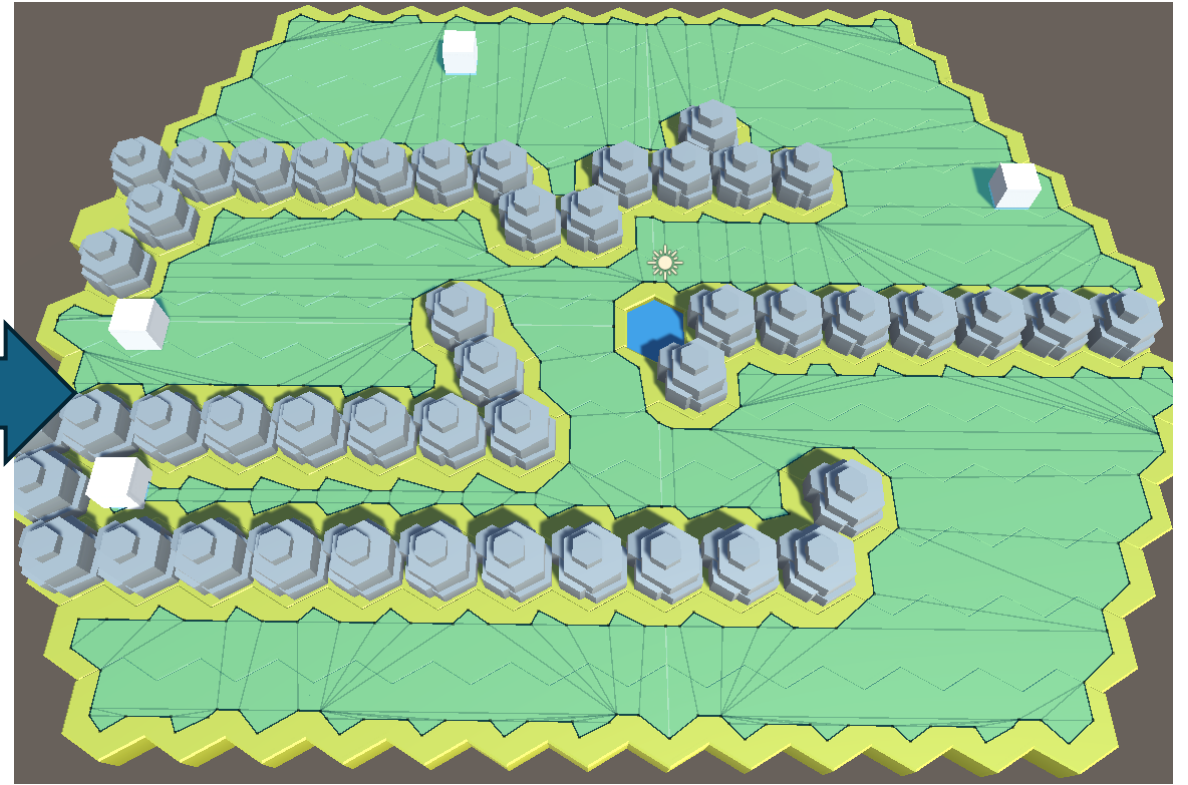
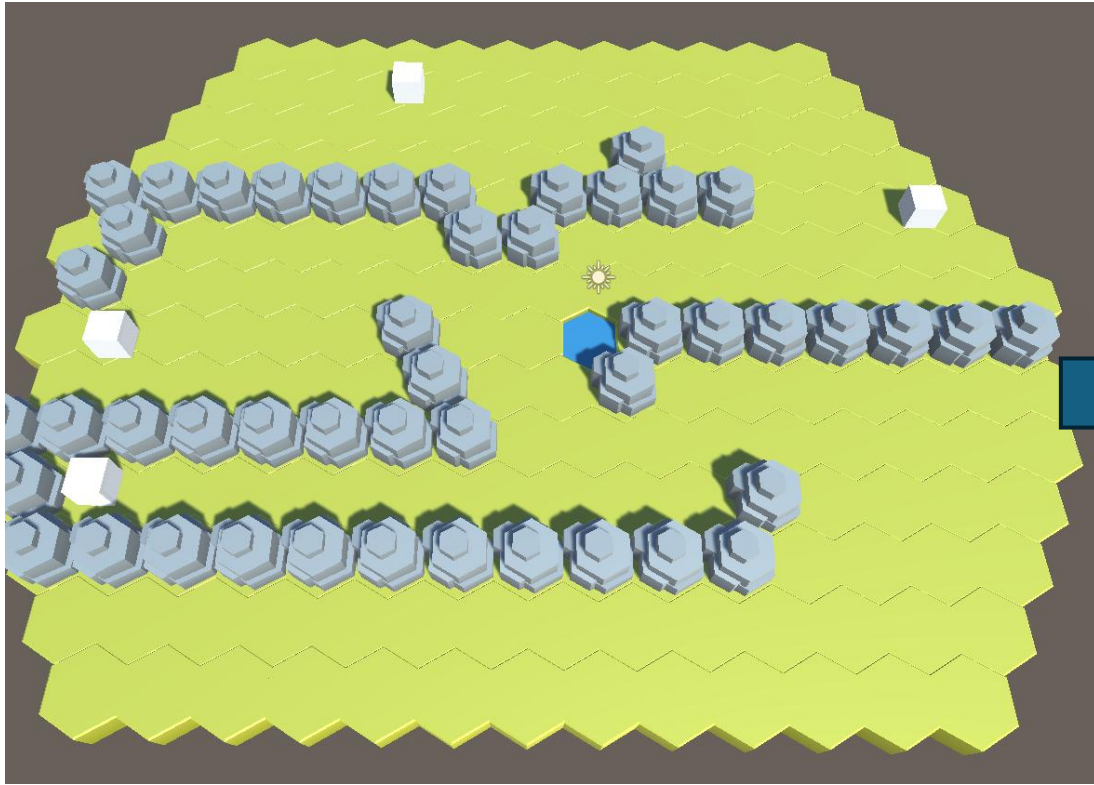


