docs.blender.org

Mesh(ID) — Blender Python API

13-17 minutes

Mesh Data

The mesh data is accessed in object mode and intended for compact storage, for more flexible mesh editing from python see `bmesh`.

Blender stores 4 main arrays to define mesh geometry.

- `Mesh.vertices` (3 points in space)
- `Mesh.edges` (reference 2 vertices)
- `Mesh.loops` (reference a single vertex and edge)
- `Mesh.polygons`: (reference a range of loops)

Each polygon reference a slice in the loop array, this way, polygons do not store vertices or corner data such as UV's directly, only a reference to loops that the polygon uses.

`Mesh.loops`, `Mesh.uv_layers` `Mesh.vertex_colors` are all aligned so the same polygon loop indices can be used to find the UV's and vertex colors as with as the vertices.

To compare mesh API options see: NGons and Tessellation Faces

This example script prints the vertices and UV's for each polygon, assumes the active object is a mesh with UVs.

```
import bpy

me = bpy.context.object.data
uv_layer = me.uv_layers.active.data

for poly in me.polygons:
    print("Polygon index: %d, length: %d" % (poly.index,
poly.loop_total))

    # range is used here to show how the polygons reference
loops,
    # for convenience 'poly.loop_indices' can be used instead.
    for loop_index in range(poly.loop_start, poly.loop_start +
poly.loop_total):
        print("    Vertex: %d" % me.loops[loop_index].vertex_index)
        print("    UV: %r" % uv_layer[loop_index].uv)
```

base classes — `bpy_struct`, `ID`

class bpy.types.Mesh(*ID*)

Mesh data-block defining geometric surfaces

animation_data

Animation data for this data-block

Type

`AnimData`, (readonly)

attributes

Geometry attributes

Type

`AttributeGroup` `bpy_prop_collection` of `Attribute`, (readonly)

auto_smooth_angle

Maximum angle between face normals that will be considered as smooth (unused if custom split normals data are available)

Type

float in [0, 3.14159], default 0.523599

auto_texspace

Adjust active object's texture space automatically when transforming object

Type

boolean, default True

cycles

Cycles mesh settings

Type

`CyclesMeshSettings`, (readonly)

edges

Edges of the mesh

Type

`MeshEdges` `bpy_prop_collection` of `MeshEdge`, (readonly)

face_maps

Type

`MeshFaceMapLayers` `bpy_prop_collection` of `MeshFaceMapLayer`, (readonly)

has_custom_normals

True if there are custom split normals data in this mesh

Type

boolean, default False, (readonly)

is_editmode

True when used in editmode

Type

boolean, default False, (readonly)

loop_triangles

Tessellation of mesh polygons into triangles

Type

`MeshLoopTriangles` `bpy_prop_collection` of `MeshLoopTriangle`, (readonly)

loops

Loops of the mesh (polygon corners)

Type

`MeshLoops` `bpy_prop_collection` of `MeshLoop`, (readonly)

materials

Type

`IDMaterials` `bpy_prop_collection` of `Material`, (readonly)

polygon_layers_float

Type

`PolygonFloatProperties` `bpy_prop_collection` of `MeshPolygonFloatPropertyLayer`, (readonly)

polygon_layers_int

Type

`PolygonIntProperties` `bpy_prop_collection` of `MeshPolygonIntPropertyLayer`, (readonly)

polygon_layers_string

Type

`PolygonStringProperties` `bpy_prop_collection` of `MeshPolygonStringPropertyLayer`, (readonly)

polygon_normals

The normal direction of each polygon, defined by the winding order and position of its vertices

Type

`bpy_prop_collection` of `MeshNormalValue`, (readonly)

polygons

Polygons of the mesh

Type

`MeshPolygons` `bpy_prop_collection` of `MeshPolygon`, (readonly)

remesh_mode

- `VOXEL` Voxel – Use the voxel remesher.
- `QUAD` Quad – Use the quad remesher.

Type

enum in [‘VOXEL’, ‘QUAD’], default ‘VOXEL’

remesh_voxel_adaptivity

Reduces the final face count by simplifying geometry where detail is not needed, generating triangles. A value greater than 0 disables Fix Poles

Type

float in [0, 1], default 0.0

remesh_voxel_size

Size of the voxel in object space used for volume evaluation. Lower values preserve finer details

Type

float in [0.0001, inf], default 0.1

sculpt_vertex_colors

All vertex colors

Type

`VertColors` `bpy_prop_collection` of `MeshVertColorLayer`, (readonly)

shape_keys

Type

`Key`, (readonly)

skin_vertices

All skin vertices

Type

`bpy_prop_collection` of `MeshSkinVertexLayer`, (readonly)

texco_mesh

Derive texture coordinates from another mesh

Type

`Mesh`

texspace_location

Texture space location

Type

float array of 3 items in [-inf, inf], default (0.0, 0.0, 0.0)

texspace_size

Texture space size

Type

float array of 3 items in [-inf, inf], default (1.0, 1.0, 1.0)

texture_mesh

Use another mesh for texture indices (vertex indices must be aligned)

