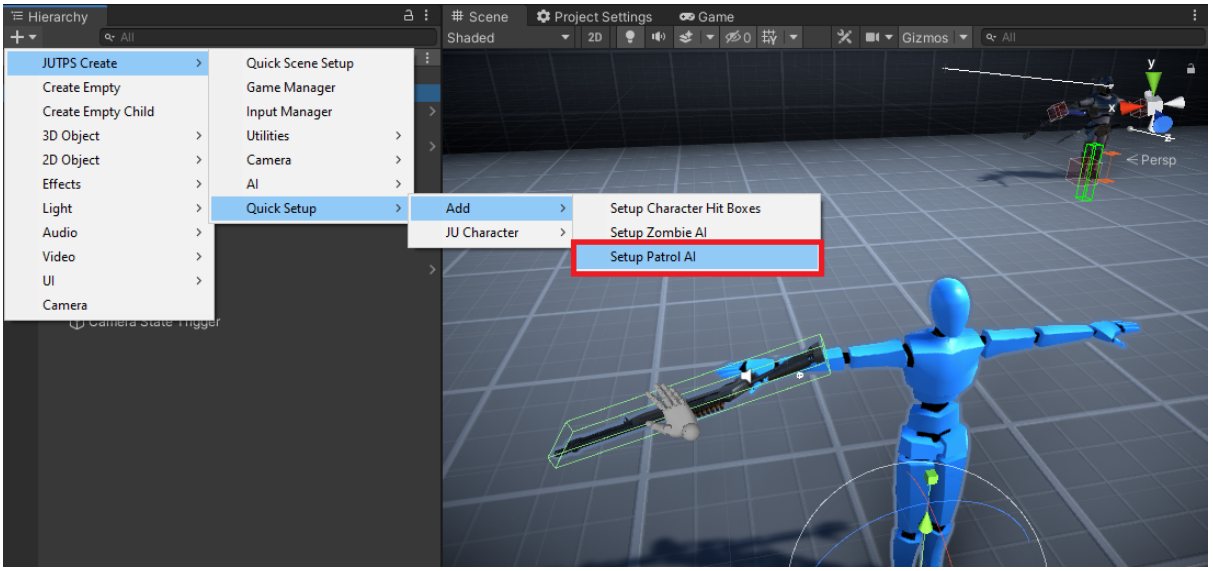
To be able to deal damage by punches, make the Hitbox setup:



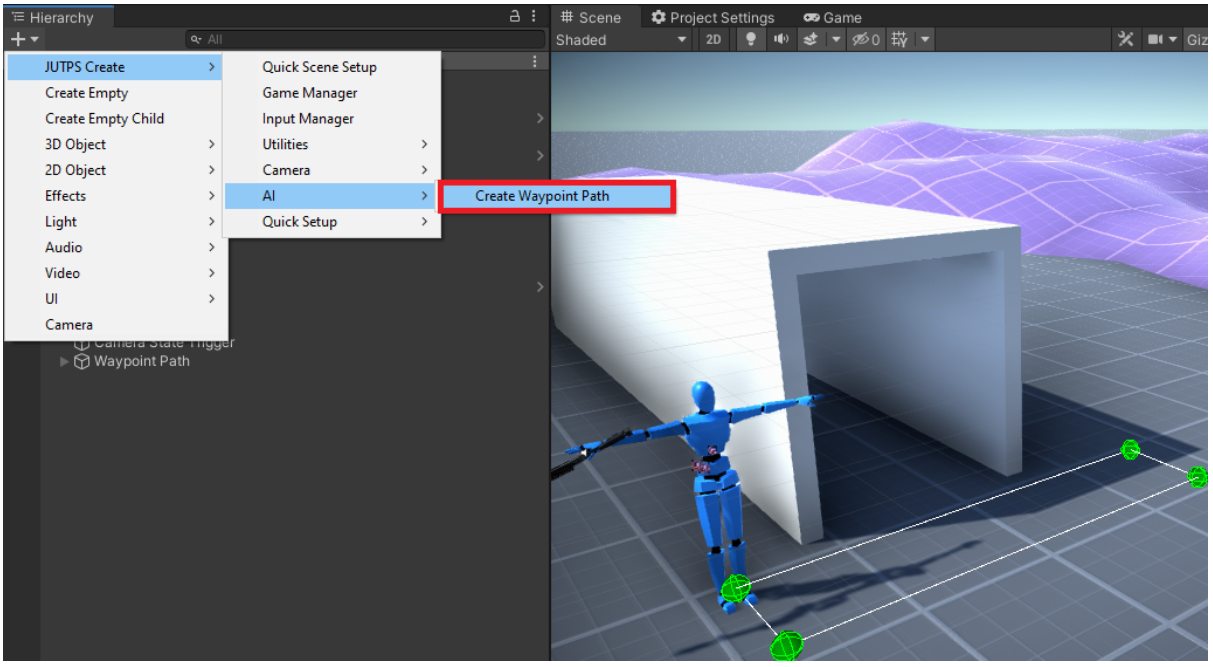
Adjust the size of the hit boxes the way that best fits your character

Patrol AI

The Patrol AI has an extremely similar setup to the Zombie AI, with the difference that it adds a Inventory and an Item Switch Manager. You can add a weapon normally and the AI will start using it, and in the Item Switch Manager you can set the ID of the item it will equip by default.



