# Dynamic Spider Web

Copyright (c) Pampel Games e.K. All Rights Reserved.

**Support** **Contact** **More Tools**

## UNLOCKED DISCOUNTS

| | |
|---|---|
|  |  |
| **Gore Simulator** | **Blood Factory** |

# SUMMARY


Dynamic Spider Web
SUMMARY
INSTALLATION
    Requirements
    Installation
QUICK START
    Create Webs
    Interaction
    Prefabs
TROUBLESHOOTING
    Cut Edges
API
    SpiderWeb.cs
    SpiderWebCreator.cs

# INSTALLATION

## Requirements

- Minimum Unity Version 2022 LTS +

## Installation

If you have other tools from Pampel Games installed in your project, it is recommended that you update them to the latest version before importing the asset.

After importing, the 'PampelGames' folder can be moved to a different location within the 'Assets/' folder if desired.

# QUICK START

## Create Webs

*There are many ready-made examples in the SpiderWebDemo01_Examples demo scene, which you can modify and recreate to your liking.*

*To create a new web from scratch, please follow this quick guide.*

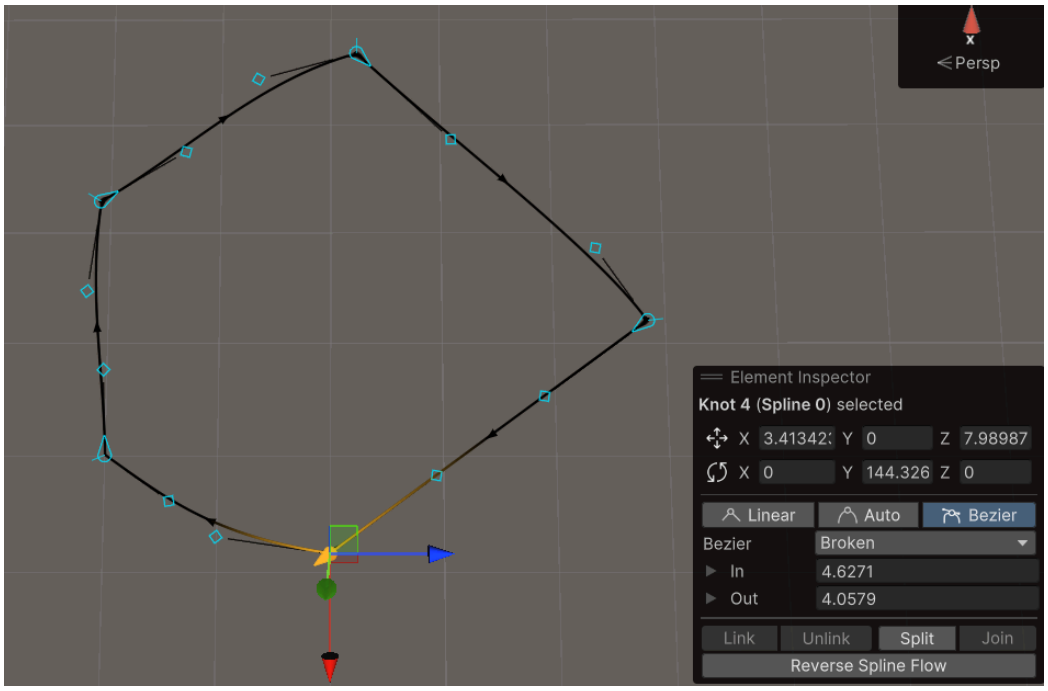
First, create and open a new empty scene.

Open the Unity Spline tool by right-clicking in the Hierarchy and selecting Spline > Draw Spline Tool.

Draw a spline in your scene. This spline will be used for the outer strings of the web.

