# 2팀

# WATERPARK

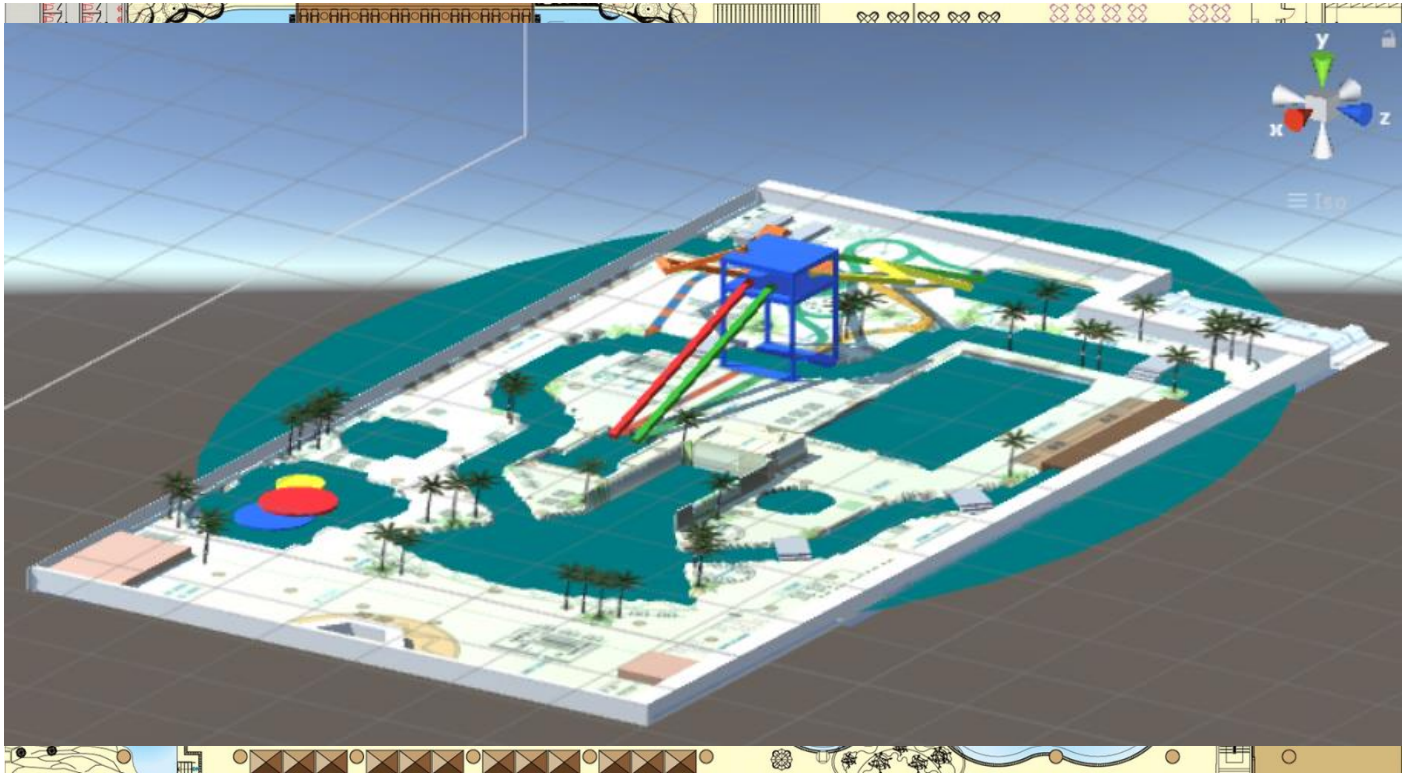
구해린 박희정 도예진 차진혁 신규화

# 목차

# 중간 단계

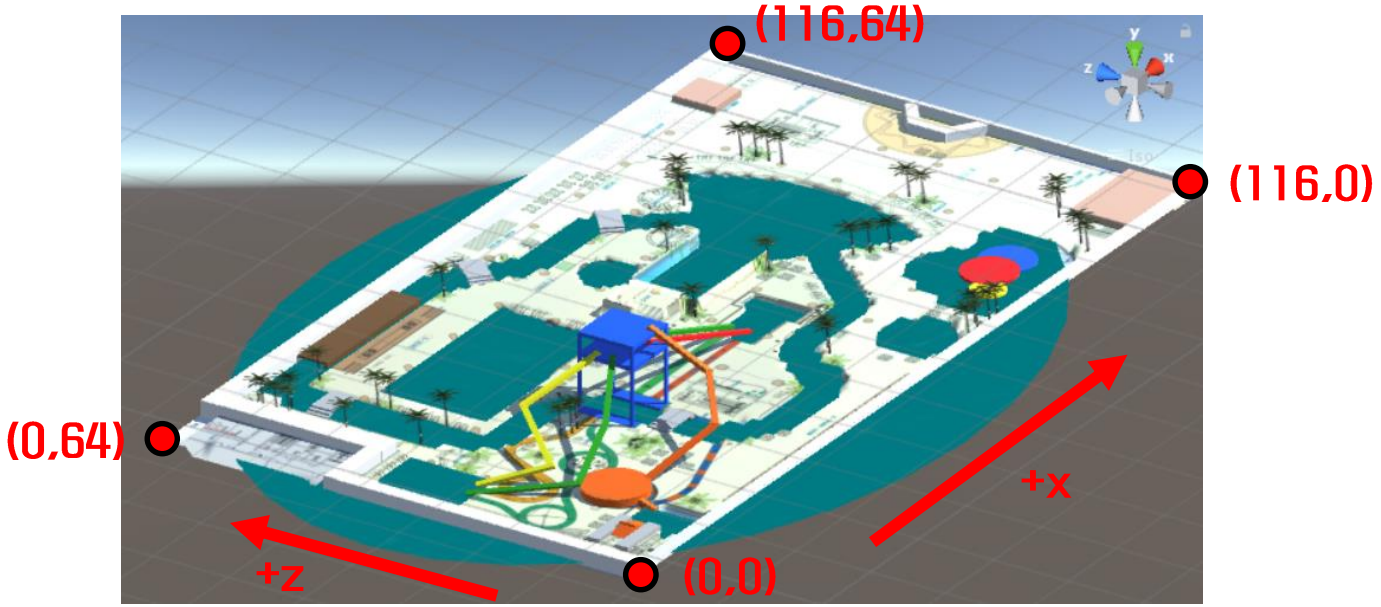
# 3D 맵 제작

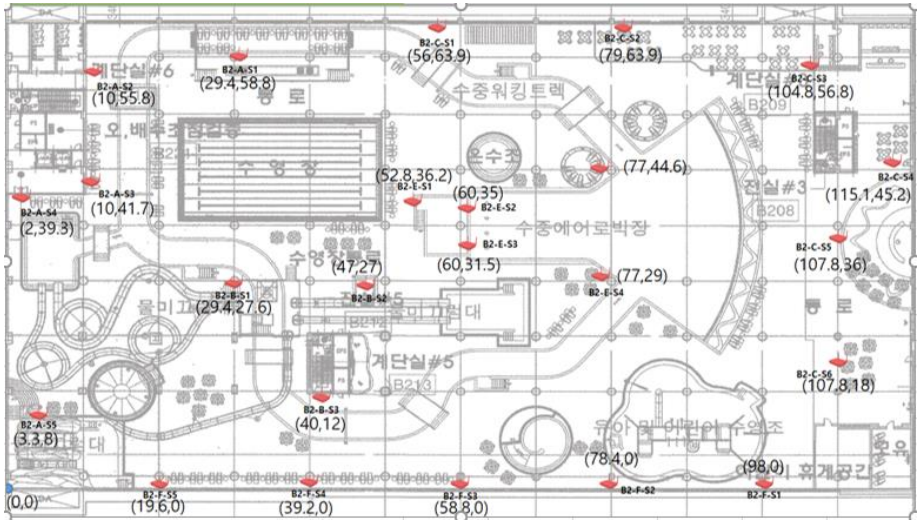
unity 유니티 사용



# 3D 맵 제작




(116,64)
(116,0)
(0,64)
(0,0)
+x
+z


B2-C-S1 (56,63.9)
B2-C-S2 (79,63.9)
B2-A-S1 (29.4,58.8)
B2-A-S2 (10,55.8)
B2-C-S3 (104.8,56.8)
B2-A-S3 (10,41.7)
B2-A-S4 (2,39.3)
(52.8,36.2)
B2-E-S1
(60,35)
B2-E-S2
(77,44.6)
B2-C-S4 (115.1,45.2)
B2-E-S3 (60,31.5)
(47,27)
(77,29)
B2-E-S4
B2-C-S5 (107.8,36)
B2-B-S1 (29.4,27.6)
B2-B-S2
B2-C-S6 (107.8,18)
B2-B-S3 (40,12)
B2-A-S5 (3.3.8)
(78,4,0)
(98,0)
B2-F-S2
B2-F-S1
(0,0)
B2-F-S5 (19.6,0)
B2-F-S4 (39.2,0)
B2-F-S3 (58.8,0)

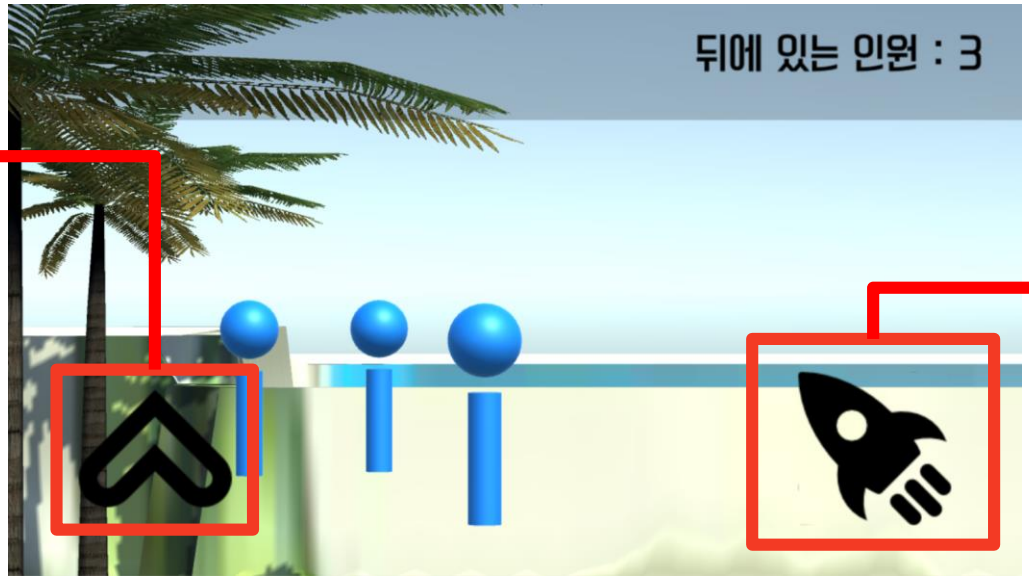
# 메인카메라 이동 및 자이로 구현



## 메인카메라 이동

**순간이동**

-> 기업체에서 어플리케이션 이용자의 gps 데이터 값을 받기 전, VR,AR 기능 테스트를 위한 버튼




**전진**

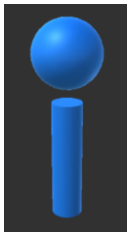
-> 어플리케이션 사용자의 위치에 따라 움직이고 있는 이용객 캐릭터가 눈앞에 제대로 출력되는지 확인하기 위한 버튼

## 자이로 구현

-> quaternion회전을 이용하는 유니티 내장 자이로를 이용

# 방문객 캐릭터 구현





## 유니티 Prefab 사용

### 〈 새로 만든 데이터 〉

txt - 메모장

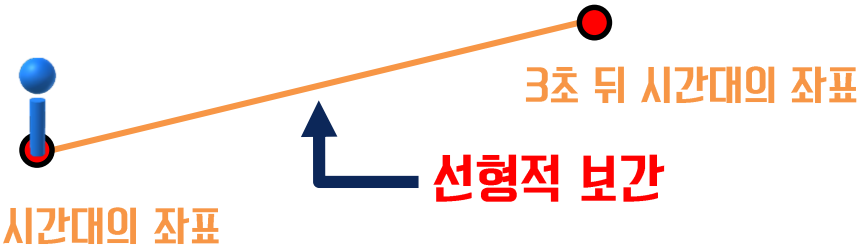
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

| | | | | |
|---|---|---|---|---|
| A0:E6:F8:34:8B:B6 | 86.86173698 | 34.97421271 | 20170925 | 132030 |
| A0:E6:F8:34:8B:B6 | 86.86173698 | 34.97421271 | 20170925 | 132033 |
| A0:E6:F8:34:8B:B6 | 83.25575681 | 31.4564004 | 20170925 | 132036 |
