MANUAL

Version 1.0 (18.11.2018)

Contact: info@ironsight.fi

Table of Contents

OVERVIEW

You need a way to show effects on your character model such as damage, bullet holes, scratches, stains, dirt and more. Skinned Decals will allow you to add impressive decals to your characters during gameplay (and editor) with ease, just one line of code.

The system does not require the skinned mesh to have any colliders, the decals are placed according the mesh data.

The mesh based solution allows flexibility in shaders, for example it is possible to use Tesselation to enhance the depth of the effects. Skinned decals will also perfectly follow the animation of your model.

Sample character, bullet hole decals and shaders are included!

FEATURES

- Run-time generation (with optional multithreading)
- Does not require collider
- One line of code to add decal
- Supports characters composed of many skinned meshes
- Atlasing, draw many decal types with one draw call.
- Option to add decals to character in the editor and save mesh for later use.

COMPATIBILITY

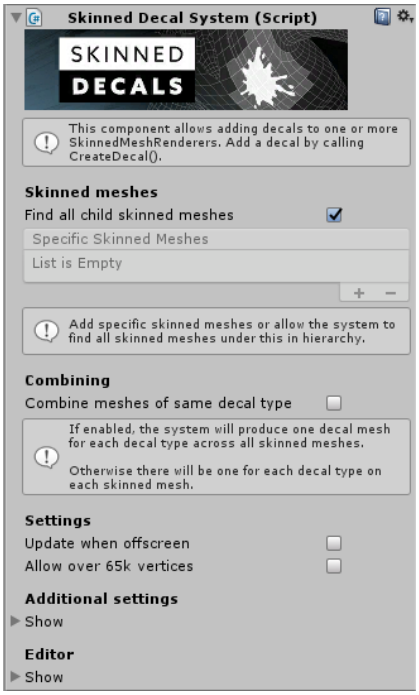
- Works with GPU skinning
- Combatible with UMA
- Doesn’t support Blendshape-animation.
- Doesn’t support cloth and dynamic meshes that are animated by something else than bones.

GUIDE

How to setup Skinned Decal System

1. Add **SkinnedDecalSystem**-component to your character gameobject.
2. Now you can add decals to this character in Editor and during run-time using **CreateDecal**-function.

   By default **"Find all child skinned meshes"**-box is checked and the system will find all skinned meshes under its hierarchy.



How to add decals in script

1. Call **CreateDecal()** function of **SkinnedDecalSystem**.cs

```
public void CreateDecal(SkinnedDecal decalType, Vector3 origin, Vector3 direction)

public void CreateDecal(SkinnedDecal decalType, Vector3 origin, Vector3 direction, Vector3 up)
```

2. This function takes in:
   a. **SkinnedDecal**-object, that defines the decal material and properties,
   b. **origin**-vector (where the decal is projected from)
   c. **direction**-vector (direction from origin where the decal is projected to).
   d. Optionally an **up**-vector for orientation.

Projecting from a transform

1. Call **CreateDecal**() with the transforms position as the origin, and forward as direction.

```
public SkinnedDecal decalType;
public SkinnedDecalSystem decalSystem;

void Start() {
    decalSystem.CreateDecal(decalType, transform.position, transform.forward, transform.up);
}
```

Projecting from a camera

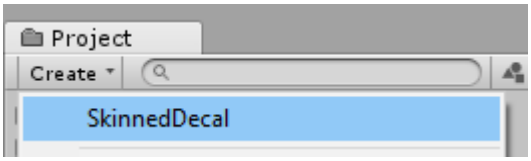
1. Create a ray from mouse position using Camera.ScreenPointToRay().
2. Call CreateDeca() with the resulting rays origin and direction, and additionally the camera transform up.

```
public new Camera camera;
public SkinnedDecal decal;
public SkinnedDecalSystem skinnedDecalSystem;

void Update () {
    if(Input.GetMouseButtonDown(0)) {
        Ray ray = camera.ScreenPointToRay(Input.mousePosition);
        skinnedDecalSystem.CreateDecal(decal, ray.origin, ray.direction, camera.transform.up);
    }
}
```

How to create a new decal type

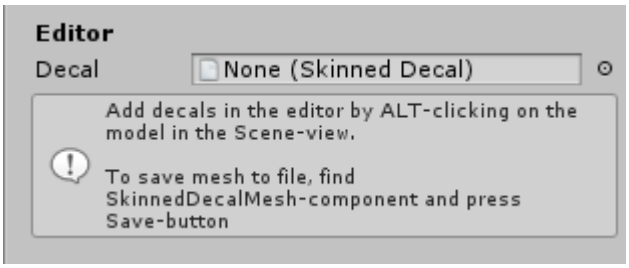
1. In the Project-window press "Create" button and find SkinnedDecal.



2. This will create a new SkinnedDecal -scriptable object that you can use to define a decal type.
3. Create a new Material with your textures and shader (you can find our included shaders under "SkinnedDecals" in the shader dropdown view) and drag it in to the material slot.
4. Set size and other settings to your liking.

