HS_Impact_position – Quick Usage

How to send impacts to shield

1. Add **HS_Impact_position** to the shield object.
2. Make sure the shield has:
    - **Collider**
    - **MeshRenderer**
    - material with shader property `_hit`
3. In shader, read `_hit` as a **Vector4 array**:
    - `xyz` = impact world position
    - `w` = hit time
4. When shield is clicked or hit by collision, script automatically writes impact data into `_hit`.
5. Shader uses this array to draw impact effects on shield surface.

Manual send from code:

```
shieldImpact.AddHitFromWorldPosition(hitPoint);
```

Important:

- `maxHits` in script should match shader array size.
- Shield collider is required for mouse hit detection.

HS_ObjectActivator – Quick Usage

1. Attach script to object with **Renderer + Collider**.
2. Assign:
    - **targetRenderers** (auto if empty)
    - **targetCollider** (auto if empty)
3. Shader must have float property (default: `_Shield_step`).

How it works:

- **Left mouse click** toggles ON/OFF.
- Value smoothly animates **0** → **1** → **0**.
- Sent to shader via **MaterialPropertyBlock**.

Features:

- Enables/disables **Collider** based on value
- Plays/stops **Particle Systems** at full ON
- Toggles **GameObjects** (with optional delay)

Settings:

- `transitionSpeed` – animation speed
- `colliderDisableValue` – when collider turns off
- `objectsActivateDelay` – delay before activating objects

Notes:

- Works without modifying materials (per-renderer control).
- Requires shader support for `_Shield_step`.