| A0:E6:F8:34:8B:B6 | 83.25575681 | 31.4564004 | 20170925 | 132039 |
| A0:E6:F8:34:8B:B6 | 84.83691768 | 37.22553889 | 20170925 | 132042 |
| A0:E6:F8:34:8B:B6 | 87.49696282 | 35.41081668 | 20170925 | 132045 |
| A0:E6:F8:34:8B:B6 | 87.49696282 | 35.41081668 | 20170925 | 132048 |
| A0:E6:F8:34:8B:B6 | 84.96202946 | 36.1742537 | 20170925 | 132051 |
| A0:E6:F8:34:8B:B6 | 84.96202946 | 36.1742537 | 20170925 | 132054 |
| A0:E6:F8:34:8B:B6 | 84.01435224 | 37.06611403 | 20170925 | 132057 |
| A0:E6:F8:34:8B:B6 | 91.55312064 | 37.23593995 | 20170925 | 132101 |
| A0:E6:F8:34:8B:B6 | 91.55312064 | 37.23593995 | 20170925 | 132104 |
| A0:E6:F8:34:8B:B6 | 83.48149848 | 31.86832465 | 20170925 | 132107 |
| A0:E6:F8:34:8B:B6 | 83.48149848 | 31.86832465 | 20170925 | 132110 |
| A0:E6:F8:34:8B:B6 | 89.7878814 | 37.36204144 | 20170925 | 132113 |
| A0:E6:F8:34:8B:B6 | 84.41164018 | 32.98039433 | 20170925 | 132116 |
| A0:E6:F8:34:8B:B6 | 84.41164018 | 32.98039433 | 20170925 | 132119 |
| A0:E6:F8:34:8B:B6 | 97.4109147 | 34.45038416 | 20170925 | 132122 |
| A0:E6:F8:34:8B:B6 | 97.4109147 | 34.45038416 | 20170925 | 132125 |
| A0:E6:F8:34:8B:B6 | 86.54213905 | 34.86391225 | 20170925 | 132128 |
| A0:E6:F8:34:8B:B6 | 84.99319065 | 36.20987131 | 20170925 | 132131 |
| A0:E6:F8:34:8B:B6 | 84.99319065 | 36.20987131 | 20170925 | 132135 |
| A0:E6:F8:34:8B:B6 | 83.26180883 | 35.58737378 | 20170925 | 132138 |
| A0:E6:F8:34:8B:B6 | 83.26180883 | 35.58737378 | 20170925 | 132141 |

