MODEL SHARK STUDIO

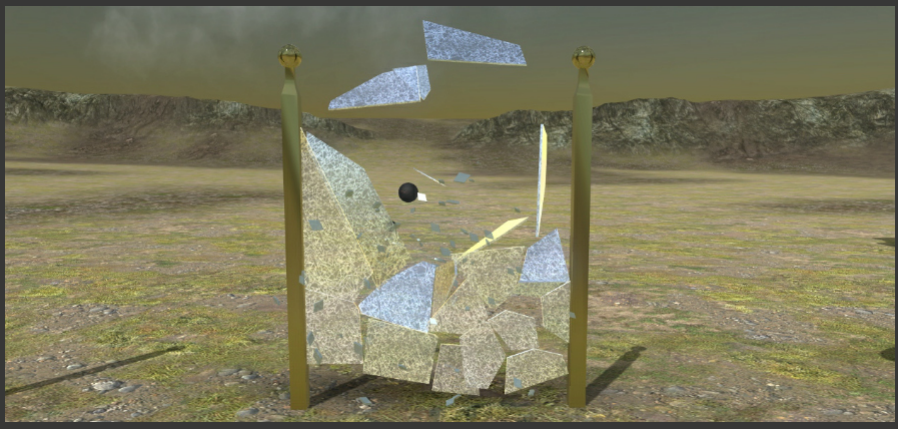
DestroyIt

DESTRUCTION SYSTEM FOR UNITY

User Guide

Contents

What is DestroyIt?



What it Does

DestroyIt is a framework that helps you add Destructible objects to your games. The system doesn't rely on a single strategy for destroying objects. Rather, it uses a mix of strategies to balance visuals, realism, and performance. DestroyIt was designed to meet the following requirements:

Highly Scalable. Have you ever dreamed of blowing up entire buildings full of furniture, fixtures, and support structures in your game? Or even leveling a city block? DestroyIt comes complete with a Destruction Manager, a Particle Manager, and Object Pooling for high performance.

Realistic Destruction. DestroyIt was designed to use destroyed prefabs that break apart realistically and produce convincing, persistent debris. With this system, flying debris can damage players or enemies, and debris can even be used for cover.

Emergent Gameplay. DestroyIt enables games that allow players to breach walls with explosives, or ram a car into a tree, causing it to fall on a power generator, disabling the electric security fence around the base. When objects can be destroyed into realistic debris, it adds a facet of emergent gameplay and player choices that goes beyond visual effects.

What You Get

The DestroyIt Core asset comes with:
Full C# source code, no DLLs
Many Particle Effects
Progressive Damage textures
Interactive Feature Scenarios in Main Demo
SUV Showcase scene
All Demo Scene Assets

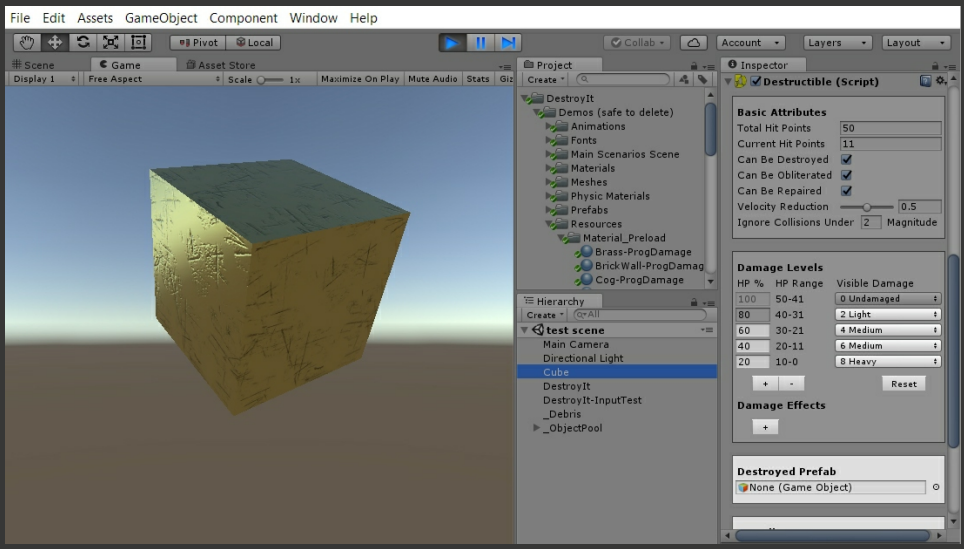
It's not necessary to know programming in order to use the DestroyIt system. However, if you do, you'll be able to further extend and enhance the framework for your game.

Quick Start

New Scene to Destructible Object in 60 Seconds!

These quick start instructions will get you up and running with a bare minimum scene for the DestroyIt system.

