STL DYNAMICS

SYSTEM GUIDE

# Human Shaders - Skin Visibility System

HS

URP

Skin Visibility

The Skin Visibility system lets clothing or gameplay hide selected areas of a Human Skin material without permanently deleting or modifying the character mesh.

The runtime system is built around:

```
HumanSkinVisibilityController
HumanSkinPaintProfile
HumanSkinVisibilitySet
HumanSkinVisibilityRequester
HumanSkinHideZone
```

## 1. HumanSkinVisibilityController

The controller owns visibility state for one character.

Add it to the character root or another suitable parent object.

It manages:

- Human Skin target renderers.
- Painted hide profiles.
- Runtime hide requests.
- Primitive hide zones.
- Custom convex hide zones.
- Edit-mode preview.
- Per-renderer visibility data.

If its explicit renderer list is empty, it can discover child renderers that use the Human Skin shader.

### Key limits

```
MaxZones = 32
MaxConvexPlanes = 128
```

### Important public methods

```
SetTargetRenderers(...)
RefreshVisibility()
SetProfileHidden(profile, hidden)
SetProfileHidden(regionId, hidden)
IsProfileHidden(profile)
AddProfile(profile, hiddenAtStart)
RemoveProfile(profile)
ApplyVisibilitySet(set)
RemoveVisibilityRequest(token)
AddZone(zone)
RemoveZone(zone)
```

## 2. Painted Geometry Regions

Painted regions are stored in `HumanSkinPaintProfile` assets.

The current painter stores geometry-weight data tied to a source mesh. This avoids the shared-UV bleed that can occur with legacy UV masks.

A paint profile contains:

- Display / region name
- Runtime region ID
- Geometry mesh reference
- Geometry vertex count
- Per-vertex geometry weights

Legacy texture masks are still represented by the profile format for migration/compatibility, but Diagnostics warns when a profile still relies on the older UV visibility-mask workflow.

### Authoring

Open:

Tools > STL Dynamics > Human Shaders Studio > Skin Visibility

In Surface Region Painter:

1. Assign a Skin Renderer.
2. Assign the Visibility Controller.
3. Create or select a Paint Profile.
4. Paint in Scene view.
5. Save the profile.

Painter controls include:

- Surface Radius
- Hardness
- Opacity
- Paint / Erase
- Save After Each Stroke
- Undo Stroke

- Clear Region
- Fill Region
- Refresh Posed Surface

### Scene-view controls

```
Left drag = Paint
Alt = Orbit normally
```

The brush follows connected mesh topology instead of simply painting by UV location.

## 3. Direct Painted-Profile Control

A profile can be made part of the controller's persistent region list and toggled directly.

```
using STLDynamics.HumanShader;
using UnityEngine;

public class Example : MonoBehaviour
{
public HumanSkinVisibilityController controller;
public HumanSkinPaintProfile torso;

public void HideTorso()
{
controller.SetProfileHidden(torso, true);
}

public void ShowTorso()
{
controller.SetProfileHidden(torso, false);
}
}
```

You can also address an existing region by its runtime ID:

```
controller.SetProfileHidden("region-id", true);
```

For string lookup, each profile should have a unique Region ID. Diagnostics warns about missing or duplicated IDs.

## 4. Visibility Sets

`HumanSkinVisibilitySet` is a ScriptableObject containing a group of paint profiles that should be hidden together.

Create one with:

Create > STL Dynamics > Human Shaders > Profiles > Skin Visibility Set

Typical examples:

```
Shirt_Hide
Pants_Hide
Dress_Hide
Boots_Hide
FullBodySuit_Hide
```

A visibility set is ideal for a garment, armor piece, outfit layer, or runtime state.

## 5. Reference-Counted Runtime Requests

`ApplyVisibilitySet()` returns a token.

```
int token = controller.ApplyVisibilitySet(mySet);
```

Release that exact request later:

```
controller.RemoveVisibilityRequest(token);
```

The controller reference-counts active profiles.

This is important for overlapping clothing.

Example:

```
Garment A requests: Torso
Garment B requests: Torso + Arms
```

Removing Garment A does not reveal Torso while Garment B still needs it hidden.

This avoids the classic problem where unequipping one garment incorrectly restores skin required by another garment.

## 6. HumanSkinVisibilityRequester

For normal clothing objects, you usually do not need to manage tokens manually.

Add `HumanSkinVisibilityRequester` to the garment.

It can reference:

- A target controller
- A visibility set
- One or more hide zones
- Apply On Enable

When enabled, the requester applies its visibility set and registers its zones.

When disabled, it releases only the requests it owns.

This makes GameObject activation a simple clothing-visibility workflow.

```
Enable Shirt -> apply Shirt_Hide
Disable Shirt -> release Shirt_Hide
```

If `Target Controller` is not assigned, the requester searches its parent hierarchy.

## 7. Exact Hide Zones

`HumanSkinHideZone` defines a hard world-space clipping volume.

Supported shapes:

```
Sphere
Box
Capsule
Custom Convex Mesh
```

Create zones from:

Human Shaders Studio > Skin Visibility > Exact Hide Zones

### Sphere

Uses a radius around the zone transform.

### Box

Uses a box size and the zone transform.

### Capsule

Uses radius and height.

### Custom Convex Mesh

Converts a closed convex mesh into plane equations.

Requirements:

- Mesh must be closed.
- Mesh must be manifold.
- Volume must be convex.
- Read/Write must be enabled while the zone is baking its planes.
- At least four vertices / four triangles are required.

If the intended shape is concave, build it from multiple convex zones.

## 8. Choosing Painted Regions vs Zones

### Use painted regions when:

- The clothing should always hide the same body surface.
- You need named reusable regions.
- Outfits overlap.
- You want garment-driven enable/disable behavior.
- Exact geometry-surface control matters.

### Use exact zones when:

- Hiding is spatial.
- The affected body area should follow a transform or object.
- A primitive shape gives a quick clean result.
- A custom convex volume closely matches the intended area.

### Use both when:

A garment has a stable primary coverage region plus a few spatial areas that need extra clipping.

## 9. Edit Mode Preview

The controller can preview visibility in Edit Mode.

This is useful for authoring clothing coverage without entering Play Mode.

If you need to force an immediate update after changing data from editor tooling or code:

```
controller.RefreshVisibility();
```

## 10. Recommended Region Design

Prefer reusable anatomical or garment-coverage regions rather than creating a unique body mask for every outfit.

Example library:

```
UpperTorso
LowerTorso
Chest
UpperArms
Forearms
UpperLegs
LowerLegs
Feet
Neck
```

Then compose them into garment-specific `HumanSkinVisibilitySet` assets.

For clothing that needs unusually precise borders, create a dedicated painted profile for that garment.

## 11. Diagnostics

Human Shaders Diagnostics validates visibility profiles and warns about:

- Profiles with no geometry mask or legacy mask.
- Legacy UV visibility masks.
- Missing runtime region IDs.
- Duplicate runtime region IDs.
- Visibility-mask import settings that are not appropriate for legacy texture masks.

Run it from:

Tools > STL Dynamics > Human Shaders Studio > Diagnostics