DinoFracture

2.8.3

Generated by Doxygen 1.9.6

Chapter 1

DinoFracture

1.1 About DinoFracture

DinoFracture is a tool that can shatter meshes either in real time or in the editor. It works on any mesh, skinned or static, but works best with water-tight meshes. A water-tight mesh is one that does not have holes exposing the backfaces of the mesh polygons. Well-formed holes, like in donuts, *are* allowed.

1.2 Basic Usage

Scripts are located in the **`DinoFracture\Plugin\Scripts`** directory. There are three kinds of scripts in the folder:

- Trigger scripts that cause fractures.
- Scripts that allow this object to be fractured
- Notify scripts that perform an action when fractured
- Engine scripts that are not meant to be used by the user The general flow is to apply either the PreFracturedGeometry or RuntimeFracturedGeometry to an object with a mesh. Apply a trigger script to either the object being fractured (ex: FractureOnCollision) or an external object that will cause a fracture. Apply notify scripts (ex: PlaySoundOnFracture) to perform actions when the fracture occurs. See the **Playground** scene in **`DinoFracture\Demo`** for an example of some different usage patterns.

1.3 Support

If you have any questions about how the plugin works or run into any issues, please contact us at **`support@entropysoftware.io`**

Chapter 2

Namespace Index

2.1 Package List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 DinoFracture Namespace Reference

Classes

- class AsyncFractureOperation

  Tracks completion and returns the results of a fragmentation operation.
- class AsyncFractureResult

  The result of a fracture.
- class AsyncShatterOperation

  Tracks completion and returns the results of a shatter.
- class AsyncSliceOperation

  Tracks completion and returns the results of a slice.
- class ChipOnFracture

  This component allows a fracture geometry to lose only bits of geometry at a time instead of completely collapsing after fracture. It works best when fracturing is done at a point instead and in conjunction with the GlueEdgeOnFracture script on the fracture template.
- class CleanupMeshOnDestroy

  Applying this script to the fracture template will ensure that the generated fracture mesh will be cleaned up properly when the fracture piece is destroyed.
- struct CoroutineHandle

  Handle to a coroutine in the CoroutinePump.
- class CoroutinePump

  Allows for pumping of a list of coroutines for a certain maximum of time per frame / update.
- class DisableObjectsOnFracture

  Adding this to the fracturing game object will allow other game objects to be turned off (set inactive) when this game object is fractured.
- struct EdgeError

  Used to display geometry errors on the mesh in the editor.
- class FractureDetails

  Basic information for any sort of mesh fragmentation.
- struct FracturedMesh

  An individual fracture piece's geometry.
- class FracturedObject

  This class is automatically added by the engine and is not meant to be added by users.
- class FractureEngine

This component is created on demand to manage the fracture coroutines. It is not intended to be added by the user.

- class FractureEngineBase

  Base class for the fracture engine.

- class FractureGeometry

  This is the base class for the PreFractureGeometry and RuntimeFractureGeometry components. As such, it is not intended to be directly added to any game object even though fracture initiator components rely on it.

- class FractureOnClick

  Casts a simple mouse ray on left click and calls Fracture() on the hit collider game object.

- class FractureOnCollision

  This component will cause a fracture to happen at the point of impact.

- class FractureOnInput

  Apply this on the fracturing game object. When the specified key is pressed, the object will fracture.

- class FractureOnParticleCollision

  This component will cause a fracture to happen at the point of impact with a particle.

- class FractureResult

  The result of a fracture.

- class GlueEdgeOnFracture

  If the fracture pieces intersects with a specified trigger when created, the rigid body is destroyed and the piece becomes static. Otherwise, the piece will turn on gravity. It's best used if the FractureTemplate's rigid body is set to not use gravity initially.

- class NotifyOnFracture

  When added to the same game object as the FractureGeometry, this script can be used to notify external game objects of this object's fracture completion. The external objects need a script with the "OnFracture" callback method.

- class OnFractureEventArgs

  Argument passed to OnFracture message.

- class PlaySoundOnFracture

  An object with this component will play the audio source when fractured.

- class PreFracturedGeometry

  Apply this component to any game object you wish to pre-fracture. Pre-fracturing is a way of baking fracture pieces into the scene. Each time the object is fractured, the same set of pieces will activate. This is very useful when creating a large number of pieces or high poly meshes, which would be too slow to create at runtime. The pieces will be in the scene as a disabled root object with piece children. When the object is fractured, those pieces will activate.

- class RuntimeFracturedGeometry

  Apply this component to any game object you wish to fracture while running in game mode. Runtime fractures will produce a unique set of pieces with each fracture. However, this is at the cost of computational time. It is recommended that both the piece count and poly count are kept low. This component is most effective when FractureRadius is set to a value in-between 0 and 1.

- class ShatterDetails

  Required information needed by the engine to produce a fracture.

- struct Size

  Structure that defines the bounds of the shatter fracture planes.

- class SliceDetails

  Required information needed by the engine to slice a mesh.

- struct SlicePlane

  Defines a plane that slices the mesh in half.

- class TransferJointsOnFracture

  When this object is fractured, the joint component on the object will be copied to this piece if this piece is sufficiently close to the joint position. Without this component, joints are broken after fracturing.

- class TriggerExplosionOnCollision

  Triggers a fracture + explosion when this game object is collided with.

- struct UVBounds

  Defines a rectangular bounds across a UV map.

Enumerations

- enum FractureUVScale { EntireMesh , Piece }

  Algorithm used to generate UVs on inside faces.
- enum FractureIssueResolution { NoAction , DisableGameObject , ReplaceMeshCollider }

  Technique used to handle pieces that generated with potential issues.
- enum MeshValidity { Unknown , Valid , NeedsCleaning , Unrecoverable }

  Denotes the state of the mesh's topology and readiness to be fractured.
- enum MeshTopologyError {
  None = 0 , DegenerateTriangles = 1 << 0 , OpenFaces = 1 << 1 , CloseVertices = 1 << 2 ,
  FanFaces = 1 << 3 }

  Returns a list of errors in the mesh's topology.
- enum SizeSpace { RelativeToBounds = 0 , WorldSpace }

  The space used to determine the shatter fracture planes bounds.
- enum FracturedMeshResultFlags { NoIssues = 0 , SmallVertexCount = 1 << 0 , ZeroVolume = 1 << 1 }

  Additional flags describing the generated fracture mesh.
- enum LogLevel {
  Statistic , Debug , Info , Warning ,
  Error , UserDisplayedInfo , UserDisplayedWarning , UserDisplayedError }

  Logging severity values.
- enum FractureType { Shatter , Slice }

  The type of fracture to perform.

Functions

- delegate void LogHandler (LogLevel level, string message, UnityEngine.Object context)

  User-specified handler to override the default logging behavior.

5.1.1 Enumeration Type Documentation

5.1.1.1 FracturedMeshResultFlags

`enum DinoFracture.FracturedMeshResultFlags`

Additional flags describing the generated fracture mesh.

Enumerator

NoIssues	Everything is okay.
SmallVertexCount	There is a substantially small number of vertices in this mesh. This can cause problems when generating a mesh collider.
ZeroVolume	All vertices on the mesh are coplanar and cannot be used for creating a convex hull with a mesh collider.

5.1.1.2 FractureIssueResolution

enum DinoFracture.FractureIssueResolution

Technique used to handle pieces that generated with potential issues.

Enumerator

NoAction	Do nothing internally - let the user take action.
DisableGameObject	Completely disable the game object.
ReplaceMeshCollider	Replaces the mesh collider with a sphere collider if the mesh collider could potentially fail to generate.

5.1.1.3 FractureType

enum DinoFracture.FractureType

The type of fracture to perform.

Enumerator

Shatter	Traditional fracture. Divide the mesh into many random sized pieces.
Slice	Use one or more user-defined planes to cut the mesh.

5.1.1.4 FractureUVScale

enum DinoFracture.FractureUVScale

Algorithm used to generate UVs on inside faces.

Enumerator

EntireMesh	The fracture UVs map to the size of the original mesh.
Piece	The fracture UVs map to the size of each individual piece.

5.1.1.5 LogLevel

enum DinoFracture.LogLevel

Logging severity values.

Enumerator

Statistic	Performance, timings, etc.
Debug	Internal verification checks.
Info	General information.
Warning	Non-fatal error containing detailed information.
Error	Fatal error containing detailed information.
UserDisplayedInfo	Information summary message in user language.
UserDisplayedWarning	Information summary message in user language.
UserDisplayedError	Fatal error summary in user language.

5.1.1.6 MeshTopologyError

`enum DinoFracture.MeshTopologyError`

Returns a list of errors in the mesh's topology.

Enumerator

None	No errors. Mesh is valid.
DegenerateTriangles	The mesh has zero-area triangles (either 3 vertices in a line or collapsed to a single point) This can be caused by poor export from DCC tools or by Unity doing a bad triangulation job importing NGons from an FBX. Degenerate triangles can be fixed through automatic mesh cleaning.
OpenFaces	The mesh is not water-tight. Mathematically, this means there is at least one edge that has exactly one associated triangle. Normal edges are shared by exactly 2 triangles. Open faces can be fixed through automatic mesh cleaning as long as there are at 3 connected open edges that all belong a single plane. Multiple groups of open faces can be fixed for a single mesh.
CloseVertices	The mesh has vertices that are extremely close, but not exactly on the same position. These vertices will be merged before during the cleaning process.
FanFaces	An edge is being shared by 3 or more triangles. It is no longer possible to determine what is inside vs outside of the mesh. This type of error cannot be fixed automatically.

5.1.1.7 MeshValidity

`enum DinoFracture.MeshValidity`

Denotes the state of the mesh's topology and readiness to be fractured.

Enumerator

Unknown	Meshes should only be in this state when the fracture script is first created and has yet to be clicked on in the editor. The mesh will be first checked and cleaned before fracturing.
Valid	The mesh is known to be valid and no checks or cleaning will be performed.

Enumerator

NeedsCleaning	The mesh is known to be invalid for fracturing. The mesh will be cleaned internally before starting the fracture.
Unrecoverable	The mesh is beyond what can be done through automatic cleaning. Fracturing can still happen, but there is a high chance that random slices will result in a failed fracture.

5.1.1.8 SizeSpace

```
enum DinoFracture.SizeSpace
```

The space used to determine the shatter fracture planes bounds.

