

# Decal Master:

## Advanced Deferred Decals

Deferred Decals extends Unity default render pipeline with projected decals. Decals can project on opaque surfaces its own materials (PBR) in real-time. This technique useful if you want add some details on your level or in real-time spawn some details like blood, bullet holes etc. Deferred Decals system works only in Deferred Rendering.

Video tutorial <https://www.youtube.com/watch?v=x8vEQaMj01M>

### How it works?

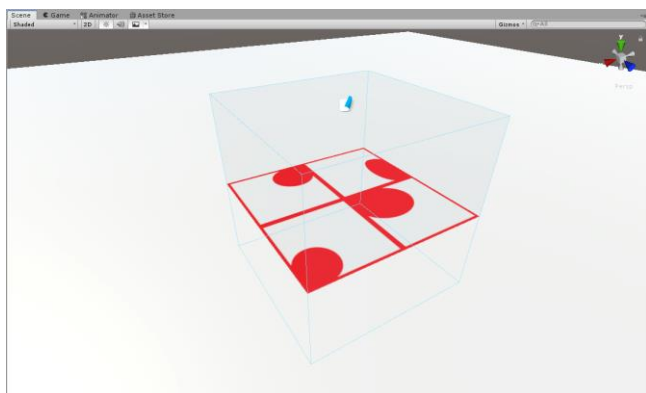
1. Register all decals
2. Setup command buffers for each camera on BeforeReflections camera event (after gbuffers)
3. Render all decals with decal shader direct to GBuffers (Diffuse, Normal, Specular Smoothness, Emission)

### How decal shader works?

Each decal reconstructs world position from CameraDepthTexture (depth) and read world normals from GBuffer2. World position is used in calculating UV and world normal is used in clipping and blending by normals.

### Quick start to use Deferred Decals System

1. Create new Scene
2. Select main camera and change 'Rendering Path' to 'Deferred'
3. Find DecalsSystem prefab in project by path 'Assets/Knife/Decal Master/Prefabs/DecalsSystem.prefab'
4. Drag and drop this prefab to scene
5. Create 'Plane' and place to (0, 0, 0) position
6. Create new Material
7. Change material shader to 'Knife/Decals/PBR'
8. Setup any opaque or semi-transparent texture to Diffuse texture slot in material
9. Create decal from menu 'GameObject/3D Object/Decal'
10. Place new decal to (0, 0.5 ,0) position
11. Assign new material to 'Material' field in inspector



## Parameters

### Deferred Decals System

Lock Rebuild	If enabled command buffers will not be recreated every frame
Terrain Decals	Defines how decals can work with terrains. Values: None, One Terrain, Multi Terrains
TerrainDecalsType:None	No decal blending with terrains height (regular projection)
TerrainDecalsType:OneTerrain	Decal blending with one terrain height (main)
TerrainDecalsType:MultiTerrain	Decal blending with all terrains heights
Terrain Height Map Size	Size of terrains heightmaps (this parameter is not change real terrain heightmap, it controls only copies of heightmaps)
Use Exclusion Mask	If enabled, all decals will be not projected on objects which have exclusion mask layer
Exclusion Mask	Decal projection exclusion layermask. All object with layer that included in Exclusion Mask will ignore decals projection. For example, you can add characters mask to ignore decals projection by characters renderers.
Frustum Culling	Enable or disable decals frustum culling
Distance Culling	Enable or disable decals distance culling
Start Fade Distance	Decals distance culling start fade distance. Begins from that distance value decals will be hidden smoothly by distance.
Fade Length	Length of distance fading
Cube Mesh	Decal mesh used in rendering
Terrain Textures	Created heightmaps
Specular Smoothness Blitter	Shader that used in specular smoothness blitting with GBuffer1

### Decal

Material	With that material decal will be rendering to GBuffers
Sorting Order	Rendering order of decal (affect only in decals rendering order)
Instanced Color	Diffuse instanced color, used when GPU Instancing enabled in material. For example, you can use one decal material but different colors on each decal.
Fade	Decal alpha blending parameter
UV Tiling	Decal tiling multiplier (useful with decals atlases)
UV Offset	Decal offset (useful with decals atlases)
Need Draw Gizmos	Does we need draw blue box when decal selected

