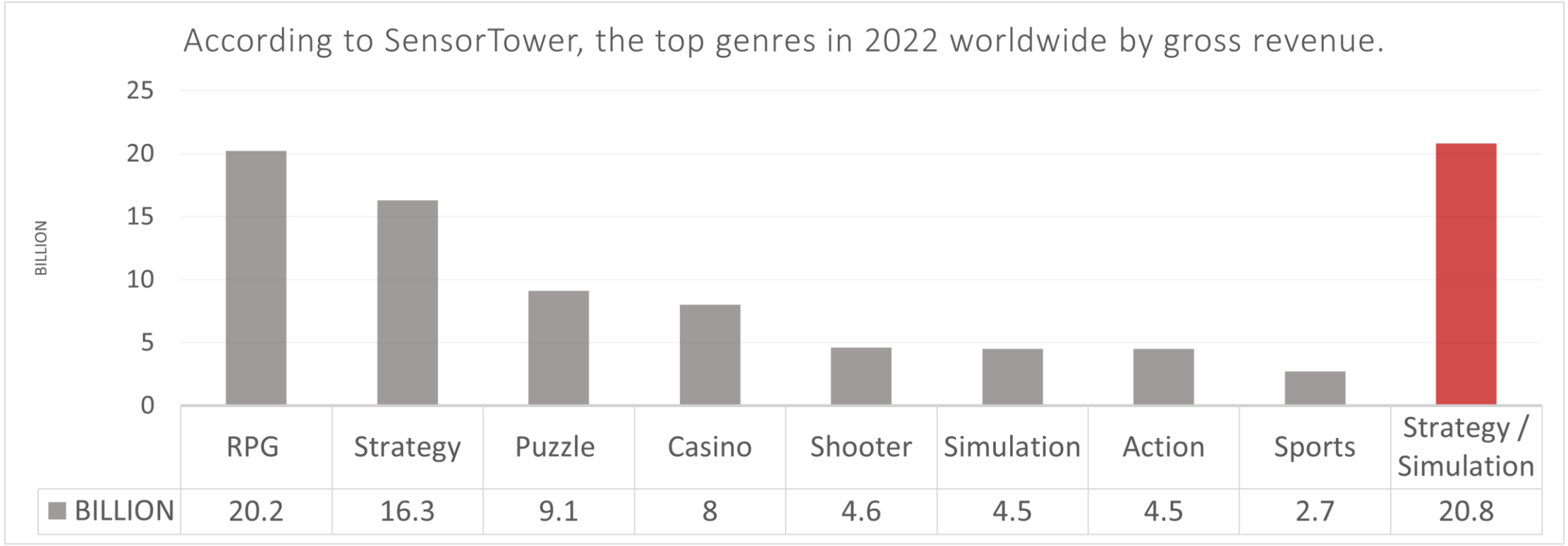
Do Not Disturb

2017180002 고선우
2020184015 박가현
2020184025 이승희

목차

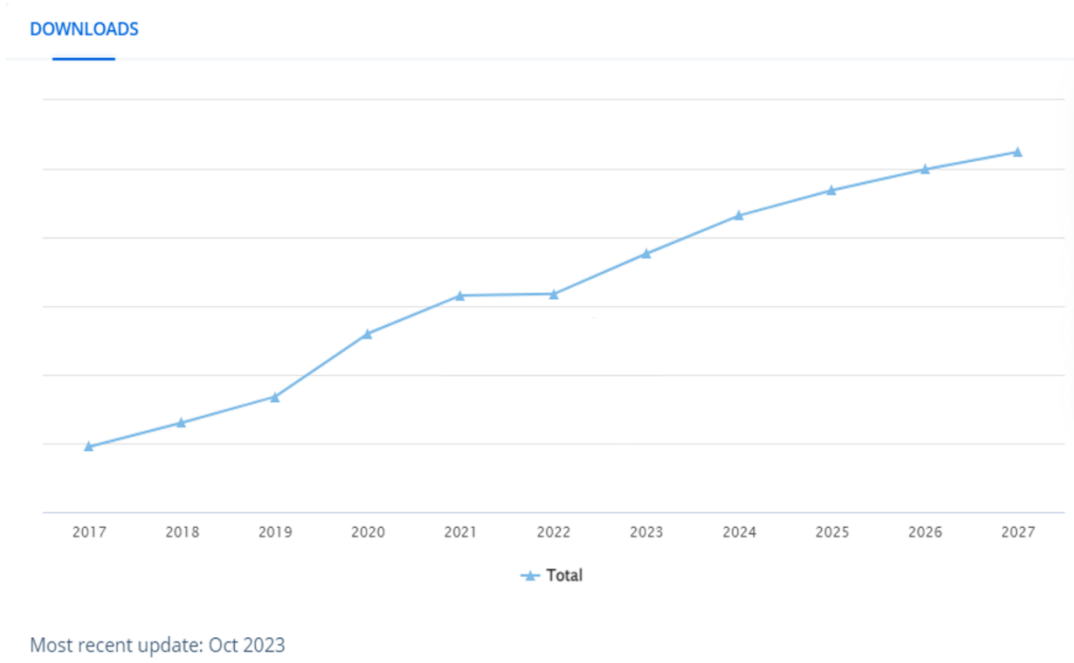
시장 환경



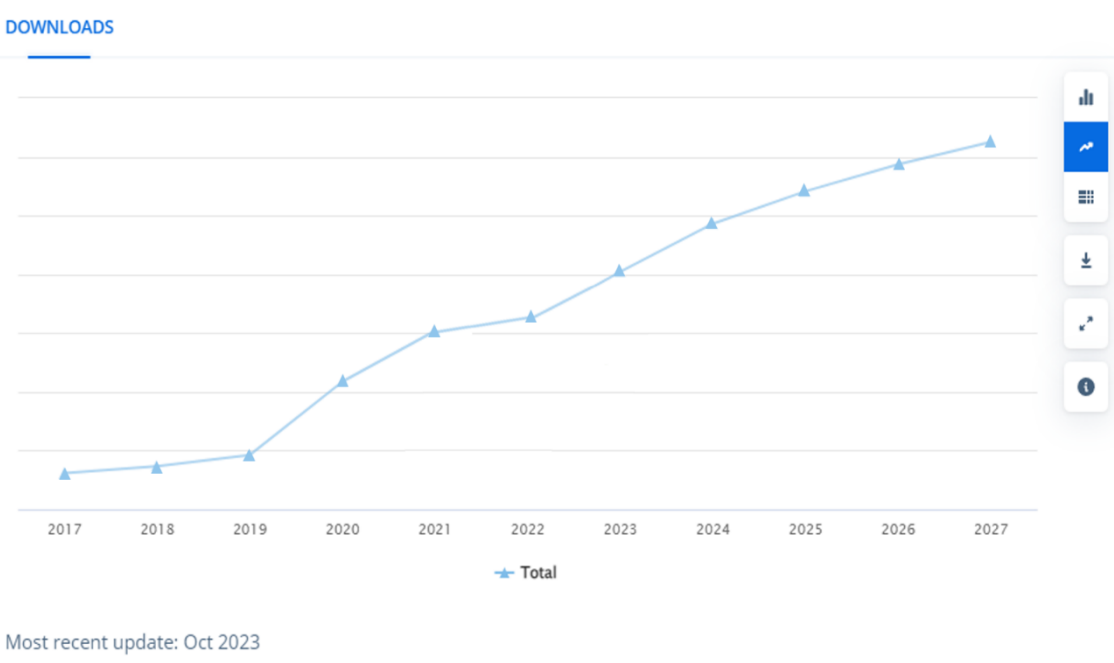
최근의 통계에서 보듯이 시뮬레이션 장르는 다른 장르에 비해 **높은 수익률**을 보인다.

장기적으로는 시뮬레이션 게임이 **높은 관심을 받고 지속적으로 성장할 것**으로 판단된다.

시장 환경



시뮬레이션 장르



전략 장르

최근의 통계에서 보듯이 시뮬레이션 장르는 다른 장르에 비해 **높은 수익률**을 보인다.

장기적으로는 시뮬레이션 게임이 **높은 관심을 받고 지속적으로 성장할 것**으로 판단된다.

게임 소개

장르 : 전략 시뮬레이션

각 지역마다 존재하는 **자원을 채취**하며 **포탑을 건설**한다.

