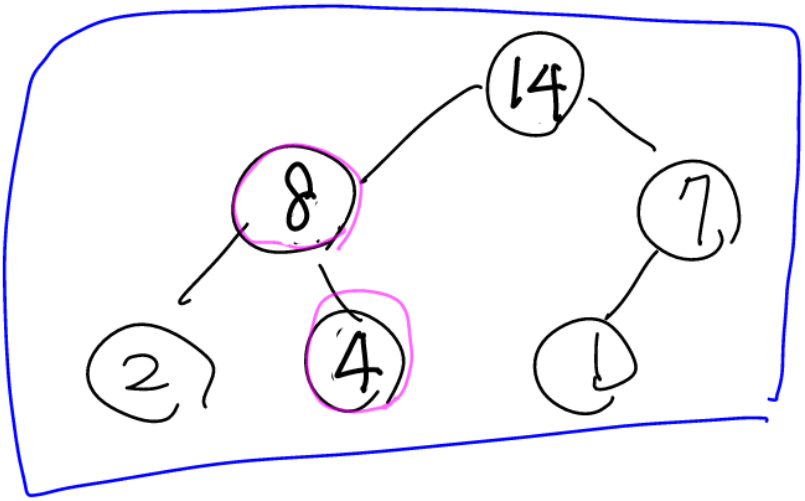
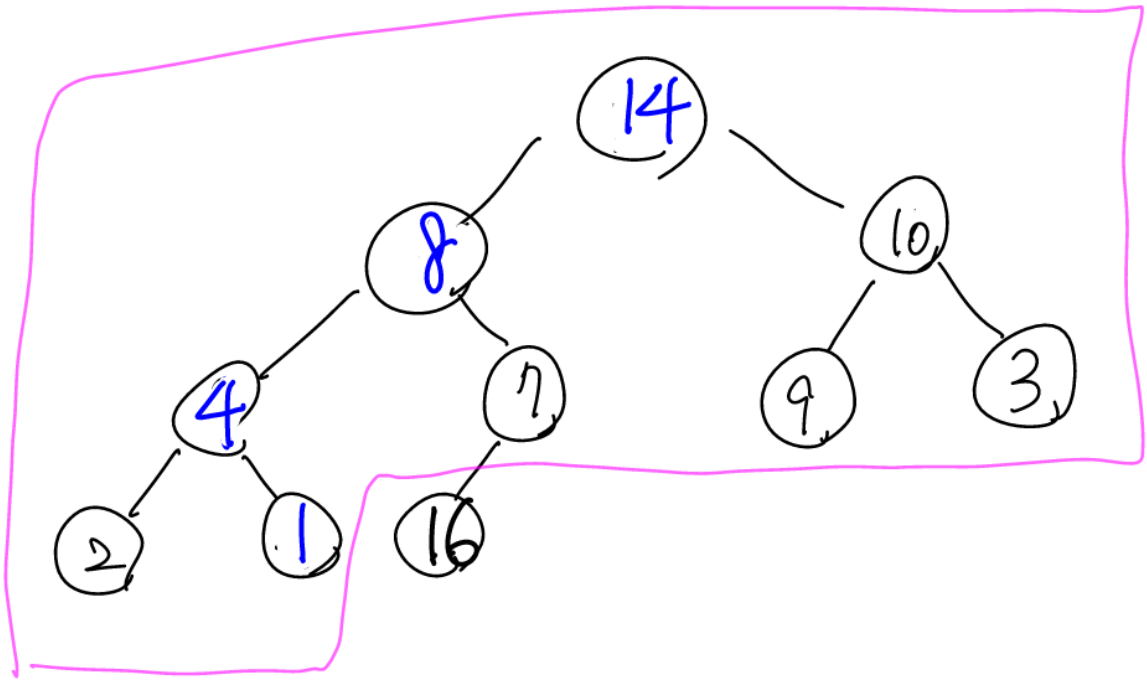
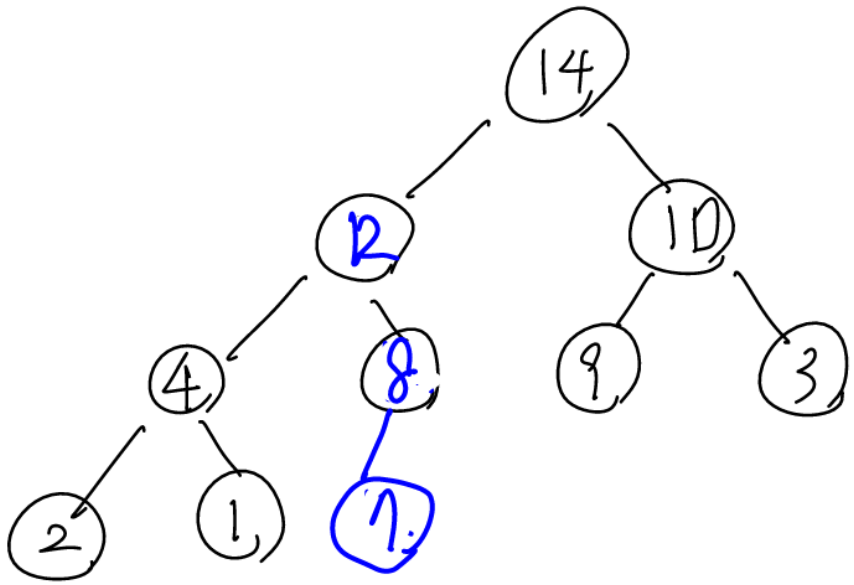
② Navigation Mesh

heap (data structure)
priority queue
hash container (unordered_set)
A star algorithm
Navigation Mesh in Unity



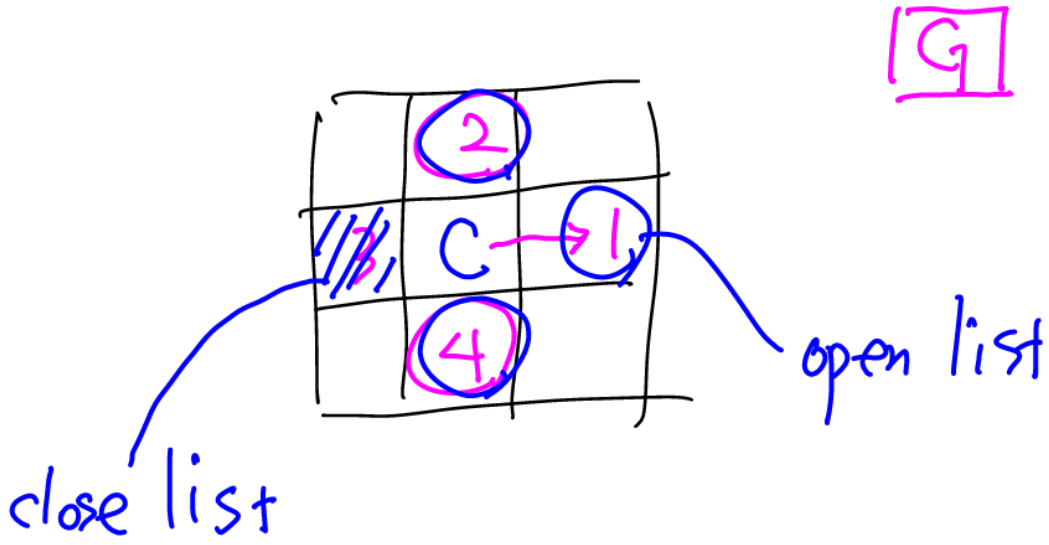




std::priority_queue<>

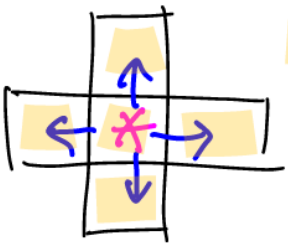
→ uses heap

std::priority_queue<AStarNode>

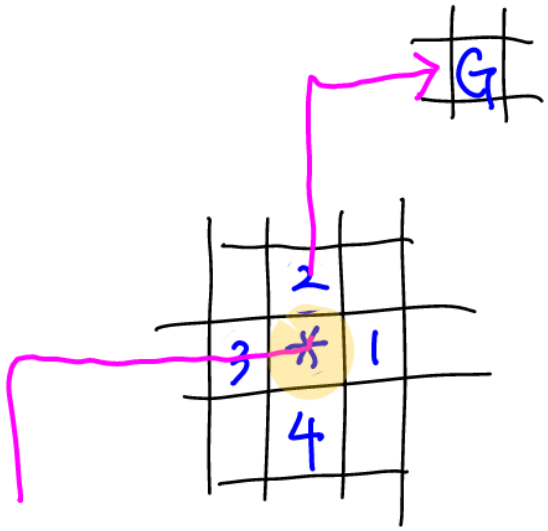


std::unordered_set<AStarNode>

@ AStar algorithm (basic)

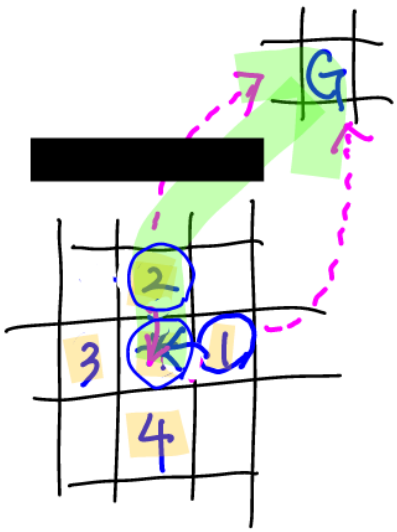


4-connected neighbors



close list

open list (priority queue)



close list

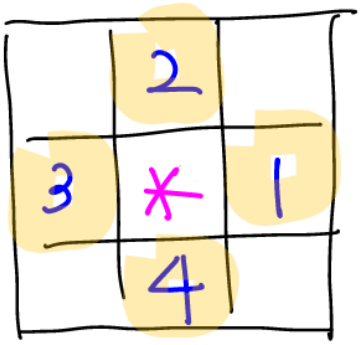
open list (priority queue)

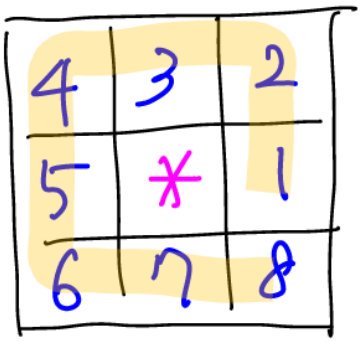
new cost
new heuristic

?

