RIGONIX THIRD PERSON CONTROLLER DOCUMENTATION

Official Documentation & User Guide

Version: 1.0

Framework: Unity 2021.3+ (Built-in / URP / HDRP)

Input System: Both Legacy And New Input System

Supports: Keyboard Mouse, Gamepad, And Mobile Controls

Scripting Backend: Mono / IL2CPP

From: Rigonix3D.com

Table of Contents

1. Introduction & Overview
2. Getting Started (The Easy Way)
    - The Rigonix Character Creator
3. Setting Up Ragdoll In Single Click
4. The Camera System
5. Core Modules (Deep Dive)
    - Locomotion Module
    - Airborne Module (Jumping & Falling)
    - Dodge Module (Roll, Slide)
    - Health & Stamina Module
    - Head IK Module
    - Emotes Module
    - Audio Module
    - Input Module
6. Advanced Systems
    - Footstep System (Surface Detection)
    - Interaction System

1. Introduction & Overview

Rigonix Ultimate is a modular, component-based character controller framework designed to bridge the gap between "Starter Assets" and "AAA Production Frameworks."

Unlike monolithic controllers where all logic is stuffed into one 3,000-line script, Rigonix uses a **Module Architecture**. Every major feature (Locomotion, Ragdoll, Footsteps) is a self-contained module that can be enabled, disabled, or customized directly from the custom Inspector.

Key Features:

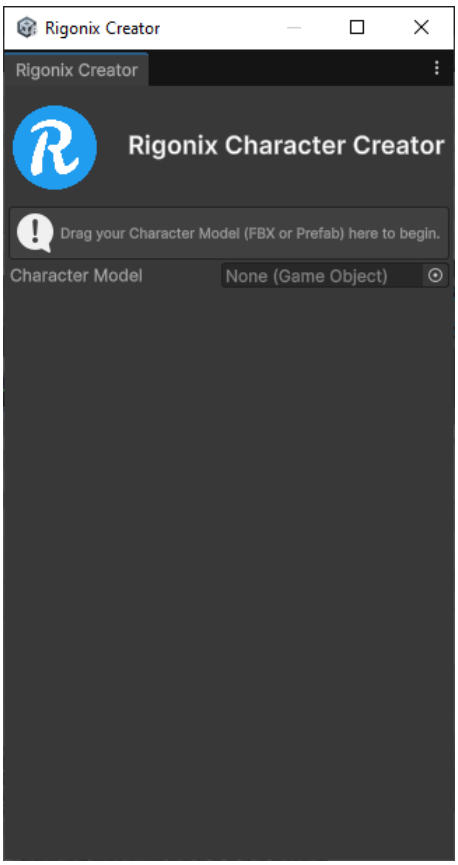
- **Unified Editor:** A custom GUI with tabs, cards, and automation buttons.
- **Dynamic Physics:** Colliders that resize in real-time for crouching/crawling.
- **Hybrid Ragdolls:** Seamless transitions from animation to ragdoll physics and back (Get-Up system).
- **Surface Engine:** Texture-based and Tag-based footstep detection.

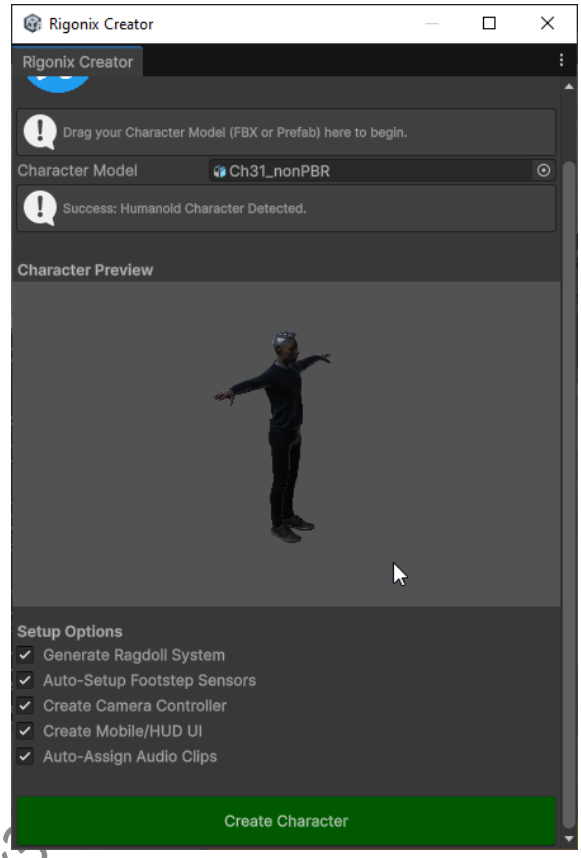
2. Getting Started (The Easy Way)

Rigonix includes a **Character Creator Wizard** that automates 95% of the setup process.

Using the Character Creator

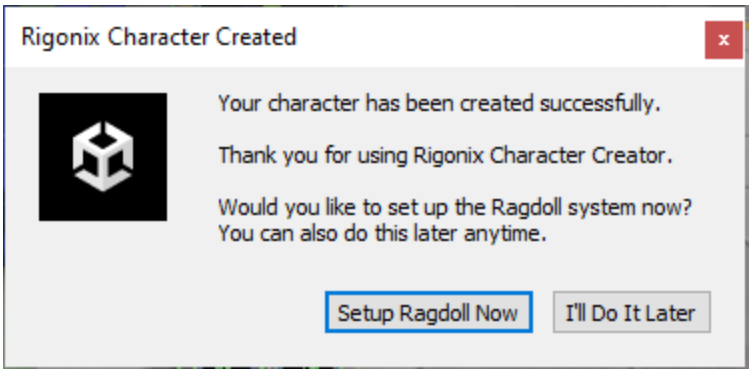
1. Navigate to the Top Menu: **Rigonix -> Character Creator**.
2. A window will appear. **Drag and Drop** your character model (FBX or Prefab) into the "Character Model" slot.
3. The tool will validate if the model is **Humanoid**. If not, you must fix the Rig type in the model import settings.

4. 



4. **Select Options:**
    - **Generate Ragdoll:** Automatically analyzes bones and adds colliders/rigidbodies.
    - **Auto-Setup Footsteps:** Adds sensors to feet bones.
    - **Create Camera:** Spawns and links the Rigonix Camera system.
    - **Auto-Assign Audio:** Scans your project for default audio clips (Walk, Jump, Land) and assigns them.
5. Click **Create Character**.

Your character is now ready to play.

