DINOFRACTURE

A DYNAMIC FRACTURE LIBRARY

USAGE

Scripts are located in the DinoFracture\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 **RuntimeFractureGeometry** 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 DinoFractureDemo\Scenes for an example of some different usage patterns.

SCRIPT REFERENCE

DestroyOnAudioFinish

This component is automatically added to temporary sound game objects created by the **PlaySoundOnFracture** component. It is not intended to be added by the user.

FractureEngine

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

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.

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.

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.

PiecesParent: The parent of the generated pieces. Each fracture produces a root object with fracture pieces (clones of FractureTemplate) as children. The root object is parented to PiecesParent.

NumFracturePieces: 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.

NumIterations: 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. ***It is recommended you use an iteration count of 1 when 0 < FractureRadius < 1.***

NumGenerations: 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. ***Note: this value is ignored if the FractureTemplate does not have a FractureGeometry component.***

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.

UVScale: 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.

DistributeMass: 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.

FractureOnCollision

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

ForceThreshold: The minimum amount of force required to fracture this object. Set to 0 to have any amount of force cause the fracture.

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.

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.

NotifyOnFracture

When this object is fractured, the specified game objects will also get the message.

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

PlaySoundOnFracture

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

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.

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.***

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

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.

Asynchronous: 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.***

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.

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.

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

CONTACT

Questions? Suggestions? Don't hesitate to contact us!

support@fatcatgames.net