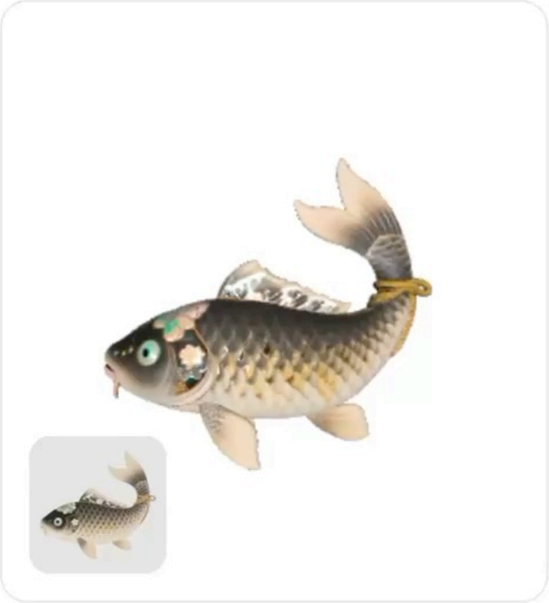
Lecture 14:

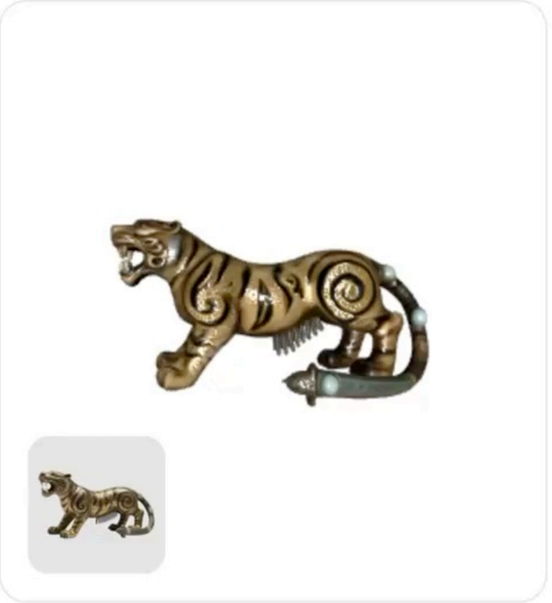
Automating Design Feedback for Interactive 3D Worlds

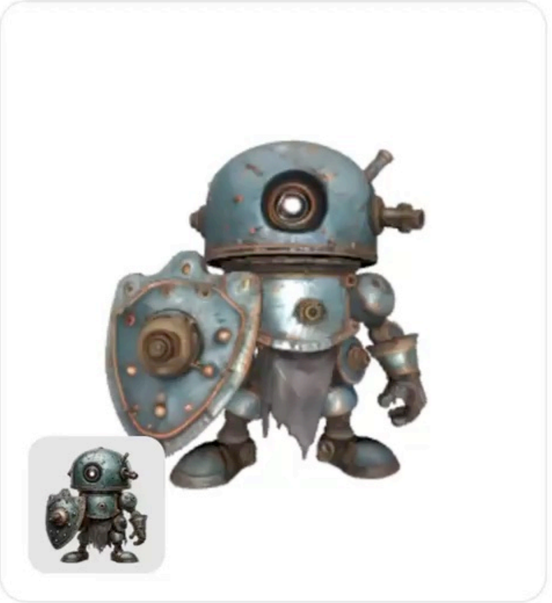
Visual Computing Systems
Stanford CS348K, Spring 2026

A photo of a 40-something male professor with brown hair wearing a cashmere gray hoodie speaking to an audience at the I3D 2026 conference, which is hosted in a conference hall at Lucasfilm in San Francisco. The audience is clearly loving what they are hearing.

3D objects, textures, and materials



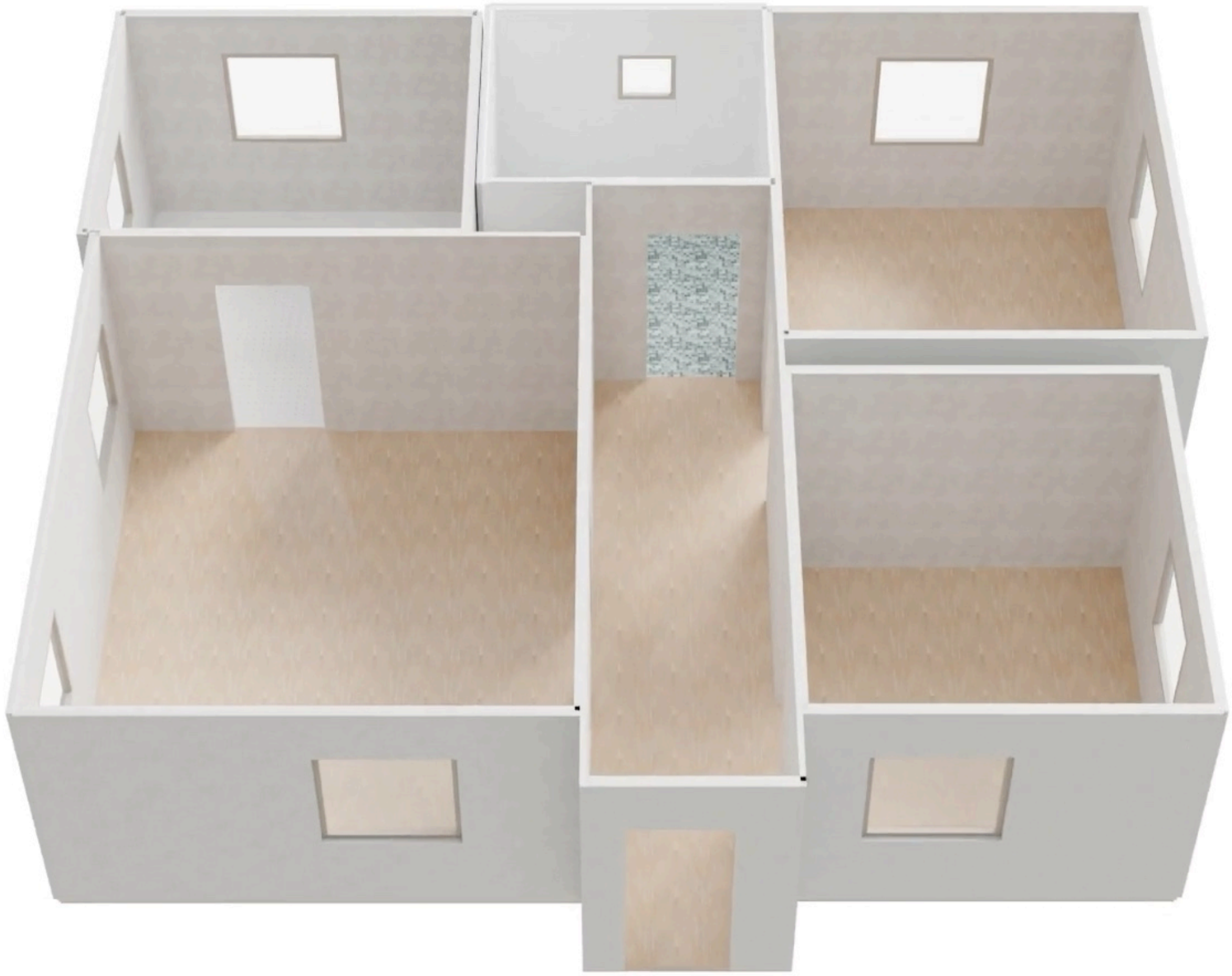




Hunyuan3D Model 2.1

Articulated 3D objects

Full 3D scenes



[SceneSmith: Pfaff et al 2026]

Full 3D scenes

[WorldLabs]

ROSEBUD AI

Create from Scratch

Describe your game below, or **pick a template** →

Create me a racing game set on the streets of San Francisco where the key challenge is to avoid ice patches on the road which cause me to skid out of control.

gemini-flash 158/500

UPLOAD IMAGE

Create →


SPEED: 0 MPH
0 0 0
SKIDS: 0

Output from Genie 3

Cost of 3D content generation dropping rapidly…
Ubiquitous creation and simulation of high-fidelity 3D worlds

Generated 3D content
(Objects, materials, lighting, 3D gaussians, scripts)

Render generated
3D content + Neural 2D “uplift”

Fully action-conditioned video-generation (video-based world model)

