Procedural Spider Documentation

Philipp Schofield

Version: 1.1.1
March 30, 2025

Contents

1 License

If this Asset has been purchased through the Unity Asset Store, the corresponding EULA holds. If this Asset has been acquired through any other means than the Unity Asset Store, no license is granted.

2 Installation

Import the downloaded package into your Unity project as:

Assets/ProceduralSpider/....

3 Playing the Demo

Open one of the following scenes in:

ProceduralSpider/Scenes/...

- **SpiderPlayerDemo**: You should be able to play as the procedural spider and move around. There should also be another NPC spider walking around.
- **SpiderNPCDemo**: You should see a static camera scene with a lot of NPC spiders running around.

4 Getting Started

To add a spider to your scene, drag&drop one of the following prefabs into to your scene. They can be found at:

ProceduralSpider/Spider/Prefabs/...

- **Spider Controller (Player)**: A Third-Person Character Controller of the spider. Has a camera and reacts to input.
- **Spider Controller (NPC)**: A random-walking NPC spider. It uses perlin noise to move around.

Make sure to head to **Spider Component** [**5.1**] and set **Walkable Layer** to all the layers in your scene the spider should be able to move on. Repeat this for **IK Step Manager Component** [**5.5**] and set **Step Layer** to all layers the spider should be able to place its legs on.

5 Components

5.1 Spider Component

This component provides the spider movement behaviour and is also responsible for glueing the spider to the surfaces around it. This is accomplished by creating a fake gravitational force in the direction of the surface normal it is standing on. The surface normal is determined by spherical raycasting downwards, as well as forwards for wall-climbing. The spider does not move on its own. A controller must call the provided functions SetVelocity and Jump from code for the desired control. See **Spider Player Controller Component** [**5.2**] and **Spider NPC Controller Component** [**5.3**] for controllers that do just that.

- **Turn Speed**: The turn speed of the spider. Determines how fast the spider will turn.
- **Is Wall Walking**: If enabled, the spider can walk on walls and ceilings and will rotate to match the ground normal. If disabled, it can't climb them and will not rotate to match the ground.
- **Walkable Layer**: The layer on which the spider can walk. Will stick to colliders in this layer.
- **Gravity Multiplier**: The spider uses fake gravity to either stick it to surfaces or make it fall. This is a multiplier on it.
- **Ground Adjust Rotation Speed**: Determines how fast the spider will rotate to match the ground surface it sticks to.
- **Wall Adjust Rotation Speed**: Determines how fast the spider will rotate to match the ground surface it climbs onto.
- **Forward Ray Length**: How long the forwards sphere ray is. Higher value will make spider notice walls to climb earlier.
- **Down Ray Length**: How long the downwards sphere ray is. Higher value will make it notice ground underneath more aggressively.
- **Show Debug**: Enable this to draw debug drawings in the viewport.

5.2 Spider Player Controller Component

This component will apply movement to **Spider Component** [**5.1**]. It uses player input and will request movement relative to the **Camera Third Person Component** [**5.10.1**] that has to exist on a child object.

- **Walk Speed**: The speed at which the spider will walk.
- **Run Speed**: The speed at which the spider will run.
- **Jump Force**: The force at which the spider will jump.
- **Key Code Run**: The key code associated to making the spider run.
- **Key Code Jump**: The key code associated to making the spider jump.

5.3 Spider NPC Controller Component

This component will apply movement to **Spider Component** [**5.1**]. It can either randomly move around, or move towards a given target (or a blend between the two). It can frequently stop movement erratically to mimic the stop and go nature of spiders. It uses perlin noise for the randomness of the movement.

- **Speed**: The speed at which we will move.
- **Random Movement Frequency**: Describes how often the movement direction will change. A higher value corresponds to changing direction more often. A lower value corresponds to committing to a direction for longer.
- **Follow Weight**: Blend value between the random movement direction and the follow direction. A Value of 0 corresponds to full random movement. A value of 1 corresponds to full target follow.
- **Follow Target**: If this target is set and a non-zero follow weight is set, this controller will move towards this transform. Leave unassigned if you want to set target position manually by calling SetTargetPosition.
- **Stop And Go Weight**: Weight that describes how much we will move and how much we will stop. A value of 0 corresponds to always moving. A value of 1 corresponds to always stopping. A value of 0.8 corresponds to 80% moving and 20% stopping.

