Destructible Structure Builder Manual Version: 1.0.0
Compatibility: Unity 6 LTS, 2022.3 LTS, 2021.3 LTS

Table of Contents

1. Introduction

Overview

Destructible Structure Builder (DSB) is a Unity®-Editor toolkit for assembling gameplay-ready buildings that can splinter, crumble, and collapse in real time. You author everything inside the main DSB window using a toolbar of Build Modes for placing connections, members, and walls. At runtime, dedicated components handle stress propagation, pooling, and event dispatch.

Important: *All construction happens in the Unity Editor; the toolkit provides* ***no*** *in-game building or editing capabilities.*

Feature Highlights

- **Stress Solver** – Graph-based system that finds the shortest path to ground, splits off unsupported chunks, and propagates structural load to detect overstressed members.
- **Voxel-Based Damage System** – Structures are built from voxels that split and detach individually on impact.
- **Chunk Merging & Splitting** – Voxels combine into chunks for performance and dynamically split when damaged.
- **Build Modes Toolbar** – Visual, mode-based system for placing nodes, beams, walls, stairs, and more.
- **Wall Designer** – Cubes, windows, triangular cut-outs, per-cell health, and live rotation controls.
- **Design Presets** – Save any wall as a WallDesign ScriptableObject and re-apply instantly.
- **Object Pooling** – Pooled debris with adjustable lifetime, pool size, and spawn-rate.
- **Profile-driven Effects** – Plug-and-play audio/particles via EffectsLibrary and MaterialEffectsLibrary assets.
- **Mesh Cache** – Persist generated meshes between sessions; safely prefab structures.
- **Full Unity Undo Support** – Every spawn, delete, or property change is wrapped in a single Undo group.

Package Contents

Folder	Contents
Scripts/Runtime/	MonoBehaviours that run in play mode (destruction logic, stress solver, pooling).
Scripts/Editor/	Custom UI (Structure Manager, scene tools, inspectors).
Materials/, Textures/, Audio/, ParticleEffects/	Included assets.
Samples/Demo/Scene/	A sample scene demonstrating structures and destruction.

System Architecture Overview

The diagram below illustrates the core components of DSB and how they interact.