Enumerator

RelativeToBounds	The size value is a percentage relative to the object's bounds. Use a value [0..1].
WorldSpace	The size value is in world space units.

5.1.2 Function Documentation

5.1.2.1 LogHandler()

```
delegate void DinoFracture.LogHandler (
            LogLevel level,
            string message,
            UnityEngine.Object context )
```

User-specified handler to override the default logging behavior.

Parameters

level	
message	
context	

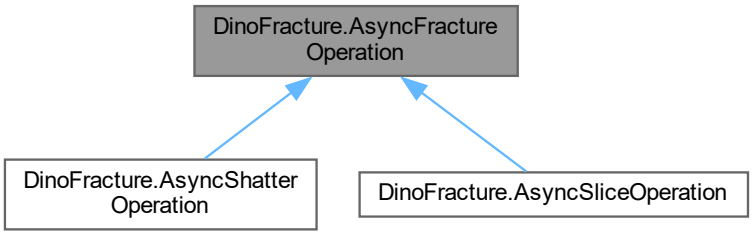
Chapter 6

Class Documentation

6.1 DinoFracture.AsyncFractureOperation Class Reference

Tracks completion and returns the results of a fragmentation operation.

Inheritance diagram for DinoFracture.AsyncFractureOperation:



Public Member Functions

- void Wait (int msTimeout)

  Wait on the fracture to complete.

Protected Member Functions

- void **SetComplete** ()

  Marks this operation as completed.

Properties

- FractureDetails **Details** `[get, set]`

  The original details passed into the operation.
- FractureResult **Result** `[get, protected set]`

  The result of the fracture. This is not set until IsComplete is true.
- bool **IsComplete** `[get]`

  True if the fracture has completed, false otherwise. If this is a synchronous fracture, this value will always be true by the end of the fracture call.
- bool **ErrorDuringFracture** `[get, protected set]`

  If true, results may be empty or not completely valid.
- float **ProgressPercent** `[get]`

  A number [0..1] denoting the completion percentage of the fracture. Computed by dividing CompletedOperationCount by TotalOperationCount.
- int **CompletedOperationCount** `[get]`

  The number of completed operations / segments in this fracture. The percent complete is this value divided by TotalOperationCount.
- int **TotalOperationCount** `[get]`

  The total number of operations / segments in this fracture. The percent complete is CompletedOperationCount divided by this number.

6.1.1 Detailed Description

Tracks completion and returns the results of a fragmentation operation.

6.1.2 Member Function Documentation

6.1.2.1 Wait()

```
void DinoFracture.AsyncFractureOperation.Wait (
            int msTimeout )
```

Wait on the fracture to complete.

Parameters

msTimeout	Max time to wait. 0 to not wait, -1 to wait forever.

6.2 DinoFracture.AsyncFractureResult Class Reference

The result of a fracture.

Properties

- bool **IsComplete** `[get]`

  Returns true if the operation has finished; false otherwise. This value will always be true for synchronous fractures.
- bool **IsSuccessful** `[get]`

  Returns true if the operation has finished and returned valid results.
- FractureGeometry **FractureGeometry** `[get]`

  The original script that initiated the fracture.
- GameObject **PiecesRoot** `[get]`

  The root of the pieces of the resulting fracture.
- Bounds **EntireMeshBounds** `[get]`

  The bounds of the original mesh.
- float **ProgressPercent** `[get]`

  A number [0..1] denoting the completion percentage of the fracture. Computed by dividing CompletedOperationCount by TotalOperationCount.
- int **CompletedOperationCount** `[get]`

  The number of completed operations / segments in this fracture. The percent complete is this value divided by TotalOperationCount.
- int **TotalOperationCount** `[get]`

  The total number of operations / segments in this fracture. The percent complete is CompletedOperationCount divided by this number.

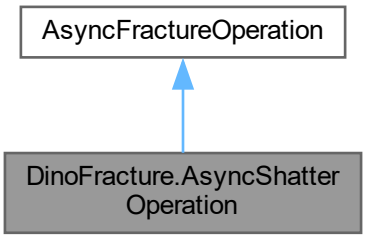
6.2.1 Detailed Description

The result of a fracture.

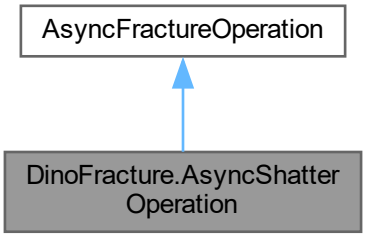
6.3 DinoFracture.AsyncShatterOperation Class Reference

Tracks completion and returns the results of a shatter.

Inheritance diagram for DinoFracture.AsyncShatterOperation:



Collaboration diagram for DinoFracture.AsyncShatterOperation:



Properties

- new ShatterDetails **Details** `[get, set]`

  The original details passed into the operation.

Properties inherited from DinoFracture.AsyncFractureOperation

- FractureDetails **Details** `[get, set]`

  The original details passed into the operation.
- FractureResult **Result** `[get, protected set]`

  The result of the fracture. This is not set until IsComplete is true.
- bool **IsComplete** `[get]`

  True if the fracture has completed, false otherwise. If this is a synchronous fracture, this value will always be true by the end of the fracture call.
- bool **ErrorDuringFracture** `[get, protected set]`

  If true, results may be empty or not completely valid.
- float **ProgressPercent** `[get]`

  A number [0..1] denoting the completion percentage of the fracture. Computed by dividing CompletedOperationCount by TotalOperationCount.
- int **CompletedOperationCount** `[get]`

  The number of completed operations / segments in this fracture. The percent complete is this value divided by TotalOperationCount.
- int **TotalOperationCount** `[get]`

  The total number of operations / segments in this fracture. The percent complete is CompletedOperationCount divided by this number.

Additional Inherited Members

Public Member Functions inherited from DinoFracture.AsyncFractureOperation

- void Wait (int msTimeout)

  Wait on the fracture to complete.

Protected Member Functions inherited from DinoFracture.AsyncFractureOperation

- void **SetComplete** ()

  Marks this operation as completed.

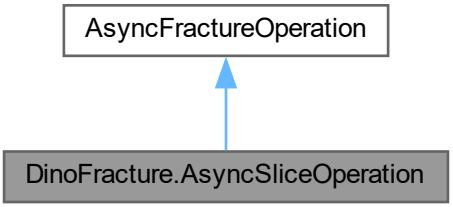
6.3.1 Detailed Description

Tracks completion and returns the results of a shatter.

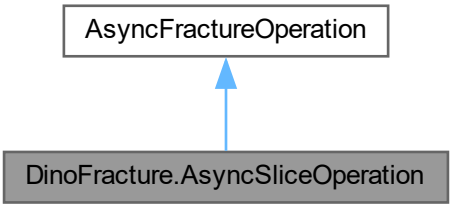
6.4 DinoFracture.AsyncSliceOperation Class Reference

Tracks completion and returns the results of a slice.

Inheritance diagram for DinoFracture.AsyncSliceOperation:



Collaboration diagram for DinoFracture.AsyncSliceOperation:



Properties

- new SliceDetails **Details** `[get, set]`

  The original details passed into the operation.

Properties inherited from DinoFracture.AsyncFractureOperation

- FractureDetails **Details** `[get, set]`

  The original details passed into the operation.
- FractureResult **Result** `[get, protected set]`

  The result of the fracture. This is not set until IsComplete is true.
- bool **IsComplete** `[get]`

  True if the fracture has completed, false otherwise. If this is a synchronous fracture, this value will always be true by the end of the fracture call.
- bool **ErrorDuringFracture** `[get, protected set]`

  If true, results may be empty or not completely valid.
- float **ProgressPercent** `[get]`

  A number [0..1] denoting the completion percentage of the fracture. Computed by dividing CompletedOperationCount by TotalOperationCount.
- int **CompletedOperationCount** `[get]`

  The number of completed operations / segments in this fracture. The percent complete is this value divided by TotalOperationCount.
- int **TotalOperationCount** `[get]`

  The total number of operations / segments in this fracture. The percent complete is CompletedOperationCount divided by this number.

Additional Inherited Members

Public Member Functions inherited from DinoFracture.AsyncFractureOperation

- void Wait (int msTimeout)

  Wait on the fracture to complete.

Protected Member Functions inherited from DinoFracture.AsyncFractureOperation

- void **SetComplete** ()

  Marks this operation as completed.

6.4.1 Detailed Description

Tracks completion and returns the results of a slice.

6.5 DinoFracture.ChipOnFracture Class Reference

This component allows a fracture geometry to lose only bits of geometry at a time instead of completely collapsing after fracture. It works best when fracturing is done at a point instead and in conjunction with the GlueEdgeOnFracture script on the fracture template.

Inherits MonoBehaviour.

Public Attributes

- float Radius

  The radius, in world space units, around the point of fracture that we will consider fracture pieces to be chipped off.
- bool **EnsureChildComponents** = true

  Add this component to child items if not already present in the FractureTemplate. It is recommended to keep this true unless you explicitly add it to the FractureTemplate.
- bool DetectDetachedChunks = false

  If true, the collection of unchipped objects will be scanned when a fracture happens to see if major sections of the collection have been split apart. If so, each section will be separated with distinct rigid bodies. This allows for the chipping away of structural foundations to cause pieces above to collapse.
- bool **DestroyIfEmpty** = true

  If true, unchipped roots will be destroyed if all the child objects have been fractured. It is recommended to set this to true.

6.5.1 Detailed Description

This component allows a fracture geometry to lose only bits of geometry at a time instead of completely collapsing after fracture. It works best when fracturing is done at a point instead and in conjunction with the GlueEdgeOnFracture script on the fracture template.

6.5.2 Member Data Documentation

6.5.2.1 DetectDetachedChunks

```
bool DinoFracture.ChipOnFracture.DetectDetachedChunks = false
```

If true, the collection of unchipped objects will be scanned when a fracture happens to see if major sections of the collection have been split apart. If so, each section will be separated with distinct rigid bodies. This allows for the chipping away of structural foundations to cause pieces above to collapse.

All detected separate chunks will be made non-kinematic. Use the GlueEdgeOnFracture script on the original fracture template to keep structure foundations from moving.

6.5.2.2 Radius

```
float DinoFracture.ChipOnFracture.Radius
```

The radius, in world space units, around the point of fracture that we will consider fracture pieces to be chipped off.

A fracture piece must be fully contained within the radius to be considered chipped.

If this is set to <= 0.0, the FractureSize on the FractureGeometry will be used.

