

Fractal Spring Tree Generator

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
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	12
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	bounds	15
5.3.2.2	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.LRule Class Reference	19
5.7	FractalTree.LTree Class Reference	19
5.7.1	Detailed Description	19
5.7.2	Constructor & Destructor Documentation	20

5.7.2.1	LTree()	20
5.7.3	Member Function Documentation	20
5.7.3.1	Generate< T >()	20
5.8	FractalTree.MovingBranch Interface Reference	21
5.8.1	Detailed Description	21
5.8.2	Property Documentation	21
5.8.2.1	endPoint	21
5.8.2.2	startPoint	22
5.9	FractalTree.MovingBranchImpl Class Reference	22
5.9.1	Detailed Description	23
5.9.2	Member Function Documentation	23
5.9.2.1	DoBranching< T >()	23
5.9.2.2	Setup() [1/4]	24
5.9.2.3	Setup() [2/4]	24
5.9.2.4	Setup() [3/4]	24
5.9.2.5	Setup() [4/4]	25
5.9.3	Property Documentation	25
5.9.3.1	endPoint	25
5.9.3.2	endPos	25
5.9.3.3	startPoint	26
5.9.3.4	startPos	26
5.10	FractalTree.MovingTreeBuilder Class Reference	26
5.10.1	Member Function Documentation	27
5.10.1.1	ApplyDirectedForce()	27
5.10.1.2	ApplyPullForce()	27
5.10.1.3	ApplyPushForce()	27
5.10.1.4	Build()	28
5.10.2	Property Documentation	28
5.10.2.1	branches	28
5.11	FractalTree.PointMass Class Reference	28

5.11.1 Detailed Description	29
5.11.2 Constructor & Destructor Documentation	29
5.11.2.1 PointMass()	29
5.11.3 Member Function Documentation	29
5.11.3.1 ApplyForce()	29
5.11.3.2 DoUpdate()	29
5.11.3.3 IncreaseDamping()	29
5.11.4 Property Documentation	30
5.11.4.1 forceApplied	30
5.11.4.2 position	30
5.11.4.3 velocity	30
5.12 FractalTree.Spring Class Reference	30
5.12.1 Detailed Description	31
5.12.2 Member Function Documentation	31
5.12.2.1 DoUpdate()	31
5.12.2.2 Setup()	31
5.12.3 Member Data Documentation	31
5.12.3.1 end	32
5.12.3.2 start	32
5.13 FractalTree.StationaryBranch Class Reference	32
5.13.1 Detailed Description	33
5.13.2 Member Function Documentation	34
5.13.2.1 DoBranching< T >()	34
5.13.2.2 DoColonizationReset()	34
5.13.2.3 Setup() [1 / 4]	34
5.13.2.4 Setup() [2 / 4]	35
5.13.2.5 Setup() [3 / 4]	35
5.13.2.6 Setup() [4 / 4]	36
5.13.2.7 UpdateColor()	36
5.13.2.8 UpdateSprite()	36

5.13.3 Member Data Documentation	36
5.13.3.1 LengthDegradation	36
5.13.3.2 m_Renderer	37
5.13.3.3 m_Width	37
5.13.3.4 SPRITE_SIZE	37
5.13.4 Property Documentation	37
5.13.4.1 colonizationDir	37
5.13.4.2 colonizationLeafCount	37
5.13.4.3 color	37
5.13.4.4 endPos	38
5.13.4.5 hasBranched	38
5.13.4.6 startPos	38
5.14 FractalTree.StationaryTreeBuilder Class Reference	38
5.14.1 Detailed Description	39
5.14.2 Member Function Documentation	39
5.14.2.1 Build()	39
5.14.3 Property Documentation	39
5.14.3.1 branches	39
5.15 FractalTree.Tree Interface Reference	39
5.16 FractalTree.TreeBuilder Class Reference	40
5.16.1 Detailed Description	41
5.16.2 Member Enumeration Documentation	41
5.16.2.1 TreeType	41
5.16.3 Member Function Documentation	42
5.16.3.1 CreateTree()	42
5.16.3.2 DoBuild< T >()	42
5.16.4 Member Data Documentation	42
5.16.4.1 branchPrefab	42
5.16.4.2 buildOnStart	42
5.16.4.3 colonizationInitialLength	42

