JU TPS : Documentation

Hello! Thank you for purchasing my JU TPS: Third Person Shooter System. I hope you are satisfied with my work :)

You can play the free demos for Windows, WebGL(HTML5), and Android on Itch.io:
https://julhiecio.itch.io/unity-third-person-shooter-template

If you want to know more about my work:
http://www.youtube.com/c/JulhiecioGameDev

INFO

NAME OF TEMPLATE: JU TPS Third Person Shooter System

TEMPLATE VERSION: 2.0

OLDER COMPATIBLE UNITY VERSION: Unity 2019 LTS

VERSION RECOMMENDED: Unity 2020 LTS

EMAIL FOR SUPPORT AND QUESTIONS: JULHIECIOGAMES1@GMAIL.COM

HOW TO START NEW SCENE:

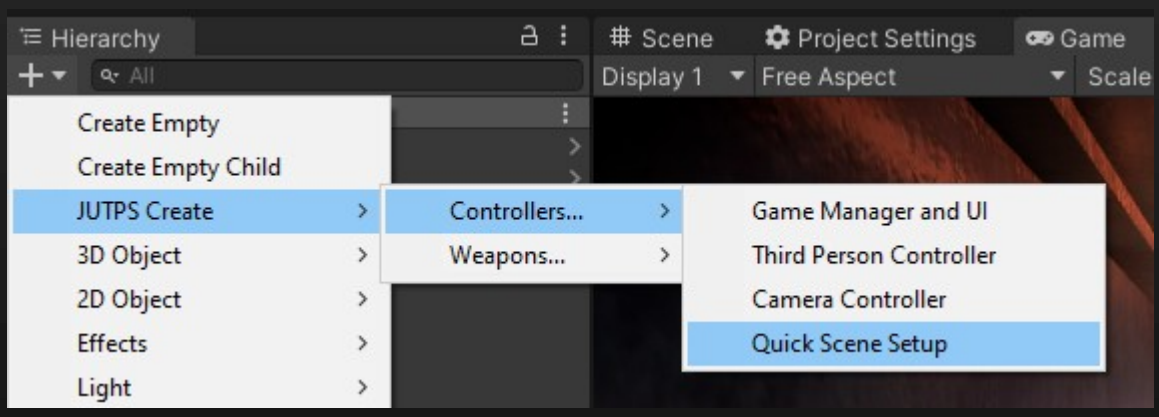
Playlist Tutorials:

https://www.youtube.com/playlist?list=PLznOHnSwmVcGcbDpXtElYKFVFYE9DvCgz

Above is a playlist with tutorials that can be useful, if you have a video suggestion you can send me an email.

I recommend checking the playlist first, as I can update the playlist more often.

Basically in two clicks you have your scene ready to start creating your game.



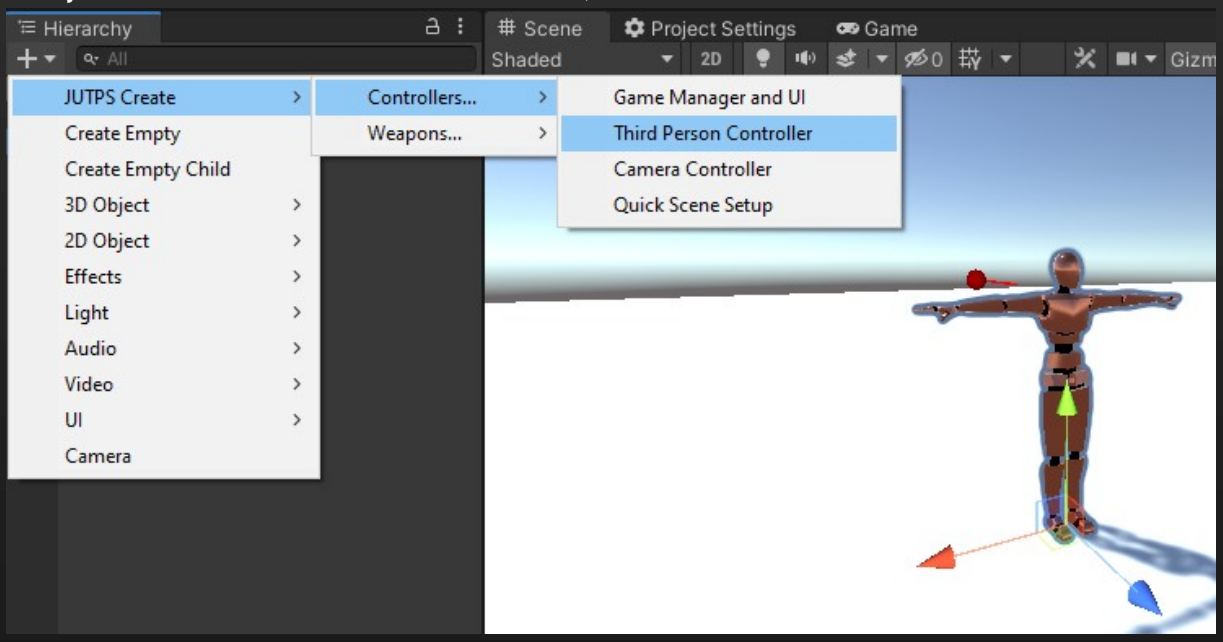
This function creates a standard JUTPS Camera Controller and Game Manager.
It is recommended to save the created Camera Controller as Prefab.
NOTE: there is already a Camera prefab with Post-Processing in the Prefabs folder.

Now you can drag and drop a character's Prefab onto the scene and play to see the game working.

HOW TO CREATE A TPS COTROLLER

⚠️ Attention: Your character must have a Humanoid rig, and an Upper Chest bone. If the code cannot find this bone automatically by the animator, you must link it manually in the “Third Person Controller” component.

Put your character's model on the scene, select it and click:



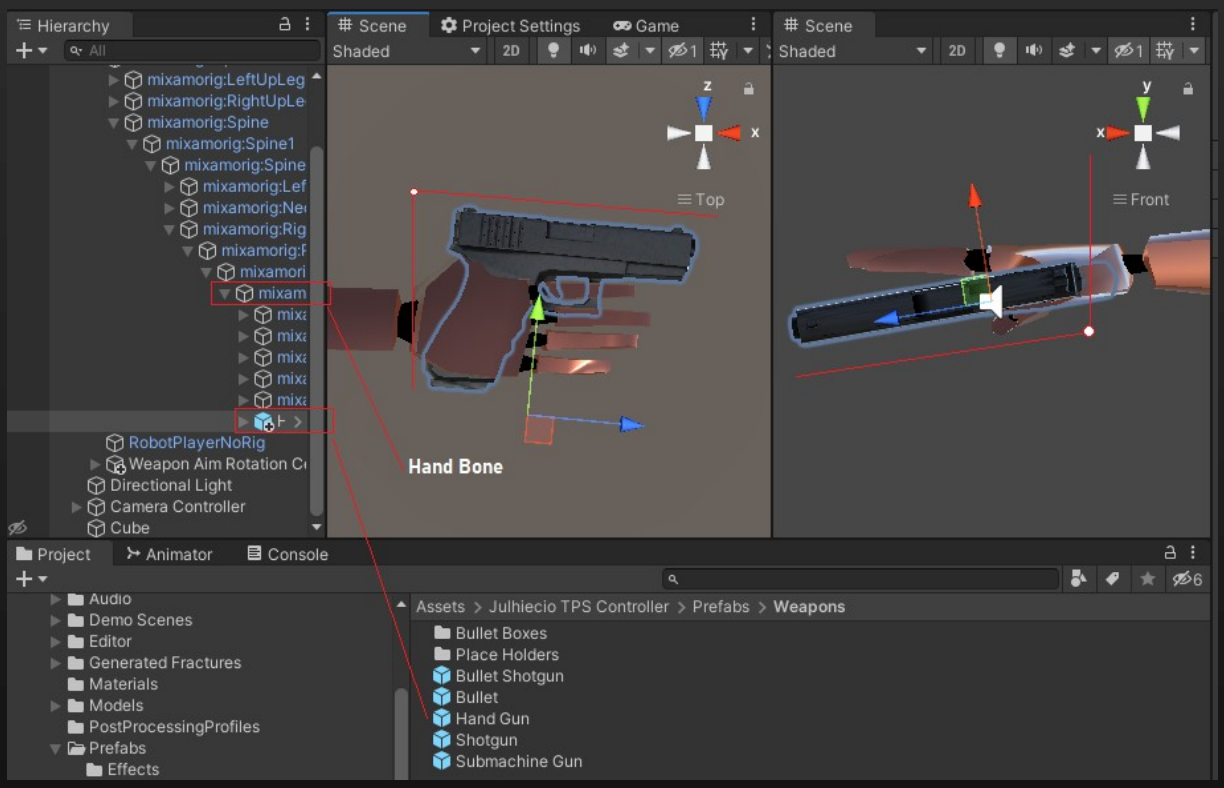
And ready, your character is already working with just two clicks.