### 〈 데이터 이동 〉

주어진 데이터는 사람의 위치가 3초 단위로 전송

-> 데이터를 그대로 사용 X

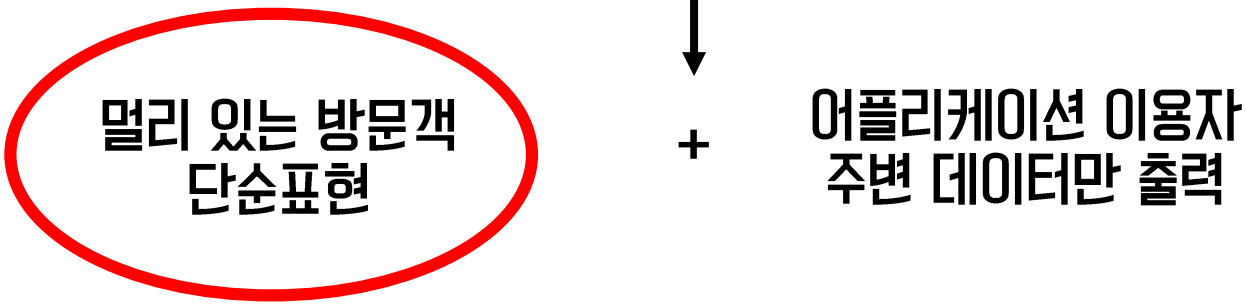



-> 시간 변화에 따라 사람이 자연스럽게 움직이는 것처럼 구현

# 일정 거리 밖 방문객 단순 표현



주어지는 방문객 데이터 수가 많아질 경우
어플리케이션 실행 시 발생할 수 있는
과부하 해결법



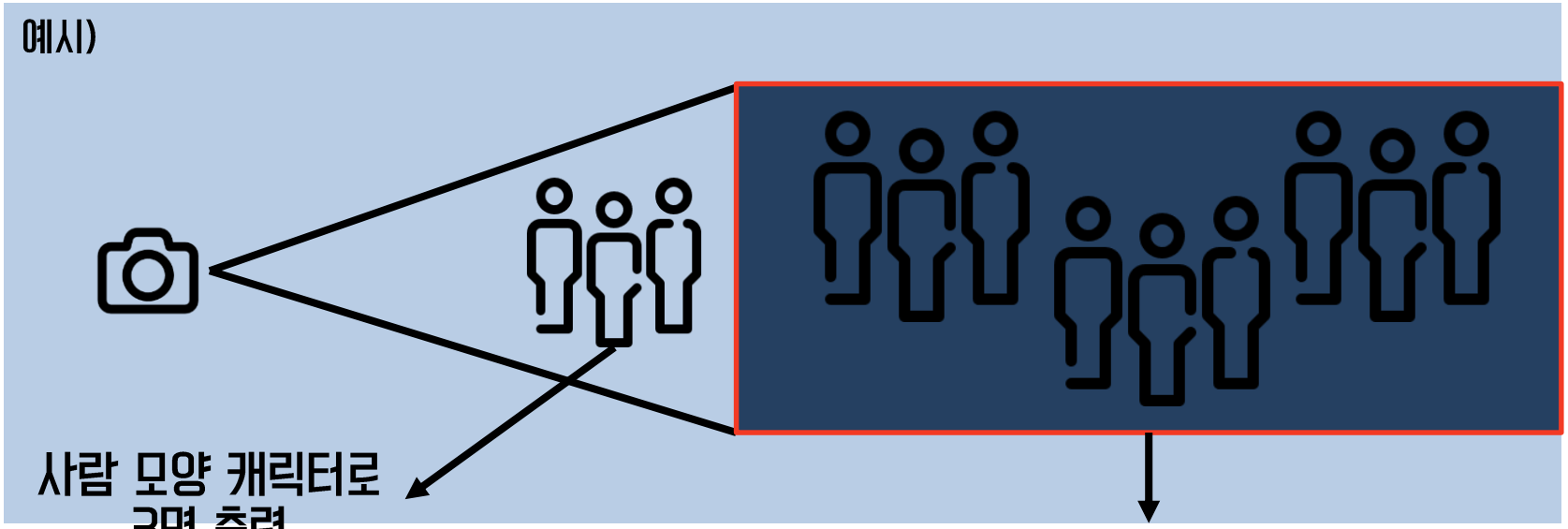
# 일정 거리 밖 방문객 단순 표현



어플리케이션 이용자의 좌표,
바라보는 방향에 상관없이
항상 시야 안에서 일정 거리 이상 떨어진
방문객 수를 합산해야 함

→

Main camera의
하위 오브젝트로
사각형모양의 오브젝트를 추가해서
항상 camera를 따라다니게 하여
사각형 오브젝트의 좌표 안에 속한
이용객 데이터 수를 합산



