NINJA CHARACTER

-DOCUMENTATION-

Author: Raquel Garcia Guillem

Asset: Itch.io

Version: 1.0.0

Content

Features 2

Technical requirements 2

How to import the asset 2

Asset content 3

How to use the asset 4

Prefabs 4

Ninja 4

Katana and Kunai 6

Weapon manager 7

Top Cam 7

3rd Person Cam 7

Camera Manager 8

HUD 9

Sound Manager 9

Animations 10

Animation controller 10

Input 11

Input Action 11

Sounds 12

Audio Mixer 12

Scripts 12

Animator 12

Camera 13

Enemy 13

Physics 13

Player 14

Sound 14

Weapon 15

Demo Scene 15

Features

- Version: 1.0.0
- Models: 3
- Prefabs: 11
- Textures: 5
- Materials: 5
- Type of materials: PBR
- Input action: keyboard & mouse and gamepad
- Rigging: yes
- Animation controller & Avatar: yes
- Animation clips: 38
- Scripts: 15
- Demo Scene: yes
- Modeled program: Blender
- Animations and rig program: Mixamo
- Root Motion: yes

Technical requirements

- Unity: 2021.3.1
- Cinemachine 2.8.9
- Input System 1.4.3
- TextMeshPro 3.0.6

How to import the asset

1. Open a Unity Project

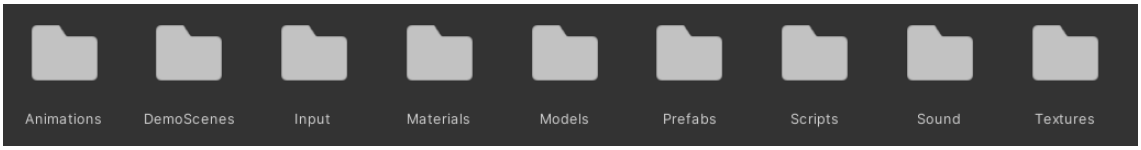
2. Asset>Import Package>Custom package

3. If not import dependencies automatically:

    1. Open Asset Manager>Unity Registry

    2. Import **cinemachine**, **inputsystem** and **textmeshpro**

Asset content



- Animation
  - Animation Controller
  - Avatar
  - Animation clips (38)
- DemoScenes
  - DemoScene
- Input
  - Input action
- Materials (5)
- Models
  - Ninja
  - Katana
  - Kunai
- Prefabs (11)
- Scripts (15)
  - Animator
  - Camera
  - Enemy
  - Physics
  - Player
  - Sound
  - Weapon
- Sound
  - AudioMixer
- Textures (5)

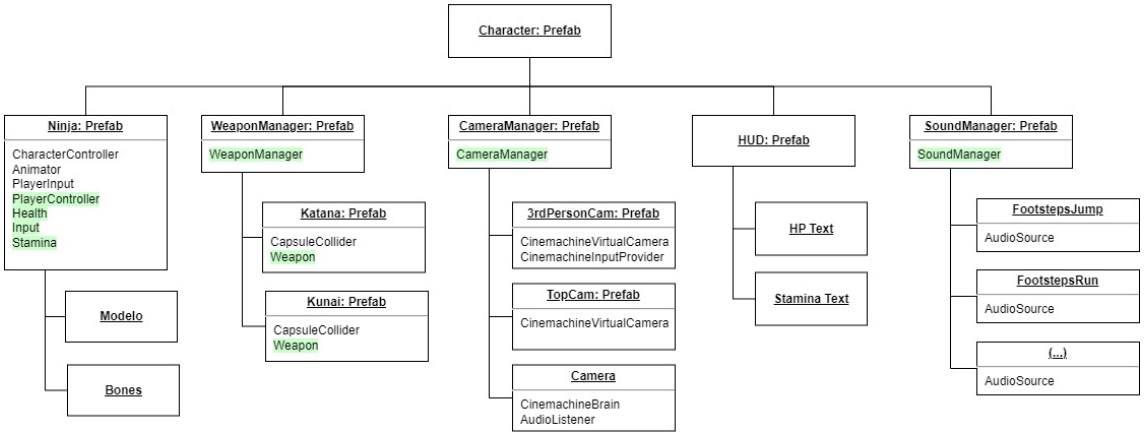
How to use the asset

The prefabs are prepared with the necessary scripts and objects. If you don't want to use the predefined prefabs, you can include the scripts you want in the chosen object.

Prefabs

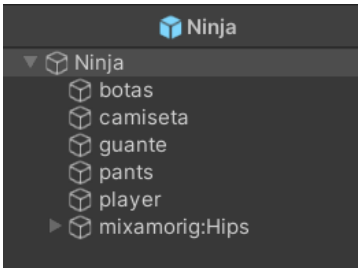
The main prefab is the **Character**, which includes all the others.