## CustomDecalExclusionMaskRenderer

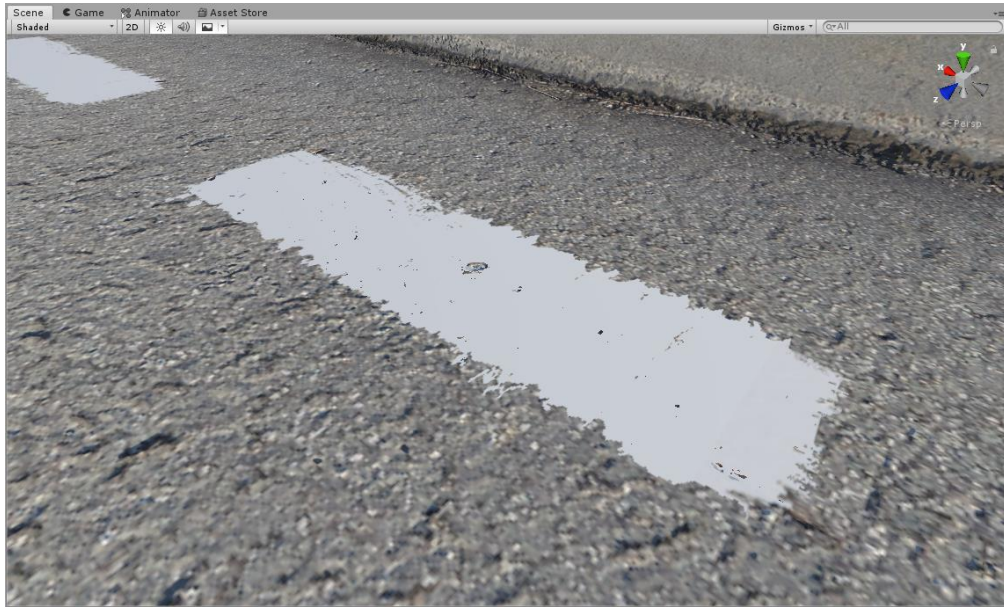
Used to exclude specific submeshes from decal projection.

Submeshes	Submeshes index array, selected submeshes will be excluded from decal projection (renders to exclusion mask). Useful if you want exclude some submesh, for example you want project decals onto trees, but don't want project them onto leaves of this tree, and tree and leaves it is one mesh, but different submeshes.
-----------	---

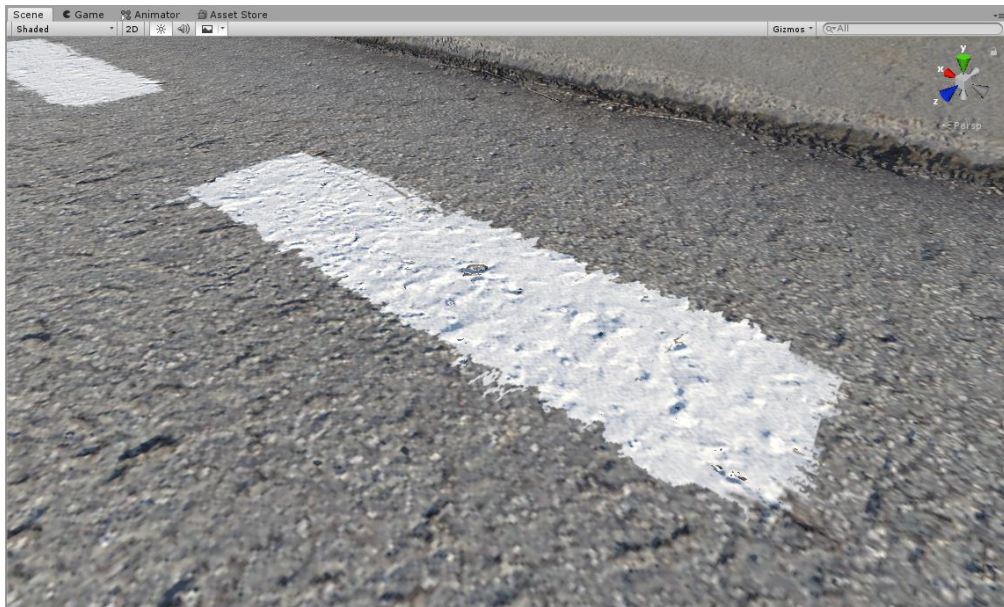
## Decal shader (Knife/Decals/PBR)

