Skip to content

Freestyle Shaders (freestyle.shaders)

This module contains stroke shaders used for creation of stylized strokes. It is also intended to be a collection of examples for shader definition in Pythor

User-defined stroke shaders inherit the `freestyle.types.StrokeShader` class.

class freestyle.shaders.**BackboneStretcherShader**

Class hierarchy: `freestyle.types.StrokeShader` > `BackboneStretcherShader`

[Geometry shader]

__init__(amount=2.0)

Builds a BackboneStretcherShader object.

PARAMETERS:

amount (*float*) – The stretching amount value.

shade(stroke)

Stretches the stroke at its two extremities and following the respective directions: v(1)v(0) and v(n-1)v(n).

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**BezierCurveShader**

Class hierarchy: `freestyle.types.StrokeShader` > `BezierCurveShader`

[Geometry shader]

__init__(error=4.0)

Builds a BezierCurveShader object.

PARAMETERS:

error (*float*) – The error we're allowing for the approximation. This error is the max distance allowed between the new curve and the original geometry.

shade(stroke)

Transforms the stroke backbone geometry so that it corresponds to a Bezier Curve approximation of the original backbone geometry.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**BlenderTextureShader**

Class hierarchy: `freestyle.types.StrokeShader` > `BlenderTextureShader`

[Texture shader]

__init__(texture)

Builds a BlenderTextureShader object.

PARAMETERS:

texture (`bpy.types.LineStyleTextureSlot` or `bpy.types.ShaderNodeTree`) – A line style texture slot or a shader node tree to define a set of textures.

shade(stroke)

Assigns a blender texture slot to the stroke shading in order to simulate marks.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**CalligraphicShader**

Class hierarchy: `freestyle.types.StrokeShader` > `CalligraphicShader`

[Thickness Shader]

__init__(thickness_min, thickness_max, orientation, clamp)

Builds a CalligraphicShader object.

PARAMETERS:

- **thickness_min** (*float*) – The minimum thickness in the direction perpendicular to the main direction.
- **thickness_max** (*float*) – The maximum thickness in the main direction.
- **orientation** (`mathutils.Vector`) – The 2D vector giving the main direction.
- **clamp** (*bool*) – If true, the strokes are drawn in black when the stroke direction is between -90 and 90 degrees with respect to the main direction and drawn in white otherwise. If false, the strokes are always drawn in black.

shade(stroke)

Assigns thicknesses to the stroke vertices so that the stroke looks like made with a calligraphic tool, i.e. the stroke will be the thickest in a main direction, and the thinnest in the direction perpendicular to this one, and an interpolation in between.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**ColorNoiseShader**

Class hierarchy: `freestyle.types.StrokeShader` > `ColorNoiseShader`

[Color shader]

__init__(amplitude, period)

Builds a ColorNoiseShader object.

PARAMETERS:

- **amplitude** (*float*) – The amplitude of the noise signal.
- **period** (*float*) – The period of the noise signal.

shade(stroke)

Shader to add noise to the stroke colors.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**ConstantColorShader**

Class hierarchy: `freestyle.types.StrokeShader` > `ConstantColorShader`

[Color shader]

__init__(red, green, blue, alpha=1.0)

Builds a ConstantColorShader object.

PARAMETERS:

- **red** (*float*) – The red component.
- **green** (*float*) – The green component.
- **blue** (*float*) – The blue component.
- **alpha** (*float*) – The alpha value.

shade(stroke)

Assigns a constant color to every vertex of the Stroke.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**ConstantThicknessShader**

Class hierarchy: `freestyle.types.StrokeShader` > `ConstantThicknessShader`

[Thickness shader]

__init__(thickness)

Builds a ConstantThicknessShader object.

PARAMETERS:

thickness (*float*) – The thickness that must be assigned to the stroke.

shade(stroke)

Assigns an absolute constant thickness to every vertex of the Stroke.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**ConstrainedIncreasingThicknessShader**

Class hierarchy: `freestyle.types.StrokeShader` > `ConstrainedIncreasingThicknessShader`

[Thickness shader]

__init__(thickness_min, thickness_max, ratio)

Builds a ConstrainedIncreasingThicknessShader object.

