Skip to content

Uv Operators

bpy.ops.uv.**align(*, axis='ALIGN_AUTO')**

Aligns selected UV vertices on a line

PARAMETERS:

axis (*enum in ['ALIGN_S', 'ALIGN_T', 'ALIGN_U', 'ALIGN_AUTO', 'ALIGN_X', 'ALIGN_Y'], (optional)*) –

Axis, Axis to align UV locations on

- `ALIGN_S` Straighten – Align UV vertices along the line defined by the endpoints.
- `ALIGN_T` Straighten X – Align UV vertices, moving them horizontally to the line defined by the endpoints.
- `ALIGN_U` Straighten Y – Align UV vertices, moving them vertically to the line defined by the endpoints.
- `ALIGN_AUTO` Align Auto – Automatically choose the direction on which there is most alignment already.
- `ALIGN_X` Align Vertically – Align UV vertices on a vertical line.
- `ALIGN_Y` Align Horizontally – Align UV vertices on a horizontal line.

bpy.ops.uv.**align_rotation(*, method='AUTO', axis='X', correct_aspect=False)**

Align the UV island's rotation

PARAMETERS:

- **method** (*enum in ['AUTO', 'EDGE', 'GEOMETRY'], (optional)*) –

  Method, Method to calculate rotation angle

  - `AUTO` Auto – Align from all edges.
  - `EDGE` Edge – Only selected edges.
  - `GEOMETRY` Geometry – Align to Geometry axis.

- **axis** (*enum in ['X', 'Y', 'Z'], (optional)*) –

  Axis, Axis to align to

  - `X` X – X axis.
  - `Y` Y – Y axis.
  - `Z` Z – Z axis.

- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Take image aspect ratio into account

FILE:

startup/bl_operators/uvcalc_transform.py:299

bpy.ops.uv.**average_islands_scale(*, scale_uv=False, shear=False)**

Average the size of separate UV islands, based on their area in 3D space

PARAMETERS:

- **scale_uv** (*boolean, (optional)*) – Non-Uniform, Scale U and V independently
- **shear** (*boolean, (optional)*) – Shear, Reduce shear within islands

bpy.ops.uv.**copy()**

Copy selected UV vertices

bpy.ops.uv.**cube_project(*, cube_size=1.0, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh over the six faces of a cube

PARAMETERS:

- **cube_size** (*float in [0, inf], (optional)*) – Cube Size, Size of the cube to project on
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun

- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**cursor_set(*, location=(0.0, 0.0))**

Set 2D cursor location

PARAMETERS:

location (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Cursor location in normalized (0.0 to 1.0) coordinates

bpy.ops.uv.**cylinder_project(*, direction='VIEW_ON_EQUATOR', align='POLAR_ZX', pole='PINCH', seam=False, radius=1.0, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh over the curved wall of a cylinder

PARAMETERS:

- **direction** (*enum in ['VIEW_ON_EQUATOR', 'VIEW_ON_POLES', 'ALIGN_TO_OBJECT'], (optional)*) –

  Direction, Direction of the sphere or cylinder

  - `VIEW_ON_EQUATOR` View on Equator – 3D view is on the equator.
  - `VIEW_ON_POLES` View on Poles – 3D view is on the poles.
  - `ALIGN_TO_OBJECT` Align to Object – Align according to object transform.
- **align** (*enum in ['POLAR_ZX', 'POLAR_ZY'], (optional)*) –

  Align, How to determine rotation around the pole

  - `POLAR_ZX` Polar ZX – Polar 0 is X.
  - `POLAR_ZY` Polar ZY – Polar 0 is Y.
- **pole** (*enum in ['PINCH', 'FAN'], (optional)*) –

  Pole, How to handle faces at the poles

  - `PINCH` Pinch – UVs are pinched at the poles.
  - `FAN` Fan – UVs are fanned at the poles.
- **seam** (*boolean, (optional)*) – Preserve Seams, Separate projections by islands isolated by seams
- **radius** (*float in [0, inf], (optional)*) – Radius, Radius of the sphere or cylinder
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**export_layout(*, filepath='', export_all=False, export_tiles='NONE', modified=False, mode='PNG', size=(1024, 1024), opacity=0.25, check_existing=True)**

Export UV layout to file

PARAMETERS:

