Vertigons

Quake 3 Surfacesprites for Soldier of Fortune 2 and Jedi Knight 2

Usage Guide

By Pat Lipo

September 13, 2025

Vertigons are a shader pass similar to those you are already familiar with. It shares kinship with surface textures, lightmaps, and other multipass texture layers. The main difference is that surfacesprites are not flat on the surface they are placed on, but rather stand up away from the surface. Another important distinction is that a single surfacesprite pass typically doesn't generate one single image, but rather multiple images evenly distributed over that surface. They usually stick vertically off the surface, hence the term vertigons.

Example:
Take any shader:

```
textures/prague/cobblestone
{
    {
        map $lightmap
    }
    {
        map textures/prague/cobblestone
        blendFunc GL_DST_COLOR GL_SRC_COLOR
    }
```

…and add a pass to it like this:

```
    {
        map textures/colombia/grass_clump
        surfaceSprites vertical 24 24 16 1024
        ssWind 0.5
        alphaFunc GE192
        blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
        depthWrite
    }
}
```

As in any shader pass, the map is the graphic used, but in this case the graphic is not laid on top of the surface, it instead is distributed over it. The alpha and blend properties are optional as with any shader, but the above configuration is highly recommended for "solid" materials. Surfacesprites will not be placed unless the polygon is facing more-or-less upward (with a couple of exceptions below), so if cube room is textured with a surfacesprite shader, the stuff will only appear on the floor.

Main Command Syntax:

surfaceSprites <type> <width> <height> <density> <fadedist>

<type>: One of three keywords are valid:

vertical: Stuff that sticks up off the ground, such as grass, reeds, etc.

oriented: An image that always faces the user. This can be used for rocks, debris, whatever you want to break up a surface. These should stay fairly small if possible.

effect: An image that always faces the user, but it grows and fades out over time. When the effect fades completely out, it reappears in another location.

<width>: Horizontal size of each individual sprite, in world dimensions. This is the *base* width, which can be added to with the variance command below.

<height>: Vertical size of each individual sprite, in world dimensions. This is the *base* height, which can be added to with the variance command below.

<density>: Density defines the interval at which the sprites are placed, e.g. a 32 means that a sprite will be placed every 32 units on the ground. Lower equals denser. Note that the more dense the sprites are, the more slowdown you will see, so you should lower the fadedist if you up the density.

<fadedist>: Fade distance defines the **minimum** distance at which the surfacesprites will be drawn in full. At a distance greater than this, sprites are visible, but various sprites start to fade out at different distances. 1000 is a fair draw distance for normal grass-like items, smaller numbers are more appropriate for smaller sprites.

Optional Parameters:

The Surfacesprites have several optional parameters that, unless overridden, use default values. These are entered as separate lines beginning with "ss", after the initial surfacesprite call:

ssFademax <fademax>

Fade max defines the **maximum** distance at which any Surfacesprites at all will be drawn at all. The sprites will fade out between the fadedist and the fademax. This value defaults to 1.33 times the fadedist.

ssFadescale <fadescale>

Fade scale indicates a fractional amount of the sprite's width that it will scale between the fadedist and fademax. That is, if:

Surfacesprites vertical 8.0 8.0 32 1000
SsFademax 1500
SsFadescale 1.0

…then between distances of 1000 and 1500, this sprite of base width 8 will grow from 8 to 16. Of course after 1500, the sprites are no longer drawn (see fademax). Default for fadescale is zero.

ssVariance <varwidth> <varheight>

Variance defines a factored amount to add to the width and height randomly. That is, if the base width is 8 and the varwidth is 1.0, the width will be randomly between 8 and 16. If the variance is not specified, they default to zero (e.g. every sprite is the same width and height).

<varwidth>: Fraction of the base width of the sprite to add randomly. Default is zero.

<varheight>: Fraction of the base height of the sprite to add randomly. Default is zero.

ssHangdown

Hangdown is an optional keyword that when set, the sprites are placed on the ceiling instead of on the ground, such as for hanging vines.

ssFaceup

Oriented and Effect sprites only

Faceup is an optional keyword for oriented and effect sprites. When set, the sprites are placed horizontally on the ground. Currently the surface must be completely horizontal, or the sprites won't appear. Used for stains on the ground, or water rings. Height is ignored on faceup sprites, they are always square. Not valid for vertical sprites.

ssWind <wind>

Vertical and Effect sprites only

Wind is an optional variable for vertical and effect sprites. Vertical sprites (such as grass) can wave according to wind settings (see below). Without this variable, the sprites are not affected by wind.