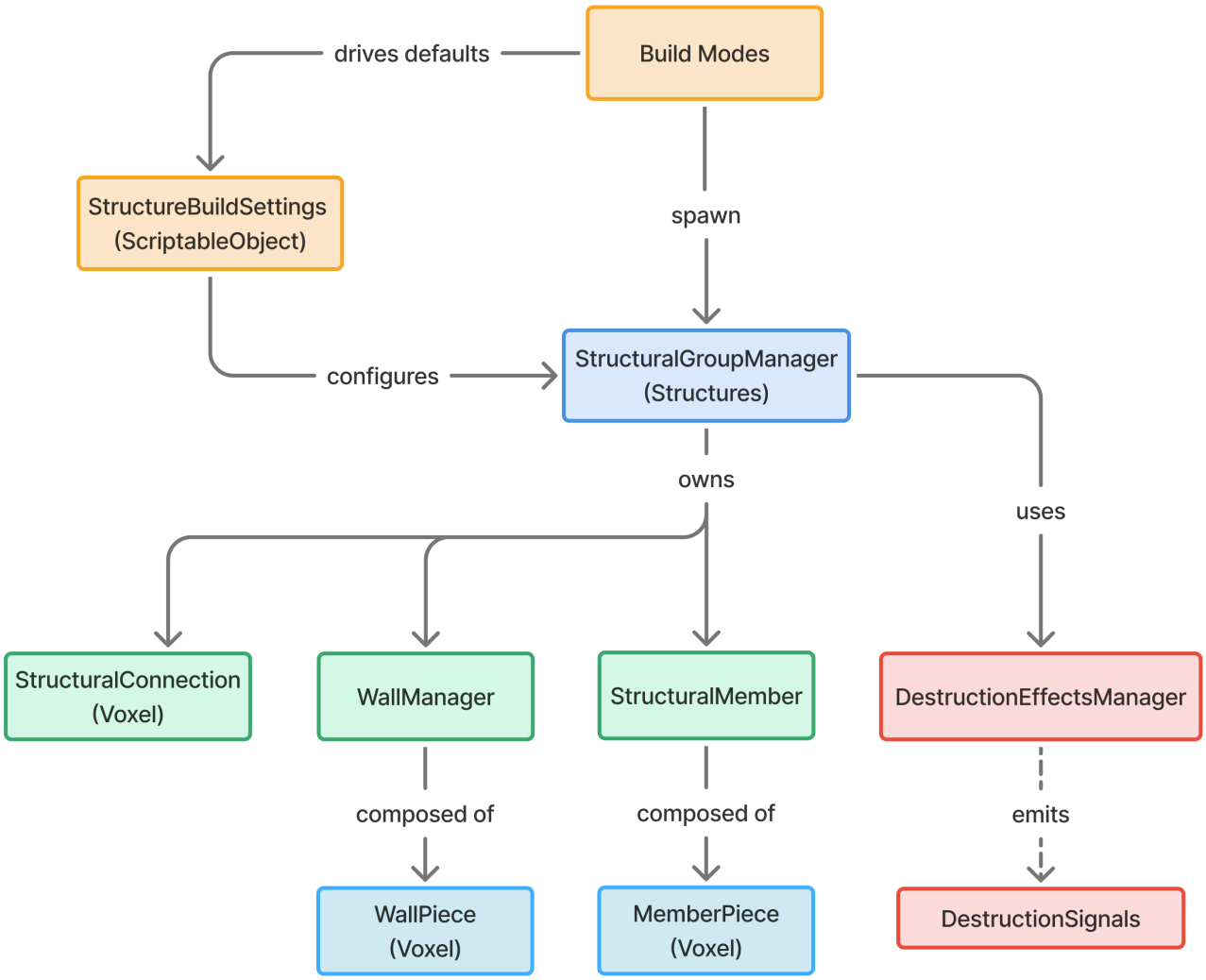

Figure 1: DSB System Overview

Component Overview

Component	Role
StructuralGroupManager	Root component on every structure. Tracks members, walls, and connections; runs stress validation; handles chunk splitting/detaching; and stores structure-wide defaults such as voxel health and density.
StructuralMember / Wall / StructuralConnection	Runtime pieces that represent beams, wall panels, and joints. They hold voxel data, health, colliders, and references used by the stress solver and damage pipeline.
DestructionEffectsManager	Scene singleton that spawns pooled debris, audio, and particle effects for crumble/stress/collapse events. Also schedules delayed destruction and optional runtime wall builds.
DestructionSignals	Static event hub that broadcasts piece-level crumble/collapse notifications so external systems can react (e.g., UI, achievements, custom FX).
StructureBuildSettings (ScriptableObject)	Asset referenced by structures and the effects manager to centralize material defaults, mesh-cache paths, voxel scales, and other authoring settings.
StructureNavmeshRebaker *(optional)*	Listens for crumble/collapse signals and triggers NavMeshSurface rebuilds after debris or detached groups change walkable geometry (requires `com.unity.ai.navigation`).

Requirements & Compatibility

Category	Details
Unity Editor	**Unity 6 LTS, 2022.3 LTS, 2021.3 LTS**
Platforms	**Windows, Mac, Linux, Mobile, Console, WebGL**
Render Pipelines	Built-in, URP, and HDRP
Assembly Definitions	Mayuns.DSB, Mayuns.DSB.Editor, Mayuns.DSB.Demo, Mayuns.DSB.Navigation
Optional Dependencies	com.unity.ai.navigation (Unity AI Navigation) – required only for navigation utilities

Limitations

- Editor-only authoring. No in-game construction or live editing.
- Destruction effects and stress simulation run at runtime, but stock tools cannot spawn new beams or walls.
- No built-in support for multiplayer/networking frameworks.
- Structures must be scaled uniformly; non-uniform scaling across axes is unsupported and can cause detached pieces or members to spawn distorted. Generating new voxels (placing walls or members) and rebuilding or caching very large structures may display a progress bar or hitch the editor.

Asset Licensing & Third-Party Content

- **Author-created assets:** Every texture, material, mesh, audio clip, particle effect, and prefab that ships in `Materials/`, `Textures/`, `Audio/`, `ParticleEffects/`, `Samples/`, and the demo scene was produced exclusively for this toolkit. You can use, remix, and ship these assets inside your own games without any additional licensing requirements or attribution.
- **No bundled third parties:** The package does not embed marketplace purchases, creative-commons audio, or any other third-party content. There are no downstream license files to track when you import or redistribute your builds.
- **Optional Unity package:** The only external dependency is Unity's `com.unity.ai.navigation` package, and it is **optional**. It is required solely if you want to use the navigation rebake utilities described under Navigation. All core editor tools, runtime scripts, and included assets function without it.

2. Installation & Setup

2.1 Install the Package

1. In Unity, open **Window → Package Manager**.
2. In the top-left dropdown, select **My Assets**.
3. Find the asset in the list, click **Download**, then click **Import**.
4. Unity will open the import dialog; click **Import** again to bring the files into your project.

Render Pipeline Notes

- **URP/HDRP users:**
  Sample materials target the Built-in RP.
  Convert them via **Edit → Render Pipeline → ... Upgrade Project Materials**.

Optional Dependencies

- **Navigation Rebaking:**
  Install Unity's AI Navigation package (`com.unity.ai.navigation`) for navmesh rebaking support.