- **filepath** (*string, (optional, never None)*) – filepath
- **export_all** (*boolean, (optional)*) – All UVs, Export all UVs in this mesh (not just visible ones)
- **export_tiles** (*enum in ['NONE', 'UDIM', 'UV'], (optional)*) –

  Export Tiles, Choose whether to export only the [0, 1] range, or all UV tiles

  - `NONE` None – Export only UVs in the [0, 1] range.
  - `UDIM` UDIM – Export tiles in the UDIM numbering scheme: 1001 + u_tile + 10*v_tile.
  - `UV` UVTILE – Export tiles in the UVTILE numbering scheme: u(u_tile + 1)_v(v_tile + 1).
- **modified** (*boolean, (optional)*) – Modified, Exports UVs from the modified mesh
- **mode** (*enum in ['SVG', 'EPS', 'PNG'], (optional)*) –

  Format, File format to export the UV layout to

  - `SVG` Scalable Vector Graphic (.svg) – Export the UV layout to a vector SVG file.

    - `EPS` Encapsulated PostScript (.eps) – Export the UV layout to a vector EPS file.
    - `PNG` PNG Image (.png) – Export the UV layout to a bitmap image.

- **size** (*int array of 2 items in [8, 32768], (optional)*) – Size, Dimensions of the exported file
- **opacity** (*float in [0, 1], (optional)*) – Fill Opacity, Set amount of opacity for exported UV layout
- **check_existing** (*boolean, (optional)*) – check_existing

FILE:
addons_core/io_mesh_uv_layout/__init__.py:137

bpy.ops.uv.**follow_active_quads(*, mode='LENGTH_AVERAGE')**

Follow UVs from active quads along continuous face loops

PARAMETERS:

mode (*enum in ['EVEN', 'LENGTH', 'LENGTH_AVERAGE'], (optional)*) –

Edge Length Mode, Method to space UV edge loops

- `EVEN` Even – Space all UVs evenly.
- `LENGTH` Length – Average space UVs edge length of each loop.
- `LENGTH_AVERAGE` Length Average – Average space UVs edge length of each loop.

FILE:
startup/bl_operators/uvcalc_follow_active.py:297

bpy.ops.uv.**hide(*, unselected=False)**

Hide (un)selected UV vertices

PARAMETERS:

unselected (*boolean, (optional)*) – Unselected, Hide unselected rather than selected

bpy.ops.uv.**lightmap_pack(*, PREF_CONTEXT='SEL_FACES', PREF_PACK_IN_ONE=True, PREF_NEW_UVLAYER=False, PREF_BOX_DIV=12, PREF_MARGIN_DIV=0.1)**

Pack each face’s UVs into the UV bounds

PARAMETERS:

- **PREF_CONTEXT** (*enum in ['SEL_FACES', 'ALL_FACES'], (optional)*) –

  Selection

    - `SEL_FACES` Selected Faces – Space all UVs evenly.
    - `ALL_FACES` All Faces – Average space UVs edge length of each loop.

- **PREF_PACK_IN_ONE** (*boolean, (optional)*) – Share Texture Space, Objects share texture space, map all objects into a single UV map
- **PREF_NEW_UVLAYER** (*boolean, (optional)*) – New UV Map, Create a new UV map for every mesh packed
- **PREF_BOX_DIV** (*int in [1, 48], (optional)*) – Pack Quality, Quality of the packing. Higher values will be slower but waste less space
- **PREF_MARGIN_DIV** (*float in [0.001, 1], (optional)*) – Margin, Size of the margin as a division of the UV

FILE:
startup/bl_operators/uvcalc_lightmap.py:662

bpy.ops.uv.**mark_seam(*, clear=False)**

Mark selected UV edges as seams

PARAMETERS:

clear (*boolean, (optional)*) – Clear Seams, Clear instead of marking seams

bpy.ops.uv.**minimize_stretch(*, fill_holes=True, blend=0.0, iterations=0)**

Reduce UV stretching by relaxing angles

PARAMETERS

PARAMETERS:

- **fill_holes** (*boolean, (optional)*) – Fill Holes, Virtually fill holes in mesh before unwrapping, to better avoid overlaps and preserve symmetry
- **blend** (*float in [0, 1], (optional)*) – Blend, Blend factor between stretch minimized and original
- **iterations** (*int in [0, inf], (optional)*) – Iterations, Number of iterations to run, 0 is unlimited when run interactively

bpy.ops.uv.**pack_islands(*, udim_source='CLOSEST_UDIM', rotate=True, rotate_method='ANY', scale=True, merge_overlap=False, margin_method='SCALED', margin=0.001, pin=False, pin_method='LOCKED', shape_method='CONCAVE')**