Color	Diffuse color, multiplied on diffuse map value
Diffuse (RGBA)	Regular color map, RGB – color value, A – transparency value
Normals (XYZ)	Regular Unity normal map
Normal Scale	Power of normal map
Specular (RGBA)	Regular specular workflow texture. RGB – specular color, A – smoothness value
Emission Color (HDR)	Emission color, multiplied on emission map value
Emission (RGB)	Regular emission color map, RGB – color value
Smoothness [0;1]	Smoothness value, multiplied on specular map smoothness value
Blend normal [0;1]	Control how amount normals will be blended. Zero – use normals of the surface on which decal projected. One – use normals of decal
Terrain Decal	Defines that all decals with that material will be projected only on terrain height
Clip by Normals	Controls how decal will be blended on normals. Enabled – clipping, Disabled – alpha blending
Normal Edge Blending	If enabled normals will not use diffuse color * color value alpha in normals blending. It will be use procedural soft circle mask to blend normal.
Normal Mask	If enabled normals will use diffuse map color alpha. Requires Enabled Normal Edge Blending
Clip normals	Decal normals clipping threshold
Terrain height clip	Terrain heightmap blending threshold
Terrain height clip power	Softness of terrain heightmap blending
Ignore Exclusion Mask	If true decal will ignore exclusion mask and project everywhere

## Blend normals equals 1



## Blend normals equals 0

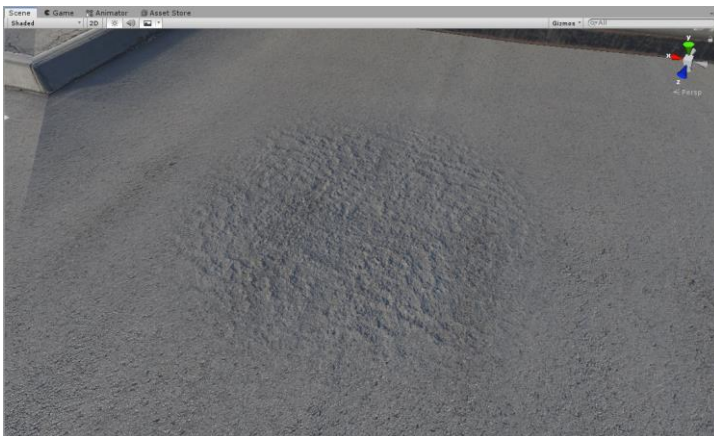




Normal map only decal and diffuse is white



Normal map only decal Normal Edge Blending enabled and diffuse is transparent



Normal map only decal Normal Edge Blending enabled and diffuse is semi-transparent texture



# Decal Master:

## Decal Placement Tool

Decal Placement Tool provide easy decal placement workflow. You don't need colliders to place decal direct on surfaces.

### How it works (GPU Raycaster)?

1. When you click in scene view specific camera renders same as scene view camera but with custom shader
2. Custom shader writes world position, world normal and object ID to 3 render targets
  - a. If we use full screen data compute shader copy data from textures to compute buffers and data from compute buffers copies from GPU to CPU (`ComputeBuffer.GetData(Array array)`)
  - b. If we need only one pixel (mouse click position) textures data copies from GPU to CPU with `Texture2D.ReadPixels` with 1 pixel rectangle
3. We have data (world position, normal and object ID if needed) and place decal with that data

### Quick Start to use Decal Placement Tool

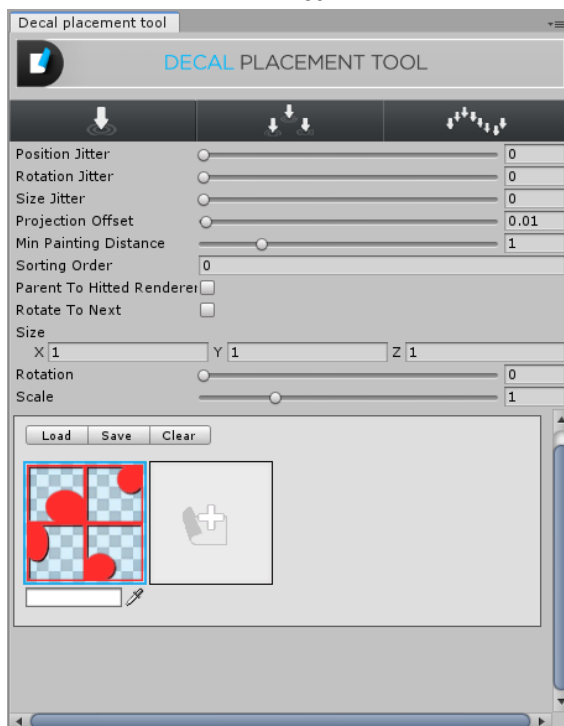
1. Open Decal Placement Tool Window 'Window/Knife/Decal Placement Tool'
2. Create new Decal Material (skip if you already have decals materials)
3. Drag and drop decals materials to Decal Placement Tool Window (pict. 1, 2)
4. Click to new decal template in Decal Placement Tool Window (pict. 3) to select template
5. Click Simple Placement Mode (pict. 4) to enable placement mode
6. Move mouse on some surface in scene view. You would see decal that projecting on surface in current mouse position
7. Click left mouse button in place where you want place decal



