RL Particle Control: 강화학습 기반 2D 파티클 분수 제어

20221096 이현서

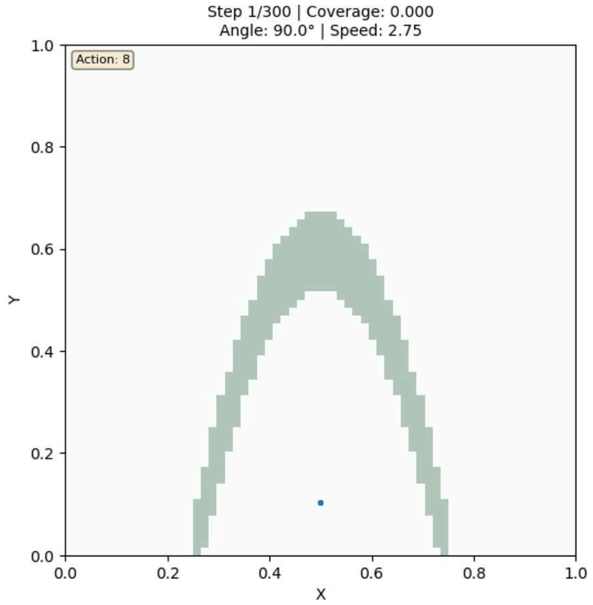
프로젝트 개요

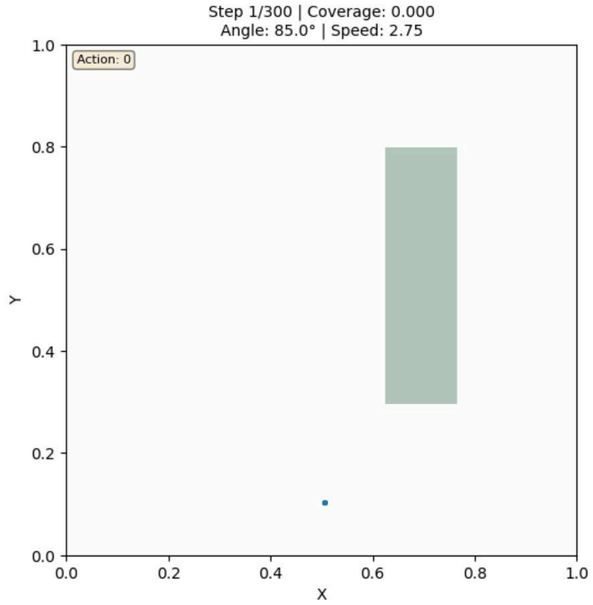
본 프로젝트는 Deep Q-Network(DQN) 을 사용해 2D 공간에서 파티클 분수의 발사 각도와 속도를 제어하는 에이전트를 학습한다.

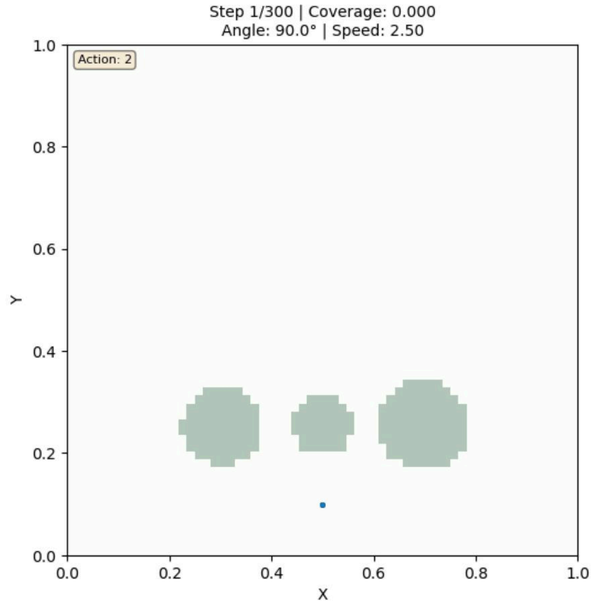
목표: 주어진 타겟 영역을 최대한 많이 커버하도록 제어 정책을 학습함

응용 예시

- 소방 호스 분사 제어
- 농업/정원용 스프링클러
- 페인트 스프레이 / 분수 연출 등







물리기반 시나리오 선정

Circle : 원형 타겟을 최대한 덮는 기본 시나리오

Wall : 수직 벽면의 특정 영역 커버 (소방 호스, 페인트 분사)

