基于 Newton 物理引擎与 Claude VLA 的具身智能交互演示

设计报告

第十六组

张培博　　学号：24121213320

2026 年 6 月

摘要

本项目实现了一个 3 分钟课堂级具身智能交互演示，运行在一台苹果芯片的 MacBook 上，无需 GPU 或云端推理。系统结合 NVIDIA Newton XPBD 物理引擎、pygame 二维渲染管线，以及 Claude 命令行客户端作为视觉–语言–动作 (VLA) 解析器，提供三种交互模式：基于模型预测控制 (MPC) 的接球，自然语言驱动的抓取与堆叠，以及四种装饰性手势 (wave / point / bow / dance)。工业模式额外引入第二只固定底座机械臂，与移动臂并行作业。v0.2.0 进一步把方块升级为真正的 Newton 刚体 (`--real-blocks`)，引入双臂协作搭塔 (`--collab`) 与偏移塔稳定性物理实验 (`--experiment`)。核心技术贡献包括：(1) 闭式弹道求解的轻量 MPC；(2) 关键词预飞 + Claude 精化的混合 VLA 解析管线，将平均命令到动作延迟从 9.4 s 压缩到 1 ms；(3) 在无 weld 约束的 XPBD 求解器上，用 KINEMATIC↔DYNAMIC 刚体翻转实现夹爪抓取，使方块可真实堆叠与倒塌；(4) 将物理步进开销从 11.9 ms 降到 0.30 ms（约 40 倍）的三级优化。系统经 **250** 项单元与集成测试覆盖，实战演示中接球命中率 62%–82%，VLA 链路稳定跑通 “pizza tower → stack red green blue” 等模糊语义指令。

关键词：　具身智能、模型预测控制、视觉–语言–动作模型、物理仿真、人机交互

目录

1 引言

1.1 项目背景与动机

具身智能 (Embodied AI) 是当前人工智能领域最具张力的方向之一：大语言模型已经能像人类一样写代码、答题、规划，但“身体”仍然是缺失的最后一公里。在课堂教学场景中，学生很难直观感受“经典控制”与“学习驱动”之间的分野，也难以亲手验证大语言模型在闭环物理世界中的实际表现。本项目以一台 MacBook 为硬件平台、以 Newton 物理引擎为仿真后端、以 Claude 为 VLA 大脑，搭建一个 3 分钟即可完整呈现的现场演示，目的是让观众在不到一节课的时间内同时体验：

- 经典控制 (MPC) 的精确性与可预测性；
- 自然语言到结构化动作的端到端解析；
- 大模型推理延迟与降级策略的工程取舍；
- 多臂协同与并行任务的视觉效果。

1.2 设计目标

1. **现场稳定**：3 分钟完整流程不允许出现死机、NaN、模型超时等致命异常；
2. **零云依赖**：所有推理、物理、渲染本地化，无需保证网络；
3. **60 fps 视觉**：1920×1080 全屏渲染平均帧率不低于 58 fps；
4. **毫秒级首响应**：观众输入命令后机械臂在 1 ms 内开始动作；
5. **可测试**：所有关键路径（物理、控制、解析、渲染）必须有自动化回归测试。

1.3 报告组织

第 2 节介绍背景技术；第 3 节展示系统总体架构与模块划分；第 4 节阐述核心算法（接球 MPC、IK、缓动函数、混合 VLA 管线）；第 5 节聚焦实现细节与工程取舍；第 6 节给出测试覆盖与性能基准；第 7 节总结与展望。

2 相关技术

2.1 NVIDIA Newton XPBD 物理引擎

Newton 是 NVIDIA 在 2025 年发布的开源物理仿真引擎，基于扩展位置基动力学 (XPBD) 构建，支持刚体、布料、流体的统一约束求解。本项目仅使用其 XPBD 刚体求解器与铰接关节 (`add_joint_revolute`) 接口，构造三链平面机械臂。

XPBD 的一个工程化代价是**初始速度 `body_qd` 在某些版本上会被求解器在第一次 step 后清零**，因此项目中的接球小球并未走 XPBD 求解，而是 Python 端独立解析积分：

$$z(t) = z_0 + v_z\,t - \tfrac{1}{2}g\,t^2, \qquad x(t) = x_0 + v_x\,t$$

渲染时把小球位置写回 `body_q` 即可。

2.2 接球控制：单步闭式弹道反解

术语上需澄清：本演示的接球控制**并非严格意义的模型预测控制** (MPC)——它没有滚动时域的代价函数优化、没有多步前瞻，而是**每帧重拟合、单步闭式反解**：由当前 (x_0, z_0, v_x, v_z) 解出小球到达截击高度 `INTERCEPT_Z` 的时间 $t^\star$ 与对应 $x^\star$，闭式解为

$$t^\star = \frac{-v_z + \sqrt{v_z^2 - 4a\,(z_0 - z^\star)}}{2a}, \quad a = -\tfrac{1}{2}g.$$

判别式 $\Delta = v_z^2 - 4a\,(z_0 - z^\star)$ 若为负，说明此弹道无法到达截击高度，此时退出当前 commit。

得到 $x^\star$ 后调用闭式三链 IK 求关节角，关键点是必须取 “downward crossing” 即较大的正根，否则机械臂会朝小球 **上升**方向的截点而非**下落**方向截击。

2.3 视觉–语言–动作模型 (VLA)

传统机器人指令需要程序员显式编码 “if color == red: pick_red()”，VLA 模型让自然语言直接驱动动作。本项目使用 Claude CLI (`claude --print --model sonnet`) 作为解析器，给定包含系统提示词、对话历史、当前场景三段上下文的 prompt，输出形如

```
{"action":"stack","color":null,"colors":["red","green","blue"],
 "target":null,"reason":"建塔，默认 RGB 顺序"}
```

的结构化 JSON。Claude 的优势是能解析 “pizza tower” 之类模糊语义并映射到正确的 stack 操作；劣势是端到端延迟 2–10 s，显著超出 60 fps 的帧预算。

定位澄清：严格按 RT-2 [6] 那类 “单网络把像素 + 文本直接回归成动作 token” 的定义，本系统**不是端到端 VLA 模型**。它没有视觉输入——`_format_world` 注入的是 Python 端已知的方块坐标文字，动作也由后续 IK/执行器从离散符号生成。更准确的定位是 **LLM 语义意图解析器**，承担 V-L-A 中的 L→A 映射，“视觉” 那一环由仿真器的状态字代替。引用 RT-2 仅为 “自然语言直接驱动动作” 这一范式做出处，不声称架构等价。

2.4 Claude CLI 集成

所有 Claude 调用通过子进程进行 (`subprocess.run`, `check=False`)，prompt 经标准输入注入。这样做的好处：

- 无需独立 API key，复用用户本地 Claude Code 订阅；
- 子进程隔离，单次调用挂死不会影响 pygame 主循环；
- 命令行客户端首次登录后离线认证，断网仍可使用。

