

---

**lacecore**

**Mar 01, 2020**



---

## Contents

---

<b>1</b>	<b>Selection operations</b>	<b>7</b>
<b>2</b>	<b>Groups</b>	<b>11</b>
<b>3</b>	<b>Tesselated shapes</b>	<b>13</b>
<b>4</b>	<b>Indices and tables</b>	<b>15</b>
	<b>Python Module Index</b>	<b>17</b>
	<b>Index</b>	<b>19</b>



**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  $k \times 3$  array of vertices. It will be marked read-only.
- **f** (*np.ndarray*) – A  $k \times 3$  or  $k \times 4$  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.

**faces\_flipped** ()

Flip the orientation of the faces.

**Returns** A mesh with transformed faces.

**Return type** *lacecore.Mesh*

**flipped** (*dim, preserve\_vertex\_centroid=False*)

Flip about the given axis.

#### Parameters

- **dim** (*int*) – The axis to flip around: 0 for *x*, 1 for *y*, 2 for *z*.
- **preserve\_vertex\_centroid** (*bool*) – When *True*, translate after flipping to preserve the original vertex centroid.

**Returns** A mesh with transformed vertices.

**Return type** *lacecore.Mesh*

**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>

**non\_uniformly\_scaled** (*x\_factor*, *y\_factor*, *z\_factor*)

Scale along each axis by the given factors.

**Parameters**

- **x\_factor** (*float*) – The scale factor along the *x* axis.
- **y\_factor** (*float*) – The scale factor along the *y* axis.
- **z\_factor** (*float*) – The scale factor along the *z* axis.

**Returns** A mesh with transformed vertices.

**Return type** *lacecore.Mesh*

**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*

**reoriented** (*up, look*)

Reorient using up and look.

**Returns** A mesh with transformed vertices.

**Return type** *lacecore.Mesh*

**rotated** (*rotation*)

Rotate by the given 3x3 rotation matrix or a Rodrigues vector.

**Returns** A mesh with transformed vertices.

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

**transform** ()

Begin a composite transform operation. After invoking *.transform()*, apply transformations, then invoke *.end()* to create a mesh with transformed vertices.

Does not mutate the callee.

**Returns** The transform operation.

**Return type** *lacecore.Transform*



## Example

```
>>> transformed = (  
    mesh.transform()  
    .translate(3.0 * vg.basis.x)  
    .uniform_scale(3.0)  
    .end()  
)
```

### **translated** (*translation*)

Translate by the vector provided.

**Parameters** **vector** (*np.arraylike*) – A 3x1 vector.

**Returns** A mesh with transformed vertices.

**Return type** *lacecore.Mesh*

### **uniformly\_scaled** (*factor*)

Scale by the given factor.

**Parameters** **factor** (*float*) – The scale factor.

**Returns** A mesh with transformed vertices.

**Return type** *lacecore.Mesh*

### **units\_converted** (*from\_units, to\_units*)

Convert the mesh from one set of units to another.

Support the length units from Ounce: <https://github.com/lace/ounce/blob/master/ounce/core.py#L26>

**Returns** A mesh with transformed vertices.

**Return type** *lacecore.Mesh*

### **vertex\_centroid**

The centroid or geometric average of the vertices.



---

## 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>



**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



---

## Tessellated shapes

---

Functions for creating meshes for tessellated 3D shapes.

**See also:**

[https://en.wikipedia.org/wiki/Tessellation\\_\(computer\\_graphics\)](https://en.wikipedia.org/wiki/Tessellation_(computer_graphics))

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

Tessellate 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*)

Tessellate 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*)

Tessellate 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`



### I

`lacecore.shapes`, [13](#)



## Symbols

`__getitem__()` (*lacecore.GroupMap method*), 11  
`__iter__()` (*lacecore.GroupMap method*), 11  
`__len__()` (*lacecore.GroupMap method*), 11

## C

`cube()` (*in module lacecore.shapes*), 13

## E

`end()` (*lacecore.Selection method*), 7

## F

`faces_flipped()` (*lacecore.Mesh method*), 1  
`flipped()` (*lacecore.Mesh method*), 1  
`from_dict()` (*lacecore.GroupMap class method*), 11

## G

`GroupMap` (*class in lacecore*), 11

## K

`keeping_vertices_above()` (*lacecore.Mesh method*), 1  
`keeping_vertices_at_or_above()` (*lacecore.Mesh method*), 1  
`keeping_vertices_at_or_below()` (*lacecore.Mesh method*), 2  
`keeping_vertices_behind_plane()` (*lacecore.Mesh method*), 2  
`keeping_vertices_below()` (*lacecore.Mesh method*), 2  
`keeping_vertices_in_front_of_plane()` (*lacecore.Mesh method*), 2  
`keeping_vertices_on_or_behind_plane()` (*lacecore.Mesh method*), 2  
`keeping_vertices_on_or_in_front_of_plane()` (*lacecore.Mesh method*), 3  
`keys()` (*lacecore.GroupMap method*), 12

## L

`lacecore.shapes` (*module*), 13

## M

`Mesh` (*class in lacecore*), 1

## N

`non_uniformly_scaled()` (*lacecore.Mesh method*), 3  
`num_f` (*lacecore.Mesh attribute*), 3  
`num_v` (*lacecore.Mesh attribute*), 3

## P

`pick_faces()` (*lacecore.Selection method*), 7  
`pick_vertices()` (*lacecore.Selection method*), 8  
`picking_faces()` (*lacecore.Mesh method*), 3  
`picking_vertices()` (*lacecore.Mesh method*), 3

## R

`rectangle()` (*in module lacecore.shapes*), 14  
`rectangular_prism()` (*in module lacecore.shapes*), 13  
`reoriented()` (*lacecore.Mesh method*), 4  
`rotated()` (*lacecore.Mesh method*), 4

## S

`select()` (*lacecore.Mesh method*), 4  
`Selection` (*class in lacecore*), 7

## T

`transform()` (*lacecore.Mesh method*), 4  
`translated()` (*lacecore.Mesh method*), 5  
`triangular_prism()` (*in module lacecore.shapes*), 13

## U

`uniformly_scaled()` (*lacecore.Mesh method*), 5  
`union()` (*lacecore.GroupMap method*), 12  
`union()` (*lacecore.Selection method*), 8

`units_converted()` (*lacecore.Mesh method*), 5

## V

`vertex_centroid` (*lacecore.Mesh attribute*), 5

`vertices_above()` (*lacecore.Selection method*), 8

`vertices_at_or_above()` (*lacecore.Selection method*), 8

`vertices_at_or_below()` (*lacecore.Selection method*), 8

`vertices_behind_plane()` (*lacecore.Selection method*), 9

`vertices_below()` (*lacecore.Selection method*), 9

`vertices_in_front_of_plane()`  
(*lacecore.Selection method*), 9

`vertices_on_or_behind_plane()`  
(*lacecore.Selection method*), 9

`vertices_on_or_in_front_of_plane()`  
(*lacecore.Selection method*), 9