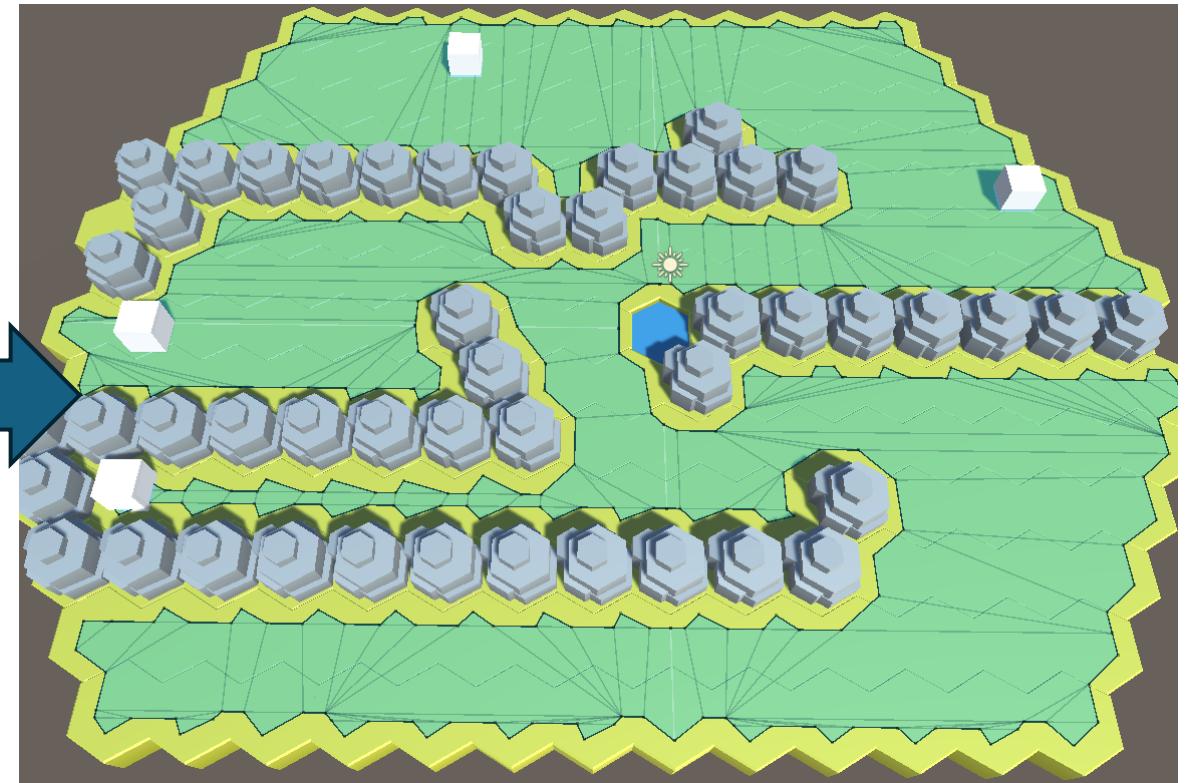
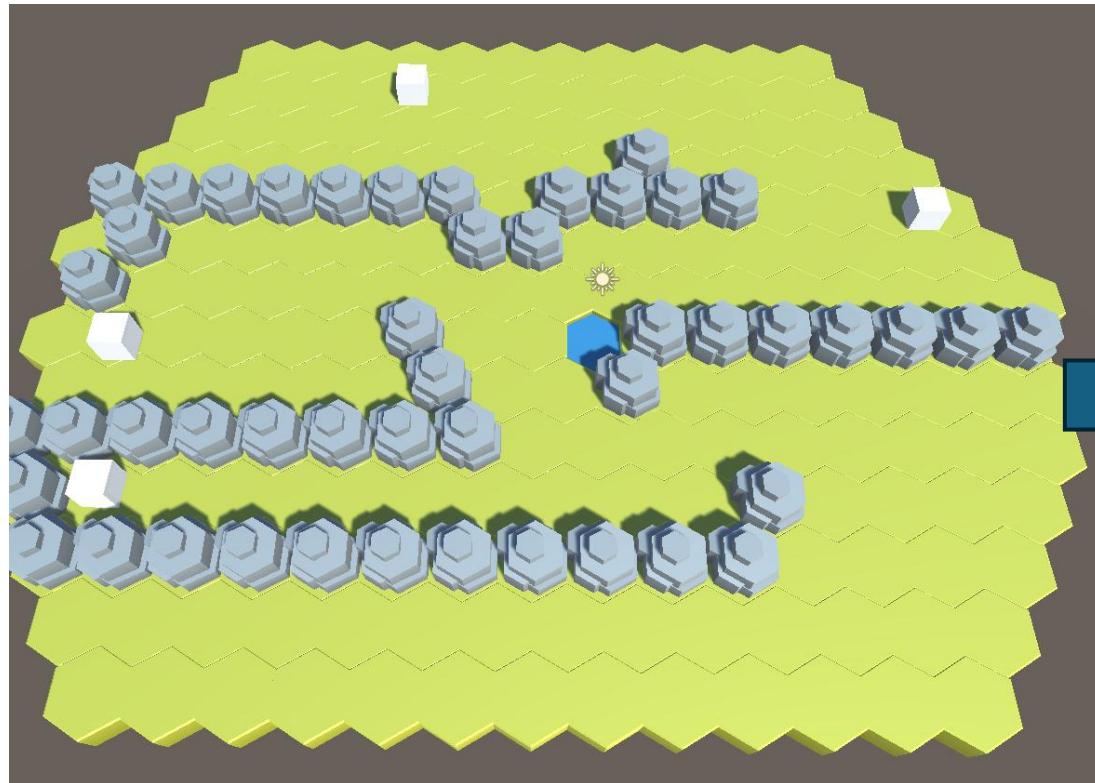


