공사현장 합성데이터를 위한 시뮬레이터

전과자들

임아랑 홍성민 김민성

CONTENTS

프로젝트 개요 및 개발 동기

〈표 2-3〉 주요 사고유형별 사고사망자 추이

구분	2020		2021		2022		합계
	사망자 수	비율(%)	사망자 수	비율(%)	사망자 수	비율(%)	
떨어짐	110	44.0%	147	54.2%	115	52.8%	372
깔림	35	14.0%	56	20.7%	47	21.6%	138
물체에 맞음	20	8.0%	25	9.2%	21	9.6%	66
화상	38	15.2%	0	0.0%	0	0.0%	38
끼임	5	2.0%	13	4.8%	15	6.9%	33
부딪힘	3	1.2%	8	3.0%	4	1.8%	15
교통사고	8	3.2%	4	1.5%	1	0.5%	13
감전	4	1.6%	3	1.1%	3	1.4%	10
질식	7	2.8%	1	0.4%	1	0.5%	9
기타	20	8.0%	14	5.2%	11	5.0%	45
합계	250	100.0%	271	100.0%	218	100.0%	739

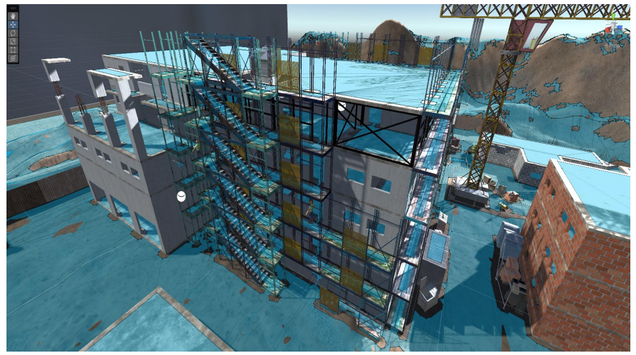
공사현장 시뮬레이션을 제작한 후
노동자 행동 데이터를 추출하여
행동분류 모델을 개발

