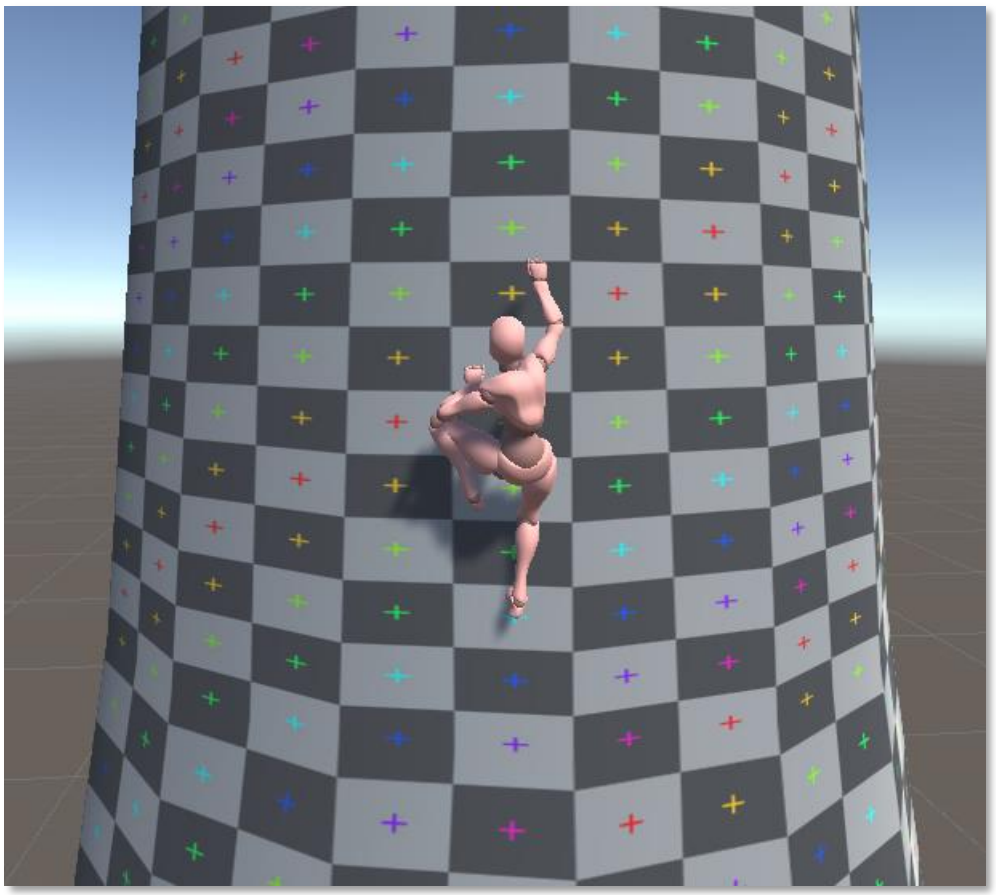
FCS – Fur Climbing System

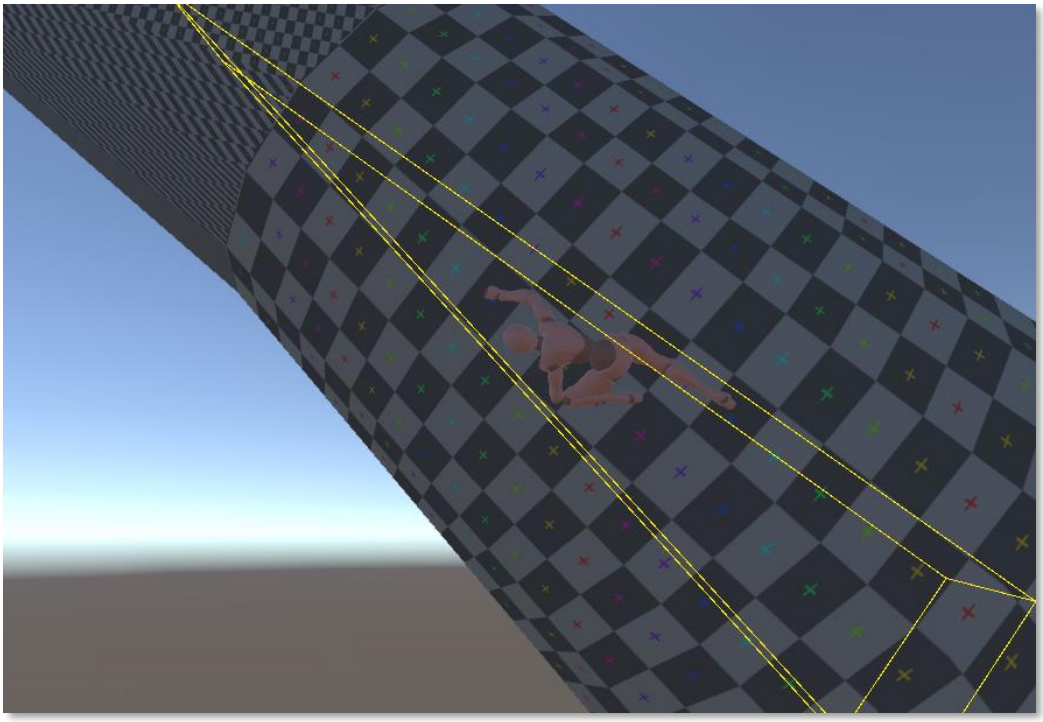
User Guide

v1.8.3

Contents

1. Welcome to FCS - Fur Climbing System





1.1. Introduction

This package will bring a solution for free climbing in some places like: colossus fur, mosses, etc.

You are free to move the player however you like, i.e. as the player moves on the climbable object, the engine automatically attaches the player to the current triangle of the climbable object, thus allowing the player to stay fixed in position current of the climbable object.

Feature requests and support: https://discord.gg/R5S7WeWyqc

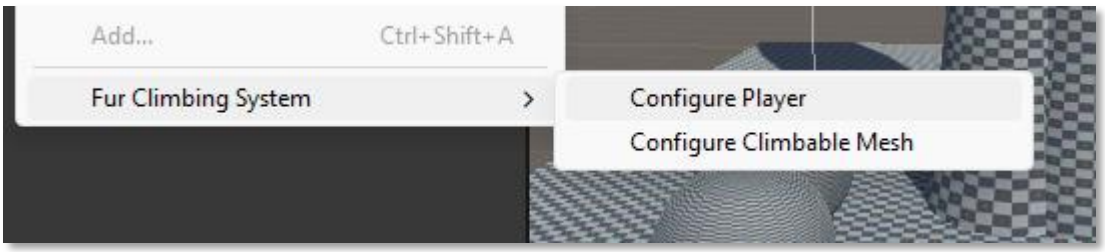
2. Getting Started

Our package is ready-to-use, that is, you can configure your player and climbable mesh by one click, through the Unity menu or by dragging the Fur Climbing System prefab.

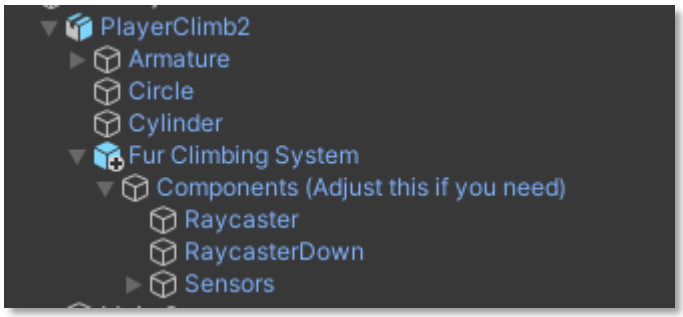
2.1. Ready-to-use Player configuration

First select your player.