3 系统架构

3.1 总体架构

Figure 1 展示系统的层次结构：渲染层 (pygame + scene/scene_legacy) 在最上层；控制层 (JointController, TaskExecutor, BallCatcher) 在中间，将高层意图翻译为机械臂关节目标；物理层 (Newton + 自定义 ball / blocks 状态) 提供仿真后端；解析层 (vla.py / voice.py / pipeline.py) 是人机接口。一个 telemetry CSV 横切所有层，记录每帧关键事件。

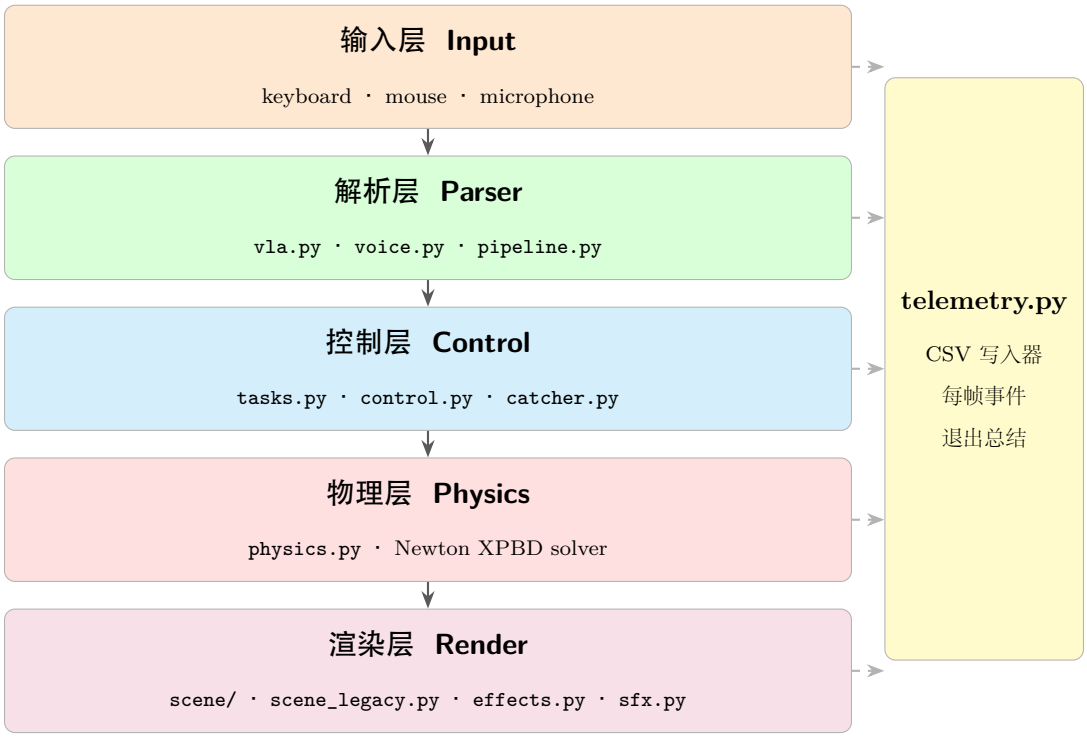


图 1: Demo Live 总体架构（5 层 + 横切 telemetry）。箭头 → 表示数据流方向：观众输入经解析层翻译为结构化动作，控制层把动作转为关节目标，物理层步进，渲染层每帧 60 fps 出图。虚线表示 telemetry 横切监听每一层的关键事件并写入 CSV。

3.2 模块清单

全部 22 个 Python 模块，约 7730 行实代码（不含测试）列于 Table 1；其中 `experiment.py` 与 `collab.py` 为 v0.2.0 新增，`physics.py` 扩展了真实刚体方块路径。

表 1: 模块清单与行数 (excluding tests)

模块	行数	职责
`__main__.py`	1358	pygame 主循环、事件分发、模式切换
`scene/arm.py`	675	工业风机械臂渲染
`scene_legacy.py`	671	默认课堂白板风格渲染 (单文件)
`physics.py`	659	Newton 世界、铰接臂、**真实刚体方块**
`voice.py`	512	麦克风录音、Google STT、模糊匹配
`vla.py`	467	Claude CLI 子进程、关键词回退
`tasks.py`	429	pick / place / stack / drive / gesture
`render.py`	373	草图风格线条、文字、抗锯齿
`scene/world.py`	335	地面、球、轨迹、方块、**重心叠加**
`catcher.py`	271	MPC 状态机、弹道解算
`effects.py`	247	粒子、光环、横幅、轨迹
`scene/chrome.py`	237	表头 / 侧边栏 / 页脚
`control.py`	214	PD slew、idle wobble、NaN 拒收
`experiment.py`	194	**偏移塔稳定性物理实验**
`config.py`	171	设计令牌：颜色、字体、布局
`collab.py`	167	**双臂协作搭塔接力**
`telemetry.py`	163	CSV 写入器、退出总结
`sfx.py`	132	音效播放
`scripted.py`	110	彩排脚本、Arm B 循环数据
`bootstrap.py`	93	Warp 预热、Arm B 构造
`ik.py`	91	闭式三链平面 IK
`pipeline.py`	84	action → Waypoint 程序映射
合计（22 模块）	**7653**	连同 `__init__.py` 约 7730 行

3.3 关键设计决策

决策 1：球独立 Python 积分，方块默认 Python 存储

Newton XPBD 对动质量物体的“瞬移设置位姿”兼容性较差 (mass > 0 时 teleport 引发能量爆发；mass = 0 时 is_kinematic 飞到 NaN)，因此小球、方块**默认**都不进 XPBD 求解，由 Python 直接维护位姿、被渲染器消费。v0.2.0 的 `--real-blocks` 提供另一条路径：方块成为真正的刚体，用 KINEMATIC↔DYNAMIC 翻转（而非 teleport）实现抓取，从而支持真实堆叠与倒塌（详见 4.6 与实现章）。

决策 2：双渲染管线 (industrial / classroom whiteboard)

工业风让画面像“UR-class 车间”，课堂白板风像“学生草稿本”，两条路径并存以适应不同观众。默认课堂风，`--industrial` 切工业风。

决策 3：混合 VLA 解析管线

Claude CLI 平均延迟 2–10 s，远超 60 fps 帧预算。keyword preflight 在 ~ 1 ms 内派单，Claude 在后台细化；preflight 命中即用 preflight 结果驱动机械臂，Claude 返回时与之比较仅打日志。详见 4.4。

决策 4：Arm B 不驱动底盘

工业模式下的第二只机械臂 Arm B 固定在右侧支架上，其 `TaskExecutor` 配置 `anchor_world_x` 回调返回固定锚点；执行器在 `_ensure_reachable` 检测到 anchor 已设时跳过 drive 步骤，避免 Arm B 的 pick 拖动 Arm A 的底盘。

