CIS 6280 · FALL 2026

WORLD MODELS

LECTURE 19

LLMs as World Models?

Jiatao Gu

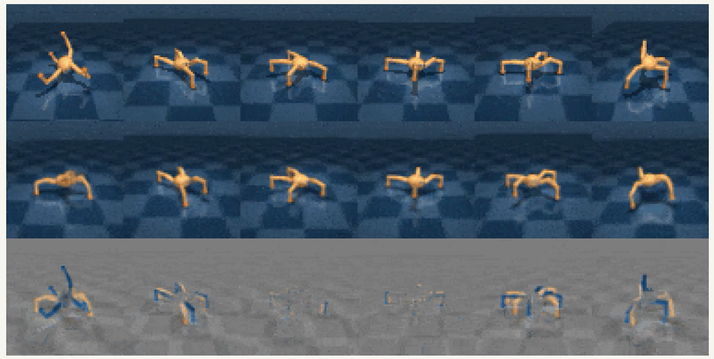
University of Pennsylvania · November 19, 2026

Announcements

Nov 24	Invited lecture by Randall.
Final project	Report and code due December 14.
Cloud compute	$100 per team. More details will follow.

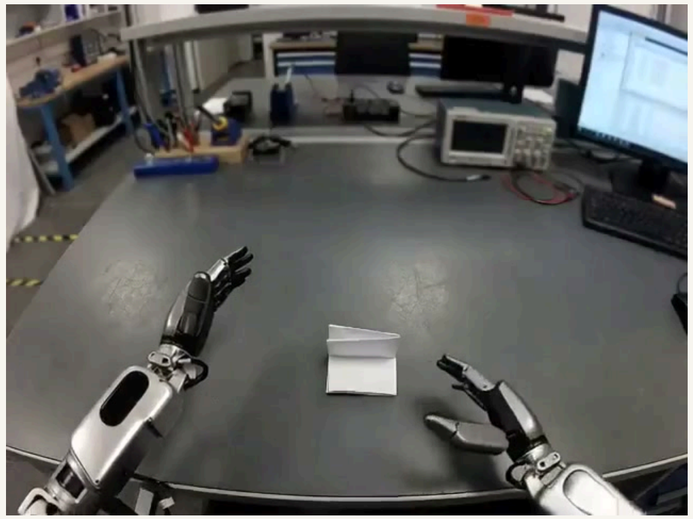
How have we represented the world?

Latent variables



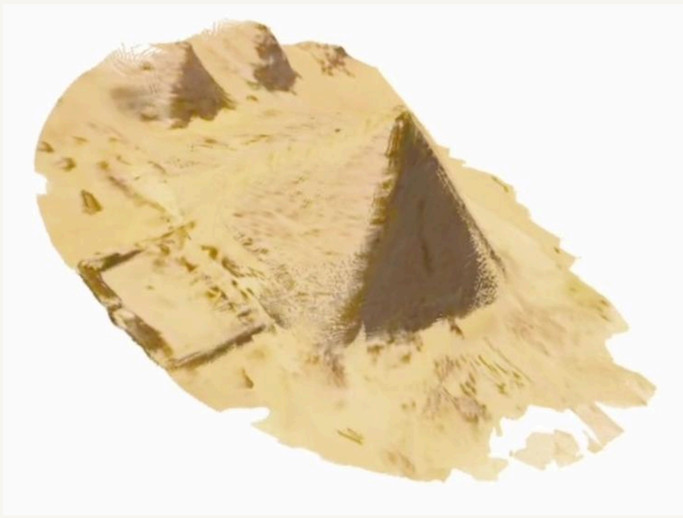
Predict learned features; decode when needed.

Video



Predict observations over time.

3D structure

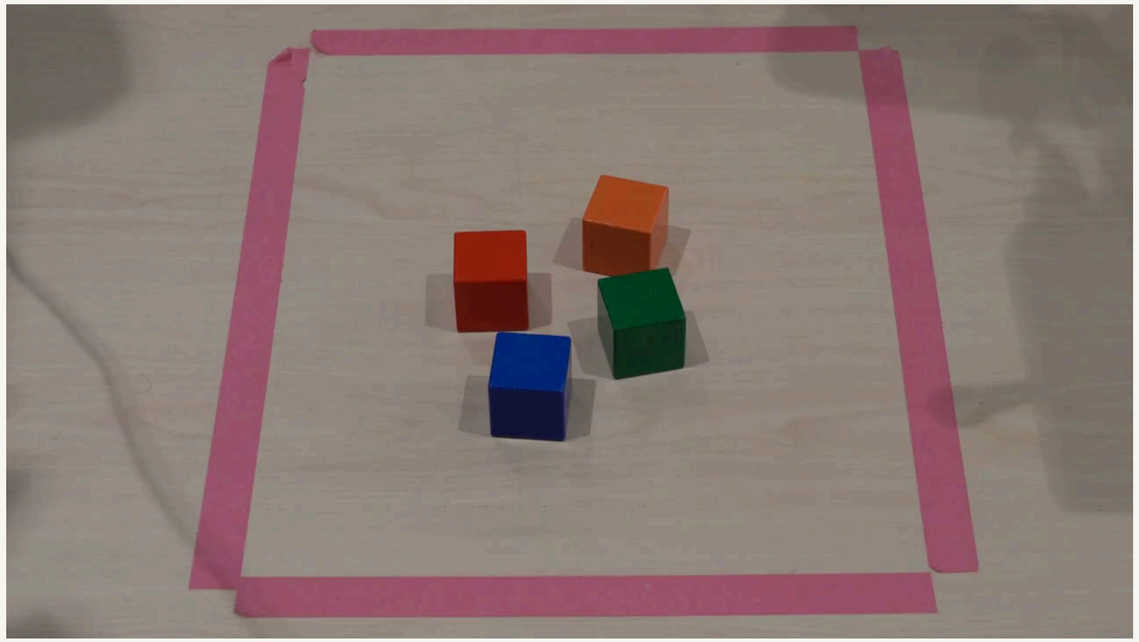


Represent geometry and spatial relations.

Reasoning can use these representations to predict consequences and choose what to do.

Hafner et al. · Dreamer · ICLR 2020 · NVIDIA · DreamDojo · Wang et al. · VGGT · CVPR 2025

Can language describe how a world changes?



A scene from Code as Policies. The action below is a teaching example.

Current situation

The red and blue blocks are on the table.

Candidate action

Place the red block on the blue block.

