penkit Documentation

Release 0.0.1

Paul Butler

Jan 24, 2021

Contents:

CHAPTER 1

penkit

1.1 penkit package

1.1.1 Subpackages

1.1.1.1 penkit.fractal package

Submodules

penkit.fractal.l_systems module

The `l_systems` module contains an implementation of Lindemeyer systems.

`penkit.fractal.l_systems.`**`l_system`**(*axiom*, *transformations*, *iterations=1*, *angle=45*, *resolution=1*)

Generates a texture by running transformations on a turtle program.

First, the given transformations are applied to the axiom. This is repeated *iterations* times. Then, the output is run as a turtle program to get a texture, which is returned.

For more background see: https://en.wikipedia.org/wiki/L-system

Parameters

- **`axiom`** (*str*) – the axiom of the Lindenmeyer system (a string)
- **`transformations`** (*dict*) – a dictionary mapping each char to the string that is substituted for it when the rule is applied
- **`iterations`** (*int*) – the number of times to apply the transformations
- **`angle`** (*float*) – the angle to use for turns when interpreting the string as a turtle graphics program
- **`resolution`** (*int*) – the number of midpoints to create in each turtle step

Returns A texture

penkit.fractal.l_systems.**transform_multiple**(*sequence*, *transformations*, *iterations*)
Chains a transformation a given number of times.

Parameters

- **sequence** (*str*) – a string or generator onto which transformations are applied
- **transformations** (*dict*) – a dictionary mapping each char to the string that is substituted for it when the rule is applied
- **iterations** (*int*) – how many times to repeat the transformation

Yields *str* – the next character in the output sequence.

penkit.fractal.l_systems.**transform_sequence**(*sequence*, *transformations*)
Applies a given set of substitution rules to the given string or generator.

For more background see: https://en.wikipedia.org/wiki/L-system

Parameters

- **sequence** (*str*) – a string or generator onto which transformations are applied
- **transformations** (*dict*) – a dictionary mapping each char to the string that is substituted for it when the rule is applied

Yields *str* – the next character in the output sequence.

Examples

```
>>> ''.join(transform_sequence('ABC', {}))
'ABC'
>>> ''.join(transform_sequence('ABC', {'A': 'AC', 'C': 'D'}))
'ACBD'
```

Module contents

penkit.fractal.**flowsnake**(*iterations=4*, *resolution=1*)
Generates a Peano-Gosper curve using an L-System.

For more information see: https://en.wikipedia.org/wiki/Gosper_curve

Parameters

- **iterations** (*int*) – the number of times to iterate the transformation
- **resolution** (*int*) – the number of midpoints along each line

Returns A texture

penkit.fractal.**hilbert_curve**(*iterations=5*, *resolution=1*)
Generates a Hilbert space-filling curve using an L-System.

For more information see: https://en.wikipedia.org/wiki/Hilbert_curve

Parameters

- **iterations** (*int*) – the number of times to iterate the transformation
- **resolution** (*int*) – the number of midpoints along each line

Returns A texture

penkit.fractal.**tree**(*iterations=4*, *resolution=1*, *angle=22.5*)
Generates an organic-looking tree.

Parameters

- **iterations** (*int*) – the number of times to iterate the transformation
- **resolution** (*int*) – the number of midpoints along each line
- **angle** (*float*) – the angle of branching

Returns A texture

1.1.1.2 penkit.polargraph package

Submodules

penkit.polargraph.gcode module

penkit.polargraph.segment module

Module contents

1.1.1.3 penkit.textures package

Submodules

penkit.textures.util module

The textures.util module contains utility functions for working with textures.

penkit.textures.util.**fit_texture**(*layer*)
Fits a layer into a texture by scaling each axis to (0, 1).

Does not preserve aspect ratio (TODO: make this an option).

Parameters **layer** (*layer*) – the layer to scale

Returns A texture.

Return type texture

penkit.textures.util.**rotate_texture**(*texture*, *rotation*, *x_offset=0.5*, *y_offset=0.5*)
Rotates the given texture by a given angle.