NOTE: You may need to adjust the Capsule Collider.

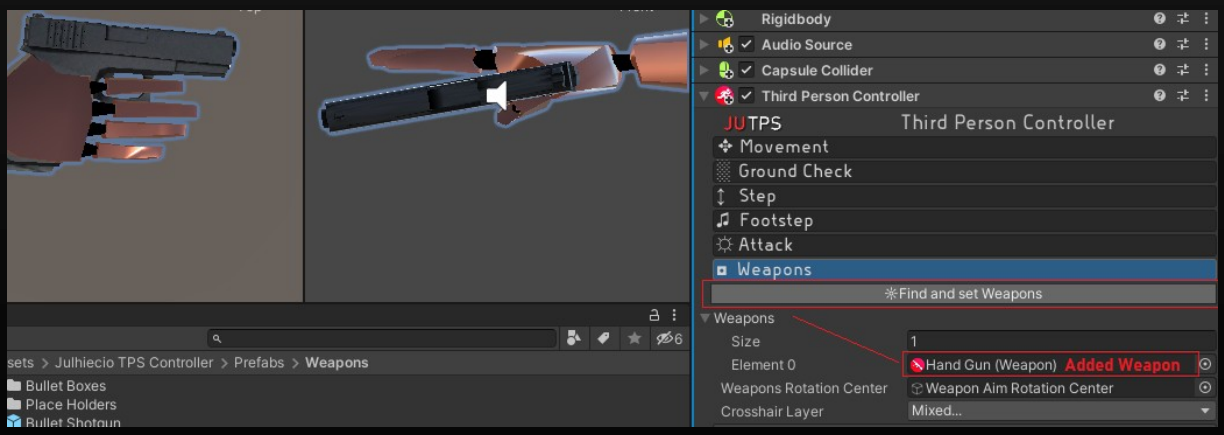
HOW TO ADD WEAPONS:

NOTE: this is for pre-made weapons, if you want to create a new weapon use the Weapon Creator tool to create a weapon.

Go to the Prefabs / Weapons folder and drag a weapon prefab to the character's hand bone, and check that the weapon is correctly in these angles:



Now go to your character's Third Person Controller component, in the Weapons tab, and click on “Find and set Weapons”.



And ready, your weapon was added, now just play to test it.

HOW JU TPS COMPONENTS WORKS:

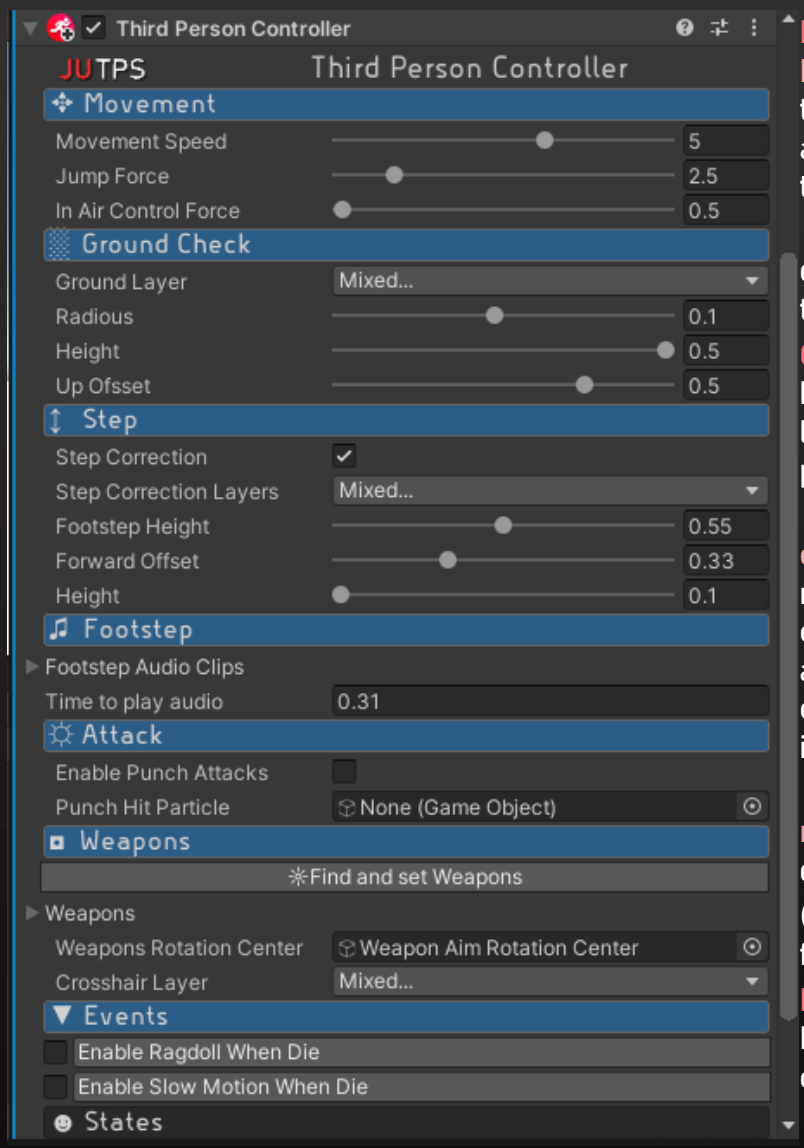
This topic shows what each script does and its functionality. (Not all variables, because most variables are self-describing).

So let's separate by script types:

TYPES:

Gameplay Third Person Controller Camera Controller Weapon AmmoBox Weapon Aim Rotation Center
Mobile Inputs Button Virtual Joystick Touchfield
Physics Bullet Vehicle Advanced Ragdoll Controller Destructible Objects.
UI Game Manager Mobile HUD Animator
Optimization: Pixel Quality Scale
Scene Manager Trigger Load Level Scene Controller

THIRD PERSON CONTROLLER



Movement/ In Air Control Force: Controls the force of the movement while in the air, if you don't want "air control" leave it at 0.

Ground Check/ Radious: radius of the box that checks the floor.

Ground Check/ Up Offset: height of the check box in relation to the position of the player.

