Dreamteck Splines – API Reference

http://dreamteck.io

https://www.facebook.com/dreamteckstudio

Contact the team: team@dreamteck.io

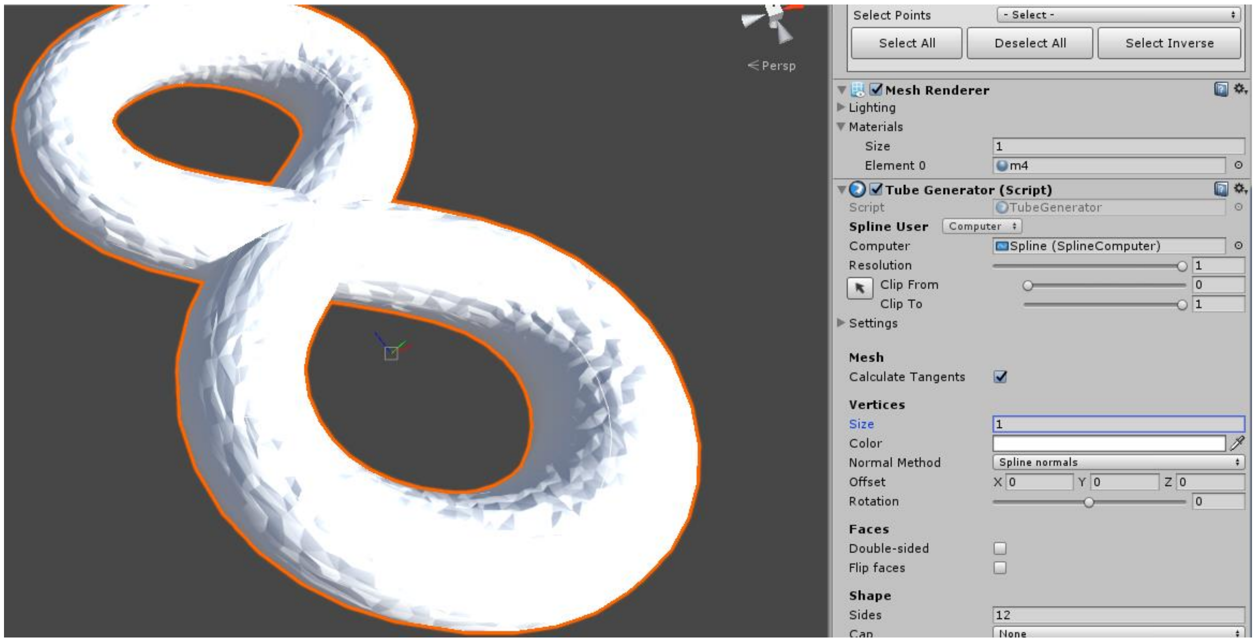
Contact support: support@dreamteck.io

Contents

API Reference Notice

Only some SplineUser classes are listed in this API reference. The reason is that most of the SplineUsers have their properties listed in the Inspector and whenever a property is listed in the inspector, it's also available in the API.

For example, in the case of this TubeGenerator:



Select Points: - Select -

Mesh Renderer
Lighting
Materials
Size: 1
Element 0: m4

Tube Generator (Script)
Script: TubeGenerator
Spline User: Computer
Computer: Spline (SplineComputer)
Resolution: 1
Clip From: 0
Clip To: 1
Settings

Mesh
Calculate Tangents: ✓

Vertices
Size: 1
Color
Normal Method: Spline normals
Offset: X 0 Y 0 Z 0
Rotation: 0

Faces
Double-sided: ☐
Flip faces: ☐

Shape
Sides: 12
Cap: None

Its size property is exposed and available to edit in the inspector. This means that it can be modified with code in the following way:

```
GetComponent<TubeGenerator>().size = 0.5f;
```

Namespaces

In order to use the Dreamteck Splines API, the following namespaces have to be included:

Dreamteck.Splines	The core namespace where all components and classes, needed for working with splines are.
Dreamteck.Splines.IO	The namespace which contains classes for importing and exporting functionality
Dreamteck.Splines.Primitives	The namespace which contains the spline Primitives

The IO and Primitives namespaces are not required unless importing/exporting and primitive creating functionality is needed.

SplinePoint

Struct in Dreamteck.Splines

Description

Representation of a control point. SplinePoint is used to define Splines. Editing the points of a Spline edits the spline too.

