DREAMTECK
SPLINES

USER MANUAL & API REFERENCE

Contents

New to Dreamteck Splines and already have a question on your mind? **Jump to the FAQ** to see if it hasn't been asked before!

1. Getting Started

Dreamteck Splines is a Spline system and extension for Unity which comes with a collection of tools and components for mesh generation, particle control, object spawning and much more. For a full list of features, refer to Spline Users and Editor Tools. The tool was initially created to aid our company (Dreamteck Ltd.) in the process of level editing and creating game mechanics and it has been growing ever since it was first created in 2013. Over the years it has been used for many different purposes in many different projects (action games, TCG games, racing games, endless runners, children's games and more).

Our support team is always at your disposal to help resolve issues at https://dreamteck.io/support/contact.php

1.1. Key Features

- ✓ Rapid spline creation and editing via custom editor in Unity.
- ✓ Four types of spline: Hermite, Bezier, B-Spline and Linear
- ✓ Following splines with uniform speed
- ✓ Extruding geometry along splines
- ✓ Creating splines with code
- ✓ Procedural primitives and saving presets for later use
- ✓ Junctions
- ✓ Morph states
- ✓ Multithreading
- ✓ Open source

1.2. Install

After the Dreamteck Splines package is downloaded from the Asset Store and imported into the project, a welcome window will pop up with information about the current version and additional actions.

- **What's new?** – Opens a changelog from the previous version up until this one. He changelog is reset on every major version.
- **Get Started** – Links to video tutorials and the user manual
- **Community & Support** – Links to contact support and get in touch with other Dreamteck Splines users.
- **Rate** – We would really appreciate it if you rate our efforts on the Asset Store – this is a link to the product's page where a review can be written.
- **Playmaker Actions** – Imports a package with Playmaker actions for Dreamteck Splines

To get the example scenes, go to **Get Started** and click the **Examples** tab. The examples package will be imported in the project under Dreamteck/Splines/Examples

1.3. Updating Dreamteck Splines

Always make a backup of the project before updating to a newer version of any plugin.

If a previous version of Dreamteck Splines is installed, it is recommended that the previous version is removed before updating. Before deleting the current Dreamteck Splines version, make sure that the Dreamteck/Splines/Presets folder doesn't contain any custom presets and if it does, make a backup of them.

1.4. Preferences configuration

To access the Preferences screen go to Edit/Preferences/Dreamteck/Splines. This will open the following screen:
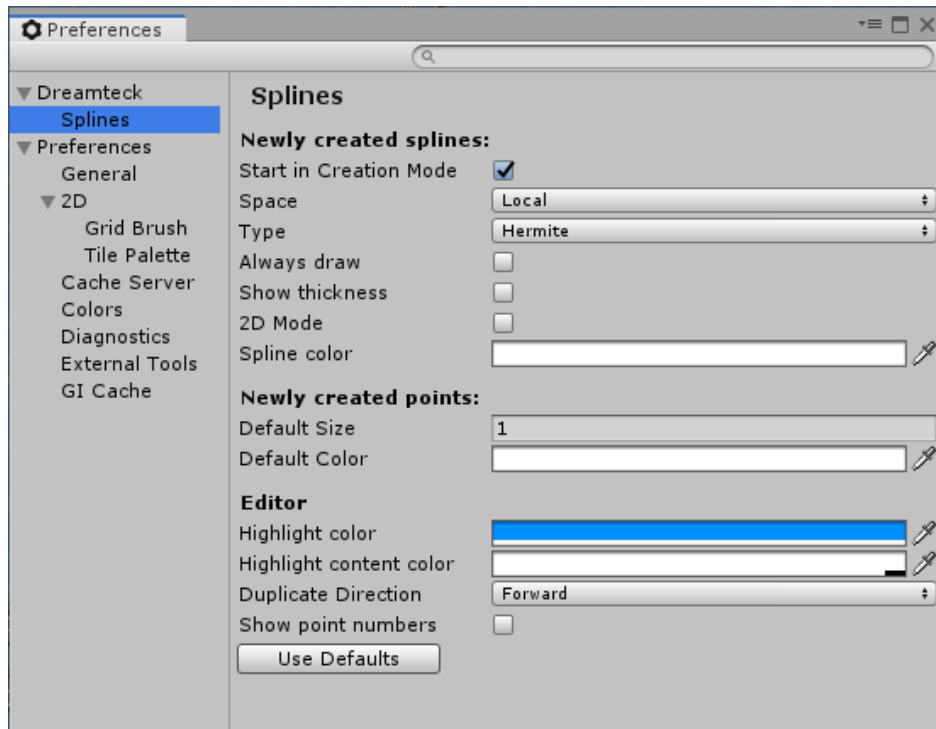
The options under "Newly created splines" define the default settings for each newly created spline. For example, if the game scene is full of light colors, then the default spline color could be set to black and that way each newly created spline will be black upon creation without the need of setting the color every time.

The options under "Newly created points" define the size and color of newly created points.

The options under the "Editor" section let the user customize the UI and scene view of the editor.

1.5. Creating a new Spline

To create a new spline object in the scene, go to **GameObject/3D Object/Spline Computer**. This will create and select a new Game Object called "Spline" with a **Spline Computer** component. If the "Start in Creation Mode" setting is toggled in the preferences, the spline will be automatically put in point creation mode and a green grid will follow the mouse inside the scene. Clicking inside the scene view will create new spline points. To exit point creation mode, go to the Spline Computer's inspector and click the plus button inside the Edit tab.

1.6.   Adding control points

A spline needs at least two control points in order to visualize. To add control points, click the Plus button inside the Edit tab in the Spline Computer's inspector. This will enter point creation mode. While in point creation mode, clicking inside the scene creates spline points.
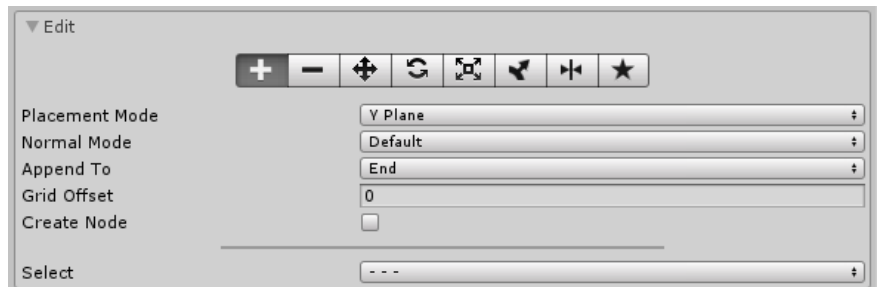
The point creation mode has a couple of properties which define how points are created and added to the spline.

1.6.1. Placement Mode

Defines where the point will be created.

- Y Plane – Creates points on the World-Y plane
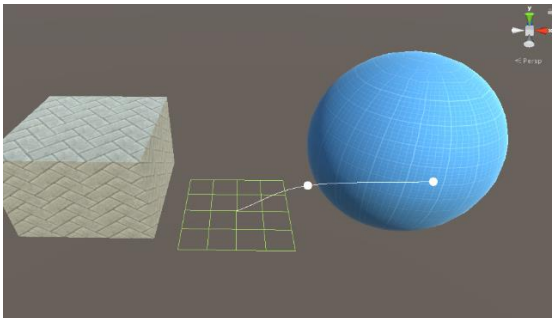
- X Plane – Creates points on the World-X plane
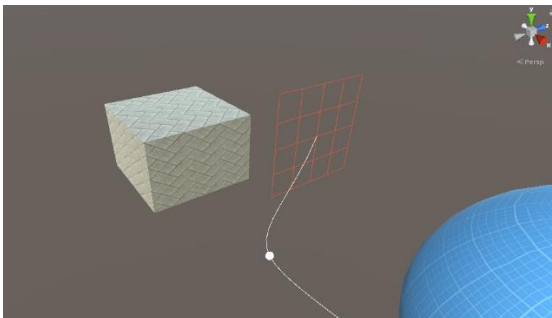
- Z Plane – Creates points on the World-Z plane
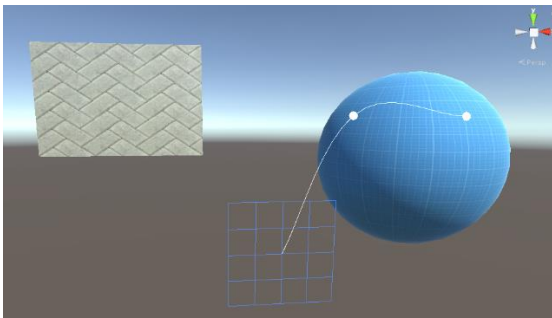
- Camera Plane - Y Plane – Creates points on the editor camera's plane.
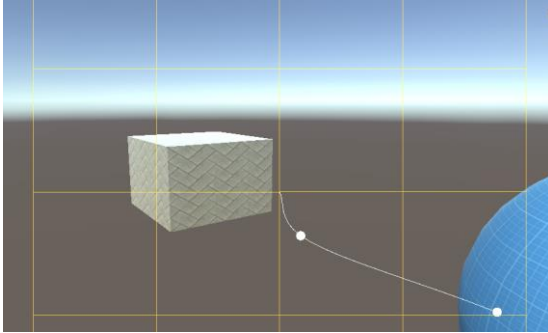
- Surface – Creates points on any surface which has a 3D Collider (uses Raycasting)
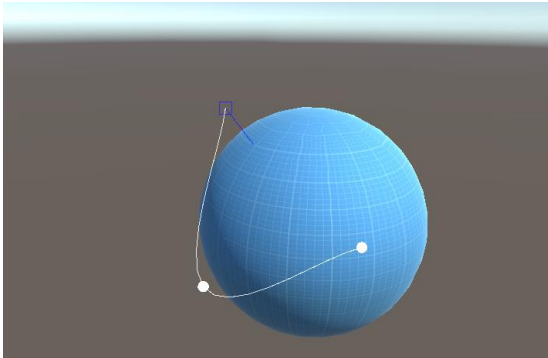
- Insert – Inserts a point in between two other spline points

All placement modes have an accompanying offset property. For the world plane methods, it is called the "Grid Offset" and for the camera plane it's called "Far Plane". In surface mode it is called "Surface offset". This property adds offset along the given creation method's normal.

1.6.2. Normal Mode

Defines where the new points' normals will point at.

- Default – Will use the default normal for the selected placement method. For example, Y Plane will make the normal point upwards.
- Look at Camera – The normal will look towards the editor camera's position
- Align with Camera – The normal will align with the editor camera's direction
- Calculate – The normal will automatically be calculated based on the location of the previous control points
- Left – World Negative X
- Right – World Positive X
- Up – World Positive Y
- Down – World Negative Y
- Forward – World Positive Z
- Back – World Negative Z

1.6.3. Append To

Defines to which end of the spline the new points will beaded.

- Beginning – Will add the point before the first spline point
- End – Will add the point after the last spline point

1.6.4. Add Node

When toggled, a new Node object will be created and bound to each new created point. Nodes are game objects which are linked to spline points and update the points when they are moved in the scene. A more detailed explanation is given in the Nodes chapter.

1.7. Selecting points

To select a point either click on it in the scene view or select it from the inspector using the "Select" drop down menu inside the Edit tab.

When a point is selected, its parameters will be displayed in the inspector:
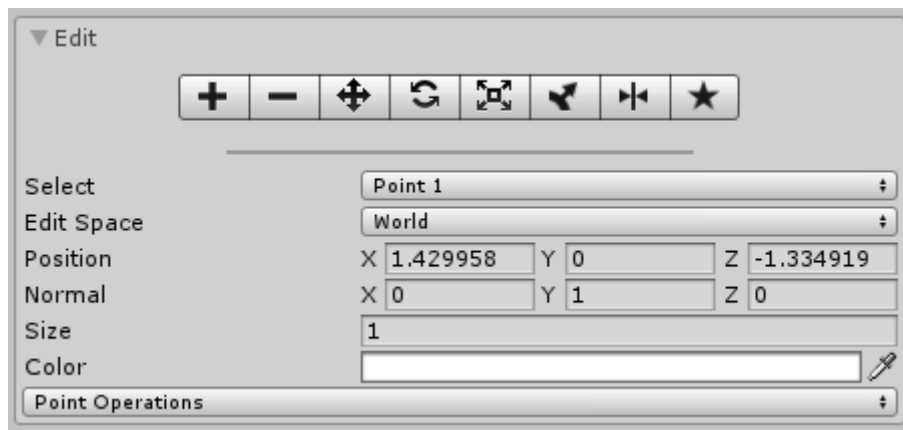
Selected points are highlighted in blue by default and are drawn with an additional hollow circle surrounding them:
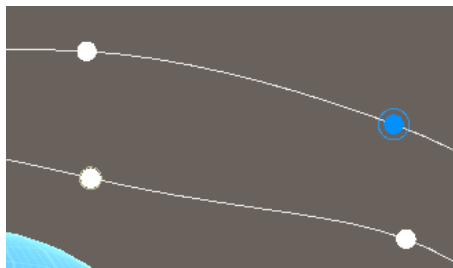
1.7.1. Selecting multiple points

To select multiple points either drag-select them in the scene view or select them one-by-one while holding Ctrl on the keyboard.

- Pressing Ctrl + A while in the scene view, will select all points inside the spline.
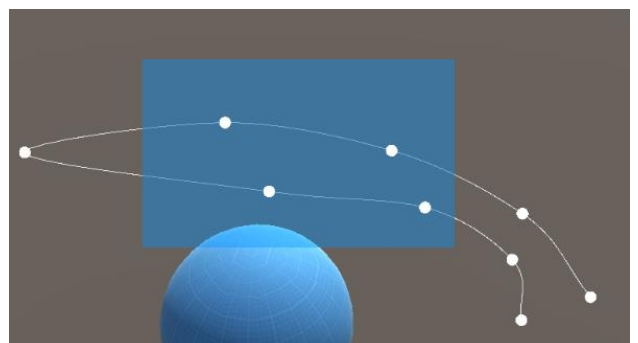
1.8. Deleting points

Points can be deleted by selecting them and pressing Delete key on the keyboard or by entering Delete point mode by clicking on the Minus button inside the Edit panel.
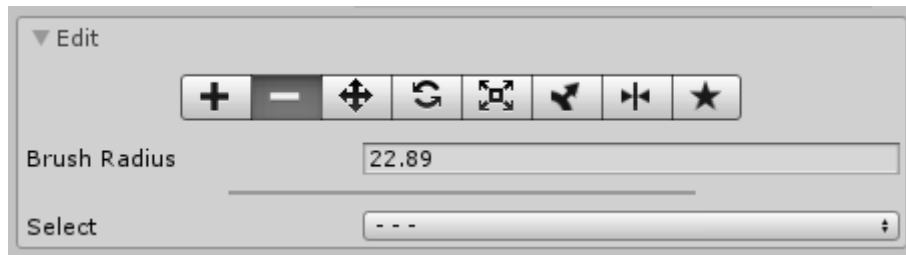
The Delete Mode toggles the delete brush tool. The brush tool has a radius in which it looks for points. Clicking and holding while dragging along the scene view will delete any spline points that fall under the delete brush' range.
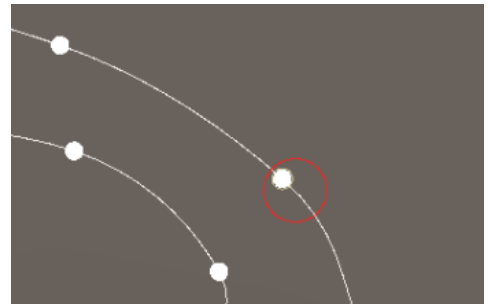
1.9. Undo

Any action can be undone using Unity's undo (Ctrl+Z) or re-done (Ctrl+Y).

2. Editing Splines

A spline control point has a couple of properties:

- Position – the position of the point
- Normal – a normal vector used for calculating rotations around the spline
- Tangent 1 and 2 – Tangent positions for Bezier splines
- Size – A size of the point (1 by default)
- Color – a color for the point (white by default)

These parameters can be edited quickly in a user-friendly editor, dedicated for that.

Once the control points of a spline are created, they can be edited one by one or together in a group. The most basic editing mechanism is dragging the points directly inside the scene view when not in any of the editor modes (Create, Delete, Move, Rotate, etc.). This moves the points along the camera plane so when working in 3D it is recommended that a Top, Right or Forward perspective is used in the editor. If more than one point is selected, when dragging a point on screen, the rest of the selected points will also move.

2.1. Move Tool (W)

The spline point Move Tool (which is different than Unity's move tool) can be toggled from the Edit panel toolbar by clicking the Move icon, or by pressing W while the Edit panel is open and the mouse is inside the scene view.
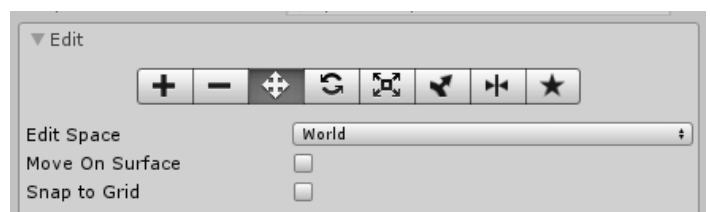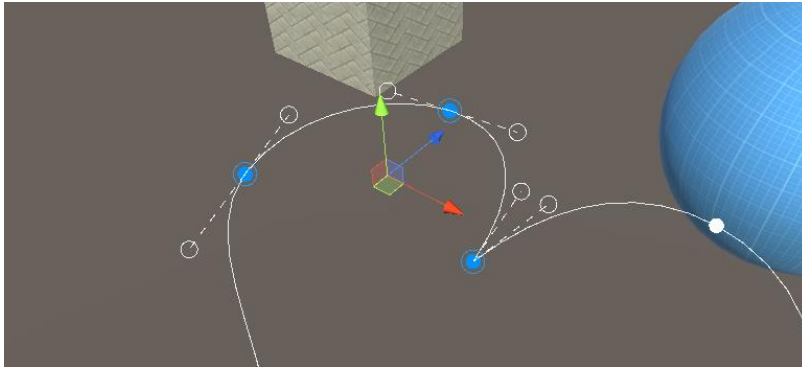
The Move tool provides a position handle which works exactly like the regular editor move tool. The difference is that it only operates on spline points and when the spline point Move tool is on, the editor move tool cannot be selected.

The Edit Space property of the move tool defines how the handle will be oriented. World will orient the handle along the world's axis. Transform will orient the handle along the Spline Computer's transform rotation and "Spline" will orient the handle along the spline. Note that "Spline" edit space only works when a single spline point is selected.
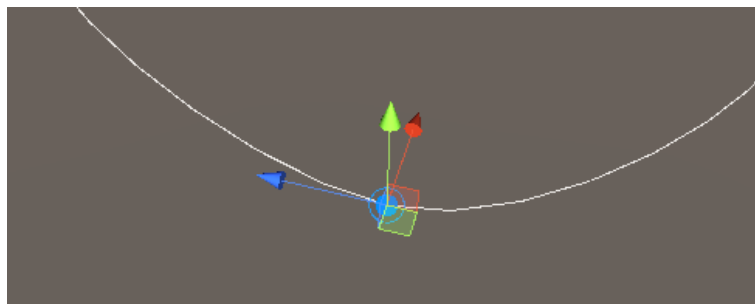
The Move On Surface toggle allows the points to be projected onto a collider when moving them (uses Raycasting). This is very similar to the Surface point creation mode, except this option allows to project already created points onto surfaces. When the Move On Surface toggle is checked, the position handle is replaced with a circle handle. Clicking and dragging the circle handle over a surface will position the points at the surface point where the mouse hovers.
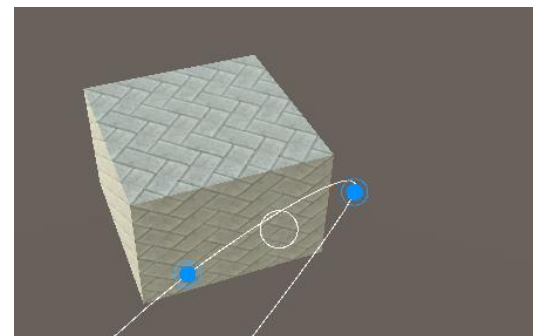
2.2. Rotate Tool (E)

The Rotate tool will rotate a single point's tangents and normal or rotate multiple points around their average center.

Two checkboxes define whether the normals of the points should be rotated and whether the tangents of the points should be rotated.

14
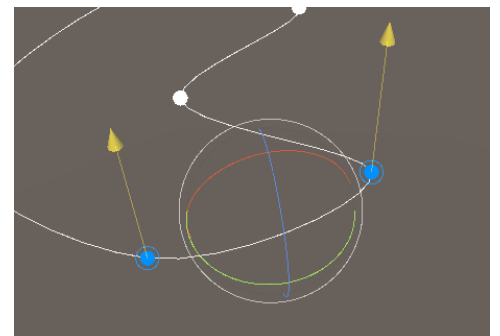
2.3.   Scale Tool (R)

The Scale tool will scale a single point's tangents and size. When multiple points are selected, the points' positions will also get scaled and offset based on their average center.

Similarly to the Rotate tool, scaling tangents and sizes can be toggled on or off separately using the tool's properties displayed under the toolbar when the tool is selected.
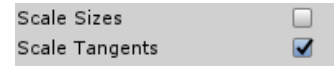
2.4.   Normal Tool (T)

The Normal Tool is a new kind of tool developed to ensure easy and intuitive normal editing for spline points. It has two modes.

2.4.1. Auto Mode

In Automatic mode, the normal are automatically calculated to always be perpendicular to the spline. This is usually the desired result so Auto is the mode that is selected by default. When a point is selected, a circular handle is drawn around it with a line going from the point, towards the normal direction. At the tip of the line, there is a circular free-move handle. Dragging this handle will rotate the normal around the spline.

2.4.2. Free Mode

Free mode allows the circular handle to be moved freely in each direction. This is a legacy tool which is usually not used anymore, however, there might be cases where spline normals don't need to be perpendicular to the spline and this is where this mode comes in handy.

2.4.3. Normal Operations

The Normal tool offers a range of normal operations which can be performed on the currently selected points. Using a normal operation is done by simply selecting it from the Normal Operations dropdown list while the desired spline points are selected.

The available normal operations are:

- Flip – Inverts the direction of the normal
- Look at Camera – The normal will look towards the editor camera's position
- Align with Camera – The normal will align with the editor camera's direction
- Calculate – The normal will automatically be calculated based on the location of the previous control points
- Left – World Negative X
- Right – World Positive X
- Up – World Positive Y
- Down – World Negative Y
- Forward – World Positive Z
- Back – World Negative Z
- Look at Avg. Center – will look at the average center of the selected points (works only when multiple points are selected)
- Perpendicular to Spline – Will make the normal perpendicular to the spline while trying to preserve the general direction

2.5. Mirror Tool (Y)

The Mirror tool does what its name suggests – it duplicates and mirrors the spline points along a given axis.
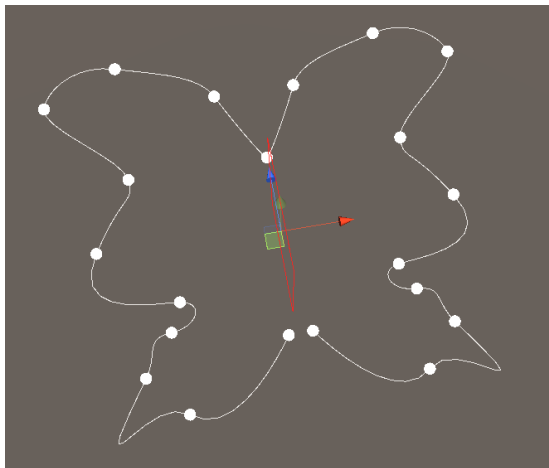
The mirror tool reflects the spline based on the mirror center which by default matches the position of the SplineComputer's Transform. If the symmetry center isn't visible, make sure to look for it around the computer's transform position.