과정

목표

공사현장 시뮬레이션제작

Unity 시뮬레이터에 공사현장 배치

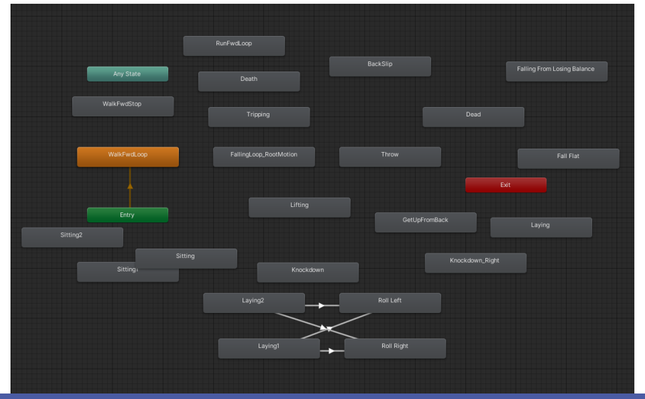


시나리오 제작

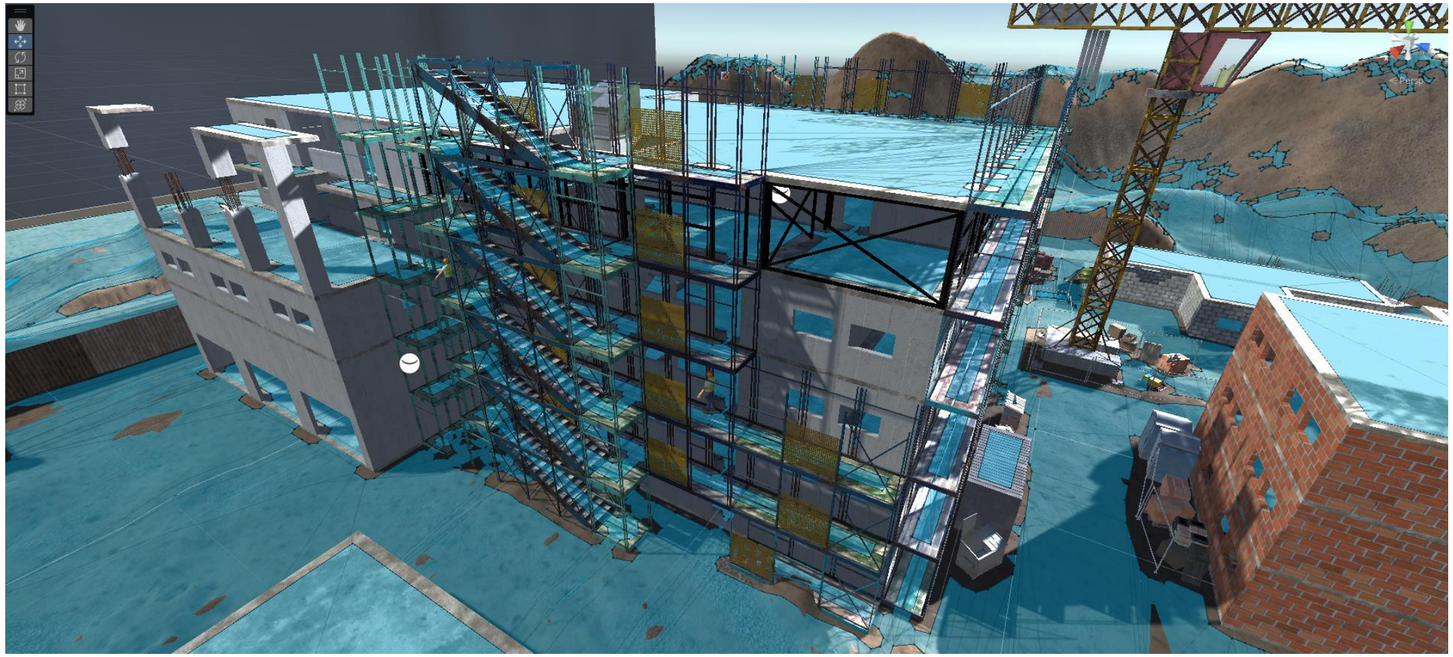
Cctv 형식의 카메라 4개를 구현해 각각의 시나리오 제작

행동 제작

여러 개의 애니메이션을 이용한 다양한 행동 제작



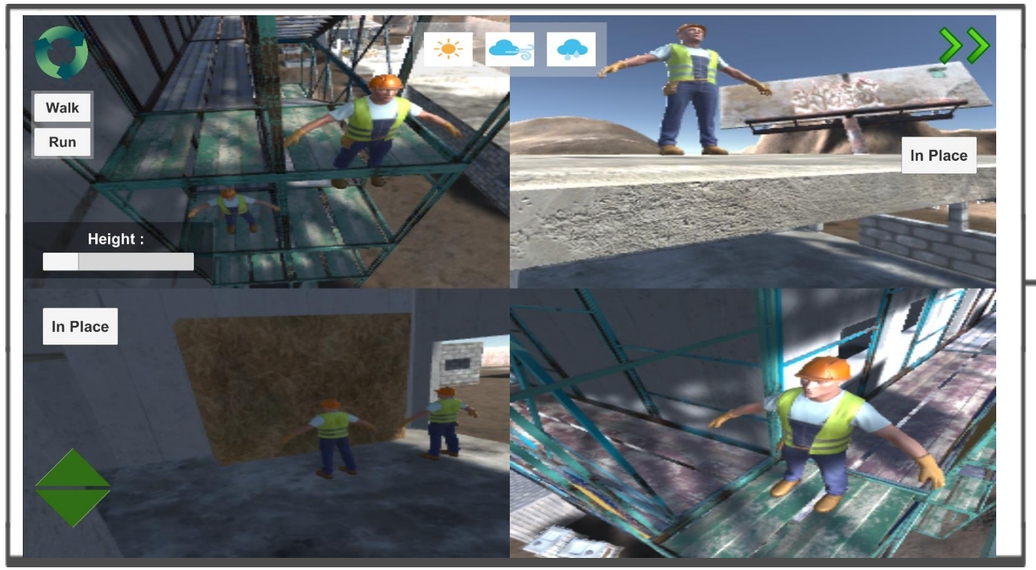
공사현장 시뮬레이션



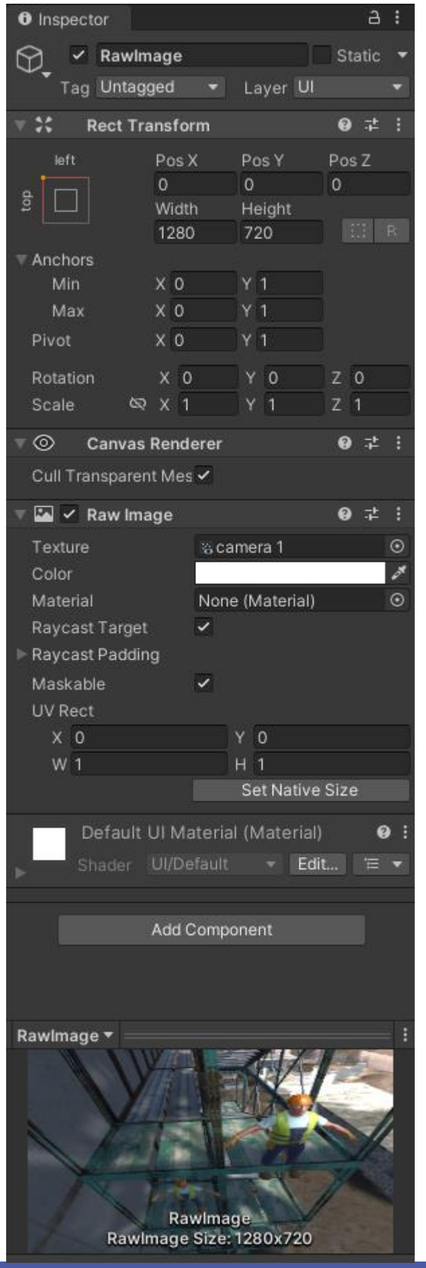
공사현장 배치

작업자 배치

Cctv 형식의 카메라 4개를 구현

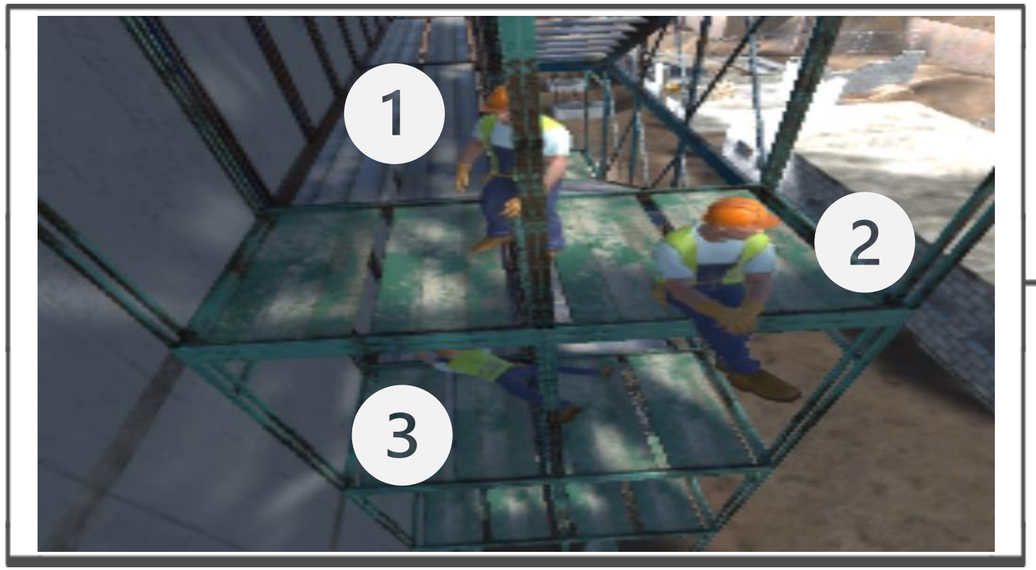


각각의 **camera**와
raw image를 연결

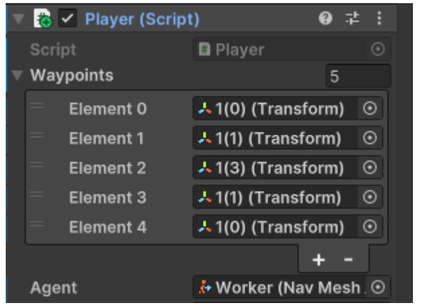


각각의 화면
확대 기능도 추가

시나리오 1



1. Worker들은 기본적으로
Bake된 맵을 **NavMeshAgent**로 이동
2. Waypoint를 배열로 받아
해당 **waypoint**를 반복적으로 이동함




```
public Transform[] waypoints;
```

```
void SetNextWaypoint()
{
    currentWaypointIndex = (currentWaypointIndex + 1) % waypoints.Length;
    agent.SetDestination(waypoints[currentWaypointIndex].position);
}
```

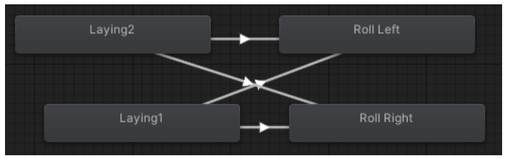
1. 기본 행동 : 걷기, 뛰기 (버튼 **UI**로 행동 선택 가능)
걷기와 뛰기 속도는 랜덤으로 설정, 속도에 따른 확률 설정 후 낙상 사고 발생

