# Player-Owner's Manual

## brief

The player is an important concept of the game. The default player already has basic functions such as blood, movement, backpack, etc. By editing the player, you can customise the player's functions, attributes, or appearance and movement performance. In this article, we will introduce the important concept of player from two aspects: player module and player script.

The player module allows you to set up the player before the game runs.

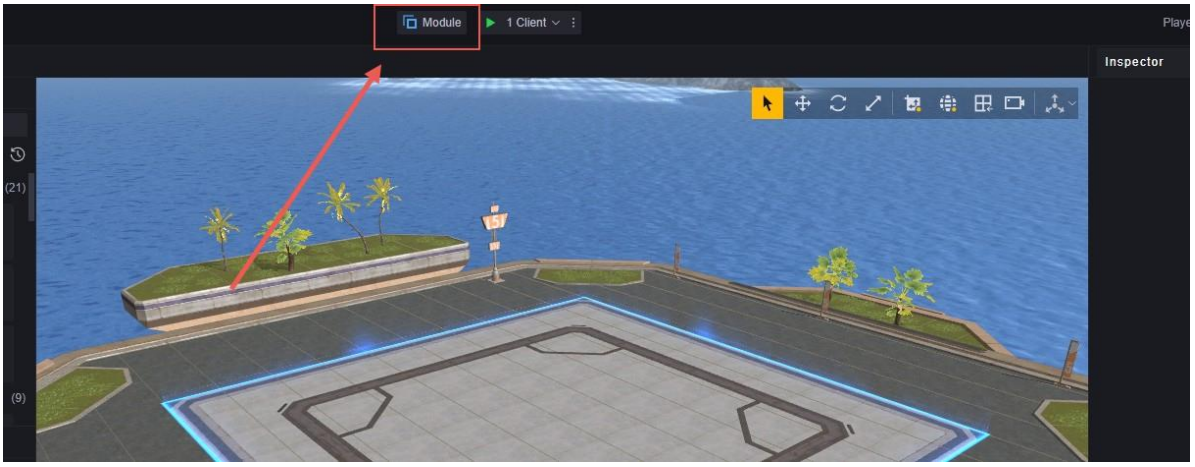
Player scripts allow for adjustments to be made to the player while the game is running.

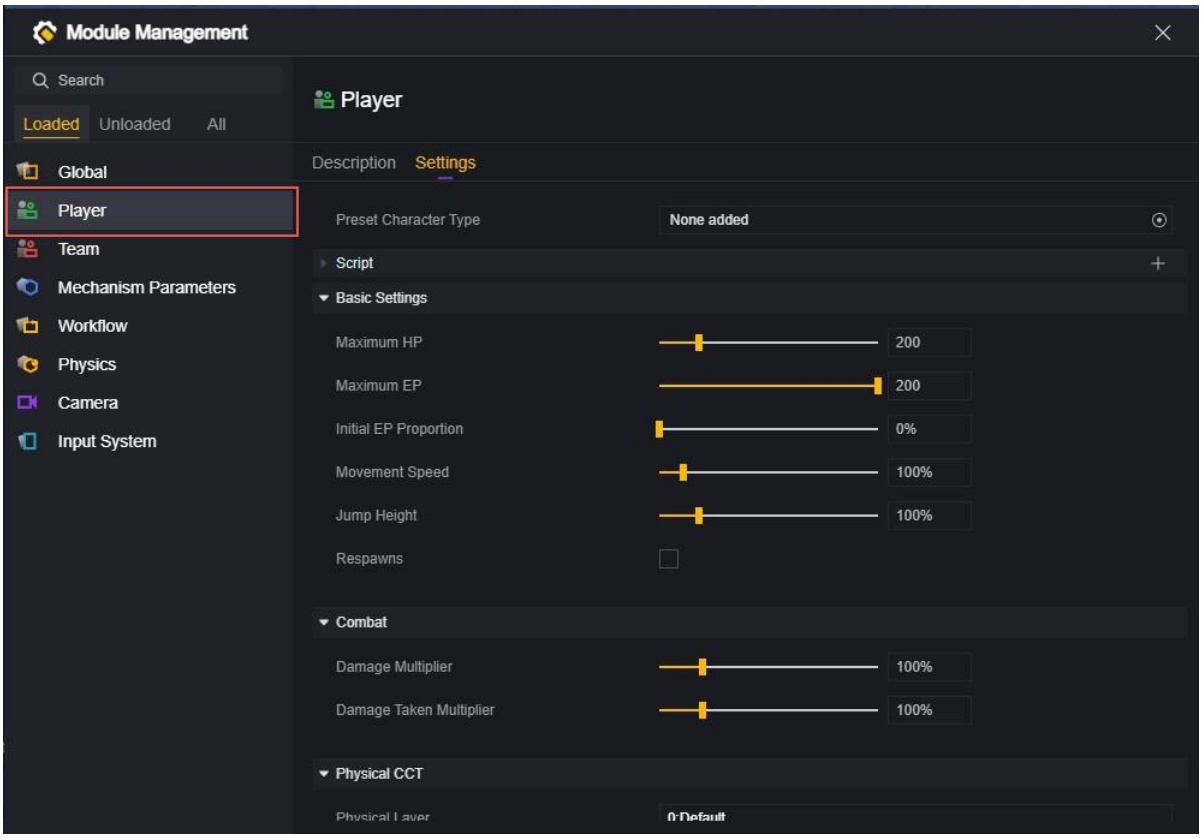
> Players exist in the game as instances of player entities, which means that edits in the module are mapped to each created instance of a player entity, and for scripts it's important to note whether they are scoped to "a specific player" or "all players".

## Gamer Module

The player module allows you to set various attributes of the player so that the player can start the game in the state you want. The player module corresponds to the player entity, and changes made here will take effect for every player created in the game.

Go to the Player menu in Editor - Modules:



