Using Maskshaders for Blending Ground Tiles

In tile based computer games, ground textures are often just linearly blended. This can result in unnatural looking transitions, e.g. when half transparent stones overlay grass, letting the green grass shine through the usually solid rock. With the use of Maskshaders, the transparency of textures can be controlled per-pixel, thus making more or less of the texture transparent instead of making the whole texture more or less shine-through.

Introduction

When blending ground textures only by linearly adapting alpha, unnatural looking transitions like the one depicted below are the norm.

Fig. 1: Transitions only with alpha

What we want is a way to increase aesthetics of transitions from one material to another, respecting its natural components and their opacity, preferably like this:

Fig. 2: Natural transitions

Theory

For this, in addition to the colour texture, a material needs sort of a mask that tells us at what level to hide or show a certain pixel. With this, it's possible to control the transparency per-pixel (fragment) instead of just per-vertex. One straight-forward approach is to take the red channel fragment value of a second image, called mask, as mask value.

Furthermore, it's necessary to have the possibility to set a mask value threshold, below which fragments will not be rendered. When deriving the threshold from interpolated vertex alpha, it can easily be stored in mesh geometry.

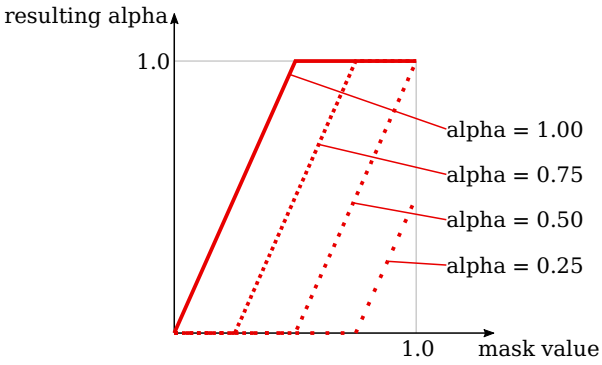

Fig. 3: Alpha curves for different alpha values

Read: The higher the alpha, the more visible fragments.

Note: Threshold must decrease for increased alpha to follow the logic that higher alpha values result in more visible fragments. To be exact, it is $Threshold = 1 - alpha$.

Finally, it would be handy to have a sharpness, that defines how quickly fragments will fade in. This fade length, over which fragments will linearly blend from zero visibility to full opacity, most likely stays the same for one material. Therefore it makes sense to store this value in the shader itself.

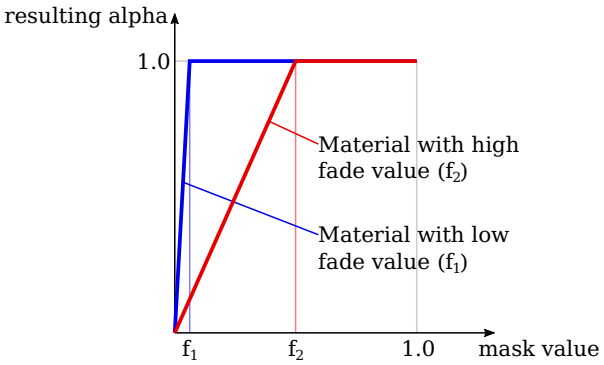

Fig. 4: Alpha curves for different fade lengths

Note: When threshold is zero, the alpha curve always goes through the origin and reaches full alpha just at the alpha fade length value. This means, in order to be able to make a texture fully opaque, the lowest mask value must be higher than the fade length value.

To obtain a mask, so far the only way is to hand-craft it depending on the texture and adjust it to our needs according to the material. But maybe someday we'll see algorithms doing this work – just taking a texture image, analysing it and masking different features randomly.

GLSL Shaders

To keep it simple, the GLSL shaders with Cerberus X sources and -examples (all licensed under CC0), are accessible under following location:

https://cerberus-x.com/community/threads/holzchopf-mojo2maskshaders-image-masking-made-easy.374/

Application

The table below shows the RedMaskShader applied to two textures with corresponding masks for different alpha values (alpha set for the whole quad).

Texture	Mask	Fade length	Alpha				
			0.2	0.4	0.6	0.8	1.0
		0.500					
		0.001