# 서버에서 어떻게 NavMesh를 이용할까?

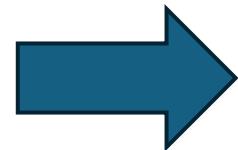
- 클라이언트 엔진에서 타일맵(지형)을 만든다.
- 해당 지형으로 엔진의 NavMesh를 굽는다.
- NavMesh 정보를 텍스트 형태로 추출하여 C++서버에서 파싱한다.
- 파싱된 정보를 그래프화하여 실제 월드 좌표로 경로를 계산한다.

엔진의 씬에서 지형을 만들고 NavMesh를 굽는다.



만들어진 NavMesh를 수치화하여 저장한다.

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



```
"navmesh": {  
    "vertices": [  
        [0.3333333, 0.08333313, 1.5],  
        [1.333333, 0.08333313, 1.5],  
        [0.6666667, 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]  
    ]  
}
```

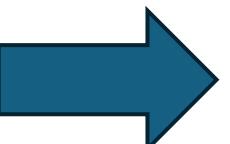
```
"navmesh": {  
    "vertices": [],  
    "triangles": [  
        [0, 1, 2],  
        [3, 4, 5],  
        [6, 7, 8],  
        [9, 10, 11],  
        [12, 13, 14],  
        [15, 16, 17]  
    ]  
}
```

