

Fractal Trees With Springs

0.1

Gandhi Games

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Namespace Documentation	7
4.1	FractalTree Namespace Reference	7
4.2	FractalTree.Demo Namespace Reference	8
5	Class Documentation	9
5.1	FractalTree.Branch Interface Reference	9
5.1.1	Detailed Description	10
5.1.2	Member Function Documentation	10
5.1.2.1	DoBranching< T >()	10
5.1.2.2	DoColonizationReset()	11
5.1.2.3	Setup() [1 / 4]	11
5.1.2.4	Setup() [2 / 4]	11
5.1.2.5	Setup() [3 / 4]	12
5.1.2.6	Setup() [4 / 4]	12
5.1.3	Property Documentation	12
5.1.3.1	colonizationDir	13

5.1.3.2	colonizationLeafCount	13
5.1.3.3	endPos	13
5.1.3.4	hasBranched	13
5.1.3.5	startPos	13
5.1.3.6	transform	13
5.2	FractalTree.ColonizationLeaf Class Reference	14
5.2.1	Detailed Description	14
5.2.2	Property Documentation	14
5.2.2.1	hasBeenReached	14
5.2.2.2	position	14
5.3	FractalTree.ColonizationLeafGenerator Class Reference	15
5.3.1	Detailed Description	15
5.3.2	Member Data Documentation	15
5.3.2.1	numToCreate	15
5.4	FractalTree.ColonizationTree Class Reference	16
5.4.1	Detailed Description	16
5.4.2	Constructor & Destructor Documentation	16
5.4.2.1	ColonizationTree()	16
5.4.3	Member Function Documentation	17
5.4.3.1	Generate< T >()	17
5.5	FractalTree.DefaultTree Class Reference	17
5.5.1	Detailed Description	17
5.5.2	Constructor & Destructor Documentation	18
5.5.2.1	DefaultTree()	18
5.5.3	Member Function Documentation	18
5.5.3.1	Generate< T >()	18
5.6	FractalTree.Demo.DemoBranchTreeUI Class Reference	19
5.6.1	Detailed Description	19
5.7	FractalTree.Demo.DemoColonizationTreeUI Class Reference	20
5.7.1	Detailed Description	20

5.8	FractalTree.Demo.DemoControls Class Reference	20
5.9	FractalTree.Demo.DemoForceController Class Reference	21
5.9.1	Detailed Description	21
5.10	FractalTree.Demo.LeafPlacement Class Reference	22
5.10.1	Detailed Description	22
5.10.2	Member Function Documentation	22
5.10.2.1	Clear()	22
5.11	FractalTree.Demo.DemoLTreeUI Class Reference	22
5.11.1	Detailed Description	23
5.12	FractalTree.Demo.DemoSceneSwitcher Class Reference	23
5.13	FractalTree.Demo.DemoTreeCreator Class Reference	24
5.13.1	Member Data Documentation	24
5.13.1.1	leafGenerator	24
5.13.1.2	showingStationary	24
5.13.1.3	treeBuilders	25
5.13.2	Property Documentation	25
5.13.2.1	activeTree	25
5.14	FractalTree.Demo.DemoTreeData Class Reference	25
5.15	FractalTree.LRule Class Reference	25
5.16	FractalTree.LTree Class Reference	26
5.16.1	Detailed Description	26
5.16.2	Constructor & Destructor Documentation	26
5.16.2.1	LTree()	26
5.16.3	Member Function Documentation	27
5.16.3.1	Generate< T >()	27
5.17	FractalTree.MovingBranch Interface Reference	27
5.17.1	Detailed Description	28
5.17.2	Property Documentation	28
5.17.2.1	endPoint	28
5.17.2.2	startPoint	28

5.18 FractalTree.MovingBranchImpl Class Reference	29
5.18.1 Detailed Description	29
5.18.2 Member Function Documentation	30
5.18.2.1 DoBranching< T >()	30
5.18.2.2 Setup() [1 / 4]	30
5.18.2.3 Setup() [2 / 4]	31
5.18.2.4 Setup() [3 / 4]	31
5.18.2.5 Setup() [4 / 4]	32
5.18.3 Property Documentation	32
5.18.3.1 endPoint	32
5.18.3.2 endPos	32
5.18.3.3 startPoint	32
5.18.3.4 startPos	33
5.19 FractalTree.MovingTreeBuilder Class Reference	33
5.19.1 Detailed Description	34
5.19.2 Member Function Documentation	34
5.19.2.1 ApplyDirectedForce()	34
5.19.2.2 ApplyPullForce()	34
5.19.2.3 ApplyPushForce()	34
5.19.2.4 Build()	35
5.19.2.5 Remove()	35
5.19.3 Member Data Documentation	35
5.19.3.1 forceDistanceMulti	35
5.19.4 Property Documentation	35
5.19.4.1 branches	36
5.20 FractalTree.MovingTreeBuilderEditor Class Reference	36
5.21 FractalTree.PointMass Class Reference	36
5.21.1 Detailed Description	37
5.21.2 Constructor & Destructor Documentation	37
5.21.2.1 PointMass()	37

5.21.3	Member Function Documentation	37
5.21.3.1	ApplyForce()	37
5.21.3.2	DoUpdate()	38
5.21.3.3	IncreaseDamping()	38
5.21.4	Property Documentation	38
5.21.4.1	forceApplied	38
5.21.4.2	position	38
5.21.4.3	velocity	38
5.22	FractalTree.Spring Class Reference	39
5.22.1	Detailed Description	39
5.22.2	Member Function Documentation	39
5.22.2.1	DoUpdate()	39
5.22.2.2	Setup()	39
5.22.3	Member Data Documentation	40
5.22.3.1	end	40
5.22.3.2	start	40
5.23	FractalTree.StationaryBranch Class Reference	40
5.23.1	Detailed Description	42
5.23.2	Member Function Documentation	42
5.23.2.1	DoBranching< T >()	42
5.23.2.2	DoColonizationReset()	43
5.23.2.3	Setup() [1 / 4]	43
5.23.2.4	Setup() [2 / 4]	43
5.23.2.5	Setup() [3 / 4]	44
5.23.2.6	Setup() [4 / 4]	44
5.23.2.7	UpdateColor()	45
5.23.2.8	UpdateSprite()	45
5.23.3	Member Data Documentation	45
5.23.3.1	LengthDegradation	45
5.23.3.2	m_Renderer	46

5.23.3.3	m_Width	46
5.23.3.4	SPRITE_SIZE	46
5.23.4	Property Documentation	46
5.23.4.1	colonizationDir	46
5.23.4.2	colonizationLeafCount	46
5.23.4.3	color	46
5.23.4.4	endPos	47
5.23.4.5	hasBranched	47
5.23.4.6	startPos	47
5.24	FractalTree.StationaryTreeBuilder Class Reference	47
5.24.1	Detailed Description	48
5.24.2	Member Function Documentation	48
5.24.2.1	Build()	48
5.24.2.2	Remove()	48
5.24.3	Property Documentation	48
5.24.3.1	branches	48
5.25	FractalTree.StationaryTreeBuilderEditor Class Reference	49
5.26	FractalTree.Tree Interface Reference	49
5.26.1	Detailed Description	49
5.27	FractalTree.TreeBuilder Class Reference	50
5.27.1	Detailed Description	51
5.27.2	Member Enumeration Documentation	51
5.27.2.1	TreeType	51
5.27.3	Member Function Documentation	52
5.27.3.1	CreateTree()	52
5.27.3.2	DoBuild< T >()	52
5.27.4	Member Data Documentation	52
5.27.4.1	branchPrefab	52
5.27.4.2	buildOnStart	52
5.27.4.3	colonizationInitialLength	53

5.27.4.4	colonizationLeafParent	53
5.27.4.5	colonizationMaxDistance	53
5.27.4.6	colonizationMinDistance	53
5.27.4.7	colonizationWidth	53
5.27.4.8	defaultAngle	53
5.27.4.9	defaultBranchColour	54
5.27.4.10	defaultGrowthCount	54
5.27.4.11	defaultInitialLength	54
5.27.4.12	defaultLengthReductionOnNewGeneration	54
5.27.4.13	defaultWidth	54
5.27.4.14	ITreeAngle	54
5.27.4.15	ITreeAutoWidth	55
5.27.4.16	ITreeAxiom	55
5.27.4.17	ITreeBranchLength	55
5.27.4.18	ITreeColours	55
5.27.4.19	ITreeGrowthCount	55
5.27.4.20	ITreeMassBasedOnWidth	55
5.27.4.21	ITreeRules	56
5.27.4.22	ITreeWidth	56
5.27.4.23	treeType	56
5.28	FractalTree.TreeBuilderEditor Class Reference	56
5.28.1	Detailed Description	57
5.29	FractalTree.Demo.TreesToDemo Class Reference	57
5.29.1	Detailed Description	57
5.29.2	Member Function Documentation	58
5.29.2.1	BuildMoving()	58
5.29.2.2	BuildStationary()	58
5.29.2.3	Hide()	58
5.29.2.4	Show()	58
5.29.3	Member Data Documentation	58
5.29.3.1	movingTree	58
5.29.3.2	stationaryTree	59
5.29.4	Property Documentation	59
5.29.4.1	active	59

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

FractalTree	7
FractalTree.Demo	8

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

FractalTree.Branch	9
FractalTree.MovingBranch	27
FractalTree.MovingBranchImpl	29
FractalTree.StationaryBranch	40
FractalTree.MovingBranchImpl	29
Editor	
FractalTree.TreeBuilderEditor	56
FractalTree.MovingTreeBuilderEditor	36
FractalTree.StationaryTreeBuilderEditor	49
FractalTree.LRule	25
MonoBehaviour	
FractalTree.ColonizationLeaf	14
FractalTree.ColonizationLeafGenerator	15
FractalTree.Demo.DemoBranchTreeUI	19
FractalTree.Demo.DemoColonizationTreeUI	20
FractalTree.Demo.DemoControls	20
FractalTree.Demo.DemoForceController	21
FractalTree.Demo.DemoLTreeUI	22
FractalTree.Demo.DemoSceneSwitcher	23
FractalTree.Demo.DemoTreeCreator	24
FractalTree.Demo.DemoTreeData	25
FractalTree.DemoLeafPlacement	22
FractalTree.Spring	39
FractalTree.StationaryBranch	40
FractalTree.TreeBuilder	50
FractalTree.MovingTreeBuilder	33
FractalTree.StationaryTreeBuilder	47
FractalTree.PointMass	36
FractalTree.Tree	49
FractalTree.ColonizationTree	16
FractalTree.DefaultTree	17
FractalTree.LTree	26
FractalTree.Demo.TreesToDemo	57

