MODEL SHARK STUDIO

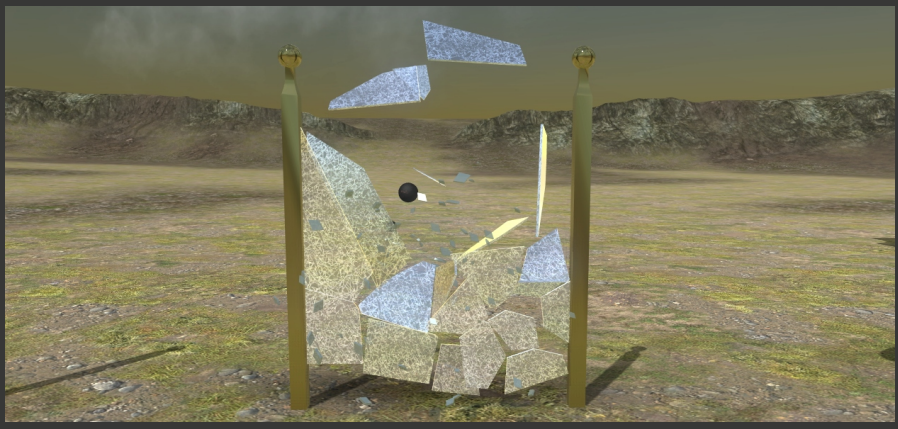
DestroyIt

DESTRUCTION SYSTEM FOR UNITY

User Guide

Contents

What is DestroyIt?



What it Does

DestroyIt is a framework that helps you add object destruction 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, Object Pooling, and a Material Preloader 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 could even damage players and enemies, and can also 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
23 Interactive Feature Scenarios in demo
SUV Showcase scene
All Demo Scene Assets

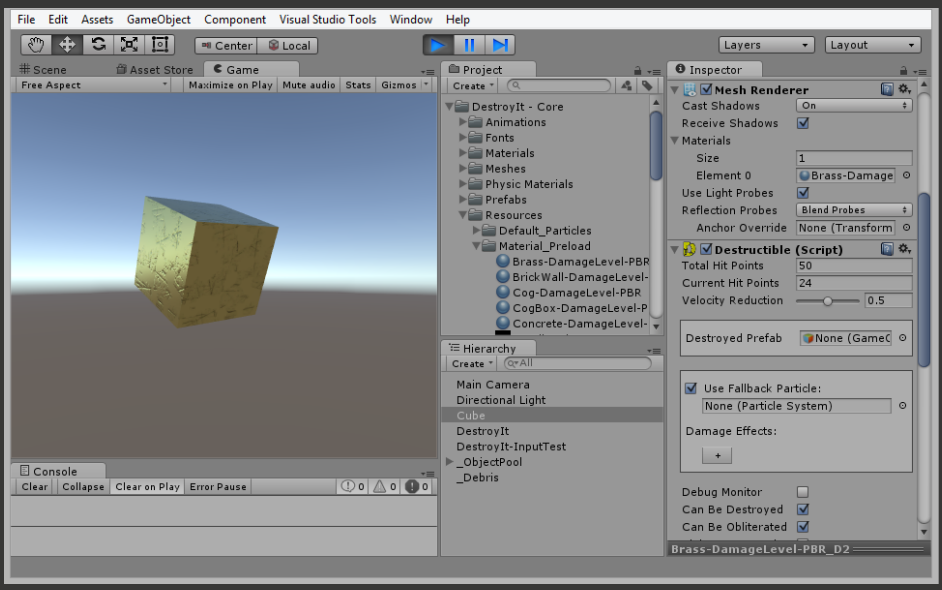
It's not necessary to know programming in order to use the DestroyIt system. However, if you do, you'll be able to further extend and enhance the framework for your game.

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 package.**
3. **Add a new scene.**
4. **Add terrain and a directional light.**
5. **Add a cube object.** Focus your main camera on the cube so you can see it in the game window.
6. **From the main menu, choose Window\DestroyIt\Setup - Minimal.** This will add the required game objects and scripts to your scene.
7. **Attach the Destructible script to the cube.** This is what makes it a Destructible object.
8. **Add a Progressive Damage Material** from the DestroyIt - Core\Resources\Material_Preload folder (such as Brass-DamageLevel-PBR) to your object.
9. **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.

