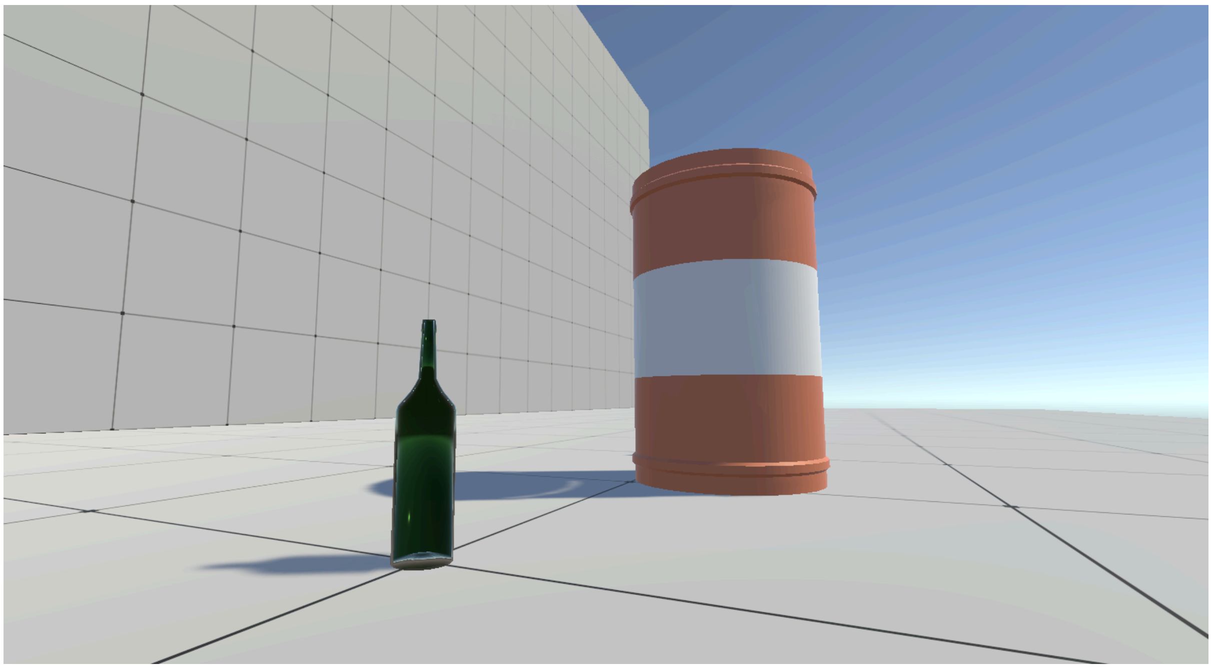
Explosive Objects

Unity Explosion & Destruction Framework



by FaberBrizio

Overview

Explosive Objects is a modular Unity asset that allows any object to trigger a configurable explosion sequence, including sound effects, particle effects, and optional destruction into physical fragments. Explosions can be triggered externally from other scripts, enabling seamless integration with gameplay systems such as weapons, triggers, puzzles, or AI behaviors.

The asset currently ships with:

- A **barrel** model and exploded fragments
- A **bottle** model and exploded fragments

Both are fully compatible with the included scripts and serve as reference implementations.

Core Concepts

- **ExplosiveObject**
  Represents an object that can explode when triggered.
- **BrokenObject**
  Controls the physics behavior and lifetime of exploded fragments.
- **ExplosionController**
  Example manager for triggering explosions and respawning objects (demo / testing utility).

The system is intentionally decoupled: any script can call `Explode()` on an `ExplosiveObject`.

Script Documentation

ExplosiveObject.cs

Purpose

Handles the full explosion lifecycle:

- Initializes sound and visual effects
- Plays randomized explosion audio
- Triggers particle systems
- Spawns a broken version of the object
- Destroys the original object

Key Features

- External trigger via `Explode()`
- Randomized audio clip and pitch variation
- Supports nested particle systems
- Optional broken object instantiation
- Automatic cleanup of effects

Inspector Fields

Explosion Assets

- **Broken Bottle Prefab**
  Optional prefab containing fractured pieces (must have `BrokenObject` attached).

Sound Effects

- **Sound FX**
  Prefab containing an `AudioSource`.
- **Explosion Clips**
  Array of explosion audio clips. One is selected randomly.

- **Pitch Variation Range**
  Adds subtle randomness to explosion pitch.
- **Destroy Sound Effect After**
  Time (seconds) before the sound object is destroyed.

Visual Effects

- **Explosion VFX**
  Particle system prefab for explosion visuals.
- **Destroy Visual Effect After**
  Time (seconds) before the VFX object is destroyed.

Public Methods

`Explode()`

Triggers the explosion sequence.

Behavior:

1. Plays a randomized explosion sound.
2. Emits particles from the main and child particle systems.
3. Instantiates the broken object prefab (if assigned).
4. Applies randomized velocities to fragments.
5. Destroys the original object.

Usage Example:

```
explosiveObject.Explode();
```

This method is safe to call from:

- Player weapons
- Collision events
- Timers
- AI logic
- Trigger volumes

BrokenObject.cs

Purpose

Controls the physical behavior of an object after it has exploded into pieces.

Key Features

- Applies randomized velocity to each fragment
- Automatically destroys fragments after a set time
- Simple and lightweight physics handling

Inspector Fields

Breaking Settings

- **Pieces**
  Array of fragment GameObjects (must have `Rigidbody` components).
- **Velocity Multiplier**
  Scales the random velocity applied to each piece.
- **Time Before Destroying**
  Lifetime of the broken object before cleanup.

Public Methods

`RandomVelocities()`

Applies a random velocity vector to each fragment.

This method is typically called automatically by `ExplosiveObject` when the object explodes.

ExplosionController.cs

Purpose

Example controller used for testing, demos, or prototyping.
Not required for runtime use in production games.

Key Features

- Triggers explosions on multiple objects
- Respawns explosive objects at predefined positions
- Keyboard-controlled for rapid testing

Inspector Fields

- **Explosive Objects**
  Runtime list of active `ExplosiveObject` instances.
- **Explosive Object**
  Prefab used for respawning.
- **Explosive Obj Transforms**
  Spawn points for respawned objects.

Controls (Demo Only)

Key	Action
E	Explode all registered explosive objects
R	Respawn explosive objects at saved positions

Typical Workflow

1. Add **ExplosiveObject** to a GameObject.
2. Assign:
   - Sound FX prefab
   - Explosion VFX prefab
   - Explosion audio clips
   - Optional broken object prefab
3. Add **BrokenObject** to the fractured prefab and assign pieces.
4. Call `Explode()` from any script when needed.

Integration Notes

- Fragment pieces must have `Rigidbody` components.
- Particle systems should be set to **Emit via script**, not auto-play.
- The asset is compatible with:
  - FPS / TPS weapons
  - Environmental destruction
  - Puzzle mechanics
  - Physics-based gameplay

Extensibility

The system is intentionally minimal and can be extended to include:

- Damage radius
- Force-based explosions
- Camera shake
- Decals
- Chain reactions
- Network replication