강화학습개론 프로젝트
GIRP with Reinforcement Learning

영차영차 (2조)

120250338 이주현
320240077 이아현
320250068 한윤상

(github: https://github.com/juhynl/GIRP-with-RL.git)

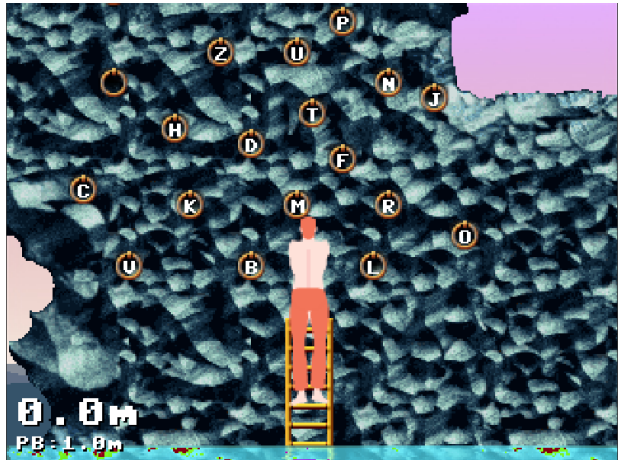
서강대학교
SOGANG UNIVERSITY

Index

- Background
- 프로젝트 문제 정의
- Observation 추출 구현 디테일
- Environment구성
- 데이터셋 구성
- State, Action, Reward 설계
- 강화학습 알고리즘
- 실험 셋업/ 실험 결과
- Discussion

Background :GIRP game (1)

- GIRP 플래시게임(링크)은 암벽을 등반하여 정상에 있는 목표 지점에 도달하는 것을 목표로 하는 게임이다.
  - Youtube에 공개된 플레이 영상 (링크)
- How to Play
  - 게임을 시작하면 플레이어가 선택 가능한 버튼이 화면상에 렌더링 됨
  - 버튼에 해당하는 알파벳 키를 누르면 해당 버튼으로 캐릭터가 손을 뻗음
  - 버튼에 손이 닿으면 캐릭터가 해당 버튼을 잡을 수 있음
  - Shift/Ctrl/Click 버튼을 누르면 버튼을 잡고 있는 팔을 가슴 쪽으로 당겨 캐릭터의 위치를 이동시킬 수 있음
  - 캐릭터 위치를 조정해가며 새로운 버튼을 선택해 암벽 정상까지 올라가는 것이 목적인 게임



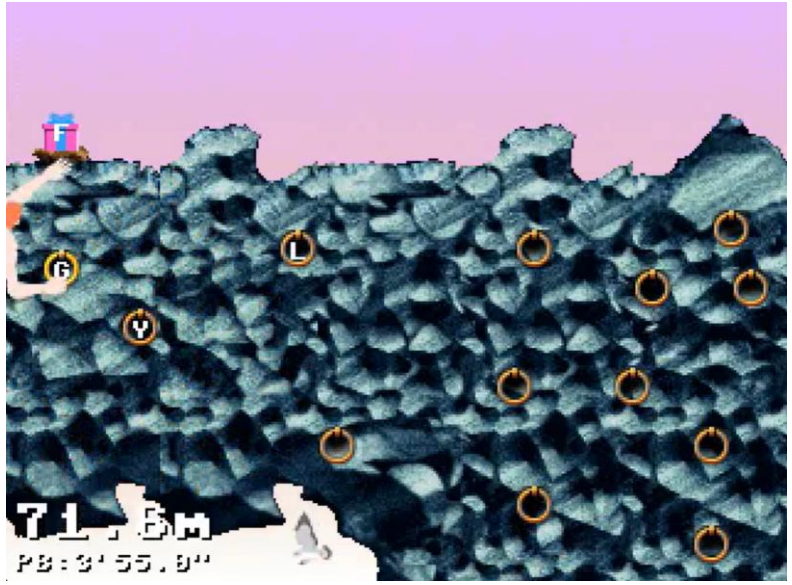
게임 시작

L을 잡고 있는 상태에서
R을 누른 모습

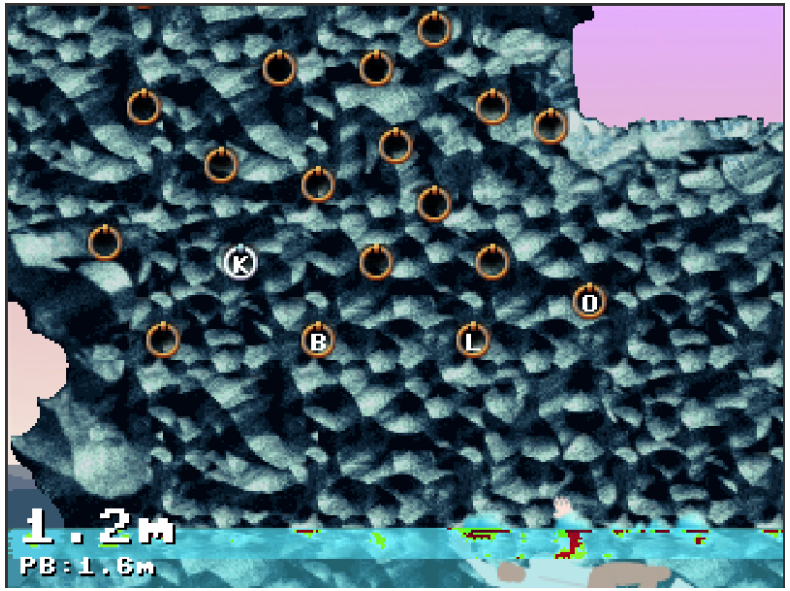
B를 잡고 있는 상태에서
Shift를 누른 모습

Background :GIRP game (2)

- 게임 종료 조건
  - 약 72m 지점에 있는 암벽 꼭대기의 “F” 버튼을 캐릭터가 잡음 (성공)
  - 서서히 올라오는 수면에 캐릭터가 빠짐 (실패)
  - 암벽 꼭대기의 “F” 버튼에 새가 먼저 도달(실패)



