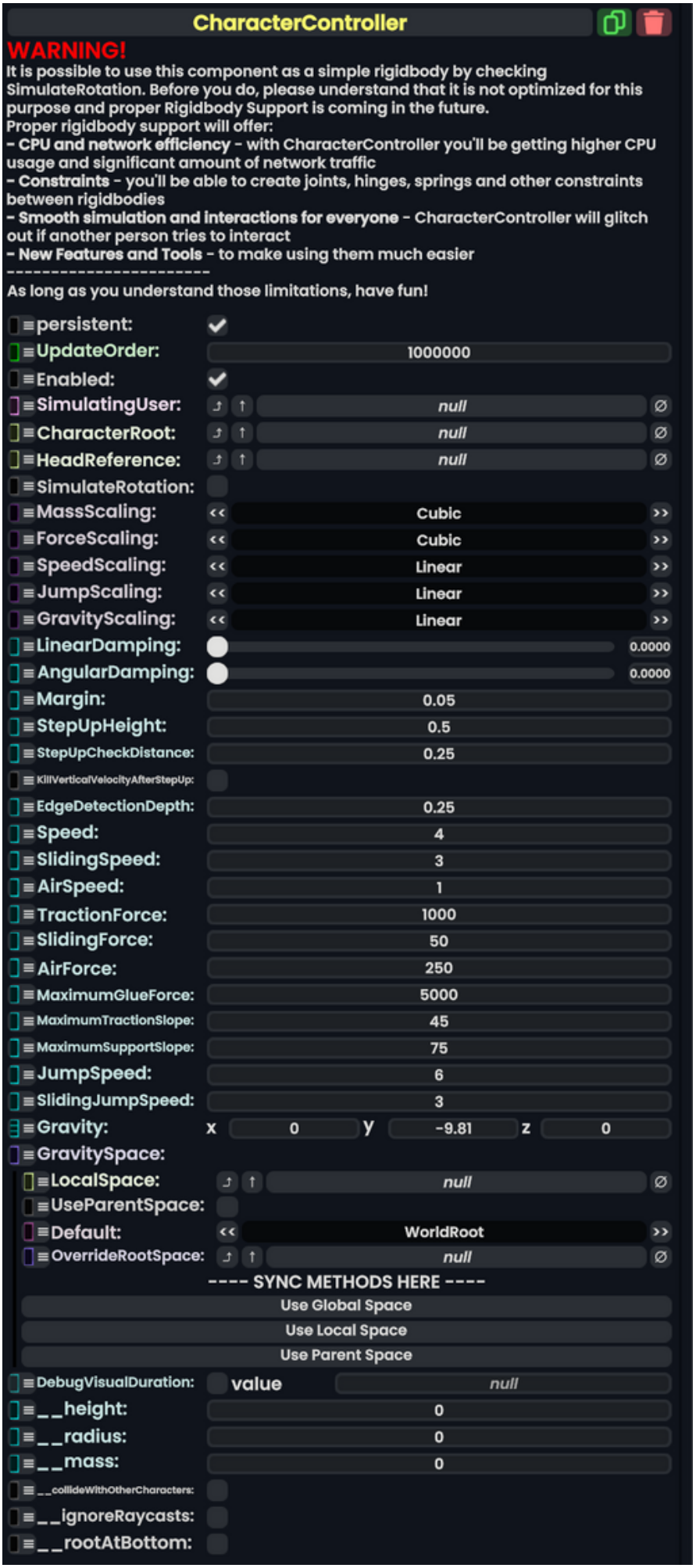
https://wiki.resonite.com/Component:CharacterController

Component:CharacterController

This article or section is a stub. You can help the Resonite wiki by expanding it.