PARAMETERS:

- **thickness_min** (*float*) – The minimum thickness.
- **thickness_max** (*float*) – The maximum thickness.
- **ratio** (*float*) – The thickness/length ratio that we don't want to exceed.

shade(stroke)

Same as the `IncreasingThicknessShader` , but here we allow the user to control the thickness/length ratio so that we don't get fa short lines.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**GuidingLinesShader**

Class hierarchy: `freestyle.types.StrokeShader` > `GuidingLinesShader`

[Geometry shader]

__init__(offset)

Builds a GuidingLinesShader object.

PARAMETERS:

offset (*float*) – The line that replaces the stroke is initially in the middle of the initial stroke bounding box. offset is the value of the displacement which is applied to this line along its normal.

shade(stroke)

Shader to modify the Stroke geometry so that it corresponds to its main direction line. This shader must be used together with the splitting operator using the curvature criterion. Indeed, the precision of the approximation will depend on the size of the stroke's pieces. The bigger the pieces are, the rougher the approximation is.

PARAMETERS:

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**IncreasingColorShader**

Class hierarchy: `freestyle.types.StrokeShader` > `IncreasingColorShader`

[Color shader]

__init__(red_min, green_min, blue_min, alpha_min, red_max, green_max, blue_max, alpha_max)

Builds an IncreasingColorShader object.

PARAMETERS:

- **red_min** (*float*) – The first color red component.
- **green_min** (*float*) – The first color green component.
- **blue_min** (*float*) – The first color blue component.
- **alpha_min** (*float*) – The first color alpha value.
- **red_max** (*float*) – The second color red component.
- **green_max** (*float*) – The second color green component.
- **blue_max** (*float*) – The second color blue component.
- **alpha_max** (*float*) – The second color alpha value.

shade(stroke)

Assigns a varying color to the stroke. The user specifies two colors A and B. The stroke color will change linearly from A to B between the fi and the last vertex.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**IncreasingThicknessShader**

Class hierarchy: `freestyle.types.StrokeShader` > `IncreasingThicknessShader`

[Thickness shader]

__init__(thickness_A, thickness_B)

Builds an IncreasingThicknessShader object.

PARAMETERS:

- **thickness_A** (*float*) – The first thickness value.
- **thickness_B** (*float*) – The second thickness value.

shade(stroke)

Assigns thicknesses values such as the thickness increases from a thickness value A to a thickness value B between the first vertex to the midpoint vertex and then decreases from B to a A between this midpoint vertex and the last vertex. The thickness is linearly interpolated from to B.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**PolygonalizationShader**

Class hierarchy: `freestyle.types.StrokeShader` > `PolygonalizationShader`

[Geometry shader]

__init__(error)

Builds a PolygonalizationShader object.

PARAMETERS:

error (*float*) – The error we want our polygonal approximation to have with respect to the original geometry. The smaller, the closer the new stroke is to the original one. This error corresponds to the maximum distance between the new stroke and the old one.

shade(stroke)

Modifies the Stroke geometry so that it looks more "polygonal". The basic idea is to start from the minimal stroke approximation consisting in line joining the first vertex to the last one and to subdivide using the original stroke vertices until a certain error is reached.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**RoundCapShader**

round_cap_thickness(x)

shade(stroke)

class freestyle.shaders.**SamplingShader**

Class hierarchy: `freestyle.types.StrokeShader` > `SamplingShader`

[Geometry shader]

__init__(sampling)

Builds a SamplingShader object.

PARAMETERS:

sampling (*float*) – The sampling to use for the stroke resampling.

shade(stroke)

Resamples the stroke.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**SmoothingShader**

Class hierarchy: `freestyle.types.StrokeShader` > `SmoothingShader`

[Geometry shader]

__init__(num_iterations=100, factor_point=0.1, factor_curvature=0.0, factor_curvature_difference=0.2, aniso_point=0.0, aniso_normal=0.0, aniso_curvature=0.0, carricature_factor=1.0)

Builds a SmoothingShader object.

