Python in the VE

VE Scripts – working with geometry

What? In VE Python it is sensible to limit room geometry adjustments to changes in % windows/doors/openings and dimensional adjustments to shading objects. This script demonstrates adjusting these elements using some generic functions that can also be then used in iteration.

Why? Moving space surfaces is quite likely to lead to over-lapping spaces and geometry errors; similarly you can move openings on a surface but again this can lead to geometry errors. As shading objects can over-lap with other objects without causing geometry errors adjusting shade surfaces will not lead to such errors.

```python
"""
==============================================
Model geometry - glazing percent and shade size
==============================================

Module description
------------------
Demonstration of adjusting model glazing percent
Demonstration of adjusting body glazing percent
Demonstration of adjusting local shade dimensions
"""

import iesve
import time

def reduce_model_glazing_percent(percent):
    """
    Reduces total model glazing to the percent parameter
    The method reduce_ext_windows(percent) reduces the existing glazing proportionately

    Parameters
    ----------
    percent : required glazing percent int

    Notes
    -----
    Limitation: percent parameter is >= existing WWR %

    """

    geom = iesve.VEGeometry
    old_percent = geom.get_wwr()
    if percent >= old_percent:
        print('Cannot reduce as requested percent is >= than existing WWR')
        return
    geom.reduce_ext_windows(percent)
    print('Reducing WWR for model from: ', old_percent, ' % to: ', percent, ' %')


def change_body_glazing_percent(bodies, percent):
    """
    Adjusts room body glazing percent
    Removes existing glazing and applies new glazing

    Parameters
    ----------
    bodies : list of model bodies
    percent : required glazing percent int

    """

    geom = iesve.VEGeometry
    print('Replacing glazing for selected bodies to: ', percent, ' %')
    for body in bodies:
        if body.type == iesve.VEBody_type.room:
            body.select()
    geom.remove_openings_below_area_threshold(1000)
    geom.set_percent_wall_glazing(percent)


def change_shade_size(bodies, overhang_change, depth_change):
    """
    Adjusts existing shade size for local shades
    Checks shade body name to check for an assigned elevation north, south, east, west
    Surface orientation is compass north, south, east, west +-45 degrees

    Parameters
    ----------
    shades : list of model bodies
    overhang_change : left and right overhang size change (+ is an increase) m
    depth_change : depth size change (+ is an increase) m

    Notes
    -----
    Limitation: surfaces at exactly the test orientation boundaries (os set to XYZ.01)

    """

    for body in bodies:
        # Check if body is a local shade
        if body.type == iesve.VEBody_type.local_shade:
            # Get body name
            body_object = body.get_room_data(type = iesve.attribute_type.real_attributes)
            body_data = body_object.get_general()

            # Check NSEW via name and adjust 3 unrestrained shade sides (walls) only
            surfaces = body.get_surfaces()
            if 'north' in body_data['name']:
                for surface in surfaces:
                    properties = surface.get_properties()
                    if properties['type'] == 'Wall' and properties['orientation'] > 45.01 and properties['orientation'] <= 135.01:
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 225.01 and properties['orientation'] <= 315.01:
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and (properties['orientation'] > 315.01 or properties['orientation'] <= 45.01):
                        surface.move(depth_change)
            if 'south' in body_data['name']:
                for surface in surfaces:
                    properties = surface.get_properties()
                    if properties['type'] == 'Wall' and properties['orientation'] > 45.01 and properties['orientation'] <= 135.01:
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 225.01 and properties['orientation'] <= 315.01:
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 135.01 and properties['orientation'] <= 225.01:
                        surface.move(depth_change)
            if 'east' in body_data['name']:
                for surface in surfaces:
                    properties = surface.get_properties()
                    if properties['type'] == 'Wall' and (properties['orientation'] > 315.01 or properties['orientation'] <= 45.01):
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 135.01 and properties['orientation'] <= 225.01:
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 45.01 and properties['orientation'] <= 135.01:
                        surface.move(depth_change)
            if 'west' in body_data['name']:
                for surface in surfaces:
                    properties = surface.get_properties()
                    if properties['type'] == 'Wall' and (properties['orientation'] > 315.01 or properties['orientation'] <= 45.01):
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 135.01 and properties['orientation'] <= 225.01:
                        surface.move(overhang_change)
                    if properties['type'] == 'Wall' and properties['orientation'] > 225.01 and properties['orientation'] <= 315.01:
                        surface.move(depth_change)


if __name__ == '__main__':

    # This is a unit test to check the functions using the current body selection set
    # Select some bodies in the VE then run the script; watch the changes in VE Model view

    project = iesve.VEProject.get_current_project()
    model = project.models[0]
    bodies = model.get_bodies(True)

    # Set some test values
    reduce_percent = 30
    percent = 50
    overhang_change = 0.25
    depth_change = 0.1

    print('1. Demo model glazing reduction')
    reduce_model_glazing_percent(reduce_percent)

    time.sleep(1)

    print('2. Demo body glazing adjustment')
    if not bodies:
        print('No bodies selected; exiting')
        quit()
    else:
        change_body_glazing_percent(bodies, percent)

    time.sleep(1)

    print('3. Demo shade adjustment')
    if not bodies:
        print('No bodies selected; exiting')
        quit()
    else:
        for i in range(0,3):
            change_shade_size(bodies, overhang_change, depth_change)
            time.sleep(1)
```

Properly structuring & commenting your code will help maintenance and allow others to read and use your code.

In this example good commenting and meaningful function / variable names explains use and differences to the reviewer.