5.16.4.4	colonizationLeafParent	42
5.16.4.5	colonizationMaxDistance	43
5.16.4.6	colonizationMinDistance	43
5.16.4.7	colonizationWidth	43
5.16.4.8	defaultAngle	43
5.16.4.9	defaultGrowthCount	43
5.16.4.10	defaultInitialLength	43
5.16.4.11	defaultLengthDegradation	43
5.16.4.12	defaultWidth	43
5.16.4.13	ITreeAngle	44
5.16.4.14	ITreeAutoWidth	44
5.16.4.15	ITreeAxiom	44
5.16.4.16	ITreeBranchLength	44
5.16.4.17	ITreeColours	44
5.16.4.18	ITreeGrowthCount	44
5.16.4.19	ITreeMassBasedOnWidth	44
5.16.4.20	ITreeRules	45
5.16.4.21	ITreeWidth	45
5.16.4.22	treeType	45

Chapter 1

Namespace Index

1.1 Packages

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

FractalTree	7
---------------------------------------	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

FractalTree.Branch	9
FractalTree.MovingBranch	21
FractalTree.MovingBranchImpl	22
FractalTree.StationaryBranch	32
FractalTree.MovingBranchImpl	22
FractalTree.LRule	19
MonoBehaviour	
FractalTree.ColonizationLeaf	14
FractalTree.ColonizationLeafGenerator	15
FractalTree.Spring	30
FractalTree.StationaryBranch	32
FractalTree.TreeBuilder	40
FractalTree.MovingTreeBuilder	26
FractalTree.StationaryTreeBuilder	38
FractalTree.PointMass	28
FractalTree.Tree	39
FractalTree.ColonizationTree	16
FractalTree.DefaultTree	17
FractalTree.LTree	19

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.LRule	19
FractalTree.LTree	
Spawns a fractal true using the L-system: http://www.allenpike.com/modeling-plants-with-l-system 19	
FractalTree.MovingBranch	
Extends branch with point data for moving branches.	21
FractalTree.MovingBranchImpl	
Extends a normal branch and adds spring functionality. Force can be applied to the start and end point of the branch.	22
FractalTree.MovingTreeBuilder	26
FractalTree.PointMass	
Added to the start and end of movable branches. Used to add spring force to a branch.	28
FractalTree.Spring	
Connects two point masses and applies a pull force to ensure points stay within a target length.	30
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.	32
FractalTree.StationaryTreeBuilder	
Builds a stationary tree.	38
FractalTree.Tree	39
FractalTree.TreeBuilder	
The base tree builder class. Provides paramaters for default, L, and colonization tree generation.	40

Chapter 4

Namespace Documentation

4.1 FractalTree Namespace Reference

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 **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](#)
- 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.
- interface [Tree](#)
- class [TreeBuilder](#)
The base tree builder class. Provides paramaters for default, L, and colonization tree generation.

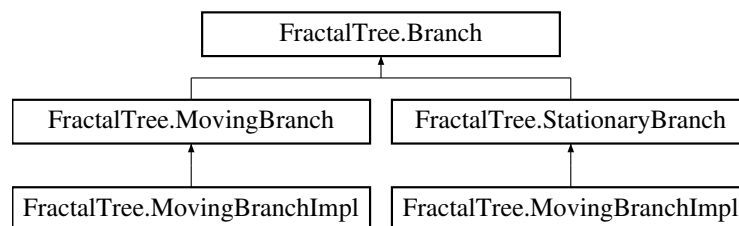
Chapter 5

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 <code>true</code> 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.