6.6 DinoFracture.CleanupMeshOnDestroy Class Reference

Applying this script to the fracture template will ensure that the generated fracture mesh will be cleaned up properly when the fracture piece is destroyed.

Inherits MonoBehaviour.

6.6.1 Detailed Description

Applying this script to the fracture template will ensure that the generated fracture mesh will be cleaned up properly when the fracture piece is destroyed.

It is always a good idea to add this to the fracture template.

6.7 DinoFracture.CoroutineHandle Struct Reference

Handle to a coroutine in the CoroutinePump.

6.7.1 Detailed Description

Handle to a coroutine in the CoroutinePump.

6.8 DinoFracture.CoroutinePump Class Reference

Allows for pumping of a list of coroutines for a certain maximum of time per frame / update.

Public Member Functions

- CoroutineHandle **AddCoroutine** (IEnumerator coroutine)

  Adds a coroutine to the list. The coroutine will execute on the next call to Update if it can be run.
- bool **CancelCoroutine** (in CoroutineHandle handle)

  Cancels / removes a running coroutine.
- void **Update** (TimeSpan maxTimeToRun)

  Pumps all coroutines up to the specified amount of time.

6.8.1 Detailed Description

Allows for pumping of a list of coroutines for a certain maximum of time per frame / update.

6.9 DinoFracture.DisableObjectsOnFracture Class Reference

Adding this to the fracturing game object will allow other game objects to be turned off (set inactive) when this game object is fractured.

Inherits MonoBehaviour.

6.9.1 Detailed Description

Adding this to the fracturing game object will allow other game objects to be turned off (set inactive) when this game object is fractured.

6.10 DinoFracture.EdgeError Struct Reference

Used to display geometry errors on the mesh in the editor.

Public Attributes

- readonly Vector3 **V0**

  First edge vertex, in object space.
- readonly Vector3 **V1**

  Second edge vertex, in object space.
- readonly MeshTopologyError **Errors**

  Error(s) associated with this edge. Useful for coloring the edges.
- readonly UnityEngine.GameObject **GameObject**

  The game object associated with the errors.

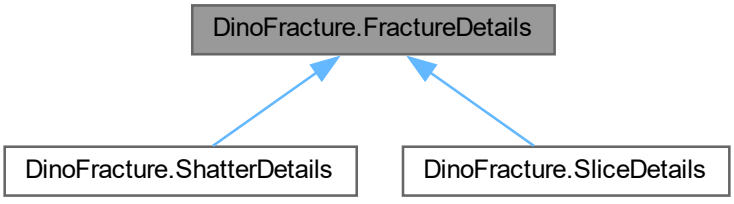
6.10.1 Detailed Description

Used to display geometry errors on the mesh in the editor.

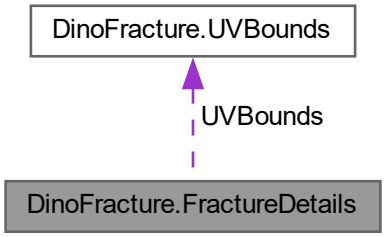
6.11 DinoFracture.FractureDetails Class Reference

Basic information for any sort of mesh fragmentation.

Inheritance diagram for DinoFracture.FractureDetails:



Collaboration diagram for DinoFracture.FractureDetails:



Public Member Functions

- virtual bool IsValid ()

  Returns true if the details are filled in correctly, false otherwise.

Public Attributes

- UnityEngine.Mesh **Mesh**

  The mesh to fracture.
- Vector3 **MeshScale**

  The scale of the mesh's game object. The meshes of fracture pieces will be scaled by this amount to allow their game object's scales to be one.
- FractureUVScale **UVScale**

  Scaling algorithm used on triangles produced during the fracture.
- UVBounds **UVBounds** = new UVBounds(Vector2.zero, Vector2.one)

  Final 'inside' triangles will be remapped to be within this range after the UV scale is applied.
- FractureIssueResolution IssueResolution

  How to deal with potentially poorly generated pieces.
- bool **Asynchronous**

  If true, fracturing is done on a background thread and results may not be ready by the time FractureBuilder.Fracture() finishes. If false, fracturing is guaranteed to be done by the time FractureBuilder.Fracture() finishes.
- int InsideMaterialIndex

  The material / sub-mesh index that newly formed triangles should be put in.
- bool SeparateDisjointPieces

  If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.
- MeshValidity **Validity** = MeshValidity.Unknown

  Highly recommended to call.

Properties

- int **FractureFrame** `[get, set]`

  The frame the fracture started on. Can be used to group separate fractures that should be treated as one.

6.11.1 Detailed Description

Basic information for any sort of mesh fragmentation.

6.11.2 Member Function Documentation

6.11.2.1 IsValid()

```
virtual bool DinoFracture.FractureDetails.IsValid ( )  [virtual]
```

Returns true if the details are filled in correctly, false otherwise.

Returns

Reimplemented in DinoFracture.ShatterDetails, and DinoFracture.SliceDetails.

6.11.3 Member Data Documentation

6.11.3.1 InsideMaterialIndex

```
int DinoFracture.FractureDetails.InsideMaterialIndex
```

The material / sub-mesh index that newly formed triangles should be put in.

Specify -1 to put at the end of the list.

6.11.3.2 IssueResolution

```
FractureIssueResolution DinoFracture.FractureDetails.IssueResolution
```

How to deal with potentially poorly generated pieces.

Note: Any generated mesh with zero triangles is automatically removed.

6.11.3.3 SeparateDisjointPieces

```
bool DinoFracture.FractureDetails.SeparateDisjointPieces
```

If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.

This process can be slow. It is recommended to be off for runtime fractures unless there is a good chance of disjoint pieces.

6.12 DinoFracture.FracturedMesh Struct Reference

An individual fracture piece's geometry.

Public Attributes

- UnityEngine.Mesh **Mesh**

  The generated Unity mesh.
- FracturedMeshResultFlags **Flags**

  Additional information about the generated result.
- Vector3 **Offset**

  The offset from the origin of the original mesh to the center of this piece.
- int **EmptyTriangleCount**

  The number of materials that have no triangles and have been removed.
- List< bool > **EmptyTriangles**

  A true for each material from the original mesh that now have zero triangles and have been removed from the mesh.

6.12.1 Detailed Description

An individual fracture piece's geometry.

6.13 DinoFracture.FracturedObject Class Reference

This class is automatically added by the engine and is not meant to be added by users.

Inherits MonoBehaviour.

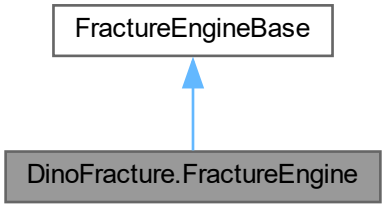
6.13.1 Detailed Description

This class is automatically added by the engine and is not meant to be added by users.

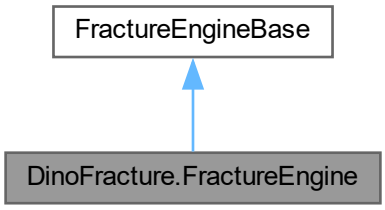
6.14 DinoFracture.FractureEngine Class Reference

This component is created on demand to manage the fracture coroutines. It is not intended to be added by the user.

Inheritance diagram for DinoFracture.FractureEngine:



Collaboration diagram for DinoFracture.FractureEngine:



Static Public Member Functions

- static AsyncFractureResult StartFracture (FractureDetails details, FractureGeometry callback, Transform piecesParent, bool transferMass, bool hideAfterFracture)

  Starts a fracture operation.

Static Public Member Functions inherited from DinoFracture.FractureEngineBase

- static void ClearCachedFractureData ()

  During both slicing and shattering, temporary data is created and cached to greatly improve future fracture performance.
- static CoroutineHandle **QueueCoroutine** (IEnumerator enumerator)

  Adds a coroutine that will start processing on the next update.
- static bool CancelCoroutine (CoroutineHandle handle)

  Cancels a running coroutine.

Protected Member Functions

- override void **Update** ()

  Update.

Protected Member Functions inherited from DinoFracture.FractureEngineBase

- virtual void **Update** ()

  Update.
- void RunOnMainThread (Action action)

  Run delegate on main thread.

Properties

- static bool **Suspended** `[get, set]`

  True if all further fracture operations should be a no-op.
- static bool **HasFracturesInProgress** `[get]`

  Returns true if there are fractures currently in progress.
- static int MaxRunningFractures `[get]`

  The maximum number of async fractures we can process at a time. If this is set to 0 (default), an unlimited number can be run.

Properties inherited from DinoFracture.FractureEngineBase

- static FractureEngineBase **Instance** `[get, set]`

  Internal instance.
- static bool ForceSynchronousPreFractureInEditor `[get]`

  If true, pre-fracturing in the editor will always be synchronous.

6.14.1 Detailed Description

This component is created on demand to manage the fracture coroutines. It is not intended to be added by the user.

6.14.2 Member Function Documentation

6.14.2.1 StartFracture()

```
static AsyncFractureResult DinoFracture.FractureEngine.StartFracture (
            FractureDetails details,
            FractureGeometry callback,
            Transform piecesParent,
            bool transferMass,
            bool hideAfterFracture )  [static]
```

Starts a fracture operation.

Parameters

details	Fracture info
callback	The object to fracture
piecesParent	The parent of the resulting fractured pieces root object
transferMass	True to distribute the original object's mass to the fracture pieces; false otherwise
hideAfterFracture	True to hide the originating object after fracturing

Returns

6.14.3 Property Documentation

6.14.3.1 MaxRunningFractures

```
int DinoFracture.FractureEngine.MaxRunningFractures  [static], [get]
```

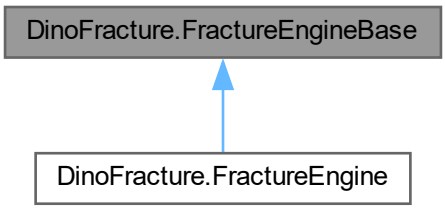
The maximum number of async fractures we can process at a time. If this is set to 0 (default), an unlimited number can be run.

NOTE: Synchronous fractures always run immediately

6.15 DinoFracture.FractureEngineBase Class Reference

Base class for the fracture engine.