The mirror center can be moved with the position handle attached to it or by modifying the center coordinates in the inspector.

By default, the spline is reflected along the X axis, but this can be changed using the Axis dropdown menu. If flip is unchecked, the points to
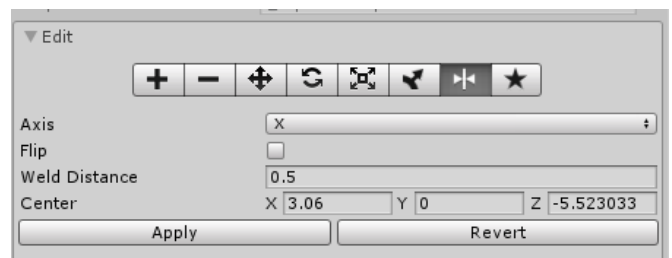
the left of the center will be reflected to the right, otherwise the points on the right will be reflected to the left.

The "Weld Distance" setting defines at what distance from the symmetry plane two reflected points will be merged into one. If the spline is of Bezier type, the merged point will have broken tangents.

The Apply and Revert buttons are used to either finalize or revert the changes made with the mirror tool. If the mirror tool is exited without clicking the Apply button, a dialog will appear asking if the made changes should be kept or reverted.

2.6. Primitives

Dreamteck Splines provides a list of procedural primitives and a preset saving system. Both are located in the Primitives & Presets tab inside the Edit panel (last button on the right).

2.6.1. Using primitives

To enter the Primitives mode, click on the "Primitives" button under the toolbar. Immediately, the first available primitive will be loaded and the spline will be changed.
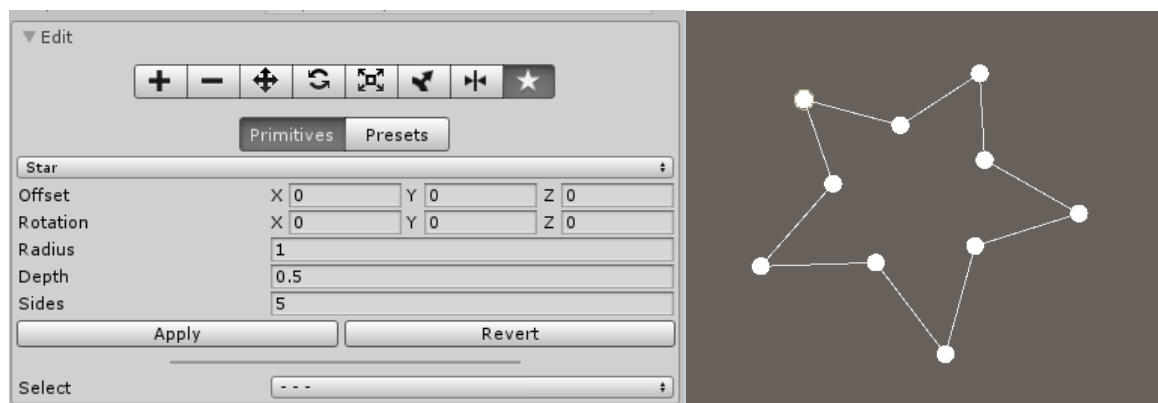
Different primitives can be selected using the dropdown menu just bellow the Primitives button. Each primitive has its own set of settings. Clicking "Apply" will save the primitive and make it into an editable spline. Revert will revert the previous state of the spline.

2.6.2. Using presets

When preset mode is entered, the spline is not immediately changed. Instead, there is a Use and a Create button:
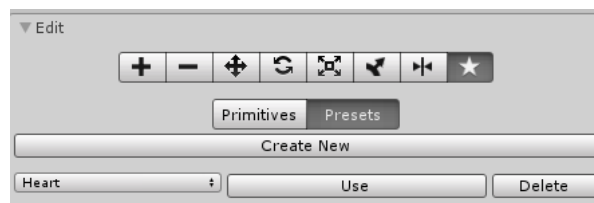
The Create New button will create a new preset out of the current spline.

The Use button will use the currently selected preset from the dropdown menu. Delete will delete this preset.

Spline presets are saved as files under Dreamteck/Splines/Presets When updating the plugin version, it is a good idea to backup this folder if custom presets are used.

2.7.   Point Operations

Dreamteck Splines provides a set of simple and handy operations for editing points. These operations can be found in a dropdown menu at the bottom of the Edit panel when at least one point is selected.

When an operation is selected from the dropdown menu it is automatically executed on the selected points.
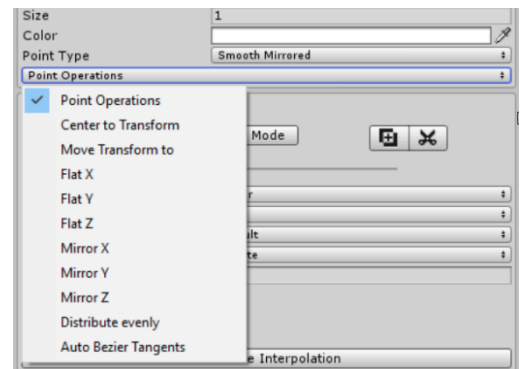
2.7.1.   Center to Transform

When splines are created, their points can be positioned very far from the position of the Spline Computer's Transform. Sometimes the control points need to be brought to the center of their transform so proper transformation can be applied. This operation brings all selected points to the center of the transform while maintaining their relative position.
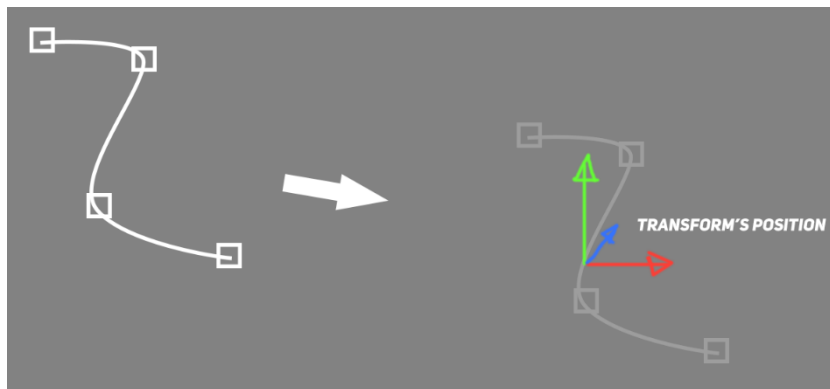
2.7.2.   Move Transform To

This is the reverse equivalent of Center to Transform. This operation will move the position of the Spline Computer's transform to the average center of the selected points without moving any points.

2.7.3.   Flat X, Y and Z

These operations flat the selected points' positions along the given axis.
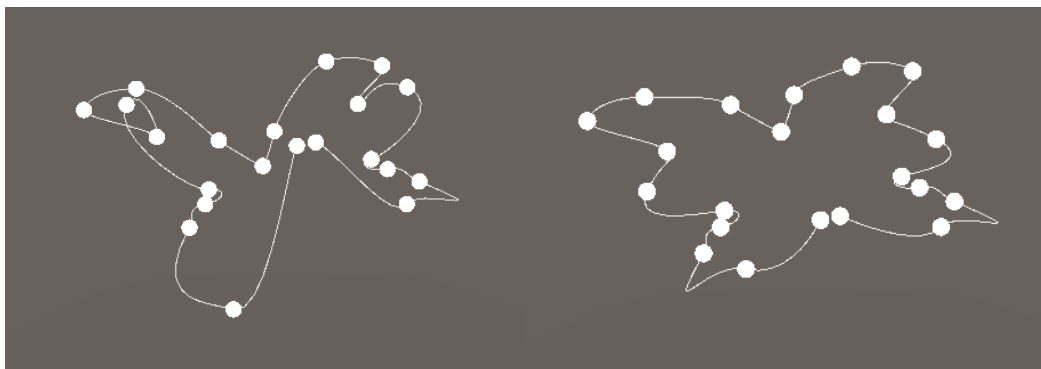
If the spline type is Bezier, a dialog will appear upon selecting the flat operation giving the option to flat only the points' positions, only the points' tangents or everything.

2.7.4. Mirror X, Y and Z

Will mirror the selected points around their average center.

2.7.5. Distribute evenly

Attempts to distribute the points at even distances from each other.

Before:
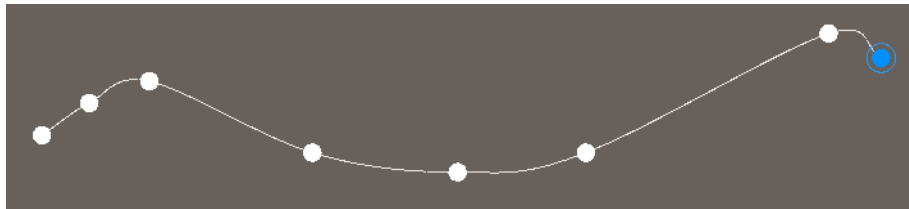
After:

2.7.6. Auto Bezier Tangents

Will automatically calculate the position for the Bezier points' tangents so that the spline starts to look like a Hermite spline.

2.8. Editing the Spline Computer's Transform

When a spline object is selected and the Edit panel is open in the inspector, the editor tools are not available and therefore its Transform cannot be changed unless values are directly typed inside the transform's property fields. In order to move, rotate and scale a spline using the editor tools, Edit mode must be exit by folding up the Edit panel:
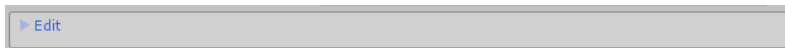
Once the Edit panel is closed, the editor tools will become available for the spline's object.

2.9. Framing control points

When a Spline Computer is selected and is in edit mode, the framing functionality of the Unity editor can be used to frame all or only the selected spline points. If the spline computer has one or more spline points, then going to Edit->Frame selected will frame the selected points or all points if none are selected. If the Spline Computer doesn't have spline points, then the default command behavior is executed.

3. Spline Computer Settings

Moving down in the Inspector, below the Edit panel, there is the Spline Computer panel. This is where the general settings for the spline are located as well as some operations which can be performed on the splines.
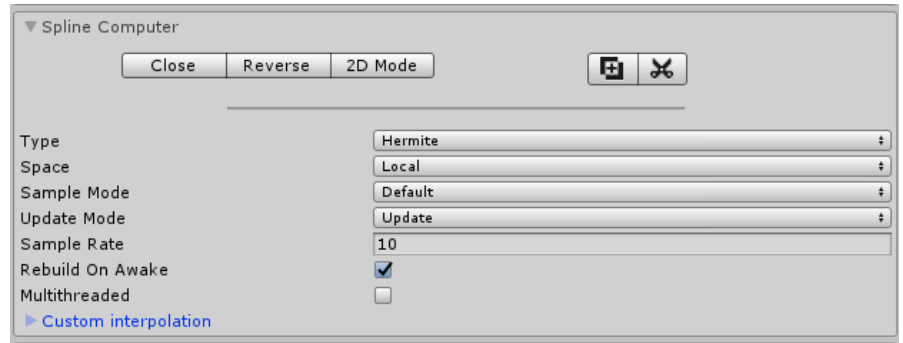
3.1. Type

The type of spline interpolation to be used. Dreamteck Splines supports four types of spline interpolation:

- Hermite
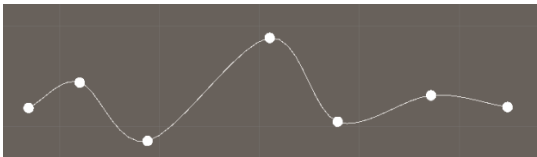
- Bezier
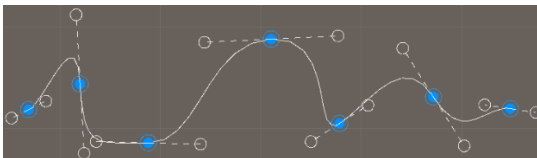
- B-Spline
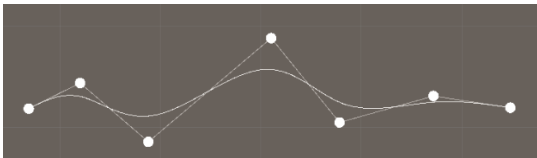
- Linear

For the Bezier spline type, each point gets additional tangent points which control the interpolation of the curve (connected with a dotted line).

3.2. Space

The coordinate system the spline should use:

- World – the spline uses world coordinates regardless of how the spline object's transform is positioned, oriented and scaled
- Local – the spline will transform with the object's transform

3.3.   Sample Mode

Splines, by default are evaluated with a [0-1] percent which spans across all spline points. For example, if a spline, consists of 3 points and it's evaluated at 0.5 percent this will always return the second point's position because it is in the middle.
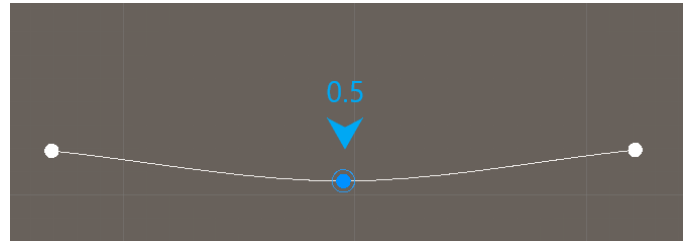
However, if the first and the second points are very close to each other and the third is very far apart, evaluating at 0.5 percent will not return the middle of the spline because it will still be returning point 2.

A lot of people who are new to splines encounter this issue when they try to extrude meshes or place objects along splines as some regions are denser than others. To illustrate this, here is the spline, drawn with thickness on:
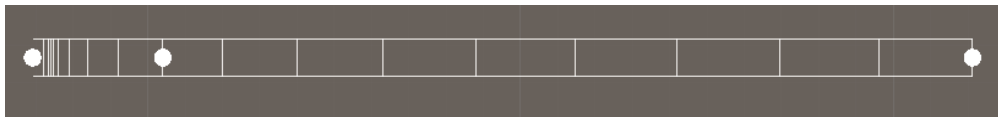
Each vertical line represents a spline sample. In this case, there are 10 samples between points 1 and 2 but there are also 10 samples in between points 2 and 3.

This is how the **Default** sample mode works. It is the fastest one in terms of performance but if the points are not distributed evenly, some spline regions may stretch or squash. There are three sample modes in Dreamteck Splines:

- Default – the default sample mode
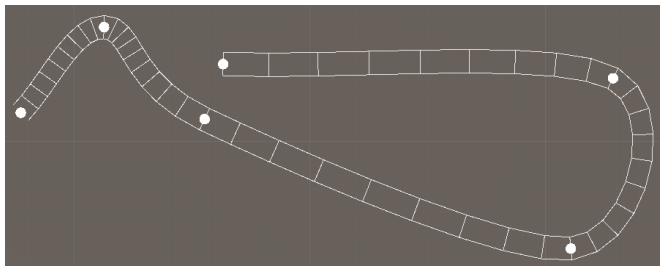
- Uniform – distributes the samples evenly along the spline but is heavy especially for larger splines
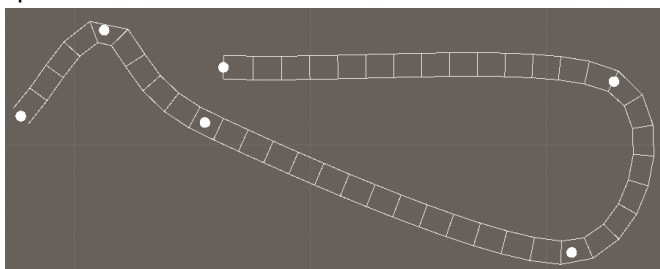
- Optimized – same as Default but performs an optimization operation which removes unnecessary samples. This also comes at the cost of performance.
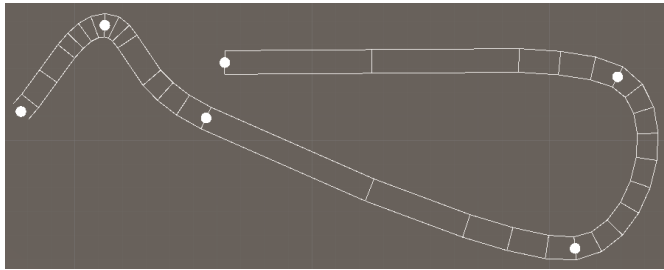
Note that in Default and Optimized modes, when moving a control point, the splines only update the samples in the region that is affected by the point. In Uniform mode, the entire spline is re-calculated.

In Optimized mode, an additional slider is presented to control the angle threshold for optimization.
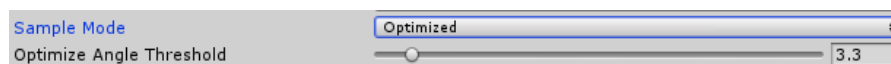
3.4. Update Mode

This property concerns when the splines are updated in runtime when there is a change to the spline's object transform or one of the spline points. In a single frame, a spline can be modified by a script multiple times. For example, the spline type can be changed and then some spline points can be modified:

```
SplineComputer spline = GetComponent<SplineComputer>();
spline.type = Spline.Type.BSpline;
spline.space = Space.Local;
spline.SetPoint(0, new SplinePoint(new Vector3(10f, 24f, 1f)));
```

Here, there are three operations on the spline. If the spline updated immediately after each operation, at the end of the frame, the spline would have re-calculated three times while it only needs to perform calculations once at the end of the command set. This is why Dreamteck Splines delays updates for the next frame by default. The Update Mode field defines when the update takes place.

- Update
- FixedUpdate
- LateUpdate
- AllUpdate – all three
- None – the spline will never update automatically during runtime and needs to be manually updated by calling Rebuild()

3.5. Sample Rate

The resolution at which the spline is sampled at. Higher sample rate means higher resolution. The number defines how many spline samples there must be between each two control points.

Sample Rate: 4	Sample Rate: 10
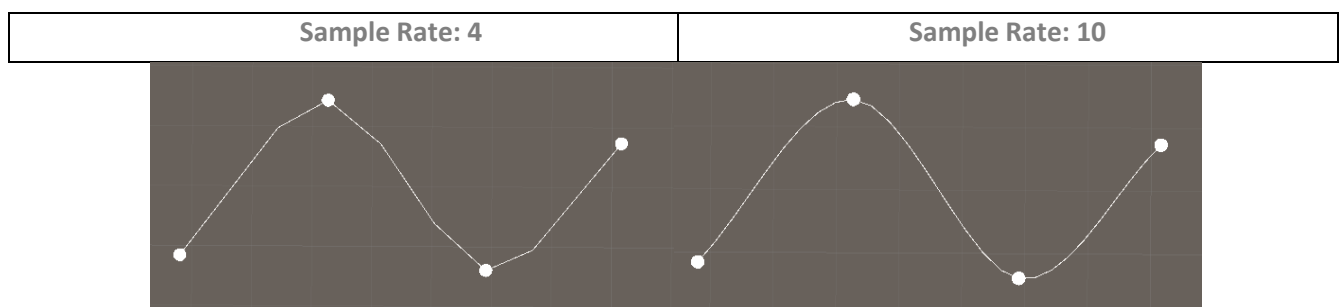
3.6. Rebuild on Awake

Should the Spline Computer rebuild as soon as it is active in the scene? This is often a good idea when instantiating splines from prefabs.

3.7. Multithreaded

Should the Spline Computer use a separate thread for calculating the samples? This is useful when either the spline is big, the sample rate is very high, the sample mode is set to Optimized or Uniform or all of the aforementioned.

Multithreaded splines might not update on the same frame. They usually lag one or two frames behind but that is only noticeable if an in-game spline editing is presented to the player (if the player has control over some spline points or properties).

3.8. Custom Interpolation

The custom interpolation foldout contains two buttons which are used to create animation curves that control how some control point values are interpolated as the spline is evaluated.
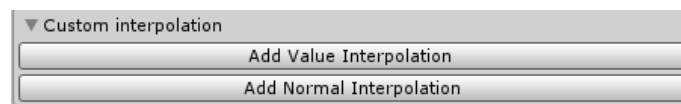
What can be interpolated?

Interpolation happens for the points' sizes, colors and normals. By default, linear interpolation is used but sometimes it is useful to have other types of interpolation. For example, when following a spline, it might be a good idea to use an ease-in, ease-out curve for the normal interpolation if point normals point in different directions.

3.8.1. Value Interpolation

Value interpolation refers to the interpolation of point sizes and point colors together. For example, here is a spline (thickness drawing is enabled), consisting of only two control points. The left one has a size of 0.5 and the right one has a size of 2.

This is how the size interpolation looks like when custom interpolation is used:
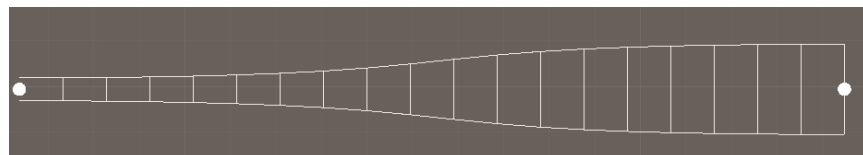
3.8.2. Normal Interpolation

How spline normals are interpolated in between two control points.

4. Spline Computer Operations

Going back to the top of the Spline computer panel, there are a couple of buttons which execute some useful operations for the currently selected spline(s).
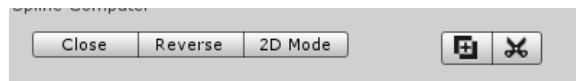
4.1. Closing and breaking a spline

A spline path consisting of four or more points can be closed. This will connect the first and the last points of the spline making the spline go in a loop. To close a spline, first ensure that there are at least four control points present, go to the inspector and click the "Close" button. If the close was successful, the last spline point will be moved to the first one and the Close button will read "Break".
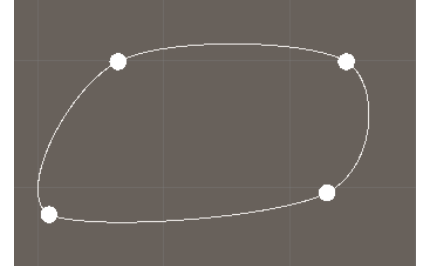
4.1.1. Closing a Spline During Point Creation

If a spline consists of three or more control points, it can be closed in point creation mode if the place where the newly created point on screen overlaps with the position of the first point from the spline's opposite end. Doing so will open a dialog asking whether the spline should be closed.

4.2. Breaking

To break a closed spline, press the "Break" button. The first point of the spline will be separated into two points.

Splines can also be broken at a different point. If a single point is selected when clicking "Break", that point will be separated.

4.3. Reversing a Spline

Sometimes it's useful to be able to reverse the control point order without changing the spline's shape. Doing so will reverse the spline's direction. The Reverse button does exactly that. When a spline is reversed, visually nothing changes but selecting the first point and looking it up in the inspector will reveal that it has become the spline's last point and vice versa.