Step/ Step Correction Layers:Layer Mask that separates what should have step correction, is recommended at specific times, such as climbing a ladder, otherwise it can cause flickering.

the rest below are adjustments of the step correction detections

(Use a player setting viewer for view this settings)

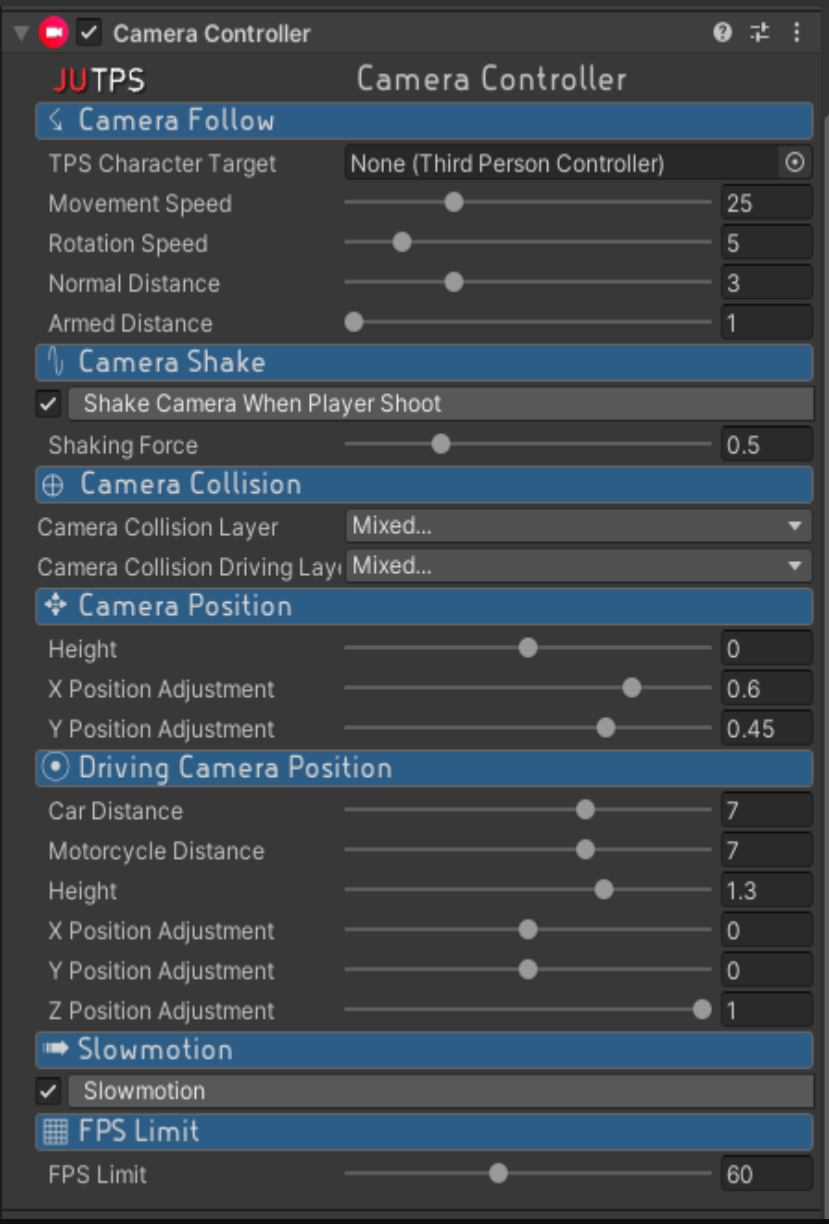
Footstep/ Time to play audio: It is the maximum time from one step to another.

Weapons/ ►Weapons: It is literally a list that should have all the weapons that the player can hold in his hand.

Weapons/ Weapons Rotation Center: It is a game object that serves as a pivot for the aim, inside it has other game objects that serves as a reference for the IK containing the position of the right hand for each type of weapon.

Weapons/ Crosshair Layer: It is a Layer Mask that separates what the ray that leaves from the camera can collide and cannot.

CAMERA CONTROLLER



Shake Camera When Player Shoot: As the name says, the camera shakes up a little when you shoot. It's great for feeling heavy.

Camera Shake Sensibility: is the sensitivity of the camera shake.

Camera Collision Layer: It is the Layer Mask that separates what the camera is going to collide in and what is not.

Camera Collision Driving Layer: It is the same thing as the variable above, except that when driving the camera you do not need to collide with the car.

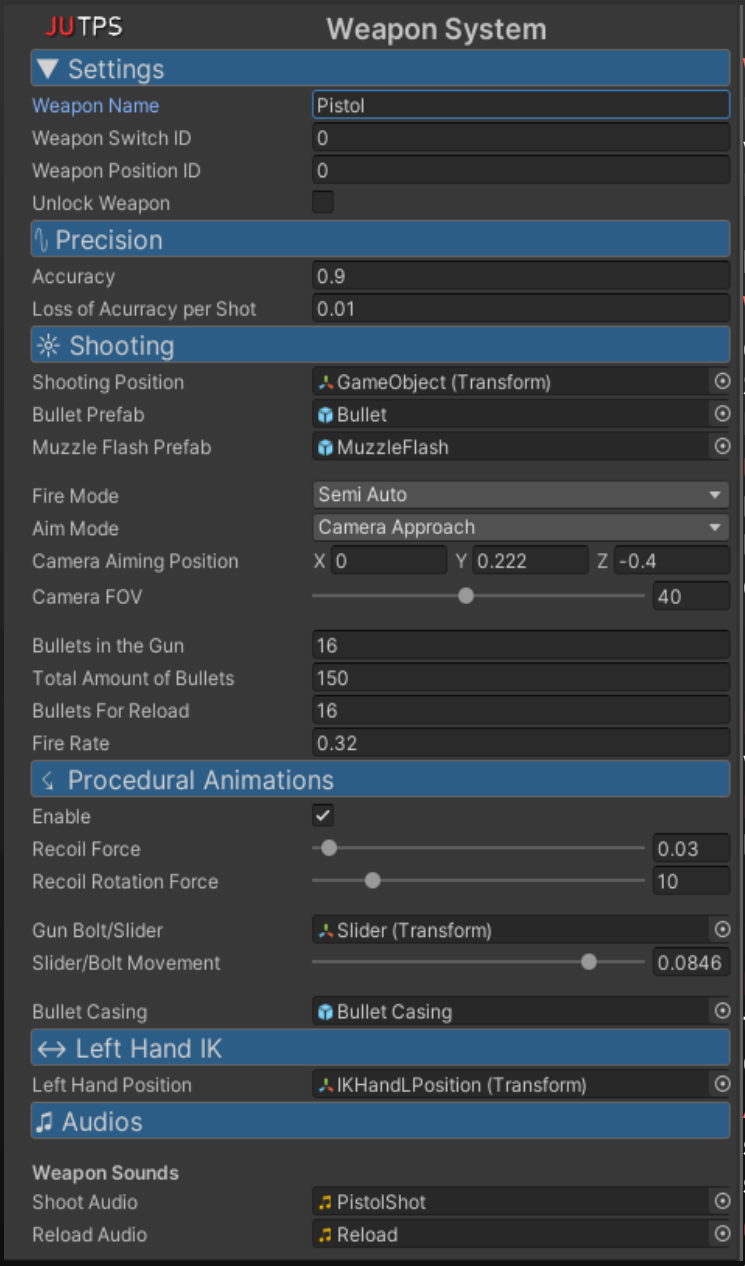
Height: It is the height of the camera, normally leave it at 0, since the camera copies the height of the Upper Chest Bone.

X Position Adjustment: It is the camera position adjustment on the X axis (Horizontal direction)

Y Position Adjustment: Even from the top only on the Y axis (Vertical direction)

Driving Camera Position: In this area of the script is where the camera is configured when the player is driving, it follows the same pattern as the variables above.

WEAPON



[Settings]

Weapon Switch ID: It is the weapon ID, it has to be in the correct order in which it is in "Weapons []" in the Player Controller, in the case [0,1,2,3 ...], the same weapon cannot have the same ID.

Weapon Position ID: It is the position of the weapon according to the Switch ID component Weapon Aim Rotation Center.

Unlock Weapon: If the weapon is unlocked, the player will be able to select it at any time, otherwise it will only be found on the ground.

[Precision]

Precision: It's the precision of the weapon.

Loss Of Accuracy Per Shot: It is the loss of accuracy for each shot.

[Shooting]

Fire Mode: Type of shot, whether automatic, semi-automatic, bolt action or shotgun.

Aim Mode: Whether it's aiming with a sniper or looking through a gun sight.

Camera Aiming Position: It is the position of the aiming camera relative to the weapon.

Camera FOV: It is the field of view of the aiming camera, you can decrease it to give a zoom effect.