Predicted change

The red block is supported by the blue block.

The same event can be described through objects, actions, and relations.

An LLM generates one token at a time

Context Task, observations, instructions, and earlier generated text.

The red block is **on** the blue block.

$$p_\theta(y \mid x) = \prod_k p_\theta(y_k \mid x, y_{<k})$$

x: context · y_k: next text or code token · θ: learned model parameters

Each generated token becomes context for the next one.

Brown et al. · Language Models are Few-Shot Learners · 2020

Learning from sequences and demonstrations

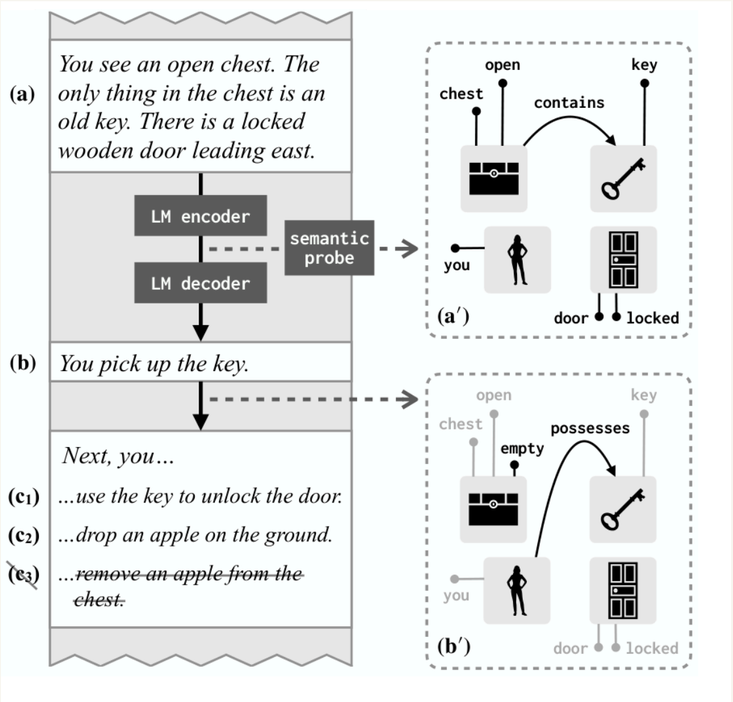
Training data	What the model predicts
Pretraining: broad text and code	The next token in a recorded sequence.
SFT: instruction and demonstration	The next token of the desired response.

Demonstration	Context: red is on the table. Task: stack red on blue.
Target response	Locate red and blue, then call pick_place(red, blue).

Training updates θ. A new prompt changes the context supplied to θ.

Brown et al. · Language Models are Few-Shot Learners · 2020 · Ouyang et al. · InstructGPT · 2022

Tracking a state described in language



Li, Nye & Andreas · entity-state representations as a discourse advances.

Before

A key is inside an open chest.

Event

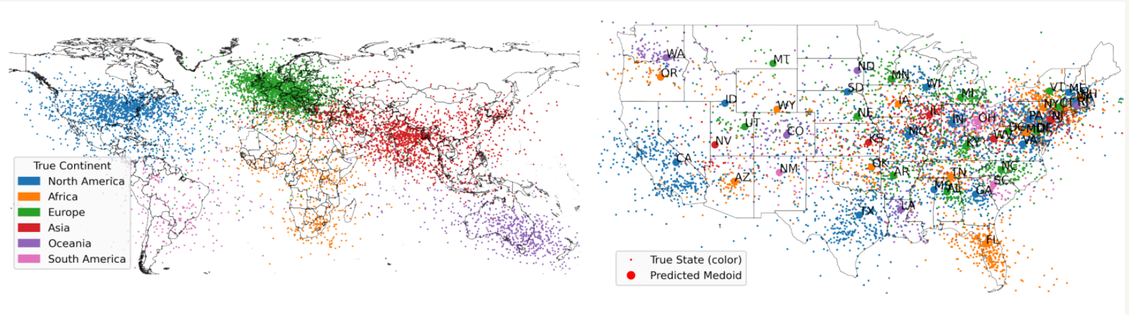
You pick up the key.

After

You hold the key. The chest is empty.

The study tests whether entity properties can be read from BART/T5 representations.

Spatial knowledge and transition prediction



Locations decoded from Llama-2 representations · source figure crop.

Where is this place?

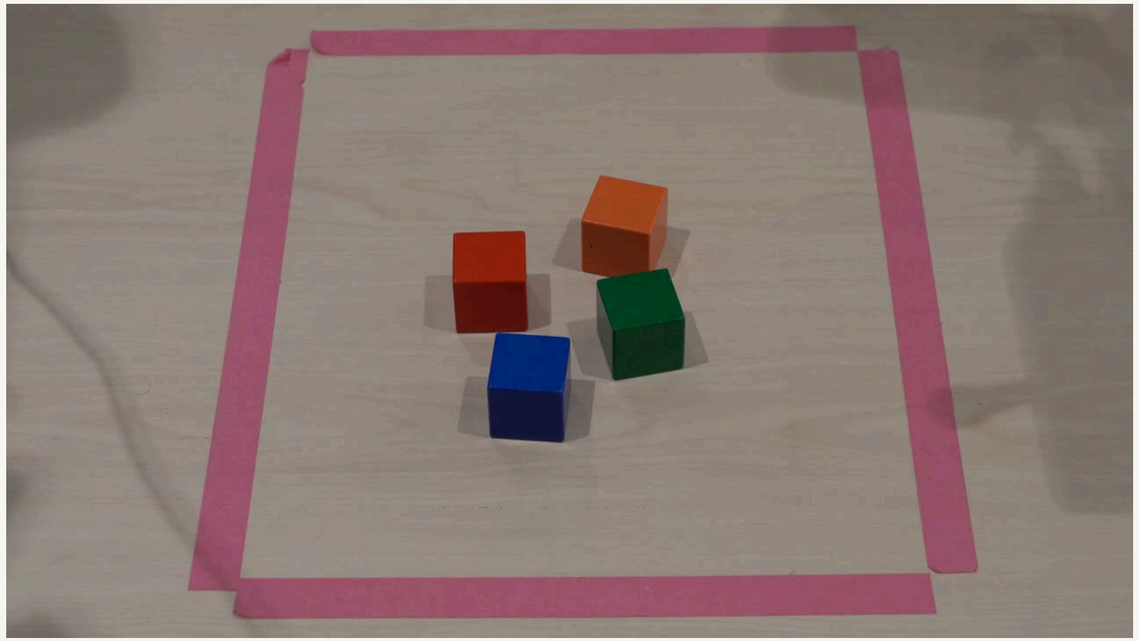
A probe can read out spatial information.