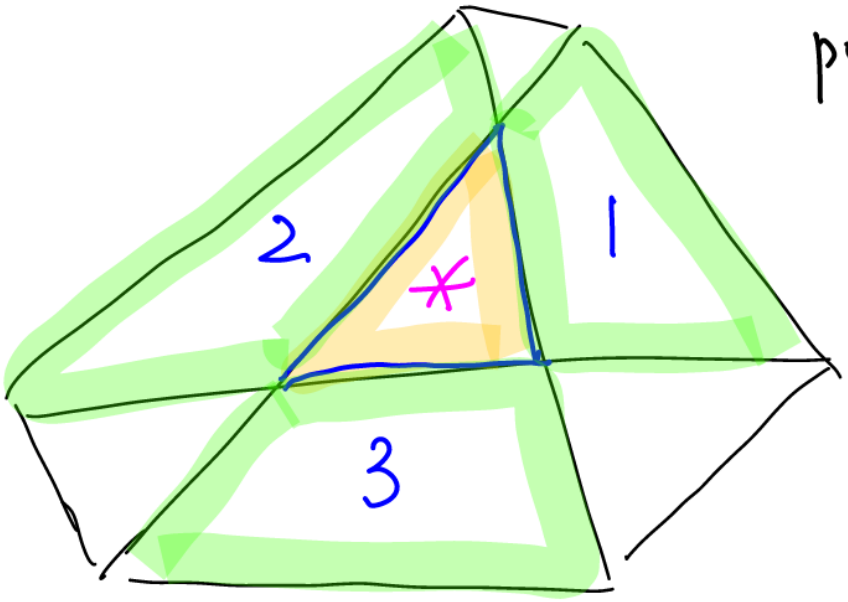
(2018.5.4)

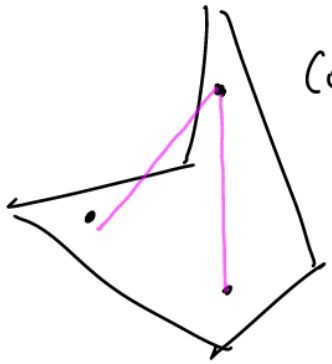
@ Extending neighbors





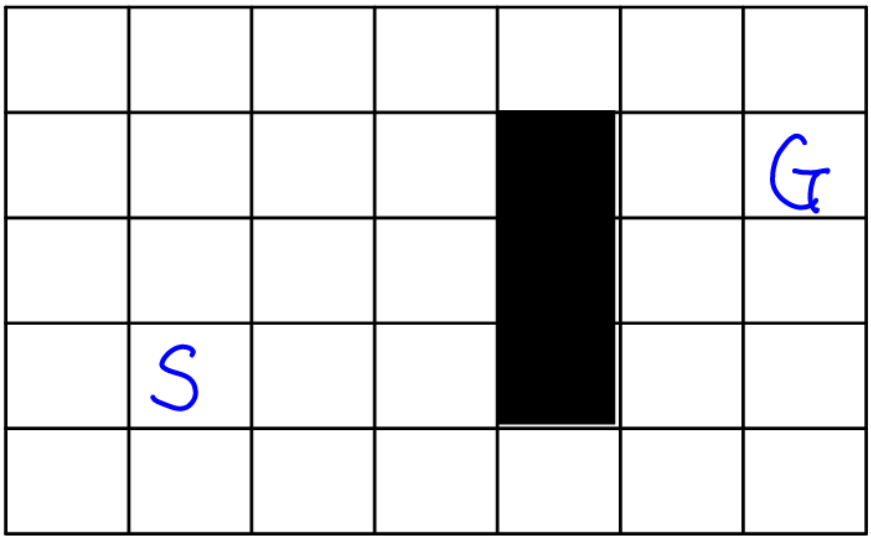
* convex polygon





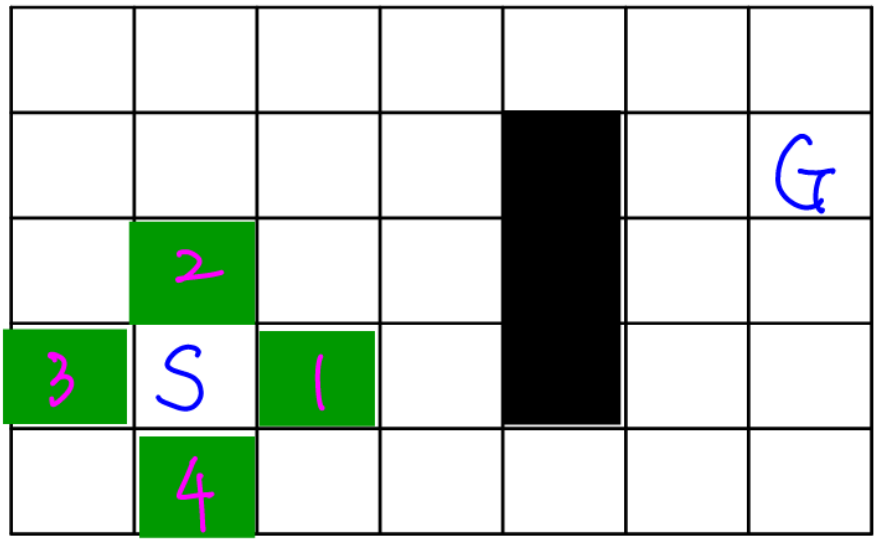
concave.

@ A Star algorithm (detail)



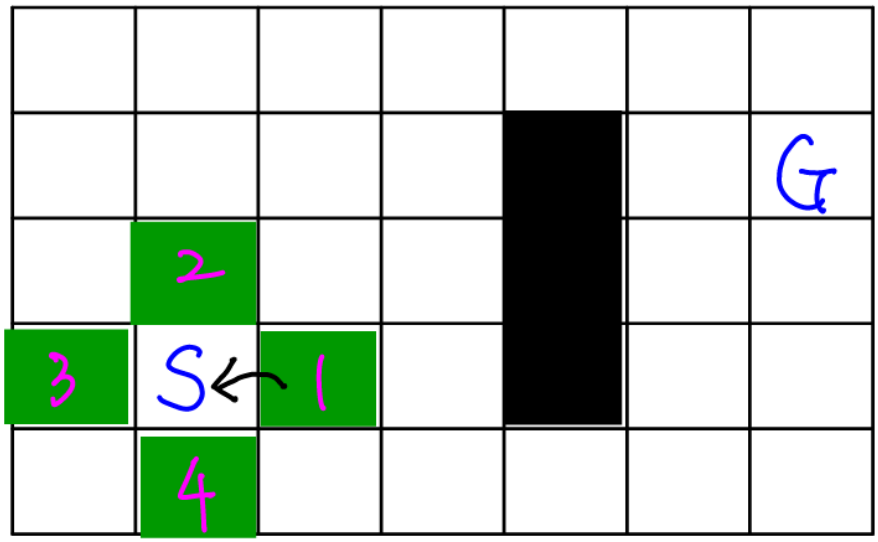
Close (null)