Bullets in the Gun: It's the total number of bullets in the gun.

Total Amounts of Bullets: It is the total number of bullets stored.

Bullets For Reload: is the number of bullets per magazine, is the number of bullets that will be reloaded.

[Procedural Animation]

Enable: When checked, the recoil animation will be animated procedurally with IK, and the variables below control the strength of that animation.

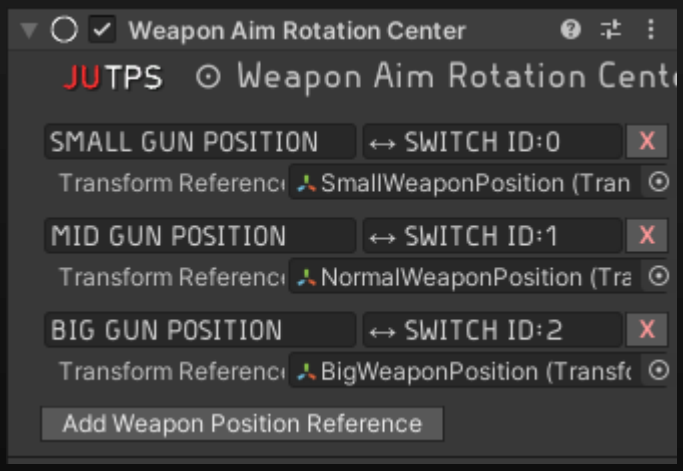
Gun Slider: Weapon slider, also used as a reference position for instantiating Bullet Casing.
Bullet Casing Prefab: It is the Bullet Casing prefab that will be instantiated at each shot, in the Slider position.

[Left Hand IK]
Left Hand Position: It is the position where the left hand will be on the weapon, it is an empty game object.

WEAPON AIM ROTATION CENTER

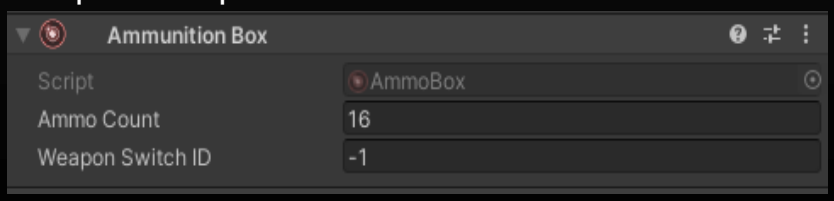
This Component stores the position and rotation information for each weapon or type of weapon, you can easily add and delete this information.

If you want a weapon to have a specific position, click on “Add Weapon Position Reference” and add a transform with desired rotation and position to the “Transform Reference” field, and set the Weapon Position ID value to the same value as the Switch ID that created.



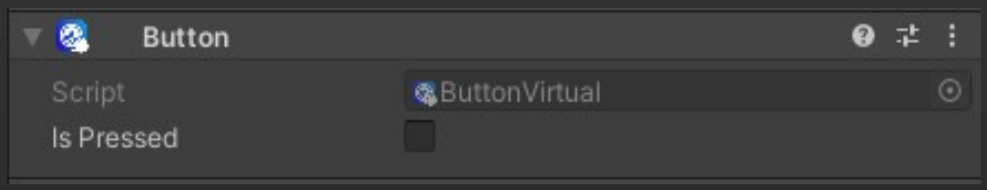
AMMUNITION BOX

It is the script placed in an ammunition box, it allows to place ammunition for any weapon or a specific one.




```
(Weapon Switch ID == -1) = Any Weapon
(Weapon Switch ID >= 0) = Weapon this Weapon ID
```

BUTTON



It is a script that basically serves to transform any UI image into a button, you can perform actions by scripts easily with one:

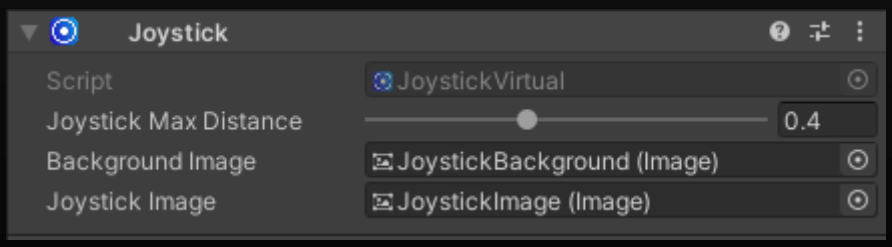
```
If(Button.IsPresses == true)
{
     Debug.Log("Pressed the button");
}
```

TOUCHFIELD

It is a script that serves to take the position of the touchscreen, making it possible to easily create a camera rotation with a touchscreen:

```
Xrotation = Touchfield.TouchDistance.y / 10;
Yrotation = Touchfield.TouchDistance.x / 10;
```

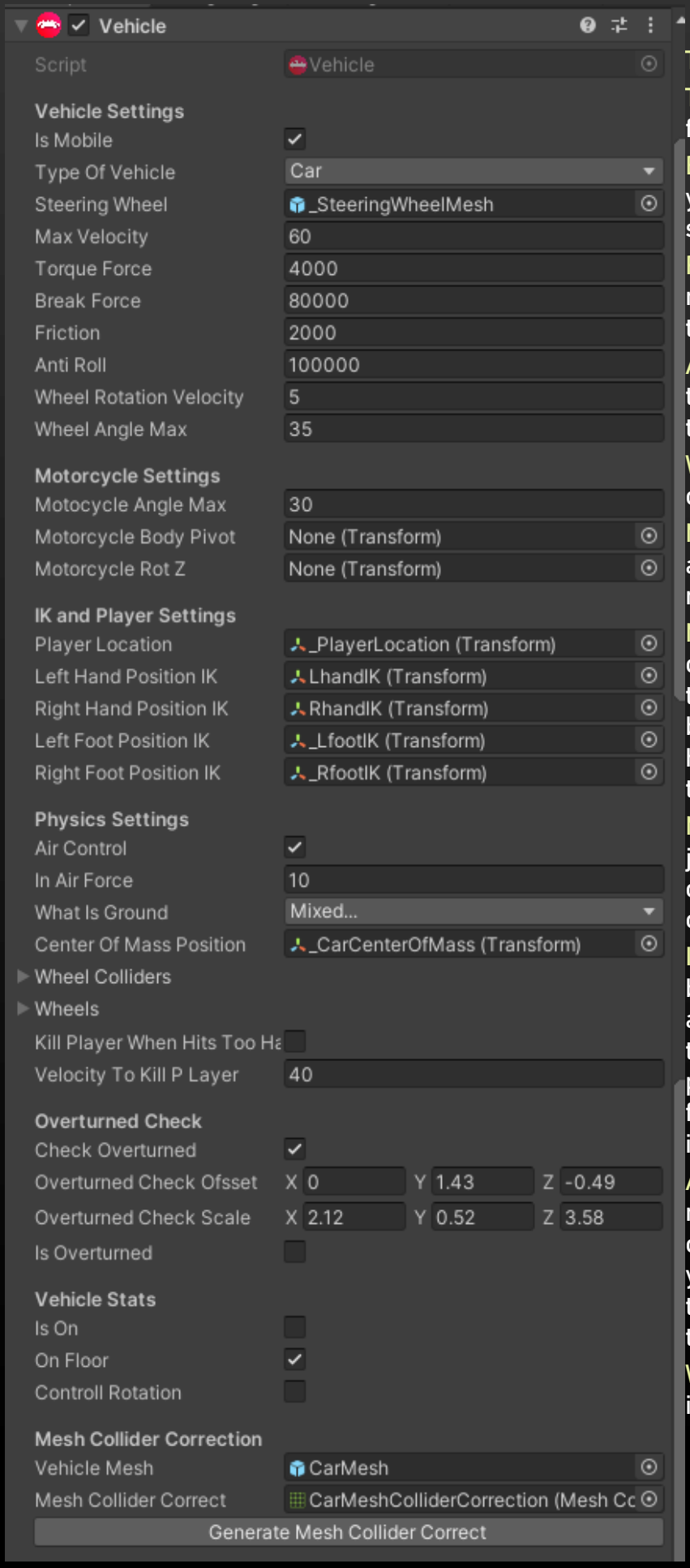
JOYSTICK



It allows you to easily create a Virtual Joystick. (NOTE: The rect canvas must have the Pivot (X = 1, Y = 0))

```
horizontal_input = Joystick.InputVector.x;
vertical_input = Joystick.InputVector.y;
```

VEHICLE



Type Of Vehicle: Car or Motorcycle

Torque Force: is the acceleration force of your vehicle's engine.