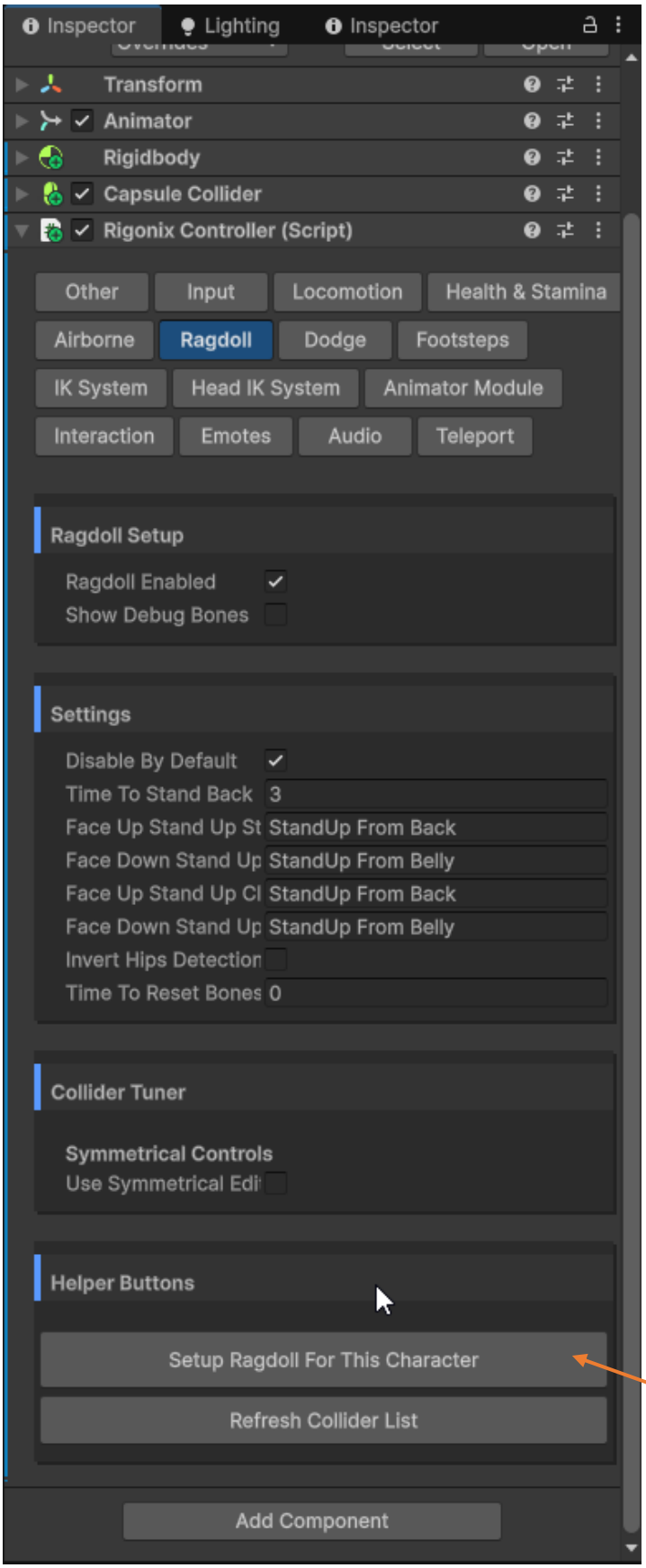


You will get this message once the character is created. You can setup ragdoll by our custom editor by clicking on SetupRagdollNow Button, or you can do it later by and click I'll Do It Later For now.

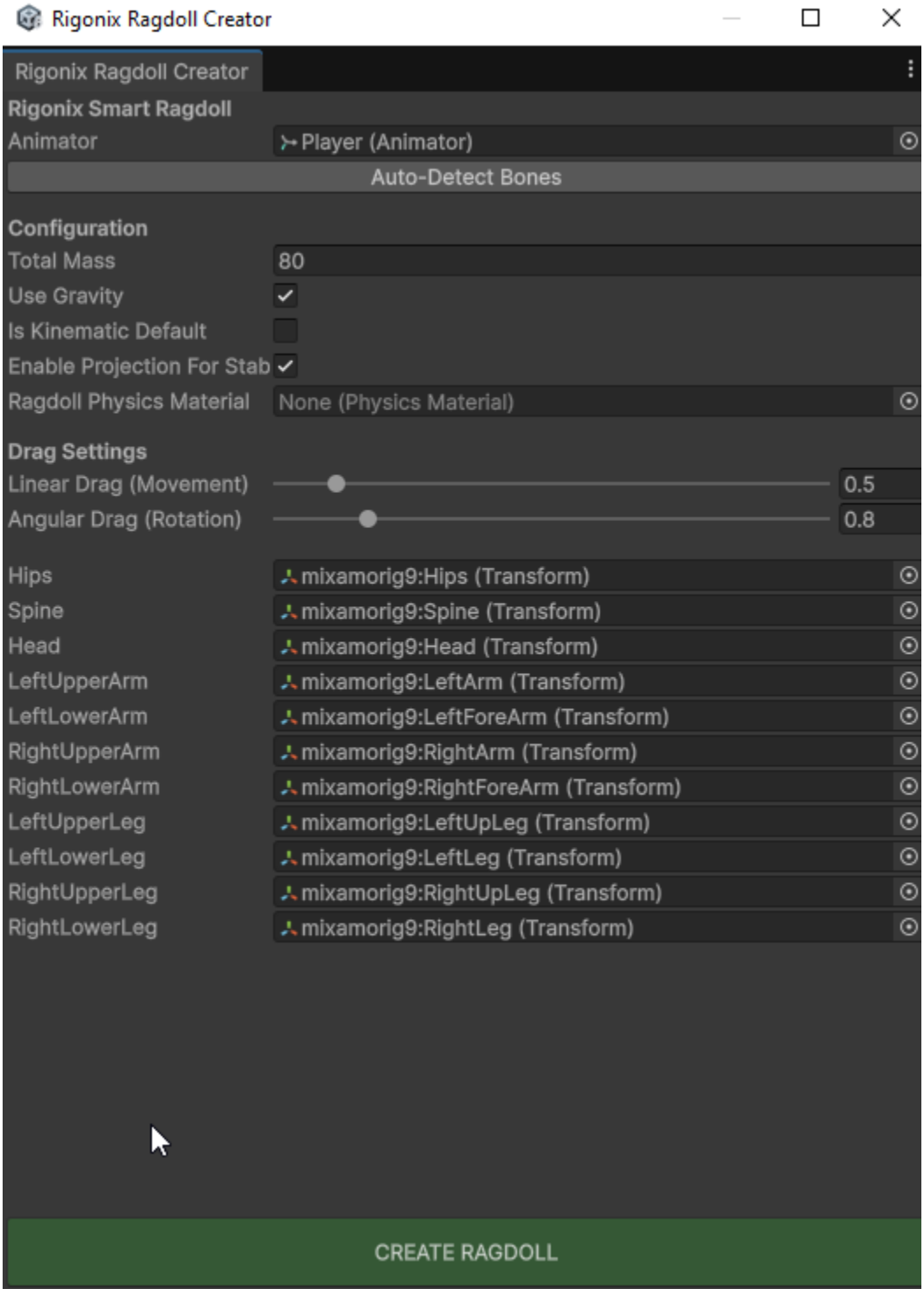
3. Setting Up Ragdoll in Single Click

You can setup the ragdoll while creating the character. Or you haven't setup the ragdoll at that time you can still do it. To create ragdoll follow these steps.

1. Select your character controller which will have the script "RigonixController.cs"


Inspector
Lighting
Inspector
Transform
Animator
Rigidbody
Capsule Collider
Rigonix Controller (Script)
Other
Input
Locomotion
Health & Stamina
Airborne
Ragdoll
Dodge
Footsteps
IK System
Head IK System
Animator Module
Interaction
Emotes
Audio
Teleport
Ragdoll Setup
Ragdoll Enabled
Show Debug Bones
Settings
Disable By Default
Time To Stand Back 3
Face Up Stand Up St StandUp From Back
Face Down Stand Up StandUp From Belly
Face Up Stand Up Cl StandUp From Back
Face Down Stand Up StandUp From Belly
Invert Hips Detection
Time To Reset Bones 0
Collider Tuner
Symmetrical Controls
Use Symmetrical Edi
Helper Buttons
Setup Ragdoll For This Character
Refresh Collider List
Add Component

2. Click on Setup Ragdoll For This Character Button. Then Ragdoll Creator Window will appear.



4. Click on the CREATE RAGDOLL Button, the Ragdoll will be created for that character.

4. Camera System 📷

The camera system is built using two scripts:

Script	Responsibility
rCameraController	Camera movement, rotation, offsets, distance, FOV, states
rCameraCollision	Prevents camera clipping into environment

Together they create a smooth, responsive third-person camera.

🎮 rCameraController

✔ Purpose

Controls the overall camera behavior:

- Player follow logic
- Rotation handling
- Camera states (Normal / Crouch / Prone / Aim / Sprint)
- Camera distance
- Field of View (FOV)
- Smooth transitions

🔑 Key Variables

Variable	Description
target	Player transform the camera follows
followOffset	Base camera position offset
followSpeed	Smoothness of camera movement
inputSensitivity	Mouse look sensitivity

Variable	Description
mobileInputSensitivity	Touch sensitivity
clampAngleUp / Down	Vertical rotation limits
invertY	Invert vertical look
cameraDistance	Current camera distance
stateTransitionSpeed	Smooth state blending
fovTransitionSpeed	Smooth FOV blending

Camera States

Each state defines:

- Offset
- Distance
- FOV
- Collision smoothing

State	Usage
normalState	Default gameplay
crouchState	When crouching
proneState	When prone
aimState	During aiming

Core Behaviour

System	Function
Input Handling	Reads mouse / mobile rotation

System	Function
Rotation	Updates horizontal & vertical angles
Follow Logic	Moves camera toward target + offset
State Blending	Smooth transitions between states
FOV Control	Dynamic zoom / sprint effects

Important Functions

Function	Purpose
SetNormal()	Switch to normal camera
SetCrouching()	Switch to crouch camera
SetProne()	Switch to prone camera
SetAiming()	Switch to aim camera
SetSprinting(bool)	Adjust sprint FOV
ResetCameraBehindTarget()	Snap camera behind player
AddRotation()	Manual rotation input
SetTarget()	Assign new target

CameraStateSettings

Container for state-specific camera tuning.