2.2 Required Assets & Managers (Project-Level Setup)

These steps only need to be done when first setting up your project.

- Open **Tools → Destructible Structure Builder** and click **Create BuildSettings Asset**.
  This asset stores build settings/preferences and other global settings used by buildmodes and runtime systems
- In **Global Settings**, click **Create Destruction Effects Manager** to add pooled particle/audio/debris management. Note: The Destruction Effects Manager is not optional and the component must be present in any scene containing destructible structures.
- If using navigation rebaking, add **StructureNavmeshRebaker** to structure's root (a button appears in the structure inspector).

3. Quick Start Tutorial

Full Quick Start

Step 1: Setup

1. Open a new scene.
2. Launch **Tools → Destructible Structure Builder**.
3. Ensure a **BuildSettings** asset is assigned (create one if prompted).
4. Ensure a **DestructionEffectsManager** is present in scene (create one if prompted).
5. Create **EffectsLibrary** via **Assets → Create → DSB → Effects Library** or use default under **Samples/** and assign to **DestructionEffectsManager → Effects Profile** (see Assigning Effects).

Step 2: Create your first structure

- **Create Structure:** Place the root connection node via **Create Structure**. Once placed, the mode will automatically switch to **Grid Member Build**.
- **Members:** Add beams via **Grid Member Build** (snapped) or **Free Member Build** (any angle). Ghost previews show valid axes; click to confirm placement.
- **Walls:** With **Wall Build**, hover members to see magenta ghost walls, drag for size, then click to finalize. Use **Apply Design** to bring in saved layouts.
- **Stairs:** Enter **Stair Build**. Click once to set the start, move to set width, click, move to choose the end point, then click again to place.
- **Grounding:** Use **Grounded Toggle** or **Auto-Detect Grounded** to ensure members that physically touch the ground are marked **grounded** so stress simulation can propagate support correctly.
- Use the **Show Stress Gizmos** toggle in the **Structural Group Manager** (attached to the root of the structure object) to visualize load propagation.
- Adjust **Density** and **Structural Strength** as desired. Use the stress visualizer to ensure no members are over capacity (They will fail/split upon first stress solver update).
- Press **Play** and interact with the structure (collisions, projectiles, etc.) to see crumble, stress reactions, and debris pooling in action.

Assigning Effects

- **EffectsLibrary** — per-event default effects (audio + particles). Create via **Assets → Create → DSB → Effects Library** and assign to **DestructionEffectsManager → Effects Profile**.
- **MaterialEffectsLibrary** — per-material overrides. Create via **Assets → Create → DSB → Material Effects Library** and assign to **DestructionEffectsManager → Material Overrides Profile**.
- A ready-to-use **EffectsLibrary** with preset audio and particle values lives under **Samples/**; assign it to start using included effects immediately.

Applying Damage to Structures

- **Use IDamageable** – All runtime pieces (members, walls, connections) expose `IDamageable`. Call `TakeDamage(value)` from your own scripts when a raycast, projectile, or trigger hits a DSB object. Damage accumulates per voxel until the piece splits or crumbles.
- **Sample hit logic** – Raycast against the scene and call `TakeDamage` on the collider you hit:

```
using Mayuns.DSB;

if (hit.collider.TryGetComponent<IDamageable>(out var damageable))
    damageable.TakeDamage(25f);
```

- **Collision damage on detached groups** – When pieces detach and become dynamic groups, optional impact damage can be driven by **DestructionEffectsManager** → **Group Collision Damage** settings. Configure **Collision Damage Layers**, **Collision Impulse Threshold**, **Collision Impulse Sensitivity**, and **Collision Damage** to control which layers can hurt debris and how much force is required before damage is applied.
- **Effects on crumble** – Ensure a **DestructionEffectsManager** is in the scene with an **Effects Library** (and optional **Material Effects Library**) so debris, particles, and audio play when damage destroys a voxel. Without a manager, pieces still crumble but no pooled effects spawn.

Core Workflow UX Explanations

- **Ghost Previews:** Hovering connections or members shows cyan beams (members) or magenta panels (walls) that update live with snapping options.
- **Rotation:** Rotate walls/stairs before placement with the overlay rotate button or bound hotkeys.
- **Snapping:** Use the Scene overlay or hotkeys (*C* to toggle snap; arrow keys to lock X/Y/Z) to lock axes. Wall/stair placement also exposes **Edge Alignment Snapping** and **Endpoint & Midpoint Snapping** toggles in the Structure Manager.