Open S



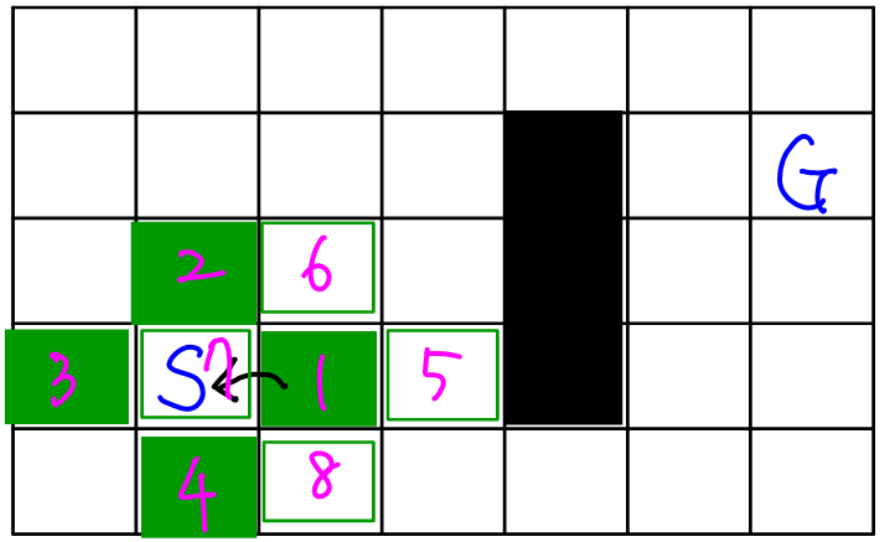
Close S

Open 1 2 3 4

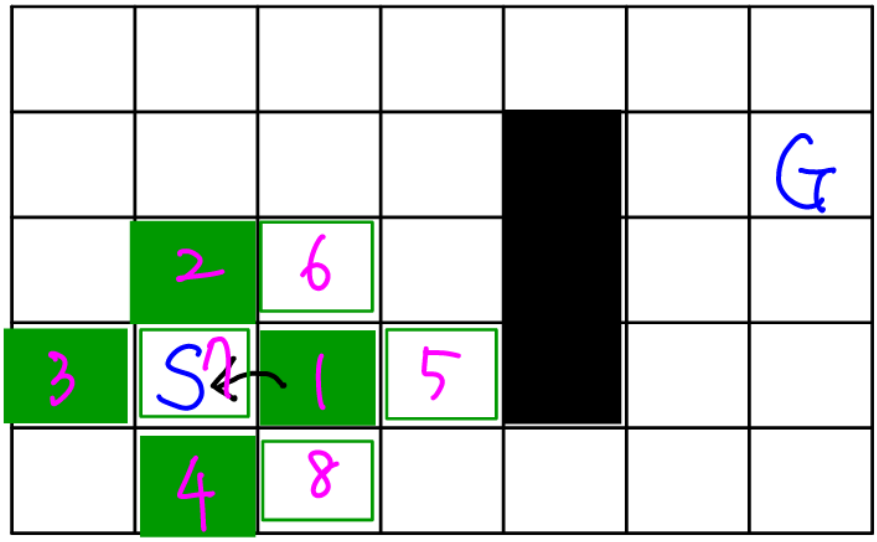


Close S 1

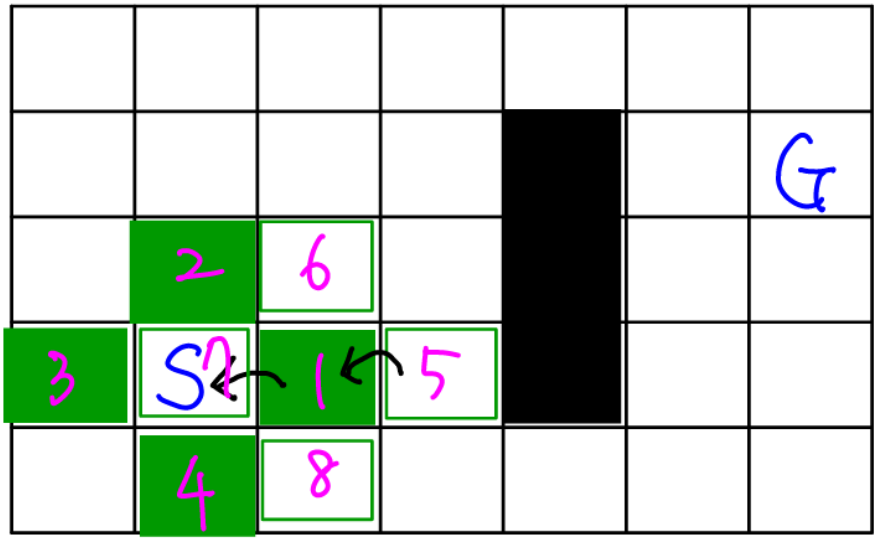
Open ~~1~~ 2 3 4



close　open
S　2
1　3
　　4

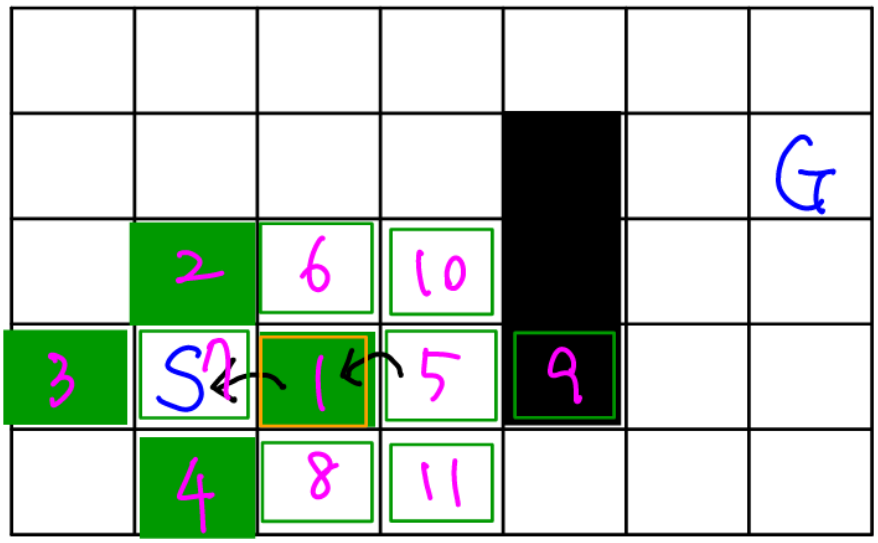


close　open
S　2
1　3
　　4
　　5
　　6
　　8



close　open
S　2
1　3
5　4
　　~~5~~
　　6
　　8

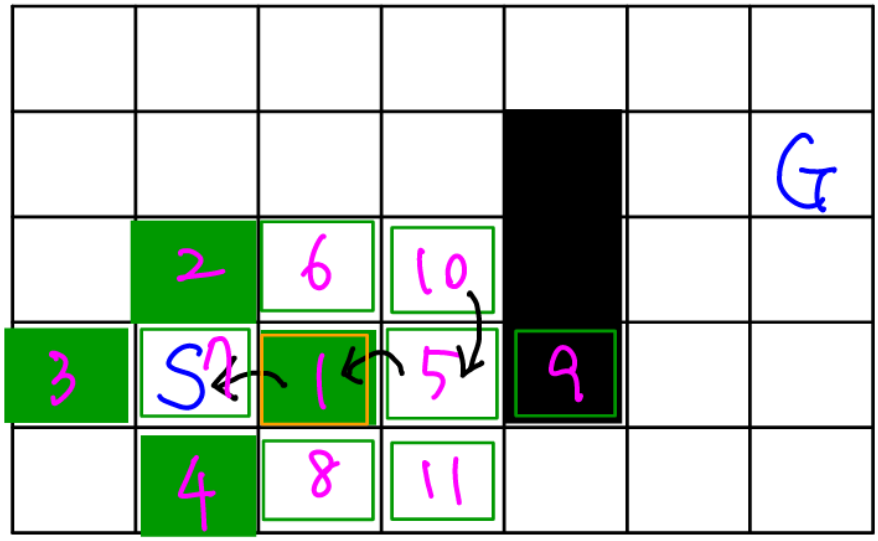
η = S



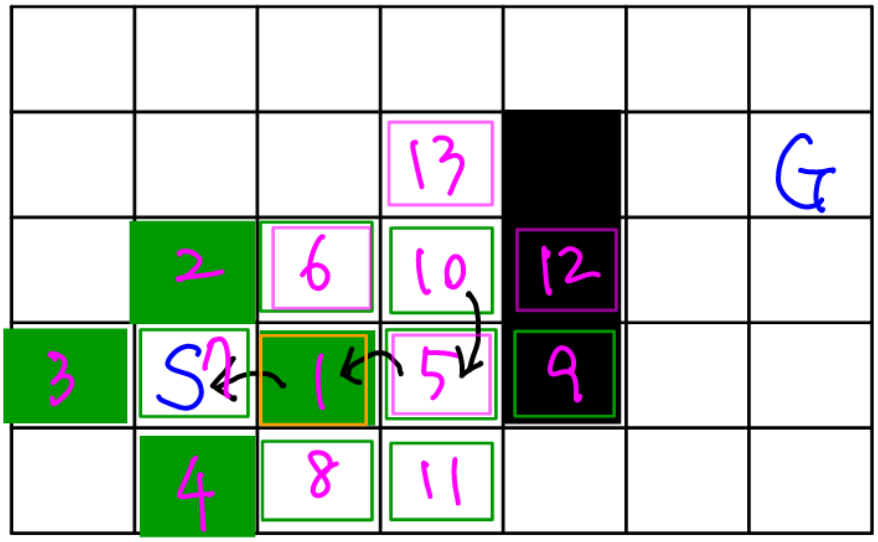
close	open
S	2
1	3
5	4
	6
	8
	10
	11

5's neighbors ≡ 9, 10, 1, 11

9 = wall (not a neighbor)



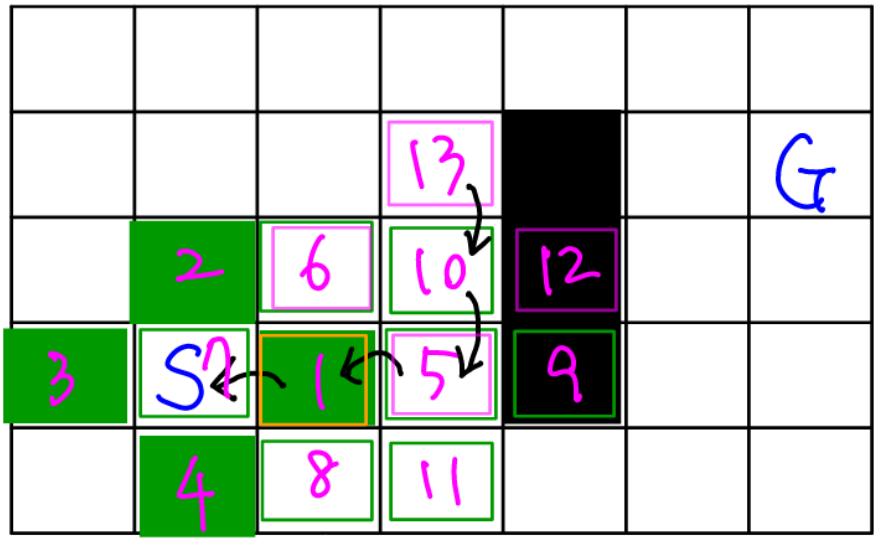
close	open
S	2
1	3
5	4
10	6
	8
	~~10~~
	11