Property	Description
offset	Camera positional offset
distance	Camera distance
fov	Field of View
collisionSmoothTime	Collision response smoothness

---

rCameraCollision

Purpose

Prevents camera clipping into:

- Walls
- Objects
- Environment geometry

Key Variables

Variable	Description
minDistance	Minimum allowed distance
maxDistance	Desired distance from controller
collisionLayerMask	Defines collision surfaces

Behaviour

Step	Description
Desired Position	Calculates ideal camera position

Step	Description
Collision Check	Uses Linecast to detect obstacles
Distance Adjustment	Moves camera closer if blocked
Smooth Recovery	Restores distance when clear

Collision Logic

Condition	Result
Obstacle detected	Camera moves closer
No obstacle	Camera returns to max distance

Smoothness controlled by:

controller.currentCollisionSmoothTime

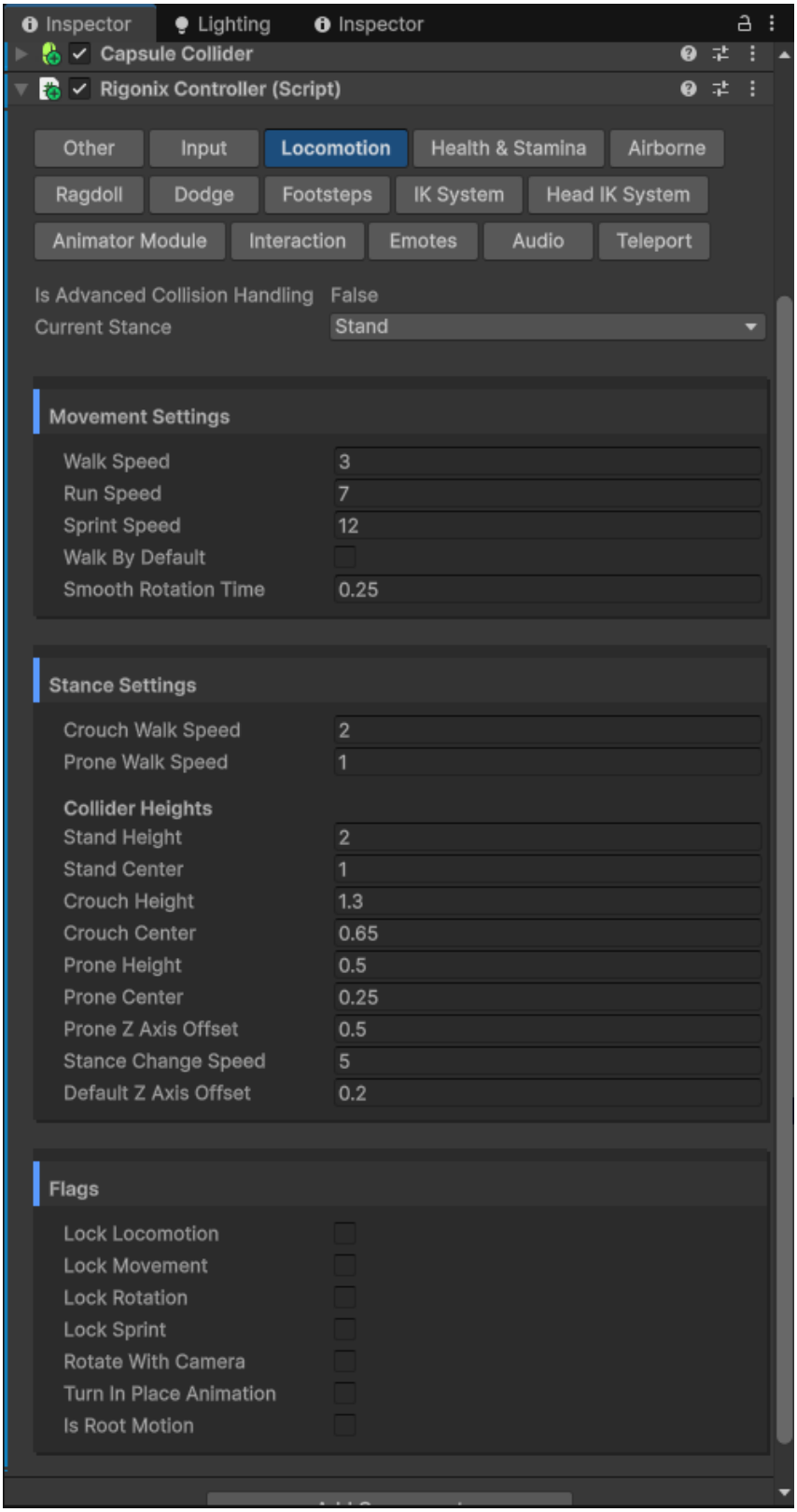
✅ Customization Notes

- ✔ Adjust **CameraStateSettings** for gameplay feel
- ✔ Increase **FOV** for sprint effect
- ✔ Reduce **Distance** for tighter combat camera
- ✔ Tune **collisionSmoothTime** for wall responsiveness
- ✔ Modify **Sensitivity** for camera control feel

5.Core Modules

The controller is divided into modular systems.
Each module is responsible for a specific gameplay feature and can be configured independently.

Locomotion Module



☑ Purpose

Handles all grounded movement behavior.

🔑 Key Features

- Walking / Running / Sprinting
- Crouch & Prone movement
- Speed control
- Rotation handling
- Ground alignment

🔑 Important Variables

Variable	Description
runSpeed	Running movement speed
sprintSpeed	Sprinting speed
crouchWalkSpeed	Movement speed while crouching
proneWalkSpeed	Movement speed while prone
walkByDefault	If true, character will walk by default
lockLocomotion	If true, locks the locomotion [Movement + Rotatation]
lockMovement	If true, locks the movement only
lockRotation	If true, locks the rotation only
Lock Sprint	If True, character will not sprint
rotateWithCamera	If true, character will rotate with camera like GTA VC

Useful Public Functions

Function	Purpose
LockPlayerMovement()	Locks The Player Movment
LockPlayerRotation()	Locks The Player’s Rotation
RemoveVelocity()	Remove Velocity In Every Direction

Airborne Module

Purpose

Controls jumping & falling logic.

Key Features

- Jump handling
- Gravity application
- Falling detection
- Landing transitions


Inspector
Lighting
Inspector
Overrides
Select
Open
Transform
Animator
Rigidbody
Capsule Collider
Rigonix Controller (Script)
Other
Input
Locomotion
Health & Stamina
Airborne
Ragdoll
Dodge
Footsteps
IK System
Head IK System
Animator Module
Interaction
Emotes
Audio
Teleport
Is Grounded
True
Disable Gravity
Gravity & Ground
Gravity
9.8
Ground Check Start Point
0.5
Ground Check Distance
0.8
Ground Layers
Everything
Jump Settings
Stationary Jump Force
4
Stationary Jump Force Applyi
0.35
Walking Jump Force
7
Walking Jump Up Force
3
Walking Jump Force Applying
0.25
Running Jump Force
7
Running Jump Up Force
2.5
Running Jump Duration
3.45
Jump Cooldown
0.2
Fall Damage & Ragdoll
Soft Land Max Velocity
-5
Medium Land Max Velocity
-10
Hard Land Max Velocity
-15
Ragdoll Land Velocity
-20
Enable Ragdoll Fall
Add Component

🔑 Important Variables

Variable	Description
Gravity	Force Of Gravity Acting On Character
GroundCheckStartPoint	Point From Where Ground Checking Will Start
Ground Layers	Layers which are treated as ground.
StationaryJumpForce	Jump Force When Character Is Not Moving
WalkingJumpForce	Force When character is walking/running.
EnableRagdollFall	Whether Ragdoll Fall Will Be Trigged Or Not