4. Core Editor Workflow

Build Modes

Build Modes Toolbar

Mode	Description
Create Structure	Click anywhere in the Scene view to place a new root connection. This node serves as a branching point for members and walls.
Grid Member Build	Place members snapped to grid directions. Hover a connection or existing member to preview cyan beams, then click to add the member and its end connection.
Free Member Build	Place members at any angle without grid snapping. Click a start connection or member, position the other end with the mouse, and click again to confirm.
Grounded Toggle	Mark or unmark members as grounded. Grounded members are fixed to the world and pass support into the structure during stress simulation.
Wall Build	Attach walls to members. Hover a member to preview a magenta ghost wall, click and drag to set height, then release to place the wall.
Stair Build	Attach stairs to members. Hover a member to preview purple stairs, then click to place. Adjust alignment in the Inspector if needed.
Apply Design	Apply the active WallDesign to a wall. Select or create a design, then click an existing wall to update its grid layout.
Apply Material	Apply the currently selected material to structure parts. Click a connection, member, or wall to assign it instantly.
Apply Member Properties	Apply edited member settings. Adjust properties in the manager, then click a member to overwrite its current settings.
Delete	Delete a connection, member, or wall. Deleting a connection also removes any attached members or walls.

Build Overlay

Activating any build mode displays a small overlay in the Scene view. It offers **Rotate Design** and **Cancel** buttons and shows the state of snapping and axis locks. Use the overlay or hotkeys (*C* to toggle snap; arrow keys to lock X/Y/Z) while placing members, walls, or stairs.

Wall and stair placement expose two dedicated snapping toggles in **Structure Manager → Wall Build** and **Stair Build**:

- **Edge Alignment Snapping** pulls the wall preview toward the nearest edge on the host face. Disable it when you need free placement anywhere across the surface.
- **Endpoint & Midpoint Snapping** keeps the preview aligned with the host member's midpoint or endpoints along its length. Toggle it quickly from the Scene view with the *M* hotkey.

Hotkeys

Hotkeys are provided for exiting the build mode easily, toggling snapping, and rotating walls before placement. Toggle them via **Structure Manager → Hotkeys** and configure bindings from **Edit → Shortcuts** (search for *DSB/Structure Build*).

- **Esc** – Exit the current build mode.
- **C** – Toggle snapping.
- **LeftArrow/RightArrow/UpArrow** – Lock/unlock axis constraints when free-building.

Wall & Member Authoring

Wall Build Workflow

- Hover a member to preview a magenta ghost wall; click and drag to adjust width, click again to start adjusting height, and click a final time to create the wall with the **Default Wall Design** (or an empty grid if none).
- Rotate the preview via the overlay or hotkeys before placing.
- Use **Apply Design** after placement to swap layouts.

Wall Design Inspector

The **Wall Design Inspector** is a custom editor that appears when a WallDesign asset is selected. It shows a color-coded grid of cells and lets you modify layouts in real time.

Features

- **Interactive Grid** - Shows each cell in a color-coded preview. Clicking or dragging edits live.
- **Hover Preview** - Moving the cursor over cells reveals a transparent preview of the active brush.
- **Brush Palette** - A legend below the grid lets you pick the current cell type to paint with.
- **Multi-cell Editing** - Click and drag across multiple cells to apply the brush quickly.
- **Auto Grid Resize** - Changing the rows or columns fields automatically adjusts the cell list.

How to Use

1. **Create a Wall Design** – **Assets → Create → DSB → Wall Design** or use **Apply Design** mode to create or open a WallDesign asset.
2. **Edit the Design** – Select the asset so the Inspector shows the editable cell grid, or open the asset in **Apply Design**.
3. **Paint Cell Types** – Activate a brush from the legend then click or drag in the grid. Hovering shows a semi-transparent preview.
4. **Use Triangle or Sloped Cells** – Pick the triangle or sloped brush and use the **Rotate** button to cycle orientations for angled pieces.
5. **Erase or Reset** – Select the **Empty** brush to clear a cell.
6. **Apply in Scene** – Use **Apply Design** from the Build Modes Toolbar to apply the saved layout to any wall.

Copy Wall Design From Scene

While in **Apply Design** or **Wall Build**, click **Copy Design From Wall in Scene** in the Structure Manager window. Then click an existing wall to load its layout into the active design or a WallDesign asset.

Saving a Wall Design

1. Enter **Apply Design** and make a new design or copy one from the scene.
2. Click **Select...** to choose a folder under *Assets/*.
3. Type a unique *Design Name* (indicator turns green when valid).
4. Press **Save Design** to create a .asset file storing cell types, rows, columns, and materials.

Apply Material / Apply Member Properties

- **Apply Material** mode assigns the selected material to connections, members, or walls with a click.
- **Apply Member Properties** pushes edited member settings from the manager onto individual members by clicking them in the scene.

Stairs Build Workflow

- Switch to **Stair Build**. Click once to set the start, move the mouse to set width, click to lock it, then move again to choose the end point and click a final time to place the stairs.
- Step orientation is determined relative to the structure's orientation; ensure parents are uniformly scaled and unrotated to avoid unexpected step rotations.