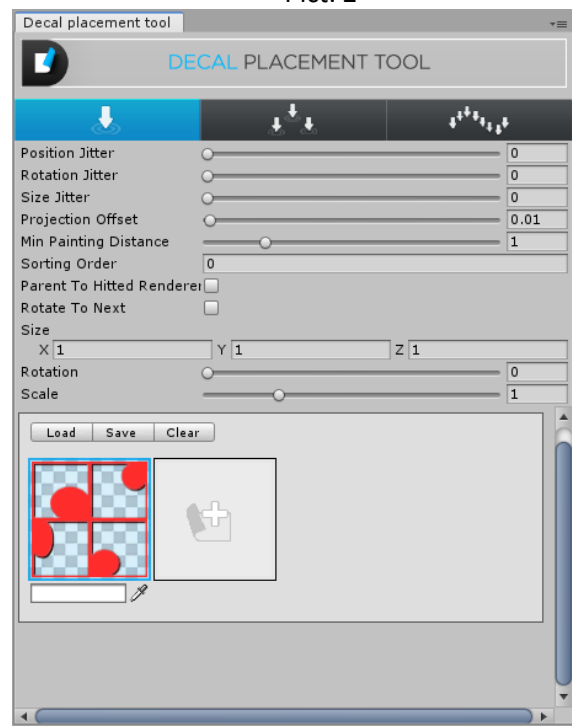
Pict. 1



Pict. 2



Pict. 3



Pict. 4

## Placement Modes

1. Simple placement mode provides one decal per one click
2. Burst placement mode provides many decals per one click with random clamped positions
3. Painting placement mode provides many decals by left mouse button holding and lines drawing.

## Control hotkeys

Hotkeys works only if some placement mode enabled and some decal template selected.

Left Mouse Button	Selected placement mode spawn selected decals
CTRL + Left Mouse Button + Mouse Move	Scale decal
CTRL + Right Mouse Button + Mouse Move	Rotate decal
CTRL + Mouse Wheel	Scale decal
SHIFT + Mouse Wheel	Rotate Decal