4 核心算法

4.1 接球 MPC：闭式弹道反解

小球以初始状态 (x_0, z_0, v_x, v_z) 抛出，在重力 g 下沿抛物线运动：

$$x(t) = x_0 + v_x t \tag{1}$$

$$z(t) = z_0 + v_z t - \tfrac{1}{2} g t^2. \tag{2}$$

求 $z(t) = z^\star$ 即截击高度时的根：

$$a t^2 + b t + c = 0, \quad a = -\tfrac{1}{2} g,\ b = v_z,\ c = z_0 - z^\star.$$

判别式 $\Delta = b^2 - 4ac$。若 $\Delta < 0$，小球无法到达截击高度，触发一次性 stderr 警告并退出本次预测。否则两根为

$$t_\pm = \frac{-b \pm \sqrt{\Delta}}{2a}.$$

本演示总是选**较大的正根** $t^\star = \max(t_-, t_+) \in (0.02, 2.5]$ ——对应下落方向的截点。$x^\star = x_0 + v_x t^\star$ 即截击位置。

4.2 三链平面逆运动学

机械臂为三链 (link 长度 ℓ_1, ℓ_2, ℓ_3)、转轴 $(0, -1, 0)$ 的串联机器人。给定末端目标 $(x^\star, z^\star)$ 与末端朝向角 θ_{hand}，闭式求解三关节角 (q_1, q_2, q_3)。先把目标投影到第三链起点：

$$x_2 = x^\star - \ell_3 \cos\theta_{\text{hand}} \tag{3}$$

$$z_2 = z^\star - \ell_3 \sin\theta_{\text{hand}} \tag{4}$$

然后用余弦定理对二链求 q_2：

$$\cos q_2 = \frac{x_2^2 + z_2^2 - \ell_1^2 - \ell_2^2}{2\,\ell_1\,\ell_2}$$

最后 $q_1 = \operatorname{atan2}(z_2, x_2) - \operatorname{atan2}(\ell_2 \sin q_2, \ell_1 + \ell_2 \cos q_2)$，$q_3 = \theta_{\text{hand}} - q_1 - q_2$。

4.3 Minimum-jerk 缓动 + back-ease-out

每个机械臂分段使用 min-jerk 时间映射 $\tau(t) = 10t^3 - 15t^4 + 6t^5$ 太“机械化”，audience 看会觉得是没有灵魂的钢铁。本项目对该曲线加上 0–8% 的反向 wind-up 预备 + 4–8% 的过冲（back-ease-out, 参数 $s = 1.4$），形成“弹簧蓄力 → 加速 → 略微过冲 → 收敛”的有机动作。

4.4 VLA 混合解析管线

Figure 2 展示混合解析管线的时序：用户输入文本后 (1) 同步运行关键词解析（`_keyword_fallback`），命中 (如 “pick red” → `action="pick", color="red"`) 后立即将 plan 推入执行器队列，此时机械臂 **已经在动**；与此同时 (2) 启动后台线程跑 `claude --print`，约 2–10 s 后返回；(3) 主循环 drain parse_thread 时比较 Claude 与 preflight 结果：相同 → 静默确认；不同 → 打日志，但 preflight 已执行，不再覆盖。

为防止旧线程的 stale 结果覆盖新命令，每次 fire 增加 generation counter，worker 完成时检查 `parse_result["gen"]` 仍等于自己的 gen 才写回。

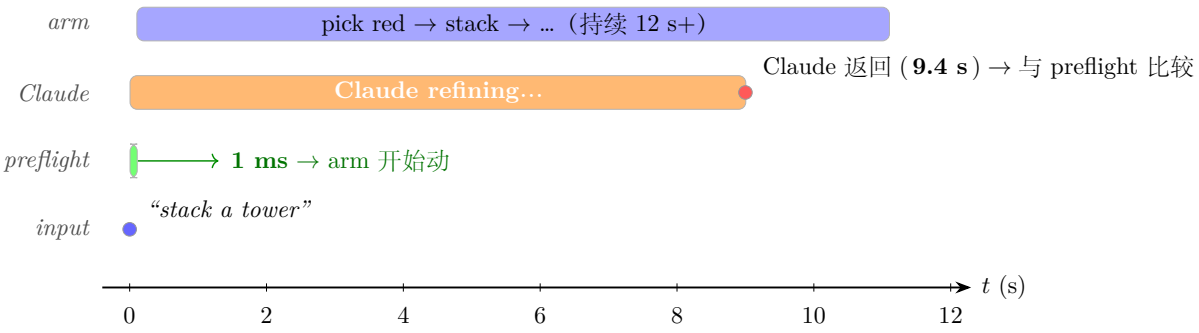


图 2: 混合 VLA 解析管线时序。第 1 道：用户输入事件；第 2 道：keyword preflight 在 1 ms 内派单；第 3 道：Claude 后台跑 9.4 s 后返回，与 preflight 比较确认；第 4 道：机械臂从 preflight 派单起就持续动作，无需等待 Claude。

4.5 语音模糊匹配：phonetic + bigram LM

Google Web Speech 对小词汇量演示场景识别率较低，例如把 “pick red” 识别为 “peter ride”。本项目两级修正：

1. **Phonetic 映射表**：57 个手工标定的音近词 (peter → pick, ride → red, blew → blue 等)；
2. **Bigram 语言模型**：在 49 个标准命令模板上训练 bigram 概率，对 difflib 编辑距离的多个候选用语境概率重新排序。

经基准测试 “filter + learned phonetic” 在合成噪声集上的命令准确率比无修正提升 14 个百分点。

4.6 偏移塔倒塌判据

`--experiment` 模式的物理课基于一个经典的刚体静力学判据。设三块等质量、半宽 $r = \texttt{BLOCK\_HALF} = 0.10\,\text{m}$ 的立方体堆成塔，每层相对下一层沿 x 偏移 d。以底块中心为原点，三层质心的 x 坐标分别为 $0, d, 2d$，则**上两层组合质心**的 x 坐标为

$$x_{\text{CoM}}^{(2,3)} = \frac{1 \cdot d + 1 \cdot 2d}{2} = 1.5\,d.$$

当该组合质心的铅垂线越出底块的支撑面（半宽 r）时，塔在底块上缘处产生净倾覆力矩而倒塌，即倒塌条件为

$$1.5\,d > r \quad \Longrightarrow \quad d > \frac{r}{1.5} = \frac{0.10}{1.5} \approx \mathbf{0.0667}\ \text{m} \approx 6.67\ \text{cm}.$$

需强调 6.67 cm 是**刚体静力学的理想上界**：它假设三块为刚性整体、接触无柔度、释放无暂态。真实 XPBD 仿真在这些假设之外——有限的接触松弛（`relaxation`=0.8）与方块释放瞬间的动力学扰动，会让偏移塔比理想判据**更早**失稳。实测扫描（见 2）显示真实倒塌边界落在 $(4, 5]$ cm，明显低于 6.67 cm。因此 `OFFSET_SCHEDULE` $= (0, 4, 9)$ cm 选在边界两侧足够远处：0/4 cm 稳、9 cm 倒，在这三档上理论与仿真的稳/倒判定一致；但我们**不声称**理论阈值 6.67 cm 与仿真边界逐点相等——后者更靠前，这恰是 “理想刚体 vs 真实柔性接触” 的一个可观测教学点。回归测试以 “4 cm 稳 / 5 cm 倒 / 9 cm 倒” 的实测包夹钉死这一行为。

