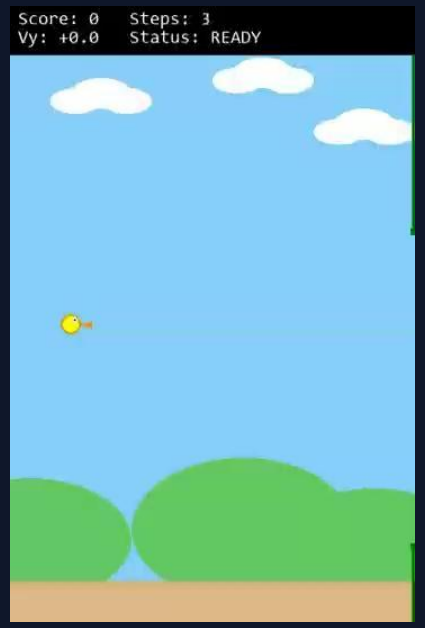
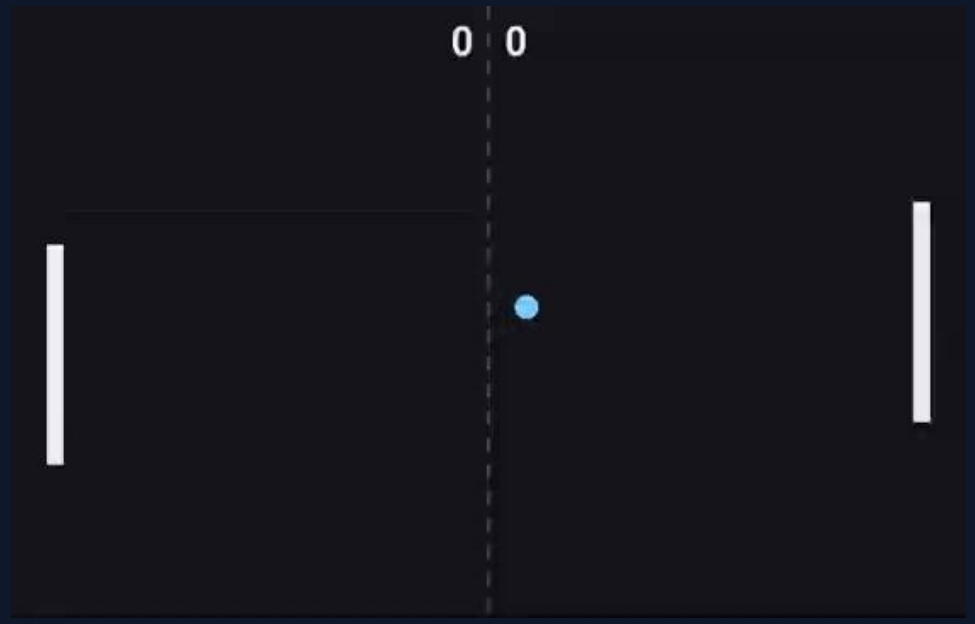
A 2D-Gaming Benchmark for Indie Developers Using Deep Reinforcement Learning

Alex Lowe · Ontario Tech University · Honours Thesis 2026
Supervisors: Cristiano Politowski, Ali Neshati



Flappy Bird (easy)

Snake (easy)



Pong (easy)

The Problem and Our Solution

Game testing is costly

One of the most time-consuming phases of development. Large studios have dedicated QA teams, most indie teams wont.

DRL can play at human level and beyond

But existing solutions require full engines: Unity, Unreal, Godot. Academic benchmarks target algorithm research, not developer workflows.

The gap

Can a small developer use DRL for game balance testing on a consumer PC? Is the game appropriately difficult?

3 games	**4 algorithms**	**3 difficulty levels**	**no discrete GPU**

Research Questions

RQ1 **Feasibility**

Is DRL-based playtesting practical for indie developers?

RQ2 **Performance**

How do PPO, A2C, TRPO, and RPPO compare across 3 games?