4.4. Merging Splines

It is possible to merge two spline objects into one. To enter merge mode, click on the enclosed "+" button in the top right of the Spline Computer panel.
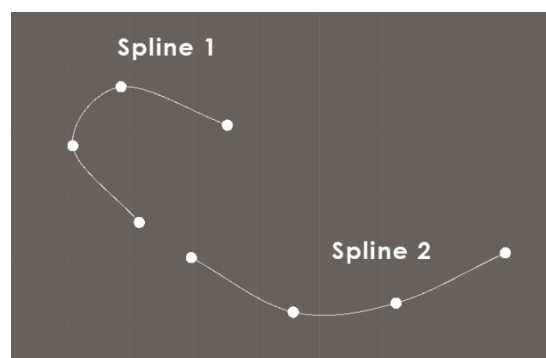
When Merge mode is active on one spline, the rest of the splines in the scene will be drawn with their ends marked as big hollow circles.

Clicking on either circle, will merge the given spline with the currently selected one.

There are two options for merging. One is the merge side. By default, the merge side is set to "End", meaning that whichever end is selected for the foreign spline, it will be merged to the selected spline's end point. Setting this option to "Start" will work the other way around.

"Merge Endpoints" defines whether the end points of the spline should be merged into one or not when merging.
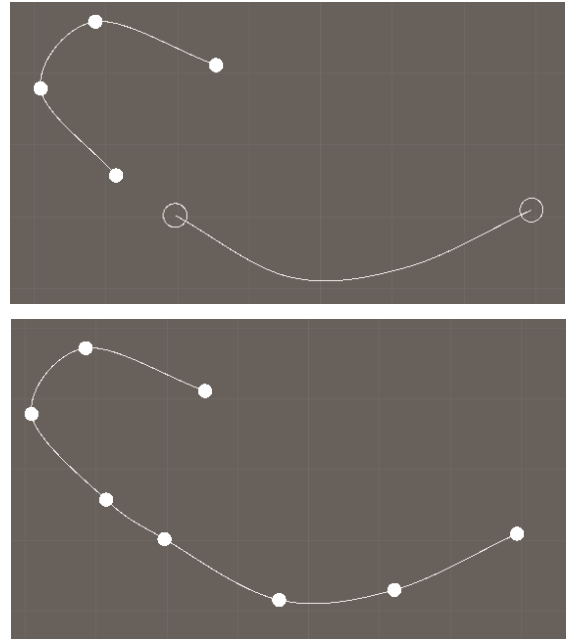
4.5. Splitting a Spline

A single spline can be split into two splines using the Split mode. In Split mode, a cursor is projected along the spline where the mouse is. Clicking on that cursor will split the spline into two splines immediately.

4.6. Transform-only Editing

If the Space property of the Spline Computer is set to Local, an additional toolbar will appear at the bottom of the Spline Computer panel with some transform tools:

These tools allow for editing the Spline Computer's transform without changing the spline.

When either of the tools is selected, the transform handle will appear at the game object's position and will have a label with the game object's name beneath.

This is useful when the spline's pivot point needs to be changed.
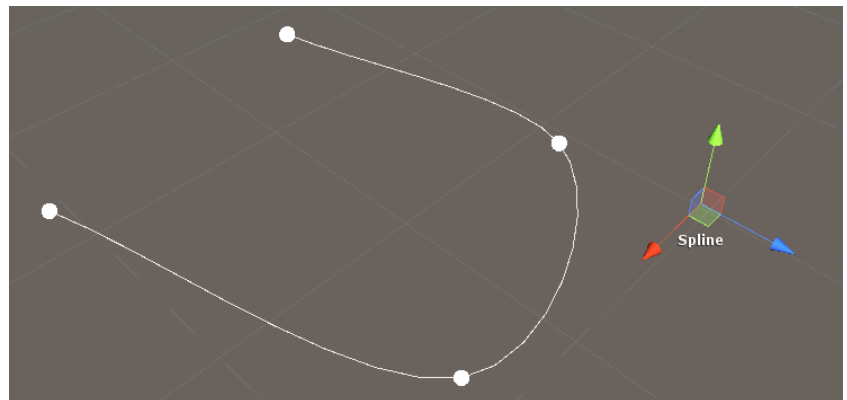
Switching to any of the tools from the Edit panel will deselect the currently selected transform tool automatically.

5. Spline Drawing and Info in The Editor

By default, splines are drawn only when selected in the editor but that can be changed from the bottom panel of the Spline Computer's inspector – Info & Debug.
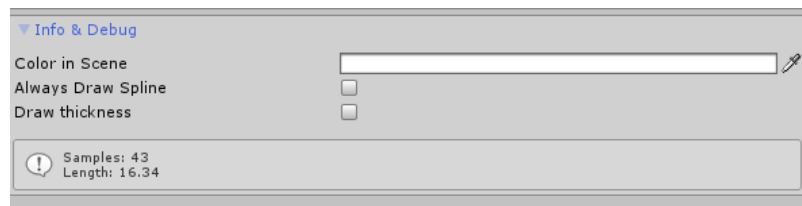
Expanding the Info & Debug panel, a couple options are presented:

- Color in Scene – The color of the spline used to draw it in the editor
- Always Draw Spline – Should the spline always be drawn regardless of whether it is selected?
- Draw Thickness – Toggling it draws the spline's thickness, based on the control points' sizes

If Draw Thickness is selected, each spline sample is drawn as a perpendicular line:
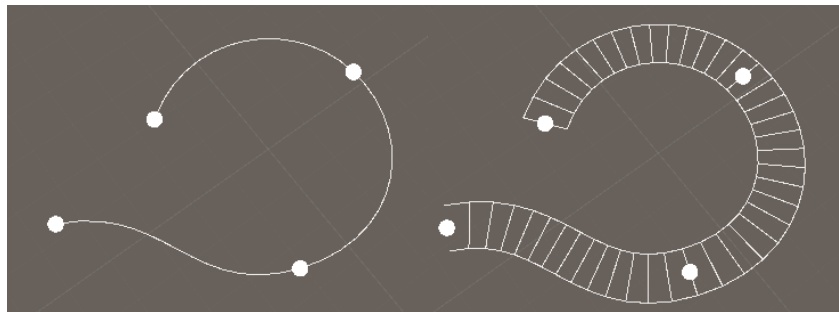
6. Using Splines (Spline Users)

Splines by themselves are intangible and invisible during runtime so to make use of a spline, the spline needs to be used. For that, Dreamteck Splines comes with a group of components called Spline Users. The Spline User components take a spline reference and utilize it to generate geometry, make objects follow splines, control particles/objects and much more. The Spline Users are all components in Unity and can be added to an object either through the editor or in runtime by calling the **AddComponent** method.

When an object with a Spline User component is selected in the editor, the referenced spline is automatically drawn.

Note that the Spline User component itself is rarely directly used since it doesn't do anything. The Spline User component is used as a base for the rest of the components that utilize the splines. In the following chapters, various Spline User components are described.

6.1. General Spline User Properties

All Spline User components have several base properties. Starting with the top, the first and most important property is the Spline field which defines which spline this user is using. If a spline isn't referenced, an error

26
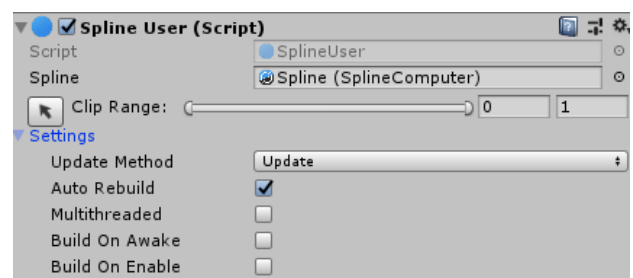
will be displayed in the inspector and the Spline User will not work.

Example in code:

```
    SplineUser genericUser = GetComponent<SplineUser>();
    genericUser.spline = mySpline;
```

6.1.1. Clip Range

The segment of the spline that this user will be using. A Spline User could operate on just a fraction of the spline.

The clip range is defined by the Clip From and Clip To values. By default, the editor displays the range with a MinMax slider:
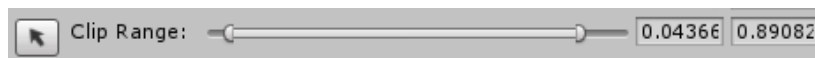
This type of control prevents the "clipFrom" value from being bigger than the "clipTo" value. However, if the sampled spline is closed, the clip range can be looped, meaning that "clipFrom" can be bigger than "clipTo" and the samples will be looped across the spline's connected start and end.

Example in code:

```
SplineUser genericUser = GetComponent<SplineUser>();
genericUser.clipFrom = 0.25;
genericUser.clipTo = 0.75;
```

6.1.2. Update Method

Similarly to the Spline Computer's Update Method property – it defines in which update cycle the logic for this Spline User will be executed – Update, LateUpdate or FixedUpdate.

6.1.3. Auto Rebuild

Should the Spline User automatically re-calculate if there is a change to the spline? If set to false, RebuildImmediate should be manually called to force the Spline User to update.

```
SplineUser genericUser = GetComponent<SplineUser>();
genericUser.RebuildImmediate();
```

6.1.4. Multithreaded

Enables multithreading for the Spline User's calculations.

Note: For mesh generating Spline Users, only the vertex calculation operations are performed on the separate thread – the information still needs to be written to the Mesh object in the main Unity thread.

6.1.5. Build on Awake

Forces the Spline User to re-calculate on Awake.

6.1.6. Build on Enable

Forces the Spline User to re-calculate each time it is enabled.

7. Tracing Splines

The Spline Tracers are a group of Spline Users, meant specifically for positioning a single object along a spline. This class of Spline Users fosters the Spline Follower component which is one of the most commonly used components when it comes to splines.

The Spline Tracer component expands on the SplineUser class by adding another set of properties that the rest of the tracers share.
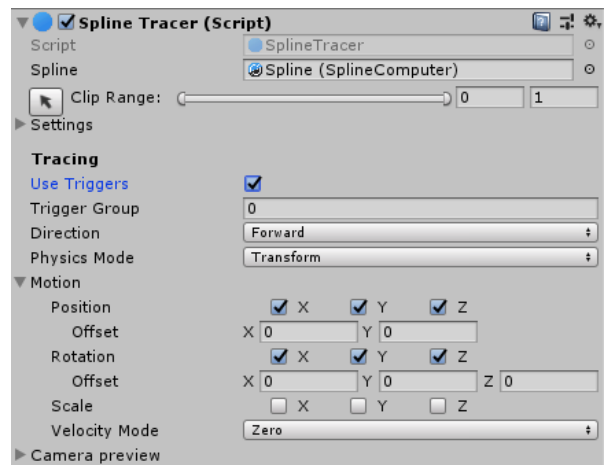
7.1. Use Triggers

Should this tracer use the triggers provided in the spline. Note that this refers to spline triggers (See Spline Triggers) and not Unity colliders set to act as triggers. Collision and physics are not used here.

If "Use Triggers" is toggled, a Trigger Group field is shown. This is the trigger group that this tracer will be using.

7.2. Direction

Defines the orientation of the object along the spline (Forward or Backward along the spline). In the case of the Spline Follower component, the direction also defines the direction in which the follower will move.

7.3. Physics Mode

The Physics mode of a SplineTracer can be either Transform, Rigidbody and Rigidbody2D. By default, the Physics mode is set to Transform which modifies the property of the Transform directly. Using Rigidbody or Rigidbody2D will modify the rotation and position of a Rigidbody component instead of the Transform. Rigidbody mode is preferred when physical interactions are needed.

Using PhysicsMode Rigidbody with a combination of the proper motion application settings can create realistically behaving followers.

7.4. Motion

The motion module defines how the transformation from evaluating the spline is applied to the object. There are three axes for each position, rotation and scale. When an axis is toggled, transformation will be applied along it. The position and rotation components use world axes while the scale axes are local to the object.
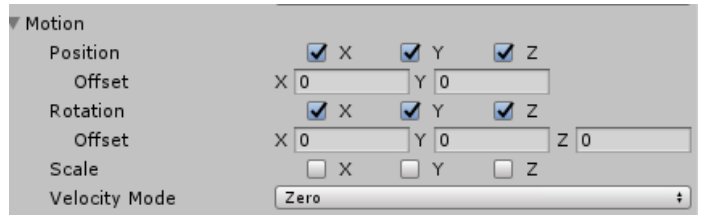
Applying scale is disabled by default meaning that all axes are unchecked. When applying scale is enabled, an additional "Base Scale" Vector3 field is presented. The base scale is the scale that is going to be applied to the object. If the spline has various sizes, the base scale will be multiplied by the size value of the current spline evaluation.

7.4.1. Velocity Mode

If the Physics Mode is set to Rigidbody, this defines how the velocity of the Rigidbody will be handled.

- Zero – rigidbody.velocity will always be zero
- Preserve – velocity will not be changed and will be kept the same
- Align – the velocity will be set to point in the direction of the spline
- Align Realistic – aligns the velocity with the spline but also reduces the velocity's magnitude based on the angle between the direction of the velocity and the spline.

7.5. Spline Follower

As the name suggests, the Spline Follower component provides spline following functionality for a single object.

The Spline Follower has two modes of following:

- Uniform
- Time
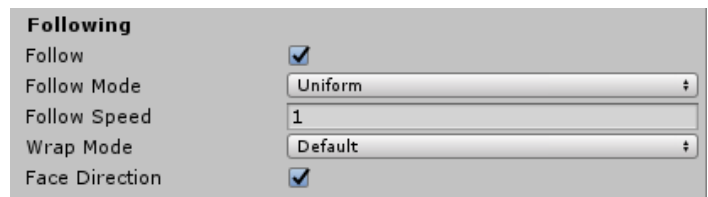
In Uniform mode, the Spline Follower will follow the spline with uniform speed, regardless of how the spline is set up. The Follow Speed property defines the distance in world units the object will travel along the spline in one second.

In Time mode, the "Follow Speed" field turns into the "Follow Duration" and the object moves along the spline by incrementing / decrementing its current spline percent [0-1]. This means that for non-uniform splines where points are closer or further apart, the object will follow with variable speed. The Time follow mode, however, will make sure that the follower has finished following the spline in exactly the time I is given to do so.

The "Follow" property defines whether or not the Spline Follower will be following the spline automatically. If Follow is disabled, the follower can be caused to follow manually each frame by calling it's public method **Move(float distance).**

Three wrap modes control the Follower's behavior when it reaches the end of a spline.:

- Default – when the end of the spline is reached, the follower will stop
- Loop – When the end of the spline is reached, the follower will start again from the beginning.
- PingPong – When the end of the spline is reached, the follower will change its direction

If "Face Direction" is toggled, the follower will face the direction of movement along the spline. If unchecked, the follower will always face the forward direction even if it moves backwards.

Where the follower starts along the spline is dictated by the Start position settings. If the "Project" checkbox is checked, the start position

will be calculated automatically based on the closest point along the spline to the Follower object in world space. The Set Distance button opens a small window which allows a distance to be input. This is a distance in world units from the start of the spline and it will be converted to a percent.
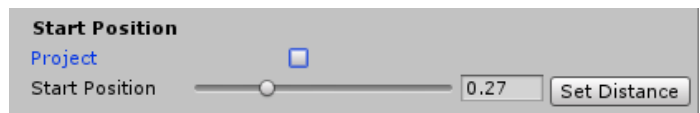
7.6.   Spline Projector

The Spline projector projects a single game object onto a spline. It will find the nearest point on the spline to the object's current world position and return the result at that point. By default, the Spline Projector is passive, meaning that it doesn't apply any results to objects but it can be set to automatically apply the projected result to any object by setting the reference of the Apply Target Object field.
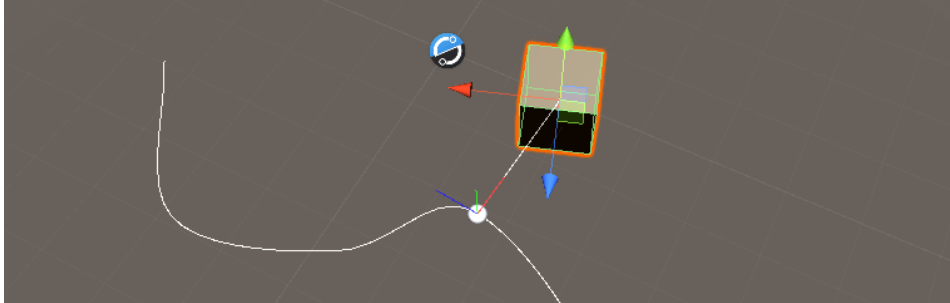
The Spline Projector has two modes:

- Accurate
- Cached

By default, the mode is set to Cached which means that the cached samples of the spline will be used. However, this in some cases results in jagged movement if the samples are not enough. This is why the Accurate mode is also introduced. In Accurate mode, the projector re-calculates the spline path from scratch using subdivision.

7.7.   Spline Positioner

The simplest Spline Tracer, the Positioner simply places a single object along a spline at a given percent.

The positioner has two modes – Percent and Distance

- Percent maps the whole spline to the range [0-1]
- Distance positions the object at the given distance from the spline's start
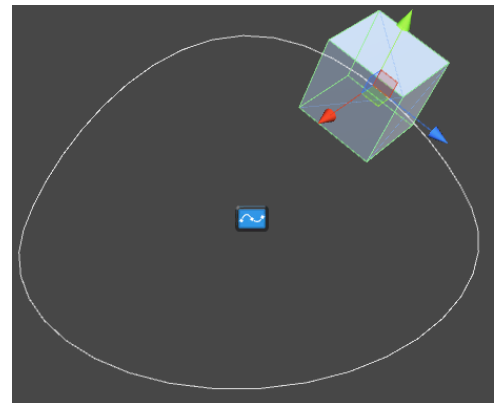
8. Spline Mesh Generation

The Mesh Generators are a special kind of Spline User class, dedicated to generating mesh geometry using a spline. The Mesh Generator component by itself doesn't do anything but is used as a base for creating other components. It provides basic properties, functionality and pipeline which are needed for the generation of meshes. It comes with its own custom editor which extends the SplineUser editor.
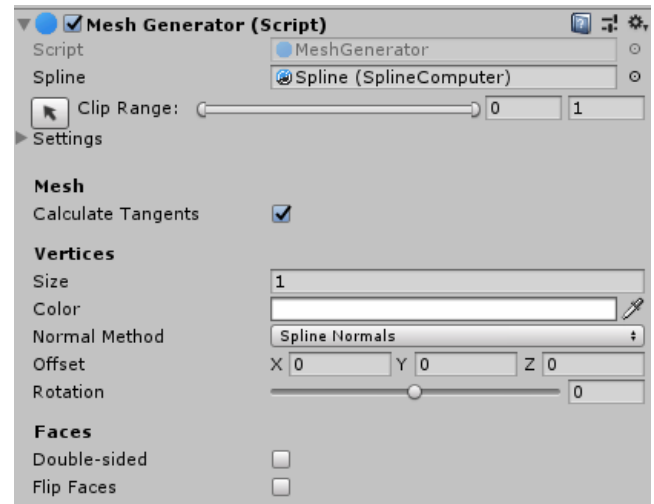
The MeshGeneration properties and functions are:

- Calc. Tangents: If checked, tangents will be calculated for normal mapping
- Size: The size of the generated geometry's circumference around the spline.
- Color: A color multiplier for the vertices
- Normal Method: the method used for the vertex normal calculation
- Offset: A Vector3 offset direction local to the spline.
- Rotation: Rotation in degrees around the spline
- Double-sided: If checked, the generated mesh will be double-sided
- Flip faces: If checked, the generated mesh will have flipped faces
- Info & Components: a foldout with information about the mesh and the mesh components
- Bake mesh button: Button used for mesh baking (See Baking Mesh Generators)

Each MeshGenerator's behavior is affected by the spline points' colors and most of them are affected by the points' sizes.
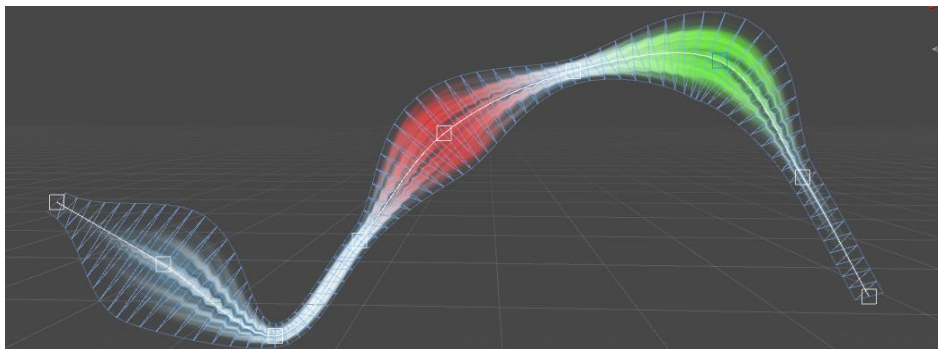
8.1. Collision Generation

If a Mesh Collider component is present, the Mesh Generator will offer to update it when geometry is generated. Since updating mesh colliders is a heavy task, when a Mesh Collider is present an additional value will be exposed in the inspector. This value is called "Collider Update Interval" and defines how frequently the mesh collider will be updated.
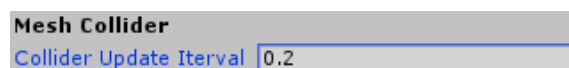
Setting this value to 0 will cause the Mesh Collider to update as soon as there is a change in the mesh. Note that the Mesh Collider will not update if there aren't changes in the mesh.

If multithreading is enabled, the mesh generation algorithm will be executed on another thread. However, due to the thread-unsafe nature Unity's API, mesh writing, collider updates and optimization are done on the main thread.

Note: All mesh generators can generate a mesh with up to 64 000 vertices since this is the maximum vertex count per mesh supported by Unity. If the vertex limit is exceeded for the generated mesh, an error message will appear in the console and the mesh will stop updating.

8.2. Spline Mesh

This is a universal component for extruding and placing meshes along a spline. It produces a single mesh which can have multiple material IDs.
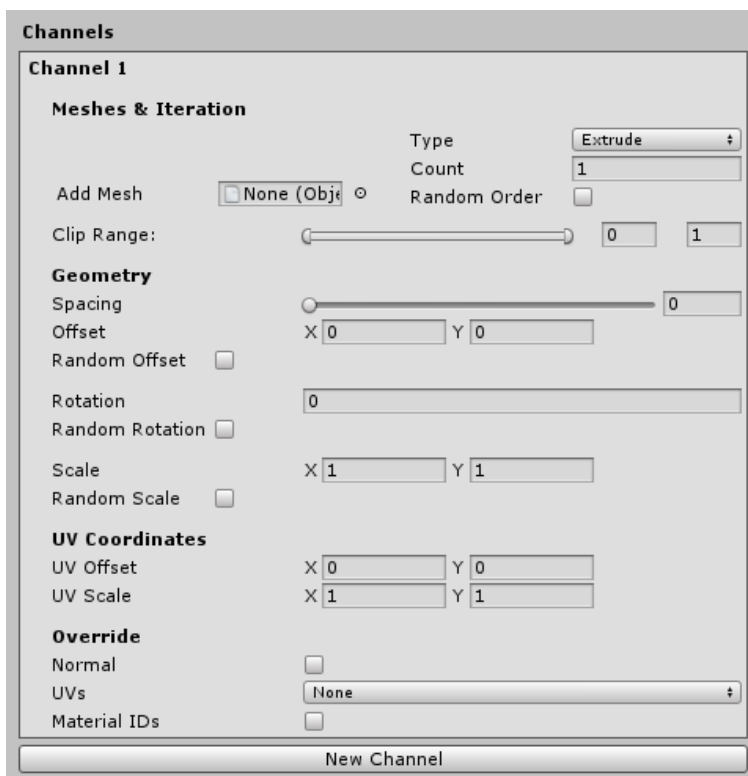
The Spline Mesh component contains channels which contain meshes. Each channel is computed independently which makes it possible for complex meshes with lots of different elements to be composed.