1. **Create a new project.**
2. **Import the DestroyIt asset package.**
3. **Add a new scene.**
4. **Add a 3D cube object.** Focus your main camera on the cube so you can see it in the game window. (Click on your main camera and CNTL+SHIFT+F.)
5. **From the main menu, choose Window\DestroyIt\Setup - Minimal.** This will add the required DestroyIt scripts to your scene.
6. **Attach the Destructible script to the cube.** This is what makes it a Destructible object.
7. **Add a Progressive Damage Material** from the DestroyIt\Demos...\Materials folder (such as Brass-ProgDamage) to your object. Any of the materials that end in "ProgDamage" will do.
8. **Run the Scene.** Click Play. You should see your cube. Press the "0" key. You should see the cube take progressive damage with each press of the "0" key until it explodes into chunks.

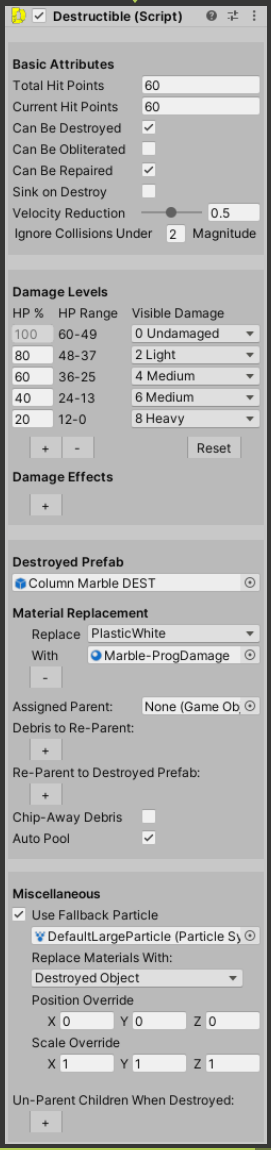


Destructible Objects

At the core of the DestroyIt system is the Destructible script. All other components exist to support this script. The scenarios in the demo scene illustrate the various features of Destructible objects, so playing with the demo scene is the best way to get familiar with them. This is simply a quick reference.

Total/Current Hit Points are the maximum and current health of the object, or rather, how close it is to being destroyed.
Can be Destroyed if unchecked, this object cannot be destroyed, but will still show progressive damage and kick off damage effects.
Can be Obliterated an object is obliterated and destroyed as a particle effect if it takes an excessive amount of damage, usually from area affect weapons like explosives. Turn this off sparingly, as it will impact performance.
Can be Repaired if checked, this object can be repaired.
Sink on Destroy will make the object fall through the terrain when destroyed, useful for some games like real-time strategies.
Velocity Reduction is how much this object slows down rigidbody impacts when it's destroyed. A brick wall would have high velocity reduction. A thin pane of glass would be low.
Ignore Collisions Under [X] Magnitude means any collision under this magnitude will be ignored. Use low numbers for objects that do not absorb impacts (ie, a ceramic vase). Use high numbers for objects that absorb impacts (ie, a rubber tire).
Damage Levels allow you to specify how many damage levels your object has, and the hit point range and visible damage of each. You must have at least 1 damage level, and can have as many as 10.
Damage Effects allow you to specify particle effects that are played at certain damage levels. For instance, you could have an

Destructible



Destructible Objects (continued)

engine that begins smoking when it is 50% damaged, and bursts into flame when it is 90% damaged.

Destroyed Prefab is the prefab that replaces this object upon destruction.

Material Replacement is a list of materials on the Destroyed Prefab that will get replaced at the time of destruction. This allows you to use a single destroyed prefab for many different materials. For instance, one wall made of stone and another made of wood could use the same prefab.

Assigned Parent: when destroyed, the destroyed prefab will spawn in as a child of this game object. If left blank, the destroyed prefab will have no parent.

Debris to Re-Parent is an optional list of debris in the Destroyed Prefab to re-parent under the object's current parent after it is destroyed. For example, the hilt of a sword would be re-parented to a character's hand bone. The wheel of a tire would be re-parented back to the axle.

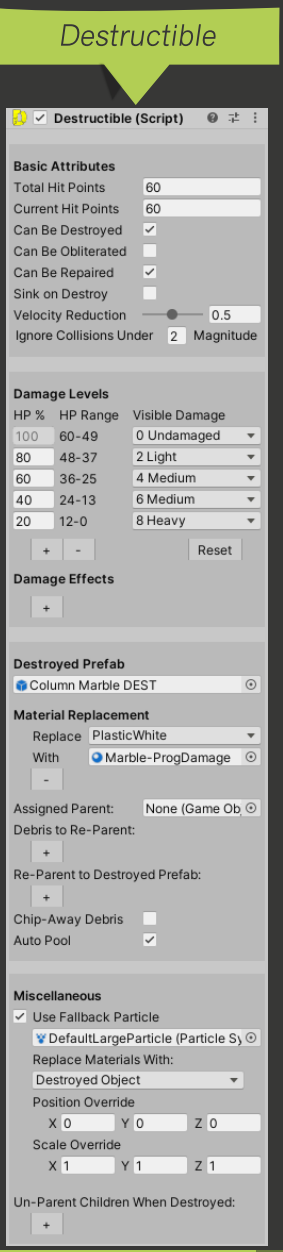
Re-Parent to Destroyed Prefab allows you to choose one or more children under the destructible object that will be re-parented under the destroyed prefab. This is useful for things like car doors that have a window child, which needs to be re-parented under the destroyed prefab's door. (See SUV Showcase scene)

Chip-Away Debris makes debris cling to a supported core structure in your destroyed prefab. This is useful for columns or walls that have solid rebar centers. The debris can be chipped away from the rebar once the object is destroyed.