\* 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;  
};
```

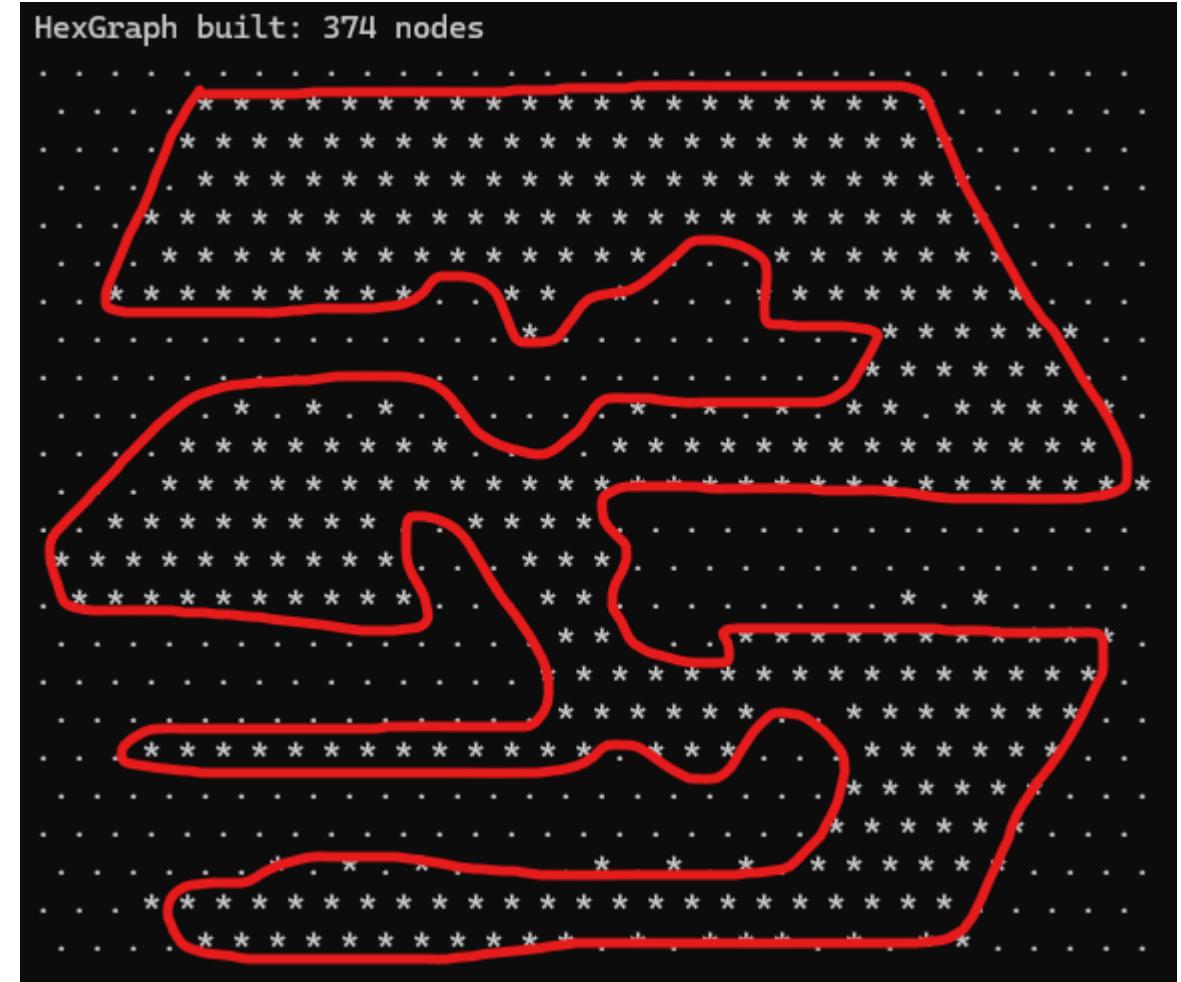
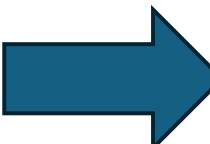
## 시각화 하여 출력 ('\*' : 갈 수 있는 길 )

```
struct MapObject
{
    string name;
    float worldX, worldY, worldZ;
    int cellX, cellY, cellZ;
};