Unity

UNREAL ENGINE

ROBLOX

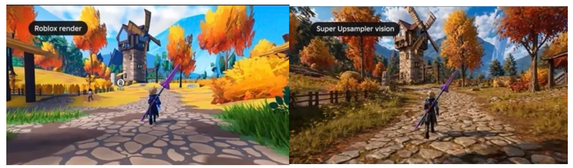
SPARK

Genie 3
BY GOOGLE DEEPMIND

Odyssey

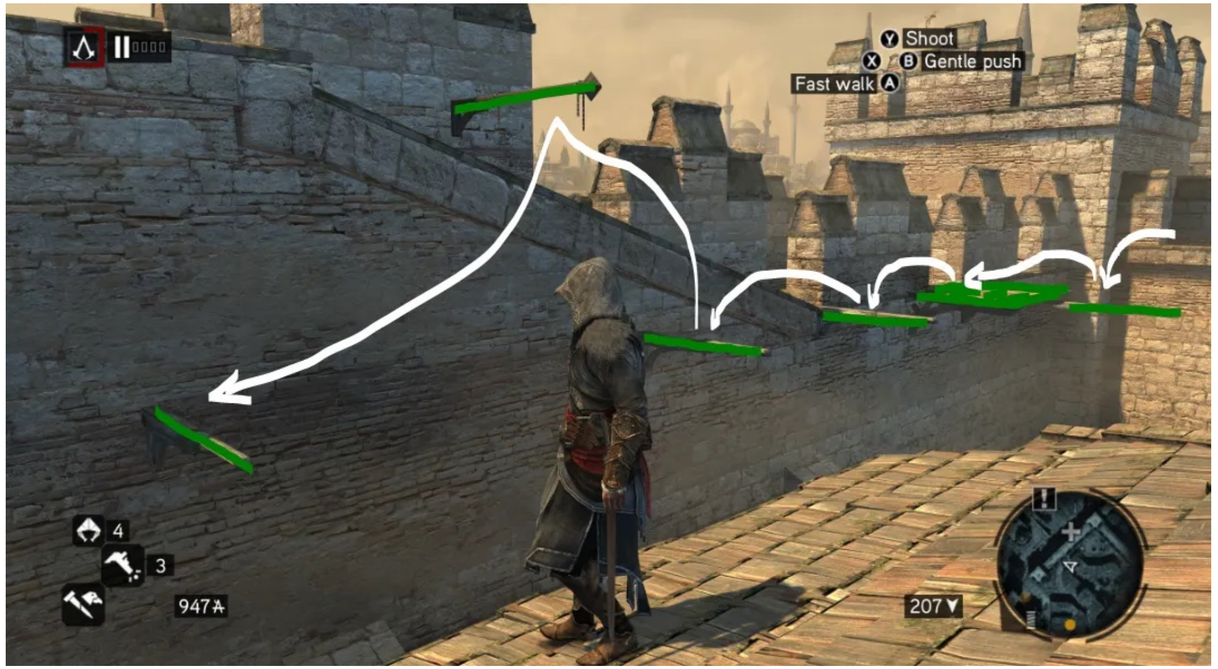
GODOT
Game engine

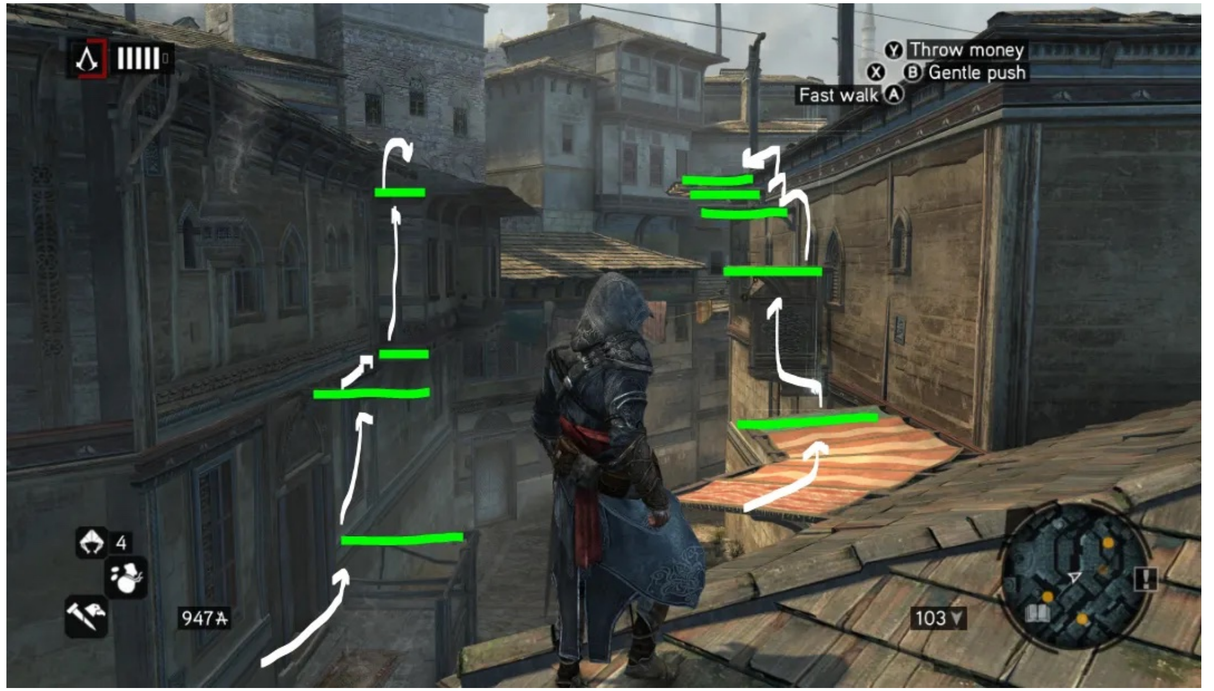
ROBLOX Hybrid

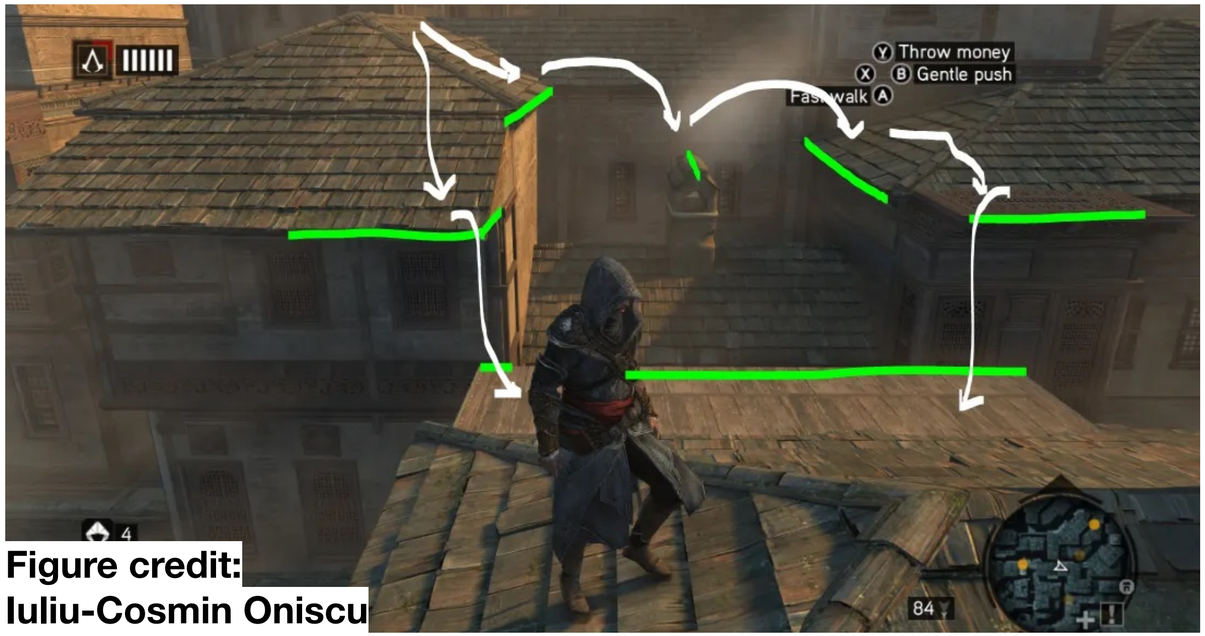


NVIDIA
Cosmos

Great aesthetics != great experiences







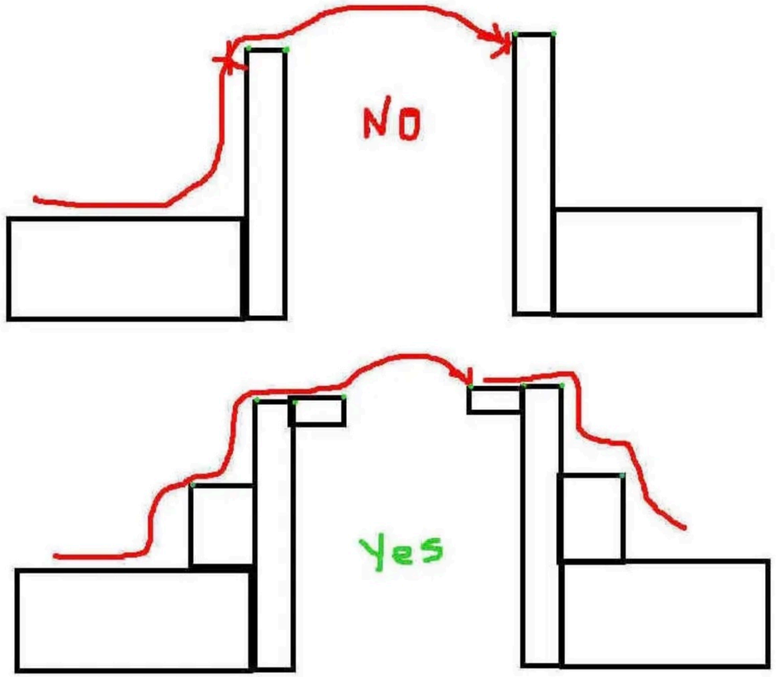
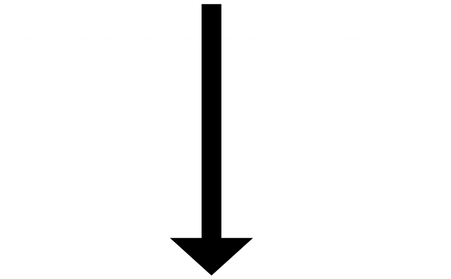


Figure credit:
Iuliu-Cosmin Oniscu
"A level design look at parkour rooftop connectivity in Assassins Creed Urban Spaces."

Balance and design of tools/components in a crafting experience

AI increases generation velocity



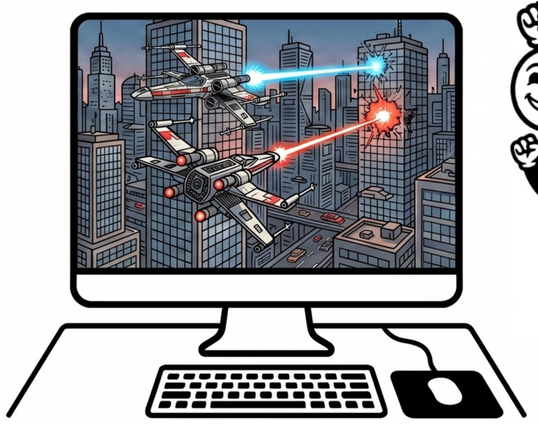
+ Claude

10x more capability

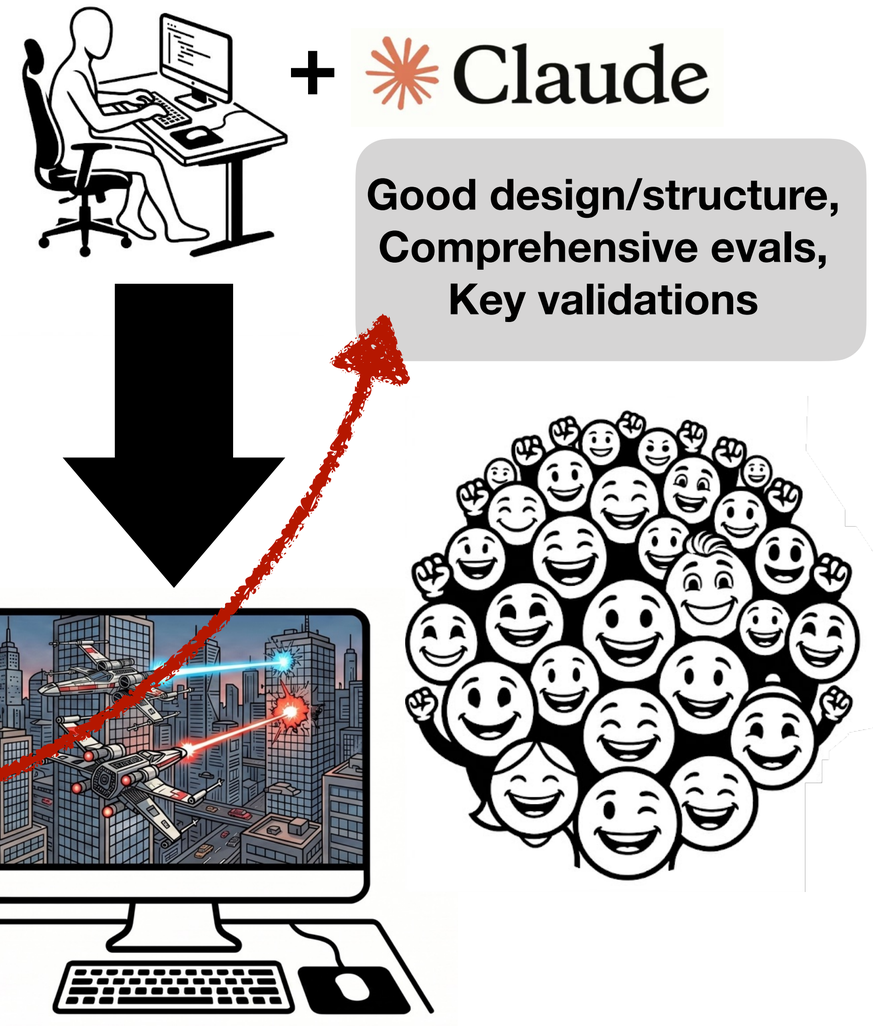
STUCK!

+ Claude

Good design/structure, Comprehensive evals, Key validations

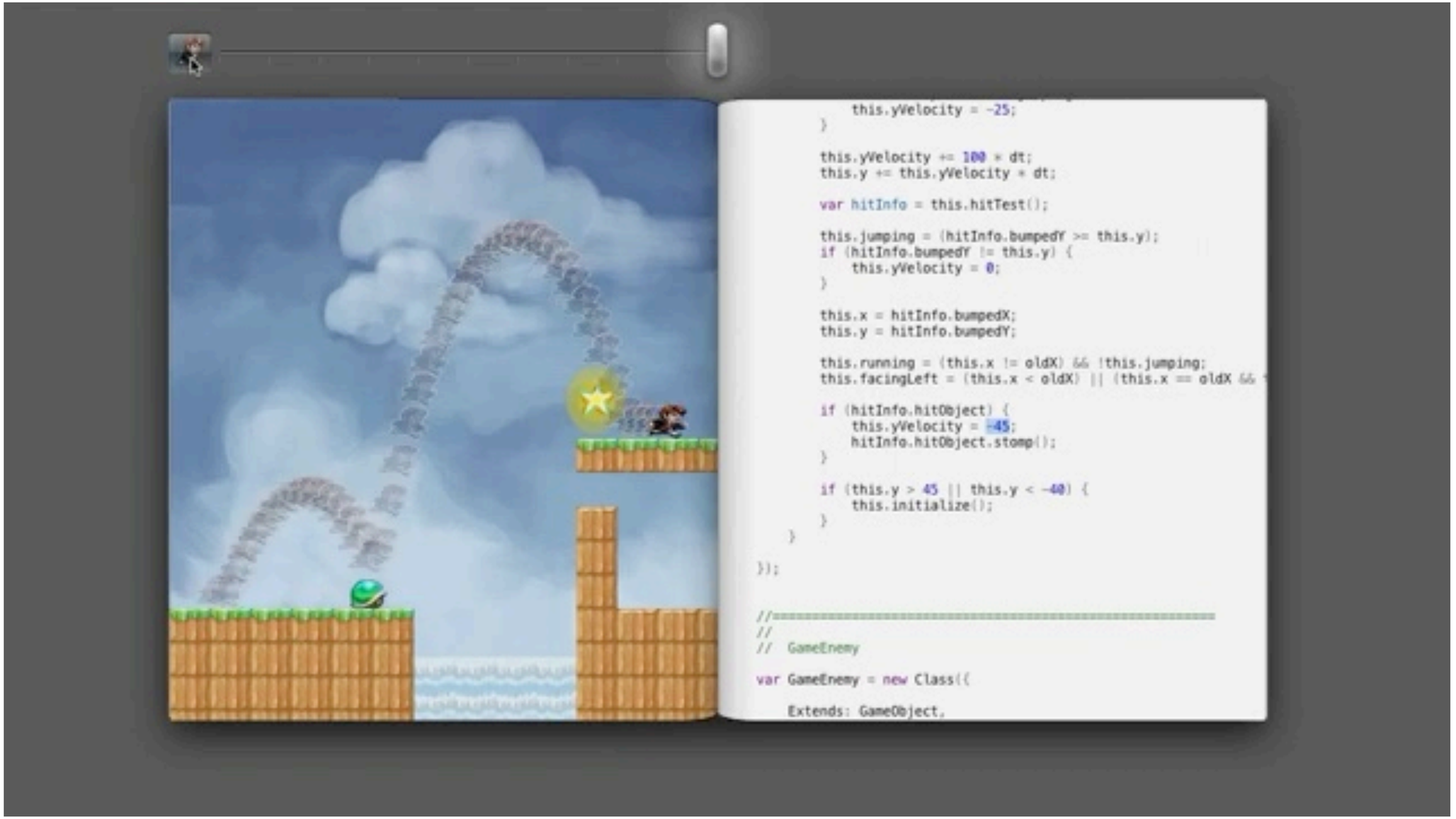


High velocity generation will be limited in impact without structure and tooling to manage, analyze, and control it

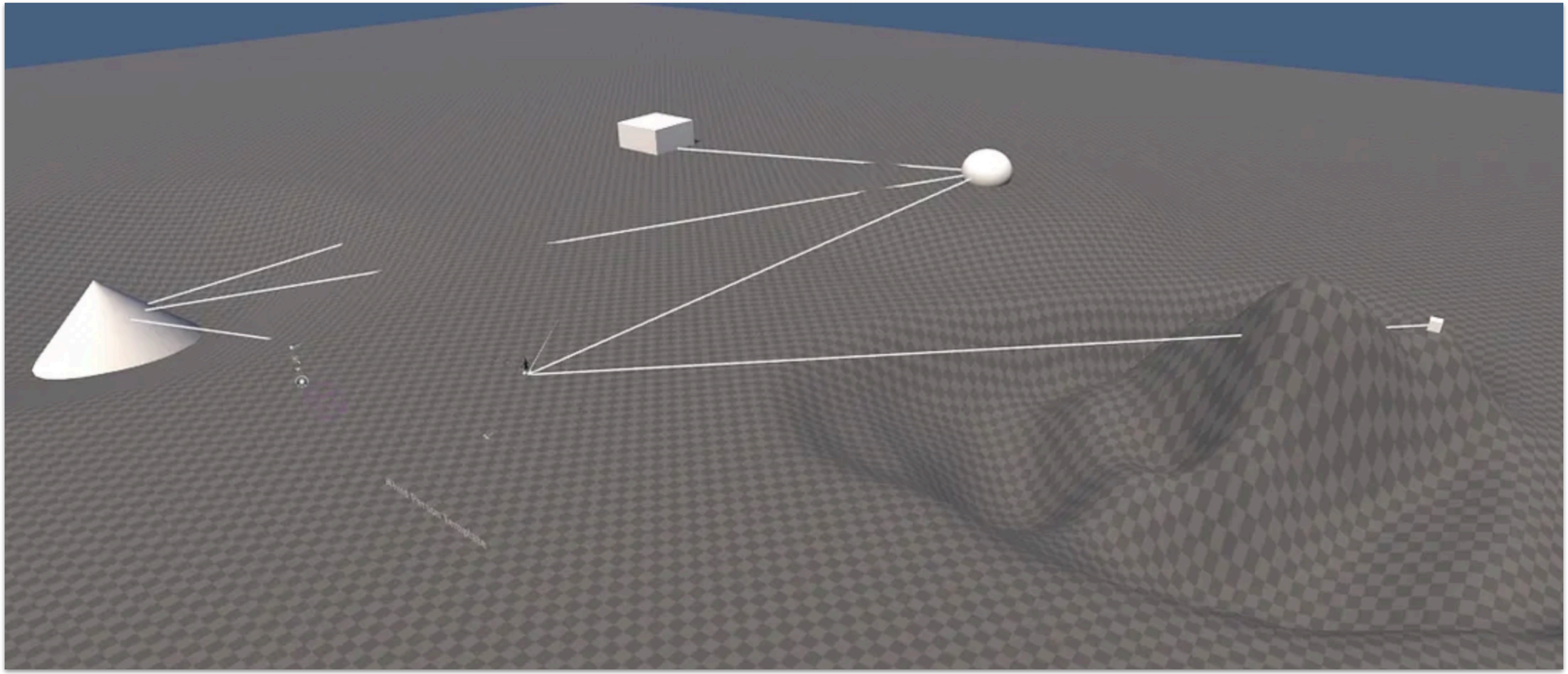


So how can we provide this support for high-velocity creation of interactive 3D experiences?

