Paint Operators

bpy.ops.paint.**add_simple_uvs()**

Add cube map UVs on mesh

bpy.ops.paint.**add_texture_paint_slot(*, type='BASE_COLOR', slot_type='IMAGE', name='Untitled', color=(0.0, 0.0, 0.0, 1.0), width=102 height=1024, alpha=True, generated_type='BLANK', float=False, domain='POINT', data_type='FLOAT_COLOR')**

Add a paint slot

PARAMETERS:

- **type** (*enum in ['BASE_COLOR', 'SPECULAR', 'ROUGHNESS', 'METALLIC', 'NORMAL', 'BUMP', 'DISPLACEMENT'], (optional)*) – Material Layer Type, Material layer type of new paint slot
- **slot_type** (*enum in ['IMAGE', 'COLOR_ATTRIBUTE'], (optional)*) – Slot Type, Type of new paint slot
- **name** (*string, (optional, never None)*) – Name, Name for new paint slot source
- **color** (*float array of 4 items in [0, inf], (optional)*) – Color, Default fill color
- **width** (*int in [1, inf], (optional)*) – Width, Image width
- **height** (*int in [1, inf], (optional)*) – Height, Image height
- **alpha** (*boolean, (optional)*) – Alpha, Create an image with an alpha channel
- **generated_type** (enum in Image Generated Type Items, (optional)) – Generated Type, Fill the image with a grid for UV map testing
- **float** (*boolean, (optional)*) – 32-bit Float, Create image with 32-bit floating-point bit depth
- **domain** (enum in Color Attribute Domain Items, (optional)) – Domain, Type of element that attribute is stored on
- **data_type** (enum in Color Attribute Type Items, (optional)) – Data Type, Type of data stored in attribute

bpy.ops.paint.**brush_colors_flip()**

Swap primary and secondary brush colors

bpy.ops.paint.**face_select_all(*, action='TOGGLE')**

Change selection for all faces

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.paint.**face_select_hide(*, unselected=False)**

Hide selected faces

PARAMETERS:

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

bpy.ops.paint.**face_select_less(*, face_step=True)**

Deselect Faces connected to existing selection

PARAMETERS:

face_step (*boolean, (optional)*) – Face Step, Also deselect faces that only touch on a corner

bpy.ops.paint.**face_select_linked()**

Select linked faces

bpy.ops.paint.**face_select_linked_pick(*, deselect=False)**

Select linked faces under the cursor

PARAMETERS:

deselect (*boolean, (optional)*) – Deselect, Deselect rather than select items

bpy.ops.paint.**face_select_loop(*, select=True, extend=False)**

Select face loop under the cursor

PARAMETERS:

- **select** (*boolean, (optional)*) – Select, If false, faces will be deselected
- **extend** (*boolean, (optional)*) – Extend, Extend the selection

bpy.ops.paint.**face_select_more(*, face_step=True)**

Select Faces connected to existing selection

PARAMETERS:

face_step (*boolean, (optional)*) – Face Step, Also select faces that only touch on a corner

bpy.ops.paint.**face_vert_reveal(*, select=True)**

Reveal hidden faces and vertices

PARAMETERS:

select (*boolean, (optional)*) – Select, Specifies whether the newly revealed geometry should be selected

bpy.ops.paint.**grab_clone(*, delta=(0.0, 0.0))**

Move the clone source image

PARAMETERS:

delta (`mathutils.Vector` of 2 items in [-inf, inf], (optional)) – Delta, Delta offset of clone image in 0.0 to 1.0 coordinates

bpy.ops.paint.**hide_show(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, action='HIDE', area='Inside', use_front_faces_only=False)**

Hide/show some vertices

PARAMETERS:

- **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
- **action** (*enum in ['HIDE', 'SHOW'], (optional)*) –

  Visibility Action, Whether to hide or show vertices

  - `HIDE` Hide – Hide vertices.
  - `SHOW` Show – Show vertices.
- **area** (*enum in ['OUTSIDE', 'Inside'], (optional)*) –