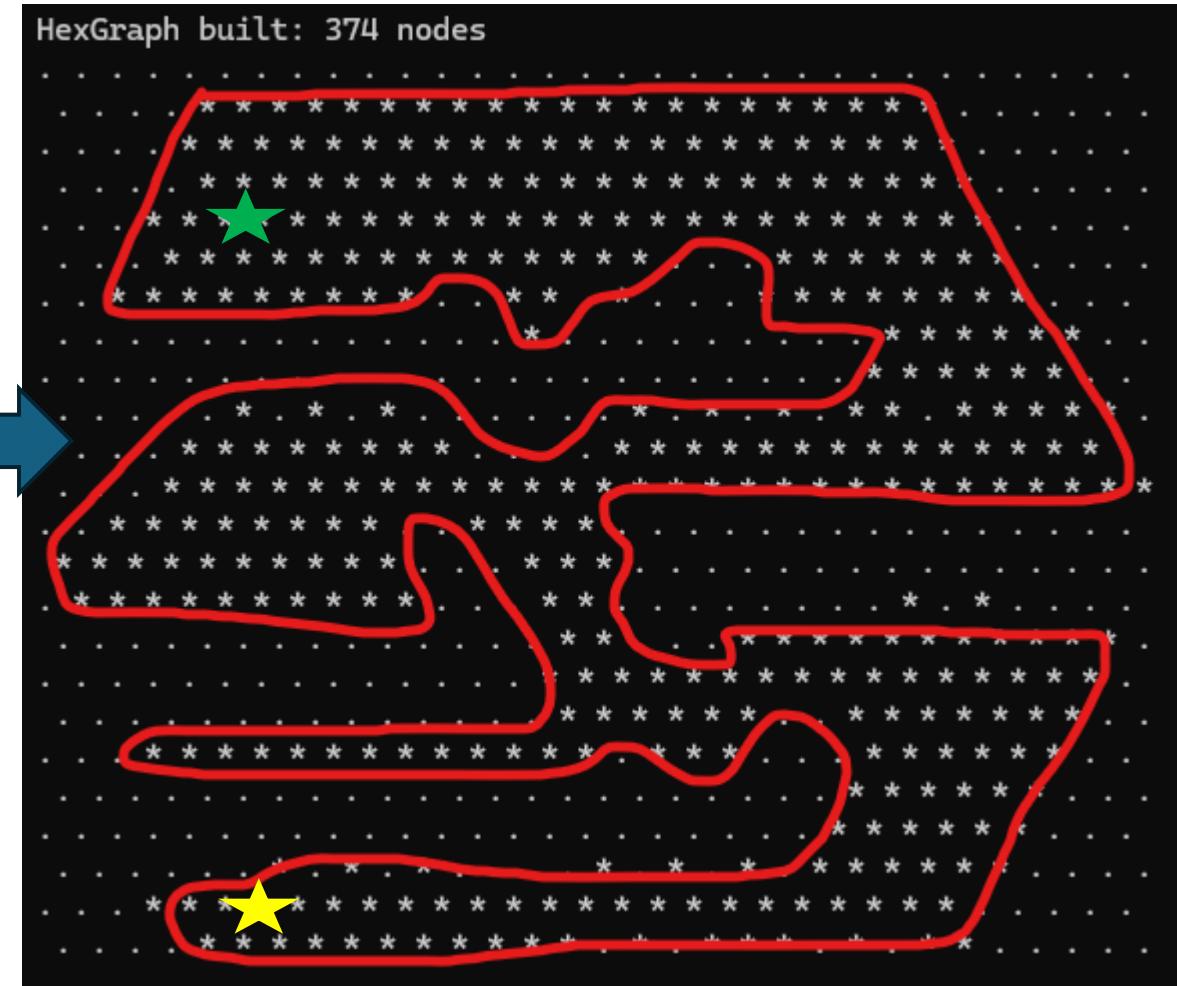
// 네비메시 데이터
struct NavMeshData
{
    struct Vertex { float x, y, z; };
    struct Triangle { int a, b, c; };