Type

Mesh

total_edge_sel

Selected edge count in editmode

Type

int in [0, inf], default 0, (readonly)

total_face_sel

Selected face count in editmode

Type

int in [0, inf], default 0, (readonly)

total_vert_sel

Selected vertex count in editmode

Type

int in [0, inf], default 0, (readonly)

use_auto_smooth

Auto smooth (based on smooth/sharp faces/edges and angle between faces), or use custom split normals data if available

Type

boolean, default False

use_auto_texspace

Adjust active object's texture space automatically when transforming object

Type

boolean, default True

use_customdata_edge_bevel

Type

boolean, default False

use_customdata_edge_crease

Type

boolean, default False

use_customdata_vertex_bevel

Type

boolean, default False

use_customdata_vertex_crease

Type

boolean, default False

use_mirror_topology

Use topology based mirroring (for when both sides of mesh have matching, unique topology)

Type

boolean, default False

use_mirror_vertex_groups

Mirror the left/right vertex groups when painting. The symmetry axis is determined by the symmetry settings

Type

boolean, default True

use_mirror_x

Enable symmetry in the X axis

Type

boolean, default False

use_mirror_y

Enable symmetry in the Y axis

Type

boolean, default False

use_mirror_z

Enable symmetry in the Z axis

Type

boolean, default False

use_paint_mask

Face selection masking for painting

Type

boolean, default False

use_paint_mask_vertex

Vertex selection masking for painting

Type

boolean, default False

use_remesh_fix_poles

Produces less poles and a better topology flow

Type

boolean, default True

use_remesh_preserve_paint_mask

Keep the current mask on the new mesh

Type

boolean, default False

use_remesh_preserve_sculpt_face_sets

Keep the current Face Sets on the new mesh

Type

boolean, default False

use_remesh_preserve_vertex_colors

Keep the current vertex colors on the new mesh

Type

boolean, default False

use_remesh_preserve_volume

Projects the mesh to preserve the volume and details of the original mesh

Type

boolean, default True

uv_layer_clone

UV loop layer to be used as cloning source

Type

`MeshUVLoopLayer`

uv_layer_clone_index

Clone UV loop layer index

Type

int in [0, inf], default 0

uv_layer_stencil

UV loop layer to mask the painted area

Type

`MeshUVLoopLayer`

uv_layer_stencil_index

Mask UV loop layer index

Type

int in [0, inf], default 0

uv_layers

All UV loop layers

Type

`UVLoopLayers` `bpy_prop_collection` of `MeshUVLoopLayer`, (readonly)

vertex_colors

All vertex colors

Type

`LoopColors` `bpy_prop_collection` of `MeshLoopColorLayer`, (readonly)

vertex_creases

Sharpness of the vertices

Type

`bpy_prop_collection` of `MeshVertexCreaseLayer`, (readonly)

vertex_layers_float

Type

`VertexFloatProperties` `bpy_prop_collection` of `MeshVertexFloatPropertyLayer`, (readonly)

vertex_layers_int

Type

`VertexIntProperties` `bpy_prop_collection` of `MeshVertexIntPropertyLayer`, (readonly)

vertex_layers_string

Type

`VertexStringProperties` `bpy_prop_collection` of `MeshVertexStringPropertyLayer`, (readonly)

vertex_normals

The normal direction of each vertex, defined as the average of the surrounding face normals

Type

bpy_prop_collection of MeshNormalValue, (readonly)

vertex_paint_masks

Vertex paint mask

Type

bpy_prop_collection of MeshPaintMaskLayer, (readonly)

vertices

Vertices of the mesh

Type

MeshVertices bpy_prop_collection of MeshVertex, (readonly)

edge_keys

(readonly)

transform(*matrix*, *shape_keys=False*)

Transform mesh vertices by a matrix (Warning: inverts normals if matrix is negative)

Parameters

- **matrix** (*float multi-dimensional array of 4 * 4 items in [-inf, inf]*) – Matrix
- **shape_keys** (*boolean, (optional)*) – Transform Shape Keys

flip_normals()

Invert winding of all polygons (clears tessellation, does not handle custom normals)

calc_normals()

Calculate vertex normals

create_normals_split()

Empty split vertex normals

calc_normals_split()

Calculate split vertex normals, which preserve sharp edges

free_normals_split()

Free split vertex normals

split_faces(*free_loop_normals=True*)