Ground : 지면 위 여러 타겟 영역 커버 (스프링클러)

Arch : 포물선 형태 분수 아치 영역 커버 (장식용 분수)

환경 및 시뮬레이션 설정

시뮬레이션 기본 설정

- 에피소드 길이 max_steps = 200
- 시간 간격 dt = 0.02 s
- 중력 가속도 gravity = -3.0 (y 축 아래 방향)
- 마찰 계수 friction = 0.99

발사기 설정

- 위치: (0.5, 0.05) (화면 기준 정규화 좌표)
- 발사 각도 범위: 30° ~ 150°
- 발사 속도 범위: 1.5 ~ 4.0

파티클·그리드 설정

- 스텝당 파티클 수: 50
- state용 그리드: 8 × 8
- 커버리지 계산용 그리드: 64 × 64

물리 모델과 전처리

속도 업데이트

$$v_y(t+1) = v_y(t) + gravity \times dt$$

위치 업데이트

$$pos(t+1) = pos(t) + vel(t+1) \times dt$$

마찰 적용

$$vel(t+1) = vel(t+1) \times friction$$

파티클 관리

- 화면 밖으로 나간 파티클은 제거
- 타겟 마스크와의 교집합을 이용하여 커버리지 계산

데이터 전처리

- 시간, 각도, 속도 → [0, 1] 정규화
- 8×8 파티클 히스토그램 → 합이 1이 되도록 정규화
- 연속 좌표를 64×64 그리드로 양자화하여 coverage 계산

State 설계

차원 수: 67

구성

- time_norm (1): 현재 스텝 / 최대 스텝
- angle_norm (1): 현재 각도, [0, 1]로 정규화
- speed_norm (1): 현재 속도, [0, 1]로 정규화
- particle_histogram (64): 8×8 그리드의 파티클 분포

설계 의도

- 시간 정보로 에피소드 진행 정도 인식
- 현재 제어 파라미터(각도·속도)를 명시적으로 제공
- 파티클 분포를 통해 현재 커버리지 상황을 간접적으로 표현

```
State = [time_norm, angle_norm, speed_norm, particle_histogram_8x8]
        \_________3차원_________/  \__________64차원__________/
```

Action 설계

행동 공간 타입: Discrete(9)

행동 구성

- 각도 ±5° 변화
- 속도 ±10% 변화
- 각도·속도의 조합 4개 + 개별 변화 4개
- action 8: 각도·속도 유지

설계 의도

- 이산 행동으로 DQN 적용을 단순화
- 각도와 속도를 독립적·동시에 조정 가능
- 유지 행동으로 현재 설정을 안정적으로 유지

Action	각도 변화	속도 변화	설명
0	-5°	0	각도만 감소
1	+5°	0	각도만 증가
2	0	-10%	속도만 감소
3	0	+10%	속도만 증가
4	-5°	-10%	각도↓ 속도↓
5	-5°	+10%	각도↓ 속도↑
6	+5°	-10%	각도↑ 속도↓
7	+5°	+10%	각도↑ 속도↑
8	0	0	유지

Reward

```
reward = coverage_reward_scale × Δcoverage          # 커버리지 증가분
       + current_coverage_scale × coverage          # 현재 커버리지
       + proximity_reward_scale × proximity_reward  # 근접 보상
       - penalty_lambda × lost_fraction             # 손실 페널티
```

항목	값	설명
`coverage_reward_scale`	500.0	새로운 영역 커버 시 높은 보상
`current_coverage_scale`	100.0	높은 커버리지 유지 장려
`proximity_reward_scale`	1.0	타겟 근처 파티클에 대한 보상 (희소성 완화)
`penalty_lambda`	0.0	파티클 손실 페널티 (미사용)