Public Properties

SplinePoint.Type type	The type of the point
Vector3 position	The position of the point
Color color	The color of the point
Vector3 normal	The normal direction of the point
float size	The size of the point
Vector3 tangent	The first tangent position for Bezier splines
Vector3 tangent2	The second tangent position for Bezier splines

Public Methods

SetPosition(Vector3 pos)	Sets the position of the point and moves its tangents too
SetTangentPosition(Vector3 pos)	Sets the tangent position of the point
SetTangent2Position(Vector3 pos)	Sets the second tangent's position of the point

Static Methods

SplinePoint Lerp(SplinePoint a, SplinePoint b, float t)	Interpolation between two spline points

Enumerations

Type {Smooth, Broken}	Smooth mirrors the tangents, Broken make the tangents independent

Constructors

SplinePoint(Vector3 p)	Creates a new smooth point
SplinePoint(Vector3 p, Vector3 t)	Creates a new smooth point
SplinePoint(Vector3 pos, Vector3 tan, Vector3 nor, float s, Color col)	Creates a new fully defined smooth point
SplinePoint(Vector3 pos, Vector3 tan, Vector3 tan2, Vector3 nor, float s, Color col)	Creates a new fully defined broken point

SplineResult

Class in Dreamteck.Splines

Description

When a spline is evaluated, multiple values are returned. This is the result of spline evaluation.

Public Properties

Vector3 position	The position of the evaluation result
Vector3 normal	The normal of the evaluation result
Vector3 direction	The direction of the evaluation result
Color color	The color of the evaluation result
float size	The size of the evaluation result
double percent	The time (0-1) the spline was evaluated at

Read-only Properties

Quaternion rotation	Returns Quaternion.LookRotation(direction, normal)
Vector3 right	Returns a perpendicular vector to direction and normal

Public Methods

Lerp(SplineResult b, double t)	Interpolates between the current values and b’s values
Lerp(SplineResult b, float t)	Interpolates between the current values and b’s values
void CopyFrom(SplineResult input)	Copies the values of the input SplineResult object

Static Methods

SplineResult Lerp(SplineResult a, SplineResult b, double t)	Interpolates between two spline results

SplineResult Lerp(SplineResult a, SplineResult b, float t)	Interpolates between two spline results
public static void Lerp(SplineResult a, SplineResult b, double t, SplineResult target)	Interpolates between two spline results and stores the result in the target spline result
public static void Lerp(SplineResult a, SplineResult b, float t, SplineResult target)	Interpolates between two spline results and stores the result in the target spline result

Constructors

SplineResult()	Creates a spline result with default values
SplineResult(Vector3 p, Vector3 n, Vector3 d, Color c, float s, double t)	Creates a fully-defined spline result
SplineResult(SplineResult input)	Creates a deep copy of input

Spline

Class in Dreamteck.Splines

Description

A spline in world space. This class stores a single spline and provides methods for evaluation, length calculation, raycasting and more.

Example

```
using UnityEngine;
using System.Collections;
using Dreamteck.Splines; //Include the Splines namespace

public class SimpleSplineController : MonoBehaviour {
       void Start () {
        //Create a new B-spline with precision 0.9
        Spline spline = new Spline(Spline.Type.BSpline, 0.9);
        //Create 3 control points for the spline
        spline.points = new SplinePoint[3];
        spline.points[0] = new SplinePoint(Vector3.left);
        spline.points[1] = new SplinePoint(Vector3.up);
        spline.points[2] = new SplinePoint(Vector3.right);
        //Evaluate the spline and get an array of values
        SplineResult[] results = new SplineResult[spline.iterations];
        spline.Evaluate(results);
        //Display the values in the editor
        for (int i = 0; i < results.Length; i++)
        {
            Debug.DrawRay(results[i].position, results[i].normal, results[i].color);
        }
    }
}
```

Public Properties

SplinePoint[] points	The control points of the spline
Spline.Type type	The type of the spline which defines what interpolation should be used.
double precision	The approximation rate (0-0.9999) of the spline
AnimationCurve customValueInterpolation	Custom curve for size and color interpolation between points

AnimationCurve customNormalInterpolation	Custom curve for normal interpolation between points

Read-only Properties

bool isClosed	Whether or not the spline is closed
double moveStep	The step size of the percent incrementation when evaluating a spline (based on percision)
int iterations	The total count of samples for the spline (based on the precision)

Public Methods

SplineResult Evaluate(double percent)	Evaluates the spline at the given time and returns a **SplineResult** object
void Evaluate(ref SplineResult[] samples, double from = 0.0, double to = 1.0)	Evaluates the spline using its precision and writes the results to the array
Vector3 EvaluatePosition(double percent)	Evaluates the spline and returns the position. This is simpler and faster than Evaluate.
void EvaluatePositions(Vector3[] positions, double from = 0.0, double to = 1.0)	Evaluates the spline using its precision and writes the result positions to the array.
public float CalculateLength(double from = 0.0, double to = 1.0, double resolution = 1.0)	Calculates the length of the spline
double Project(Vector3 point, int subdivide = 4, double from = 0.0, double to = 1.0)	Projects a point on the spline. Returns evaluation percent.
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)	Casts rays along the spline against all colliders in the scene
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)	Casts rays along the spline against all colliders in the scene and returns all hits. Order is not guaranteed.

double Travel(double start, float distance, Direction direction)	Returns the percent from the spline at a given distance from the start point
void Close()	Closes the spline (requires the spline to have at least 4 points)
void Break()	Breaks the closed spline
void Break(int at)	Breaks the closed spline at a given point
void ConvertToBezier()	Converts the spline from Hermite to Bezier without losing its initial shape.