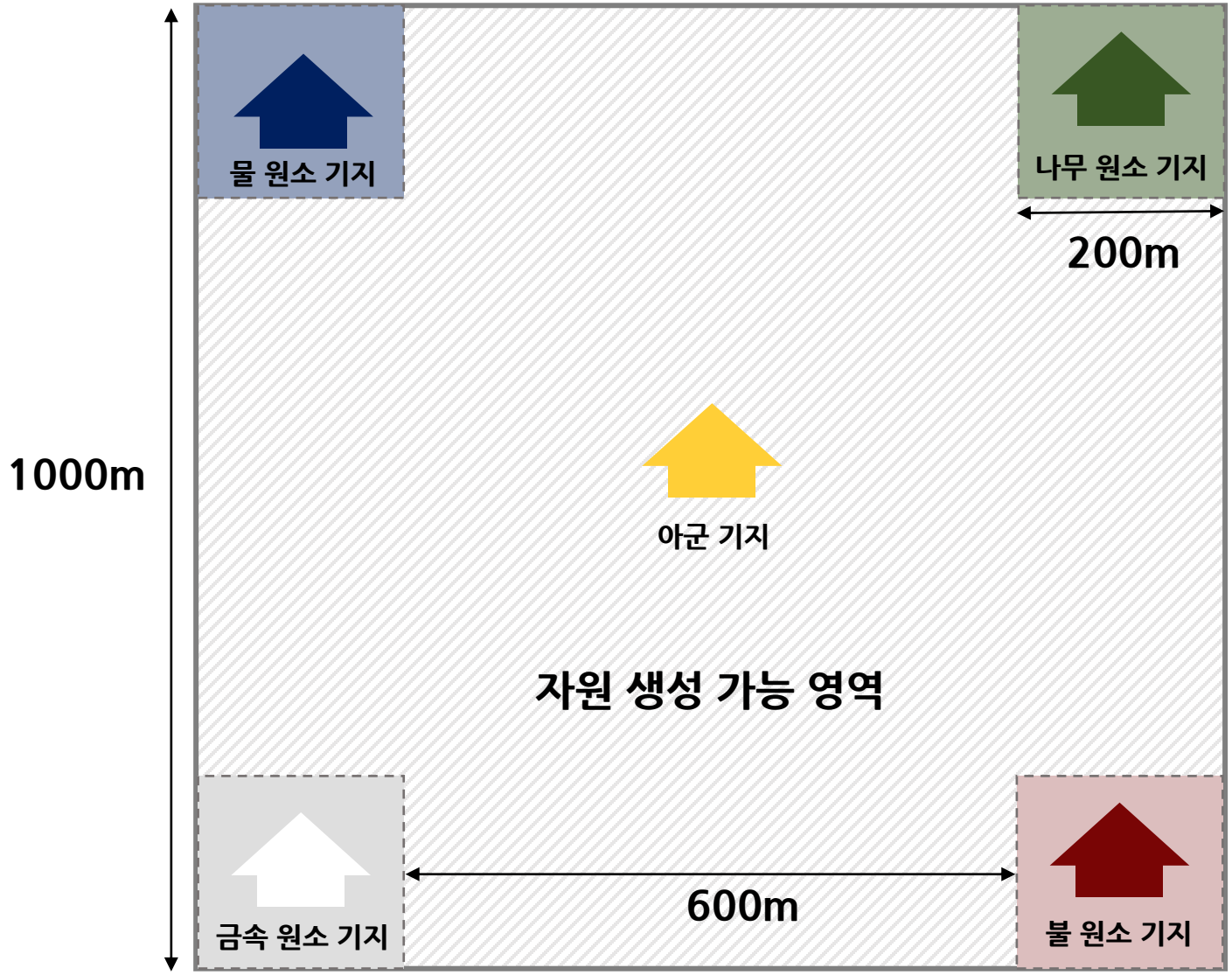
플레이어는 자원을 활용하여 포탑을 업그레이드하는 **연구를 진행**한다.

정해진 시간마다 **웨이브**가 진행되며 설치된 포탑으로 **공격 및 방어**를 한다.

게임 소개

예상 플레이 시간	1판당 약 25 ~ 30분
시점	3인칭
플레이 인원	1~2명
포탑 종류	5개 (공격 포탑 2개, 방어 포탑 3개)
적 유형	4개 (근거리, 원거리, 중간 보스, 최종보스)
자원 종류	물, 불, 금속, 나무
건물 종류	연구소, 기지, 적 기지, 포탑

게임 소개



게임 소개

[플레이어]

자원 채취, 건물 건설

웨이브 중 포탑 수리 가능

[포탑]

공격 포탑

4가지 속성의 공격 포탑(근거리, 원거리) 존재

방어 포탑

단일 , 광역 , 감속 포탑 존재

[원소]

속성 간의 약점 존재

지속적으로 생성된다.

연구 진행도에 따라 강해진다.