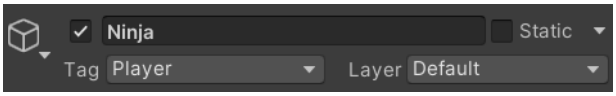
In the following diagram we see how each prefab is formed and the components it contains. The components in green are scripts that are in the Scripts folder.



Ninja

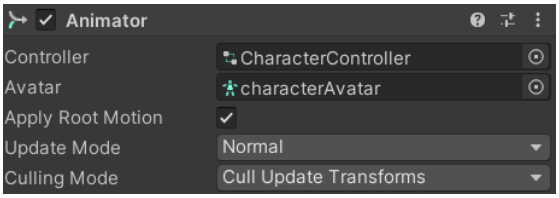
Formed by the model and the bones. Has the **Player tag**.





Animator

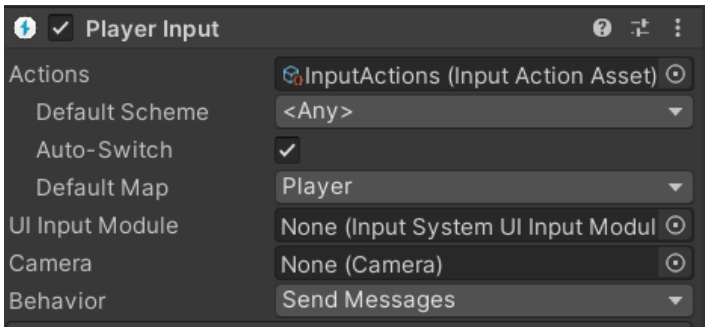
References to **characterAvatar** and **characterController** in the Animations folder. **Root Motion** enabled.



Player Input

Reference to the **InputActions** in the Input folder.

Auto switch, the default map of the **Player** and the **Send Messages** mode have been selected to receive keystrokes.

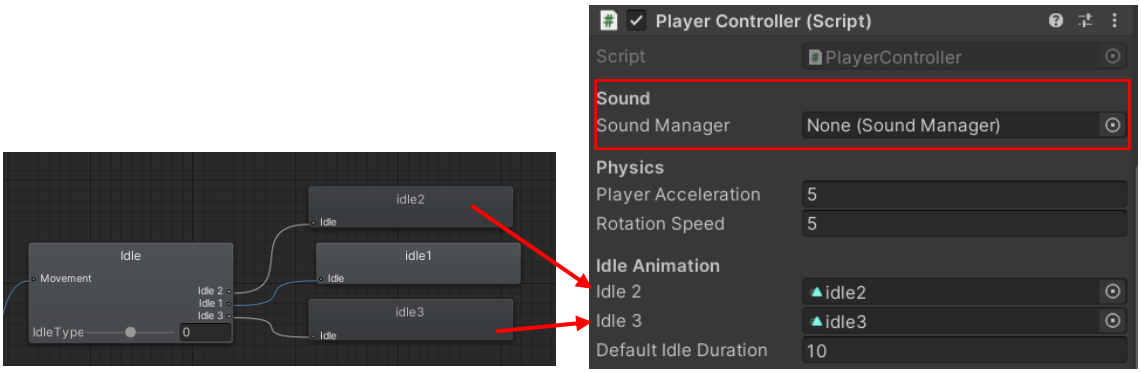


PlayerController

The physical values refer to the **acceleration** to reach the maximum speed or braking and the **speed** with which the model rotates when moving the camera.

The idle animations section contains two **secondary animations of inactivity** and the **waiting time** during which the default animation will be running. When the time is up, one of the other two animations is executed. **These child animations must match the animations in the Animator Controller**.

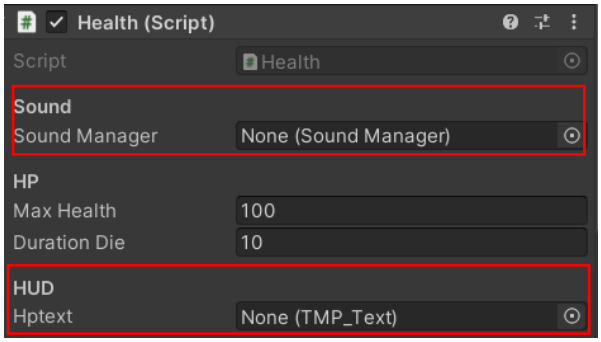
A reference to the **SoundManager** can be included.