  Visibility Area, Which vertices to hide or show

  - `OUTSIDE` Outside – Hide or show vertices outside the selection.
  - `Inside` Inside – Hide or show vertices inside the selection.
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view

bpy.ops.paint.**hide_show_all(*, action='HIDE')**

Hide/show all vertices

PARAMETERS:

action (*enum in ['HIDE', 'SHOW'], (optional)*) –

Visibility Action, Whether to hide or show vertices

- `HIDE` Hide – Hide vertices.
- `SHOW` Show – Show vertices.

bpy.ops.paint.**hide_show_lasso_gesture(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, action='HIDE', area='Inside', use_front_faces_only=False)**

Hide/show some vertices

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
- **action** (*enum in ['HIDE', 'SHOW'], (optional)*) –

  Visibility Action, Whether to hide or show vertices

  - `HIDE` Hide – Hide vertices.
  - `SHOW` Show – Show vertices.
- **area** (*enum in ['OUTSIDE', 'Inside'], (optional)*) –

  Visibility Area, Which vertices to hide or show

  - `OUTSIDE` Outside – Hide or show vertices outside the selection.
  - `Inside` Inside – Hide or show vertices inside the selection.
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view

bpy.ops.paint.**hide_show_line_gesture(*, xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5, action='HIDE', area='Inside', use_front_faces_only=False, use_limit_to_segment=False)**

Hide/show some vertices

PARAMETERS:

- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator
- **action** (*enum in ['HIDE', 'SHOW'], (optional)*) –

  Visibility Action, Whether to hide or show vertices

  - `HIDE` Hide – Hide vertices.
  - `SHOW` Show – Show vertices.
- **area** (*enum in ['OUTSIDE', 'Inside'], (optional)*) –

  Visibility Area, Which vertices to hide or show

  - `OUTSIDE` Outside – Hide or show vertices outside the selection.
  - `Inside` Inside – Hide or show vertices inside the selection.
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **use_limit_to_segment** (*boolean, (optional)*) – Limit to Segment, Apply the gesture action only to the area that is contained within the segment without extending its effect to the entire line

bpy.ops.paint.**hide_show_masked(*, action='HIDE')**

Hide/show all masked vertices above a threshold

PARAMETERS:

action (*enum in ['HIDE', 'SHOW'], (optional)*) –

Visibility Action, Whether to hide or show vertices

- `HIDE` Hide – Hide vertices.
- `SHOW` Show – Show vertices.

bpy.ops.paint.**hide_show_polyline_gesture(*, path=None, action='HIDE', area='Inside', use_front_faces_only=False)**

Hide/show some vertices

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **action** (*enum in ['HIDE', 'SHOW'], (optional)*) –
  Visibility Action, Whether to hide or show vertices
  - `HIDE` Hide – Hide vertices.
  - `SHOW` Show – Show vertices.
- **area** (*enum in ['OUTSIDE', 'Inside'], (optional)*) –
  Visibility Area, Which vertices to hide or show
  - `OUTSIDE` Outside – Hide or show vertices outside the selection.
  - `Inside` Inside – Hide or show vertices inside the selection.
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view

bpy.ops.paint.**image_from_view(*, filepath='')**

Make an image from biggest 3D view for reprojection

PARAMETERS:

filepath (*string, (optional, never None)*) – File Path, Name of the file

bpy.ops.paint.**image_paint(*, stroke=None, mode='NORMAL', pen_flip=False)**

Paint a stroke into the image

PARAMETERS:

- **stroke** (`bpy_prop_collection` of `OperatorStrokeElement`, (optional)) – Stroke
- **mode** (*enum in ['NORMAL', 'INVERT', 'SMOOTH', 'ERASE'], (optional)*) –
  Stroke Mode, Action taken when a paint stroke is made
  - `NORMAL` Regular – Apply brush normally.
  - `INVERT` Invert – Invert action of brush for duration of stroke.
  - `SMOOTH` Smooth – Switch brush to smooth mode for duration of stroke.
  - `ERASE` Erase – Switch brush to erase mode for duration of stroke.
- **pen_flip** (*boolean, (optional)*) – Pen Flip, Whether a tablet's eraser mode is being used

bpy.ops.paint.**mask_box_gesture(*, xmin=0, xmax=0, ymin=0, ymax=0, wait_for_input=True, use_front_faces_only=False, mode='VALUE value=1.0)**

Mask within a rectangle defined by the cursor

PARAMETERS:

- **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
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **mode** (*enum in ['VALUE', 'VALUE_INVERSE', 'INVERT'], (optional)*) –
  Mode
  - `VALUE` Value – Set mask to the level specified by the 'value' property.
  - `VALUE_INVERSE` Value Inverted – Set mask to the level specified by the inverted 'value' property.
  - `INVERT` Invert – Invert the mask.
- **value** (*float in [0, 1], (optional)*) – Value, Mask level to use when mode is 'Value'; zero means no masking and one is fully masked

bpy.ops.paint.**mask_flood_fill(*, mode='VALUE', value=0.0)**

Fill the whole mask with a given value, or invert its values

PARAMETERS:

- **mode** (*enum in ['VALUE', 'VALUE_INVERSE', 'INVERT'], (optional)*) –
  Mode
  - `VALUE` Value – Set mask to the level specified by the 'value' property.
  - `VALUE_INVERSE` Value Inverted – Set mask to the level specified by the inverted 'value' property.
  - `INVERT` Invert – Invert the mask.
- **value** (*float in [0, 1], (optional)*) – Value, Mask level to use when mode is 'Value'; zero means no masking and one is fully masked

bpy.ops.paint.**mask_lasso_gesture(*, path=None, use_smooth_stroke=False, smooth_stroke_factor=0.75, smooth_stroke_radius=35, use_front_faces_only=False, mode='VALUE', value=1.0)**

Mask within a shape defined by the cursor

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
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **mode** (*enum in ['VALUE', 'VALUE_INVERSE', 'INVERT'], (optional)*) –
  Mode
  - `VALUE` Value – Set mask to the level specified by the 'value' property.
  - `VALUE_INVERSE` Value Inverted – Set mask to the level specified by the inverted 'value' property.
  - `INVERT` Invert – Invert the mask.
- **value** (*float in [0, 1], (optional)*) – Value, Mask level to use when mode is 'Value'; zero means no masking and one is fully masked

bpy.ops.paint.**mask_line_gesture(*, xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5, use_front_faces_only=False, use_limit_to_segment=False, mode='VALUE', value=1.0)**

Mask to one side of a line defined by the cursor

PARAMETERS:

- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator

- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **use_limit_to_segment** (*boolean, (optional)*) – Limit to Segment, Apply the gesture action only to the area that is contained within the segment without extending its effect to the entire line
- **mode** (*enum in ['VALUE', 'VALUE_INVERSE', 'INVERT'], (optional)*) –

  Mode

  - `VALUE` Value – Set mask to the level specified by the 'value' property.
  - `VALUE_INVERSE` Value Inverted – Set mask to the level specified by the inverted 'value' property.
  - `INVERT` Invert – Invert the mask.
- **value** (*float in [0, 1], (optional)*) – Value, Mask level to use when mode is 'Value'; zero means no masking and one is fully masked

bpy.ops.paint.**mask_polyline_gesture(*, path=None, use_front_faces_only=False, mode='VALUE', value=1.0)**

Mask within a shape defined by the cursor

PARAMETERS:

- **path** (`bpy_prop_collection` of `OperatorMousePath`, (optional)) – Path
- **use_front_faces_only** (*boolean, (optional)*) – Front Faces Only, Affect only faces facing towards the view
- **mode** (*enum in ['VALUE', 'VALUE_INVERSE', 'INVERT'], (optional)*) –

  Mode

  - `VALUE` Value – Set mask to the level specified by the 'value' property.
  - `VALUE_INVERSE` Value Inverted – Set mask to the level specified by the inverted 'value' property.
  - `INVERT` Invert – Invert the mask.
- **value** (*float in [0, 1], (optional)*) – Value, Mask level to use when mode is 'Value'; zero means no masking and one is fully masked

bpy.ops.paint.**project_image(*, image='')**

Project an edited render from the active camera back onto the object

PARAMETERS:

image (*enum in [], (optional)*) – Image

bpy.ops.paint.**sample_color(*, location=(0, 0), merged=False, palette=False)**

Use the mouse to sample a color in the image

PARAMETERS:

- **location** (*int array of 2 items in [0, inf], (optional)*) – Location
- **merged** (*boolean, (optional)*) – Sample Merged, Sample the output display color
- **palette** (*boolean, (optional)*) – Add to Palette

bpy.ops.paint.**texture_paint_toggle()**

Toggle texture paint mode in 3D view

bpy.ops.paint.**vert_select_all(*, action='TOGGLE')**

Change selection for all 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.paint.**vert_select_hide(*, unselected=False)**

Hide selected vertices

PARAMETERS:

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

bpy.ops.paint.**vert_select_less(*, face_step=True)**

Deselect Vertices connected to existing selection

PARAMETERS:

face_step (*boolean, (optional)*) – Face Step, Also deselect faces that only touch on a corner

bpy.ops.paint.**vert_select_linked()**

Select linked vertices

bpy.ops.paint.**vert_select_linked_pick(*, select=True)**

Select linked vertices under the cursor

PARAMETERS:

select (*boolean, (optional)*) – Select, Whether to select or deselect linked vertices under the cursor

bpy.ops.paint.**vert_select_more(*, face_step=True)**

Select Vertices connected to existing selection

PARAMETERS:

face_step (*boolean, (optional)*) – Face Step, Also select faces that only touch on a corner

bpy.ops.paint.**vert_select_ungrouped(*, extend=False)**

Select vertices without a group

PARAMETERS:

extend (*boolean, (optional)*) – Extend, Extend the selection

bpy.ops.paint.**vertex_color_brightness_contrast(*, brightness=0.0, contrast=0.0)**

Adjust vertex color brightness/contrast

PARAMETERS:

- **brightness** (*float in [-100, 100], (optional)*) – Brightness
- **contrast** (*float in [-100, 100], (optional)*) – Contrast

bpy.ops.paint.**vertex_color_dirt(*, blur_strength=1.0, blur_iterations=1, clean_angle=3.14159, dirt_angle=0.0, dirt_only=False, normalize=True)**

Generate a dirt map gradient based on cavity

PARAMETERS:

- **blur_strength** (*float in [0.01, 1], (optional)*) – Blur Strength, Blur strength per iteration
- **blur_iterations** (*int in [0, 40], (optional)*) – Blur Iterations, Number of times to blur the colors (higher blurs more)
- **clean_angle** (*float in [0, 3.14159], (optional)*) – Highlight Angle, Less than 90 limits the angle used in the tonal range
- **dirt_angle** (*float in [0, 3.14159], (optional)*) – Dirt Angle, Less than 90 limits the angle used in the tonal range
- **dirt_only** (*boolean, (optional)*) – Dirt Only, Don't calculate cleans for convex areas
- **normalize** (*boolean, (optional)*) – Normalize, Normalize the colors, increasing the contrast

FILE:

startup/bl_operators/vertexpaint_dirt.py:179

bpy.ops.paint.**vertex_color_from_weight()**

Convert active weight into gray scale vertex colors bpy.ops.paint.**vertex_color_hsv(*, h=0.5, s=1.0, v=1.0)**

Adjust vertex color Hue/Saturation/Value

PARAMETERS:

- **h** (*float in [0, 1], (optional)*) – Hue
- **s** (*float in [0, 2], (optional)*) – Saturation
- **v** (*float in [0, 2], (optional)*) – Value

bpy.ops.paint.**vertex_color_invert()**

Invert RGB values

bpy.ops.paint.**vertex_color_levels(*, offset=0.0, gain=1.0)**

Adjust levels of vertex colors

PARAMETERS:

- **offset** (*float in [-1, 1], (optional)*) – Offset, Value to add to colors
- **gain** (*float in [0, inf], (optional)*) – Gain, Value to multiply colors by

bpy.ops.paint.**vertex_color_set(*, use_alpha=True)**

Fill the active vertex color layer with the current paint color

PARAMETERS:

use_alpha (*boolean, (optional)*) – Affect Alpha, Set color completely opaque instead of reusing existing alpha

bpy.ops.paint.**vertex_color_smooth()**

Smooth colors across vertices

bpy.ops.paint.**vertex_paint(*, stroke=None, mode='NORMAL', pen_flip=False, override_location=False)**

Paint a stroke in the active color attribute layer

PARAMETERS:

- **stroke** (`bpy_prop_collection` of `OperatorStrokeElement`, (optional)) – Stroke
- **mode** (*enum in ['NORMAL', 'INVERT', 'SMOOTH', 'ERASE'], (optional)*) –
  Stroke Mode, Action taken when a paint stroke is made
  - `NORMAL` Regular – Apply brush normally.
  - `INVERT` Invert – Invert action of brush for duration of stroke.
  - `SMOOTH` Smooth – Switch brush to smooth mode for duration of stroke.
  - `ERASE` Erase – Switch brush to erase mode for duration of stroke.
- **pen_flip** (*boolean, (optional)*) – Pen Flip, Whether a tablet's eraser mode is being used
- **override_location** (*boolean, (optional)*) – Override Location, Override the given *location* array by recalculating object space positions fro
  the provided *mouse_event* positions

bpy.ops.paint.**vertex_paint_toggle()**

Toggle the vertex paint mode in 3D view

bpy.ops.paint.**visibility_filter(*, action='GROW', iterations=1, auto_iteration_count=True)**

Edit the visibility of the current mesh

PARAMETERS:

- **action** (*enum in ['GROW', 'SHRINK'], (optional)*) –
  Action
  - `GROW` Grow Visibility – Grow the visibility by one face based on mesh topology.
  - `SHRINK` Shrink Visibility – Shrink the visibility by one face based on mesh topology.

- **iterations** (*int in [1, 100], (optional)*) – Iterations, Number of times that the filter is going to be applied
- **auto_iteration_count** (*boolean, (optional)*) – Auto Iteration Count, Use an automatic number of iterations based on the number of vertices the sculpt

bpy.ops.paint.**visibility_invert()**

Invert the visibility of all vertices

bpy.ops.paint.**weight_from_bones(*, type='AUTOMATIC')**

Set the weights of the groups matching the attached armature's selected bones, using the distance between the vertices and the bones

PARAMETERS:

type (*enum in ['AUTOMATIC', 'ENVELOPES'], (optional)*) –

Type, Method to use for assigning weights

- `AUTOMATIC` Automatic – Automatic weights from bones.
- `ENVELOPES` From Envelopes – Weights from envelopes with user defined radius.

bpy.ops.paint.**weight_gradient(*, type='LINEAR', xstart=0, xend=0, ystart=0, yend=0, flip=False, cursor=5)**

Draw a line to apply a weight gradient to selected vertices

PARAMETERS:

- **type** (*enum in ['LINEAR', 'RADIAL'], (optional)*) – Type
- **xstart** (*int in [-inf, inf], (optional)*) – X Start
- **xend** (*int in [-inf, inf], (optional)*) – X End
- **ystart** (*int in [-inf, inf], (optional)*) – Y Start
- **yend** (*int in [-inf, inf], (optional)*) – Y End
- **flip** (*boolean, (optional)*) – Flip
- **cursor** (*int in [0, inf], (optional)*) – Cursor, Mouse cursor style to use during the modal operator

bpy.ops.paint.**weight_paint(*, stroke=None, mode='NORMAL', pen_flip=False, override_location=False)**

Paint a stroke in the current vertex group's weights

PARAMETERS:

- **stroke** (`bpy_prop_collection` of `OperatorStrokeElement`, (optional)) – Stroke
- **mode** (*enum in ['NORMAL', 'INVERT', 'SMOOTH', 'ERASE'], (optional)*) –

  Stroke Mode, Action taken when a paint stroke is made

  - `NORMAL` Regular – Apply brush normally.
  - `INVERT` Invert – Invert action of brush for duration of stroke.
  - `SMOOTH` Smooth – Switch brush to smooth mode for duration of stroke.
  - `ERASE` Erase – Switch brush to erase mode for duration of stroke.
- **pen_flip** (*boolean, (optional)*) – Pen Flip, Whether a tablet's eraser mode is being used
- **override_location** (*boolean, (optional)*) – Override Location, Override the given *location* array by recalculating object space positions fro the provided *mouse_event* positions

bpy.ops.paint.**weight_paint_toggle()**

Toggle weight paint mode in 3D view

bpy.ops.paint.**weight_sample()**

Use the mouse to sample a weight in the 3D view

bpy.ops.paint.**weight_sample_group()**

Select one of the vertex groups available under current mouse position bpy.ops.paint.**weight_set()**

Fill the active vertex group with the current paint weight