게임 성공

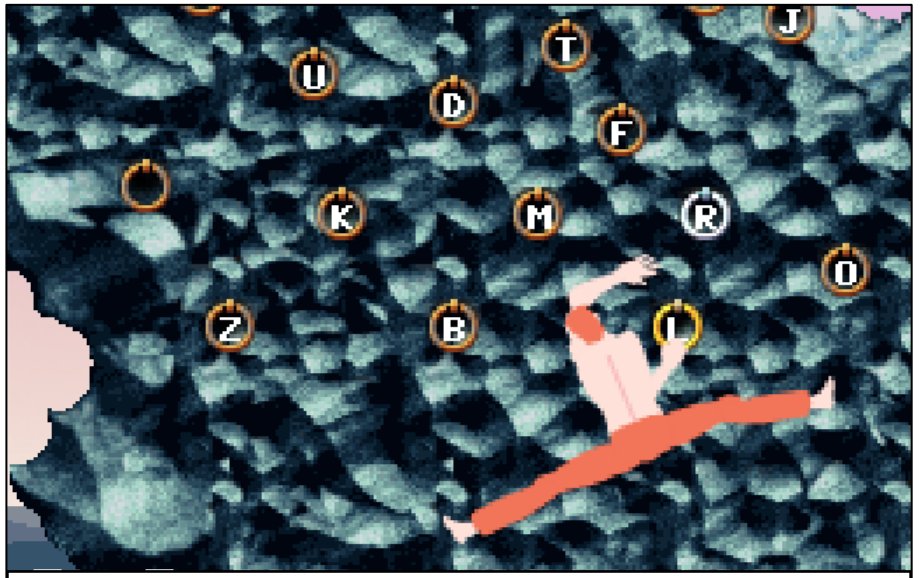


게임 실패

프로젝트 문제 정의

- GIRP은 물리엔진 기반 캐릭터를 움직여, 정해진 키보드 입력(알파벳)을 통해 손잡이를 잡고 **최대한 높이 올라가는** 의사결정 과정을 수행.
- 이를 Markov Decision Process로 모델링하여 tuple $\mathcal{M} =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$에서 누적 보상을 최대화 하는 Optimal Policy Search 문제로 정의.

State space $\mathcal{S} \subset \mathbb{R}^n$	게임 환경과 캐릭터 상태 집합
Action space $\mathcal{A}$	Discrete한 알파벳 집합 (A~Z, idle)
Transition Probability $\mathcal{P}$	현재 게임, 캐릭터 state가 다음 state로 이동할 확률
Reward Function $\mathcal{R}$	현재 State와 action에 대한 보상 및 페널티 값
Discount factor γ	Future reward에 대한 반영 비율



현재 게임 환경과 캐릭터 상태에서
어떤 알파벳을 선택해야 최대 보상을 얻을까?

프로젝트 문제 정의

- objective function J를 최대화 하는 policy π을 찾는 것이 목적.
- Policy π에서 누적 reward의 합이 최대가 되도록 설계.

$$\pi^* = \underset{\pi}{\operatorname{argmax}} J(\pi), where\ J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})\right]$$

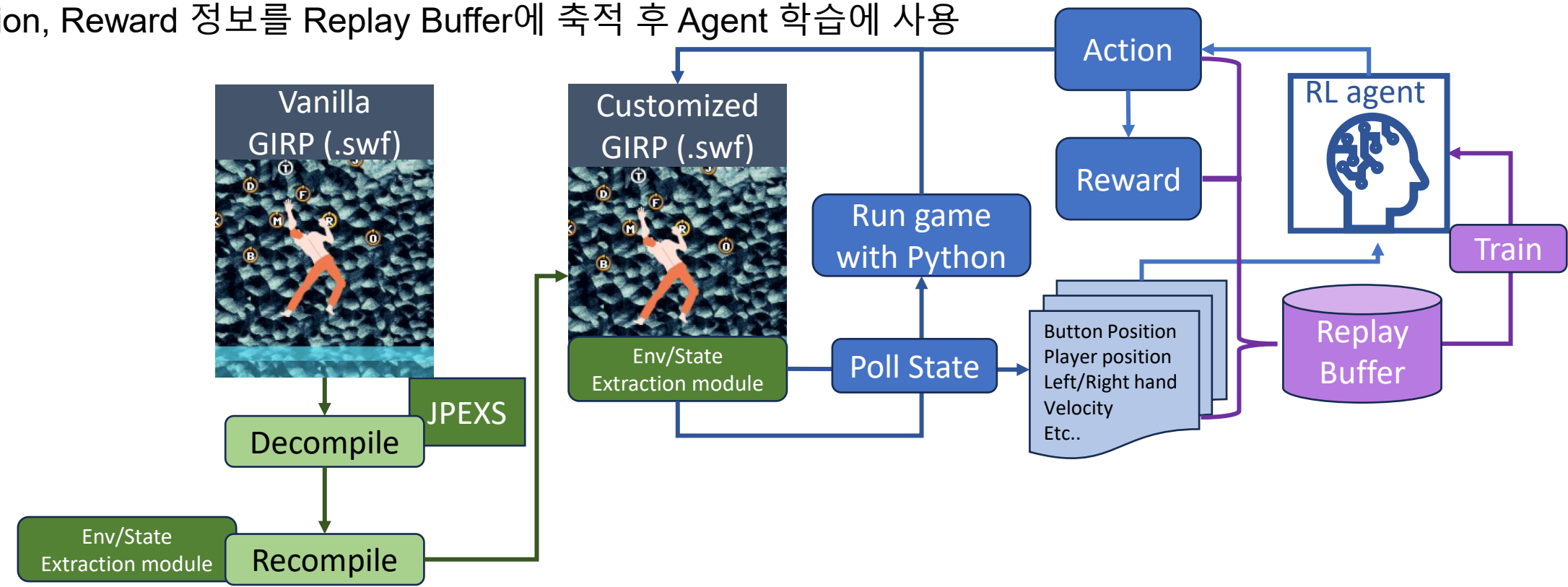
- 이 때 다음 state $\mathbf{s}_{t+1}$는 현재 state $\mathbf{s}_t$와 action a_t에 대해 인게임 물리엔진 시스템인 $\mathcal{P}$에 의해 결정된다.

$$\text{subject to} \quad \mathbf{s}_{t+1} \sim \mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t), \qquad a_t \sim \pi(a_t|\mathbf{s}_t)$$

- 따라서 GIRP는 Transition probability $\mathcal{P}$를 알 수 없는 Model-Free environment를 가지고 있음.