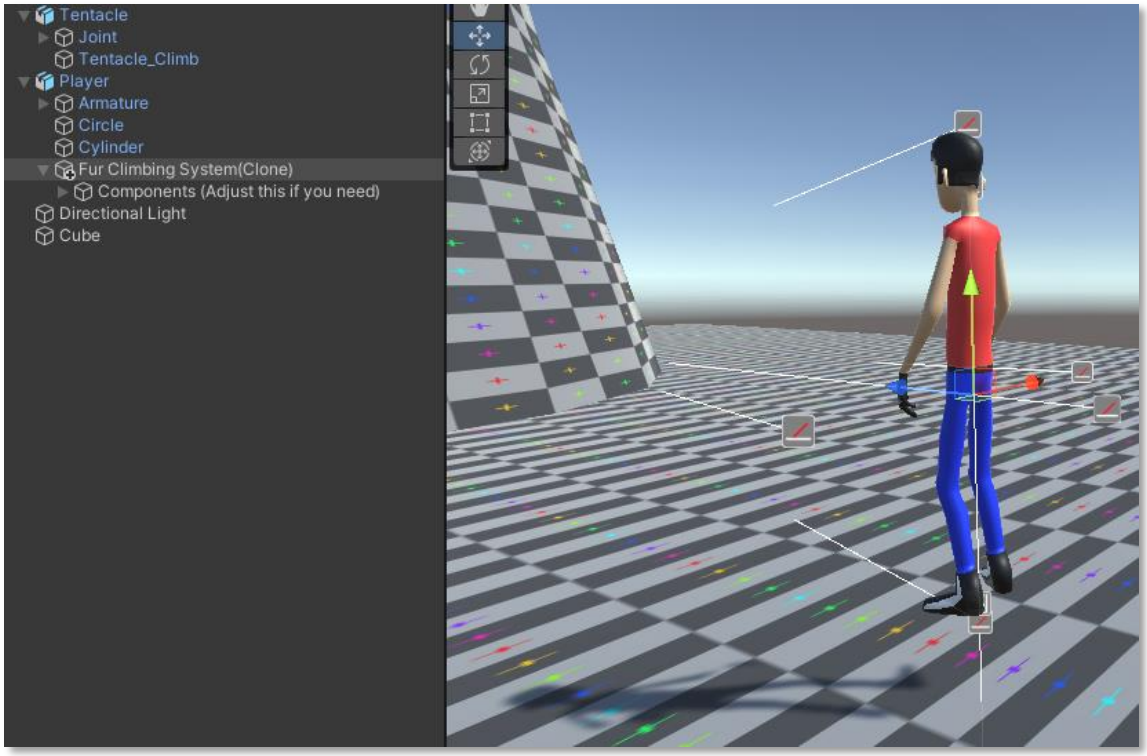
Select Component > Fur Climbing System > **Configure Player** option to configure your player.



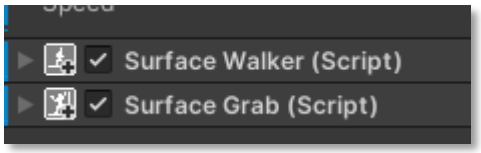
The prefab needed for climbing to work will be added inside your player.



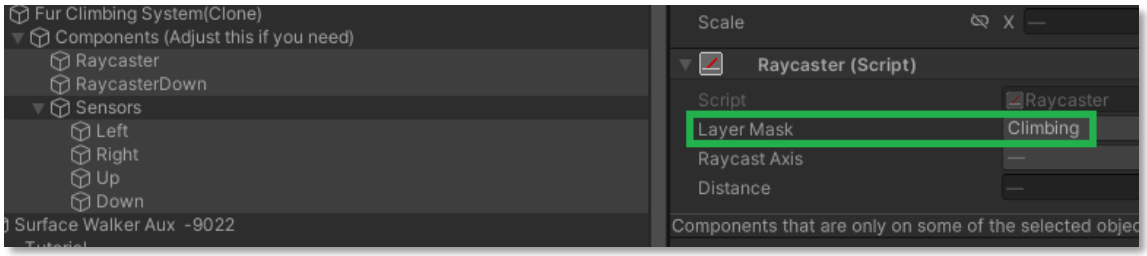
The expectation is that the prefab is aligned in the center of your Character during **Play-mode**.



Two components will be added to your player, so you can configure the component's properties according to your desire.



You will have to configure a Layer to detect the climbable meshes.



Obviously, the climbing activation logic should be implemented by you and according to your needs and that of your character. In the example below we can see that to perform the climb the player must hold down the right mouse click, if the player releases the mouse, the character will release the climbable mesh and stop climb animations. You can read the example of the PlayerDemo.cs file to learn how climb works.

```
Script do Unity (3 referências de ativo) | 6 referências
public class PlayerDemo : MonoBehaviour
{
    public SurfaceGrab surfaceGrab;
```

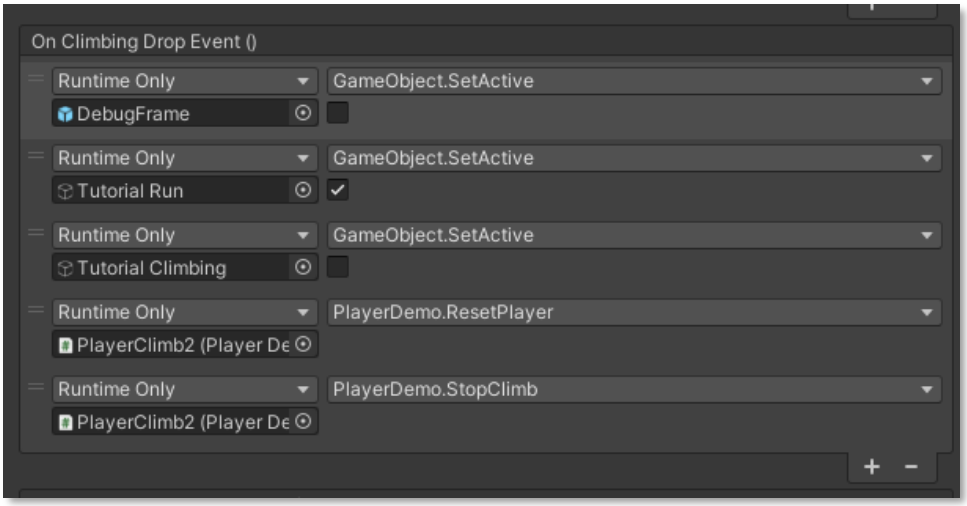
```
if (Input.GetKey(KeyCode.Mouse1))
{
    surfaceGrab.GrabSurface();
}
else
{
    surfaceGrab.Drop();
```

The code below detects if there is no "hole" above the player. This sensor is present in the prefab Fur Climbing System. His logic can be implemented in the movements of Climb Right, Left, Up and Down too.

```
public Sensor sRight, sLeft, sUp, sDown;
```