Health

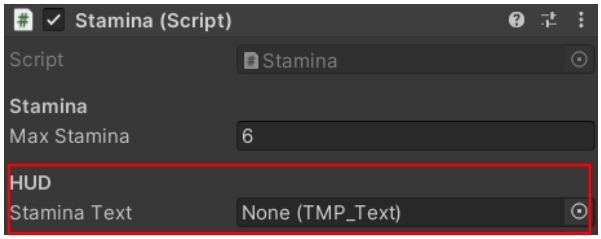
You can modify the maximum **health** values and the **duration of death** time (time from death to hiding the object).

You can include a reference to the **SoundManager** and a **text** to show the value of current health.



Stamina

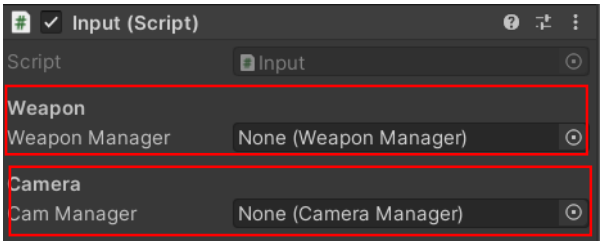
Value of the maximum number of **stamina** that can be loaded. A text reference can be included to show the value of the current stamina.



Input

It is responsible for collecting the data when pressing the buttons and passing the data to the components that need it.

References to the **Weapon Manager** for changing weapons and the **Camera Manager** for changing cameras can be included.



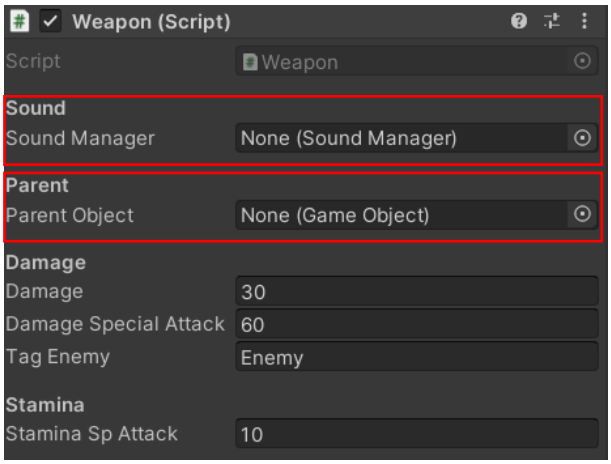
Katana and Kunai

They have the model of the weapon.

Weapon

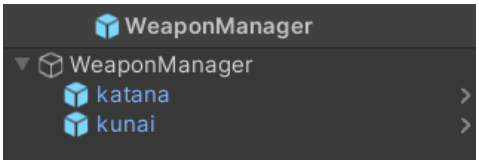
Configuration of the **damage** that makes the **attack** and **special attack**. You need the **tag** of the objects you can damage and the amount of **stamina** that the special attack uses.

You can include a reference **to the object you need to follow** (hand bone) and the **Sound Manager.**



Weapon manager

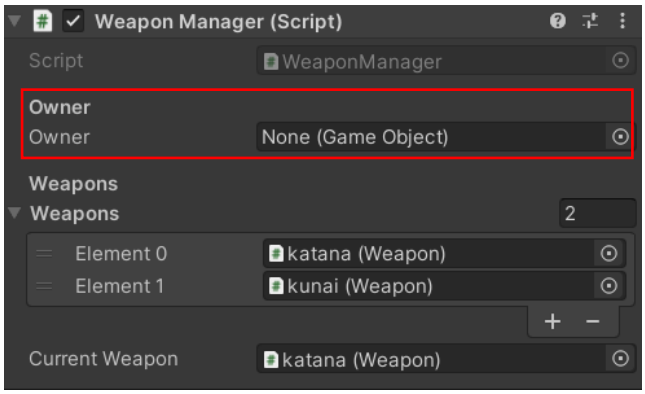
It contains the prefabs of Kunai and Katana.



Weapon Manager

It is responsible for managing a set of weapons. It has the katana and kunai as weapons and the katana as an active weapon.

A reference to the **object that will carry these weapons** (ninja prefab) can be included .

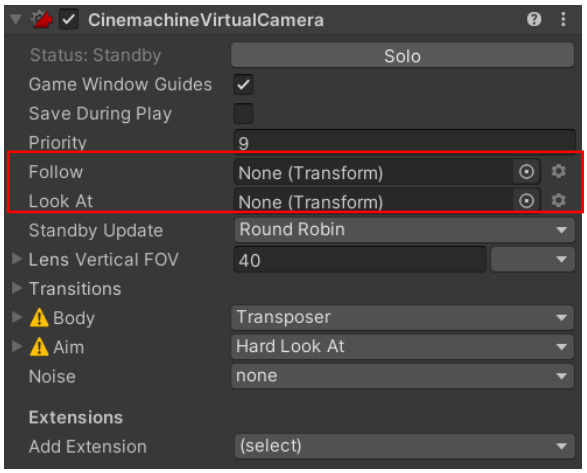


Top Cam

Camera with aerial view that follows an object.

