**RayFire Shatter**

It's main purpose is **Prefragmentation** in Editor mode. For now it provides 7 fragmentation types: **Voronoi**, **Splinters, Slabs, Radial, Custom, Slices, Bricks, Voxels** and **Tetrahedron** based fragmentation.

**Rayfire Shatter** component provides two fragmentation engines:

* **V1**: FIrst fragmention engine based on 3ds max fragmentation algorithms. Developed in 2019 and is not optimized for Unity.
* **V2**: Second fragmention engine. Developed in 2025 for Unity engine. Much faster and more flexible than V1.

***Management***

After fragmentation you will see a line with few buttons and a name of fragments root under Fragment button. Here you can manage created fragments.

**Delete**: Delete root and fragments.

**Selection bar**: Single click to highlight in Hierarchy, Double Click to select root with fragments.

**Load**: Allows to restore properties which were used to create these fragments.

**Export**: Allows to export fragment meshes into asset file.

**IMPORTANT**: It is absolutely necessary to export fragment meshes into asset before create prefab. Otherwise Prefab will not store fragment meshes because until Export only Scene stores fragment meshes and Prefab can not reference to scene to get meshes.

**Minus**: Forget fragments and remove them from the list.

***Fragments***

**Fragment Type**: Defines fragmentation type for object.

* **Voronoi**: Low poly, convex, physics friendly fragments.
* **Splinters**: Low poly, convex, physics friendly fragments, stretched along one axis.
* **Slabs**: Low poly, convex, physics friendly fragments, stretched along two axes.
* **Radial**: Low poly, convex, physics friendly fragments, creates radial fragments pattern.
* **Custom**: Low poly, convex, physics friendly fragments, allows use of custom point cloud for fragments distribution.
* **Slice**: Slice object by planes.
* **Bricks**: Low poly, physics friendly fragments
* **Voxels**. Fragments object into cubes.
* **Tets**: Tetrahedron based fragments, this type is mostly useless as is and should be used with Gluing, in this case it creates high poly concave fragments. Not supported by V2 engine.
* **Decompose**: Detach every element (separated surface which is not connected with the rest of surface) into separate fragments.

***Voronoi***

[RayFire for Unity. Shatter. 1 Voronoi.](https://www.youtube.com/watch?v=qJ6NVrB1SNk)

**Amount**: Defines the amount of points in the point cloud, every point represents the rough center of the fragment.

**Center Bias**: Defines offset of points in point cloud towards **Center**. Using high values it is possible to create more tiny fragments closer to **Center** and less bigger fragments far from it.

***Splinters***

[RayFire for Unity. Shatter. 2 Splinters](https://www.youtube.com/watch?v=GyqYus7mH78)

**Axis**: Fragments will be stretched over the defined axis.

**Amount**: Defines the amount of points in the point cloud, every point represents the rough center of the fragment.

**Strength**: Defines sharpness of stretched fragments.

**Center Bias**: Defines offset of points in point cloud towards **Center**. Using high values it is possible to create more tiny fragments closer to **Center** and less bigger fragments far from it.

***Slabs***

[RayFire for Unity. Shatter. 3 Slabs](https://www.youtube.com/watch?v=GlZl93WvzHs)

**Axis**: Fragments will be stretched over the defined axis.

**Amount**: Defines the amount of points in the point cloud, every point represents the rough center of the fragment.

**Strength**: Defines sharpness of stretched slabs.

**Center Bias**: Defines offset of points in point cloud towards **Center**. Using high values it is possible to create more tiny fragments closer to **Center** and less bigger fragments far from it.

***Radial***

[RayFire for Unity. Shatter. 4 Radial.](https://www.youtube.com/watch?v=6uE5YuXBnaI)

**Radius**: Radius of radial fragmentation type.

**Divergence**: Radial type creates point cloud which creates Voronoi fragments. Divergence property defines a random offset for points in this point cloud. High divergence provides more chaotic fragments.

**Restrict**: Divergence offset points in all directions. If Restrict property is On points will be restricted to plane.

**Rings**: Defines the amount of Rings.

**Focus**: Defines bias to center for rings.

**Focus Str**: Defines Focus bias strength.

**Random Rings**: Add random offset to rings position.

**Rays**: Defines amount of Rays.

**Random Rays**: Add random offset to rings direction.

**Twist**: Add rotation to rings relative to center.

***Custom***

[RayFire for Unity. Shatter. 5 Custom.](https://www.youtube.com/watch?v=4TuIgnkczag)

***Point Cloud***

**Source**: Defines source of custom point cloud for fragments distribution

* **Children Transform**: Every child object position represents one point in point cloud.
* **Transform Array**: Array of object's transforms.
* **Vector3 Array**: Array of Point 3 coordinates.