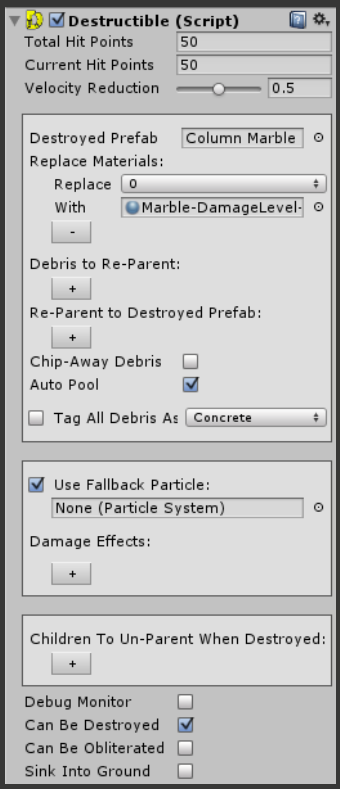


Destructible Objects

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.
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.
Destroyed Prefab is the prefab that replaces this object when it is destroyed.
Replace Materials 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 "destroyed wall" prefab.
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. Examples: the hilt of a sword, or the wheel of a tire.
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 reparented under the destroyed prefab's door. (See SUV Showcase scene)
Chip-Away Debris: choose this option to make 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.

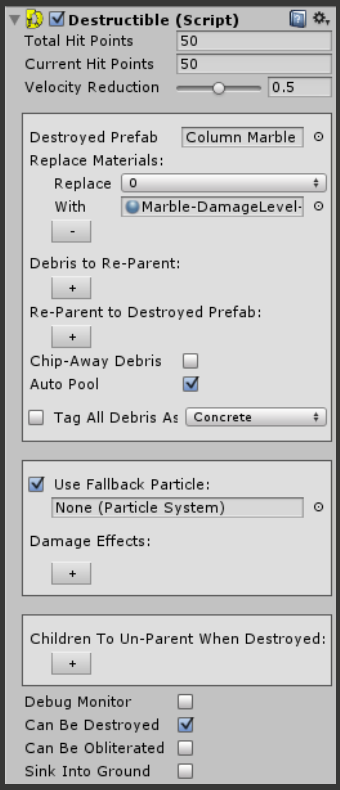
Destructible



Destructible Objects (cont.)

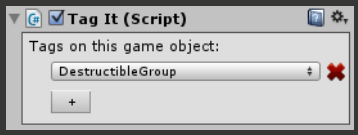
Auto Pool: by default, destroyed prefabs are added to the Object Pool automatically, and materials pre-transferred for speed. You can un-check this option and handle the pooling yourself if your Destructible objects can be altered significantly during gameplay.
Tag All Debris As this option provides an easy way for you to tag all debris pieces on the destroyed prefab automatically instead of having to manually add TagIt scripts to the debris beforehand.
Fallback Particle is the particle effect to play for this object if it is destroyed and the destroyed prefab cannot be instantiated. This can happen if the Destruction Manager's debris limit has been reached, if the object does not have a Destroyed Prefab, or if the object is obliterated (takes an excessive amount of damage). Most of the time, you will want to use a fallback particle.
Damage Effects are optional effects that happen at different damage stages. There are five damage stages (including destroyed). You can kick off effects at each of these stages.
Children to Un-Parent are children of this object 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.
Debug Monitor if checked, displays the object's hit points in the HUD for debug purposes.
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 can impact performance.
Sink Into Ground will make the object fall through the terrain when destroyed, useful for some games like real-time strategies.

Destructible



Destructible Groups

DestructibleGroup



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.

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, please!

Check out Scenario #19 - Support Points in the demo scene. The entire tower is a DestructibleGroup.

Progressive Damage

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" so it shows signs of increased wear and tear.

How do I use it?

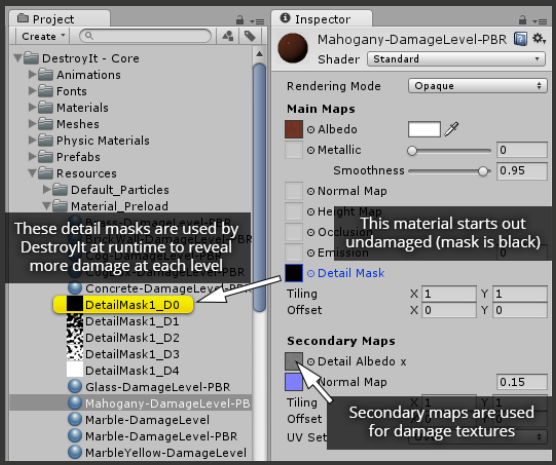
DestroyIt uses Detail Masks and Secondary Damage Textures on the Standard Shader to provide progressive damage. Also, it's important to note that **all progressive damage materials must be in the Resources\Material_Preload folder**.