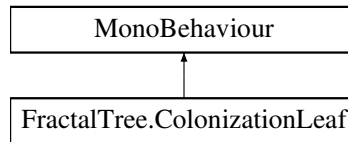
The documentation for this interface was generated from the following file:

- 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.

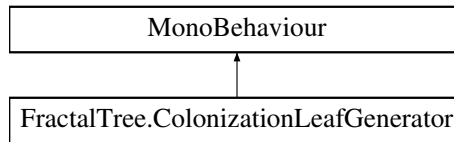
The documentation for this class was generated from the following file:

- 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 Attributes

- Rect `bounds`
The bounds within which to spawn the leafs.
- int `numToCreate` = 100
The number of leaves to spawn.

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 bounds

```
Rect FractalTree.ColonizationLeafGenerator.bounds
```

The bounds within which to spawn the leafs.

5.3.2.2 numToCreate

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

The number of leaves to spawn.

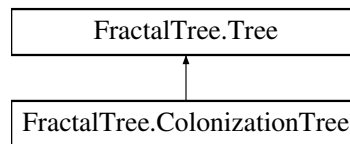
The documentation for this class was generated from the following file:

- 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)
*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 )
  
```

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

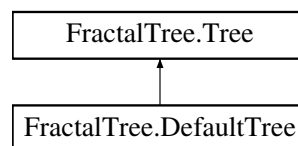
The documentation for this class was generated from the following file:

- 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, 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,
    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 degradation.
<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:

- DefaultTree.cs

5.6 FractalTree.LRule Class Reference

Public Member Functions

- **LRule** (char from, string to)

Public Attributes

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

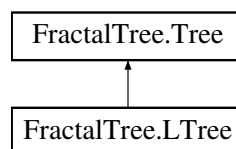
The documentation for this class was generated from the following file:

- LTree.cs

5.7 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 initialLength, 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.

5.7.1 Detailed Description

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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 LTree()

```
FractalTree.LTree.LTree (
    GameObject branchPrefab,
    int steps,
    string axiom,
    LRule [] rules,
    float initialLength,
    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.

5.7.3 Member Function Documentation

5.7.3.1 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](#)**

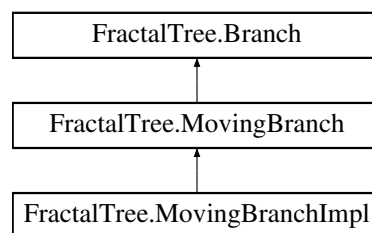
The documentation for this class was generated from the following file:

- LTree.cs

5.8 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

5.8.1 Detailed Description

Extends branch with point data for moving branches.

5.8.2 Property Documentation

5.8.2.1 endPoint

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

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

The end point.

5.8.2.2 startPoint

`PointMass FractalTree.MovingBranch.startPoint [get]`

Gets the start point mass. Used to add spring force

The start point.

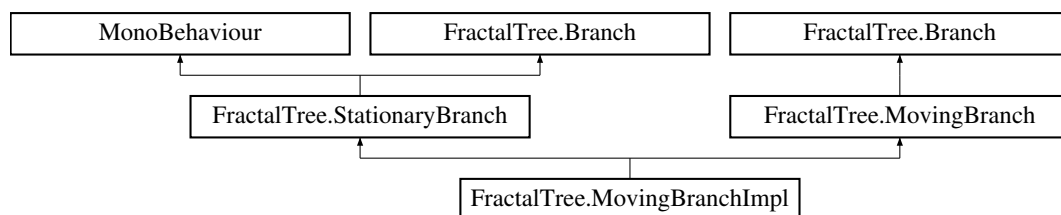
The documentation for this interface was generated from the following file:

- Branch.cs

5.9 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

5.9.1 Detailed Description

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

5.9.2 Member Function Documentation

5.9.2.1 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**

5.9.2.2 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](#).