1 AR 전환

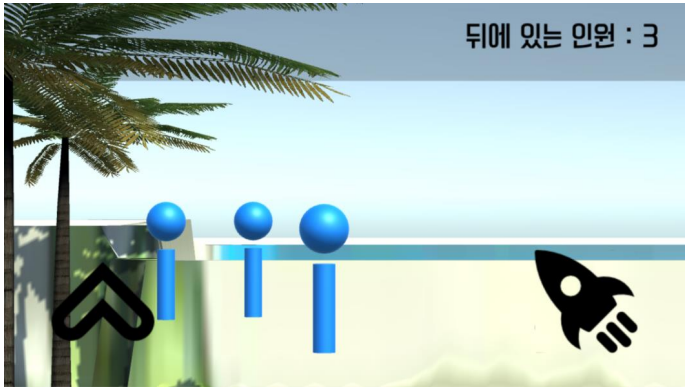
2 렌더링

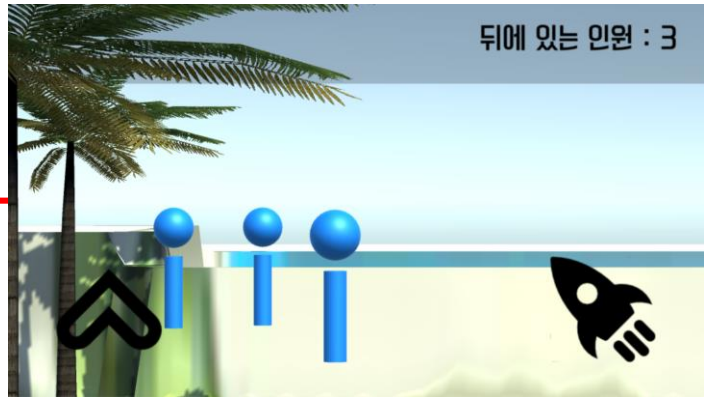
## 최종 단계 목차

3 화면 이동

4 기능 구현

# VR에서 AR 전환




**카메라** 자이로 센서와 그라운드는 그대로 사용하고
배경에 핸드폰 카메라 화면을 투영

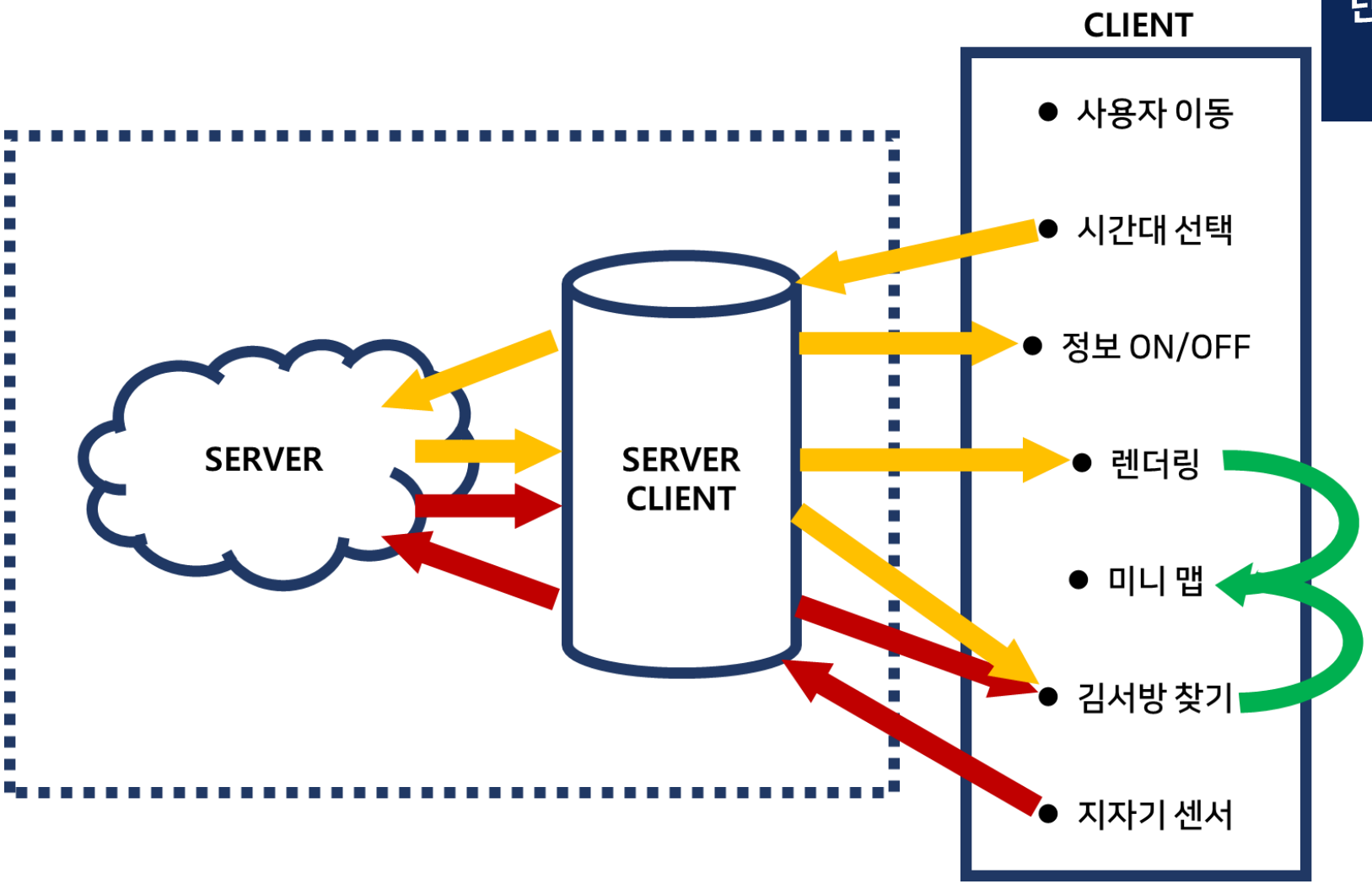
+

**방위조정** Local좌표를 World좌표로 변환

→ 이후의 김서방 찾기 기능을 위한 전처리

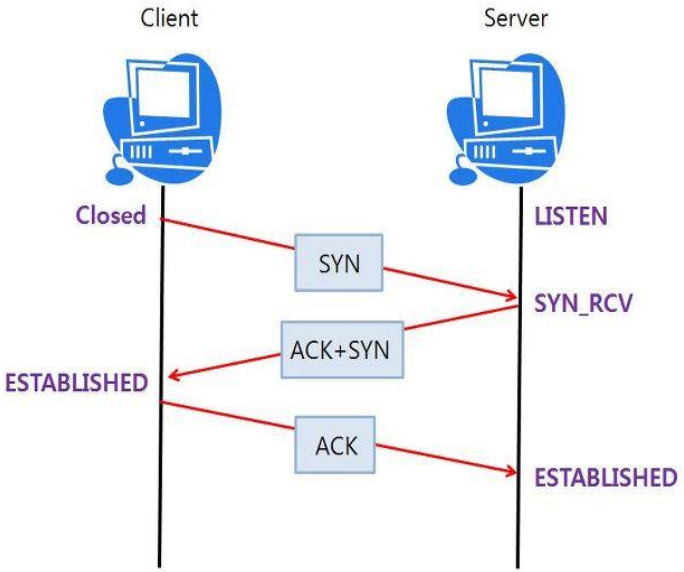
# 네트워크 구성




CLIENT
사용자 이동
시간대 선택
정보 ON/OFF
렌더링
미니 맵
김서방 찾기
지자기 센서
SERVER
SERVER
CLIENT

# 프로토콜






TCP/IP 연결

1초 마다 통신

클라이언트

시간, 방위 값

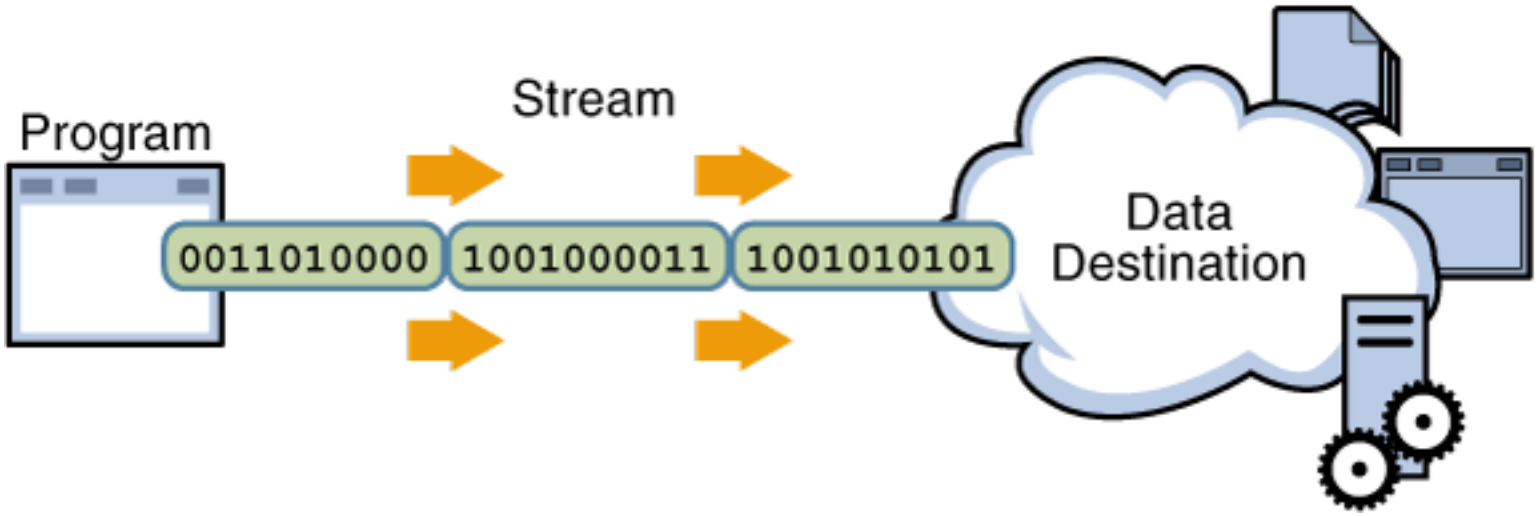
사용자 위치
정보 데이터

서버

바이트 스트림으로 송수신,
UTF-8 형식으로 읽고 씀

# 프로토콜

String 으로 저장된 데이터를
byte로 변환, stream으로 전송




Byte stream을 읽어 들여 다시
string의 형태로 저장해준다.

# 서버 구성



- 위치 정보를 행렬로 저장
- 프로토콜에 맞춘 문자열 생성
- 클라이언트로 전송

# 위치 정보를 행렬로 저장



- 비콘을 통해서 들어온 고객의 위치와 정보를 행렬 형태로 만드는 부분
- 한 단위시간(0.2초) 마다 행렬을 하나씩 만들어 디스크에 저장

| ID | X | Y | DATE | TIME |
|---|---|---|---|---|

| 0,0 | 1,0 | 2,0 | 3,0 | 4,0 | 5,0 | … | Width,0 |
|---|---|---|---|---|---|---|---|