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

FractalTree.Branch	Contract for all fractal tree branches. Includes positional data and initialisation.	9
FractalTree.ColonizationLeaf	Attach to leaf objects for space colonization. The branches move towards the leaves.	14
FractalTree.ColonizationLeafGenerator	Spawns a set number of leaves within a bounds. Used by space colonization.	15
FractalTree.ColonizationTree	Spawns a fractal tree using space colonization: http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf	16
FractalTree.DefaultTree	Spawns a fractal tree.	17
FractalTree.Demo.DemoBranchTreeUI	For demo purposes. Listens for changes in UI and updates Branch variables.	19
FractalTree.Demo.DemoColonizationTreeUI	For demo purposes. Listens for changes in UI and updates Space Colonization variables. . . .	20
FractalTree.Demo.DemoControls	20
FractalTree.Demo.DemoForceController	Applies forces to the currently active tree in the demo scene.	21
FractalTree.Demo.LeafPlacement	For demo purposes. Spawns a leaf object (used for space colonization tree algorithm) at mouse position on left-click.	22
FractalTree.Demo.DemoLTreeUI	For demo purposes. Listens for changes in UI and updates L-Tree variables.	22
FractalTree.Demo.DemoSceneSwitcher	23
FractalTree.Demo.DemoTreeCreator	24
FractalTree.Demo.DemoTreeData	25
FractalTree.LRule	25
FractalTree.LTree	Spawns a fractal true using the L-system: http://www.allenpike.com/modeling-plants-with-l-system 26	
FractalTree.MovingBranch	Extends branch with point data for moving branches.	27
FractalTree.MovingBranchImpl	Extends a normal branch and adds spring functionality. Force can be applied to the start and end point of the branch.	29

FractalTree.MovingTreeBuilder	Builds a moving tree and provides methods of applying forces to generated trees.	33
FractalTree.MovingTreeBuilderEditor	36
FractalTree.PointMass	Added to the start and end of movable branches. Used to add spring force to a branch.	36
FractalTree.Spring	Connects two point masses and applies a pull force to ensure points stay within a target length.	39
FractalTree.StationaryBranch	A stationary branch. Forces cannot be applied to it. It is a line drawn onscreen by rotating and scaling a sprite between a start and end point.	40
FractalTree.StationaryTreeBuilder	Builds a stationary tree.	47
FractalTree.StationaryTreeBuilderEditor	49
FractalTree.Tree	All trees should have a method of generating themselves.	49
FractalTree.TreeBuilder	The base tree builder class. Provides parameters for default, L, and colonization tree generation.	50
FractalTree.TreeBuilderEditor	Custom editor for tree builder class. Hides variables not in use based on TreeType.	56
FractalTree.Demo.TreesToDemo	Trees to demo. Stationary and Moving tree builder pairs.	57

Chapter 4

Namespace Documentation

4.1 FractalTree Namespace Reference

Namespaces

Classes

- interface [Branch](#)
Contract for all fractal tree branches. Includes positional data and initialisation.
- class [ColonizationLeaf](#)
Attach to leaf objects for space colonization. The branches move towards the leaves.
- class [ColonizationLeafGenerator](#)
Spawns a set number of leaves within a bounds. Used by space colonization.
- class [ColonizationTree](#)
Spawns a fractal tree using space colonization: [http://algorithmicbotany.org/papers/colonization.←egwnp2007.pdf](http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf)
- class [DefaultTree](#)
Spawns a fractal tree.
- class [DemoLeafPlacement](#)
For demo purposes. Spawns a leaf object (used for space colonization tree algorithm) at mouse position on left-click.
- class **Extensions**
Extension methods used by the Fractal [Tree](#) generator.
- class [LRule](#)
- class [LTree](#)
Spawns a fractal true using the L-system: <http://www.allenpike.com/modeling-plants-with-l-systems/>
- interface [MovingBranch](#)
Extends branch with point data for moving branches.
- class [MovingBranchImpl](#)
Extends a normal branch and adds spring functionality. Force can be applied to the start and end point of the branch.
- class [MovingTreeBuilder](#)
Builds a moving tree and provides methods of applying forces to generated trees.
- class [MovingTreeBuilderEditor](#)
- class [PointMass](#)
Added to the start and end of movable branches. Used to add spring force to a branch.
- class [Spring](#)

- Connects two point masses and applies a pull force to ensure points stay within a target length.*
- class [StationaryBranch](#)
A stationary branch. Forces cannot be applied to it. It is a line drawn onscreen by rotating and scaling a sprite between a start and end point.
- class [StationaryTreeBuilder](#)
Builds a stationary tree.
- class [StationaryTreeBuilderEditor](#)
- interface [Tree](#)
All trees should have a method of generating themselves.
- class [TreeBuilder](#)
The base tree builder class. Provides parameters for default, L, and colonization tree generation.
- class [TreeBuilderEditor](#)
Custom editor for tree builder class. Hides variables not in use based on TreeType.

4.2 FractalTree.Demo Namespace Reference

Classes

- class [DemoBranchTreeUI](#)
For demo purposes. Listens for changes in UI and updates [Branch](#) variables.
- class [DemoColonizationTreeUI](#)
For demo purposes. Listens for changes in UI and updates Space Colonization variables.
- class [DemoControls](#)
- class [DemoForceController](#)
Applies forces to the currently active tree in the demo scene.
- class [DemoLTreeUI](#)
For demo purposes. Listens for changes in UI and updates L-Tree variables.
- class [DemoSceneSwitcher](#)
- class [DemoTreeCreator](#)
- class [DemoTreeData](#)
- class [TreesToDemo](#)
Trees to demo. Stationary and Moving tree builder pairs.

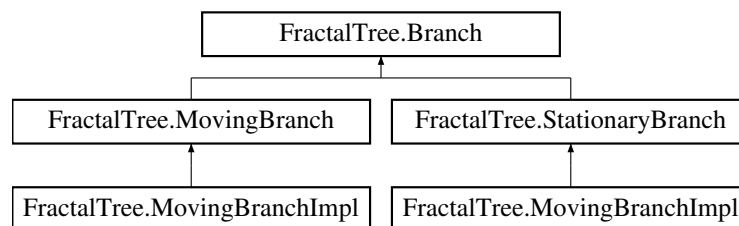
Chapter 5

Class Documentation

5.1 FractalTree.Branch Interface Reference

Contract for all fractal tree branches. Includes positional data and initialisation.

Inheritance diagram for FractalTree.Branch:



Public Member Functions

- void **Setup** (Branch owner, Vector2 end, float thickness, Color color)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.
- void **Setup** (Branch owner, Vector2 end, float thickness, Color color, bool autoMass)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.
- void **Setup** (Vector2 start, Vector2 end, float thickness, Color color)
Setup the specified start, end, thickness and color. Creates a stand alone branch that is not connected to any other branch.
- void **Setup** (Vector2 start, Vector2 end, float width, Color color, bool autoMass)
Setup the specified start, end, thickness and color. Creates a stand alone branch that is not connected to any other branch that has its mass autogenerated based on line width.
- T **DoBranching**< T > (float angle)
Returns a new branch based on current branch angle plus parameter angle.
- void **DoColonizationReset** ()
Resets the colonization paramater. Used only for space colonization generation.

Properties

- Vector2 [startPos](#) [get]
Gets the start position.
- Vector2 [endPos](#) [get]
Gets the end position.
- Vector2 [colonizationDir](#) [get, set]
Gets or sets the colonization direction. Used for space colonization tree generation. Defines the direction of the next branch in relation to nearby leaves.
- int [colonizationLeafCount](#) [get, set]
Gets or sets the number of nearby colonizaion leaves.
- bool [hasBranched](#) [get, set]
Gets or sets a value indicating whether this [FractalTree.Branch](#) has branched.
- Transform [transform](#) [get]
Gets the transform.

5.1.1 Detailed Description

Contract for all fractal tree branches. Includes positional data and initialisation.

5.1.2 Member Function Documentation

5.1.2.1 DoBranching< T >()

```
T FractalTree.Branch.DoBranching< T > (
    float angle )
```

Returns a new branch based on current branch angle plus parameter angle.

Returns

The branching.

Parameters

<i>angle</i>	Angle.
--------------	--------

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Implemented in [FractalTree.StationaryBranch](#), and [FractalTree.MovingBranchImpl](#).

Type Constraints

***T* : [Branch](#)**

5.1.2.2 DoColonizationReset()

```
void FractalTree.Branch.DoColonizationReset ( )
```

Resets the colonization paramater. Used only for space colonization generation.

Implemented in [FractalTree.StationaryBranch](#).

5.1.2.3 Setup() [1/4]

```
void FractalTree.Branch.Setup (
    Branch owner,
    Vector2 end,
    float thickness,
    Color color )
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.

Parameters

<i>owner</i>	The attached branch.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.

Implemented in [FractalTree.StationaryBranch](#), and [FractalTree.MovingBranchImpl](#).

5.1.2.4 Setup() [2/4]

```
void FractalTree.Branch.Setup (
    Branch owner,
    Vector2 end,
    float thickness,
    Color color,
    bool autoMass )
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.

Parameters

<i>owner</i>	Owner.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>autoMass</i>	If set to true auto mass.

Implemented in [FractalTree.StationaryBranch](#), and [FractalTree.MovingBranchImpl](#).

5.1.2.5 Setup() [3/4]

```
void FractalTree.Branch.Setup (
    Vector2 start,
    Vector2 end,
    float thickness,
    Color color )
```

Setup the specified start, end, thickness and color. Creates a stand alone branch that is not connected to any other branch.

Parameters

<i>start</i>	Start.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.

Implemented in [FractalTree.StationaryBranch](#), and [FractalTree.MovingBranchImpl](#).

5.1.2.6 Setup() [4/4]

```
void FractalTree.Branch.Setup (
    Vector2 start,
    Vector2 end,
    float width,
    Color color,
    bool autoMass )
```

Setup the specified start, end, thickness and color. Creates a stand alone branch that is not connected to any other branch that has its mass autogenerated based on line width.

Parameters

<i>start</i>	Start.
<i>end</i>	End.
<i>width</i>	Width.
<i>color</i>	Color.
<i>autoMass</i>	If set to <code>true</code> auto mass.

Implemented in [FractalTree.StationaryBranch](#), and [FractalTree.MovingBranchImpl](#).

5.1.3 Property Documentation

5.1.3.1 colonizationDir

```
Vector2 FractalTree.Branch.colonizationDir [get], [set]
```

Gets or sets the colonization direction. Used for space colonization tree generation. Defines the direction of the next branch in relation to nearby leaves.

The colonization dir.

5.1.3.2 colonizationLeafCount

```
int FractalTree.Branch.colonizationLeafCount [get], [set]
```

Gets or sets the number of nearby colonizaion leaves.

The colonization leaf count.

5.1.3.3 endPos

```
Vector2 FractalTree.Branch.endPos [get]
```

Gets the end position.