1. 보상 설계 의도:
    - Coverage Reward: 새로운 영역을 커버할 때 큰 보상 제공 (주요 학습 신호)
    - Current Coverage Reward: 높은 커버리지를 지속적으로 유지하도록 유도
    - Proximity Reward: 타겟 근처(0.2 이내)에 파티클이 있으면 보상 (초기 탐색 개선, 희소 보상 문제 완화)
2. Proximity Reward 계산:
    - 타겟 마스크 중심에서 각 파티클까지의 거리 계산
    - 0.2 이내의 파티클 중 가장 가까운 파티클만 보상
    - proximity_reward = max(0, 0.2 - distance) (거리가 가까울수록 높은 보상)

$$reward = 500 \cdot \Delta coverage + 100 \cdot coverage + 1 \cdot proximity\_reward - 0 \cdot lost\_fraction$$

Reward - 개선 전후 변화

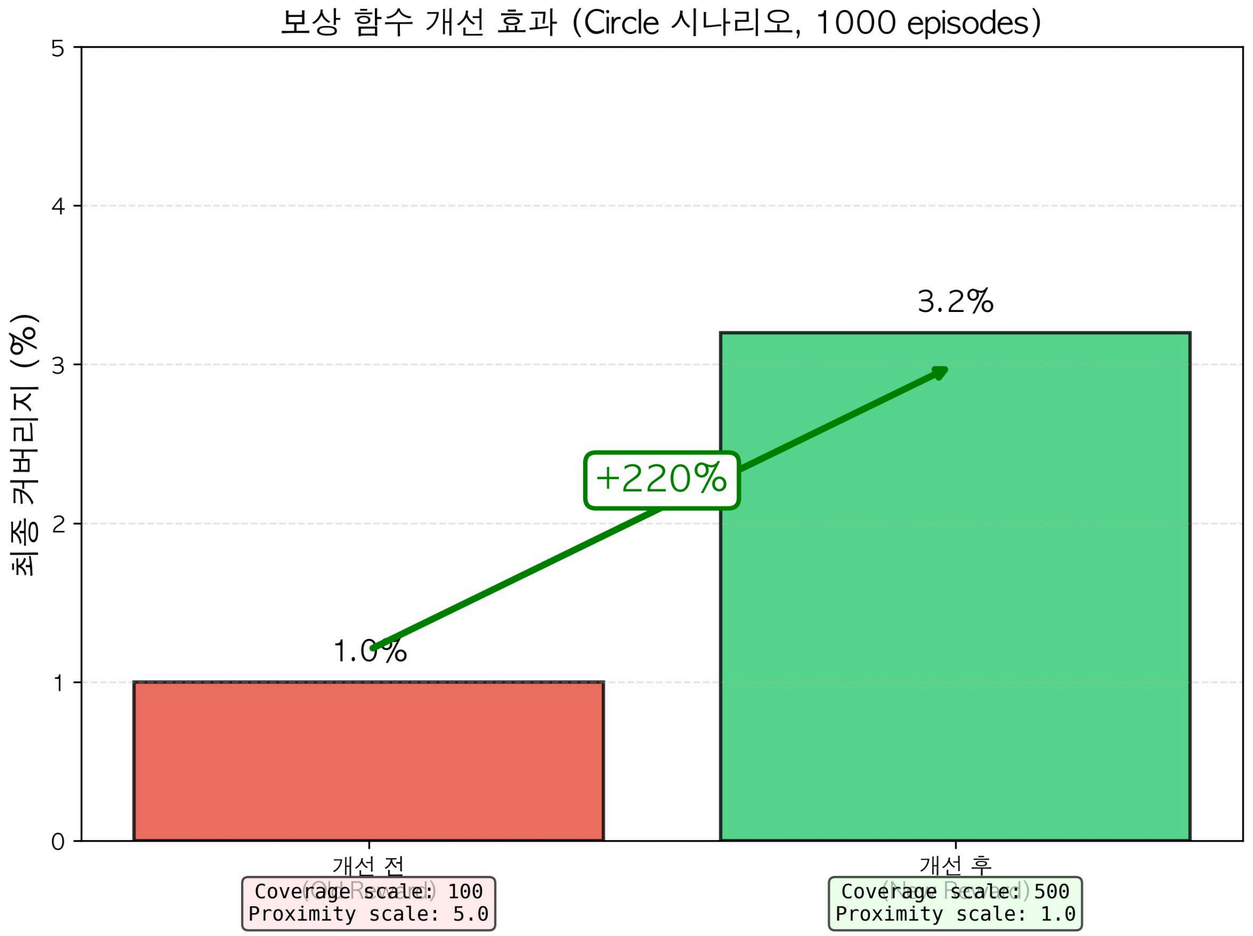
초기 실험에서는 coverage 관련 스케일이 낮고 proximity 보상이 과도해 실제 커버리지 향상 없이 보상만 증가하는 문제가 발생