| 0,1 | ID | 0 | 0 | 0 | 0 | … | 0 |
| 0,2 | 0 | 0 | 0 | 0 | ID | … | 0 |
| 0,3 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| … | … | … | … | … | … | … | … |
| 0,Height | 0 | 0 | ID | 0 | 0 | … | ID |

# 프로토콜에 맞춘 문자열 생성



- 0이 아닌 원소가 있으면 그 원소의 인덱스는 x, y 좌표
- 0이 아닌 값들을 모으면 ID의 존재 여부를 알 수 있음

| 0,0 | 1,0 | 2,0 | 3,0 | 4,0 | 5,0 | … | Width,0 |
|---|---|---|---|---|---|---|---|
| 0,1 | ID=6 | 0 | 0 | 0 | 0 | … | 0 |
| 0,2 | 0 | 0 | 0 | 0 | ID | … | 0 |
| 0,3 | 0 | 0 | 0 | 0 | 0 | … | 0 |
| … | … | … | … | … | … | … | … |
| 0,Height | 0 | 0 | ID | 0 | 0 | … | ID |

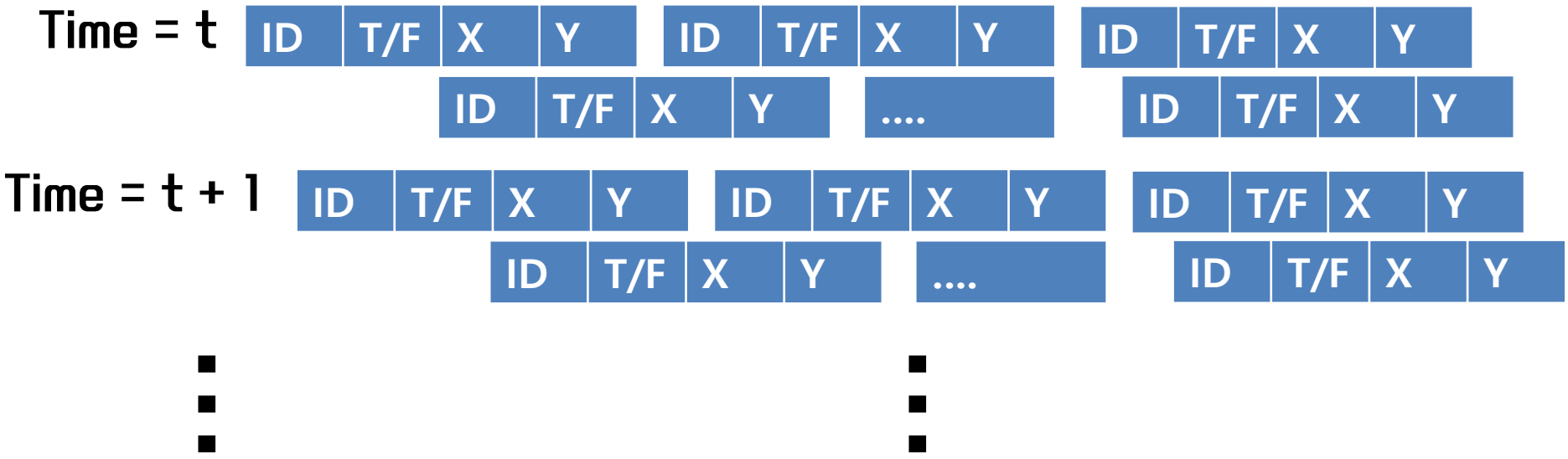
| ID = 6 | T/F = T | X = 1 | Y = 1 |
|---|---|---|---|

〈각 ID별 필요한 정보〉

# 프로토콜에 맞춘 문자열 생성



- 한 단위시간별 (ID, T/F, X, Y) * 30 을 하나의 행으로 만들어 저장한다

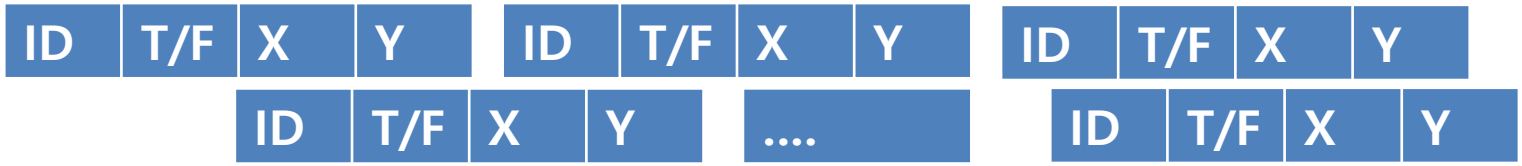


# 클라이언트로 전송



- 리스트의 한 행을 string으로 만들어 전송

Time = t



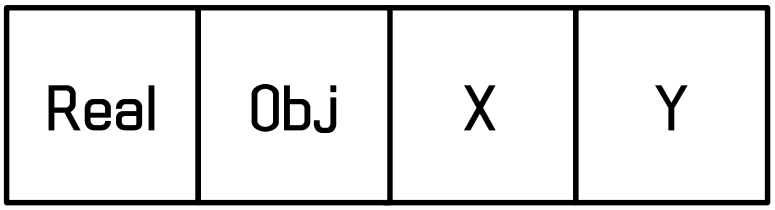

" 1 0 0 0 2 0 0 0 3 0 0 0 4 1 59.6 26.7 5 0 0 0 ……
28 0 0 0 29 1 100.5 30.7 30 0 0 0 "