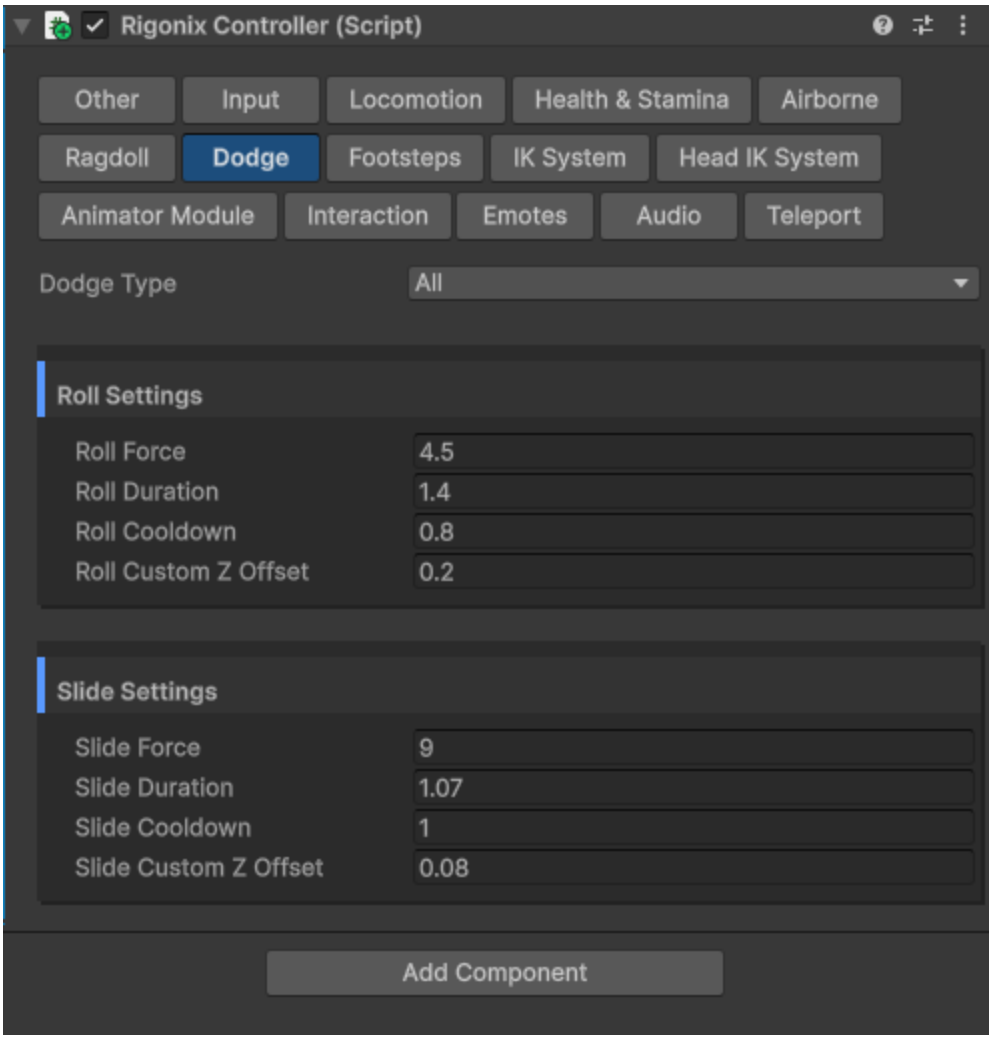
🌀 Dodge Module

☑ Purpose

Handles evasive movement mechanics.

🔑 Key Features

- Dodge / Roll actions
- Directional dodge
- Cooldown management



Important Variables

Variable	Description
DodgeType	Which Types Of Dodge Character Can Perfrom. All = Both, Roll = Only Roll Slide = Only Slide
RollForce	Force Applied At Roll
Roll Duration	Duration To Which Roll Force Is Applied. Matches the animation length of dodge.
SlideFroce	Force Applied While Sliding

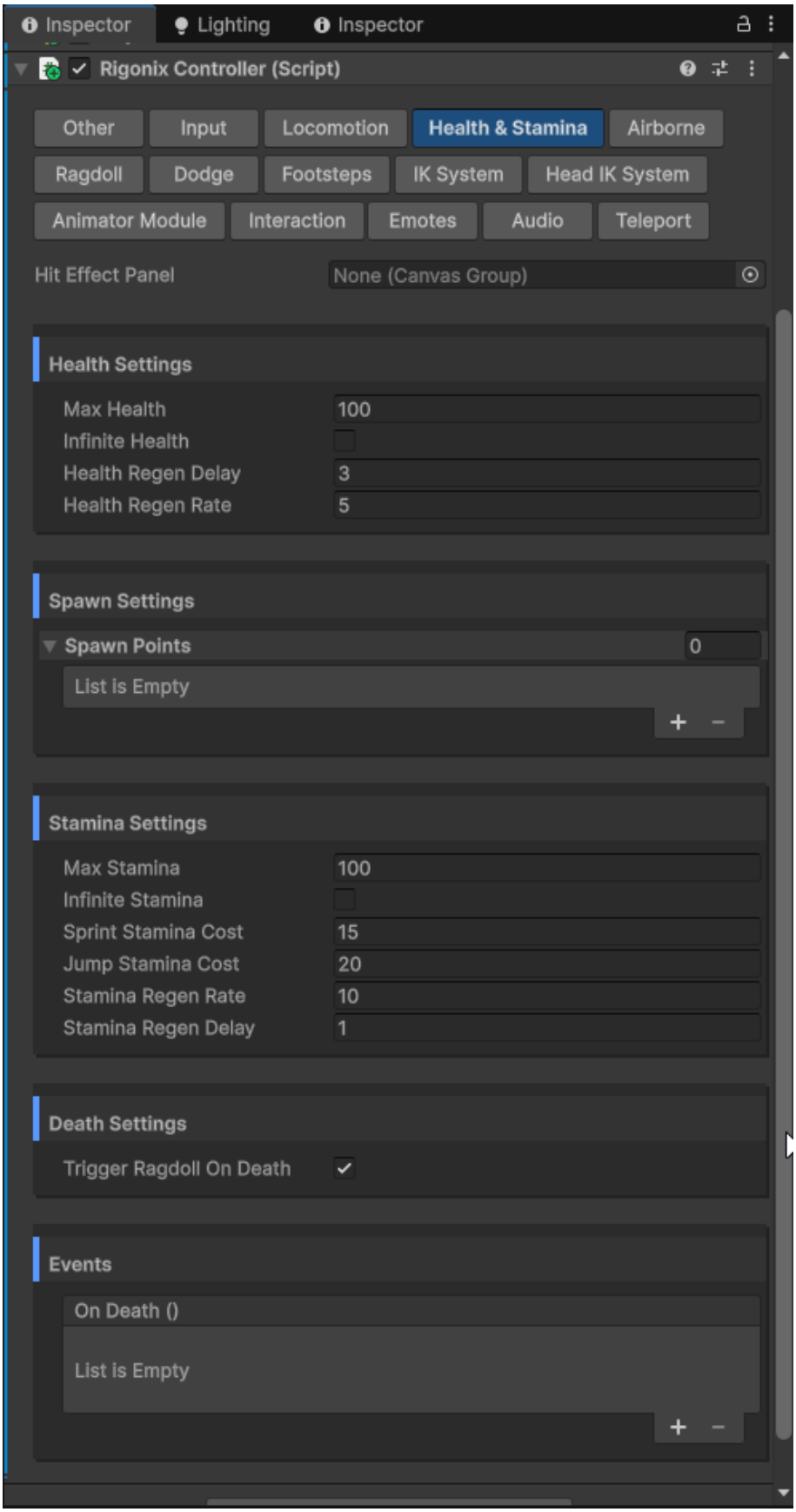
♡ Health & Stamina Module

☑ Purpose

Manages survival & energy systems.

🔑 Key Features

- Health system
- Damage handling
- Stamina consumption / regeneration
- Death logic


Inspector
Lighting
Inspector
Rigonix Controller (Script)
Other
Input
Locomotion
Health & Stamina
Airborne
Ragdoll
Dodge
Footsteps
IK System
Head IK System
Animator Module
Interaction
Emotes
Audio
Teleport
Hit Effect Panel
None (Canvas Group)
Health Settings
Max Health
100
Infinite Health
Health Regen Delay
3
Health Regen Rate
5
Spawn Settings
Spawn Points
0
List is Empty
Stamina Settings
Max Stamina
100
Infinite Stamina
Sprint Stamina Cost
15
Jump Stamina Cost
20
Stamina Regen Rate
10
Stamina Regen Delay
1
Death Settings
Trigger Ragdoll On Death
Events
On Death ()
List is Empty

🔧 Important Variables

Variable	Description
maxHealth	Maximum health
currentHealth	Runtime health value
maxStamina	Maximum stamina
staminaRegenRate	Recovery speed
SpawnPoints	Points Where Character Spawn Again, When Spawn() function is called. If List Is null, then character will spawn at origin.