As soon as a channel has a single mesh assigned to it, the mesh will be extruded along the spline. Adding more meshes will cause the channel to iterate through them in the order of adding. It is possible to randomize the iteration by checking the "Random Order" checkbox. Note that when more than one meshes are added to a channel, they will not appear unless the channel's mesh count (to the right) is set to at least the number of the meshes.

There are two types of channels:

- Extrude – deforms the meshes along the spline
- Place – places the meshes along the spline without deforming

Each channel has its own clip range which defines the spline region it is going to take up.

33

When a mesh is added to a channel, it can be edited prior to the generation. This helps for correcting pivot orientations, unwanted offsets or sizes. To configure a mesh, click on it in the Channel tab – it will open the Configure mesh window:
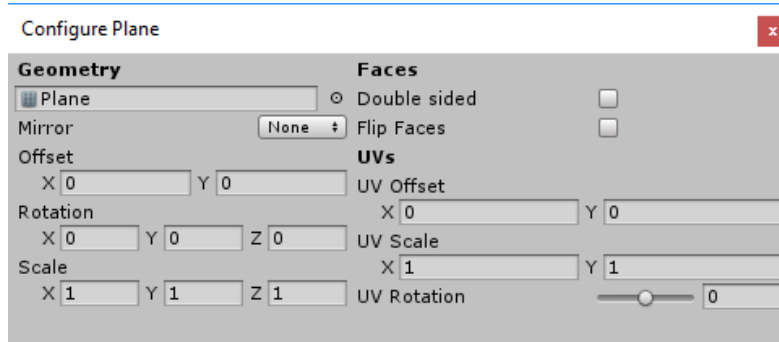
It is also possible to use a custom handler function for controlling the offset, rotation and scale along the extrusion. This can be used for some interesting effects and deformations (see API Reference).

8.3.   Spline Renderer

The Spline Renderer is an analogue of Unity's Built-in Line renderer. It uses a spline to visualize instead of a set of points. It's the first of a set of behaviors, derived from a special Spline User class called MeshGenerator. This class is used to provide basic functionality for creating procedural geometry using a spline.

The generated geometry will be oriented towards the camera on each render cycle meaning that this mesh is constantly updated and therefore bigger meshes might cost performance.
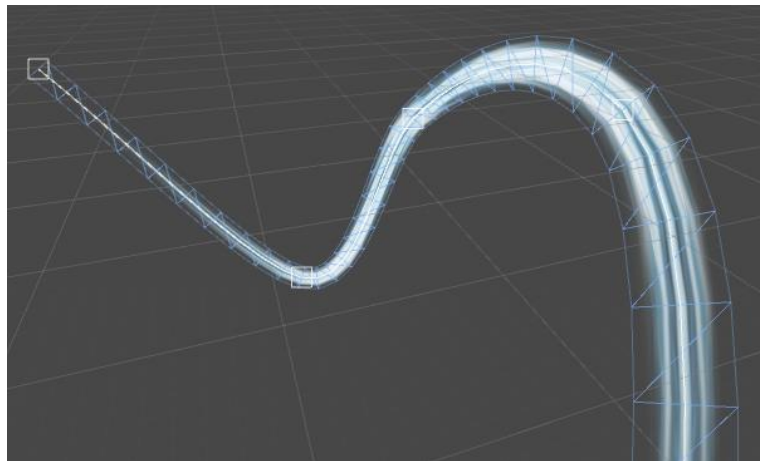
Because of the triangular nature of the mesh, in some cases the texture coordinates of the Spline Renderer might appear jagged.
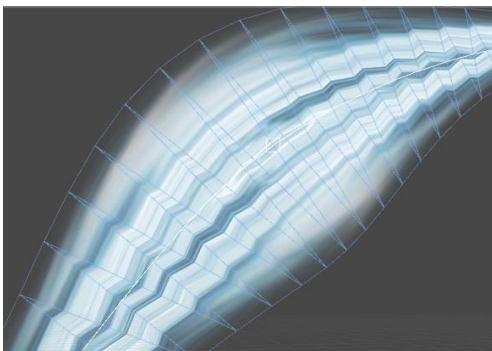
This is most often observed when there are spline points with different sizes and the solution to this is to add more edge loops. To do so go to the inspector and increase the number of Slices:

This will reduce the jagged artefacts. The same applies for the Path Generator component.
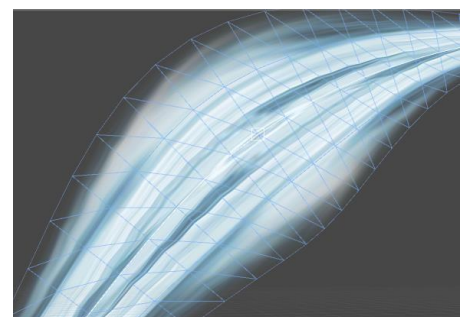
The Spline Renderer component is the only mesh generator which does not update Mesh Colliders.

8.4. Path Generator

The Path Generator is very similar to the Spline Renderer with a few exceptions. First, unlike the Spline Renderer, the Path Generator does not orientate the generated geometry to face the camera. Instead the geometry is orientated in the direction of the spline normal.

Like the Spline Renderer, the Path Generator has the option for multiple edge loops (called Slices) to prevent jagged texture coordinates. However, the Path Generator's slices also serve a function of deforming the generated path. Turning on the "Use Shape Curve" option will expose a curve editor. If the slices of the Path Generator are set to a value bigger than 1, then the path can be deformed using the curve editor.
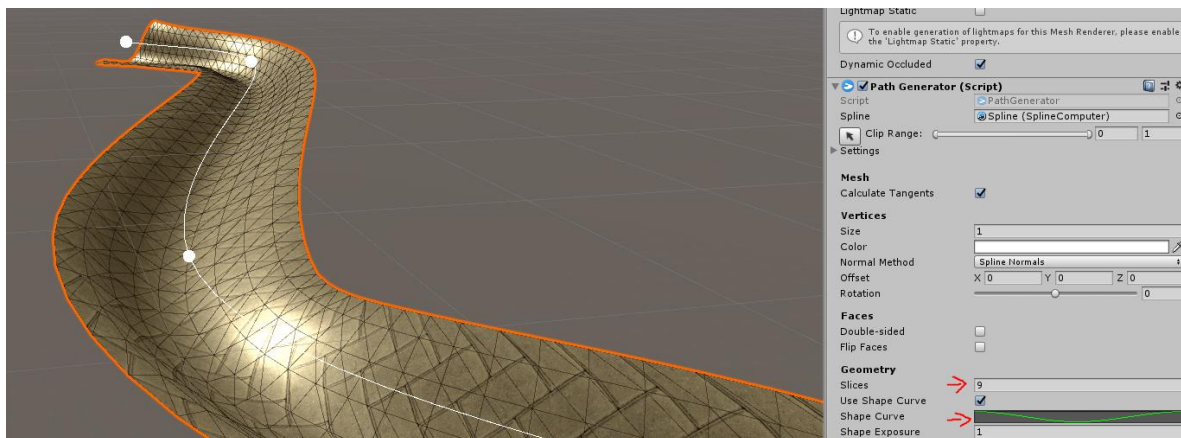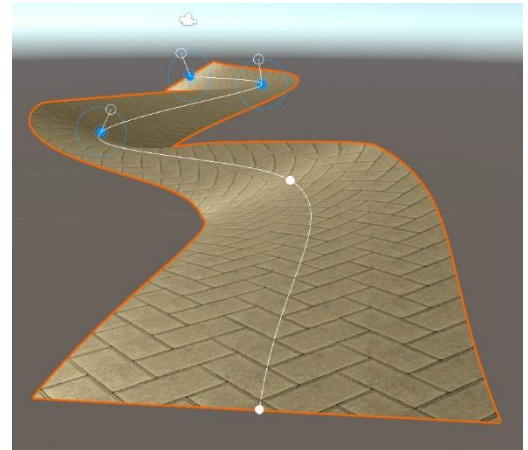
8.5. Tube Generator

The Tube Generator is useful for creating geometry, revolving around a spline like cables and tubes. It generates a hollow mesh which can have caps at both ends.

The type of caps is defined by the Cap property:

- Flat
- Round

The Revolve property [0-360] controls the spin of the generated tube around the spline. If less than 360, the Tube Generator will generate open geometry.
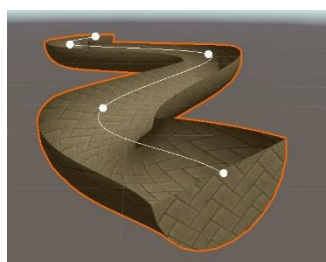
8.6.   Surface Generator

The Surface Generator creates a 3D surface from a spline. These surfaces can be used for creating different kinds of platforms, natural water pools, shape animations etc. They don't require the spline to be closed to work.

By default, the generated surfaces don't have thickness which may not look good in some situations. To add thickness to the surface, increase the "Extrude" value in the inspector.
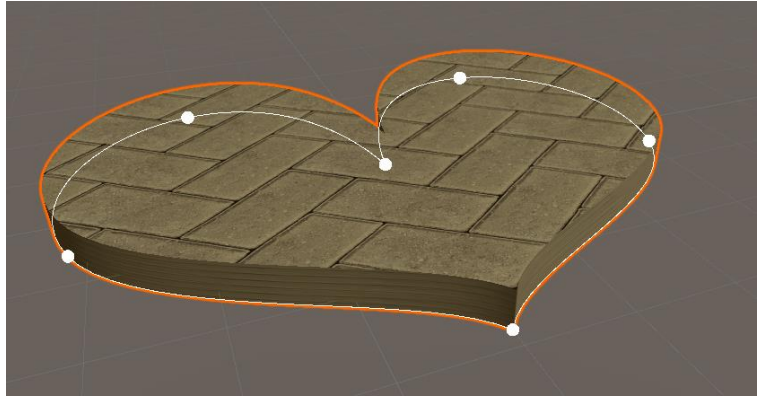
If Extrude is different than zero, a secondary set of UV coordinates which control the side UVs will be exposed in the inspector.
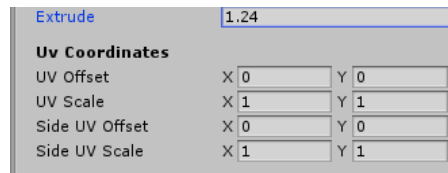
Surfaces can also be extruded along another spline. Drag the SplineComputer component of the extrusion spline in the "Extrude Path" field in the Inspector:
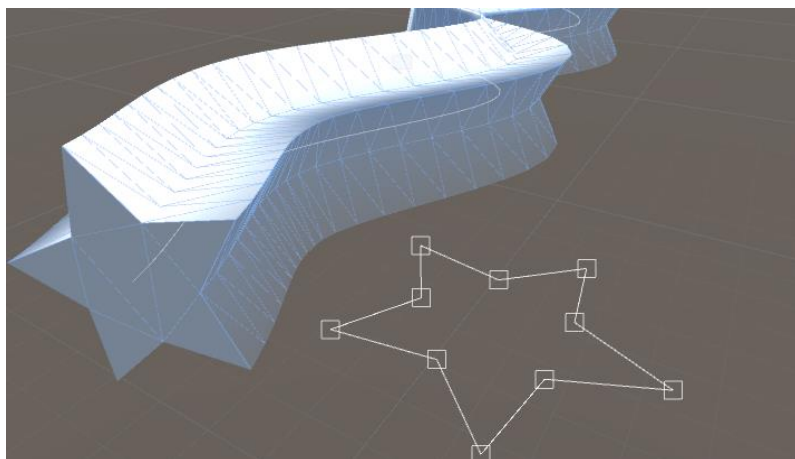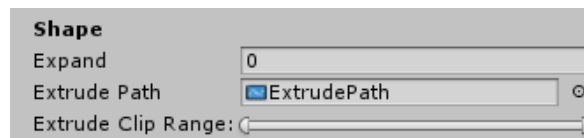
8.7. Baking Mesh Generators

Each Mesh Generator component can be baked into a static mesh with lightmap UVs. To do that, click the "Bake Mesh" button at the bottom of the Mesh Generator's inspector. This will open the Bake Mesh window.
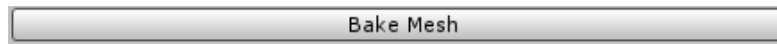
Bake Mesh

The Bake Mesh window provides a range of settings for baking and a Bake button.
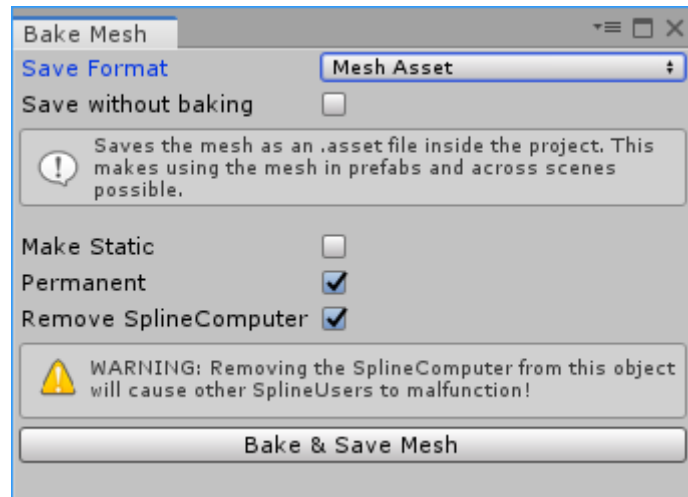
- Save Format – In what format to save the mesh?
  - o Mesh Asset – Will save the mesh as a Unity .asset file in the project. This creates a full-featured Unity mesh asset
  - o OBJ – Will save the mesh as an open source OBJ file which can be edited in modeling programs. Some vertex properties like color are not supported.
- Make static – will make the object static when bake is finished.
- Permanent – will remove the Mesh Generator component. Usually it's best to leave this unchecked.
- Remove SplineComputer - will remove the Spline Computer component that is referenced by the Mesh Generator

9. Particle Controller

The Particle Controller uses a Shuriken particle system and places its particles along a spline.

This component offers a preview right in the editor if it's attached to the particle system that it controls.

The Particle Controller has an Emit point which controls where new particles will be spawned. The modes are:

- Beginning – Emits particles from the beginning of the spline
- Ending – Emits particles from the final point of the spline
- Ordered – Emits particles from the beginning of the spline towards it's end based on particle index
- Random – Emits particles between the beginning and the end of the spline on a random basis

Another very important option of the Particle Controller is the Motion type. This controls the particle behavior over its lifetime. The available motion types are:

- Use Particle System – Uses the motion, defined by the Particle System component
- Follow Forward – Particles move forward along the spline based on their lifetime
- Follow Backward – Particles move backward along the spline based on their lifetime
- By Normal – Particles get their initial velocity set in the direction of the spline sample's normal
- By Normal Randomized – Same as above but the direction vector is randomized
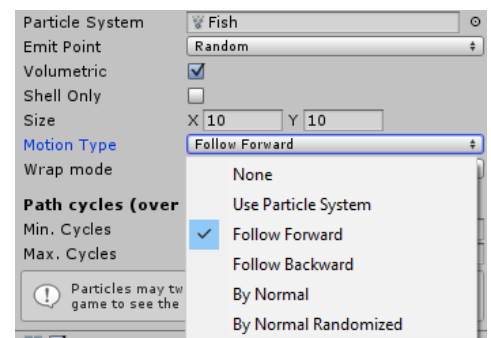
10.   Object Controller

The Object Controller component instantiates and positions Game Objects along a spline. This could be done both during runtime and in the editor.
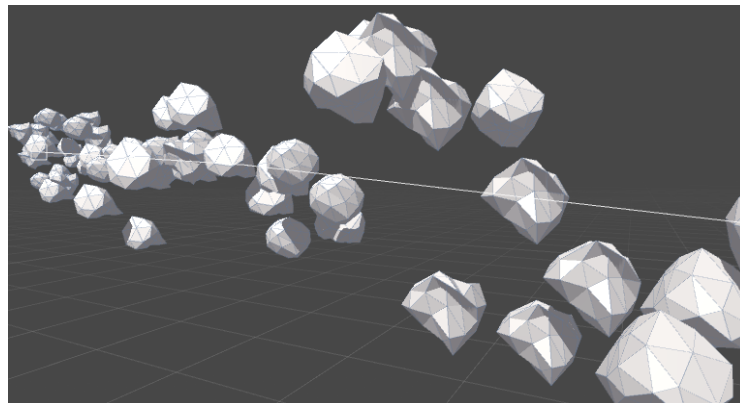
The Object Controller has two Object Methods which define how the objects it controls are created.

- Instantiate – This will instantiate new objects
- Get Children – This will not instantiate new objects but instead will take the existing children of the Object Controller's Transform and use them.

In Instantiate mode, a Game Object array is presented. For Instantiate to work there must be at least one added Game Object reference.

A Spawn Count property is presented. Modifying it will create or destroy objects.

The Iteration dropdown defines in what order the Game Objects are instantiated. If Ordered is selected, the Game Objects will be spawned in the order they are added. To change this order, use the up and down arrows next to the objects. In Random mode, objects will be instantiated in a random order.

If Delayed spawn is turned on, the objects will spawn over a certain period of time. This is mostly used to prevent spike lags caused by spawning many objects at once. Delayed spawn does not work in the editor.

In Get Children mode, the child objects of the Object Controller's Transform will be used. Adding or removing objects from the hierarchy will include or exclude them from the controller.
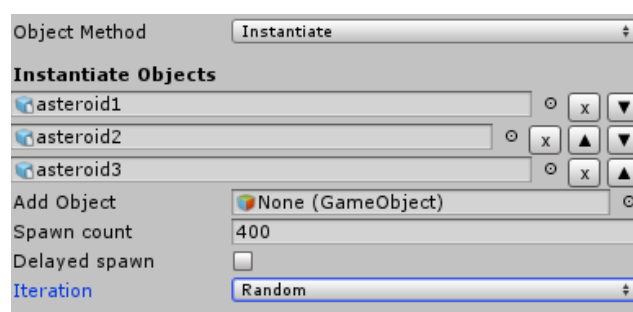
11.   Length Calculator

A basic component which calculates the length of the spline using the approximation rate defined by the Spline Computer's precision and the Resolution of the Length Calculator component.

The Length Calculator can have length events added to it which fire when the spline has reached a certain length.

To add an event, click the "Add Length Event" button. This will create a new empty event. Set the "target length" value to the value that should be listened for, select the target object (an object with a certain behavior), select the method and assign the argument's value. The Dropdown menu tells the Length Calculator when to listen for the specific event.
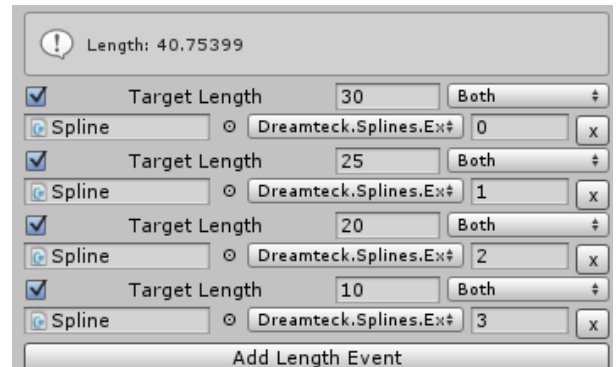
- Growing: will invoke the event when the target length has been reached by growing
- Shrinking: will invoke the event when the target length has been reached by shrinking
- Both: will invoke the event when this value has been reached or passed in both directions

This component has one additional module that controls its behavior. Refer to Triggers and Modules for more information.

12.   Object Bender

The purpose of the Object Bender is to bend a game object and all of its children along a spline. The Object Bender will position and orient all of the objects in the hierarchy along the spline. The difference between the Object Bender and the Object Controller however is that the Object Bender will also bend any Meshes, MeshColliders and SplineComputers in the hierarchy.

When an Object Bender is added in the editor or in runtime, it will be in edit mode by default. This means that the object bender will not work. The edit mode allows the developer to edit the children in the hierarchy.
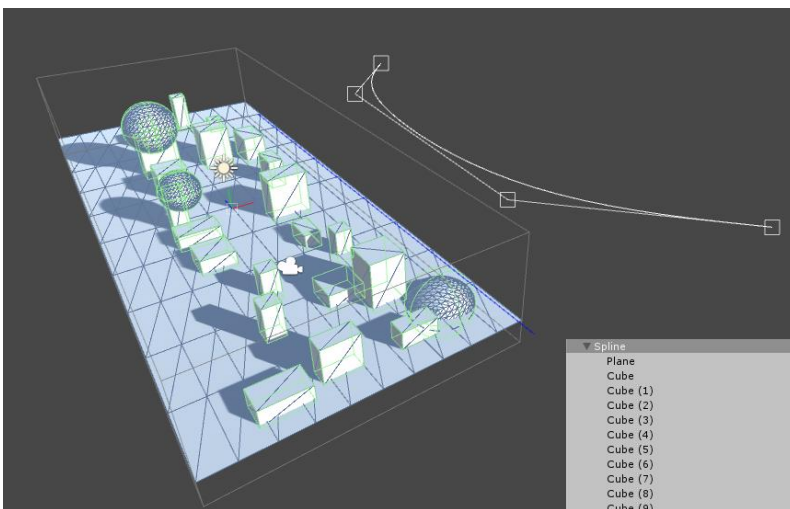
In edit mode, the ObjectBender editor will visualize the bounds of the parent object and all of the children in the hierarchy. These bounds include the Transform components, meshes, colliders and SplineComputers.

When in edit mode, there is a "Bend" button at the bottom of the ObjectBender's inspector. Clicking it will exit edit mode and will start bending the objects along the spline.

When in Bend mode, all objects will be controlled by the spline. Each object from the hierarchy will be editable after it has been bent but if the spline updates, it will get recalculated and repositioned.

In Bend mode, the "Bend" button will be replaced with an "Enter edit mode" button which will revert the bend and enable editing.

The ObjectBender bends all the objects along one of the three local axes. The axes are local to the parent object. The bend axis can be selected through the ObjectBender's inspector and will be applied immediately even if the ObjectBender is in edit mode.

The ObjectBender component gives complete control over what is bent and how it's bent. A bend property is created for each object from the hierarchy. Each bend property allows the developer to toggle certain aspects of the bending process on and off for each object individually or toggle off bending for the object completely.