The end position.

5.1.3.4 hasBranched

```
bool FractalTree.Branch.hasBranched [get], [set]
```

Gets or sets a value indicating whether this [FractalTree.Branch](#) has branched.

true if has branched; otherwise, false.

5.1.3.5 startPos

```
Vector2 FractalTree.Branch.startPos [get]
```

Gets the start position.

The start position.

5.1.3.6 transform

```
Transform FractalTree.Branch.transform [get]
```

Gets the transform.

The transform.

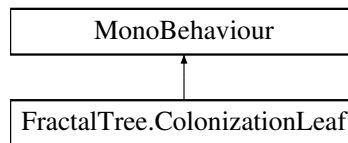
The documentation for this interface was generated from the following file:

- FT/Scripts/Branch/Branch.cs

5.2 FractalTree.ColonizationLeaf Class Reference

Attach to leaf objects for space colonization. The branches move towards the leaves.

Inheritance diagram for FractalTree.ColonizationLeaf:



Properties

- bool `hasBeenReached` [get, set]
Within the minimum distance of a branch. To be removed from the simulation.
- Vector2 `position` [get]
Gets the position of the leaf.

5.2.1 Detailed Description

Attach to leaf objects for space colonization. The branches move towards the leaves.

5.2.2 Property Documentation

5.2.2.1 hasBeenReached

`bool FractalTree.ColonizationLeaf.hasBeenReached` [get], [set]

Within the minimum distance of a branch. To be removed from the simulation.

true if has been reached; otherwise, false.

5.2.2.2 position

`Vector2 FractalTree.ColonizationLeaf.position` [get]

Gets the position of the leaf.

The position.

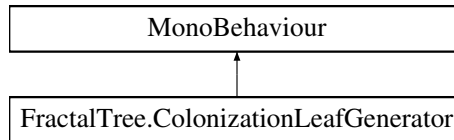
The documentation for this class was generated from the following file:

- FT/Scripts/Trees/ColonizationLeaf.cs

5.3 FractalTree.ColonizationLeafGenerator Class Reference

Spawns a set number of leaves within a bounds. Used by space colonization.

Inheritance diagram for FractalTree.ColonizationLeafGenerator:



Public Member Functions

- void **Generate** ()
- void **Remove** ()

Public Attributes

- float **radius** = 6f
- int **numToCreate** = 100
The number of leaves to spawn.
- bool **buildOnStart** = false

5.3.1 Detailed Description

Spawns a set number of leaves within a bounds. Used by space colonization.

5.3.2 Member Data Documentation

5.3.2.1 numToCreate

```
int FractalTree.ColonizationLeafGenerator.numToCreate = 100
```

The number of leaves to spawn.

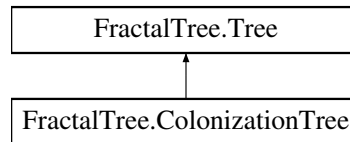
The documentation for this class was generated from the following file:

- FT/Scripts/Trees/ColonizationLeafGenerator.cs

5.4 FractalTree.ColonizationTree Class Reference

Spawns a fractal tree using space colonization: <http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf>

Inheritance diagram for FractalTree.ColonizationTree:



Public Member Functions

- **ColonizationTree** (List< [ColonizationLeaf](#) > leaves, Transform owner, float initialLength, GameObject branchPrefab, float width, float minDistance, float maxDistance, Color branchColor)
Initializes a new instance of the [FractalTree.ColonizationTree](#) class.
- List< T > **Generate**< T > ()
Generates a tree using space colonization.

5.4.1 Detailed Description

Spawns a fractal tree using space colonization: <http://algorithmicbotany.org/papers/colonization.egwnp2007.pdf>

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ColonizationTree()

```

FractalTree.ColonizationTree.ColonizationTree (
    List< ColonizationLeaf > leaves,
    Transform owner,
    float initialLength,
    GameObject branchPrefab,
    float width,
    float minDistance,
    float maxDistance,
    Color branchColor )
  
```

Initializes a new instance of the [FractalTree.ColonizationTree](#) class.

Parameters

<i>leaves</i>	Leaves.
<i>owner</i>	Owner.
<i>initialLength</i>	Initial length.
<i>branchPrefab</i>	Branch prefab.
<i>width</i>	Width.
<i>minDistance</i>	Minimum distance.
<i>maxDistance</i>	Max distance.

5.4.3 Member Function Documentation

5.4.3.1 Generate< T >()

```
List<T> FractalTree.ColonizationTree.Generate< T > ( )
```

Generates a tree using space colonization.

Template Parameters

<i>T</i>	Branch type.
----------	--------------

Implements [FractalTree.Tree](#).

Type Constraints

***T* : Branch**

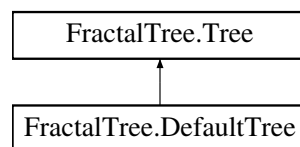
The documentation for this class was generated from the following file:

- FT/Scripts/Trees/ColonizationTree.cs

5.5 FractalTree.DefaultTree Class Reference

Spawns a fractal tree.

Inheritance diagram for FractalTree.DefaultTree:



Public Member Functions

- [DefaultTree](#) (int growth, float initialLength, float lengthDegredation, float angle, float thickness, GameObject branchPrefab, Color branchColor, Transform owner)
Initializes a new instance of the [FractalTree.DefaultTree](#) class.
- List< T > [Generate< T > \(\)](#)
Generates a fractal tree.

5.5.1 Detailed Description

Spawns a fractal tree.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 DefaultTree()

```
FractalTree.DefaultTree.DefaultTree (
    int growth,
    float initialLength,
    float lengthDegredation,
    float angle,
    float thickness,
    GameObject branchPrefab,
    Color branchColor,
    Transform owner )
```

Initializes a new instance of the [FractalTree.DefaultTree](#) class.

Parameters

<i>growth</i>	Growth.
<i>initialLength</i>	Initial length.
<i>lengthDegredation</i>	Length degredation.
<i>angle</i>	Angle.
<i>thickness</i>	Thickness.
<i>branchPrefab</i>	Branch prefab.
<i>owner</i>	Owner.

5.5.3 Member Function Documentation

5.5.3.1 Generate< T >()

```
List<T> FractalTree.DefaultTree.Generate< T > ( )
```

Generates a fractal tree.

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Implements [FractalTree.Tree](#).

Type Constraints

***T* : [Branch](#)**

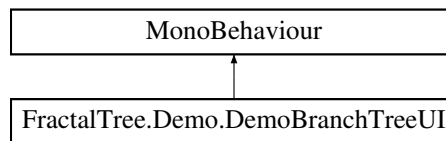
The documentation for this class was generated from the following file:

- FT/Scripts/Trees/DefaultTree.cs

5.6 FractalTree.Demo.DemoBranchTreeUI Class Reference

For demo purposes. Listens for changes in UI and updates [Branch](#) variables.

Inheritance diagram for FractalTree.Demo.DemoBranchTreeUI:



Public Member Functions

- void **OnGenerationChange** (string value)
- void **OnLengthChanged** (string value)
- void **OnMultiplierChanged** (string value)
- void **OnAngleChanged** (string value)
- void **OnWidthChanged** (string value)
- void **Generate** ()

Public Attributes

- [TreeBuilder](#) **treeBuilder**
- InputField **genInput**
- InputField **lengthInput**
- InputField **multiplierInput**
- InputField **angleInput**
- InputField **widthInput**

5.6.1 Detailed Description

For demo purposes. Listens for changes in UI and updates [Branch](#) variables.

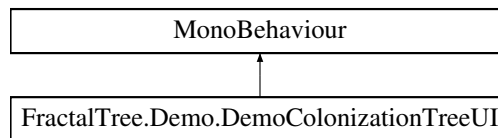
The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoBranchTreeUI.cs

5.7 FractalTree.Demo.DemoColonizationTreeUI Class Reference

For demo purposes. Listens for changes in UI and updates Space Colonization variables.

Inheritance diagram for FractalTree.Demo.DemoColonizationTreeUI:



Public Attributes

- [TreeBuilder](#) **treeBuilder**
- [DemoLeafPlacement](#) **leafPlacement**
- InputField **lengthInput**
- InputField **widthInput**
- InputField **minDistanceToLeafInput**
- InputField **maxDistanceToLeafInput**
- Button **clearButton**
- Button **generateButton**

5.7.1 Detailed Description

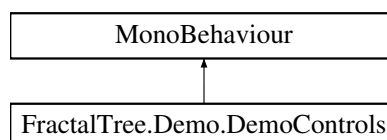
For demo purposes. Listens for changes in UI and updates Space Colonization variables.

The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoColonizationTreeUI.cs

5.8 FractalTree.Demo.DemoControls Class Reference

Inheritance diagram for FractalTree.Demo.DemoControls:



Public Member Functions

- void **OnRadiusChanged** (string value)
- void **OnPushChanged** (string value)
- void **OnPullChanged** (string value)
- void **OnNextTreePressed** ()
- void **OnChangeTreeStatePressed** ()

Public Attributes

- GameObject **controls**
- [DemoForceController](#) **forceController**
- [DemoTreeCreator](#) **treeCreator**
- InputField **radiusInput**
- InputField **pushInput**
- InputField **pullInput**
- Text **stationaryButtonText**
- Text **warningLabel**

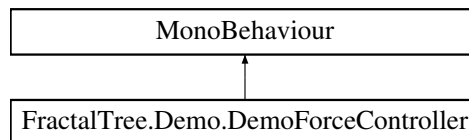
The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoControls.cs

5.9 FractalTree.Demo.DemoForceController Class Reference

Applies forces to the currently active tree in the demo scene.

Inheritance diagram for FractalTree.Demo.DemoForceController:



Public Attributes

- [DemoTreeCreator](#) **treeCreator**
- float **radius** = 10
- float **pushForce** = 2f
- float **pullForce** = 0.1f
- float **directedForce** = 0.5f

5.9.1 Detailed Description

Applies forces to the currently active tree in the demo scene.

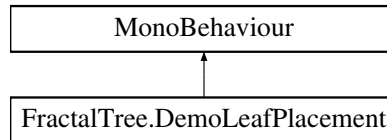
The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoForceController.cs

5.10 FractalTree.DemoLeafPlacement Class Reference

For demo purposes. Spawns a leaf object (used for space colonization tree algorithm) at mouse position on left-click.

Inheritance diagram for FractalTree.DemoLeafPlacement:



Public Member Functions

- void [Clear](#) ()
Remove all child leaves.

Public Attributes

- GameObject **leafPrefab**

5.10.1 Detailed Description

For demo purposes. Spawns a leaf object (used for space colonization tree algorithm) at mouse position on left-click.