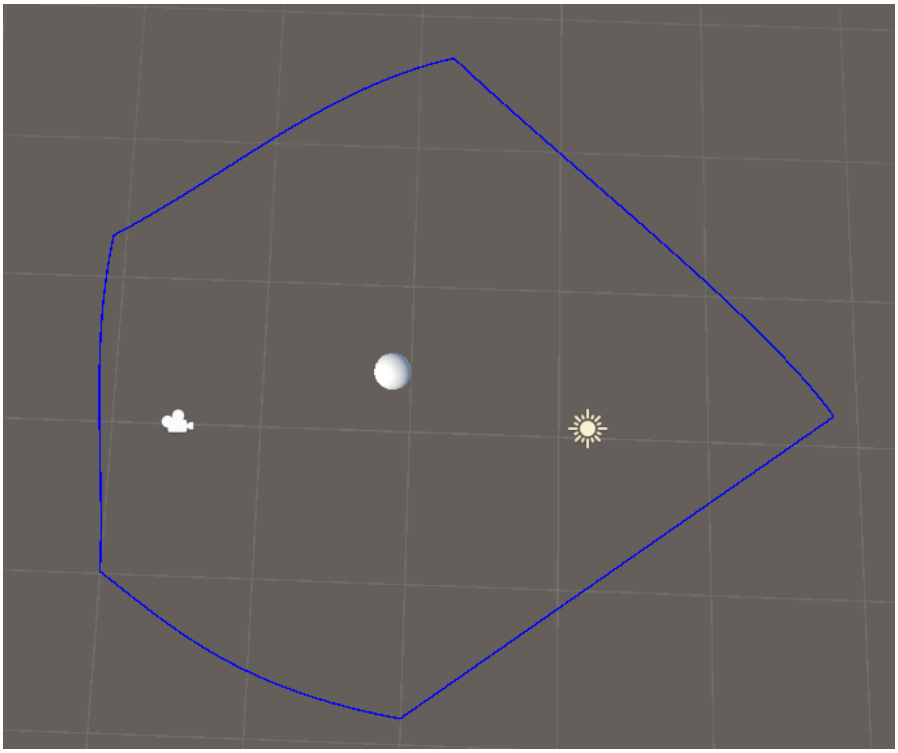
*For more information on using the Spline package, refer to the Unity documentation and search for "Getting started with Splines."*

For best results, the spline should be closed, and the knots set to “Bezier > Broken.” This provides greater flexibility and ensures the edges remain sharp.



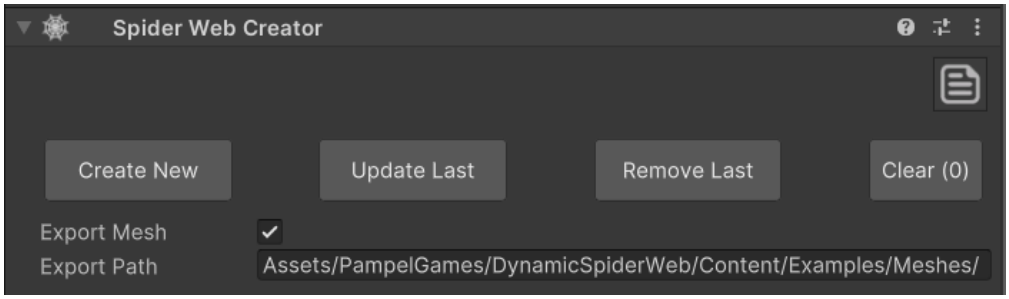

Example of a web spline for the outer strings

Create a sphere and place it roughly at the center of the spline. This transform will serve as the center of the circular web.



Sphere in the center of the spline (Unity scene gizmos enabled)

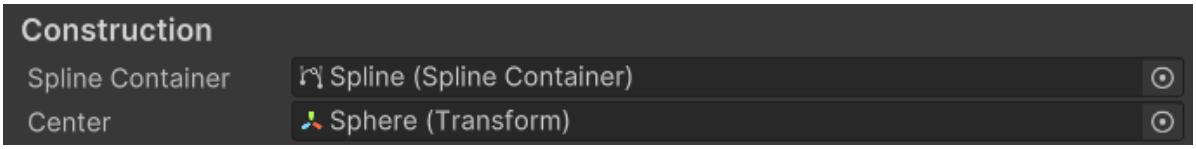
Create an empty GameObject in the Hierarchy and name it "WebCreator."
Then, add the SpiderWebCreator component to it.