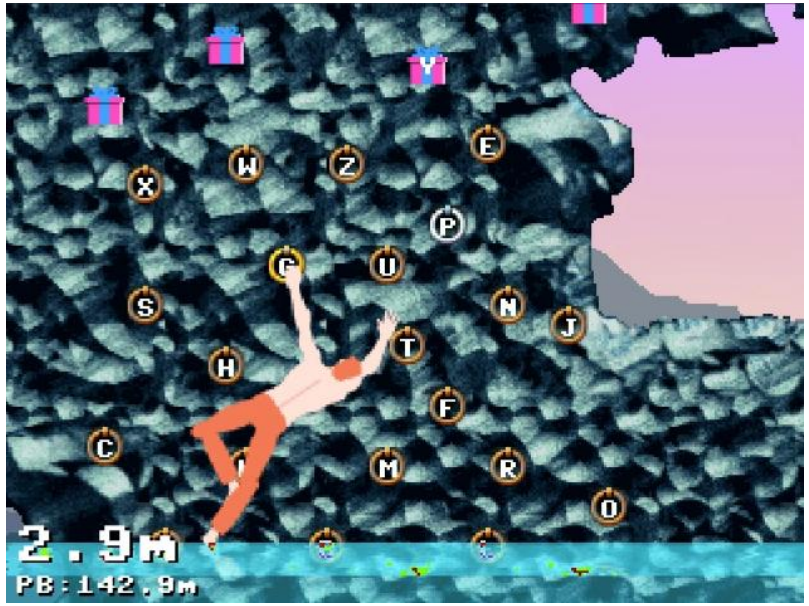
Observation 추출 구현 디테일

- GIRP는 빌드된 상태로 배포되므로, 내부에 접근하여 Environment 및 State 정보를 얻을 수 없음
- JPEXS Flash Decompiler (source)를 이용하여 순정 GIRP 게임에 Environment 와 State를 추출하는 모듈을 이식하여 Recompile 함
- Http server를 통해 Recompile한 게임을 구동하고, 일정 시간 간격으로 Observation 추출
- 추출한 State를 기반으로 강화학습 Agent가 Action을 결정하고, 이에 대한 Reward를 계산
- State, Action, Reward 정보를 Replay Buffer에 축적 후 Agent 학습에 사용



Observation 추출 구현 디테일

- GIRP_for_RL_mini.swf
  - 강화학습에 최적화된 커스텀 GIRP 환경 구현
  - 시뮬레이션 및 학습 시간 단축을 위해 물리 엔진 연산을 10배 가속
  - 목표 지점의 높이를 동적으로 조절할 수 있는 기능 추가



Custom GIRP_for_RL_mini.swf 플레이 화면

State, Action, Reward 설계: State Space

- State Space $\mathcal{S}$는 게임 환경 상태와 캐릭터 상태에 대한 feature vector의 집합으로 구성된다.
- t 시점에 대한 state $s_t \in \mathcal{S}$는 게임 환경 상태 ENV_t 캐릭터 상태 $PLAYER_t$ 로 구성된다.

$$s_t = [ENV_t, PLAYER_t]$$

t에서 게임 환경 상태
ENV_t

t에서 캐릭터 상태
$PLAYER_t$

State, Action, Reward 설계: State Space

- State Space S는 게임 환경 상태와 캐릭터 상태에 대한 feature vector의 집합으로 구성된다.
- t 시점에 대한 게임 환경 상태 ENV_t 는 아래와 같이 정의된다.

$$s_t = [ENV_t, PLAYER_t]$$

$$ENV_t = [BIRD_t, TH_t, WL_t, P_t, END_t]$$

t에서 게임종료 여부

$$END_t = \begin{cases} -1, & lose \\ 0, & playing \\ 1, & win \end{cases}$$

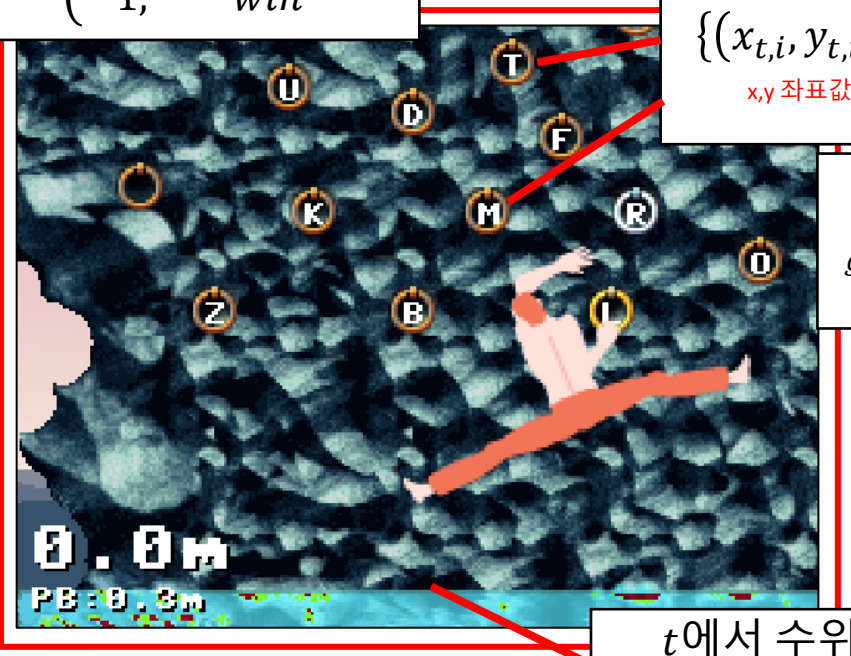
t에서 존재하는 발판들

$$\{(x_{t,i}, y_{t,i}, grip_{t,i}, L_i)\}_{i=1}^{N} = TH_t$$

x,y 좌표값 잡기 여부 발판의 알파벳

t에서 잡기 상태

$$grip_t = \begin{cases} 0, & \text{안 잡음} \\ 1, & \text{발판을 향해 뻗음} \\ 2, & \text{발판을 잡음} \end{cases}$$



t에서 수위

$$w_t = WL_t$$

t에서 새가 앉은 상태

$$grip_t = \begin{cases} 0, & \text{발판에 앉지 않음} \\ 1, & \text{발판에 앉음} \end{cases}$$

t에서 새의 정보

$$(x_{bird_t}, y_{bird_t}, landed_t) = BIRD_t$$

x,y 좌표값 앉은 상태

t에서 플레이 타임

$$p_t = P_t$$

State, Action, Reward 설계: State Space

- State Space $\mathcal{S}$는 게임 환경 상태와 캐릭터 상태에 대한 feature vector의 집합으로 구성된다.
- t 시점에 대한 캐릭터 상태 $PLAYER_t$ 는 아래와 같이 정의된다.

