# VertexDirt
# Manual for Alpha 1 version

## Overview
VertexDirt is a Unity3D plugin for baking indirect lighting to vertex colors. It comes with simple and easy to use editor tools and with fully documented source code. VertexDirt is compatible with both versions of Unity. *Please note that vertex colors only visible on mesh if you use a shader that supports them. If you do not have any, you can check the "VD Vertex Color" shader family in your shaders.*

## VertexDirt baking tools
Select one of the VD tools from the 'Tools/VertexDirt' menu in the menubar. Alpha 1 comes with the Ambient Occlusion tool. Further versions will introduce more baking tools.

## Ambient Occlusion
This is the basic baking tool. Ambient Occlusion is a way to visualize indirect lighting and shadows.

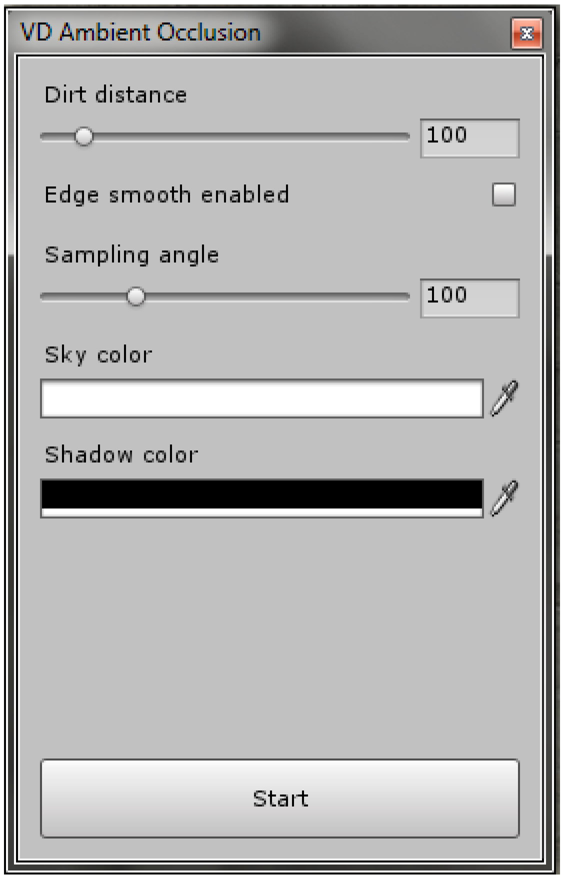
*Dirt distance*
The distance in units after the geometry clipped. Use short distances in interiors and / or if you want sharp, dark corners.

*Edge smooth enabled*
Edge smooth averages the normals of vertices in the same position during baking, and produce smoother results.

*Sampling angle*
Wider angles results darker corners. Narrow angles may need for long dirt distances in interiours.

*Sky color*
Color of the sky. This color is used when the vertex completely unoccluded.

*Shadow color*
This color is used when the vertex completely occluded.
If a vertex partially occluded it will get a color mixed from the sky and shadow colors.



Dirt distance 100
Edge smooth enabled ☐
Sampling angle 100
Sky color
Shadow color

## Usage

Select some object, set the desired parameters and hit the Start button.
The baking time depends on the complexity of the selected objects. Baking time not affected by unselected objects, but every visible object fill affect the results of baking. Baked object automatically gets a VDColorHandler component. This component stores the baked colors and handle mesh instancing.

## Under the hood

During the baking VertexDirt create a sampler camera. You can set the camera near and far clipping planes, fov and background color/cubemap through VD static variables (see scripting reference). Also, a replacement shader taken account. This shader repaint all object in the scene in a way that we want to capture to the vertex colors. For example, for ambient occlusion using a simple unlit color shader which is black in default and color the camera background to white. After this setup, the camera renders a small texture positioned to every vertex rotated in the normal vectors direction. This texture then downsampled to get the final color of each vertex.

The generated colors then storen in VDColorHandler components in each GameObject. If you want to write your own VDColorHandler script, you can extends it from the VDColorHandlerBase class. If you do so, just modify the colorHandlerClass variable under VertexDirt, so it will put your own component to the objects after the bake.

**Please note, VertexDirt is under development. Alpha-1 is the first public version, mostly for testing purpose. If you like this project please write a review for describing bugs or share me new ideas.**