Fractal Spring Tree Generator

0.1

Gandhi Games

Contents

Chapter 1

Namespace Index

1.1 Packages

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

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`
- 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 apllies 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.

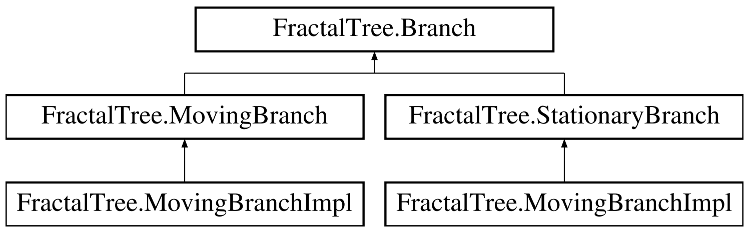
Chapter 5

Class Documentation

5.1 FractalTree.Branch Interface Reference

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

Inheritance diagram for FractalTree.Branch:


FractalTree.Branch
FractalTree.MovingBranch
FractalTree.StationaryBranch
FractalTree.MovingBranchImpl
FractalTree.MovingBranchImpl

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

angle	Angle.

Template Parameters

T	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

owner	The attached branch.
end	End.
thickness	Thickness.
color	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

owner	Owner.
end	End.
thickness	Thickness.
color	Color.
autoMass	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

start	Start.
end	End.
thickness	Thickness.
color	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

start	Start.
end	End.
width	Width.
color	Color.
autoMass	If set to `true` 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.

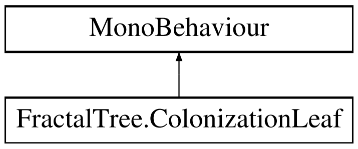
The documentation for this interface was generated from the following file:

- 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.

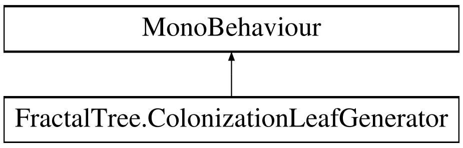
The documentation for this class was generated from the following file:

- 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.

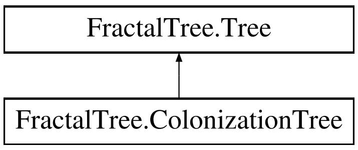
The documentation for this class was generated from the following file:

- 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

leaves	Leaves.
owner	Owner.
initialLength	Initial length.
branchPrefab	Branch prefab.
width	Width.
minDistance	Minimum distance.
maxDistance	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

T	Branch type.

Implements FractalTree.Tree.

Type Constraints

T* : *Branch

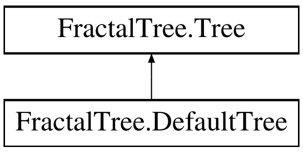
The documentation for this class was generated from the following file:

- 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 lengthDegradation, 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

growth	Growth.
initialLength	Initial length.
lengthDegredation	Length degredation.
angle	Angle.
thickness	Thickness.
branchPrefab	Branch prefab.
owner	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

T	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**

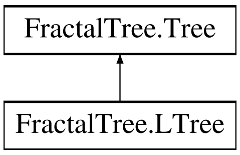
The documentation for this class was generated from the following file:

- LTree.cs

5.7 FractalTree.LTree Class Reference

Spawns a fractal true 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 true 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

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

T	The 1st type parameter.

Implements FractalTree.Tree.

Type Constraints

T* : *Branch

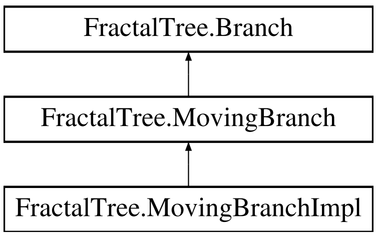
The documentation for this class was generated from the following file:

- 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.

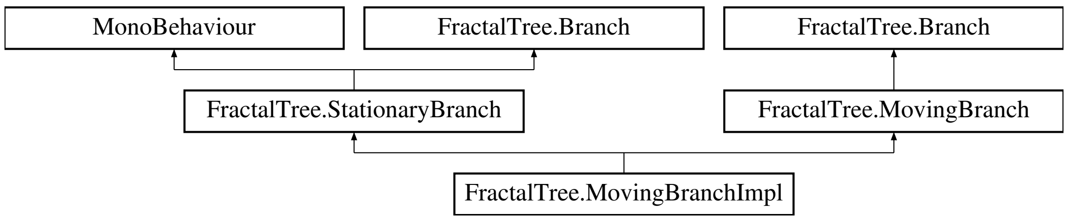
The documentation for this interface was generated from the following file:

- 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

angle	Angle.

Template Parameters

T	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

owner	The attached branch.
end	End.
thickness	Thickness.
color	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

owner	Owner.
end	End.
thickness	Thickness.
color	Color.
autoMass	If set to `true` 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

owner	The attached branch.
end	End.
thickness	Thickness.
color	Color.
start	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

owner	Owner.
end	End.
thickness	Thickness.
color	Color.
start	Start.
width	Width.
autoMass	If set to `true` 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.