<wind>: The wind value is how much to factor in these effects, as a multiple of the effects of wind. 1.0 would be for limp grass blowing in the wind. 0.5 would be more rigid. 0.0 means the sprite sits rigidly upward. A rigid sprite is cheaper to render, so leave the wind variable out if you don't need it! NOTE: Currently not implemented on effect sprites. Default is zero.

ssWindidle <windidle>
Vertical sprites only

The wind idle value is a factor from 0 to 1.0. The higher this number is, the more vertical sprites will sway when there is no wind.

<windidle>: The windidle value is how much to factor in these effects, and typically varies from 0 to 1. 1.0 is limp grass swaying when there is no wind. 0.25 offers very subtle movement. 0.0 means the sprite sits rigidly upward when there is no wind. A rigid sprite is cheaper to render, so leave windidle at zero if you do not need it. This value defaults to zero, or to the wind value if it is set.

ssVertSkew <skew>
Vertical sprites only

The vertical skew value is a random factor that is added to the surfacesprites to make their "base" position something other than vertical. The best use for this is for example a thin graphic for bamboo, which might stick up at strange angles.

<skew>: This value indicates the fraction of the sprites height to add randomly to the sprites' top point. For example, if a sprite has a height of 16, and the skew value is 1.0, then the sprite may lean between 16 to the left and 16 to the right. In exact terminology, the X and Y coordinates of the top of the sprite could be between a -16 and 16 offset. This value defaults to zero.

ssFXDuration <duration>
Effect sprites only

The duration value is a length of time that an effect sprite grows or shrinks before it disappears. This is how long the effect lasts before looping, *in milliseconds*. That is, 1000 equals 1 second. Duration defaults to 1000 (1 second).

ssFXGrow <growwidth> <growheight>
Effect sprites only

The grow value is a proportional value that indicates how much of the base width and height changes over an effects lifetime (i.e. its duration). A base width of 8 and a grow value of 1.0 means the effect grows from 8 to 16. A negative value means it shrinks, so a sprite with a base width of 8 (or any base width) and a grow value of –1.0, will shrink to nothing over its duration. The default for growwidth and growheight are 0.0.

ssFXAlphaRange <alphastart> <alphaend>
Effect sprites only

The effect alpha range sets two values between 0 and 1, that indicates what amount to fade out to over the course of the effect's lifetime. That is, if the alpha range is 0.8 and 0.3, then an effect surfacesprite will start out with an alpha of 0.8 and end with an alpha of 0.3.

<alphastart>: This indicates the alpha value that is shown at the start of each individual surfacesprite effect. The alpha transparency of a sprite will smoothly go from this value to the alphaend value. The default for the alphastart is 1.0, or opaque.

<alphaend>: This indicates the alpha value that is shown at the end of an individual surfacesprite effect. The alpha transparency of a sprite will smoothly go from the alphaend value to this value. The default for the alphaend is 0.0, or fully transparent.

ssFXWeather

Effect sprites only

Weather is an optional keyword for effect sprites. If this keyword is present, then the density of the sprites is reliant on the **r_surfaceWeather** cvar as detailed below. If r_surfaceWeather is zero, then the given sprites are keyed off the weather system, and will appear only if the weather system is currently showing snow or rain.

There are a few cvars that this feature uses right now:
r_surfaceSprites 0: Turns surfacesprites off.
r_surfaceSprites 1: Turns surfacesprites on.
r_surfaceSprites 3: Reports how many are being drawn per frame

r_windSpeed <speed>: Sets the windspeed. Defaults to 0. This value is ignored if the weather system is present, as the weather wind will take control.
r_windAngle <angle>: Sets the angle that the wind blows at. Defaults to 0. This value is ignored if the weather system is present, as the weather wind will take control.
r_windDampFactor <factor>: Sets how quickly the wind responds to changes. Defaults to 0.1
r_windGust: Sets the minimum number of seconds between wind gusts. Ignored if a weather system is present.

r_surfaceWeather <value>: Sets the amount of weather-oriented effects to display. Defaults to 1.0. This value is typically 0 to 1, representing 0-100% of the weather, so in the case of rain splashes, 0.4 is a light rain.

r_windPointForce <value>:
r_windPointX <value>:
r_windPointY <value>:
Call it a failed experiment. If r_windPointForce is given a non-zero value, additional wind is generated from a central point defined by the windpoint X and Y. We didn't use it because it didn't give pleasing results.