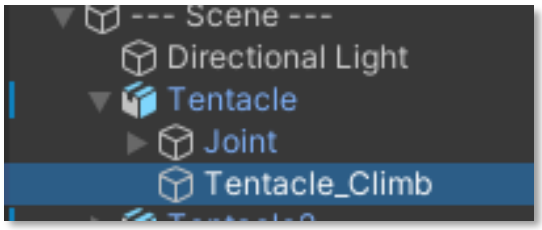
```
if (Input.GetKey(KeyCode.W) && sUp.IsDetecting)
{
    animator.SetBool("Up", true);
}
else
{
    animator.SetBool("Up", false);
}
```

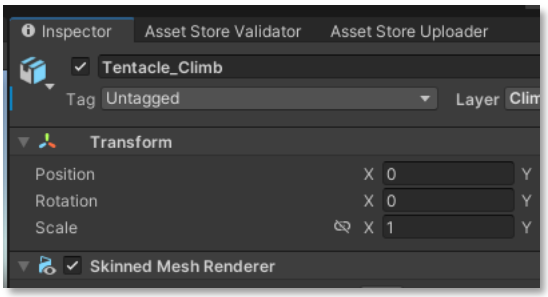
Example below of using the event "On Climbing Drop" in Surf Grab.

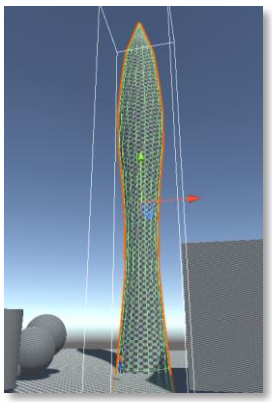


3.1. Ready-to-use Climbable Mesh configuration

First select your target mesh.

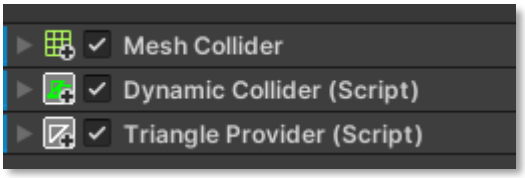




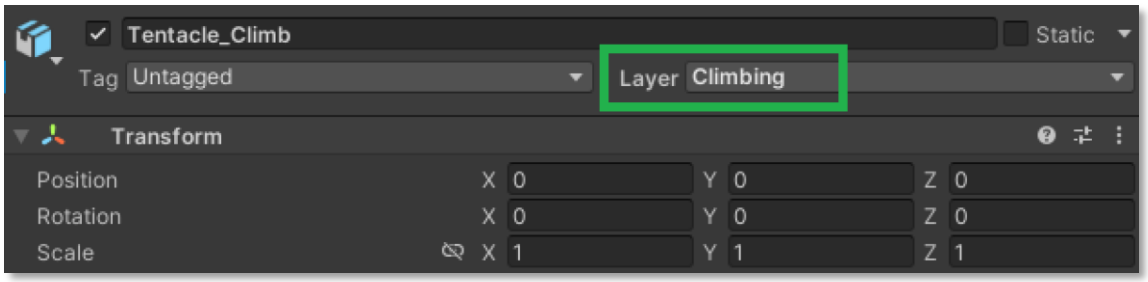


Select Component > Fur Climbing System > **Configure Climbable Mesh** option to configure your mesh. Note that this option will enable the "Read/Write" option on the mesh in the Unity Model Importer.

Three components will be added to your mesh, so you can configure the component's properties according to your desire.



You can set the layer to be detected by the player's SurfaceGrab and SurfaceWalker.



By configuring the player and the climbable mesh, your player is already able to climb the mesh.

3. Understanding The Components

Obs.: This section is just reading to show how the components work and how it would be to manually configure the component. As the package is ready-to-use, just use section 2. Getting Started to configure your climb.

3.1. Adding Components to climbable object

In this topic we will add the necessary components for climbing. Obviously, note that the mesh of your model must follow the following criteria:

- Have a good quality
- Have a good weight mapping
- No distortions occur in the triangles
- Climbable area should be a separate object, so you can add the components without affecting the rest of the colossus' body
- Disable some physics components (Example: CharacterController, Rigidbody, etc) while climbing so gravity and other effects don't interfere.

Some of these logics will have to be implemented manually, as they depend on the developer's will:

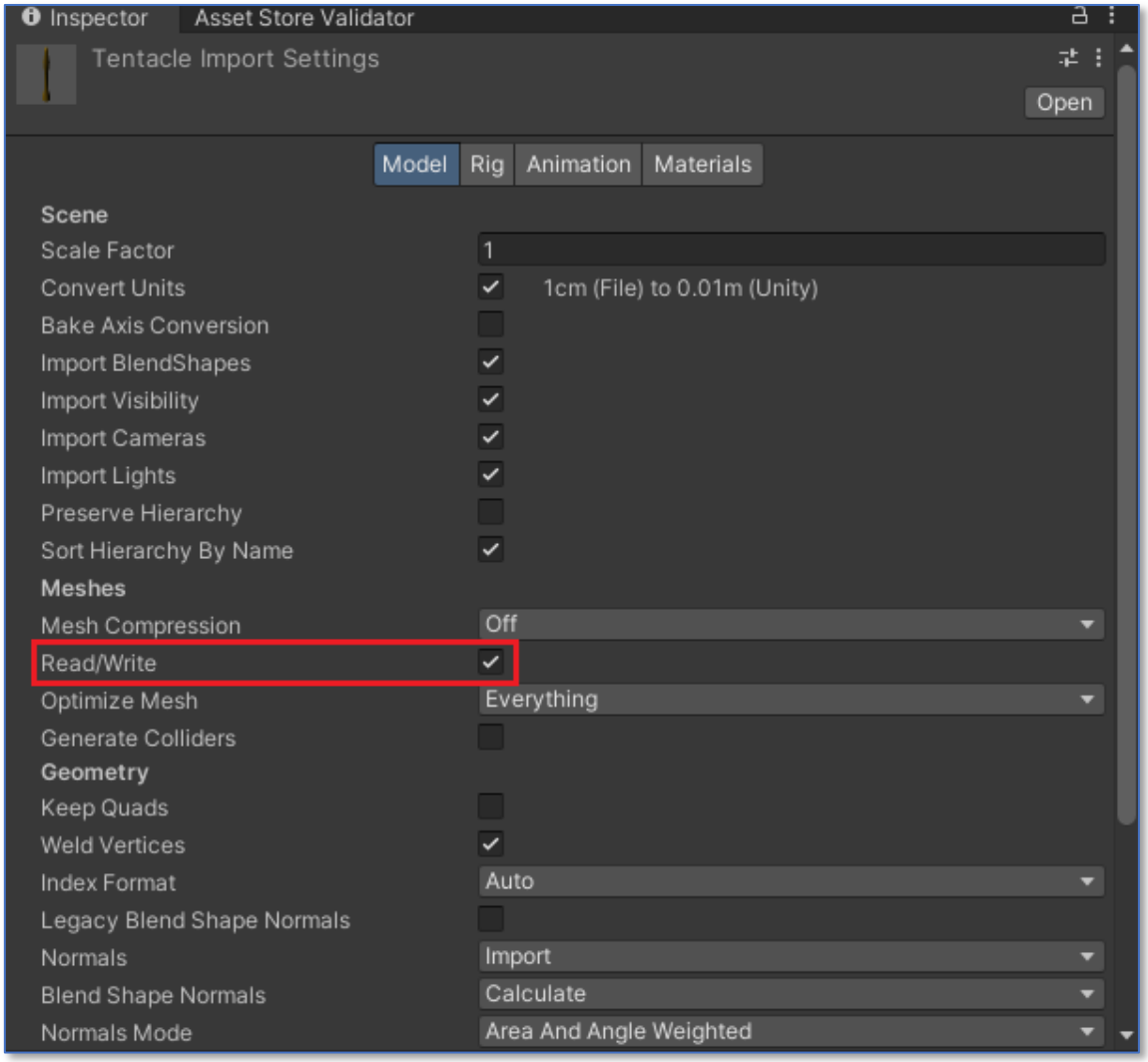
- Climbing restriction sensors
- Inverse Kinematics
- Pendulum
- Imbalance
- Player
- Disable or configure other components that will interfere with climbing.