Texture Scaling

Every wall and structural member stores **Texture Scale X** and **Texture Scale Y**, representing meters per texture repeat along local axes. DSB divides the object's world size by these scales to derive tiling, then

multiplies mesh UVs so textures repeat consistently regardless of dimensions. A scale of 1 means one meter per tile; 0.5 doubles tiling; 2 halves it; 0 falls back to one meter. Global defaults live in **Structure Build Settings** (*Member Texture Scale X/Y* and *Wall Texture Scale X/Y*), while individual walls and members can override their own values. Rebuild after changing them.

Structure Authoring & Management

Structural Group Inspector

Selecting a **Structural Group Manager** exposes foldouts that mirror the custom inspector:

- **Overview** – Toggle *Show Stress Gizmos* to overlay stress colours in the Scene view while in Play Mode. A non-uniform scale warning appears automatically if the hierarchy uses mismatched axes.
- **HP** – *Wall Voxel Health*, *Window Voxel Health*, and *Member Voxel Health* each include **Apply to All** buttons that push the entered value across the entire structure (Undo-supported, disabled while playing).
- **Simulation**
  - **Density** (with **Apply to All**) recalculates voxel masses.
  - **Structural Strength** controls the support capacity of each member in the structure.
  - **Validation Interval** control the frequency of stress solver updates.
- **Utilities (Editor only)** –
  - **Grounded Members → Auto-Detect Grounded** marks members touching the grounded layers configured in Global Settings.
  - **Navigation → Add NavMesh Rebaker** attaches **StructureNavmeshRebaker** (disabled if already present).
  - **Voxel Operations → Rebuild Voxels** regenerates meshes/colliders.
  - **Validate Cache** repairs cached mesh references (available when mesh caching and the cache folder are configured) and reports missing caches or members needing rebuilds, plus the current cache folder path.
  - Additional info messages appear if mesh caching is disabled or the global cache folder is invalid.
- **Diagnostics** – **Refresh Counts** updates readouts for wall, member, and combined totals to help verify structure contents.

Stress Visualizer

Enable **Show Stress Gizmos** in any **Structural Group Manager** inspector to color over-stressed beams while the game is running (Editor only).

Piece-level Debug Inspectors

Selecting a **Member Piece** or **Wall Piece** shows editable/read-only diagnostics such as *Accumulated Damage*, *Max/Current Health*, and voxel counts.

5. Mesh Cache & Prefab Workflow

Warning / Important

- Cache folders are **not** cleaned up automatically; delete old structure folders manually.
- Cache content is editor-generated and should **not** be committed blindly to version control.
- Treat cache folders as temporary output rather than source assets.
- Cache be deleted and regenerated at any time. The serialized data needed for regeneration lives on each piece's components (wall, member, connection).

Mesh Cache Management

DSB persists generated meshes to disk so structures can be prefabricated and reopened quickly. When a structure is built, the system looks for meshes on disk before generating new ones. If a cached version exists, it is loaded and used immediately; otherwise the mesh is produced and written to the cache for the next session. This avoids rebuilding complex geometry every time the scene is opened.

How the cache is organized

- Choose a cache root in **Structure Manager → Build Settings → Mesh Cache → Change…**.
- Each **Structural Group Manager** writes its meshes to a subfolder derived from the group's name, scene GUID, and instanceID to prevent collisions.
- Mesh files inside the group folder are stored as Unity Mesh assets named after the individual piece identifiers. Deleting a mesh file only affects that single piece.

Maintaining the cache

- Use **Validate Cache** in a group's inspector after moving the folder, enabling caching for the first time, or changing the structure significantly. This option scans the group's folder and rebuilds any missing or out-of-date meshes. If the folder isn't present, it will automatically create one.
- If cached meshes appear incorrect or pink, clear the affected group's cache folder to force a full rebuild on the next structure bake.
- **Cache folders are not cleaned up automatically.** Remove old structure group folders manually to reclaim disk space or to prevent the project from shipping unused meshes. Treat cache housekeeping as part of your release checklist to avoid shipping stale assets.

Common issues and fixes

- **Missing meshes after relocating the project** – The absolute cache path stored in the settings may no longer be valid. Reassign the root folder and run **Validate Cache**.
- **Renamed or deleted groups still consume disk space** – Manually delete their corresponding sub-folders in the cache root.
- **Unexpected geometry after editing a structure** – Stale meshes may still be loaded. Delete the group's cache folder or use **Validate Cache** to regenerate.

Prefab a Structure

1. Build your structure and select its **root object** in the Hierarchy (the parent with **Structural Group Manager**).
2. In the Inspector's **Mesh Cache** section confirm a cache folder and use **Validate Cache** if the inspector warns of missing pieces.
3. Drag the root object from the Hierarchy into the Project window to create a prefab asset.