5.9.2.3 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](#).

5.9.2.4 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](#).

5.9.2.5 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](#).

5.9.3 Property Documentation

5.9.3.1 endPoint

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

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

The end point.

5.9.3.2 endPos

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

Gets the end position.

The end position.

5.9.3.3 startPoint

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

Gets the start point mass. Used to add spring force

The start point.

5.9.3.4 startPos

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

Gets the start position.

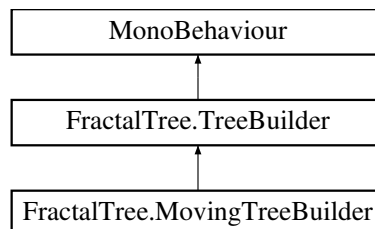
The start position.

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

- `MovingBranchImpl.cs`

5.10 FractalTree.MovingTreeBuilder Class Reference

Inheritance diagram for `FractalTree.MovingTreeBuilder`:



Public Member Functions

- `override void` `Build` ()
Build this instance.
- `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.

Properties

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

Additional Inherited Members

5.10.1 Member Function Documentation

5.10.1.1 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.

5.10.1.2 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.

5.10.1.3 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.

5.10.1.4 Build()

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

Build this instance.

Implements [FractalTree.TreeBuilder](#).

5.10.2 Property Documentation

5.10.2.1 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:

- MovingTreeBuilder.cs

5.11 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.

5.11.1 Detailed Description

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

5.11.2 Constructor & Destructor Documentation

5.11.2.1 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.

5.11.3 Member Function Documentation

5.11.3.1 ApplyForce()

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

Applies a force.

Parameters

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

5.11.3.2 DoUpdate()

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

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

5.11.3.3 IncreaseDamping()

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

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

Parameters

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

5.11.4 Property Documentation

5.11.4.1 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.

5.11.4.2 position

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

The current position of the branch point.

The position.

5.11.4.3 velocity

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

Gets the velocity.

The velocity.

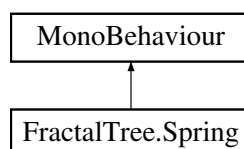
The documentation for this class was generated from the following file:

- PointMass.cs

5.12 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.

5.12.1 Detailed Description

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

5.12.2 Member Function Documentation

5.12.2.1 DoUpdate()

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

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

5.12.2.2 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.

5.12.3 Member Data Documentation

5.12.3.1 end

`PointMass FractalTree.Spring.end`

The end point mass.

5.12.3.2 start

`PointMass FractalTree.Spring.start`

The start point mass.

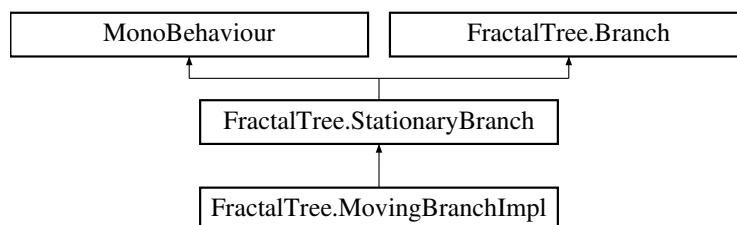
The documentation for this class was generated from the following file:

- Spring.cs

5.13 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.

5.13.1 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.

5.13.2 Member Function Documentation

5.13.2.1 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](#)**

5.13.2.2 DoColonizationReset()

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

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

Implements [FractalTree.Branch](#).

5.13.2.3 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](#).

5.13.2.4 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](#).

5.13.2.5 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](#).

5.13.2.6 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](#).

5.13.2.7 UpdateColor()

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

Updates the sprite renderer color.

Parameters

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

5.13.2.8 UpdateSprite()

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

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

5.13.3 Member Data Documentation

5.13.3.1 LengthDegradation

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

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

5.13.3.2 m_Renderer

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

The renderer associated with the branch.