Cinemachine Virtual Camera

Transposer option in Body and **Hard Look At** in Aim. **You need references in Follow and Look At to the object to follow (ninja).**

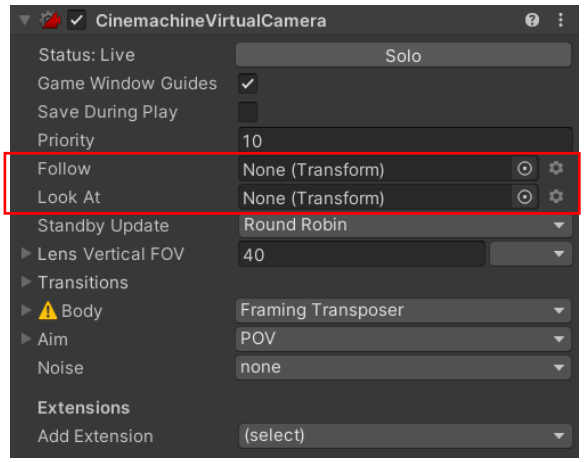


3rd Person Cam

Third-person camera that moves around an object.

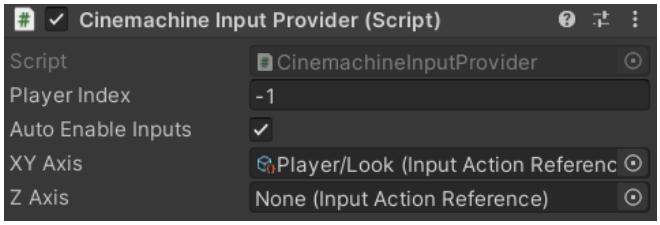
Cinemachine Virtual Camera

Option of Framing Transposer in Body and **POV** in Aim. **You need references in Follow and Look At to the object to follow (ninja).**



Cinemachine Input Provider

It has a reference to the Look action of the input action to move in XY according to the captured values.

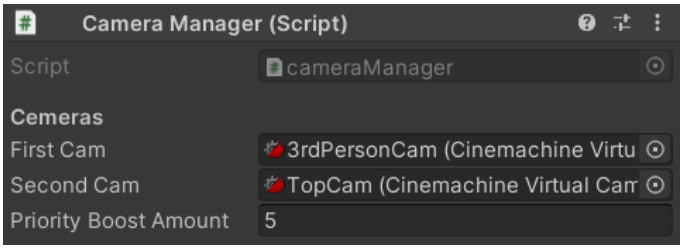


Camera Manager

Formed by the main camera, topCam and 3rdPersonCam. The main camera contains a **cinemachineBrain** component to switch between the positions of the other cameras.

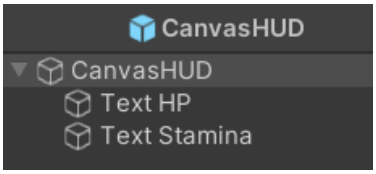
Camera Manager

The 3rdPersonCam is configured as the first camera and the topCam as the **second camera**. It also includes the **amount of priority to add** in the second chamber when you want to switch to it.



HUD

It contains **two** 2D texts to show the life and stamina of the character. They can be referenced from the Health and Stamina component.

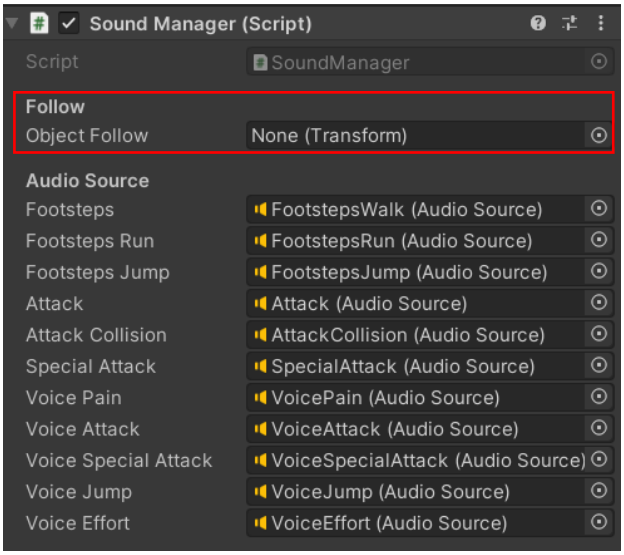


Sound Manager

Formed by 11 objects that each contains a different Audio Source. Manage the character's sound events.