$$s_t = [ENV_t, PLAYER_t]$$

$$PLAYER_t = \left[CHEST_t, HEAD_t, LH_t, RH_t, A_t, \phi_t^{(L)}, \phi_t^{(R)}\right]$$

t에서 오른손의 정보

$\left(x_t^{(R)}, y_t^{(R)}, vx_t^{(R)}, vy_t^{(R)}, grip_t^{(R)}\right) = RH_t$

x,y 좌표값 x,y 축에서 속도

t에서 머리의 정보

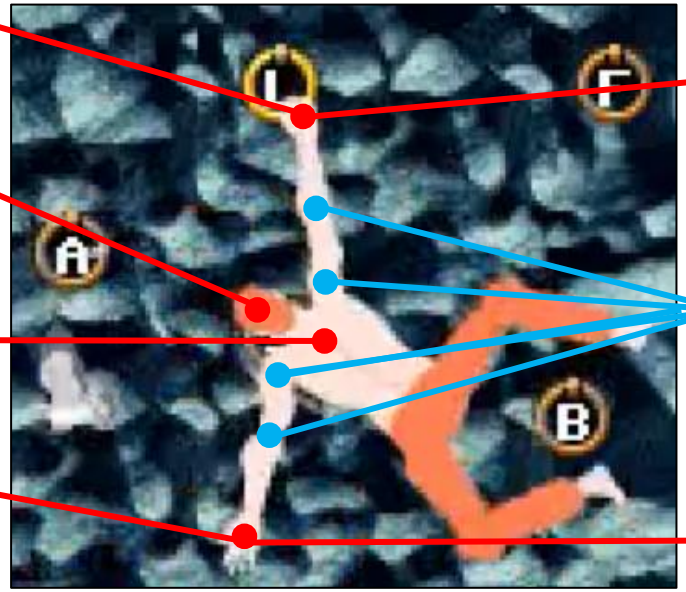
$\left(x_{HEAD_t}, y_{HEAD_t}, vx_{HEAD_t}, vy_{HEAD_t}\right) = HEAD_t$

t에서 가슴의 정보

$\left(x_{CHEST_t}, y_{CHEST_t}, vx_{CHEST_t}, vy_{CHEST_t}\right) = CHEST_t$

t에서 왼손의 정보

$\left(x_t^{(L)}, y_t^{(L)}, vx_t^{(L)}, vy_t^{(L)}, grip_t^{(L)}\right) = LH_t$



t에서 오른손과 가장 가까운 k개의 발판

$\left\{\left(dx_{t,i}^{(R)}, dy_{t,i}^{(R)}, \mathbf{o}_{t,i}\right)\right\}_{i=1}^{k} \in \phi_t^{(R)}$

손과 발판의 거리 발판 문자의 one hot vector

t에서 팔꿈치, 어깨 각도 값

$\left(a_{elbow_t}^{(R)}, a_{shoulder_t}^{(R)}, a_{elbow_t}^{(L)}, a_{shoulder_t}^{(L)}\right) = A_t$

t에서 왼손과 가장 가까운 k개의 발판

$\left\{\left(dx_{t,i}^{(L)}, dy_{t,i}^{(L)}, \mathbf{o}_{t,i}\right)\right\}_{i=1}^{k} \in \phi_t^{(L)}$

State, Action, Reward 설계: Action Space

- GIRP는 물리 엔진 기반의 게임으로, 손을 잡은 위치와 뻗는 각도, 반동을 받는 타이밍에 따라 잡을 수 있는 발판이 달라진다.
- 잡을 수 있는 발판의 후보는 위치, 각도, 타이밍 등 continuous한 환경에 영향을 받는다.
- 그러나 RL에서 이러한 continuous dynamics를 고려해 action space를 설계하는 것은 매우 복잡하다.
- 따라서 단순화를 위해 발판에 적힌 문자를 기준으로 discrete한 action space를 정의한다.
- Action space $\mathcal{A}$는 해당 알파벳이 적힌 발판으로 뻗는 동작과 아무것도 누르지 않고 대기하는 idle 동작으로 구성된다.

$$\mathcal{A} = \{0, 1, 2, \dots, 26\}$$

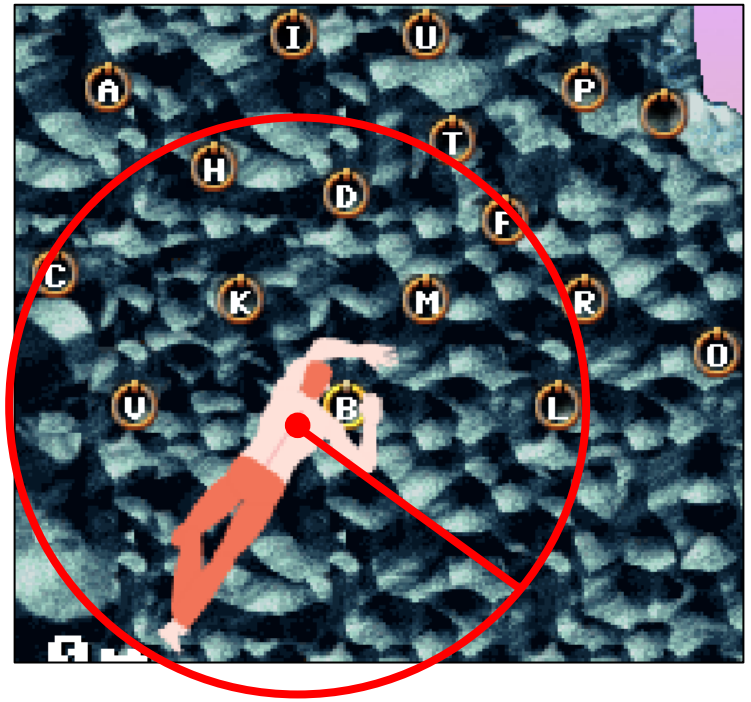
$$a_t = \begin{cases} Press\ (A + a_t), & 0 \le a_t \le 25 \\ Idle, & a_t = 26 \end{cases}$$

State, Action, Reward 설계: Action Space

- Action Space $\mathcal{A}$는 discrete하게 동작하지만, 팔의 위치나 각도에 따라 물리적으로 잡을 수 없는 발판도 잠재적으로 포함될 수 있다.
- 따라서 우리는 실제로 잡을 수 있는 발판만을 허용하는 action mask m_t를 도입했다.
- Action mask m_t는 현재 state s_t에서 캐릭터가 발판에 도달하지 못하는 action을 0으로 만드는 vector로, 아래와 같이 정의된다.

$$m_t = \{0, 1\}^{|\mathcal{A}|}$$

- 물리적으로 잡을 수 없는 발판의 기준은 캐릭터의 가슴 높이를 기준으로 반경 3.5 픽셀로 설정했다.