Transform all islands so that they fill up the UV/UDIM space as much as possible

PARAMETERS:

- **udim_source** (*enum in ['CLOSEST_UDIM', 'ACTIVE_UDIM', 'ORIGINAL_AABB'], (optional)*) –
  Pack to
  - `CLOSEST_UDIM` Closest UDIM – Pack islands to closest UDIM.
  - `ACTIVE_UDIM` Active UDIM – Pack islands to active UDIM image tile or UDIM grid tile where 2D cursor is located.
  - `ORIGINAL_AABB` Original bounding box – Pack to starting bounding box of islands.
- **rotate** (*boolean, (optional)*) – Rotate, Rotate islands to improve layout
- **rotate_method** (*enum in ['ANY', 'CARDINAL', 'AXIS_ALIGNED', 'AXIS_ALIGNED_X', 'AXIS_ALIGNED_Y'], (optional)*) –
  Rotation Method
  - `ANY` Any – Any angle is allowed for rotation.
  - `CARDINAL` Cardinal – Only 90 degree rotations are allowed.
  - `AXIS_ALIGNED` Axis-aligned – Rotated to a minimal rectangle, either vertical or horizontal.
  - `AXIS_ALIGNED_X` Axis-aligned (Horizontal) – Rotate islands to be aligned horizontally.
  - `AXIS_ALIGNED_Y` Axis-aligned (Vertical) – Rotate islands to be aligned vertically.
- **scale** (*boolean, (optional)*) – Scale, Scale islands to fill unit square
- **merge_overlap** (*boolean, (optional)*) – Merge Overlapping, Overlapping islands stick together
- **margin_method** (*enum in ['SCALED', 'ADD', 'FRACTION'], (optional)*) –
  Margin Method
  - `SCALED` Scaled – Use scale of existing UVs to multiply margin.
  - `ADD` Add – Just add the margin, ignoring any UV scale.
  - `FRACTION` Fraction – Specify a precise fraction of final UV output.
- **margin** (*float in [0, 1], (optional)*) – Margin, Space between islands
- **pin** (*boolean, (optional)*) – Lock Pinned Islands, Constrain islands containing any pinned UV's
- **pin_method** (*enum in ['SCALE', 'ROTATION', 'ROTATION_SCALE', 'LOCKED'], (optional)*) –
  Pin Method
  - `SCALE` Scale – Pinned islands won't rescale.
  - `ROTATION` Rotation – Pinned islands won't rotate.
  - `ROTATION_SCALE` Rotation and Scale – Pinned islands will translate only.
  - `LOCKED` All – Pinned islands are locked in place.
- **shape_method** (*enum in ['CONCAVE', 'CONVEX', 'AABB'], (optional)*) –
  Shape Method
  - `CONCAVE` Exact Shape (Concave) – Uses exact geometry.
  - `CONVEX` Boundary Shape (Convex) – Uses convex hull.
  - `AABB` Bounding Box – Uses bounding boxes.

bpy.ops.uv.**paste()**

Paste selected UV vertices

bpy.ops.uv.**pin(*, clear=False, invert=False)**

Set/clear selected UV vertices as anchored between multiple unwrap operations

PARAMETERS:

- **clear** (*boolean, (optional)*) – Clear, Clear pinning for the selection instead of setting it
- **invert** (*boolean, (optional)*) – Invert, Invert pinning for the selection instead of setting it

bpy.ops.uv.**project_from_view(*, orthographic=False, camera_bounds=True, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh as seen in current 3D view

PARAMETERS:

- **orthographic** (*boolean, (optional)*) – Orthographic, Use orthographic projection
- **camera_bounds** (*boolean, (optional)*) – Camera Bounds, Map UVs to the camera region taking resolution and aspect into account
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**randomize_uv_transform(*, random_seed=0, use_loc=True, loc=(0.0, 0.0), use_rot=True, rot=0.0, use_scale=True, scale_even=False, scale=(1.0, 1.0))**

Randomize the UV island's location, rotation, and scale

PARAMETERS:

- **random_seed** (*int in [0, 10000], (optional)*) – Random Seed, Seed value for the random generator
- **use_loc** (*boolean, (optional)*) – Randomize Location, Randomize the location values
- **loc** (`mathutils.Vector` of 2 items in [-100, 100], (optional)) – Location, Maximum distance the objects can spread over each axis
- **use_rot** (*boolean, (optional)*) – Randomize Rotation, Randomize the rotation value
- **rot** (*float in [-6.28319, 6.28319], (optional)*) – Rotation, Maximum rotation
- **use_scale** (*boolean, (optional)*) – Randomize Scale, Randomize the scale values
- **scale_even** (*boolean, (optional)*) – Scale Even, Use the same scale value for both axes
- **scale** (*float array of 2 items in [-100, 100], (optional)*) – Scale, Maximum scale randomization over each axis

FILE:

startup/bl_operators/uvcalc_transform.py:473

bpy.ops.uv.**remove_doubles(*, threshold=0.02, use_unselected=False, use_shared_vertex=False)**

Selected UV vertices that are within a radius of each other are welded together

PARAMETERS:

- **threshold** (*float in [0, 10], (optional)*) – Merge Distance, Maximum distance between welded vertices
- **use_unselected** (*boolean, (optional)*) – Unselected, Merge selected to other unselected vertices
- **use_shared_vertex** (*boolean, (optional)*) – Shared Vertex, Weld UVs based on shared vertices

bpy.ops.uv.**reset()**

Reset UV projection

bpy.ops.uv.**reveal(*, select=True)**

Reveal all hidden UV vertices

PARAMETERS:

select (*boolean, (optional)*) – Select

bpy.ops.uv.**rip(*, mirror=False, release_confirm=False, use_accurate=False, location=(0.0, 0.0))**

Rip selected vertices or a selected region

PARAMETERS:

- **mirror** (*boolean, (optional)*) – Mirror Editing
- **release_confirm** (*boolean, (optional)*) – Confirm on Release, Always confirm operation when releasing button
- **use_accurate** (*boolean, (optional)*) – Accurate, Use accurate transformation
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**rip_move(*, UV_OT_rip=None, TRANSFORM_OT_translate=None)**

Unstitch UVs and move the result

PARAMETERS:

- **UV_OT_rip** (`UV_OT_rip`, (optional)) – UV Rip, Rip selected vertices or a selected region
- **TRANSFORM_OT_translate** (`TRANSFORM_OT_translate`, (optional)) – Move, Move selected items

bpy.ops.uv.**seams_from_islands(*, mark_seams=True, mark_sharp=False)**

Set mesh seams according to island setup in the UV editor

PARAMETERS:

- **mark_seams** (*boolean, (optional)*) – Mark Seams, Mark boundary edges as seams
- **mark_sharp** (*boolean, (optional)*) – Mark Sharp, Mark boundary edges as sharp

bpy.ops.uv.**select(*, extend=False, deselect=False, toggle=False, deselect_all=False, select_passthrough=False, location=(0.0, 0.0))**

Select UV vertices

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection instead of deselecting everything first
- **deselect** (*boolean, (optional)*) – Deselect, Remove from selection
- **toggle** (*boolean, (optional)*) – Toggle Selection, Toggle the selection
- **deselect_all** (*boolean, (optional)*) – Deselect On Nothing, Deselect all when nothing under the cursor
- **select_passthrough** (*boolean, (optional)*) – Only Select Unselected, Ignore the select action when the element is already selected
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_all(*, action='TOGGLE')**

Change selection of all UV vertices

PARAMETERS:

action (*enum in ['TOGGLE', 'SELECT', 'DESELECT', 'INVERT'], (optional)*) –

Action, Selection action to execute

- `TOGGLE` Toggle – Toggle selection for all elements.
- `SELECT` Select – Select all elements.
- `DESELECT` Deselect – Deselect all elements.
- `INVERT` Invert – Invert selection of all elements.

bpy.ops.uv.**select_box(*, pinned=False, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, mode='SET')**

Select UV vertices using box selection

PARAMETERS:

- **pinned** (*boolean, (optional)*) – Pinned, Border select pinned UVs only
- **xmin** (*int in [-inf, inf], (optional)*) – X Min
- **xmax** (*int in [-inf, inf], (optional)*) – X Max
- **ymin** (*int in [-inf, inf], (optional)*) – Y Min
- **ymax** (*int in [-inf, inf], (optional)*) – Y Max
- **wait_for_input** (*boolean, (optional)*) – Wait for Input

- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –

  Mode

  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.uv.**select_circle(*, x=0, y=0, radius=25, wait_for_input=True, mode='SET')**

Select UV vertices using circle selection

PARAMETERS:

- **x** (*int in [-inf, inf], (optional)*) – X
- **y** (*int in [-inf, inf], (optional)*) – Y
- **radius** (*int in [1, inf], (optional)*) – Radius
- **wait_for_input** (*boolean, (optional)*) – Wait for Input
- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –

  Mode

  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.uv.**select_edge_ring(*, extend=False, location=(0.0, 0.0))**

Select an edge ring of connected UV vertices

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_lasso(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, mode='SET')**

Select UVs using lasso selection

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_smooth_stroke** (*boolean, (optional)*) – Stabilize Stroke, Selection lags behind mouse and follows a smoother path
- **smooth_stroke_factor** (*float in [0.5, 0.99], (optional)*) – Smooth Stroke Factor, Higher values gives a smoother stroke
- **smooth_stroke_radius** (*int in [10, 200], (optional)*) – Smooth Stroke Radius, Minimum distance from last point before selection continues
- **mode** (*enum in ['SET', 'ADD', 'SUB'], (optional)*) –

  Mode

  - `SET` Set – Set a new selection.
  - `ADD` Extend – Extend existing selection.
  - `SUB` Subtract – Subtract existing selection.

bpy.ops.uv.**select_less()**

Deselect UV vertices at the boundary of each selection region

bpy.ops.uv.**select_linked()**

Select all UV vertices linked to the active UV map

bpy.ops.uv.**select_linked_pick(*, extend=False, deselect=False, location=(0.0, 0.0))**

Select all UV vertices linked under the mouse

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection
- **deselect** (*boolean, (optional)*) – Deselect, Deselect linked UV vertices rather than selecting them
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_loop(*, extend=False, location=(0.0, 0.0))**

Select a loop of connected UV vertices

PARAMETERS:

- **extend** (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection
- **location** (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Location, Mouse location in normalized coordinates, 0.0 to 1.0 is within the image bounds

bpy.ops.uv.**select_mode(*, type='VERTEX')**

Change UV selection mode

PARAMETERS:

type (enum in Mesh Select Mode Uv Items, (optional)) – Type

bpy.ops.uv.**select_more()**

Select more UV vertices connected to initial selection

bpy.ops.uv.**select_overlap(*, extend=False)**

Select all UV faces which overlap each other

PARAMETERS:

extend (*boolean, (optional)*) – Extend, Extend selection rather than clearing the existing selection

bpy.ops.uv.**select_pinned()**

Select all pinned UV vertices

bpy.ops.uv.**select_similar(*, type='PIN', compare='EQUAL', threshold=0.0)**

Select similar UVs by property types

PARAMETERS:

- **type** (*enum in ['PIN', 'LENGTH', 'LENGTH_3D', 'AREA', 'AREA_3D', 'MATERIAL', 'OBJECT', 'SIDES', 'WINDING', 'FACE'], (optional)*) – Type
- **compare** (*enum in ['EQUAL', 'GREATER', 'LESS'], (optional)*) – Compare
- **threshold** (*float in [0, 1], (optional)*) – Threshold

bpy.ops.uv.**select_split()**

Select only entirely selected faces

bpy.ops.uv.**shortest_path_pick(*, use_face_step=False, use_topology_distance=False, use_fill=False, skip=0, nth=1, offset=0, object_index=-1, index=-1)**

Select shortest path between two selections

PARAMETERS:

- **use_face_step** (*boolean, (optional)*) – Face Stepping, Traverse connected faces (includes diagonals and edge-rings)
- **use_topology_distance** (*boolean, (optional)*) – Topology Distance, Find the minimum number of steps, ignoring spatial distance
- **use_fill** (*boolean, (optional)*) – Fill Region, Select all paths between the source/destination elements
- **skip** (*int in [0, inf], (optional)*) – Deselected, Number of deselected elements in the repetitive sequence
- **nth** (*int in [1, inf], (optional)*) – Selected, Number of selected elements in the repetitive sequence
- **offset** (*int in [-inf, inf], (optional)*) – Offset, Offset from the starting point

bpy.ops.uv.**shortest_path_select(*, use_face_step=False, use_topology_distance=False, use_fill=False, skip=0, nth=1, offset=0)**