PARAMETERS:

- **num_iterations** (*int*) – The number of iterations.
- **factor_point** (*float*) – 0.1
- **factor_curvature** (*float*) – 0.0
- **factor_curvature_difference** (*float*) – 0.2
- **aniso_point** (*float*) – 0.0
- **aniso_normal** (*float*) – 0.0
- **aniso_curvature** (*float*) – 0.0
- **carricature_factor** (*float*) – 1.0

shade(stroke)

Smooths the stroke by moving the vertices to make the stroke smoother. Uses curvature flow to converge towards a curve of constant curvature. The diffusion method we use is anisotropic to prevent the diffusion across corners.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**SpatialNoiseShader**

Class hierarchy: `freestyle.types.StrokeShader` > `SpatialNoiseShader`

[Geometry shader]

__init__(amount, scale, num_octaves, smooth, pure_random)

Builds a SpatialNoiseShader object.

PARAMETERS:

- **amount** (*float*) – The amplitude of the noise.
- **scale** (*float*) – The noise frequency.
- **num_octaves** (*int*) – The number of octaves
- **smooth** (*bool*) – True if you want the noise to be smooth.
- **pure_random** (*bool*) – True if you don't want any coherence.

shade(stroke)

Spatial Noise stroke shader. Moves the vertices to make the stroke more noisy.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**SquareCapShader**

shade(stroke)

class freestyle.shaders.**StrokeTextureStepShader**

Class hierarchy: `freestyle.types.StrokeShader` > `StrokeTextureStepShader`

[Texture shader]

__init__(step)

Builds a StrokeTextureStepShader object.

PARAMETERS:

step (*float*) – The spacing along the stroke.

shade(stroke)

Assigns a spacing factor to the texture coordinates of the Stroke.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**ThicknessNoiseShader**

Class hierarchy: `freestyle.types.StrokeShader` > `ThicknessNoiseShader`

[Thickness shader]

__init__(amplitude, period)

Builds a ThicknessNoiseShader object.

PARAMETERS:

- **amplitude** (*float*) – The amplitude of the noise signal.
- **period** (*float*) – The period of the noise signal.

shade(stroke)

Adds some noise to the stroke thickness.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**TipRemoverShader**

Class hierarchy: `freestyle.types.StrokeShader` > `TipRemoverShader`

[Geometry shader]

__init__(tip_length)

Builds a TipRemoverShader object.

PARAMETERS:

tip_length (*float*) – The length of the piece of stroke we want to remove at each extremity.

shade(stroke)

Removes the stroke's extremities.

PARAMETERS:

stroke (`freestyle.types.Stroke`) – A Stroke object.

class freestyle.shaders.**py2DCurvatureColorShader**

Assigns a color (grayscale) to the stroke based on the curvature. A higher curvature will yield a brighter color.

shade(stroke)

class freestyle.shaders.**pyBackboneStretcherNoCuspShader**

Stretches the stroke's backbone, excluding cusp vertices (end junctions).

shade(stroke)

class freestyle.shaders.**pyBackboneStretcherShader**

Stretches the stroke's backbone by a given length (in pixels).

shade(stroke)

class freestyle.shaders.**pyBluePrintCirclesShader**

Draws the silhouette of the object as a circle.

shade(stroke)

class freestyle.shaders.**pyBluePrintDirectedSquaresShader**

Replaces the stroke with a directed square.

shade(stroke)

class freestyle.shaders.**pyBluePrintEllipsesShader**

shade(stroke)

class freestyle.shaders.**pyBluePrintSquaresShader**

shade(stroke)

class freestyle.shaders.**pyConstantColorShader**

Assigns a constant color to the stroke.

shade(stroke)

class freestyle.shaders.**pyConstantThicknessShader**

Assigns a constant thickness along the stroke.

shade(stroke)

class freestyle.shaders.**pyConstrainedIncreasingThicknessShader**

Increasingly thickens the stroke, constrained by a ratio of the stroke's length.

shade(stroke)

class freestyle.shaders.**pyDecreasingThicknessShader**

Inverse of pyIncreasingThicknessShader, decreasingly thickens the stroke.