- **Stop And Go Frequency**: Describes how frequent the movement will stop. A higher value corresponds to switching between stop and go more often. A lower value corresponds to committing to either stop or go for longer.
- **Show Debug**: If enabled, will draw debug drawings to viewport.

5.4 Procedural Root Motion Component

Requires an **Animator Component** with a set **Avatar** somewhere in children.

This component adds procedural root motion to the spider. The root bone will move and rotate in accordance to its legs (**IK Chain Component** [**5.9**]), specifically the height of each leg. It can also add a breathing movement. The root bone is given by the Avatar of the Animator Component.

- **Root Height**: The height we will set the root bone to be at runtime.
- **Use Root Movement**: If enabled, will translate the root bone according to legs.
- **Root Movement Speed**: How fast the root bone will move.
- **Root Movement Vertical Weight**: The higher the vertical weight, the more the root bone will move upwards.
- **Root Movement Horizontal Weight**: The higher the horizontal weight, the more the root bone will move sideways.
- **Use Root Rotation**: If enabled, will rotate the root bone according to legs.
- **Root Rotation Speed**: How fast the root bone will rotate.
- **Root Rotation Weight**: The higher the value, the more the root bone will rotate according to legs.
- **Use Breathing**: If enabled, will periodically move root bone up and down like breathing.
- **Breathe Period**: The time in seconds to finish one breath. Higher values will make it move slower.

- **Breathe Magnitude**: The magnitude of how much the root bone should move up/down.
- **Show Debug**: Enable this to draw debug drawings in the viewport.

5.5 IK Step Manager Component

This component manages stepping behaviour of the spider for all legs. It will find all **IK Chain Component** [**5.9**] defined in children, which are further referred to as legs. It will set the IK targets for each leg to mimic stepping. A leg desires to step if the IK solver can not solve the leg sufficiently anymore. This component will orchestrate each legs desire to step and make sure asynchronicity to other legs is adhered to. A new step target is found using a system of ray casts and the current velocity. The leg will move in an arc to reach the new target.

- **Step Layer**: The Layer in which we raycast to find a suitable target (Make sure to at least include the same layers as **Spider Component** [**5.1**]).
- **Step Height**: The height each step should take.
- **Step Curve**: The Animation Curve each step should follow. An inverse parabola is recommended.
- **Step Duration Multiplier**: A higher value will lead to slower steps. A smaller value to faster steps. The step duration is dynamic and relative to current velocity.
- **Step Duration (Max)**: The maximum step duration. This makes sure we never step slower than this value.
- **Leg Properties**: For each leg (**IK Chain Component** [**5.9**]) found in children, one can edit some properties:
  - **Auto Setup Asynchronous Legs**: A useful button to automatically setup asynchronous legs. Will try to add only the leg to the left/right and the leg in front of the current leg as asynchronous legs.
  - **Async Chains**: The legs to which we should step asynchronously. If one of them is in the process of stepping, this leg is blocked from stepping. It is recommended to only add the leg to the left/right and the leg in front.

  - **Anchor Reach**: A higher value will set targets more outwards. A lower value will set targets more inwards.
  - **Anchor Stride**: A higher value will set targets more to the right. A lower value will set targets more to the left.
  - **Anchor Multiplier**: A higher value will lead to larger steps. A value of 1 will make every step land exactly on the anchor position.

- **Draw Debug:** Enable this to draw debug drawings in the viewport.

5.6 Spider Mode Switch Component

A simple component that will switch between **Spider Player Controller Component** [**5.2**] and **Spider NPC Controller Component** [**5.3**] with a specified key code. It will always start with the former enabled.

- **Key Code Switch**: The key code that will trigger the switch.
- **Player Camera**: The camera that the **Spider Player Controller Component** [**5.2**] component uses.
- **NPC Camera**: The camera that the **Spider NPC Controller Component** [**5.3**] component uses.

5.7 Visibility Culling Component

A component that will disable given mono behaviours if it determines a mesh to be not visible. It will use the first **Renderer Component** found in children and check for its visibility. Spider is setup to disable components **Procedural Root Motion Component** [**5.4**] and **IK Step Manager Component** [**5.5**] if culled. This will disable all procedural animation and IK solving to optimize performance. Additionally one can set a cull distance at which the object is always regarded as culled.