5.10.2 Member Function Documentation

5.10.2.1 Clear()

```
void FractalTree.DemoLeafPlacement.Clear ( )
```

Remove all child leaves.

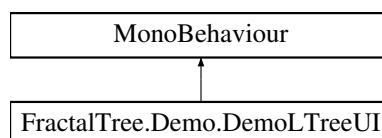
The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoLeafPlacement.cs

5.11 FractalTree.Demo.DemoLTreeUI Class Reference

For demo purposes. Listens for changes in UI and updates L-Tree variables.

Inheritance diagram for FractalTree.Demo.DemoLTreeUI:



Public Attributes

- [TreeBuilder](#) **treeBuilder**
- Toggle **autoWidthToggle**
- InputField **widthInput**
- InputField **genInput**
- InputField **lengthInput**
- InputField **angleInput**
- InputField **axiomInput**
- InputField [] **ruleInputs**
- Button **generate**

5.11.1 Detailed Description

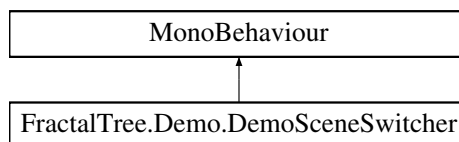
For demo purposes. Listens for changes in UI and updates L-Tree variables.

The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoLTreeUI.cs

5.12 FractalTree.Demo.DemoSceneSwitcher Class Reference

Inheritance diagram for FractalTree.Demo.DemoSceneSwitcher:



Public Member Functions

- void **LoadNextScene** ()

Public Attributes

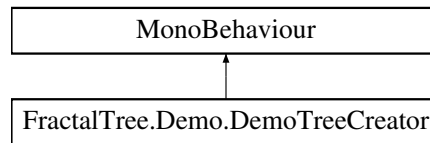
- int **numOfScenes** = 2

The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoSceneSwitcher.cs

5.13 FractalTree.Demo.DemoTreeCreator Class Reference

Inheritance diagram for FractalTree.Demo.DemoTreeCreator:



Public Member Functions

- void **ShowNextTree** ()
- bool **SwitchTreeState** ()

Public Attributes

- [TreesToDemo](#) [] [treeBuilders](#)
A list of stationary and moving tree pairs with helper methods to switch between them.
- [ColonizationLeafGenerator](#) [leafGenerator](#)
A leaf generator used for space colonization trees.
- bool [showingStationary](#) = true
Showing a static or moving version of the tree.
- int **startIndex** = 0

Properties

- [TreeBuilder](#) [activeTree](#) [get]
Gets the active tree or null if there is none.

5.13.1 Member Data Documentation

5.13.1.1 leafGenerator

[ColonizationLeafGenerator](#) FractalTree.Demo.DemoTreeCreator.leafGenerator

A leaf generator used for space colonization trees.

5.13.1.2 showingStationary

bool FractalTree.Demo.DemoTreeCreator.showingStationary = true

Showing a static or moving version of the tree.

5.13.1.3 treeBuilders

`TreesToDemo [] FractalTree.Demo.DemoTreeCreator.treeBuilders`

A list of stationary and moving tree pairs with helper methods to switch between them.

5.13.2 Property Documentation

5.13.2.1 activeTree

`TreeBuilder FractalTree.Demo.DemoTreeCreator.activeTree [get]`

Gets the active tree or null if there is none.

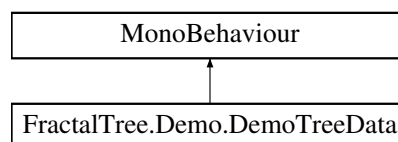
The active tree.

The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoTreeCreator.cs

5.14 FractalTree.Demo.DemoTreeData Class Reference

Inheritance diagram for FractalTree.Demo.DemoTreeData:



Public Attributes

- `DemoTreeCreator treeCreator`

The documentation for this class was generated from the following file:

- FT/Scripts/Demo/DemoTreeData.cs

5.15 FractalTree.LRule Class Reference

Public Member Functions

- `LRule (char from, string to)`

Public Attributes

- char **from**
- string **to**

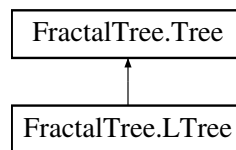
The documentation for this class was generated from the following file:

- FT/Scripts/Trees/LTree.cs

5.16 FractalTree.LTree Class Reference

Spawns a fractal tree using the L-system: <http://www.allenpike.com/modeling-plants-with-l-systems/>

Inheritance diagram for FractalTree.LTree:



Public Member Functions

- **LTree** (GameObject branchPrefab, int steps, string axiom, **LRule**[] rules, float branchLength, float angle, Transform owner, Color[] colors, float width, bool autoWidth, bool autoMass)
*Initializes a new instance of the **FractalTree.LTree** class.*
- List< T > **Generate**< T > ()
Generates the tree.

5.16.1 Detailed Description

Spawns a fractal tree using the L-system: <http://www.allenpike.com/modeling-plants-with-l-systems/>

5.16.2 Constructor & Destructor Documentation

5.16.2.1 LTree()

```

FractalTree.LTree.LTree (
    GameObject branchPrefab,
    int steps,
    string axiom,
    LRule [] rules,
    float branchLength,
    float angle,
    Transform owner,
    Color [] colors,
    float width,
    bool autoWidth,
    bool autoMass )
  
```

Initializes a new instance of the **FractalTree.LTree** class.

Parameters

<i>branchPrefab</i>	Branch prefab.
<i>steps</i>	Steps.
<i>axiom</i>	Axiom.
<i>rules</i>	Rules.
<i>initialLength</i>	Initial length.
<i>angle</i>	Angle.
<i>owner</i>	Owner.
<i>colors</i>	Colors.
<i>width</i>	Width.
<i>autoWidth</i>	If set to <code>true</code> auto width.
<i>autoMass</i>	If set to <code>true</code> auto mass.

5.16.3 Member Function Documentation

5.16.3.1 Generate< T >()

```
List<T> FractalTree.LTree.Generate< T > ( )
```

Generates the tree.

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Implements [FractalTree.Tree](#).

Type Constraints

***T* : [Branch](#)**

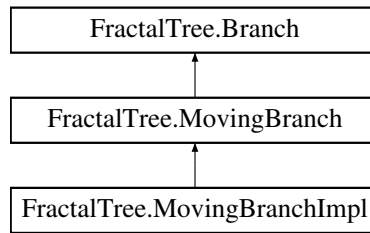
The documentation for this class was generated from the following file:

- FT/Scripts/Trees/LTree.cs

5.17 FractalTree.MovingBranch Interface Reference

Extends branch with point data for moving branches.

Inheritance diagram for FractalTree.MovingBranch:



Properties

- [PointMass startPoint](#) [get]
Gets the start point mass. Used to add spring force
- [PointMass endPoint](#) [get]
Gets the end point mass. Used to add spring force.

Additional Inherited Members

5.17.1 Detailed Description

Extends branch with point data for moving branches.

5.17.2 Property Documentation

5.17.2.1 endPoint

[PointMass](#) `FractalTree.MovingBranch.endPoint` [get]

Gets the end point mass. Used to add spring force.

The end point.

5.17.2.2 startPoint

[PointMass](#) `FractalTree.MovingBranch.startPoint` [get]

Gets the start point mass. Used to add spring force

The start point.

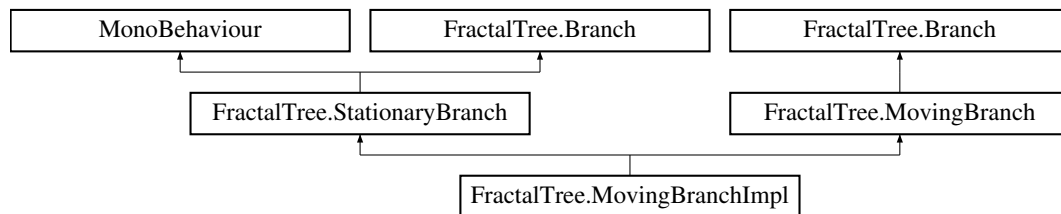
The documentation for this interface was generated from the following file:

- FT/Scripts/Branch/Branch.cs

5.18 FractalTree.MovingBranchImpl Class Reference

Extends a normal branch and adds spring functionality. Force can be applied to the start and end point of the branch.

Inheritance diagram for FractalTree.MovingBranchImpl:



Public Member Functions

- override void **Setup** (Branch owner, Vector2 end, float thickness, Color color)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.
- override void **Setup** (Branch owner, Vector2 end, float thickness, Color color, bool autoMass)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.
- override void **Setup** (Vector2 start, Vector2 end, float thickness, Color color)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.
- override void **Setup** (Vector2 start, Vector2 end, float width, Color color, bool autoMass)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.
- new T **DoBranching**< T > (float angle)
Returns a new branch based on current branch angle plus parameter angle.

Protected Member Functions

- override void **Awake** ()

Properties

- PointMass **startPoint** [get]
Gets the start point mass. Used to add spring force
- PointMass **endPoint** [get]
Gets the end point mass. Used to add spring force.
- override Vector2 **startPos** [get]
Gets the start position.
- override Vector2 **endPos** [get]
Gets the end position.

Additional Inherited Members

5.18.1 Detailed Description

Extends a normal branch and adds spring functionality. Force can be applied to the start and end point of the branch.

5.18.2 Member Function Documentation

5.18.2.1 DoBranching< T >()

```
new T FractalTree.MovingBranchImpl.DoBranching< T > (
    float angle )
```

Returns a new branch based on current branch angle plus parameter angle.

Returns

The branching.

Parameters

<i>angle</i>	Angle.
--------------	--------

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Implements [FractalTree.Branch](#).

Type Constraints

T* : *Branch

5.18.2.2 Setup() [1/4]

```
override void FractalTree.MovingBranchImpl.Setup (
    Branch owner,
    Vector2 end,
    float thickness,
    Color color ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.

Parameters

<i>owner</i>	The attached branch.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.

Reimplemented from [FractalTree.StationaryBranch](#).

5.18.2.3 Setup() [2/4]

```
override void FractalTree.MovingBranchImpl.Setup (
    Branch owner,
    Vector2 end,
    float thickness,
    Color color,
    bool autoMass ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.

Parameters

<i>owner</i>	Owner.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>autoMass</i>	If set to <code>true</code> auto mass.

Reimplemented from [FractalTree.StationaryBranch](#).

5.18.2.4 Setup() [3/4]

```
override void FractalTree.MovingBranchImpl.Setup (
    Vector2 start,
    Vector2 end,
    float thickness,
    Color color ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.

Parameters

<i>owner</i>	The attached branch.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>start</i>	Start.

Reimplemented from [FractalTree.StationaryBranch](#).

5.18.2.5 Setup() [4/4]

```
override void FractalTree.MovingBranchImpl.Setup (
    Vector2 start,
    Vector2 end,
    float width,
    Color color,
    bool autoMass ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.