🎯 Useful Public Functions

Function	Purpose
SetHealth(value)	Sets the health to given value
TakeDamage(amountOfDamage,clipToPlay)	Gives the damange to character and play certain clip [If not given play default hurt sound].
Heal(value)	Increases the health by given amount.
Spawn()	Spawn The Character At New Place. Used after character dies.

👓 Head IK Module

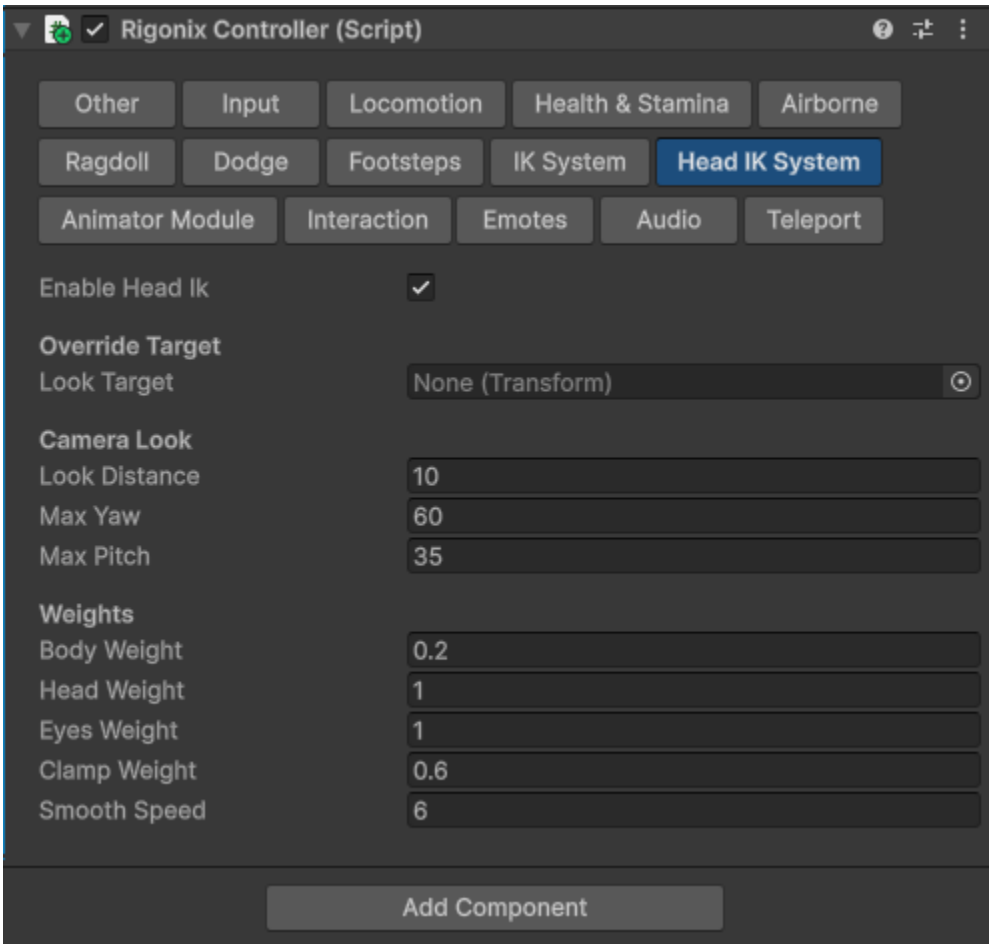
✅ Purpose

Controls character head look behavior.

🔧 Key Features

- Dynamic head tracking

- Target look system
- Smooth blending



Important Variables

Variable	Description
lookTarget	Object being tracked. If null then will look as per camera movement.
LookDistance	Distance till which character will prioritize the look target.
Max Yaw	How much character can rotate its head.

Useful Public Functions

Function	Purpose
SetLookTarget()	Assign tracking target
ClearLookTarget()	Reset head tracking

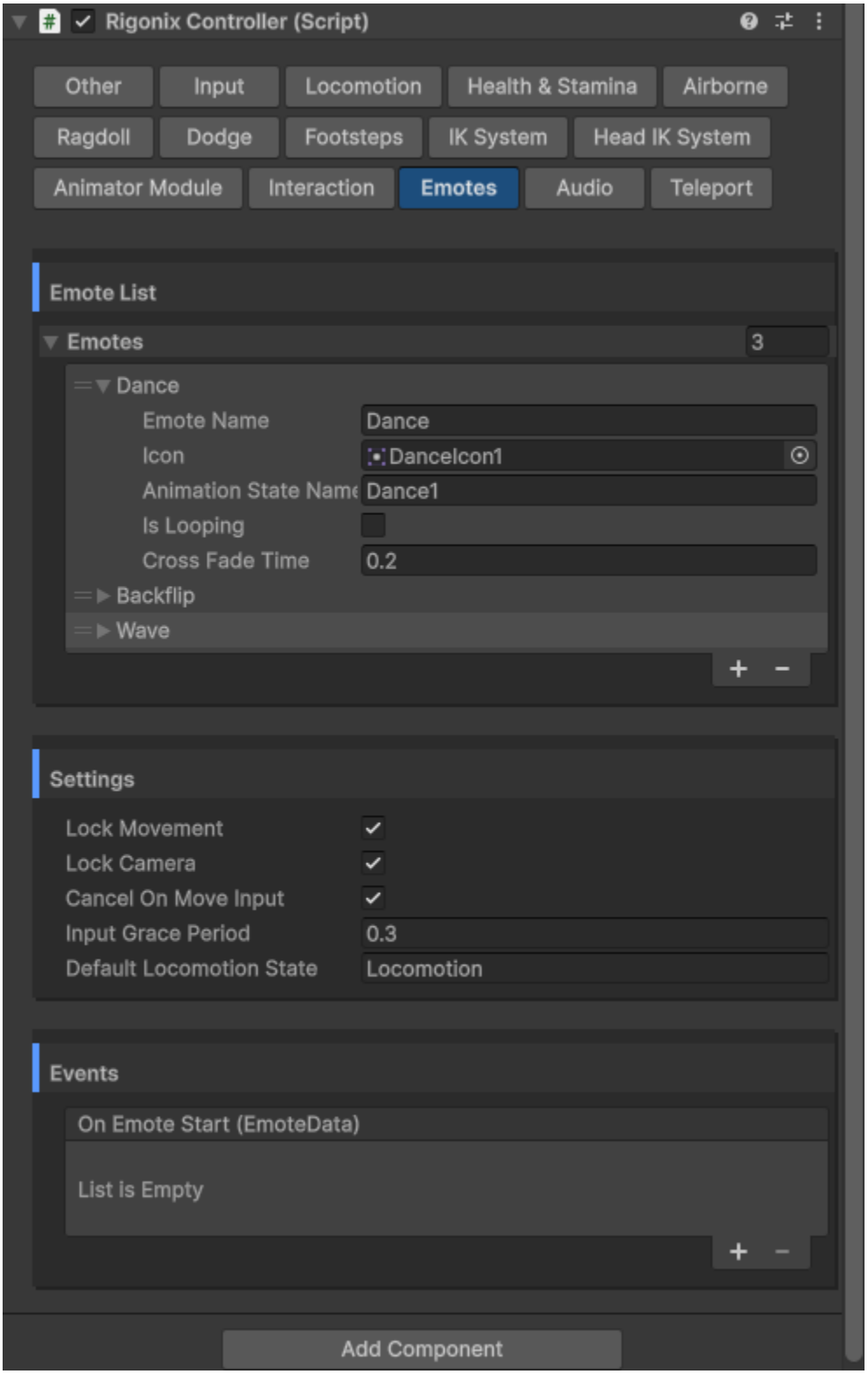
Emote Module

Purpose

Handles character emotes and expressive animations.