Brake Force: it's brake force and yes I confused break with brake, sorry I'm Brazilian.

Friction: the friction is directly connected with the Brake Force, it is the hardness of the brake.

Anti Roll: The higher this number, the less likely the car will overturn.

Wheel Angle Max: Maximum angle of rotation of the wheels.

Motorcycle Angle Max: Maximum angle that the bike will tilt when rolling.

Motorcycle Body Pivot: It is a game object that serves as a pivot for the rotation of the motorcycle body, it adapts to the surface and has a direct influence on the rotation of the motorcycle.

Motorcycle Rot Z: is the game object that must be inside “Motorcycle Body Pivot”, this game object controls the inclination of the bike.

IK and Player Settings: this area is basically gameobjects that serve as a target for the IK, this serves to position the entire body of the player in their proper positions: feet under, and hand on the steering wheels.

Air Controll: Do you want to maneuver with your vehicle? Leave it checked. With this you can rotate your vehicle in the air according to the "In Air Force" that will control the rotation force.

What Is Ground: Layer Mask that identifies what the floor is.

Center Of Mass Position: It is an empty game object that passes the coordinate of the center of mass of your vehicle, this will directly affect the physics of your vehicle.

Wheel Colliders: is the list of all Wheel Colliders, which are components integrated into Unity that simulate wheel physics.

Wheels: It is the wheel models of your vehicle, which will receive all data from the Wheel Collider.

Kill The Player When Hits Too Hard: It's a lot of fun to fall on a motorcycle with this with the ragdoll enabled.

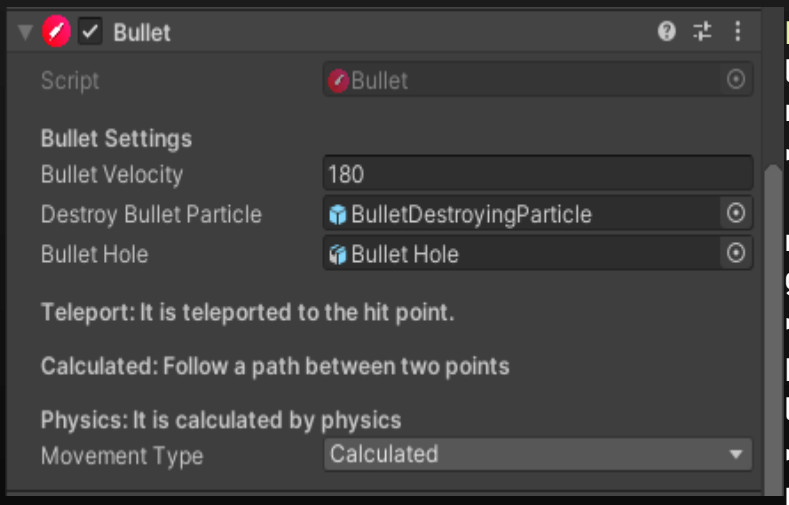
Overturned Check[HEADER]: This checks if the vehicle has overturned and automatically adjusts its speed, if you want you can deactivate it and create a manual decapping system.

Vehicle States [HEADER]: This area shows the vehicle status in the inspector.

Mesh Collider Correction [HEADER]: Unity does not support Mesh Colliders not convex since Unity 5, so I created this as a form of Mesh Collider correction, just link the mesh of the vehicle and click on "Generate Mesh Collider Correct".

With this you will sacrifice some reactions of physics, however the bullet holes will be instantiated correctly.

BULLET



Movement Type: It is how the bullet will move, it has three types of movement:

► Calculated (recommended):

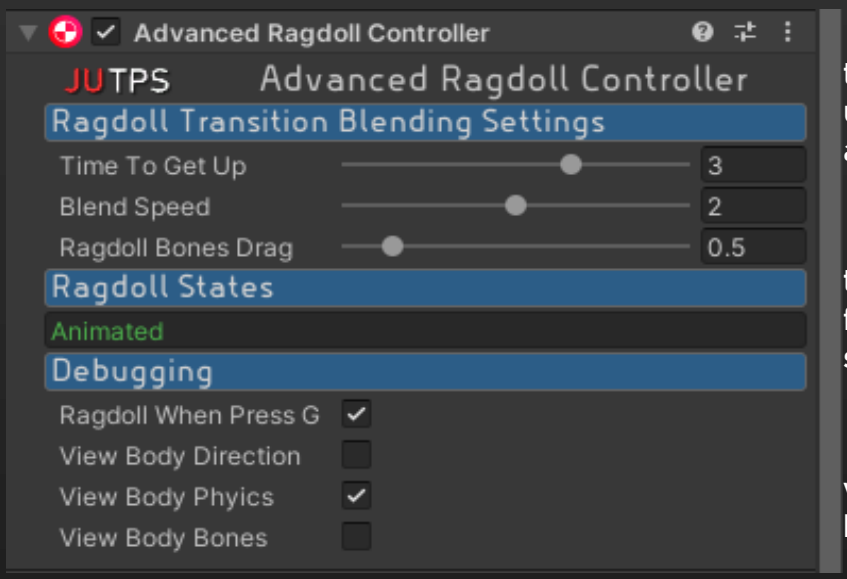
This will calculate the entire movement of the bullet to the target.

► Physic:

It will physically calculate the bullet path

► Teleport: teleports the bullet projectile to the target (realistic).

ADVANCED RAGDOLL CONTROLLER



Time To Get Up: It is the time that the character takes to start to get up after having fallen ragdolled and to become static.

Blend Speed: It is the speed of transition between the position of falling on the ground and the position of the animation rising

Ragdoll Bones Drag: It is the value set in Drag of each Rigidbody of each bone.

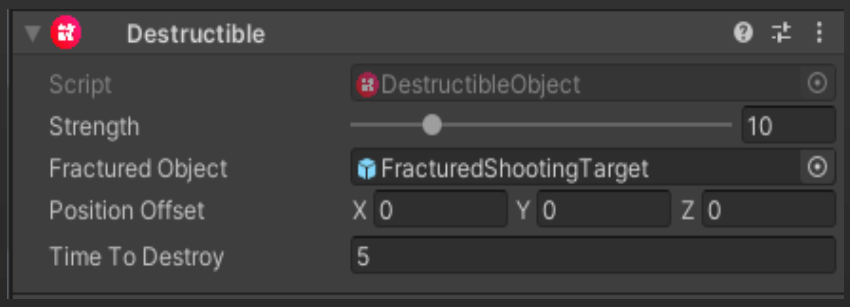
Ragdoll States: Shows the status of the character, whether it is ragdolled or animated.

Debbuging: Shows options used to adjust the ragdoll, test or view the states in gizmos.