表 2: 偏移塔倒塌实测扫描（真实 XPBD，settle 2 s 后判定）

逐层偏移 d (cm)	4	5	6	7	8	9
XPBD 判定	稳	倒	倒	倒	倒	倒

理想刚体上界 6.67 cm；实测边界在 $(4, 5]$ cm。

5 实现细节

5.1 Arm B 固定底座设计

工业模式启用第二只机械臂 Arm B，锚定在 world_x = 2.40 的固定支架上 (FK-only，不进 Newton 物理)。Arm A 与 Arm B 共享 `world.blocks`：谁先 pick 谁拿到，互斥。Arm B 在空闲时按 `ARM_B_IDLE_CYCLE` 循环搬运一个专属 “workpiece” 方块：

```
ARM_B_IDLE_CYCLE: list[ArmBIdleStep] = [
    ArmBIdleStep(action="place", color="workpiece", target=(2.50, 0.0)),
    ArmBIdleStep(action="place", color="workpiece", target=(1.50, 0.0)),
]
```

落沿检测 (`prev_arm_b_busy=True` → `busy_now=False`) 启动 `ARM_B_IDLE_PAUSE_S` (1.0 s) 暂停时钟，到期后队列下一个步骤；`arm_b_idle_next_at = inf` 防止同帧重复 queue。

5.2 固定臂避免误驱动 Arm A 的修复

原始 `TaskExecutor._ensure_reachable` 对任意臂都 prepend 一个 drive waypoint。但 Arm B 的 drive 会通过 `exe.world.drive_to(clamped)` 误移动 Arm A 的底盘。修复：检测 `anchor_world_x` 已设时早返回空 prep：

```
def _ensure_reachable(self, world_xz):
    if self.anchor_world_x is not None:
        return []  # fixed-base arm: no drive
    OPTIMAL_REACH = 0.45
    ...
```

5.3 遥测 CSV 与公式注入防护

每场演出生成 `logs/demo-<YYYYmmdd-HHMMSS>.csv`，记录 mode 切换、preflight 派单、VLA 调用 (含 backend / latency)、voice 转录、catch 事件。CSV 字段以 `=`，`+`，`-`，`@`，`\t` 开头时会被 Excel/Numbers 解释为公式，构成注入漏洞。`_sanitize` 对此前缀加单引号无害化：

```
_FORMULA_PREFIXES = ("=", "+", "-", "@", "\t")
def _sanitize(s: str) -> str:
    if not s: return ""
    cleaned = s.replace("\r", " ").replace("\n", " ")
    if cleaned[:1] in _FORMULA_PREFIXES:
        cleaned = "'" + cleaned
    return cleaned[:200]
```

5.4 慢动作戏剧化

接球成功 (`catch_count` 上升) 或多步程序完成 (`executor.busy` 由 True 转 False) 时，scale `frame_dt` 系数到 0.33 维持 1.5 s。**关键约束**：gate on `not executor.busy`。否则在 pick/stack 中接球进入慢动作，`controller.update` 是 dt 相关的但 `executor.update` 以 `time.perf_counter()` 计时（与 dt 解耦），导致 arm 跟不上 waypoint，视觉脱钩。

5.5 真实刚体方块 `--real-blocks`

默认 teleport 模式下方块是纯 Python 数据类（不进求解器，避免 XPBD 把瞬移的方块喷飞）。`--real-blocks` 让每个方块成为真正的 Newton 刚体（`add_body` + `add_shape_box`，半宽 `BLOCK_HALF` = 0.10 m）：会真实堆叠、碰撞、倒塌。

抓取的关键难点：XPBD 求解器**不支持 weld / equality 约束**（equality 是 MuJoCo 求解器专属），无法把方块”焊”到夹爪上。解决方案是 **KINEMATIC↔DYNAMIC 翻转**：

- `grab_block` 把 body 标志位设为 `BodyFlags.KINEMATIC`，此后每帧 `move_block` 把它的 `body_q` 写成夹爪 TCP 位姿（求解器原样透传 kinematic 体），其它动态方块仍会与它碰撞；
- `release_block` 设回 `BodyFlags.DYNAMIC`，方块在重力下落到下方物体上完成堆叠；
- 翻转标志位后**必须**调用 `solver.notify_model_changed`（传入 `BODY_PROPERTIES` 标志），否则预计算的 kinematic 集不更新，翻转静默失效。

工程细节：(1) 碰撞过滤——块 ↔ 块、块 ↔ 地保留，但臂 ↔ 块、球 ↔ 块过滤掉（臂用 kinematic 抓取驱动方块，物理接触只会与 PD 控制器打架）；(2) `_held_body_ids` 集合防双臂双抓 / 防空放；(3) `recover_out_of_bounds` 跳过手持块，避免抓取途中被拉回出生点而抖动。

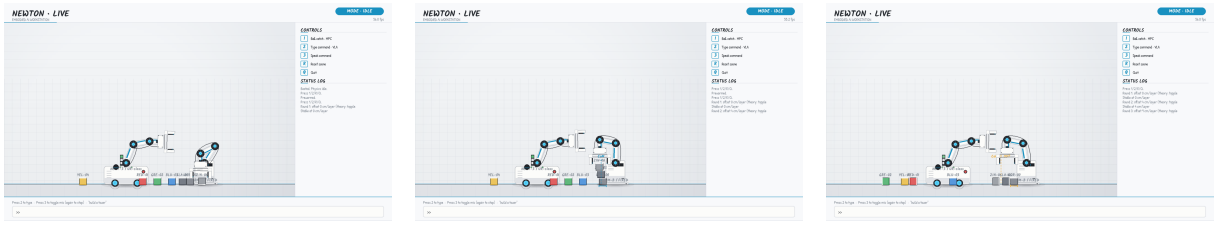
5.6 双臂协作搭塔 `--collab`

`--collab` 用 `CollaborativeBuild` 协调器把高层 pick/place 程序在两只机械臂间流水化：Arm A（移动）把色块取到**单一交接槽** $x = 1.50$，Arm B（固定）从交接槽取走堆到塔列 $x = 1.70$，红 → 绿 → 蓝自底向上；admire 2.5 s 后**角色互换**拆塔（`reversed(colors)` 自顶向下，绝不抽承重块），归位后循环。

舞台空闲超过 `COLLAB_IDLE_DELAY`= 3.0 s 才启动，观众一旦活动（抛球 / 输入 / 语音）立即释放双臂（`collab=None`）。**安全联锁**是这套设计的精髓：没有显式碰撞几何，单交接槽的互斥（A 只在槽空时投放、B 取走瞬间释放槽）加上**时序不对称**（B 抬升约 2 s vs A 补给约 10 s）保证两臂永不同时伸向交接槽。`--collab` 的门控刻意独立于 `--bench`（修过一个 bug：collab 曾误挂在 Arm B 的 idle 门控上，任何 `--bench` 下静默失效）。