Selected shortest path between two vertices/edges/faces

PARAMETERS:

- **use_face_step** (*boolean, (optional)*) – Face Stepping, Traverse connected faces (includes diagonals and edge-rings)
- **use_topology_distance** (*boolean, (optional)*) – Topology Distance, Find the minimum number of steps, ignoring spatial distance
- **use_fill** (*boolean, (optional)*) – Fill Region, Select all paths between the source/destination elements
- **skip** (*int in [0, inf], (optional)*) – Deselected, Number of deselected elements in the repetitive sequence
- **nth** (*int in [1, inf], (optional)*) – Selected, Number of selected elements in the repetitive sequence
- **offset** (*int in [-inf, inf], (optional)*) – Offset, Offset from the starting point

bpy.ops.uv.**smart_project(*, angle_limit=1.15192, margin_method='SCALED', rotate_method='AXIS_ALIGNED_Y', island_margin=0.0, area_weight=0.0, correct_aspect=True, scale_to_bounds=False)**

Projection unwraps the selected faces of mesh objects

PARAMETERS:

- **angle_limit** (*float in [0, 1.5708], (optional)*) – Angle Limit, Lower for more projection groups, higher for less distortion
- **margin_method** (*enum in ['SCALED', 'ADD', 'FRACTION'], (optional)*) –
  Margin Method
  - `SCALED` Scaled – Use scale of existing UVs to multiply margin.
  - `ADD` Add – Just add the margin, ignoring any UV scale.
  - `FRACTION` Fraction – Specify a precise fraction of final UV output.
- **rotate_method** (*enum in ['AXIS_ALIGNED', 'AXIS_ALIGNED_X', 'AXIS_ALIGNED_Y'], (optional)*) –
  Rotation Method
  - `AXIS_ALIGNED` Axis-aligned – Rotated to a minimal rectangle, either vertical or horizontal.
  - `AXIS_ALIGNED_X` Axis-aligned (Horizontal) – Rotate islands to be aligned horizontally.
  - `AXIS_ALIGNED_Y` Axis-aligned (Vertical) – Rotate islands to be aligned vertically.
- **island_margin** (*float in [0, 1], (optional)*) – Island Margin, Margin to reduce bleed from adjacent islands
- **area_weight** (*float in [0, 1], (optional)*) – Area Weight, Weight projection's vector by faces with larger areas
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into account
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**snap_cursor(*, target='PIXELS')**

Snap cursor to target type

PARAMETERS:

target (*enum in ['PIXELS', 'SELECTED', 'ORIGIN'], (optional)*) – Target, Target to snap the selected UVs to

bpy.ops.uv.**snap_selected(*, target='PIXELS')**

Snap selected UV vertices to target type

PARAMETERS:

target (*enum in ['PIXELS', 'CURSOR', 'CURSOR_OFFSET', 'ADJACENT_UNSELECTED'], (optional)*) – Target, Target to snap the selected UVs to

bpy.ops.uv.**sphere_project(*, direction='VIEW_ON_EQUATOR', align='POLAR_ZX', pole='PINCH', seam=False, correct_aspect=True, clip_to_bounds=False, scale_to_bounds=False)**

Project the UV vertices of the mesh over the curved surface of a sphere

PARAMETERS:

- **direction** (*enum in ['VIEW_ON_EQUATOR', 'VIEW_ON_POLES', 'ALIGN_TO_OBJECT'], (optional)*) –
  Direction, Direction of the sphere or cylinder

- `VIEW_ON_EQUATOR` View on Equator – 3D view is on the equator.
- `VIEW_ON_POLES` View on Poles – 3D view is on the poles.
- `ALIGN_TO_OBJECT` Align to Object – Align according to object transform.