Detail Masks

Progressive damage detail masks are a series of grayscale images with successive levels of white that reveal more of the damage texture. The first mask should be solid black and the last mask solid white. After you've imported the masks into Unity, set them to Alpha From Grayscale in the texture import settings.

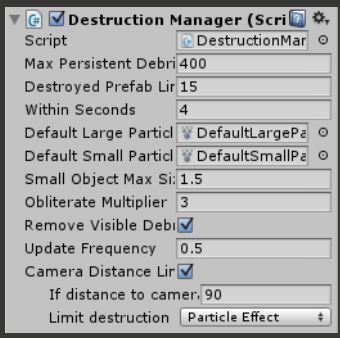
Secondary Damage Textures

When creating a new damage texture, set the background fill layer to 50% gray (128 128 128). The damage itself can be any color, though black or white will give the best results most of the time.



Destruction Manager

DestructionManager



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.

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

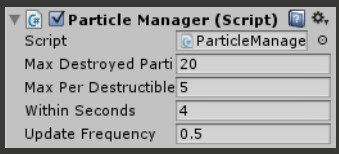
The Particle Manager is an optional script you can add to your scene that primarily acts as a throttle for Destructible object particle effects. When the Destruction Manager determines it should destroy an object with a particle effect, it checks to see if the Particle Manager exists. If it does, it 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.
Load/Save to File - use these options to save your object pool entries to a text file (located under Assets/DestroyIt - Core) or load them from a previously-saved file.

ObjectPool

Clinging Debris

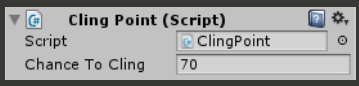
Clinging debris is an optional feature of the DestroyIt system that gives your destroyed objects realism by making some of the debris cling to adjacent colliders.

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.

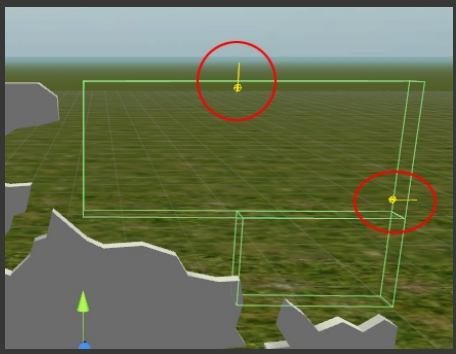
To the right is an illustration showing how cling points look in the inspector. In this case, we have disabled the mesh on the chipboard wall so you can see the "pin" gizmos that indicate the direction raycasts will be fired from the debris piece to check for adjacent debris.

ClingPoint



Properties

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



Cling Point location and direction gizmos.

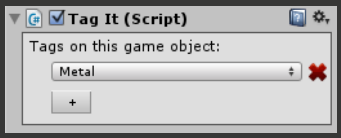
TagIt

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. You'll need to replace references to TagIt throughout the DestroyIt namespace with your own script.

DestroyIt uses the TagIt system to indicate whether objects have power (like lights), whether materials have been transferred from a destructible object to its destroyed prefab, and what type of particle effect to play for bullet and cannonball collisions. For example, when a bullet strikes a collider that has been tagged as Wood, it will play a wood bullet strike particle effect at the hit normal.

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

TagIt



Destructible Trees

The DestroyIt system provides a way for you to create destructible trees in your game, as seen by the Palm tree in the demo scene. You may have noticed that the Palm tree in the demo is a regular game object and not part of the terrain. This prevents you from being able to paint destructible trees on the terrain, or taking advantage of the built-in tree swaying terrain shaders. So why do it this way?

The reason is, you can't attach scripts to individual terrain objects, since they are part of the terrain. And realistically, you probably wouldn't want to. If you had 10,000 trees in your scene (not unrealistic for a large map), then you would have 10,000 Destructible scripts running, which needless to say would grind performance to a halt.

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 debug logging on for 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 and you have assigned a detail mask and secondary damage textures (see progressive damage section).
- Make sure your material is getting pre-loaded by putting it in the DestroyIt - Core\Resources\Material_Preload folder.
- Check your destructible object's hit point value to make sure it's not a really big number. (You can use the debug monitor checkbox to monitor your object's status in the HUD.)
- 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 have a joint that's trying to bind two objects together but the anchor point is setup wrong.

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

- Increase your Solver Iteration Count under Edit => Project Settings => Physics to 20 or higher.
- 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.