서버에서 어떻게 NavMesh를 이용할까?

- 클라이언트 엔진에서 타일맵(지형)을 만든다.
- 해당 지형으로 엔진의 NavMesh를 굽는다.
- NavMesh정보를 텍스트형태로 추출하여 C++서버에서 파싱한다.
- 파싱된 정보를 그래프화하여 실제 월드 좌표로 경로를 계산한다.

엔진의 씬에서 지형을 만들고 NavMesh를 굽는다.



만들어진 NavMesh를 수치화하여 저장한다.

이때 실제 월드 좌표인 vertices값과 해당 값을 이용하여 삼각형을 구성하는 triangles을 저장한다.



```
"navmesh": {  
  "vertices": [  
    [0.333333, 0.0833331, 1.5],  
    [1.33333, 0.0833331, 1.5],  
    [0.666667, 0.0833331, 1.16667],  
    [2.16667, 0.0833331, 1.5],  
    [3.16667, 0.0833331, 1.5],  
    [2.5, 0.0833331, 1.16667],  
    [4, 0.0833331, 1.5],  
    [5, 0.0833331, 1.5],  
    [4.33333, 0.0833331, 1.16667]  ]  
}
```

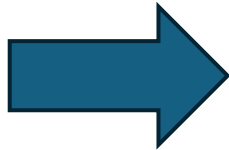
```
"navmesh": {  
  "vertices": [],  
  "triangles": [  
    [0, 1, 2],  
    [3, 4, 5],  
    [6, 7, 8],  
    [9, 10, 11],  
    [12, 13, 14],  
    [15, 16, 17]  ]  
}
```

* triangles[0]은 vertices[0], vertices[1], vertices[2]를 이용하여 실제 월드의 삼각형 구역을 이용한다.

파싱한 값을 서버 메모리에서 저장 할 수 있게 구조체화한다.

```
"navmesh": {  
  "vertices": [  
    [0.333333, 0.08333313, 1.5],  
    [1.333333, 0.08333313, 1.5],  
    [0.666667, 0.08333313, 1.166667],  
    [2.166667, 0.08333313, 1.5],  
    [3.166667, 0.08333313, 1.5],  
    [2.5, 0.08333313, 1.166667],  
    [4, 0.08333313, 1.5],  
    [5, 0.08333313, 1.5],  
    [4.333333, 0.08333313, 1.166667]
```

```
"navmesh": {  
  "vertices": [...],  
  "triangles": [  
    [0, 1, 2],  
    [3, 4, 5],  
    [6, 7, 8],  
    [9, 10, 11],  
    [12, 13, 14],  
    [15, 16, 17]
```



```
struct MapObject  
{  
    string name;  
  
    float worldX, worldY, worldZ;  
    int cellX, cellY, cellZ;  
};  
  
// 네비메시 데이터  
struct NavMeshData  
{  
    struct Vertex { float x, y, z; };  
    struct Triangle { int a, b, c; };  
  
    vector<Vertex> vertices;  
    vector<Triangle> triangles;  
    vector<int> areas;  
};  
  
// 전체 맵 데이터  
struct SceneMap  
{  
    vector<MapObject> objects;  
    NavMeshData navmesh;  
};
```