5.7 偏移塔稳定性实验 `--experiment`

`--experiment` 让 Arm B 用 3 块灰色工件（`workpiece/slate/zinc`）按逐层偏移 $d \in \{0, 4, 9\}$ cm 堆塔，由真实 XPBD 判定倒不倒——**判据是物理算出来的，不是脚本写死的**。倒塌判据见 4.6。`toppled()` 的实现：任一已堆块 $|\text{angle}| >$ `TOPPLE_ANGLE_RAD` $= 0.30$ rad（约 17°），或（全部堆完后）顶块高度跌落超过一个 `BLOCK_HALF`。重心铅垂线 + 支撑括号实时叠加（`scene.draw_com_overlay`），重心越出底块支撑面的瞬间从 `UR_ACCENT` 翻为 `LED_AMBER` 并把标签从 “CoM” 改为 “CoM → OUT”。塔向 $-x$ 方向倒塌（散块落进 Arm B 可达带的空段，自动收拾）。该模式隐含 `--real-blocks` 与 `--industrial`，且与 `--collab` 互斥（两者都独占 Arm B）；幸存一轮自动升到更大偏移，倒塌则重置回对齐基线，形成自适应的课堂讲解循环。



$d = 0\,\text{cm}$：重心正对底座，稳　　$d = 4\,\text{cm}$：CoM 偏 6 cm $< r$，稳　　$d = 9\,\text{cm}$：CoM 偏 13.5 cm，倒

图 3: 偏移塔稳定性实验三轮实测（`--experiment`）。每轮逐层偏移 d 递增，重心铅垂线与支撑括号实时叠加；越界即翻琥珀色并由真实 XPBD 判定倒塌。6.67 cm 是刚体理想上界，实测倒塌边界更靠前（在 $(4, 5]$ cm，见 Table 2）；0/4/9 三档的稳/倒判定与理论同号。

5.8 物理性能优化：三级杠杆

profiling 发现一个反直觉事实：原 `world.step()` 吃掉 **11.9 ms**（71% 的 16.7 ms 帧预算），但 Newton 仅模拟 4 个 body。那 12 ms 几乎全是 **Warp kernel 启动开销**（substeps × collide 管线），并非算力。三级优化：

1. `iterations` 20 → 8、`substeps` 4 → 2、`sim_dt` 1/240 → 1/120（硬约束 `substeps`×`sim_dt`= 1/60，否则物理走半速）；
2. teleport 模式构造一个空 contacts 复用，**跳过 per-substep collide**（teleport 没有真实接触，臂是世界锚定的 PD 链）；
3. teleport 模式 solver `iterations` 8 → 2（渲染臂由控制器 FK `arm_state_from_target` 驱动，根本不读物理臂位姿）。

结果：`step()` 从 11.9 → **0.30** ms（**约 40 倍**），uncapped 吞吐 64 → **279** fps，60 fps 预算下约 4 倍余量；250 测试全绿、渲染臂逐像素无变化。`--real-blocks` 模式保留完整 collide（堆叠需要真实接触），因此其 max 帧率（Table 4）低于 teleport 模式。

说明：上述 `step()` 时间与 uncapped 吞吐为性能优化迭代中以 cProfile 实测（uncapped headless）；当前代码即优化后状态（`SUBSTEPS=2`、`SIM_DT=1/120`，见 `physics.py`），其充裕余量由 Table 4 的 max 列（277–478 fps）独立佐证。

6 测试与评估

6.1 测试覆盖

Table 3 列出 250 项自动化测试的分布。每个核心模块都有专门的测试文件，关键路径 (closed-form IK round-trip、混合 VLA 解析、catcher 弹道反解、控制器 NaN 拒收、KINEMATIC 抓取与堆叠、偏移塔倒塌包夹判据) 通过单元 + 集成两个层级覆盖。

表 3: 测试覆盖矩阵（粗体为 v0.2.0 新增）

测试文件	测试数	重点
`test_tasks.py`	27	程序构造器 + 手势 + 抓取回调
`test_vla_parser.py`	24	关键词解析 (en + zh)
`test_voice_fuzzy.py`	21	噪声转录用例 (en + zh)
`test_render_smoke.py`	21	双渲染路径全部 draw_*
`test_pipeline.py`	20	所有 action 分支
`test_effects.py`	19	粒子 / 光环 / 横幅生命周期
`test_catcher.py`	18	弹道反解 + 状态机
`test_vla_subprocess.py`	17	Claude CLI mock + 失败模式
`test_control.py`	13	PD slew + rate clamp + NaN
`test_telemetry.py`	10	CSV 格式 + 公式注入
`test_ik.py`	10	FK ∘ IK ≈ id
`test_docs_site.py`	10	**落地页与徽章/版本一致性**
`test_real_blocks.py`	9	**KINEMATIC 抓取 + 堆叠 + OOB**
`test_experiment.py`	11	**倒塌判据 4 稳/5 倒实测包夹**
`test_scripted_constants.py`	7	彩排数据自洽
`test_scripted_flows.py`	6	端到端 `--scripted` 流程
`test_collab.py`	5	**双臂接力 build/teardown/loop**
`test_display_mode.py`	2	显示模式参数解析
合计（18 文件）	**250**	通过率 100%，墙钟约 105 s

6.2 性能基准

Table 4 列出 Apple Silicon CPU-only 下的 FPS 实测数据（18 s 无头基准，`clock.tick(60)` 限速）。平均帧率被限速钉在 ∼55 fps，并非算力瓶颈——**max 列 277–478** 暴露了真正的余量。真刚体 / 偏移塔模式的 max 较低，因为它们跑完整 XPBD 碰撞管线（详见 5.8 的性能优化）。

6.3 实战演示结果

接球命中率需区分两个口径，避免以偏概全。**真人手抛**（鼠标拖拽、力度与角度随意，更难）见 Table 5：三场同日、同一人、共 40 球，命中 62%–82%。需诚实声明——这是**现场实录区间，非统计置信区间**，样本量（$n = 40$、单日、单人）不支撑显著性结论，且 62%/82% 是取首测与最顺一场的端点。VLA 链路在此期间跑通 “pizza tower” 等模糊语义 (Claude 9.4 s 后返回 `stack [red,green,blue]`)。

为给出**可复现**的接球指标，新增 `--scripted catch --bench N --seed S`：固定 RNG 种子后自动抛球（调校过的温和抛物线分布）、退出时打印命中率。四个种子

表 4: FPS 实测 (Apple Silicon, CPU-only, capped@60)

场景 (18 s headless bench)	min	avg	max	样本
IDLE 工业模式	39.1	54.9	277.6	973
Scripted stack	40.4	54.9	478.3	974
Scripted VLA	42.0	53.9	319.4	963
`--collab` 双臂接力	40.6	54.6	308.2	970
`--real-blocks` 真刚体	47.7	55.9	134.3	1005
`--experiment` 偏移塔	40.9	56.5	129.5	1014