Constructors

Spline(Type t)	Creates a spline of a given type

SplineComputer : MonoBehaviour

Class in Dreamteck.Splines

Description

The SplineComputer is attached to a Game Object in the scene and serves as a thread-safe MonoBehaviour wrapper for the Spline class. This is a component that holds a single Spline object, has all the public methods and properties the Spline class has and applies transformation to the spline according to its Game Object’s Transform component.

Example

```
        //Get the SplineComputer component
        SplineComputer computer = GetComponent<SplineComputer>();
        //Make sure it's set to local space
        computer.space = Space.Local;
        //Get the computer's control poitns
        SplinePoint[] points = computer.GetPoints();
        //if no contorl points are found - stop - there is nothing to do
        if (points.Length == 0) return;
        //Edit the first and the last point's positions
        points[0].SetPosition(points[0].position + Vector3.up);
        points[points.Length - 1].SetPosition(points[points.Length - 1].position +
Vector3.down);
        //Set the new points for the computer
        computer.SetPoints(points);
        //Transform the computer a little
        computer.transform.localScale *= 1.5f;
        computer.transform.Rotate(Vector3.one * 45f);
        //Get the evaluated results which will also be transformed
        SplineResult[] results = new SplineResult[computer.iterations];
        computer.Evaluate(ref results);
        //Display the values in the editor
        for (int i = 0; i < results.Length; i++)
        {
            Debug.DrawRay(results[i].position, results[i].normal, results[i].color);
        }
```

Public Properties

Space space	The space in which the spline is evaluated.
Spline.Type type	The type of the spline which defines what interpolation should be used.

double precision	The approximation rate (0-0.9999) of the spline
AnimationCurve customValueInterpolation	Custom curve for size and color interpolation between points
AnimationCurve customNormalInterpolation	Custom curve for normal interpolation between points

Read-only Properties

bool isClosed	Whether or not the spline is closed
double moveStep	The step size of the percent incrementation when evaluating a spline (based on percision)
int iterations	The total count of samples for the spline (based on the precision)
int pointCount	The number of control points the spline is defined with
NodeLinks[] nodeLinks	The node links of the SplineComputer (used for connecting to Nodes)
SplineComputer.SplineMorph morph	The morph module of the SplineComputer
bool hasMorph	Does the SplineComputer have at least one morph channel?
Vector3 position	The position of the SplineComputer's Transform (thread safe)
Quaternion rotation	The rotation of the SplineComputer's Transform (thread safe)
Vector3 scale	The localScale of the SplineComputer's Transform (thread safe)
int subscriberCount	The number of SplineUser that are subscribed to this SplineComputer

Public Methods

SplineResult Evaluate(double percent)	Evaluates the spline at the given time and returns a **SplineResult** object
void Evaluate(ref SplineResult[] samples, double from = 0.0, double to = 1.0)	Evaluates the spline using its precision and writes the results to the array
Vector3 EvaluatePosition(double percent)	Evaluates the spline and returns the position. This is simpler and faster than Evaluate.
void EvaluatePositions(Vector3[] positions, double from = 0.0, double to = 1.0)	Evaluates the spline using its precision and writes the result positions to the array.
public float CalculateLength(double from = 0.0, double to = 1.0, double resolution = 1.0)	Calculates the length of the spline
double Project(Vector3 point, int subdivide = 4, double from = 0.0, double to = 1.0)	Projects a point on the spline. Returns evaluation percent.
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)	Casts rays along the spline against all colliders in the scene
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)	Casts rays along the spline against all colliders in the scene and returns all hits. Order is not guaranteed.
double Travel(double start, float distance, Direction direction)	Returns the percent from the spline at a given distance from the start point
void Close()	Closes the spline (requires the spline to have at least 4 points)
void Break()	Breaks the closed spline
void Break(int at)	Breaks the closed spline at a given point
void Subscribe(SplineUser input)	Subscribes a SplineUser to the SplineComputer
void Unsubscribe(SplineUser input)	Unsubscribes a SplineUser from the SplineComputer