시각화 하여 출력 ('*' : 갈 수 있는 길)

```

struct MapObject
{
    → string name;

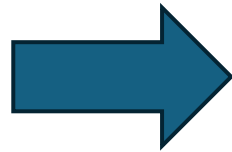
    → float worldX, worldY, worldZ;
    → int cellX, cellY, cellZ;
};

// 네비메시 데이터
struct NavMeshData
{
    → struct Vertex { float x, y, z; };
    → struct Triangle { int a, b, c; };

    → vector<Vertex> vertices;
    → vector<Triangle> triangles;
    → vector<int> areas;
};

// 전체 맵 데이터
struct SceneMap
{
    → vector<MapObject> objects;
    → NavMeshData navmesh;
};

```

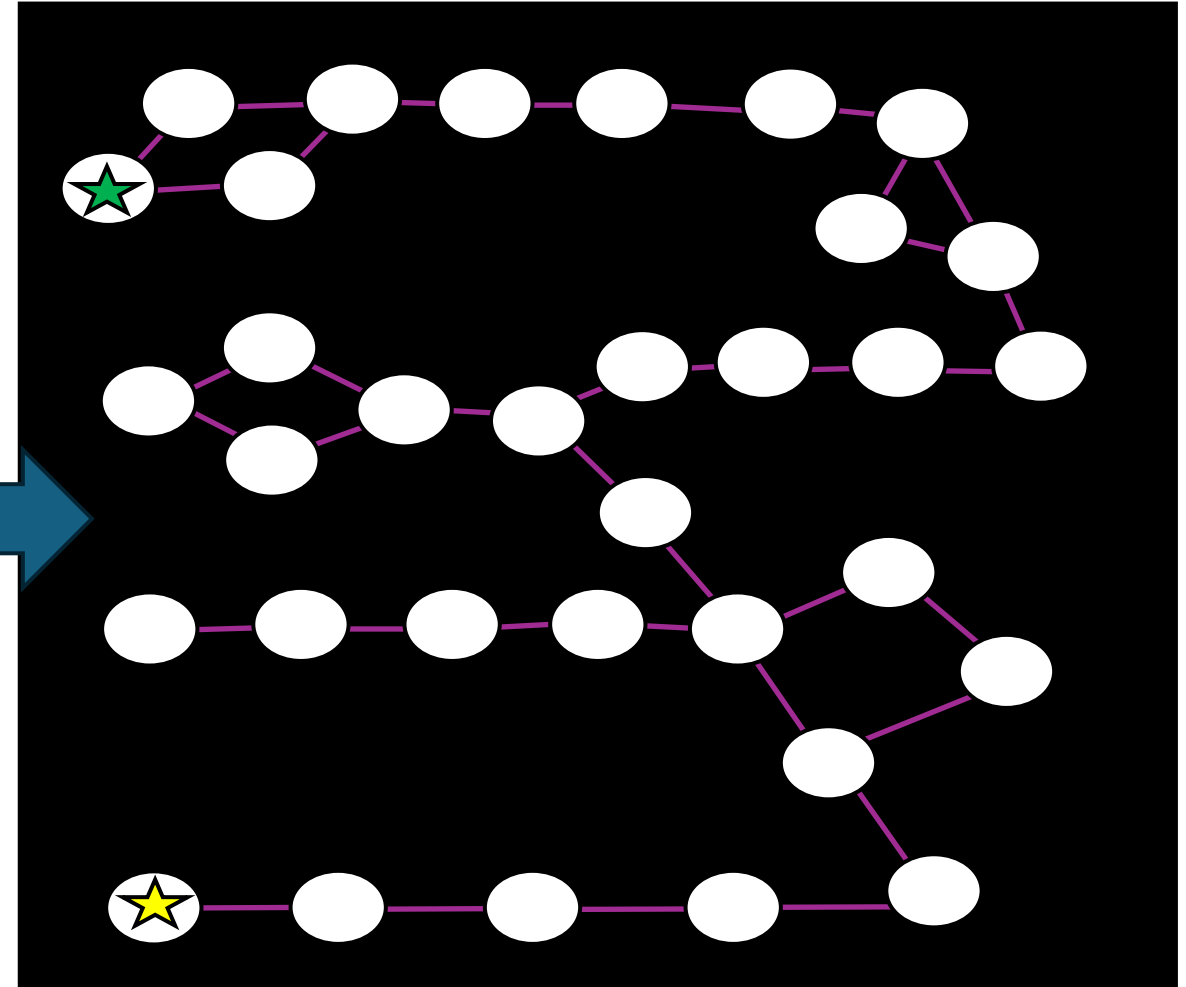
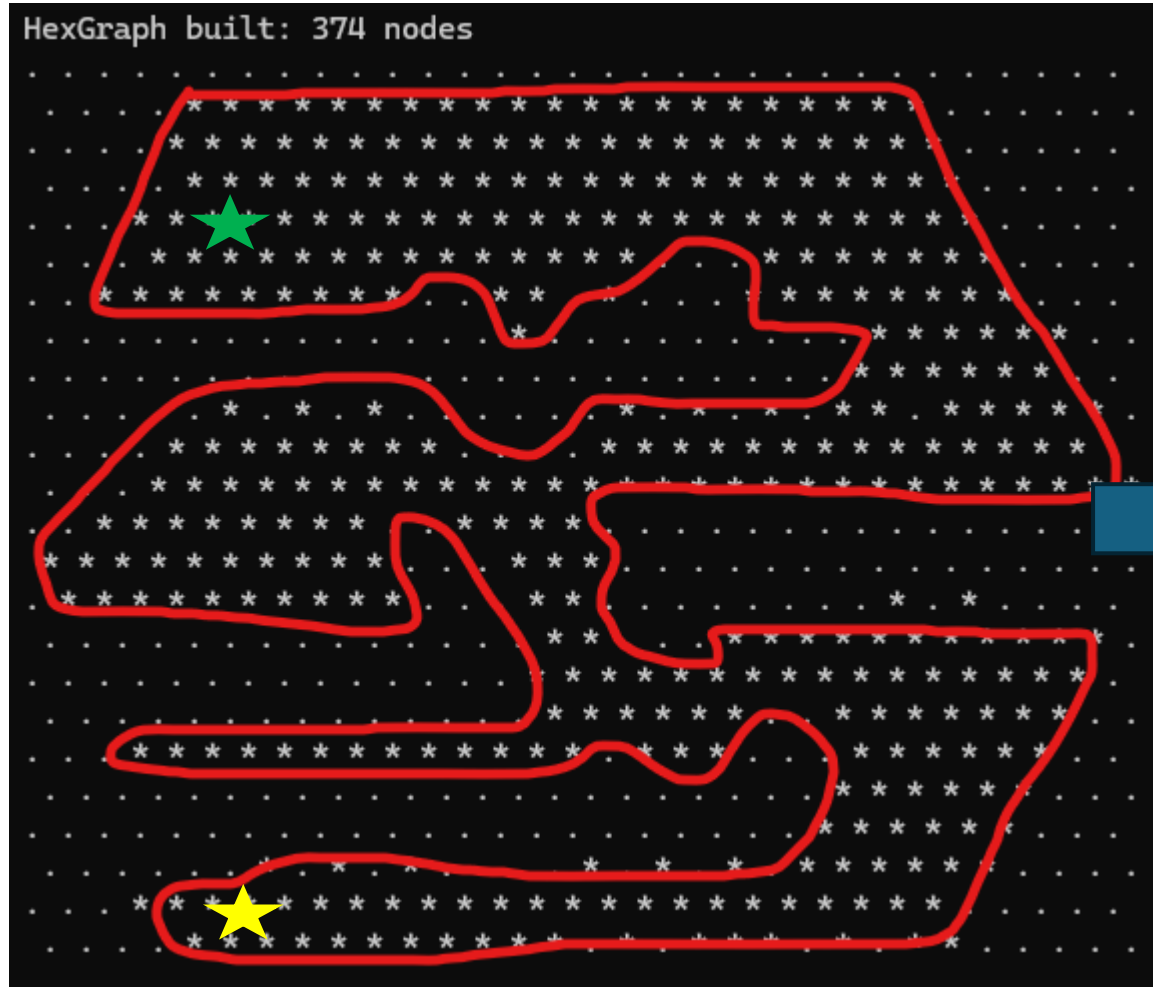