Examples of climbable objects:

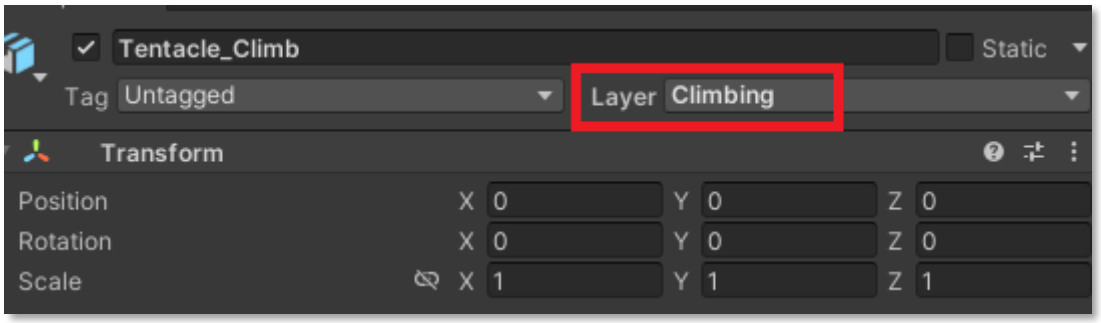
- Mosses
- Boss's fur

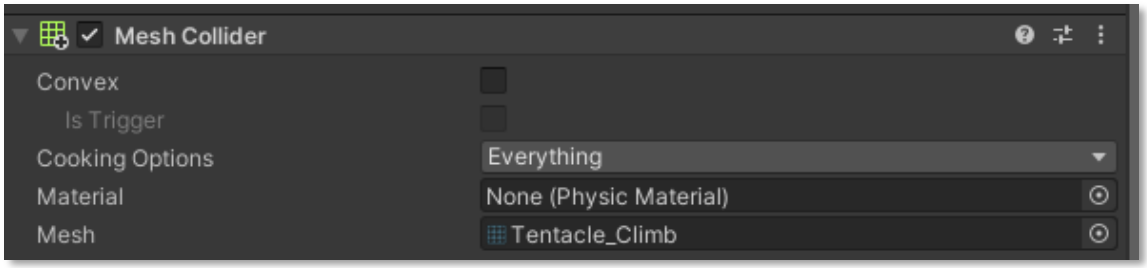
3.1.1 Dynamic Collider

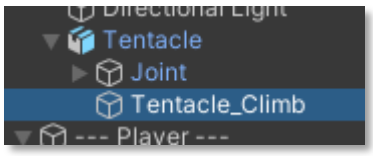
First, for the climbing to work you must set the model's Read/Write property to true.



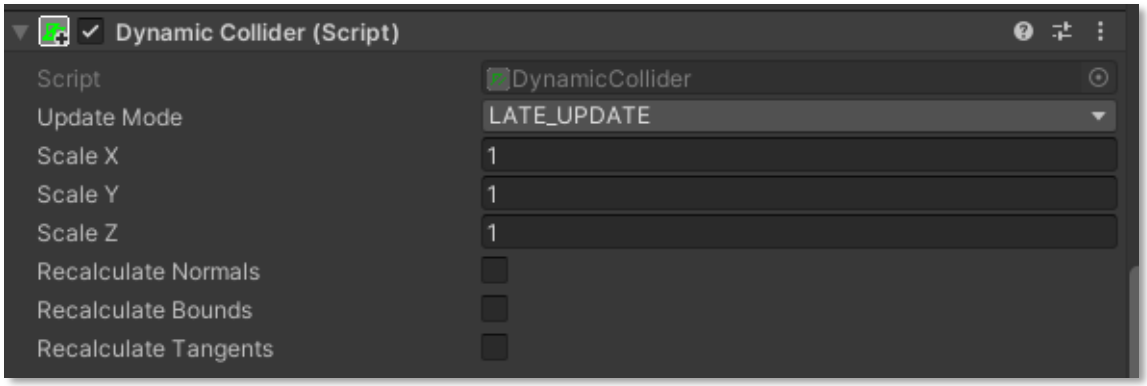
Configure a layer of your choice.

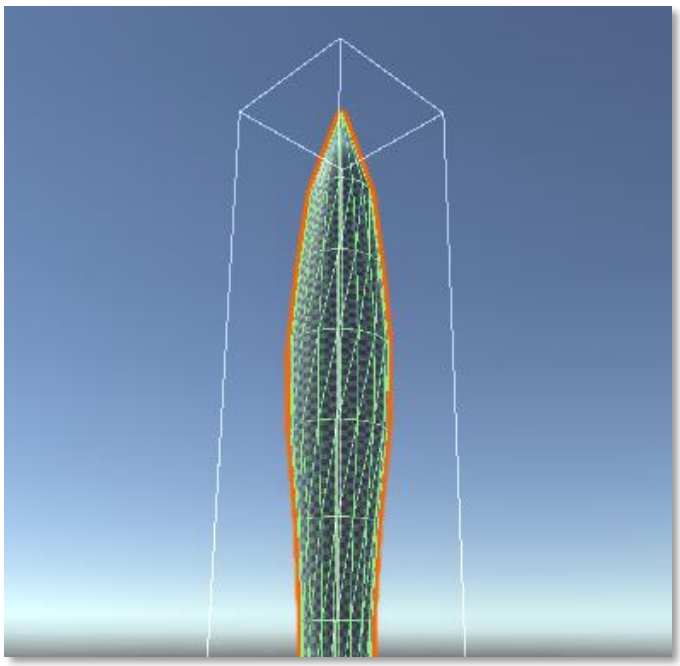


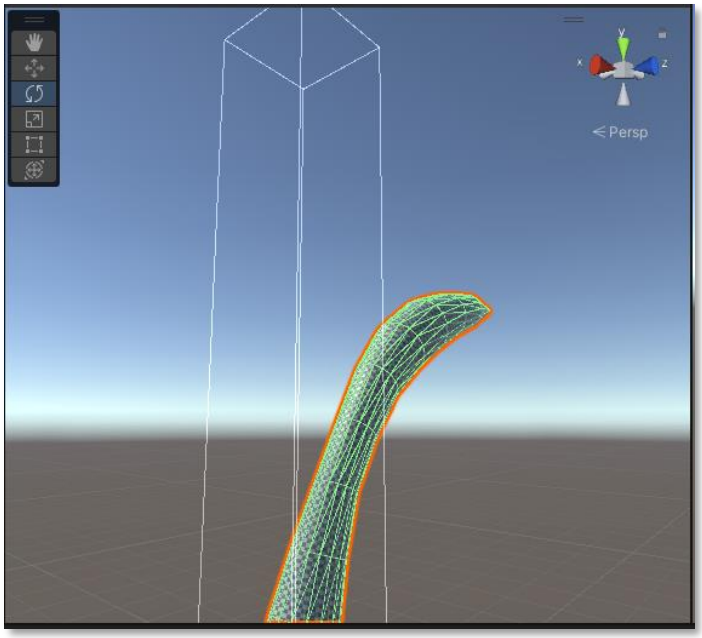




The DynamicCollider component will cause the object's MeshCollider to be updated and adjusted according to the mesh deformation. This will make it possible for the climbing Raycast to be able to follow the mesh deformation.