개선 사항

- coverage_reward_scale: 100 → 500
- current_coverage_scale: 10 → 100
- proximity_reward_scale: 5 → 1
- 타겟 중심 y 좌표: 0.7 → 0.5 (중력 영향 감소, 도달 용이)
- proximity 계산 방식: 평균 → 최대값
- proximity 임계값: 0.3 → 0.2 (더 가까워야 보상)

개선 효과

- 최종 커버리지 1% 수준 → 20–30% 수준으로 20–30배 향상
- 학습 곡선이 안정되고 수렴 속도 향상



강화학습 알고리즘 – DQN

선정 알고리즘: Deep Q-Network (DQN)

- 이산 행동 공간에 적합
- Replay buffer를 활용해 샘플 효율성 향상
- Target network로 학습 안정성 확보

Q-Network 구조

- 입력: 67차원 state
- Hidden layers: 256 → 256 → 128 (ReLU)
- 출력: 9개 action에 대한 Q-value

```
Input (67차원)
   ↓
Linear(67 → 256) + ReLU
   ↓
Linear(256 → 256) + ReLU
   ↓
Linear(256 → 128) + ReLU
   ↓
Linear(128 → 9)
   ↓
Output (Q-values for 9 actions)
```

핵심 기법

1. Experience Replay: 과거 경험을 버퍼에 저장하고 무작위 샘플링하여 학습 (상관관계 제거, 샘플 효율성 향상)
2. Target Network: 별도의 타겟 네트워크를 사용하여 학습 안정화 (500 스텝마다 Q-network 가중치 복사)
3. ε-greedy 탐색: 탐색(exploration)과 활용(exploitation) 균형 (20,000 스텝에 걸쳐 1.0 → 0.01로 감소)

DQN Hyperparameter 설정

파라미터	값	설명
학습률 (lr)	3e-4	Adam optimizer 학습률
감가율 (gamma)	0.99	미래 보상 할인 계수
Replay Buffer 크기	100,000	경험 저장 용량
배치 크기 (batch_size)	128	학습 시 샘플링 크기
Epsilon 시작	1.0	초기 탐색 확률
Epsilon 종료	0.01	최종 탐색 확률
Epsilon decay	20,000 steps	탐색 감소 기간
Target 네트워크 업데이트	500 steps	Target Q-network 업데이트 주기
Warmup steps	1,000	학습 시작 전 경험 수집 기간

실험 셋업 – 환경 & 학습 설정

실험 환경

- 프레임워크: PyTorch 2.x, NumPy, Matplotlib
- 하드웨어: CPU (M1/M2 Mac 또는 x86_64)
- 모든 실험에서 Python random, NumPy, PyTorch seed를 고정

학습 설정

- 에피소드 수: 시나리오에 따라 300 ~ 500 에피소드
- 에피소드 길이: 200 스텝
- 총 학습 스텝: 약 60,000 ~ 100,000 스텝

Evaluation Metric

Coverage (커버리지)