2. 기본 행동 : 앉기 (랜덤으로 **3**개의 애니메이션 실행)

```
string[] animations = { "Sitting", "Sitting1", "Sitting2" };

startAnimation = animations[Random.Range(0, animations.Length)];
animator.Play(startAnimation);
```

3. 기본 행동 : 눕기 (랜덤으로 **2**개의 애니메이션 실행)
동작의 가시성을 위해 누워서 움직이는 기능도 구현



시나리오 2

기본 행동 : 무너지는 건물에서
낙상사고 발생

건물의 높이가 일정 높이가 되면
넘어지는 애니메이션을 취함

```
agent.enabled = false;
rb.isKinematic = false;
rb.useGravity = true;
```

**Nav에서는 중력이 발생하지 않으므로
중력이 작용해 떨어지는 코드 작성**

```
if (gameObject.CompareTag("Worker4"))
{
    if (im.item4.transform.position.y < 2.8)
    {
        animator.Play("Death");
    }
}
```

시나리오 3

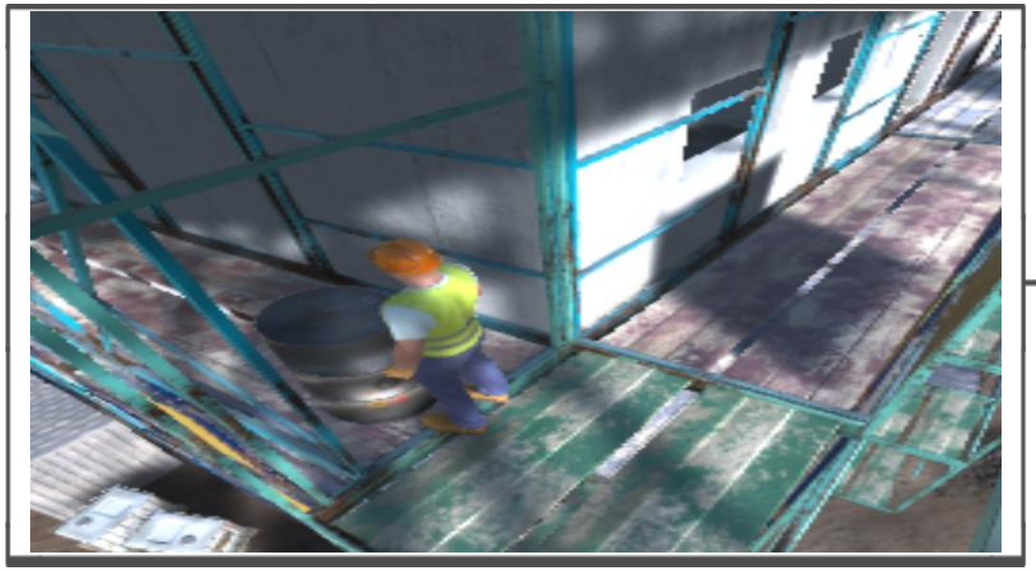
UI로 인원수 조절 가능

기본 행동 : 공사현장의 사물이 쓰러져 발생하는 낙상 사고

행동의 다양성을 위해 **worker**마다 다른 애니메이션을 사용

```
if (gameObject.CompareTag("Worker2"))
{
    animator.Play("Death");
}
else if (gameObject.CompareTag("Worker3"))
{
    animator.Play("Knockdown_Right");
}
```