Note: *If the prefab asset is deleted later and the instance appears red in the Hierarchy, build modes may not work correctly. Unpack the object or create a new prefab before editing.*

Common Mesh/Prefab Issues

- Prefab instances missing pieces: verify the cache folder on the root **Structural Group Manager** and click **Validate Cache**.
- Non-uniform scaling on prefab parents causes distorted debris/pieces. Ensure uniform scale before prefabbing.

6. Runtime Systems

Damage & Destruction

Damage and Destructible Workflow

Objects that splinter inherit from **Voxel** and implement **IDamageable** via concrete voxel scripts.

Script / Type	Responsibilities
IDamageable	Single-method contract: `void TakeDamage(float damage)` – opt-in to DSB's damage pipeline.
Voxel (abstract)	1. Stores an array of **DebrisData** (pre-cut meshes + materials) via `CreateAndStoreDebrisData()`. 2. Finds scene **DestructionEffectsManager** in `Awake()`. 3. On `Crumble()` spawns pooled debris, then destroys itself.

Heads-up: *If no* ***DestructionEffectsManager*** *exists,* `Crumble()` *still destroys the object, but no debris spawns.*

All voxels implement `IDamageable`. Call `TakeDamage(amount)` from your game scripts whenever a raycast, trigger, or explosion hits an object with an IDamageable component attached. Damage accumulates per voxel and eventually leads to splitting or a call to `Crumble()`.

```
using Mayuns.DSB;

if (hit.collider.TryGetComponent<IDamageable>(out var dmg))
    dmg.TakeDamage(25f);
```

Voxel System Summary

- Voxels store debris meshes and materials up front so runtime crumble can pull from pooled data quickly.
- Stress simulation propagates through grounded members; ungrounded groups can detach as single masses.
- Detached structural groups can optionally take collision damage using **Collision Damage Layers**, **Collision Impulse Threshold**, **Collision Impulse Sensitivity**, and **Collision Damage** settings on the **DestructionEffectsManager**.

Destruction Effects Manager

Effects Manager Overview

The singleton **DestructionEffectsManager** owns all runtime destruction feedback: it prepares reusable debris shells, pools particle systems and audio sources, and reacts to the events raised through **DestructionSignals** (`PieceCrumble`, `MemberStress`, `LargeCollapse`, `WindowShatter`, and `DebrisImpact`). Create one automatically from **Structure Manager → Runtime Debris Settings → Create DestructionEffectsManager in Scene**

or add the component manually. The script enforces a single instance at runtime by destroying duplicate components during `Awake()`.

Assign an **Effects Library** asset for the baseline particle/audio response and optionally a **Material Effects Library** so individual materials can override those defaults. When an event fires, the manager locates the best matching effect, plays pooled audio, spawns pooled particle prefabs at the event location, and schedules both for automatic cleanup. Without an assigned profile no effects will play, so plug in the provided sample library from **Samples/** or create your own.

Effects Libraries

- **Effects Profile** (global defaults) and **Material Overrides Profile** (per-material overrides) control which audio/particle prefabs spawn.
- Profiles can be swapped per scene to adapt effects to different environments.

Debris Behaviour & Pooling

- **Audio & Particle Effects** – *Effects Profile*, *Material Overrides Profile*, plus *Particle Pool Size* and *Audio Pool Size* to cap retained inactive systems.
- **Debris Pooling** – *Max Active Debris*, read-only *Current Active Debris*, *Max Pool Size* (pooled shell capacity), and *Initial Pool Size* (pre-generated shells).
  Debris Layering & Shadows – *Debris Layer* assigns a single layer to every debris shell; **Debris Shadows** toggles shadow casting.
- **Persistent Debris** – Enable to keep resting debris in the scene. Anything that settles on the *Persistent Debris Ground Layers* configured on **DestructionEffectsManager** (or the global grounded layers when left empty) is merged into combined meshes every few seconds so rubble piles persist without thousands of active rigidbodies. **Debris Colliders** – *Debris Collider Scale* adjusts automatically generated box colliders (default 0.5 = 50% shrink) to trade precision for stability.
- **Debris Spawn Motion** – *Debris Explosiveness* and *Debris Spin* apply random linear and angular impulses when debris activates.
- **Debris Lifetimes (seconds)** – *Debris Lifetime* controls how long pooled debris remains active before returning to the pool. *Detached Group Lifetime* removes large detached structural groups after they have been spawned (these are not pooled).
- **Collapse Thresholds** – *Large Collapse Mass Threshold* defines how much mass must detach before the “Large Collapse” effect and audio are allowed to trigger.