Key Features

- Emote animation triggering
- Movement blocking during emotes
- Smooth animation transitions
- Optional looping emotes
- Interrupt control


Rigonix Controller (Script)
Other
Input
Locomotion
Health & Stamina
Airborne
Ragdoll
Dodge
Footsteps
IK System
Head IK System
Animator Module
Interaction
Emotes
Audio
Teleport
Emote List
Emotes
3
Dance
Emote Name
Dance
Icon
DanceIcon1
Animation State Name
Dance1
Is Looping
Cross Fade Time
0.2
Backflip
Wave
Settings
Lock Movement
Lock Camera
Cancel On Move Input
Input Grace Period
0.3
Default Locomotion State
Locomotion
Events
On Emote Start (EmoteData)
List is Empty
Add Component

Important Variables

Variable	Description
Emotes	List of Emotes Chracter Can Perform.
EmoteName	Name of emote displayed on list
Icon	Icon to show of that emote
AnimationStateName	Name of animation's state to play.

Useful Public Functions

Function	Purpose
PlayEmote()	Triggers an emote animation
StopEmote()	Ends the active emote
PlayEmoteByName(name)	Plays a emote by given name

Behaviour Notes

- ✔ Emotes can temporarily override locomotion
- ✔ Emotes may block actions (configurable)
- ✔ Supports both one-shot & looping animations
- ✔ Can be interrupted depending on settings

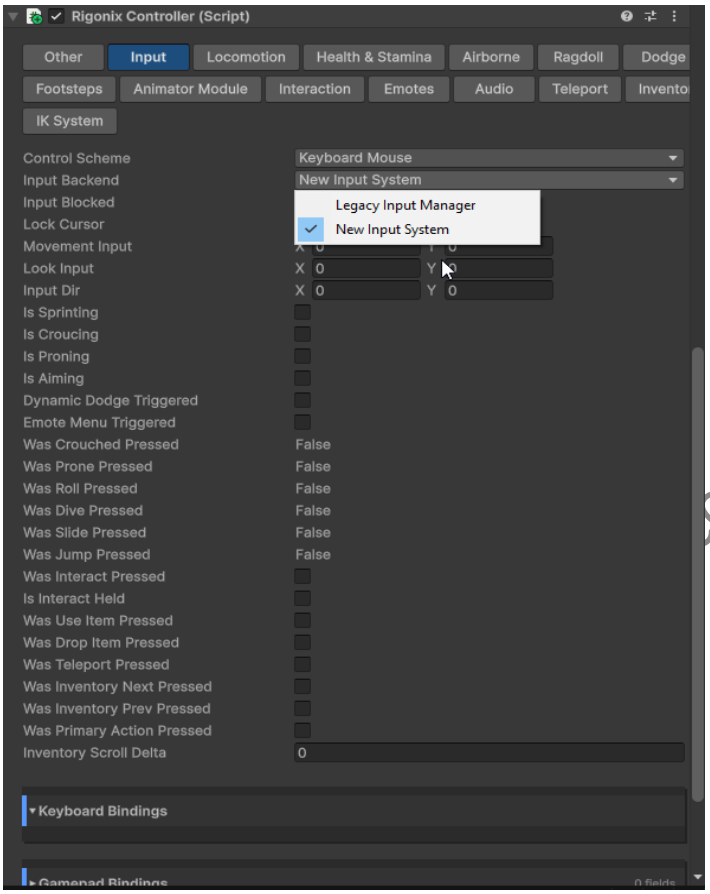
Input Module

Purpose

Centralized input management system.

Key Features

- Movement input
- Camera input
- Action triggers
- Mobile & PC support



Important Variables

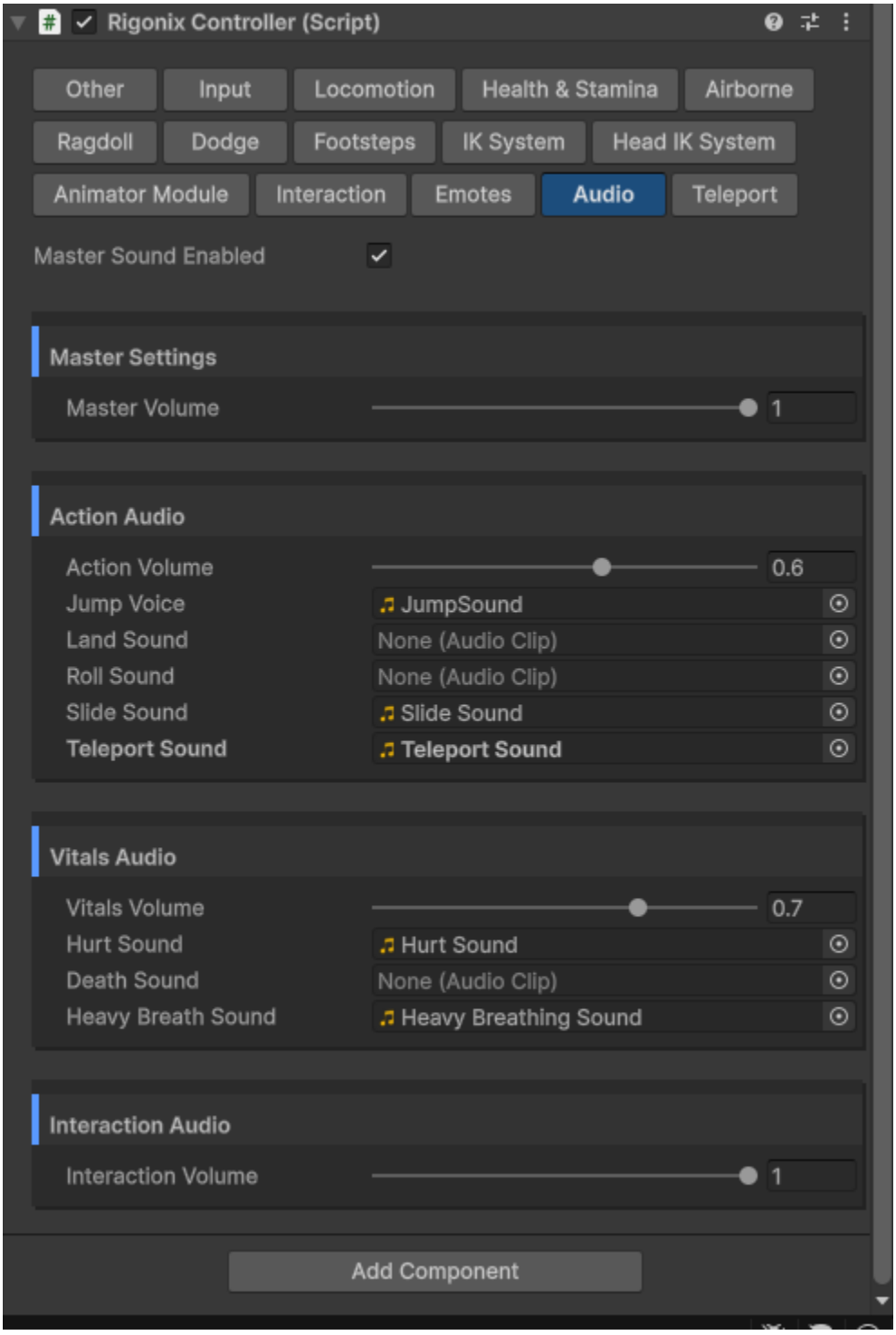
Variable	Description
Control Scheme	The Type Of Controls System Will Support Keybord Mouse: For PC Mobile: For Mobile Based
Input Backend	Legacy Input System: For Old Input System

Variable	Description
	New Input System: For using the new input system, requires the action asset.
InputBlocked	Blocks all kind of input, can be used in cutscenes and all.

Audio Module

Purpose

Controls character-related sound effects.