Sound Manager

It contains references to all Audio Sources and an object can be referenced **to track its position** (the ninja).



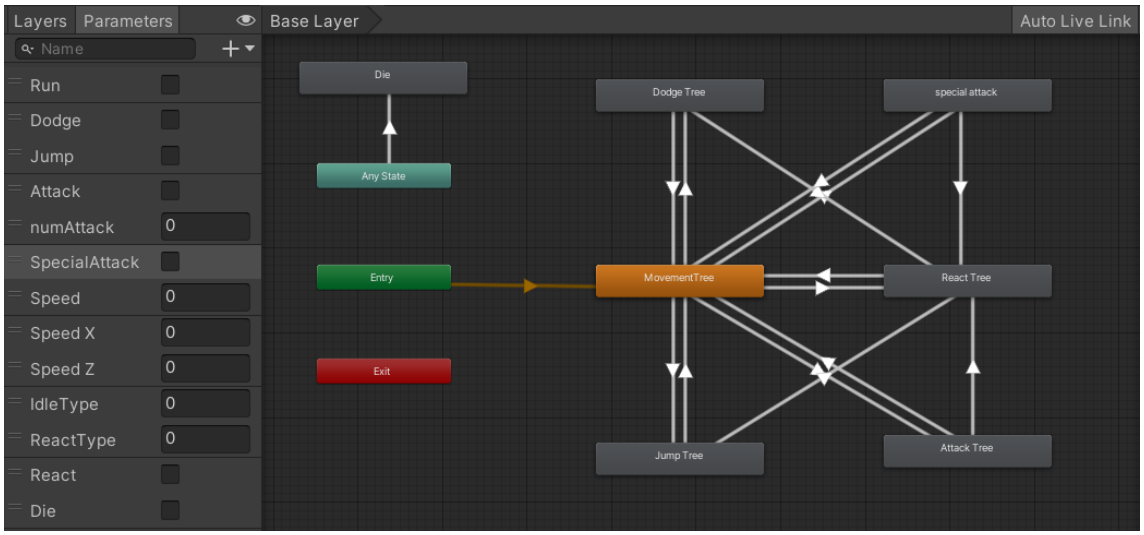
Animations

All animations have been obtained from **Mixamo** and the clips have been modified to adapt them to the programmed movement.

If you want to add more animations, they must be configured as **humanoid** to be compatible with the avatar of the character.

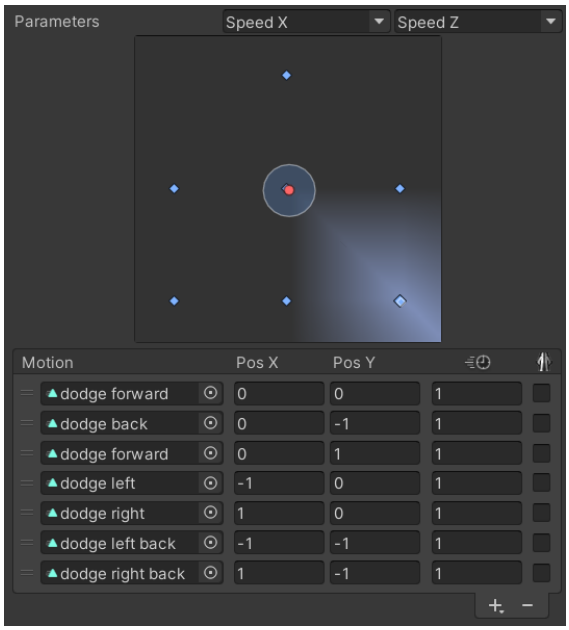
Animation controller

It contains **13 parameters** to handle animations and **7 states**. The Dodge, Special attack, React, Attack, and Jump states contain scripts so that when you exit these states, the parameters that correspond to each case are modified.



- **DodgeTree**: 2D tree. It is executed when the Dodge parameter is activated and when it is exited it is set to false.
- **AttackTree**: 1D tree. According to the num parameterAttack executes one animation or another. It is executed when the Attack parameter is activated and when exiting it is set to false and modifies the value of numAttack.
- **Special Attack:** Motion: It is executed when the Special Atack parameter is activated and when it leaves it sets it to false next to the Attack parameter and the numAttack to 0.
- **ReactTree**: 1D tree. Depending on the param, ReactType executes one animation or another. It is executed when the React parameter is activated and when it leaves it puts all bools to false and modifies the value of ReactType.
- **JumpTree**: 1D tree and 2D tree. It is executed when the Jump parameter is activated and when it leaves it sets to false.
- **Die**: Motion. It is executed when the Die parameter is activated and when finished it does not return to another state, it is the end of the animations.
- **MovementTree: 1D tree** containing 1 1D tree and 2 2D trees. It uses the speeds and is the default state.