RQ3 **Generalisation**

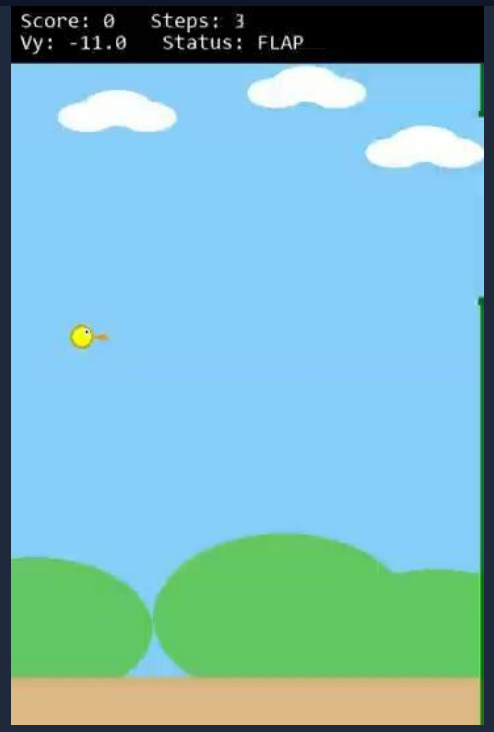
Do agents trained on Normal transfer to Easy and Hard?

RQ4 **Dev Perspective**

What are the real challenges and trade-offs?

Approach

Pygame + Gymnasium + Stable-Baselines3 · doubling-and-halving difficulty · 10M steps · 100 ep eval per difficulty