$m_t =$

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	idle
0	1	1	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

State, Action, Reward 설계: Reward function

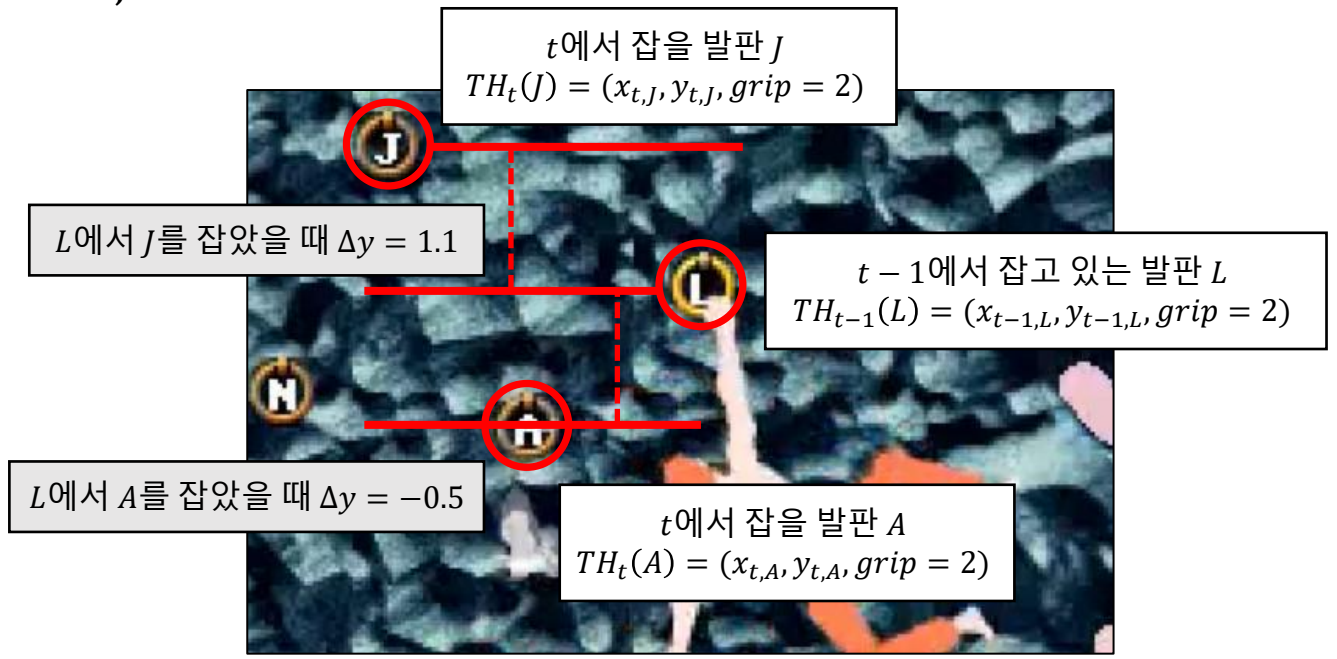
- t 시점에서의 reward function r_t는 아래와 같이 세 파트로 구성되어 있다.

$$r(t) = r_{climb}(t) + r_{time}(t) + r_{winResult}(t)$$

- 목표하는 발판을 잡았을 때의 보상 $r_{climb}(t) = \begin{cases} +0.5, & \Delta y(t) > 0.5 \\ -1.0, & \Delta y(t) < 0.5 \\ -0.01, & else \end{cases}$
  - $\Delta y(t) = (y_{t,i_t} - y_{t-1,j_{t-1}})$

    $where\ i_t = \{i | grip_{t,i} = 2\},\ \ j_{t-1} = \{j | grip_{t-1,j} = 2\}$

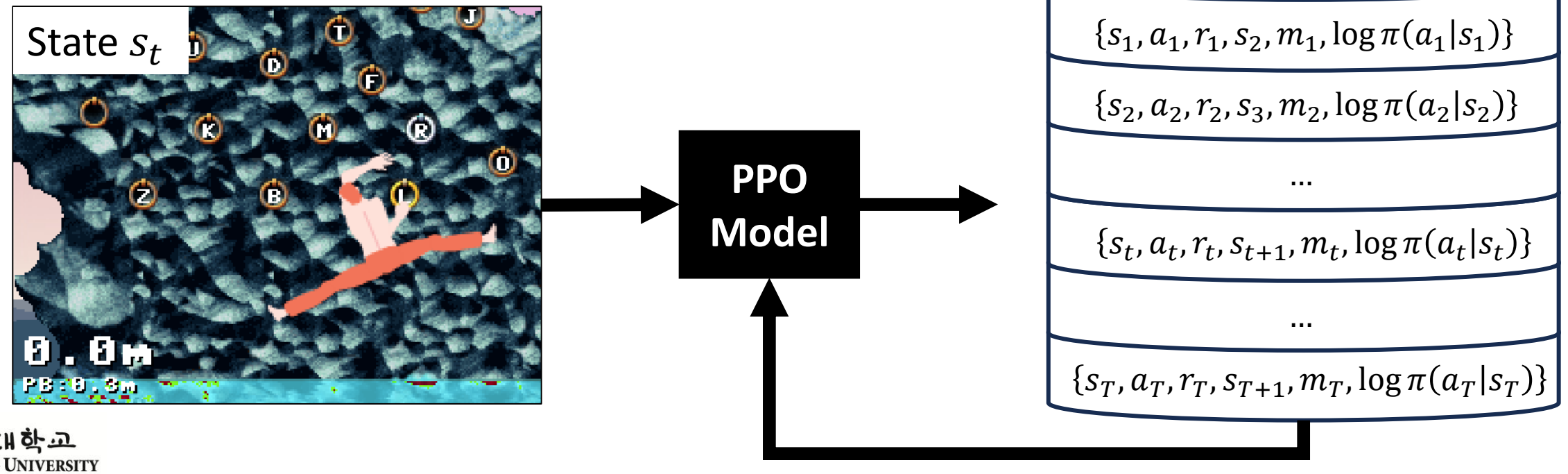
- Time penalty $r_{time}(t) = -0.01$
- 게임 종료 시 보상 $r_{winResult}(t) = \begin{cases} +10.0, & if\ win \\ -1.0, & if\ lose \end{cases}$



데이터셋 구성

- 데이터셋 $\mathcal{D}$는 buffer에 저장된 인게임 transition들의 집합으로 구성된다.
- 0.5초 간격으로 인게임 State s_t를 추출하고, PPO Model을 통해 추론된 action a_t, reward r_t, action mask m_t, 다음 state s_{t+1} , log probability $\log \pi(a_t|s_t)$ 를 buffer에 저장.
- Buffer가 가득 찼을 때 buffer 안의 데이터들을 하나의 batch로 사용하여 model update 수행
- 따라서 데이터셋 $\mathcal{D}$는 아래와 같이 정의된다.

$$\mathcal{D} = \{(s_t, a_t, r_t, s_{t+1}, m_t, \log \pi(a_t|s_t))\}_{t=1}^{T}$$



강화학습 알고리즘: PPO + Action Masking

- PPO
  - GIRP는 복잡한 물리적 상호작용과 고도의 전략이 요구되는 게임이다.
  - 에이전트는 다양한 요소의 위치와 속도를 고려하여 다음 발판을 즉시 잡을지, 혹은 기다렸다가 잡을지를 결정해야 한다.
  - PPO는 Clipping 메커니즘을 통해 정책 업데이트의 폭을 제한함으로써 학습 붕괴를 방지하고 안정적인 학습을 가능하게 한다.
  - 따라서 복잡한 환경을 가지면서 Model-Free 환경인 GIRP 에는 안정성이 뛰어난 PPO를 사용하는 것이 유리하다.