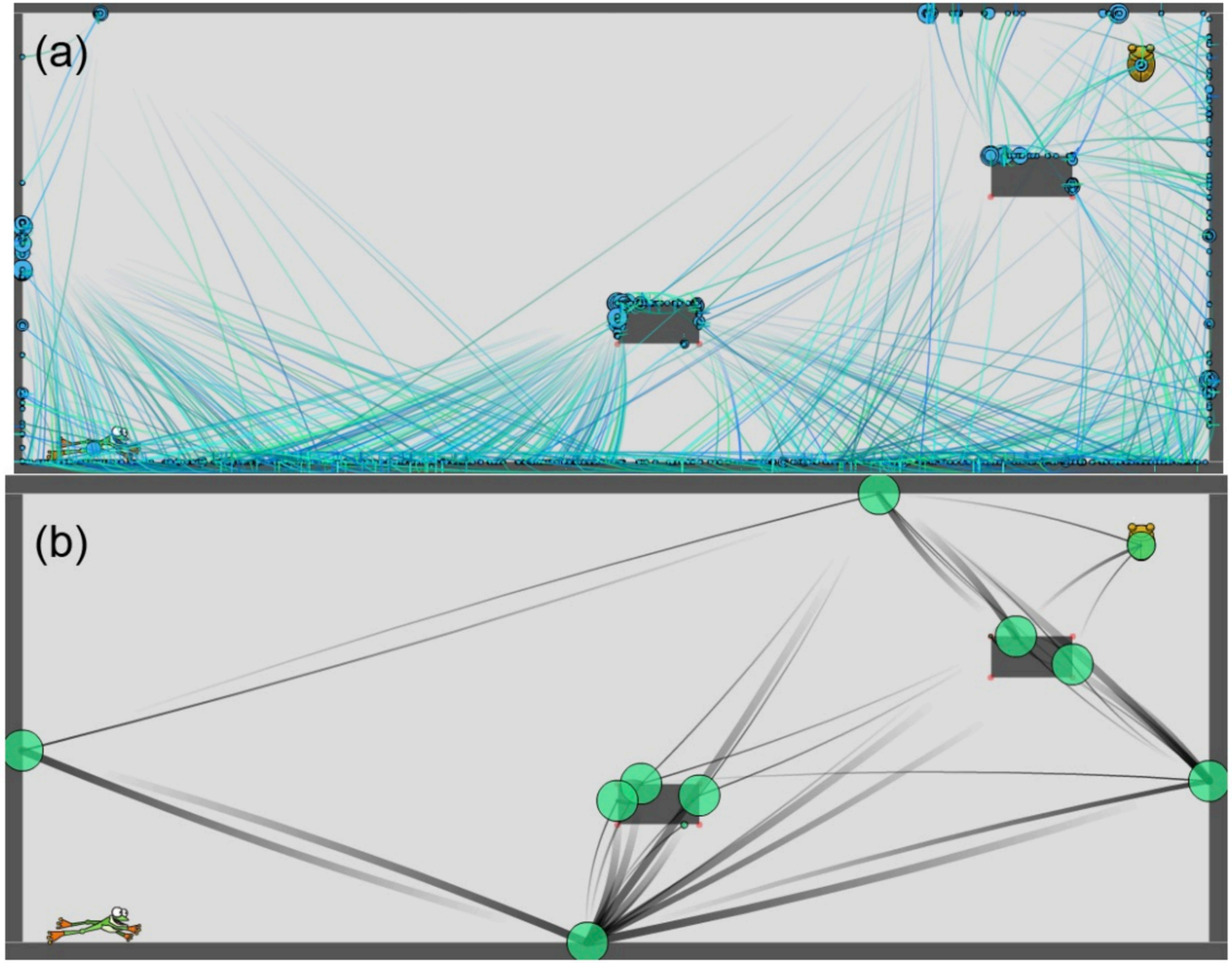
Automating generation of design feedback



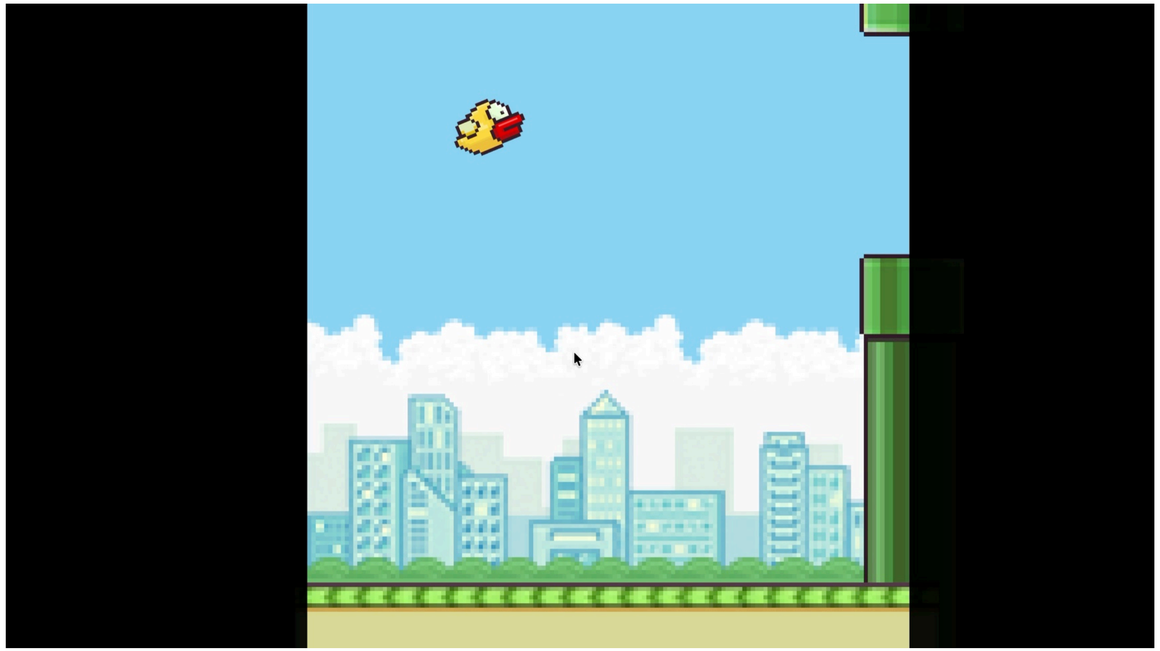
Automating generation of design feedback

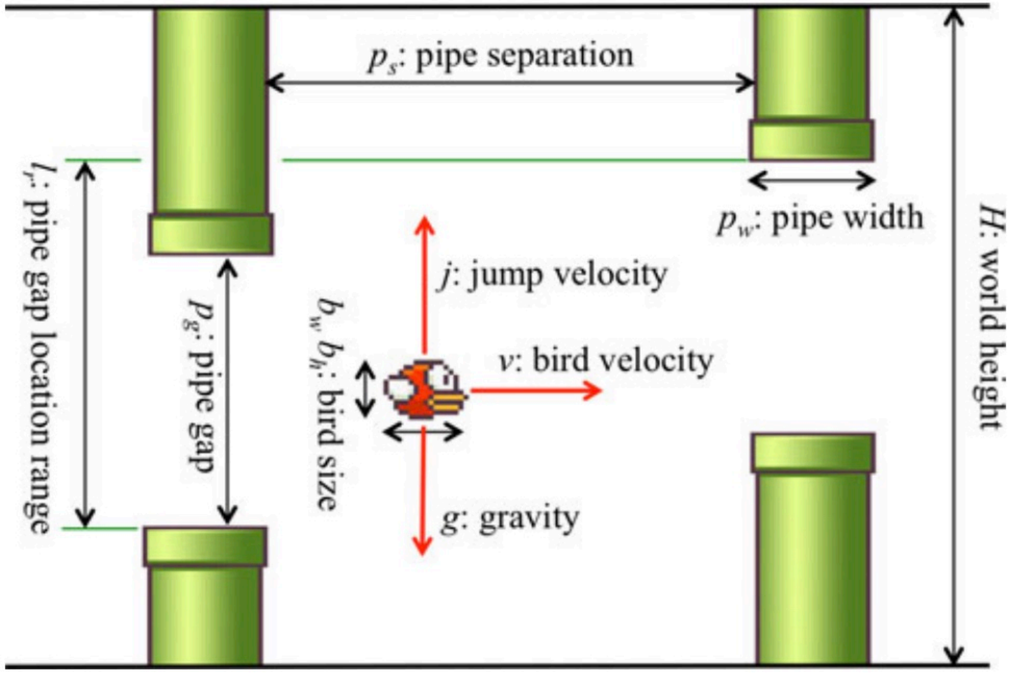


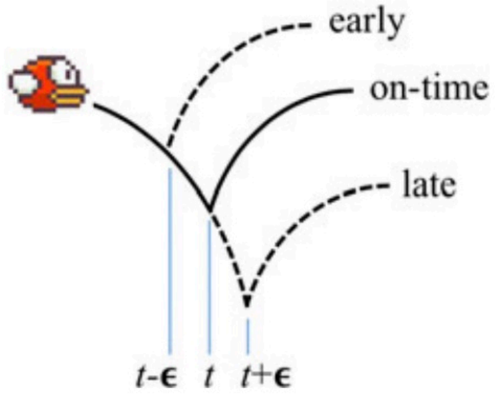
Automating generation of design feedback

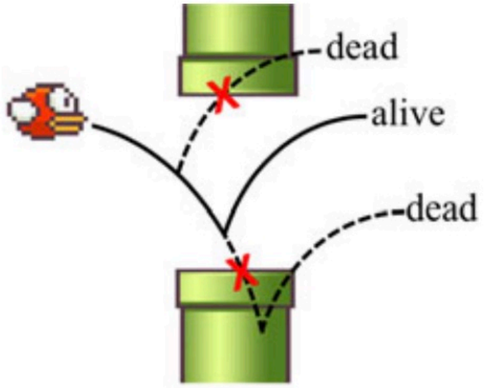


Automation to provide design feedback

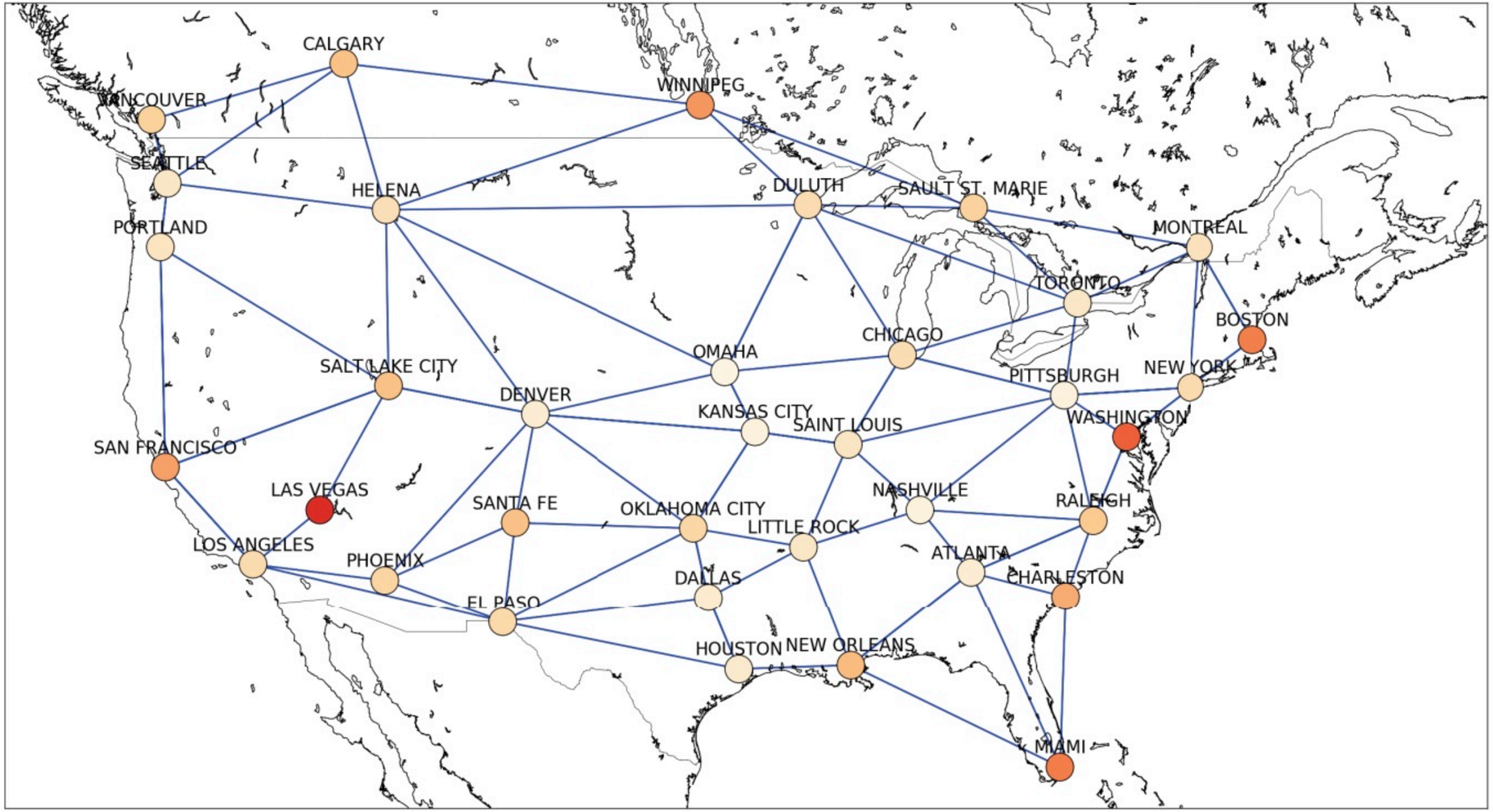








Automation to provide design feedback



Modern AI “bots”

Large-scale RL

4. Mass Scale Training Infrastructure

The DART platform had access to over 1,000 PlayStation 4 (PS4) consoles. Each was used to collect data for training GT Sophy or evaluate a trained version. The platform consisted of the necessary computing components (GPUs, CPUs) to interact with a large number of PS4s and support large scale training over an extended period of time.