How to use in code:

```
AdvancedRagdoll.SetActiveRagdoll(Active?, use Inertia?);
```

DESTRUCTIBLE

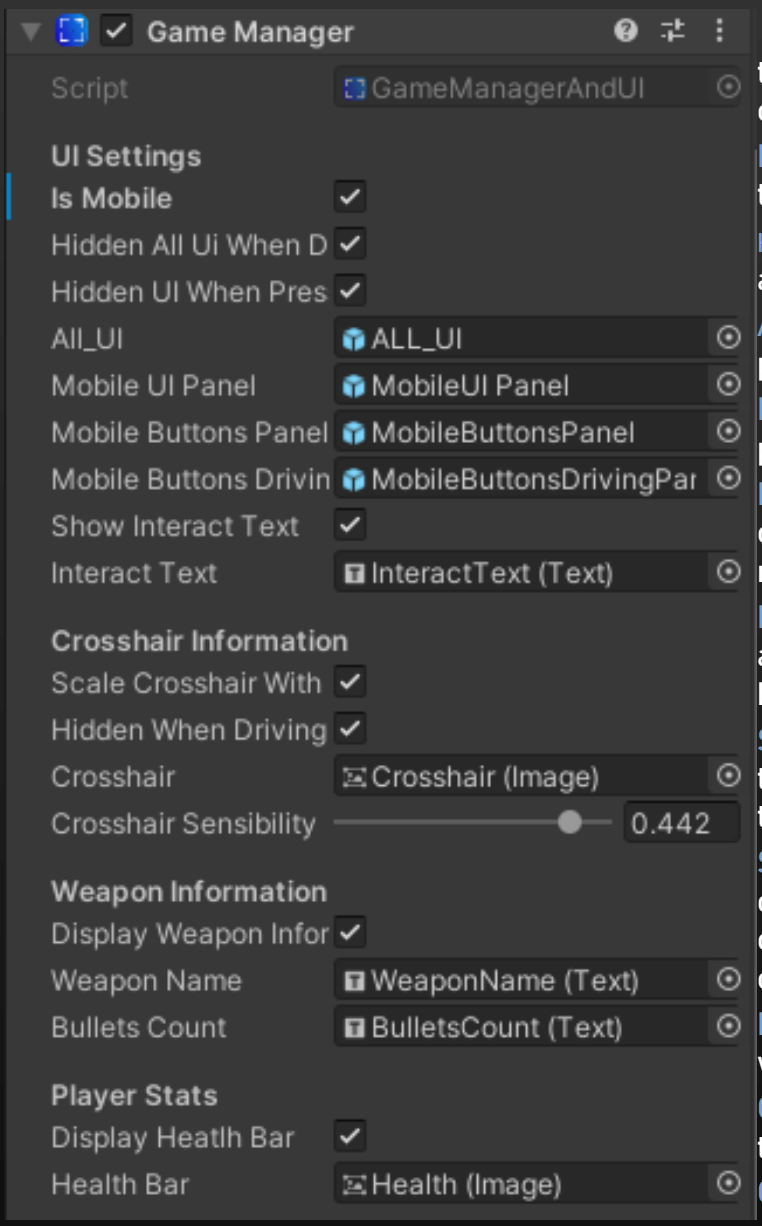


Strength: It is the object's strength, the smaller the easier it is to break by colliding with an object at high speed.
Fractured Object: It is the prefab of the fractured object.
►To fracture a object in JUTPS Template, add a Fracture Tool component to the object and click on "Generate Fractured Object" and if it's OK click on "Save Generated Mesh as Asset". Add a rigidbody and mesh collider to each fracture of the object and save as a prefab.
►To create a fractured object just fracture an object in Blender with the Cell Fracture addon, export to Unity, add a Convex Mesh Collider and Rigidbody to each fracture of the object, and that's it, just save the prefab.

Position Offset: This is useful if you need to adjust the position of your fractured prefab.
Time To Destroy: Time to destroy the fractures created.

```
DestructibleObject._DestroyObject();
```

GAME MANAGER



IsMobile: when checked it will activate the mobile panels, with the control vehicle control buttons.

Hidden All UI When Die: This will disable the entire HUD when the player dies.

Hidden All UI When Press f2: even from the above variable only by pressing f2.

ALL_UI: It is a parent panel of all UI components.

Mobile UI Panel: is the mobile control panel, every mobile control is his child.

Mobile Buttons Panel: It is the panel that contains the buttons and joystick for moving the player.

Mobile Buttons Driving Panel: the same as above, only with the vehicle control buttons

Show Interact Text: Shows the interaction text on the screen that says “Press [F] to ...”

Scale Crosshair With Precision: When checked this will change the aim size according to the accuracy, the lower the accuracy, the greater the aim.

Hidden When Driving:Hide the crosshairs while driving a vehicle.

Crosshair: crosshair UI image in the center of the screen.

Crosshair Sensibility: The greater the greater the change of scale in the sight.

Display Weapon Information: shows information for the selected weapon.

Weapon Name: Informational text of the weapon name.

Bullets Count: Informative text of the number of bullets in the weapon and the number of bullets stored.

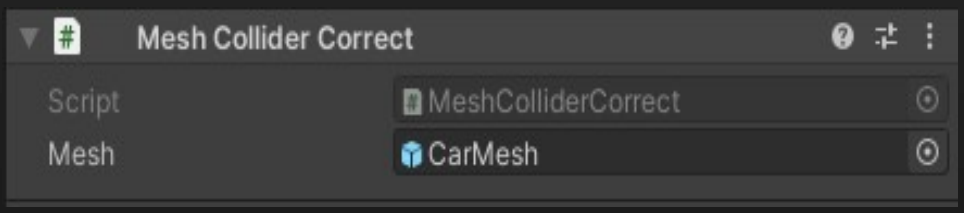
Display Health Bar: When checked it will show the amount of life in a life bar.

Health Bar: Filled type UI image, the UI Gameplay Manager uses the `.fillAmount` according to the amount of life of the player.

PIXEL QUALITY SCALE

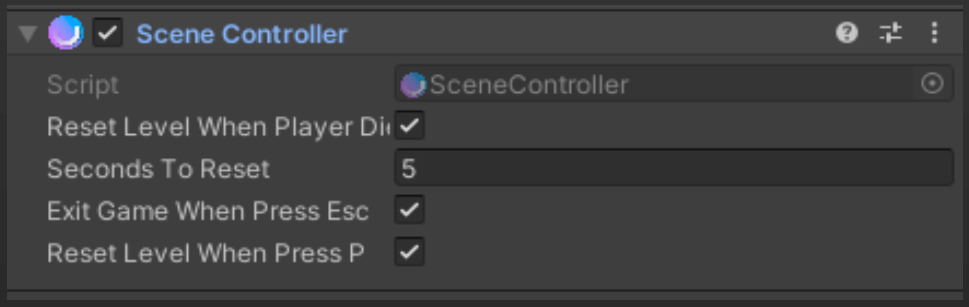
This will slightly decrease the game's resolution, consequently improving performance, I recommend using it on mobile platforms and leaving it at 1.2 to 1.4, after that the image will lose a relevant amount of quality.

MESH COLLIDER CORRECT



Mesh: The vehicle's original mesh, this will prevent the bullet holes from bugging while the car is in motion. The Mesh Collider Correct is updated physically and not every frame, this will make the bullet holes hit end up being left behind when the vehicle is in motion, and this script serves to fix this.

SCENE CONTROLLER



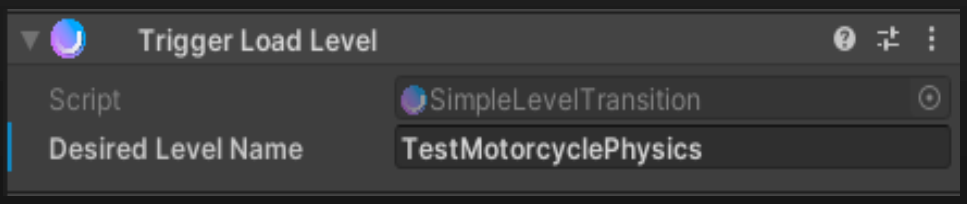
Reset Level When Player Die: will reset the loaded scene when the player dies.

Seconds To Reset: Time to reload the scene after the player's death.

Exit Game When Press Esc: Close the game when pressing Esc, I recommend deselecting it and creating a menu with the option to exit the game or continue playing.

Reset Level When Press P: Reset the scene when you press the letter P, I recommend leaving it unchecked in the final build of the game and only leaving it to reset when the player dies.