- **Use Camera Distance Culling**: If enabled, will regard the object as invisible if its a certain distance away from the current camera.
- **Cull Distance**: The distance from the main camera at which we regard the object as culled if distance culling is enabled.
- **Mono Behaviours To Disable:** The mono behaviours that should be disabled when this component culls.

5.8 Joint Hinge Component

A component that represents a hinge joint with specified rotation axis and limits. It exposes a Rotate function, that can be called from code to apply the rotation given the setup constraints.

- **Rotation Axis**: The axis around which this hinge joint rotates.
- **Angle Constraints**: The lower and upper bound for our hinge joint.

5.9 IK Chain Component

A component that represents a chain of **Joint Hinge Component** [**5.8**] with an end effector. Given a target transform, the chain will perform IK (inverse kinematics) to reach it on late update stage. The IK Solver implemented is a CCD (Cyclic Coordinate Descent) algorithm. One can manually supply the target transform, or leave it empty which allows **IK Step Manager Component** [**5.5**] to set it.

- **Joints**: All **Joint Hinge Component** [**5.8**] found in children (Not editable and found automatically).
- **Weight**: The weight of this joint. The IK Solver will use this value. Lower values will make this joint tend to rotate less.
- **End Effector**: The end effector of this chain. Is the child transform of last joint.
- **Auto Create Joints**: A helpful button to automatically create a **Joint Hinge Component** [**5.8**] on all children except the last which is used as the end effector.
- **Has Foot**: If enabled, the last joint will rotate to the specified foot angle. This will make it act like a foot.
- **Foot Angle**: The foot angle in degrees used if foot is enabled. 0 corresponds to foot flat on the ground.
- **IK Target**: If set, will use it as target for this IK Chain. Useful for debugging purposes. Leave this empty if **IK Step Manager Component** [**5.5**] should manage IK targets.

- **Physics Based End Effector Velocity**: If set, will calculate end effectors velocity based on physics fixed update. This will make velocity frame rate independent. If not set, a leg may step slower/faster for lower/higher frame rates as the step time of the leg scales with this velocity.

5.10 Camera Components

There are two implemented Camera Components:

- **Camera Third Person Component** [**5.10.1**]
- **Camera Spectating Component** [**5.10.2**]

They both require a Camera component and will move the camera to smoothly follow the parent transform, referred to as target. They react to mouse input and will turn the camera horizontally/vertically. They support minimum and maximum angles. They support obstruction hiding and clip zoom.

- **Player Camera**: The camera of this controller. The input will be relative to this camera.
- **Translation Speed**: The speed at which the camera will move to follow the target.
- **Rotation Speed**: The speed at which the camera will rotate to follow the target.
- **X Sensitivity**: The sensitivity of horizontal mouse movement.
- **Y Sensitivity**: The sensitivity of vertical mouse movement.
- **Position Interpolation Type**: Determines how the translation should be interpolated smoothly.
- **Cam Upper Angle Margin**: The lower bound angle the camera should be constrained to.
- **Cam Lower Angle Margin**: The upper bound angle the camera should be constrained to.
- **Enable Clip Zoom**: Enables clip zoom. This will move the camera closer towards the target whenever an object obstructs sight.

- **Clip Zoom Layer**: The layers used to determines obstructing objects for clip zoom.
- **Clip Zoom Padding Factor**: Adds padding from camera position to obstructing object. Value of 0 means that camera will place itself directly at obstructing object.
- **Clip Zoom Min Distance Factor**: Determines the minimum distance the camera should keep from the target. Value of 0 means the camera can move into the target, value of 1 means the camera can not zoom at all.
- **Enable Obstruction Hiding**: Enables obstruction hiding. Will make all obstructing objects invisible.
- **Obstruction Hiding Layer**: The layers used to determines obstructing objects for obstruction hiding.
- **Ray Radius Obstruction Hiding**: The radius of the sphere cast used to determines obstructing objects.
- **Show Debug**: Enable this to draw debug drawings in the viewport.

The difference between the two components is as follows:

5.10.1 Camera Third Person Component

Stays parented to the parent transform and will follow the parents rotation. It will rotate around the parents Y-Axis. How much it will follow the rotation can be customized.