What happens if I act here?

This requires a separate prediction test.

Gurnee & Tegmark · ICLR 2024 · Fig. 1

Language can specify an action



Perception and control tools connect a program to the robot.

Instruction

Place the red block on the blue block.

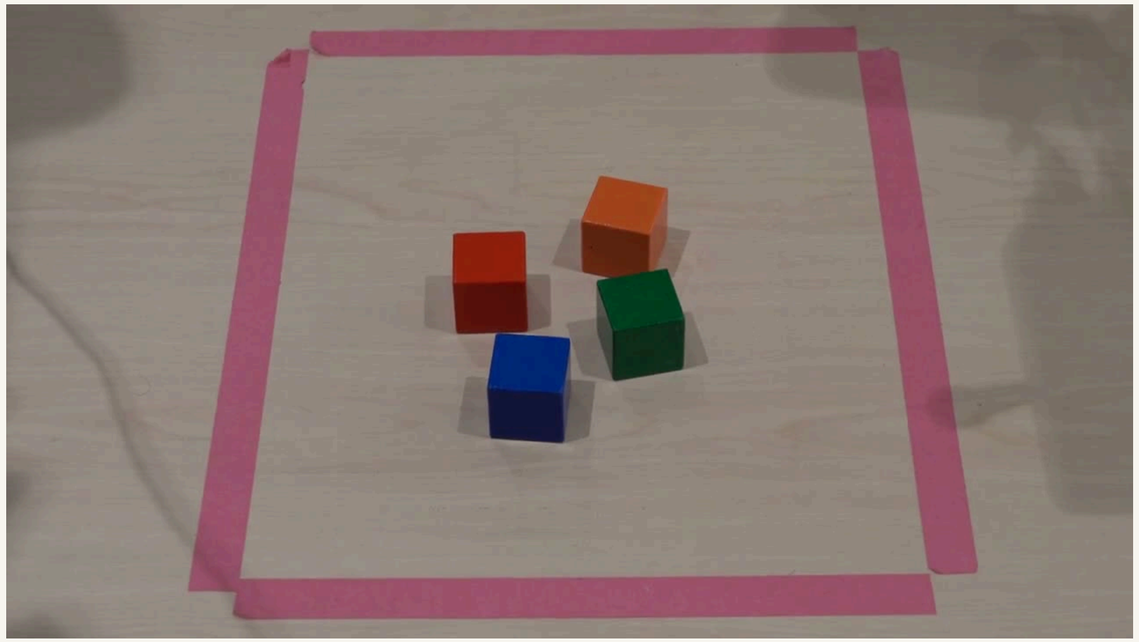
Executable interface

```
pick_place("red block", "blue block")
```

$$a_t \sim \pi_\theta(\cdot \mid h_t, g)$$

h_t: observation–action history · g: goal · a_t: next command

Language can predict a transition



Use the same scene and supply a candidate action.

Given

The red block is on the table. Place it on the blue block.

Predict

The red block is now on the blue block.

$$\hat{o}_{t+1} \sim p_\theta(\cdot \mid h_t, a_t)$$

$\hat{o}_{t+1}$: a predicted observation, before execution

Choosing an action and predicting its consequence are different queries.

Hao et al. · RAP · EMNLP 2023 · Liang et al. · Code as Policies · ICRA 2023

Language can specify and evaluate success

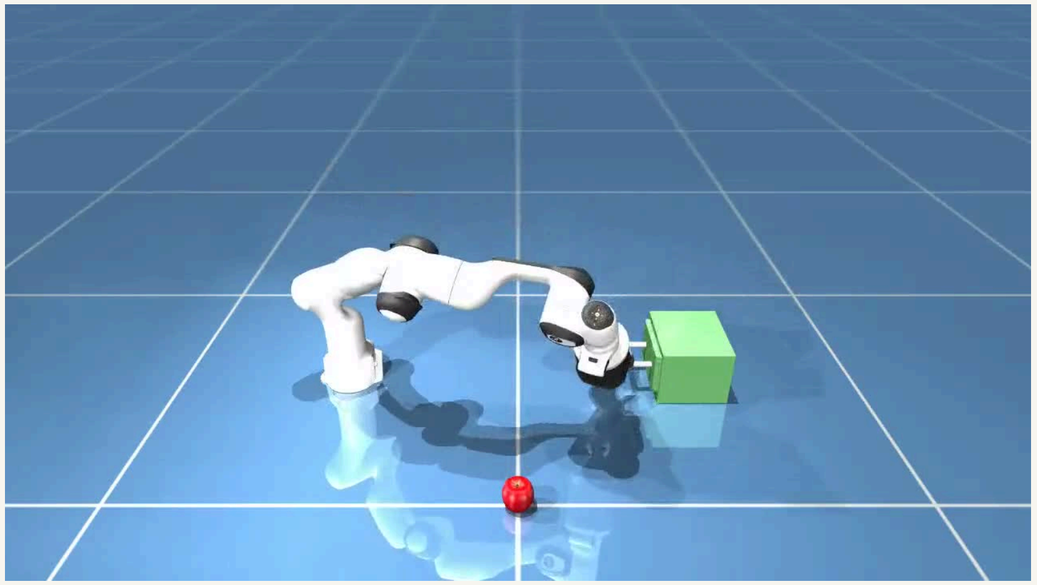
Goal	Put the red block on the blue block.
Success condition	The red block rests on the blue block after release.
Evaluation	Check the resulting support relation.

A goal becomes an optimization signal through a scoring rule.

Yu et al. · Language to Rewards · CoRL 2023 · Ma et al. · Eureka · ICLR 2024

Language to Rewards

Yu et al. · Google DeepMind and collaborators · CoRL 2023



Author demonstration · simulated object manipulation

Input	A request describing the desired behavior.
LLM output	A motion description and reward parameters / code.
Execution	MuJoCo MPC searches using the physics model.

The language model specifies what the controller should optimize.

Eureka

Ma et al. · NVIDIA, Penn, Caltech and UT Austin · ICLR 2024



Author demonstration · learned pen-spinning policy

Write	Generate reward code from task and environment code.
Train	Train the robot policy by RL using that reward.
Revise	Use training feedback to improve the reward.