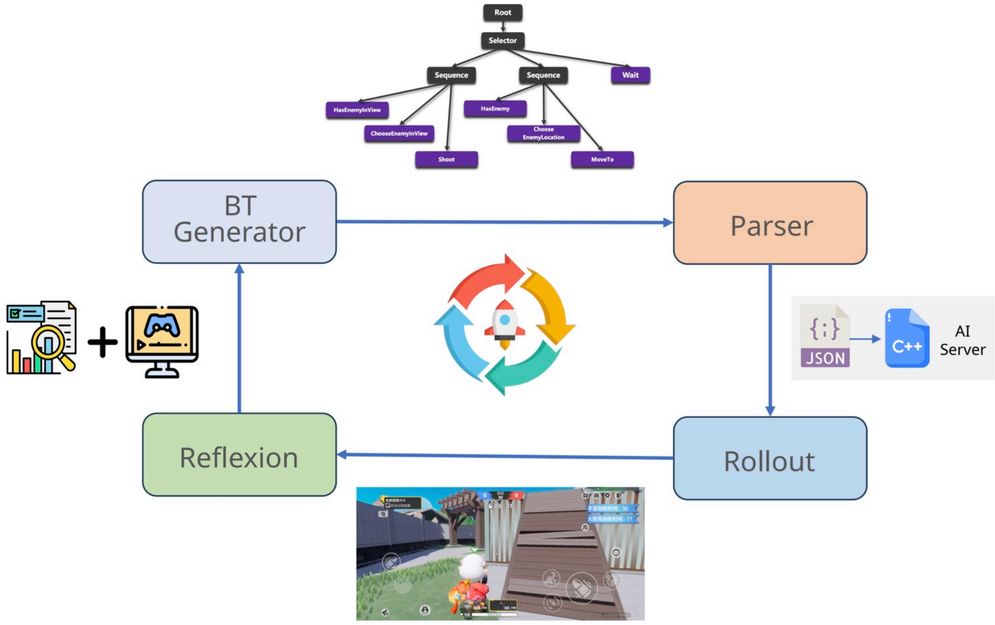
for over 1500 epochs[5] from scratch, which takes approximately ten days to train on a single A100 GPU and a dozen Playstation 4 systems. For our experiments, we reduced the training time of each iteration to 300 epochs, and only train the policy from the final iteration for 1500 epochs without a secondary replay buffer (to compare with GT Sophy) for evaluation. From now on, we refer to

Modern AI “bots”

VLM directly plays game, learns from it’s own experience

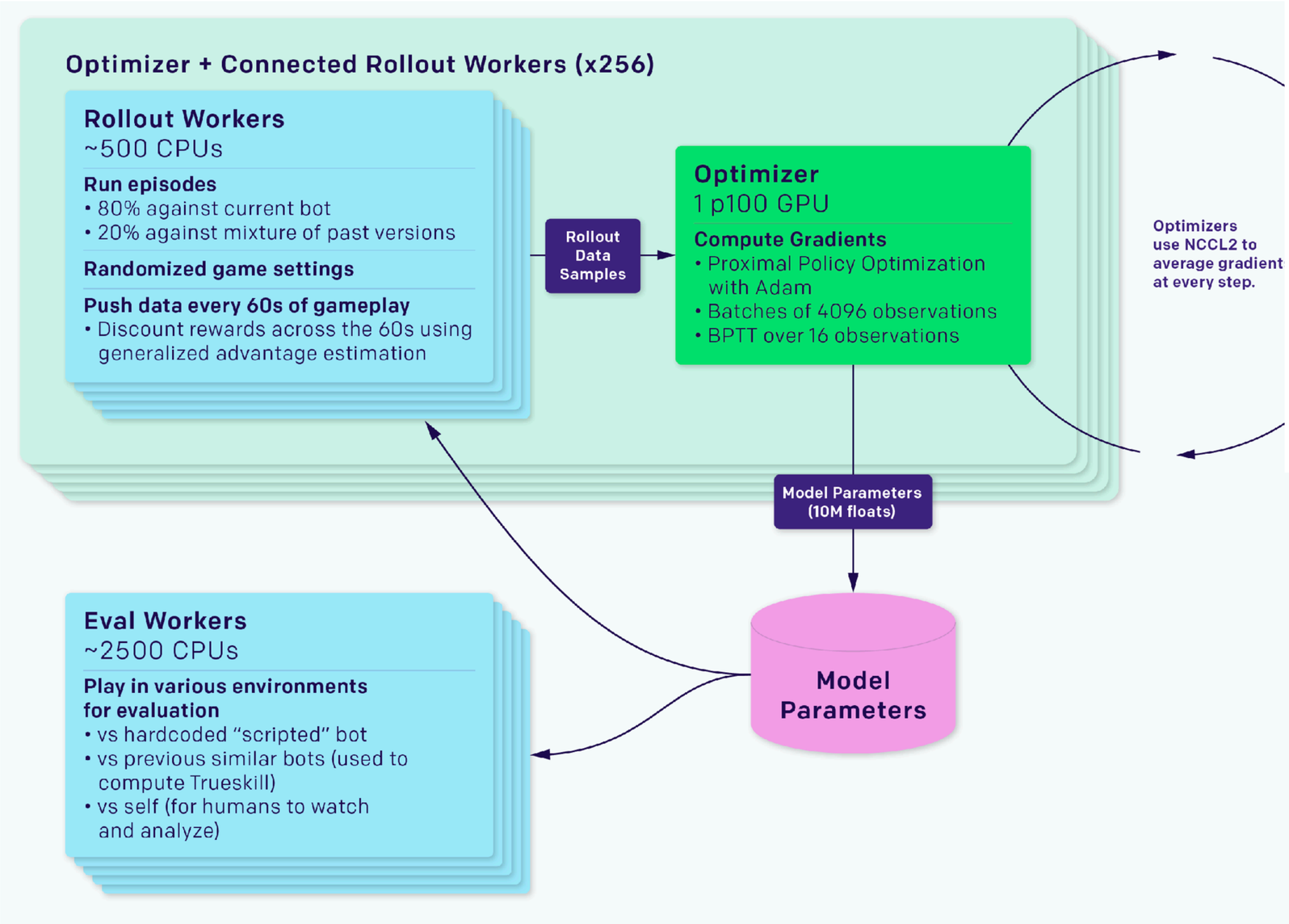
Modern AI “bots”

VLM authors a behavior tree that defines bot’s behavior



Top bots in 2026...

Experience collection for Dota 2 bots

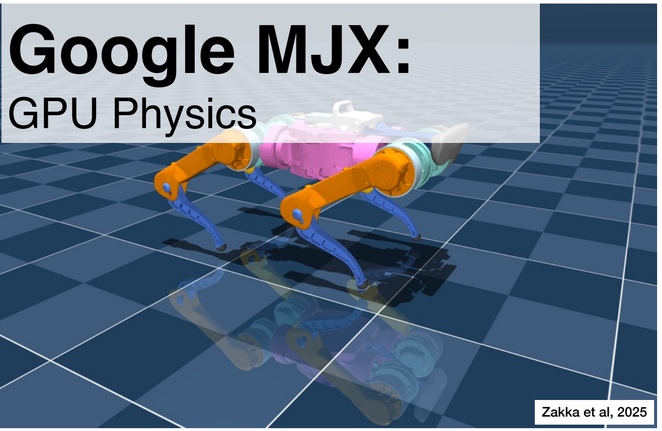


OpenAI Five stats (Dota 2)

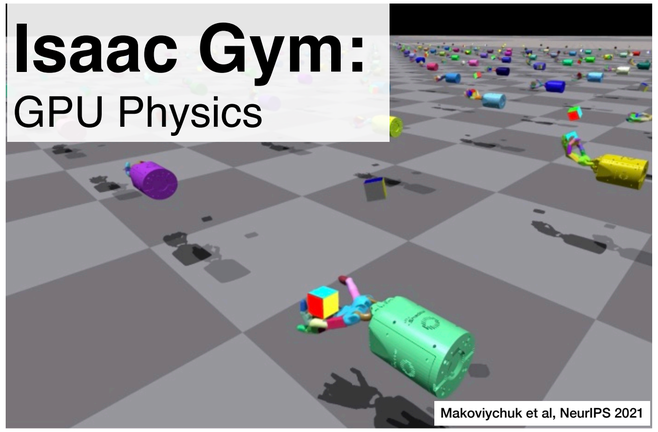
	OPENAI FIVE
CPUs	128,000 preemptible CPU cores on GCP
GPUs	256 P100 GPUs on GCP
Experience collected	~180 years per day (~900 years per day counting each hero separately)
Size of observation	~36.8 kB
Observations per second of gameplay	7.5
Batch size	1,048,576 observations
Batches per minute	~60

2023-2025 “batch simulators” that achieve millions of steps/sec by executing thousands of environments in parallel on a single GPU