It's a good idea to change the export path to a new, empty folder, as it may be difficult to locate later among the demo meshes. Alternatively, you can disable mesh export for this example.

In general, exporting meshes is necessary if you plan to create prefabs from the webs or move them to other scenes. To export meshes after creation, you can use the Unity package “FBX Exporter”.

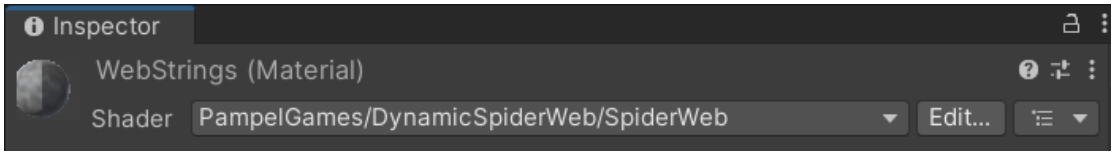
In the Construction section of the SpiderWebCreator component, assign the Spline Container and the Sphere as the center.




The first spline in the container defines the outer strings, which the circular web attaches to. Additional splines in the container are optional and can be used to create extra strings for added detail.

Before we can create a new web, we need to reference a material for the strings and at least one material for the inner circle.

Create a new material in your project folder and name it “WebStrings.”
Set the shader to PampelGames > DynamicSpiderWeb > SpiderWeb.
The default textures should already be assigned automatically.

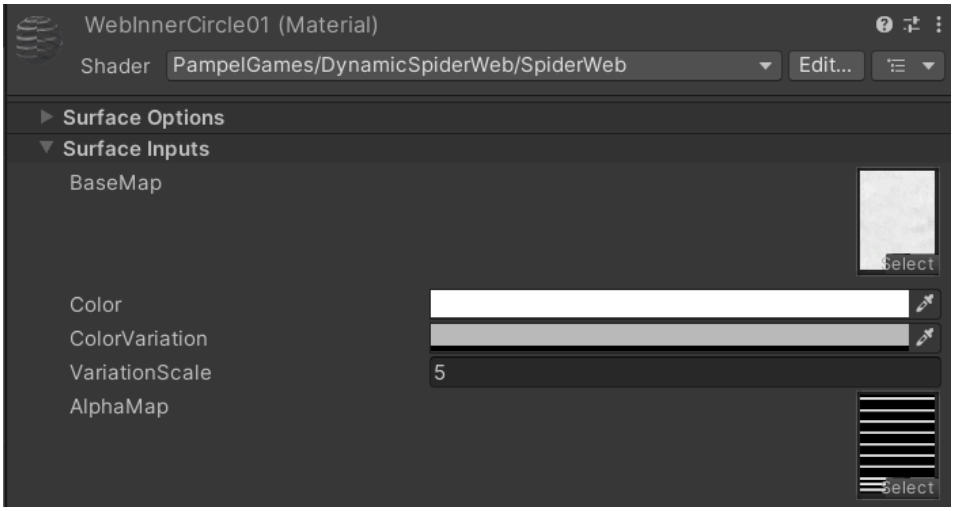



Reference this material as the “Strings” material in the SpiderWebCreator component.

