Tangled Program Graphs

Capstone Project

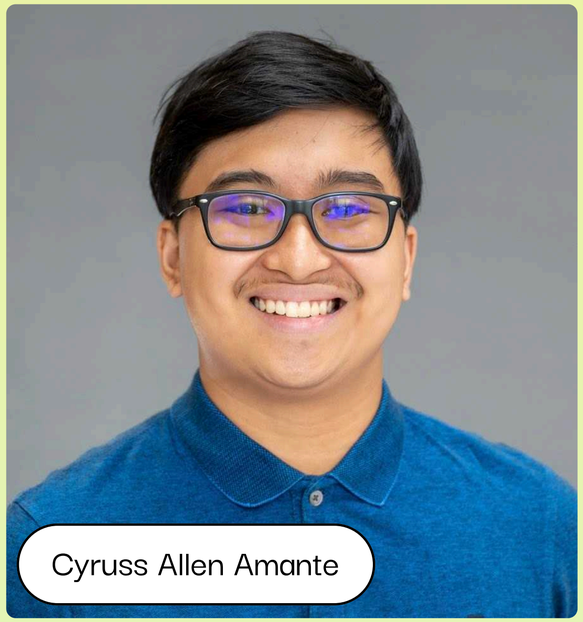
TEAM 03 - TANGLE

CYRUSS ALLEN AMANTE, MARK ANGELO CRUZ, EDWARD GAO,
RICHARD LI & CALVYN SIONG

MARCH 2025

UNDER THE SUPERVISON OF DR. STEPHEN KELLY

OUR TEAM


Cyruss Allen Amante


Mark Angelo Cruz


Edward Gao


Richard Li


Calvyn Siong

Here’s what we’re going to cover today.

- Context
- Goals
- Demo
- Key Benefits
- Next Steps

How can we teach a robot how to walk on it’s own?