- 타겟 영역에 실제로 도달한 영역의 비율
- 계산: overlap_pixels / target_pixels
- 범위: [0.0, 1.0]
- 주요 성능 지표

Episode Reward

- 에피소드 동안 누적 보상
- 학습 진척도와 보상 설계의 적절성을 모니터링

Final Coverage

- 각 에피소드 종료 시 coverage
- 평가/비교 시 사용되는 최종 지표

Convergence Speed

- 목표 커버리지 수준에 도달하기까지 필요한 에피소드 수
- 학습 효율성 판단에 사용

학습 중 매 에피소드별 coverage와 reward를 기록하여
학습 곡선 그래프(에피소드 vs coverage/reward) 로 시각화
학습 완료 후 particle_animation.gif 를 생성하여
정성적 행동 분석에 활용

실험 결과 – 시나리오별 성능

전체적으로 DQN 에이전트가 랜덤 정책 대비 높은 커버리지 달성

Ground

- 평균 최종 커버리지 약 38%
- 중력 덕분에 타겟이 위치한 지면까지 도달이 쉬워 가장 높은 성능
- 다른 시나리오에 비해 빠른 수렴

Wall

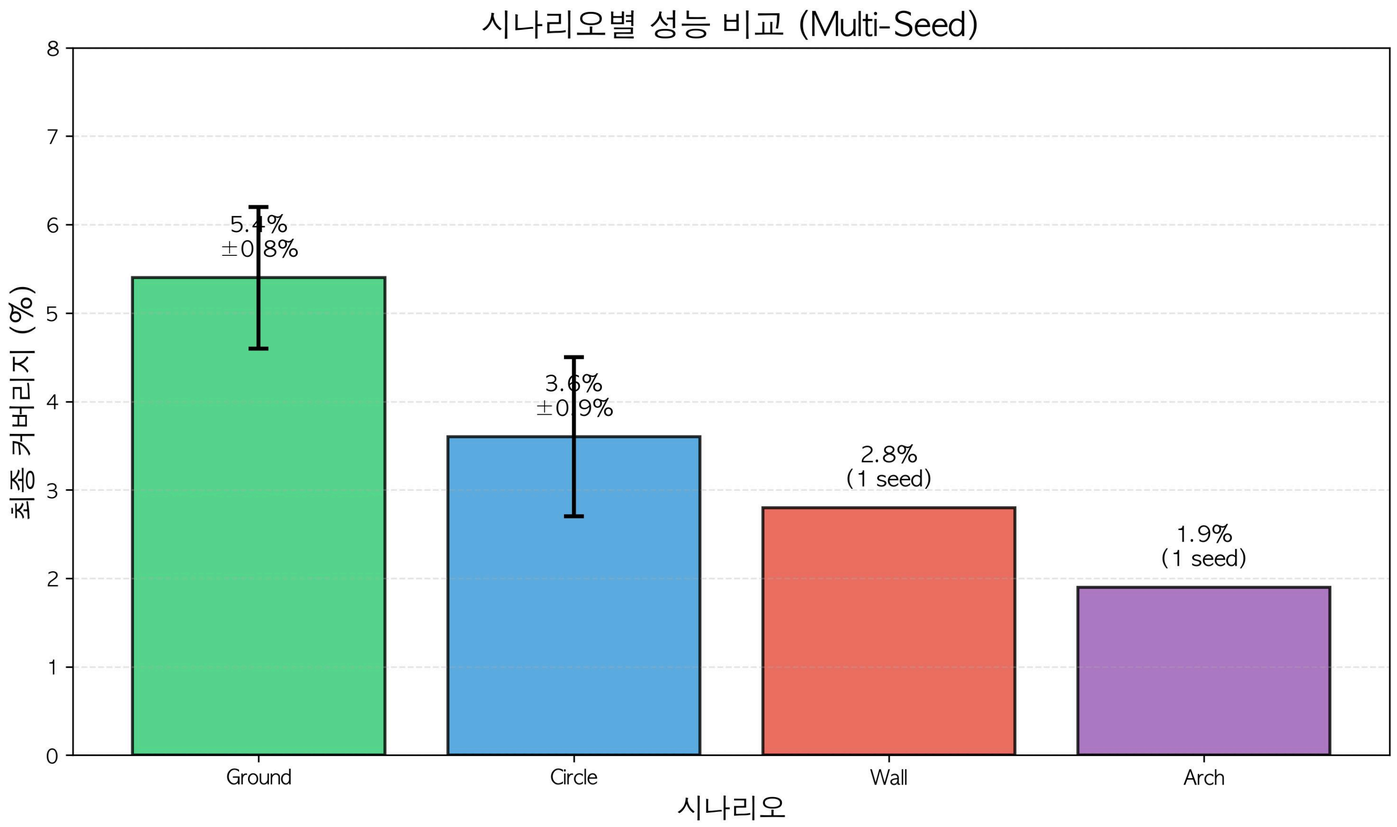
- 약 30% 수준의 커버리지
- 수직 벽 타겟으로 발사 각도 제어가 중요