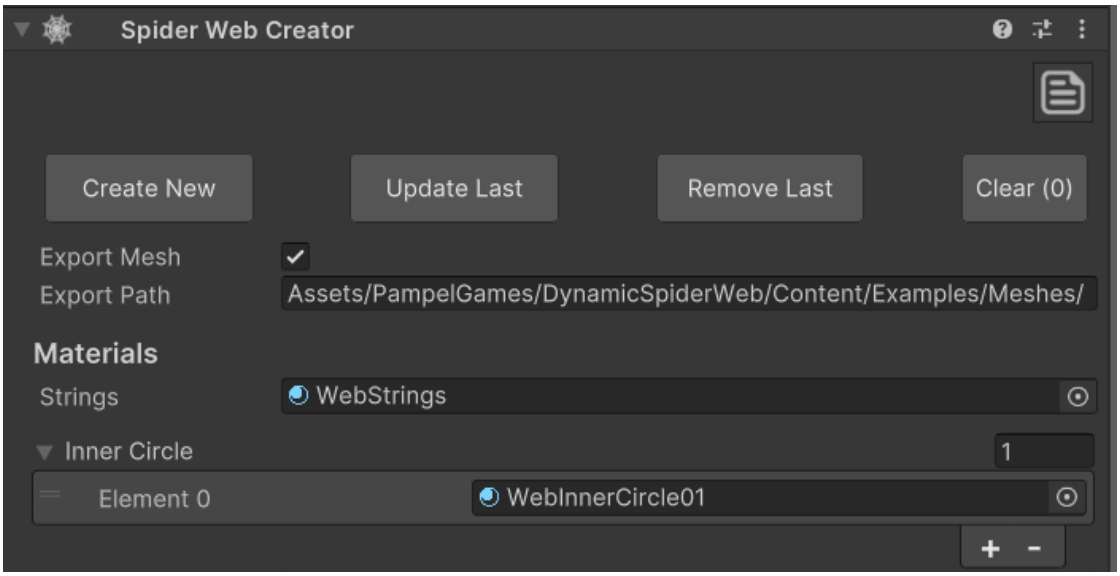
Copy the material in the project folder and rename the copy to “WebInnerCircle01.”

For this material, you'll need an alpha texture to define a specific shape. Various textures are included in the asset, or you can create your own.

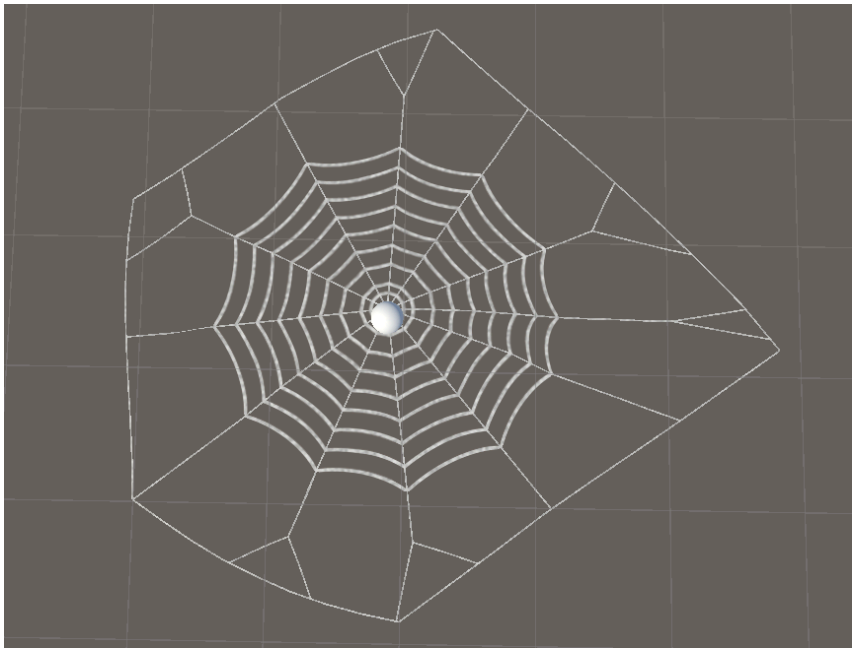
Search for the “SpiderWebAlphas01_1” texture and assign it to the Alpha Map slot of the WebInnerCircle01 material.




Add this material to the Inner Strings material list in the SpiderWebCreator component.




All right, now we have the bare minimum of what we need to create a new spider web. Hit the “Create New” button and find the new web in your scene.



Example spider web with the radius increased to 5

After changing settings or materials, simply click the “Update Last” button - this will also remove and recreate the project mesh.

From this point, you should be able to modify and expand the example on your own. Most fields include tooltips explaining their function (if not self-explanatory), or you can experiment directly in the editor.

Note: All settings in the generated webs can be modified later, except for the Construction settings at the top. To make changes such as switching materials in the Skinned Mesh Renderer or adjusting interaction settings in the SpiderWeb component, locate the generated webs in the hierarchy.