시나리오 4

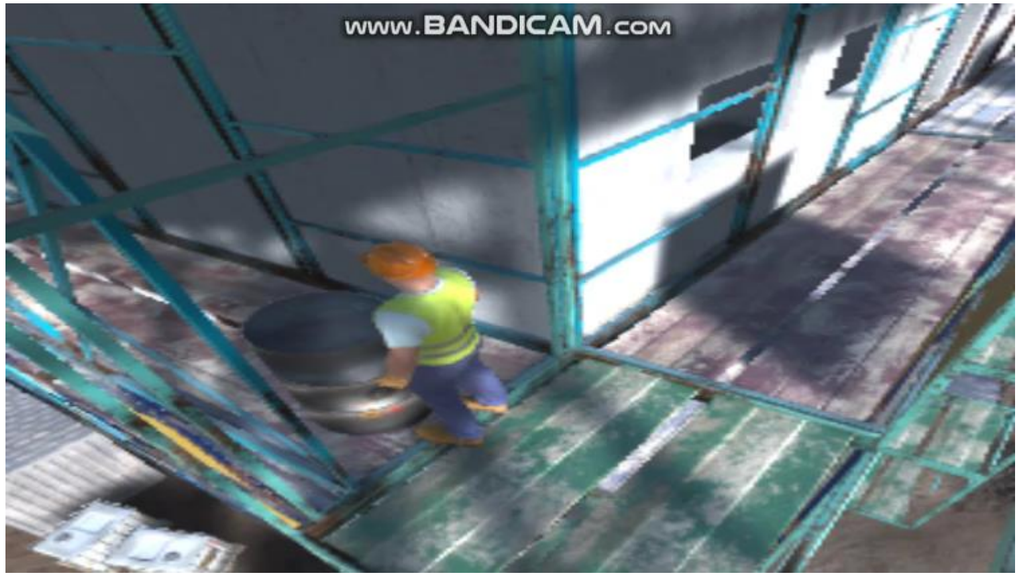


기본 행동 : 공사현장의 사물을 드는 행동

20초마다 랜덤으로 사물 생성
(사물 무게도 랜덤 생성**)**

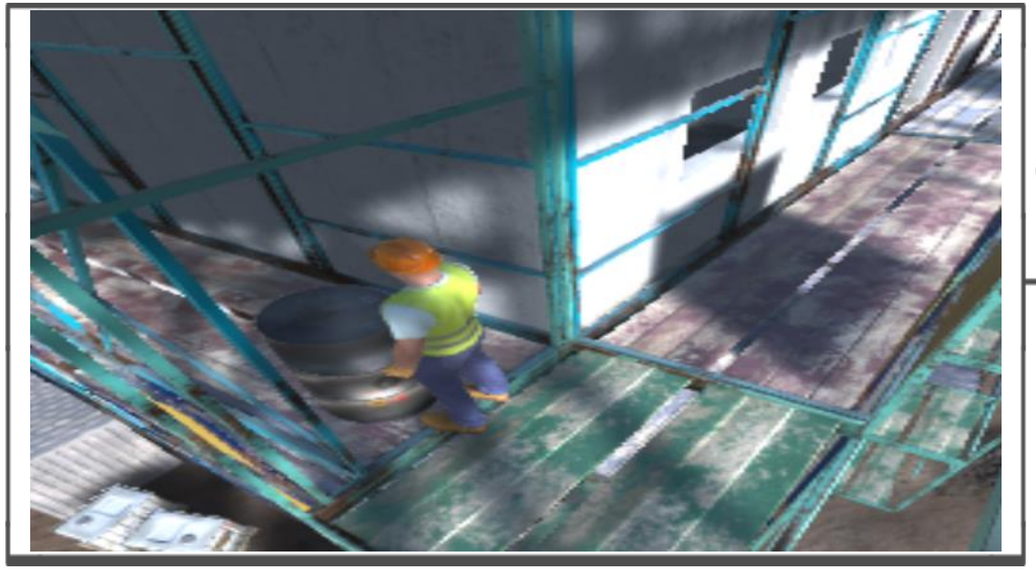
1. 일정 무게를 넘으면 넘어짐
2. 넘지 않으면 들고 지나감

①



②

시나리오 4 - 코드



20초마다 랜덤으로
사물 생성
(사물 무게도 랜덤 생성)