All bend properties are automatically exposed in the inspector. Clicking on a property will highlight it in blue and will open the property editor. Several properties can be edited simultaneously. To select multiple properties, hold down Ctrl and click on properties to add them to the selection. Shift + click will select multiple properties in a row.
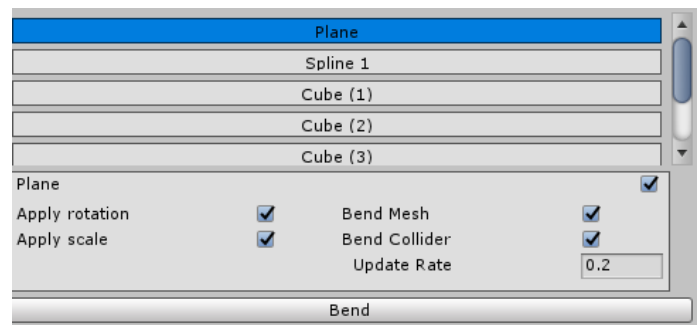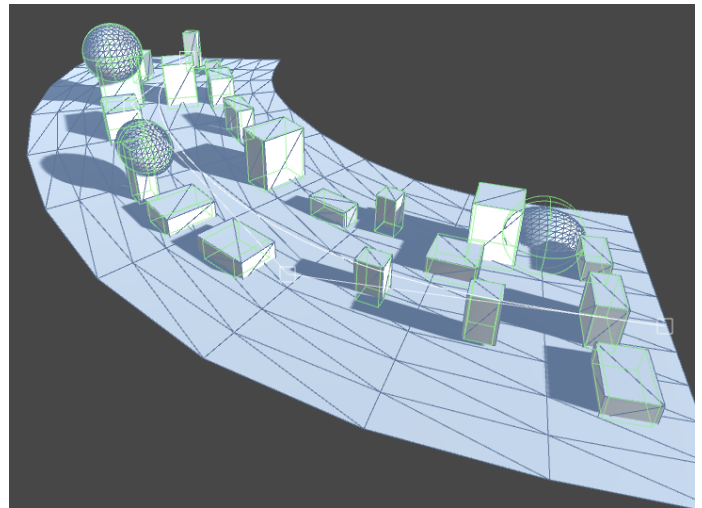
13. 2D Collision

Dreamteck Splines has means of generating 2D colliders.

13.1. Polygon Collider Generator

As the name suggests, this component generates a 2D Polygon Collider to be used with 2D physics. The collider is immediately updated on change in the editor and has an update interval property used to restrict the number of updates during runtime.

The Polygon Collider Generator has two modes:

- Path: creates a path along the spline, much like the Path Generator
- Surface: creates a solid shape, much like the surface generator
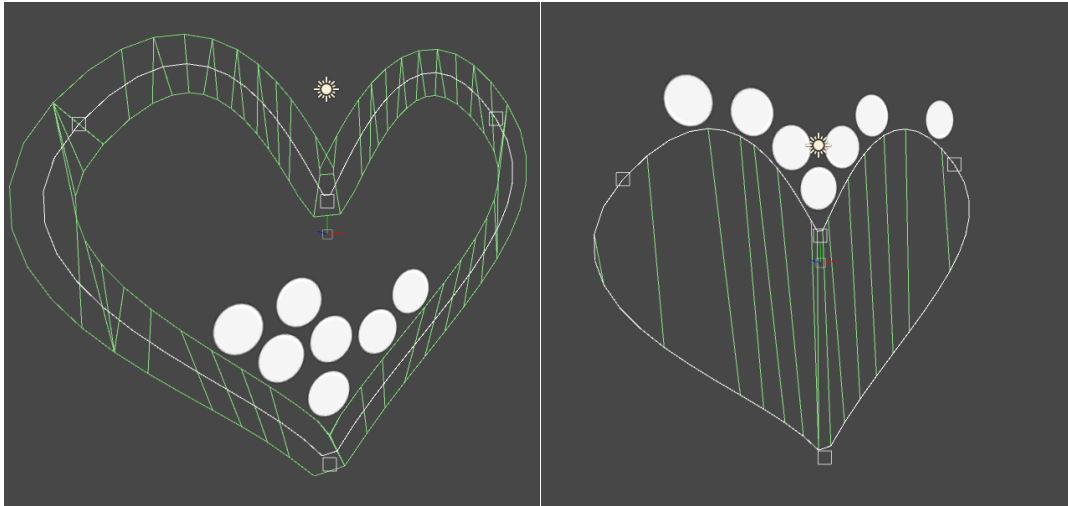
13.2. Edge Collider Generator

A very simple component that generates an edge collider. It has no custom settings.

Note: The edge collider will be generated at the position of the spline and therefore the spline rendering will cover the collider visualization making it seem like nothing is happening.

14.   Spline Triggers

The spline triggers are a way to call events when an object reaches a certain point along the spline. Spline Triggers are not physical collider triggers but rather a percent along the spline [0-1] which when reached or passed, will call an event.

Currently, triggers are supported only by the Spline Tracer components (Follower, Positioner and Projector) but trigger-checking functionality can be easily implemented by calling **SplineComputer.CheckTriggers** and providing the start percent along the spline and the end percent. If there are any triggers that fall between these two values, they will get invoked.

14.1. Creating Triggers

The trigger panel is located below the "Spline Computer" panel of the Spline Computer component. Expanding it reveals a single button, called "New Group". Clicking the button will create a new trigger group with index 0.

After a group is created, triggers can be added inside it by pressing the "Add Trigger" button.

When a trigger is created, it is added to the trigger list under the trigger group. Clicking on the trigger's name will open the trigger's panel:
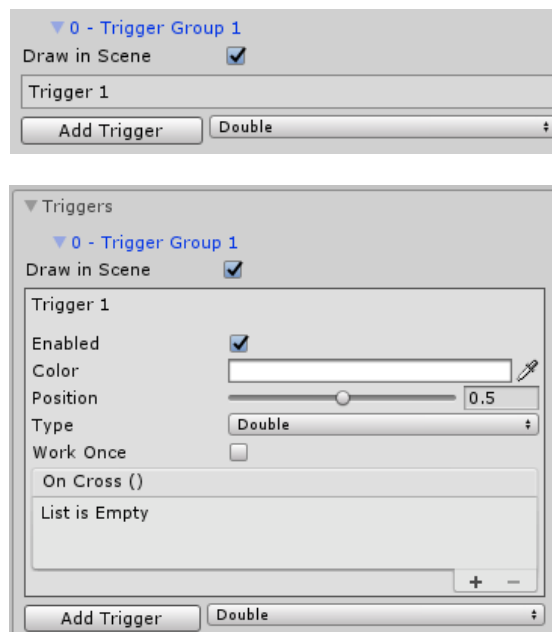
Inside the trigger panel, several properties can be set:

- Enabled – is the trigger active?
- Color – Color of the trigger for rendering it in the scene view
- Position – the position in percent [0-1] of the trigger along the spline
- Type – The direction the trigger works in: forward, backward or double (works in both directions).
- Work once – The trigger will work only the first time it is crossed

The On Cross event is the event that gets fired when the trigger is crossed.

43

14.2. Managing and Editing Triggers

When a trigger group is expanded in the inspector, its triggers will be drawn in the scene.
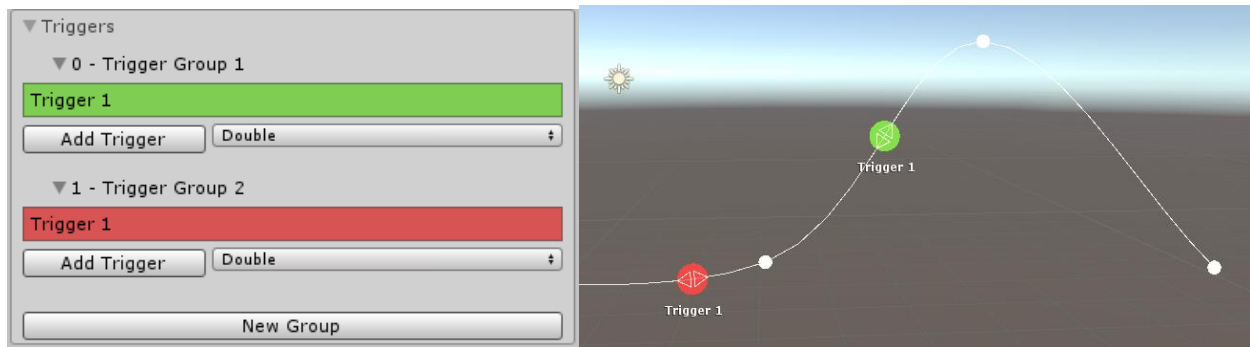
Dragging the trigger handles along the spline in the scene view will set their position.

Triggers and trigger groups can be renamed and deleted by right clicking on their names in the inspector and choosing the corresponding option.

14.3. Using Triggers

The Spline Tracer components like the Spline Follower for example, all have the "Use Triggers" option inside the inspector. When "Use Triggers" is checked, an additional field for the trigger group is presented. When the tracer moves along the spline, it will check all triggers from the given group and if a trigger is crossed, its event will be invoked.
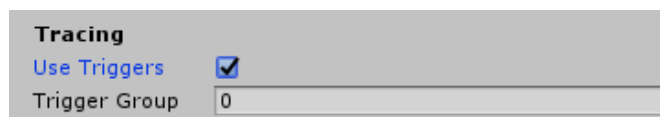
Currently, Spline Tracers can use only one trigger group but in future updates using multiple groups will be added.

14.3.1. Using Triggers from Code

To use the triggers from code, first a reference of the Spline Computer should be present:

```
SplineComputer spline = GetComponent<SplineComputer>();
```

From here, all triggers from all trigger groups can be checked like this:

```
spline.CheckTriggers(0.0, 0.5);
```

This will invoke all triggers which are in the range [0-0.5] and their type is set to either Forward or Double. To check triggers of type Backward, switch the position of the values:

```
spline.CheckTriggers(0.5, 0.0);
```

If there are triggers, set to work once, they can be reset by calling:

```
spline.ResetTriggers();
```

To check and reset triggers only from a given trigger group, the "triggerGroups" array needs to be used:

```
spline.triggerGroups[0].Check(0.0, 0.5);
spline.triggerGroups[0].Reset();
```

This will check and reset all triggers in group 0.

Individual triggers can also be changed through code by modifying the same properties, exposed in the inspector:

```
spline.triggerGroups[0].triggers[0].name = "Explosion";
spline.triggerGroups[0].triggers[0].position = 0.75;
```

15.   Sample Modifiers

Inside the Inspector of all Spline User component, there is a section for the so-called Sample Modifiers. These are operations that are performed on the spline samples before the given user uses them. This way, it is possible to add custom offsets and change the spline output per user.

All modifiers work in the same way – they just modify different values of the spline samples. When a modifier foldout is expanded, there are always these UI elements:
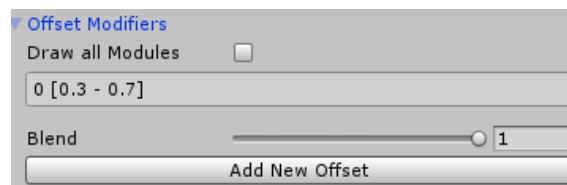
- Draw All Modules – will draw all modules from the given group in the scene view regardless of whether they are selected.
- Blend – a general multiplier that applies to all modules under the given group. Moving the value towards zero will decrease the effect of all modules in the group.
- Add New ### - The "Add New" button will add a new module inside the given group.

When a module is selected, it is drawn in the scene view and int can be edited there directly. The two bigger handles define the range where this modifier works. The smaller handles define the area of full strength (between the small handles).
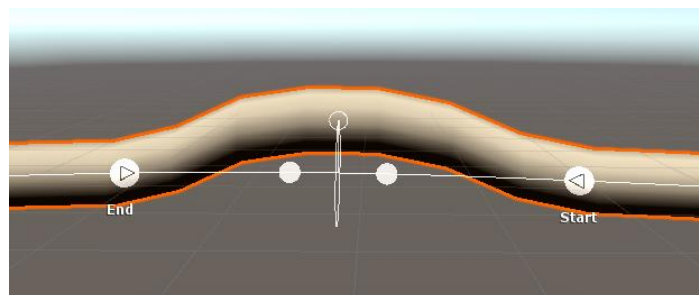
Pressing Ctrl will display an additional handle in the middle which when dragged, moved all handles together along the spline.

15.1. Offset Modifiers

The offset modifiers apply offset to the spline samples. **The offset is always local to the spline**.
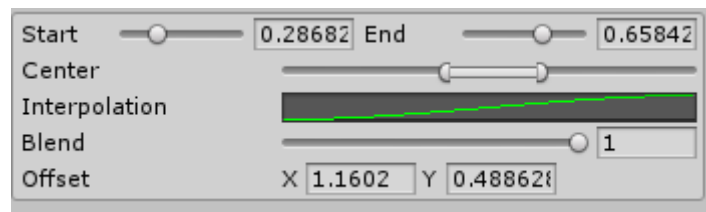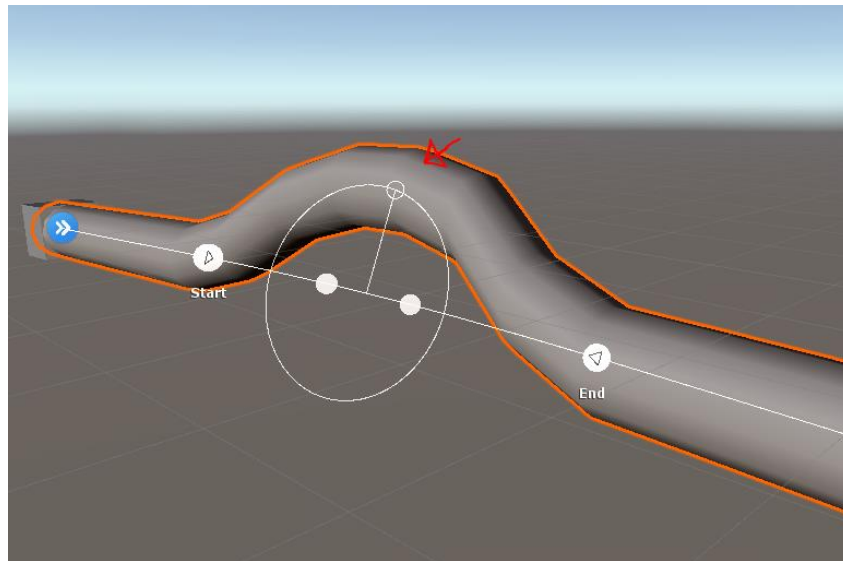
The offset is defined by either directly dragging the offset handle in the scene (the small circle) or through the inspector's offset property.

15.2. Rotation Modifiers

The rotation modifiers modify the direction and normal vectors of the spline samples so that different rotation can be computed. This works especially well with Spline Tracers:
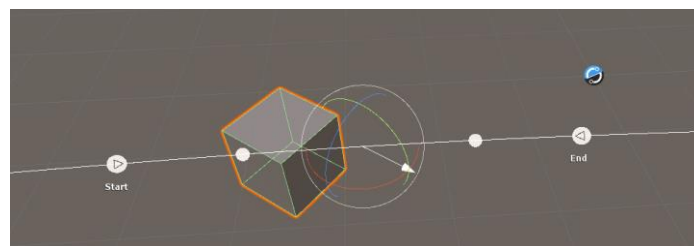
The rotation offset vector is local to the spline and can be edited either by using the rotation handle in the scene or by modifying the rotation field of the modifier inside the inspector.

The rotation modifiers can use a specific look target (a Transform in the scene) instead of a rotation offset property. When a look target is used, the rotation offset will be computed in such a way that the rotation will point towards the target.

46
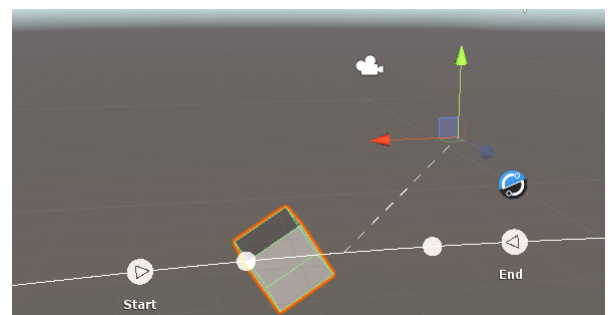
15.3. Color Modifiers

The color modifiers offset the color of the spline samples:
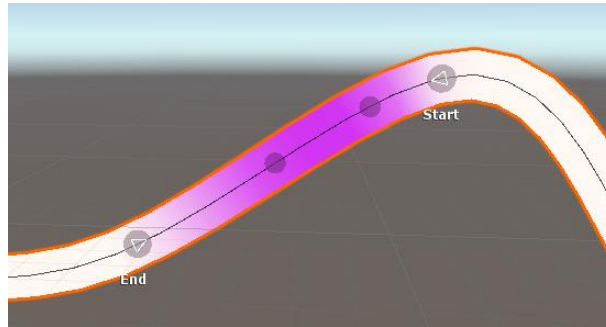
There are four blending modes that can be used for this operation:

- Lerp
- Multiply
- Add
- Subtract

15.4. Size Modifiers

The size modifiers change the size of the spline samples. If the value is positive, size will be added to the spline and if the value is negative, size will be subtracted from the spline samples.
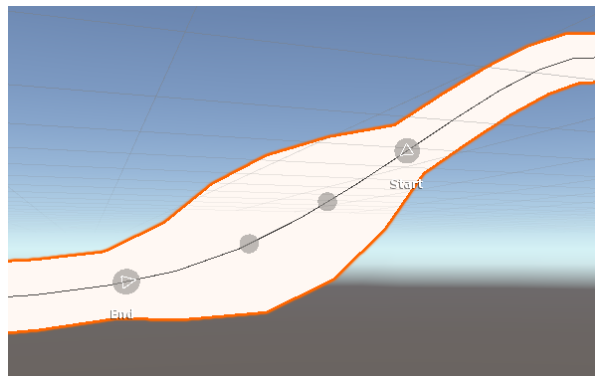
16.   Morphing Splines

Dreamteck Splines provides easy morphing functionality which allows different spline shapes to be blended into each other. In order for this to work, all versions of the spline should have exactly the same point count.

To use spline morphing, add the Spline Morph component to a Spline Computer object. Inside the inspector of the Spline Morph component, there is a "+" button with a text box next to it. Clicking the button will add the first morph channel.
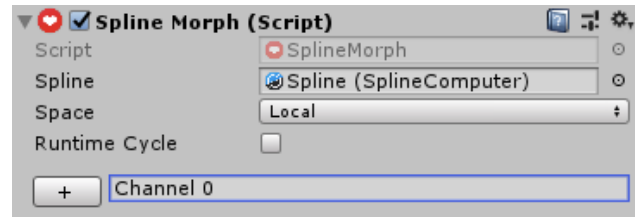
The first morph channel is the base channel and holds the original spline shape. Each channel that is added afterwards will have a blend slider which controls how much the shape in this channel is morphed towards.

In order to blend between two spline shapes, create a base channel.
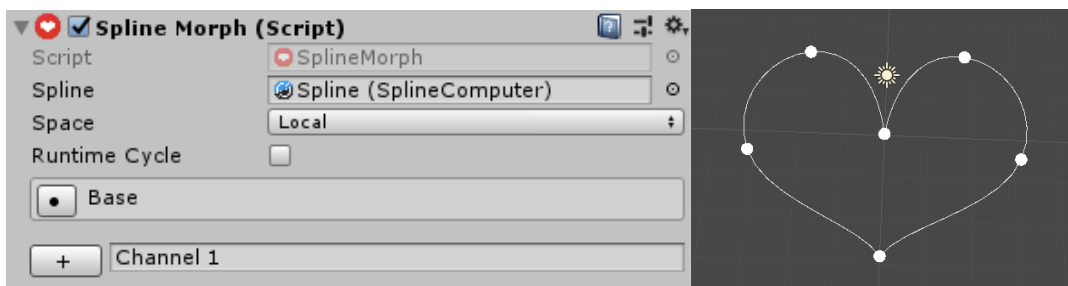
After this, re-arrange the points of the spline and click the Create channel button again. A new morph channel will be created that holds the new shape.
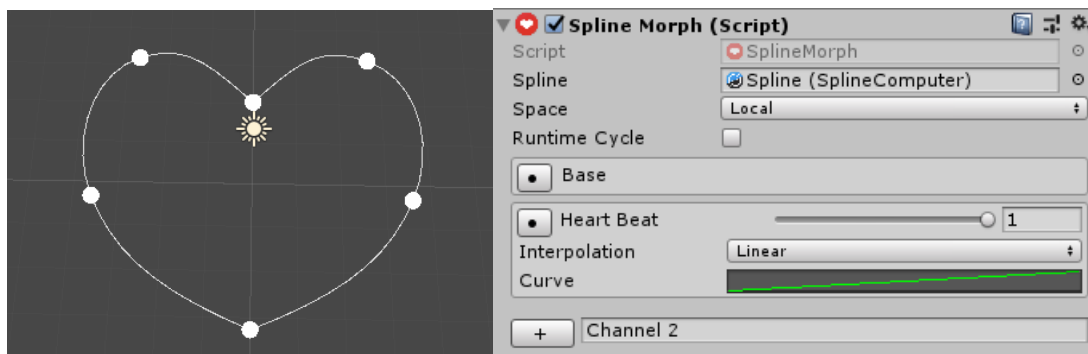
With that new channel created, the two shapes can now be blended between by pulling the slider of the second channel left and right.

Each channel can interpolate the spline points linearly or spherically – this is set using the Interpolation property. The curve property controls the blend interpolation rate.

To edit a channel's points, edit the spline points and click the channel record button:

16.1. Runtime Cycle

The morph component can automatically cycle through its states during runtime if "Runtime Cycle" is checked. Doing so will expose three additional parameters:

- Cycle wrap
- Update Mode
- Cycle Duration

Cycle wrap defines what happens when the morph cycle ends. Default mode will simply stop at the last channel. Loop will start again from the first channel and Ping Pong will play the morph cycle backwards.
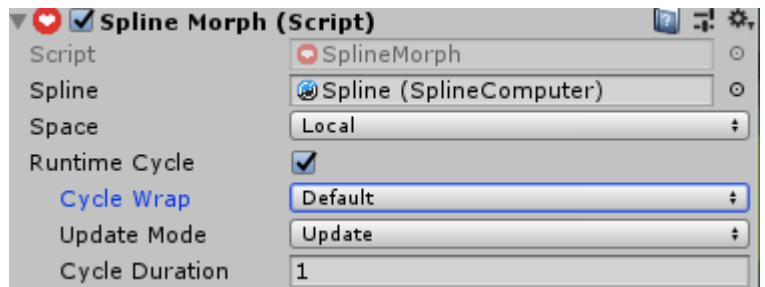
Update mode defines the update cycle in which the logic will be performed – Update, LateUpdate or FixedUpdate

Cycle duration is the time the cycle needs in order to run A to Z.

17.   Nodes

Nodes are a key feature and serve two very important purposes - to bind points of Spline Computers to other scene objects and to create junctions.
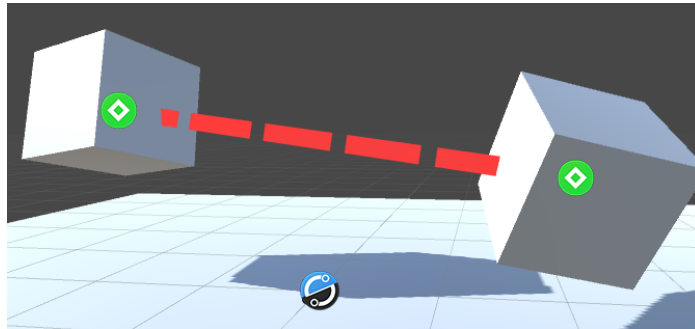
To create a Node, go to **GameObject/3D Object/Spline Node**. This will create a new Game Object with a Node component in the scene.