5.13.3.3 m_Width

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

The width of the branch.

5.13.3.4 SPRITE_SIZE

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

Pixels of line sprite / pixels per units.

5.13.4 Property Documentation

5.13.4.1 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.

5.13.4.2 colonizationLeafCount

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

Gets or sets the number of nearby colonizaion leaves.

The colonization leaf count.

5.13.4.3 color

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

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

The color.

5.13.4.4 endPos

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

Gets the end position.

The end position.

5.13.4.5 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.

5.13.4.6 startPos

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

Gets the start position.

The start position.

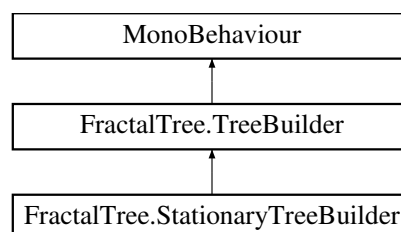
The documentation for this class was generated from the following file:

- StationaryBranch.cs

5.14 FractalTree.StationaryTreeBuilder Class Reference

Builds a stationary tree.

Inheritance diagram for FractalTree.StationaryTreeBuilder:



Public Member Functions

- override void [Build](#) ()
Build this instance.

Properties

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

Additional Inherited Members

5.14.1 Detailed Description

Builds a stationary tree.

5.14.2 Member Function Documentation

5.14.2.1 Build()

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

Build this instance.

Implements [FractalTree.TreeBuilder](#).

5.14.3 Property Documentation

5.14.3.1 branches

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

A list of all branches associated with the tree.

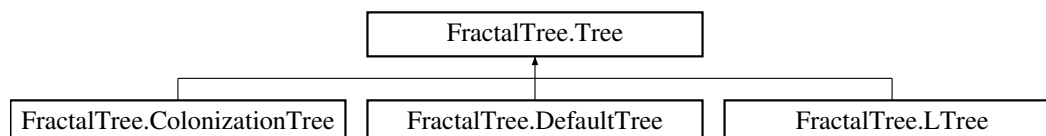
The branches.

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

- StationaryTreeBuilder.cs

5.15 FractalTree.Tree Interface Reference

Inheritance diagram for FractalTree.Tree:



Public Member Functions

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

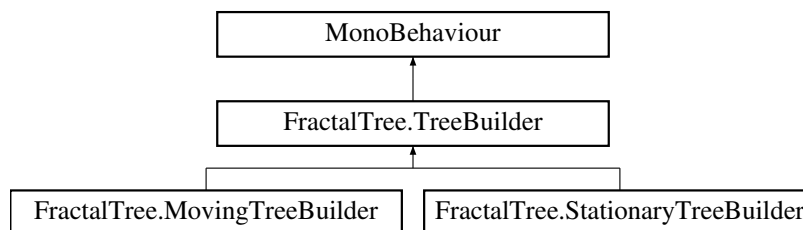
The documentation for this interface was generated from the following file:

- MovingTreeBuilder.cs

5.16 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](#) { **Default**, **LTree**, **Colonization** }
[Tree](#) type.

Public Member Functions

- abstract void **Build** ()

Public Attributes

- bool [buildOnStart](#) = true
If true, builds tree on start.
- [TreeType](#) [treeType](#) = TreeType.Default
The tree type to generate.
- GameObject [branchPrefab](#)
The branch prefab. If tree to generate is moving then prefab should have [MovingBranch](#) script attached.
- int [defaultGrowthCount](#) = 8
The number of tree generations.
- float [defaultInitialLength](#) = 5f
The default length of the initial branches for the default tree generation.
- float [defaultLengthDegradation](#) = 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 I 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 I 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.
- 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.

5.16.1 Detailed Description

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

5.16.2 Member Enumeration Documentation

5.16.2.1 TreeType

```
enum FractalTree.TreeBuilder.TreeType [strong]
```

`Tree` type.