The LLM writes the reward. RL trains the robot policy that executes the skill.

Reward and future value

Reward

Did this transition help?

A score for a state, action, or outcome.

$$r_t$$

Value

How promising is this situation?

Expected future return under a specified continuation.

$$V^{\pi}(s_t)$$

Language-based evaluation can guide search, but its calibration must be checked.

Hao et al. · RAP · EMNLP 2023

One LLM, several roles

Query	Condition on	Generate
State inference	History of observations and actions	A current-state description
Policy	History + goal	A candidate action
Transition	History + candidate action	A predicted consequence
Evaluation	Outcome + task criteria	A judgment or score

The question and context select the role; the model weights can remain the same.

Hao et al. · RAP · EMNLP 2023

Reasoning supplies context for the next decision

Given context	Place red on blue. Blue is covered by another block.
Generated reasoning	Blue must be cleared first. Move the covering block to the table.
Final command	Move the covering block.

$$p_\theta(z, a \mid h, g) = p_\theta(z \mid h, g)\, p_\theta(a \mid h, g, z)$$

z: generated intermediate text · a: final command · θ stays fixed during this reasoning

Intermediate text can organize a decision before any command is executed.

Wei et al. · Chain-of-Thought · NeurIPS 2022 · Hao et al. · RAP · EMNLP 2023

World modeling inside reasoning

Infer the state	Blue is covered. Red is on the table.
Set a subgoal	Clear blue before stacking red.
Predict a consequence	Moving the covering block will leave blue clear.
Evaluate and decide	That enables the goal. Move it, then check the result.

Reasoning can combine world prediction with goal decomposition, evaluation, and verification.

Teaching illustration · One possible reasoning process; these functions need not be separate networks.

Wei et al. · Chain-of-Thought · NeurIPS 2022 · Hao et al. · RAP · EMNLP 2023

RL trains the language model itself

Sample	Generate several reasoning traces and answers for a task.
Score	Use a test, verifier, or reward model to evaluate the outcome.
Update	Adjust the LLM so relatively successful outputs become more likely.

$$J(\theta) = \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}[R(x, y)]$$

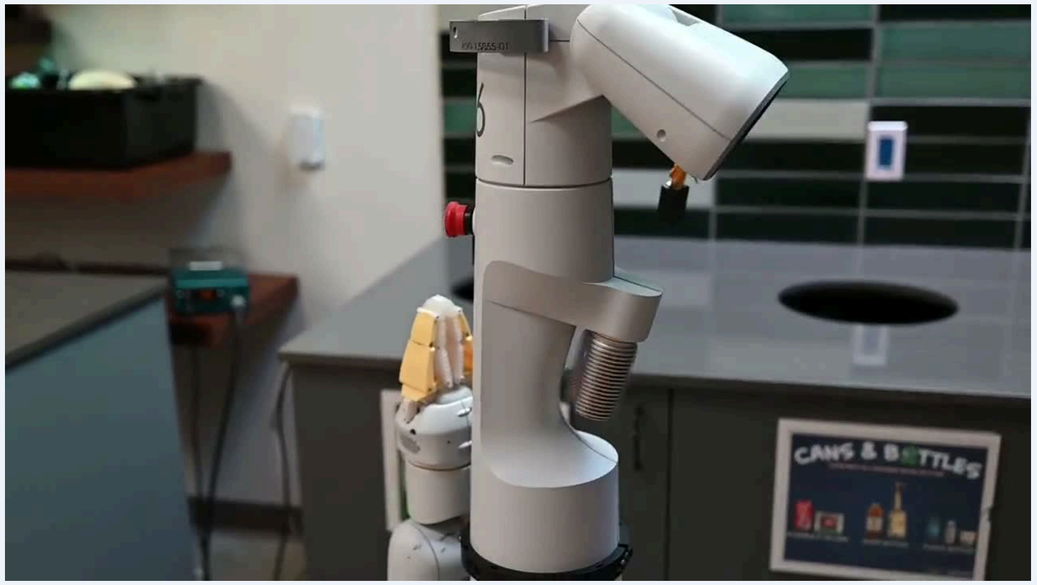
x: task context · y: sampled response, including reasoning · R: reward

SFT supplies an output to imitate. RL evaluates outputs the model generated.

DeepSeek-AI · DeepSeek-R1 · 2025

SayCan: choosing a feasible skill

Ahn et al. · Google Robotics · 2022



Author robot demonstration · instruction-following with a skill library

Task relevance	Does this skill help satisfy the instruction?
Feasibility	Can the skill succeed in the current situation?
Execution	Run the selected skill, then choose again.

Choose a skill using task relevance and its predicted chance of success.

ReAct: reasoning with actual feedback

Yao et al. · ICLR 2023 · ALFWorld task, condensed teaching trace

LLM thought	I need the keychain before I can put it in the safe.
LLM action	Open the drawer.
Environment	The drawer contains a keychain and a watch.
LLM continuation	The keychain is visible. Take it from the drawer.

Execute the action, append the real observation to history, then continue generating.

Agentic RL trains on an interaction trajectory

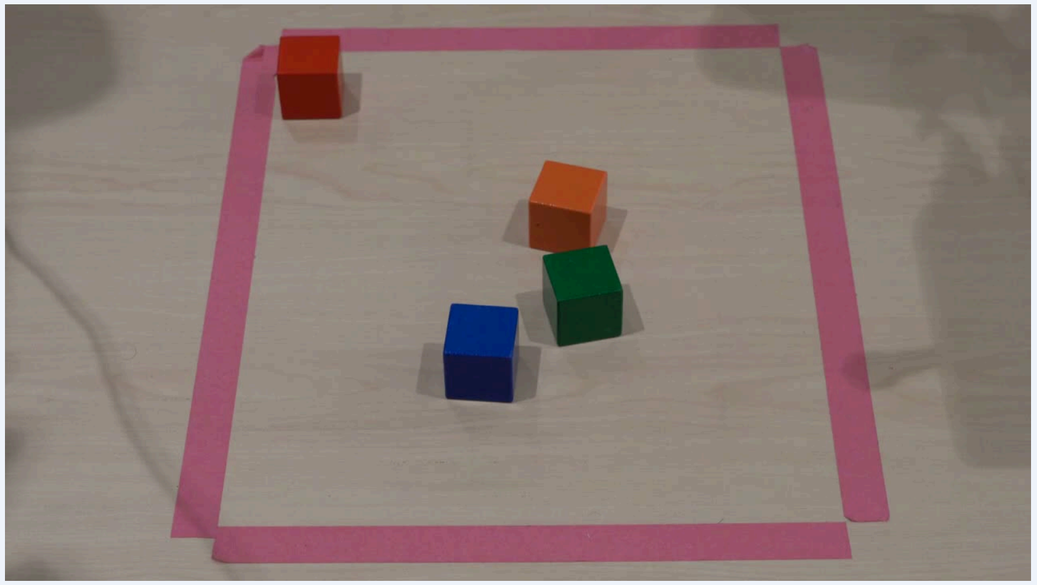
LLM output	Inspect the relevant code and run the tests.
Tool observation	The tests fail; the tool returns the error.
LLM output	Use that feedback to edit the program and test again.
Task outcome	External tests score the submitted solution.