Auto Pool if checked, the destroyed prefab for this object will be added to the object pool automatically for you at runtime.

Use Fallback Particle if checked, the specified particle effect will be played for this object whenever: the Destruction Manager's debris limit has been reached, the object does not have a Destroyed Prefab, or the object is obliterated (takes excessive damage). You can also specify what materials you want to replace on the particle effect's mesh renderers, as well as overriding the position the particle effect is played at, and the scale of the particle effect when it is spawned.

Un-Parent Children When Destroyed identifies children of this object you want to release (un-parent) when this object is destroyed. An example might be a ceiling fan (child) that falls when the ceiling (parent) is destroyed.

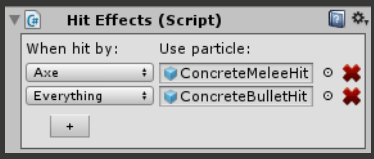


Hit Effects

DestroyIt includes a hit effects system that allows you to specify what particle effect plays when a weapon hits an object. It works with more than just Destructible objects, too - you can use the HitEffects script on any object with a collider.

The way it works is, each weapon or projectile checks for the HitEffects script on the object it hit. If it finds the script, it searches the "When hit by" field for a match from top to bottom. If it finds a match, it plays the effect for the first one it finds. This allows you to specify different hit effects for different weapons, and also have a default effect play for everything else.

HitEffects



How do I use it?

Simply put the HitEffect script on your game object and specify the particle effects to use when it is hit by the listed weapons or projectiles. You can list out specific hit effects to play for each weapon, and leave a default for everything else at the bottom of the list.

If you have a complex object like the glass pane with metal posts (left), you can put a single HitEffects script on the parent game object and specify the Metal particle effect, then put another HitEffects script on the glass pane child and specify the Glass effect. When the metal posts are hit, the script will search up the parent hierarchy for a HitEffects script and find it at the parent level and play the Metal hit effect. When the glass pane is hit, it will find the HitEffects script on the child and stop searching.

Example

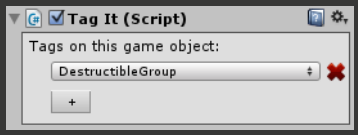
Check out Scenario #2 - Hit Effects in the demo scene, or virtually any of the scenarios after #2 as they all use hit effects!

Destructible Groups

When you have a complex structure made up of many Destructible objects (like the tower shown above), you may want to treat the individual Destructible objects as a group when it comes to particle/debris culling to get the best performance.

An exploding tower like the one above would quickly fill the Active Particle queue in the Particle Manager, essentially "hogging up" all the particles for itself. This could cause other nearby objects destroyed during the same time to simply disappear. But by designating the tower as a destructible group, you are telling the system to treat the entire structure as one when it comes to particle/debris culling.

DestructibleGroup



How do I use it?

To designate a destructible group, select a parent gameobject that has multiple Destructible object children you would like to group. Drag the TagIt script onto the parent object and select the DestructibleGroup tag. That's it! The system will handle the rest.

You may want to check your Particle Manager's MaxPerDestructible setting at this time, to make sure it fits your needs. If it's too low, your entire structure might look silly as it explodes into a few measly particle effects. Too high, and your game's performance may tank.

Example

Check out Scenario #20 - Support Points in the demo scene. The entire tower is a DestructibleGroup.

Progressive Damage (Standard Shader)

What is it?

Progressive Damage is a feature of the DestroyIt system that gives your objects visual damage before (and after) they are destroyed. As a Destructible object takes damage, it progresses through each damage level and changes in appearance.

How do I use it?

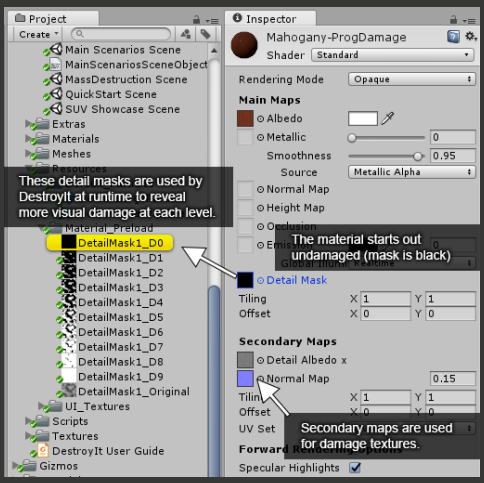
For the built-in Standard Shader, assign Detail Masks and Secondary Damage Textures to your material, as illustrated (right).