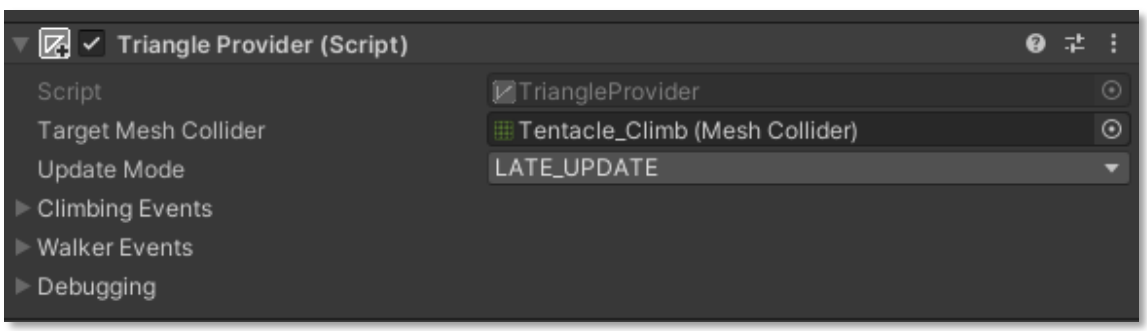






3.1.2 TriangleProvider

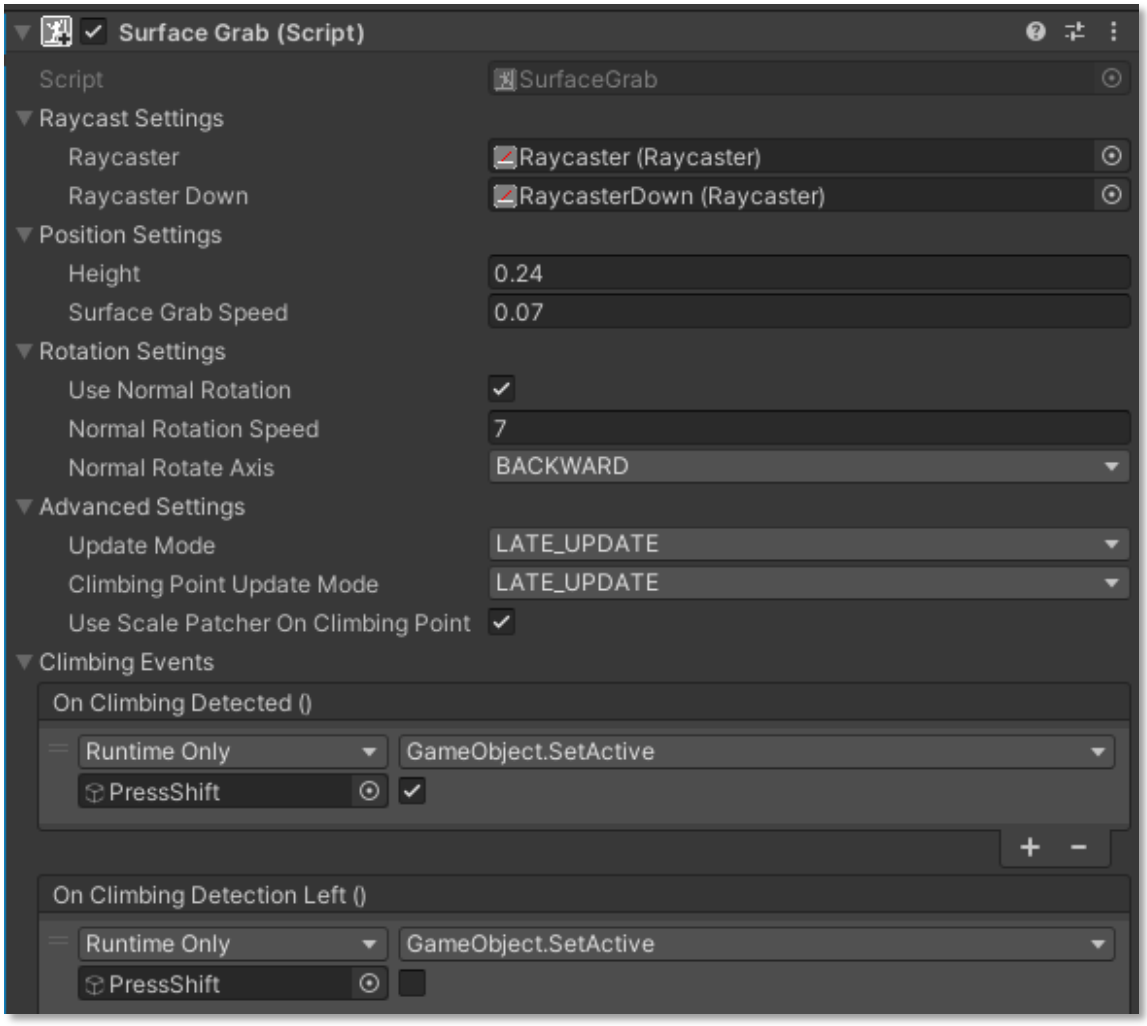
The TriangleProvider is the component responsible for getting all the mesh vertices and delivering them to other components. This component must be added for the system to work.