bool IsSubscribed(SplineUser user)	Checks if a SplineUser is subscribed to the computer
void AddNodeLink(Node node, int pointIndex)	Add a link to a node. This is used by the Node class
void RemoveNodeLink(int pointIndex)	Removes a link to a node.
SplinePoint GetPoint(int index, Space getSpace = Space.World)	Returns a spline control point by its index (Transformed)
SplinePoint[] GetPoints(Space getSpace = Space.World)	Returns all spline control points.
Vector3 GetPointPosition(int index, Space getSpace = Space.World)	Returns the position of the given point
Vector3 GetPointNormal(int index, Space getSpace = Space.World)	Returns the normal of the given point
Vector3 GetPointTangent(int index, Space getSpace = Space.World)	Returns the tangent of the given point
Vector3 GetPointTangent2(int index, Space getSpace = Space.World)	Returns the second tangent of the given point
float GetPointSize(int index, Space getSpace = Space.World)	Returns the size of the given point
Color GetPointColor (int index, Space getSpace = Space.World)	Returns the color of the given point
void SetPoint(int index, SplinePoint point,, Space setSpace = Space.World)	Sets the value of a control point.
void SetPoints(SplinePoint[] points, Space setSpace = Space.World)	Sets the points of the spline.
void SetPointPosition(int index, Vector3 pos, Space setSpace = Space.World)	Sets the position of a single control point
SetPointTangents(int index, Vector3 tan1, Vector3 tan2, Space setSpace = Space.World)	Set the tangents of a single control point
void SetPointNormal(int index, Vector3 nrm, Space setSpace = Space.World)	Set the normal of a single control point

void SetPointSize(int index, float size)	Set the size of a single control point
void SetPointColor(int index, Color color)	Set the color of a single control point
void Rebuild()	Forces the SplineComputer to rebuild all subscribed users on the next update cycle
void RebuildImmediately()	Forces the SplineComputer to rebuild all subscribed users immediately
int[] GetAvailableNodeLinksAtPosition (double percent, Spline.Direction direction)	Returns an array of node link indices
void SetMorphState(int index)	Set the selected morph's weight to 1 and all others to 0
void SetMorphState(string morphName)	Set the selected morph's weight to 1 and all others to 0
void SetMorphState(int index, float percent)	Set the selected morph's weight to percent and reduce all other morph weights automatically
void SetMorph(string morphName, float percent)	Set the selected morph's weight to percent and reduce all other morph weights automatically
void SetMorphState(float percent)	Automatically assigns a weight value to all morph states based on the percent. 0 will set the first morph's weight to 1 and all others to 0. 1 will set the last morph's weight to 1 and all others to 0.
List<SplineComputer> GetConnectedComputers()	Returns a list of all connected computers using Nodes. The list includes the current computer too.
void GetConnectedComputers(List<SplineComputer> computers, List<int> connectionIndices, List<int> connectedIndices, double percent, Spline.Direction direction, bool includeEqual)	Gets the connected computers of a computer at percent, along direction. This overload of the method also gets the points they are connected to and the points they are connected at.

Enumerations

Space {World, Local}	The space that the SplineComputer uses to transform the spline.

SplineComputer.Morph

Class in Dreamteck.Splines

Description

A morph module for the SplineComputer component which stores and blends different spline paths with the same amount of control points. It's used for shape animations during runtime.

Public Methods

void AddChannel(string name)	Creates a new morph channel which can be blended
void RemoveChannel(string name)	Removes a morph channel by name
int GetChannelCount()	Returns the channel count of the morph
string[] GetChannelNames()	Returns the channel names of the morph
float GetWeight(int index)	Gets the weight of a channel by index
float GetWeight(string name)	Gets the weight of a channel by name
void SetWeight(int index, float weight)	Sets the weight of a channel by index
void SetWeight(string name, float weight)	Sets the weight of a channel by name
void CaptureSnapshot(int index)	Saves the points of the spline into a channel (by index)
void CaptureSnapshot(string name)	Saves the points of the spline into a channel (by name)
SplinePoint[] GetSnapshot(int index)	Gets the points of the spline from a channel (by index)
SplinePoint[] GetSnapshot(string name)	Gets the points of the spline from a channel (by name)
void Clear()	Removes all morph channels

Node : MonoBehaviour

Class in Dreamteck.Splines

Description

The Node class is used to bind the control points of SplineComputers to Game Objects in the scene and also to create junctions.

Public Properties

Node.Type type	Defines the way the node connects the points.
bool transformNormals	Should it transform the normals of the connected points?
bool transformSize	Should it transform the sized of the connected points?
bool transformTangents	Should it transform the tangents of the connected points?

Public Methods

void UpdateConnectedComputers(SplineComputer excludeComputer = null)	Forces the connected computers to rebuild their subscribed users. Can exclude one computer from update.
void UpdatePoint(SplineComputer computer, int pointIndex, SplinePoint point)	Updates the values of a connected point
void AddConnection(SplineComputer computer, int pointIndex)	Adds a new Spline Computer and point to the connections
void RemoveConnection(SplineComputer computer, int pointIndex)	Removes a Spline Computer and point from the connections
bool HasConnection(SplineComputer computer, int pointIndex)	Checks if a computer is connected at the given point
void ClearConnections()	Removes all connections from the node
Connection[] GetConnections()	Returns the connections the node has

Enumerations

Type {Smooth, Free}	Smooth makes the values of the connected points the same while free allows each point to retain its normal, size and color.