Inheritance diagram for DinoFracture.FractureEngineBase:



Static Public Member Functions

- static void ClearCachedFractureData ()

  During both slicing and shattering, temporary data is created and cached to greatly improve future fracture performance.
- static CoroutineHandle **QueueCoroutine** (IEnumerator enumerator)

  Adds a coroutine that will start processing on the next update.
- static bool CancelCoroutine (CoroutineHandle handle)

  Cancels a running coroutine.

Protected Member Functions

- virtual void **Update** ()

  Update.
- void RunOnMainThread (Action action)

  Run delegate on main thread.

Properties

- static FractureEngineBase **Instance** `[get, set]`

  Internal instance.
- static bool ForceSynchronousPreFractureInEditor `[get]`

  If true, pre-fracturing in the editor will always be synchronous.

6.15.1 Detailed Description

Base class for the fracture engine.

6.15.2 Member Function Documentation

6.15.2.1 CancelCoroutine()

```
static bool DinoFracture.FractureEngineBase.CancelCoroutine (
            CoroutineHandle handle )  [static]
```

Cancels a running coroutine.

Returns

True if the coroutine was stopped; false if the coroutine has already finished.

6.15.2.2 ClearCachedFractureData()

```
static void DinoFracture.FractureEngineBase.ClearCachedFractureData ( )  [static]
```

During both slicing and shattering, temporary data is created and cached to greatly improve future fracture performance.

Calling this method releases that temporary data back to be reclaimed by the GC.

Calling this method may cause a large GC spike soon after. It is recommended to call this during 'downtime' in the game, such as during level load.

6.15.2.3 RunOnMainThread()

```
void DinoFracture.FractureEngineBase.RunOnMainThread (
            Action action )  [protected]
```

Run delegate on main thread.

Parameters

action	

6.15.3 Property Documentation

6.15.3.1 ForceSynchronousPreFractureInEditor

```
bool DinoFracture.FractureEngineBase.ForceSynchronousPreFractureInEditor  [static], [get]
```

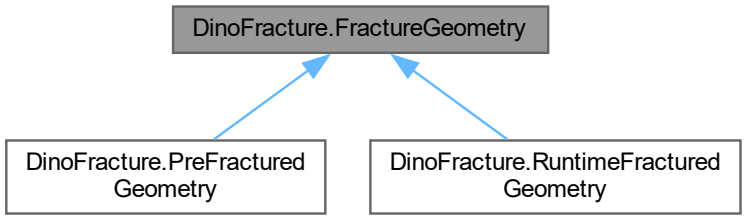
If true, pre-fracturing in the editor will always be synchronous.

This is mainly used for debugging purposes.

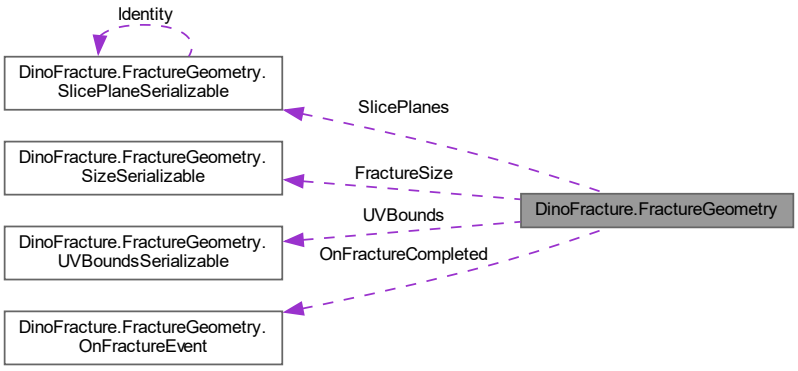
6.16 DinoFracture.FractureGeometry Class Reference

This is the base class for the PreFractureGeometry and RuntimeFractureGeometry components. As such, it is not intended to be directly added to any game object even though fracture initiator components rely on it.

Inheritance diagram for DinoFracture.FractureGeometry:



Collaboration diagram for DinoFracture.FractureGeometry:



Classes

- class OnFractureEvent

  OnFracture() Unity event wrapper.
- struct SizeSerializable

  Unity cannot handle the serializable attribute on types defined in dlls. This wraps a DinoFracture.Size type.
- struct SlicePlaneSerializable

  Unity cannot handle the serializable attribute on types defined in dlls. So, we have to duplicate the SlicePlane structure here in order to save it.
- struct UVBoundsSerializable

  Unity cannot handle the serializable attribute on types defined in dlls. This wraps a DinoFracture.UVBounds type.

Public Member Functions

- void FractureAndForget ()

  Initiate a fracture at the origin and does not return a handle to the async operation.
- void FractureAndForget (Vector3 localPos)

  Initiate a fracture at the specified position relative to this object and does not return a handle to the async operation.
- AsyncFractureResult Fracture ()

  Initiate a fracture at the origin.
- AsyncFractureResult Fracture (Vector3 localPos)

  Initiate a fracture at the specified position relative to this object.
- MeshTopologyError **CheckMeshValidity** ()

  This is called automatically when viewing the component in the inspector. However, it should be called whenever the mesh changes through other means.
- void ForceValidGeometry ()

  This will force the geometry to pass all validity checks.
- List< EdgeError > **GetMeshEdgeErrors** ()

  Returns any bad mesh edges. Used by the editor script for debugging.

Static Public Member Functions

- static SlicePlaneSerializable CreateSlicePlane (Plane worldPlane, Transform targetGameObject)

  This will create a valid slice plane for slicing a mesh.

Public Attributes

- Material **InsideMaterial**

  The material assigned to the "inside" triangles of the fracture pieces. These are the triangles that DinoFracture creates. The surface triangles of the original mesh retain their materials.
- bool OptimizeMaterialUsage = true

  If true, newly generated triangles using the "InsideMaterial" will attempt to be part of the same existing material in th mesh.
- GameObject **FractureTemplate**

  This game object will be cloned for each facture piece. It is required to have a MeshFilter component. If a MeshCollider component is added, it will be assigned the fracture mesh.
- Transform **PiecesParent**

  The parent of the generated pieces. Each fracture produces a root object with fracture pieces (clones of Fracture↩ Template) as children. The root object is parented to PiecesParent.
- FractureType **FractureType** = FractureType.Shatter

  The type of fracture to produce when Fracture() is called.
- SlicePlaneSerializable[ ] SlicePlanes

  The planes to use when slicing the mesh. Not used when fracturing into pieces.
- int **NumFracturePieces** = 5

  The number of fracture pieces generated per iteration. Fault lines are spread evenly around the fracture point. The number of total pieces generated is NumFracturePieces ^ NumIterations.
- int NumIterations = 2

  The number of passes of fracturing. Using lower piece count with a higher iteration count is computationally faster than a higher piece count with a lower iteration count. Ex: 5 pieces with 2 iterations is faster than 25 pieces and 1 iteration. The downside to using more iterations is fractures can become less uniform. In general, keep this number below 4. The number of total pieces generated is NumFracturePieces ^ NumIterations.

- bool EvenlySizedPieces = true

If true, the engine will attempt to make all the randomly generated pieces roughly the same size. This adds a little processing time to the fracture.

- int NumGenerations = 1

  To allow for fracture pieces to be further fractured, the FractureTemplate should have a FractureGeometry component. NumGenerations dictates how many times the geometry can be re-fractured. The count is decremented and passed on to the component in each generated piece. Ex: A value of 2 means this piece can be fractured and each generated piece can be fractured. The second generation of fractures cannot be fractured further.

- float **FractureRadius**

  A value between 0 and 1 that indicates how clustered the fracture lines are. A value of 0 or 1 means fractures are evenly distributed across the mesh. A value between means they are clustered within a percentage of the mesh bounds. Ex: a value of 0.3 means fractures are clustered around the fracture point in a volume 30% the size of the mesh. Pre-fracture geometry typically has this value set to 0 or 1 because there isn't always a pre-determined point of fracture.

- SizeSerializable **FractureSize**

  The approximate size of fractures to create during shatter. When combined with a position through a call to Fracture(), the shatter planes will be clustered within this bounds.

- FractureUVScale **UVScale** = FractureUVScale.Piece

  If set to EntireMesh, the UV map for each inside triangle will be mapped to a box the size of the original mesh. If set to piece, inside triangles will be mapped to a box the size of the individual fracture piece.

- UVBoundsSerializable UVBounds = new UVBoundsSerializable(Vector2.zero, Vector2.one)

  Final 'inside' triangles will be remapped to be within this range. This does not affect the UVs on the incoming mesh and works with any value set for the UVScale.

- bool **DistributeMass** = true

  If true and both this game object and the FractureTemplate have a RigidBody component, each fracture piece will have a mass set to a value proportional to its volume. That is, the density of the fracture piece will equal the density of the original mesh. If false, the mass property goes untouched.

- bool SeparateDisjointPieces = false

  If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.

- int **RandomSeed** = 0

  The random seed to use when initiating the fracture. If set to zero, then the system clock will be used to create a random seed.

- OnFractureEvent **OnFractureCompleted**

  Unity event that fires whenever a fracture on this object completes.

- bool **SkipMeshCleaning** = false

  Not recommended to be set. But if set to true, no mesh cleaning will occur during the fracture process. This has no effect if the mesh is already clean.

Properties

- MeshValidity **MeshValidity** `[get]`

  Used to determine if the mesh is of known good quality to fracture.

- bool **IsProcessingFracture** `[get]`

  Are we in the middle of computing a fracture for this object?

6.16.1 Detailed Description

This is the base class for the PreFractureGeometry and RuntimeFractureGeometry components. As such, it is not intended to be directly added to any game object even though fracture initiator components rely on it.

6.16.2 Member Function Documentation

6.16.2.1 CreateSlicePlane()

```
static SlicePlaneSerializable DinoFracture.FractureGeometry.CreateSlicePlane (
            Plane worldPlane,
            Transform targetGameObject )  [static]
```

This will create a valid slice plane for slicing a mesh.

While valid, it is not intended to be displayed in the editor and is meant for runtime use.

6.16.2.2 ForceValidGeometry()

```
void DinoFracture.FractureGeometry.ForceValidGeometry ( )
```

This will force the geometry to pass all validity checks.

This method can be useful to call in very custom scenarios of fracturing through script when you know the mesh you are passing in is valid.

6.16.2.3 Fracture() [1/2]

```
AsyncFractureResult DinoFracture.FractureGeometry.Fracture ( )
```

Initiate a fracture at the origin.

Returns

6.16.2.4 Fracture() [2/2]

```
AsyncFractureResult DinoFracture.FractureGeometry.Fracture (
            Vector3 localPos )
```

Initiate a fracture at the specified position relative to this object.