Detail Masks

Detail masks are a series of black and white images that reveal more of the damage texture as the image gets whiter. The first mask in the series should be solid black (no damage texture will show through), and the last mask solid white (completely covered in damage texture).

Damage Textures (Secondary Maps)

To create a new damage texture for the Standard Shader, be sure to set the background fill layer to 50% gray (128 128 128). The damage details (scratches, gouges, cracks) can be any color.



Progressive Damage (URP)

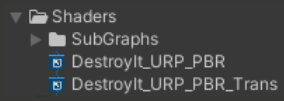
What is it?

Progressive Damage is a feature of the DestroyIt system that gives your objects visual damage before (and after) they are destroyed. As a Destructible object takes damage, it progresses through each damage level and changes in appearance.

How do I use it?

For the Universal Render Pipeline (URP), we created a custom Shader Graph shader for progressive damage.

Change your material's shader to one of our custom ones (below) and assign Detail Mask and Detail Map textures to your material, as illustrated (right).

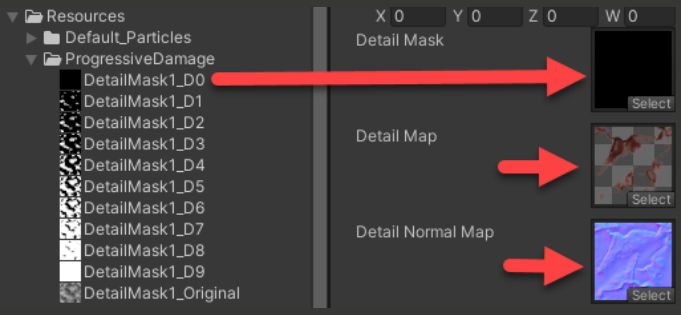


Detail Masks

Detail masks are a series of black and white images that reveal more of the damage texture as the image gets whiter. The first mask in the series should be solid black (no damage texture), and the last mask solid white (completely covered in damage texture).

Damage Textures (Detail Maps)

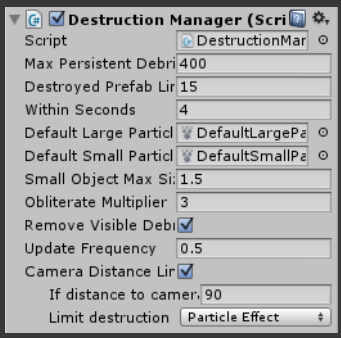
To create a new damage texture for URP, make sure your image has an alpha channel for the parts of the texture you don't want to be seen. The damage details (scratches, gouges, cracks) can be any color.



Destruction Manager

The Destruction Manager controls how destructible objects behave, primarily under heavy load. It limits how many concurrent pieces of debris are allowed to persist in the world, and when they can be recycled. It also throttles destructible prefab instantiation when there's heavy destruction in a short amount of time.

DestructionManager



Properties

Max Persistent Debris: The number of debris pieces allowed to persist in the scene. Lower for performance.
Destroyed Prefab Limit: The maximum destroyed prefab instantiations allowed within Within Seconds (below). Additional destructions will use the Destructible objects' Fallback Particle. Lower for better performance.
Within Seconds: The length of time (in seconds) before destroyed prefabs are no longer counted against the Destroyed Prefab Limit. Increase for better performance.
Default Large Particle: The particle effect used for Destructible objects that have no Fallback Particle set. Whether an object is considered large or small is determined by its mesh extent bounds and the Small Object Max Size value. Note: These default particles are just placeholders in case you forget to assign a Fallback Particle on your destructible objects. They're your last line of backups.
Default Small Particle: Same as Default Large Particle, except this is the one used for small objects.
Small Object Max Size: The maximum mesh size (in game units) of an object to be considered "small". Any object larger than this will use the Default Large Particle.
Obliterate Multiplier: This number times the destructible object's total hit points is the amount of damage that must be sustained in a single hit for the object to be obliterated (see Destructible script). If the number is 3, a destructible object with 100 hit points that sustained 300 hit points in one hit will be obliterated.
Remove Visible Debris: If checked, debris can be destroyed/recycled by the Destruction Manager even if it is currently being rendered by the main camera. Leave this option on for better performance.
Update Frequency: How often the Destruction Manager updates its list of monitored debris and the list of recently-instantiated destroyed prefabs. Increase for better performance.
Camera Distance Limit: When checked, if a destructible object is farther than the specified game units from the camera when it is destroyed, the fallback particle will be used. Useful for mass destruction when objects in the distance would not be noticed as much.