1. 일정 무게를 넘으면 넘어짐
2. 넘지 않으면 들고 지나감

```
private void HandleLiftHit()
{
    ignoreHit = true;
    StartCoroutine(ResetIgnoreHit());
    agent.isStopped = true;
    if (lm.weight >= 35)
    {
        isHeavy = true;
        rb.isKinematic = false;
        animator.Play("Lifting");
        StartCoroutine(PlayBackSlipAfterDelay(9.3f));
    }
    else
    {
        isHeavy = false;
        animator.Play("Throw");
        Destroy(lm.spawnedObject, 2.5f);
        StartCoroutine(ReWalk(3f));
    }
}
```

```
if (randomIndex == 0)
{
    spawnedObject.transform.position += new Vector3(-0.3f, 0f, -0.45f);
    weight = 70 * combined;

    BoxCollider collider = spawnedObject.GetComponent<BoxCollider>();
    if (collider != null)
    {
        Vector3 newSize = collider.size;
        newSize.y *= 1.6f;
        collider.size = newSize;
        Vector3 newCenter = collider.center;
        newCenter.y = 1f;
        collider.center = newCenter;
    }
}
else if(randomIndex == 1)
{
    spawnedObject.transform.position += new Vector3(0f, 0f, 0.25f);
    weight = 60 * combined;

    CapsuleCollider collider = spawnedObject.GetComponent<CapsuleCollider>();
    if (collider != null)
    {
        collider.height *= 1.6f;
        Vector3 newCenter = collider.center;
        newCenter.y = 1f;
        collider.center = newCenter;
    }
}
else if (randomIndex == 2)
{
    spawnedObject.transform.position += new Vector3(0.1f, 0f, -0.26f);
    weight = 30 * combined;

    BoxCollider collider = spawnedObject.GetComponent<BoxCollider>();
    if (collider != null)
    {
        Vector3 newSize = collider.size;
        newSize.y *= 2f;
        collider.size = newSize;
        Vector3 newCenter = collider.center;
        newCenter.y = 1f;
        collider.center = newCenter;
    }
}
```

전체 재실행

UI

날씨 조절

빨리 감기

걷기, 뛰기
조절

시나리오 **1**
키 조절

시나리오 **3**
재실행

시나리오 **3**
재실행

인원수 증감

날씨 기능

```
case WeatherState.None:
    foreach (var worker in workers)
    {
        worker.SetActive(true);
    }
    nonePrefab.SetActive(true);
    foreach (Player player in players)
    {
        player.agent.speed = 1.5f;
        player.agent.acceleration = 5;
        player.agent.stoppingDistance = 2;
        player.agent.autoBraking = true;
    }
    break;
case WeatherState.Rainy:
    foreach (var worker in workers)
    {
        worker.SetActive(false);
    }
    rainPrefab.SetActive(true);
    break;
case WeatherState.Windy:
    foreach (var worker in workers)
    {
        worker.SetActive(true);
    }
    windPrefab.SetActive(true);
    foreach (Player player in players)
    {
        player.agent.speed = 1f;
        player.agent.acceleration = 8;
        player.agent.stoppingDistance = 1;
        player.agent.autoBraking = false;
    }
    break;
```

맑음, 바람, 비 **3**개로 나눔
날씨마다 각각 속도, 가속도,
제어능력, 마찰력을 임의 조절

```
private void IncreaseFunctionality()
{
    // 이전 날씨가 Rainy였을 때 비가 그친 후 모든 기능을 1.5배로 증가시킴
    foreach (Player player in players)
    {
        player.agent.acceleration *= 1.5f;
        player.agent.stoppingDistance /= 2;
        player.agent.autoBraking = false;
    }
    fallProbabilities[WeatherState.None] *= 1.5f;
    fallProbabilities[WeatherState.Windy] *= 1.5f;
}
```

```
// 이전 날씨가 Rainy이었을 때, 비가 그친 후에 모든 기능을 1.5배로 증가
if (previousWeather == WeatherState.Rainy && currentWeather != WeatherState.Rainy)
{
    IncreaseFunctionality();
    GameManager.Instance.Restart();
}
```

```
private void InitializeFallProbabilities()
{
    // 각 날씨별 확률을 딕셔너리에 저장
    fallProbabilities.Add(WeatherState.None, Random.Range(10f, 60f));
    fallProbabilities.Add(WeatherState.Windy, Random.Range(30f, 80f));
}

참조 1개
public void CheckPlayerFall()
{
    fallProbability = fallProbabilities[currentWeather];

    foreach (Player player in players)
    {
        if (player.startAnimation == "RunFwdLoop")
        {
            fallProbability *= 1.2f;
            break;
        }
    }

    isSlipped = fallProbability > 50;
}
```

최근에 비가 왔었다면
가속도 증가, 제어능력
감소, 마찰력 감소

최근에 비가 왔었다면
사고 발생 확률 증가

비가 오고 있다면
경고 **UI**와 함께
Worker 비활성화

데이터 로깅

사고 발생 원인 분석

```
writer.WriteLine(
tm.elapsedTime >= tm.dayLengthInSeconds ? "오전입니다." :
tm.elapsedTime >= tm.dayLengthInSeconds / 3f ? "오후입니다." :
tm.elapsedTime >= (tm.dayLengthInSeconds / 3f) * 2f ? "저녁입니다." : ""
);
writer.WriteLine("사고 발생 원인 : 미끄러짐");
writer.WriteLine("날씨 : " + wm.currentWeather);
writer.WriteLine("최근 비의 유무 : " + (wm.previousWeather == WeatherManager.WeatherState.Rainy ? "O" : "X"));
writer.WriteLine("습도 : " + humidity + "%");
writer.WriteLine("바람의 유무 : " + (wm.previousWeather == WeatherManager.WeatherState.Windy ? "O" : "X"));
writer.WriteLine();
```

+

```
writer.WriteLine(
tm.elapsedTime >= tm.dayLengthInSeconds ? "오전입니다." :
tm.elapsedTime >= tm.dayLengthInSeconds / 3f ? "오후입니다." :
tm.elapsedTime >= (tm.dayLengthInSeconds / 3f) * 2f ? "저녁입니다." : ""
);
writer.WriteLine("사고 발생 원인 : 무거운 물체를 들다가");
writer.WriteLine("무게 : " + (int)lm.weight + "kg");
writer.WriteLine();
```