Parameters

<i>owner</i>	Owner.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>start</i>	Start.
<i>width</i>	Width.
<i>autoMass</i>	If set to <code>true</code> auto mass.

Reimplemented from [FractalTree.StationaryBranch](#).

5.18.3 Property Documentation

5.18.3.1 endPoint

```
PointMass FractalTree.MovingBranchImpl.endPoint [get]
```

Gets the end point mass. Used to add spring force.

The end point.

5.18.3.2 endPos

```
override Vector2 FractalTree.MovingBranchImpl.endPos [get]
```

Gets the end position.

The end position.

5.18.3.3 startPoint

```
PointMass FractalTree.MovingBranchImpl.startPoint [get]
```

Gets the start point mass. Used to add spring force

The start point.

5.18.3.4 startPos

```
override Vector2 FractalTree.MovingBranchImpl.startPos [get]
```

Gets the start position.

The start position.

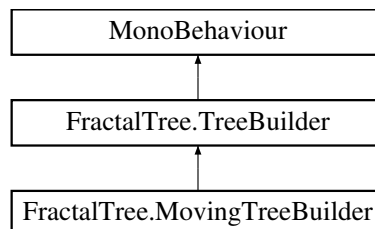
The documentation for this class was generated from the following file:

- FT/Scripts/Branch/MovingBranchImpl.cs

5.19 FractalTree.MovingTreeBuilder Class Reference

Builds a moving tree and provides methods of applying forces to generated trees.

Inheritance diagram for FractalTree.MovingTreeBuilder:



Public Member Functions

- override void **Build** ()
Build this instance.
- override void **Remove** ()
Deletes all child branches and clears branch list.
- void **ApplyDirectedForce** (Vector2 force, Vector2 position, float radius)
Applies a directed force to all branches within range.
- void **ApplyPushForce** (float force, Vector2 position, float radius)
Applies a push force to all branches within range.
- void **ApplyPullForce** (float force, Vector2 position, float radius)
Applies a pull force to all branches in range.

Public Attributes

- float **forceDistanceMulti** = 4f
All forces applies to this tree are multiplied by this value. Use this to create trees that are more sensitive to forces than others.

Properties

- List< **MovingBranch** > **branches** [get]
A list of all branches associated with the tree.

Additional Inherited Members

5.19.1 Detailed Description

Builds a moving tree and provides methods of applying forces to generated trees.

5.19.2 Member Function Documentation

5.19.2.1 ApplyDirectedForce()

```
void FractalTree.MovingTreeBuilder.ApplyDirectedForce (
    Vector2 force,
    Vector2 position,
    float radius )
```

Applies a directed force to all branches within range.

Parameters

<i>force</i>	Force.
<i>position</i>	Position.
<i>radius</i>	Radius.

5.19.2.2 ApplyPullForce()

```
void FractalTree.MovingTreeBuilder.ApplyPullForce (
    float force,
    Vector2 position,
    float radius )
```

Applies a pull force to all branches in range.

Parameters

<i>force</i>	Force.
<i>position</i>	Position.
<i>radius</i>	Radius.

5.19.2.3 ApplyPushForce()

```
void FractalTree.MovingTreeBuilder.ApplyPushForce (
    float force,
```

```
Vector2 position,  
float radius )
```

Applies a push force to all branches within range.

Parameters

<i>force</i>	Force.
<i>position</i>	Position.
<i>radius</i>	Radius.

5.19.2.4 Build()

```
override void FractalTree.MovingTreeBuilder.Build ( ) [virtual]
```

Build this instance.

Implements [FractalTree.TreeBuilder](#).

5.19.2.5 Remove()

```
override void FractalTree.MovingTreeBuilder.Remove ( ) [virtual]
```

Deletes all child branches and clears branch list.

Implements [FractalTree.TreeBuilder](#).

5.19.3 Member Data Documentation

5.19.3.1 forceDistanceMulti

```
float FractalTree.MovingTreeBuilder.forceDistanceMulti = 4f
```

All forces applies to this tree are multiplied by this value. Use this to create trees that are more sensitive to forces than others.

5.19.4 Property Documentation

5.19.4.1 branches

`List<MovingBranch> FractalTree.MovingTreeBuilder.branches [get]`

A list of all branches associated with the tree.

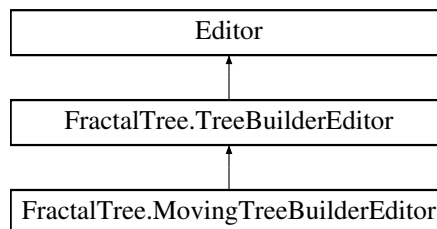
The branches.

The documentation for this class was generated from the following file:

- FT/Scripts/Tree Builders/MovingTreeBuilder.cs

5.20 FractalTree.MovingTreeBuilderEditor Class Reference

Inheritance diagram for FractalTree.MovingTreeBuilderEditor:



Public Member Functions

- override void **OnInspectorGUI** ()

Additional Inherited Members

The documentation for this class was generated from the following file:

- FT/Scripts/Editor/MovingTreeBuilderEditor.cs

5.21 FractalTree.PointMass Class Reference

Added to the start and end of movable branches. Used to add spring force to a branch.

Public Member Functions

- **PointMass** (Vector2 **position**, float invMass, float bounceBackForce)
Initializes a new instance of the [FractalTree.PointMass](#) class.
- void **ApplyForce** (Vector2 force)
Applies a force.
- void **IncreaseDamping** (float factor)
Increases the damping factor. This dampens the velocity each step.
- void **DoUpdate** ()
Updates position based on current force and distance from initial position.

Properties

- Vector2 [position](#) [get, set]
The current position of the branch point.
- Vector2 [velocity](#) [get]
Gets the velocity.
- bool [forceApplied](#) [get]
Gets a value indicating whether this [FractalTree.PointMass](#) has had a force applied.

5.21.1 Detailed Description

Added to the start and end of movable branches. Used to add spring force to a branch.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 PointMass()

```
FractalTree.PointMass.PointMass (
    Vector2 position,
    float invMass,
    float bounceBackForce )
```

Initializes a new instance of the [FractalTree.PointMass](#) class.

Parameters

<i>position</i>	Initial position.
<i>invMass</i>	Inverse mass, lower numbers result in more force required to move the point.
<i>bounceBackForce</i>	Bounce back force. The force applied when moving the spring back to its initial position.

5.21.3 Member Function Documentation

5.21.3.1 ApplyForce()

```
void FractalTree.PointMass.ApplyForce (
    Vector2 force )
```

Applies a force.

Parameters

<i>force</i>	Force.
--------------	--------

5.21.3.2 DoUpdate()

```
void FractalTree.PointMass.DoUpdate ( )
```

Updates position based on current force and distance from initial position.

5.21.3.3 IncreaseDamping()

```
void FractalTree.PointMass.IncreaseDamping (
    float factor )
```

Increases the damping factor. This dampens the velocity each step.

Parameters

<i>factor</i>	Factor.
---------------	---------

5.21.4 Property Documentation

5.21.4.1 forceApplied

```
bool FractalTree.PointMass.forceApplied [get]
```

Gets a value indicating whether this [FractalTree.PointMass](#) has had a force applied.

true if force applied; otherwise, false.

5.21.4.2 position

```
Vector2 FractalTree.PointMass.position [get], [set]
```

The current position of the branch point.

The position.

5.21.4.3 velocity

```
Vector2 FractalTree.PointMass.velocity [get]
```

Gets the velocity.

The velocity.

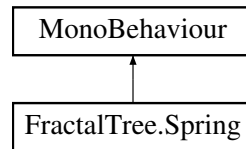
The documentation for this class was generated from the following file:

- FT/Scripts/Springs/PointMass.cs

5.22 FractalTree.Spring Class Reference

Connects two point masses and applies a pull force to ensure points stay within a target length.

Inheritance diagram for FractalTree.Spring:



Public Member Functions

- void [Setup](#) ([PointMass start](#), [PointMass end](#), float stiffness, float damping)
Setup the specified start, end, stiffness and damping.
- void [DoUpdate](#) ()
Applies force to start and point based on distance between points.

Public Attributes

- [PointMass start](#)
The start point mass.
- [PointMass end](#)
The end point mass.

5.22.1 Detailed Description

Connects two point masses and applies a pull force to ensure points stay within a target length.

5.22.2 Member Function Documentation

5.22.2.1 DoUpdate()

```
void FractalTree.Spring.DoUpdate ( )
```

Applies force to start and point based on distance between points.

5.22.2.2 Setup()

```
void FractalTree.Spring.Setup (
    PointMass start,
    PointMass end,
    float stiffness,
    float damping )
```

Setup the specified start, end, stiffness and damping.

Parameters

<i>start</i>	Start.
<i>end</i>	End.
<i>stiffness</i>	Stiffness.
<i>damping</i>	Damping.

5.22.3 Member Data Documentation

5.22.3.1 end

`PointMass FractalTree.Spring.end`

The end point mass.

5.22.3.2 start

`PointMass FractalTree.Spring.start`

The start point mass.

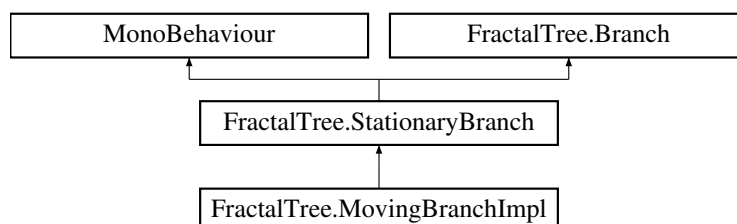
The documentation for this class was generated from the following file:

- FT/Scripts/Springs/Spring.cs

5.23 FractalTree.StationaryBranch Class Reference

A stationary branch. Forces cannot be applied to it. It is a line drawn onscreen by rotating and scaling a sprite between a start and end point.

Inheritance diagram for `FractalTree.StationaryBranch`:



Public Member Functions

- virtual void [Setup](#) ([Branch](#) owner, Vector2 end, float thickness, Color [color](#))
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.
- virtual void [Setup](#) ([Branch](#) owner, Vector2 end, float thickness, Color [color](#), bool autoMass)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.
- virtual void [Setup](#) (Vector2 start, Vector2 end, float thickness, Color [color](#))
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.
- virtual void [Setup](#) (Vector2 start, Vector2 end, float thickness, Color [color](#), bool autoMass)
Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.
- T [DoBranching](#)< T > (float angle)
Returns a new branch based on current branch angle plus parameter angle.
- void [DoColonizationReset](#) ()
Resets the colonization paramater. Used only for space colonization generation.

Static Public Attributes

- static float [LengthDegradation](#) = 0.67f
Used by the default tree algorithm. Each branchings length is multiplied by this value.