Tool results are appended to history. Generation then continues with the updated context.

Collect trajectories, compute rewards, update the LLM, and repeat.

Code as Policies

Liang et al. · Google Research and collaborators · ICRA 2023



Author tabletop demonstration · code calls perception and motion APIs

```
red = locate("red block")
blue = locate("blue block")
pick_place(red, blue)
```

Illustrative API syntax

Program output

Commands that act on the real environment.

The generated program determines what the robot does next.

ASPIRE: improve an executable policy

Lu et al. · NVIDIA GEAR and collaborators · 2026



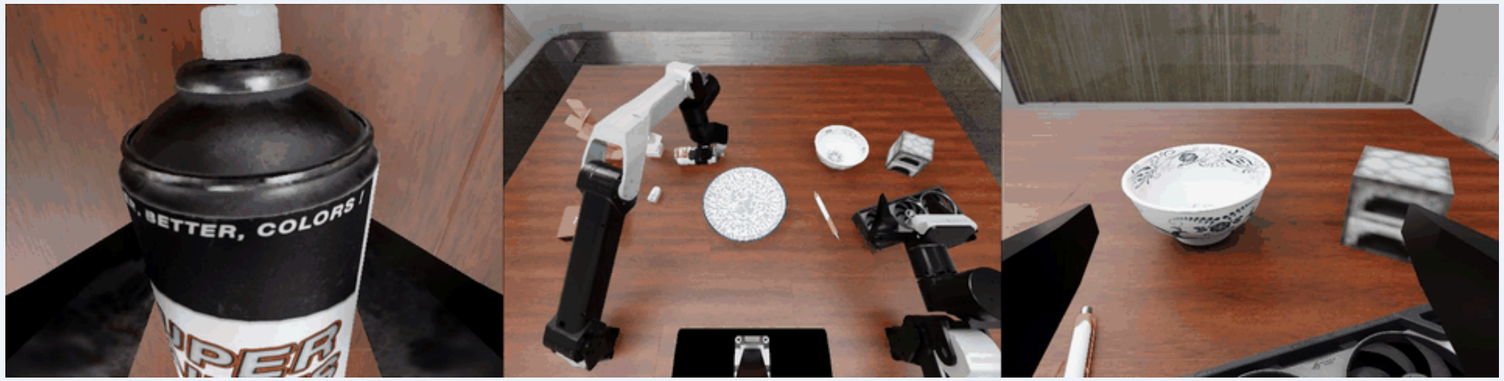
Author robot demonstration · stacking cups

Write	Generate a robot program using the available APIs.
Inspect	Run it and inspect the actual execution trace.
Repair and reuse	Revise the program and save useful skills.

Here experience changes the program and skill library; the LLM weights stay fixed.

Where is the world model in PEAR?

Qwen3.8-27B-PEAR · RoboDojo · one recorded spray-can pickup



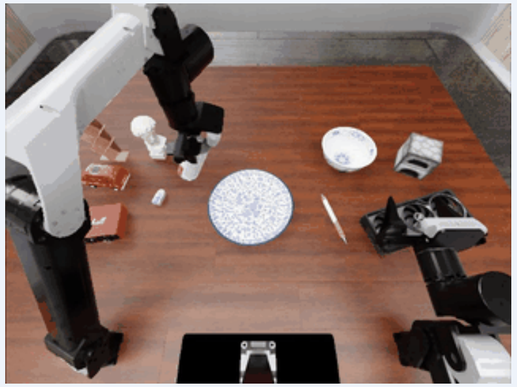
Input: camera views, robot pose, and gripper width.

Policy computation	Choose the next arm or gripper command.
World-model computation	Reason about what that command will do in the current physical situation.

Look for action-conditioned consequences inside the reasoning.

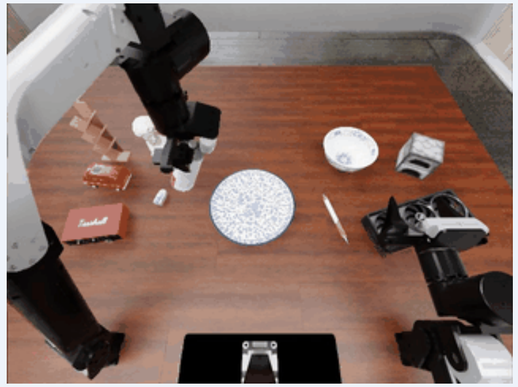
Arm configuration changes what a lift can do

Lift straight up



Observed: no further upward movement.

Retract while lifting



Observed: the arm reaches a higher position.

> “So the arm has reached its limit and can’t lift any higher at this x.”
>
> Visible model reasoning · turn 11

Consequence prediction Pulling the arm back may make further upward motion possible.

The prediction depends on both the current configuration and the proposed action.

Prediction paraphrased from the report; the inferred constraint is a hypothesis.

A model of closing helps infer contact



After closing, the fingers remain partly open.

Expected action effect

If the can blocks the fingers, closing should stop at contact.

Actual feedback

The fingers stop before reaching the fully closed target.

Inferred state

The can is probably between the fingers.

An action-to-consequence rule gives meaning to the observed gripper width.

The report says the stopping rule was supplied in the interface; the LLM applies it to this scene. Contact does not yet guarantee a secure grasp.

PEAR uses local predictions inside reasoning

Representation	“The can is probably grasped; the arm may be at its reach limit.”
Prediction	“Retracting while lifting may let the hand rise farther.”
Interaction	Execute the command, observe the motion, and revise the hypothesis.