5.16.3 Member Function Documentation

5.16.3.1 CreateTree()

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

Creates a tree based on treeType.

Returns

The tree.

5.16.3.2 DoBuild< T >()

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

Build this instance of the tree.

Type Constraints

T : Branch

5.16.4 Member Data Documentation

5.16.4.1 branchPrefab

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

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

5.16.4.2 buildOnStart

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

If true, builds tree on start.

5.16.4.3 colonizationInitialLength

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

The initial length for a colonization tree trunk.

5.16.4.4 colonizationLeafParent

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

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

5.16.4.5 colonizationMaxDistance

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

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

5.16.4.6 colonizationMinDistance

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

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

5.16.4.7 colonizationWidth

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

The width of the colonization tree branches.

5.16.4.8 defaultAngle

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

The angle for default tree branching.

5.16.4.9 defaultGrowthCount

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

The number of tree generations.

5.16.4.10 defaultInitialLength

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

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

5.16.4.11 defaultLengthDegradation

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

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

5.16.4.12 defaultWidth

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

The width of the branches for the default tree generator.

5.16.4.13 lTreeAngle

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

The angles used to branch an L tree.

5.16.4.14 lTreeAutoWidth

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

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

5.16.4.15 lTreeAxiom

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

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

5.16.4.16 lTreeBranchLength

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

The length of the l tree branch.

5.16.4.17 lTreeColours

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

The L tree colours.

5.16.4.18 lTreeGrowthCount

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

The number of L tree generations.

5.16.4.19 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.

5.16.4.20 lTreeRules

`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.

5.16.4.21 lTreeWidth

`float FractalTree.TreeBuilder.lTreeWidth = 0.03f`

The max branch width for L trees.

5.16.4.22 treeType

`TreeType FractalTree.TreeBuilder.treeType = TreeType.Default`

The tree type to generate.

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

- TreeBuilder.cs

Index

- ApplyDirectedForce
 - FractalTree::MovingTreeBuilder, [27](#)
- ApplyForce
 - FractalTree::PointMass, [29](#)
- ApplyPullForce
 - FractalTree::MovingTreeBuilder, [27](#)
- ApplyPushForce
 - FractalTree::MovingTreeBuilder, [27](#)
- bounds
 - FractalTree::ColonizationLeafGenerator, [15](#)
- branchPrefab
 - FractalTree::TreeBuilder, [42](#)
- branches
 - FractalTree::MovingTreeBuilder, [28](#)
 - FractalTree::StationaryTreeBuilder, [39](#)
- Build
 - FractalTree::MovingTreeBuilder, [28](#)
 - FractalTree::StationaryTreeBuilder, [39](#)
- buildOnStart
 - FractalTree::TreeBuilder, [42](#)
- colonizationDir
 - FractalTree::Branch, [12](#)
 - FractalTree::StationaryBranch, [37](#)
- colonizationInitialLength
 - FractalTree::TreeBuilder, [42](#)
- colonizationLeafCount
 - FractalTree::Branch, [12](#)
 - FractalTree::StationaryBranch, [37](#)
- colonizationLeafParent
 - FractalTree::TreeBuilder, [42](#)
- colonizationMaxDistance
 - FractalTree::TreeBuilder, [42](#)
- colonizationMinDistance
 - FractalTree::TreeBuilder, [43](#)
- ColonizationTree
 - FractalTree::ColonizationTree, [16](#)
- colonizationWidth
 - FractalTree::TreeBuilder, [43](#)
- color
 - FractalTree::StationaryBranch, [37](#)
- CreateTree
 - FractalTree::TreeBuilder, [42](#)
- defaultAngle
 - FractalTree::TreeBuilder, [43](#)
- defaultGrowthCount
 - FractalTree::TreeBuilder, [43](#)
- defaultInitialLength
 - FractalTree::TreeBuilder, [43](#)
- defaultLengthDegradation
 - FractalTree::TreeBuilder, [43](#)
- DefaultTree
 - FractalTree::DefaultTree, [18](#)
- defaultWidth
 - FractalTree::TreeBuilder, [43](#)
- DoBranching< T >
 - FractalTree::Branch, [10](#)
 - FractalTree::MovingBranchImpl, [23](#)
 - FractalTree::StationaryBranch, [34](#)
- DoBuild< T >
 - FractalTree::TreeBuilder, [42](#)
- DoColonizationReset
 - FractalTree::Branch, [10](#)
 - FractalTree::StationaryBranch, [34](#)
- DoUpdate
 - FractalTree::PointMass, [29](#)
 - FractalTree::Spring, [31](#)
- end
 - FractalTree::Spring, [31](#)
- endPoint
 - FractalTree::MovingBranch, [21](#)
 - FractalTree::MovingBranchImpl, [25](#)
- endPos
 - FractalTree::Branch, [13](#)
 - FractalTree::MovingBranchImpl, [25](#)
 - FractalTree::StationaryBranch, [37](#)
- forceApplied
 - FractalTree::PointMass, [30](#)
- FractalTree, [7](#)
- FractalTree.Branch, [9](#)
- FractalTree.ColonizationLeaf, [14](#)
- FractalTree.ColonizationLeafGenerator, [15](#)
- FractalTree.ColonizationTree, [16](#)
- FractalTree.DefaultTree, [17](#)
- FractalTree.LRule, [19](#)
- FractalTree.LTree, [19](#)
- FractalTree.MovingBranch, [21](#)
- FractalTree.MovingBranchImpl, [22](#)
- FractalTree.MovingTreeBuilder, [26](#)
- FractalTree.PointMass, [28](#)
- FractalTree.Spring, [30](#)
- FractalTree.StationaryBranch, [32](#)
- FractalTree.StationaryTreeBuilder, [38](#)
- FractalTree.Tree, [39](#)
- FractalTree.TreeBuilder, [40](#)
- FractalTree::Branch