Split faces based on the edge angle

Parameters

free_loop_normals (*boolean, (optional)*) – Free Loop Normals, Free loop normals custom data layer

calc_tangents(*uvmap=''*)

Compute tangents and bitangent signs, to be used together with the split normals to get a complete tangent space for normal mapping (split normals are also computed if not yet present)

Parameters

uvmap (*string, (optional, never None)*) – Name of the UV map to use for tangent space computation

free_tangents()

Free tangents

calc_loop_triangles()

Calculate loop triangle tessellation (supports editmode too)

calc_smooth_groups(*use_bitflags=False*)

Calculate smooth groups from sharp edges

Parameters

use_bitflags (*boolean, (optional)*) – Produce bitflags groups instead of simple numeric values

Return (poly_groups, groups)

poly_groups, Smooth Groups, int array of 1 items in [-inf, inf]

groups, Total number of groups, int in [0, inf]

normals_split_custom_set(*normals*)

Define custom split normals of this mesh (use zero-vectors to keep auto ones)

Parameters

normals (*float multi-dimensional array of 1 * 3 items in [-1, 1]*) – Normals

normals_split_custom_set_from_vertices(*normals*)

Define custom split normals of this mesh, from vertices' normals (use zero-vectors to keep auto ones)

Parameters

normals (*float multi-dimensional array of 1 * 3 items in [-1, 1]*) – Normals

update(*calc_edges=False*, *calc_edges_loose=False*)

update

Parameters

- **calc_edges** (*boolean, (optional)*) – Calculate Edges, Force recalculation of edges
- **calc_edges_loose** (*boolean, (optional)*) – Calculate Loose Edges, Calculate the loose state of each edge

update_gpu_tag()

update_gpu_tag

unit_test_compare(*mesh=None*, *threshold=7.1526e-06*)

unit_test_compare

Parameters

- **mesh** (Mesh, (optional)) – Mesh to compare to
- **threshold** (*float in [0, inf], (optional)*) – Threshold, Comparison tolerance threshold

Returns

Return value, String description of result of comparison

Return type

string, (never None)

clear_geometry()

Remove all geometry from the mesh. Note that this does not free shape keys or materials

validate(*verbose=False*, *clean_customdata=True*)

Validate geometry, return True when the mesh has had invalid geometry corrected/removed

Parameters

- **verbose** (*boolean, (optional)*) – Verbose, Output information about the errors found
- **clean_customdata** (*boolean, (optional)*) – Clean Custom Data, Remove temp/cached custom-data layers, like e.g. normals…

Returns

Result

Return type

boolean

validate_material_indices()

Validate material indices of polygons, return True when the mesh has had invalid indices corrected (to default 0)

Returns

Result

Return type

boolean

count_selected_items()

Return the number of selected items (vert, edge, face)

Returns

Result

Return type

int array of 3 items in [0, inf]

from_pydata(*vertices*, *edges*, *faces*)

Make a mesh from a list of vertices/edges/faces Until we have a nicer way to make geometry, use this.

Parameters

- **vertices** (*iterable object*) – float triplets each representing (X, Y, Z) eg: [(0.0, 1.0, 0.5), …].
- **edges** (*iterable object*) –

  int pairs, each pair contains two indices to the *vertices* argument. eg: [(1, 2), …]

  When an empty iterable is passed in, the edges are inferred from the polygons.
- **faces** (*iterable object*) – iterator of faces, each faces contains three or more indices to the

vertices argument. eg: [(5, 6, 8, 9), (1, 2, 3), …]

Warning

Invalid mesh data *(out of range indices, edges with matching indices, 2 sided faces… etc)* are **not** prevented. If the data used for mesh creation isn't known to be valid, run `Mesh.validate` after this function.

classmethod bl_rna_get_subclass(*id*, *default=None*)

Parameters

id (*string*) – The RNA type identifier.

Returns

The RNA type or default when not found.

Return type

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(*id*, *default=None*)

Parameters

id (*string*) – The RNA type identifier.

Returns

The class or default when not found.

Return type

type

Inherited Properties

• `bpy_struct.id_data`	• `ID.tag`
• `ID.name`	• `ID.is_library_indirect`
• `ID.name_full`	• `ID.library`
• `ID.is_evaluated`	• `ID.library_weak_reference`
• `ID.original`	• `ID.asset_data`

- ID.users
- ID.use_fake_user
- ID.is_embedded_data
- ID.override_library
- ID.preview

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set

- bp
- bp
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I
- I

• bpy_struct.property_unset	• ID

References

• bpy.context.mesh	• Mesh.texco_mes
• BlendData.meshes	• Mesh.texture_r
• BlendDataMeshes.new	• Mesh.unit_test
• BlendDataMeshes.new_from_object	• Object.to_mesh
• BlendDataMeshes.remove	