All states with 2D trees are handled with the **parameters Speed X** and **Speed Z** and have 7 motions placed as follows:



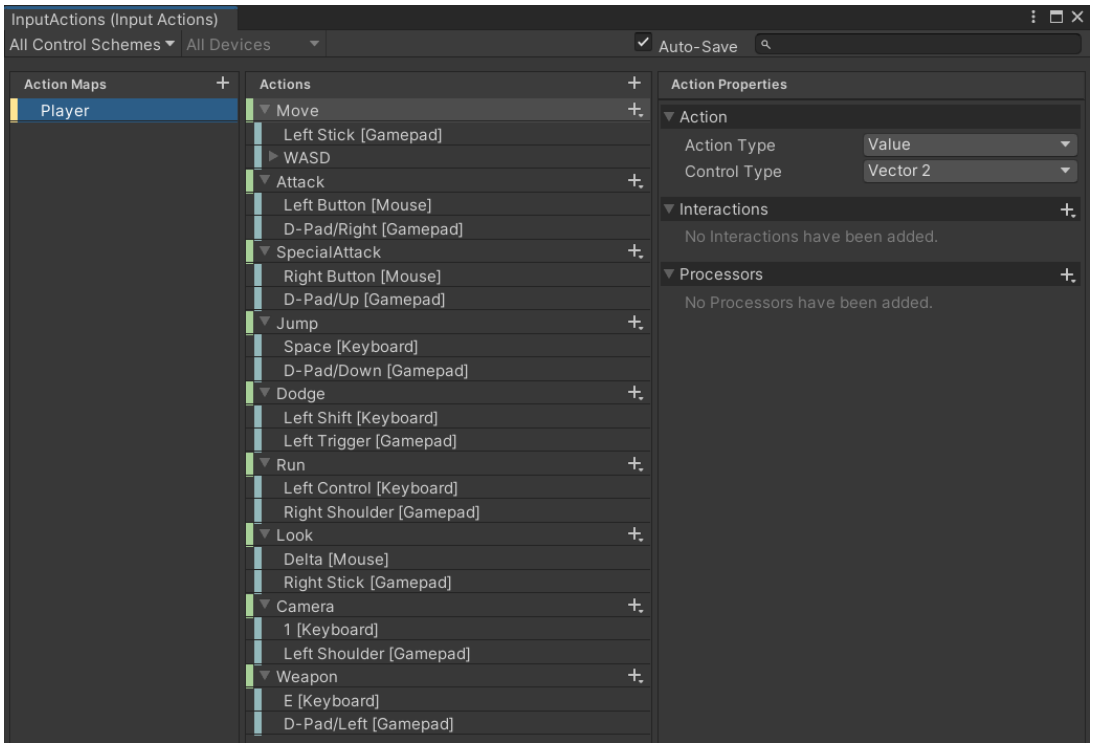
Input

An asset input action has been created to capture the player's actions. Contains the basic actions used from the input script.

Input Action

It has a single action map for the character. All actions are prepared for keyboard & mouse or gamepad but can be configured as preferred.

The Move and Look actions store the information in a vector2, while the rest of the actions are buttons. If they are modified, the corresponding code of the input script that processes the data must be changed.



Sounds

To add sounds to the Sound Manager, place the audio clips in each Audio Source.