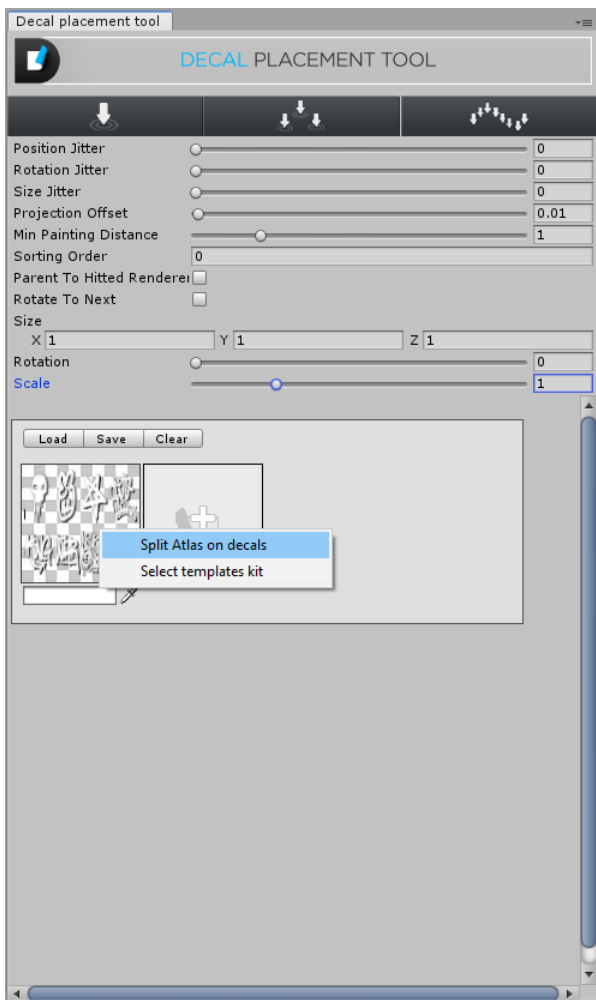
## Parameters

Position jitter	Randomness of decal position
Rotation jitter	Randomness of decal rotation
Scale jitter	Randomness of decal scale
Projection Offset	Distance between decal pivot position and projecting surface
Projection Distance	Length of projection (decal height)
Min Painting Distance	All decals will spawn only if distance between current and previous position is greater than that value. Works only in Painting Placement Mode
Sorting Order	Decals sorting order number. Decals with greater the value will be drawn above than decals with smaller the value
Parent to Hitted Renderer	If enabled all decals will be setparented object below mouse
Rotate to Next	If enabled all decals will rotate forward axis to next decal. Works only in Painting Placement Mode
Line As One Decal	If enabled only one decal will be spawned and its UV will be tiled. Works only in Painting Placement Mode
Size	Non-uniform size of decal
Rotation	Decal rotation value
Scale	Decal size value
Burst count	Decal count that will be spawned. Works only in Burst Placement Mode
Burst size	Radius of decals position randomness in screen-space coordinates. Works only in Burst Placement Mode



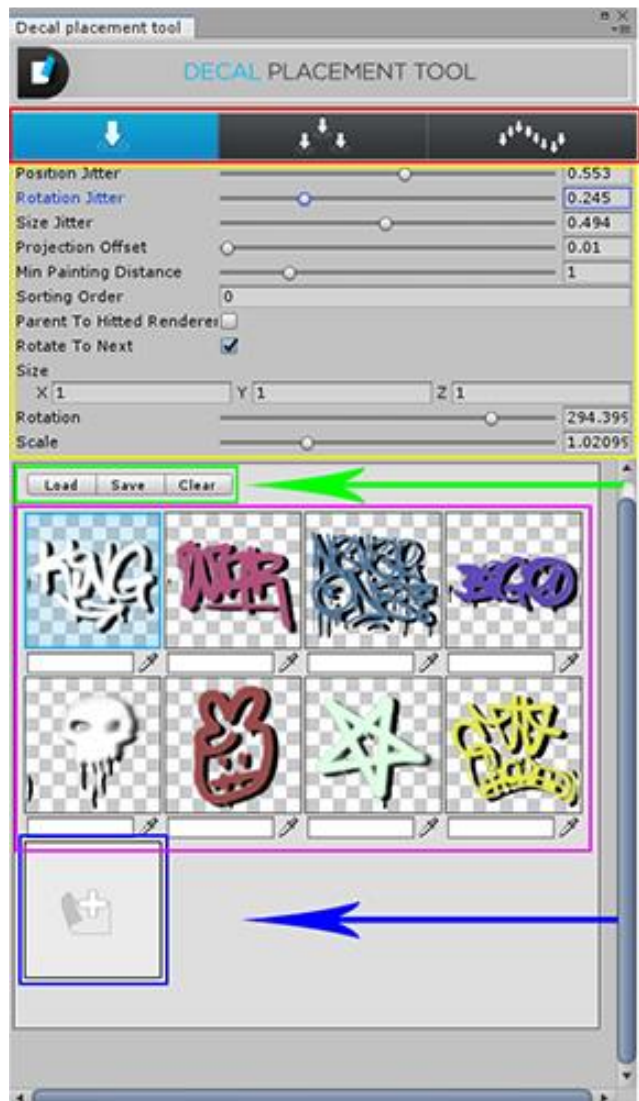
## Decals Atlas Splitter. How to use.

1. Add decal material to Decal Placement Tool
2. Click right mouse button on template
3. Select Split Atlas on decals
4. In Atlas Splitter window you can manually create rectangles (click and hold left mouse button on canvas and drag to another corner of rectangle)
5. Or you can automatically split atlas by grid
  - a. In left top corner you can see Auto Split options
  - b. Select tiles counts by x and y axes
  - c. Click auto split button and your atlas will be automatically split by grid
6. Click Split button in left top corner, window will be closed and decals will be added in Decal Placement Tool





## Decal Placement Tool Window overview



The screenshot shows the 'Decal placement tool' window. It features a top bar with three icons for placement modes. Below this is a 'Parameters' section with sliders for Position Jitter (0.553), Rotation Jitter (0.245), Size Jitter (0.494), Projection Offset (0.01), Min Painting Distance (1), and Sorting Order (0). There are checkboxes for 'Parent To Hitted Renderes' and 'Rotate To Next' (checked). A 'Size' section has input fields for X (1), Y (1), and Z (1). A 'Rotation' slider is set to 294.395, and a 'Scale' slider is set to 1.02095. Below the parameters are 'Load', 'Save', and 'Clear' buttons. A grid of eight decal thumbnails is shown, with the first one selected. A blue arrow points to a button with a plus icon at the bottom left, labeled 'Add new Decal Template button'. A green arrow points to the 'Load', 'Save', and 'Clear' buttons, labeled 'Presets control buttons'. A pink box highlights the grid of decal thumbnails, labeled 'Current preset of decals'. The label 'Placement modes' points to the top bar icons.