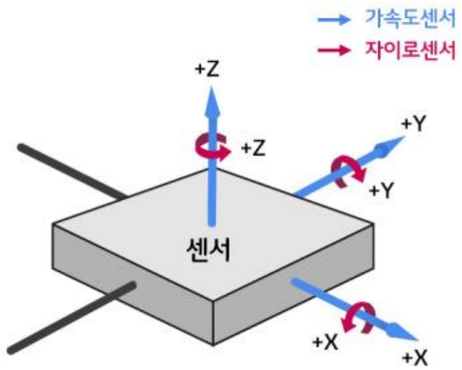
→

```
오후입니다.
사고 발생 원인 : 미끄러짐
날씨 : None
최근 비의 유무 : O
습도 : 70
바람의 유무 : X
오후입니다.
사고 발생 원인 : 무거운 물체를 들다가
무게 : 52.48882
```

목표

자이로센서, 가속도계

Unity의 각 시나리오마다
가속도 센서와
자이로센서를 가져옴

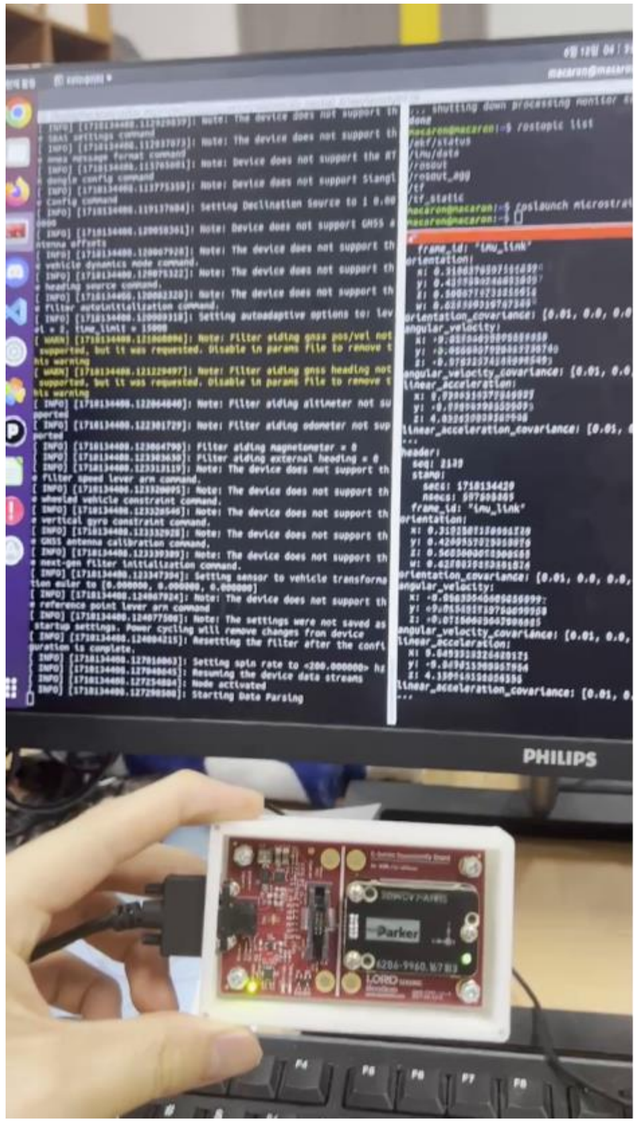


IMU

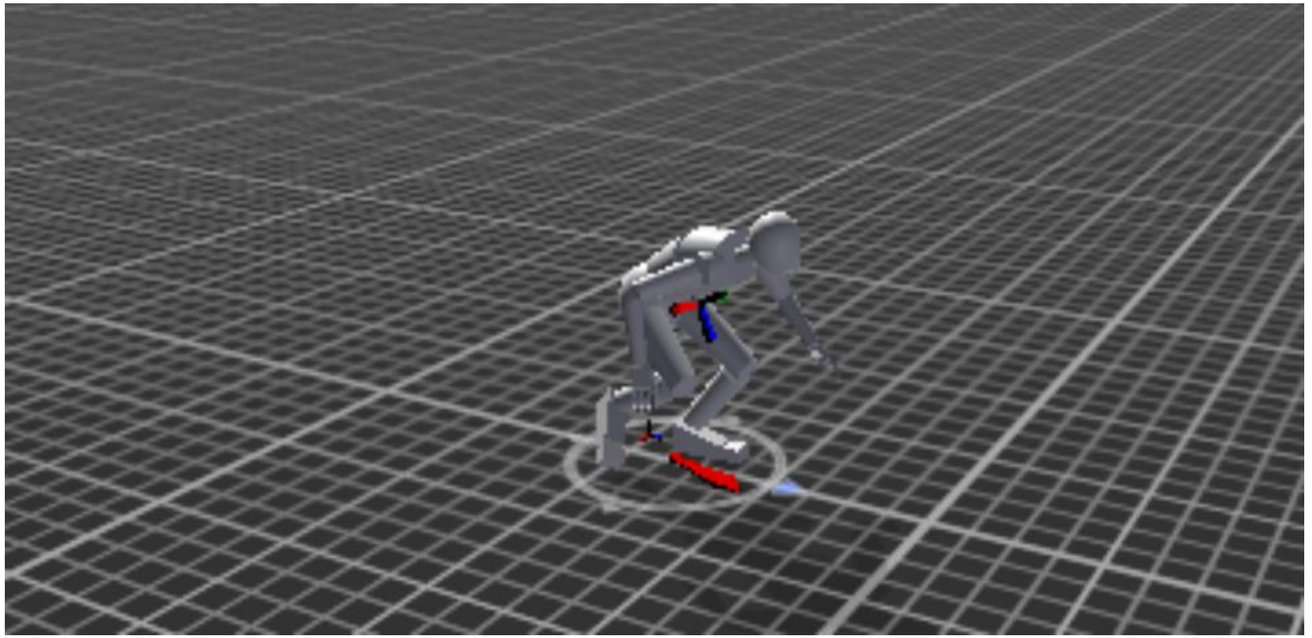
실제 가속도 센서와
자이로센서의 값을
IMU센서를 통해 가져옴

행동분류

걷기, 뛰기의 행동을
분류하는 것을 목표로 함



IMU 센서에 이동에 따른, 각 축 값의 변화 확인

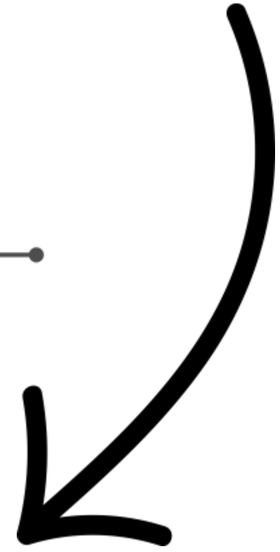


Unity상에서의 X, Y, Z축 정의

Unity의 Row Data

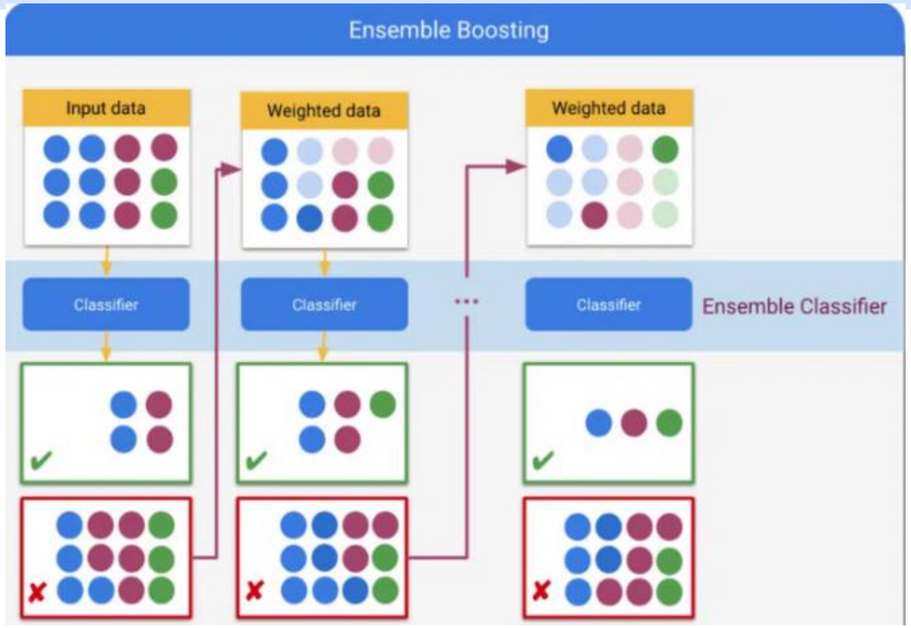
	Time	Gravity Acceleration	Angular Velocity	Linear Acceleration	Activity
0	0.02	(9.73, -1.26, 0.0)	(0.0, 0.0, 0.0)	(-2.62, 0.38, 0.0)	running
1	0.04	(9.73, -1.26, 0.0)	(0.0, 0.0, 0.0)	(-4.46, 0.64, 0.0)	running
2	0.06	(8.55, -3.45, -3.34)	(-5.62, 4.46, 6.82)	(-6.36, 1.35, 0.65)	running
3	0.08	(8.55, -3.45, -3.34)	(0.0, 0.0, 0.0)	(-5.87, 1.93, 1.34)	running
4	0.10	(8.55, -3.45, -3.34)	(0.0, 0.0, 0.0)	(-6.41, 2.34, 1.82)	running

1. 중복값 제거
2. 각 3-axis에 맞게, Gravity, Linear값들을 더함
3. IQR을 통한 이상치 제거
4. 파생 변수