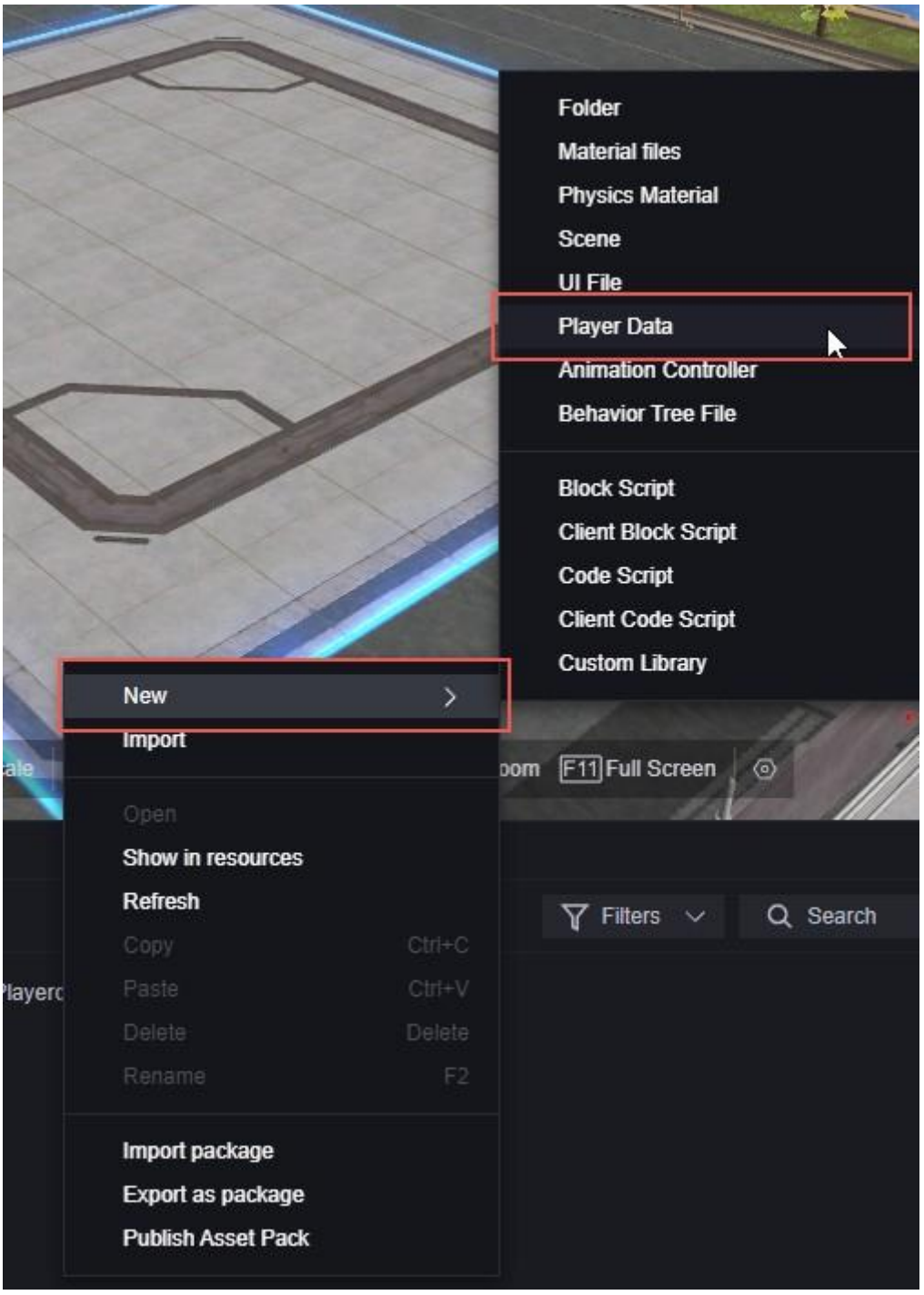



# Preset Player Types

At this setting, you can use custom player resources instead of the default image. This means that you can make the player appearance different, as well as use custom actions in part or in full.

## How to create a player data:

Create a new player data file at Engineering - Assets:



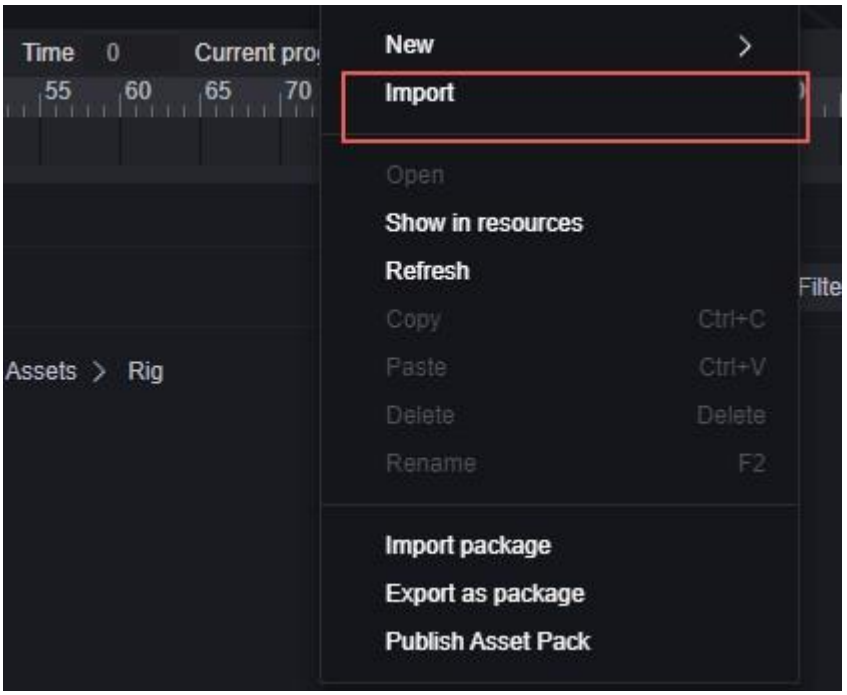

The player data file contains the model and action resources used by the player.

## How to edit player data:

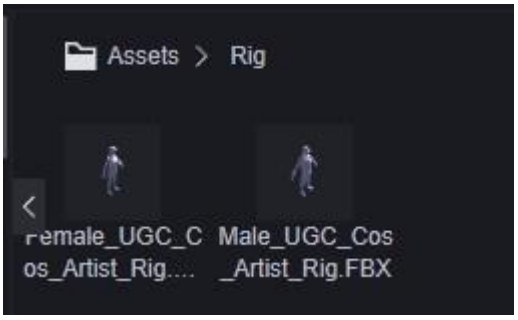
First of all, you need to prepare the required model and action resources. It is recommended to use the same character model as the one located in your local editor path.

\resources\LocalData\Utilities\UGC_Cos_Artist_Rig in the default model to match the bones.

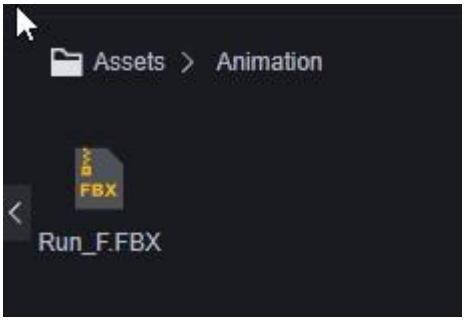
Add resources:



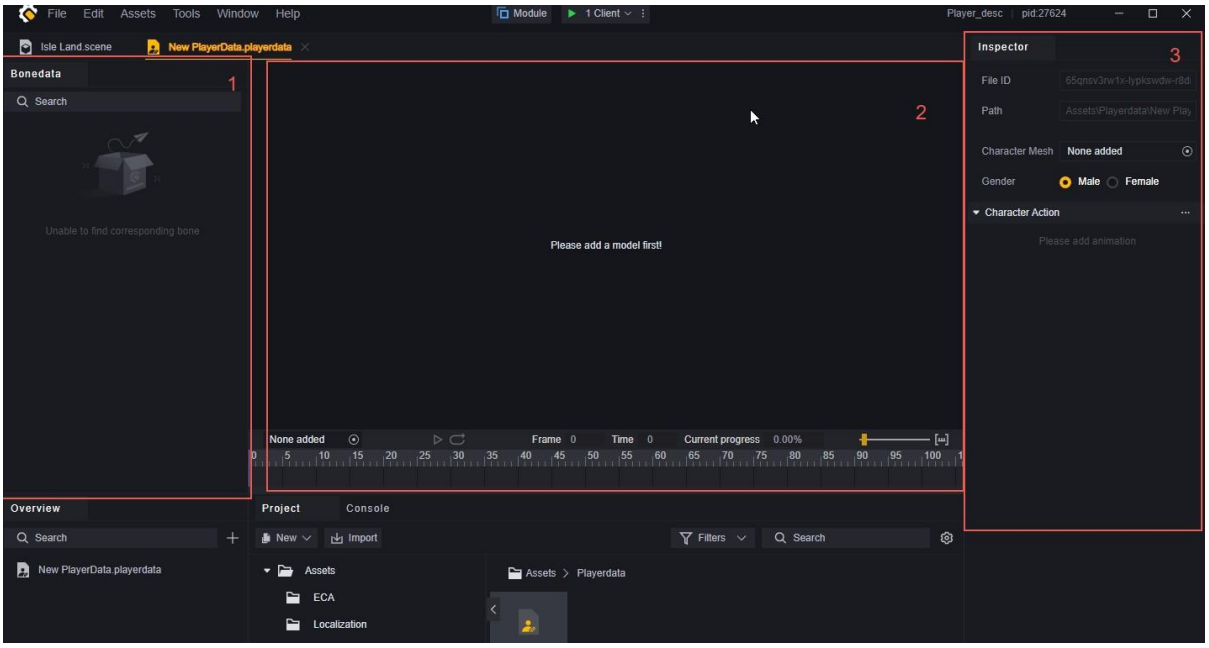

Role Modelling (example):