How to add decals in editor

1. In **SkinnedDecalSystem**-inspector find Decal-field under Editor:



2. Drag a **SkinnedDecal** -scriptable object to this slot
3. **ALT - left click** on the character in Scene-view to add decals.

How to save decal mesh to the project

1. When you have placed decals on the model and want to save it for later use:
2. Find the **SkinnedDecalMesh**-component with your decals in the hierarchy under you SkinnedDecalSystem.
3. In the SkinnedDecalMesh-inspector, press **"Save to Assets…**" button and select the location where you want the mesh to be saved.

4. Keep the generated SkinnedMeshRenderer in the character hierarchy.
5. Use by enabling the renderer when the effect is needed.

SHADERS

Following shaders are included in the package.

Texture

Minimal shader with just color texture.

Textures:

- Albedo map

Specular

Basic shader for specular workflow.

Textures:

- Albedo map (alpha = opacity)
- Normal map
- Specular map (alpha = smoothness)

Parameters:

- Specular color
- Shininess

Parallax

Shader with parallax mapping.

Textures:

- Albedo map (alpha = opacity)
- Normal map
- Specular map (alpha = smoothness)
- Height map

Parameters:

- Parallax amount

Tessellated Parallax

Shader with parallax and tessellation.

Textures:

- Albedo map (alpha = opacity)
- Normal map
- Specular map (alpha = smoothness)

- Height map

Parameters:

- Parallax amount
- Tessellation amount
- Displacement amount

NOTE about Tessellation shaders:

Tessellation shaders supplied by this package only displace mesh to the direction of mesh normal with values that are above 0.5(50% grey to white) in the height map. Parallax effect is applied only to values below 0.5 (50% grey to black) to create illusion of depth. This method makes sure that there won't be any clipping geometry. It is extremely important that your height maps have their middle ground at 0.5 (50% grey) value.

Atlas

Shader that supports atlas textures.

Textures:

- Albedo map (alpha = opacity)
- Normal map
- Specular map (alpha = smoothness)

Parameters:

- Atlas width
- Atlas height

Packed Parallax

Same as Parallax, but with height, normal and smoothness packed to a single texture.

Textures:

- Albedo map (alpha = opacity)
- Packed normal (r = height, g = normal up, b = smoothness, a = normal right)

