

ERAGANX / UNITY TOOLKIT

2D Soft Body Pro

A practical guide to creating, tuning, and scripting deformable 2D bodies in Unity.



Choose a sprite. Pick a profile. Press play.

Unity 6000.3.13f1 • Built on Unity Physics 2D • Optional Input System integration

1. Quick start

You can make a working soft body in a few minutes. The package uses a closed Sprite outline to build a physics-driven perimeter and a deforming mesh.

- 01** Prepare a Sprite
Import a Sprite with one simple, connected silhouette. In Sprite Editor, define a clean Physics Shape. Holes and separate islands are not supported.
- 02** Open the Creator
In Unity, choose Tools > ERAGANX > Softbody 2D > Softbody Creator.
- 03** Choose settings
Assign the source Sprite and choose a profile. Set recovery, rotation lock, perimeter colliders, or trigger mode as needed.
- 04** Create and test
Click Create Soft Body. Put the generated object above a Collider2D floor and press Play. Save the whole generated hierarchy as a prefab.

What the Creator does

The Creator creates a root object with a SoftBody2D component, generated physics nodes, colliders, and a mesh renderer. Use the Inspector or call Generate again when rebuilding after source shape or quality changes. Save the generated child hierarchy with the root prefab.

Try the included first scene: Demo/Basic/01-Basic.unity. Open it, enter Play mode, then drag a jelly and release to observe its response.

2. Profiles and tuning

Profiles are reusable starting points. The Creator copies the selected profile settings onto the generated body, so runtime changes do not edit the shared profile asset. Duplicate a profile when you want a reusable custom preset.

Profile	Starting character
Jelly	Soft impact squash, elastic wobble, and shape recovery.
Jelly Mobile	Jelly preset with 12 nodes for a lower-cost starting point.
Jelly High Quality	Jelly preset with 32 nodes for a smoother physical outline.
Water-like	Loose response with recovery disabled; still a deformable solid approximation.
Rubber / Bounce	Firmer recovery or stronger contact rebound.
Wobble / Soft / Hard	Longer oscillation, balanced response, or tighter shape return.

Settings you will tune most

Setting	Effect
Softness	Reduces spring and recovery frequencies for a looser feel.
Bounce	Controls contact restitution.
Wobble and Damping	Set how much oscillation remains and how quickly it settles.
Stretch and Squish	Adjust neighboring-node separation and perimeter response.
Shape Recovery	Pulls the outline toward its authored shape. Disable for freer deformation.
Visual Smoothness	Adds interpolated mesh vertices between physical nodes; it does not add physics detail.
Node count	More nodes improve outline detail and increase physics cost.

Tip: Compare 12-node Mobile, 20-node Balanced, and 32-node High Quality on your target device. The labels are quality starting points, not measured frame-rate guarantees.

3. Drag interaction

Add one `SoftBodyDragController` to a scene object. Assign its camera, or tag the camera `MainCamera`. The controller selects a soft body by its simulated outline, then applies a mass-aware grip. It does not teleport the root. Release preserves momentum.

For a quick prototype, open `Demo/Drag/05-Drag.unity`. The sample already includes the controller.

Legacy input

When the new Input System integration is not installed or enabled, the controller supports mouse and touch through the legacy Input Manager. Use the camera conversion method and pointer API below to connect a custom input source.

Optional Input System

The separate `ERAGANX.SoftBody2D.InputSystem` assembly provides `SoftBodyPointerInput`. Add it beside the drag controller to forward touchscreen and mouse pointer events. This assembly references the Input System package; it is optional and isolated from the base runtime.

Custom pointer flow

```
using ERAGANX.SoftBody2D;
using UnityEngine;

public class DragInput : MonoBehaviour
{
    [SerializeField] SoftBodyDragController drag;

    public void Press(int pointerId, Vector2 screenPosition)
    {
        Vector2 world = drag.ScreenToWorld(screenPosition);
        drag.BeginDrag(pointerId, world);
    }

    public void Move(int pointerId, Vector2 screenPosition)
    {
        drag.MoveDrag(pointerId, drag.ScreenToWorld(screenPosition));
    }

    public void Release(int pointerId) => drag.EndDrag(pointerId);
}
```

Pointer IDs should be unique while active. One pointer controls one body at a time; another pointer may control a different body. Call `ReleaseAll()` when disabling your input layer.