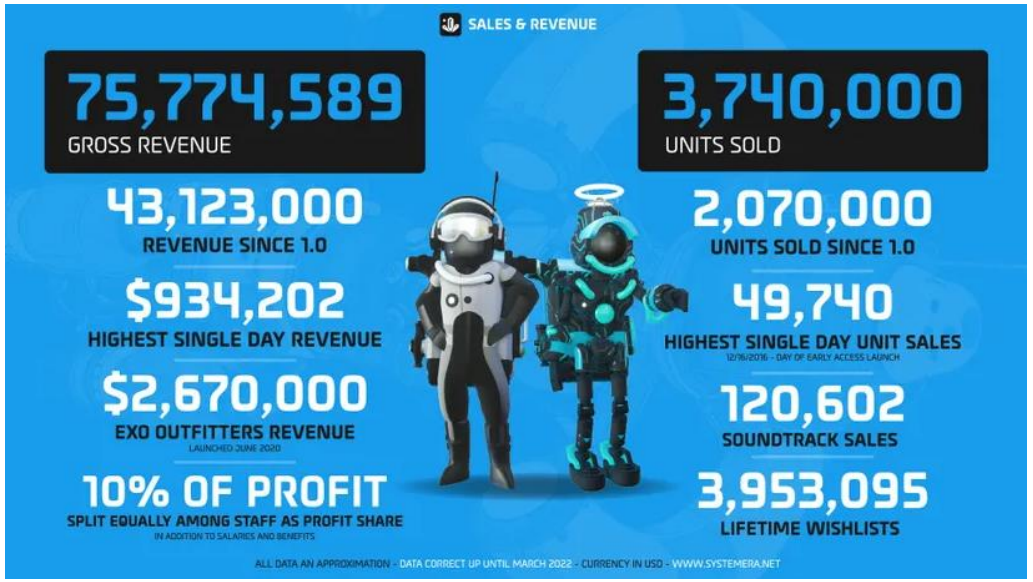
플레이어 (0.5m * 0.5m * 1.5m)	중간 보스 (0.6m * 0.6m * 2m)	최종 보스 (1.2m * 1.2m * 3m)
		
포탑 (1m * 1m * 3m)		원소 (0.25m * 0.25m * 0.5m)
		

타 게임과의 차이

아스트로니어

- 지표면/지하의 자원을 채취하여 발전하는 것이 주 목적
- 도구 제작 및 생물 연구를 통해 상위 도구, 건물을 해금하는 등 생존과 탐색

-> 반복된 게임 플레이
-> 생존 게임인데 위기감이 없다.
= 지루한 게임 플레이



개발 환경

Unity

3Ds Max

Swit

GitHub

ZBrush

Visual Studio

구현할 기술

- A* 알고리즘을 이용한 길찾기
- 툰 쉐이더 제작
- 메쉬 슬라이스
- TCP/IP 서버

역할 분담 및 개인 별 준비 현황

고선우	박가현	이승희
그래픽 리소스 제작 맵 제작 이펙트 사운드	A* 알고리즘을 이용한 길 찾기 툰 쉐이더 제작 메쉬 슬라이스 TCP/IP 서버 구현	
게임엔진1 3D 모델링 3D 애니메이션	C++,STL,3D게임프로그래밍,자료구조,알고리즘, 윈도우 프로그래밍, 컴퓨터 그래픽스, 쉐이더 프로그래밍, 네트워크 게임 프로그래밍	