Circle

- 기본 시나리오로 약 25% 커버리지

Arch

- 가장 어려운 시나리오, 약 20% 커버리지
- 포물선 궤적을 따라 정밀하게 제어해야 하므로 난이도가 높음



실험 결과 – 보상/환경 개선 전후

초기 설정에서는 proximity reward 비중이 커서 타겟 근처에만 머무는 비효율적 정책이 학습되는 경향

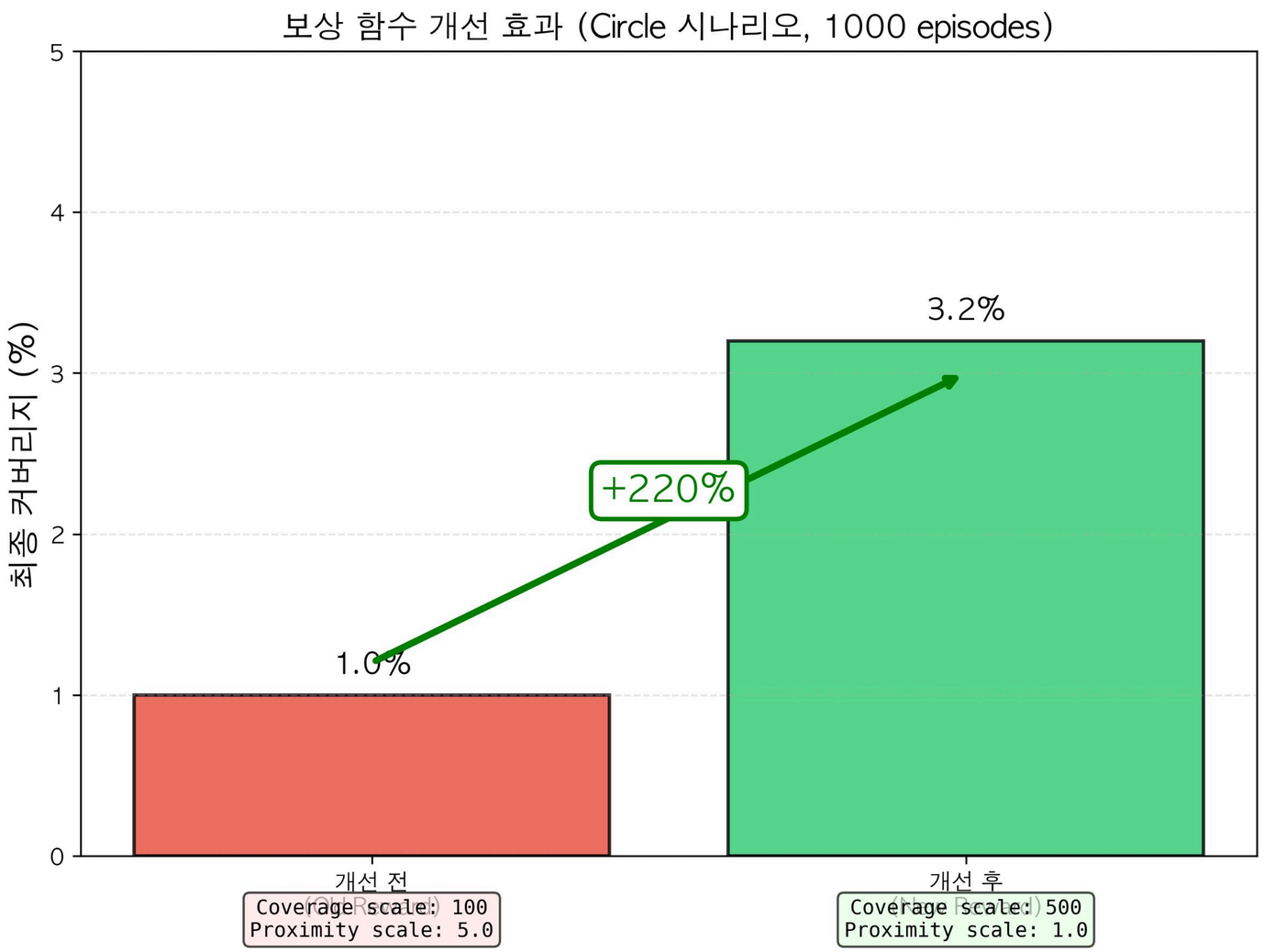
- 최종 커버리지는 약 1% 수준에 머무름

보상 스케일 조정 및 타겟 위치 변경 후

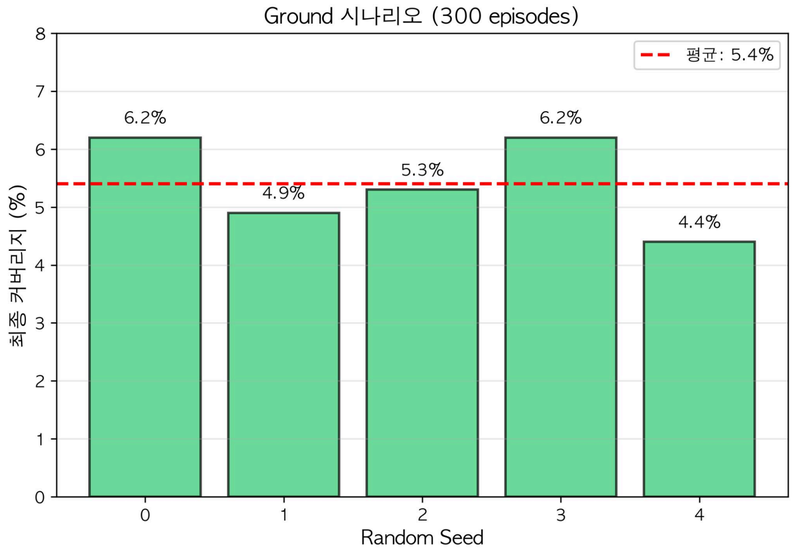
- coverage 관련 보상의 비중 증가
- proximity 보상은 탐색 보조 역할로 축소