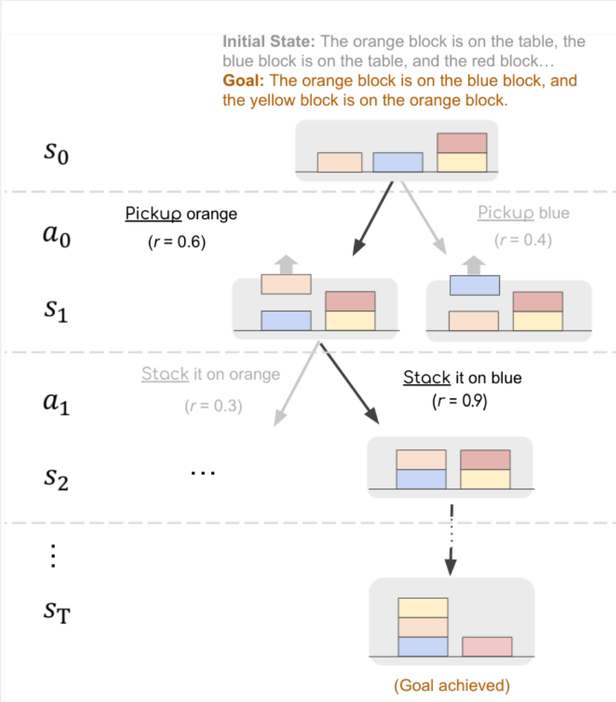
Teaching paraphrases of the episode’s local judgments, not verbatim model output.

Here, world modeling is a task-relevant consequence prediction within the LLM’s reasoning.

Next: RAP makes next-state predictions explicit and chains them into imagined rollouts.

RAP: predicting candidate futures

Hao et al. · UC San Diego and collaborators · EMNLP 2023



Original Blocksworld planning example · Fig. 2, left.

Propose	Enumerate legal actions and score them with the LLM.
Predict	Ask the LLM for the resulting state.
Search	Explore promising action sequences.

In this Blocksworld instance, the LLM scores actions and predicts state changes.

Expand two possible futures

Initial state: orange and blue are clear on the table; the hand is empty.

Goal: place orange on blue.

Stage in every branch	Candidate A	Candidate B
Legal action	Pick up orange.	Pick up blue.
LLM-predicted state	Hold orange; blue stays clear.	Hold blue; orange stays on the table.

$$a_t \in \mathcal{A}(s_t), \qquad \hat{s}_{t+1} \sim p_\theta(\cdot \mid s_t, a_t)$$

$\mathcal{A}(s_t)$: legal actions in the current state. Each predicted state becomes the context for the next step.

Both branches are imagined. Neither action has been executed.

Use predicted outcomes to choose a plan

After picking up orange

Stack orange on blue.

This can directly establish the target relation.

After picking up blue

The goal still requires orange on blue.

Further actions are needed to free the hand and reposition orange.

Goal-condition score	Compare predicted block relations with the specified goal.
Search value	How promising is the continuation explored from this branch?

Search compares continuations; only the selected action is sent to the environment.

What does an execution trace teach?

Executed statement

```
inventory = ["key", "map"]
held = inventory.pop()
```

A compact teaching example of traced program state.

Resulting local state

```
inventory = ["key"]
held = "map"
```

Execution supplies the target for prediction.

During data collection	Run the program and record what changed.
During model prediction	Generate the likely next runtime state from context.

Execution prediction learns effects that static code alone leaves implicit.

The same experience can train two capabilities

Training question	Target or feedback	Capability
What followed this action?	The recorded next observation.	Predict consequences.
Which choices led to success?	A reward on the generated behavior.	Improve the policy.

One interaction Inspect files, run code, observe the result, and complete a task.

A reasoning model can learn both how a world changes and how to act within it.

Train the model to predict the recorded outcome

Observed context	`inventory = ["key", "map"]`
Executed action	`held = inventory.pop()`
Observed result	`inventory = ["key"]; held = "map"`

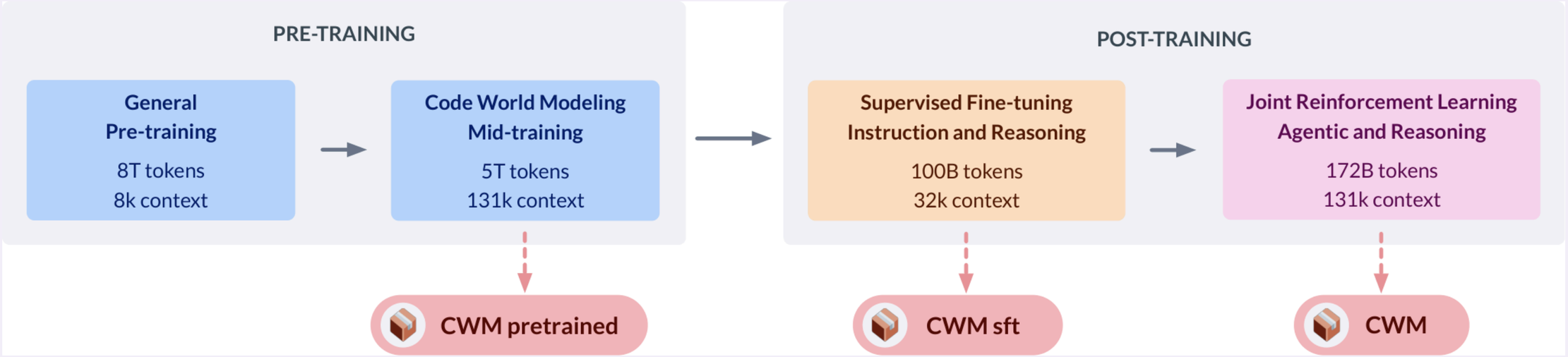
$$\mathcal{L}(\theta) = -\log p_\theta(o_{t+1} \mid h_t, a_t)$$

Train on recorded outcomes. The illustrated transition is a teaching example.

The recorded next observation is the target, including after an unsuccessful action.

CWM: learning from program execution

FAIR CodeGen team · Meta · 2025 · 32B decoder-only LLM



CWM training stages · original paper figure.

Python traces

Executed lines paired with changes in local program state.

Docker interaction

Commands paired with feedback from a software environment.

Execution data teaches consequences; SFT and RL also teach how to solve tasks.

Policy code and world-model code

A program that acts

```
def choose_action(history, goal):
    return pick_place("red", "blue")
```

Returns a command to execute.