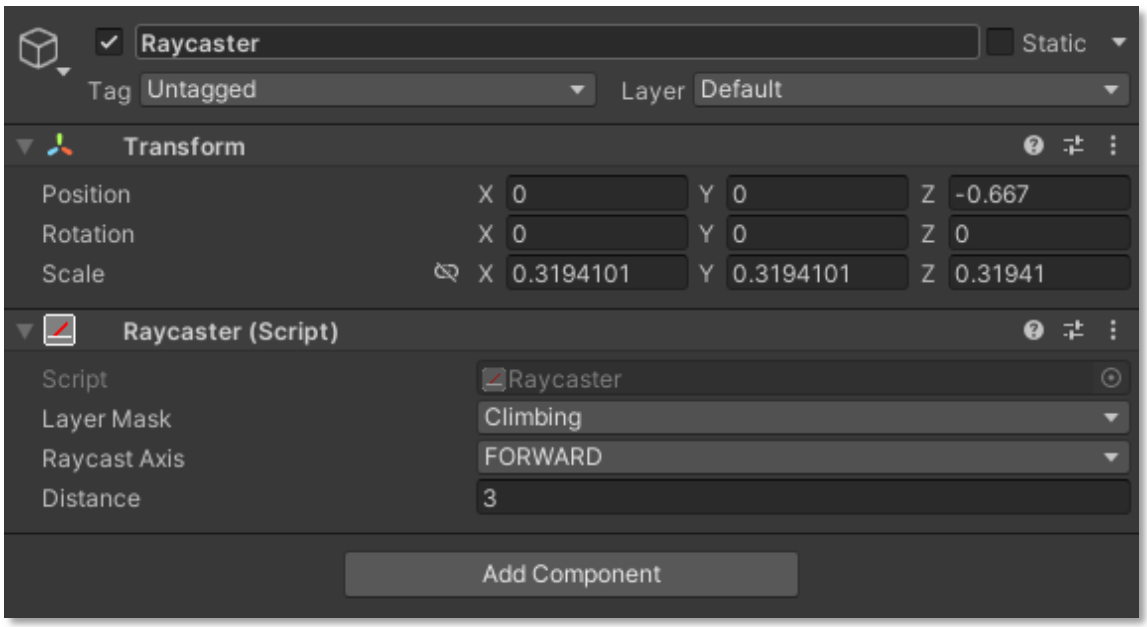


3.2. Adding Components to player

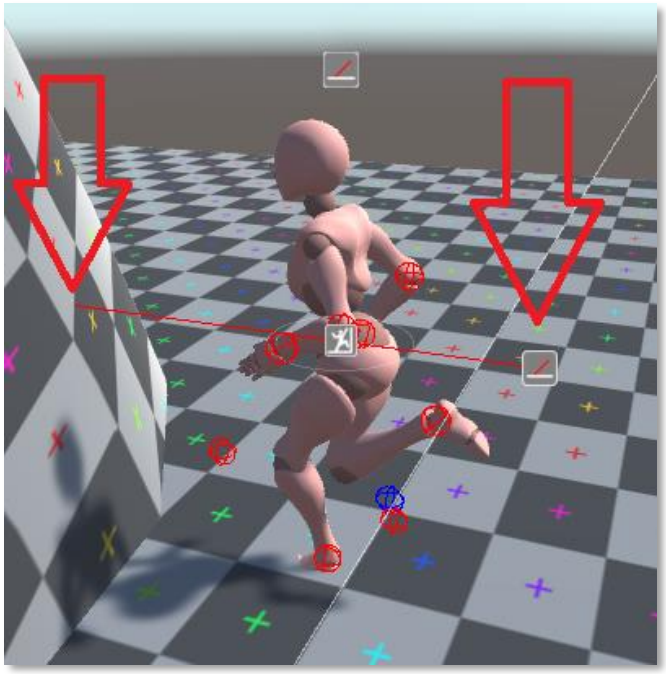
3.2.1 SurfaceGrab

The SurfaceGrab component will be attached to the player or to an object that will climb the climbable object. This component is responsible for attaching the player to the current triangle of the climbable object and also "sticking" the player to the climbable object as the player moves.

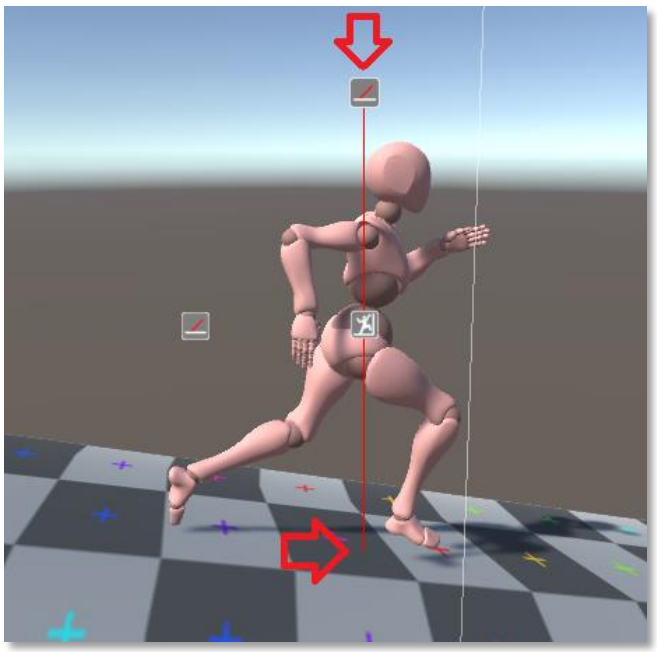




Raycaster: The Raycaster will be an object that will be behind the player, that is, it is the object where a raycast will be sent with the objective of fixing the player on the climbable object. It is recommended that the Raycaster be behind the player and with zero positions.



Raycaster Down: Same as Raycaster, but will check if the player is stepping on a climbable object.



Layer Mask: The Layer Mask property is where the layer to be climbed will be configured, that is, which layer the Raycast can reach in order to fix the player on the climbable object.

Raycast Axis: The Raycast Axis property will tell you which axes the Raycast will cast the Raycast through. In the image above it shows that the Raycast is leaving the Axis forward (blue) for example.

Distance: The distance at which the Raycast's Raycast will reach.

Height: The height at which the player will stand above the climbable object.

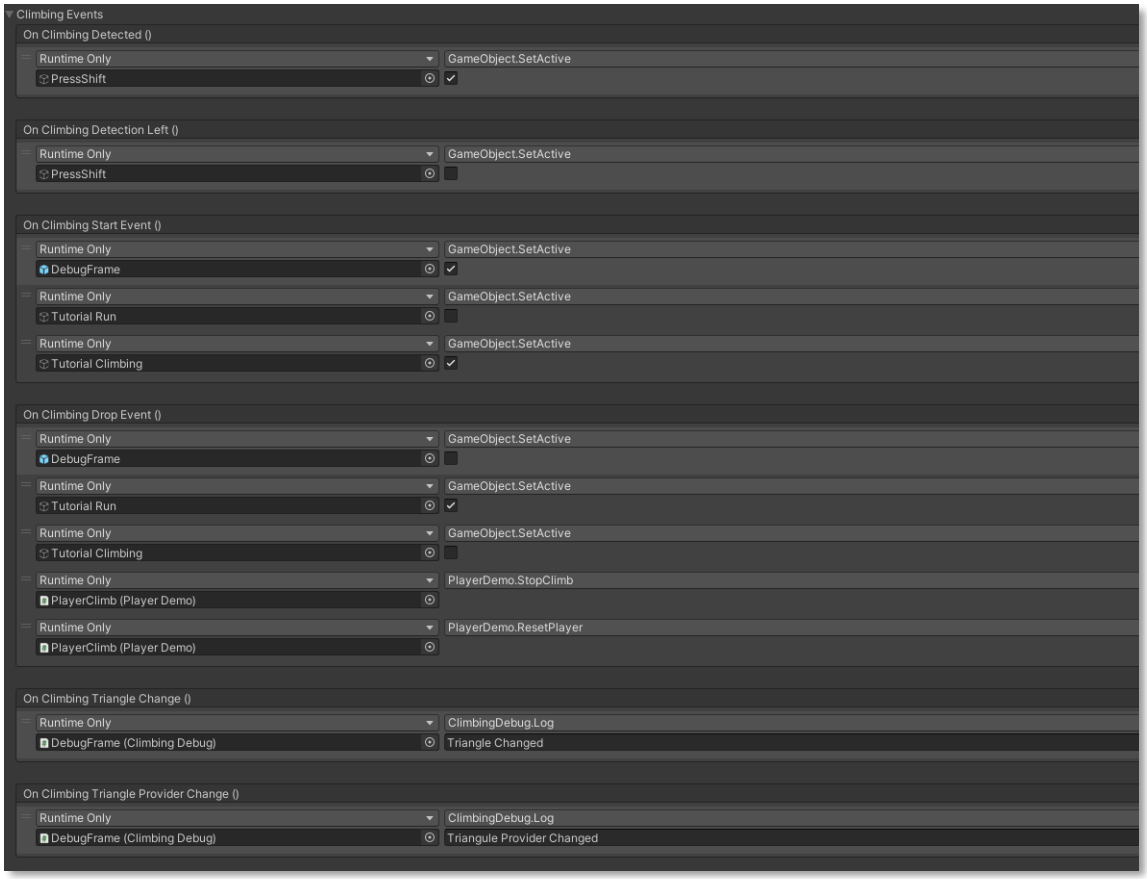
Use Normal Rotation: Says if the player will have their rotation adjusted according to the relief of the climbable object. The rotation speed can be configured in **Normal Rotation Speed**. The developer will also be able to configure the Axis of the player in which the rotation will take effect in the **Normal Rotate Axis option**.