Character Actions (example):

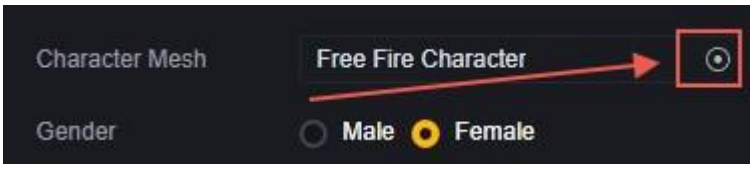


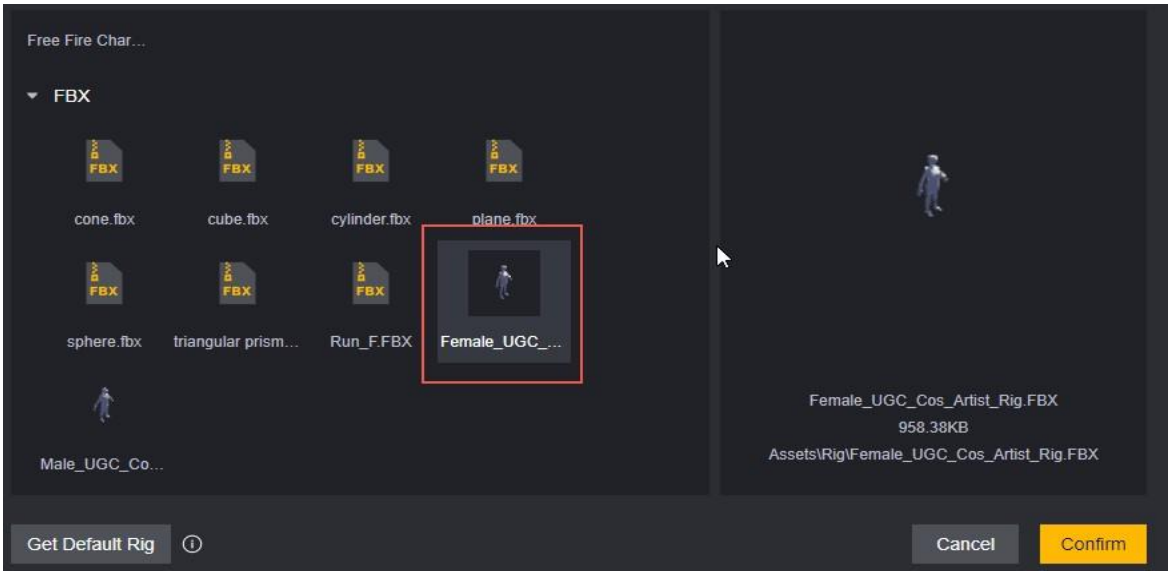

Double-click the newly created player data file to open the edit page:



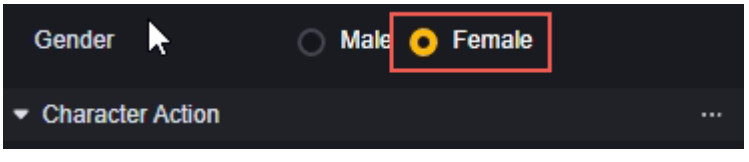

1. Skeletal data
2. preview window
3. Edit Window

Firstly, in the 3. Edit window, you need to select the character model you need and select the gender.




> It is also possible to quickly modify the model used by dragging the model from the asset screen to the 2. preview screen.

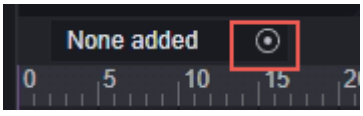


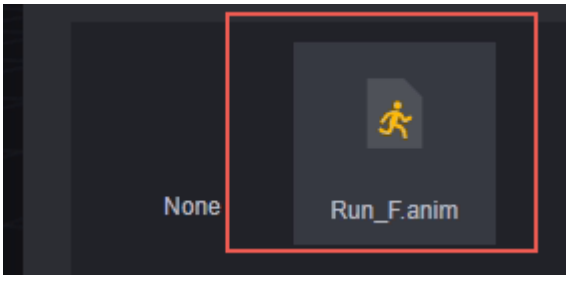

Here we are using a female model as an example, so choose female.

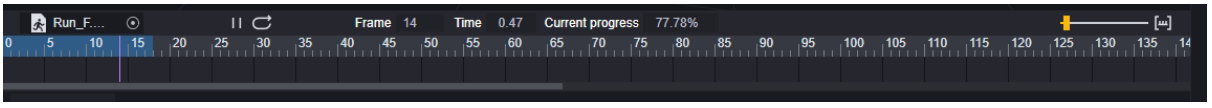
After adding the model, you can already see what the model looks like in the 2. preview screen. Here is just a white model because it is for demo.

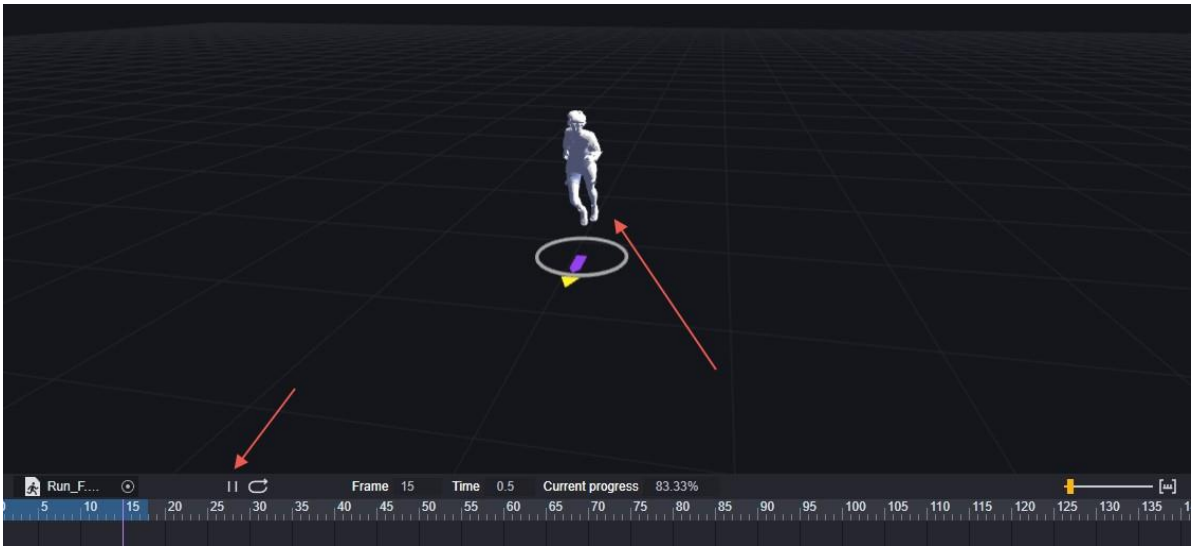
You can zoom in and out with the scroll wheel, and rotate the camera by holding down the right button to see model details.