Real-World Applications

Humanoid Robotics

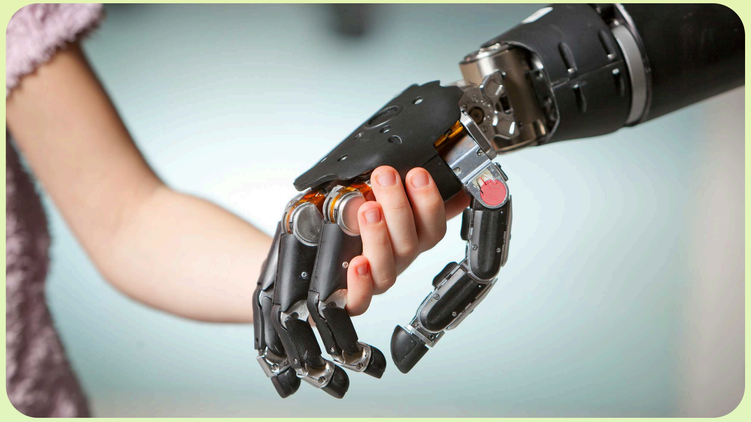
General-purpose AI for a human world

Advanced Prosthetics

Creating more natural and responsive prosthetic limbs for amputees

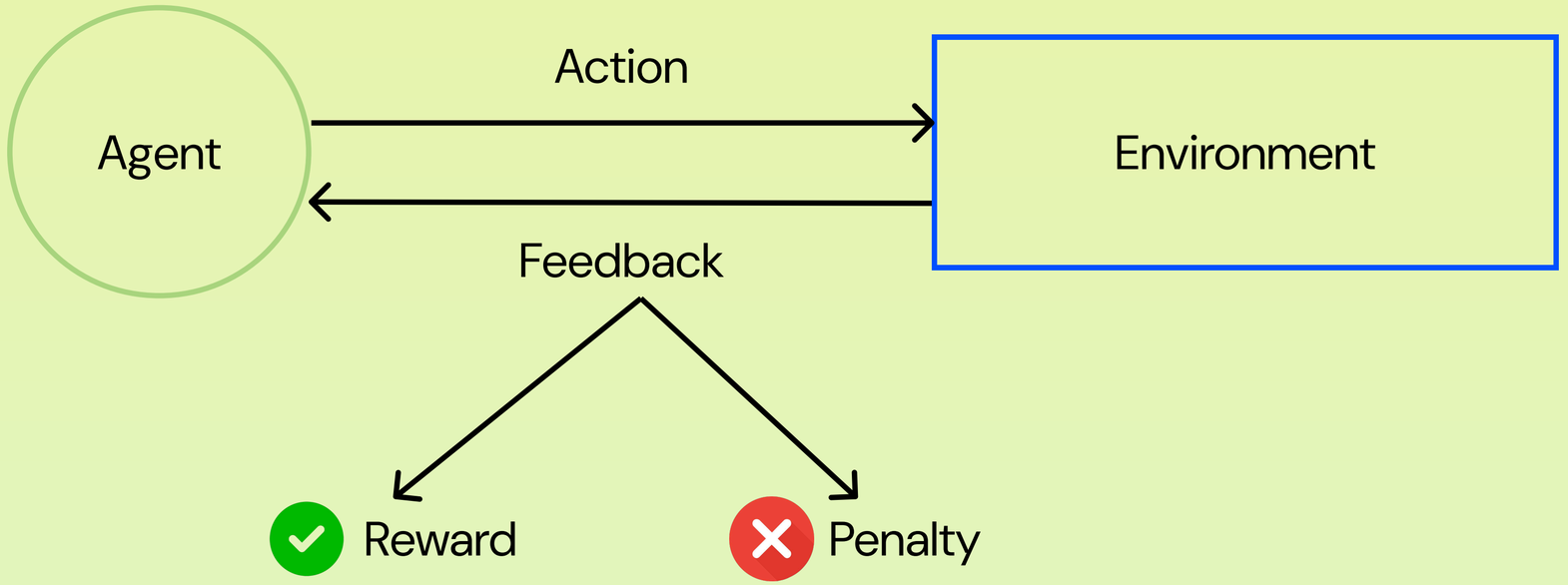
Autonomous Vehicles

Enabling self-driving cars to navigate complex urban environments safely and efficiently



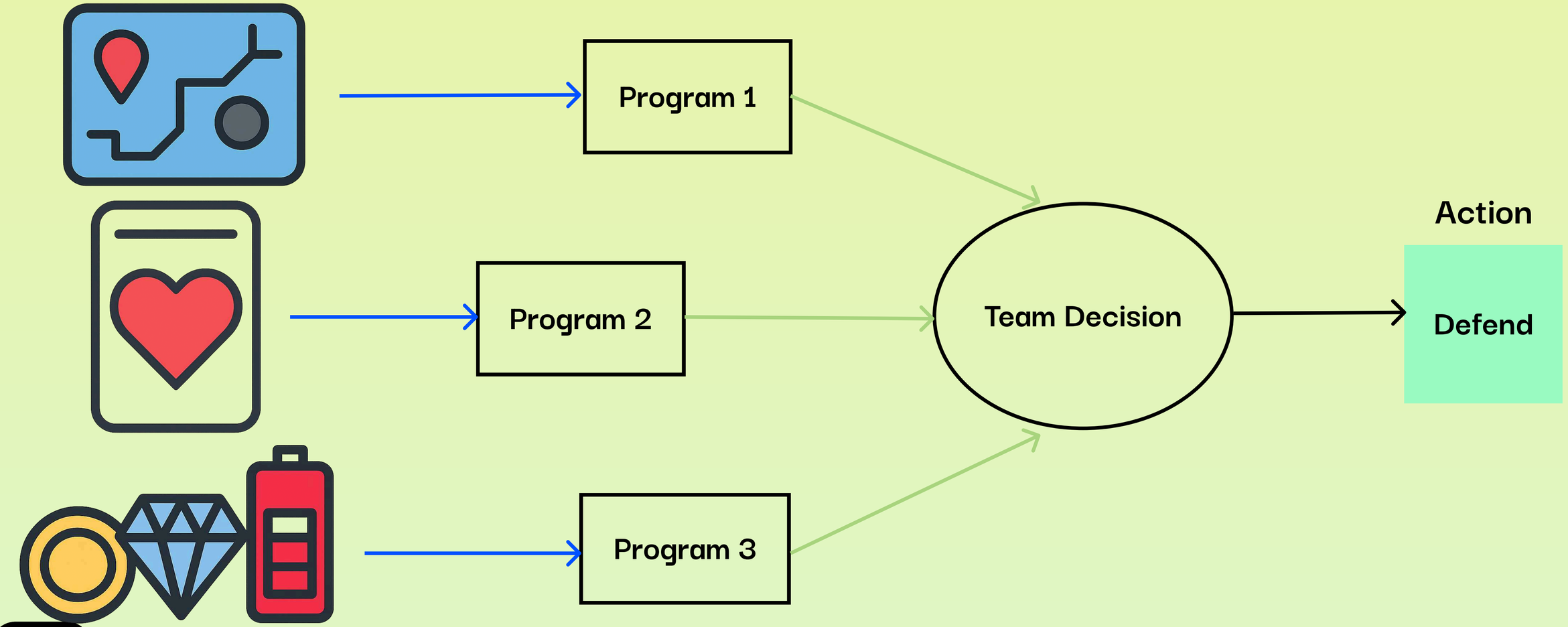
How do we create AI that can learn complex movements and adapt to different environments?