Key Features

- Footsteps
- Jump / Land sounds
- Action-based audio

- Surface detection support

🔑 Important Variables

Variable	Description
Master Volume	Total Volume Of All The Sound Effects
jumpClip	Jump sound
landClip	Landing sound
audioSource	Playback source

🎯 Useful Public Functions

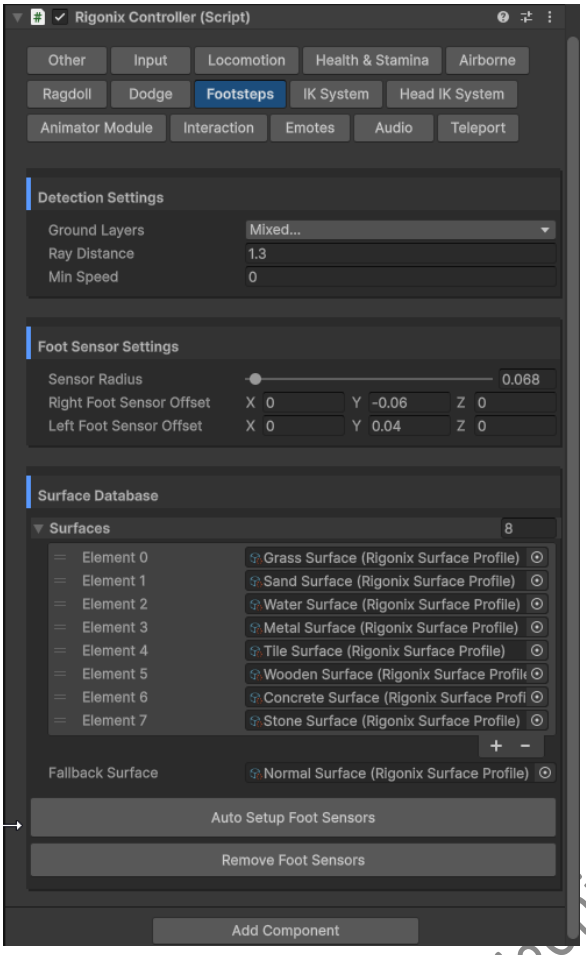
Function	Purpose
PlayInteractionSound(AudioClip)	Players any interaction sound, using the interaction volume
PlayClip(AudioClip)	Play custom sounds

Advance Systems

FootSteps System

Overview

The **FootstepModule** is a Unity script that creates realistic footstep audio and visual effects based on surface detection. It automatically identifies what surface the character is walking on (terrain textures, material names, or tags) and plays corresponding sounds and particle effects.



Key Features

- **Surface Detection** - Automatically detects terrain textures, mesh materials, and object tags
- **Realistic Audio** - Plays randomized footstep sounds with pitch/volume variation
- **Visual Effects** - Spawns particle effects and decals (footprints)
- **Animation Event Support** - Integrates with Unity's animation system
- **Editor Tools** - One-click setup and removal of foot sensors
- **Speed Threshold** - Only plays sounds when character is moving fast enough

---

Important Settings

Detection Settings

Variable	Type	Purpose
groundLayers	LayerMask	Defines which layers can be detected as ground

Variable	Type	Purpose
rayDistance	float	How far down to raycast from feet (default: 1.3m)
minSpeed	float	Minimum movement speed to trigger footsteps (default: 0.1)

Foot Sensor Settings

Variable	Type	Purpose
sensorRadius	float	Size of the foot collision sensor (0.05-1.0)
rightFootSensorOffset	Vector3	Position offset for right foot sensor
leftFootSensorOffset	Vector3	Position offset for left foot sensor

Surface Database

Variable	Type	Purpose
surfaces	List	Collection of surface profiles (grass, wood, metal, etc.)
fallbackSurface	RigonixSurfaceProfile	Default surface when no match is found

---

How It Works

1. **Animation Event** - Your character's walk/run animation calls TriggerFootstep(0) for left foot or TriggerFootstep(1) for right foot
2. **Surface Detection** - Raycasts downward to detect the ground and identifies the surface type
3. **Effect Playback** - Plays the appropriate audio clip and spawns visual effects based on the detected surface

Surface Detection Priority

The system checks surfaces in this order:

1. **Terrain Textures** - Reads Unity terrain splatmaps
2. **Material/Texture Names** - Checks the mesh's material and texture names
3. **Object Tags** - Falls back to Unity tags
4. **Fallback Surface** - Uses default if nothing matches

Editor Buttons

Button	Function
Auto Setup Foot Sensors	Automatically adds sensor components to left/right foot bones
Remove Foot Sensors	Cleans up all foot sensor components

Usage Example

1. Add surface profiles to the surfaces list (grass, concrete, wood, etc.)
2. Click "Auto Setup Foot Sensors" to configure foot bones
3. Add animation events in your walk/run animations:
    - TriggerFootstep(0) - Left foot
    - TriggerFootstep(1) - Right foot
4. Adjust sensorRadius and offsets if needed

The system will automatically play the correct sounds and effects based on what your character is walking on.

Rigonix Interaction System Documentation

How It Works - The Complete Story

Imagine your character walking through a game world. As they approach a button, a prompt appears: **"Press E to Activate"**. When they press E, the character walks up to the button, plays a pressing animation, and the button triggers - maybe opening a door or turning on lights.

This is the **Rigonix Interaction System** - and here's how it works behind the scenes:

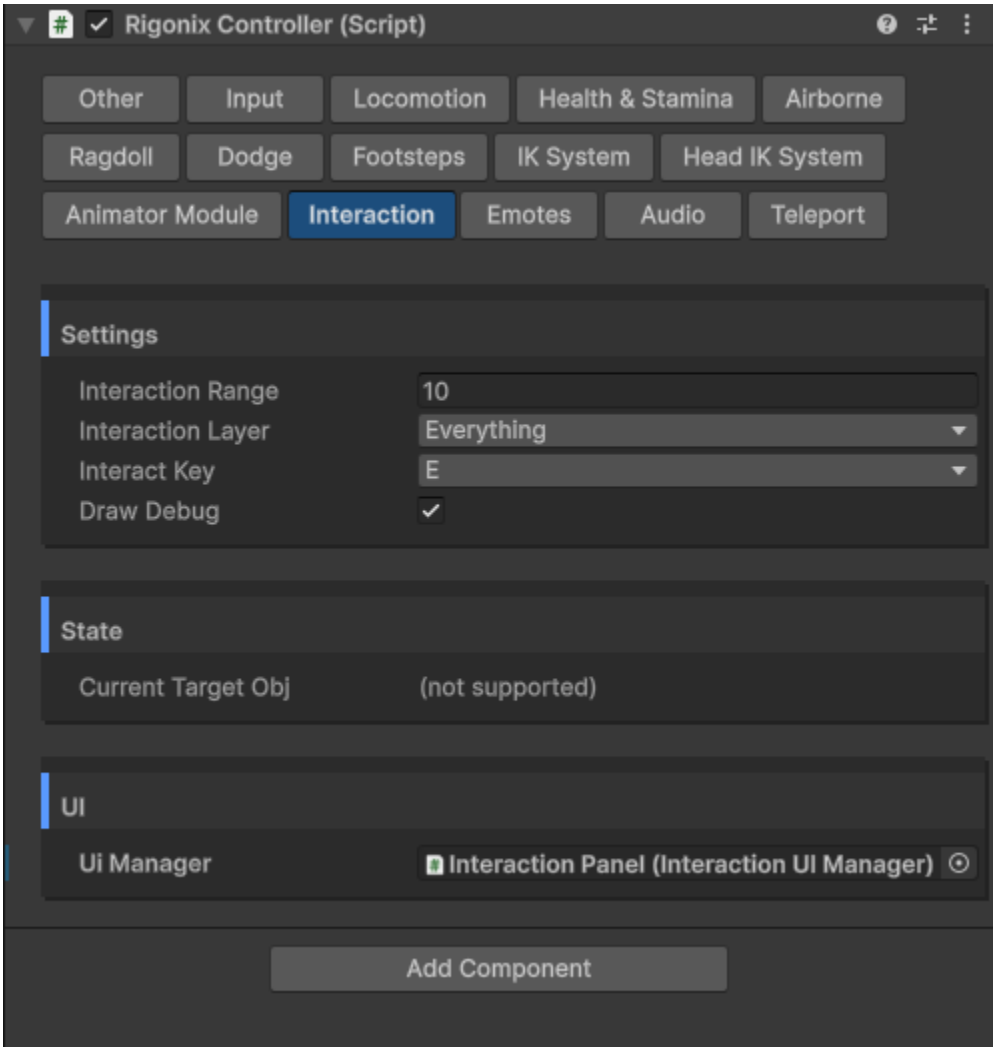
The Three-Part System

1. The Scanner (InteractionModule)

This constantly looks around the player, asking: *"Is there anything nearby I can interact with?"*

What it does:

- Scans in a radius around the player (default: 2 meters)
- Picks the object you're facing most directly
- Shows the UI prompt (e.g., "Press E to Open")
- Listens for your input (keyboard or mobile button)
- Starts the interaction when you press the key



Setting	Purpose	Example
interactionRange	How far you can interact	2.0m = arms reach
interactKey	Which button to press	E key (PC)

---

2. The Base Rules (RigonixInteractable)

This defines *how* the player approaches the object before interacting.