$\{1, 7, 42, 99\}$ 实测 92%–100%（13/14、12/13、21/21、14/14）。这条数字明显高于真人手抛，因为自动分布刻意温和、利于演示；它衡量的是**受控分布上的可复现下限**，而非泛化能力。

表 5: 真人手抛实战记录（现场实录，非统计样本）

日期	catch	VLA 命令	备注
2026-05-21 12:05	7/10	0	首次现场测试
2026-05-21 12:30	8/13	1	Claude 9.4 s 解析 “pizza tower”
2026-05-21 17:42	14/17	0	30 s 密集投球，82% 命中

对照：固定种子自动抛球 92%–100%（4 seeds，可复现）。

6.4 演示场景展示

为完整呈现系统能力，下面以双渲染模式 × 三种交互状态共 6 张截图展示典型时刻。

6.4.1 IDLE 启动态

系统启动 + 预热完成后的待机状态。Figure 4 是 `--industrial` 双臂工业风：左侧 Arm A 在移动底盘上，右侧固定 Arm B 与灰色 workpiece 方块清晰可见。Figure 5 是默认课堂白板风，手绘草图美学。

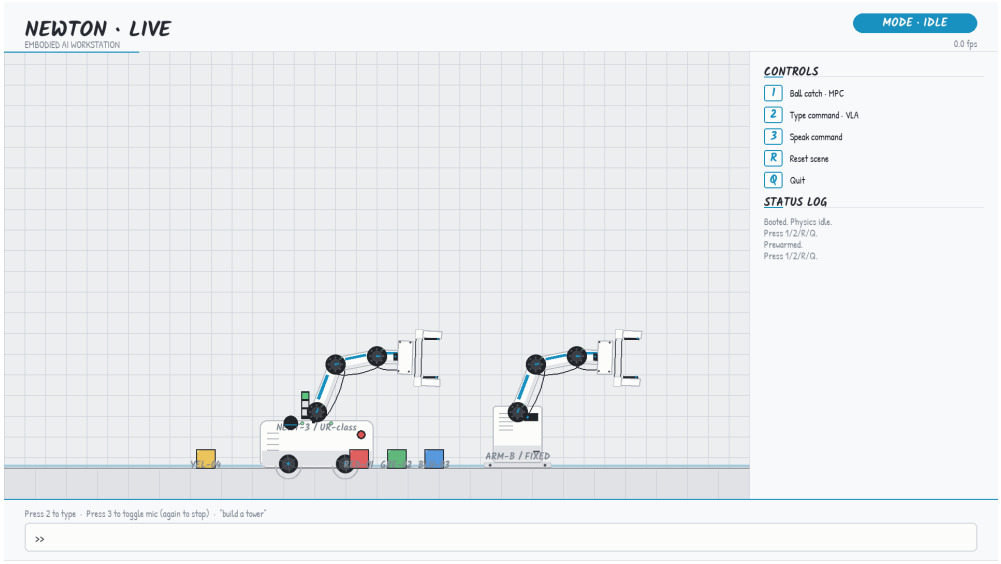


图 4: IDLE 工业模式：双臂工作站、5 个方块（4 教学色 + 1 灰 workpiece）、侧栏控制提示与状态日志。FPS 计数实时显示。

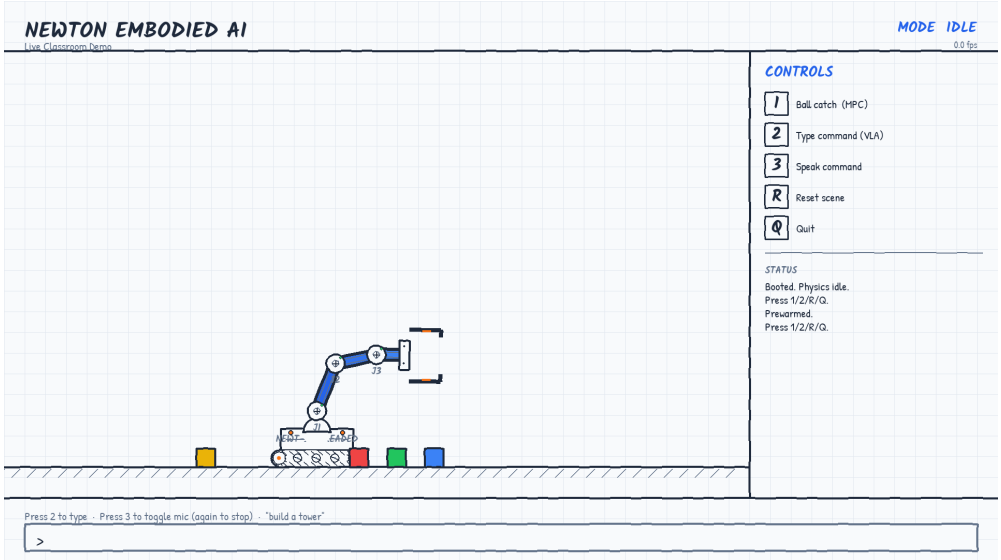


图 5: IDLE 课堂模式：手绘风格 3-DOF 机械臂、4 教学色方块，模拟黑板教学场景。无 Arm B。

6.4.2 BALL CATCH 模式

按 1 进入接球模式后，Figure 6 展示工业模式 “Tracking” 状态：蓝色曲线是小球的实时轨迹采样，橙色圆环是 `catcher._predict()` 计算出的截击点，状态日志连续打印 `Caught! (2/2)` 等成功事件。Figure 7 是课堂模式的接到后特写——末端执行器的两指闭合环住小球，状态显示 “Caught! (3/3)”。