Reinforcement Learning

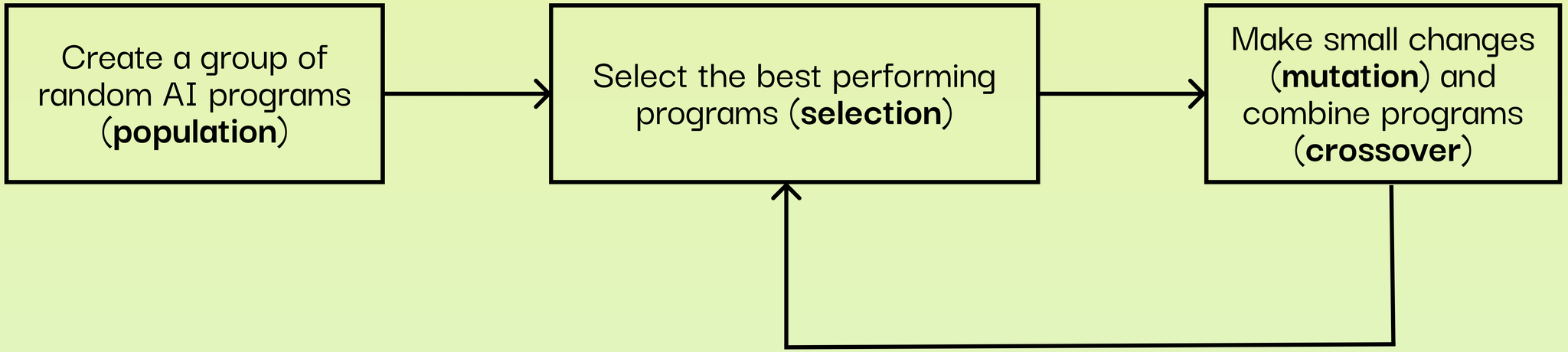


But how do we structure the Agent's "brain" to learn these complex movements?

Tangled Program Graphs (TPG)



Genetic Programming



So we have our AI 'brain' (TPG). But where does it practice walking? Where's the environment?

MuJoCo (Multi-Joint dynamics with Contact)