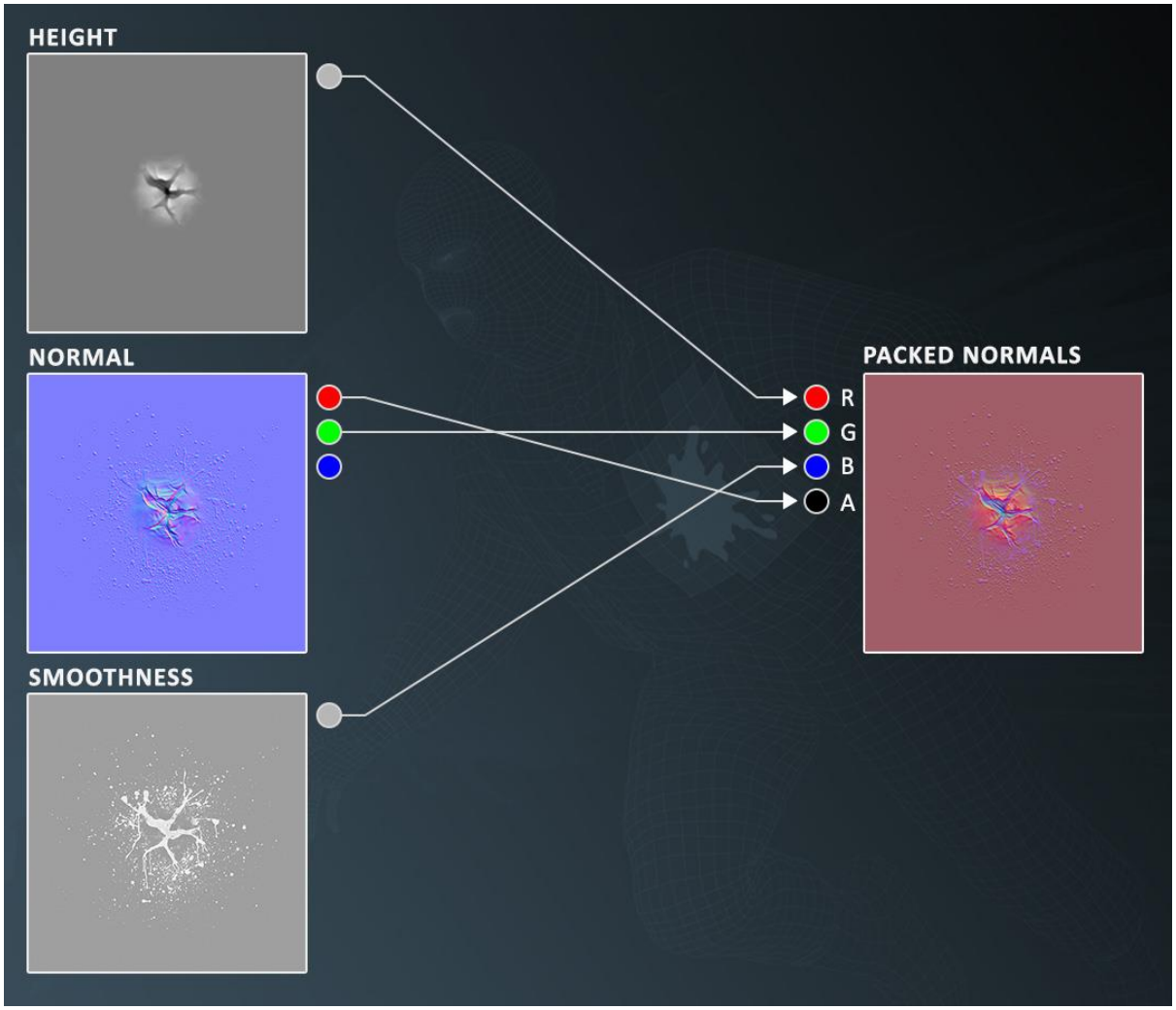
Packed Atlas

Same as Atlas, but with height, normal and smoothness packed to a single texture.

Textures:

- Albedo map (alpha = opacity)
- Packed normal (r = height, g = normal up, b = smoothness, a = normal right)
- Normal map

How to assemble Packed Normals texture:



Important about texture import settings:

- Height maps
  - sRGB(Color Texture) = False
- Packed Normals
  - sRGB(Color Texture) = False

COMPONENT OVERVIEW

Skinned Decal System (SkinnedDecalSystem.cs)

Provides the interface to add decal using script and defines which Skinned Mesh Renderers are affected. Creates Skinned Decal Builders for each child Skinned Mesh Renderer.

OPTIONS

Skinned meshes

- **Find all child skinned meshes**
  - If enabled, the system will use GetComponentsInChildren() to find all Skinned Mesh Renderers under it.
- **Specific skinned meshes**
  - If "find all child skinned meshes" is disabled, this list determines which Skinned Mesh Renderers are used.

Combining

- **Combine meshes of same decal type**
  - If enabled, the system will produce one decal mesh for each decal type across all skinned meshes. This means that if the system has for example 10 Skinned Mesh Renderers, the decal geometry from all these will be combined to a single mesh.
  - If disabled, each skinned mesh renderer will produce its own mesh for each decal type.

Settings

- **Update when offscreen**
  - Enables "update when offscreen" on each generated decal mesh.
- **Allow over 65k vertices**
  - If enabled, the system will continue to add vertices to decal mesh even after this limit. **NOTE**: make sure your Unity version supports this.
  - If disabled a new mesh is generated after previous exceeds this limit.

Editor

- **Decal**
  - Decal type that is used when adding decals in the Editor (using **ALT-left click**).

Splitting

- **Split vertex limit**
  - A vertex count that causes the decal mesh to be started over. The old mesh will remain but all new decal geometry will be added to the new one (until it reaches vertex limit again).

Scripting interface for creating decals:

```
public void CreateDecal(SkinnedDecal decalType, Vector3 origin, Vector3 direction)
```

```
public void CreateDecal(SkinnedDecal decalType, Vector3 origin, Vector3 direction,
Vector3 up)
```

Scripting interface for adding / removing Skinned Mesh Renderers:

```
public void AddSkinnedMesh(SkinnedMeshRenderer smr)

public void RemoveSkinnedMesh(SkinnedMeshRenderer smr)
```

Skinned Decal (SkinnedDecal.cs)

A Scriptable Object that holds data related to specific decal type. These are created by the user in Editor and need to be supplied when calling CreateDecal() on the Skinned Decal System.

SETTINGS

- **Material**
- **Size**
- **Size randomization**
- **Rotation randomization**
- **Normal clipping**
  - This normal clip is tested as a dot product between the decal projection direction (supplied to CreateDecal) and the skinned mesh normal, when selecting vertices for the decal mesh.
- **Atlasing**
  - Atlasing provides a way to include many decals in a single material, texture and therefore decal type. This allows many different decals to be rendered with a single draw call.
  - User can specify a specific item from the atlas using “selected item”, or enable “select random item” which selects item randomly each time.
  - NOTE: Atlasing works by encoding the item index in to the **vertex color** buffer or the generated decal mesh and therefore requires a shader that can calculate the correct UV from it (atlas shader provided).

Skinned Decal Builder (SkinnedDecalBuilder.cs)

These are added by the decal system to each Skinned Mesh Renderer. This component holds data related to the Skinned Mesh Renderer it manages and this is where the mesh operations are performed.

Handled by Skinned Decal System and has no functions for the user.

Skinned Decal Mesh (SkinnedDecalMesh.cs)

Holds the generated mesh data.

OPTIONS

- **Save to assets**
  - Saves the current decal mesh to user defined location in the project.
  - NOTE: The mesh is only usable if the skinned mesh using it remains in the character hierarchy.
  - NOTE: Can only be used in the editor