- **align** (*enum in ['POLAR_ZX', 'POLAR_ZY'], (optional)*) –
  Align, How to determine rotation around the pole
  - `POLAR_ZX` Polar ZX – Polar 0 is X.
  - `POLAR_ZY` Polar ZY – Polar 0 is Y.
- **pole** (*enum in ['PINCH', 'FAN'], (optional)*) –
  Pole, How to handle faces at the poles
  - `PINCH` Pinch – UVs are pinched at the poles.
  - `FAN` Fan – UVs are fanned at the poles.
- **seam** (*boolean, (optional)*) – Preserve Seams, Separate projections by islands isolated by seams
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **clip_to_bounds** (*boolean, (optional)*) – Clip to Bounds, Clip UV coordinates to bounds after unwrapping
- **scale_to_bounds** (*boolean, (optional)*) – Scale to Bounds, Scale UV coordinates to bounds after unwrapping

bpy.ops.uv.**stitch(*, use_limit=False, snap_islands=True, limit=0.01, static_island=0, active_object_index=0, midpoint_snap=False, clear_seams=True, mode='VERTEX', stored_mode='VERTEX', selection=None, objects_selection_count=(0, 0, 0, 0, 0, 0))**

Stitch selected UV vertices by proximity

PARAMETERS:

- **use_limit** (*boolean, (optional)*) – Use Limit, Stitch UVs within a specified limit distance
- **snap_islands** (*boolean, (optional)*) – Snap Islands, Snap islands together (on edge stitch mode, rotates the islands too)
- **limit** (*float in [0, inf], (optional)*) – Limit, Limit distance in normalized coordinates
- **static_island** (*int in [0, inf], (optional)*) – Static Island, Island that stays in place when stitching islands
- **active_object_index** (*int in [0, inf], (optional)*) – Active Object, Index of the active object
- **midpoint_snap** (*boolean, (optional)*) – Snap at Midpoint, UVs are stitched at midpoint instead of at static island
- **clear_seams** (*boolean, (optional)*) – Clear Seams, Clear seams of stitched edges
- **mode** (*enum in ['VERTEX', 'EDGE'], (optional)*) – Operation Mode, Use vertex or edge stitching
- **stored_mode** (*enum in ['VERTEX', 'EDGE'], (optional)*) – Stored Operation Mode, Use vertex or edge stitching
- **selection** (`bpy_prop_collection` of `SelectedUvElement`, (optional)) – Selection
- **objects_selection_count** (*int array of 6 items in [0, inf], (optional)*) – Objects Selection Count

bpy.ops.uv.**unwrap(*, method='CONFORMAL', fill_holes=False, correct_aspect=True, use_subsurf_data=False, margin_method='SCALED', margin=0.001, no_flip=False, iterations=10, use_weights=False, weight_group='uv_importance', weight_factor=1.0)**

Unwrap the mesh of the object being edited

PARAMETERS:

- **method** (*enum in ['ANGLE_BASED', 'CONFORMAL', 'MINIMUM_STRETCH'], (optional)*) – Method, Unwrapping method (Angle Based usually gives better results than Conformal, while being somewhat slower)
- **fill_holes** (*boolean, (optional)*) – Fill Holes, Virtually fill holes in mesh before unwrapping, to better avoid overlaps and preserve symmetry
- **correct_aspect** (*boolean, (optional)*) – Correct Aspect, Map UVs taking aspect ratio of the image associated with the material into accoun
- **use_subsurf_data** (*boolean, (optional)*) – Use Subdivision Surface, Map UVs taking vertex position after Subdivision Surface modifier has been applied
- **margin_method** (*enum in ['SCALED', 'ADD', 'FRACTION'], (optional)*) –
  Margin Method
  - `SCALED` Scaled – Use scale of existing UVs to multiply margin.
  - `ADD` Add – Just add the margin, ignoring any UV scale.

    - `FRACTION` Fraction – Specify a precise fraction of final UV output.
- **margin** (*float in [0, 1], (optional)*) – Margin, Space between islands
- **no_flip** (*boolean, (optional)*) – No Flip, Prevent flipping UV’s, flipping may lower distortion depending on the position of pins
- **iterations** (*int in [0, 10000], (optional)*) – Iterations, Number of iterations when “Minimum Stretch” method is used
- **use_weights** (*boolean, (optional)*) – Importance Weights, Whether to take into account per-vertex importance weights
- **weight_group** (*string, (optional, never None)*) – Weight Group, Vertex group name for importance weights (modulating the deform)
- **weight_factor** (*float in [-10000, 10000], (optional)*) – Weight Factor, How much influence the weightmap has for weighted parameterization, 0 being no influence

bpy.ops.uv.**weld()**

Weld selected UV vertices together