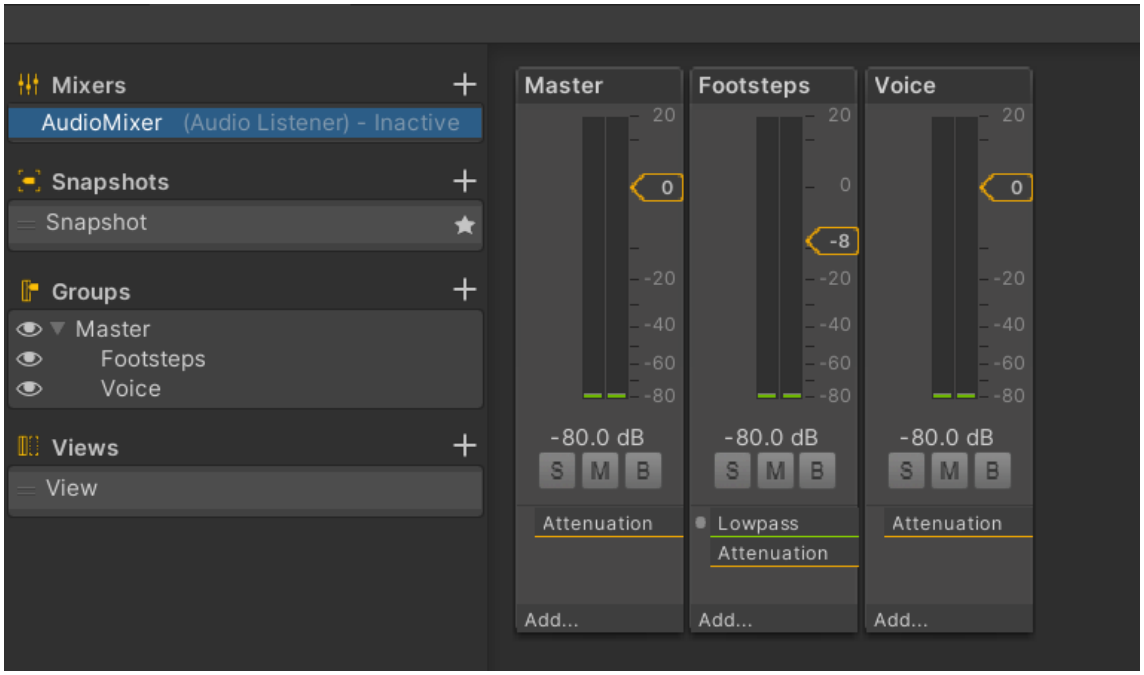
Some Audio Sources are part of an Audio Mixer group to apply effects to them.

Audio clips can be obtained from the following unity assets:

- Voices - Essentials de NoxSound
- Footsteps - Essentials de NoxSound
- Blades & Bludgeoning Free Sample Pack – Idia Software LLC
- SwordSoundPack – Edoardo Gigante

Audio Mixer

As we can see there are 2 groups, one for footsteps and one for voices. Footsteps have a low pass filter to eliminate high-pitched sounds and an attenuation of -8db. As for the voices they have the pitch at 125%.

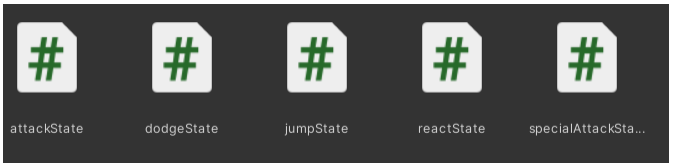


Scripts

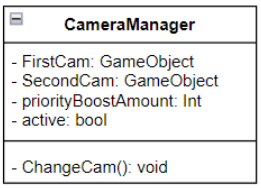
Below is the definition of each script by folders

Animator

Contains scripts used by the animation handler when it changes state.

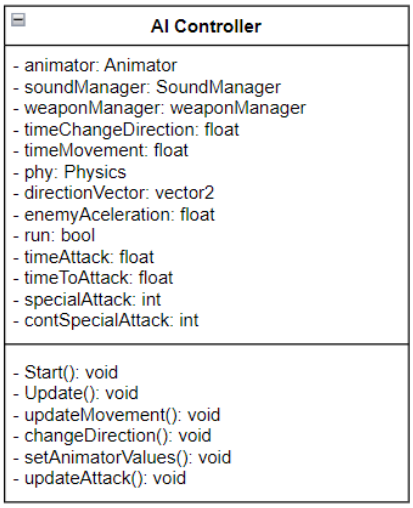


Camera



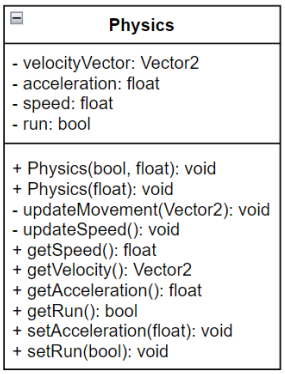
Enemy

AI with basic movement and time-based attacks for testing.