- colonizationDir, 12
 - colonizationLeafCount, 12
 - DoBranching< T >, 10
 - DoColonizationReset, 10
 - endPos, 13
 - hasBranched, 13
 - Setup, 11, 12
 - startPos, 13
 - transform, 13
- FractalTree::ColonizationLeaf
 - hasBeenReached, 14
 - position, 14
- FractalTree::ColonizationLeafGenerator
 - bounds, 15
 - numToCreate, 15
- FractalTree::ColonizationTree
 - ColonizationTree, 16
 - Generate< T >, 17
- FractalTree::DefaultTree
 - DefaultTree, 18
 - Generate< T >, 18
- FractalTree::LTree
 - Generate< T >, 20
 - LTree, 20
- FractalTree::MovingBranch
 - endPoint, 21
 - startPoint, 21
- FractalTree::MovingBranchImpl
 - DoBranching< T >, 23
 - endPoint, 25
 - endPos, 25
 - Setup, 23–25
 - startPoint, 25
 - startPos, 26
- FractalTree::MovingTreeBuilder
 - ApplyDirectedForce, 27
 - ApplyPullForce, 27
 - ApplyPushForce, 27
 - branches, 28
 - Build, 28
- FractalTree::PointMass
 - ApplyForce, 29
 - DoUpdate, 29
 - forceApplied, 30
 - IncreaseDamping, 29
 - PointMass, 29
 - position, 30
 - velocity, 30
- FractalTree::Spring
 - DoUpdate, 31
 - end, 31
 - Setup, 31
 - start, 32
- FractalTree::StationaryBranch
 - colonizationDir, 37
 - colonizationLeafCount, 37
 - color, 37
 - DoBranching< T >, 34
 - DoColonizationReset, 34
 - endPos, 37
 - hasBranched, 38
 - LengthDegradation, 36
 - m_Renderer, 36
 - m_Width, 37
 - SPRITE_SIZE, 37
 - Setup, 34, 35
 - startPos, 38
 - UpdateColor, 36
 - UpdateSprite, 36
- FractalTree::StationaryTreeBuilder
 - branches, 39
 - Build, 39
- FractalTree::TreeBuilder
 - branchPrefab, 42
 - buildOnStart, 42
 - colonizationInitialLength, 42
 - colonizationLeafParent, 42
 - colonizationMaxDistance, 42
 - colonizationMinDistance, 43
 - colonizationWidth, 43
 - CreateTree, 42
 - defaultAngle, 43
 - defaultGrowthCount, 43
 - defaultInitialLength, 43
 - defaultLengthDegradation, 43
 - defaultWidth, 43
 - DoBuild< T >, 42
 - ITreeAngle, 43
 - ITreeAutoWidth, 44
 - ITreeAxiom, 44
 - ITreeBranchLength, 44
 - ITreeColours, 44
 - ITreeGrowthCount, 44
 - ITreeMassBasedOnWidth, 44
 - ITreeRules, 44
 - ITreeWidth, 45
 - TreeType, 41
 - treeType, 45
- Generate< T >
 - FractalTree::ColonizationTree, 17
 - FractalTree::DefaultTree, 18
 - FractalTree::LTree, 20
- hasBeenReached
 - FractalTree::ColonizationLeaf, 14
- hasBranched
 - FractalTree::Branch, 13
 - FractalTree::StationaryBranch, 38
- IncreaseDamping
 - FractalTree::PointMass, 29
- LTree
 - FractalTree::LTree, 20
- ITreeAngle
 - FractalTree::TreeBuilder, 43