Also remember that the materials offer a variety of settings to help you achieve the exact look you're aiming for.

Enjoy 🙂

## Interaction

To explore ways of interacting with the spider webs, please refer to the included demo scenes (SpiderWebDemo02 – SpiderWebDemo04).

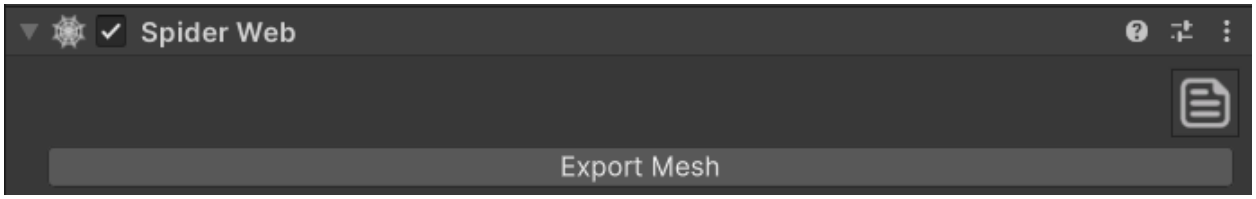
These scenes provide straightforward examples, and within their hierarchies, you'll find READ ME components and example scripts demonstrating how to execute the logic.

Also refer to the API provided below.

## Prefabs

To create a prefab from a spider web, the mesh must first be serialized to the project folder.

Use the Export Mesh button at the top of the SpiderWeb component to save the mesh as an asset.

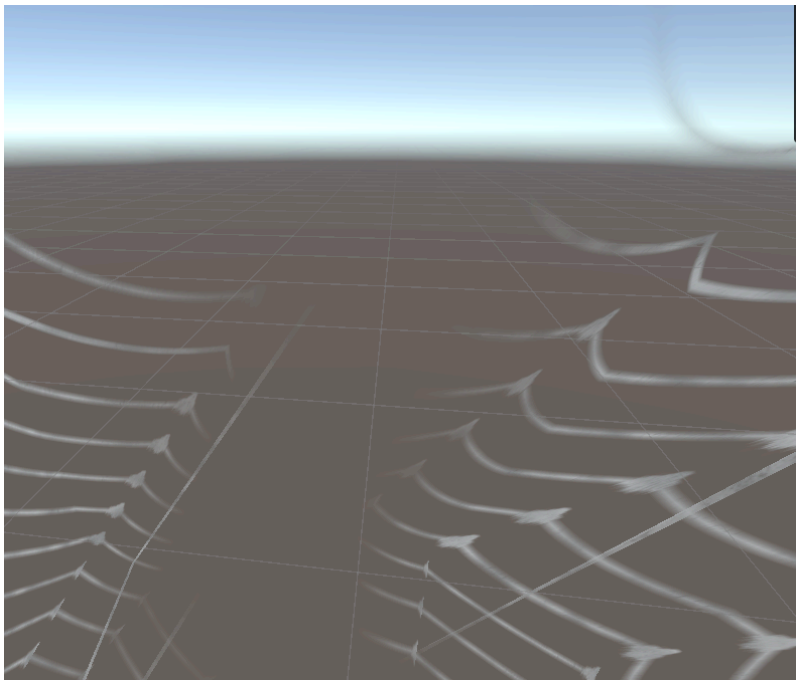



After exporting the mesh, you can create a prefab from it and instantiate the prefab wherever needed.