Protected Member Functions

- virtual void **Awake** ()
- void [UpdateSprite](#) ()
Updates the sprite position, rotation, and scale in relation to the start and point.
- void [UpdateColor](#) (Color [color](#))
Updates the sprite renderer color.

Protected Attributes

- float [m_Width](#)
The width of the branch.
- SpriteRenderer [m_Renderer](#)
The renderer associated with the branch.

Static Protected Attributes

- static readonly float [SPRITE_SIZE](#) = 100f / 100f
Pixels of line sprite / pixels per units.

Properties

- Vector2 [colonizationDir](#) [get, set]
Gets or sets the colonization direction. Used for space colonization tree generation. Defines the direction of the next branch in relation to nearby leaves.
- int [colonizationLeafCount](#) [get, set]
Gets or sets the number of nearby colonizaion leaves.
- virtual Vector2 [startPos](#) [get]
Gets the start position.
- virtual Vector2 [endPos](#) [get]
Gets the end position.
- bool [hasBranched](#) [get, set]
Gets or sets a value indicating whether this [FractalTree.StationaryBranch](#) has branched.
- Color [color](#) [set]
Sets the color of the branch sprite and updates the sprite renderer.

5.23.1 Detailed Description

A stationary branch. Forces cannot be applied to it. It is a line drawn onscreen by rotating and scaling a sprite between a start and end point.

5.23.2 Member Function Documentation

5.23.2.1 DoBranching< T >()

```
T FractalTree.StationaryBranch.DoBranching< T > (
    float angle )
```

Returns a new branch based on current branch angle plus parameter angle.

Returns

The branching.

Parameters

<i>angle</i>	Angle.
--------------	--------

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Implements [FractalTree.Branch](#).

Type Constraints

T : Branch

5.23.2.2 DoColonizationReset()

```
void FractalTree.StationaryBranch.DoColonizationReset ( )
```

Resets the colonization paramater. Used only for space colonization generation.

Implements [FractalTree.Branch](#).

5.23.2.3 Setup() [1/4]

```
virtual void FractalTree.StationaryBranch.Setup (
    Branch owner,
    Vector2 end,
    float thickness,
    Color color ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.

Parameters

<i>owner</i>	The attached branch.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.

Implements [FractalTree.Branch](#).

Reimplemented in [FractalTree.MovingBranchImpl](#).

5.23.2.4 Setup() [2/4]

```
virtual void FractalTree.StationaryBranch.Setup (
    Branch owner,
    Vector2 end,
    float thickness,
    Color color,
    bool autoMass ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.

Parameters

<i>owner</i>	Owner.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>autoMass</i>	If set to <code>true</code> auto mass.

Implements [FractalTree.Branch](#).

Reimplemented in [FractalTree.MovingBranchImpl](#).

5.23.2.5 `Setup()` [3/4]

```
virtual void FractalTree.StationaryBranch.Setup (
    Vector2 start,
    Vector2 end,
    float thickness,
    Color color ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch.

Parameters

<i>owner</i>	The attached branch.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>start</i>	Start.

Implements [FractalTree.Branch](#).

Reimplemented in [FractalTree.MovingBranchImpl](#).

5.23.2.6 `Setup()` [4/4]

```
virtual void FractalTree.StationaryBranch.Setup (
    Vector2 start,
    Vector2 end,
    float thickness,
    Color color,
    bool autoMass ) [virtual]
```

Setup the specified owner, end, thickness and color. Used to create a branch that is attached to another branch that has its mass autogenerated based on line width.

Parameters

<i>owner</i>	Owner.
<i>end</i>	End.
<i>thickness</i>	Thickness.
<i>color</i>	Color.
<i>start</i>	Start.
<i>autoMass</i>	If set to <code>true</code> auto mass.

Implements [FractalTree.Branch](#).

Reimplemented in [FractalTree.MovingBranchImpl](#).

5.23.2.7 UpdateColor()

```
void FractalTree.StationaryBranch.UpdateColor (
    Color color ) [protected]
```

Updates the sprite renderer color.

Parameters

<i>color</i>	Color.
--------------	--------

5.23.2.8 UpdateSprite()

```
void FractalTree.StationaryBranch.UpdateSprite ( ) [protected]
```

Updates the sprite position, rotation, and scale in relation to the start and point.

5.23.3 Member Data Documentation

5.23.3.1 LengthDegradation

```
float FractalTree.StationaryBranch.LengthDegradation = 0.67f [static]
```

Used by the default tree algorithm. Each branchings length is multiplied by this value.

5.23.3.2 m_Renderer

```
SpriteRenderer FractalTree.StationaryBranch.m_Renderer [protected]
```

The renderer associated with the branch.

5.23.3.3 m_Width

```
float FractalTree.StationaryBranch.m_Width [protected]
```

The width of the branch.

5.23.3.4 SPRITE_SIZE

```
readonly float FractalTree.StationaryBranch.SPRITE_SIZE = 100f / 100f [static], [protected]
```

Pixels of line sprite / pixels per units.

5.23.4 Property Documentation

5.23.4.1 colonizationDir

```
Vector2 FractalTree.StationaryBranch.colonizationDir [get], [set]
```

Gets or sets the colonization direction. Used for space colonization tree generation. Defines the direction of the next branch in relation to nearby leaves.

The colonization dir.

5.23.4.2 colonizationLeafCount

```
int FractalTree.StationaryBranch.colonizationLeafCount [get], [set]
```

Gets or sets the number of nearby colonizaion leaves.

The colonization leaf count.

5.23.4.3 color

```
Color FractalTree.StationaryBranch.color [set]
```

Sets the color of the branch sprite and updates the sprite renderer.

The color.

5.23.4.4 endPos

```
virtual Vector2 FractalTree.StationaryBranch.endPos [get]
```

Gets the end position.

The end position.

5.23.4.5 hasBranched

```
bool FractalTree.StationaryBranch.hasBranched [get], [set]
```

Gets or sets a value indicating whether this [FractalTree.StationaryBranch](#) has branched.

true if has branched; otherwise, false.

5.23.4.6 startPos

```
virtual Vector2 FractalTree.StationaryBranch.startPos [get]
```

Gets the start position.

The start position.

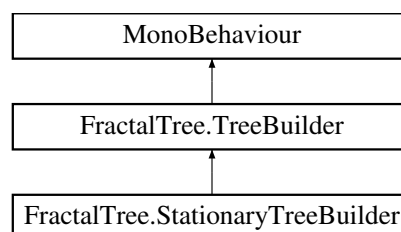
The documentation for this class was generated from the following file:

- FT/Scripts/Branch/StationaryBranch.cs

5.24 FractalTree.StationaryTreeBuilder Class Reference

Builds a stationary tree.

Inheritance diagram for FractalTree.StationaryTreeBuilder:



Public Member Functions

- override void [Build](#) ()
Build this instance.
- override void [Remove](#) ()
Deletes all child branches and clears branch list.

Properties

- List< [Branch](#) > [branches](#) [get]
A list of all branches associated with the tree.

Additional Inherited Members

5.24.1 Detailed Description

Builds a stationary tree.

5.24.2 Member Function Documentation

5.24.2.1 Build()

```
override void FractalTree.StationaryTreeBuilder.Build ( ) [virtual]
```

Build this instance.

Implements [FractalTree.TreeBuilder](#).

5.24.2.2 Remove()

```
override void FractalTree.StationaryTreeBuilder.Remove ( ) [virtual]
```

Deletes all child branches and clears branch list.

Implements [FractalTree.TreeBuilder](#).

5.24.3 Property Documentation

5.24.3.1 branches

```
List<Branch> FractalTree.StationaryTreeBuilder.branches [get]
```

A list of all branches associated with the tree.

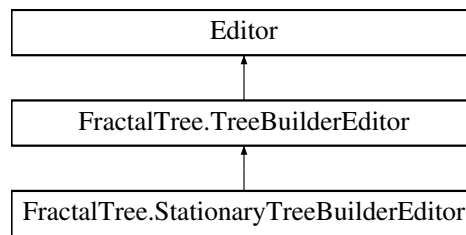
The branches.

The documentation for this class was generated from the following file:

- FT/Scripts/Tree Builders/StationaryTreeBuilder.cs

5.25 FractalTree.StationaryTreeBuilderEditor Class Reference

Inheritance diagram for FractalTree.StationaryTreeBuilderEditor:



Public Member Functions

- override void **OnInspectorGUI** ()

Additional Inherited Members

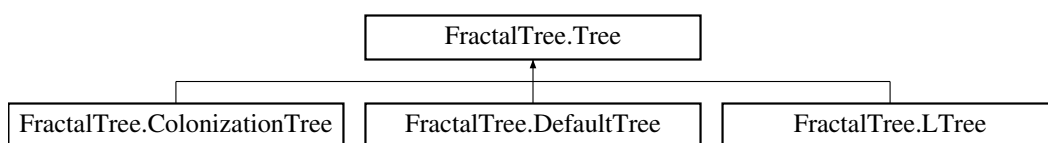
The documentation for this class was generated from the following file:

- FT/Scripts/Editor/StationaryTreeBuilderEditor.cs

5.26 FractalTree.Tree Interface Reference

All trees should have a method of generating themselves.

Inheritance diagram for FractalTree.Tree:



Public Member Functions

- List< T > **Generate**< T > ()

5.26.1 Detailed Description

All trees should have a method of generating themselves.

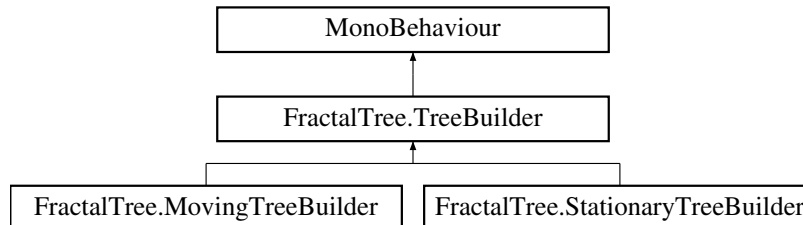
The documentation for this interface was generated from the following file:

- FT/Scripts/Tree Builders/TreeBuilder.cs

5.27 FractalTree.TreeBuilder Class Reference

The base tree builder class. Provides paramaters for default, L, and colonization tree generation.

Inheritance diagram for FractalTree.TreeBuilder:



Public Types

- enum [TreeType](#) { **Branching**, **LTree**, **Colonization** }
Tree type.

Public Member Functions

- abstract void **Build** ()
- abstract void **Remove** ()

Public Attributes

- bool [buildOnStart](#) = true
If true, builds tree on start.
- [TreeType](#) [treeType](#) = TreeType.Branching
The tree type to generate.
- GameObject [branchPrefab](#)
The branch prefab. If tree to generate is moving then prefab should have [MovingBranch](#) script attached.
- Color [defaultBranchColour](#) = Color.white
The branch colour for the default tree.
- int [defaultGrowthCount](#) = 8
The number of tree generations.
- float [defaultInitialLength](#) = 5f
The default length of the initial branches for the default tree generation.
- float [defaultLengthReductionOnNewGeneration](#) = 0.67f
The length degradation for the default tree. Branches are reduced in size by this factor.
- float [defaultAngle](#) = 45f
The angle for default tree branching.
- float [defaultWidth](#) = 0.04f
The width of the branches for the default tree generator.
- bool [ITreeAutoWidth](#) = true
When true, the width of the branches will be set automatically based on the colours.
- bool [ITreeMassBasedOnWidth](#) = true
When true, the mass of the branches will be set automatically based on colours. Used only when generating a moving tree.

- float `ITreeWidth` = 0.03f
The max branch width for L trees.
- int `ITreeGrowthCount` = 5
The number of L tree generations.
- string `ITreeAxiom` = "FX"
The l tree axiom. The initial seed used to generate a L tree.
- `LRule` [] `ITreeRules`
The rules applied to the axiom.
- float `ITreeBranchLength` = 0.17f
The length of the l tree branch.
- float `ITreeAngle` = 25f
The angles used to branch an L tree.
- Color [] `ITreeColours`
The L tree colours.
- Transform `colonizationLeafParent`
The parent of the game object that holds the colonization leaves.
- Color `colonizationBranchColor` = Color.white
- float `colonizationInitialLength` = 1f
The initial length for a colonization tree trunk.
- float `colonizationWidth` = 0.04f
The width of the colonization tree branches.
- float `colonizationMinDistance` = 1f
The minimum distance between the branch and a colonization leaf for it to be registered.
- float `colonizationMaxDistance` = 10f
The maximum distance between the branch and a colonization leaf for it to be registered.

Protected Member Functions

- List< T > `DoBuild< T > ()`
Build this instance of the tree.
- `Tree CreateTree ()`
Creates a tree based on treeType.

5.27.1 Detailed Description

The base tree builder class. Provides paramaters for default, L, and colonization tree generation.

5.27.2 Member Enumeration Documentation

5.27.2.1 TreeType

enum `FractalTree.TreeBuilder.TreeType` [strong]

`Tree` type.

5.27.3 Member Function Documentation

5.27.3.1 CreateTree()