최종적으로 유니티에서 만들어진 NavMesh가 서버 메모리에 반영된 상태



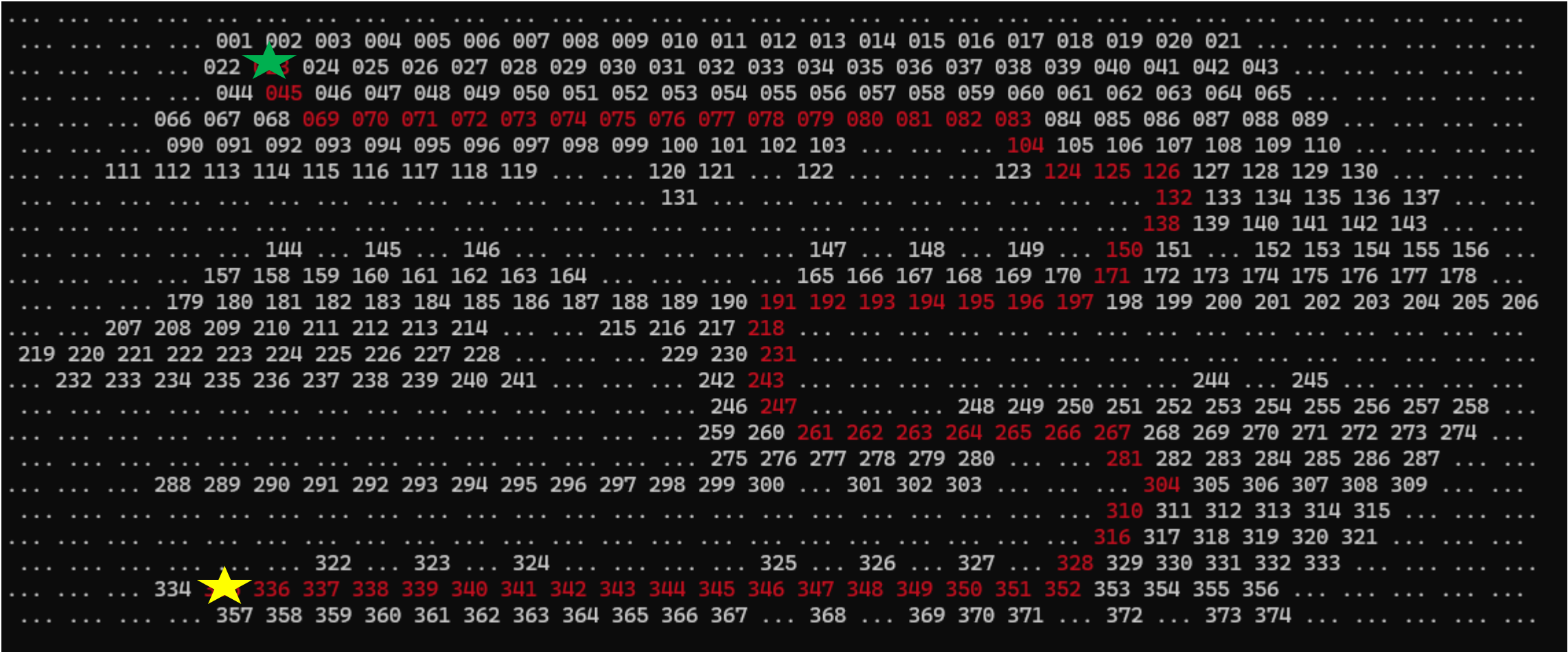
* ★ 에서 ★ 로 가는 길은?

갈 수 있는 구역(타일) 위치를 하나의 노드로 변환하여 그래프로 변경한다.



(간략한 예시)

그래프의 노드들을 이용하여 최종적으로 최단경로를 추출. A* 알고리즘 이용



실제 게임썬에서 NavMesh를 적용후 계산한 최단 거리를 실제 월드 좌표에서 표현시킴.
(검은 구체 오브젝트는 서버에서 알려주는 다음 좌표)

