Skip to content

Modes and Mesh Access

When working with mesh data you may run into the problem where a script fails to run as expected in Edit-Mode. This is caused by Edit-Mode having i own data which is only written back to the mesh when exiting Edit-Mode.

A common example is that exporters may access a mesh through `obj.data` (a `bpy.types.Mesh`) when the user is in Edit-Mode, where the mesh data is available but out of sync with the edit mesh.

In this situation you can…

- Exit Edit-Mode before running the tool.
- Explicitly update the mesh by calling `bmesh.types.BMesh.to_mesh`.
- Modify the script to support working on the edit-mode data directly, see: `bmesh.from_edit_mesh`.
- Report the context as incorrect and only allow the script to run outside Edit-Mode.

N-Gons and Tessellation

Since 2.63 n-gons are supported, this adds some complexity since in some cases you need to access triangles still (some exporters for example).

There are now three ways to access faces:

- `bpy.types.MeshPolygon` – this is the data structure which now stores faces in Object-Mode (access as `mesh.polygons` rather than `mesh.faces`).
- `bpy.types.MeshLoopTriangle` – the result of tessellating polygons into triangles (access as `mesh.loop_triangles`).
- `bmesh.types.BMFace` – the polygons as used in Edit-Mode.

For the purpose of the following documentation, these will be referred to as polygons, loop triangles and BMesh-faces respectively.

Faces with five or more sides will be referred to as `ngons`.

Support Overview

Usage	`bpy.types.MeshPolygon`	`bpy.types.MeshLoopTriangle`	`bmesh.types.BMFace`
Import/Create	Poor *(inflexible)*	Unusable *(read-only)*.	Best
Manipulate	Poor *(inflexible)*	Unusable *(read-only)*.	Best
Export/Output	Good *(n-gon support)*	Good *(When n-gons cannot be used)*	Good *(n-gons, extra memory overhead)*

Note

Using the `bmesh` API is completely separate API from `bpy`, typically you would use one or the other based on the level of editing needed, not simply for a different way to access faces.

Creating

All three data types can be used for face creation:

- Polygons are the most efficient way to create faces but the data structure is *very* rigid and inflexible, you must have all your vertices and faces ready a create them all at once. This is further complicated by the fact that each polygon does not store its own vertices, rather they reference an index and si in `bpy.types.Mesh.loops` which are a fixed array too.
- BMesh-faces are most likely the easiest way to create faces in new scripts, since faces can be added one by one and the API has features intended f mesh manipulation. While `bmesh.types.BMesh` uses more memory it can be managed by only operating on one mesh at a time.

Editing

Editing is where the three data types vary most.

- Polygons are very limited for editing, changing materials and options like smooth works, but for anything else they are too inflexible and are only intended for storage.
- Loop-triangles should not be used for editing geometry because doing so will cause existing n-gons to be tessellated.
- BMesh-faces are by far the best way to manipulate geometry.

Exporting

All three data types can be used for exporting, the choice mostly depends on whether the target format supports n-gons or not.

- Polygons are the most direct and efficient way to export providing they convert into the output format easily enough.
- Loop-triangles work well for exporting to formats which don't support n-gons, in fact this is the only place where their use is encouraged.
- BMesh-Faces can work for exporting too but may not be necessary if polygons can be used since using BMesh gives some overhead because it's no the native storage format in Object-Mode.

Copyright © Blender Authors
Made with Furo