Surface Grab Speed: Configures the speed at which the player sticks to the mesh, that is, in other words, this is the speed at which the climbable mesh attracts the player to the surface.

Climbing Events: They are where the actions that will be called during certain climbing events and behaviors will be configured. For example, when climbing is released, player goes to another triangle, etc.

Use Scale Patcher On Climbing Point: This will force the Climbing Point to always be on a uniform global scale.

4. Climbing Events



On Climbing Detected: This event will be called when the player is facing a climbable object.

On Climbing Detection Left: This event will be called when the player walks away from a climbable object.

On Climbing Start Event: This event will be called when the player starts climbing an object.

On Climbing Drop Event: This event will be called when the player stops climbing.

On Climbing Triangle Change: When a player is moving while climbing and he migrates to another triangle, this event is called.

On Climbing Triangle Provider Change: This event will be called when during climbing, the player goes to another climbable object.

5. Enum Types

Type	Equivalent
FCSTypes.Axis.**UP**	transform.up
FCSTypes.Axis.**DOWN**	-transform.up
FCSTypes.Axis.**FORWARD**	transform.forward
FCSTypes.Axis.**BACKWARD**	-transform.forward
FCSTypes.Axis.**RIGHT**	transform.right
FCSTypes.Axis.**LEFT**	-transform.right

Type	Description
FCSTypes.UpdateMode.UPDATE	Update is called every frame, if the MonoBehaviour is enabled.
FCSTypes.UpdateMode.FIXED_UPDATE	Frame-rate independent MonoBehaviour.FixedUpdate message for physics calculations.
FCSTypes.UpdateMode.LATE_UPDATE	LateUpdate is called every frame, if the Behaviour is enabled. LateUpdate is called after all Update functions have been called. This is useful to order script execution.
FCSTypes.UpdateMode.IENUMERATOR	This update mode will use a Coroutine to call the actions.

IENUMERATOR mode example:

```
1 referência
IEnumerator RecalculateBounds()
{
    while (true)
    {
        if (updateMode == FCSTypes.UpdateMode.IENUMERATOR)
        {
            Recalculate();
        }
        yield return null;
    }
}
```

6. Functions and Properties

6.1. SurfaceGrab.cs

Function	Description	Type
GrabSurface()	This function will cause the player to start climbing an object when available.	void
Drop()	This function will make the player drop the climb.	void
CreateClimbingPointForCurrentTriangle()	With this function you can create a Climbing Point based on the player's current triangle, if you want to attach something to the triangle.	ClimbingPoint

Example:

```
if (Input.GetKey(KeyCode.Mouse1))
{
    surfaceGrab.GrabSurface();
}
else
{
    surfaceGrab.Drop();
}
```

Property	Description	Type
IsOnClimbing	Check player is climbing.	Bool
IsClimbingAvailableNow	Check if the player can now climb near a climbable object.	Bool
CurrentTriangleIndex	Gets the index of the current triangle.	Int
CurrentTriangle	Gets the current Triangle object.	Triangle
CurrentTriangleProvider	Gets the TriangleProvider that the player is currently climbing.	TriangleProvider
CurrentObjectVelocity	Gets the Object Velocity component of the current triangle.	ObjectVelocity

6.2. SurfaceWalker.cs

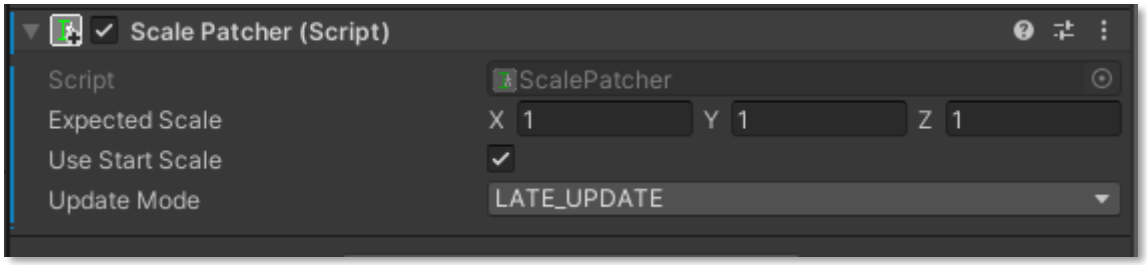
Function	Description	Type
CreateClimbingPointForCurrentTriangle()	With this function you can create a Climbing Point based on the player's current triangle, if you want to attach something to the triangle.	ClimbingPoint

Property	Description	Type
IsWalkingInMesh	Checks if the player is currently walking over a climbable mesh.	Bool
CurrentTriangleIndex	Gets the index of the current triangle.	Int
CurrentTriangle	Gets the current Triangle object.	Triangle
CurrentTriangleProvider	Gets the TriangleProvider that the player is	TriangleProvider

	currently climbing.	
CurrentObjectVelocity	Gets the Object Velocity component of the current triangle.	ObjectVelocity
CurrentSlopeAngle	Obtain the angle of inclination of the object the player is standing on.	float

6.3. ScalePatcher.cs

ScalePatcher is a component that will keep the player always in the same scale. This is useful when the parent's scale is not uniform and ends up deforming the player's scale. Note that instead of using this, it is recommended that the scale of your model be uniform, thus not needing to use this component.