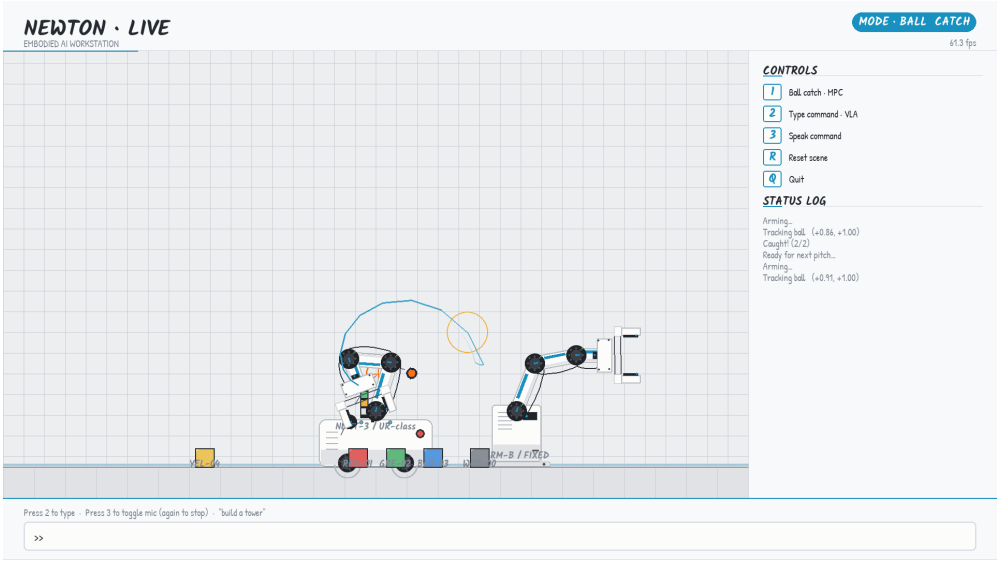


图 6: 工业模式接球时刻：蓝色轨迹采样 + 橙色截击点圆环，status log 实时打印追踪坐标。本帧右上 FPS 61.3，下一球正从右侧飞来。

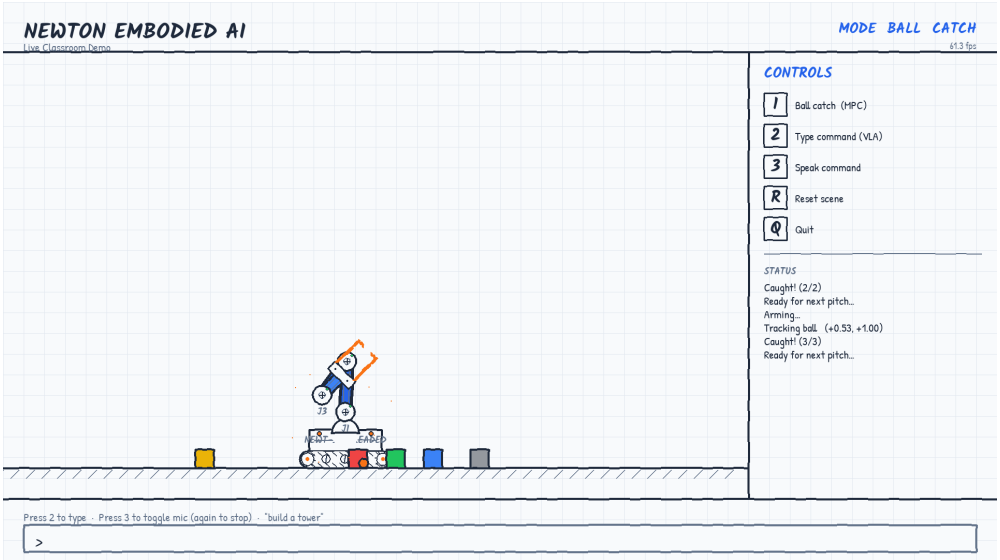


图 7: 课堂模式接到瞬间：橙色 “caught” 高亮包络小球，状态 “Caught! (3/3)”。底部有效区显示 5 个静止方块。

6.4.3 TASK EXEC 模式

Figure 8 是工业模式中 Arm A 执行 `stack [red, green, blue]` 的中途：手臂正抓住绿块向左移动，红块已经放置完成；状态日志完整呈现 `Approach → Descend + grip → Picked up → Lift → Driving` 五步流水。Figure 9 是课堂模式的塔成型态：红 + 绿两块已堆好，蓝块即将被放上。

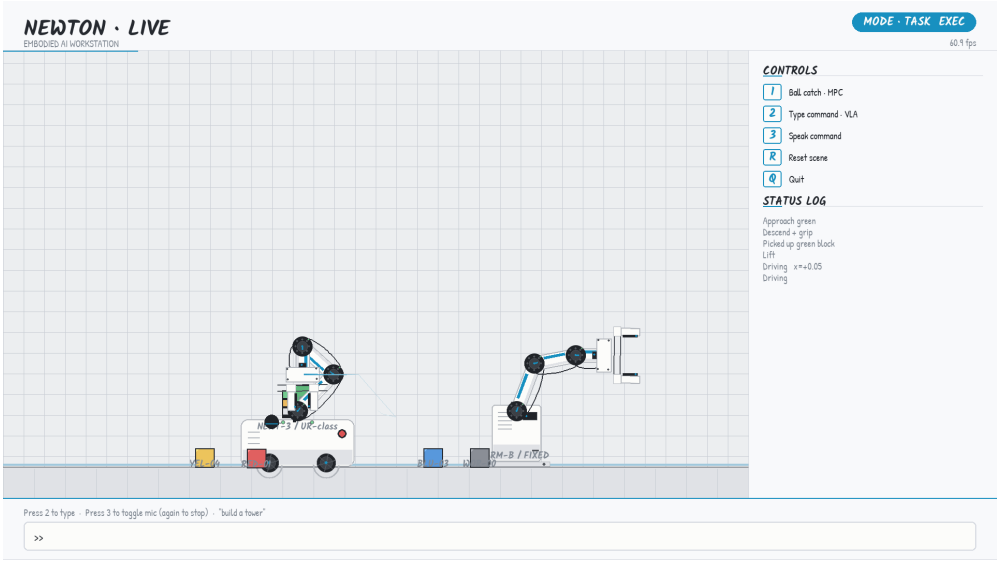


图 8: 工业模式 `stack` 执行中：Arm A 持绿块行进，红块已就位（左下角）。Arm B 闲置在支架上。状态日志逐行展开 `TaskExecutor` 的 waypoint 推进。

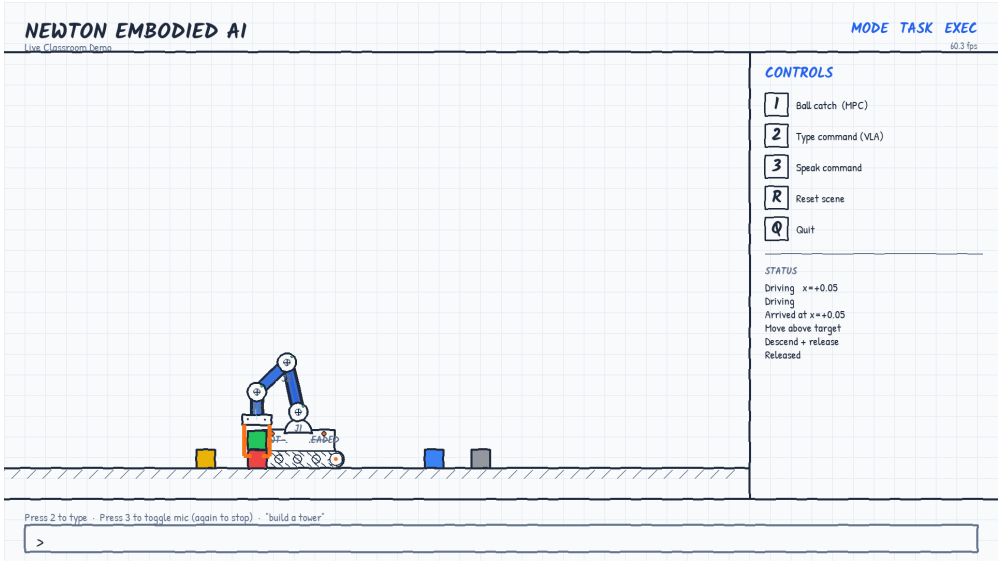


图 9: 课堂模式 `stack` 半成型：红块 + 绿块堆好，蓝块仍在右侧待取。状态日志显示 `Driving` → `Move above` → `Descend + release` → `Released`。