You should also add a few elements to make the surface light properly:
First you generally should not use lightmaps, the surface should be instead vertex lit, to match the lighting of the grass.

```
q3map_nolightmap
q3map_onlyvertexlighting
```

If the polygon surfaces are large, you will not see the sprites fade properly. You might want to force the BSP to break up the surface with a command like this:

```
q3map_tesssize          256
```

The sprites or graphics that it uses are hard-alpha channeled images of reasonable resolution. They should have sides (e.g. not tile). Grass is done with "clump" images. A wide, flat edge at the bottom of the sprite is not usually desirable. It is highly recommended that the sprites have the same overall coloring as the ground texture.

The system currently randomly flips any surfacesprites horizontally for variety.

You can use any blend mode you wish, so if you want the sprites to be additive, simply set **blendfunc GL_ONE GL_ONE**. Beware of highly transparent objects if the sprites are very tall (e.g. vertical surfacesprites such as tall grass). The transparency will reveal seams if you are not careful. So for tall "in your face" sprites such as grass, use a hard-edged alpha mask and an alphatest setting such as **alphafunc GE128**, or even crisper like **alphafunc GE192** (a new

alphafunc recently added). However, there is a tradeoff. When the alphafunc is set higher, the sprites will not be able to fade as smoothly, and will appear to "pop" in a bit more.

If you want additional sprites in a single texture, you can add in another pass, but don't abuse it or your overtaxed machine will punish you. If you use the exact same density on a second pass, the second set of sprites will be directly on top of the first, which can be useful. If you want roughly the same density but don't wish for this, simply add or subtract a small amount from the density, the overall effect will be the same. BE CAREFUL WITH THIS SYSTEM, BECAUSE THE SPRITES AIN'T FREE!

If you want to have a texture covered with sprites in some places, not in others (such as raindrops outdoors but not inside), two shaders should be made with the same graphic, but one not possessing the surfacesprite pass. I'd suggest a clever naming convention for this.

Shader examples:

Grasses that blow in the wind:

```
textures/test/grnd01c_sprites
{
        qer_editorimage       textures/kamchatka/grnd01c
        q3map_material Gravel
        q3map_nolightmap
        q3map_onlyvertexlighting
    {
        map textures/kamchatka/grnd01c
          blendFunc GL_DST_COLOR GL_ZERO
    }

    {
        map models/objects/colombia/jungle/tall_grass
        surfaceSprites vertical 16 32 48 1000
            ssVariance 1.0 2.0
            ssWind 1.0
        blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
        alphaFunc GE128
        depthWrite
    }

}
```

Rain splashes on water:

```
textures/colombia/rain_water
{
        qer_editorimage       textures/colombia/water_terrain
        q3map_material Water
        q3map_nolightmap
        q3map_onlyvertexlighting
        q3map_tesssize 256
    {
        map textures/colombia/water_terrain
        blendFunc GL_ONE GL_ONE_MINUS_SRC_ALPHA
        depthWrite
        rgbGen exactVertex
        alphaGen const 0.9
        tcMod turb 0 0.08 0.04 0.08
        tcMod scroll -0.05 -0.001
        tcMod scale 3 3
    }
    {
        map textures/colombia/water_terrain2
        blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
```

```
        rgbGen exactVertex
        tcMod turb 0 0.08 0.04 0.08
        tcMod scale 3 3
    }

    {
        map gfx/sprites/rainhit
        surfaceSprites effect 2.5 2.5 16 400
            ssFXDuration 135
            ssFXGrow 6 6
            ssVariance 1.0 0.75
            ssFXWeather
        blendFunc GL_ONE GL_ONE
    }
    {
        map gfx/sprites/rainring
        surfaceSprites effect 2.0 2.0 28 350
            ssFXDuration 220
            ssFXGrow 2.5 2.5
            ssFaceup
            ssVariance 2.0 1.0
            ssWeather
        blendFunc GL_ONE GL_ONE
    }
}
```

Random debris on the ground:

```
textures/common/ground
{
        qer_editorimage        textures/colombia/ground_jungle
        q3map_material gravel
        q3map_nolightmap
        q3map_onlyvertexlighting
    {
        map textures/colombia/ground_jungle
        blendFunc GL_ONE GL_ZERO
    }
    {
        map gfx/sprites/rock1
        surfaceSprites oriented 4 4 80 256
            ssVariance 0.25 0.0
        blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
        alphaFunc GE128
        depthWrite
    }
    {
        map gfx/sprites/rock2
        surfaceSprites oriented 2 2 92 200
            ssVariance 0.25 0.0
        blendFunc GL_SRC_ALPHA GL_ONE_MINUS_SRC_ALPHA
        alphaFunc GE128
        depthWrite
    }
}
```