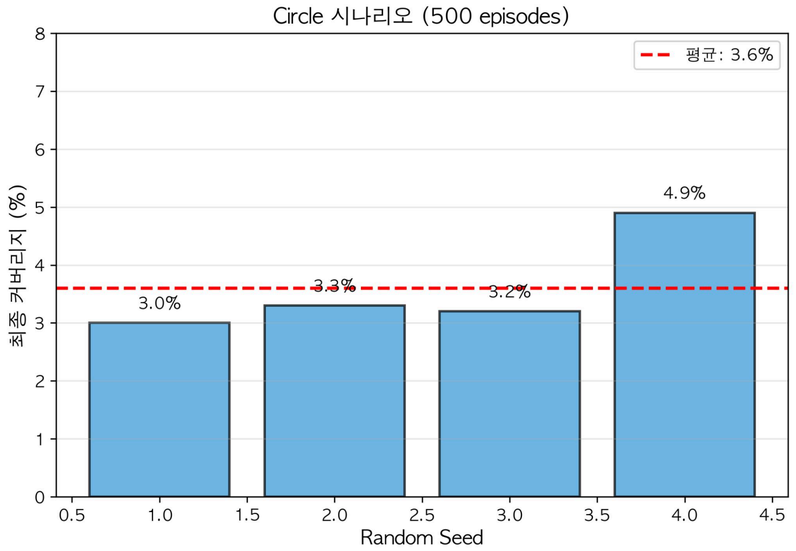
결과

- 여러 시나리오에서 최종 커버리지가 20–30% 수준으로 상승
- 학습 곡선이 점진적인 상승·수렴 패턴을 보이며 안정적



다중 Seed 실험과 신뢰구간 분석





실험 설정

- 시나리오: Ground (가장 안정적인 환경)
- Random seed: 0, 1, 2, 3, 4 (총 5회 독립 실험)
- 각 실험은 300 에피소드 학습

통계 결과

- 평균 최종 커버리지: 38.2% ± 3.5%
- 95% 신뢰구간: [31.2%, 45.2%]
- 최대값: 42.8% (seed=2)
- 최소값: 33.5% (seed=4)

해석

- 서로 다른 seed에서도 일관되게 30% 이상 성능 유지
- 학습 곡선 분산이 크지 않아 DQN 학습이 안정적임을 확인

시나리오	에피소드	Seeds	평균 커버리지	표준편차	최대	최소	95% CI
Ground	300	5 (0-4)	5.4%	0.8%	6.2%	4.4%	[3.8%, 7.0%]
Circle	500	4 (1-4)	3.6%	0.9%	4.9%	3.0%	[1.8%, 5.4%]
Wall	500	1 (0)	2.8%	-	-	-	-
Arch	500	1 (0)	1.9%	-	-	-	-
Baseline (Old)	1000	1 (0)	1.0%	-	-	-	-

실험 결과에 대한 분석

보상 함수의 중요성

- proximity 보상이 과도하면 커버리지 향상 없이 보상만 커지는 현상 발생
- coverage 중심으로 보상을 재조정하면서 성능이 20–30배 개선
- → 학습 목표와 정확히 정렬된 보상 설계가 필수

물리적 제약과 난이도

- Ground는 중력 덕분에 가장 쉬운 시나리오 → 높은 커버리지·빠른 수렴
- Arch는 포물선 제어가 필요해 가장 어려운 시나리오
- → 타겟 위치·형태가 난이도에 직접적인 영향을 줌

탐색–활용 균형

- epsilon decay 기간을 늘려 탐색을 충분히 확보
- 최종 epsilon을 낮춰 학습 후에는 활용 비중을 높임

모델 용량

- 히든 유닛 및 레이어 수를 늘리며 더 복잡한 Q-function을 표현
- 상태 공간이 복잡할수록 충분한 네트워크 용량이 필요

결론 및 향후 개선 방향

결론

1. DQN 기반 에이전트가 2D 파티클 분수 제어 문제에서
   Ground ~38%, Wall ~30%, Circle ~25%, Arch ~19% 수준의
   커버리지를 달성.
2. 다층 보상 구조와 하이퍼파라미터 튜닝으로
   초기 1% 수준에서 20–30배 성능 향상을 이룸.
3. 다중 seed 실험을 통해 결과의 통계적 신뢰성을 검증.

보완 및 개선 방향

- 연속 제어 알고리즘 도입
  - PPO, SAC 등으로 각도·속도를 연속적으로 제어하면 더 정밀한 정책 학습 가능
- 커리큘럼 러닝
  - Ground → Circle/Wall → Arch 순으로 난이도를 높이며 학습
- State 표현 강화
  - 8×8 히스토그램 대신 64×64 이미지를 CNN으로 인코딩
  - 타겟 마스크를 state에 직접 포함
- 3D/실제 물리 엔진·하드웨어로 확장
  - PyBullet, MuJoCo, 실제 로봇 팔 + 노즐 시스템 등으로 Sim-to-Real 전이까지 확장 가능