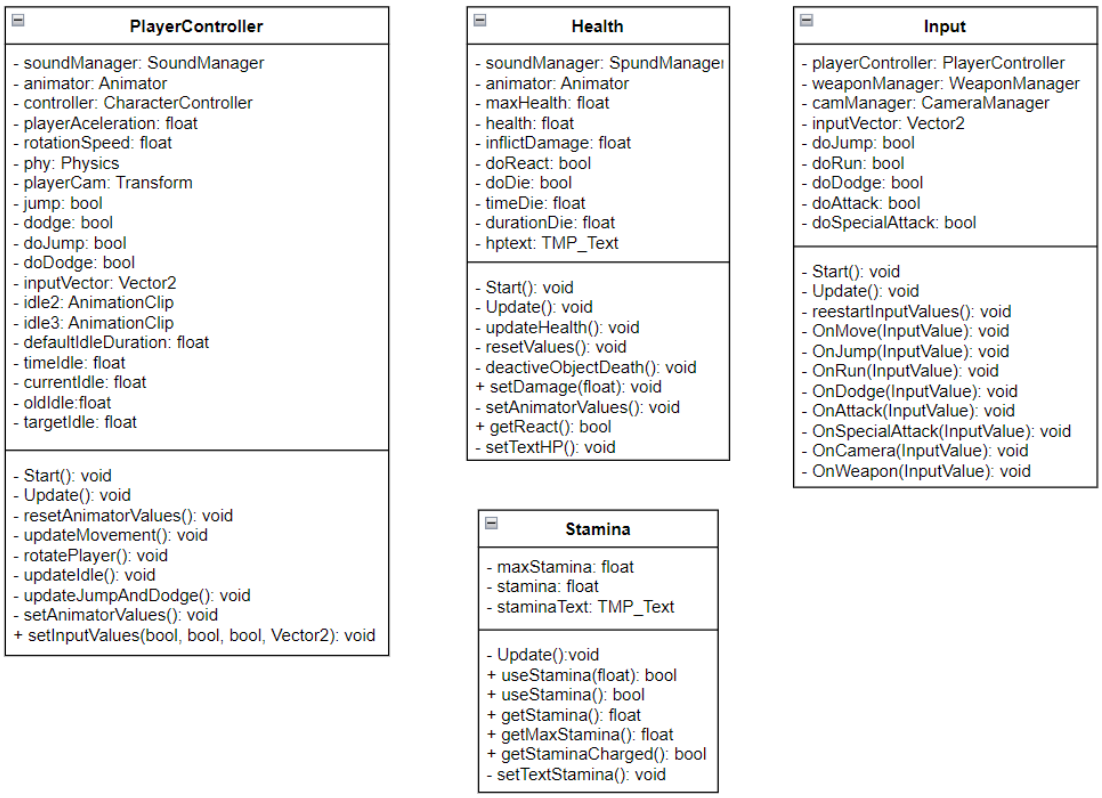


Physics

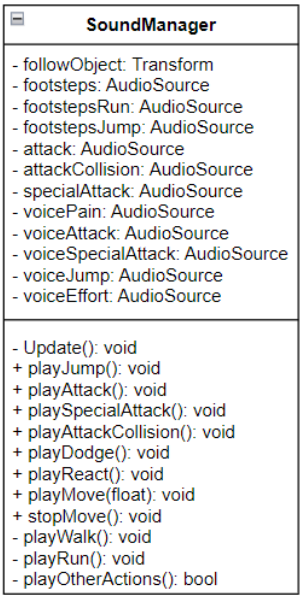
Manage the physics that can be applied to the character.



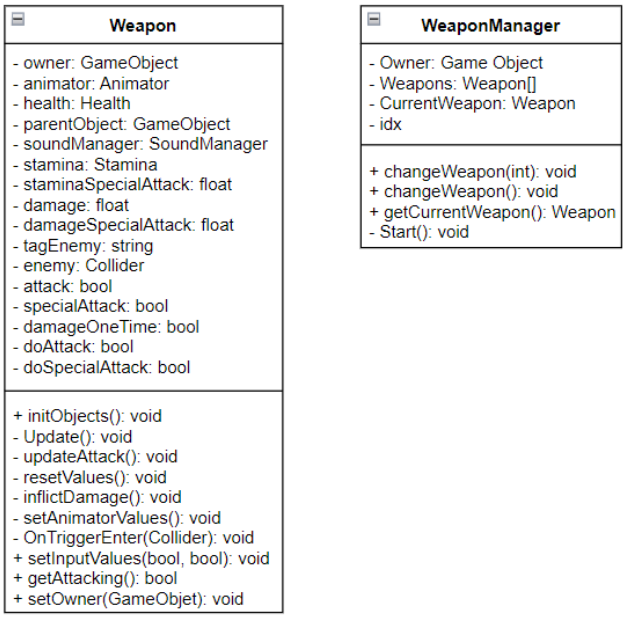
Player



Sound



Weapon



Demo Scene

The demo scene contains a **player**, **5 enemies** and a **terrain** to check the correct functioning of the character. On the right, there are several objects to check collisions while on the other side are the enemies.

These enemies have been created using the character, but with another of the available textures. The components of the player control have also been removed and the **AiController** component has been included. In addition, they have been placed the **Enemy tag**.