- **Roll Ignore**: If set to 0, camera will not follow any X-Axis rotation of the parent object. If set to 1, camera will fully follow it.
- **Yaw Ignore**: If enabled, the camera will not follow any Y-Axis rotation of the parent object.

5.10.2 Camera Spectating Component

Will not follow the parents rotation and will always rotate around the global Y axis.

6 Additional

6.1 Scaling

Rescaling of spider is supported. Simply change the controller transform scale. The components will adjust to the scale through the actual size of the **Capsule Collider** of the spider. Be aware that changing radius/length of this collider will therefore have scaling effects.

- For **Spider Component** [**5.1**] it scales the velocity and grounding raycasts.
- For **Procedural Root Motion Component** [**5.4**] it scales root bone translation.
- For **IK Step Manager Component** [**5.5**] it scales step height, step time and all of the raycasts.
- For **IK Chain Component** [**5.9**] it scales the IK solver (Tolerance etc.).
- In general it scales debug drawings.

6.2 Using a custom Model/Rig

It is possible to use another model/rig than the provided one, but this is not straightforward to set up and it is not guaranteed to work for your model. There are a couple of requirements your model/rig has to fulfill:

- Must have enough joints per leg (The provided one has five per leg, plus end effector).
- Each legs last joint must represent the end effector (Can also be added in manually by adding a child game object).
- The first joint of each leg must be designed to rotate around the spiders root bone up axis.
- The other joints of each leg must be designed to rotate in such a way that they curl (e.g. local Z-Axis).
- The leg/body length ratio needs to be similar to the provided one.
- Rig should be imported as follows:

- Animation Type: 'Generic'
- Avatar Definition: 'Create From This Model'
- Root Node: The root bone your spider model. This bone will procedurally animate.

To set it up, do the following:

1. Import your model as required above.
2. Drag one of the prefabs into scene, e.g. 'Spider Controller (Player)'.
3. Right-click on it, hover over 'Prefab' and click on 'Unpack Completely'.
4. Delete 'Spider_Model' game object (This is the provided model).
5. Replace it with your model by dragging it onto 'Spider Controller (Player)'.
6. Add an Animator Component to your dragged in model and setup the Avatar to the automatically created one from import.
7. For each leg in your model, do:
   (a) Find the start joint of the leg (the joint designed to rotate around root bone up-axis).
   (b) Add **IK Chain Component** [**5.9**] to it.
   (c) Click on 'Auto Create Joints'. This will create **Joint Hinge Component** [**5.8**] on every child joint (including this one). If this fails, do it manually. Make sure the first joint rotates around 'Root Y' and the others around the proper local axis (e.g. 'Local Z'). Adjust constraints if applicable.
   (d) Make sure **IK Chain Component** [**5.9**] has found all **Joint Hinge Component** [**5.8**] and the End Effector is setup correctly.
8. Head back to 'Spider Controller (Player)', and set up:
   (a) For **IK Step Manager Component** [**5.5**] make sure it has found all your added **IK Chain Component** [**5.9**]. Setup 'Asynchronous Legs' for each leg. Either click on 'Auto Setup Asynchronous Legs', or do it manually. A leg should be asynchronous to the leg in front and the leg opposite of it.

(b) For the **Capsule Collider**, adjust radius and height. Do not change direction or collider type, only Z-Axis aligned Capsule Collider is supported! Be aware, that its size has side effects on other components as they use this collider for scaling reference, see **Scaling** [**6.1**].

(c) Optionally, adjust relative position of your model so it fits into the collider nicely.

6.3 Using Animations Clips

This asset does not include any animation clips. All animation is fully procedural. It is not possible to blend animation clips on top of this procedural animation. It is however possible to disable **IK Step Manager Component** [**5.5**] and **Procedural Root Motion Component** [**5.4**], play the animation clip and reenable the components when the animation clip has finished playing. With this approach adding e.g. a jump animation should be straightforward.

6.4 Creating a custom Controller

To create a custom controller for the spider, simply call the SetVelocity and Jump functions on **Spider Component** [**5.1**] as you wish. As an example refer to **Spider Player Controller Component** [**5.2**] and **Spider NPC Controller Component** [**5.3**].