```
Tree FractalTree.TreeBuilder.CreateTree ( ) [protected]
```

Creates a tree based on treeType.

Returns

The tree.

5.27.3.2 DoBuild< T >()

```
List<T> FractalTree.TreeBuilder.DoBuild< T > ( ) [protected]
```

Build this instance of the tree.

Type Constraints

T : Branch

5.27.4 Member Data Documentation

5.27.4.1 branchPrefab

```
GameObject FractalTree.TreeBuilder.branchPrefab
```

The branch prefab. If tree to generate is moving then prefab should have [MovingBranch](#) script attached.

5.27.4.2 buildOnStart

```
bool FractalTree.TreeBuilder.buildOnStart = true
```

If true, builds tree on start.

5.27.4.3 colonizationInitialLength

```
float FractalTree.TreeBuilder.colonizationInitialLength = 1f
```

The initial length for a colonization tree trunk.

5.27.4.4 colonizationLeafParent

```
Transform FractalTree.TreeBuilder.colonizationLeafParent
```

The parent of the game object that holds the colonization leaves.

5.27.4.5 colonizationMaxDistance

```
float FractalTree.TreeBuilder.colonizationMaxDistance = 10f
```

The maximum distance between the branch and a colonization leaf for it to be registered.

5.27.4.6 colonizationMinDistance

```
float FractalTree.TreeBuilder.colonizationMinDistance = 1f
```

The minimum distance between the branch and a colonization leaf for it to be registered.

5.27.4.7 colonizationWidth

```
float FractalTree.TreeBuilder.colonizationWidth = 0.04f
```

The width of the colonization tree branches.

5.27.4.8 defaultAngle

```
float FractalTree.TreeBuilder.defaultAngle = 45f
```

The angle for default tree branching.

5.27.4.9 defaultBranchColour

```
Color FractalTree.TreeBuilder.defaultBranchColour = Color.white
```

The branch colour for the default tree.

5.27.4.10 defaultGrowthCount

```
int FractalTree.TreeBuilder.defaultGrowthCount = 8
```

The number of tree generations.

5.27.4.11 defaultInitialLength

```
float FractalTree.TreeBuilder.defaultInitialLength = 5f
```

The default length of the initial branches for the default tree generation.

5.27.4.12 defaultLengthReductionOnNewGeneration

```
float FractalTree.TreeBuilder.defaultLengthReductionOnNewGeneration = 0.67f
```

The length degradation for the default tree. Branches are reduced in size by this factor.

5.27.4.13 defaultWidth

```
float FractalTree.TreeBuilder.defaultWidth = 0.04f
```

The width of the branches for the default tree generator.

5.27.4.14 lTreeAngle

```
float FractalTree.TreeBuilder.lTreeAngle = 25f
```

The angles used to branch an L tree.

5.27.4.15 lTreeAutoWidth

```
bool FractalTree.TreeBuilder.lTreeAutoWidth = true
```

When true, the width of the branches will be set automatically based on the colours.

5.27.4.16 lTreeAxiom

```
string FractalTree.TreeBuilder.lTreeAxiom = "FX"
```

The l tree axiom. The initial seed used to generate a L tree.

5.27.4.17 lTreeBranchLength

```
float FractalTree.TreeBuilder.lTreeBranchLength = 0.17f
```

The length of the l tree branch.

5.27.4.18 lTreeColours

```
Color [ ] FractalTree.TreeBuilder.lTreeColours
```

The L tree colours.

5.27.4.19 lTreeGrowthCount

```
int FractalTree.TreeBuilder.lTreeGrowthCount = 5
```

The number of L tree generations.

5.27.4.20 lTreeMassBasedOnWidth

```
bool FractalTree.TreeBuilder.lTreeMassBasedOnWidth = true
```

When true, the mass of the branches will be set automatically based on colours. Used only when generating a moving tree.

5.27.4.21 ITreeRules

`LRule [] FractalTree.TreeBuilder.lTreeRules`

Initial value:

```
= new LRule[] {

    new LRule ('F', "C0FF-[C1-F+F]+[C2+F-F]"),
    new LRule ('X', "C0FF+[C1+F]+[C3-F] ")
}
```

The rules applied to the axoim.

5.27.4.22 ITreeWidth

`float FractalTree.TreeBuilder.lTreeWidth = 0.03f`

The max branch width for L trees.

5.27.4.23 treeType

`TreeType FractalTree.TreeBuilder.treeType = TreeType.Branching`

The tree type to generate.

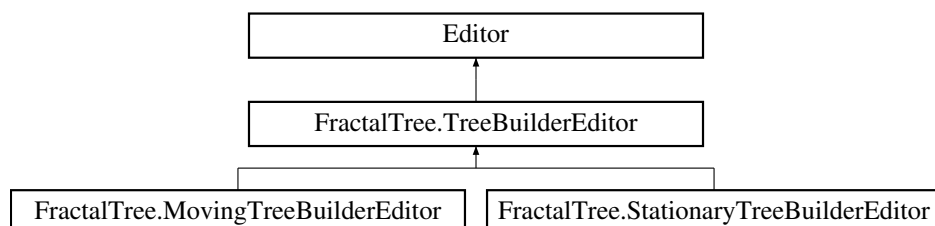
The documentation for this class was generated from the following file:

- FT/Scripts/Tree Builders/TreeBuilder.cs

5.28 FractalTree.TreeBuilderEditor Class Reference

Custom editor for tree builder class. Hides variables not in use based on TreeType.

Inheritance diagram for FractalTree.TreeBuilderEditor:



Protected Member Functions

- void **DrawEditor** ()

5.28.1 Detailed Description

Custom editor for tree builder class. Hides variables not in use based on TreeType.

The documentation for this class was generated from the following file:

- FT/Scripts/Editor/TreeBuilderEditor.cs

5.29 FractalTree.Demo.TreesToDemo Class Reference

Trees to demo. Stationary and Moving tree builder pairs.

Public Member Functions

- void **BuildStationary** ()
Builds the stationary tree and then disables game object.
- void **BuildMoving** ()
Builds the moving tree and then disables game object.
- void **Show** (bool showStationary)
Enables either the stationary or moving tree.
- void **Hide** ()
Disables both trees.

Public Attributes

- bool **preload**
- bool **built**
- **TreeBuilder** stationaryTree
The stationary tree.
- **TreeBuilder** movingTree
The moving tree.

Properties

- **TreeBuilder** active [get]
Gets the active tree builder.

5.29.1 Detailed Description

Trees to demo. Stationary and Moving tree builder pairs.

5.29.2 Member Function Documentation

5.29.2.1 BuildMoving()

```
void FractalTree.Demo.TreesToDemo.BuildMoving ( )
```

Builds the moving tree and then disables game object.

5.29.2.2 BuildStationary()

```
void FractalTree.Demo.TreesToDemo.BuildStationary ( )
```

Builds the stationary tree and then disables game object.

5.29.2.3 Hide()

```
void FractalTree.Demo.TreesToDemo.Hide ( )
```

Disables both trees.

5.29.2.4 Show()

```
void FractalTree.Demo.TreesToDemo.Show (
    bool showStationary )
```

Enables either the stationary or moving tree.

Parameters

<i>showStationary</i>	If set to <code>true</code> show stationary else show moving tree.
-----------------------	--

5.29.3 Member Data Documentation

5.29.3.1 movingTree