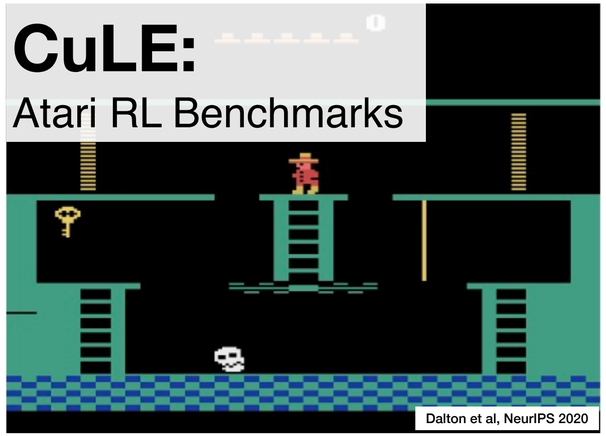
Google MJX:
GPU Physics



Isaac Gym:
GPU Physics



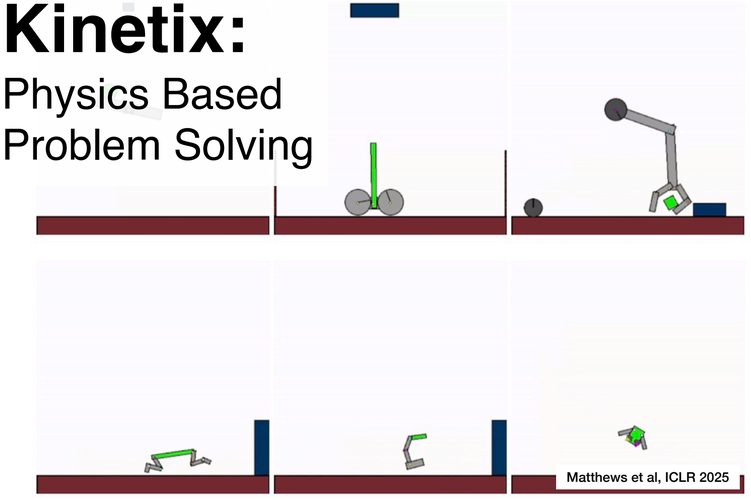
CuLE:
Atari RL Benchmarks



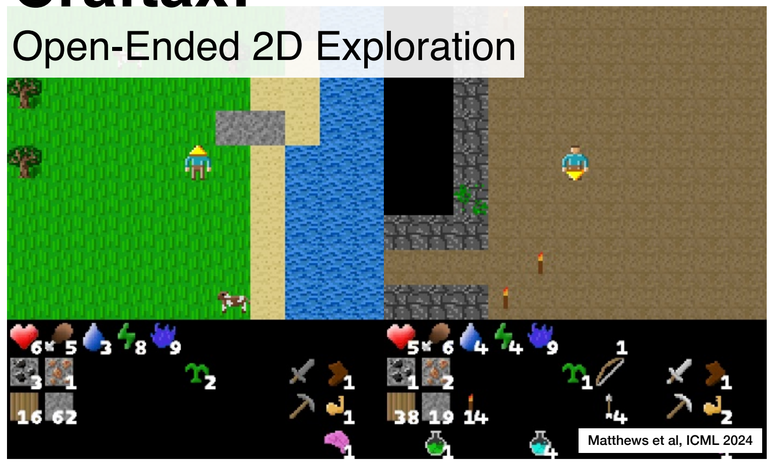
BPS3D:
Home Navigation

Maniskill:
Robotic Manipulation

Kinetix:
Physics Based
Problem Solving



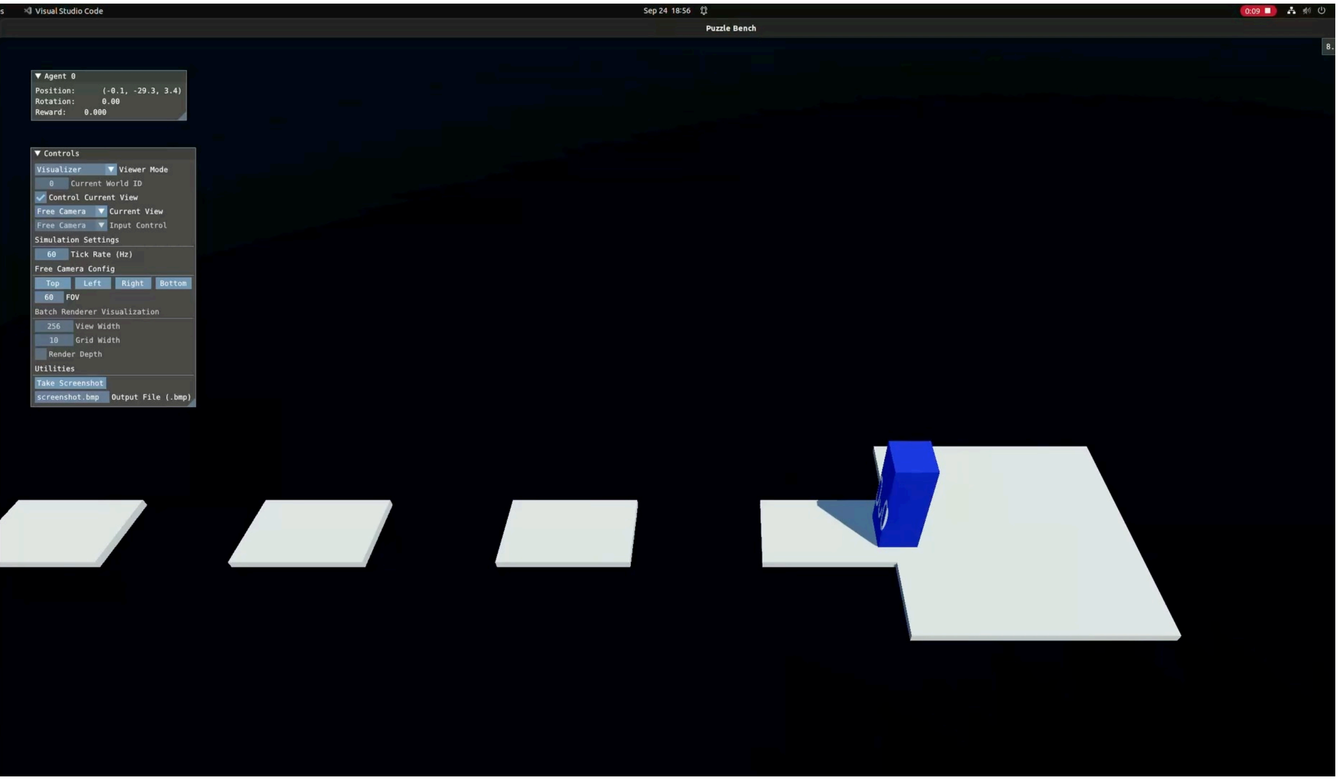
Craftax:
Open-Ended 2D Exploration



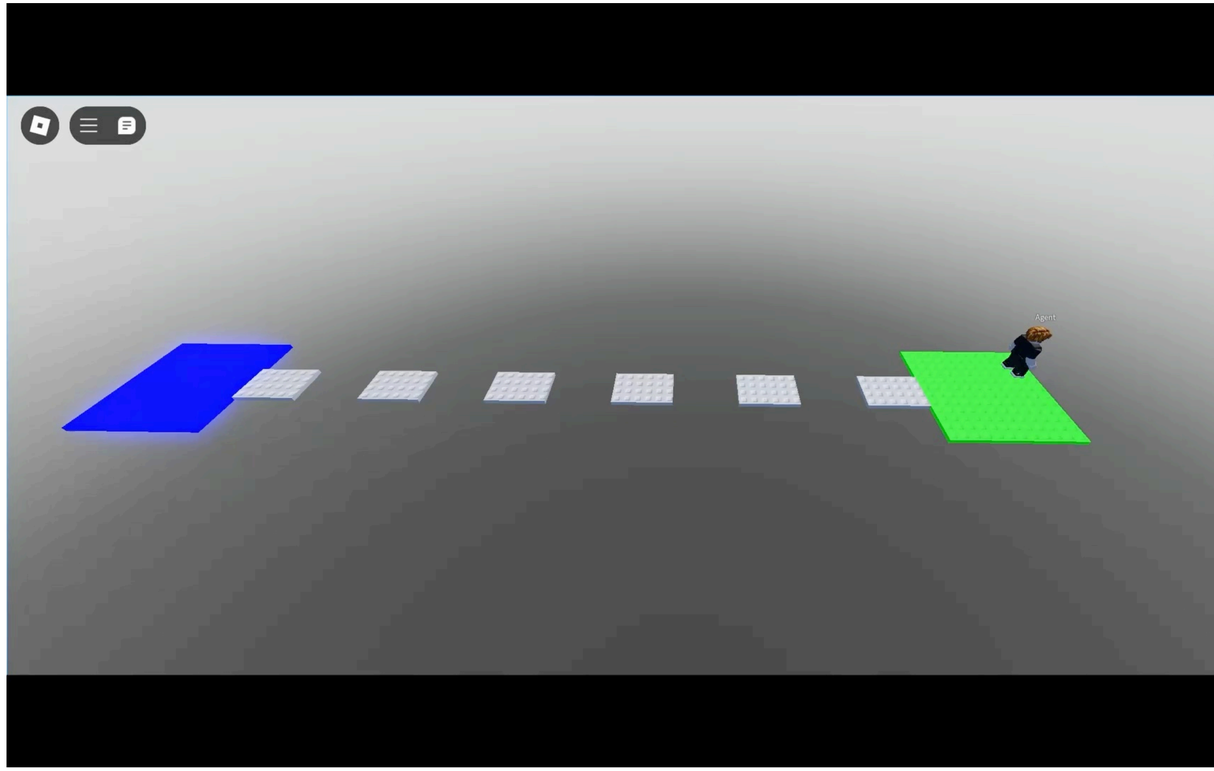
But… requires a simulator rewrite for the GPU

Implement simple proxies of real games to train agents: then sim-to-sim transfer

1M steps/sec simulation in low-fi simulator



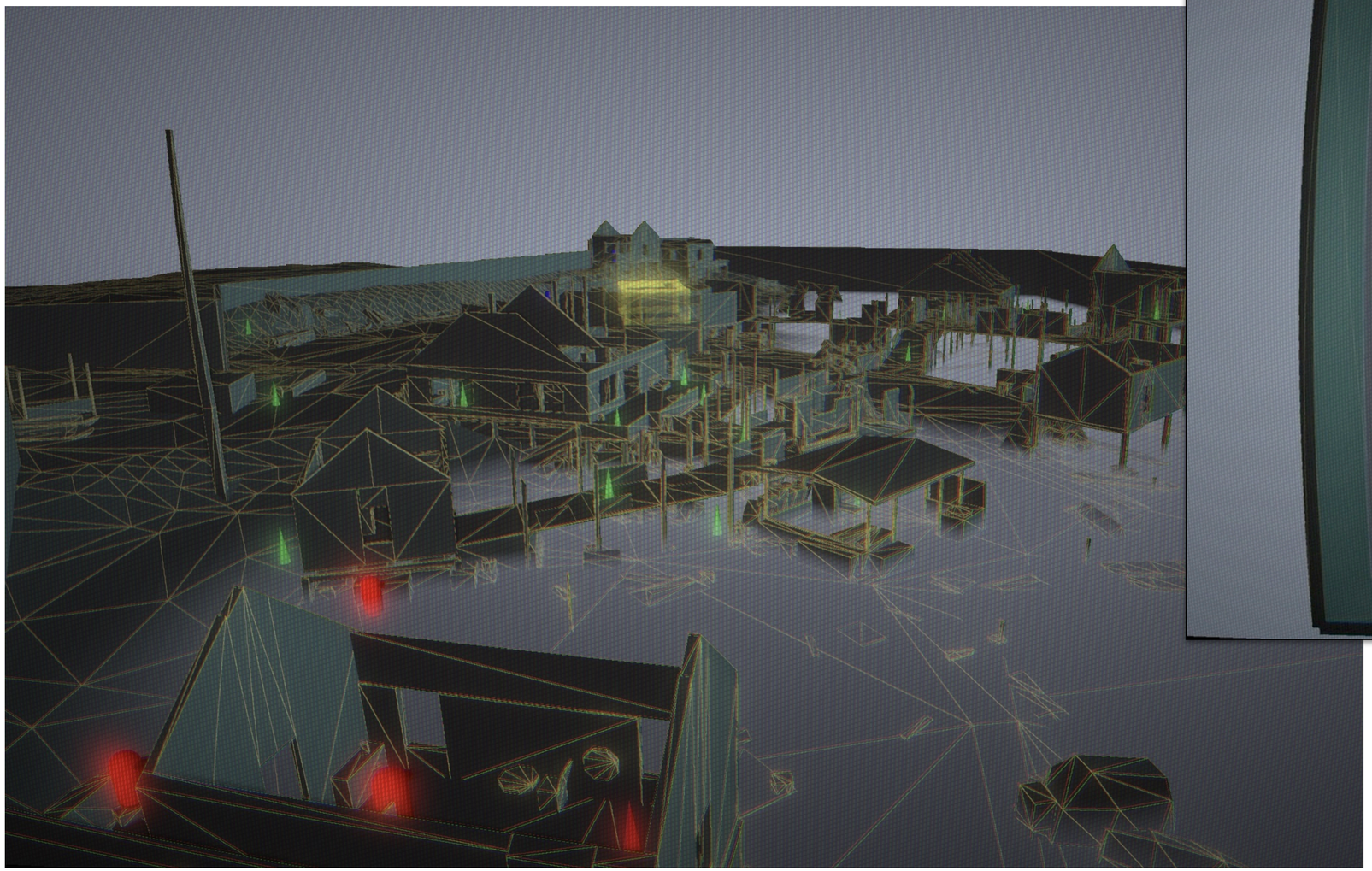
Same policy running in 60Hz Roblox engine

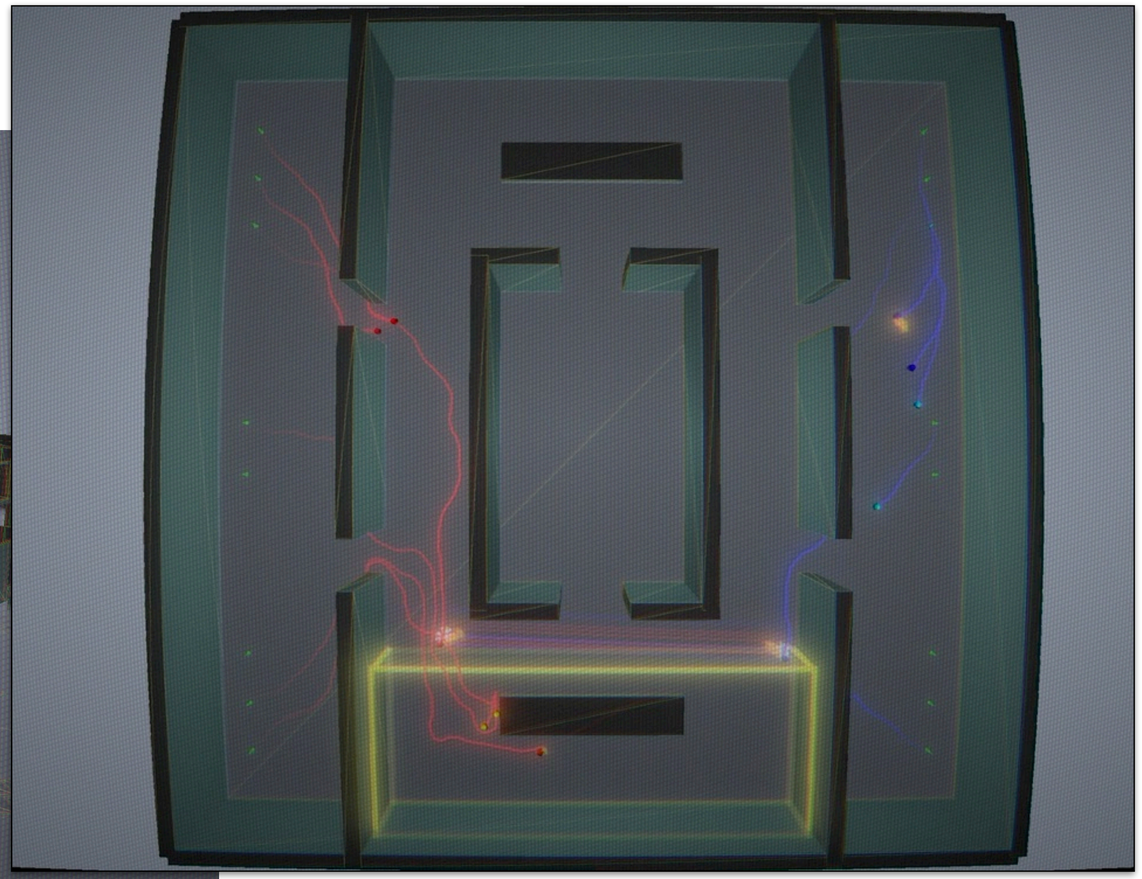


Sim-to-Sim: policies trained in coarse sim in a day can transfer directly to unseen obstacles in a higher-fidelity simulator (Roblox)

We tried learning competitive strategies for 6 vs 6 FPS play

6 x 6 FPS simulator (collaboration with Activision)