Node.Connection

Class in Dreamteck.Splines

Description

A connection definition for the Node class. It stores a computer and the point index it’s connected at.

Public Properties

SplineComputer computer	The computer that is connected
int pointIndex	The index of the point that the computer is connected at

SplineUser : MonoBehaviour

Class in Dreamteck.Splines

Description

A base class that utilizes the SplineComputer component. It samples a single SplineComputer and provides useful sample information that can be used for path following, mesh generation, object positioning and everything else that imagination is capable of.

The SplineUser has a resolution multiplier [0-1] which can be used to reduce the sample rate for the SplineComputer as well as clip from and clip to values [0-1] which control what segment of the spline is sampled.

The SplineUser implements a multithreading framework which allows developers to easily make their code multithreaded.

Changing the public properties of a SplineUser causes it to automatically resample the Spline Computer the on the next update cycle.

Please refer to the User Manual for all available SplineUser-derived components.

Refer to BlankUser.cs in the Components folder for how to derive your own SplineUser classes.

All of the SplineUser's public properties update the user automatically when changed. For example, there is no need to call Subscribe if computer is set and there is no need to call Rebuild when resolution or clipFrom/clipTo are set.

Public Properties

UpdateMethod updateMethod	When should the user update?
SplineComputer computer	The computer that the SplineUser samples
SplineUser user	A SplineUser reference to use instead of a SplineComputer
double resolution	The resolution multiplier [0-1] to sample the SplineComputer with
double clipFrom	Where to start evaluating the spline from? [0-1]

double clipTo	Where to evaluate the spline to? [0-1]
bool averageResultVectors	Should normals and directions be averaged when sampling?
bool multithreaded = false;	Should the SplineUser use multithreading?
SplineAddress address	The junction address of the SplineUser. Used for taking different routes when there are junctions

Read-only Properties

double span	Always equal to clipTo - clipFrom

Protected Properties

SplineResult[] samples	The samples from the SplineComputer component
SplineResult[] clippedSamples	The clipped samples from the SplineComputer component defined by clipFrom and clipTo

Public Methods

void Rebuild(bool sampleComputer)	Forces a rebuild on the next update cycle. sampleComputer controls whether the user should re-evaluate the SplineComputer
void RebuildImmediate(bool sampleComputer)	Forces a rebuild immediately
void EnterAddress(Node node, int pointIndex, Spline.Direction direction = Spline.Direction.Forward)	Adds a new element to the junction address
void AddComputer(SplineComputer computer, int connectionIndex, int connectedIndex, Spline.Direction direction = Spline.Direction.Forward)	Adds a new element to the junction address but instead of a node, this method directly uses a SplineComputer reference and the two connection point indices. This method lets the user add any computer to the junction address. The computer doesn't have to be connected with a Node.
void ExitAddress(int depth)	Removes the last (depth) elements from the junction

	address
void ClearAddress()	Clears the junction address
void CollapseAddress()	Flattens the junction address so that all other address entries except the last one are removed
int GetSampleIndex(double percent)	Get a sample index from a percent
SplineResult Evaluate(double percent)	Evaluates the sampled spline
void Evaluate(SplineResult result, double percent)	Evaluates the sampled spline and stores the result in the result parameter
Vector3 EvaluatePosition(double percent)	Evaluates the sampled spline and returns the position. This is simpler and faster than Evaluate.
SplineResult Project(Vector3 point, double from = 0.0, double to = 1.0)	Projects a point on the sampled spline
void Project(SplineResult result, Vector3 point, double from = 0.0, double to = 1.0)	Projects a point on the sampled spline and writes the result in to the object in the “result” parameter
double Travel(double start, float distance, Direction direction)	Returns the percent from the sampled spline at a given distance from the start point

Enumerations

enum UpdateMethod { Update, FixedUpdate, LateUpdate}	When should the SplineUser rebuild?

SplineTracer : SplineUser

Class in Dreamteck.Splines

Description

A base class inherited by the SplineFollower, SplinePositioner and SplineProjector. It provides positioning and trigger functionality. This component doesn't do anything by itself but can be used through the API to get position results along a spline although this would rarely be needed.

Public Properties

PhysicsMode physicsMode	Whether to apply the follow motion to a transform or to a rigidbody.
Spline.Direction direction	The follow direction of the follower. It can be either forward or backward.
TransformModule motion	The transform module of the follower which defines how the follow motion is applied
CustomOffsetModule customOffsets	A Custom offset module that creates custom offset regions along the spline's evaluation
CustomRotationModule customRotations	A custom rotations module that creates custom rotation offsets along the spline's evaluation

Read-only Properties

SplineResult result	The current spline result
SplineResult offsettedResult	The current spline result with applied offsets

Public Methods

void SetPercent(double distance)	Set the position of the follower along the spline immediately
void SetDistance(float distance)	Set the follower position to a certain distance from the start of the spline
void AddTrigger(SplineTrigger.Type t, UnityAction call, double position = 0.0, SplineTrigger.Type type =	Add a trigger with a parameterless action