   - Magnitude, Variance
   - Frequency domain Tranform
   - 절댓값 합, 평균, 표준편차 등의 값을 추가함



다양한 파생변수 추가

	Activity	Angular-Magnitude	Linear-Magnitude	Angular-Var	Linear-Var	Total-Magnitude	Angular-FFT-Strength	Angular-FFT-Phase	Linear-FFT-Strength	Linear-FFT-Phase	Angular-XYZ-Abs-Sum	Linear-XYZ-Abs-Sum	Angular-XYZ-Mean	Linear-XYZ-Mean	Angular-XYZ-Std	Linear-XYZ-Std
0	walking	0.875957	6.615701	0.033233	20.765733	7.491658	1.45	0.315753	2.59	7.892858	1.45	9.13	-0.483333	0.863333	0.182300	4.556943
1	walking	0.683667	16.896230	0.046433	57.463900	17.579897	1.06	0.373229	22.62	13.129802	1.06	25.04	-0.353333	-7.540000	0.215484	7.580495
2	running	0.890505	22.310977	0.383433	94.559033	23.201483	0.28	1.072520	30.43	16.842717	1.24	31.83	-0.093333	-10.143333	0.619220	9.724147
3	running	0.560357	3.819149	0.108400	2.196133	4.379506	0.54	0.570263	5.53	2.566788	0.58	5.53	0.180000	-1.843333	0.329242	1.481936
4	walking	0.512055	0.076811	0.107033	0.002533	0.588866	0.38	0.566657	0.05	0.087178	0.82	0.11	0.126667	-0.016667	0.327159	0.050332