Parameters

localPos	

Returns

6.16.2.5 FractureAndForget() [1/2]

```
void DinoFracture.FractureGeometry.FractureAndForget ( )
```

Initiate a fracture at the origin and does not return a handle to the async operation.

The OnFracture() callback will still fire. This method is compatible with Unity events.

6.16.2.6 FractureAndForget() [2/2]

```
void DinoFracture.FractureGeometry.FractureAndForget (
            Vector3 localPos )
```

Initiate a fracture at the specified position relative to this object and does not return a handle to the async operation.

The OnFracture() callback will still fire. This method is compatible with Unity events.

6.16.3 Member Data Documentation

6.16.3.1 EvenlySizedPieces

```
bool DinoFracture.FractureGeometry.EvenlySizedPieces = true
```

If true, the engine will attempt to make all the randomly generated pieces roughly the same size. This adds a little processing time to the fracture.

Do not set this to true if FractureRadius > 0.

6.16.3.2 NumGenerations

```
int DinoFracture.FractureGeometry.NumGenerations = 1
```

To allow for fracture pieces to be further fractured, the FractureTemplate should have a FractureGeometry component. NumGenerations dictates how many times the geometry can be re-fractured. The count is decremented and passed on to the component in each generated piece. Ex: A value of 2 means this piece can be fractured and each generated piece can be fractured. The second generation of fractures cannot be fractured further.

Specify a negative value on the main piece to allow for infinite repeated fractures

6.16.3.3 NumIterations

```
int DinoFracture.FractureGeometry.NumIterations = 2
```

The number of passes of fracturing. Using lower piece count with a higher iteration count is computationally faster than a higher piece count with a lower iteration count. Ex: 5 pieces with 2 iterations is faster than 25 pieces and 1 iteration. The downside to using more iterations is fractures can become less uniform. In general, keep this number below 4. The number of total pieces generated is NumFracturePieces $^{\wedge}$ NumIterations.

It is recommended you use an iteration count of 1 when $0 <$ FractureRadius < 1.

6.16.3.4 OptimizeMaterialUsage

```
bool DinoFracture.FractureGeometry.OptimizeMaterialUsage = true
```

If true, newly generated triangles using the "InsideMaterial" will attempt to be part of the same existing material in th mesh.

If false, newly generated triangles will always be put in a new material placed after all *used* existing materials. Additionally, new materials will continue to be appended upon further fractures.

6.16.3.5 SeparateDisjointPieces

```
bool DinoFracture.FractureGeometry.SeparateDisjointPieces = false
```

If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.

This process can be slow. It is recommended to be off for runtime fractures unless there is a good chance of disjoint pieces.

6.16.3.6 SlicePlanes

```
SlicePlaneSerializable [] DinoFracture.FractureGeometry.SlicePlanes
```

The planes to use when slicing the mesh. Not used when fracturing into pieces.

Each slice plane must be in local space. Use FractureGeometry.CreateSlicePlane() to create a compatible local space plane from a Unity world space plane.

6.16.3.7 UVBounds

```
UVBoundsSerializable DinoFracture.FractureGeometry.UVBounds = new UVBoundsSerializable(Vector2.↩
zero, Vector2.one)
```

Final 'inside' triangles will be remapped to be within this range. This does not affect the UVs on the incoming mesh and works with any value set for the UVScale.

This can be used with an atlas texture to constrain the generated triangles to use only a specific portion of the texture map.

6.17 DinoFracture.FractureOnClick Class Reference

Casts a simple mouse ray on left click and calls Fracture() on the hit collider game object.

Inherits MonoBehaviour.

6.17.1 Detailed Description

Casts a simple mouse ray on left click and calls Fracture() on the hit collider game object.

6.18 DinoFracture.FractureOnCollision Class Reference

This component will cause a fracture to happen at the point of impact.

Inherits MonoBehaviour.

Public Attributes

- float **ForceThreshold**

  The minimum amount of force required to fracture this object. Set to 0 to have any amount of force cause the fracture.
- float **ForceFalloffRadius** = 1.0f

  Falloff radius for transferring the force of the impact to the resulting pieces. Any piece outside of this falloff from the point of impact will have no additional impulse set on it.
- bool **AdjustForKinematic** = true

  If true and this is a kinematic body, an impulse will be applied to the colliding body to counter the effects of hitting a kinematic body. If false and this is a kinematic body, the colliding body will bounce off as if this were an unmovable wall.
- bool ConsistentImpactForce = false

  If true, the force calculations will become more consistent with all incoming collisions. The force will be the same given the same incoming rigid body properties instead of changing as this mass changes.
- LayerMask **CollidableLayers** = (LayerMask)(-1)

  The collision layers that are allowed to cause a fracture.
- bool **OutputDebugCollisionInfo** = false

  If true, collision energies will be output in the editor to help tune required force thresholds.

6.18.1 Detailed Description

This component will cause a fracture to happen at the point of impact.

6.18.2 Member Data Documentation

6.18.2.1 ConsistentImpactForce

```
bool DinoFracture.FractureOnCollision.ConsistentImpactForce = false
```

If true, the force calculations will become more consistent with all incoming collisions. The force will be the same given the same incoming rigid body properties instead of changing as this mass changes.

Recommended to set to true for consistency. Defaults to false for backwards compatibility.

The incoming force calculations will change, requiring re-tweaking of the ForceThreshold value.

6.19 DinoFracture.FractureOnInput Class Reference

Apply this on the fracturing game object. When the specified key is pressed, the object will fracture.

Inherits MonoBehaviour.

6.19.1 Detailed Description

Apply this on the fracturing game object. When the specified key is pressed, the object will fracture.

6.20 DinoFracture.FractureOnParticleCollision Class Reference

This component will cause a fracture to happen at the point of impact with a particle.

Inherits MonoBehaviour.

Public Attributes

- float **VelocityThreshold**

  The minimum velocity of incoming particles required to fracture this object. Set to 0 to have any amount of velocity cause the fracture.
- float **ParticleMass** = 1e-4f

  The mass of an individual particle to allow transferring of forces to the fracture pieces. If zero, no forces will be transferred.
- float **ForceFalloffRadius** = 1.0f

  Falloff radius for transferring the force of the impact to the resulting pieces. Any piece outside of this falloff from the point of impact will have no additional impulse set on it.
- LayerMask **CollidableLayers** = (LayerMask)int.MaxValue

  The collision layers of the particle system that are allowed to cause a fracture.

6.20.1 Detailed Description

This component will cause a fracture to happen at the point of impact with a particle.

Note: Particle collisions won't be activated unless "Send Collision Messages" is checked.

6.21 DinoFracture.FractureResult Class Reference

The result of a fracture.

Public Member Functions

- IReadOnlyList< FracturedMesh > GetMeshes ()
  Returns a list of pieces produced by the fracture.

Properties

- Bounds **EntireMeshBounds** `[get]`
  Bounds of the original mesh, in local space.

6.21.1 Detailed Description

The result of a fracture.

6.21.2 Member Function Documentation

6.21.2.1 GetMeshes()

```
IReadOnlyList< FracturedMesh > DinoFracture.FractureResult.GetMeshes ( )
```

Returns a list of pieces produced by the fracture.

Returns

6.22 DinoFracture.GlueEdgeOnFracture Class Reference

If the fracture pieces intersects with a specified trigger when created, the rigid body is destroyed and the piece becomes static. Otherwise, the piece will turn on gravity. It's best used if the FractureTemplate's rigid body is set to not use gravity initially.

Inherits MonoBehaviour.

Public Attributes

- string **CollisionTag** = ""

  The piece will be glued if it intersects a trigger with this collision tag. Set to empty to allow any trigger to glue the piece.
- bool **DestroyRigidBody** = true

  If true, the rigid body will be destroyed when glued. This will prevent further fractures that rely on collision events, but will reduce processing on Unity.

6.22.1 Detailed Description

If the fracture pieces intersects with a specified trigger when created, the rigid body is destroyed and the piece becomes static. Otherwise, the piece will turn on gravity. It's best used if the FractureTemplate's rigid body is set to not use gravity initially.

6.23 DinoFracture.NotifyOnFracture Class Reference

When added to the same game object as the FractureGeometry, this script can be used to notify external game objects of this object's fracture completion. The external objects need a script with the "OnFracture" callback method.

Inherits MonoBehaviour.

Public Attributes

- GameObject[ ] **GameObjects** = new GameObject[1]

  The array of game objects to notify. They do not need to be in this object's tree.

6.23.1 Detailed Description

When added to the same game object as the FractureGeometry, this script can be used to notify external game objects of this object's fracture completion. The external objects need a script with the "OnFracture" callback method.

6.24 DinoFracture.FractureGeometry.OnFractureEvent Class Reference

OnFracture() Unity event wrapper.

Inherits UnityEngine.Events.UnityEvent< OnFractureEventArgs >.

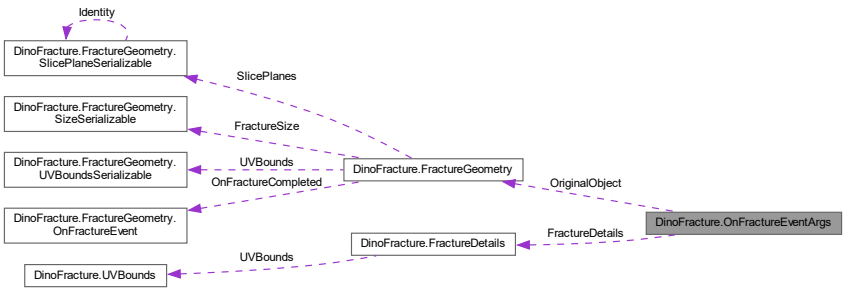
6.24.1 Detailed Description

OnFracture() Unity event wrapper.

6.25 DinoFracture.OnFractureEventArgs Class Reference

Argument passed to OnFracture message.

Collaboration diagram for DinoFracture.OnFractureEventArgs:



Public Member Functions

- IEnumerable< UnityEngine.Mesh > GetMeshes ()

  Returns an enumerable of just the generated Unity meshes.

Public Attributes

- FractureGeometry **OriginalObject**

  The object that fractured.
- Bounds **OriginalMeshBounds**

  The bounds of the original mesh.
- GameObject **FracturePiecesRootObject**

  The root of the pieces of the resulting fracture.
- FractureDetails **FractureDetails**

  The parameters used for the fracture.