You can create a Waypoint Path and assign it to the Patrol AI, and you will have a Patrol Enemy:



✅ Patrol AI configured successfully

Zombie AI

The Zombie AI component makes the character walk to a destination or a Waypoint, follow the player and attack him with punches. Your character doesn't necessarily have to be a zombie, you can use this script to make an aggressive NPC for example.

it only takes 3 clicks to setup a zombie enemy, similar to Patrol AI above.
You can also adjust the parameters below to better adapt to what you want

Default AI Class Parameters	Description
Destination	Destination
Waypoint Path	Waypoint Path
Distance To Finish One Point	Distance to go to next waypoint/path index
Check Nearest Point On Part	Checks the nearest point on the waypoint/path and prevents the AI from going back to a previous Waypoint
On Start Path [Unity Event]	Event called when AI start path
On Following Path [Unity Event]	Event called when AI walk along the path
On Ended [Unity Event]	Event called when AI reaches the end

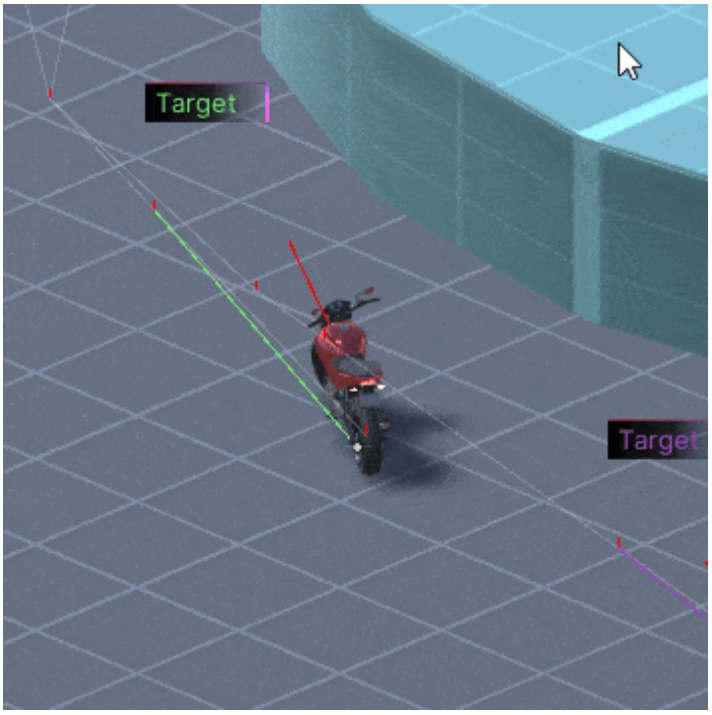
Zombie AI Parameters	Description
Target Tags	Filter targets by tags
Field Of View	AI field of view, it is by this sensor that will detect targets
Sensor Layer Mask	Layer Mask of Field of View
Start Running At Distance	Runs when the target is greater than this distance
Attack At Distance	Attack at distance
Aim Up Offset	Target direction height
Look At Target Speed	Look at rotation speed

Min Time To Attack	Min Time To Attack
Max Time To Attack	Max Time To Attack
On Follow Waypoint [Unity Event]	Event called when AI follow a WAYPOINT path
On Follow AI Path [Unity Event]	Event called when AI follow a NAVMESH path
On See Target [Unity Event]	Event called when AI sees a target
On Stop Seeing Target [Unity Event]	Event called when AI stop seeing a target

Patrol AI's parameters are similar to Zombie AI's, so you can use the parameters above to configure Patrol AI in the same way.

How to use Vehicles AI

The Vehicle AI is very similar to the enemy AI in operation, it also uses Waypoint Path and calculates paths through NavMesh. The Vehicle AI can be used on any vehicle in the JUTPS Vehicle class.
There's no secret, just add the Vehicle AI script to your vehicle and configure the variables.



Vehicle being controlled completely by AI

See the parameters to configure below

Vehicle AI Parameters	Description
Enable Pathfinding	Enable NavMesh Pathfinding
Recalculate Path Refresh Rate	Time to recalculate path
Destination	Destination
Waypoint Path	Waypoint Path
Distance to Continue Path	Distance to go to next waypoint/path index
Vehicle Deceleration Intensity	Deceleration in curves
Front Check	Front sensor

Check Nearest Point On Path	Checks the nearest point on the waypoint/path and prevents the AI from going back to a previous Waypoint
On End Path [Stop, Reverse, Restart]	What the vehicle will do after finishing the path
On Start Path [Unity Event]	Event called when AI start path
On Following Path [Unity Event]	Event called when AI walk along the path
On Ended [Unity Event]	