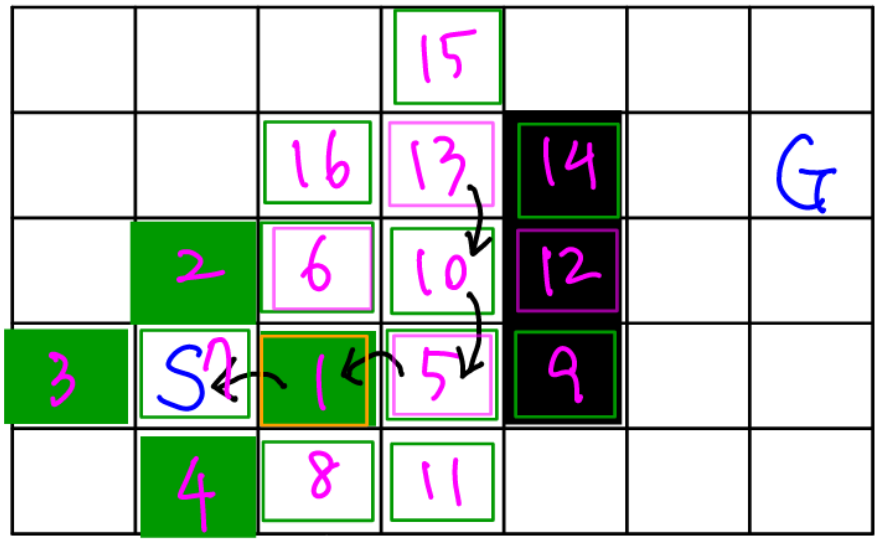
close	open
S	2
1	3
5	4
10	6 : update
	8
	11
	13

10's neighbor ≡ 12, 13, 6, 5

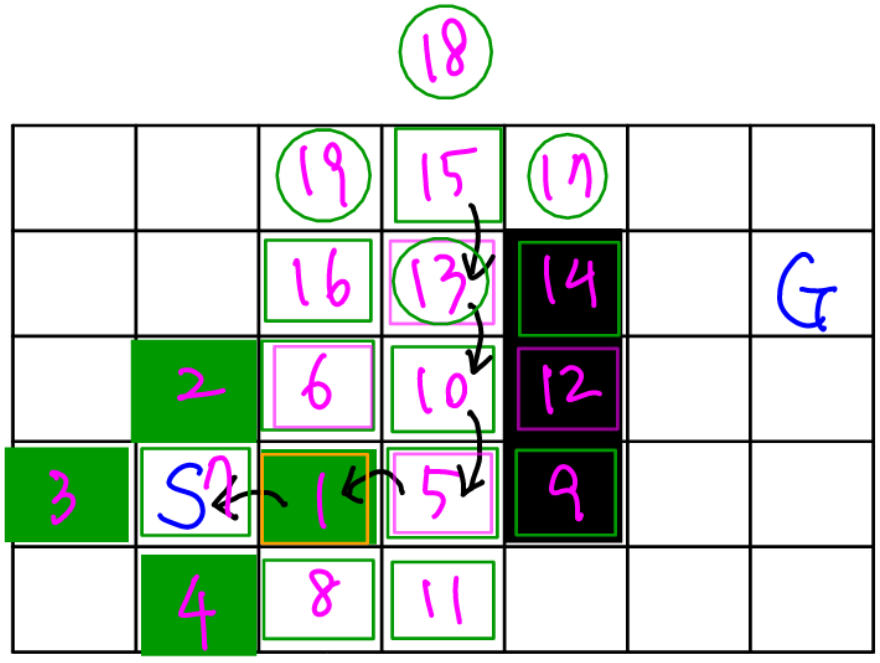
↙ wall ↓ open ↓ close



close	open
S	2
1	3
5	4
10	6
13	8
	11
	~~13~~

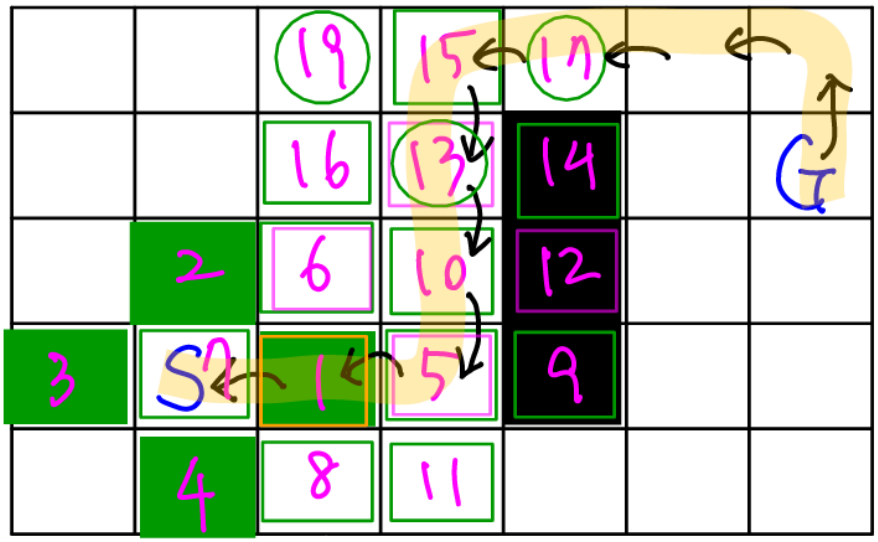


close	open
S	2
1	3
5	4
10	6
13	8
	11
	15
	16



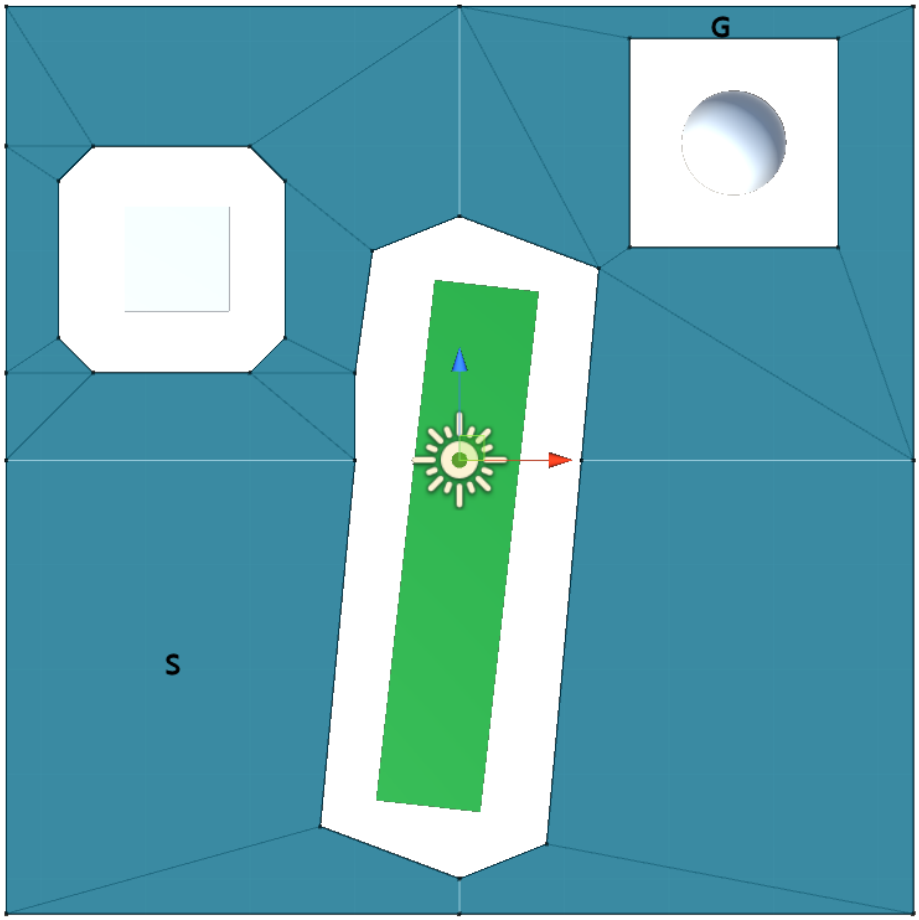
close	open
S	2
1	3
5	4
10	6
13	8
15	11
	~~15~~
	16