Parameters

- **texture** (*texture*) – the texture to rotate
- **rotation** (*float*) – the angle of rotation in degrees
- **x_offset** (*float*) – the x component of the center of rotation (optional)
- **y_offset** (*float*) – the y component of the center of rotation (optional)

Returns A texture.

Return type texture

Module contents

The `textures` module includes functions to generate textures.

`penkit.textures.`**`make_grid_texture`**(*num_h_lines=10*, *num_v_lines=10*, *resolution=50*)
Makes a texture consisting of a grid of vertical and horizontal lines.

Parameters

- **`num_h_lines`** (*int*) – the number of horizontal lines to draw
- **`num_v_lines`** (*int*) – the number of vertical lines to draw
- **`resolution`** (*int*) – the number of midpoints to draw on each line

Returns A texture.

`penkit.textures.`**`make_hex_texture`**(*grid_size=2*, *resolution=1*)
Makes a texture consisting on a grid of hexagons.

Parameters

- **`grid_size`** (*int*) – the number of hexagons along each dimension of the grid
- **`resolution`** (*int*) – the number of midpoints along the line of each hexagon

Returns A texture.

`penkit.textures.`**`make_lines_texture`**(*num_lines=10*, *resolution=50*)
Makes a texture consisting of a given number of horizontal lines.

Parameters

- **`num_lines`** (*int*) – the number of lines to draw
- **`resolution`** (*int*) – the number of midpoints on each line

Returns A texture.

`penkit.textures.`**`make_spiral_texture`**(*spirals=6.0*, *ccw=False*, *offset=0.0*, *resolution=1000*)
Makes a texture consisting of a spiral from the origin.

Parameters

- **`spirals`** (*float*) – the number of rotations to make
- **`ccw`** (*bool*) – make spirals counter-clockwise (default is clockwise)
- **`offset`** (*float*) – if non-zero, spirals start offset by this amount
- **`resolution`** (*int*) – number of midpoints along the spiral

Returns A texture.

1.1.2 Submodules

1.1.3 penkit.mpl_preview module

Preview plots with matplotlib.

Alternative to the `preview` module for non-Jupyter environments.

`penkit.mpl_preview.`**`draw_layer`**(*ax*, *layer*)
Draws a layer on the given matplotlib axis.

Parameters

- **ax** (*axis*) – the matplotlib axis to draw on
- **layer** (*layer*) – the layers to plot

penkit.mpl_preview.**draw_plot**(*ax*, *plot*)
Draws a plot on the given matplotlib axis.

Parameters

- **ax** (*axis*) – the matplotlib axis to draw on
- **plot** (*list*) – the layers to plot

penkit.mpl_preview.**show_layer**(*layer*)
Shortcut for `show_plot` when only one layer is needed.

Parameters **layer** (*layer*) – the layer to plot

penkit.mpl_preview.**show_plot**(*plot*)
Draws a preview of the given plot with matplotlib.

Parameters **plot** (*list*) – the plot as a list of layers

1.1.4 penkit.plot module

1.1.5 penkit.preview module

Functions for displaying plots inline inside a Jupyter notebook.

These functions are useful for iterative development of plots.

penkit.preview.**show_layer**(*layer*, **args*, ***kwargs*)
Shortcut for `show_plot` when the plot has only one layer.

Parameters

- **layer** (*layer*) – the layer to plot
- **width** (*int*) – the width of the preview
- **height** (*int*) – the height of the preview

Returns An object that renders in Jupyter as the provided plot

penkit.preview.**show_plot**(*plot*, *width=330*, *height=255*)
Preview a plot in a jupyter notebook.

Parameters

- **plot** (*list*) – the plot to display (list of layers)
- **width** (*int*) – the width of the preview
- **height** (*int*) – the height of the preview

Returns An object that renders in Jupyter as the provided plot

1.1.6 penkit.projection module

The `projection` module provides functions for rotating 2D objects (surfaces and textures) in 3D space and projecting them back to 2D.

penkit.projection.**map_texture_to_surface**(*texture*, *surface*)
Returns values on a surface for points on a texture.