1 binary action

Pipe gap width controls difficulty

4 directional actions

Acceleration per apple controls difficulty

3 actions: up / stay / down

Paddle height controls difficulty

10M

Timesteps

10

Parallel Envs

CPU

Ryzen 5 8600G

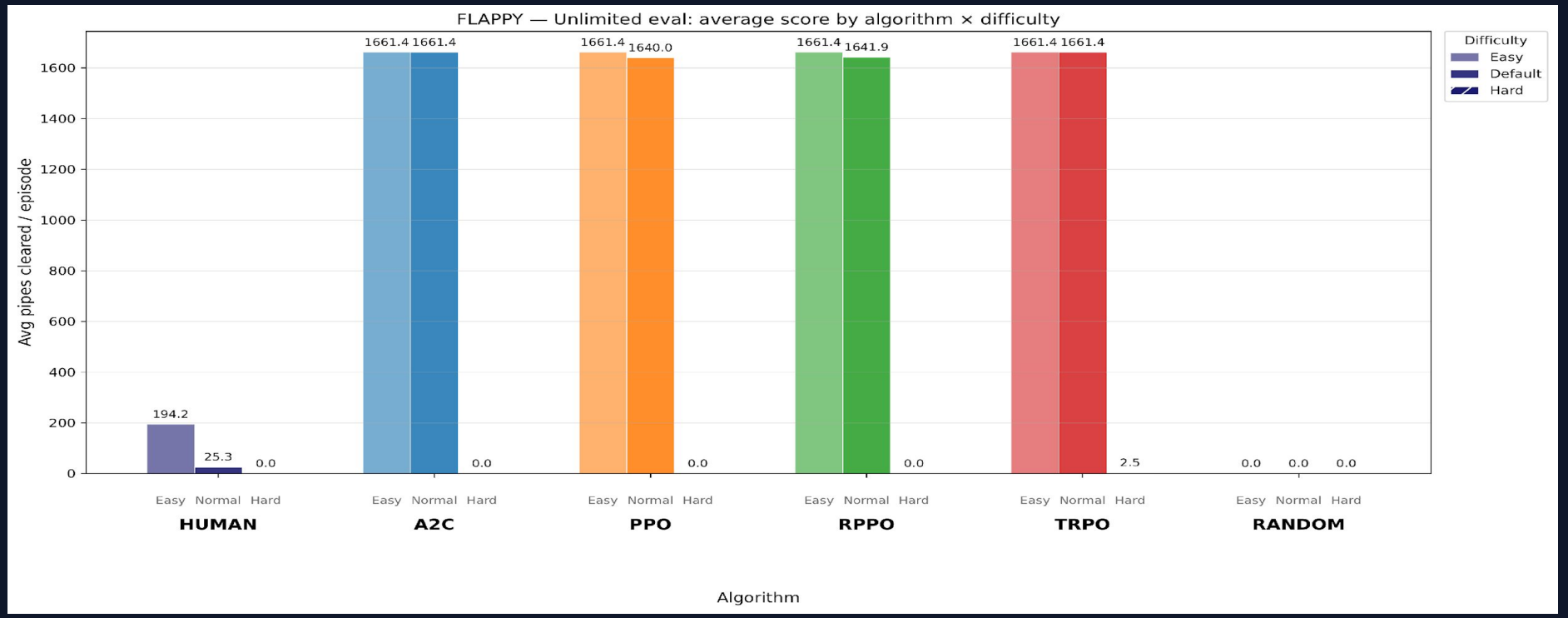
100 episodes

Per Difficulty

Flappy Bird Results

Reactive control · single binary action · pipe gap difficulty

Hard Difficulty



Easy & Normal

All 4 algorithms hit the 30-min cap
(1,661 pipes) every episode

Hard

3 of 4 scored zero, TRPO managed
just 2.5 mean pipes (best episode: 11)

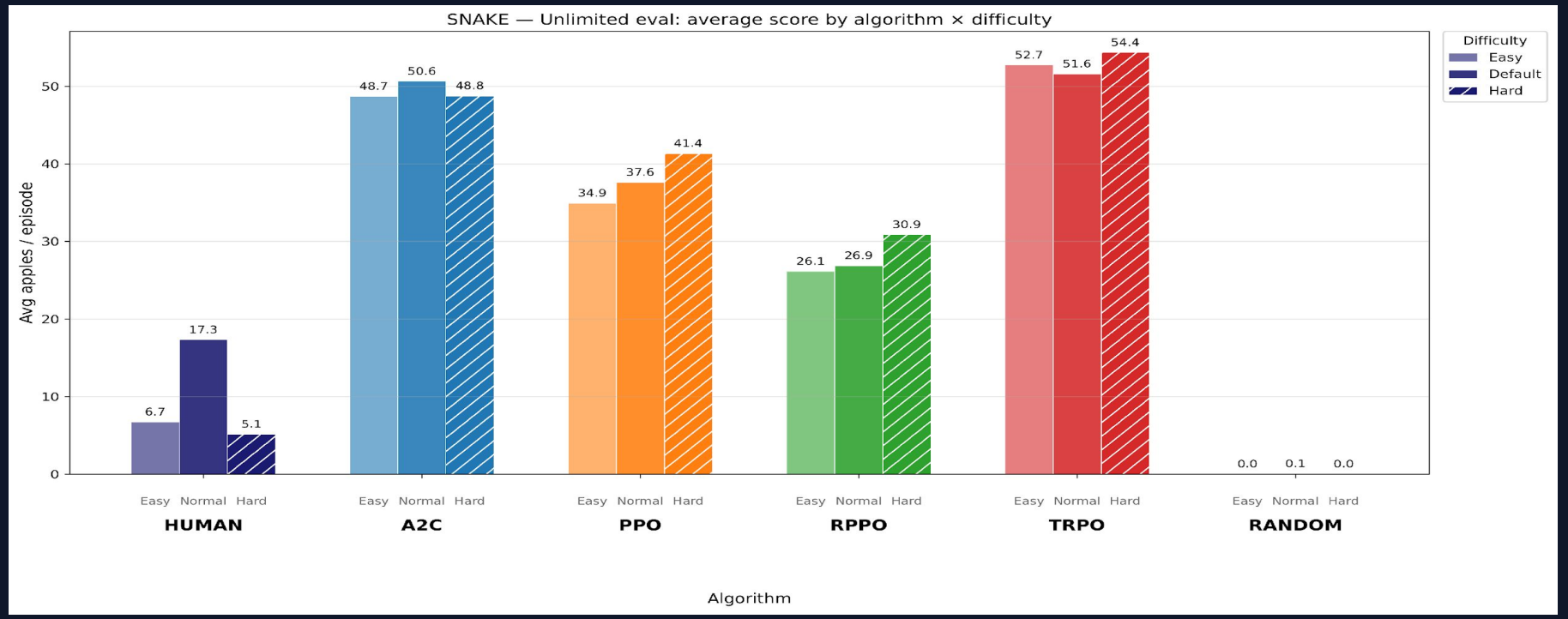
Human Baseline

194 pipes Easy · 25 Normal · 0 Hard
Well below DRL ceiling on Easy/Normal

Snake Results

Long-horizon planning · most reward iterations of any game

Hard Difficulty



DRL Dominated

Even RPPO Easy (26.1 apples) was 4x the human mean (6.7)