Decal placement tool

DECAL PLACEMENT TOOL

Placement modes

Parameters

Presets control buttons

Current preset of decals

Add new Decal Template button

# GPU Raycast

## What is it?

GPU Raycast provide object raycasting without colliders.

## GPU Raycast Hit Info contains:

Vector3 position	World Position of hit point
Vector3 normal	World Normal of hit point
Renderer hitRenderer	Renderer which was hit
Int VertexIndex	Index of vertex which was hit
DecalsTarget hitTarget	DecalsTarget component of object which was hit (may be parent relative to hitRenderer)
Bool IsHit	Is any object was hit

## GPU Raycast class:

Bool RaycastToRegisteredTargets (Camera camera, Vector2 uv, **out** GPURaycastDecalsTargetInfo hitInfo) - raycasts to registered targets from one camera and special screen space uv coordinates (may be normalized mouse position).

Bool RaycastToRegisteredTargets (Camera camera, **float** pixelX, **float** pixelY, **out** GPURaycastDecalsTargetInfo hitInfo) - raycasts to registered targets from one camera and special screen space pixel coordinates.

Bool RaycastToRegisteredTargets (Camera camera, **out** GPURaycastDecalsTargetInfo hitInfo) - raycasts to registered targets from one camera in mouse position.

Returns hitInfo.IsHit.

Parameters:

Camera camera	From which camera you want raycast
Vector2 uv	Normalized screen space position
<b>out</b> GPURaycastDecalsTargetInfo hitInfo	Hit info result

Camera camera	From which camera you want raycast
<b>float</b> pixelX	Screen space pixel position X
<b>float</b> pixelY	Screen space pixel position Y
<b>out</b> GPURaycastDecalsTargetInfo hitInfo	Hit info result

## Example

1. Open Knife/Decal Master/Samples/Scenes/GPURaycastRealtime scene.
2. Enter playmode
3. Click mouse by boxes and spheres

As you can see all boxes and sphere do not have colliders but we can hit them and spawn decals.

## How to use

1. Your object must have Decals Target component (it needs to register renderer for Raycasting)
2. Your fire script (in example scene it is DecalsTargetsRaycastTest object and component) must use `GPURaycast.RaycastToRegisteredTargets` function to hit and must send hit data to targets components
3. In examples scene our targets components are SimpleDecalReceiver and interface IDecalReceiver
  - a. Also, you can find SkinnedMeshDecalReceiver component, it can spawn decals on skinnedmeshrenderers and parent these decals to bones.

# Decal Spawn API

## What is it?

This API provides easy decals spawn at runtime.

## How to use?

1. You need decide on which events you want spawn decals.
  - a. On Physics RaycastHit (common shooting method)  
For this you need attach colliders on every object on which you want spawn decals.
  - b. On Collision (common projectile)  
For this you need attach colliders on every object on which you want spawn decals. (same as for RaycastHit). Also, you need:
    - i. Attach SpawnDecalOnCollision to Projectile's Collider
  - c. Select SpawnControl (by receiver or by spawn controller)  
SpawnControl.ByReceiver – all collision events will be send to receiver  
SpawnControl.BySpawnController – all collision events will be handled by this script and this script will spawn its own decals.  
On GPU RaycastHit (Decal Master GPURaycast shooting method)  
For this you don't need attach colliders, but you need attach register component DecalsTarget on every object, or on parent object.
  - d. On Particle CollisionEvent (in places which particle was collided)  
For this you need attach colliders on every object on which you want spawn decals. (same as for RaycastHit). Also, you need:
    - i. Attach SpawnDecalOnParticle to ParticleSystem
    - ii. Select SpawnControl (by receiver or by spawn controller)  
SpawnControl.ByReceiver – all collision events will be send to receiver  
SpawnControl.BySpawnController – all collision events will be handled by this script and this script will spawn its own decals.
    - iii. Enable Collision module in ParticleSystem
    - iv. Select World Collision Type in Collision Module
    - v. Enable "Send Collision Messages" in Collision Module
2. You need attach SimpleDecalReceiver on every object or on parent object. And select SpawnMode you want use in SpawnController.
  - a. Instantiate – generic method to clone prefabs and objects. May be cause of performance impact if you spawn a large number of decals.  
If you select this method, you should select DecalPrefab from project (or instance from scene)
  - b. Pool – prewarm method. You can prewarm (pre instantiate) many objects (or prefabs).  
Enable and Disable them when you need Spawn and Destroy decal instead of classic Object.Instantiate and Object.Destroy methods.  
If you select this method, you should select DecalSpawner from scene (our implementation is SimpleDecalSpawner component). To setup SimpleDecalSpawner you need:
    - i. Create empty GameObject and attach SimpleDecalSpawner component to it.
    - ii. Select prewarm Prefab