Component image


CharacterController
WARNING!
It is possible to use this component as a simple rigidbody by checking SimulateRotation. Before you do, please understand that it is not optimized for this purpose and proper Rigidbody Support is coming in the future.
Proper rigidbody support will offer:
- CPU and network efficiency - with CharacterController you'll be getting higher CPU usage and significant amount of network traffic
- Constraints - you'll be able to create joints, hinges, springs and other constraints between rigidbodies
- Smooth simulation and interactions for everyone - CharacterController will glitch out if another person tries to interact
- New Features and Tools - to make using them much easier
As long as you understand those limitations, have fun!
persistent:
UpdateOrder: 1000000
Enabled:
SimulatingUser: null
CharacterRoot: null
HeadReference: null
SimulateRotation:
MassScaling: Cubic
ForceScaling: Cubic
SpeedScaling: Linear
JumpScaling: Linear
GravityScaling: Linear
LinearDamping: 0.0000
AngularDamping: 0.0000
Margin: 0.05
StepUpHeight: 0.5
StepUpCheckDistance: 0.25
KillVerticalVelocityAfterStepUp:
EdgeDetectionDepth: 0.25
Speed: 4
SlidingSpeed: 3
AirSpeed: 1
TractionForce: 1000
SlidingForce: 50
AirForce: 250
MaximumGlueForce: 5000
MaximumTractionSlope: 45
MaximumSupportSlope: 75
JumpSpeed: 6
SlidingJumpSpeed: 3
Gravity: x 0 y -9.81 z 0
GravitySpace:
LocalSpace: null
UseParentSpace:
Default: WorldRoot
OverrideRootSpace: null
---- SYNC METHODS HERE ----
Use Global Space
Use Local Space
Use Parent Space
DebugVisualDuration: value null
__height: 0
__radius: 0
__mass: 0
__collideWithOtherCharacters:
__ignoreRaycasts:
__rootAtBottom:

Character Controller component as seen in the Scene Inspector

Description

The Character Controller component is primarily designed to be part of the Locomotion system and is what used for user movement and simulating physics relating to this.

It has many settings and values that can alter the way on how a user can control either their self or another slot (For example: Vehicles). These Settings allow you to control the mass, speed, force, etc. This flexibility also allows for uses far beyond it's original intended design.

> ?
> When getting the character controller of a user, avoid using the character controller under the user root slot. Instead use the node Find Character Controller From User.

Keep in mind that the character controller under the user root slot is currently useless (as in, not used besides it needing to be there) at this time. And removing it just recreates it back on the user root slot.

For this component, when using some features, you may need to either write/drive the field `SimulatingUser`. This will activate the character controller on the slot and make it start simulating. Doing this also lets you use some nodes such as Is Character Controller and Is Character On Ground, and without a simulating user, the values from some nodes would not be updated.

Fields

Fields

Name	Type	Description
`persistent`	**Bool**	Determines whether or not this item will be saved to the server.
`UpdateOrder`	**Int**	Controls the order in which this component is updated.
`Enabled`	**Bool**	Controls whether or not this component is enabled. Some components stop their functionality when this field is disabled, but some don't.
`SimulatingUser`	**User**	The user who is currently simulating and networking this Character Controller.
`CharacterRoot`	**Slot**	The slot to simulate, can be any inputted slot.

HeadReference	**Slot**	Dictates where to simulate the collider and based on the slot's forward direction, will decide what a "forward" movement is based on your joystick/input.
SimulateRotation	**Bool**	Allow the object simulated by this object to rotate, essentially making it a rigid body. See warning.
MassScaling	**PhysicsScalingMode**	How to scale the mass based on this component's slot scale.
ForceScaling	**PhysicsScalingMode**	How to scale the force based on this component's slot scale.
SpeedScaling	**PhysicsScalingMode**	How to scale the speed based on this component's slot scale.
JumpScaling	**PhysicsScalingMode**	How to scale the jump force based on this component's slot scale.
GravityScaling	**PhysicsScalingMode**	How to scale the gravity effect based on this component's slot scale.
LinearDamping	**Float**	how fast the character controller slows down when no other forces are being applied to it
AngularDamping	**Float**	how fast thus character slows down its rotation when no other forces are being applied to it.
Margin	**Float**	Unused.
StepUpHeight	**Float**	How big of a ledge the character controller can step up
StepUpCheckDistance	**Float**	determines the distance the shape is swept in the movement direction to determine if it's free of other obstacles
KillVerticalVelocityAfterStepUp	**Bool**	will convert any vertical velocity into directional one in the current planar direction of the movement and push the player back down, preventing the player from getting too much air when running off a ramp or stairs.