**Use As**: Defines source of custom point cloud for fragments distribution

* **Volume Points**: Every point offspring additional points around so total amount of points in cloud will be equal to Amount property in Volume group.
* **Point Cloud**: Every point cloud is used as the approximate center for a fragment.

***Volume***

**Amount**: Total amount of points in the point cloud if Use As set to Volume Points .

**Radius**: Every Volume point generates new points around it in defined Radius.

***Preview***

**Enable**: Show point cloud in viewport.

**Size**: Size of preview point.

***Arrays***

**Transforms**: Array of object transforms in case Source set to Transform Array.

**Vector 3**: Array of global Vector 3 coordinates in case Source set to Vector3 Array.

***Slice***

[RayFire for Unity. Shatter. 6 Slice.](https://youtu.be/JUcQtUGCuRU)

**Plane**: Defines slicing plane by two axes.

**Slice List**: List with transforms for slice planes.

***Tets***

[RayFire for Unity. Shatter. 7 Tets.](https://youtu.be/QKkBr5l0p88)

**Density**: Defines density of tetrahedron cage.

**Noise**: Add noise for tetrahedrons

**Mult**: Multiplier for **Density**.

***Decompose***

[RayFire for Unity. Shatter. 11 Decompose](https://www.youtube.com/watch?v=fMPGXiLJDo0)

***Inner Surface***

**Mapping Scale**: Defines mapping scale for inner surface. Using this property you can increase or decrease the size of texture for the inner surface.

**Material**: Allows to define material for fragment's inner surface. If not applied, RayFire will use original object material for the inner surface.

**Vertex Color**: Allows to define Vertex Color for inner surface vertices.

**UV Region**: Allows to define UV region for all inner surface vertices. Useful when you have material with texture which contains special texture for inner surface of fragments and you need inner surface triangles to have UV mapping inside specific UV region.

***Properties V1 engine***

**Mode**: Defines fragmentation mode.

* **Runtime**: Can be used to test if an object can be fragmented in Runtime by Rigid component. Optimized for Runtime fragmentation. Sensitive to input mesh topology. Object won't be fragmented if the mesh has open edges (not closed volume), not welded vertices or double faces. Meshes with such issues if fragmented will produce fragments with the same issues and attempt to generate a convex hull for collider or simulate such an object may show error and in some rare cases even crash Unity engine. To prevent this all meshes with such meshes will show Bad Mesh warning and will not be fragmented.
* **Editor**: Should be used in Editor for prefragmentation. Not sensitive to input mesh and can fragment meshes with open edges and other issues but will provide fragments with the same issues. In this case such fragments can be tested in Editor and refragmented if there are any simulation issues detected. Also provides advanced properties which allows to automatically fix some issues like capping open edges. Takes more time then Runtime mode for fragmentation because of additional mesh checks and fixes.

**Seed**: Seed for all random parameters.

**Copy**: Copy components from fragmented object to fragments.

**Smooth**: Smooth inner surface. Allows to avoid gaps using dispalce mapping for inner surface.

**Combine**: Combines all children meshes into one mesh and fragment this mesh.

**Collinear**: Removes collinear vertices for fragments. Allows to reduce amount of triangles but may produce UV mapping artifacts.

**Decompose**: Decompose fragments to several meshes by connectivity of the original mesh.

**Input Precap**: Attempt to close all open edges of input mesh to create closed volume. Otherwise only the surface will be cut and fragments won't have closed volume as well.

**Output Precap**: Attempt to close all open edges of output fragment mesh.

Limitations

**Size Limitation**: Allows to fragment again all new fragments with Size bigger than defined Max Size value.

**Vertex Limitation**: Allows to fragment again all new fragments with an amount of vertices higher than defined Vertex Amount value.

**Triangle Limitation**: Allows to fragment again all new fragments with an amount of triangles higher than defined Max Amount value.

Filters

**Inner**: Do not create fragments which are completely inside of fragmented object and can not be seen until demolition and create only fragments that have part of the original object surface.

**Planar**: Do not create planar fragments.

**Relative Size**: Do not create fragments with size smaller than defined value. Mesures in Percentage realtive to original object size.

**Absolute Size**: Do not create fragments with size smaller than defined value. Mesures in units.

Editor

**Element Size**: Measures in percentage relative to original object size. If size of the mesh element (group of triangles connected with each other but not conencted with the rest of the mesh, for instance, rivet on metalic surface) is less than defined value, then such element skips fragmentation process and transferred as is to output fragments array. This allows to avoid fragmenting all small mesh elements on main mesh.

**Double Faces**: Attempt to delete all double faces except one for input mesh.

***Properties V2 engine***

**Hierarchy**: Defines hierarchy for fragments.

* **Instantiate**: Root object and all children with mesh will be instantiated, mesh and renderer components will be destroyed and objects will be used as root for fragments.
* **Copy**: Root object and all children hierarchy will be created from scratch and will be used as roots for fragments.
* **Flat**: All fragments will be parented to one main root.