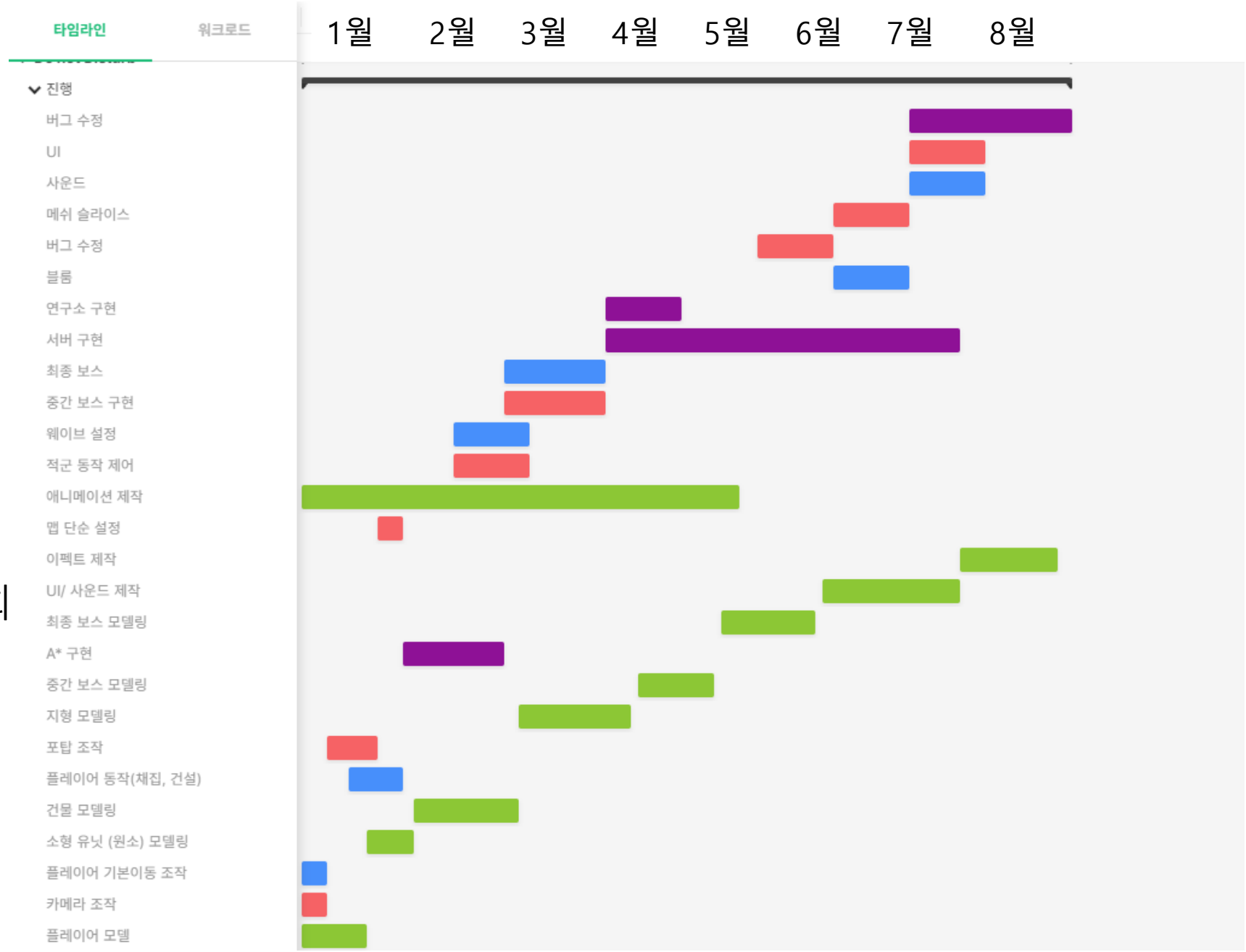
일정

박가현

이승희

고선우

박가현 + 이승희



참고 문헌

문서

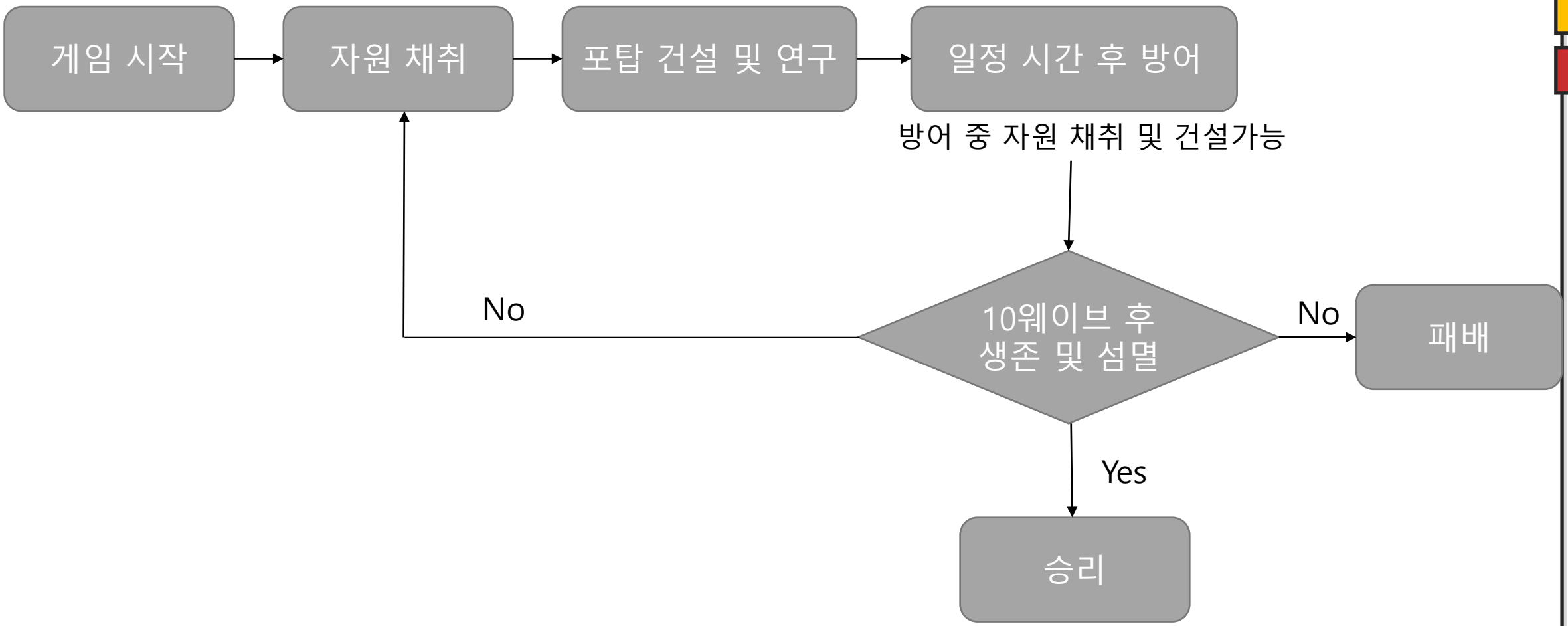
1. https://www.vintageisthenewold.com/game-pedia/what-genre-of-game-are-the-most-profitable
2. https://www.dooit.co.kr/survey/report/index/193996/2
3. https://web.archive.org/web/20220804165535/
4. https://www.gamedeveloper.com/blogs/3-years-of-astroneer-live-a-marketing-comms-post-mortem
5. https://learn.64bitdragon.com/articles/computer-science/procedural-generation/the-diamond-square-algorithm
6. https://movingai.com/jps.html
7. https://darkcatgame.tistory.com/27
8. https://pin.it/1G6hTxh
9. https://pin.it/2XJRdaa
10. https://blog.naver.com/shol9570/222224199117

