lacecore

Mar 02, 2020

Contents

class lacecore.**Mesh**(*v*, *f*, *copy_v=False*, *copy_f=False*)

A triangular or quad mesh. Vertices and faces are represented using NumPy arrays. Instances are read-only, at least for now. This class is optimized for cloud computation.

Parameters

- **v** (*np.ndarray*) – A *kx3* array of vertices. It will be marked read-only.
- **f** (*np.ndarray*) – A *kx3* or *kx4* array of vertex indices which make up the faces. It will be marked read-only.
- **copy_v** (*bool*) – When *True*, the input vertices will be copied before they are marked read-only.
- **copy_f** (*bool*) – When *True*, the input faces will be copied before they are marked read-only.

keeping_vertices_above(*dim*, *point*)

Select vertices which, when projected to the given axis, lie further along that axis than the projection of the given point.

Return a new mesh, without mutating the callee.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.
- **point** (*np.arraylike*) – The point of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

keeping_vertices_at_or_above(*dim*, *point*)

Select vertices which, when projected to the given axis, are either coincident with the projection of the given point, or lie further along the axis.

Return a new mesh, without mutating the callee.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.
- **point** (*np.arraylike*) – The point of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

keeping_vertices_at_or_below(*dim*, *point*)

Select vertices which, when projected to the given axis, are either coincident with the projection of the given point, or lie before it.

Return a new mesh, without mutating the callee.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.
- **point** (*np.arraylike*) – The point of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

keeping_vertices_behind_plane(*plane*)
Select the vertices which are behind the given plane.

Return a new mesh, without mutating the callee.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

keeping_vertices_below(*dim*, *point*)
Select vertices which, when projected to the given axis, lie before the projection of the given point.

Return a new mesh, without mutating the callee.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.
- **point** (*np.arraylike*) – The point of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

keeping_vertices_in_front_of_plane(*plane*)
Select the vertices which are in front of the given plane.

Return a new mesh, without mutating the callee.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

keeping_vertices_on_or_behind_plane(*plane*)
Select the vertices which are either on or behind the given plane.

Return a new mesh, without mutating the callee.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

keeping_vertices_on_or_in_front_of_plane(*plane*)
Select the vertices which are either on or in front of the given plane.

Return a new mesh, without mutating the callee.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

num_f
The number of faces.

Returns The number of faces.

Return type int

num_v
The number of vertices.

Returns The number of vertices.

Return type int

picking_faces(*indices_or_boolean_mask*)
Select only the given faces.

Return a new mesh, without mutating the callee.

Parameters **indices_or_boolean_mask** (*np.arraylike*) – Either a list of vertex indices, or a boolean mask the same length as the vertex array.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

picking_vertices(*indices_or_boolean_mask*)
Select only the given vertices.

Return a new mesh, without mutating the callee.

Parameters **indices_or_boolean_mask** (*np.arraylike*) – Either a list of vertex indices, or a boolean mask the same length as the vertex array.

Returns A submesh containing the selection.

Return type *lacecore.Mesh*

select()
Begin a chained selection operation. After invoking *.select()*, apply selection criteria, then invoke *.end()* to create a submesh.

Include *.union()* in the chain to combine multiple sets of selection criteria into a single submesh.

Does not mutate the callee.

Returns The selection operation.

Return type *lacecore.Selection*

Example

```
>>> centroid = np.average(mesh.v, axis=0)
>>> upper_right_quadrant = (
    mesh.select()
    .vertices_above(centroid, dim=0)
    .vertices_above(centroid, dim=1)
    .end()
)
>>> upper_half_plus_right_half = (
```

(continues on next page)

(continued from previous page)

```
    mesh.select()
    .vertices_above(centroid, dim=0)
    .union()
    .vertices_above(centroid, dim=1)
    .end()
)
```

CHAPTER 1

Selection operations

class lacecore.**Selection**(*target*, *union_with=[]*)

Encapsulate a chained submesh selection operation.

Invoke *.end()* to apply the selection operation and create a submesh. By default, orphaned vertices are pruned. However you can keep them by invoking *.end(prune_orphan_vertices=True)*.

Include *.union()* in the chain to combine more than one set of selection criteria into a single submesh.

Parameters

- **target** (lacecore.Mesh) – The mesh on which to operate.
- **union_with** (lacecore.Selection) – The operation with which the new instance should combine itself. Normally this is reserved for internal use.

end(*prune_orphan_vertices=True*, *ret_indices_of_original_faces_and_vertices=False*)

Apply the selection to construct a submesh.

Parameters

- **prune_orphan_vertices** (*bool*) – When *True*, remove vertices which are referenced only by faces which are being removed.
- **ret_indices_of_original_faces_and_vertices** – When *True*, also return the indices of the original faces and vertices.

Returns

Either the submesh as an instance of *lacecore.Mesh*, or a tuple *(submesh, indices_of_original_faces, indices_of_original_vertices)*. The index arrays contain the new indices of the original vertices, and *-1* for each removed face and vertex.

Return type object

pick_faces(*indices_or_boolean_mask*)

Select only the given faces.

Parameters **indices_or_boolean_mask** (*np.arraylike*) – Either a list of face indices, or a boolean mask the same length as the face array.

Returns self

pick_vertices(*indices_or_boolean_mask*)
Select only the given vertices.

Parameters **indices_or_boolean_mask** (*np.arraylike*) – Either a list of vertex indices, or a boolean mask the same length as the vertex array.

Returns self

union()
Chain on a new selection object. This works like a boolean "or" to combine two sets of submesh operations.