Property	Description	Type
ExpectedScale	The scale the climbing player must be on.	Vector3
useStartScale	Defines whether ExpectedScale will be the object's initial scale.	Bool
CurrentTriangle	Gets the current Triangle object.	Triangle

updateMode	Defines what the update method is.	Enum FCSTypes.UpdateMode

6.4. ObjectVelocity.cs

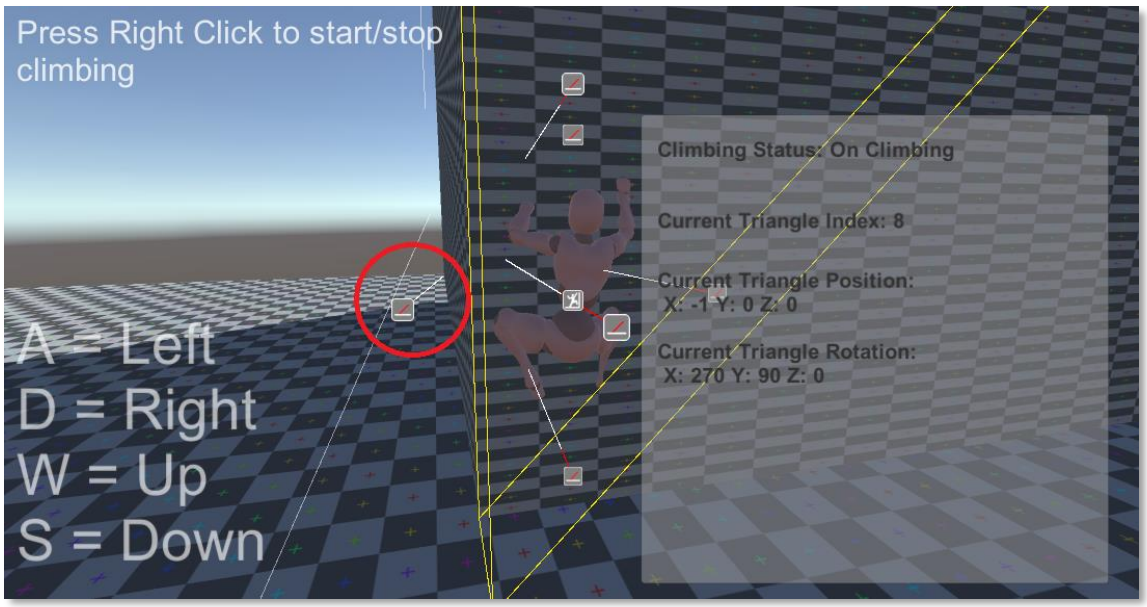
Property	Description	Type
VelocityInfo.Velocity	Get velocity of moving object.	Vector3
VelocityInfo.VelocityNormalized	Get the normalized speed.	Vector3
VelocityInfo.Magnitude	Get the magnitude of the velocity.	Float

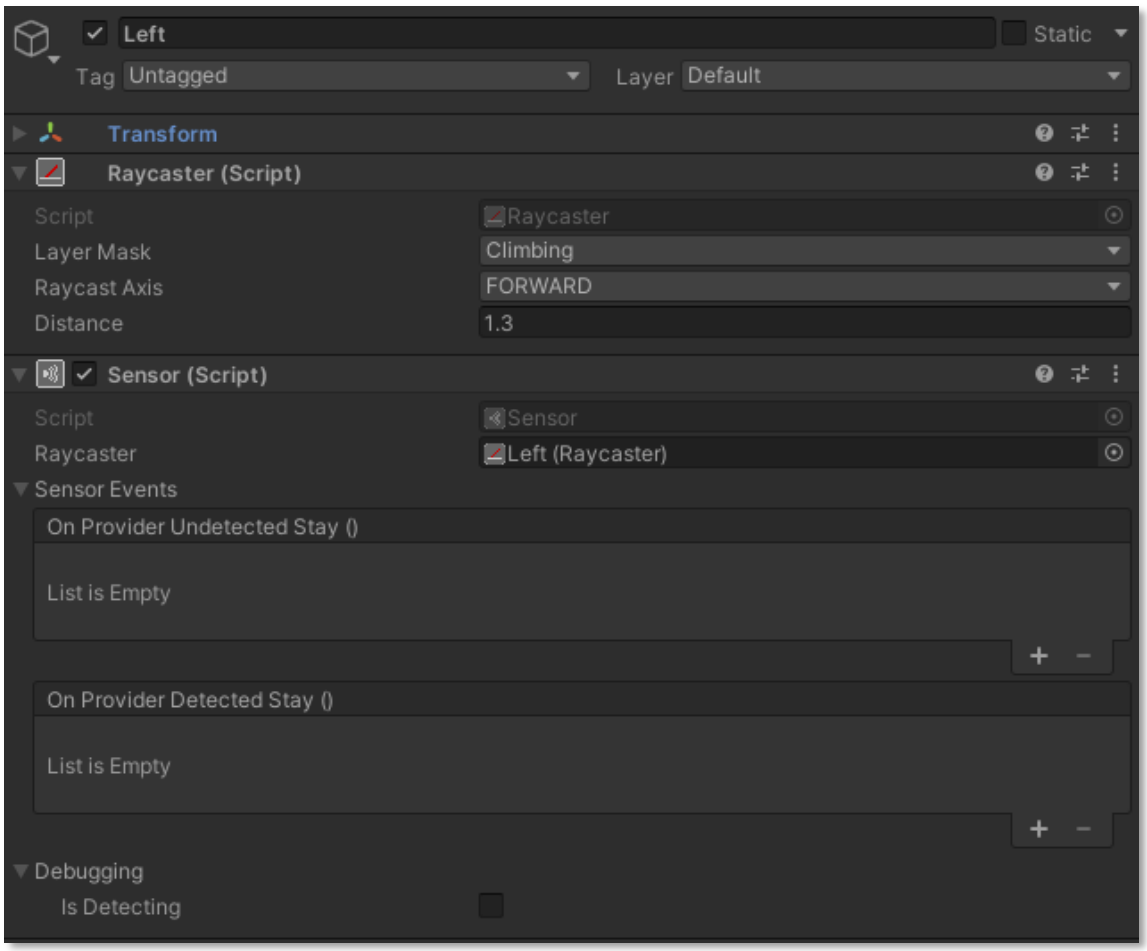
Function	Description	Type
velocityDirectionInfo.GetDirection(float distanceMultiplier, bool inverse)	Gets the current direction	Vector 3

	of the object's velocity. You can specify if you want to get the **inverse** direction (useful for pendulums) and also the **distance multiplier**.	
velocityDirectionInfo.GetRawDirection(bool inverse)	Get just the normalized of the calculated velocity delta. Useful for you to calculate the direction manually.	Vector 3

6.5. Sensor.cs

The Sensor component prevents your character from leaving the climbing area.


Press Right Click to start/stop climbing
A = Left
D = Right
W = Up
S = Down
Climbing Status: On Climbing
Current Triangle Index: 8
Current Triangle Position:
X: -1 Y: 0 Z: 0
Current Triangle Rotation:
X: 270 Y: 90 Z: 0


Left
Static
Tag Untagged
Layer Default
Transform
Raycaster (Script)
Script Raycaster
Layer Mask Climbing
Raycast Axis FORWARD
Distance 1.3
Sensor (Script)
Script Sensor
Raycaster Left (Raycaster)
Sensor Events
On Provider Undetected Stay ()
List is Empty
On Provider Detected Stay ()
List is Empty
Debugging
Is Detecting

```
if (Input.GetKey(KeyCode.A) && sLeft.IsDetecting)
{
    animator.SetBool("Left", true);
}
else
{
    animator.SetBool("Left", false);
}
```

Property	Description	Type
Sensor.IsDetecting	Checks if the Sensor is detecting a TriangleProvider.	Bool
Sensor.TriangleHit	Gets the TriangleHit with some useful properties. `public class TriangleHit { public RaycastHit raycastHit; public Vector3 normal = Vector3.zero, point; public TriangleProvider triangleProviderHit; public Transform hitTransform = null; public int triangleIndex = -999; public bool raycast = false; }`	TriangleHit