```
TreeBuilder FractalTree.Demo.TreesToDemo.movingTree
```

The moving tree.

5.29.3.2 stationaryTree

`TreeBuilder` `FractalTree.Demo.TreesToDemo.stationaryTree`

The stationary tree.

5.29.4 Property Documentation

5.29.4.1 active

`TreeBuilder` `FractalTree.Demo.TreesToDemo.active` `[get]`

Gets the active tree builder.

The active.

The documentation for this class was generated from the following file:

- `FT/Scripts/Demo/DemoTreeCreator.cs`

Index

active

FractalTree::Demo::TreesToDemo, [59](#)

activeTree

FractalTree::Demo::DemoTreeCreator, [25](#)

ApplyDirectedForce

FractalTree::MovingTreeBuilder, [34](#)

ApplyForce

FractalTree::PointMass, [37](#)

ApplyPullForce

FractalTree::MovingTreeBuilder, [34](#)

ApplyPushForce

FractalTree::MovingTreeBuilder, [34](#)

branchPrefab

FractalTree::TreeBuilder, [52](#)

branches

FractalTree::MovingTreeBuilder, [35](#)

FractalTree::StationaryTreeBuilder, [48](#)

Build

FractalTree::MovingTreeBuilder, [35](#)

FractalTree::StationaryTreeBuilder, [48](#)

BuildMoving

FractalTree::Demo::TreesToDemo, [58](#)

buildOnStart

FractalTree::TreeBuilder, [52](#)

BuildStationary

FractalTree::Demo::TreesToDemo, [58](#)

Clear

FractalTree::DemoLeafPlacement, [22](#)

colonizationDir

FractalTree::Branch, [12](#)

FractalTree::StationaryBranch, [46](#)

colonizationInitialLength

FractalTree::TreeBuilder, [52](#)

colonizationLeafCount

FractalTree::Branch, [13](#)

FractalTree::StationaryBranch, [46](#)

colonizationLeafParent

FractalTree::TreeBuilder, [53](#)

colonizationMaxDistance

FractalTree::TreeBuilder, [53](#)

colonizationMinDistance

FractalTree::TreeBuilder, [53](#)

ColonizationTree

FractalTree::ColonizationTree, [16](#)

colonizationWidth

FractalTree::TreeBuilder, [53](#)

color

FractalTree::StationaryBranch, [46](#)

CreateTree

FractalTree::TreeBuilder, [52](#)

defaultAngle

FractalTree::TreeBuilder, [53](#)

defaultBranchColour

FractalTree::TreeBuilder, [53](#)

defaultGrowthCount

FractalTree::TreeBuilder, [54](#)

defaultInitialLength

FractalTree::TreeBuilder, [54](#)

defaultLengthReductionOnNewGeneration

FractalTree::TreeBuilder, [54](#)

DefaultTree

FractalTree::DefaultTree, [18](#)

defaultWidth

FractalTree::TreeBuilder, [54](#)

DoBranching< T >

FractalTree::Branch, [10](#)

FractalTree::MovingBranchImpl, [30](#)

FractalTree::StationaryBranch, [42](#)

DoBuild< T >

FractalTree::TreeBuilder, [52](#)

DoColonizationReset

FractalTree::Branch, [11](#)

FractalTree::StationaryBranch, [43](#)

DoUpdate

FractalTree::PointMass, [38](#)

FractalTree::Spring, [39](#)

end

FractalTree::Spring, [40](#)

endPoint

FractalTree::MovingBranch, [28](#)

FractalTree::MovingBranchImpl, [32](#)

endPos

FractalTree::Branch, [13](#)

FractalTree::MovingBranchImpl, [32](#)

FractalTree::StationaryBranch, [46](#)

forceApplied

FractalTree::PointMass, [38](#)

forceDistanceMulti

FractalTree::MovingTreeBuilder, [35](#)

FractalTree, [7](#)

FractalTree.Branch, [9](#)

FractalTree.ColonizationLeaf, [14](#)

FractalTree.ColonizationLeafGenerator, [15](#)

FractalTree.ColonizationTree, [16](#)

FractalTree.DefaultTree, [17](#)

- FractalTree.Demo, 8
- FractalTree.Demo.DemoBranchTreeUI, 19
- FractalTree.Demo.DemoColonizationTreeUI, 20
- FractalTree.Demo.DemoControls, 20
- FractalTree.Demo.DemoForceController, 21
- FractalTree.Demo.DemoLTreeUI, 22
- FractalTree.Demo.DemoSceneSwitcher, 23
- FractalTree.Demo.DemoTreeCreator, 24
- FractalTree.Demo.DemoTreeData, 25
- FractalTree.Demo.TreesToDemo, 57
- FractalTree.DemoLeafPlacement, 22
- FractalTree.LRule, 25
- FractalTree.LTree, 26
- FractalTree.MovingBranch, 27
- FractalTree.MovingBranchImpl, 29
- FractalTree.MovingTreeBuilder, 33
- FractalTree.MovingTreeBuilderEditor, 36
- FractalTree.PointMass, 36
- FractalTree.Spring, 39
- FractalTree.StationaryBranch, 40
- FractalTree.StationaryTreeBuilder, 47
- FractalTree.StationaryTreeBuilderEditor, 49
- FractalTree.Tree, 49
- FractalTree.TreeBuilder, 50
- FractalTree.TreeBuilderEditor, 56
- FractalTree::Branch
 - colonizationDir, 12
 - colonizationLeafCount, 13
 - DoBranching< T >, 10
 - DoColonizationReset, 11
 - endPos, 13
 - hasBranched, 13
 - Setup, 11, 12
 - startPos, 13
 - transform, 13
- FractalTree::ColonizationLeaf
 - hasBeenReached, 14
 - position, 14
- FractalTree::ColonizationLeafGenerator
 - numToCreate, 15
- FractalTree::ColonizationTree
 - ColonizationTree, 16
 - Generate< T >, 17
- FractalTree::DefaultTree
 - DefaultTree, 18
 - Generate< T >, 18
- FractalTree::Demo::DemoTreeCreator
 - activeTree, 25
 - leafGenerator, 24
 - showingStationary, 24
 - treeBuilders, 24
- FractalTree::Demo::TreesToDemo
 - active, 59
 - BuildMoving, 58
 - BuildStationary, 58
 - Hide, 58
 - movingTree, 58
 - Show, 58
 - stationaryTree, 59
- FractalTree::DemoLeafPlacement
 - Clear, 22
- FractalTree::LTree
 - Generate< T >, 27
 - LTree, 26
- FractalTree::MovingBranch
 - endPoint, 28
 - startPoint, 28
- FractalTree::MovingBranchImpl
 - DoBranching< T >, 30
 - endPoint, 32
 - endPos, 32
 - Setup, 30, 31
 - startPoint, 32
 - startPos, 32
- FractalTree::MovingTreeBuilder
 - ApplyDirectedForce, 34
 - ApplyPullForce, 34
 - ApplyPushForce, 34
 - branches, 35
 - Build, 35
 - forceDistanceMulti, 35
 - Remove, 35
- FractalTree::PointMass
 - ApplyForce, 37
 - DoUpdate, 38
 - forceApplied, 38
 - IncreaseDamping, 38
 - PointMass, 37
 - position, 38
 - velocity, 38
- FractalTree::Spring
 - DoUpdate, 39
 - end, 40
 - Setup, 39
 - start, 40
- FractalTree::StationaryBranch
 - colonizationDir, 46
 - colonizationLeafCount, 46
 - color, 46
 - DoBranching< T >, 42
 - DoColonizationReset, 43
 - endPos, 46
 - hasBranched, 47
 - LengthDegradation, 45
 - m_Renderer, 45
 - m_Width, 46
 - SPRITE_SIZE, 46
 - Setup, 43, 44
 - startPos, 47
 - UpdateColor, 45
 - UpdateSprite, 45
- FractalTree::StationaryTreeBuilder
 - branches, 48
 - Build, 48
 - Remove, 48
- FractalTree::TreeBuilder

- branchPrefab, 52
- buildOnStart, 52
- colonizationInitialLength, 52
- colonizationLeafParent, 53
- colonizationMaxDistance, 53
- colonizationMinDistance, 53
- colonizationWidth, 53
- CreateTree, 52
- defaultAngle, 53
- defaultBranchColour, 53
- defaultGrowthCount, 54
- defaultInitialLength, 54
- defaultLengthReductionOnNewGeneration, 54
- defaultWidth, 54
- DoBuild< T >, 52
- ITreeAngle, 54
- ITreeAutoWidth, 54
- ITreeAxiom, 55
- ITreeBranchLength, 55
- ITreeColours, 55
- ITreeGrowthCount, 55
- ITreeMassBasedOnWidth, 55
- ITreeRules, 55
- ITreeWidth, 56
- TreeType, 51
- treeType, 56
- Generate< T >
 - FractalTree::ColonizationTree, 17
 - FractalTree::DefaultTree, 18
 - FractalTree::LTree, 27
- hasBeenReached
 - FractalTree::ColonizationLeaf, 14
- hasBranched
 - FractalTree::Branch, 13
 - FractalTree::StationaryBranch, 47
- Hide
 - FractalTree::Demo::TreesToDemo, 58
- IncreaseDamping
 - FractalTree::PointMass, 38
- LTree
 - FractalTree::LTree, 26
- ITreeAngle
 - FractalTree::TreeBuilder, 54
- ITreeAutoWidth
 - FractalTree::TreeBuilder, 54
- ITreeAxiom
 - FractalTree::TreeBuilder, 55
- ITreeBranchLength
 - FractalTree::TreeBuilder, 55
- ITreeColours
 - FractalTree::TreeBuilder, 55
- ITreeGrowthCount
 - FractalTree::TreeBuilder, 55
- ITreeMassBasedOnWidth
 - FractalTree::TreeBuilder, 55
- ITreeRules
 - FractalTree::TreeBuilder, 55
- ITreeWidth
 - FractalTree::TreeBuilder, 56
- leafGenerator
 - FractalTree::Demo::DemoTreeCreator, 24
- LengthDegradation
 - FractalTree::StationaryBranch, 45
- m_Renderer
 - FractalTree::StationaryBranch, 45
- m_Width
 - FractalTree::StationaryBranch, 46
- movingTree
 - FractalTree::Demo::TreesToDemo, 58
- numToCreate
 - FractalTree::ColonizationLeafGenerator, 15
- PointMass
 - FractalTree::PointMass, 37
- position
 - FractalTree::ColonizationLeaf, 14
 - FractalTree::PointMass, 38
- Remove
 - FractalTree::MovingTreeBuilder, 35
 - FractalTree::StationaryTreeBuilder, 48
- SPRITE_SIZE
 - FractalTree::StationaryBranch, 46
- Setup
 - FractalTree::Branch, 11, 12
 - FractalTree::MovingBranchImpl, 30, 31
 - FractalTree::Spring, 39
 - FractalTree::StationaryBranch, 43, 44
- Show
 - FractalTree::Demo::TreesToDemo, 58
- showingStationary
 - FractalTree::Demo::DemoTreeCreator, 24
- start
 - FractalTree::Spring, 40
- startPoint
 - FractalTree::MovingBranch, 28
 - FractalTree::MovingBranchImpl, 32
- startPos
 - FractalTree::Branch, 13
 - FractalTree::MovingBranchImpl, 32
 - FractalTree::StationaryBranch, 47
- stationaryTree
 - FractalTree::Demo::TreesToDemo, 59
- transform
 - FractalTree::Branch, 13
- treeBuilders
 - FractalTree::Demo::DemoTreeCreator, 24
- TreeType
 - FractalTree::TreeBuilder, 51
- treeType
 - FractalTree::TreeBuilder, 56

UpdateColor
 FractalTree::StationaryBranch, [45](#)
UpdateSprite
 FractalTree::StationaryBranch, [45](#)

velocity
 FractalTree::PointMass, [38](#)