**Slice Type**: Defines algorithm for internal per fragment cap operations. Every fragment have to be sliced several times to get its final shape. After every slice it has to be capped to close its volume in order to be sliced right next time.

* **Hybrid**: Meshes with open edges will be cut over their surface and meshes with closed edges will be sliced to fragments with inner surface. Should be used for tree like objects or objects that combine meshes wit closed volume and meshes ith open volume (open edges).
* **Forced Cut**: All meshes will be cut over their surface. There won't be inner fragments, only outer surface will be cut.
* **Forced Cap**: All meshes will be sliced to fragments with inner surface where possible. Cap will be applied even for meshes with open volume. This may produce visible artifacts.

**Seed**: Seed for all random parameters.

**Separate Elements**: Element is a part of the mesh that is not connected to the rest of the mesh. In order to speeed up fragmentation and minimize possible issues it is reasonable to fragment every element separately. When this property enabled mesh will be separated to elements and every element will be fragmented separately.

**Element Size**: This property used only when **Separate Elements** enabled and measured in percentage relative to original mesh. If element's size will be less than original mesh size, than such element will not be fragmented at all and will be returned as separate fragment.

*For instance*: If there is a box with a lot of small nail heads all over it and they are **not conencted** to box, then there is no need to put them throught fragmentation process and better to return as is. If Box size is 2 units and one the nail head size is approximately 0.02 units (1% relative to Box size) then you can set **Element Size** property to 2% (0.04 units relative to Box size) so every element that is less than 0.04 units will be detached and returned as is as fragments without going through fragmentation process

**Combine**: This property used only when **Separate Elements** enabled. It allows to combine all separated elements fragments back to single fragment.

*For instance*: Box from previous example may contain hundreds of nail heads and even if you want to fragment Box to 10 fragments with enabled **Separate Elements** property you will endup with hudreds of fragments: fragmented to 10 fragments Box and hundreds of nail heads. In order to get defined amount of fragments you can enable **Combine** property so every nail head will be combined with Box fragment it belongs.

**Input Cap**: Create extra triangles to connect open edges and close input mesh volume for correct fragmentation. Performs once before fragmentation started. You can set **Decompose** fragmentation type and Fragment object with enabled **Input Cap** to see the same mesh with capped open edges.

**Output Cap**: Create extra triangles to connect open edges and close output fragment mesh volume. Performs after fragmentation for every fragment.

**Original Scale**: Allows to keep original objects scale for fragments roots. By default scale for all fragments roots is set to [1,1,1] but in some cases you may need to keep original object scale. It is recommeneded to keep this property disabled in order to avoid fragments scale change in case they will be moved out of their root parent.

**Smooth**: Smooth inner surface. Allows to avoid gaps using dispalce mapping for inner surface.

Filters

**Inner**: Do not create fragments which are completely inside of fragmented object and can not be seen until demolition and create only fragments that have part of the original object surface.

**Planar**: Do not create planar fragments.

***Center***

**Set**: Enabled Center helper to define center for fragmentation bias and for center and direction of radial fragmentation.

**Transform**: Object that will be used as Center helper.

***Shell. V2 only***

**Add**: Cut mesh surface, then add shell to fragments.

**Bridge**: Create bridge between outer and inner surface

**Merge Submeshes**: Merge Bridhe mesh and Inner mesh to one Sub Mesh.

**Thickness**: Shell thickness.

***Partial Fragmentation. V2 only***

**Enable**: Allows to slice object around defined Bounding Box so only mesh part inside Bounding Box will be fragmented to pieces.

**Separate**: Allows to separate all sliced parts to separate fragments. When disabled all slices parts around BOundin Box will be combined into one mesh.

**Source**: Object that will be used as source of Partial Fragmentation bound box.

***Clusters***

[RayFire for Unity. Shatter. 8 Clusters.](https://youtu.be/bfbwp4GF4dM)

Main

**Enable**: Allows to glue groups of fragments into a single mesh by deleting shared faces.

**Count**: Defines total amount of glued groups.

**Seed**: Seed for random parameters.

**Relax**: Smooth inner surface.

Debris

**Amount**: Defines the amount of solo fragments which will not be part of a cluster over the group edges. Measured in percentage, 100% means that all fragments at the edges of the group will not be glued and will stay as is. Using this property you can create several complex glued clusters and small solo fragments (debris) at their surface to create more natural looking demolitions.

**Layers**: Allows to add more layers of debris.

**Scale**: Add random scale variation for debris from 0 to 1. Useful to create small debris crumbling effects.

**Min/Max**: Allows to glue debris together to create just like main clusters, but with an amount of fragments defined by Min/Max range.