Stable Across Difficulties

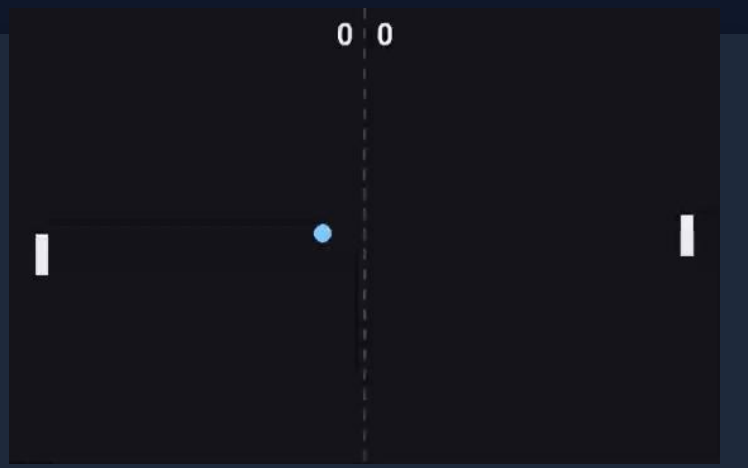
TRPO topped at 54.4 mean apples on Hard
No algorithm degraded meaningfully

Orbiting Problem

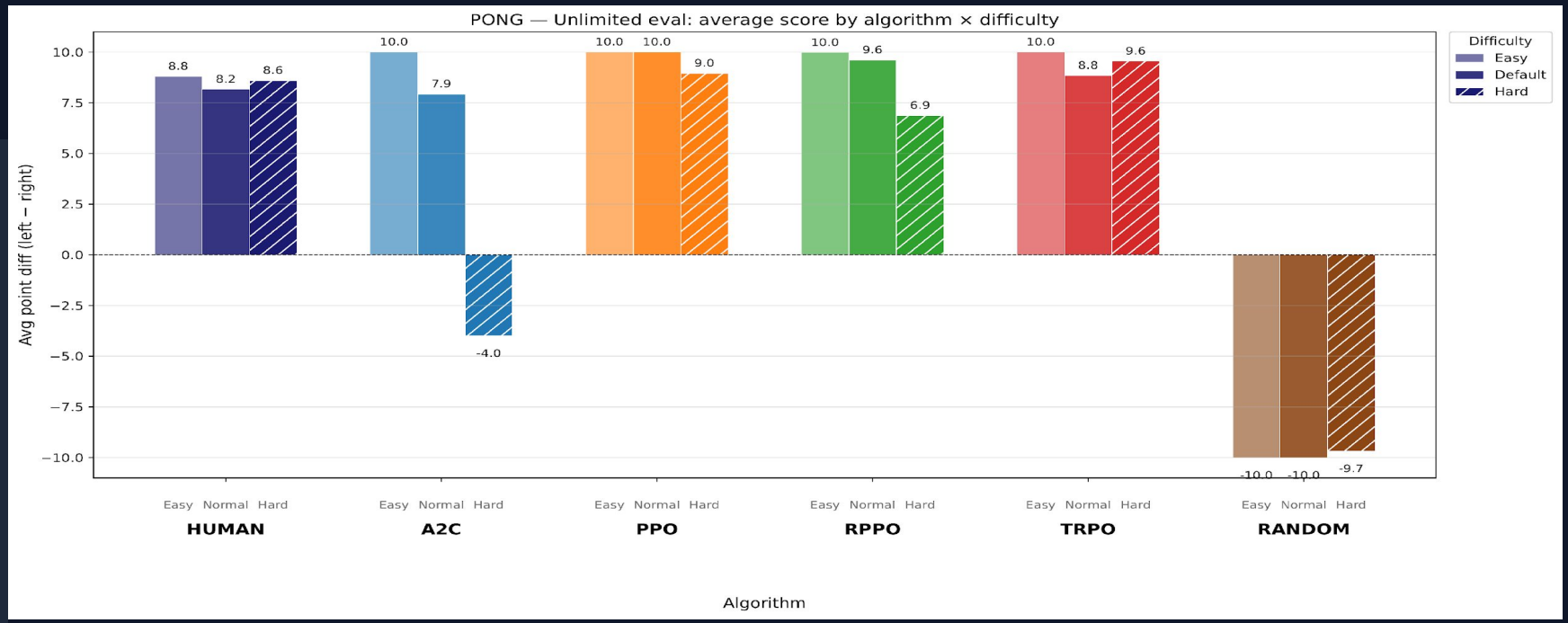
Agents circle the apple instead of collecting it, hardest reward design