Parameters

- **texture** (*texture*) – the texture to trace over the surface
- **surface** (*surface*) – the surface to trace along

Returns an array of surface heights for each point in the texture. Line separators (i.e. values that are `nan` in the texture) will be `nan` in the output, so the output will have the same dimensions as the x/y axes in the input texture.

penkit.projection.**project_and_occlude_texture**(*texture*, *surface*, *angle=45*)
Projects a texture onto a surface with occluded areas removed.

Parameters

- **texture** (*texture*) – the texture to map to the projected surface
- **surface** (*surface*) – the surface to project
- **angle** (*float*) – the angle to project at, in degrees (0 = overhead, 90 = side view)

Returns A layer.

Return type layer

penkit.projection.**project_surface**(*surface*, *angle=45*)
Returns the height of the surface when projected at the given angle.

Parameters

- **surface** (*surface*) – the surface to project
- **angle** (*float*) – the angle at which to project the surface

Returns A projected surface.

Return type surface

penkit.projection.**project_texture**(*texture_xy*, *texture_z*, *angle=45*)
Creates a texture by adding z-values to an existing texture and projecting.

When working with surfaces there are two ways to accomplish the same thing:

1. project the surface and map a texture to the projected surface
2. map a texture to the surface, and then project the result

The first method, which does not use this function, is preferred because it is easier to do occlusion removal that way. This function is provided for cases where you do not wish to generate a surface (and don't care about occlusion removal.)

Parameters

- **texture_xy** (*texture*) – the texture to project
- **texture_z** (*np.array*) – the Z-values to use in the projection
- **angle** (*float*) – the angle to project at, in degrees (0 = overhead, 90 = side view)

Returns A layer.

Return type layer

penkit.projection.**project_texture_on_surface**(*texture*, *surface*, *angle=45*)
Maps a texture onto a surface, then projects to 2D and returns a layer.

Parameters

- **texture** (*texture*) – the texture to project
- **surface** (*surface*) – the surface to project onto
- **angle** (*float*) – the projection angle in degrees (0 = top-down, 90 = side view)

Returns A layer.

Return type layer

1.1.7 penkit.shapes module

1.1.8 penkit.surfaces module

The `surfaces` module provides functions for generating **surfaces**. Surfaces are 2D matrices which act as an elevation map.

`penkit.surfaces.`**make_bubble_surface**(*dims=(500, 500)*, *repeat=3*)
Makes a surface from the product of sine functions on each axis.

Parameters

- **dims** (*pair*) – the dimensions of the surface to create
- **repeat** (*int*) – the frequency of the waves is set to ensure this many repetitions of the function

Returns A surface.

Return type surface

`penkit.surfaces.`**make_gradients**(*dims=(500, 500)*)
Makes a pair of gradients to generate textures from numpy primitives.

Parameters **dims** (*pair*) – the dimensions of the surface to create

Returns A pair of surfaces.

Return type pair

`penkit.surfaces.`**make_noise_surface**(*dims=(500, 500)*, *blur=10*, *seed=None*)
Makes a surface by generating random noise and blurring it.

Parameters

- **dims** (*pair*) – the dimensions of the surface to create
- **blur** (*float*) – the amount of Gaussian blur to apply
- **seed** (*int*) – a random seed to use (optional)

Returns A surface.

Return type surface

`penkit.surfaces.`**make_sine_surface**(*dims=(500, 500)*, *offset=0.5*, *scale=1.0*)
Makes a surface from the 3D sine function.

Parameters

- **dims** (*pair*) – the dimensions of the surface to create
- **offset** (*float*) – an offset applied to the function

- **scale** (*float*) – a scale applied to the sine frequency

Returns A surface.

Return type surface

1.1.9 penkit.turtle module

The `turtle` module contains a basic implementation of turtle graphics.

The turtle language also includes a "branching" extension that allows the turtle to return to a previously remembered state.