TRIGGER LOAD LEVEL



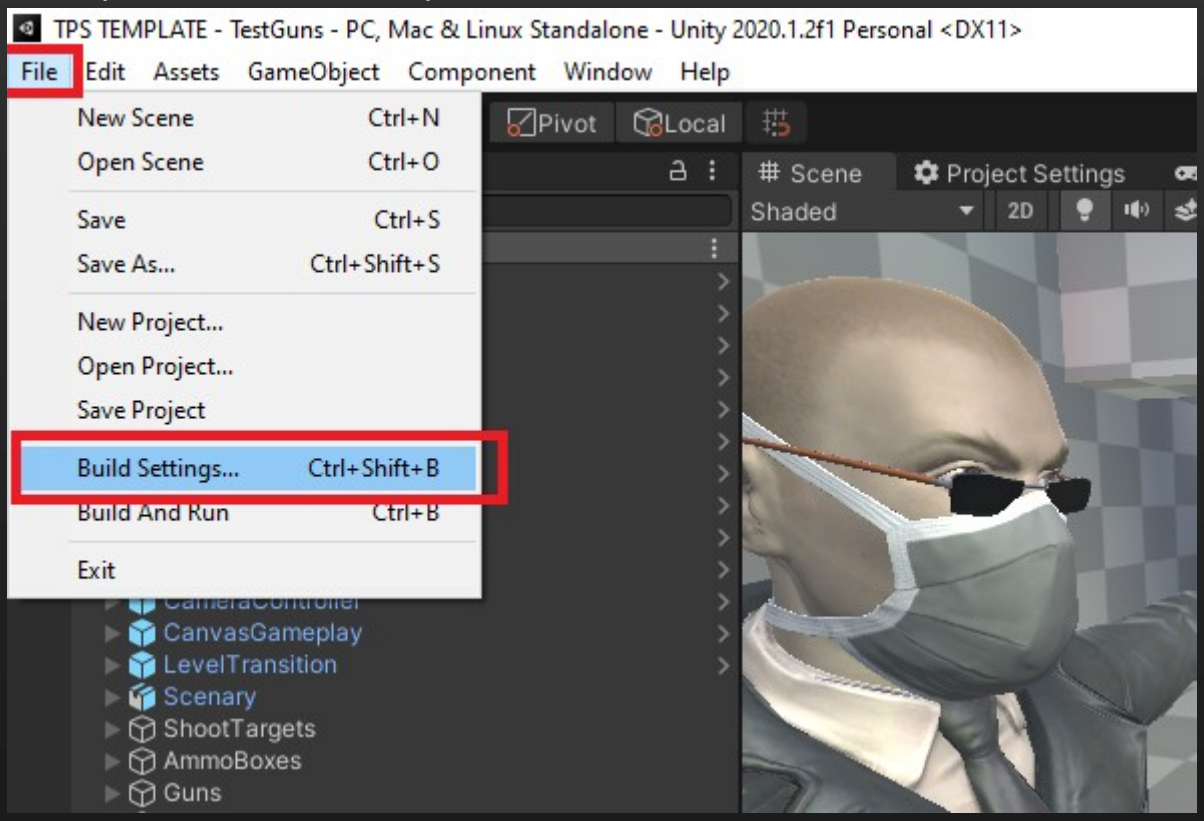
Desired Level Name: Name of the scene that will be loaded when the player collides with a collider marked as Trigger next to that script.

Note: This script needs to be placed on an object with a collider marked in Trigger to work.

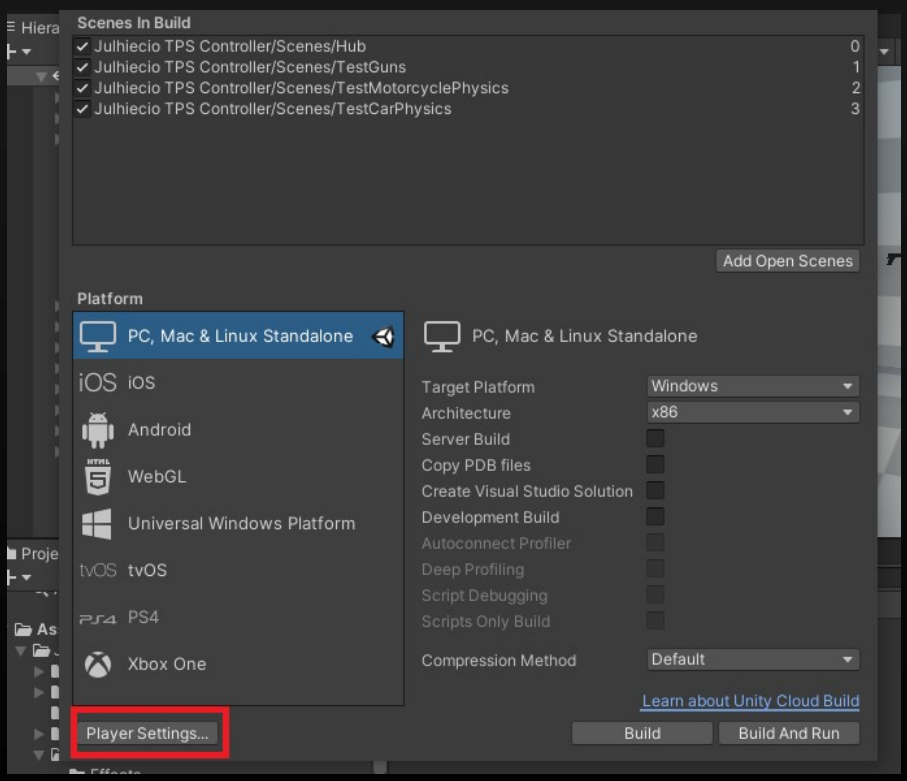
Okay, now that you know the functionality of all components, let's move on to the next topic:

HOW TO BUILD:

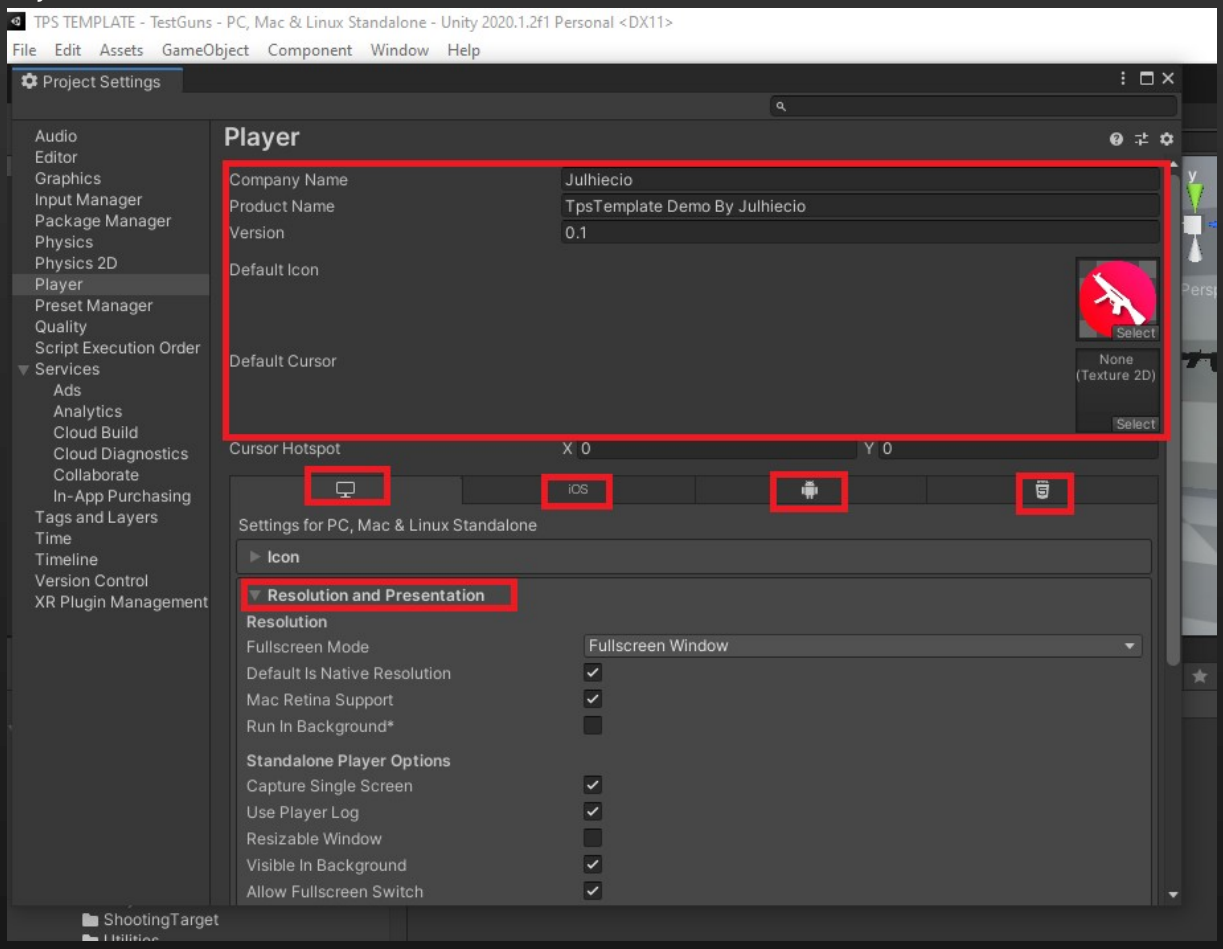
It's simple, click FILE in the top menu, and click BUILD SETTINGS.