6.25.1 Detailed Description

Argument passed to OnFracture message.

6.25.2 Member Function Documentation

6.25.2.1 GetMeshes()

```
IEnumerable< UnityEngine.Mesh > DinoFracture.OnFractureEventArgs.GetMeshes ( )
```

Returns an enumerable of just the generated Unity meshes.

Returns

6.26 DinoFracture.PlaySoundOnFracture Class Reference

An object with this component will play the audio source when fractured.

Inherits MonoBehaviour.

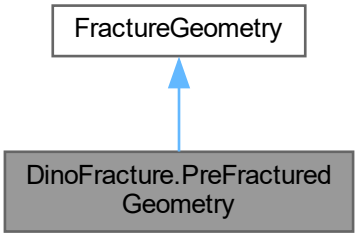
6.26.1 Detailed Description

An object with this component will play the audio source when fractured.

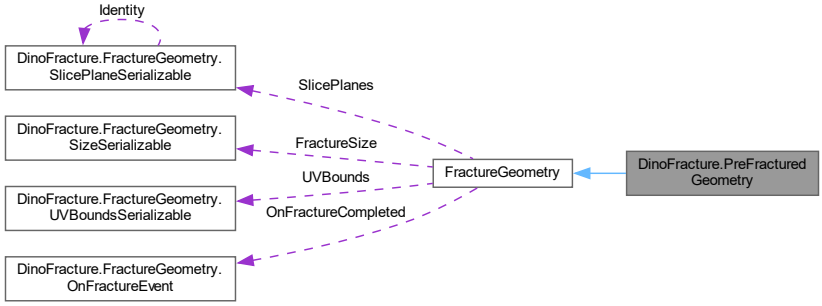
6.27 DinoFracture.PreFracturedGeometry Class Reference

Apply this component to any game object you wish to pre-fracture. Pre-fracturing is a way of baking fracture pieces into the scene. Each time the object is fractured, the same set of pieces will activate. This is very useful when creating a large number of pieces or high poly meshes, which would be too slow to create at runtime. The pieces will be in the scene as a disabled root object with piece children. When the object is fractured, those pieces will activate.

Inheritance diagram for DinoFracture.PreFracturedGeometry:



Collaboration diagram for DinoFracture.PreFracturedGeometry:



Public Member Functions

- void **Prime** ()

  Primes the pre-fractured pieces when the game starts by activating them and then deactivating them. This avoids a large delay on fracture if there are a lot of rigid bodies.

Public Member Functions inherited from DinoFracture.FractureGeometry

- void FractureAndForget ()

  Initiate a fracture at the origin and does not return a handle to the async operation.
- void FractureAndForget (Vector3 localPos)

  Initiate a fracture at the specified position relative to this object and does not return a handle to the async operation.
- AsyncFractureResult Fracture ()

  Initiate a fracture at the origin.
- AsyncFractureResult Fracture (Vector3 localPos)

  Initiate a fracture at the specified position relative to this object.
- MeshTopologyError **CheckMeshValidity** ()

  This is called automatically when viewing the component in the inspector. However, it should be called whenever the mesh changes through other means.
- void ForceValidGeometry ()

  This will force the geometry to pass all validity checks.
- List< EdgeError > **GetMeshEdgeErrors** ()

  Returns any bad mesh edges. Used by the editor script for debugging.

Public Attributes

- GameObject GeneratedPieces

  A reference to the root of the pre-fractured pieces. This is not normally set manually. Instead, you press the "Create Fractures" button in the inspector window to generate the fracture immediately.

- Bounds **EntireMeshBounds**

  The encapsulating bounds of the entire set of pieces. In local space.

Public Attributes inherited from DinoFracture.FractureGeometry

- Material **InsideMaterial**

  The material assigned to the "inside" triangles of the fracture pieces. These are the triangles that DinoFracture creates. The surface triangles of the original mesh retain their materials.
- bool OptimizeMaterialUsage = true

  If true, newly generated triangles using the "InsideMaterial" will attempt to be part of the same existing material in th mesh.
- GameObject **FractureTemplate**

  This game object will be cloned for each facture piece. It is required to have a MeshFilter component. If a MeshCollider component is added, it will be assigned the fracture mesh.
- Transform **PiecesParent**

  The parent of the generated pieces. Each fracture produces a root object with fracture pieces (clones of Fracture↩ Template) as children. The root object is parented to PiecesParent.
- FractureType **FractureType** = FractureType.Shatter

  The type of fracture to produce when Fracture() is called.
- SlicePlaneSerializable[ ] SlicePlanes

  The planes to use when slicing the mesh. Not used when fracturing into pieces.

- int **NumFracturePieces** = 5

  The number of fracture pieces generated per iteration. Fault lines are spread evenly around the fracture point. The number of total pieces generated is NumFracturePieces ^ NumIterations.

- int NumIterations = 2

  The number of passes of fracturing. Using lower piece count with a higher iteration count is computationally faster than a higher piece count with a lower iteration count. Ex: 5 pieces with 2 iterations is faster than 25 pieces and 1 iteration. The downside to using more iterations is fractures can become less uniform. In general, keep this number below 4. The number of total pieces generated is NumFracturePieces ^ NumIterations.

- bool EvenlySizedPieces = true

  If true, the engine will attempt to make all the randomly generated pieces roughly the same size. This adds a little processing time to the fracture.

- int NumGenerations = 1

  To allow for fracture pieces to be further fractured, the FractureTemplate should have a FractureGeometry component. NumGenerations dictates how many times the geometry can be re-fractured. The count is decremented and passed on to the component in each generated piece. Ex: A value of 2 means this piece can be fractured and each generated piece can be fractured. The second generation of fractures cannot be fractured further.

- float **FractureRadius**

  A value between 0 and 1 that indicates how clustered the fracture lines are. A value of 0 or 1 means fractures are evenly distributed across the mesh. A value between means they are clustered within a percentage of the mesh bounds. Ex: a value of 0.3 means fractures are clustered around the fracture point in a volume 30% the size of the mesh. Pre-fracture geometry typically has this value set to 0 or 1 because there isn't always a pre-determined point of fracture.

- SizeSerializable **FractureSize**

  The approximate size of fractures to create during shatter. When combined with a position through a call to Fracture(), the shatter planes will be clustered within this bounds.

- FractureUVScale **UVScale** = FractureUVScale.Piece

  If set to EntireMesh, the UV map for each inside triangle will be mapped to a box the size of the original mesh. If set to piece, inside triangles will be mapped to a box the size of the individual fracture piece.

- UVBoundsSerializable UVBounds = new UVBoundsSerializable(Vector2.zero, Vector2.one)

  Final 'inside' triangles will be remapped to be within this range. This does not affect the UVs on the incoming mesh and works with any value set for the UVScale.

- bool **DistributeMass** = true

  If true and both this game object and the FractureTemplate have a RigidBody component, each fracture piece will have a mass set to a value proportional to its volume. That is, the density of the fracture piece will equal the density of the original mesh. If false, the mass property goes untouched.

- bool SeparateDisjointPieces = false

  If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.

- int **RandomSeed** = 0

  The random seed to use when initiating the fracture. If set to zero, then the system clock will be used to create a random seed.

- OnFractureEvent **OnFractureCompleted**

  Unity event that fires whenever a fracture on this object completes.

- bool **SkipMeshCleaning** = false

  Not recommended to be set. But if set to true, no mesh cleaning will occur during the fracture process. This has no effect if the mesh is already clean.

Additional Inherited Members

Static Public Member Functions inherited from DinoFracture.FractureGeometry

- static SlicePlaneSerializable CreateSlicePlane (Plane worldPlane, Transform targetGameObject)

  This will create a valid slice plane for slicing a mesh.

Properties inherited from DinoFracture.FractureGeometry

- MeshValidity **MeshValidity** `[get]`

  Used to determine if the mesh is of known good quality to fracture.
- bool **IsProcessingFracture** `[get]`

  Are we in the middle of computing a fracture for this object?

6.27.1 Detailed Description

Apply this component to any game object you wish to pre-fracture. Pre-fracturing is a way of baking fracture pieces into the scene. Each time the object is fractured, the same set of pieces will activate. This is very useful when creating a large number of pieces or high poly meshes, which would be too slow to create at runtime. The pieces will be in the scene as a disabled root object with piece children. When the object is fractured, those pieces will activate.

6.27.2 Member Data Documentation

6.27.2.1 GeneratedPieces

`GameObject DinoFracture.PreFracturedGeometry.GeneratedPieces`

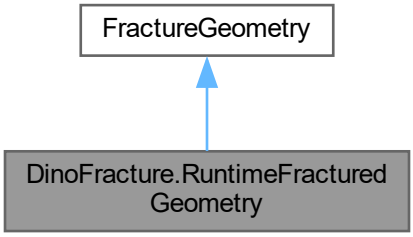
A reference to the root of the pre-fractured pieces. This is not normally set manually. Instead, you press the “Create Fractures” button in the inspector window to generate the fracture immediately.

The “Create Fractures” button is only intended to be used in edit mode; not game mode.

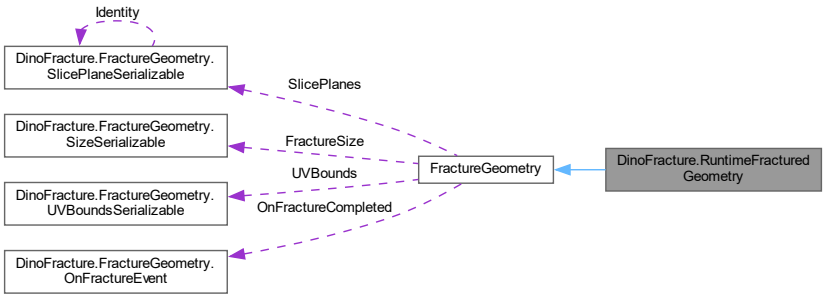
6.28 DinoFracture.RuntimeFracturedGeometry Class Reference

Apply this component to any game object you wish to fracture while running in game mode. Runtime fractures will produce a unique set of pieces with each fracture. However, this is at the cost of computational time. It is recommended that both the piece count and poly count are kept low. This component is most effective when FractureRadius is set to a value in-between 0 and 1.