CLIENT

# 클라이언트



ID 구조체

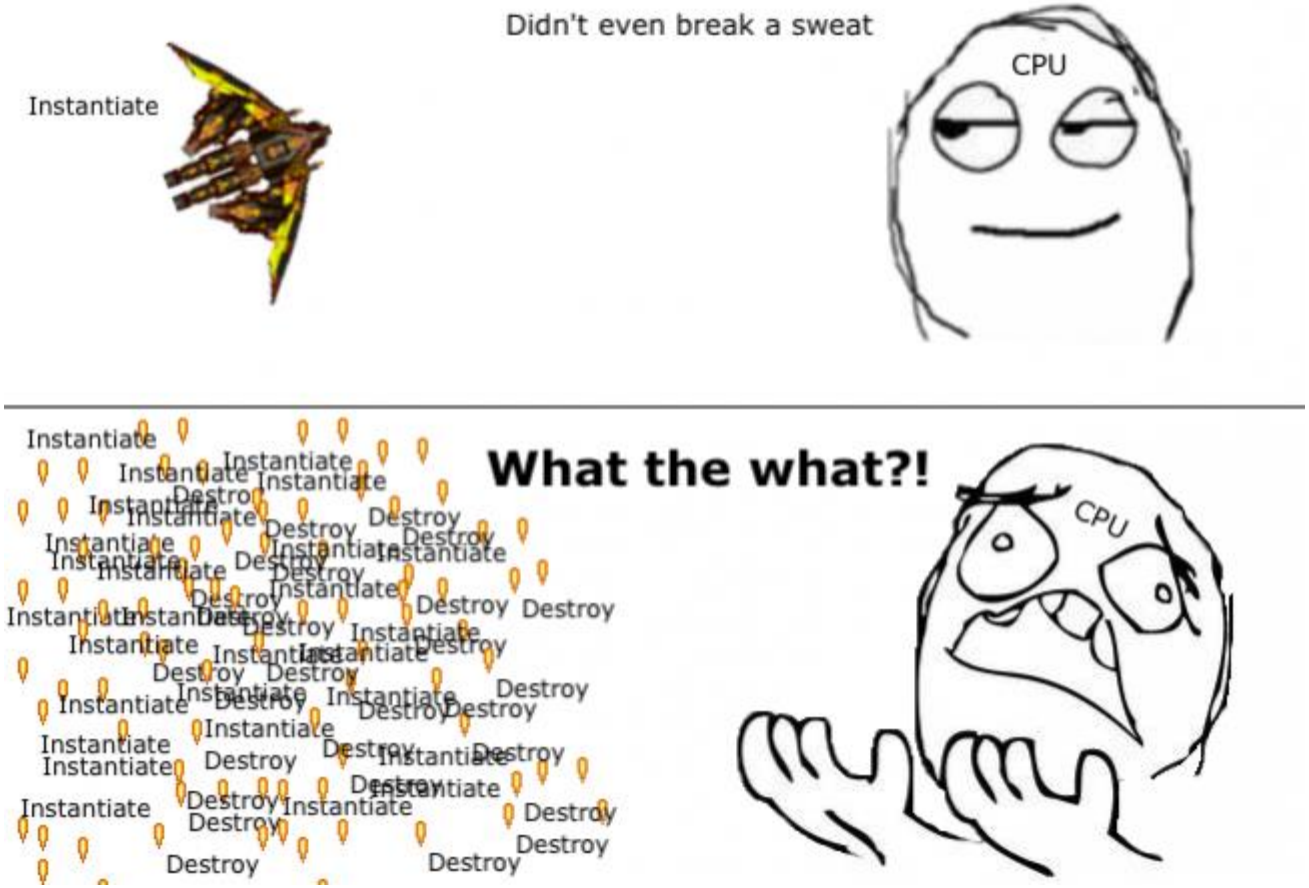



Real : 객체 존재여부
Obj : 해당 아이디의 객체
X : 객체의 X 좌표
Y : 객체의 Y 좌표

구조체 ID로 구성된 배열에 서버로 부터 받은 데이터를 저장한다.

# 오브젝트 풀링






많은 객체를 렌더링 할 때 Instance, Destroy를 이용해 렌더링을 하면 과부하가 일어남

# 오브젝트 풀링

## 만흉의 원인 : 가비지 컬렉터

가비지 컬렉터(GC)는 언제 일어날지 모른다.

- Mono의 동적 메모리 관리 때문에, 메모리 해제를 위해 GC가 자동 호출 된다.
- GC는 언제 일어날지 모른다.
- GC가 일어나면, 게임이 멈추는 현상(랙)이 발생하게 된다.
- 동적 메모리 해제가 가능한 일어나지 않도록 하는것이 GC 관리의 핵심

오브젝트 풀링

- 오브젝트(or 프리팹)의 동적 생성과 해제는 부하가 크다.
- 오브젝트가 해제되면, 언젠가는 GC가 동작하여 정리 한다 = 랙이 발생
- 오브젝트를 풀을 만들어 미리 많이 만들어 두고, 활성화/비활성화로 사용한다.
- 풀에서 가져와서 사용하고, 사용이 끝나면 비활성화 상태로 풀에 반환
- 오브젝트 풀링 사용은 선택이 아닌 필수!!

9

오브젝트 풀링을 사용해 렌더링 구현

# 오브젝트 풀링



ID 구조체

| Real | Obj | X | Y |
|---|---|---|---|

ID 구조체에서 Real 값이 true 이면 객체를 켜주고 false면 객체를 꺼준다.

객체가 켜지면, 객체에 저장된 X, Y 좌표로 객체를 이동시켜 준다.