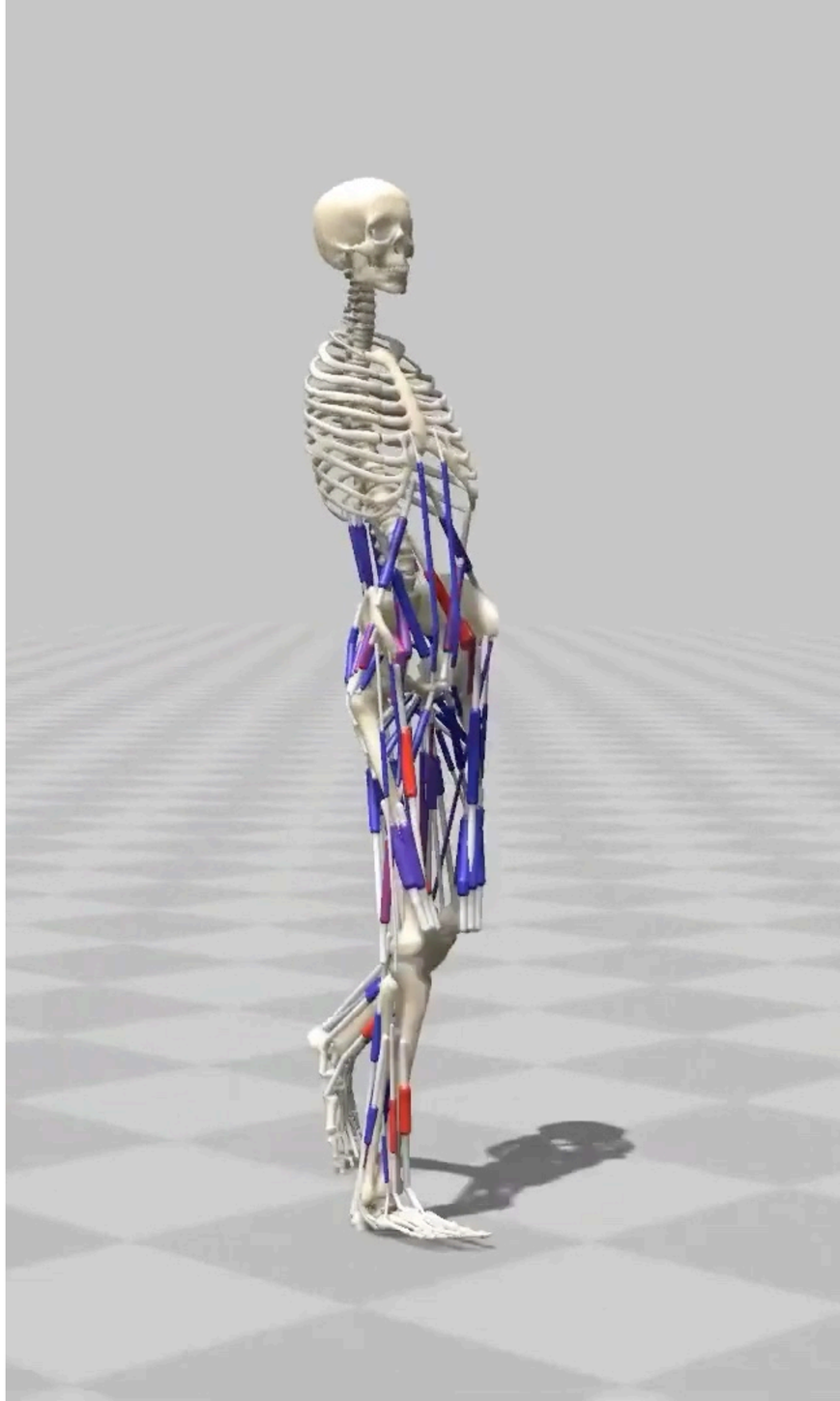




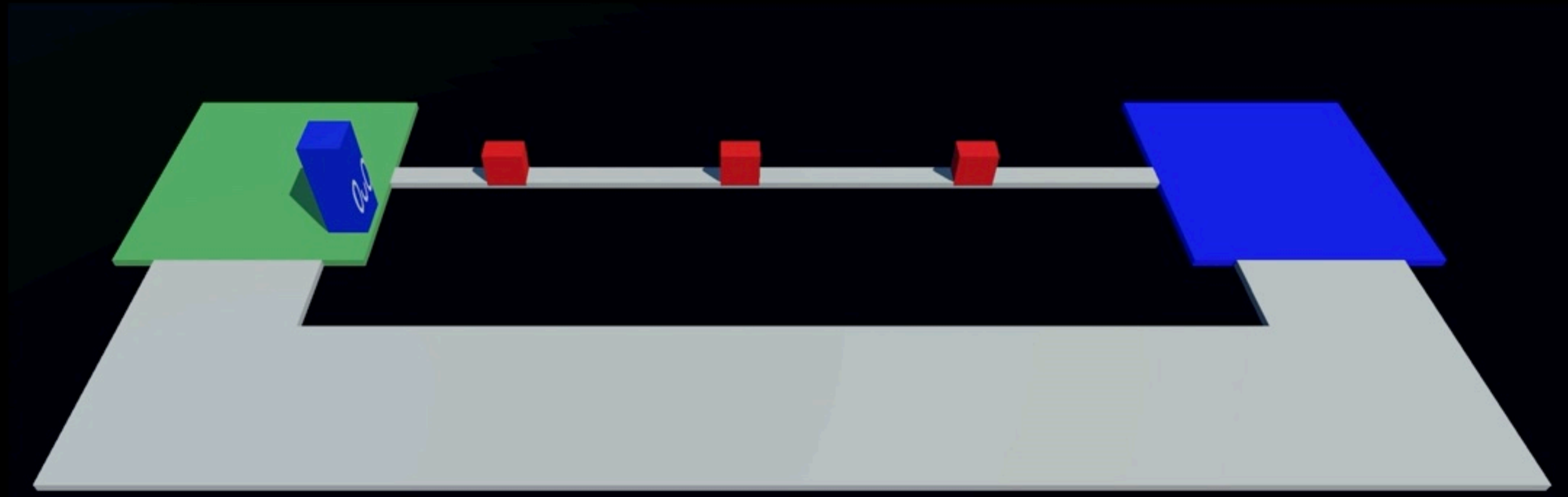
Leveraging system efficiency:
Use more accurate models of players

Analogy:

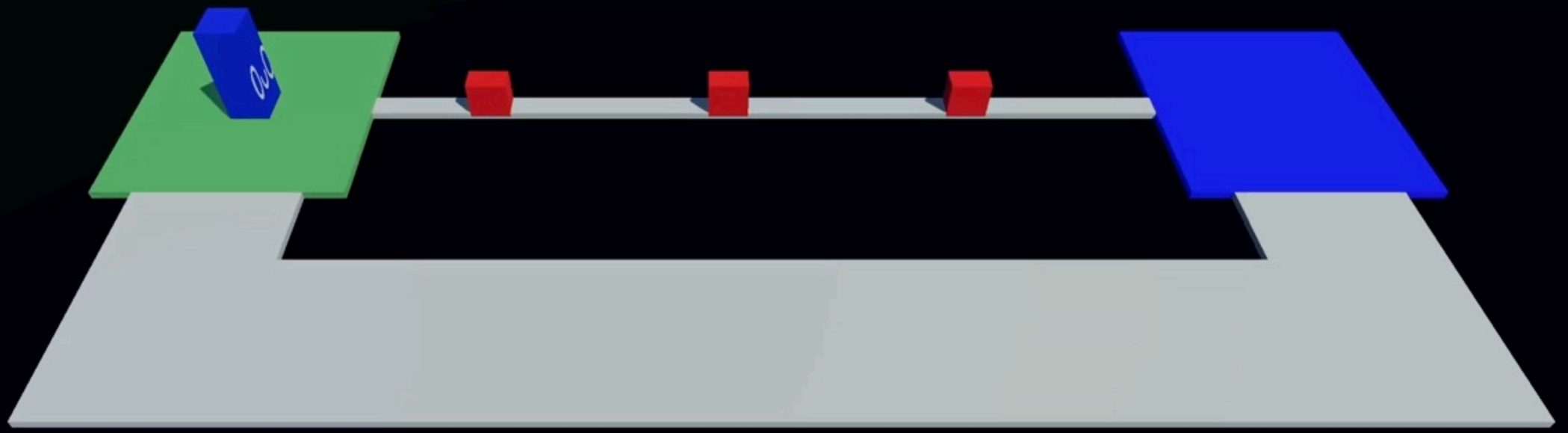
In physically-based animation, biomechanical modeling of muscles and tendons leads motion optimization toward more realistic gaits



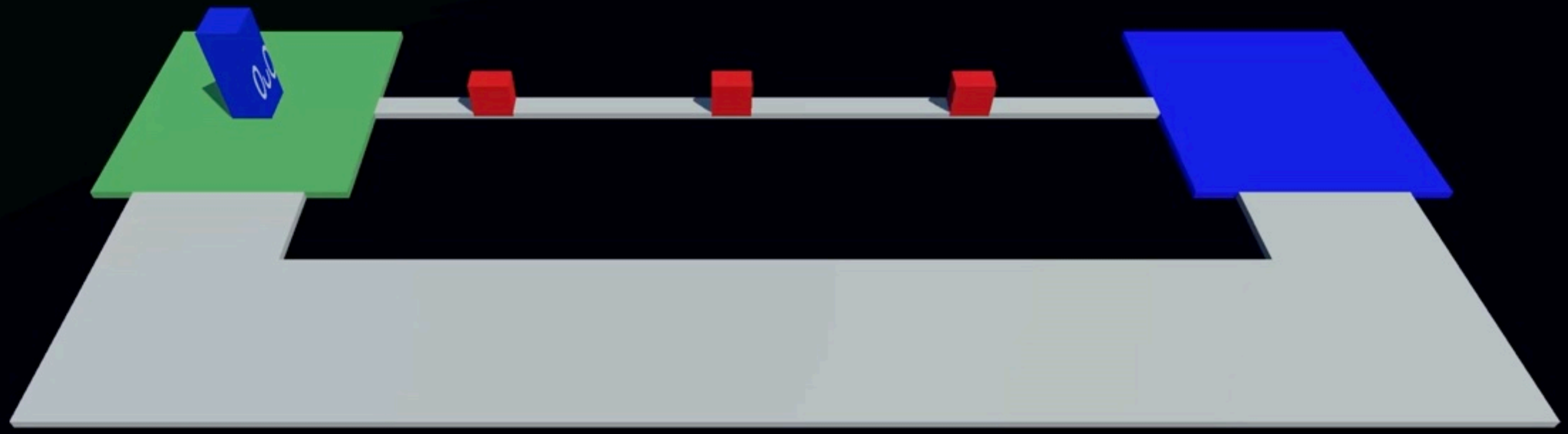
Given enough training, an agent can learn to successfully navigate the skinny path, even though that path affords a low margin for movement error.



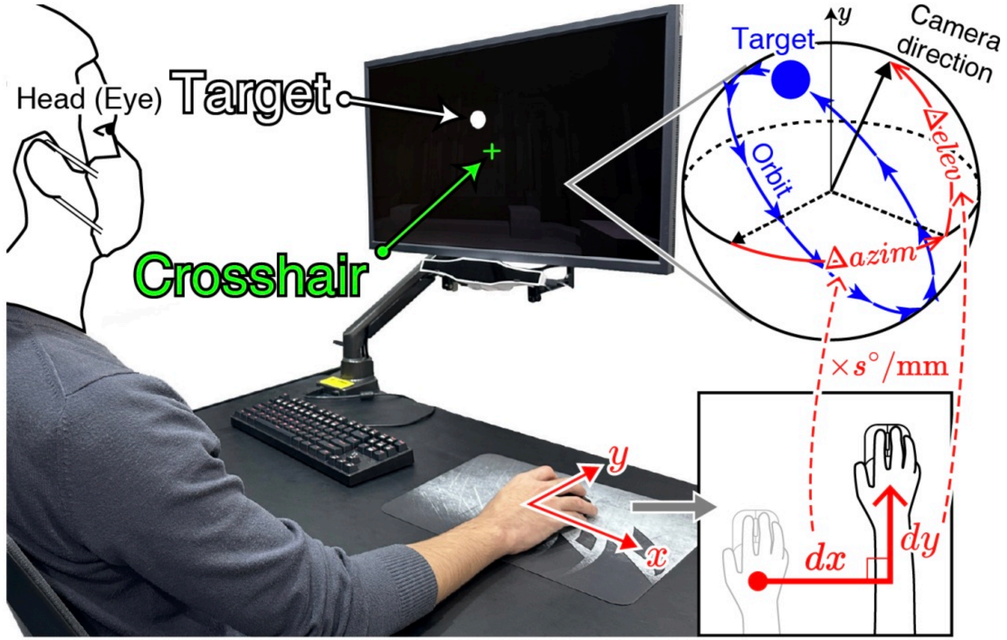
Randomly changing the agent's agents (with probability *p*) reduces success rate to near 0.

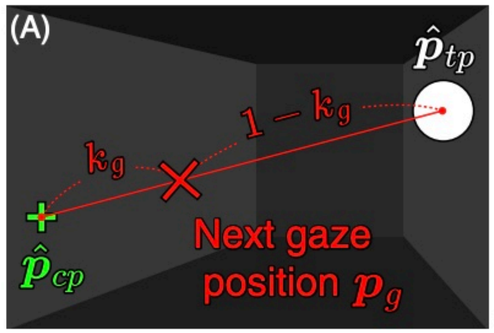


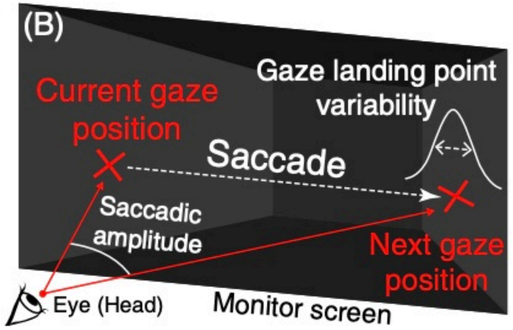
Training an agent that is “aware of” its own limitations (potential for occasional action perturbation) yields a policy that wisely takes a more conservative approach.