Add the action you want to preview with this model in the bottom left corner to see how the action will look on the model:



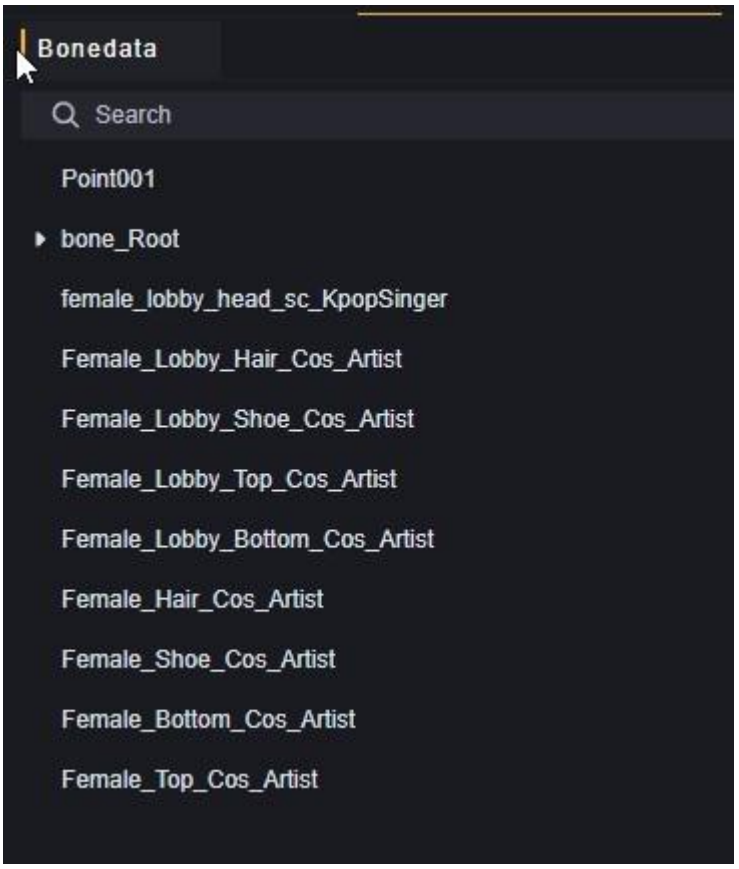



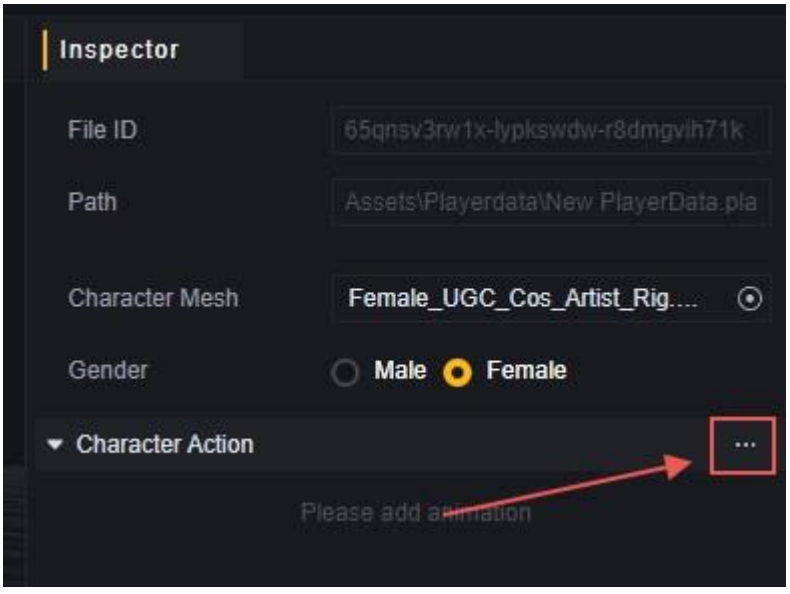

The action preview screen below now shows information about the action, which can be previewed by clicking play:

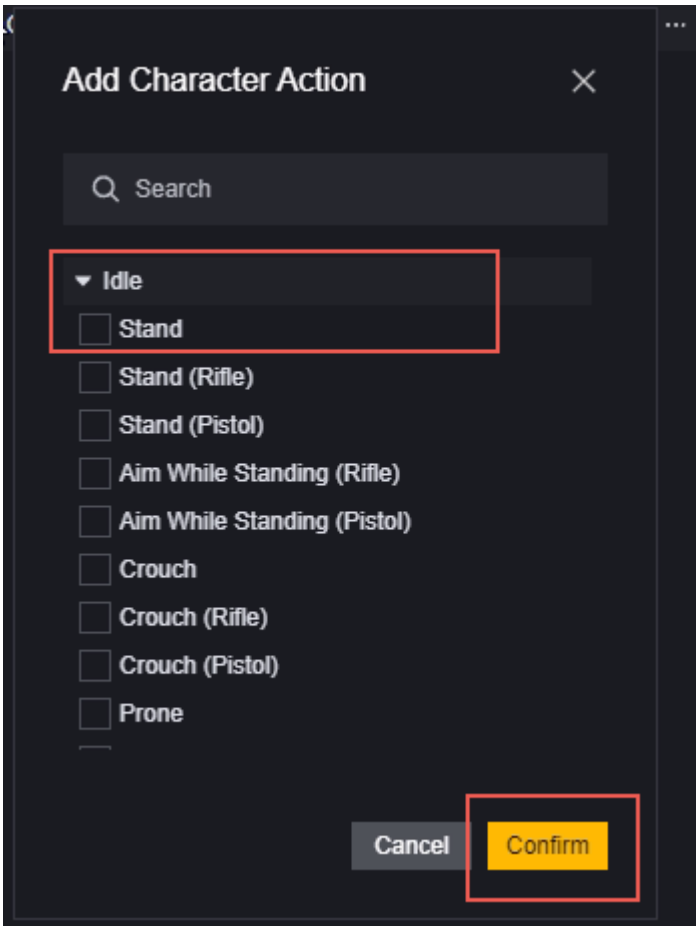
The skeletal structure of this model can be viewed in 1. Skeletal Data:

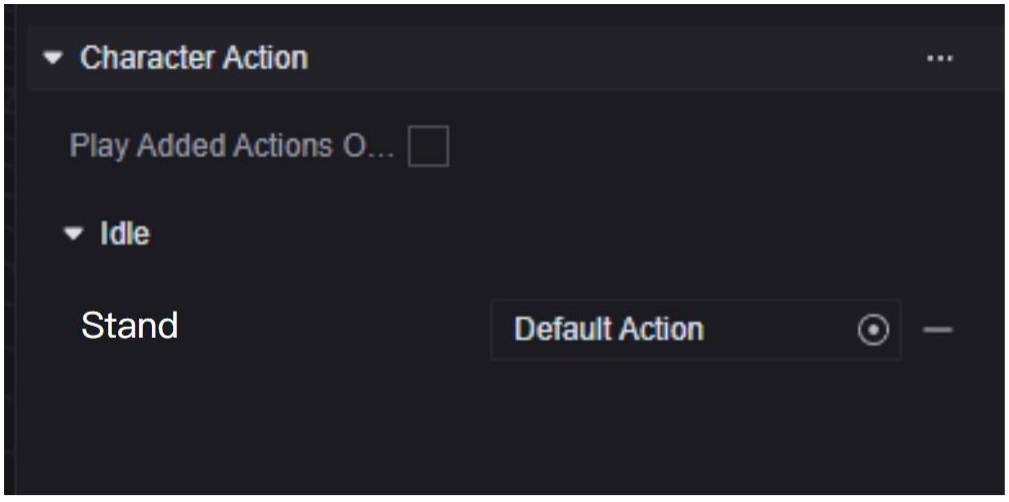



Once you have prepared your model, you can add the kind of movement you want to modify in the Character Movements window in the 3. Edit window, such as modifying Running and Standing. Movements that are not modified will play the default animation.