(100,100)　　(101,100)

# RTLS 기반 사용자 이동

3 화면 이동

**기존 계획** 사용자의 비콘의 위치를 실시간으로 회사에서 제공받아서 카메라를 이동

**현재 구현 상황** 이후에 회사에서 데이터를 받았을 시 바로 적용할 수 있도록 구현

현재는 사용자를 고정시킨 상태

위치
데이터파일

실시간으로 서버에서 전송

# UI 구현



현재 시간을 보여줌

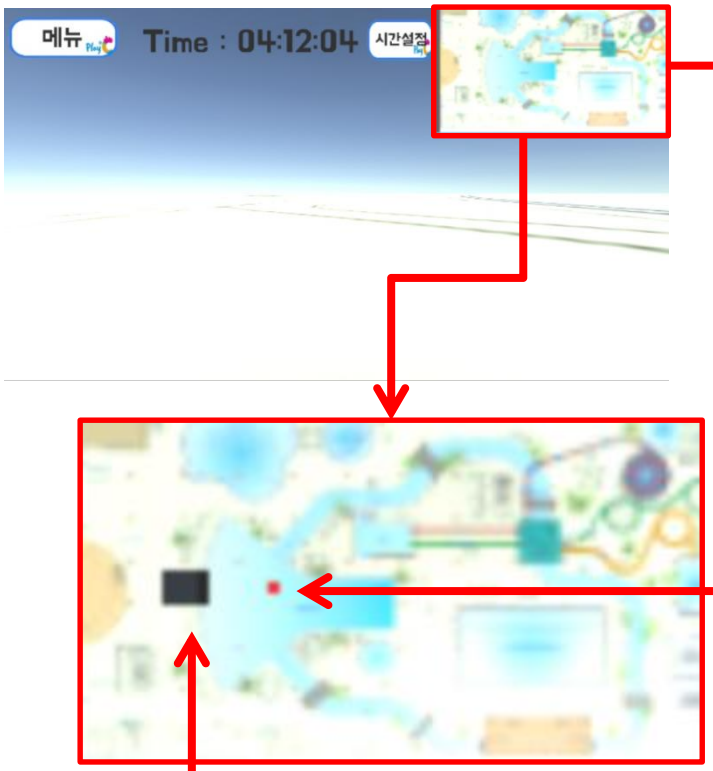
미니맵을 통해 지도를 보여줌


시간을 설정해 줄 수 있다.

기능을 선택할 수 있는 메뉴 버튼

# 미니 맵 구현






## 미니맵

-> second camera를 삽입. 위에서 아래를 보도록 해서 사용자가 언제든지 지도를 확인할 수 있도록 하고, 자신의 위치도 확인 할 수 있도록 한다.

## 사용자

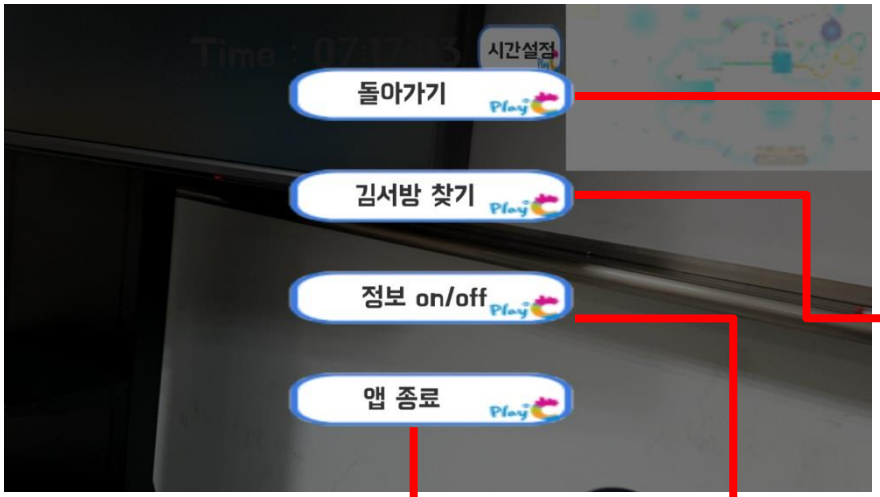
-> 카메라 위에 게임 오브젝트를 추가해주고 레이어를 변경해줘 미니 맵에만 나타나도록 해줘, 사용자의 위치를 실시간으로 알 수 있도록 해준다.

## 김서방

-> 다른 오브젝트는 미니 맵에 안 나타나도록 해주고, 김서방 프리팹에 하위로 들어가있는 오브젝트만 미니 맵에 띄워지도록 한다.

# UI 구현






기존 화면으로 돌아가게 됨

ON일 때 아이디 입력 화면으로 이동한다

OFF일 때 기능이 꺼지게 된다.

정보가 on/off 된다.

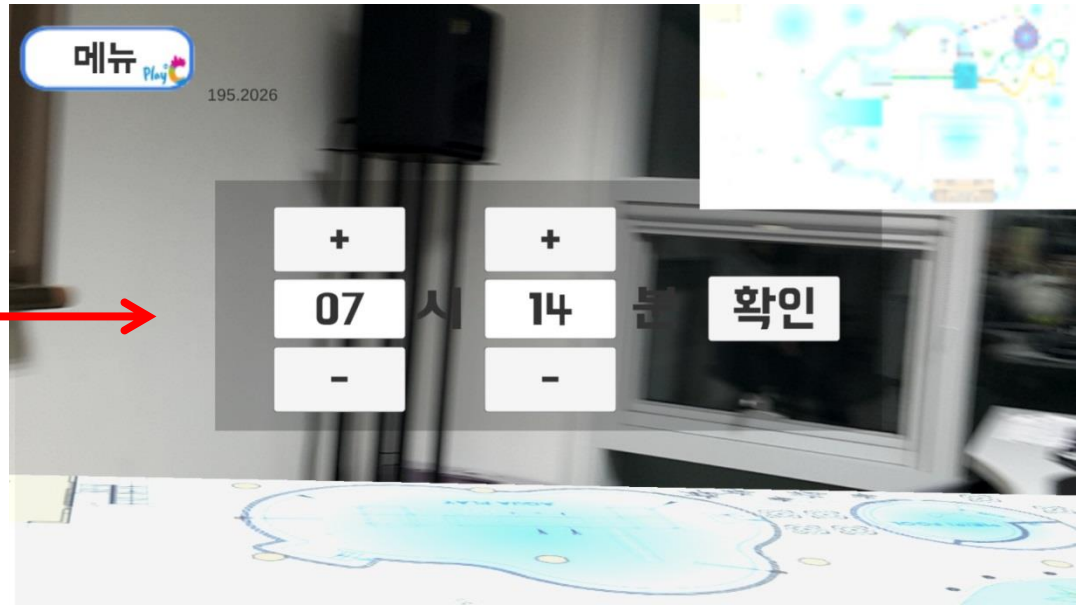
앱이 종료됨

# 타임머신 (시간대 선택)



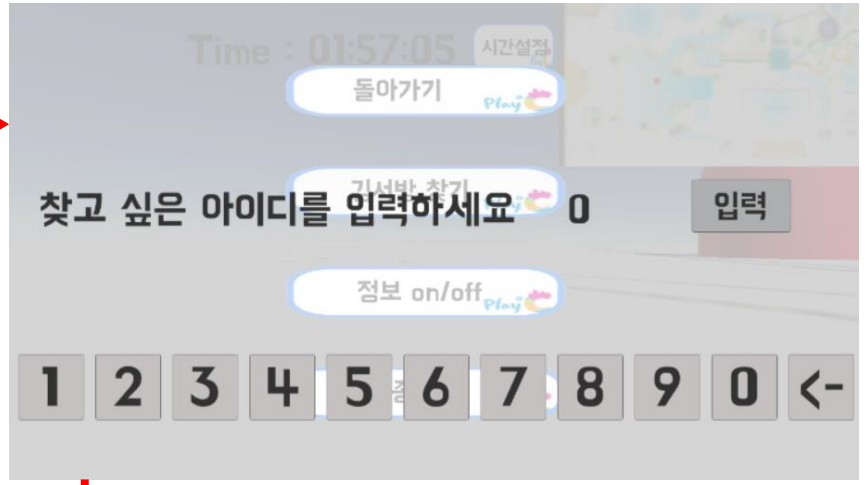
## 타임머신 과정

현재 데이터 : 13시 20분 30초 ~ 14시 14분 56초


원하는 시간대를 선택 -> 확인을 누름 -> 시간의 값을 string형의 hhmmss값으로 생성 -> 서버로 넘겨줌 -> 현재 시간이 선택한 시간으로 바뀜