After that, click on Player Settings in the bottom left corner of the window.

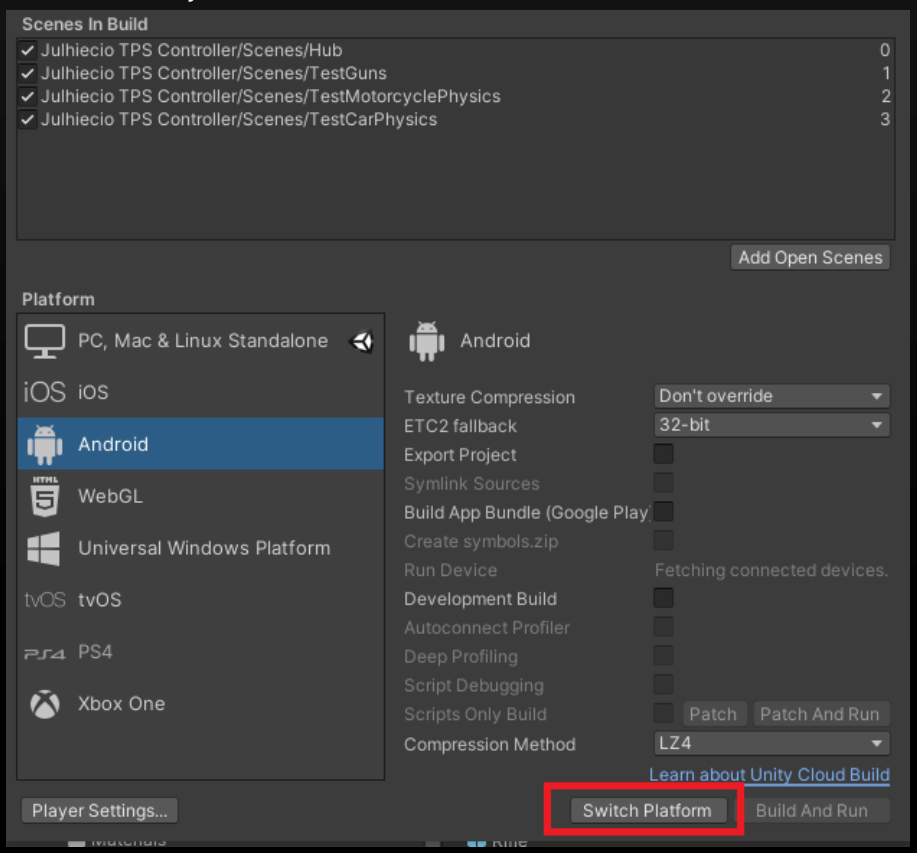


and configure your game: Company Name, Product Name, Icons, Splash Screen, Category and etc.

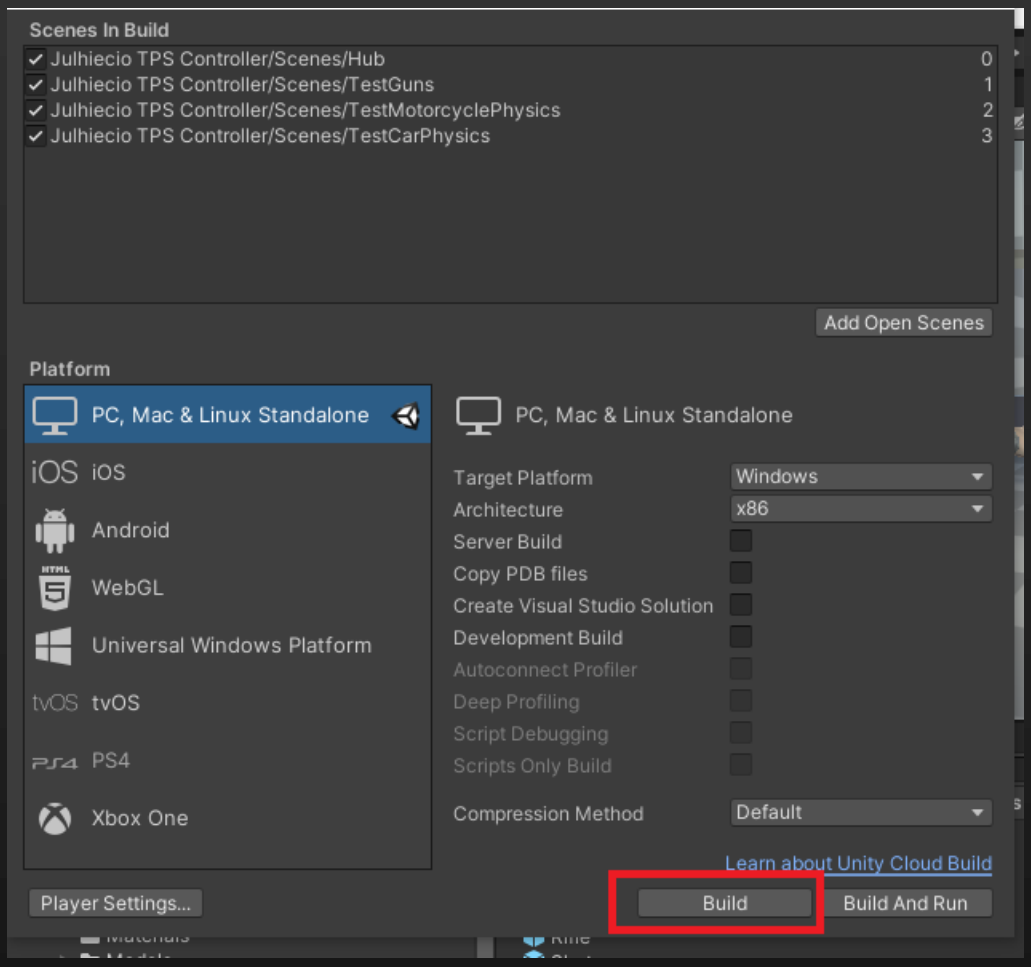


NOTE: you have to configure for each platform

After that, go back to FILE> BUILD and choose the platform you want to publish. If it is not already selected, click on “Switch Platform”.



Then just click on "Build", choose the folder and wait for Unity to create the executable for your game.



It's done!

I hope you are enjoying the Template, and thanks again. If you have any questions you can contact me.