# TROUBLESHOOTING

## Cut Edges

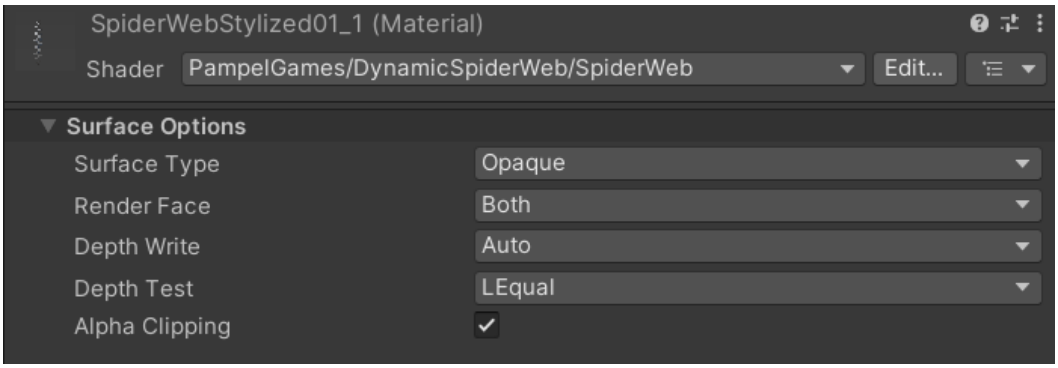
If you cut or rip out parts of the web, the edges are smoothly faded out rather than being cleanly cut:

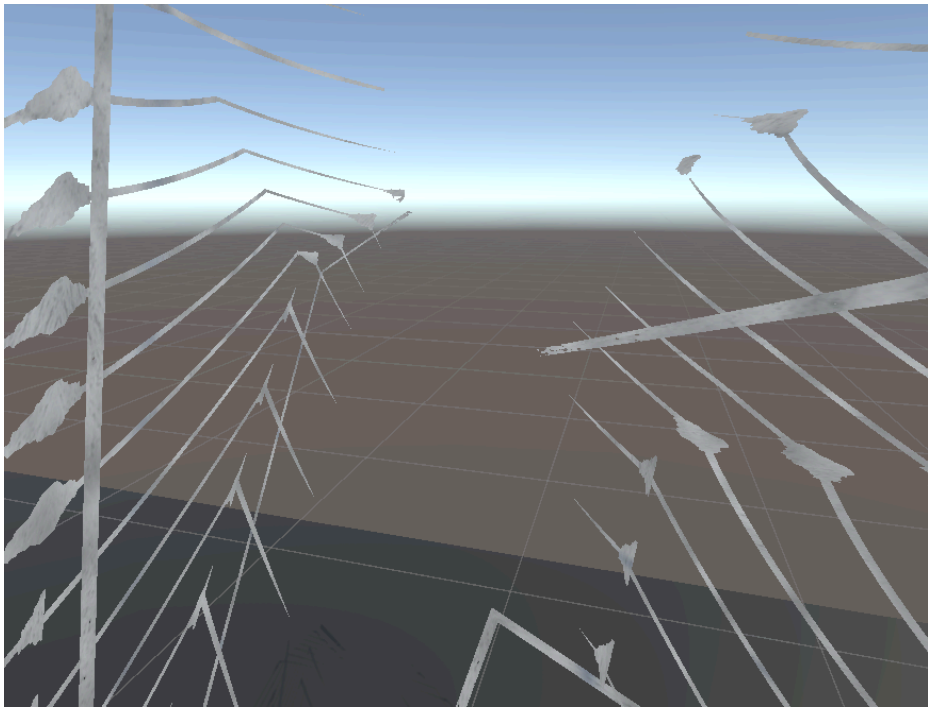


Example of a faded cut in the SpiderWebDemo02_Cut

This smooth fade effect occurs because the material's surface type is set to Transparent by default.

To achieve a clean cut instead, simply set the surface type to Opaque and enable Alpha Clipping.



Example of a clean cut with an opaque material

However, opaque materials cannot be smoothly faded out. This also affects effects like rain drops, where the base of the web can't be faded either.

Unfortunately, this means you have to choose between a smooth fade (with transparent materials) or a clean cut (with opaque materials and alpha clipping).