`penkit.turtle.`**`branching_turtle_generator`**(*turtle_program*, *turn_amount=45.0*, *initial_angle=90.0*, *resolution=1*)

Given a turtle program, creates a generator of turtle positions.

The state of the turtle consists of its position and angle. The turtle starts at the position `(0, 0)` facing up. Each character in the turtle program is processed in order and causes an update to the state. The position component of the state is yielded at each state change. A `(nan, nan)` separator is emitted between state changes for which no line should be drawn.

The turtle program consists of the following commands:

- Any letter in `ABCDEFGHIJ` means "move forward one unit and draw a path"
- Any letter in `abcdefghij` means "move forward" (no path)
- The character − means "move counter-clockwise"
- The character + means "move clockwise"
- The character `[` means "push a copy of the current state to the stack"
- The character `]` means "pop a state from the stack and return there"
- All other characters are silently ignored (this is useful when producing programs with L-Systems)

Parameters

- **turtle_program** (*str*) – a string or generator representing the turtle program
- **turn_amount** (*float*) – how much the turn commands should change the angle
- **initial_angle** (*float*) – if provided, the turtle starts at this angle (degrees)
- **resolution** (*int*) – if provided, interpolate this many points along each visible line

Yields *pair* – The next coordinate pair, or `(nan, nan)` as a path separator.

`penkit.turtle.`**`texture_from_generator`**(*generator*)

Convert a generator into a texture.

Parameters **generator** (*generator*) – a generator of coordinate pairs

Returns A texture.

Return type texture

`penkit.turtle.`**`turtle_to_texture`**(*turtle_program*, *turn_amount=45.0*, *initial_angle=90.0*, *resolution=1*)

Makes a texture from a turtle program.

Parameters

- **turtle_program** (*str*) – a string representing the turtle program; see the docstring of *branching_turtle_generator* for more details
- **turn_amount** (*float*) – amount to turn in degrees
- **initial_angle** (*float*) – initial orientation of the turtle
- **resolution** (*int*) – if provided, interpolation amount for visible lines

Returns A texture.

Return type texture

1.1.10 penkit.write module

penkit.write.**calculate_view_box**(*layers*, *aspect_ratio*, *margin=0.1*)
Calculates the size of the SVG viewBox to use.

Parameters

- **layers** (*list*) – the layers in the image
- **aspect_ratio** (*float*) – the height of the output divided by the width
- **margin** (*float*) – minimum amount of buffer to add around the image, relative to the total dimensions

Returns

a 4-tuple of floats representing the viewBox according to SVG specifications `(x, y, width, height)`.

Return type tuple

penkit.write.**layer_to_path**(*layer*)
Generates an SVG path from a given layer.

Parameters **layer** (*layer*) – the layer to convert

Returns an SVG path

Return type str

penkit.write.**layer_to_svg**(*layer*, ***kwargs*)
Converts a layer into an SVG image.

Wrapper around `plot_to_svg`.

Parameters

- **layer** (*layer*) – the layer to plot
- **width** (*float*) – the width of the resulting image
- **height** (*float*) – the height of the resulting image
- **unit** (*str*) – the units of the resulting image if not pixels

Returns A stringified XML document representing the image

Return type str

penkit.write.**plot_to_svg**(*plot*, *width*, *height*, *unit=''*)
Converts a plot (list of layers) into an SVG document.

Parameters

- **plot** (*list*) – list of layers that make up the plot
- **width** (*float*) – the width of the resulting image
- **height** (*float*) – the height of the resulting image
- **unit** (*str*) – the units of the resulting image if not pixels

Returns A stringified XML document representing the image

Return type str

penkit.write.**write_plot**(*plot*, *filename*, *width=11*, *height=8.5*, *unit='in'*)

Writes a plot SVG to a file.

Parameters

- **plot** (*list*) – a list of layers to plot
- **filename** (*str*) – the name of the file to write
- **width** (*float*) – the width of the output SVG
- **height** (*float*) – the height of the output SVG
- **unit** (*str*) – the unit of the height and width

1.1.11 Module contents

- genindex
- modindex
- search

Python Module Index

p

Index

W