- Action Masking
  - 학습 효율을 높이기 위해 실행 불가능하거나 비효율적인 action을 비활성화한다.
  - 유효한 action을 식별하는 mask를 생성하고, 그 외 action의 logits 값을 매우 작은 수(-1e9 내외)로 설정하여 해당 action이 선택되는 것을 방지한다.

강화학습 알고리즘: Curriculum Learning

- GIRP 강화학습의 한계점
  - 시작 지점과 최종 목표(정상) 간의 거리가 멀어, 학습 초기 단계에서 최종 목표에 대한 보상을 획득할 확률이 매우 희박하다.
  - 이로 인해 에이전트가 Local Optima에 빠지거나, 정상 도달을 위한 최적의 행동 정책을 학습하지 못하는 문제가 발생한다.
- 해결책: Curriculum Learning
  - 쉬운 난이도의 목표부터 시작하여 에이전트의 학습 수준에 맞춰 점진적으로 과제의 난이도를 높이는 학습 기법을 적용한다.

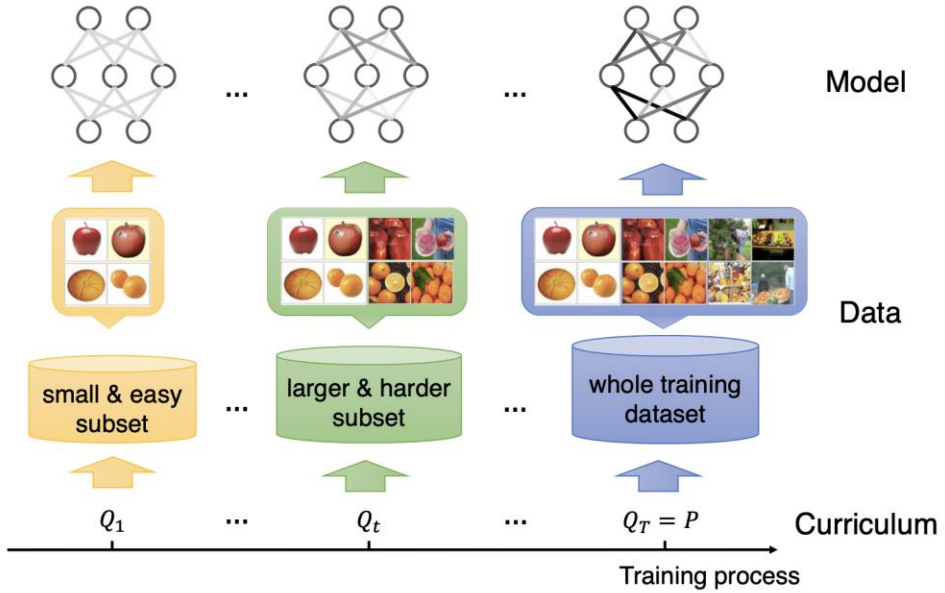


Illustration of the Curriculum Learning (CL) concept [1]

[1] Wang, Xin, Yudong Chen, and Wenwu Zhu. "A survey on curriculum learning." *IEEE transactions on pattern analysis and machine intelligence* (2021)

강화학습 알고리즘: Curriculum Learning

- Curriculum 구성
  - GIRP는 고도가 높아질수록 지형이 복잡해지고 등반 난이도가 높아지는 특성을 가진다.
  - 따라서 초기에는 목표 지점의 고도를 낮게 설정하고, 이를 점진적으로 높이는 방식으로 난이도를 조절한다.
- Curriculum Learning 과정
  - 초기 설정: 목표 지점을 시작 위치와 가깝게 설정하여, 에이전트가 비교적 쉽게 승리 보상을 획득하고 등반의 기초적인 원리를 학습하도록 유도한다.
  - 난이도 조절: 에이전트의 최근 승률이 설정된 임계값을 초과하면, 해당 난이도를 극복한 것으로 간주하여 목표 고도를 상향 조정한다.
- 기대 효과
  - 낮은 고도에서 습득한 등반 기술을 바탕으로 고난도 지형에 점진적으로 적응하게 하여 학습의 안정성을 확보한다.

강화학습 알고리즘: Hyper Parameters

- 신경망 구조와 최적화에 관련된 하이퍼파라미터
  - **ACT_DIM = 27**: 에이전트가 선택할 수 있는 행동의 차원 수
  - **LR_ACTOR = 0.0001**: Actor 네트워크의 Learning Rate
  - **LR_CRITIC = 0.001**: Critic 네트워크의 Learning Rate
- 정책 업데이트와 보상 계산 방식을 제어하는 하이퍼파라미터
  - **UPDATE_TIMESTEP = 500**: 업데이트를 위해 수집해야 하는 데이터의 타임스텝 수
  - **GAMMA = 0.99**: Discount Factor
  - **GAE_LAMBDA = 0.95**: GAE(Generalized Advantage Estimation)를 계산할 때 사용되는 가중치
  - **K_EPOCHS = 4**: 네트워크 반복 학습 횟수
  - **EPS_CLIP = 0.2**: Clipping 범위 설정
  - **ENTROPY_COEF = 0.05**: 엔트로피 항의 계수

강화학습 알고리즘: Hyper Parameters

- 커리큘럼 학습 관련 하이퍼파라미터
  - **WIN_RESULT_BUFFER_SIZE = 20**: 승률 계산에 사용되는 최근 에피소드의 개수
  - **WIN_RATE_THRESHOLD = 0.7**: 난이도를 높이기 위한 기준 승률
  - **INITIAL_GOAL_HEIGHT = 1.5**: 학습 초기에 에이전트가 도달해야 하는 목표 높이
  - **GOAL_HEIGHT_INCREMENT = 1.0**: 다음 단계로 넘어갈 때 증가시킬 목표 높이의 크기
- 전체 학습 관련 하이퍼파라미터
  - **NUM_EPISODE = 10000**: 총 실행할 에피소드의 횟수