Particle Manager

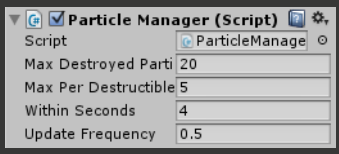
The Particle Manager is an optional script you can add to your scene that primarily acts as a throttle for Destructible object particle effects. When the Destruction Manager determines it should destroy an object with a particle effect, it checks to see if the Particle Manager exists. If it does, it calls the Particle Manager to play the effect.

The Particle Manager checks to see if the current limit of particles for Destructible objects has been reached. Too many Destructible objects blowing up at once could get out of hand and grind your framerates to a crawl. The Particle Manager will ignore any requests to play particle effects if the limit has been reached. While this may not be ideal visually, it's a lot better than crashing your game or lagging framerates.

One way to limit particles but keep destruction looking good is the MaxPerDestructible variable. This will limit each Destructible object (or DestructibleGroup) to a maximum number of particles, which keeps a complex Destructible object from hogging up all the particles.

The Particle Manager also enhances the default particles by changing their material to match the destroyed object. In other words, if you blow up a purple object and it explodes into the default particle (it doesn't have a custom fallback particle), then that particle effect will also be purple.

ParticleManager



Properties

Max Destroyed Particles: The total number of particles allowed by all Destructible objects within the specified seconds (Within Seconds). Lower for better performance.

Max Per Destructible: The maximum number of particles allowed by a single Destructible object or Destructible Group within the specified seconds (Within Seconds). Lower for better performance.

Within Seconds: The length of time (in seconds) before particles are removed from the watch list. Increase for better performance.

Update Frequency: How often the Particle Manager updates the list of particles it's watching. Increase for better performance.

Object Pool

When you need to blow up a LOT of stuff up (and who doesn't?), it can be a real performance hog to instantiate/destroy new game objects and particle systems for each object. Much better to have the objects already in the game when they're needed, and recycle them if they can be re-used. That's what the Object Pool is for.

The concept is simple - objects are instantiated and then disabled, and put under an empty game object when the game starts. They remain there until needed, at which time they're positioned and activated. Instead of destroying them when they've played out (particles, for example), they're disabled, reset, and added back to the pool, ready to be used again.

To add a game object to the pool, click the Add button and drag a prefab to the new slot. Enter the number of instances of the object you want to keep in the pool in the box provided. The lock icon determines whether to only allow the designated number of objects regardless of the number of requests (**strict, gold**), or allow additional objects to be instantiated on the fly, even if there are none available in the pool (**flexible, gray**).

Suppress Warnings - if you uncheck this option, the Object Pool will let you know when an object was instantiated or destroyed directly instead of using the pool. This can be useful information when setting up and playtesting destruction in your scene.
Import/Clear/Save - use these options to save your object pool entries to a text file (located under Assets/DestroyIt - Core), remove all objects, or load objects from a previously-saved file.

ObjectPool

Clinging Debris

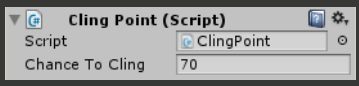
Clinging debris is an optional feature of the DestroyIt system that gives your destroyed objects realism by making some of the debris cling to adjacent colliders.

How it Works

You add the Cling Point script to colliders on your destroyed prefab. When debris is spawned, the DestroyIt system looks for cling points and raycasts against adjacent colliders. If the raycast hits something, it rolls a percent chance based on Chance to Cling that you specify. If it passes that check, it strips the rigidbodies off the debris piece and parents it under the object it clings to. This provides better performance and a rock-solid joint over hinge joints.

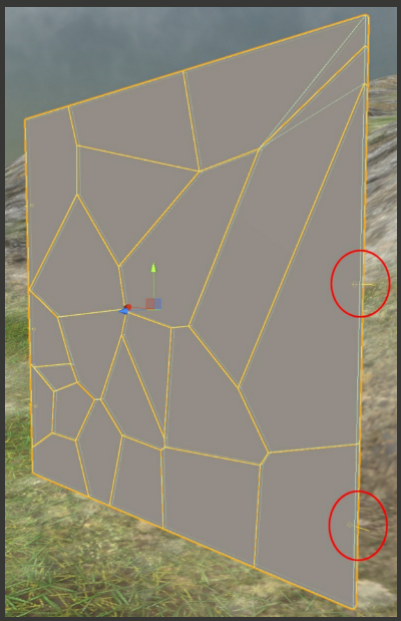
To the right is an illustration showing how cling points appear in the inspector. This is the destroyed prefab for the glass pane. On the edges, you can see the "pin" gizmos that indicate the direction the raycasts will be fired from the debris piece to check for adjacent colliders to cling to.

ClingPoint



Properties

Chance To Cling: How likely (percent chance) the cling point will attach itself to adjacent colliders.