Detached structural groups can take damage when they collide with the world. Use the **Group Collision Damage** fields to configure this behaviour:

- **Collision Damage Layers** – layers capable of dealing impact damage.
- **Collision Impulse Threshold** – minimum collision impulse before damage is considered.
- **Collision Impulse Sensitivity** – multiplier applied to impulse over the threshold.
- **Collision Damage** – base damage applied per qualifying impact.

Debris Impact Sound Hook

Every pooled debris carries a **DebrisCollisionSound** component. When debris collides, it raises `DestructionSignals.DebrisImpact`, letting you play custom impact audio or particles by subscribing to the event.

Workflow Quick Tips

- **Balance pools and lifetimes** – Increase *Max Active Debris* and *Max Pool Size* when you want debris to linger, or shorten *Debris Lifetime* for faster cleanup.
- **Persistent piles** – Enable **Persistent Debris** and set **Ground Layer** (found in global settings in main DSB window) to keep rubble piles around without maintaining active rigidbodies.
- **Collider tuning** – Lower **Debris Collider Scale** when debris jitter; raise it toward 1.0 to match the visual mesh more closely.
- **Manual spawning** – Custom scripts can call `GetReusableDebrisShell()` then `RegisterTimedDebris()` (or `RegisterDebris()`) to integrate bespoke debris flows.
- **Event hooks** – Subscribe to **DestructionSignals** (for example `DebrisImpact` or `LargeCollapse`) to layer additional VFX/SFX alongside the pooled defaults.

Navigation (Optional)

Navigation Overview

DSB ships with an optional helper that ties into Unity's AI Navigation package. Install `com.unity.ai.navigation` to keep pathfinding data aligned with collapsing structures.

StructureNavmeshRebaker

Attach this component to a **Structural Group Manager** to automatically refresh a `NavMeshSurface` whenever pieces crumble or large collapses occur.

- Auto-adds/configures a `NavMeshSurface` to collect child physics colliders, throttling rebakes via `debounceSeconds` and `minDelaySeconds`.
- Validates colliders on startup, warning about meshes that cannot be baked.

7. Performance & Best Practices

Area	Recommendation
Debris Pool	Balance *Max Active Debris* vs *Debris Lifetimes* – long lifetimes need bigger pools.
Wall & Member cell budget	Keep wall grids ≤ 10×10 and reduce member subdivision where possible.
Structure Piece Count Reduction	If you see hitches during large break events, reduce total members/walls/cells per structure.
Physics Layers	Exclude debris-to-debris collisions via the Layer Collision Matrix.
Cell count guidance	Prefer smaller voxel budgets on sprawling structures to avoid costly rebuilds.
Debris pool cost management	Tune pool sizes and lifetimes together to prevent runaway allocations.
Avoiding non-uniform scale	Keep transforms uniformly scaled to prevent distorted pieces or unstable physics.

8. Extending DSB

Event System

DSB supports two approaches to customizing the audiovisual output of destruction:

- **Configure effects without code** – Assign an **Effects Profile** and optional **Material Overrides Profile** in the **Destruction Effects Manager** inspector to swap out particles/audio per event or per material. This is the fastest way to re-skin destruction; see Assigning Effects for authoring guidance.
- **Extend with code** – Subscribe to **DestructionSignals** to run your own logic in response to destruction events, optionally alongside inspector-driven effects. The remainder of this section focuses on this scripting-based workflow.

DestructionSignals Quick Reference

Event name	Parameters	Trigger
DestructionSignals.PieceCrumble	`(StructuralGroupManager group, Material material, Vector3 position)`	A structural piece (member, wall, or connection) crumbles.
DestructionSignals.MemberStress	`(StructuralGroupManager group, Material material, Vector3 position)`	The stress solver flags a beam as overstressed.
DestructionSignals.LargeCollapse	`(StructuralGroupManager group, Material material, Vector3 position)`	A detached group of more than four members collapses.
DestructionSignals.WindowShatter	`(StructuralGroupManager group, Material material, Vector3 position)`	A window panel is destroyed and shatters.
DestructionSignals.DebrisImpact	`(StructuralGroupManager group, Material material, Vector3 position)`	A flying debris collides with the world.

Event Examples

Most code-driven integrations follow this pattern:

- Subscribe to **DestructionSignals** events to react to destruction moments (voxel crumbles, stress spikes, collapses, impacts). All events share the same signature, enabling structure-aware, material-aware, and location-based effects.
- Pair your scripts with inspector-driven content if desired. The **Effects Profile** (global defaults) and **Material Overrides Profile** (per-material overrides) continue to spawn the configured particles/audio while your code layers additional behaviour on top.