실험 셋업

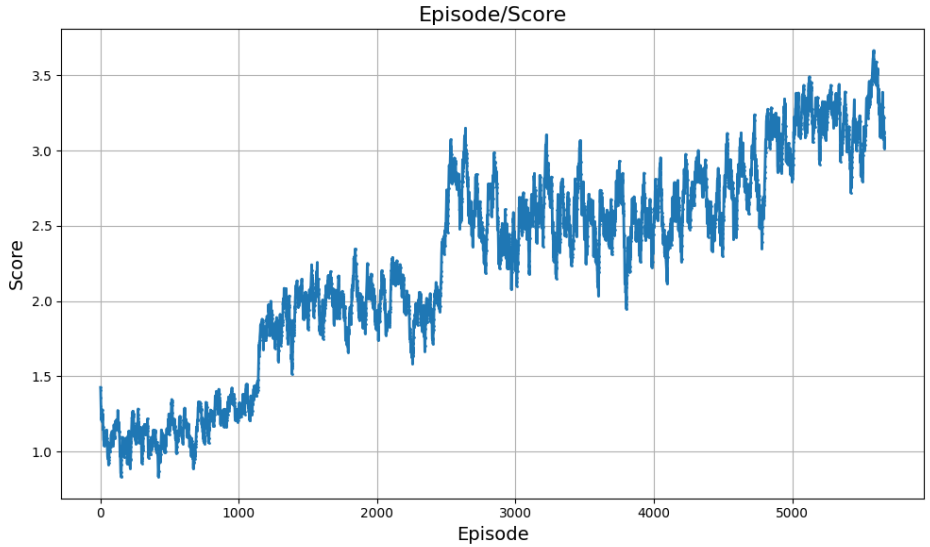
- 실험환경

OS	Ubuntu 20.04.6 LTS
CPU	12th Gen Intel(R) Core(TM) i7-12700
GPU	NVIDIA RTX 3060-12GB 28 Ampere SMs 12GB GDDR6 DRAM
python	3.10
pytorch	2.3.0+cu121

- Evaluation Metric
  - player가 에피소드에서 달성한 최대 높이 (Score)
  - player가 최근 20개의 에피소드에서 달성한 평균 승률 (winRate)

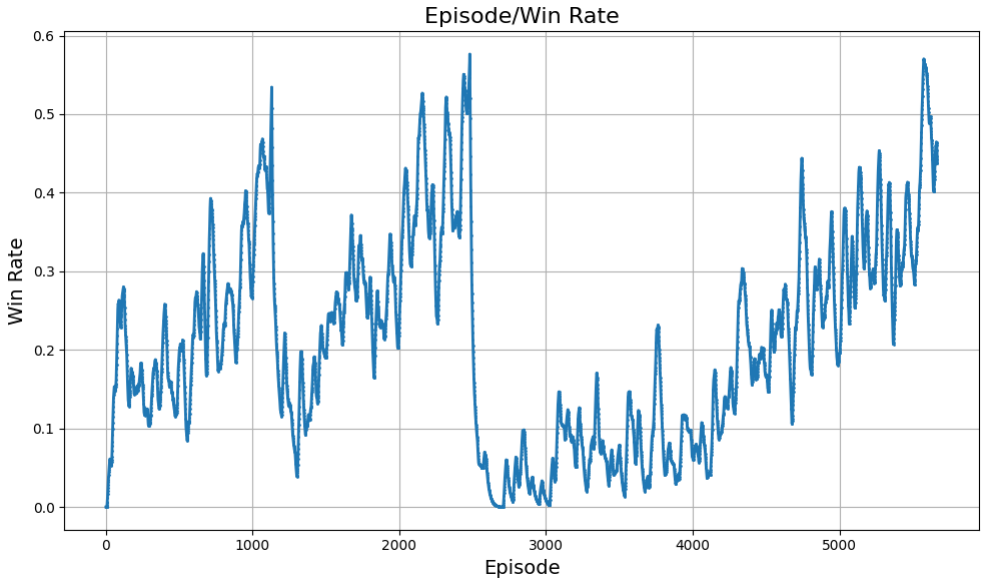
실험 결과

- Episode 진행에 따른 Score



  - Episode가 진행됨에 따라 Agent가 게임을 학습하여 더 높은 GIRP 게임 Score를 달성
  - 일정 승률을 달성하면 목표 지점의 위치를 높이면서 Agent가 더 높은 곳에 도달하는 것을 확인할 수 있다

- Episode 진행에 따른 Win Rate

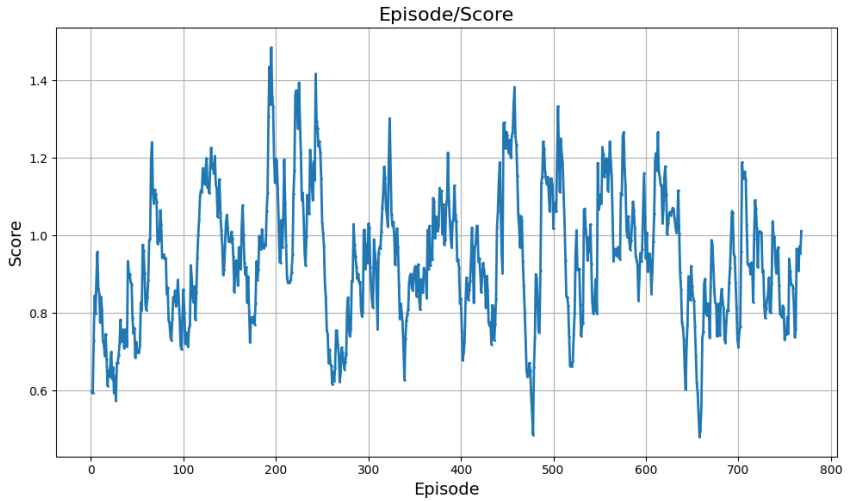


  - Win Rate는 최근 20개의 에피소드에서의 평균 승률을 계산한 값
  - Curriculum Learning 기법으로 인해 학습 중간에 승률이 급격하게 떨어지는 구간이 발생한다
  - 목표치 수정이 이루어진 후에는 안정적으로 승률이 서서히 높아지는 것을 확인할 수 있다

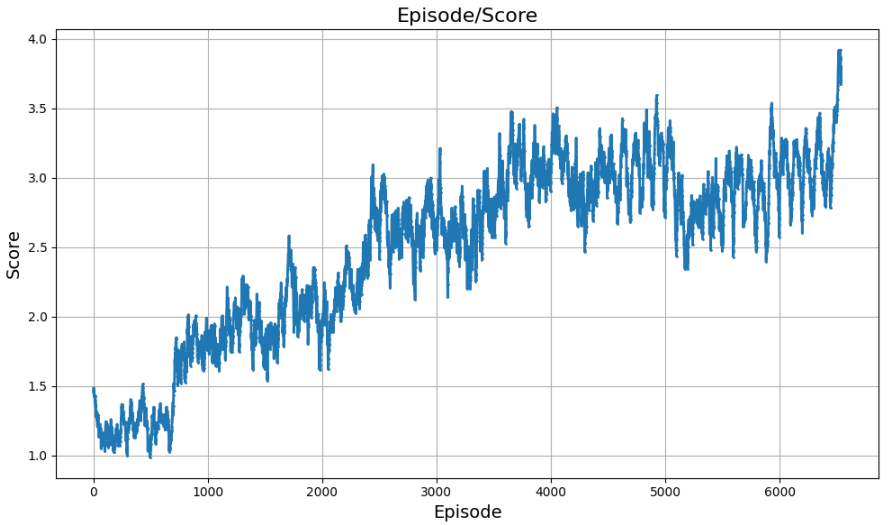
실험 결과

- Curriculum Learning 기법 적용 여부에 따른 결과

[Curriculum Learning 미적용]