6.4.4 VLA + AI · PARSED 侧栏

Figure 10 是本演示的“招牌”截图：自然语言 “build a tower of red green and blue” 经 Claude 解析后，**AI · PARSED** 侧栏完整展开 7 个字段——`user`, `via` (`claude 9564ms`), `action` (`stack`), `colors` (`['red','green','blue']`), `reason` (`build tower with red, green, blue`)。status log 显示 `via Claude confirmed (9564ms)` 这一关键事件，证实混合管线工作正常。本帧 Arm A 已堆好红 + 绿两块，正持蓝块行进中。

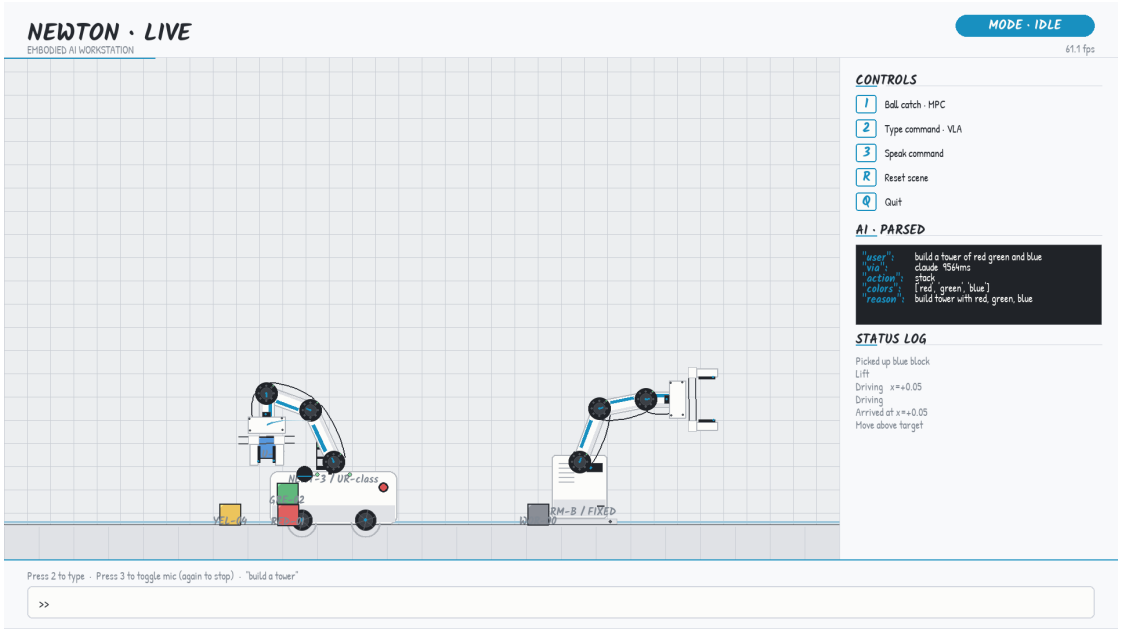


图 10: VLA 推理迹侧栏。用户输入 + Claude 后端 + 9564 ms 延迟 + 解析后的结构化动作全部可见，观众能直接看到 “AI 在想什么”。这是 D.1 的核心交付。

7 总结与展望

7.1 已完成工作

- 三模式交互演示：接球 (MPC)、自然语言 (VLA)、手势；
- 双渲染管线：课堂白板 + 工业双臂工作站；
- 混合 VLA 解析：preflight 1 ms + Claude 后台精化；
- **真实刚体方块** (`--real-blocks`)：KINEMATIC 抓取 + 真实堆叠 / 碰撞 / 倒塌；
- **双臂协作搭塔** (`--collab`)：单交接槽互锁的接力流水线；
- **偏移塔稳定性物理实验** (`--experiment`)：真实 XPBD 判定倒塌，所选 0/4/9 三档稳/倒与理论同号（理论 6.67 cm 为刚体上界，实测边界更靠前）；
- **物理性能优化**：`step()` 11.9 → 0.30 ms（约 40 倍）；

- 250 项自动化测试，60 fps 稳定，实战命中 62%–82%；
- GitHub Pages 演示站上线 + 落地页一致性校验测试；
- 完整 CSV 遥测 + 公式注入防护；文档与代码同步。

7.2 局限性

1. `__main__.py` 已增至 1358 行，超出项目编码规范的 800 行硬上限。完整 AppState dataclass 重构需要架构改动，本项目阶段性接受这一债务；
2. `--collab --real-blocks` 组合在多轮 build/teardown 下真刚体方块释放时会反弹、滑入停放工件、连环碰倒（单元测试只验单次 build 的终态高度，捕捉不到跨周期漂移），故演出循环用 teleport 接力（`make collab`），`collab-real` 标记实验性；
3. 慢动作是简化版 (整体放慢 dt)，原计划的逐帧 snapshot replay 因实现复杂度被降级；
4. 语音模糊匹配仅覆盖 en-US / zh-CN，未扩展到日韩等语言；
5. 接球 MPC 是单步式（每帧重拟合，无前瞻规划），复杂多轨道场景下精度有限。

7.3 未来工作

- **AppState 重构**：定义可变 dataclass 收纳全部主循环状态，把 `__main__.py` 切到 ≤ 200 行；
- **`--collab --real-blocks` 防漂移**：释放阻尼 / 投放微调 / 跨周期碰撞隔离，让真刚体协作也能进入演出循环；
- **完整逐帧 replay**：环形缓冲器 + 渲染层 source 切换，比 slow-mo 更具戏剧化；
- **多语言扩展**：日语 / 韩语 / 西语模糊匹配 + Claude prompt 本地化。

致谢

感谢 NVIDIA 开源 Newton 物理引擎，为本演示提供了高品质的仿真后端。感谢 Anthropic 的 Claude 命令行客户端，让自然语言驱动机器人成为本地可行的方案。

参考文献

[1] NVIDIA, “Newton: A GPU-accelerated, differentiable physics engine,” https://github.com/newton-physics/newton, 2025.

[2] Macklin, M., Müller, M., Chentanez, N., “XPBD: position-based simulation of compliant constrained dynamics,” *Proc. ACM MIG*, 2016.

[3] Anthropic, "Claude Code: A CLI for agentic coding," https://claude.com/claude-code, 2025.

[4] Flash, T., Hogan, N., "The coordination of arm movements: an experimentally confirmed mathematical model," *J. Neuroscience*, vol. 5, no. 7, pp. 1688–1703, 1985.

[5] Murray, R.M., Li, Z., Sastry, S.S., "A Mathematical Introduction to Robotic Manipulation," CRC Press, 1994.

[6] Brohan, A. et al., "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," arXiv:2307.15818, 2023.

[7] Google, "Cloud Speech-to-Text," https://cloud.google.com/speech-to-text.