Inheritance diagram for DinoFracture.RuntimeFracturedGeometry:



Collaboration diagram for DinoFracture.RuntimeFracturedGeometry:



Public Attributes

- bool Asynchronous = true

  If true, the fracture operation is performed on a background thread and may not be finished by the time the fracture call returns. A couple of frames can go by from the time of the fracture to when the pieces are ready. If this is false, the fracture will guaranteed be complete by the end of the call, but the game will be paused while the fractures are being created.

Public Attributes inherited from DinoFracture.FractureGeometry

- Material **InsideMaterial**

  The material assigned to the "inside" triangles of the fracture pieces. These are the triangles that DinoFracture creates. The surface triangles of the original mesh retain their materials.
- bool OptimizeMaterialUsage = true

  If true, newly generated triangles using the "InsideMaterial" will attempt to be part of the same existing material in th mesh.
- GameObject **FractureTemplate**

  This game object will be cloned for each facture piece. It is required to have a MeshFilter component. If a MeshCollider component is added, it will be assigned the fracture mesh.
- Transform **PiecesParent**

  The parent of the generated pieces. Each fracture produces a root object with fracture pieces (clones of Fracture↩ Template) as children. The root object is parented to PiecesParent.
- FractureType **FractureType** = FractureType.Shatter

  The type of fracture to produce when Fracture() is called.
- SlicePlaneSerializable[ ] SlicePlanes

  The planes to use when slicing the mesh. Not used when fracturing into pieces.
- int **NumFracturePieces** = 5

  The number of fracture pieces generated per iteration. Fault lines are spread evenly around the fracture point. The number of total pieces generated is NumFracturePieces ^ NumIterations.
- int NumIterations = 2

  The number of passes of fracturing. Using lower piece count with a higher iteration count is computationally faster than a higher piece count with a lower iteration count. Ex: 5 pieces with 2 iterations is faster than 25 pieces and 1 iteration. The downside to using more iterations is fractures can become less uniform. In general, keep this number below 4. The number of total pieces generated is NumFracturePieces ^ NumIterations.

- bool EvenlySizedPieces = true

If true, the engine will attempt to make all the randomly generated pieces roughly the same size. This adds a little processing time to the fracture.

- int NumGenerations = 1

  To allow for fracture pieces to be further fractured, the FractureTemplate should have a FractureGeometry component. NumGenerations dictates how many times the geometry can be re-fractured. The count is decremented and passed on to the component in each generated piece. Ex: A value of 2 means this piece can be fractured and each generated piece can be fractured. The second generation of fractures cannot be fractured further.

- float **FractureRadius**

  A value between 0 and 1 that indicates how clustered the fracture lines are. A value of 0 or 1 means fractures are evenly distributed across the mesh. A value between means they are clustered within a percentage of the mesh bounds. Ex: a value of 0.3 means fractures are clustered around the fracture point in a volume 30% the size of the mesh. Pre-fracture geometry typically has this value set to 0 or 1 because there isn't always a pre-determined point of fracture.

- SizeSerializable **FractureSize**

  The approximate size of fractures to create during shatter. When combined with a position through a call to Fracture(), the shatter planes will be clustered within this bounds.

- FractureUVScale **UVScale** = FractureUVScale.Piece

  If set to EntireMesh, the UV map for each inside triangle will be mapped to a box the size of the original mesh. If set to piece, inside triangles will be mapped to a box the size of the individual fracture piece.

- UVBoundsSerializable UVBounds = new UVBoundsSerializable(Vector2.zero, Vector2.one)

  Final 'inside' triangles will be remapped to be within this range. This does not affect the UVs on the incoming mesh and works with any value set for the UVScale.

- bool **DistributeMass** = true

  If true and both this game object and the FractureTemplate have a RigidBody component, each fracture piece will have a mass set to a value proportional to its volume. That is, the density of the fracture piece will equal the density of the original mesh. If false, the mass property goes untouched.

- bool SeparateDisjointPieces = false

  If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.

- int **RandomSeed** = 0

  The random seed to use when initiating the fracture. If set to zero, then the system clock will be used to create a random seed.

- OnFractureEvent **OnFractureCompleted**

  Unity event that fires whenever a fracture on this object completes.

- bool **SkipMeshCleaning** = false

  Not recommended to be set. But if set to true, no mesh cleaning will occur during the fracture process. This has no effect if the mesh is already clean.

Additional Inherited Members

Public Member Functions inherited from DinoFracture.FractureGeometry

- void FractureAndForget ()

  Initiate a fracture at the origin and does not return a handle to the async operation.

- void FractureAndForget (Vector3 localPos)

  Initiate a fracture at the specified position relative to this object and does not return a handle to the async operation.

- AsyncFractureResult Fracture ()

  Initiate a fracture at the origin.

- AsyncFractureResult Fracture (Vector3 localPos)

  Initiate a fracture at the specified position relative to this object.

- MeshTopologyError **CheckMeshValidity** ()

This is called automatically when viewing the component in the inspector. However, it should be called whenever the mesh changes through other means.

- void ForceValidGeometry ()

  This will force the geometry to pass all validity checks.

- List< EdgeError > **GetMeshEdgeErrors** ()

  Returns any bad mesh edges. Used by the editor script for debugging.

Static Public Member Functions inherited from DinoFracture.FractureGeometry

- static SlicePlaneSerializable CreateSlicePlane (Plane worldPlane, Transform targetGameObject)

  This will create a valid slice plane for slicing a mesh.

Properties inherited from DinoFracture.FractureGeometry

- MeshValidity **MeshValidity** `[get]`

  Used to determine if the mesh is of known good quality to fracture.

- bool **IsProcessingFracture** `[get]`

  Are we in the middle of computing a fracture for this object?

6.28.1 Detailed Description

Apply this component to any game object you wish to fracture while running in game mode. Runtime fractures will produce a unique set of pieces with each fracture. However, this is at the cost of computational time. It is recommended that both the piece count and poly count are kept low. This component is most effective when FractureRadius is set to a value in-between 0 and 1.

6.28.2 Member Data Documentation

6.28.2.1 Asynchronous

```
bool DinoFracture.RuntimeFracturedGeometry.Asynchronous = true
```

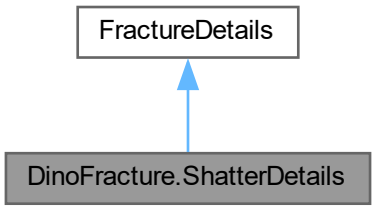
If true, the fracture operation is performed on a background thread and may not be finished by the time the fracture call returns. A couple of frames can go by from the time of the fracture to when the pieces are ready. If this is false, the fracture will guaranteed be complete by the end of the call, but the game will be paused while the fractures are being created.

It is recommended to set asynchronous to true whenever possible.

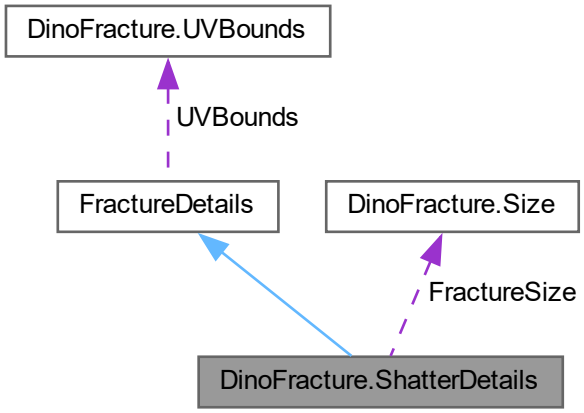
6.29 DinoFracture.ShatterDetails Class Reference

Required information needed by the engine to produce a fracture.

Inheritance diagram for DinoFracture.ShatterDetails:



Collaboration diagram for DinoFracture.ShatterDetails:



Public Member Functions

- override bool IsValid ()

  Returns true if the details are filled in correctly, false otherwise.

- virtual bool IsValid ()

  Returns true if the details are filled in correctly, false otherwise.

Public Attributes

- int **NumPieces**

  The number of new pieces to produce per iteration. The total number of pieces produced by this fracture will be NumPieces ^ NumIterations.
- int **NumIterations**

  The number of fracture iterations. The total number of pieces produced by this fracture will be NumPieces ^ NumIterations.
- bool **EvenlySizedPieces**

  If true, the engine will attempt to make all the randomly generated pieces roughly the same size. This adds a little processing time to the fracture.
- Vector3 **FractureCenter**

  The center of the fracture.
- Size **FractureSize**

  The bounds for the generated shatter fracture planes. If 0 sized, the bounds of the mesh are used.
- int **RandomSeed**

  The random seed to use when initiating the fracture. If set to zero, then the system clock will be used to create a random seed.

Public Attributes inherited from DinoFracture.FractureDetails

- UnityEngine.Mesh **Mesh**

  The mesh to fracture.
- Vector3 **MeshScale**

  The scale of the mesh's game object. The meshes of fracture pieces will be scaled by this amount to allow their game object's scales to be one.
- FractureUVScale **UVScale**

  Scaling algorithm used on triangles produced during the fracture.
- UVBounds **UVBounds** = new UVBounds(Vector2.zero, Vector2.one)

  Final 'inside' triangles will be remapped to be within this range after the UV scale is applied.
- FractureIssueResolution IssueResolution

  How to deal with potentially poorly generated pieces.
- bool **Asynchronous**

  If true, fracturing is done on a background thread and results may not be ready by the time FractureBuilder.Fracture() finishes. If false, fracturing is guaranteed to be done by the time FractureBuilder.Fracture() finishes.
- int InsideMaterialIndex

  The material / sub-mesh index that newly formed triangles should be put in.
- bool SeparateDisjointPieces

  If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.
- MeshValidity **Validity** = MeshValidity.Unknown

  Highly recommended to call.

Additional Inherited Members

Properties inherited from DinoFracture.FractureDetails

- int **FractureFrame** `[get, set]`

  The frame the fracture started on. Can be used to group separate fractures that should be treated as one.

6.29.1 Detailed Description

Required information needed by the engine to produce a fracture.

6.29.2 Member Function Documentation

6.29.2.1 IsValid()

```
override bool DinoFracture.ShatterDetails.IsValid ( )  [virtual]
```

Returns true if the details are filled in correctly, false otherwise.

Returns

Reimplemented from DinoFracture.FractureDetails.

6.30 DinoFracture.Size Struct Reference

Structure that defines the bounds of the shatter fracture planes.

Public Member Functions

- Vector3 **GetWorldSpaceSize** (Vector3 boundsSize)
  Returns the bounds size in world space units.