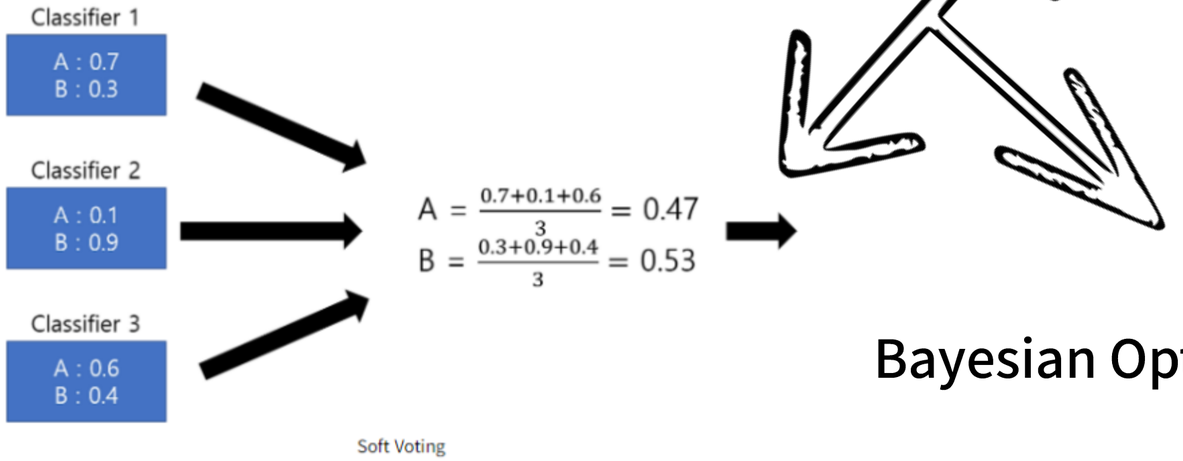


Machine Learning Algorithm

LightGBM, Gradient Boosting, Logistic Regression 모델을 사용함

[LightGBM과 Gradient Boosting]
이는 약한 분류기들을 결합하여 더 강한 분류기를 만드는 과정의 알고리즘들

[Logistic Regression]
이진 분류 문제에서 많이 사용되는 선형 모델



Bayesian Optimization을 통해 Hyperparameter Tuning

Soft Voting

여러 모델의 예측 확률을 평균하여 높은 확률을 갖는 class를 최종 예측으로 선택하게끔 하는 Ensemble

결 과

최종 결과

Enviornment	Metric	Value
Unity	Accuracy (%)	87.4
	F1 (%)	92.3
Real World	Accuracy (%)	88.1
	F1 (%)	88

아쉬운 점

파생 변수를 추가하여 모델 학습을 시도했지만 선형 가속으로 인한 노이즈가 너무 많았습니다.
따라서 다양한 필터링 방법을 시도했으나 두 클래스(Run, Walking)에 대해 약 60%의 정확도를 보이는 낮은 성능의 모델을 얻었습니다.

또한, 구부리고 걷기, 누워서 걷기, 점프하기 등 다양한 행동을 분류하려고 시도했으나,
각속도만으로는 명확한 한계가 있음을 발견했습니다.

앞으로는 Linear Accleration의 가치를 더욱 연구하여 노이즈를 정확하게 판별하고 다양한 행위를 분류하는 것이 목표입니다.

기대 효과

데이터

피실험자 없이, 다양한
데이터들을 수집할 수 있음

실시간 위험감지

실시간으로 위험 상황을
감지하여 즉각적으로 경고를
발송함으로써 작업자의 부상
및 사고를 예방할 수 있음

다양한 시나리오

시뮬레이션을 통해 실제
현장에서 발생할 수 있는
다양한 상황을 재현하고,
이를 기반으로 작업자들에게
교육 및 훈련을 제공하여 사고
예방에 기여할 수 있음

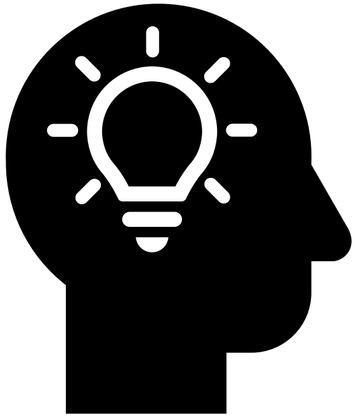
맞춤 안전 솔루션 개발

작업자의 개별 행동 패턴을
분석하여 개인별 맞춤형 안전
지침을 제공하고, 이를 통해
각 작업자의 안전을 보다
세밀하게 관리할 수 있음

역할 분담

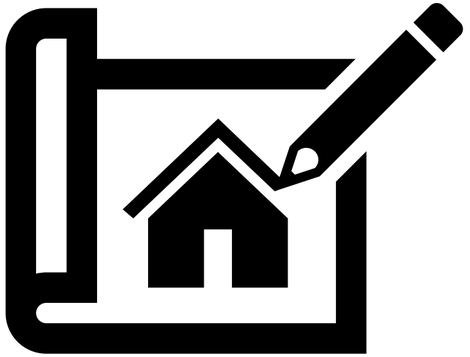
임아랑

시뮬레이션
합성데이터 추출



홍성민

머신러닝을
이용한 행동분석



김민성

유니티
시뮬레이터 개발

감사합니다.