Pong Results

Adversarial tracking · the only game where humans stayed competitive



Hard Difficulty



Human Competitive

Human averaged +8.8/+8.2/+8.6 across all 3 difficulties, beat A2C on Normal+Hard

A2C Collapsed on Hard

Dropped to −4.0 mean while TRPO held +9.6, overfitted to paddle size

PPO & TRPO Robust

PPO: +10/+10/+9 · TRPO: +10/+8.8/+9.6
Reactive tracking that generalises

RQ 1: Verdict & Takeaways

Not readily feasible for small studios without significant upfront investment.

Hardware is not the barrier.
Reward design is.

Single release: hard to justify
Live-service game: could pay off over time

Key Takeaways

Use TRPO or PPO for this application for tested games

A2C when iteration speed matters

30 min vs 7.5 h for RPPO

Avoid RPPO at all costs for 2D games

LSTM adds cost, provides no benefit

Having multiple difficulties helps

Eval time dominates once agents perform well

Flappy Bird evaluation alone: ~82 hours

Reward design is the hardest part

RQ 2: Algorithm Comparison

Consistent ranking across all 3 games

1 **TRPO**

#1 in all 3 games · only algorithm to clear Flappy Hard

~70 min

2 **PPO**

Most reliable · never collapsed on any game or difficulty

~80 min

3 **A2C**

Fastest training · strong on Easy/Normal · collapsed on Pong Hard

30–47 min

4 **RPPO**

Weakest results in every game · LSTM provided no benefit

4–7.5 h

RQ 3: Generalisation

Flappy Bird

Easy	Normal	Hard
✓	✓	✗ (3/4 scored zero)

Snake

Easy	Normal	Hard
✓	✓	✓ (stable)

Speed range was not the limiting factor

Pong

Easy	Normal	Hard
✓	✓	✓ (Minus A2C)

A2C overfit to the training paddle size

RQ 4: Development Effort

One-Time Setup (all 3 games)

Game	Dev	DRL Setup	Total
Flappy Bird	4 h	2 h	**6 h**
Snake	6 h	1 h	**7 h**
Pong	3 h	1 h	**4 h**
Total	**13 h**	**4 h**	**17 h**

Per-Game Recurring Costs

Phase	Time
DRL code iteration	30–45 min/cycle
Training (all 4 algos)	7.5–10 h
Eval — Flappy Bird	**~82 h**
Eval — Snake / Pong	~3–4 h each

Reward Design Was the Bottleneck

Snake orbiting suppression took more effort than building the game itself

Flappy gap-centring bug only visible by watching the agent play

Pong CPU opponent needed careful tuning for useful learning signal

What Worked Well

A Consumer CPU handled all 12 configurations, no GPU needed

Stable-Baselines3 required zero modification

CSV training logs and tensorboard make comparing changes straightforward

Limitations & Future Work

Limitations

- Only 3 games
- Single developer, single human baseline player
- Default SB3 hyperparameters, results represent a floor, and may represent a ceiling

Future Work

- **Donkey Kong clone in active development**
  Multi-level platformer with climbing, hazards, moving enemies
- **Systematic hyperparameter tuning**
  Per algorithm, per game, establish whether defaults are near optimal
- **Continuous difficulty sweeps**
  Using much more difficulties
- **Apply to a game under active development**
  Test the workflow when both game and agent change simultaneously