A program that predicts

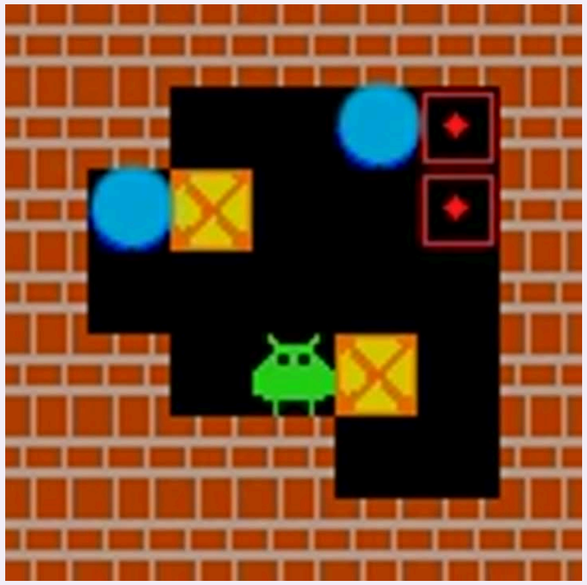
```
def step(state, action):
    next_state = apply_rules(state, action)
    return next_state
```

Returns a consequence for a supplied action.

The function’s input and output determine its role.

WorldCoder

Tang, Key & Ellis · Cornell · NeurIPS 2024

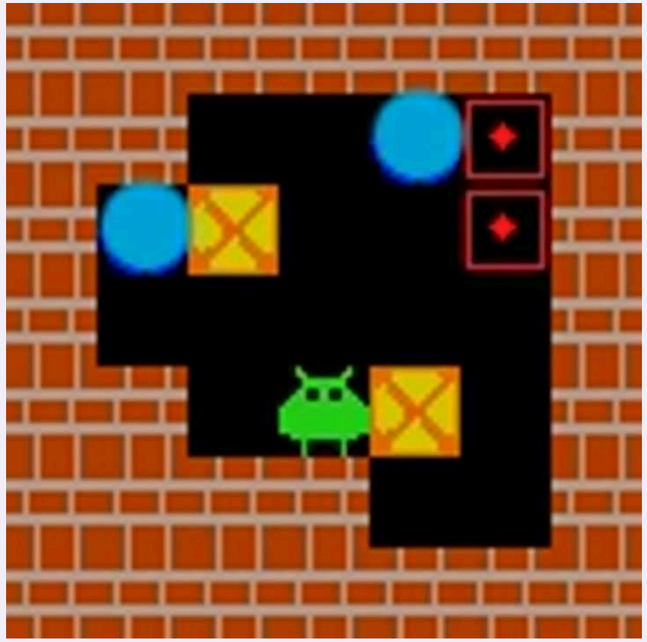


Author demonstration · Sokoban with teleporters

Experience	Observe states, actions, and their consequences.
Model	Write Python rules for the environment.
Use	Execute the model to plan and explore.

The LLM constructs a simulator that can be queried repeatedly.

Turn an observation into a rule



Sokoban with teleporters · official project demonstration.

```
if action == "activate" and
on_teleporter(state):
    exit = other_teleporter(state)
    state = move_player_to(exit, state)
```

Condensed learned transition; exit-collision handling omitted.

What this rule predicts

Activate while standing on a teleporter to move to its paired exit.

A reusable rule explains more than one observed transition.

An unexpected result changes the model

Given state and action	The player stands on a teleporter and chooses activate.
Current model predicts	The player remains at the entrance tile.
Environment returns	The player appears at the paired exit.
Revise the model	Add the activation rule and recheck earlier transitions.

Teaching illustration · A schematic counterexample to a model that omitted teleportation.

Code World Models for General Game Playing

Lehrach et al. · ICLR 2026

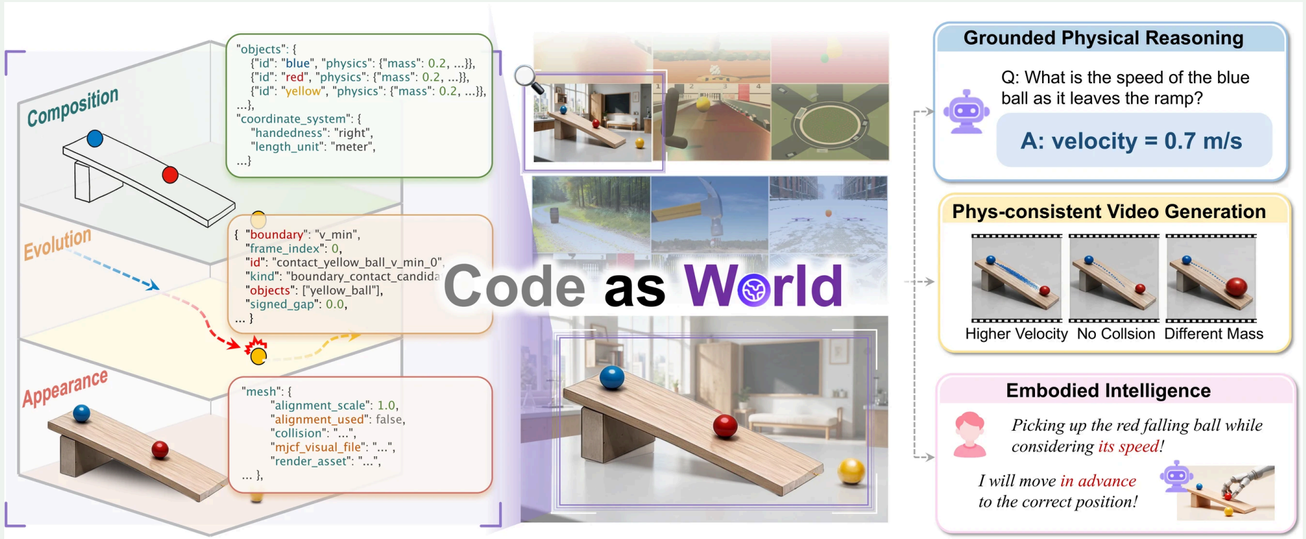
```
next_state = transition(state, action)
choices = legal_actions(state)
done = is_terminal(state)
```

Construct	Use rules and interaction trajectories to generate executable game models.
Plan	Let a search algorithm query the generated environment.

Executable interfaces let a language-generated model support many simulated rollouts.

Code as Worlds

Wang et al. · 2026



Executable composition, evolution, and appearance · author overview.

Infer a world — Use observations and task evidence to propose objects, parameters, and executable rules.