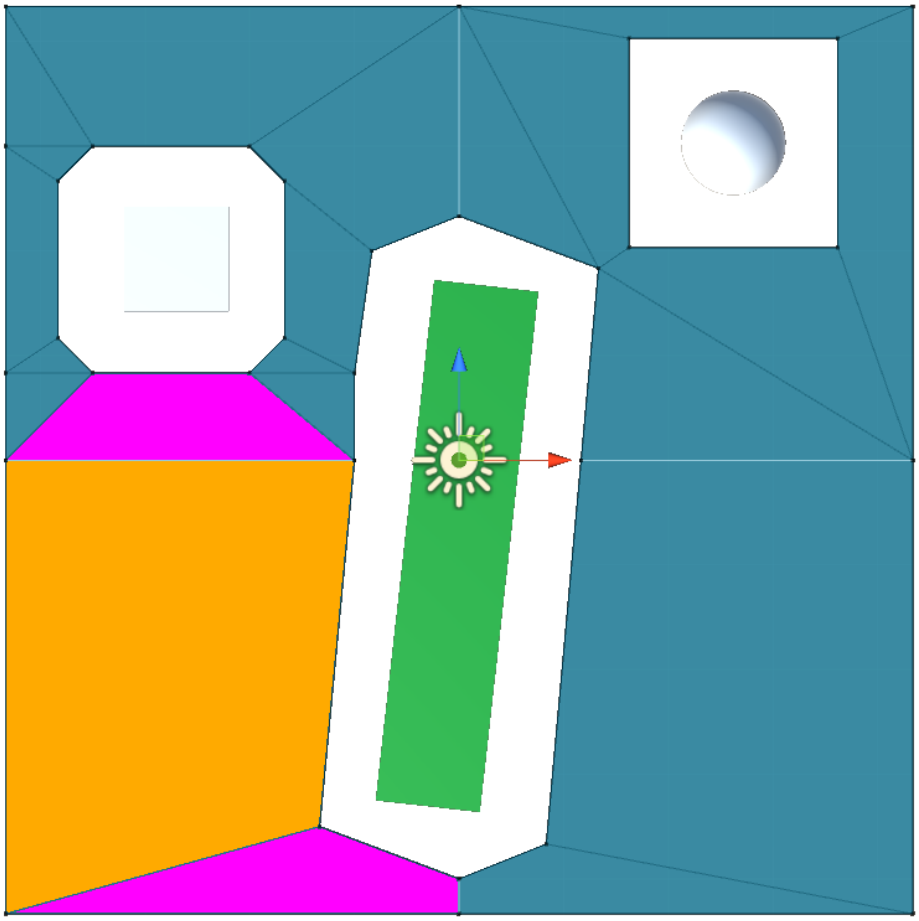
* final result

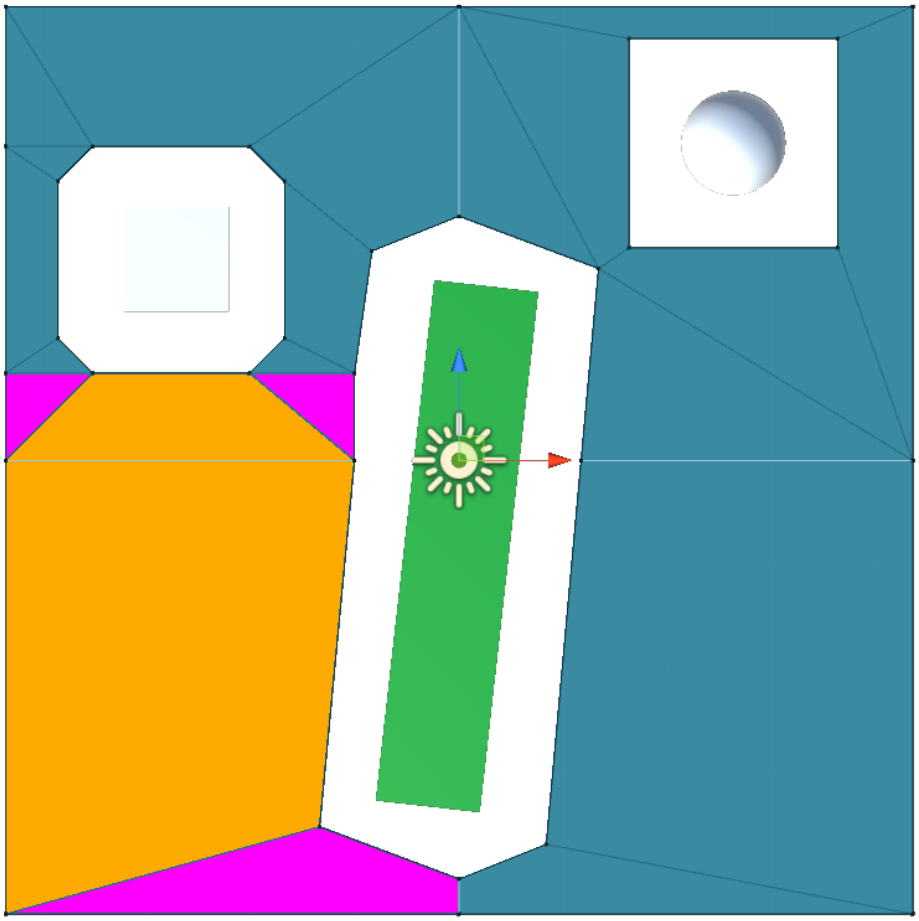


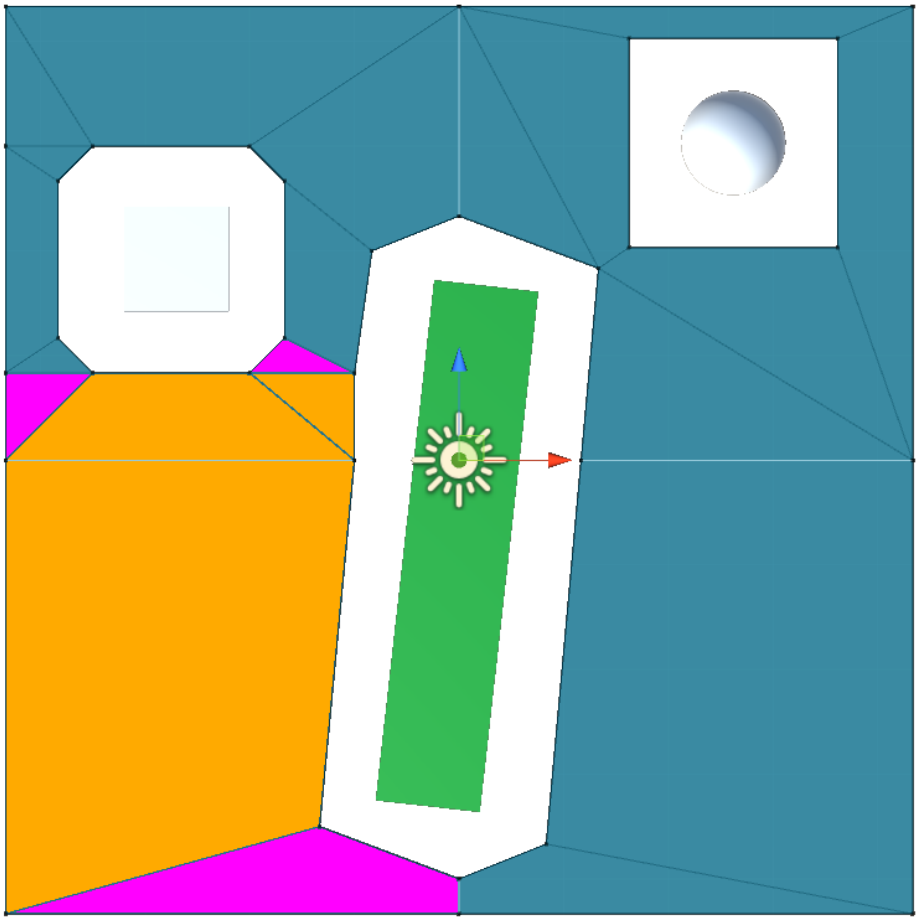
close	open
S	2
1	3
5	4
10	6
13	8
15	11
17	16
⋮	19
	⋮

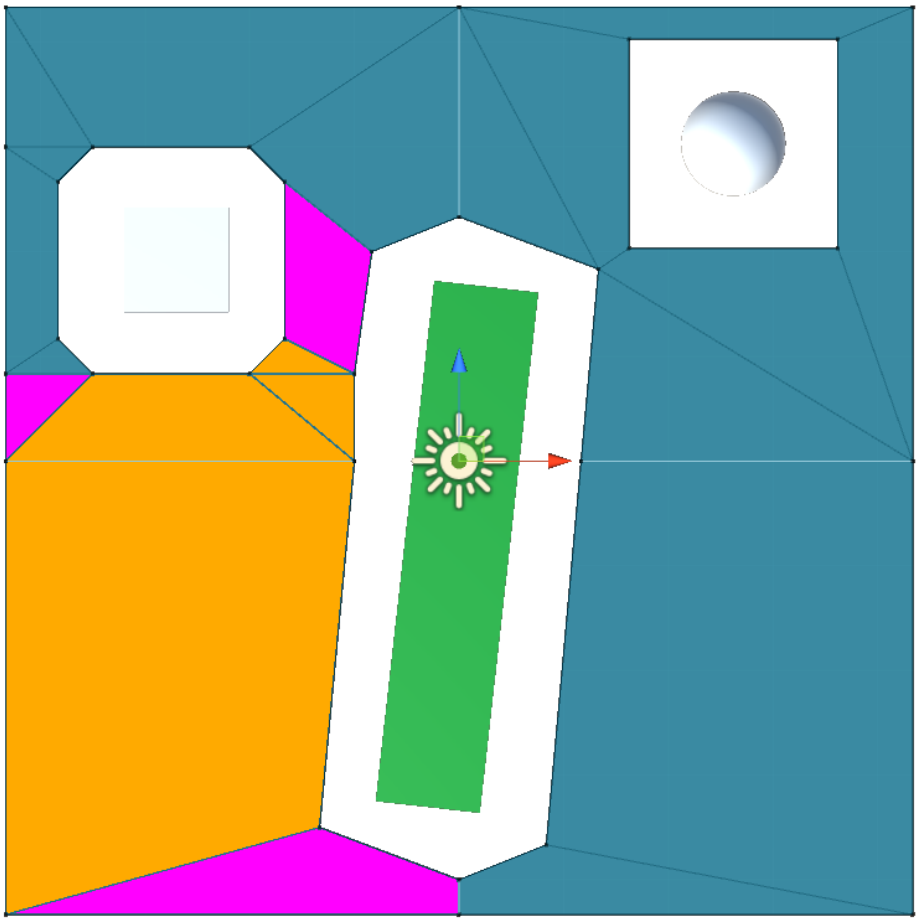
(2018. 5. 6)