SplineTrigger.Type.Double)	
void Restart(double startPosition = 0.0)	Restarts the follower and positions it at the given percent [0-1]
void AddTrigger(SplineTrigger.Type t, UnityAction<int> call, int value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an integer parameter
void AddTrigger(SplineTrigger.Type t, UnityAction<float> call, float value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an float parameter
void AddTrigger(SplineTrigger.Type t, UnityAction<double> call, double value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an double parameter
void AddTrigger(SplineTrigger.Type t, UnityAction<string> call, string value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an string parameter
void AddTrigger(SplineTrigger.Type t, UnityAction<bool> call, bool value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an bool parameter
void AddTrigger(SplineTrigger.Type t, UnityAction<GameObject> call, GameObject value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an GameObject parameter
void AddTrigger(SplineTrigger.Type t, UnityAction<Transform> call, Transform value, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with an action with an Transform parameter

Enumerations

enum PhysicsMode { Transform, Rigidbody, Rigidbody2D }	Physics mode defines how to apply the follow motion to the follower. Transform is the default option and it applies the motion to the Transform

component, disregarding physics. Rigidbody and Rigidbody2D will apply the motion to the attached rigidbody making the follower properly collide with the world

SplineFollower : SplineTracer

Class in Dreamteck.Splines

Description

A SplineUser class dedicated for following SplineComputers. It can follow splines with constant speed regardless of how close or far apart are the spline points from each other. It can also follow based on time with variable speed.

Example

```
         public SplineComputer computer;
    SplineFollower follower;

       // Use this for initialization
       void Start () {
        follower = gameObject.AddComponent<SplineFollower>(); //Create a follower
component from scratch
        follower.computer = computer; //Set the computer of the follower component
        follower.followSpeed = 5f; //Set the speed of the follower
        follower.wrapMode = SplineFollower.Wrap.PingPong; //Set the wrap mode of the
follower
        follower.onBeginningReached += OnBeginningReached; //Subscribe to the
onBeginningReached event
        follower.onEndReached += OnEndReached; //Subscribe to the onEndReached event
       }

    void OnBeginningReached()
    {
        Debug.Log("Beginning reached");
        follower.motion.offset = Vector2.up;
    }

    void OnEndReached()
    {
        Debug.Log("End reached");
        follower.motion.offset = Vector2.down;
    }
```

Public Properties

Wrap wrapMode	Defines what happens when the follower reaches the end of the spline
FollowMode followMode	The follow mode of the follower. It can either follow with

	uniform speed or follow based on a duration variable
bool autoFollow	If true, the SplineFollower will automatically move along the spline during runtime
double startPosition	The start position along the spline when autoFollow is checked
bool autoStartPosition	If set to true and autoFollow is on, the follower will disregard the startPercent value and find the closest point on the spline via projection
float followSpeed	if followMode is set to FollowMode.Uniform, this will be the constant follow speed
float followDuration	if followMode is set to FollowMode.Time this will be the time in which the follower will reach the end of the spline
bool applyDirectionRotation	If set to true and applyRotation is true, the orientation will point in the direction of movement

Public Methods

void Move(double percent)	Move along the spline with percent
void Move(float distance)	Move a certain distance along the spline
void AddTrigger(SplineTrigger.Type t, UnityAction call, double position = 0.0, SplineTrigger.Type type = SplineTrigger.Type.Double)	Add a trigger with a parameterless action
void Restart(double startPosition = 0.0)	Restarts the follower and positions it at the given percent [0-1]

Enumerations

enum FollowMode { Uniform, Time }	A follow mode enumeration defining whether the follower should use a uniform follow speed or use time for traversing
enum Wrap { Default, Loop, PingPong}	Wrap mode for the follower. Default stops when the follower reaches the end. Loop moves the follower to the opposite end of the spline when the end is

	reached and continues in the same direction. PingPong changes the direction when the end is reached.

TransformModule

Class in Dreamteck.Splines

Description

A module used by the SplineTracer class to control how the motion is applied to the object

Public Properties

SplineResult splineResult	The spline result object used to determine the ideal position, rotation and scale. It's passed by value.
Vector2 offset;	An offset vector to apply to the final position
Vector3 rotationOffset	A rotation offset to apply to the final rotation
Vector3 baseScale	The base scale that will be multiplied by the splineResult.size value
bool applyPositionX	Should position be applied along the world-X axis?
bool applyPositionY	Should position be applied along the world-Y axis?
bool applyPositionZ	Should position be applied along the world-Z axis?
bool applyPosition	Should position be applied? Will return true if position is set to be applied along at least one axis.
bool applyRotationX	Should rotation be applied around the world-X axis?
bool applyRotationY	Should rotation be applied around the world-Y axis?
bool applyRotationZ	Should rotation be applied around the world-Z axis?
bool applyRotation	Should rotation be applied? Will return true if rotation is set to be applied around at least one axis.
CustomRotationModule customRotation	A custom rotation module to be used to offset the rotation even further

Spline.Direction direction	The direction of the rotation, if set to Spline.Direction.Backward the rotation will turn at 180 degrees
bool applyScaleX	Should scale be applied along the local-X axis?
bool applyScaleY	Should scale be applied along the local-Y axis?
bool applyScaleZ	Should scale be applied along the local-Z axis?
bool applyScale	Should scale be applied? Will return true if scale is set to be applied along at least one axis.