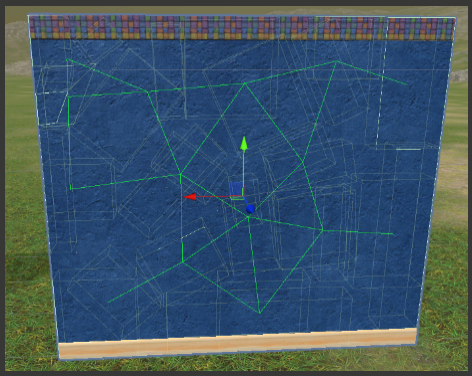
Cling Point locations and direction gizmos.

Structural Support

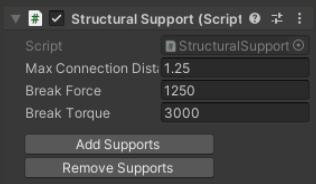
Sometimes, you may want destructible debris to remain joined together and have an amount of flex and structural integrity, while being able to react to physical forces and additional damage. For these cases, we've included the **StructuralSupport** script that helps you quickly and easily add connecting joints to your destroyed prefabs.

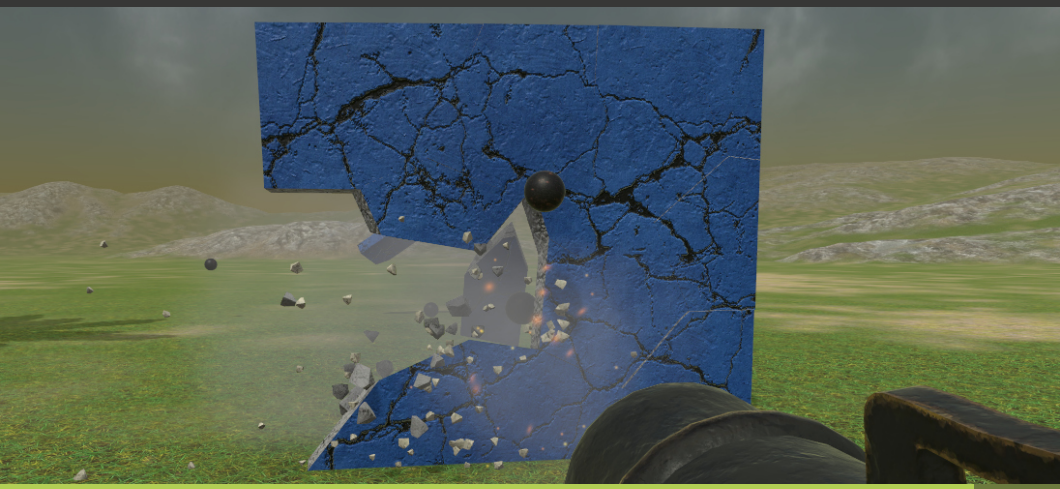
How it Works

Drag your destroyed prefab into the scene, then add the StructuralSupport script to it, at the parent level above the child debris pieces. Then click the Add Supports button, and it will add connecting joints to all the debris pieces, based on the connection distance and force/torque settings you specify. If you have Gizmos turned on in the editor, you will see green lines showing the connected rigidbodies.



StructuralSupport





DestroyIt Events

There may be times when you need to act on certain destructible events. For instance, you may need to play a sound when a destructible object is destroyed, or update the amount of "repair points" remaining in a repair kit when the player repairs an object. To help with this, DestroyIt exposes several events that you can listen to and act upon, as well as sample scripts to get you started.

Destroyed Event

Any time a Destructible object is destroyed, it will fire the Destroyed event so any listeners can act upon it.

Sample Scripts: In the demo folder, see the WhenDestroyed.cs and WhenDestroyedPlaySound.cs scripts. WhenDestroyed logs the object's name and world position to the console log when it is destroyed, and WhenDestroyedPlaySound plays the selected audio clip on destruction.

Damaged Event

Whenever a Destructible object is damaged, it will fire the Damaged event so any listeners can act upon it.

Sample Script: In the demo folder, see the WhenDamaged.cs script. This script logs the object's name and how many hit points of damage was done each time it is damaged.

Repaired Event

Whenever a Destructible object is repaired, it will fire the Repaired event so any listeners can act upon it.

Sample Script: In the demo folder, see the WhenRepaired.cs script. This script logs the object's name and how many hit points of damage was repaired each time it is repaired.

Repairing Destructible Objects

Destructible objects can be damaged in many different ways, but by default they can also be repaired.

Damage Effects and Repairing

When an object is damaged and has damaged effects, then at certain damage levels the effects will play. For instance, you may have a damage effect that emits sparks from the object when it reaches 50% damage.

When you repair an object that has damage effects, it will turn off the effects as it is repaired past the damage level threshold. So for instance the sparks will stop emitting when it is repaired to 40% damage. If the object is then damaged again to 50%, sparks will start emitting again.

Example: In the main scenarios demo scene, see the second cog box scenario that has damage effects. As you damage it, smoke and sparks will emit from the object. Switch to the wrench weapon and use it on the cog box to repair it. As you repair the object, the sparks and smoke will stop emitting at the appropriate damage levels.