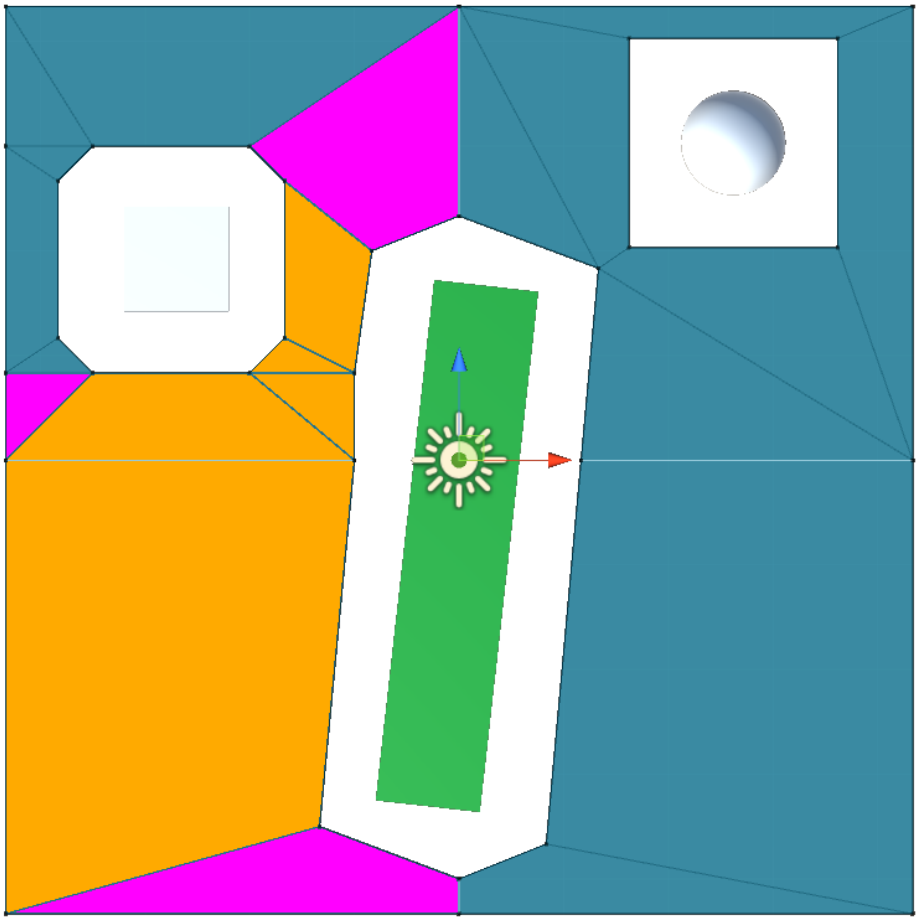


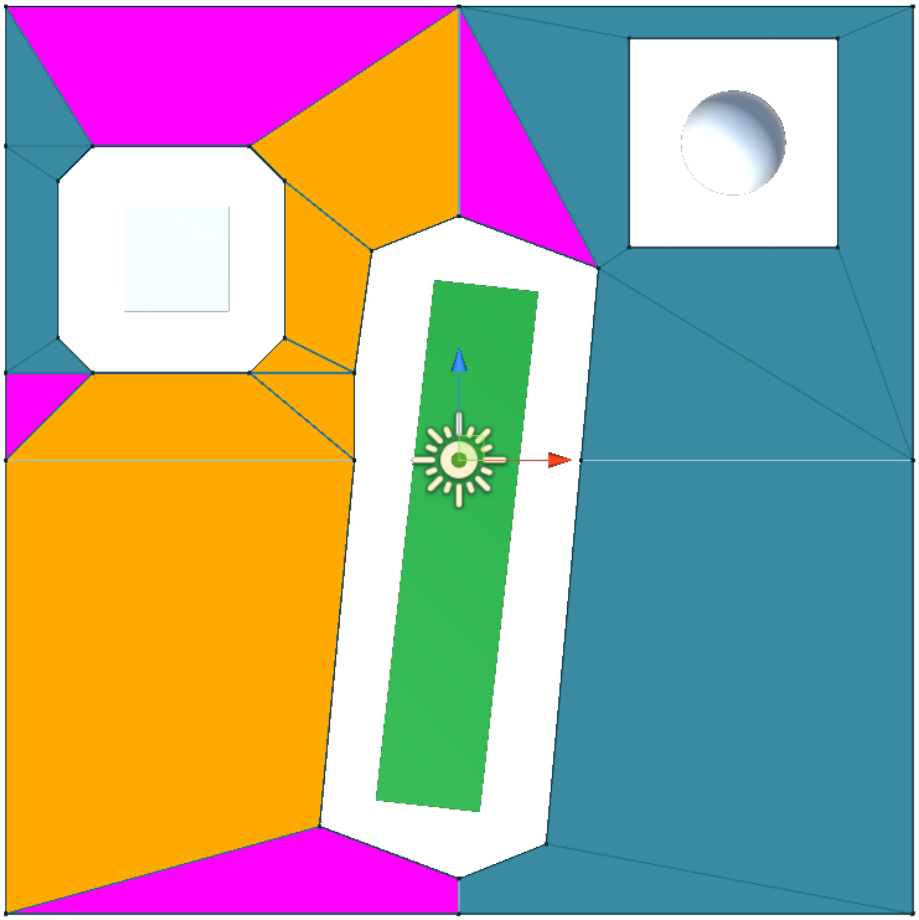


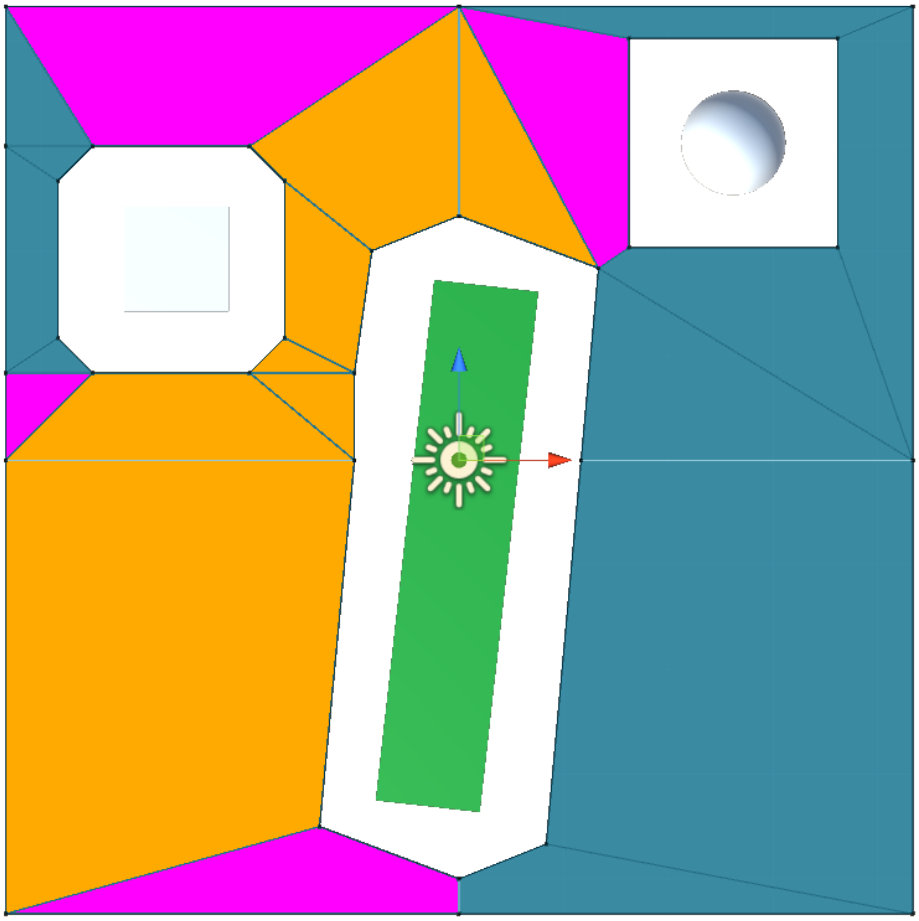


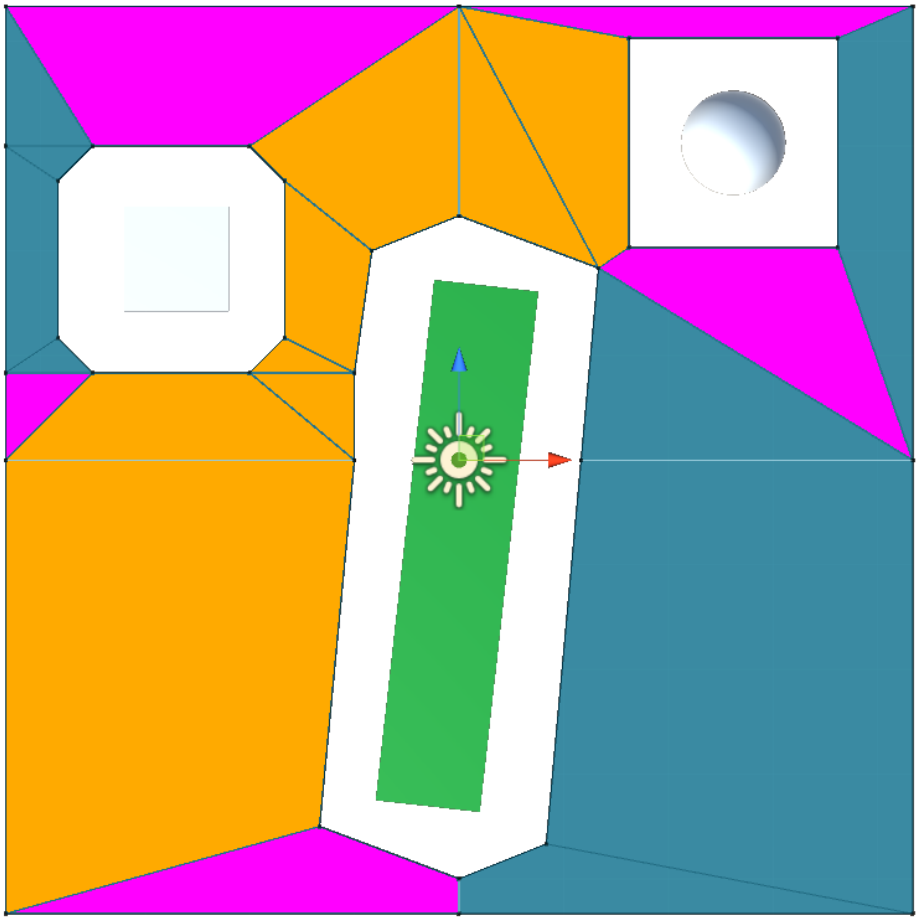


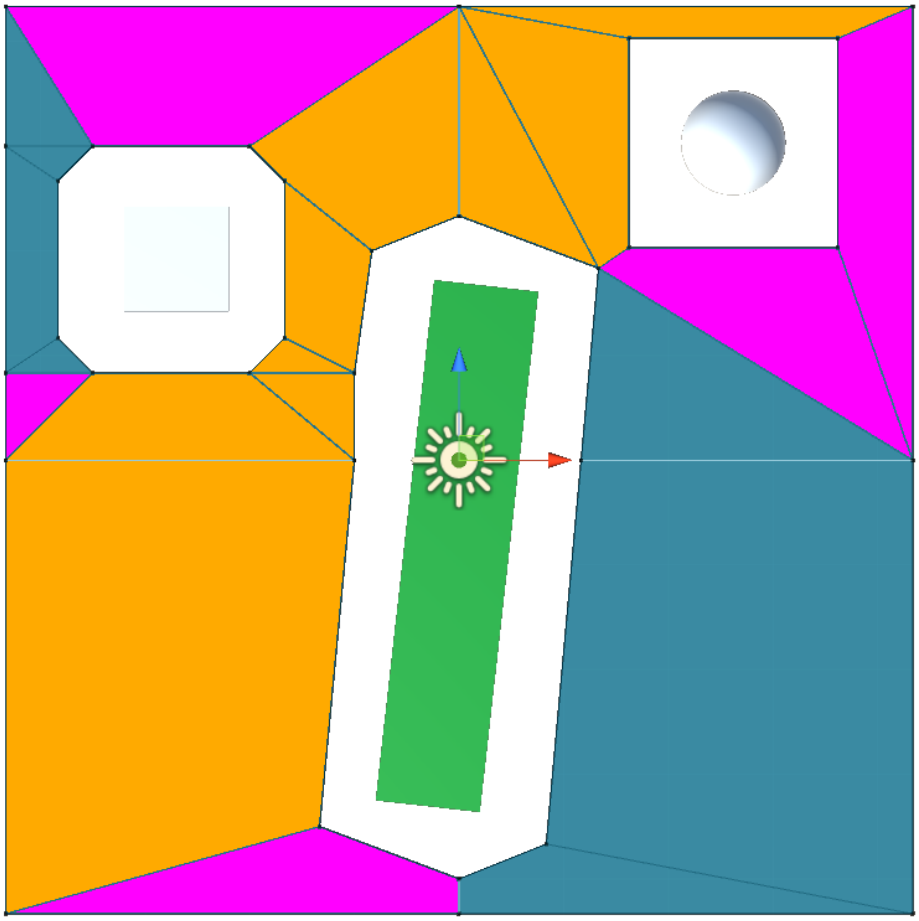


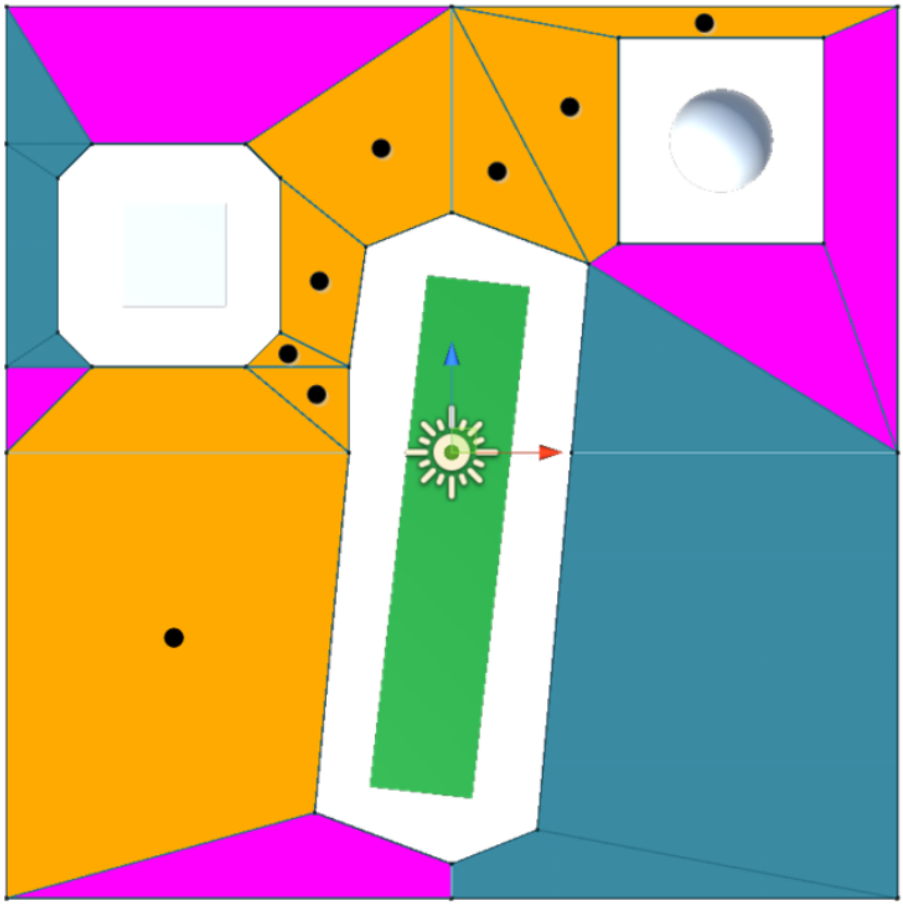