Public Methods

void ApplyTransform(Transform input)	Applies the motion to the referenced Transform component
void ApplyRigidbody(Rigidbody input)	Applies the motion to the referenced Rigidbody component
void ApplyRigidbody2D(Rigidbody2D input)	Applies the motion to the referenced Rigidbody2D component

CustomOffsetModule

Class in Dreamteck.Splines

Description

A module used by the SplineTracer class to create custom offset regions

Public Properties

float blend	Controls the blending amount [0-1] of the offset module. 0 applies no offsets and 1 applies full offsets
List<CustomOffsetModule.Key> keys	The list of custom offset regions this module has

Public Methods

AddKey(Vector2 offset, double f, double t, double c)	Adds a new offset region with offset “offset” starting at f, ending at t with a center c
Vector2 Evaluate(double time)	Returns the offset at the given time along the spline (based on the keys)

CustomOffsetModule.Key

Class in Dreamteck.Splines

Description

The key of the CustomOffsetModule.

Public Properties

Vector2 offset	The offset
double from	The start of the custom offset region

double to	The end of the custom offset region
double center	The center of the custom offset region where the offset is applied with full power. This value is local to the range [from-to] meaning that 0 equals from and 1 equals to. 0.5 produces the middle of the custom offset region
bool loop	Should the offset region be looped?
double position	Gets and sets the position of the region based on its center. Whenever the position is set, the from and to values are also moved to maintain the region’s size.

CustomRotationModule

Class in Dreamteck.Splines

Description

A module used by the SplineTracer class to create custom rotation regions

Public Properties

float blend	Controls the blending amount [0-1] of the offset module. 0 applies no offsets and 1 applies full offsets
List<CustomRotationModule.Key> keys	The list of custom offset regions this module has

Public Methods

AddKey(Vector2 offset, double f, double t, double c)	Adds a new offset region with offset “offset” starting at f, ending at t with a center c
Vector2 Evaluate(double time)	Returns the offset at the given time along the spline (based on the keys)

CustomRotationModule.Key

Class in Dreamteck.Splines

Description

The key of the CustomRotationModule

Public Properties

Vector3 `rotation`	The rotation offset in Euler angles
double from	The start of the custom offset region

double to	The end of the custom offset region
double center	The center of the custom offset region where the offset is applied with full power. This value is local to the range [from-to] meaning that 0 equals from and 1 equals to. 0.5 produces the middle of the custom offset region
bool loop	Should the offset region be looped?
double position	Gets and sets the position of the region based on its center. Whenever the position is set, the from and to values are also moved to maintain the region's size.

SplinePrimitive

Class in Dreamteck.Splines.Primitives

Description

The base class which all procedural primitives in Dreamteck Splines inherit from

Public Properties

SplinePrimitive.axis axis	The axis, along which the primitive will be facing
Vector3 offset	The offset of the primitive
Vector3 rotation	The rotation of the primitive in Euler angles

Public Methods

Spline GetSpline()	Creates and returns a Spline object based on the primitive logic
void UpdateSpline(Spline spline)	Writes the primitive spline to an existing spline object
SplineComputer CreateSplineComputer(string name, Vector3 position, Quaternion rotation)	Creates and returns a spline computer with a name, position and rotation in the scene based on the primitive logic
void UpdateSplineComputer(SplineComputer comp)	Writes the primitive spline to an existing SplineComputer

Enumerations

enum Axis { X, Y, Z, nX, nY, nZ }	A definition of an axis for the spline primitives

Ellipse

Class in Dreamteck.Splines.Primitives

Description

Creates a Bezier ellipse spline

Example

```
using UnityEngine;
using Dreamteck.Splines.Primitives;

public class EllipseCreator : MonoBehaviour {
       void Start () {
        Ellipse ellipsePrimitive = new Ellipse();
        ellipsePrimitive.xRadius = 2f;
        ellipsePrimitive.yRadius = 1f;
        ellipsePrimitive.CreateSplineComputer("My ellipse", Vector3.zero,
Quaternion.identity);
        ellipsePrimitive.xRadius = 1.5f;
        ellipsePrimitive.yRadius = 1.5f;
        ellipsePrimitive.CreateSplineComputer("My circle", Vector3.right * 5f,
Quaternion.identity);
       }
}
```

Public Properties

float xRadius	The horizontal radius of the ellipse
float yRadius	The vertical radius of the ellipse