Parameters **indices_or_boolean_mask** (*np.arraylike*) – Either a list of face indices, or a boolean mask the same length as the face array.

Returns

The new selection operation, which will combine itself with *self*.

Return type *lacecore.Selection*

Example

```
>>> upper_half_plus_right_half = (
    mesh.select()
    .vertices_above(centroid, dim=0)
    .union()
    .vertices_above(centroid, dim=1)
    .end()
)
```

vertices_above(*dim*, *point*)
Select vertices which, when projected to the given axis, lie further along that axis than the projection of the given point.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.
- **point** (*np.arraylike*) – The point of interest.

Returns self

vertices_at_or_above(*dim*, *point*)
Select vertices which, when projected to the given axis, are either coincident with the projection of the given point, or lie further along the axis.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.
- **point** (*np.arraylike*) – The point of interest.

Returns self

vertices_at_or_below(*dim*, *point*)
Select vertices which, when projected to the given axis, are either coincident with the projection of the given point, or lie before it.

Parameters

- **dim** (*int*) – The axis of interest: 0 for *x*, 1 for *y*, 2 for *z*.

- **point** (*np.arraylike*) – The point of interest.

Returns self

vertices_behind_plane(*plane*)
Select the vertices which are behind the given plane.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns self

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

vertices_below(*dim*, *point*)
Select vertices which, when projected to the given axis, lie before the projection of the given point.

Parameters

- **dim** (*int*) – The axis of interest: 0 for x, 1 for y, 2 for z.
- **point** (*np.arraylike*) – The point of interest.

Returns self

vertices_in_front_of_plane(*plane*)
Select the vertices which are in front of the given plane.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns self

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

vertices_on_or_behind_plane(*plane*)
Select the vertices which are either on or behind the given plane.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns self

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

vertices_on_or_in_front_of_plane(*plane*)
Select the vertices which are either on or in front of the given plane.

Parameters **plane** (*polliwog.Plane*) – The plane of interest.

Returns self

See also:

https://polliwog.readthedocs.io/en/latest/#polliwog.Plane

CHAPTER 2

Groups

class lacecore.**GroupMap**(*num_elements*, *group_names*, *masks*, *copy_masks=False*)

An immutable map of groups of elements, which are allowed to overlap. These can be used for face or vertex groups, as in the Wavefront OBJ standard.

Parameters

- **num_elements** (*int*) – The total number of elements. This determines the length of the masks.
- **group_names** (*list*) – The names of the groups.
- **masks** (*np.array*) – A boolean array with a row containing a boolean mask for each group.

See also:

http://paulbourke.net/dataformats/obj/

__getitem__(*group_name*)

Get the read-only mask for the requested group.

Parameters **group_name** (*string*) – The desired group.

Returns A read-only boolean array with length equal to *self.num_elements*.

Return type np.array

__iter__()

Iterate over the groups.

Returns An iterator over the groups.

Return type list_iterator

__len__()

Get the number of groups.

Returns The number of groups.

Return type int

classmethod from_dict(*group_data*, *num_elements*)
Create a group map from a dictionary of elements. The keys are the group names and the values are lists of element indices.

Parameters

- **group_data** (*dict*) – The group data.
- **num_elements** (*int*) – The total number of elements.

keys()
Get the names of all the groups.

Returns A list of the group names.

Return type list

union(**group_names*)
Construct the union of the requested groups and return it as a writable mask.

Parameters **group_names** (*list*) – The requested groups.

Returns A boolean mask with length equal to *self.num_elements*.

Return type np.array

CHAPTER 3

Tesselated shapes

Functions for creating meshes for tesselated 3D shapes.

See also:

https://en.wikipedia.org/wiki/Tessellation_(computer_graphics)

lacecore.shapes.**rectangular_prism**(*origin*, *size*)

Tesselate an axis-aligned rectangular prism. One vertex is *origin*. The diametrically opposite vertex is *origin* + *size*.

Parameters

- **origin** (*np.ndarray*) – A 3D point vector containing the point on the prism with the minimum x, y, and z coords.
- **size** (*np.ndarray*) – A 3D vector specifying the prism's length, width, and height, which should be positive.

Returns A *Mesh* instance containing the rectangular prism.

Return type *lacecore.Mesh*

lacecore.shapes.**cube**(*origin*, *size*)

Tesselate an axis-aligned cube. One vertex is *origin*. The diametrically opposite vertex is *size* units along $+x$, $+y$, and $+z$.

Parameters

- **origin** (*np.ndarray*) – A 3D point vector containing the point on the prism with the minimum x, y, and z coords.
- **size** (*float*) – The length, width, and height of the cube, which should be positive.

Returns A *Mesh* instance containing the cube.

Return type *lacecore.Mesh*

lacecore.shapes.**triangular_prism**(*p1*, *p2*, *p3*, *height*)

Tesselate a triangular prism whose base is the triangle *p1*, *p2*, *p3*. If the vertices are oriented in a counterclockwise direction, the prism extends from behind them.

Parameters

- **p1** (*np.ndarray*) – A 3D point on the base of the prism.
- **p2** (*np.ndarray*) – A 3D point on the base of the prism.
- **p3** (*np.ndarray*) – A 3D point on the base of the prism.
- **height** (*float*) – The height of the prism, which should be positive.

Returns A *Mesh* instance containing the triangular prism.

Return type *lacecore.Mesh*

lacecore.shapes.**rectangle**()

Create a rectangle.

Returns A *Mesh* instance containing the rectangle.

Return type *lacecore.Mesh*

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index