사진

1. 아스트로니어
2. 리그 오브 레전드
3. 유니티
4. 스윗
5. 깃허브
6. 3d max
7. zbrush

감사합니다

부록 - 플로우차트


게임 시작
자원 채취
포탑 건설 및 연구
일정 시간 후 방어
방어 중 자원 채취 및 건설가능
10웨이브 후
생존 및 섬멸
No
No
패배
Yes
승리

부록 - 포탑 종류

근거리 유닛	원거리 유닛	단일	감속	광역
3마리씩 생산	2마리씩 생산	단일 적을 향해 강한 공격	사거리가 길다 맞은 적은 느려짐 광역 공격	많은 적을 향해 약한 공격

부록 - 웨이브에 따른 적 수

1	2	3	4	5	6	7	8	9	10
근거리 8	근거리 8 원거리 4	근거리 12 원거리 8	중간 보스 (바람) 근거리 20 원거리 16	근거리 24 원거리 20	근거리 28 원거리 24	중간보스 (땅) 근거리 36 원거리 28	근거리 40 원거리 32	근거리 44 원거리 36	보스 2 (불,물 근거리 52원거리 44

부록 - A* 알고리즘

[f(n) = g(n) + h(n)]

속도를 위해 C++로 구현하며, 네이티브 플러그인 기능을 사용해 유니티에 연결한다.

[A* 알고리즘을 이용해 경로 탐색 시, 시간을 소요하는 부분]

1. 휴리스틱 함수
>> 현재 노드에서 목표 노드까지의 예상 비용 계산.

2. 우선 순위 관리 자료구조
>> 탐색 중인 노드의 우선 순위 관리.

3. 그래프 탐색
>> 노드의 이웃 노드를 탐색하고 새로운 경로를 계산하는 작업.

부록 - A* 알고리즘

[휴리스틱 함수 - 맨하탄 거리(Manhattan distance)]

>> 장애물과 관계 없이 해당 노드에서 목표 노드까지의 거리 값을 계산해 판단한다.

이때, 휴리스틱 함수는 **허용 가능**하고 **일관적**이어야 한다.

조건
1. 유한 공간인 경우
2. 모든 노드에 대한 휴리스틱 함수 값은 양수 값 혹은 0.
3. 노드 n과 n의 이웃 노드 m에 대해 노드 n에서 목표 노드까지의 추정 비용이 n에서 m으로 가는 실제 비용 값과 노드 m에서 목표 노드까지의 추정 비용의 합보다 작거나 같아 일관성이 보장될 때.

- $h(n) \leq g(n \rightarrow m) + h(m)$

부록 - A* 알고리즘

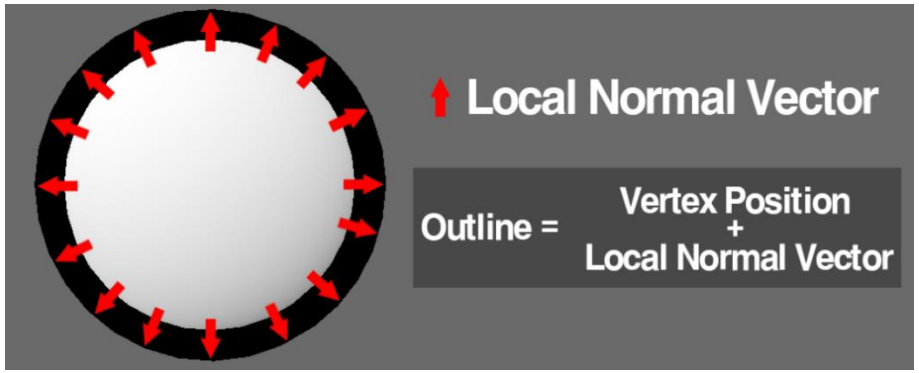
[우선 순위 관리 - 자료구조]