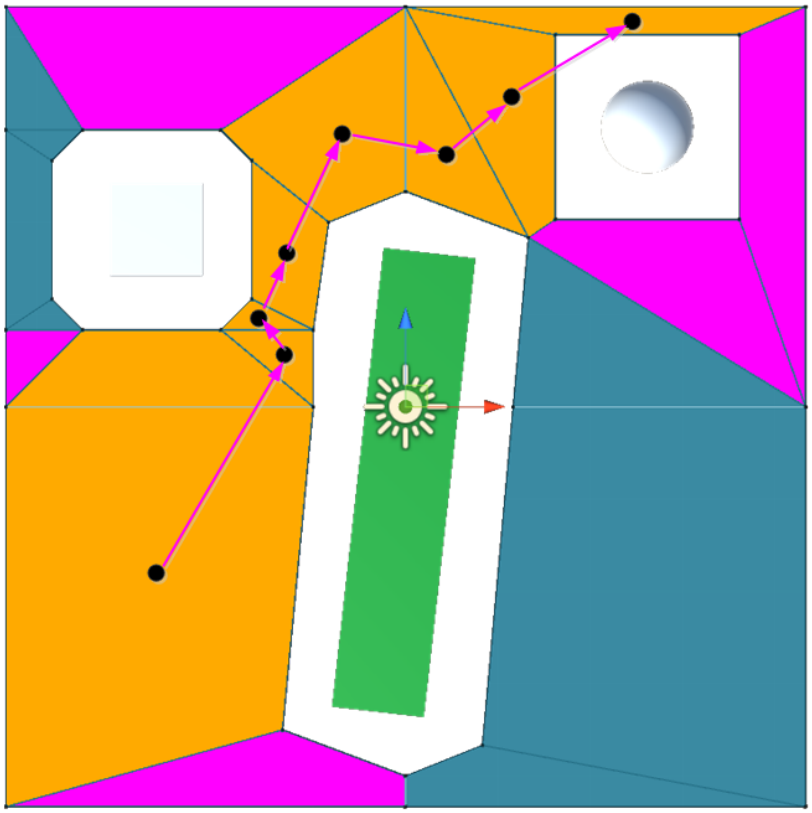


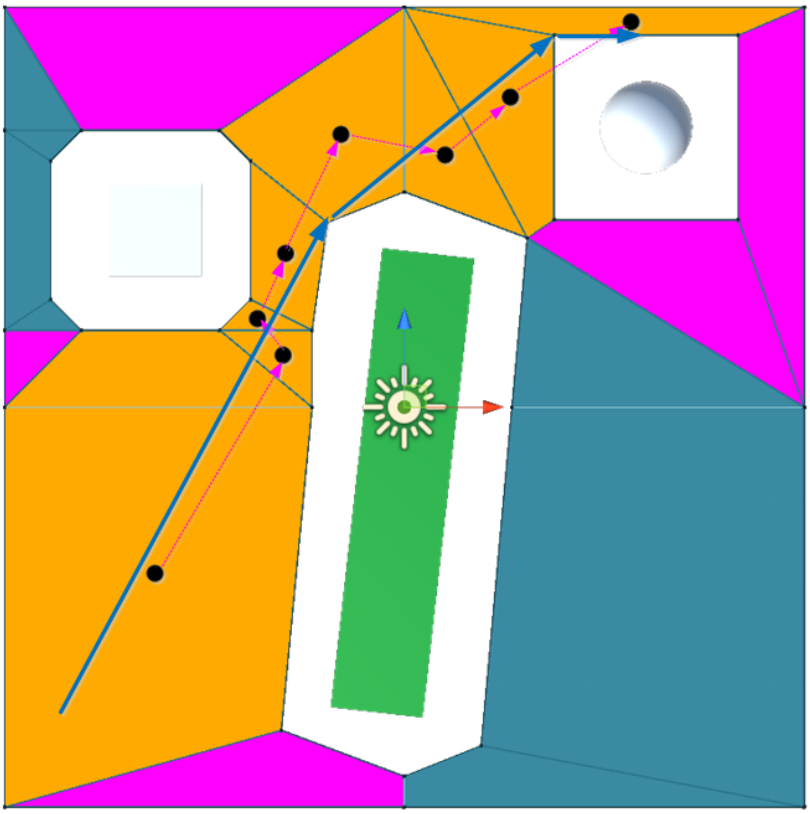


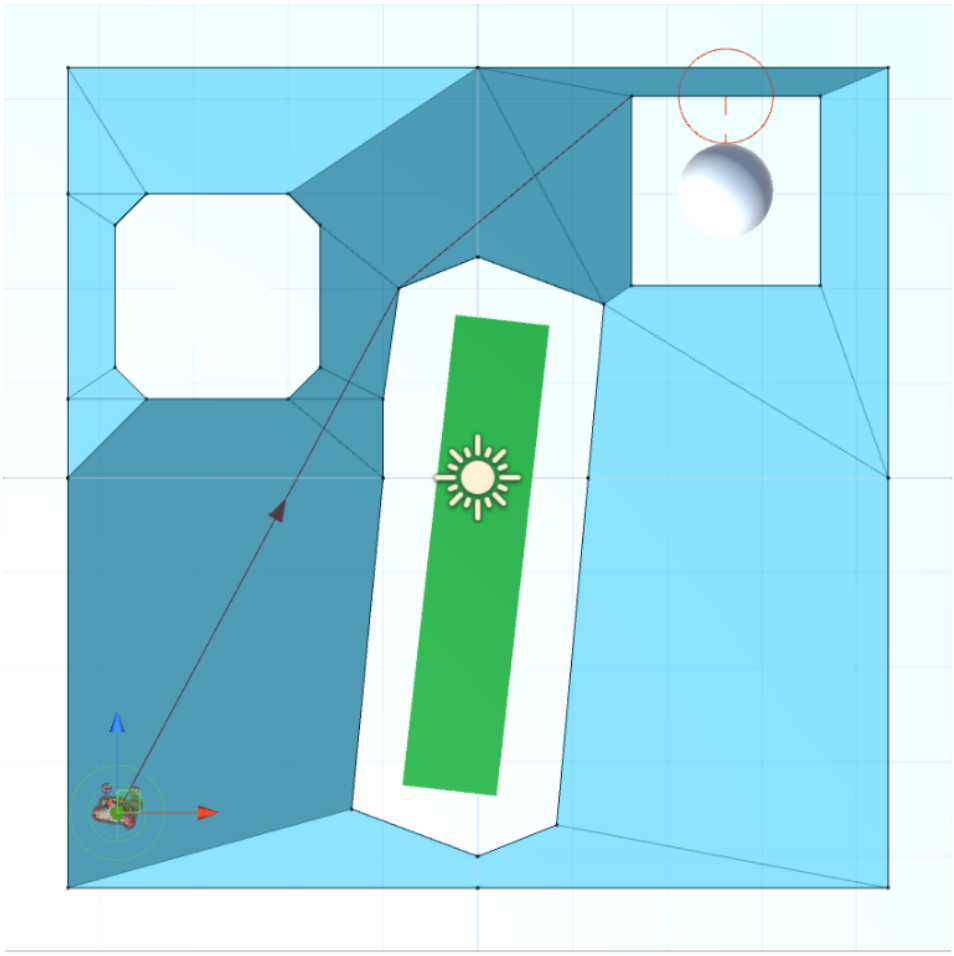


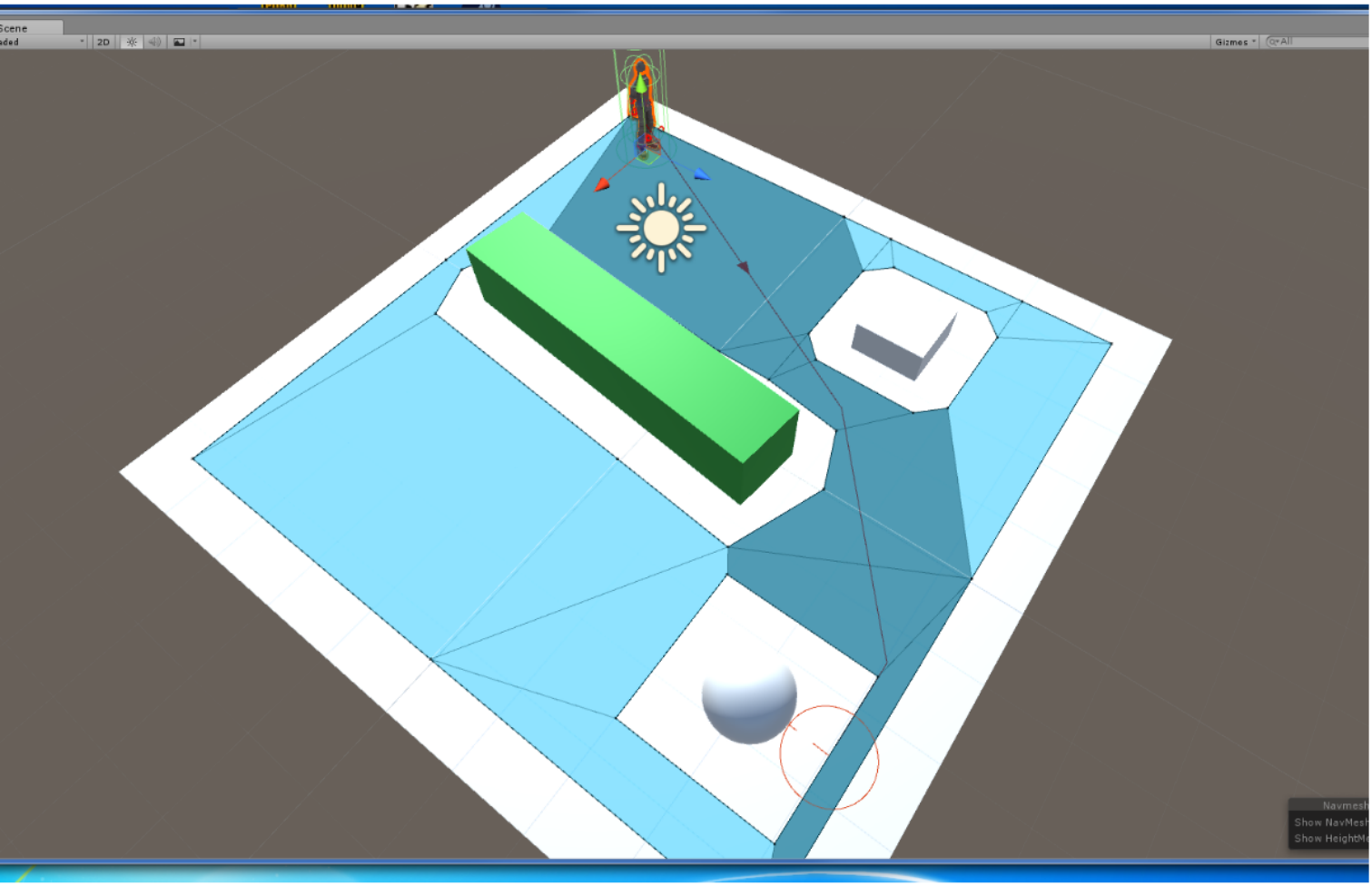











Scene
2D
Gizmos
All

```
using UnityEngine.AI;

NavMeshAgent    m_agent;

    m_agent.SetDestination( m_target.transform.
                                position);
    if( m_agent.remainingDistance >
                m_agent.stopDistance )
    {
        gameObject.transform.position +=
            m_agent.desiredVelocity * Time.deltaTime;
    }
```

(2018.5.1)