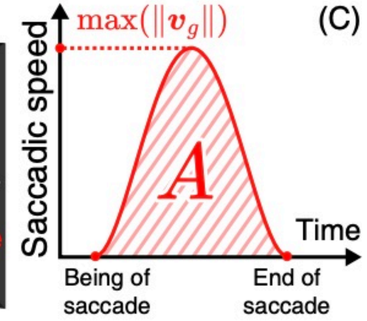


Measuring/modeling human behavior in aim-and-shoot video games



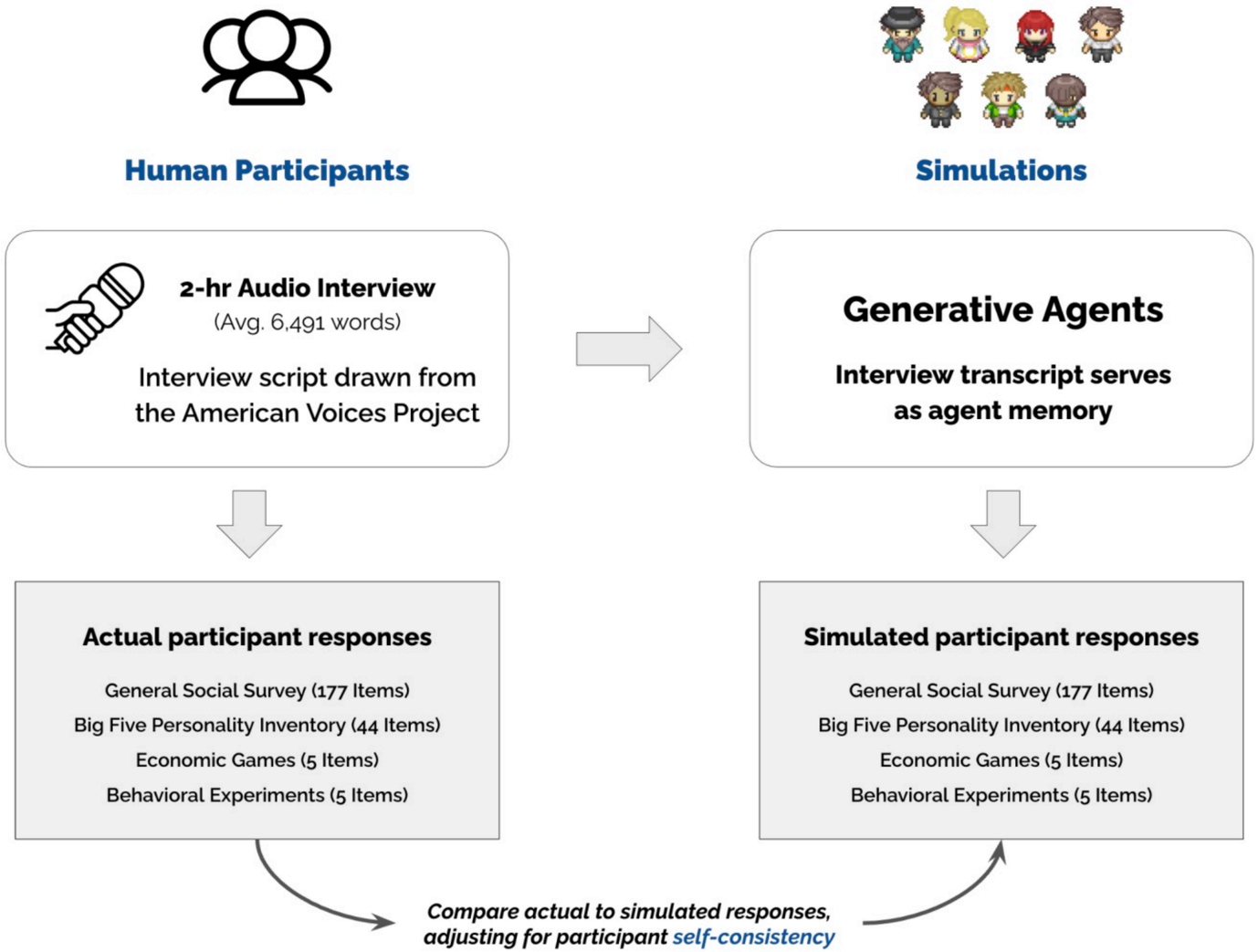






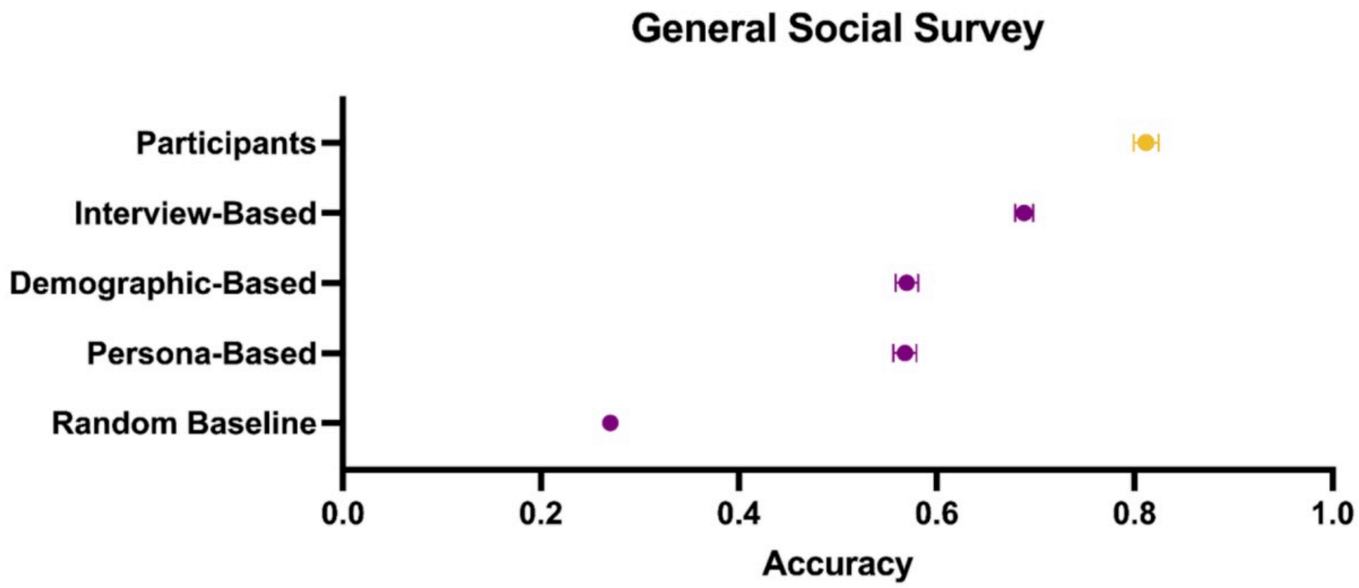
Modeling human behavior (in LLM agents)

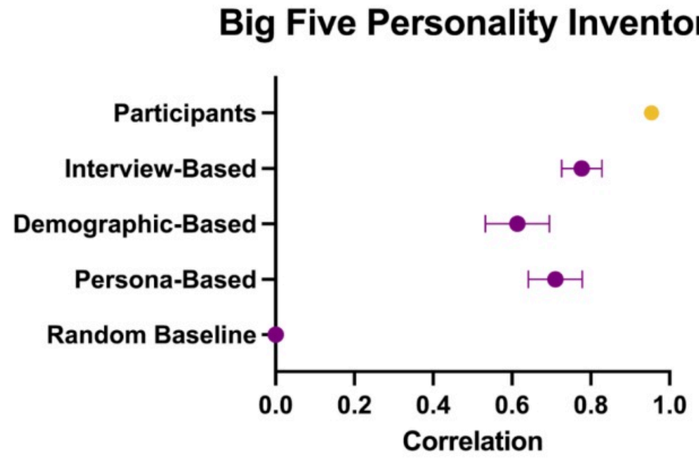
- Recent study (by the same authors as Generative agents) [Park et al. 2024]
- Interview 1000 people + ask those people a detailed set of questions about their personality, ask them to perform simple tasks, etc.
- Give transcript of interview to LLM agent… see how the result agent does on the same interview

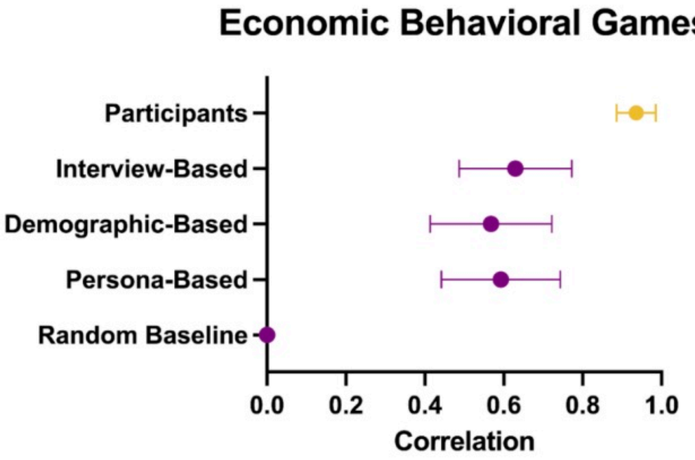


Similarity to human responses

- Generative agent built off of interview context responds to questions more like the interview than alternative ways to model a persona

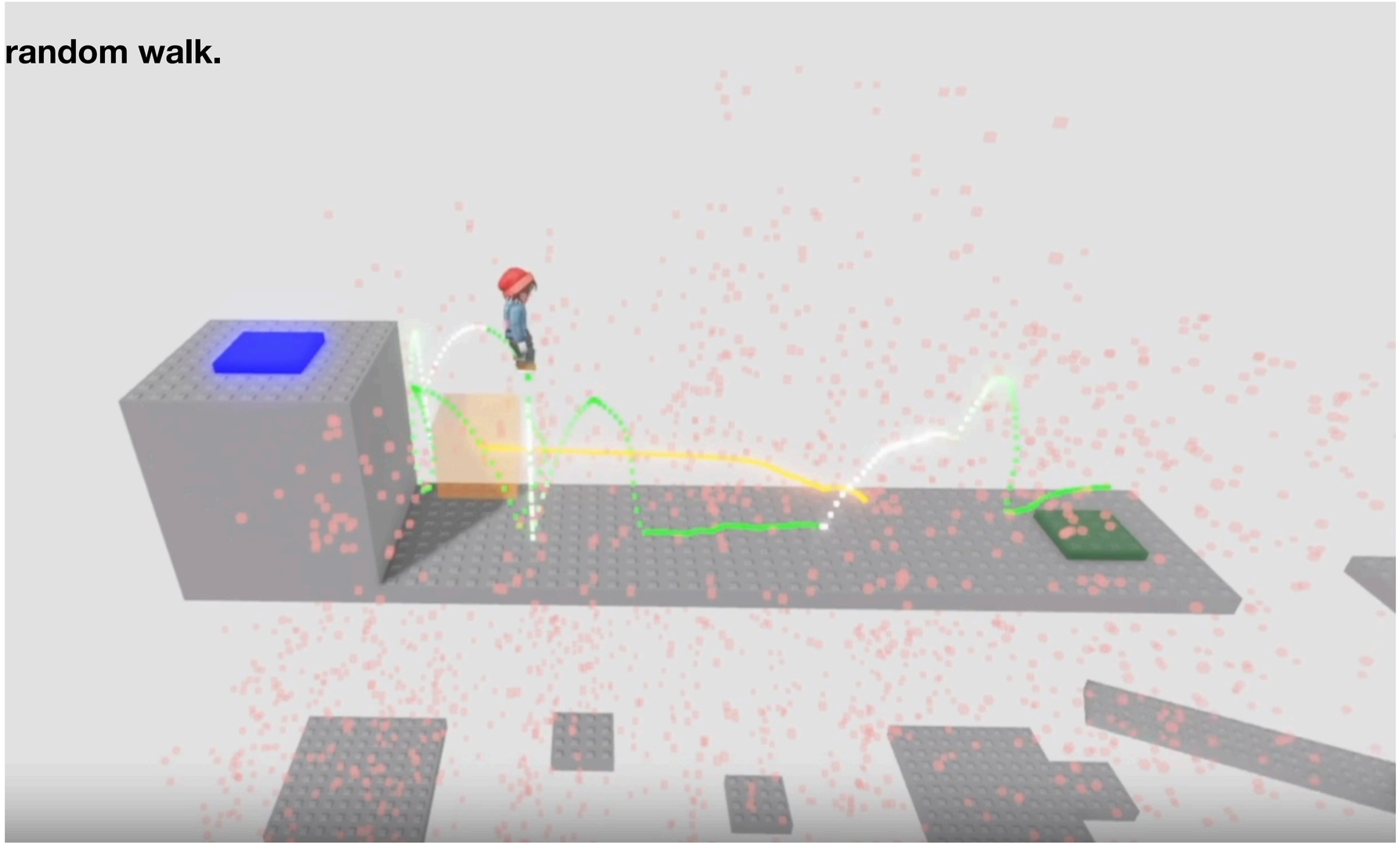






But it's not necessary true accurate behavior is needed

View results of a random walk.

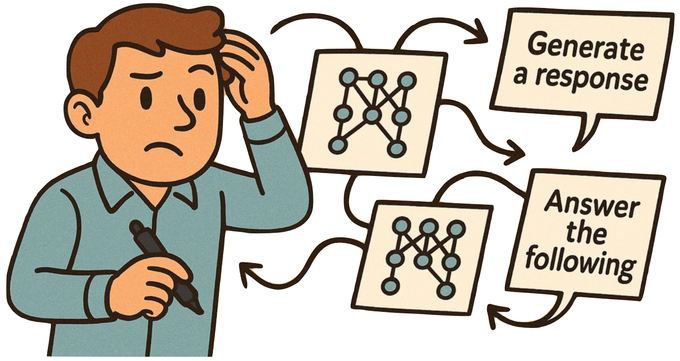


Improving LLM-agents in an afternoon without costly fine-tuning

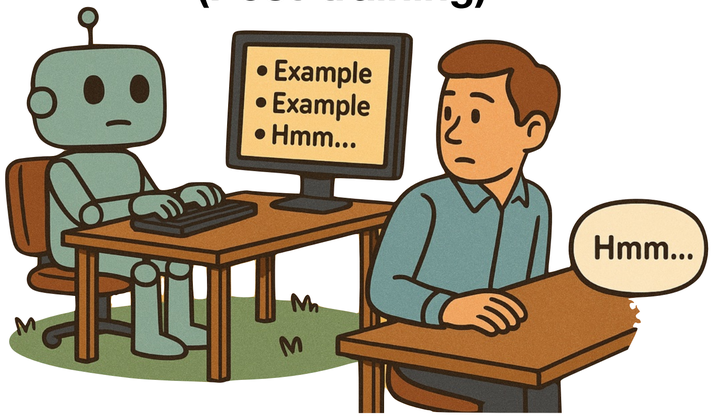
If an LLM agent can’t solve a task, what can we do?

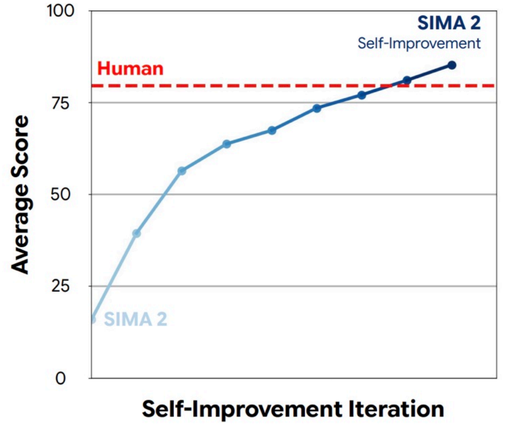
Use bigger off-the-shelf model?
Test-time scaling (take multiple tries)?
generate more thinking tokens?

Prompt engineer?
Provide better instructions?
High human cost.
Brittle. Annoying!