shade(stroke)

class freestyle.shaders.**pyDepthDiscontinuityThicknessShader**

Assigns a thickness to the stroke based on the stroke's distance to the camera (Z-value).

shade(stroke)

class freestyle.shaders.**pyDiffusion2Shader**

Iteratively adds an offset to the position of each stroke vertex in the direction perpendicular to the stroke direction at the point. The offset is scaled l
the 2D curvature (i.e. how quickly the stroke curve is) at the point.

shade(stroke)

class freestyle.shaders.**pyFXSVaryingThicknessWithDensityShader**

Assigns thickness to a stroke based on the density of the diffuse map.

shade(stroke)

class freestyle.shaders.**pyGuidingLineShader**

shade(stroke)

class freestyle.shaders.**pyHLRShader**

Controls visibility based upon the quantitative invisibility (QI) based on hidden line removal (HLR).

shade(stroke)

class freestyle.shaders.**pyImportance2DThicknessShader**

Assigns thickness based on distance to a given point in 2D space. the thickness is inverted, so the vertices closest to the specified point have the lowest thickness.

shade(stroke)

class freestyle.shaders.**pyImportance3DThicknessShader**

Assigns thickness based on distance to a given point in 3D space.

shade(stroke)

class freestyle.shaders.**pyIncreasingColorShader**

Fades from one color to another along the stroke.

shade(stroke)

class freestyle.shaders.**pyIncreasingThicknessShader**

Increasingly thickens the stroke.

shade(stroke)

class freestyle.shaders.**pyInterpolateColorShader**

Fades from one color to another and back.

shade(stroke)

class freestyle.shaders.**pyLengthDependingBackboneStretcherShader**

Stretches the stroke's backbone proportional to the stroke's length NOTE: you'll probably want an l somewhere between (0.5 - 0). A value that is too high may yield unexpected results.

shade(stroke)

class freestyle.shaders.**pyMaterialColorShader**

Assigns the color of the underlying material to the stroke.

shade(stroke)

class freestyle.shaders.**pyModulateAlphaShader**

Limits the stroke's alpha between a min and max value.

shade(stroke)

class freestyle.shaders.**pyNonLinearVaryingThicknessShader**

Assigns thickness to a stroke based on an exponential function.

shade(stroke)

class freestyle.shaders.**pyPerlinNoise1DShader**

Displaces the stroke using the curvilinear abscissa. This means that lines with the same length and sampling interval will be identically distorted.

shade(stroke)

class freestyle.shaders.**pyPerlinNoise2DShader**

Displaces the stroke using the strokes coordinates. This means that in a scene no strokes will be distorted identically.

More information on the noise shaders can be found at: https://freestyleintegration.wordpress.com/2011/09/25/development-updates-on-septembe 25/

shade(stroke)

class freestyle.shaders.**pyRandomColorShader**

Assigns a color to the stroke based on given seed.

shade(stroke)

class freestyle.shaders.**pySLERPThicknessShader**

Assigns thickness to a stroke based on spherical linear interpolation.

shade(stroke)

class freestyle.shaders.**pySamplingShader**

Resamples the stroke, which gives the stroke the amount of vertices specified.

shade(stroke)

class freestyle.shaders.**pySinusDisplacementShader**

Displaces the stroke in the shape of a sine wave.

shade(stroke)

class freestyle.shaders.**pyTVertexRemoverShader**

Removes t-vertices from the stroke.

shade(stroke)

class freestyle.shaders.**pyTVertexThickenerShader**

Thickens TVertices (visual intersections between two edges).

shade(stroke)

class freestyle.shaders.**pyTimeColorShader**

Assigns a grayscale value that increases for every vertex. The brightness will increase along the stroke.

shade(stroke)

class freestyle.shaders.**pyTipRemoverShader**

Removes the tips of the stroke.

shade(stroke)

Undocumented

class freestyle.shaders.**pyZDependingThicknessShader**

Assigns thickness based on an object's local Z depth (point closest to camera is 1, point furthest from camera is zero).

shade(stroke)