4. Runtime API

The main runtime component is `ERAGANX.SoftBody2D.SoftBody2D`. Keep your component reference and invoke these methods from gameplay code:

```
using ERAGANX.SoftBody2D;
using UnityEngine;

public class JellyExample : MonoBehaviour
{
    [SerializeField] SoftBody2D body;
    [SerializeField] SoftBodyProfile profile;

    public void PopUp()
    {
        body.ApplyImpulse(Vector2.up * 2f);
        body.ApplyLocalizedForce(body.Center, Vector2.up * 3f, 0.5f);
    }

    public void ResetJelly() => body.ResetShape();
    public void ApplyProfile() => body.SetProfile(profile);
}
```

Methods at a glance

API	Use
Generate(out string error)	Build the body from its Sprite outline; inspect the error string if generation fails.
ApplyForce / ApplyImpulse	Distribute a force or impulse across the body.
ApplyLocalizedForce(worldPoint, force, radius)	Apply a force around a world-space point.
Deform(worldPoint, displacement, radius)	Explicitly edit nearby node positions; not the normal collision response.
RestoreShape(strength)	Apply one restoring-force update at the current pose.
ResetShape()	Return to the authored root pose and clear motion.
SetProfile(profile)	Apply a profile; a changed point count can regenerate the body.
Center / NodeCount / IsGenerated	Read the current center and generation state.

Use `ApplyLocalizedForce` with world-space positions. Do not regenerate every frame. Change profiles with different node counts outside active gameplay when preserving the current deformed pose matters.

5. Events and gameplay contacts

Subscribe and unsubscribe with the owning object lifecycle. Logical collision events aggregate contacts from a body's perimeter nodes. Use collision events for physical contacts and trigger events when Trigger Mode is enabled.

```
void OnEnable()
{
    body.CollisionEntered += OnBodyCollision;
    body.TriggerEntered += OnBodyTrigger;
}

void OnDisable()
{
    body.CollisionEntered -= OnBodyCollision;
    body.TriggerEntered -= OnBodyTrigger;
}

void OnBodyCollision(SoftBodyContact contact)
{
    Debug.Log($"Contact point: {contact.Point}");
}

void OnBodyTrigger(SoftBodyContact contact)
{
    Debug.Log($"Trigger from: {contact.OtherCollider}");
}
```

Available events

Event family	Events
Physical collision	CollisionEntered, CollisionStayed, CollisionExited
Trigger contact	TriggerEntered, TriggerStayed, TriggerExited
Lifecycle / impact	Initialized, Impacted, ResetCompleted

A `SoftBodyContact` describes a contributing node-collider pair. Its point and normal are not an averaged contact manifold. Trigger colliders are boundary sensors, not a filled interior volume.

Existing Unity layers still govern collision filtering. The optional Collision Mask adds a further restriction; the asset does not edit project layer settings for you.

Note on impact settings

Collision response comes from Unity Physics 2D contacts. Serialized Impact Strength, Radius, and Maximum Compression fields remain for compatibility but do not currently inject an extra impact kick. `Impacted` is a thresholded notification.

6. Samples, setup, and limits

Included sample scenes

Scene	Demonstrates
01-Basic	Fall, squash, rebound, and recovery.
02-Materials	Water-like, Jelly, Rubber, and Hard comparison.
03-Stacking	Several interacting soft bodies.
04-Recovery	Recovery enabled and disabled.
05-Drag	Physical grip and release.
06-Collision	Mixed-material contacts.
07-Stress Test	25 Mobile bodies as an illustrative setup, not an FPS promise.

Sprite and transform requirements

Use a single closed silhouette with a valid Sprite Physics Shape. Simple concavity is supported; holes, disconnected islands, self-crossing outlines, and zero-area shapes are rejected. Use planar transforms with positive scale, and set scale and parenting before generation.

Physics expectations

The body is a physics-driven deformable solid approximation, not a fluid or finite-element simulation. Extreme speed, deep initial overlap, narrow obstacles, severe concavity, and self-intersection can destabilize a perimeter simulation. Start bodies with free space around them. Stacking behavior is approximate.

Performance and target devices

Cost rises with node count, contacts, and active bodies. Each node adds Rigidbody2D, collider, spring, bending, and distance constraints; active drags add temporary target joints. Start with Mobile and profile on the hardware and scenes you plan to ship. Editor stress scenes include editor overhead and do not establish mobile performance.

Installation and dependencies

Import the complete Assets/ERAGANX/Softbody 2D folder, including .meta files. The runtime uses Unity Physics 2D. The Input System assembly is optional. Keep generated children with their root when saving a prefab.

For the latest project-specific detail, see the package Documentation folder and its validation notes.