Another way to create a Node is just by adding a Node component to an existing object.

The newly created node will be empty and will require a connection to be added. To add a connection, drag a Game Object with a Spline Computer from the scene into the Connections box.
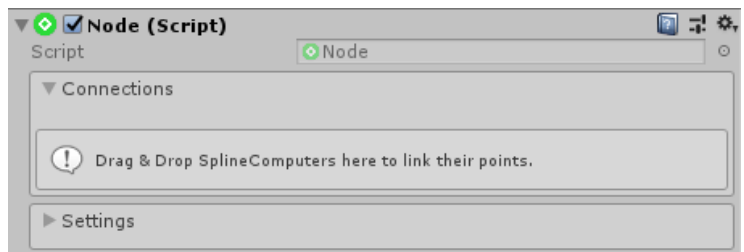
When a Spline Computer is dragged in, its points will visualize as circles and a point dropdown menu will be shown in the inspector.
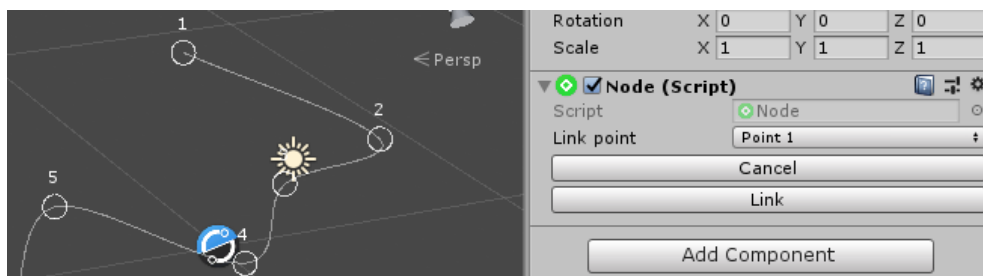
The Node will now require a point from the Spline Computer to be selected. To select a point either select it from the dropdown menu and click "Link" or click on a point in the scene view. If the point has no connections added, after the first point is selected, a dialog will appear asking what to do when connecting:
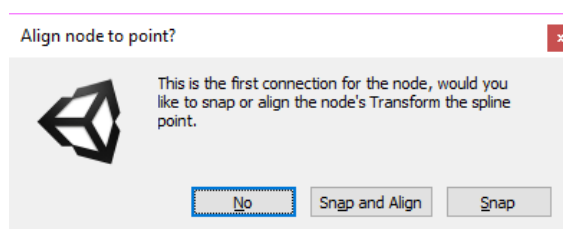
- Snap and Align – Will move the Node to the selected point and will rotate the node along the spline
- Snap – Will just move the node to the selected point's position
- No – Will move the selected point to the node

After the node has been connected, its Transform can then be animated or a Rigidbody can be attached and when the Node moves, the connected point will also be updated.

A Node can have more than one Spline Computer connected to it. Repeat the procedure with another Spline Computer to add a new connection.
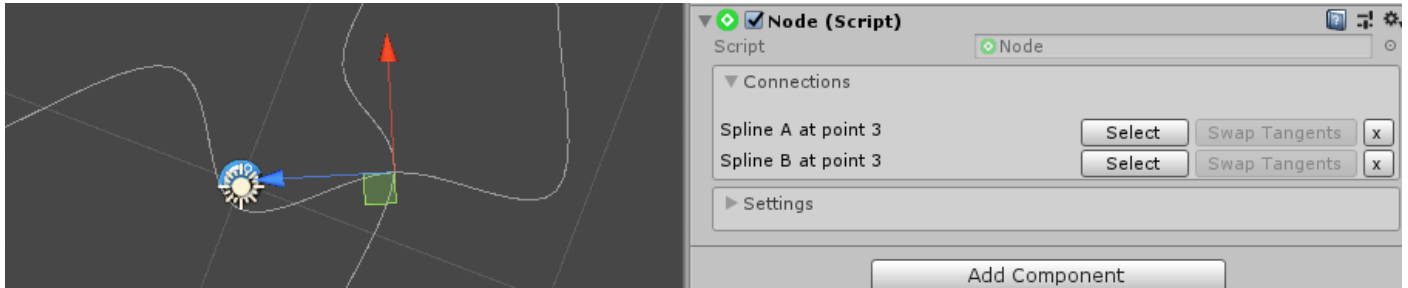
When editing a point that is connected to a Node in the Spline Editor, the node's position will also be updated and therefore all the other Spline Computers connected to it.

If the connected spline is of Bezier type, a "Swap Tangents" button will be available next to the spline's entry in the connections list. Pressing it will change the direction of the connected point's tangents.
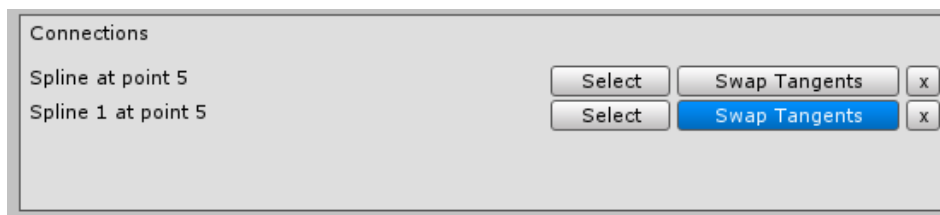
To remove a spline connection, click on the "x" button inside the connections list panel.

Clicking the "Select" button will select the connected spline's Game Object in the scene.

17.1. Node Settings

Expanding the Settings panel inside the Node's inspector will reveal additional options that apply for the entire node:
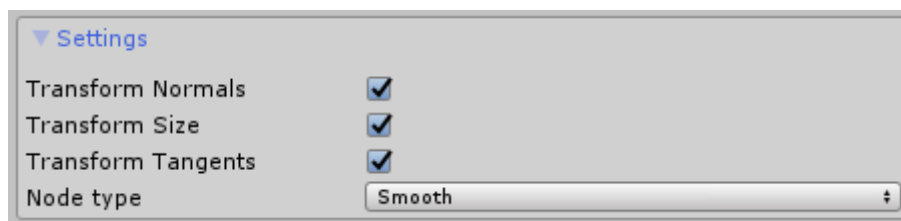
- Transform Normals – Should point normals be transformed when the Node rotates?
- Transform Size – should point sizes be transformed when the Node is scaled?
- Transform Tangents – should Bezier tangents be transformed by the Node's rotation?
- Node Type – Smooth will apply the same point properties to all connected points (size, color, normals, tangents) so that all points, connected to the Node are the same. Free will preserve the points' properties and only sync position.

17.2. Junctions

When two Spline Computers are connected to the same Node, this is considered a junction since getting the connection list of a Node provides all the information, needed to know in order to switch to another spline.
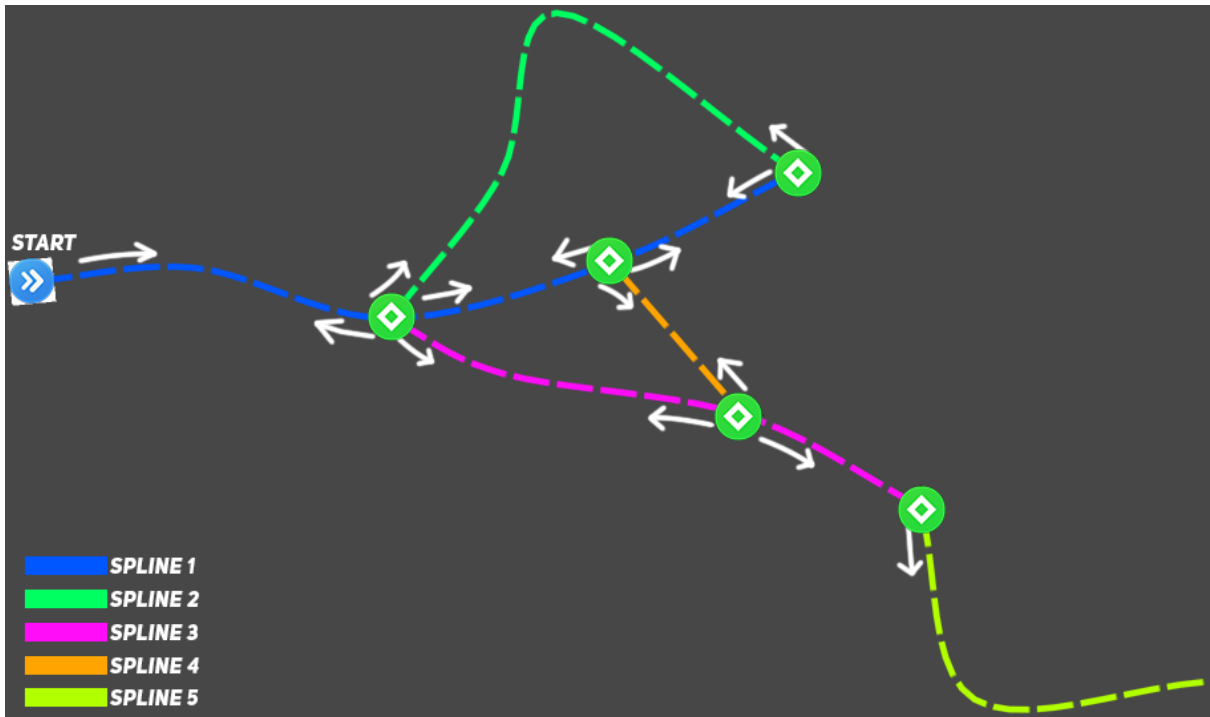
Junctions are used entirely through code. The Spline Tracer components provide a convenient API for handling junctions.

This example script can be attached to either a Spline Positioner, Follower or a Projector and will work for bopth of them:

```
using System.Collections.Generic;
using UnityEngine;
using Dreamteck.Splines;

public class ExampleJunctionHandler : MonoBehaviour
{
    SplineTracer tracer;

    private void Awake()
    {
        tracer = GetComponent<SplineTracer>();
    }

    private void OnEnable()
    {
        tracer.onNode += OnNode; //onNode is called every time the tracer passes by a Node
    }

    private void OnDisable()
    {
        tracer.onNode -= OnNode;
    }
```

```csharp
    private void OnNode(List<SplineTracer.NodeConnection> passed)
    {
        //plineTracer.NodeConnection holds information about the point index and provides a
direct reference to the Node
        //a list of Node connections is passed in case the tracer has moved over several points
in the last frame
        //this happens very rarely in particular cases where points are very close to each other
and the tracer moves very fast
        Debug.Log("Reached node " + passed[0].node.name + " connected at point " +
passed[0].point);
        //Get all available connected splines
        Node.Connection[] connections = passed[0].node.GetConnections();
        //If this node does not have other connected splines, skip everything - there is no
junction
        if (connections.Length == 1) return;
        //get the connected splines and find the index of the tracer's current spline
        int currentConnection = 0;
        for (int i = 0; i < connections.Length; i++)
        {
            if (connections[i].spline == tracer.spline && connections[i].pointIndex ==
passed[0].point)
            {
                currentConnection = i;
                break;
            }
        }
        //Choose a random connection to use that is not the current one
        //This part can be replaced with any other Junction-picking logic (see TrainEngine.cs in
Examples)
        int newConnection = Random.Range(0, connections.Length);
        //If the random index corrensponds to the current connection, change it so that it
points to another connection
        if(newConnection == currentConnection)
        {
            newConnection++;
            if (newConnection >= connections.Length) newConnection = 0;
        }
        //A good method to use which takes into account spline directions and travel distances
        //and adds compensation so that no twitching occurs
        SwitchSpline(connections[currentConnection], connections[newConnection]);
    }

    void SwitchSpline(Node.Connection from, Node.Connection to)
    {
        //See how much units we have travelled past that Node in the last frame
        float excessDistance =
tracer.spline.CalculateLength(tracer.spline.GetPointPercent(from.pointIndex),
tracer.UnclipPercent(tracer.result.percent));
        //Set the spline to the tracer
        tracer.spline = to.spline;
        tracer.RebuildImmediate();
        //Get the location of the junction point in percent along the new spline
        double startpercent = tracer.ClipPercent(to.spline.GetPointPercent(to.pointIndex));
        if (Vector3.Dot(from.spline.Evaluate(from.pointIndex).direction,
to.spline.Evaluate(to.pointIndex).direction) < 0f)
        {
            if (tracer.direction == Spline.Direction.Forward) tracer.direction =
Spline.Direction.Backward;
            else tracer.direction = Spline.Direction.Forward;
        }
        //Position the tracer at the new location and travel excessDistance along the new spline
        tracer.SetPercent(tracer.Travel(startpercent, excessDistance, tracer.direction));
```

```
    }
}
```

If it seems very complicated, here are simplified versions of both the OnNode and SwitchSpline methods:

```
private void OnNode (List<SplineTracer.NodeConnection> passed)
    {
        Debug.Log("Reached node " + passed[0].node.name + " connected at point " +
passed[0].point);
        Node.Connection[] connections = passed[0].node.GetConnections();
        if (connections.Length == 1) return;
        int newConnection = Random.Range(0, connections.Length);
        if (connections[newConnection].spline == tracer.spline &&
connections[newConnection].pointIndex == passed[0].point)
        {
            newConnection++;
            if (newConnection >= connections.Length) newConnection = 0;
        }
        SwitchSplineSimple(connections[newConnection]);
    }

    void SwitchSpline(Node.Connection to)
    {
        tracer.spline = to.spline;
        tracer.RebuildImmediate();
        double startpercent = tracer.ClipPercent(to.spline.GetPointPercent(to.pointIndex));
        tracer.SetPercent(startpercent);
    }
```

To summarize – what needs to happen inside OnNode is:

- We need to get the connections of the reached node
- We need to find a new, suitable connection which to use
- We then assign the spline from the new connection to the tracer and set the Tracer's position along the new spline to the position of the connected point (via spline.GetPointPercent tracer.SetPercent)

Junction picking can be aided with additional, custom components. For example, an additional component that is attached to the Node object which says which connections can be used. An example for this can be found inside the Train example scene. There, a component called "Junction Switch" is used to do exactly this and the logic for junction picking and switching can be found inside TrainEngine.cs where inside OnNode, a reference to the JunctionSwitch is used in order to pick the right connections.

18. Instantiating Splines

Spline Computers can safely be put into prefabs and instantiated during runtime. However, since they come from prefabs, instantiating a spline at a different position than the one that is written in the prefab will cause the spline to revert to its original prefab location.

To fix this issue, make sure that all Spline Computers inside any prefab are set to rebuild on awake.

19.   Editor Tools

The Spline User is a component that is mostly meant to work in runtime although it can be configured in the editor. Sometimes, however, level designers just need a tool to help them in the work process. For pure level design purposes, Dreamteck Splines offers the Spline Tools window which is found in **Window/Dreamteck/Splines/Tools.**

Each tool can be selected by clicking the corresponding button. Some tools require an object with a Spline Computer to be selected in the editor in order to work. Otherwise a warning will be displayed:
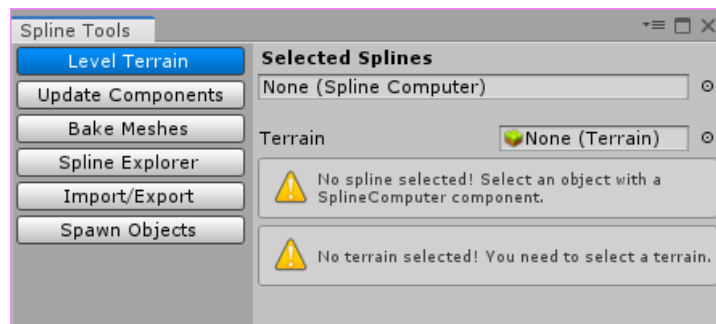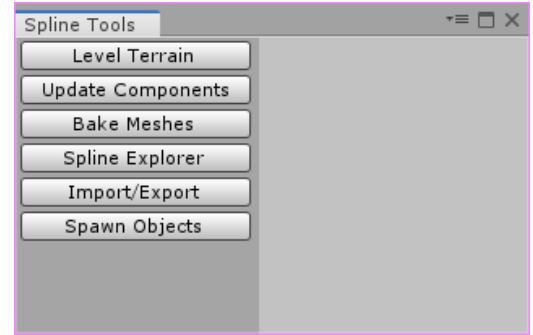
If this warning appears, go to the scene's hierarchy, select a Spline Computer and click on the tools window again. This will update it and the warning will disappear.

19.1. Mass Baking Spline-generated Meshes

If a lot of Mesh Generators need to be baked simultaneously, the Bake Meshes tool can help with that. It does not need a Spline Computer to be selected in order to work and can be opened right away.

There are three bake modes:

- All - Bakes all Mesh Generators in the scene
- Selected - Bakes only the selected Generators
- All Excluding - Bakes all Generators except for the selected ones

If "Selected" or "All Excluding" is selected, the Bake Tool will provide two panels – Available and Selected:

Inside the Available panel, a list of all Mesh Generators which are available to bake is displayed. Clicking the "+" button will add the given mesh generator to the selected panel. When "Bake" is clicked, all Mesh Generators inside the "Selected" panel will be baked or excluded from baking depending on the Bake Mode.
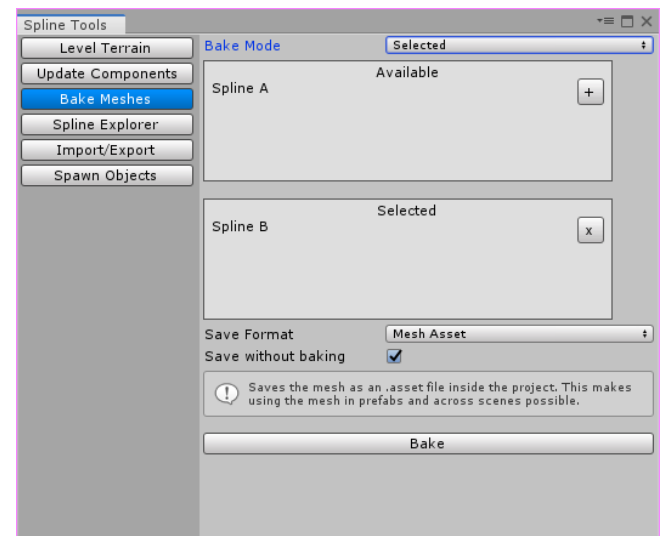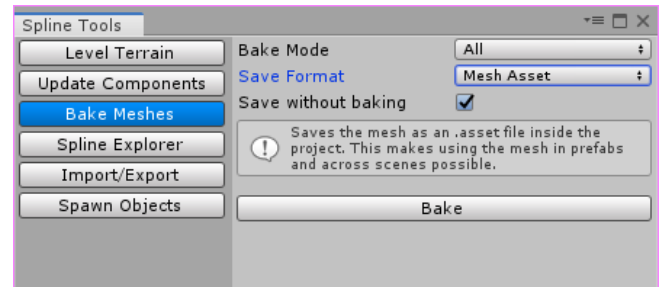
19.2. Leveling terrain

The Level Terrain tool is an editor-only tool meant for sculpting terrains using splines.

For the Level Terrain tool to work, the scene needs to have at least one terrain and a Spline Computer needs to be selected. If the scene has more than one terrain, the terrain that should be leveled should also be selected along with the Spline Computer.
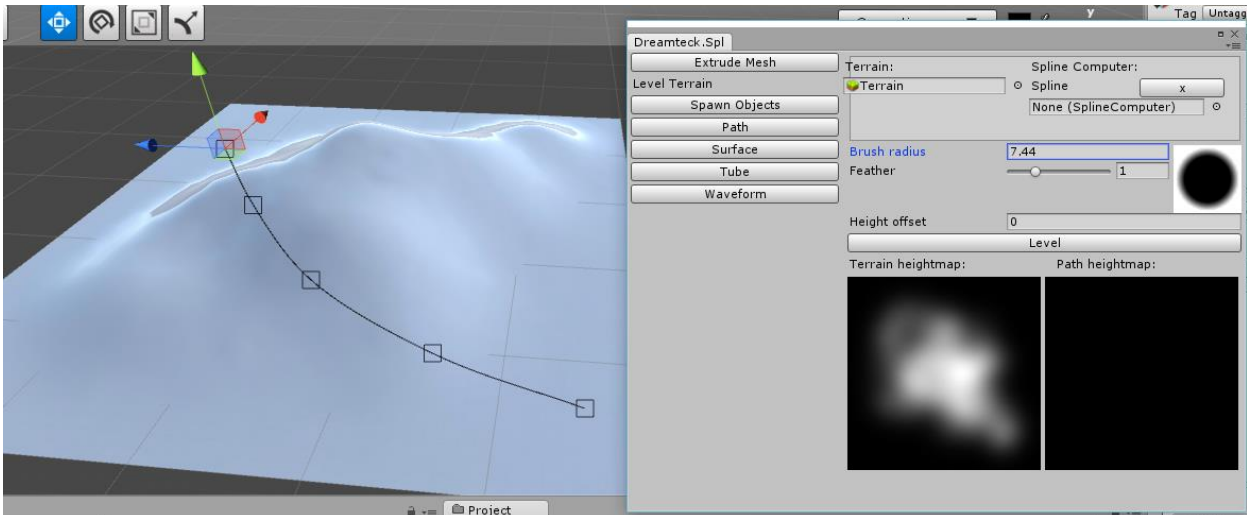
Once open, the Level Terrain tool will display a preview of the terrain's heightmap, the path heightmap which will be black and a preview of the brush feather.

The tool has only three settings:

- Brush Radius – the radius of the brush in world units
- Feather – the feather of the brush
- Height Offset – height offset

To level the terrain using the selected spline(s), click the "Level" button. The path heightmap will be drawn and the terrain will be sculpted.
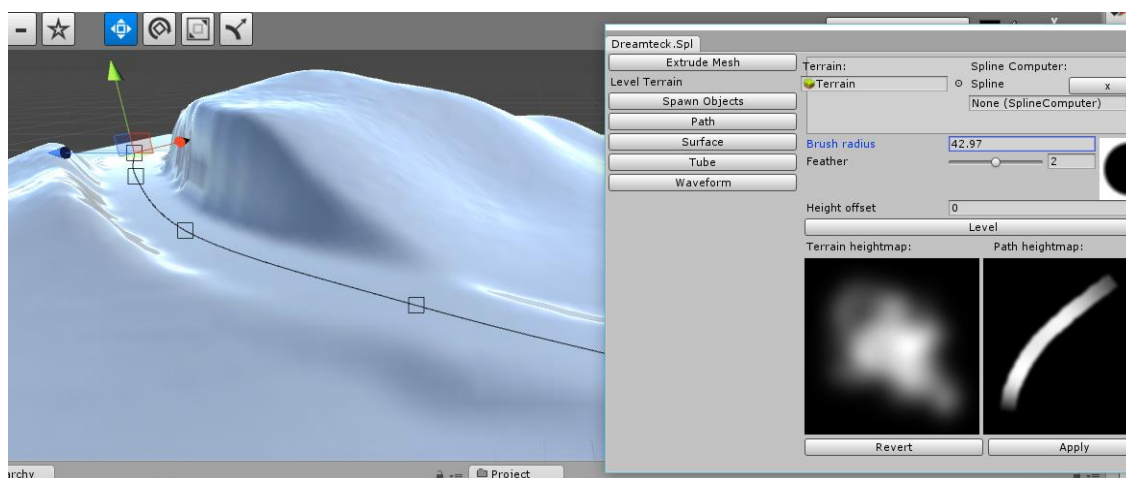
An "Apply" and "Revert" buttons will appear under the heightmap previews. If the result is not satisfying, edit the spline and the settings and click Level again – the previous level will get replaced with the new one. To save the terrain, click "Apply" – this will write the level data to the terrain permanently.