EdgeDetectionDepth	**Float**	the smallest size an edge can be that the character can stand on while against a wall
Speed	**Float**	How fast this character controller can move.
SlidingSpeed	**Float**	how fast the character controller moves when sliding down a slope
AirSpeed	**Float**	how fast the character moves in the air
TractionForce	**Float**	How much force is needed to overcome standing friction.
SlidingForce	**Float**	How much force is needed to stay moving
AirForce	**Float**	the force needed to move in the air
MaximumGlueForce	**Float**	Reduces the velocity of breaking contact from a surface vertically be under this speed. This is useful for simulating sticky surfaces, where the user cannot jump as high.
MaximumTractionSlope	**Float**	Maximum Traction angle (degrees) before user slides down a slope, even if they are actively trying to walk up it.
MaximumSupportSlope	**Float**	Maximum Support angle (degrees) before user slides down a slope, when not working against the slope.
JumpSpeed	**Float**	The speed of the velocity applied when jumping on a slope with an angle less than or equal to MaximumTractionSlope.
SlidingJumpSpeed	**Float**	The speed of the velocity applied when jumping on a slope with an angle greater than MaximuSupportSlope.
Gravity	**Float3**	Force of Gravity for this character controller
GravitySpace	*direct* **RootSpace**	The coordinate space in which Gravity is applied in. applying gravity to world space using this field, can allow a user to stay upright despite a floor they're parented under is being rotated. which generates interesting effects.

DebugVisualDuration	**Nullable`1<Float>**	Allows for the component to visualize the force vectors acting on the character simulator if set above 0.
__height	**Float**	Height of the driven CapsuleCollider (Component) Currently does nothing.
__radius	**Float**	Radius of the driven CapsuleCollider (Component) Currently does nothing.
__mass	**Float**	Mass of the of the driven CapsuleCollider (Component) Currently does nothing.
__collideWithOtherCharacters	**Bool**	Currently does nothing.
__ignoreRaycasts	**Bool**	Currently does nothing.
__rootAtBottom	**Bool**	Currently does nothing.

Usage

This component is usually used internally by Resonite to simulate user movement physics like walking and climbing.

But the component can also be manipulated to create "Fake Players" commonly known by Resonite users as "NPC's". Before doing this, know that the physics simulation is always inconsistent when viewed by different players in the same session. It also causes high amounts of networking and FPS lag. Use too many and you may start rising in Queued Packets.

The character controller can also be used as a ball and simulates rolling, falling, collisions and momentum transfer, but for individual things that fly or bounce, using a Trajectory Position ProtoFlux node is vastly superior.

This component is also used in certain types of vehicles and making them controllable by the user. This is done by having a vehicle you want to make controllable, then making an anchor for the user to sit in, then setting up this component to make that user become the simulating user, thus making their controls change in such a way where they are controlling a vehicle they are sitting on.

The CharacterRoot slot input when null will do its transforms on the slot the CharacterController component is on, however when given a slot the component will instead make those transforms happen on the slot inputted. Combined with defining the HeadReference, you can control external slots without moving where the CharacterController component is and have them work with physical obstacles. For a quick example of this behavior, you can modify your current active locomotion's CharacterController on the CharacterRoot field to point towards another slot while defining the HeadReference to instead move the inputted slot while you stand still.

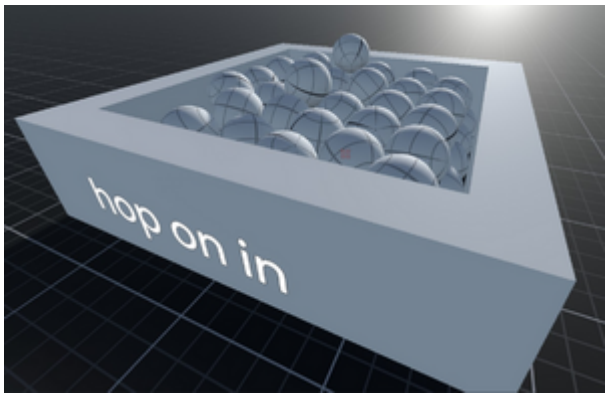
This component can use sphere, box and other colliders that have volume in addition to the capsule collider.

Note, due to the unique behavior of the HeadReference input defining where the

CharacterController's collider is in the simulation, visualizing the collider via the inspector will not be correct.

Examples

A Shrike NPC running around the cloud home:



A simple Ball Pit made with CharacterController demonstrating physics. (Worlds: 'ball pit')

See also

Protoflux Notes

Using Character Controller User will give you the Simulating User of this Character Controller Component.

Using Get User From Component will give you the Owner User of this Character Controller Component.

Using Allocating User will give you the User that spawned out a slot that has this Character Controller Component.

Components

The Physical Locomotion modules use a Capsule Collider who's bounds are generated from

the SingleShapeCharacterControllerManager component. This controls how an avatar interacts with the world.