Public Attributes

- SizeSpace **Space**
  The space Value is specified in.
- float **Value**
  The size of the bounds in the specified space.

6.30.1 Detailed Description

Structure that defines the bounds of the shatter fracture planes.

If Value is set to 0, the bounds are set to the mesh's bounds.

6.31 DinoFracture.FractureGeometry.SizeSerializable Struct Reference

Unity cannot handle the serializable attribute on types defined in dlls. This wraps a DinoFracture.Size type.

Public Attributes

- SizeSpace **Space**

  What the bounds is relative to.
- float **Value**

  The size of the bounds, in the specified space.

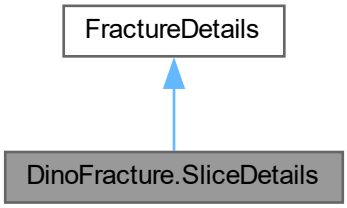
6.31.1 Detailed Description

Unity cannot handle the serializable attribute on types defined in dlls. This wraps a DinoFracture.Size type.

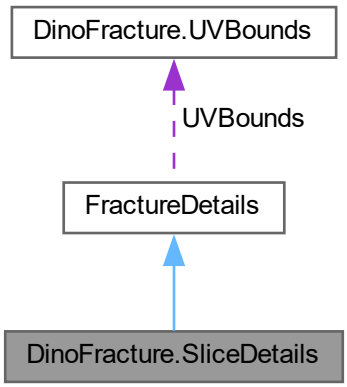
6.32 DinoFracture.SliceDetails Class Reference

Required information needed by the engine to slice a mesh.

Inheritance diagram for DinoFracture.SliceDetails:



Collaboration diagram for DinoFracture.SliceDetails:



Public Member Functions

- override bool IsValid ()

  Returns true if the details are filled in correctly, false otherwise.

- virtual bool IsValid ()

  Returns true if the details are filled in correctly, false otherwise.

Public Attributes

- readonly List< SlicePlane > **SlicingPlanes** = new List<SlicePlane>()

  User defined slicing planes.

Public Attributes inherited from DinoFracture.FractureDetails

- UnityEngine.Mesh **Mesh**

  The mesh to fracture.
- Vector3 **MeshScale**

  The scale of the mesh's game object. The meshes of fracture pieces will be scaled by this amount to allow their game object's scales to be one.
- FractureUVScale **UVScale**

  Scaling algorithm used on triangles produced during the fracture.
- UVBounds **UVBounds** = new UVBounds(Vector2.zero, Vector2.one)

  Final 'inside' triangles will be remapped to be within this range after the UV scale is applied.
- FractureIssueResolution IssueResolution

  How to deal with potentially poorly generated pieces.
- bool **Asynchronous**

  If true, fracturing is done on a background thread and results may not be ready by the time FractureBuilder.Fracture() finishes. If false, fracturing is guaranteed to be done by the time FractureBuilder.Fracture() finishes.

- int InsideMaterialIndex

  The material / sub-mesh index that newly formed triangles should be put in.
- bool SeparateDisjointPieces

  If true, a final pass will be done to separate out meshes that are not physically connected. This can only happen when the mesh has concave parts.
- MeshValidity **Validity** = MeshValidity.Unknown

  Highly recommended to call.

Additional Inherited Members

Properties inherited from DinoFracture.FractureDetails

- int **FractureFrame** `[get, set]`

  The frame the fracture started on. Can be used to group separate fractures that should be treated as one.

6.32.1 Detailed Description

Required information needed by the engine to slice a mesh.

6.32.2 Member Function Documentation

6.32.2.1 IsValid()

```
override bool DinoFracture.SliceDetails.IsValid ( )  [virtual]
```

Returns true if the details are filled in correctly, false otherwise.

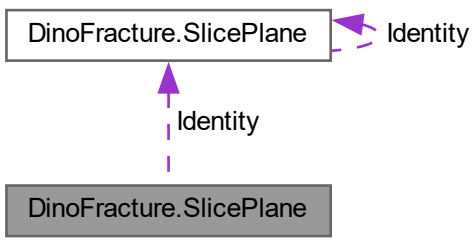
Returns

Reimplemented from DinoFracture.FractureDetails.

6.33 DinoFracture.SlicePlane Struct Reference

Defines a plane that slices the mesh in half.

Collaboration diagram for DinoFracture.SlicePlane:



Public Member Functions

- UnityEngine.Plane **ToPlane** ()

  Converts this object to a Unity plane.

Public Attributes

- Vector3 **Position**

  Local space position of the plane.
- Quaternion Rotation

  Local space rotation of the plane.
- float Scale

  Scale of the plane.

Static Public Attributes

- static readonly SlicePlane **Identity**

  Default, "identity" plane.

6.33.1 Detailed Description

Defines a plane that slices the mesh in half.

Values are in the mesh's local space.

6.33.2 Member Data Documentation

6.33.2.1 Rotation

```
Quaternion DinoFracture.SlicePlane.Rotation
```

Local space rotation of the plane.

Z dir is the plane's normal.

6.33.2.2 Scale

```
float DinoFracture.SlicePlane.Scale
```

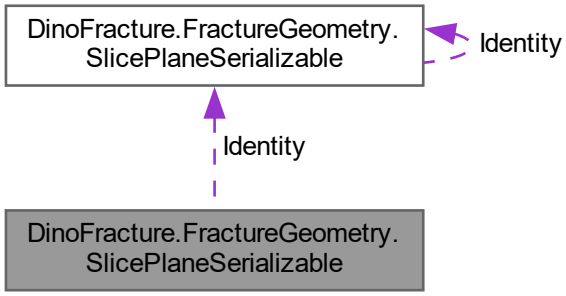
Scale of the plane.

This is only used to display it in the editor. When slicing, the plane will be treated as infinite in size.

6.34 DinoFracture.FractureGeometry.SlicePlaneSerializable Struct Reference

Unity cannot handle the serializable attribute on types defined in dlls. So, we have to duplicate the SlicePlane structure here in order to save it.

Collaboration diagram for DinoFracture.FractureGeometry.SlicePlaneSerializable:



Public Member Functions

- SlicePlane ToSlicePlane ()

  Converts this serialization helper to a normal slice plane.

6.34.1 Detailed Description

Unity cannot handle the serializable attribute on types defined in dlls. So, we have to duplicate the SlicePlane structure here in order to save it.

6.34.2 Member Function Documentation

6.34.2.1 ToSlicePlane()

```
SlicePlane DinoFracture.FractureGeometry.SlicePlaneSerializable.ToSlicePlane ( )
```

Converts this serialization helper to a normal slice plane.

Returns

6.35 DinoFracture.TransferJointsOnFracture Class Reference

When this object is fractured, the joint component on the object will be copied to this piece if this piece is sufficiently close to the joint position. Without this component, joints are broken after fracturing.

Inherits MonoBehaviour.

Public Attributes

- Transform **IncomingJointsSearchRoot**

  The tree to crawl in search for joints of other objects that need to be transferred to this joint. This search root should be as scoped as possible.
- float **DistanceTolerance** = 0.05f

  How close this object must be to the joint in order to transfer. The larger the number, the more pieces will have joints transferred.

6.35.1 Detailed Description

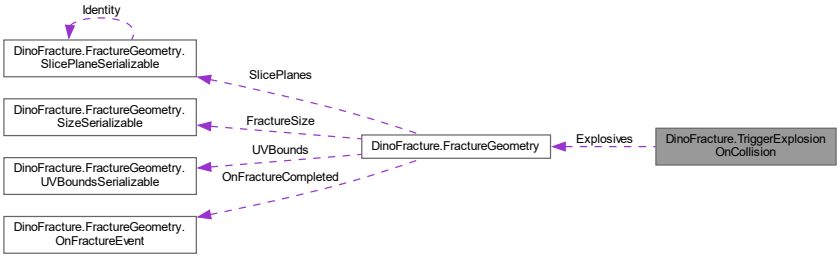
When this object is fractured, the joint component on the object will be copied to this piece if this piece is sufficiently close to the joint position. Without this component, joints are broken after fracturing.

6.36 DinoFracture.TriggerExplosionOnCollision Class Reference

Triggers a fracture + explosion when this game object is collided with.

Inherits MonoBehaviour.

Collaboration diagram for DinoFracture.TriggerExplosionOnCollision:



Public Attributes

- FractureGeometry[ ] **Explosives**

  List of explosions to trigger.
- float **Force**

  The force behind the explosions.
- float **Radius**

  The radius of the explosions.

6.36.1 Detailed Description

Triggers a fracture + explosion when this game object is collided with.

This script does not need to be applied on a fracturing game object.

6.37 DinoFracture.UVBounds Struct Reference

Defines a rectangular bounds across a UV map.

Public Member Functions

- **UVBounds** (Vector2 startUV, Vector2 endUV)
  Constructor taking the bounds.

Public Attributes

- Vector2 **Start**
  Start UV coordinates. Expected to be in the range [0..1].
- Vector2 **Size**
  UV coordinate size (End - Start)

6.37.1 Detailed Description

Defines a rectangular bounds across a UV map.

If the bounds extends backwards, the texture will be reversed when applied.

6.38 DinoFracture.FractureGeometry.UVBoundsSerializable Struct Reference

Unity cannot handle the serializable attribute on types defined in dlls. This wraps a DinoFracture.UVBounds type.

Public Member Functions

- **UVBoundsSerializable** (in Vector2 startUV, in Vector2 endUV)
  Constructor taking the bounds.

Public Attributes

- Vector2 StartUV
  Start UV coordinates. Expected to be in the range [0..1].
- Vector2 EndUV
  End UV coordinates. Expected to be in the range [0..1].

6.38.1 Detailed Description

Unity cannot handle the serializable attribute on types defined in dlls. This wraps a DinoFracture.UVBounds type.

6.38.2 Member Data Documentation

6.38.2.1 EndUV

```
Vector2 DinoFracture.FractureGeometry.UVBoundsSerializable.EndUV
```

End UV coordinates. Expected to be in the range [0..1].

Coord (0, 0) is the bottom left of the texture. Coord (1, 1) is the top right of the texture.

6.38.2.2 StartUV

```
Vector2 DinoFracture.FractureGeometry.UVBoundsSerializable.StartUV
```

Start UV coordinates. Expected to be in the range [0..1].

Coord (0, 0) is the bottom left of the texture. Coord (1, 1) is the top right of the texture.

Index