Fine-tune the model by learning from experience. (Post-training)





A very simple agent

```
def agent(task_goal, db, game):
    plan = make_plan(task_goal)

    while (!game.done)
        obs = game.getObservation()
        action = choose_action(task_goal, plan, obs)
        game.step(action)
```

TextWorld

```
Welcome!

You are in the middle of the room.
Looking around you, you see
a diningtable, a stove,
a microwave, and a cabinet.

Your task is to:
Put a pan on the diningtable.

> goto the cabinet

You arrive at the cabinet.
The cabinet is closed.

> open the cabinet

The cabinet is empty.
```

A very simple agent with a DB of prior experience

```
def agent(task_goal, db, game):
  plan = make_plan(task_goal, retrieve_for_planning(db, task_goal) )

   while (!game.done)
      obs = game.getObservation()
      action = choose_action(task_goal, plan, obs, retrieve_for_acting(db, obs) )
      game.step(action)
```

So what data do we put in the database?

- We have an agent start solving tasks in a game, and **if the task is solved, we store the trajectory of (actions, observations) in the DB, growing it by one—that's it!**

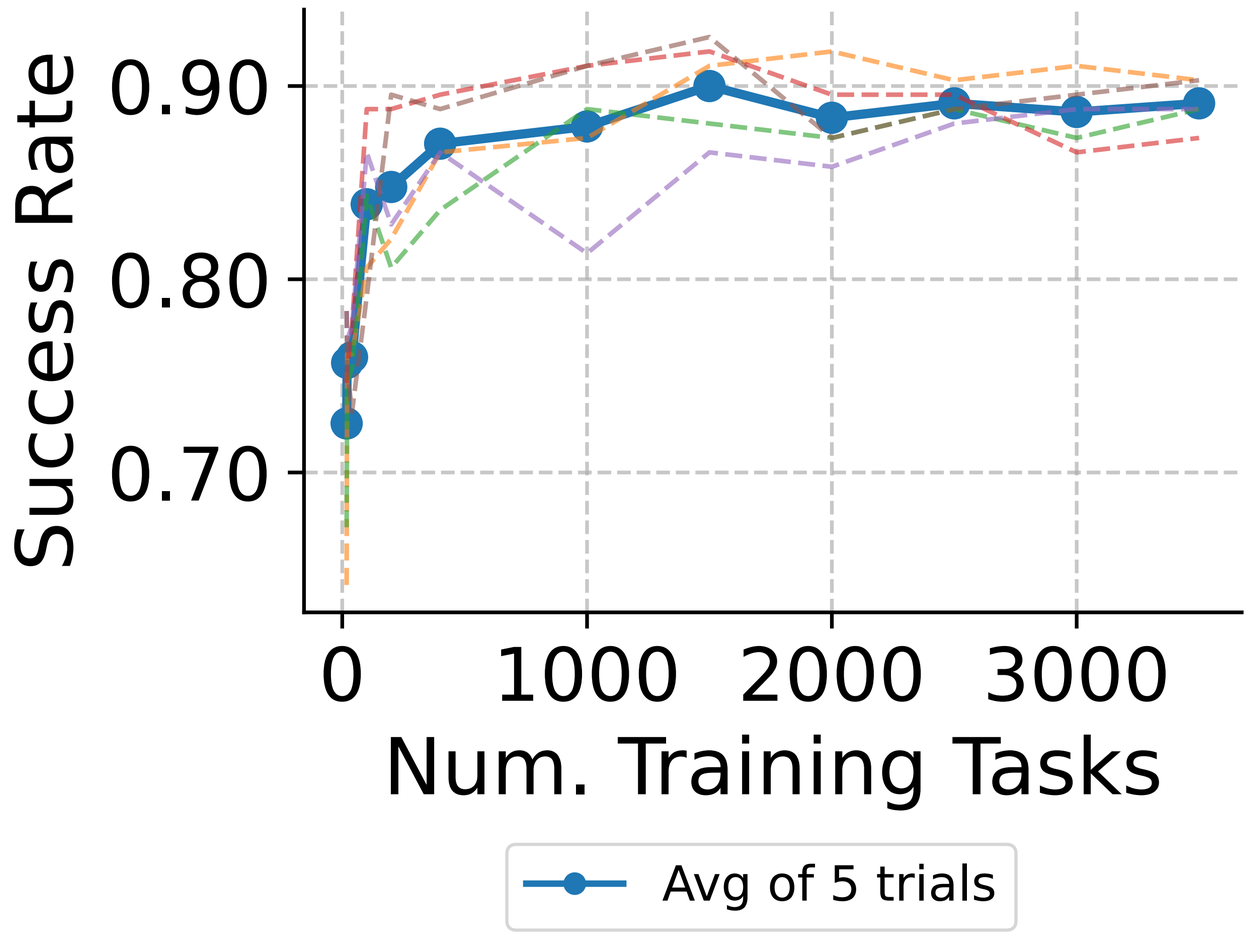
- The hope:

  1. Successful behavior on one game task is helpful to LLM when asked to do ***similar game tasks*** in the future.

  2. More complex future tasks involve composition of concepts from simpler tasks

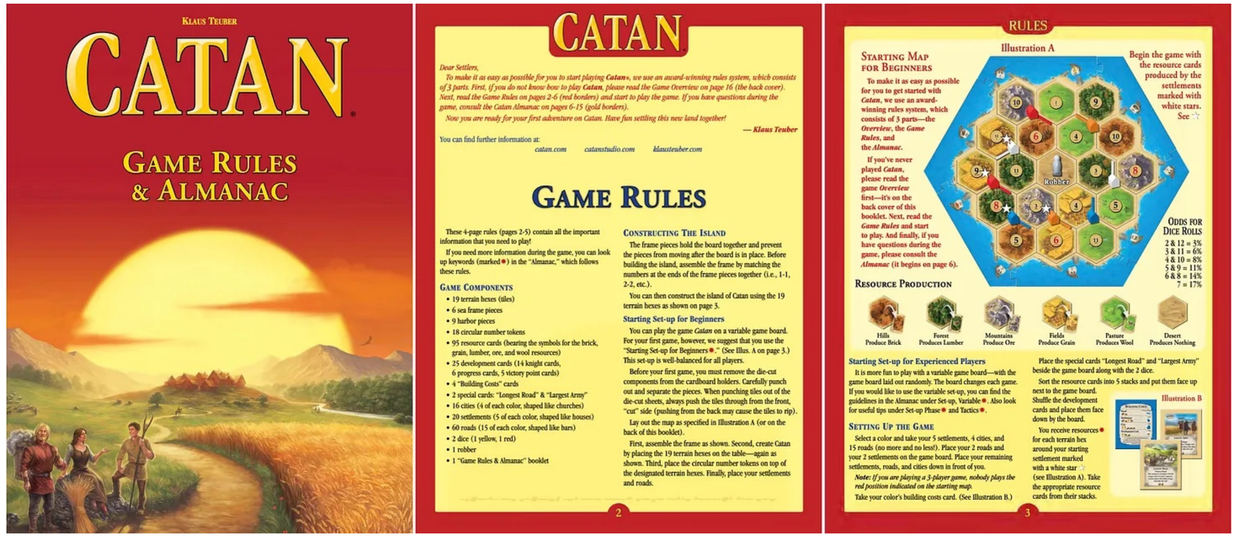
Even this simple approach gets big boosts!

73% to 89% success



- Learning from the agent's own examples can yield notable performance boosts
- Low run-time costs (only additional cost is DB retrieval)
- Immediate improvement (no waiting for fine-tuning the model)

Experiment: simultaneously implementing Catan and bots to play it!



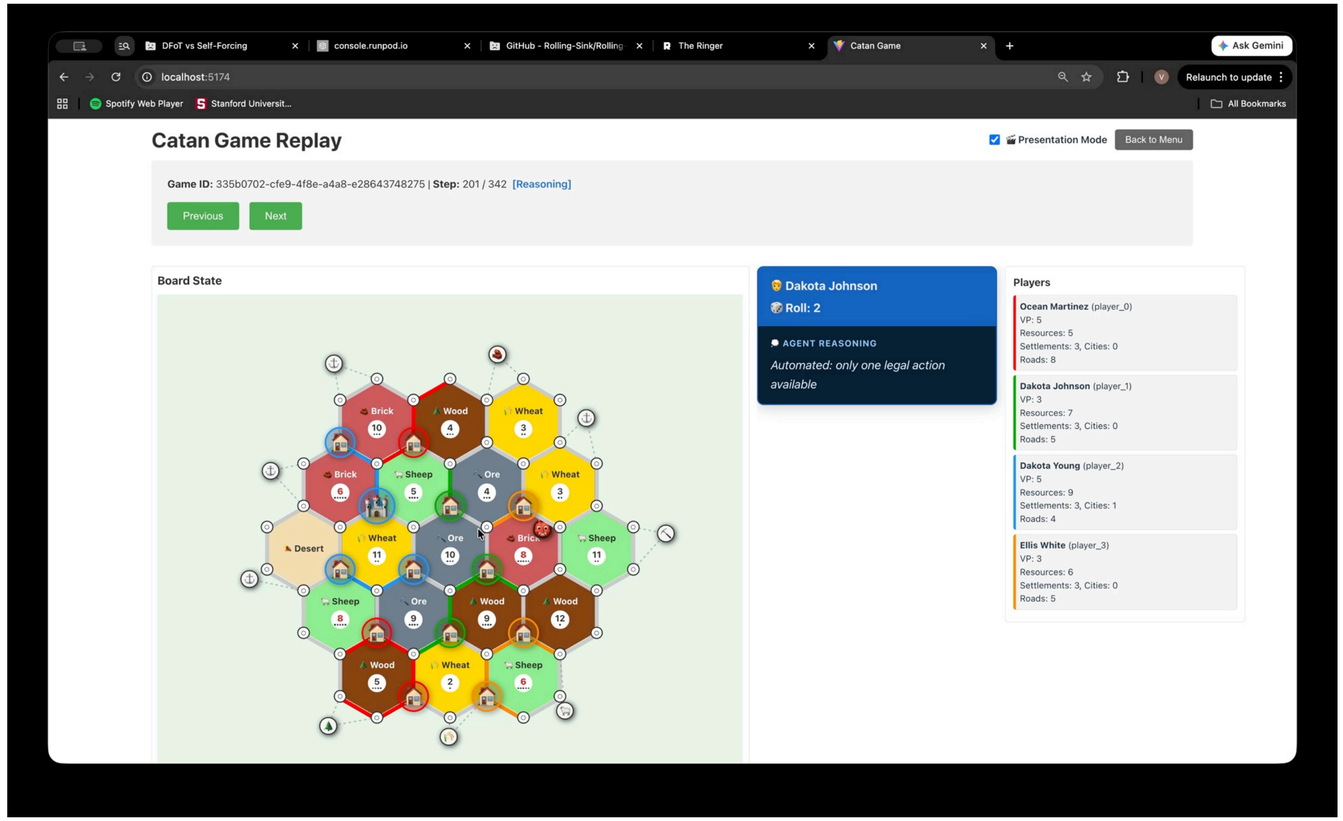
Phase 1: implementing Catan

- Gave an agent a rule book.
- Ask LLM to implement game and game tests given the rule book
    - Game implementation **can dump state** at any game step (ask LLM to write tests for given state), **this creates a growing set of regression tests.**
- Simple agents play game (fuzz test) to find errors in game (crashes/rule violations)
- Ask LLM to fix errors in game (give failing state and agent action)

Simultaneously implementing Catan and bots to play it!

Phase 2: improve agents

- Human watches agents play themselves
- If human sees a good move by the agent in a key scenario, add (state, action) to database
- If human sees a bad move in a key moment, add example to database with human-annotated solution
- Agents query from database of good moves, which grows over time — **agents improve!**
- Improved agents sample different parts of game space, further fuzz testing game correctness



Level-of-detail = reasoning at the right level of abstraction

How can we identify and generate a world simulation at the right level of abstraction for a given learning task?

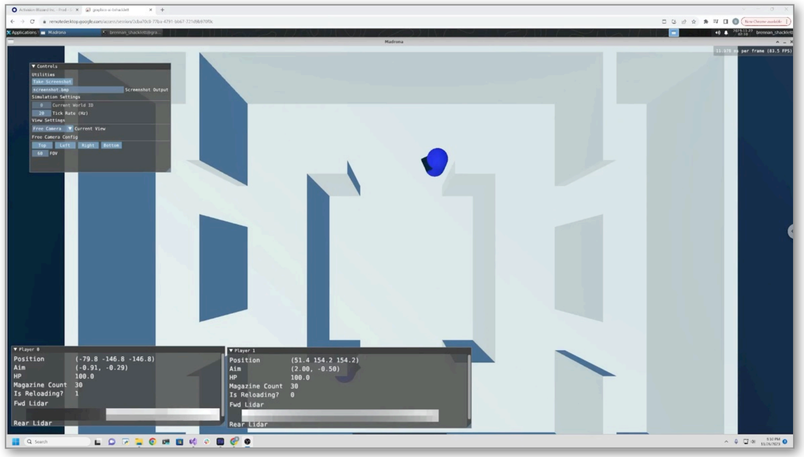
What aspects of problem solving should be done at each level of abstraction?

Text-based game

"You are a character in a battle Royale FPS game, you are standing near a corner and you see an enemy peak out…

LLM policy action:
"You should take cover"

Low-fidelity sim



Low-fi agent action:
Move to (x,y)

High-fidelity game

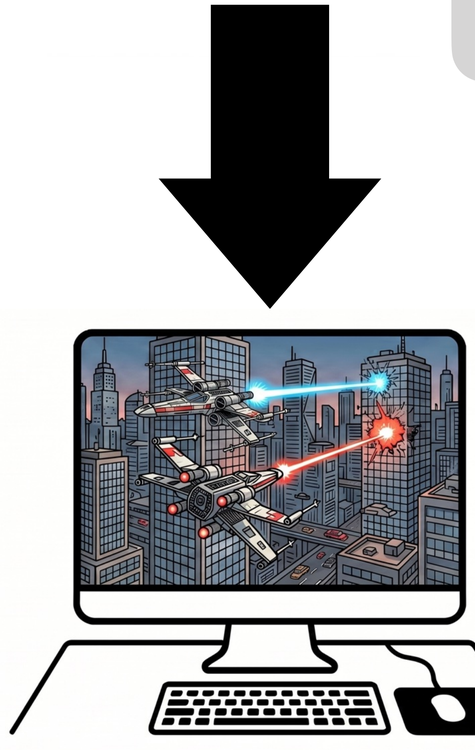
Game agent policy action:
Game controller input

Summary:

High velocity generation will be limited in impact without structure and tooling to manage, analyze, and control it

+ Claude

Good design/structure, Comprehensive evals, Key validations



Rapidly improving agent capabilities offer a vector for “keeping up” with the pace of 3D content generation.

Same questions face “regular” software face developers of interactive 3D worlds.