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

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

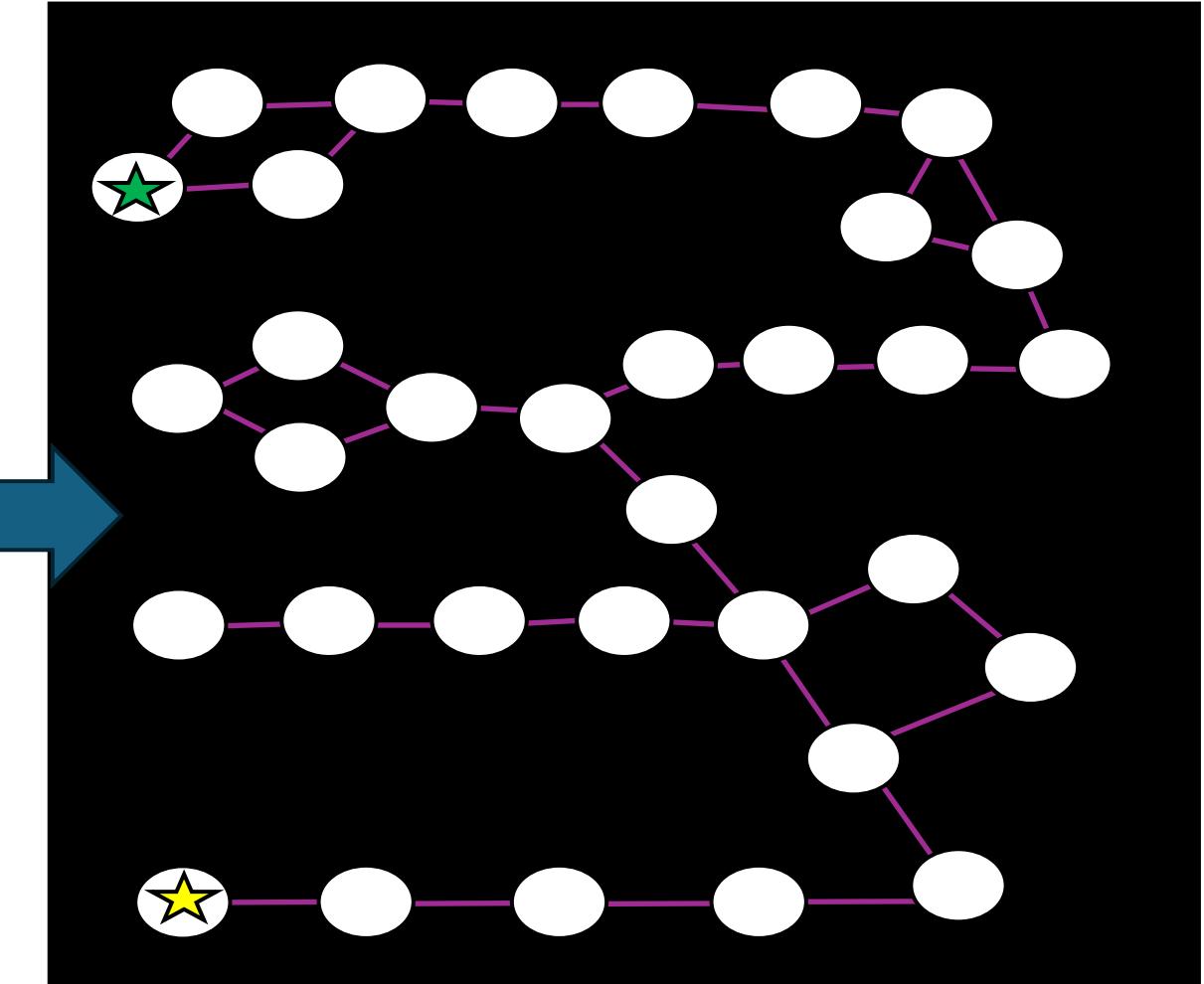


최종적으로 유니티에서 만들어진 NavMesh가 서버 메모리에 반영된 상태



\* ★ 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

.....	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	016	017	018	019	020	021	.....						
.....	022	023	024	025	026	027	028	029	030	031	032	033	034	035	036	037	038	039	040	041	042	043	.....					
.....	044	045	046	047	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063	064	065	.....					
.....	066	067	068	069	070	071	072	073	074	075	076	077	078	079	080	081	082	083	084	085	086	087	089	.....				
.....	090	091	092	093	094	095	096	097	098	099	100	101	102	103	.....	104	105	106	107	108	109	110	.....					
.....	111	112	113	114	115	116	117	118	119	.....	120	121	...	122	...	123	124	125	126	127	128	129	130	.....				
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	131	.....	.....	.....	.....	.....	132	133	134	135	136	137	.....					
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	138	139	140	141	142	143	.....						
.....	144	...	145	...	146	...	.....	147	...	148	...	149	...	150	151	...	152	153	154	155	156	.....						
.....	157	158	159	160	161	162	163	164	.....	165	166	167	168	169	170	171	172	173	174	175	176	177	178	.....				
.....	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206
.....	207	208	209	210	211	212	213	214	.....	215	216	217	218	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....			
219	220	221	222	223	224	225	226	227	228	.....	229	230	231	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....		
.....	232	233	234	235	236	237	238	239	240	241	.....	242	243	.....	.....	.....	244	...	245	.....	.....	.....	.....	.....	.....	.....		
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	246	247	.....	248	249	250	251	252	253	254	255	256	257	258	.....			
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274		
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	275	276	277	278	279	280	.....	281	282	283	284	285	286	287	.....	.....		
.....	288	289	290	291	292	293	294	295	296	297	298	299	300	...	301	302	303	.....	304	305	306	307	308	309	.....	.....	.....	
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	.....	310	311	312	313	314	315	.....	.....	.....	.....	.....		
.....	322	...	323	...	324	...	.....	325	...	326	...	327	...	328	329	330	331	332	333	.....	.....	.....	.....	.....	.....	.....	.....	
.....	334	...	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	.....	.....	.....	.....	.....
.....	.....	357	358	359	360	361	362	363	364	365	366	367	...	368	...	369	370	371	...	372	...	373	374	.....	.....	.....	.....	.....

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