Disabling Repairing

If you have a destructible object that shouldn't be able to be repaired, you can simply disable this feature by un-checking Can Be Repaired on the Destructible script.

TagIt

TagIt is a simple script that lets you add multiple tags to a single game object. If you already have a multi-tagging solution, you can use that instead. If you do, you'll need to replace references to TagIt throughout the DestroyIt namespace with your own.

DestroyIt uses the TagIt system to indicate whether objects have power (like lights), and whether materials have been transferred from a destructible object to its destroyed prefab.

To use TagIt to control particle effects for collisions, attach the script at the same level as each collider on an object and choose the appropriate material

Destructible Trees

DestroyIt provides two ways for you to create destructible trees in your game: as standalone objects, and as terrain trees. If you take a look a the Palm tree in the demo scene, you'll notice that it's a regular standalone game object and not part of the terrain.

The benefit of creating a standalone destructible tree like this is, it's simpler to setup and manage, because they work like any other destructible object. The downside is, it prevents you from being able to paint the destructible trees on the terrain, or from taking advantage of the built-in tree swaying and nature tinting of the terrain shaders. It's a good option for games where trees are less of the focus or the levels are small.

If on the other hand, you are making an open world survival game or similar, where levels are vast and destructible trees are numerous, take a look at our DestroyIt Destructible Trees PDF, which explains how to make destructible terrain trees. This way, you get all the advantages of terrain trees, and they're still destructible!

DestroyIt-Ready Assets

DestroyIt provides you with tools to import, create, and even sell DestroyIt-Ready assets that are pre-configured to use with DestroyIt. To use this feature, **unpack the DestroyIt-Ready Unity package** located under \Assets\DestroyIt\Extras (safe to delete).

This feature allows content creators to create destructible models and effects and package them as separate assets to share or sell, without the need to include our proprietary DestroyIt scripts. DestroyIt-Ready assets can then be imported into a project and converted to fully-working Destructible objects with one click.

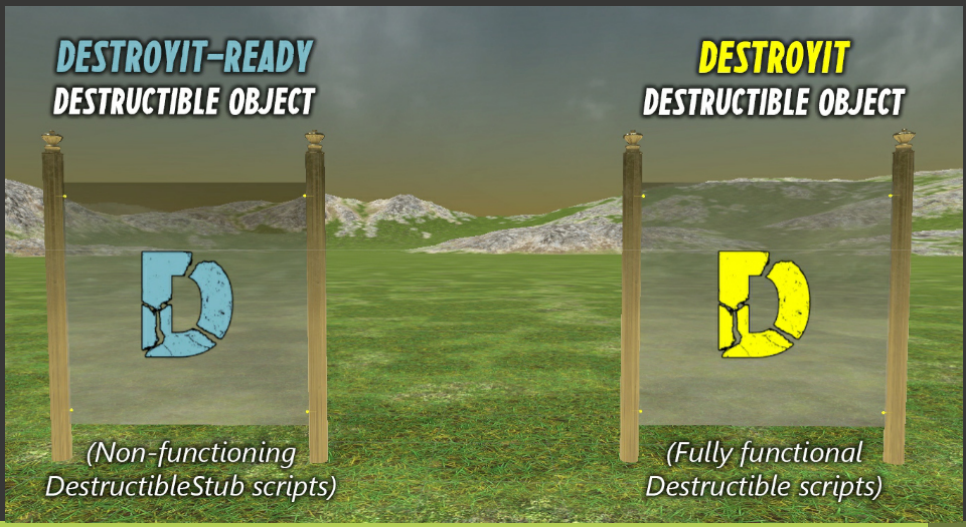
If you have Gizmos turned on in Scene view, you can easily identify a DestroyIt-Ready object by the color of its icon. DestroyIt objects will have a yellow "D" logo, and DestroyIt-Ready objects will have a blue logo (see below).

Using DestroyIt-Ready Assets

Once you've imported a DestroyIt-Ready asset, select it by clicking on it in the Project or Hierarchy tab. From the Window menu choose DestroyIt > Convert Stubs to Destructibles. A popup will ask you to confirm that you want to replace all DestructibleStub scripts on the object with Destructible scripts. Click Replace, and your object is now a fully-functional Destructible object.