The Four Approach Modes:

Mode	What Happens	Example Use
None	Player stays in place	Quick pickups, switches
LookAtObject	Player just turns to face it	Examining items
AlignToStandPoint	Player walks to exact spot & rotation	Sitting in chair, using computer
MoveToDistance	Player runs close but stops at distance	Talking to NPCs

Example: A chair has a standPoint (an invisible marker) showing exactly where to stand. When you interact, the player automatically runs to that spot and rotates to face the right direction.

---

3. The Action (UniversalInteractable)

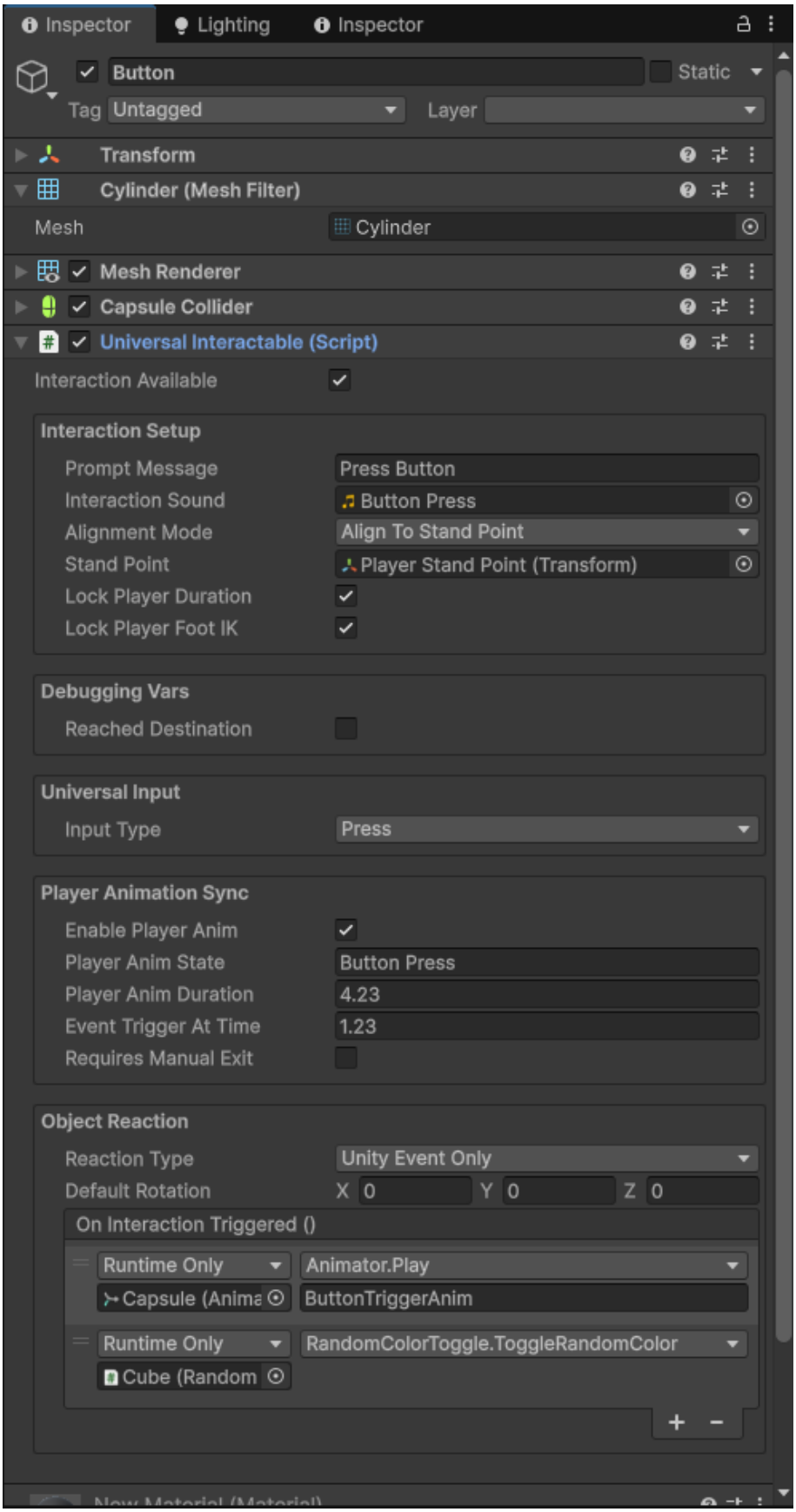
This is the actual interactable object - it defines what happens when you interact.

Real-World Examples

Example 1: Pressing a Button

Input Type: Press (one tap)
Player Animation: "PressButton" (2 seconds)
Object Reaction: Door opens (animation plays)

Story: You walk up, press E, your character reaches out and pushes the button, then the door slides open.


Inspector
Lighting
Inspector
Button
Static
Tag Untagged
Layer
Transform
Cylinder (Mesh Filter)
Mesh Cylinder
Mesh Renderer
Capsule Collider
Universal Interactable (Script)
Interaction Available
Interaction Setup
Prompt Message Press Button
Interaction Sound Button Press
Alignment Mode Align To Stand Point
Stand Point Player Stand Point (Transform)
Lock Player Duration
Lock Player Foot IK
Debugging Vars
Reached Destination
Universal Input
Input Type Press
Player Animation Sync
Enable Player Anim
Player Anim State Button Press
Player Anim Duration 4.23
Event Trigger At Time 1.23
Requires Manual Exit
Object Reaction
Reaction Type Unity Event Only
Default Rotation X 0 Y 0 Z 0
On Interaction Triggered ()
Runtime Only Animator.Play
Capsule (Anima ButtonTriggerAnim
Runtime Only RandomColorToggle.ToggleRandomColor
Cube (Random

Key Settings:

- inputType: Press
- enablePlayerAnim: true
- playerAnimState: "PressButton"
- reactionType: PlayAnimation
- objectAnimParam: "IsOpen"

Example 2: Sitting in a Chair

Alignment Mode: AlignToStandPoint
Player Animation: "SitDown" (3 seconds)
Manual Exit: true (press Escape to stand up)

Story: You press E, character walks to the chair's standpoint, plays sitting animation, stays seated until you press Escape.

Key Settings:

- alignmentMode: AlignToStandPoint
- standPoint: Position in front of chair
- requiresManualExit: true
- manualExitAnimationStateName: "StandUp"

Example 3: Holding a Lever

Input Type: Hold (2 seconds)
Object Reaction: Rotate 90 degrees

Story: You hold E for 2 seconds, the character grabs and pulls the lever, it rotates down.

Key Settings:

- inputType: Hold
- holdDuration: 2.0
- reactionType: RotateObject
- activeRotation: (90, 0, 0)

Example 4: Secret Code Entry

Input Type: Tap (5 taps required)
Object Reaction: Unlock safe

Story: You must tap E exactly 5 times to unlock a hidden safe.

Key Settings:

- inputType: Tap
- requiredTaps: 5
- reactionType: UnityEventOnly
- OnInteractionTriggered: UnlockSafe()

Key Settings Reference

Input Types

Type	Description	Use Case
Press	Single key press	Buttons, doors, pickups
Hold	Hold for X seconds	Charging actions, heavy objects
Tap	Press X times rapidly	Codes, rhythm games

Reaction Types

Type	What It Does	Example
PlayAnimation	Triggers animation on object	Door opens, chest opens
RotateObject	Rotates to new angle	Lever, valve, switch
ToggleGameObject	Show/hide object	Hidden passage
UnityEventOnly	Custom code only	Any custom logic

Player Animation Sync

Setting	Purpose
playerAnimState	Which animation to play
playerAnimDuration	How long animation lasts
eventTriggerAtTime	When to trigger object (0.5 = halfway through)
requiresManualExit	Keep player locked until Escape pressed

Thank You!

Thank you for choosing **Rigonix** for your game development needs! We hope this this person controller system helps bring your game world to life and makes creating immersive player experiences easier than ever.

Final Notes

This documentation covers the core functionality of the Rigonix Third Person Controller System. If you have questions or need additional support, don't hesitate to experiment with different settings - that's the best way to learn!

Get Free Animations!

Visit **Rigonix3d.com** to access a growing library of **free animations** and resources for your Unity projects. Whether you need locomotion, interactions, or character movements - we've got you covered!

Happy developing! 🎮

— The Rigonix Team

Have any queries? Contact us at rigonix3d.com