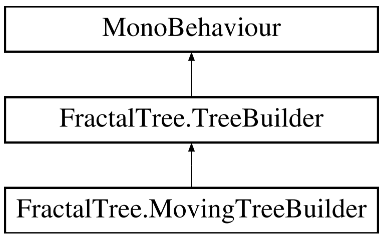
The start position.

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

force	Force.
position	Position.
radius	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

force	Force.
position	Position.
radius	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

force	Force.
position	Position.
radius	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

position	Initial position.
invMass	Inverse mass, lower numbers result in more force required to move the point.
bounceBackForce	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

force	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

factor	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.

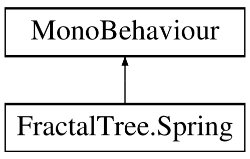
The documentation for this class was generated from the following file:

- PointMass.cs

5.12 FractalTree.Spring Class Reference

Connects two point masses and apllies 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 apllies 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

start	Start.
end	End.
stiffness	Stiffness.
damping	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.

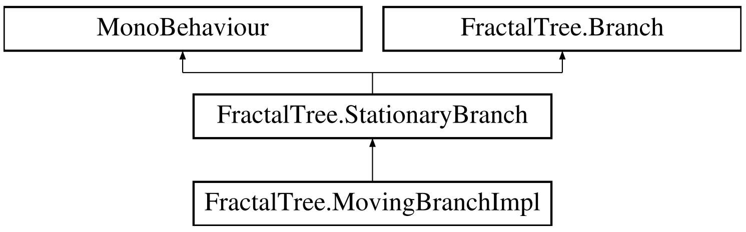
The documentation for this class was generated from the following file:

- 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

angle	Angle.

Template Parameters

T	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

owner	The attached branch.
end	End.
thickness	Thickness.
color	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

owner	Owner.
end	End.
thickness	Thickness.
color	Color.
autoMass	If set to `true` 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

owner	The attached branch.
end	End.
thickness	Thickness.
color	Color.
start	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

owner	Owner.
end	End.
thickness	Thickness.
color	Color.
start	Start.
autoMass	If set to `true` 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

color	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.

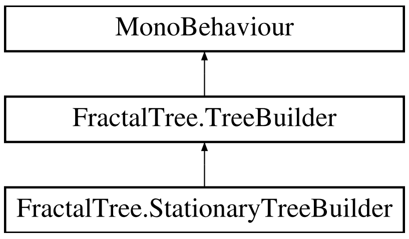
The documentation for this class was generated from the following file:

- 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.

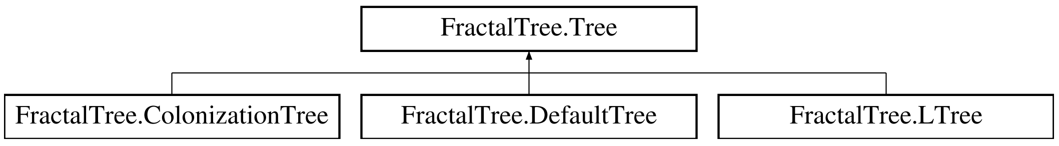
The branches.

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** > ()

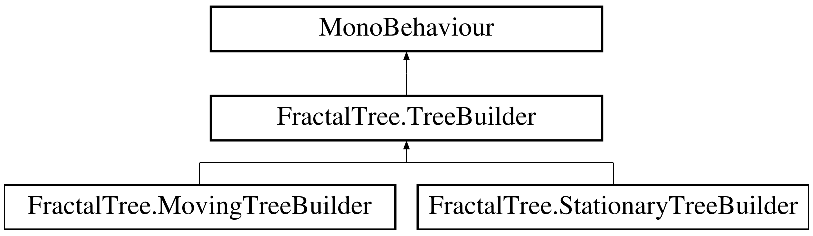
The documentation for this interface was generated from the following file:

- 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:


MonoBehaviour
FractalTree.TreeBuilder
FractalTree.MovingTreeBuilder
FractalTree.StationaryTreeBuilder

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 lTreeAutoWidth = true

    When true, the width of the branches will be set automatically based on the colours.
- bool lTreeMassBasedOnWidth = true

    When true, the mass of the branches will be set automatically based on colours. Used only when generating a moving tree.
- float lTreeWidth = 0.03f

    The max branch width for L trees.
- int lTreeGrowthCount = 5

    The number of L tree generations.
- string lTreeAxiom = "FX"

    The l tree axiom. The initial seed used to generate a L tree.
- LRule [ ] lTreeRules

    The rules applied to the axoim.
- float lTreeBranchLength = 0.17f

    The length of the l tree branch.
- float lTreeAngle = 25f

    The angles used to branch an L tree.
- Color [ ] lTreeColours

    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