Execute and revise — Compare the simulated or rendered result with evidence and revise the program.

GPT4Motion

Lv et al. · 2023 preprint / CVPR Workshops 2024 · training-free composition



Author demonstration · simulated structure guides generated appearance

Code	GPT-4 writes a Blender scene and simulation script.
Simulation	Render motion structure such as depth and edges.
Generation	Condition image diffusion on the simulated structure.

A simulator computes motion; a generative model supplies appearance.

Coding Agent as World Brain

Chen, Lin & Zhang · Westlake University and NTU · 2026



Author generated-world example · RGB observation

Maintain	Persistent state and executable rules.
Modify	A coding agent updates the world's state or behavior.
Render	A video model generates observations from proxy constraints.

Persistent state can live outside the generated video frames.

A proxy world and its visual observation

Proxy



Geometry and motion provide time-aligned constraints.

Generated RGB



A learned model fills in the visual appearance.

The video renderer learns from paired proxy / RGB data; the program maintains world state.

Mundo: a world from context images

Mundo · author project · 2026



Author scene-exploration example · laundry room

Observe	Start with images of a scene.
Construct	Build a 3D proxy and an executable scene description.
Explore	A simulator advances the proxy; a video model renders its appearance.

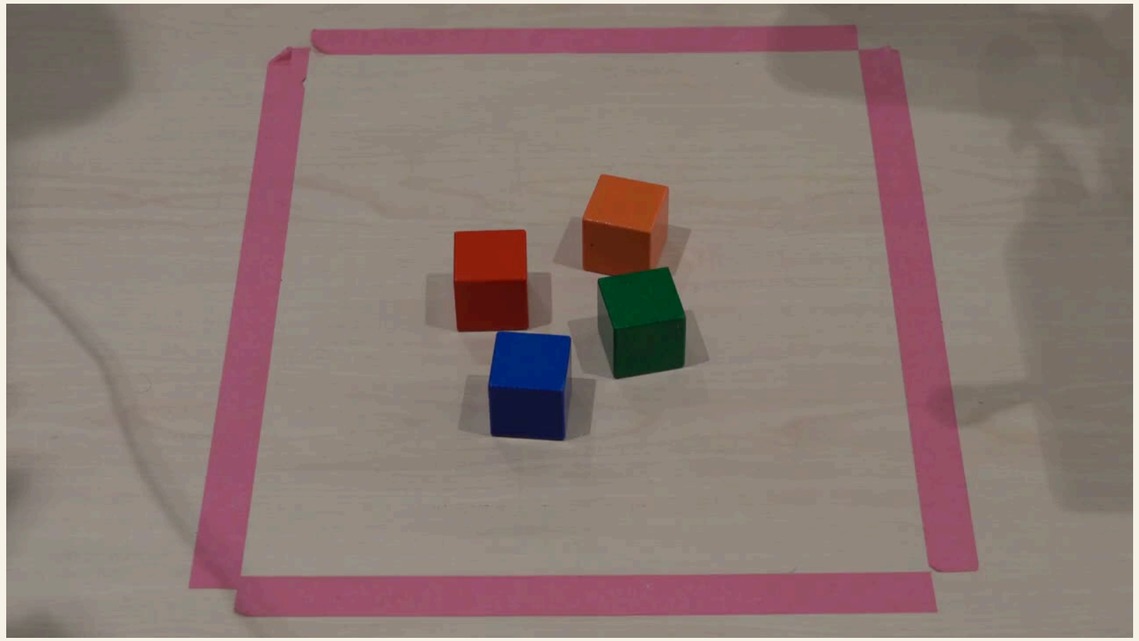
Context images specify this scene. The clip illustrates exploration, not a full physics test.

Who computes the next state?

System	Next-state computation	Observation / evaluation
Interactive agent	The actual environment evolves.	Sensors and external task feedback.
Language rollout	The LLM predicts a state description.	Task rules or a learned evaluator.
Executable world model	Generated code executes transition rules.	Program output, checked against evidence.
Simulation + video	A simulator advances the proxy world.	A generative model renders observations.

These components can be combined within one system.

What gets lost when a state becomes text?



A single sentence can omit details that matter for the next action.

“The red block is next to the blue block.”

Missing geometry

Exact pose, clearance, support, and gripper alignment.

Missing physics

Friction, mass, contact, and possible slip.

Choose an abstraction that preserves the consequences you need to predict.

What should a predicted rollout preserve?

Specified structure



A proxy defines a particular motion and arrangement.

Generated observation



The renderer produces a plausible-looking frame.

Check one step	Did the supplied action produce the correct change?
Check a rollout	Do identity, geometry, and constraints remain consistent?

Visual quality and fidelity to the evolving world require separate measurements.

Chen, Lin & Zhang · Coding Agent as World Brain · 2026

When a predictor judges its own prediction

Assumption	The gripper is securely holding the object.
Predicted result	A lift will raise the object.
Self-evaluation	This action will complete the task.
External check	After lifting, is the object actually off the table?

Teaching illustration · A shared mistaken assumption can enter both prediction and evaluation.

A reward can miss the real goal

Intended behavior	Lift the object securely.
Incomplete reward	Give credit when the hand reaches a high position.
Missing condition	The object must move with the hand and remain supported.

A correctly executed reward program can still encode the wrong objective.

Teaching illustration · A reward-design failure, not a reported Eureka experiment.

Ma et al. · Eureka · ICLR 2024

Reasoning time matters for physical action

Task-level reasoning

Choose and revise a skill.

Uses history, tools, and longer-horizon context.

Feedback control

Track motion as the world changes.

Uses frequent sensor updates and immediate corrections.

Action duration and feedback frequency must match the environment’s time scale.

Evaluate the prediction and its role in reasoning

Capability	A direct test
State inference	Check the inferred relations against observations.
World prediction	Compare predicted and executed consequences of held-out actions.
Evaluation	Check model judgments against external task criteria.
Reasoning benefit	Hold tools and compute fixed; test whether the predictions improve decisions.

PhySimCode evaluates executable physical reconstruction from video. Predicting a changed intervention requires a further test.

Test the prediction itself, then test whether reasoning uses it effectively.

World modeling can be part of reasoning

Maintain a situation	Use observations and memory to track what is true.
Predict and decide	Imagine consequences, compare goals, and choose an action or a check.
Make predictions executable	Construct world programs that reasoning can query.

Language can bring these computations together within one model or one reasoning process.