Space Planning | Solar Envelope

```
1.  //This algorithm checks if any of the boxes in the site volume potentially blocks the sunlight for
2.  //the neightbouring buildings and stores how many times each face blocks it.
3.
4.  //Inputs are the box cloud (Input1), Points of Interest (Input2), Sun Rays (Input3) and
5.  //the Surrounding Buildings (Input4)
6.  //Firstly, it checks if the sun was already blocked from the site geometry
7.  //If not, it adds on hit count to every face intersecting with the reversed sun ray "shot" from the
8.  //point of interest
9.
10. //The output is a box cloud with each face having a hit count illustrating
11. //how much sun light it is blocking
12.
13. int p_i=npoints(@OpInput2); // Amount of ""Points of Interest"
14. int n_n=nprimitives(@OpInput3); // Amount of sun rays
15. vector uvw[];
16. int target[];
17. vector pos[];
18. vector test_vec;
19. float test_u;
20. float test_v;
21.
22. for(int k=0; k<p_i;k++){
23.     for(int j =0; j<n_n;j++){
24.         vector sun_n=prim_normal(@OpInput3, j, 0.5,0.5);
25.         int test = intersect(@OpInput4 , point(@OpInput2,"P",k), sun_n*1000, test_vec, test_u, test_v);
            // Test the Sun Ray hits the Site Geometry first
26.         if (test=-1){
27.             int hit = intersect_all( @OpInput1 , "" , point(@OpInput2,"P",k), sun_n*1000, pos, target, uvw);
                //Find all Faces blocking the Sun
28.             for(int i=0; i<hit; i++){
29.                 setprimattrib(0, 'Hit_Count', target[i], 1 , "add");
30.             }
31.         }
32.     }
33. }
```