As a demo, we add a standing idle action:

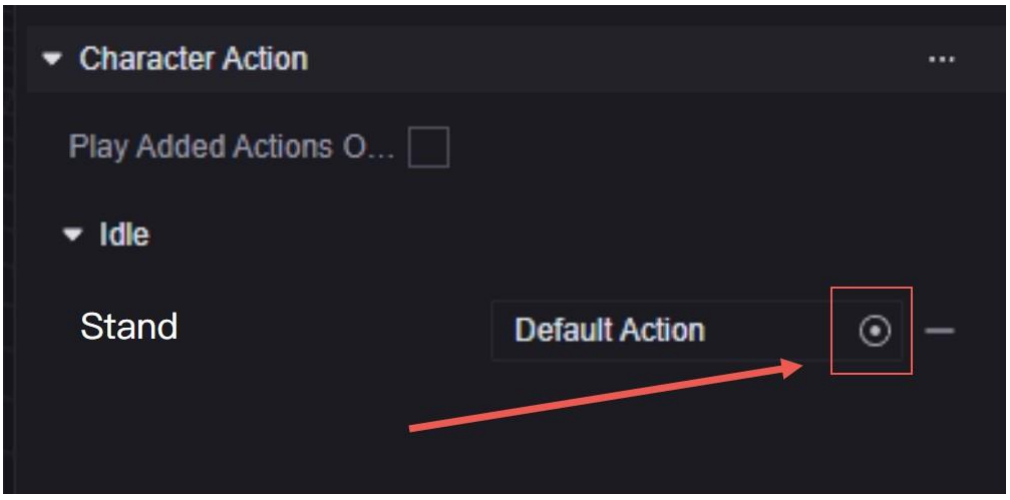


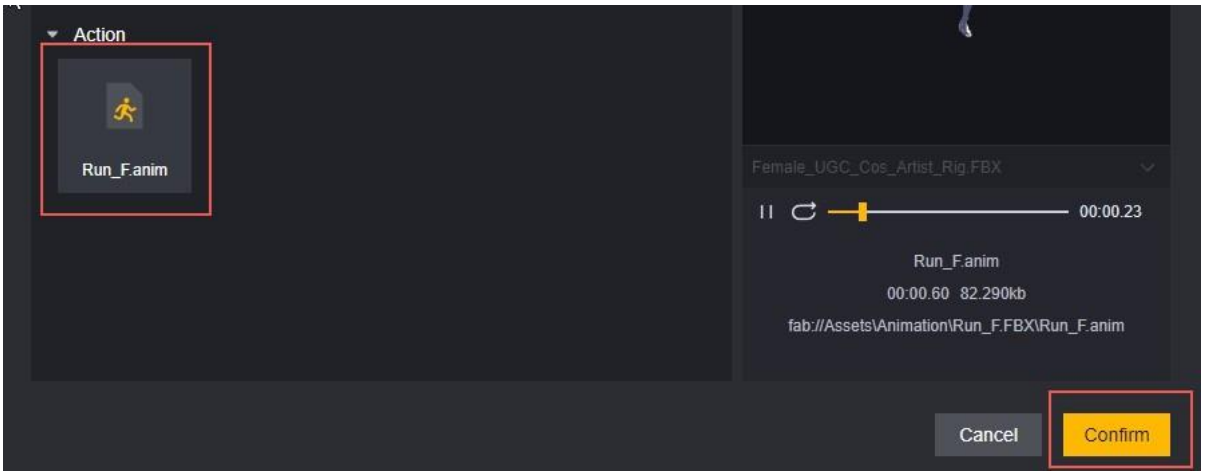



Modify this action to play the custom action you added when the game requires the player character to play the stand-idle.

For demonstration purposes only, we have configured the running action on a standing idle. Please note that this can lead to strange behaviour, and in practice please add the right action to the right trigger scene according to your design.



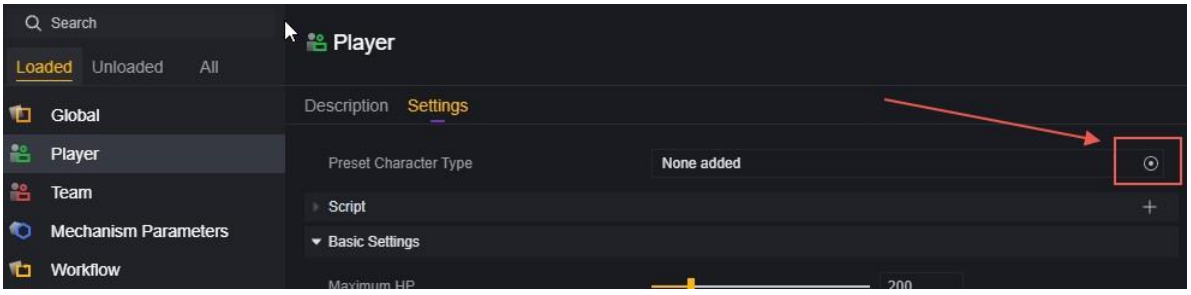



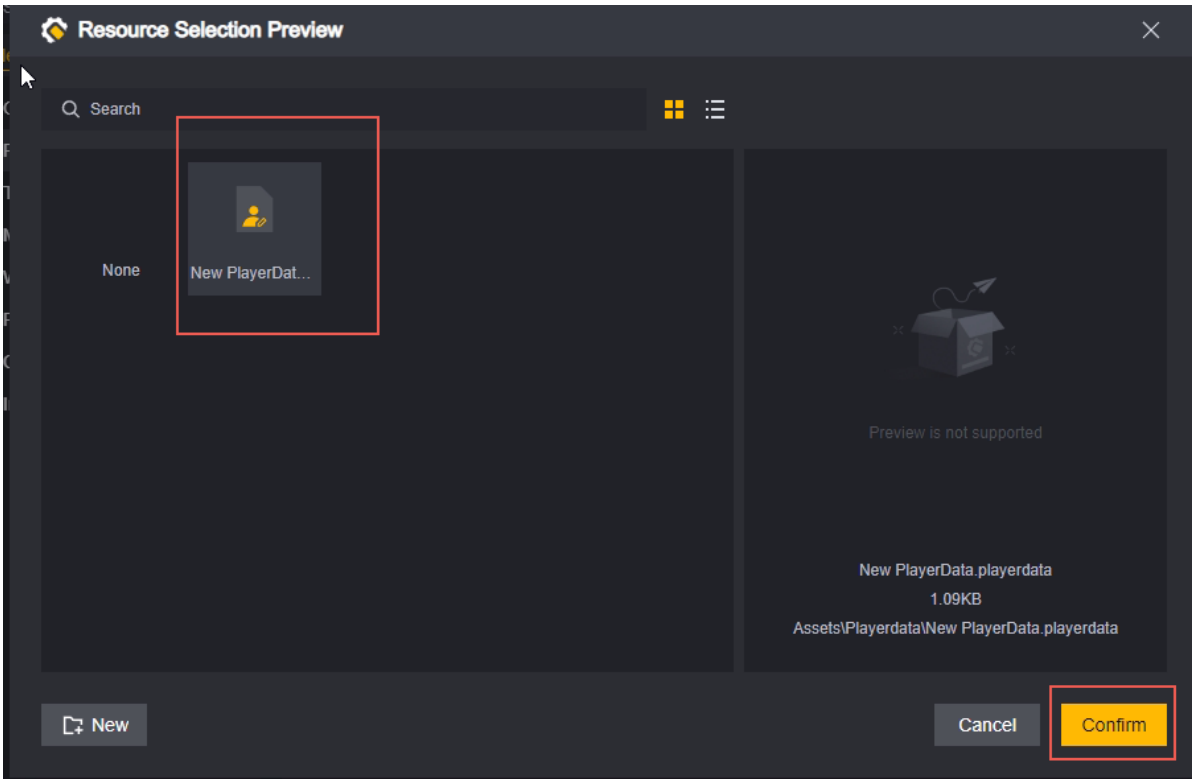

This way this player data will be modified for the player as follows:

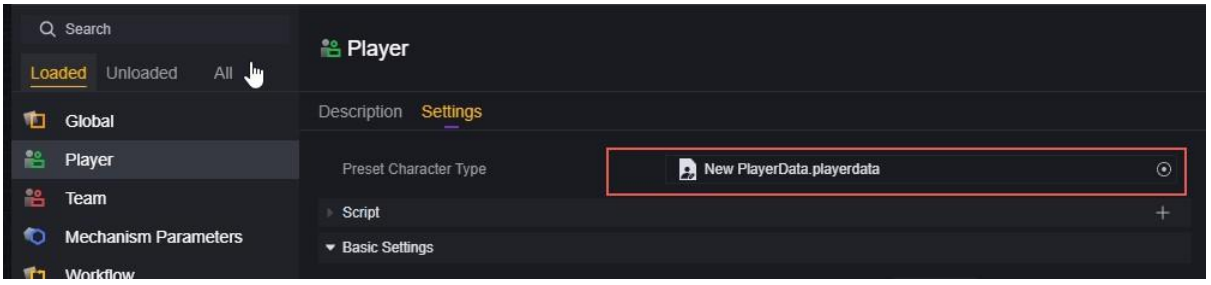
1. Replace the player model with a white-modelled female.
2. Replaces the player standing idle with a running action.

## How to add player data to a player:

Configuring it in the module applies that player data to all players:

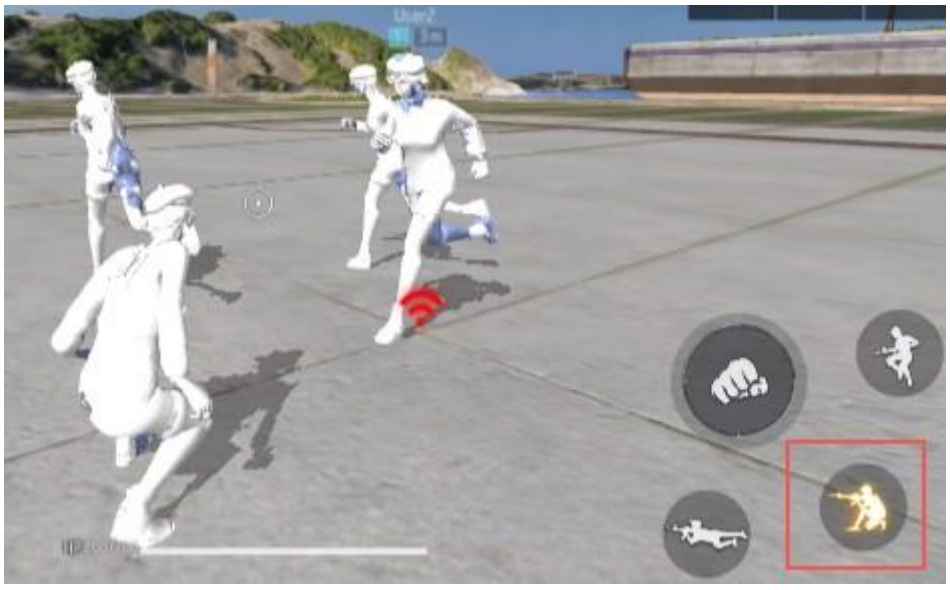




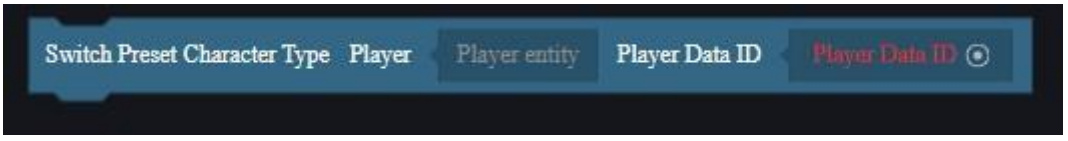

In-game performance:

Everyone performs a run action when they should be standing IDLE, no displacement actually occurs. The default action for the corresponding gender is still played when performing other actions:



> This is the reason why it is recommended to use models with default skeletons; playing default motions with non-compliant skeletons can lead to very strange behaviour if the motion modifications are not made in full.

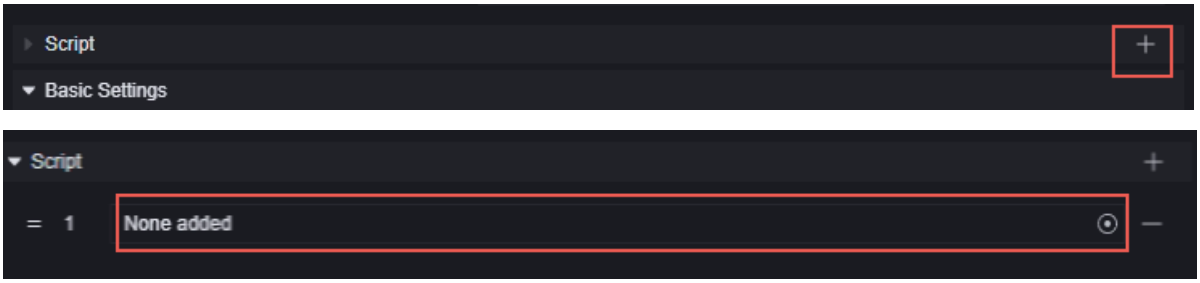
If you only want to modify player data for a specific player, you can do so by using script-specific nodes:



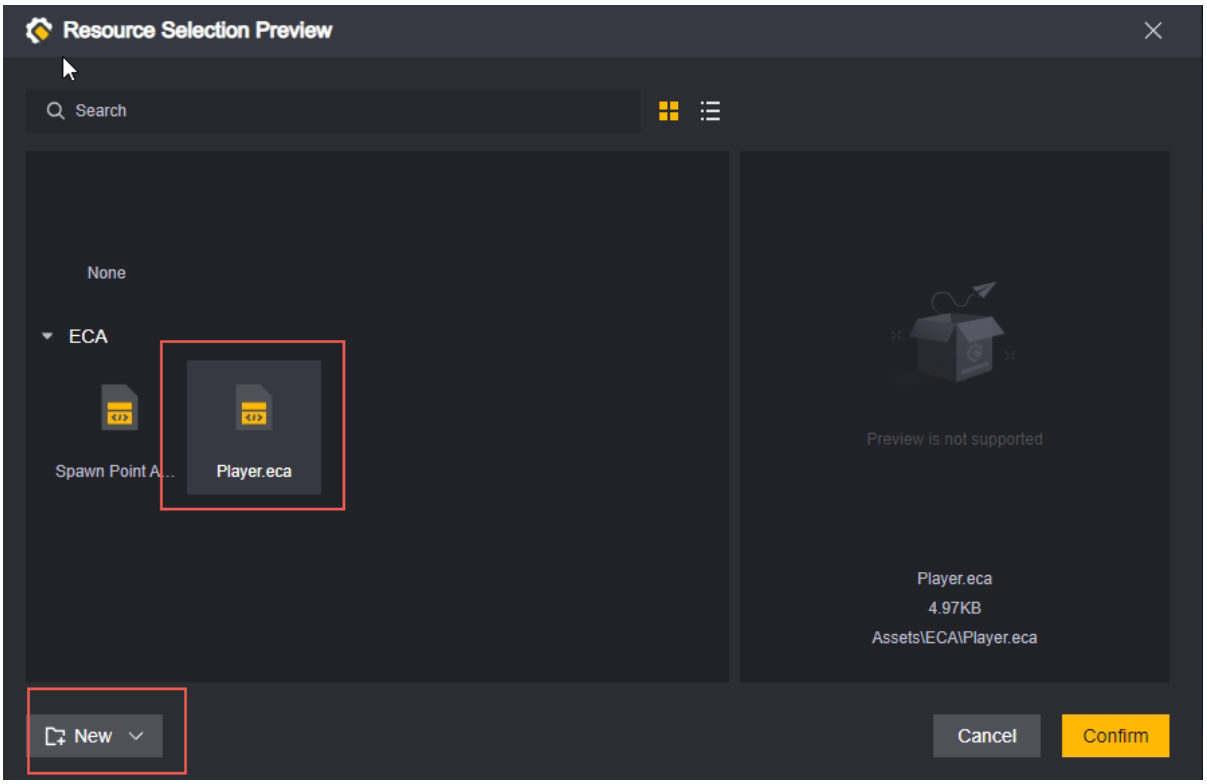

This node can add specified player data for a specific player, and can be carried out to trigger on a specified player according to your needs or after the appropriate conditions are met.

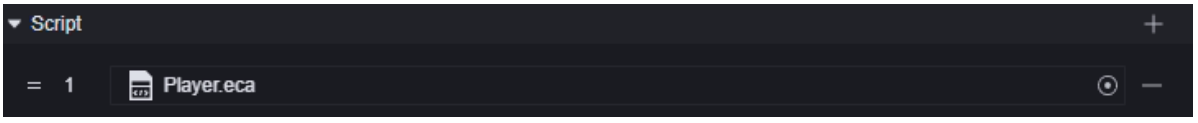
## scripts

Scripts can be mounted for player entities, and running will mount the configured script on each player.




Existing scripts or new ones can be added:




See the Player Scripts section below for scripts.

## Basic settings

Sets the player's base attributes:

**Life Limit:** The player's maximum life value, ranging from 20-1000.

**EP Limit:** The player's maximum EP, ranging from 0-200.

**Initial EP Proportion:** Initial player owned EP = Initial EP Proportion * EP Cap, range 0%-100%.

**Movement Speed:** Scaling of movement speed, less than 100% is slower than the default speed, more than 100% is faster than the default speed. Range 50%-500%. **Jump Height:** Scaling of the jump height, less than 100% is lower than the default height, more than 100% is higher than the default height. Range 10%-500%. **Respawn:** When ticked, player will respawn automatically after death, and new respawn wait time configuration.

**Respawn Wait Time:** Appears only when respawn is ticked. How long to wait to respawn after death, range 0-9999 seconds.

> Modifications to movement speed and jump height change the player's initial movement speed and jump height, and the player's default movement speed scaling and jump height scaling attributes remain at 1. Modifications to these attributes in the script are based on the scaling modifications made to the configurations in the module.

## combatants

Setting of player combat attributes:

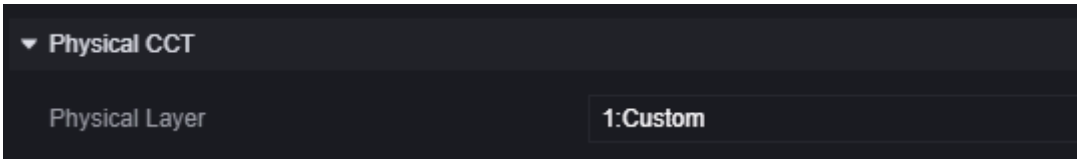
Damage Multiplier: the size of the damage dealt, less than 100% is lower than the default damage, more than 100% is higher than the default damage. Range 0%-500%.

Damage Taken Multiplier: the amount of damage taken, less than 100% is lower than the default damage, more than 100% is higher than the default damage. Range 0%-500%.

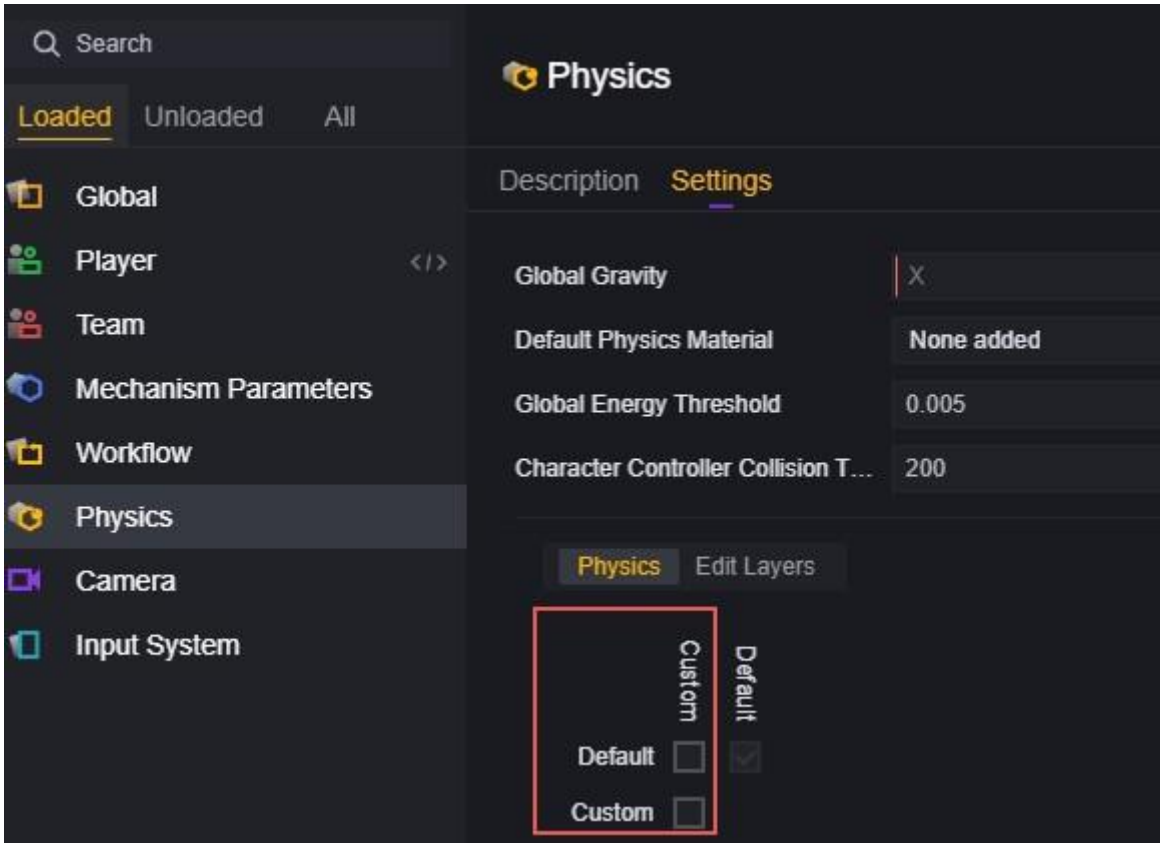
> As with movement speed and jump height, modifications to the damage multiplier and damage taken multiplier will change the player's initial damage and damage sustained, and the player's default damage multiplier and damage taken percentage attributes will remain at 1. Modifications to these attributes in the script are based on scaling modifications made to the in-module configurations.