The brush radius is affected by the sizes of the points.

19.3. Import/Export

Splines can be imported and exported from and to other software using the Import/Export tool. The currently supported formats are SVG and CSV datasets.
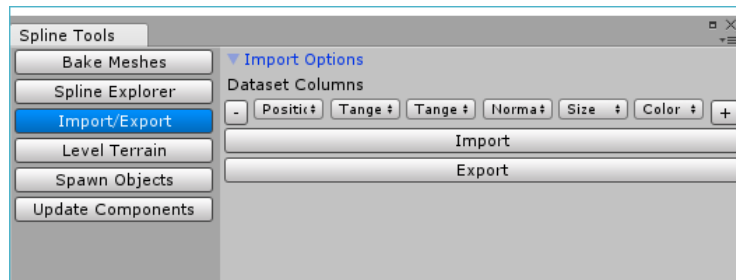
19.3.1. Importing

To import an SVG or a CSV file click the "Import" button. This will open a new file browser. Navigating to the desired SVG or CSV file and clicking "Open" will instantly import the splines from the file in the Unity scene.

NOTE: Splines from SVG files are usually very big, about the size of a Canvas or even bigger. If the imported SVG spline isn't visible after import, make sure to zoom-out further in the scene.

After the spline has been imported, a new object with the name of the file will be created in the scene and the import tool's panel will display some importing options:
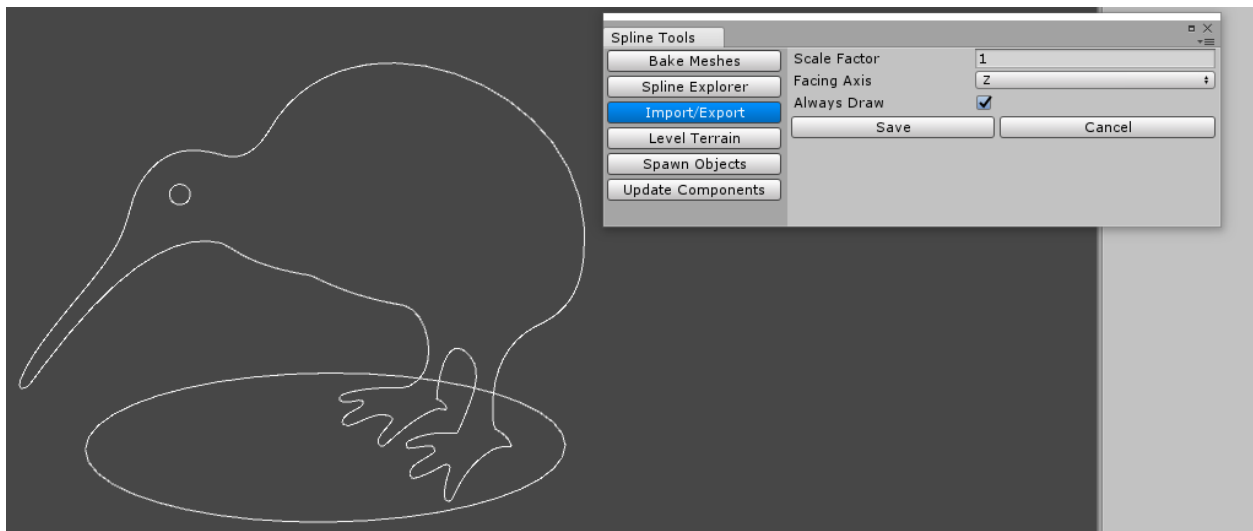
- Scale Factor – This is the scale factor of the imported graphic. A scale factor of 1 will retain the size of the graphic.
- Facing axis – The orientation of the imported graphic. By default it is Z which means no rotation
- Always draw – Should the splines be drawn in the scene constantly after closing the import tool?

19.3.2. Exporting

To export one or more splines, the splines first have to be selected in the scene. If no splines are selected in the scene, the export button will be unavailable.

When the export mode is toggled, a Format dropdown menu will be available, allowing the user to select between SVG and CSV.

19.3.2.1.   Exporting SVG

Since the SVG is a 2D vector format, when a spline is exported to SVG it is flattened.
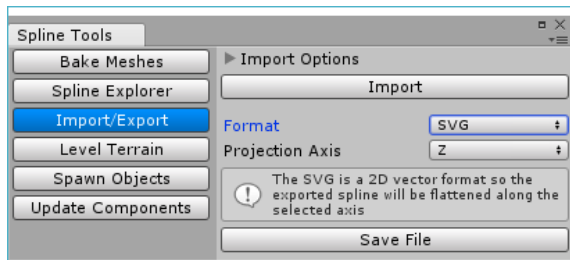
The projection axis dropdown menu defines the axis along which the spline will be flattened and projected onto the SVG canvas.

19.3.2.2.   Exporting CSV

The CSV export creates a new Excel file with each spline point's data written on each row.
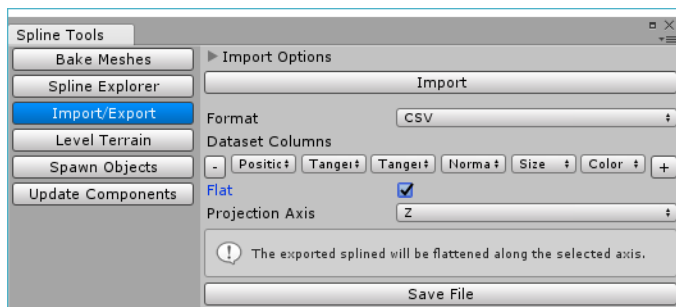
The Dataset columns field allows for customization of the columns that will be written in the CSV file. For example, if only the points' positions are needed, all columns except the position column should be removed.

CSV Export also supports flattening. If "Flat" is toggled, a projection axis dropdown menu is presented.

58

20.   Basic API Usage

To use the Dreamteck Splines API, add a "**using Dreamteck.Splines;**" to the start of the script. From there on, all components, classes and public methods/properties can be accessed.

20.1. Accessing Spline Components in Code

All Spline components derive from the **MonoBehaviour** class and therefore act as regular components which can be get using the GetComponent method.

Once a reference to the given component is obtained, all properties exposed in the editor are available in code. For example, here is how getting a reference to a SplineFollower and setting its follow speed looks like:

```
SplineFollower follower = GetComponent<SplineFollower>();
follower.followSpeed = 10f;
```

In order to change the spline used by the current user, set the user's "spline" property to the desired spline:

```
follower.spline = mySplineComputer; //Where mySplineComputer is an object of type SplineComputer
```

Setting the spline through code like that will immediately tell the user to move all of its functionality over to the new spline.

20.2. Creating and Editing Splines in Runtime

Splines are defined by a collection of control points so creating and altering them in runtime comes down to getting and setting points. There are a couple of methods for this:

- **GetPoint** – Gets a point by index
- **GetPoints** – Gets an array of all points in the spline
- **SetPoint** – Sets a point by index
- **SetPoints** – Sets the entire points array of the spline

Here is an example of creating a new spline from scratch:

```
//Add a Spline Computer component to this object
SplineComputer spline = gameObject.AddComponent<SplineComputer>();
//Create a new array of spline points
SplinePoint[] points = new SplinePoint[5];
//Set each point's properties
for (int i = 0; i < points.Length; i++)
{
    points[i] = new SplinePoint();
    points[i].position = Vector3.forward * i;
    points[i].normal = Vector3.up;
    points[i].size = 1f;
    points[i].color = Color.white;
}
//Write the points to the spline
spline.SetPoints(points);
```

This is an example of editing the points of an existing spline:

```
SplineComputer spline = gameObject.GetComponent<SplineComputer>();
//Get the spline's points
SplinePoint[] points = spline.GetPoints();
for (int i = 0; i < points.Length; i++)
{
    //Add random vertical offset to each point
    points[i].position += Vector3.up * Random.Range(-1f, 1f);
}
        spline.SetPoints(points);
```

And this is an example of removing a point:

```
int pointToRemove = 2;
SplineComputer spline = gameObject.GetComponent<SplineComputer>();
SplinePoint[] points = spline.GetPoints();
SplinePoint[] newPoints = new SplinePoint[points.Length - 1];
for (int i = 0; i < points.Length; i++)
{
    if (i < pointToRemove) newPoints[i] = points[i];
    else if (i > pointToRemove) newPoints[i - 1] = points[i];
}
spline.SetPoints(newPoints);
```

20.3. Evaluating Splines

There are two methods for evaluating a spline at a percent – **Evaluate** and **EvaluatePosition**. EvaluatePosition is lightweight but it only returns a Vector3 position along the spline while Evaluate returns all values that the spline can provide and writes them to a special object called a **SplineSample.**

For EvaluatePosition, the usage is as simple as calling the method:

```
SplineComputer spline = gameObject.GetComponent<SplineComputer>();
Vector3 position = spline.EvaluatePosition(0.5);
```

Note that, 0.5 does not necessarily mean that the position will be physically in the middle of the spline, especially if the spline's sample mode is not set to Uniform. Refer to Sample Mode to get a better understanding on how percentages translate to world positions. 6

Evaluate requires a SplineSample object to be created and passed to it in order to work. It is usually a good practice to define the SplineSample in the initial variables and re-use in order to reduce GC. However, in this example, the SplineSample is defined just before Evaluate is called so that it can work when copied.

```
SplineComputer spline = gameObject.GetComponent<SplineComputer>();
SplineSample sample = new SplineSample();
spline.Evaluate(0.5, sample);
Debug.DrawRay(sample.position, sample.forward, Color.blue, 10f);
Debug.DrawRay(sample.position, sample.up, Color.green, 10f);
Debug.DrawRay(sample.position, sample.right, Color.red, 10f);
```

This example draws the spline result with its forward, up and right vectors.

Note that the Evaluate methods use parameters of type double and not float so when hardcoding percentages, do not add the "f" behind the number. For example **spline.Evaluate(0.5f, sample)** is wrong.

Both methods also have an override where instead of a double percent (0.5 for example), an integer value defining a point index can be provided and this will return the evaluation at the position of that control

point. For example `spline.EvaluatePosition(0); spline.EvaluatePosition(1);`
`spline.EvaluatePosition(2);` Will return the positions of points 1, 2 and 3.

20.4. Converting World Units to Spline Percentages

It is possible to find the length of the whole spline or a certain region in world units by using the CalculateLength method. The method uses two parameters, from and to, which are set by default to 0 and 1.

```
Debug.Log(spline.CalculateLength()); //Will return the entire spline's length
Debug.Log(spline.CalculateLength(0.25, 0.75)); //Will return the length of the given region
```

To find the percent of a spline that corresponds to a certain distance in world units along it, the **Travel** method can be used.

```
double distancePercentFromStart = spline.Travel(0.0, 15f, Spline.Direction.Forward);
double distancePercentFromEnd = spline.Travel(1.0, 15f, Spline.Direction.Backward);
```

The first argument is the start percent, the second holds the distance in world units which will be travelled and the last argument is a direction – forward or backward since Travel can work in both directions.

With all that said, here is an example of how to find the true percent that lies in the physical middle of the spline:

```
SplineComputer spline = gameObject.GetComponent<SplineComputer>();
float splineLength = spline.CalculateLength();
double travel = spline.Travel(0.0, splineLength / 2f, Spline.Direction.Forward);
Vector3 middle = spline.EvaluatePosition(travel);
Debug.DrawRay(middle, Vector3.up, Color.red, 10f);
```

20.5. Getting Output from Spline Tracers

The Spine Tracer components cave a public property called "result" and it can be used to get information about their current location along the spline. The result property is a read-only SplineSample object that automatically gets updated by all tracers.

For example, in order to get the percentage of a Spline Follower along a spline, it can be done like this:

```
SplineFollower follower = GetComponent<SplineFollower>();
Debug.Log(follower.result.percent);
```

And this works for the Spline Positioner and Spline Projector as well.

20.6. Writing a custom SplineUser class

Dreamteck Splines' functionality can be expanded easily by creating new SplineUser components. For a SplineUser class to be created it needs to derive from the SplineUser base class.

An empty SplineUser example is provided in the Components folder. It can be used as a template for creating new custom ones.

20.6.1. Protected and virtual Methods

Each SplineUser class inherits a set of methods which are automatically called. These methods are automatically called and should be overridden in order to create custom functionality.

SplineUser classes are not supposed to have Update, FixedUpdate and LateUpdate methods. They can have a Start method and an Awake method through overriding however:

```
protected override void Awake()
{
base.Awake();
//Code here
}
```

In order to access the generated spline samples inside the custom SplineUser class the protected property "sampleCount" should be used to determine the count of the spline samples and then the GetSample(int index) method to get the sample at any given index. Note that these are the samples within the clip range of the Spline User and changing the clipFrom or clipTo properties will also change the sampleCount and the returned samples by GetSample.

The **Build** method can either run on the main thread or if Multithreading is enabled, it can run on a worker thread and therefore it is important to only have thread-safe code inside. For example, if the spline user generates a mesh, all vertex calculations should happen inside the Build method, using sampleCount and GetSample(), and all mesh operations like setting the vertices for the mesh should happen in **PostBuild**.

Here is an example of a simple custom SplineUser which draws all spline samples within its clip range:

```
public class DebugUser : SplineUser
{
    bool isPlaying = false;

    protected override void Awake()
    {
        base.Awake();
        isPlaying = true;
    }

    protected override void Build()
    {
        if (isPlaying) return; //Cannot use Application.isPlaying as it is not thread-
safe
        //Should run only in edit mode as Debug.DrawRay is not thread-safe
        for (int i = 0; i < sampleCount; i++)
        {
            SplineSample sample = GetSample(i);
            Debug.DrawRay(sample.position, sample.up * sample.size, sample.color);
        }
    }
}
```

Look at how existing SplineUser classes are implemented in order to get a better understanding of how SplineUser inheritance works.

20.6.2. Protected virtual void Run()

Run is called automatically on Update, FixedUpdate or LateUpdate based on the Update Method setting of the SplineUser. It's mainly used for game logic that will run on each update cycle.

20.6.3. Protected virtual void Build()

Build is called when there has been a change in the spline user. For example if the spline has changed or a property of the SplineUser has been set. Build can either be called from the main thread or from a separate one if multithreading is enabled, therefore it is advised that only thread-safe code is put inside. Make calculations inside Build and then apply them in PostBuild.

20.6.4. Protected virtual void PostBuild()

PostBuild is always called on the main thread after Build has finished even if multithreading is enabled. It's used to apply the results, calculated in Build(). For example, if the custom SplineUser will position objects along path. The objects' positions should be calculated in Build and then applied to the objects' transforms in PostBuild().

20.6.5. Rebuilding

The SplineUser class has the Rebuild(bool sampleComputer) method which forces the user to recalculate on the next update cycle.

SplineUsers rebuild automatically when their spline is modified or their properties are changed. That however does not happen when the properties of custom SplineUser classes are modified. In order to make the SplineUser class rebuild, custom setters should be implemented that call the Rebuild method.

Example:
```
private bool _state = false;
public bool state
{
    get { return _state; }
    set
    {
        if (value != _state) //Only rebuild if the value is different
        {
            _state = value;
            Rebuild(); //Rebuild but don't resample the spline
        }
    }
}
```

Rebuild is a public method so it can also be called from another object. Rebuilding happens once per update cycle no matter how many times it's called prior to that.

20.6.6. RebuildImmediate

Rebuild waits for the next update cycle so calling it will not yield any changes immediately. In some cases, rebuilding should happen instantly and that's what **RebuildImmediate** is for. It calls **t**he rebuild sequence regardless of update cycles and will yield results immediately.

Calling **RebuildImmediately** multiple times per update cycle will cause the SplineUser to rebuild as many times as RebuildImmediately is called.

21.   Performance, Oprimization and Under the Hood

Dreamteck Splines is optimized for speed and runs smoothly on all mobile devices, including older ones. This chapter will go over the main concepts behind the system's design and shed some light on how the main components – the Spline Computer and the Spline User work.

21.1. What impacts performance and what doesn't?

Dreamteck Splines uses caching in order to minimize calculations. When a spline is created, it is being immediately cached based on its sample rate setting. The bigger the spline and the bigger the sample rate, the more samples will be generated.

After the initial caching calculation, no other calculations will be performed unless the spline is changed.

What counts as a spline change?

- Moving spline points
- Adding / Removing spline points
- Changing spline settings like sample rate, spline type and space
- Moving the spline's transform when the Space is set to Local

The computational complexity and scale depend greatly on the spline settings. For example, if the Sample Mode of the spline is set to Default, moving a single spline point through code or inside the editor will only update the spline's samples around that point and will not recalculate the entire spline. However, if the Sample Mode is set to Uniform or Optimized, the entire spline will be re-calculated.

For splines, set to Local space, an additional Transform operation is run after the spline samples are calculated. If the Spline Computer's Transform is changed, the Transform operation will run on all samples but the spline calculation will not be performed. This is different from the legacy Dreamteck Splines (version 1.1) where both the sample calculation and the transform are performed regardless of what is happening.

Having many static Spline Computers in the scene costs nothing but memory since nothing is updated until it needs to update. Even if a lot of Spline Users are attached to the splines, no spline calculation operations will be performed as long as the spline objects are not changed.

21.2. How Spline Users Work?

When a Spline User is added to a Spline, it runs its own logic using the already cached spline samples. This means that no spline calculations are performed.

Similar to the Spline Computer, most Spline Users do not perform operations when not needed. For example, all Mesh Generators with the exception of the Spline Renderer, perform the mesh generation operation only when the spline changes or some property of the generators themselves is changed.

For the Spline Follower – the follow logic runs every frame on the cached spline and so many followers can be added to the same scene without performance drop. On mobile this number varies between 50 and 200 running simultaneously. However, if that many objects need to follow a single spline, consider using an Object Controller instead as it is meant exactly for this – to position lots of object on a spline and incrementing the "Evaluate Offset" property could give a similar result.

The Particle Controller runs every frame and for every particle but it is still pretty speedy and suitable for mobile.

Probably the most taxing Spline User is the Spline Renderer component as it runs once for each camera, every frame. If the game has many cameras running simultaneously or there are many/big splines in the scene, using a Spline Renderer, the framerate could drop fast. Always consider using the Path Generator component first before jumping onto the Spline Renderer, especially for 2D games. 2D games cannot benefit

from the Spline Renderer at all – there will not be any visual difference between using a Spline Renderer and a Path Generator in a 2D game.

21.2.1. What about the sample modifiers?

If a Spline User has sample modifiers, they are calculated each time the Spline User is updated so they do add a small overhead. However, each modifier has a range in which it operates and so calculations are not done outside this range.

21.3. Does having many SplineComputer components in the scene hurt performance?

As mentioned above, the Spline Computer by itself doesn't re-calculate the spline so there will be no Spline calculations. However, in order to ensure that the splines will be updated properly when the Spline Computer's transform changes, there will be a couple of if-checks during update. This means that having several thousand SplineComputers in the scene will indeed take up some performance.How to optimize if a thousand SplineComputers are present?

The best way to optimize would be to disable all Spline Computer components. This will not disable the splines but will disable the checks every frame. Then whenever a spline needs updating, either enable while its needed or call ResampleTransform() on it.

21.4. What happens when a scene, full of SplineComputers and SplineUsers loads?

Spline Computers and Users cache their values into the memory and everything gets serialized into the scene. By default, when a scene loads, all values will be de-serialized and no calculations will be performed.

However, if a Spline Computer is set to rebuild on Awake (by toggling "Rebuild On Awake "in the inspector), as soon as the Spline becomes active, it will be re-calculated and all connected Spline Users will run their update logic.

21.5. Common Methods

In Dreamteck Splines, a couple of methods with the same name can be found in several classes. These methods are:

- **Evaluate** – Evaluate the spline at a given percent [0-1] and return an entire object with values like position, direction, color, size, etc.
- **EvaluatePosition** – Evaluate the spline at a given percent [0-1] and return just a Vector3 object for the position
- **Travel** – Move along a spline in world units starting at a given percent. The returned result is a new percent that can be used to Evaluate the spline.
- **CalculateLength** – Calculates the distance in world units along the spline between two percentages
- **Project** – Returns the spline percent [0-1] that is closest to a given point in world space

They al can be found in the Spline class, as well as in SplineComputer, SplineUser and SampleCollection.

"So what is the difference between all of them?"

21.5.1. Methods in Spline

The Spline class represents a spline in world coordinates. It isn't transformed by any transform and it isn't used directly by any Spline User component. Instead, it is wrapped by the SplineComputer class.

21.5.2. Methods in SampleCollection

The SampleCollection is an object that holds a spline cache in the form of a list of samples. This class is used by the SplineComputer in order to store the cached samples.

When calling the methods on a SampleCollection object, they get performed on the cached samples and no spline calculation operations are run.

21.5.3. Methods in SplineComputer

Since the SplineComputer uses a SampleCollection object for its samples, calling one of these methods on a SplineComputer translates to calling the methods on a SampleCollection object – the methods are run on the cached samples.

However, there is more to that. The SplineComputer versions of the methods all have an additional, optional parameter called "mode". If the mode parameter is passed as SplineComputer.EvaluateMode.Calculate, then the spline will be calculated and the spline result will be transformed on the spot. This costs more performance but if pinpoint accuracy is needed, it can be very useful.

21.5.4. Methods in SplineUser

Calling the methods on a SplineUser component results in calling them on the SplineComputer. The catch here is that the percentages are automatically mapped to the SplineUser's clip range (clip from – clip to).

For example, if the clip range is set to [0.0 – 0.5], calling **EvaluatePosition(1.0)** on the Spline User will return the position at 0.5.

This also applies for the returned percent for **Travel** and **Project**.

22.　　　Frequently Asked Questions

This is a list of questions that our studio gets asked on a daily basis. We highly suggest checking these out to new users as this might save some time in waiting for a response from our support.

22.1.　Can I create and edit splines in runtime with code?

Yes! In fact, everything that is possible in the editor, can be done with code as well since the Editor uses the same API that is exposed during runtime. Examples of creating splines with code can be found in the Basic API Usage section.

22.2.　How to distribute objects evenly along a spline using the Object Controller?

The default behavior of all splines is such that they stretch and squash when control points are further apart or closer together (refer to Sample Mode for a better explanation). To mitigate this effect, either make sure to place all control points at even distances (see Distribute Evenly) or use a Uniform sample mode for the Spline Computer.

22.3.　How to find the middle of the spline?

Calling **Evaluate** with a percent of 0.5 will not work unless all control points are distributed evenly. In order to find the middle of the spline, use **CalculateLength** and **Travel**. Example here.

22.4.　Can I use Dreamteck Splines to create an endless runner game?

Yes, in fact, a lot of endless runner games have already been made using Dreamteck Splines. However, creating an endless runner game with this package will require some additional coding and this is why we have released Forever – it uses Dreamteck Splines as a base to create endless runner games effortlessly.