- iii. Enable PopulateOnAwake
  - iv. Set prewarm count (count of prefab instances)
  - v. Enable or disable AutoPopulateOnZero (every time when pool empty, it will populate automatically)
3. Update your scripts and GameObjects
- a. If you are using Physics Raycast you need use method `DecalReceiverHelper.SendPhysicsRaycastInfo(hitInfo)`  
hitInfo – common hit result info of PhysicsRaycast (RaycastHit)
  - b. If you are using GPU Raycast you need use method `DecalReceiverHelper.SendGPURaycastInfo(hitInfo)`  
hitInfo – common hit result info of GPURaycast (GPURaycastDecalsTargetInfo)
  - c. If you are using projectiles with Collision handling you need attach `SpawnDecalOnCollision` to projectile's collider. You may leave default settings of this script, or may setup for your needs.  
Or you can implement it by yourself. You should call `DecalReceiverHelper.SendCollision(collision)` method or spawn decals manually by contact points or something else.
  - d. If you are using particles with Collision handling you need attach `SpawnDecalOnParticle` to `ParticleSystem`. You may leave default settings of this script, or may setup for your needs.  
Or you can implement it by yourself. You should call `DecalReceiverHelper.SendParticleCollisionEvent(collisionEvent)` method or spawn decals manually by collision events or something else.

## Knife.DeferredDecals.Spawn.DecalReceiverHelper

Static helper class for messages sending

<code>public static void SendPhysicsRaycastInfo(RaycastHit hitInfo)</code>	Send RaycastHit hitInfo to IDecalReceiver component on hitInfo.collider or to parent component with IDecalReceiver
<code>public static void SendParticleCollisionEvent(ParticleCollisionEvent collisionEvent)</code>	Send ParticleCollisionEvent collisionEvent to IDecalReceiver component on collisionEvent.colliderComponent or to parent component with IDecalReceiver
<code>public static void SendGPURaycastInfo(GPURaycastDecalsTargetInfo hitInfo)</code>	Send GPURaycastDecalsTargetInfo hitInfo to IDecalReceiver component on hitInfo.hittedRenderer or to parent component with IDecalReceiver
<code>public static void SendCollision(Collision collision)</code>	Send Collision collision to IDecalReceiver component on collision.collider or to parent component with IDecalReceiver

## Knife.DeferredDecals.Spawn.DecalSpawnController

Common class to spawn Decals from pool or with Instantiate function.

<code>public</code> SpawnType SpawnMode	SpawnMode controls how decals will be spawned, from pool or with Instantiate function.
<code>public</code> Decal DecalPrefab	This prefab will be used to spawn instances. Used only if SpawnMode.Instantiate selected.
<code>public</code> DecalSpawner Spawner	This spawner will be used to spawn instances. Used only if SpawnMode.Pool selected.
<code>public float</code> DestroyDelay	After that time decal will be destroyed or removed to pool. If value is below zero, auto destroy will be disabled.
<code>public</code> Decal SpawnDecal()	Spawn decal instance, activate GameObject and return decal instance.

## Knife.DeferredDecals.Spawn.DecalSpawner

Abstract class to spawn decals. Derived from MonoBehaviour.

<code>public abstract</code> Decal Spawn()	Abstract decal spawn method.
<code>public abstract void</code> DestroyDecal(Decal decal)	Abstract destroy decal method.
<code>public void</code> DestroyDecal(Decal decal, float delay)	Starts coroutine that will call DestroyDecal(decal) with delay.

## Knife.DeferredDecals.Spawn.IDecalReceiver

Interface which can get messages with different info (RaycastHit, GPURaycastDecalsTargetInfo, Collision, ParticleCollisionEvent).

<code>void</code> HittedByGPURaycast(GPURaycastDecalsTargetInfo hitInfo)	Get message with GPURaycastDecalsTargetInfo hitInfo method.
<code>void</code> HittedByPhysicsRaycast(RaycastHit hitInfo)	Get message with RaycastHit hitInfo method.
<code>void</code> HittedByPhysicsCollision(Collision collision)	Get message with Collision info method.
<code>void</code> HittedByParticle(ParticleCollisionEvent collisionEvent)	Get message with ParticleCollisionEvent method.