Creating DestroyIt-Ready Assets

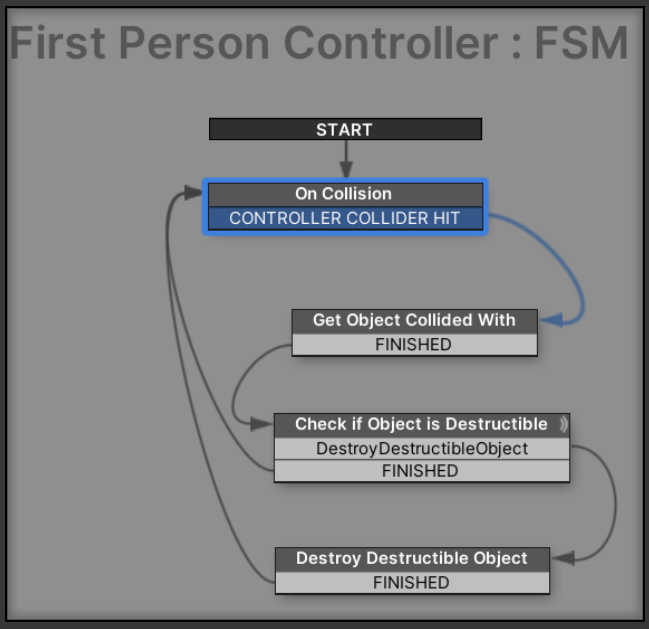
To create a DestroyIt-Ready asset, select your Destructible object, and from the Window menu choose DestroyIt > Convert Destructibles to Stubs. A popup will ask you to confirm that you want to replace all Destructible scripts with DestructibleStub scripts. Click Replace, and now your object is ready to export. Right-click the object and choose Export Package. At this time you should also select any other assets you've created that are required for the object to work correctly. **Be sure to uncheck "Include Dependencies" to prevent Unity from including proprietary DestroyIt scripts and assets with your package.**



PlayMaker Integration

If you use PlayMaker for your game development, you might be interested in our custom DestroyIt PlayMaker Actions and scripts which let you damage, destroy, and repair destructible object with actions, and listen to those same events in your PlayMaker FSM. To get started, unpack the PlayMakerDestroyIt Unity package located under \Assets\DestroyIt\Extras (safe to delete).

Open the PlayMaker Demo scene and observe how the scenarios are setup. Each scenario has a PlayMaker Action attached to something that either damages, destroys, or repairs Destructible objects. There is also an event listener that reports events sent to PlayMaker from DestroyIt scripts.



Performance Optimization Tips

Particles

How complex are your particle effects? Try to minimize the number of particles emitted. Use billboard renderers instead of meshes and turn off environment collisions when possible. Consider grouping destructible objects into DestructibleGroups to reduce particle emission. Also, check your ParticleManager settings to see if you can reduce the max particles or update frequency.

Destroyed Prefabs

How complex are your destroyed prefabs? Can you reduce the number of rigidbodies and still have enough debris? Are you using mesh colliders? Try to use primitive colliders when possible. Also, try reducing the total amount of persistent debris in the DestructionManager.

Physics Settings

What is your physics Timestep value? A lower value gives you better looking physics, but at a high performance cost. Also check your solver iteration count. Can you reduce it and still get "solid enough" joint connections?

Shadows

Can you turn off or reduce shadows in your scene to increase draw call batching? On your debris pieces, can you turn off Send/Receive shadows to increase performance?

Object Pooling

Check that you are allocating enough destroyed prefabs in your object pool. Turn off "Supress Warnings" on the ObjectPool so you get notified when objects are instantiated or destroyed directly.

FAQs

Why aren't my destructible objects showing progressive damage?

- Make sure your material is using one of the Standard Shaders (or the DestroyIt custom URP shaders if you're using the URP rendering pipeline) and you have assigned a detail mask
and secondary damage textures (see progressive damage section).
- Check your destructible object's hit point value to make sure it's not a really big number.
- If your destructible object does not have a rigidbody and is nested under a gameobject that has a rigidbody but is not destructible, use the DestructibleParent script on the parent. This will cause collisions on the parent to fire on the destructible children as well.

How do I stop my rigidbodies from jittering/dancing around?

Sometimes, you may notice rigidbodies dancing around or jittering along the ground. Most of the time, this is due to something setup improperly with the rigidbodies, such as nesting them. Or you may have a Joint that's trying to bind two objects together but the anchor point is misaligned.

However, sometimes you have everything setup correctly and you still get rigidbody jittering or wiggling. In this case, try the following to see if it fixes it:

- Increase your Solver Iteration Count under Edit => Project Settings => Physics to 20 or higher.
- Click Edit => Project Settings => Physics and check the Enable Adaptive Force checkbox.
- Reduce your Fixed Timestep under Edit => Project Settings => Time to 0.01 or lower.

How do I create my own destroyed prefabs?

You'll need to fracture your un-destroyed model with a 3D modeling tool to create the debris pieces. The difficulty of this task depends on several factors: how complex your model is, the type of materials it's made of (concrete is easier to simulate than wood), how many broken pieces you want to create, and whether you can automate it with a plugin.

For more information on how to create your own destroyed 3D models, check out our video on **Cell-Fracturing in Blender** (a free 3D modeling tool):

Does DestroyIt work with URP and HDRP?

DestroyIt is fully compatible with URP - the only thing you'll need to do differently is choose our custom Shader Graph shader for your material if you want progressive damage textures on your destructible objects.

DestroyIt also works well with HDRP, with the exception of progressive damage, which is not currently supported for HDRP.