## Player controller with physical effects

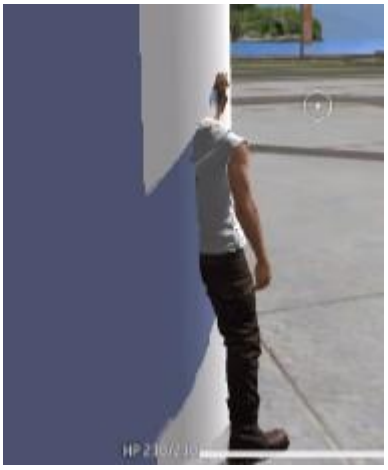
**Physics Tier:** Determines where the player's physics is located, and whether or not collisions occur at different tiers depends on the physics configuration. Configure a non-default physics tier for the player:




Cancels the collision between the Custom tier and the default tier in the physical configuration:




It will turn out that the player can pass through any object of the default physics layer without feeling it:



Props that depend on collision triggers will also not trigger:


**Whether to use player controllers with physics effects:** check to use controllers with custom physics effects and expand the related configuration. **Total height, including upper and lower hemispheres:** the height of the collision body of the character capsule.

**The radius of the upper and lower hemispheres:** the radius of the capsule body, which must not exceed 1/2 of the height.

**Quality:** player quality.

**Maximum uphill angle:** slopes beyond this angle cannot be walked up and the player will start to slide down.

**Step Vertical Offset:** allows the player to deviate a distance from this configuration without being off the ground or blocked, often used when moving up stairs. **Minimum Move Distance:** If the distance the character controller has to move is less than this value, the player will not move at all.

**Skin Thickness:** the depth at which other objects are allowed to embed into the character's collision body, used to avoid jiggling or snagging.

**Self-imposed gravitational acceleration:** the acceleration of gravity in the 3D direction. For players, this setting overrides the global physics settings. **Enable custom thrust:** when turned on, the force applied by the character when pushing a rigid body can be customised. Expand the relevant configuration when turned on.

**Custom Thrust Force:** the amount of force applied when the character pushes the rigid body. **Multi-Jump Height:** the height of the jump when performing a multi-jump.

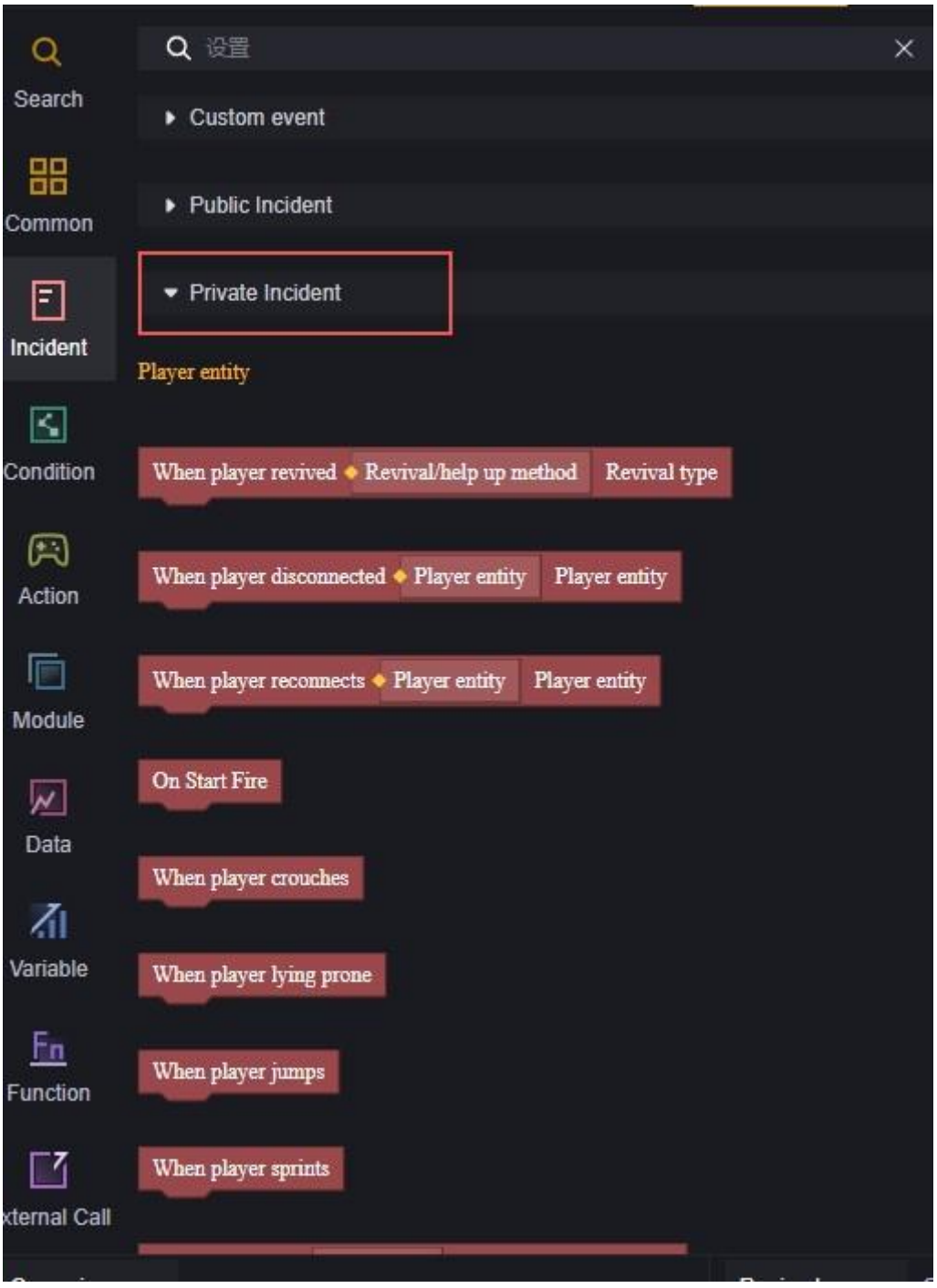
**Maximum number of multihops:** the maximum number of jumps supported per multihop.

**Aerial Movement:** when switched on, the player can change the speed and direction of movement in the air.

## Player Scripts

Scripts mounted in Module-Player will be mounted on each player, please note that global events may be triggered multiple times without design.

Typically, we perform actions on the player and its components on the player script. With private event triggering, it is possible to limit the scope of objects that are triggered:

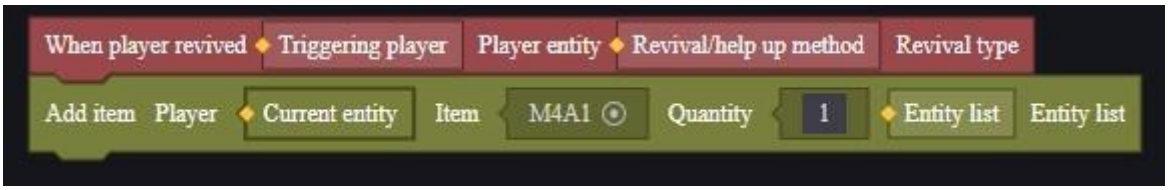



For example, when a player is revived, an M4A1 is issued for the revived player.


The private event "When player is resurrected" is only triggered when the current player entity is resurrected.

Adding props adds an M4A1 to this entity, which is the resurrected player itself, so the requirement can be met. In actual editing, you will also need to consider whether the weapon is full when you add the M4A1, and so on.

For example, using the public event "when player is resurrected" will cause the corresponding logic to be triggered once for all players when a player is resurrected. Similar logic is recommended for global scripts.




Configuring such a script on the player module would cause all players to add an M4A1 every time a player is revived.


Configuring such a script on the player module would result in that player being added to the player total of M4A1s every time a player is revived.