# UI 구현


ID 선택 -> 서버로 ID를 전송

서버에서는 김서방의 위치를 보내줌

# 김서방 찾기 (선택한 사람 찾기)



## 김서방 찾기 과정

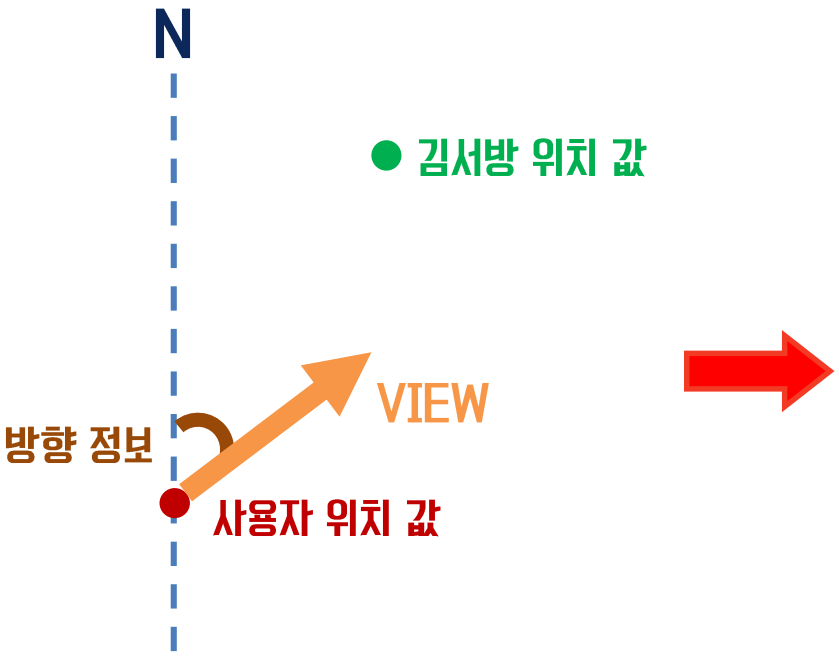

미니 맵 상에 김서방을 띄워 줌

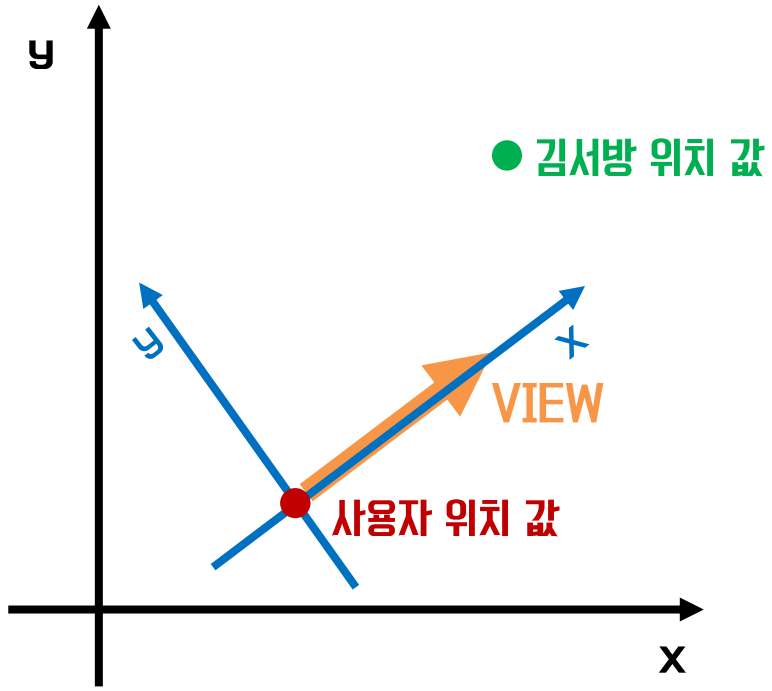
김서방의 위치를 화살표를 통해 알려 줌

# 김서방 찾기 (선택한 사람 찾기)



## 김서방 찾기 알고리즘



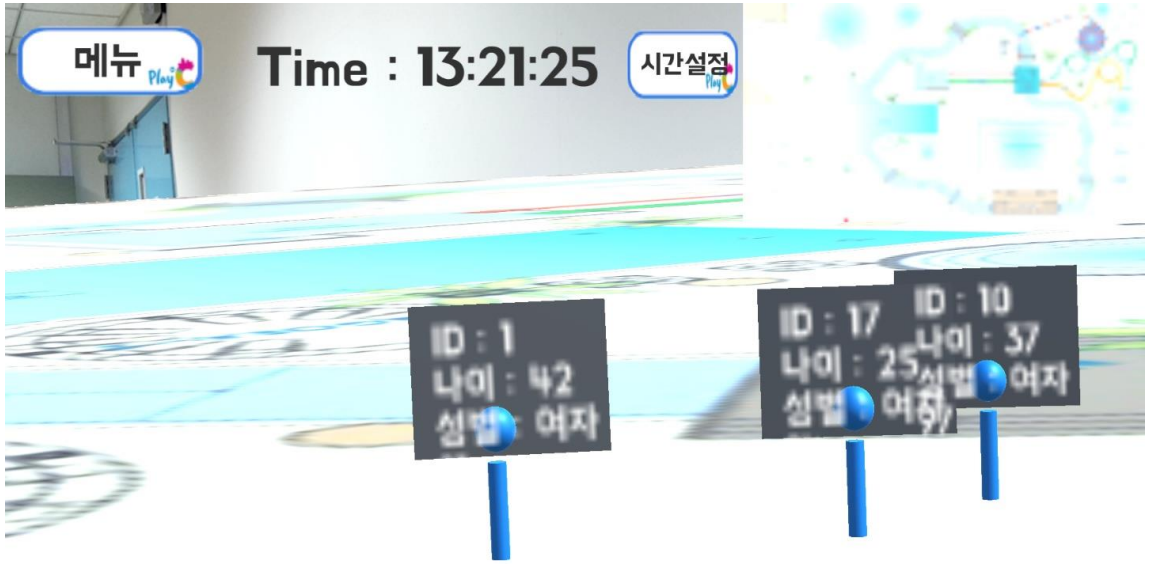

- 유니티에 내장된 Compass를 이용
- 카메라가 바라보는 방향 정보를 string 값으로 서버에 전송

- y > 0.5 -> 왼쪽 화살표
- y < - 0.5 -> 오른쪽 화살표
- - 0.5 ≤ y ≤ 0.5 -> 없음

# 정보 ON/OFF



## 정보 ON/OFF 방법




**3D text를 객체위에 띄우고 LookAtConstraint를 이용해서 빌보드를 구현**

빌보드 : 카메라가 움직이거나 회전을 해도 카메라를 향해서 바라보고있는 오브젝트

# 최종 시연

# 질의 응답

# 감사합니다.

실감미디어 2팀