>> 열린 목록, 닫힌 목록

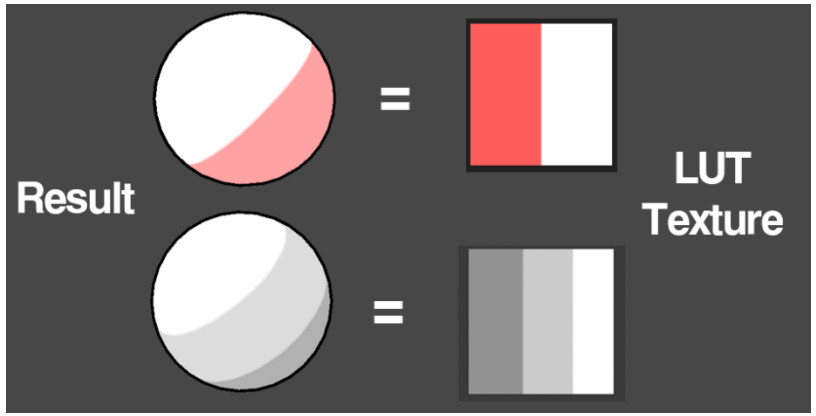
- 열린 목록: vector

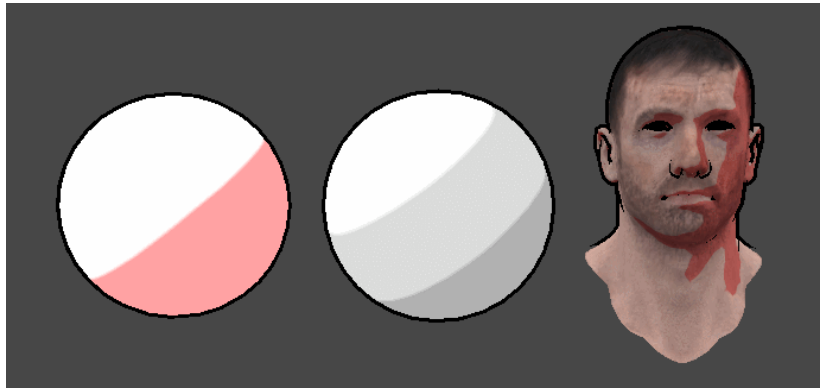
- 닫힌 목록: multiset

부록 - 툰 쉐이딩



1. 외각선 (2번 그려서 외각선 처럼 보이도록)





2. 단계별 음영 (LUT, Look Up Table 사용한다. 미리 텍스쳐를 통해 무슨색인지, 몇 단계인지 설정한다.)

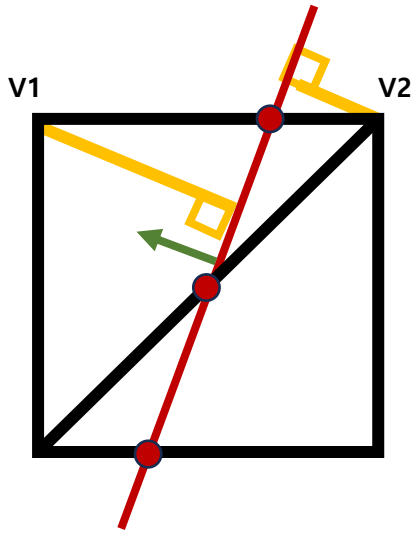
부록 - 메쉬 슬라이스

〈 프로세스 〉

1. object의 mesh를 자를 단면 기준으로 둘로 나눈다.

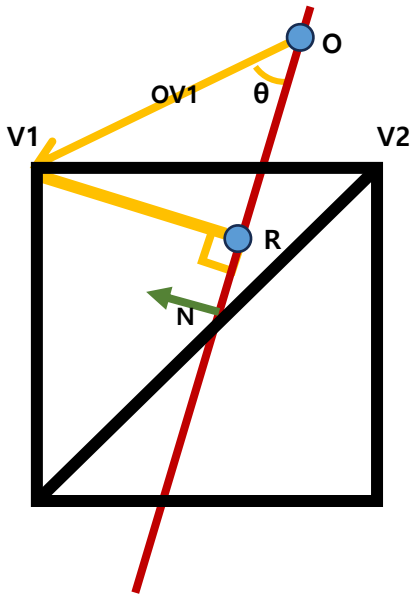
2. 나눠진 두 mesh로 새로운 mesh를 생성한다.

3. 잘려진 단면을 sub mesh로 만들어 채운다.