Capsule

Class in Dreamteck.Splines.Primitives

Description

Creates a Bezier capsule spline

Public Properties

float radius	The radius of the capsule
float height	The height of the capsule

Line

Class in Dreamteck.Splines.Primitives

Description

Creates a Linear line spline

Public Properties

float length	The length of the spline
int segments	The number of segments between the points of the spline.
bool mirror	Should the line be mirrored

Ngon

Class in Dreamteck.Splines.Primitives

Description

Creates a Linear n-gon spline

Public Properties

float radius	The radius of the n-gon
int sides	The number of sides of the n-gon

Rectangle

Class in Dreamteck.Splines.Primitives

Description

Creates a Linear rectangle spline

Public Properties

Vector2 size	The size of the rectangle

RoundedRectangle

Class in Dreamteck.Splines.Primitives

Description

Creates a Linear rectangle spline

Public Properties

Vector2 size	The size of the rectangle
float xRadius	The horizontal radius of the rectangle’s edge
Float yRadius	The vertical radius of the rectangle’s edge

Spiral

Class in Dreamteck.Splines.Primitives

Description

Creates a Bezier spiral spline

Public Properties

float startRadius	The radius of the spiral at the beginning
float endRadius	The radius of the spline at the end
float stretch	How much the spiral is stretched
int iterations	The number of iterations of the spiral
AnimationCurve curve	A curve object for controlling the radius interpolation

Star

Class in Dreamteck.Splines.Primitives

Description

Creates a Linear star spline

Public Properties

float radius	The radius of the star
float depth	How much the star’s beams poke out of the shape
int sides	The number of beams the star has

SVG

Struct in Dreamteck.Splines.IO

Description

The class that handles importing and exporting Dreamteck Splines from and to SVG files. When an SVG object is created it can load an SVG file from a path or write existing splines to an SVG file.

Example

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

public class ImportExport : MonoBehaviour {
    public string exportPath = "C:/Users/USERNAME/Desktop/graphic.svg";
    public string importPath = "C:/Users/USERNAME/Desktop/exported.svg";
    public List<SplineComputer> computerReferences = new List<SplineComputer>();
       void Start () {
        SVG exporter = new SVG(computerReferences);
        exporter.Write(exportPath, SVG.Axis.Z); //Export the projection of the splines in 
the Z plane
        SVG importer = new SVG(importPath);
        List<SplineComputer> importedComputers = 
importer.CreateSplineComputers(Vector3.zero, Quaternion.identity, 
SVG.Element.All);//Import all elements from the graphic at importPath
       }
}
```

This example demonstrates the use of the SVG object for both exporting and importing splines

Public Methods

void Write(string filePath, SVG.Axis ax = Axis.Z)	Writes the contents to an SVG file. The axis parameter defines the plane on which the spline will be projected in order to be saved as a 2D graphic.
List<Spline> CreateSplines(SVG.Element elements = SVG.Element.All)	Creates an array of Spline objects from the loaded SVG graphic. This should be called after a file has been read. The elements argument provides filtering functionality in case only certain SVG elements need to be imported (for example, only ellipses)
List<SplineComputer> CreateSplineComputers(Vector3	Same as CreateSplines but this method creates

position, Quaternion rotation, Element elements = Element.All)	SplineComputer objects in the scene. The created objects are automatically named and returned as a list

Enumerations

Axis {X, Y, Z}	A projection axis for the SVG import/export functionality
Element { All, Path, Polygon, Ellipse, Rectangle, Line }	An enumeration that servers as a filter for import of splines from a file.

Constructors

SVG(string filePath)	Creates a new SVG IMPORTER from a file. Use this to import SVG files.
SVG(List<SplineComputer> computers)	Creates a new SVG EXPORTER from a list of SplineComputers. Use this to export SVG files.

CSV

Struct in Dreamteck.Splines.IO

Description

The class that handles importing and exporting Dreamteck Splines from and to CSV dataset files. When an CSV object is created it can load an CSV file from a path or write existing splines to an CSV file.

Public Methods

void Write(string filePath)	Writes the contents to an CSV file. The CSV file must contain a single spline
Spline CreateSpline()	Creates a Spline object from the loaded CSV dataset.
SplineComputer CreateSplineComputer(Vector3 position, Quaternion rotation)	Same as CreateSpline but this method creates a SplineComputer object in the scene. The created object is automatically named and returned

Enumerations

ColumnType { Position, Tangent, Tangent2, Normal, Size, Color }	An enumeration which defines the type of a single column in the CSV dataset. Usually used in a list to define how to read from or write to CSV diles

Constructors

CSV(string filePath)	Creates a new CSV IMPORTER from a file. Use this to import CSV files.
CSV(SplineComputer computer, List<CSV.ColumnType> customColumns = null)	Creates a new CSV EXPORTER for a single SplineComputer. Use this to export CSV files.