In this example using functions makes it easy to test and to iterate.

Import modules; we will use the time module to provide a sleep method for the unit tests

We have created a number of functions; each providing a specific function so what it does it is clear

We comment what the function does, what the parameters are (type & description) and any notes like inherent limitations

We get an instance of the model geometry and assign it to a variable

We call a method of the VEGeometry API to get the existing WWR

We check if the percent parameter exceeds the existing WWR; if it does we cannot reduce the WWR so we exit the function early

We use a method of the VEGeometry API to reduce the model WWR

There is no *return* keyword as we are not returning any object from the function

We comment clearly the differences to the previous function

We comment clearly what the parameters are: type & description

We iterate through the list of bodies passed into the function, we check if it is a *room* (so the user can select any object types for the *bodies* list and it will work with all these functions) and if it is we *select* it

We use methods of the VEGeometry API to first remove then apply new glazing on the *selected* bodies

This function uses a string per elevation so that it can adjust the shades size without growing room objects (you could also use room groups)

The function is written to cater for rotated objects not just objects that are orthogonal to the XY axes

We comment clearly what the method limitations are; a wall surface angle of 45.0 is likely and logic with this as a rotation threshold would fail, so we pick an unlikely threshold

We iterate through the list of bodies passed into the function, we check if the body is a *local shade* and if it is get it's general properties DICT

We get the body's surfaces then test to see if the body's name string contains the defined elevation identifier

We iterate through the body's list of surfaces; we use the surface properties DICT entries to find the relevant shade walls to move

For a north facing wall the boundaries of north sit astride 360/0 degrees so we use OR not AND plus precedence brackets to ensure the correct logic

We use *if __name__ == '__main__':* to create a means to test the functions

We get the current project, then the actual model, then a list of bodies that are currently *selected* by the user in the VE UI by setting the *get_bodies* method parameter to *True*

We set some test values for the function parameters; note the names we use do not need to be the same as those used in the function definition

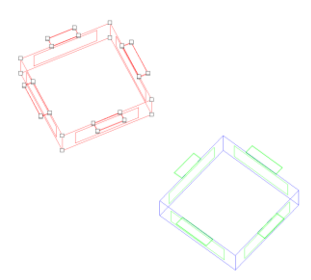
We call the first function; it is not assigned to a variable as we do need to return anything

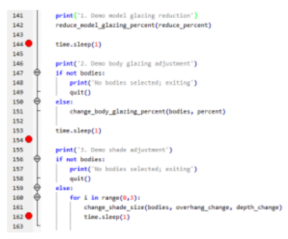
We call the time module sleep method and pass in 1 sec so the changes in the VE view are obvious

Before calling the second function the conditional statement catches if the user has not selected any bodies

For the third function we add a loop to show the function being called repeatedly; any selected shades will 'grow' in the VE view

Sample output:



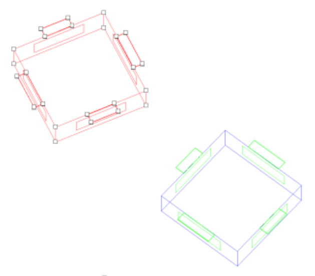


```python
    print('1. Demo model glazing reduction')
    reduce_model_glazing_percent(reduce_percent)

    time.sleep(1)

    print('2. Demo body glazing adjustment')
    if not bodies:
        print('No bodies selected; exiting')
        quit()
    else:
        change_body_glazing_percent(bodies, percent)

    time.sleep(1)

    print('3. Demo shade adjustment')
    if not bodies:
        print('No bodies selected; exiting')
        quit()
    else:
        for i in range(0,3):
            change_shade_size(bodies, overhang_change, depth_change)
            time.sleep(1)
```

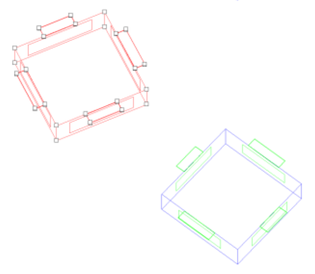
Starting test model; two identical rooms, 50% glazing, local shades, one room rotated.

I have selected the rotated room and it's local shades.

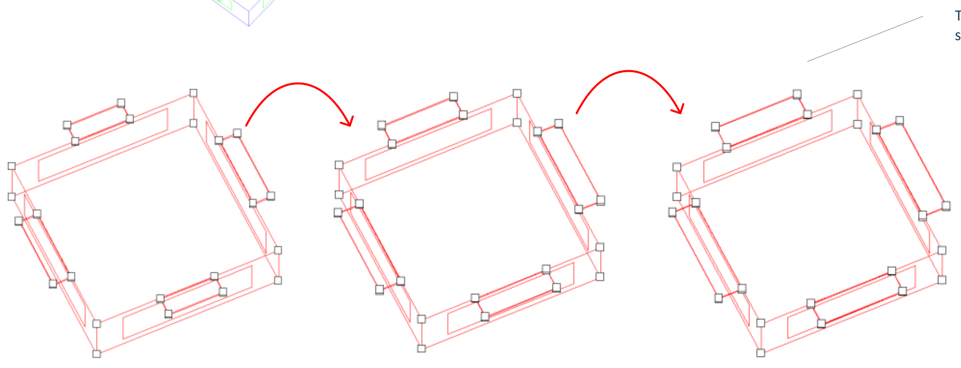
To screen grab these images I have set break points in the code and run the script in debug mode.



The first function reduces WWR on the model ... so on both rooms



The second function removes & adds windows to the selected room only



The third function 'grows' the selected local shades in steps