## Knife.DeferredDecals.Spawn.ISpawner<T>

Interface which can spawn T, where T is MonoBehaviour (or inherited classes).

<code>T</code> Spawn()	Spawn T object.
------------------------	-----------------

## Knife.DeferredDecals.Spawn.IOnePrefabPool<T>

Interface which can spawn T instance, destroy T instance, and will be populated with T instances, where T is MonoBehaviour (or inherited classes), Derived from ISpawner.

<code>void</code> Populate(T prefab)	Populates pool by T prefab.
<code>void</code> Destroy(T instance)	Remove T instance to pool.

## Knife.DeferredDecals.Spawn.OnePrefabDecalPool

Implementation class of IOnePrefabPool<Decal>

<code>public</code> OnePrefabDecalPool( <code>int</code> prewarmCount, <code>bool</code> autoPopulateOnZero)	Constructor. prewarmCount – pool cached size (instancs count). autoPopulateOnZero – repopulate pool if instances will be zero on Spawn() call.
<code>public void</code> Populate(Decal prefab)	Populate pool with prefab instances.
<code>public</code> Decal Spawn()	Spawn decal from pool.
<code>public void</code> Destroy(Decal instance)	Remove decal to pool.

## Knife.DeferredDecals.Spawn.SimpleDecalReceiver

Implementation class of IDecalReceiver.

<code>public</code> DecalSpawnController SpawnController	SpawnController for decal spawn.
<code>public bool</code> ParentOnHit	Auto parent decal to hitted object(renderer or collider) when object get message about hit.
<code>public float</code> Offset	Normal offset for spawned decal.

## Knife.DeferredDecals.Spawn.SimpleDecalSpawner

Implementation class of abstract class DecalsSpawner.

<code>public</code> Decal Prefab	Prefab that will be used for pool populating.
<code>public bool</code> PopulateOnAwake	Auto populate pool on Awake call.
<code>public int</code> PrewarmCount	Instances count to populate.
<code>public bool</code> AutoPopulateOnZero	Repopulate pool if instances will be zero on Spawn() call.
<code>public override</code> Decal Spawn()	Spawn decal from pool.
<code>public override void</code> DestroyDecal(Decal decal)	Remove decal to pool.

## Knife.DeferredDecals.Spawn.SkinnedMeshDecalReceiver

Class to control spawned decals on skinned mesh renderers. Derived from SimpleDecalReceiver. Has overridden method `void OnHitted(Decal decalInstance, GPURaycastDecalsTargetInfo hitInfo)` when decal will be spawned, it will be parented to skinnedmeshrenderer bone (with maximum weight) of hitted vertex.

## Knife.DeferredDecals.Spawn.SpawnDecalOnCollision

Class for spawning decals on collision.

<code>public SpawnControl SpawnControl</code>	This flag controls who will handle messages (receiver or this script). SpawnControl.ByReceiver – all collision events will be send to receiver SpawnControl.BySpawnController – all collision events will be handled by this script and this script will spawn its own decals.
<code>public DecalSpawnController SpawnController</code>	SpawnController that will be used to spawn decals if SpawnControl is BySpawnController .
<code>public float Offset</code>	Normal offset for spawned decals.
<code>public bool ParentOnCollide</code>	Parent spawned decals to collider.
<code>public bool SpawnOnCollisionEnter</code>	This flag controls if decal will send messages (or spawn decal) on CollisionEnter event.
<code>public bool SpawnOnCollisionStay</code>	This flag controls if decal will send messages (or spawn decal) on CollisionStay event.
<code>public bool SpawnOnCollisionExit</code>	This flag controls if decal will send messages (or spawn decal) on CollisionExit event.
<code>public string TagCheck</code>	All collision will be check tag of collided object before spawn. And script will send messages (or spawn decal) if tags are equals. If null or empty tag checks will be disabled.

## Knife.DeferredDecals.Spawn.SpawnDecalOnParticle

Class for spawning decals on particle collision.

<code>public SpawnControl SpawnControl</code>	This flag controls who will handle messages (receiver or this script). SpawnControl.ByReceiver – all collision events will be send to receiver SpawnControl.BySpawnController – all collision events will be handled by this script and this script will spawn its own decals.
<code>public DecalSpawnController SpawnController</code>	SpawnController that will be used to spawn decals if SpawnControl is BySpawnController .
<code>public float Offset</code>	Normal offset for spawned decals.
<code>public bool ParentOnCollide</code>	Parent spawned decals to collider.