Skip to content

bpy_extras submodule (bpy_extras.mesh_utils)

bpy_extras.mesh_utils.**mesh_linked_uv_islands(mesh)**

Returns lists of polygon indices connected by UV islands.

PARAMETERS:

mesh (`bpy.types.Mesh`) – the mesh used to group with.

RETURNS:

list of lists containing polygon indices

RETURN TYPE:

list[list[int]]

bpy_extras.mesh_utils.**mesh_linked_triangles(mesh)**

Splits the mesh into connected triangles, use this for separating cubes from other mesh elements within 1 mesh data-block.

PARAMETERS:

mesh (`bpy.types.Mesh`) – the mesh used to group with.

RETURNS:

Lists of lists containing triangles.

RETURN TYPE:

list[list[`bpy.types.MeshLoopTriangle`]]

bpy_extras.mesh_utils.**edge_face_count_dict(mesh)**

RETURNS:

Dictionary of edge keys with their value set to the number of faces using each edge.

RETURN TYPE:

dict[tuple[int, int], int]

bpy_extras.mesh_utils.**edge_face_count(mesh)**

RETURNS:

list face users for each item in mesh.edges.

RETURN TYPE:

list[int]

bpy_extras.mesh_utils.**edge_loops_from_edges(mesh, edges=None)**

Edge loops defined by edges

Takes me.edges or a list of edges and returns the edge loops

return a list of vertex indices. [[1, 6, 7, 2], …]

closed loops have matching start and end values.

bpy_extras.mesh_utils.**ngon_tessellate(from_data, indices, fix_loops=True, debug_print=True)**

Takes a poly-line of indices (ngon) and returns a list of face index lists. Designed to be used for importers that need indices for an ngon to create fr existing verts.

PARAMETERS:

- **from_data** (`bpy.types.Mesh` | list[Sequence[float]] | tuple[Sequence[float]]) – Either a mesh, or a list/tuple of 3D vectors.
- **indices** (*list[int]*) – a list of indices to use this list is the ordered closed poly-line to fill, and can be a subset of the data given.
- **fix_loops** (*bool*) – If this is enabled poly-lines that use loops to make multiple poly-lines are dealt with correctly.

bpy_extras.mesh_utils.**triangle_random_points(num_points, loop_triangles)**

Generates a list of random points over mesh loop triangles.

PARAMETERS:

- **num_points** (*int*) – The number of random points to generate on each triangle.
- **loop_triangles** (Sequence[`bpy.types.MeshLoopTriangle`]) – Sequence of the triangles to generate points on.

RETURNS:

List of random points over all triangles.

RETURN TYPE:

list[`mathutils.Vector`]