절단면과 메쉬의 교차점 찾기

(내적)



$$\overrightarrow{ov_1} \cdot \overrightarrow{N} = \left|\overrightarrow{ov_1}\right| \times \left|\overrightarrow{N}\right| \times \cos\theta \left(\overrightarrow{N} \text{ 은단면이 바라보는 방향}\right)$$

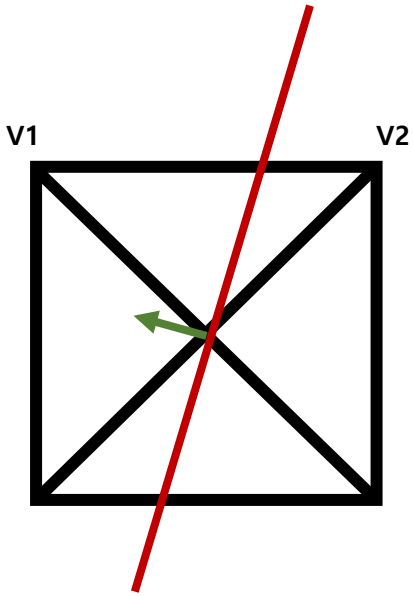
위의 공식을 바탕으로 선분R,V1 의 길이 구하기 -> v1, v2의 단면과의 거리 구할 수 있음.

v1v2 벡터에 v1:v2 비율 만큼을 곱해주어 교차점 위치 찾기

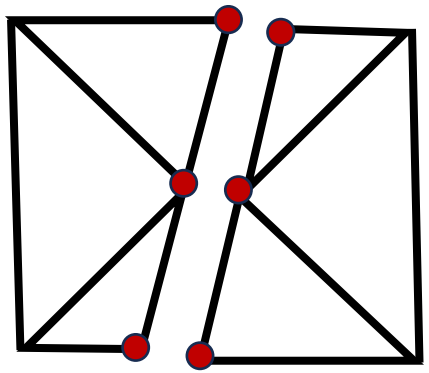
부록 - 메쉬 슬라이스

〈 프로세스 〉

1. object의 mesh를 자를 단면 기준으로 둘로 나눈다.
2. 나눠진 두 mesh로 새로운 mesh를 생성한다.
3. 잘려진 단면을 sub mesh로 만들어 채운다.



구한 정점 모두 이어주기

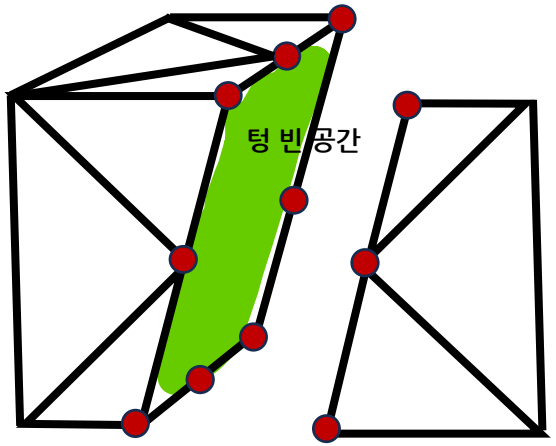


자르기

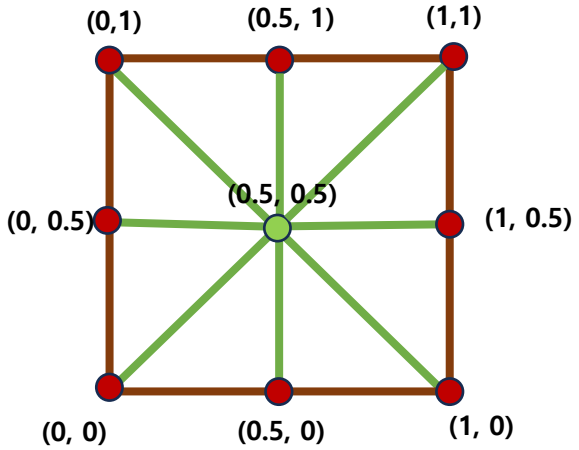
부록 - 메쉬 슬라이스

〈 프로세스 〉

1. object의 mesh를 자를 단면 기준으로 둘로 나눈다.
2. 나눠진 두 mesh로 새로운 mesh를 생성한다.
3. **잘려진 단면을 sub mesh로 만들어 채운다.**



자른 모양



모든 정점의 위치를 더하고 정점들의 개수 만큼 나눠주어 중앙 정점의 위치 구하기

중앙의 점을 texture의 중앙으로 설정

각 정점이 떨어져 있는 값에 따라서 uv를 지정하기

(단면 방향 기준으로 2차원 좌표계 만들어 사용)