22.5.　How to move along a spline in world units instead of percent?

Incrementing a percent with another percent will not move along the spline at even distances and this is why the Travel method should be used in such cases. Example here.

22.6.　I'm using the Spline Mesh component and the extruded mesh does not perfectly follow the spline or is too edgy. What is going on?

The Spline Mesh component extrudes the provided meshes along the spline without creating additional geometry. This means that if you are trying to extrude a single cube mesh along a spline, the start and end vertices will simply stretch at the both ends of the spline because there are no vertices in the middle which can be positioned along the spline:
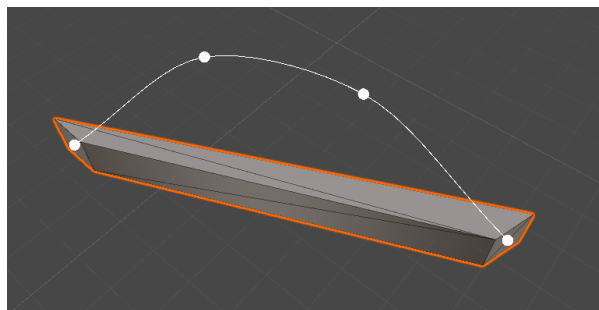
No matter how smooth the spline is, if the mesh does not have enough detail, it will be jaggy.

To fix this, either increase the repetition count of the mesh in the channel or if this isn't an option, add more vertices in between the two ends of the mesh in the modeling program.

22.7. I'm using a mesh generator to generate a big mesh but the mesh does not generate. What is going on?

Unity has a limit of 64,000 vertices per mesh object. If your mesh exceeds this limit, it cannot exist and therefore our mesh generators will stop updating the geometry.

In such cases, consider splitting the geometry into two separate objects. Create two game objects and add the same Mesh generator component to both. Have both of them use the same spline and set the clip range of the first Mesh Generator to [0-0.5] and the clip range of the second Mesh Generator to [0.5-1].

22.8. How can I change the speed of the Spline Follower component during runtime?

```
SplineFollower follower = GetComponent<SplineFollower>();
follower.followSpeed = 10f;
```

22.9. No matter where I instantiate a Spline Computer, the spline appears always in the same place. What is happening?

Double check if the Spline Computer's space is not set to World instead of Local and make sure the Spline Computer is set to rebuild on awake from the Spline Computer panel.

23.   API Reference

This chapter lists the main classes in Dreamteck Splines along with their public and protected properties and methods.

23.1. SplinePoint

A spline control point used by the Spline class.

23.1.1. Enums

Type { SmoothMirrored, Broken, SmoothFree };	• **SmoothMirrored – Mirrors both tangents** • **Broken – Each tangent moves independently** • **SmoothFree – Tangent direction is mirrored but length is independent**

23.1.2. Public Properties

SplinePoint.Type type	The type of the spline point
Vector3 position	The position of the point
Vector3 normal	The normal of the point
Vector3 tangent	The first tangent of the point
Vector3 tangent2	The second tangent of the point
Color color	The color of the point
float size	The size of the point

23.1.3. Public Methods

void SetPosition(Vector3 pos)	Set the point's position (updates the tangents automatically)
void SetTangentPosition(Vector3 pos)	Set the position of the first tangent (updates the other tangent automatically based on the point type)
void SetTangent2Position(Vector3 pos)	Set the position of the second tangent (updates the other tangent automatically based on the point type)

23.1.4. Static Methods

SplinePoint Lerp(SplinePoint a, SplinePoint b, float t)	Interpolates between two spline points
bool AreDifferent(ref SplinePoint a, ref SplinePoint b)	Returns true if any of the points' values are different

23.2. SplineSample

23.2.1. Public Properties

Vector3 position	The position of the sample
Vector3 up	The up direction of the sample (based on the spline normal)
Vector3 forward	The forward direction of the sample (based on the spline direction)
Vector3 right	A perpendicular vector to up and forward
Color color	The color of the sample
float size	The size of the sample
double percent	The percent along the spline of the sample
Quaternion rotation	A quaternion rotation, calculated from the direction vectors of the sample

23.2.2. Public Methods

void Lerp(SplineSample b, double t)	Interpolates the sample towards sample "b" with a percent t
void Lerp(SplineSample b, float t)	Interpolates the sample towards sample "b" with a percent t
void CopyFrom(SplineSample input)	Copies the properties of another sample into the current sample

23.2.3. Static Methods

SplineSample Lerp(SplineSample a, SplineSample b, float t)	Interpolates between two spline results with time t and returns the result
SplineSample Lerp(SplineSample a, SplineSample b, double t)	Interpolates between two spline results with time t and returns the result

23.3. Spline

23.3.1. Enums

Direction { Forward = 1, Backward = -1 }	Traversing direction of the spline.
Type { Hermite, BSpline, Bezier, Linear };	Type of the spline as described in the documentation

23.3.2. Public Properties

Spline.Type type	The type of the spline
SplinePoint[] points	The control points array which make up the spline
int sampleRate	The sample rate of the spline – the bigger, the smoother
AnimationCurve customValueInterpolation	Custom curve for size and color interpolation
AnimationCurve customNormalInterpolation	Custom curve for normal interpolation
bool linearAverageDirection	Should sample directions be averaged in linear mode?
bool isClosed	Is the spline closed (read only)?
double moveStep	The average percent distance between spline samples based on the sampleRate
int iterations	The total count of samples the spline is expected to have based on sampleRate and point count

23.3.3. Public Methods

float CalculateLength(double from = 0.0, double to = 1.0, double resolution = 1.0)	Calculates the length in world units between two points along the spline
double Project(Vector3 position, int subdivide = 4, double from = 0.0, double to = 1.0)	Finds the closest point along the spline to a world point and returns its percent value
bool Raycast(out RaycastHit hit, out double hitPercent, LayerMask layerMask, double resolution = 1.0, double from = 0.0, double to = 1.0, QueryTriggerInteraction hitTriggers = QueryTriggerInteraction.UseGlobal)	Raycasts along the spline and returns the first hit
bool RaycastAll(out RaycastHit[] hits, out double[] hitPercents, LayerMask layerMask, double resolution = 1.0, double from = 0.0, double to = 1.0, QueryTriggerInteraction hitTriggers = QueryTriggerInteraction.UseGlobal)	Raycasts along the spline and returns all hits
double GetPointPercent(int pointIndex)	Returns the percent along a spline of a point based on its index
Vector3 EvaluatePosition(double percent)	Evaluates the spline at the provided percent and returns the world position at that percent
SplineSample Evaluate(double percent)	Evaluates the spline at the provided percent and returns the spline sample at that percent

SplineSample Evaluate(int pointIndex)	**Evaluates the spline at a given control point and returns the spline sample at that point**
void Evaluate(SplineSample result, int pointIndex)	**A version of Evaluate which does not allocate new objects**
void Evaluate(SplineSample result, double percent)	**A version of Evaluate which does not allocate new objects**
void Evaluate(ref SplineSample[] samples, double from = 0.0, double to = 1.0)	**Evaluates the spline throughout the provided range and writes all samples to the provided sample array**
void EvaluateUniform(ref SplineSample[] samples, ref double[] originalSamplePercents, double from = 0.0, double to = 1.0)	**Evaluates the spline throughout the provided range and writes all samples to the provided sample array. The spline is sampled at uniform distances.**
void EvaluatePositions(ref Vector3[] positions, double from = 0.0, double to = 1.0)	**Evaluates the spline throughout the provided range and writes all positions to the provided sample array**
double Travel(double start, float distance, out float moved, Direction direction)	**Moves along a spline using world units and returns the new percent**
double Travel(double start, float distance, Spline.Direction direction = Spline.Direction.Forward)	**Moves along a spline using world units and returns the new percent**
void EvaluatePosition(ref Vector3 point, double percent)	**Evaluates the spline at the provided percent and writes the world position at that percent to the referenced field**
void Close()	**Closes the spline**
void Break()	**Breaks (opens) the spline at the closed point**
Break(int at)	**Breaks (opens) the spline at a specific point**
void HermiteToBezierTangents()	**Converts a Hermite spline into a Bezier spline**

23.3.4. Static Methods

void FormatFromTo(ref double from, ref double to, bool preventInvert = true)	**Used for formatting a from and to percent so that they can be safely used in internal spline methods**

23.4. SampleCollection

23.4.1. Public Properties

SplineSample[] samples	The array of spline samples this collection holds
SplineComputer.SampleMode sampleMode	The sample mode these samples have been calculated with
int Count	The count of the samples
int[] optimizedIndices	Index mapping array for optimized samples only
double clipFrom = 0.0	The start region the samples are clipped from
double clipTo = 0.0	The end region the samples are clipped to
bool loopSamples	Should the samples looped (closed)?
bool samplesAreLooped	Are the samples successfully looped?
double span	The total spline range [0-1] of the samples.

23.4.2. Public Methods

int GetClippedSampleCount(out int startIndex, out int endIndex)	Returns the number of samples which fall inside the clip range. Based on the clip range and the samples array.
double ClipPercent(double percent)	Converts a global percent [0-1] into a clip range percent [0-1]
void ClipPercent(ref double percent)	Converts a global percent [0-1] into a clip range percent [0-1]
double UnclipPercent(double percent)	Converts a clip range percent into a global percent
void UnclipPercent(ref double percent)	Converts a clip range percent into a global percent
void GetSamplingValues(double percent, out int sampleIndex, out double lerp)	Returns the sample index and the interpolation value towards the next sample based on the provided percent
Vector3 EvaluatePosition(double percent)	Evaluates the samples at the provided percent and returns the world position at that percent
SplineSample Evaluate(double percent)	Evaluates the samples at the provided percent and returns a spline sample at that percent
void Evaluate(double percent, SplineSample result)	A version of Evaluate which does not allocate new objects
void Evaluate(ref SplineSample[] results, double from = 0.0, double to = 1.0)	Evaluates the samples throughout the provided range and writes all samples to the provided sample array
void EvaluatePositions(ref Vector3[] positions, double from = 0.0, double to = 1.0)	Evaluates the samples throughout the provided range and writes all positions to the provided sample array
double Travel(double start, float distance, Spline.Direction direction, out float moved)	Moves along a sample collection using world units and returns the new percent

double Travel(double start, float distance, Spline.Direction direction = Spline.Direction.Forward)	Moves along a sample collection using world units and returns the new percent
void Project(Vector3 position, int controlPointCount, SplineSample result, double from = 0.0, double to = 1.0)	Finds the closest point along the collection to a world point and returns its percent value
float CalculateLength(double from = 0.0, double to = 1.0)	Calculates the length in world units between two points along the spline

23.5. SplineComputer

23.5.1. Enums

Space { World, Local };	• World space does not take the Spline's transform into account • Local transforms the spline using the transform
EvaluateMode { Cached, Calculate }	• Cached uses the cached spline samples to operate • Calculate recalculates the spline for better precision at the cost of performance
SampleMode { Default, Uniform, Optimized }	Sample mode as described in the documentation
UpdateMode { Update, FixedUpdate, LateUpdate, AllUpdate, None }	Which update method should the Spline Computer use? None does not update the spline and waits for manual update using the Rebuild method

23.5.2. Public Properties

SplineComputer.Space space	The transform space of the Spline Computer
Spline.Type type	The type of the spline
bool linearAverageDirection	Should sample directions be averaged in linear mode?
bool is2D	Toggle 2D mode for the spline
int sampleRate	The sample rate of the spline – the bigger, the smoother
float optimizeAngleThreshold	The angle below which samples get deleted when using Optimized sample mode
SampleMode sampleMode	The sample mode of this Spline Computer as described in the documentation
bool multithreaded	Toggle multithreading for the Spline Computer
bool rebuildOnAwake	Should the spline rebuild as soon as it is awake in the scene?
SplineComputer.UpdateMode updateMode	The update mode of the Spline Computer
TriggerGroup[] triggerGroups	A collection of triggers used by other components to invoke events
AnimationCurve customValueInterpolation	Custom curve for size and color interpolation
AnimationCurve customNormalInterpolation	Custom curve for normal interpolation
int iterations	The total count of samples the spline is expected to have based on sampleRate and point count

double moveStep	The average percent distance between spline samples based on the sampleRate
bool isClosed	Is the spline closed?
int pointCount	Returns the control point count of the spline (read only)
SplineSample[] samples	Returns the computed and transformed samples along the whole spline
int sampleCount	Returns the count of the computed samples (read only)
SplineSample[] rawSamples	Returns the raw samples before transformation
Vector3 position	Thread-safe position of the Spline Computer's transform component
Quaternion rotation	Thread-safe rotation of the Spline Computer's transform component
Vector3 scale	Thread-safe scale of the Spline Computer's transform component
int subscriberCount	Returns the number of SplineUsers that are currently using this Spline Computer

23.5.3. Public Methods

void GetSamples(SampleCollection collection)	Writes the calculated samples into the passed SampleCollection object
void ResampleTransform()	Resamples the transform so that samples can be properly transformed
bool IsSubscribed(SplineUser user)	Checks if a Spline User is subscribed to the given computer
SplineUser[] GetSubscribers()	Returns all subscribed Spline Users
SplinePoint[] GetPoints(Space getSpace = Space.World)	Returns all spline points of the Spline Computer
SplinePoint GetPoint(int index, Space getSpace = Space.World)	Returns the point at the given index
Vector3 GetPointPosition(int index, Space getSpace = Space.World)	Returns the position of the point at the given index
Vector3 GetPointNormal(int index, Space getSpace = Space.World)	Returns the normal of the point at the given index
Vector3 GetPointTangent(int index, Space getSpace = Space.World)	Returns the tangent of the point at the given index
Vector3 GetPointTangent2(int index, Space getSpace = Space.World)	Returns the second tanget of the point at the given index
float GetPointSize(int index, Space getSpace = Space.World)	Returns the size of the point at the given index
Color GetPointColor (int index, Space getSpace = Space.World)	Returns the color of the point at the given index
void SetPoints(SplinePoint[] points, Space setSpace = Space.World)	Sets the points of the Spline Computer to the ones from the array

void SetPointPosition(int index, Vector3 pos, Space setSpace = Space.World)	Sets the position of the point at the given index
void SetPointTangents(int index, Vector3 tan1, Vector3 tan2, Space setSpace = Space.World)	Sets the tangents of the point at the given index
void SetPointNormal(int index, Vector3 nrm, Space setSpace = Space.World)	Sets the normal of the point at the given index
void SetPointSize(int index, float size)	Sets the size of the point at the given index
void SetPointColor(int index, Color color)	Sets the color of the point at the given index
void SetPoint(int index, SplinePoint point, Space setSpace = Space.World)	Sets the point at the given index
double GetPointPercent(int pointIndex)	Returns the percent of a point along the spline based on its index
int PercentToPointIndex(double percent, Spline.Direction direction = Spline.Direction.Forward)	Finds the point index that corresponds to the given percent along the spline
Vector3 EvaluatePosition(double percent)	Evaluates the spline at the provided percent and returns the world position at that percent
Vector3 EvaluatePosition(double percent, EvaluateMode mode = EvaluateMode.Cached)	Evaluates the spline at the provided percent and returns the transformed position at that percent
Vector3 EvaluatePosition(int pointIndex, EvaluateMode mode = EvaluateMode.Cached)	Evaluates the spline at the given control point and returns the transformed position at that percent
SplineSample Evaluate(double percent)	Evaluates the spline at the provided percent and returns the transformed sample at that percent
SplineSample Evaluate(double percent, EvaluateMode mode = EvaluateMode.Cached)	Evaluates the spline at the provided percent and returns the transformed sample at that percent
SplineSample Evaluate(int pointIndex)	Evaluates the spline at the given control point and returns the transformed sample at that percent
void Evaluate(int pointIndex, SplineSample result)	Evaluates the spline at the given control point and writes the result to the provided sample
void Evaluate(double percent, SplineSample result)	Evaluates the spline at the provided percent and writes the result to the provided sample
void Evaluate(double percent, SplineSample result, EvaluateMode mode = EvaluateMode.Cached)	Evaluates the spline at the provided percent and writes the result to the provided sample
void Evaluate(ref SplineSample[] results, double from = 0.0, double to = 1.0)	Evaluates the spline throughout the provided range and writes all samples to the provided sample array
void EvaluatePositions(ref Vector3[] positions, double from = 0.0, double to = 1.0)	Evaluates the spline throughout the provided range and writes all sampled positions to the provided vector array
double Travel(double start, float distance, out float moved, Spline.Direction direction = Spline.Direction.Forward)	Moves along the spline based on a start percent and a distance in world

77

	units and returns the new percent at the reached position as well as the moved distance
double Travel(double start, float distance, Spline.Direction direction = Spline.Direction.Forward)	Moves along the spline based on a start percent and a distance in world units and returns the new percent at the reached position
void Project(SplineSample result, Vector3 position, double from = 0.0, double to = 1.0, EvaluateMode mode = EvaluateMode.Cached, int subdivisions = 4)	Projects a point in world space onto the spline and returns the percent along the spline that resembles the closest sample to that point
float CalculateLength(double from = 0.0, double to = 1.0)	Calculates the distance in world units of the provided range
void Rebuild(bool forceUpdateAll = false)	Recalculates the Spline Computer and all of its subscribers (Spline Users) on the next frame
void RebuildImmediate(bool calculateSamples = true, bool forceUpdateAll = false)	Recalculates the Spline Computer and all of its subscribers (Spline Users) immediately
void Close()	Closes the spline (at least 4 control points required)
void Break()	Breaks the spline
void Break(int at)	Breaks the spline at a specific control point
void HermiteToBezierTangents()	Converts a Hermite spline into a Bezier spline
bool Raycast(out RaycastHit hit, out double hitPercent, LayerMask layerMask, double resolution = 1.0, double from = 0.0, double to = 1.0 , QueryTriggerInteraction hitTriggers = QueryTriggerInteraction.UseGlobal)	Raycasts along the spline and returns hit information
bool RaycastAll(out RaycastHit[] hits, out double[] hitPercents, LayerMask layerMask, double resolution = 1.0, double from = 0.0, double to = 1.0, QueryTriggerInteraction hitTriggers = QueryTriggerInteraction.UseGlobal)	Raycasts along the spline and returns all objects that have been hit
void CheckTriggers(double start, double end)	Checks and invokes any triggers inside the start – end range
void CheckTriggers(int group, double start, double end)	Checks and invokes only the triggers inside the given trigger group inside the start – end range
void ResetTriggers()	Resets all triggers that have been set to work once
void ResetTriggers(int group)	Resets only the triggers inside the given trigger group that have been set to work once
List<Node.Connection> GetJunctions(int pointIndex)	Returns the available junctions for the given point
Dictionary<int, List<Node.Connection>> GetJunctions(double start = 0.0, double end = 1.0)	Returns all junctions for the points in the given range
void ConnectNode(Node node, int pointIndex)	Connect the referenced Node to the spline point
void DisconnectNode(int pointIndex)	Disconnects the connected node at

the given point

23.6. SplineUser

23.6.1. Enums

UpdateMethod { Update, FixedUpdate, LateUpdate }

23.6.2. Public Properties

SplineComputer spline	The spline computer that this user is using
double clipFrom	The start range of the user
double clipTo	The end range of the user
bool autoUpdate	Should this user update automatically? If not, Rebuild must be called manually
bool loopSamples	Should samples be looped (only works for closed splines)?
double span	The span of the user's clip range (read-only)
bool samplesAreLooped	Returns true if the spline is closed, loopSamples is true and the clipFrom is bigger than clipTo
RotationModifier rotationModifier	The rotation modifier module of the spline user
OffsetModifier offsetModifier	The offset modifier module of the spline user
ColorModifier colorModifier	The color modifier module of the spline user
SizeModifier sizeModifier	The size modifier module of the spline user
bool multithreaded	Should multithreading be used?
bool buildOnAwake	Should the user rebuild on Awake?
bool buildOnEnable	Should the user rebuild on Enable?

23.6.3. Public Methods

SplineSample GetSample(int index)	Returns the spline sample at the given index
void Rebuild()	Causes the user to rebuild on the next frame
void RebuildImmediate()	Causes the user to rebuild immediately
void ModifySample(SplineSample source, SplineSample destination)	Modifies a sample using the user's modifiers
void SetClipRange(double from, double to)	Sets the clip range of the user (same as setting clipFrom and clipTo)
double ClipPercent(double percent)	Converts a global spline percent [0-1] to a local percent within [clipFrom-clipTo]
void ClipPercent(ref double percent)	Converts a global spline percent [0-1] to a local percent within [clipFrom-clipTo]
double UnclipPercent(double percent)	Converts a local [clipFrom-clipTo] percent to a global [0-1] percent
void UnclipPercent(ref double percent)	Converts a local [clipFrom-clipTo] percent to a global [0-1] percent

Vector3 EvaluatePosition(double percent)	Evaluates the clip range and returns the position at the given percent
void Evaluate(double percent, SplineSample result)	Evaluates the clip range and writes the spline sample at the given percent to the given result object
SplineSample Evaluate(double percent)	Evaluates the clip range and returns a spline sample at the given percent
void Evaluate(ref SplineSample[] results, double from = 0.0, double to = 1.0)	Evaluates the entire clip range within the given range and writes them to an array of spline samples
void EvaluatePositions(ref Vector3[] positions, double from = 0.0, double to = 1.0)	Evaluates the entire clip range within the given range and writes them to the and writes them to the provide Vector3 array
double Travel(double start, float distance, Spline.Direction direction, out float moved)	Moves along the clip range, based on a start percent and a distance in world units and returns the new percent at the reached position
double Travel(double start, float distance, Spline.Direction direction = Spline.Direction.Forward)	Moves along the clip range, based on a start percent and a distance in world units and returns the new percent at the reached position as well as the moved distance
void Project(Vector3 position, SplineSample result, double from = 0.0, double to = 1.0)	Projects a point in world space onto the clip range and returns the percent along the spline that resembles the closest sample to that point
float CalculateLength(double from = 0.0, double to = 1.0)	Calculates the distance along the clip range within the provided range

23.7. SplineTrigger

23.7.1. Enums

Type { Double, Forward, Backward }	- Double works both forwards and backwards - Forward only works forwards - Backward only works backwards

23.7.2. Public Properties

string name	The name of the trigger
SplineTrigger.Type type	The type of the trigger
bool workOnce	Should this trigger only work once?
double position	The percent along the spline where this trigger is located
bool enabled	Is this trigger enabled and working?
Color color	The color of the trigger
UnityEvent onCross	UnityEvent to invoke when the trigger is triggered

23.7.3. Public Methods

void AddListener(UnityAction action)	Adds a UnityAction to the trigger which will be invoked when the trigger is triggered
void Reset()	Resets the trigger if it is set to work once and it has already worked
bool Check(double previousPercent, double currentPercent)	Checks the trigger (called automatically by SplineComputer)
void Invoke()	Invokes the trigger (called automatically by SplineComputer)

23.8. SplineTriggerGroup

23.8.1. Public Properties

bool enabled	Is the trigger group enabled?
string name	The name of the trigger group
Color color	The color of the trigger group
SplineTrigger[] triggers	The triggers in this trigger group

23.8.2. Public Methods

void Check(double start, double end)	Checks all triggers inside the trigger group (called automatically by SplineComputer)
void Reset()	Resets all triggers inside the trigger group