- ITreeAutoWidth
 - FractalTree::TreeBuilder, [44](#)
- ITreeAxiom
 - FractalTree::TreeBuilder, [44](#)
- ITreeBranchLength
 - FractalTree::TreeBuilder, [44](#)
- ITreeColours
 - FractalTree::TreeBuilder, [44](#)
- ITreeGrowthCount
 - FractalTree::TreeBuilder, [44](#)
- ITreeMassBasedOnWidth
 - FractalTree::TreeBuilder, [44](#)
- ITreeRules
 - FractalTree::TreeBuilder, [44](#)
- ITreeWidth
 - FractalTree::TreeBuilder, [45](#)
- LengthDegradation
 - FractalTree::StationaryBranch, [36](#)
- m_Renderer
 - FractalTree::StationaryBranch, [36](#)
- m_Width
 - FractalTree::StationaryBranch, [37](#)
- numToCreate
 - FractalTree::ColonizationLeafGenerator, [15](#)
- PointMass
 - FractalTree::PointMass, [29](#)
- position
 - FractalTree::ColonizationLeaf, [14](#)
 - FractalTree::PointMass, [30](#)
- SPRITE_SIZE
 - FractalTree::StationaryBranch, [37](#)
- Setup
 - FractalTree::Branch, [11](#), [12](#)
 - FractalTree::MovingBranchImpl, [23–25](#)
 - FractalTree::Spring, [31](#)
 - FractalTree::StationaryBranch, [34](#), [35](#)
- start
 - FractalTree::Spring, [32](#)
- startPoint
 - FractalTree::MovingBranch, [21](#)
 - FractalTree::MovingBranchImpl, [25](#)
- startPos
 - FractalTree::Branch, [13](#)
 - FractalTree::MovingBranchImpl, [26](#)
 - FractalTree::StationaryBranch, [38](#)
- transform
 - FractalTree::Branch, [13](#)
- TreeType
 - FractalTree::TreeBuilder, [41](#)
- treeType
 - FractalTree::TreeBuilder, [45](#)
- UpdateColor
 - FractalTree::StationaryBranch, [36](#)
- UpdateSprite
 - FractalTree::StationaryBranch, [36](#)
- velocity
 - FractalTree::PointMass, [30](#)