[Curriculum Learning 적용]

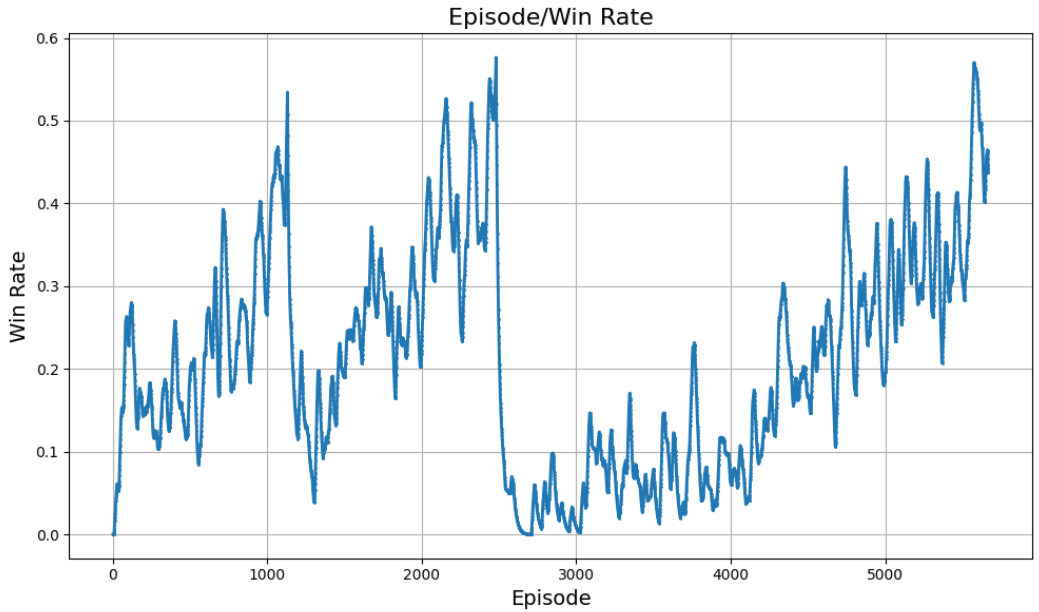


- Curriculum learning을 적용하지 않았을 때 episode마다 도달하는 최대 높이의 variance가 매우 높아 학습이 불안정한 문제가 있었음. 즉, 하나의 에피소드에서 우연히 최대 높이를 달성했다 하더라도 곧이어 떨어져 죽는 과정만 관찰되었기 때문에 agent는 최대 높이를 성공적으로 달성했다는 사실을 학습할 수 없음.
- Curriculum Learning 적용을 통해 훨씬 안정적인 학습이 이루어지는 것을 확인할 수 있다.

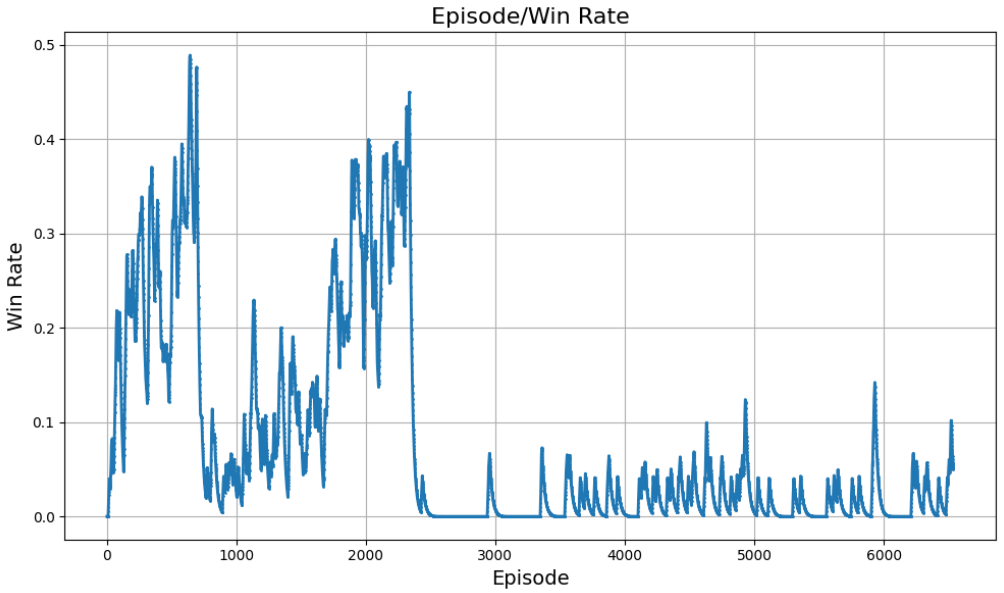
실험 결과

- Curriculum Learning 기법 적용시의 목표 높이 상향 정도에 차이를 주고 실험 진행

[Goal Height delta = 1.0]



[Goal Height delta = 1.5]



- Curriculum learning에서 목표 높이가 1.0m일 때는 win rate값이 점진적으로 증가하고 있으며, 1.5m일 때는 2000 episode 이후 win rate이 회복하지 못하는 모습을 보이고 있다. 목표 난이도가 높아지며 agent가 1.5m를 달성한 성공 경험을 충분히 반영하지 못하여 정책이 안정적으로 유지되지 못한다는 사실을 알 수 있음.
- Curriculum Learning 기법을 적용할 때 목표치의 상향 정도를 적절히 조절해야 한다는 것을 확인할 수 있음

Discussion

- 실험을 통해 Curriculum learning이 본 게임 환경에서 효과적임을 확인할 수 있었다.
- GIRP는 일반 게임과 달리, 한 번의 실패로 곧바로 원점보다 낮은 곳으로 추락하여 에피소드가 종료되도록 구성되어 있다.
- 이러한 특성으로 인해 PPO가 초기에 학습하는 불안정한 정책으로는 우연히 성공 경험을 달성하기 조차 매우 어렵다는 한계가 있다.
- 따라서 Curriculum learning을 통해 달성 가능한 목표 높이를 낮은 수준으로부터 점진적으로 증가시켜 에이전트가 초기 단계에서부터 안정적으로 성공 경험을 탐색하고 일관적인 policy를 학습할 수 있다는 것을 알 수 있었다.
- 게임에서 고도가 높아질 수록 복잡해지는 게임 환경으로 인해, 학습 안정성을 유지하기 위해서는 win rate의 buffer크기와 에피소드 수집 주기를 적절하게 조절하는 방식이 필요하다.