NOISE TRAP COMPONENT

Noise Traps allow ordinary scene objects to become reactive sources of **noise events**. When a trap is pushed, bumped, hit, or moved, it releases a noise signal through its Noise Component. Any **actor with a Senses Component and an active Hear Sense** can detect this noise based on distance, propagation, and its internal detection settings. This makes the environment feel more alive and interactive, and it allows designers to create believable stealth, investigation, and physics-driven reactions without additional scripting.

What Noise Traps Can Achieve

Noise Traps are an easy way to let the world respond to physical interactions. Once placed in a scene, they automatically emit noise values whenever they receive force—whether from thrown objects, falling debris, players, or other actors. Designers can use Noise Traps to introduce:

- **Stealth hazards**, where bumping an object can give away a position
- **Player-driven distractions**, such as kicking a can to lure a guard
- **Environmental storytelling**, like a bottle falling off a shelf
- **Dynamic physics reactions**, where objects collide and trigger noise
- **Unscripted chain events**, creating emergent gameplay moments

By attaching a Noise Trap to props, crates, cans, or any movable object, you add depth and unpredictability to your level while keeping everything fully simulation-driven.

How Actors With Senses React (Fair, Neutral, and Non-Cheating)

A unique and important behavior of Noise Traps is that the **noise is always attributed to the trap itself**, not to the actor that caused the interaction. If a player or NPC bumps into a trap, the resulting noise is linked to the trap's own TargetSenses component.

This ensures that actors with Hear Sense respond in a **neutral, world-based** way:

"A noise happened at that location. I should check it out."
-not-
"The player made that noise."

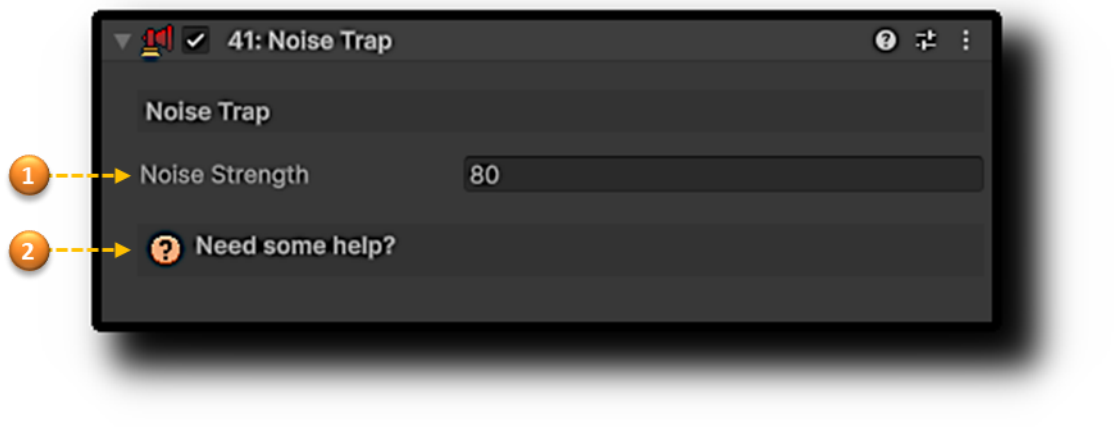
This prevents unfair or cheating behavior. Actors only react to the world as they experience it—by detecting the trap's noise value and deciding how to respond based on normal Senses rules (distance, thresholds, NavMesh path, surface propagation, etc.).

This keeps gameplay natural and consistent:

- A guard hearing a can fall will investigate the can
- A creature detecting a crate sliding will move toward the crate
- An NPC noticing a heavy object drop will check that area

Noise Traps allow the world—not invisible logic—to communicate with your actors.

Noise Trap Inspector Tab



1	Noise Strength	Defines how strong the trap's noise signal is when force is applied. Higher values travel farther and alert more AI.
2	'Need some help?'	Displays Help Information.

Integration – Two Ways

Noise Traps can be triggered in two distinct ways, depending on what type of object interacts with them. This separation exists because Unity handles physics objects and CharacterController movement very differently. Both paths ultimately produce a noise event through the Noise Component, but the way the force is applied is different.

Noise Trap Triggered by OnCollisionEnter (Physics Interaction)

This is the simplest and most automatic integration method. If any **Rigidbody** collides with the trap-such as a thrown rock, a falling object, or another physics prop—Unity will call OnCollisionEnter. The Noise Trap handles this event internally:

- It evaluates impact strength
- Applies a physical reaction force to itself
- Releases a noise value based on the impact

This requires **no setup** beyond placing the trap into your scene. Any physics-based interaction will result in a realistic noise event that actors with Hear Sense can detect.

Noise Trap Triggered By Character Controller (Example Usage)

Unity's **CharacterController** does not participate in Rigidbody physics and does **not** trigger OnCollisionEnter.
To support this movement style, the package includes an example helper script: **NoiseTrapPusher**.

This script uses OnControllerColliderHit to detect when a CharacterController touches a Noise Trap. If a trap is hit, the script calculates a push direction and calls the trap's AddForceAtPosition method to simulate a physical shove. The trap then reacts normally and emits a noise event.

```
public class NoiseTrapPusher : MonoBehaviour
{
    [SerializeField] float m_force;
    ColliderReference m_colliderReference;

    void OnControllerColliderHit(ControllerColliderHit hit)
     {
      if (!hit.gameObject.TryGetComponent<ColliderReference>(out m_colliderReference))
       {
        m_colliderReference = hit.gameObject.AddComponent<ColliderReference>();
       }

      if (m_colliderReference.isNoiseTrap(out NoiseTrap _noiseTrap))
       {
        Vector3 _forceDirection = hit.transform.position - transform.position;
        _forceDirection.y = 0;
        _forceDirection.Normalize();
        _noiseTrap.AddForceAtPosition(_forceDirection * m_force, transform.position,
        ForceMode.Impulse);
       }
     }
}
```

This method is perfect for:

- Players using CharacterController
- AI agents using CharacterController
- First-person and third-person controller templates

It ensures CharacterController movement produces realistic trap interactions, even without physics collisions.