Runtime Scripting API

```
using Mayuns.DSB;
using UnityEngine;
/// <summary>
/// Example of wiring custom logic into every stage of destruction.
/// </summary>
public class DsbEventBridge : MonoBehaviour
{
    void OnEnable()
    {
        // voxel-level crumble
        DestructionSignals.PieceCrumble += OnCrumble;
        // individual beam stress
        DestructionSignals.MemberStress += OnMemberStress;
        // wholesale collapse
        DestructionSignals.LargeCollapse += OnCollapse;
        // glass shatters
        DestructionSignals.WindowShatter += PlayGlassTinkle;
        // debris hits anything with a collider
        DestructionSignals.DebrisImpact += SpawnDustPuff;
    }
    void OnDisable()
    {
        DestructionSignals.PieceCrumble -= OnCrumble;
        DestructionSignals.MemberStress -= OnMemberStress;
        DestructionSignals.LargeCollapse -= OnCollapse;
        DestructionSignals.WindowShatter -= PlayGlassTinkle;
        DestructionSignals.DebrisImpact -= SpawnDustPuff;
    }

    /* ----------------- event handlers ----------------- */
    void OnCrumble(StructuralGroupManager group, Material m, Vector3 pos) {
        Debug.Log($"Chunk of {m.name} crumbled at {pos}");
    }
    void OnMemberStress(StructuralGroupManager group, Material m, Vector3 pos) {
        // e.g. flash a warning UI
    }
    void OnCollapse(StructuralGroupManager group, Material m, Vector3 pos) {
        // big boom - spawn camera shake, etc.
    }
    void PlayGlassTinkle(StructuralGroupManager group, Material m, Vector3 pos) {
        // trigger bonus shards
    }
    void SpawnDustPuff(StructuralGroupManager group, Material m, Vector3 pos) {
        // lighter particle on every debris impact
    }
}
```

Example Scripts

Use the sample **DsbEventBridge** above as a template for custom integrations such as analytics hooks, dynamic lighting, or gameplay modifiers that respond to destruction events.

9. Troubleshooting Guide

Performance Issues

- **Stuttering / Low FPS during destruction events (Especially on WebGL builds)**
  - Increase **Fixed Timestep** in **Project Settings → Time**. A timestep between 0.025 and 0.03 is recommended.
  - Decrease **Max Active Debris** in **Destruction Effects Manager** in your scene. Keeping this under 500 is recommended for the best performance.
  - Decrease **Debris Lifetime** in **Destruction Effects Manager** in your scene. Keeping this under 10 seconds is recommended for the best performance.
  - Assign a seperate layer for debris in **Destruction Effects Manager → Debris Layer** in your scene, and modify the physics matrix in **Project Settings → Physics** to disable self collisions or any unwanted collisions with other layers.
- **Diagonal or orthogonal ghost beams never appear.**
  - In **Build Settings → Hide Axis** set **None** or **Orthogonal**.

Editor & Build Tools

- **Build tools not working?**
  - Ensure gizmo visibility is enabled. (See Build Modes)
- **Diagonal or orthogonal ghost beams never appear.**
  - In **Build Settings → Hide Axis** set **None** or **Orthogonal**.

Prefabs & Mesh Cache

- **Structure invisible or missing pieces in a prefab?**
  - Select the root **Structural Group Manager**, verify the cache folder path, and click **Validate Cache**. (See Mesh Cache & Prefab Workflow)
- **Detached pieces look stretched or members spawn distorted.**
  - Non-uniform scaling is likely applied somewhere in the hierarchy. Use uniform scale on the structure and its parents.

Materials & Visuals

- **Walls are pink.**
  - Assign a material in **Build Settings → Wall Material** or use **Apply Material** mode.
- **Stair steps rotate improperly.**
  - Step orientation follows the structure's transform; ensure uniform, non-rotated parents.

Runtime Effects

- **Want default effects without making your own library?**
  - Use the pre-made **Effects Library** under **Samples/** and assign it in **DestructionEffectsManager → Effects Profile**. (See Assigning Effects)

- **Pieces, debris, and detached groups disappear too quickly.**
  - Increase *Max Active Debris*, adjust debris & detached group lifetime, or enable persistent debris in **Destruction Effects Manager**.
- **NavMesh doesn't rebake on collapse.**
  - Install `com.unity.ai.navigation` and add **StructureNavmeshRebaker** to your structure's root. (See Navigation)

Performance

- **Lag spikes during large collapse or damage events.**
  - Reduce total component/voxel count of the structure and max debris count. (See Performance & Best Practices)

10. Support & Legal

- **Email:** support@mayuns.com
- **Discord:** https://discord.gg/73GaMeP6JF
- **Website:** https://mayuns.com

Runtime & editor scripts are distributed under the standard **Unity Asset Store EULA**. All bundled textures, meshes, audio clips, particle presets, materials, and demo scenes were authored by Antonio Indindoli (dba Mayuns Technologies) and can be used in your shipped projects without any additional licensing restrictions or attribution.