Title: Real-Time Brax Training Viewer for Policy Visualization

Synopsis

Brax is a fast and flexible physics simulation engine that leverages JAX to train reinforcement learning (RL) agents at scale. While Brax offers high-throughput parallelized training pipelines, particularly with PPO, it currently lacks tools for **real-time visualization of policy behavior** during training. This project proposes the development of a **Brax Training Viewer**, a live visualization utility that allows users to inspect agent behavior inside MuJoCo as the training progresses.

The tool will provide real-time action updates from the currently executing PPO policy and display its behavior within a synchronized MuJoCo viewer. This aims to **bridge the gap between fast RL training and agent interpretability**, aiding debugging, research insight, and educational exploration.

Benefits to the Community

Currently, Brax users can only evaluate policies **after training concludes**, which slows down the feedback loop during model development. This lack of intermediate visibility limits researchers when:

- Debugging unstable reward curves
- Understanding whether a policy is learning meaningful behaviors
- Detecting early signs of divergence or failure modes

The proposed **Training Viewer** would allow real-time streaming of rollout frames alongside current policy actions, providing insight into:

- What behaviors the agent is learning in real time
- How exploration changes over time
- Whether the policy is improving (e.g., in locomotion, balance)

This project will:

- Provide a lightweight, optional visualization layer built on top of the official PPO pipeline
- Allow researchers and developers to **pause/resume rendering** to trade off between performance and interpretability
- Enhance Brax's utility for RL education, experiment design, and reproducible debugging

---

Deliverables

- A **Brax Training Viewer module** that connects PPO training to a MuJoCo renderer
- Real-time streaming of environment states (position/velocity) from training
- Toggle for enabling/disabling viewer updates during training (to preserve performance)
- CLI and Python API interfaces for integrating the viewer into existing training scripts
- Compatibility with the `brax.training.agents.ppo.train` function
- Optional: Logging mode that saves periodic visual rollouts for offline review

Timeline

I'm free for the summer, I don't have any exams or conflicting exam schedule

Period	Deliverables
Getting Familiar	Familiarize with Brax internals, `ppo.train` API, and MuJoCo rendering. Finalize project goals and interface specs.
Week 1–3	Build initial proof of concept for streaming rollout states into a viewer from PPO. Attach viewer to a live policy.
Week 4–6	Implement a synchronized viewer loop with toggle support. Optimize data sharing and JAX→NumPy interop.
Week 7	**Midterm Evaluation**: Basic real-time viewer with working PPO integration.
Week 8–9	Add optional parallel environment tracking and toggle for viewing specific envs. Add log-based replay viewer.
Week 10–11	Polish API, write tests, and prepare documentation/tutorials.
Week 12	Final bugfixes, complete documentation, submit final report.

Technical Details

The system will:

- Wrap around Brax's `ppo.train()` function by injecting hooks into the environment rollout
- Use shared memory or periodic polling to extract `pipeline_state` (i.e., `qpos`, `qvel`)
- Leverage `mujoco.Renderer` for efficient offscreen visualization (no OpenGL context required)
- Support control over FPS, resolution, and rendering frequency
- Use threading or async queues to isolate visualization from training performance

Optional features include:

- Saving snapshots of the simulation as MP4 or GIF
- Adding overlays (reward, step, action magnitude)

My Skill Set

I'm quite familiar with Python, JAX, Reinforcement Learning and Mujoco

About Me

I am currently pursuing a Master's in Robotics at the University of Colorado Boulder. As part of my coursework, I took *Decision Making Under Uncertainty* , where I explored reinforcement learning in depth and implemented various RL algorithms. Although the course used Julia, I am well-versed in Python and have prior experience working with MuJoCo, particularly in the Franka-Kitchen environment. Through this project, I hope to deepen my understanding of tool-building and system integration, while contributing meaningfully to the Brax ecosystem and the broader open-source RL community.

Contact Information

- Name : Thanushraam Suresh Kumar
- Email : skthanushraam@gmail.com
- Country of Residence : United States of America
- Resume : Resume
- Project Repository : Tr0612/brax_ant
- Project Videos : Playlist Link
- Github Profile : Tr0612 (Thanushraam)
- LinkedIn Profile : Thanushraam Suresh Kumar