# API

## SpiderWeb.cs

```
/// <summary>
///     Determines whether the spider web is disconnected, which
happens after cut/rip operations.
/// </summary>
public bool IsDisconnected()


/// <summary>
///     Cuts a section of the spider web.
/// </summary>
/// <param name="origin">The world position on the web where the cut
begins.</param>
/// <param name="normal">The direction perpendicular to the cutting
plane.</param>
/// <param name="force">A force applied to deform the cut vertices
over time.</param>
/// <param name="forceDuration">The duration over which the force is
applied until full deformation.</param>
/// <param name="cutSpiderWeb">The resulting new spider web
representing one side of the cut. The original web represents the
other side.</param>
/// <returns>Returns false if the cut was prohibited due to
interaction delay not being reached or the origin being out of
range.</returns>
public bool Cut(Vector3 origin, Vector3 normal, Vector3 force, float
forceDuration, out SpiderWeb cutSpiderWeb)
```

```
/// <summary>
///     Rips out a section of the spider web based on the specified
origin and radius.
/// </summary>
/// <param name="origin">The origin point from which the rip out
operation begins.</param>
/// <param name="radius">The radius within which the web section will
be detached.</param>
/// <param name="cutSpiderWeb">The resulting new spider web
representing the ripped part. The original web represents the
remaining part.</param>
/// <returns>Returns false if the rip was prohibited due to
interaction delay not being reached or the origin being out of
range.</returns>
public bool RipOut(Vector3 origin, float radius, out SpiderWeb
cutSpiderWeb)


/// <summary>
///     Initiates the process of sticking the spider web to the
specified transform, applying a spherical area.
/// </summary>
/// <param name="stickTransform">The transform to which the spider web
will stick.</param>
/// <param name="radius">The radius defining the area of the sticky
effect.</param>
/// <param name="ripOutRadius">The radius within which the web can be
"ripped out" or detached.</param>
/// <returns>Returns true if stick to started successfully.</returns>
public bool StartStickTo(Transform stickTransform, float radius,
float ripOutRadius)
```

```
/// <summary>
///     Begins the process of attaching the spider web to a specified
transform with a rectangular area defined by height and width.
/// </summary>
/// <param name="stickTransform">The transform to which the spider web
will stick.</param>
/// <param name="height">The height of the rectangular area for the
sticky effect.</param>
/// <param name="width">The width of the rectangular area for the
sticky effect.</param>
/// <param name="ripOutRadius">The radius within which the web can be
detached or "ripped out."</param>
/// <returns>Returns true if stick to started successfully.</returns>
public bool StartStickTo(Transform stickTransform, float height,
float width, float ripOutRadius)


/// <summary>
///     Stops the spider web from sticking to the current target and
deactivates the stick-to behavior.
/// </summary>
public void StopStickTo()


/// <summary>
///     Burns the spider web starting from the specified origin using
the default particles.
///     The burning process progressively destroys the web as it
spreads.
/// </summary>
/// <param name="origin">The starting point of the burning process in
world coordinates, converted to the nearest vertex.</param>
public void Burn(Vector3 origin)
```

```
/// <summary>
///     Burns the spider web starting from the specified origin.
///     The burning process progressively destroys the web as it
spreads.
/// </summary>
/// <param name="particles">The particle system used to simulate the
burning effect.</param>
/// <param name="origin">The starting point of the burning process in
world coordinates, converted to the nearest vertex.</param>
public void Burn(ParticleSystem particles, Vector3 origin)


/// <summary>
///     Gradually fades out the spider web and destroys it, using the
fade out timing settings.
/// </summary>
public void FadeOut()


/// <summary>
///     Gradually fades out the spider web and destroys it.
/// </summary>
/// <param name="delay">The time in seconds to wait before starting
the fade-out process.</param>
/// <param name="duration">The time in seconds over which the fade-out
effect occurs.</param>
public void FadeOut(float delay, float duration)


/// <summary>
///     Editor Only. Exports the web mesh into the project folder.
/// </summary>
public void ExportMesh(string exportPath)


/// <summary>
///     Editor Only. Removes the web mesh from the project folder.
/// </summary>
public void DestroyMesh()
```

## SpiderWebCreator.cs

```
/// <summary>
///     Creates a new spider web object with the provided
configurations and materials.
/// </summary>
public SpiderWeb CreateSpiderWeb()
{
   return CreateSpiderWebInternal();
}
```