3D Physics Engine

Google DeepMind

Designed for Robotic Research

Goals and Motivation

1. MuJoCo Environments Integration

2. Software Engineering Best Practices and Improvements

CMake

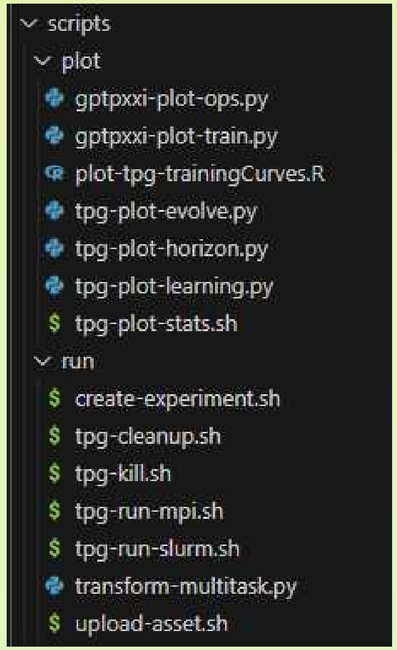
pipx

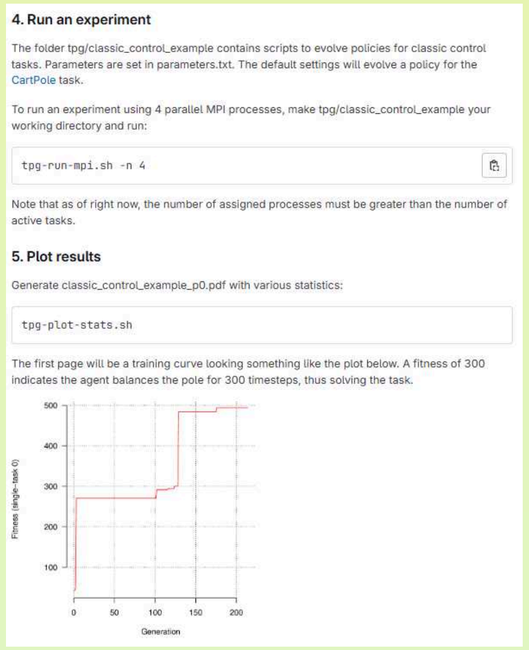
matplotlib

Initial DevEx Pain Points

Pain Point #1

Lack of standardized CLI





Available scripts (left) and README for running scripts (right)

Pain Point #2:

Overwhelming Logs

```
InitTms Msz 3000 Lsz 15000 rSz 3000 mSz 0 0 0 eLSz 0
setElTmsMTA eLSz 2996 ss 210 fm 0 minThr 0.404202 tminfo t 0 id 2978 gtm 0 phs 0 root 1 sz 5 age 0 nOut 15 0 mnOut p0t0a0 451.33946 p0t0a1 1.00000 p0t0a2 6.00000 p0t1a0 2
setElTmsST eLSz 2996 ss 2 fm 0 minThr 0.01 tminfo t 0 id 2913 gtm 0 phs 0 root 1 sz 5 age 0 nOut 15 0 mnOut p0t0a0 139.08607 p0t0a1 1.00000 p0t0a2 2.00000 p0t1a0 8.00000
setElTmsST eLSz 2996 ss 1 fm 0 minThr 0.00 tminfo t 0 id 1804 gtm 0 phs 0 root 1 sz 5 age 0 nOut 15 0 mnOut p0t0a0 131.78725 p0t0a1 1.00000 p0t0a2 8.00000 p0t1a0 298.8000
setElTmsST eLSz 2996 ss 0 fm 0 minThr 1.00 tminfo t 0 id 233 gtm 0 phs 0 root 1 sz 5 age 0 nOut 15 0 mnOut p0t0a0 927.80916 p0t0a1 1.00000 p0t0a2 5.00000 p0t1a0 18.60000
selTms t 0 Msz 2996 Lsz 14980 mrSz 3000 mSz 0 0 0 eLSz 2996 nDel 4 nOldDel 0 nOldDelPr 0.00
genTms t 1 Msz 3290 Lsz 152823 eLSz 2996 nNTms 294
setElTmsMTA eLSz 2996 ss 210 fm 0 minThr 0.40 tminfo t 1 id 2978 gtm 0 phs 0 root 1 sz 5 age 1 nOut 15 0 mnOut p0t0a0 451.33946 p0t0a1 1.00000 p0t0a2 6.00000 p0t1a0 239.2
setElTmsST eLSz 2996 ss 2 fm 0 minThr 0.01 tminfo t 1 id 2913 gtm 0 phs 0 root 1 sz 5 age 1 nOut 15 0 mnOut p0t0a0 139.08607 p0t0a1 1.00000 p0t0a2 2.00000 p0t1a0 8.00000
setElTmsST eLSz 2996 ss 1 fm 0 minThr 0.00 tminfo t 1 id 1804 gtm 0 phs 0 root 1 sz 5 age 1 nOut 15 0 mnOut p0t0a0 131.78725 p0t0a1 1.00000 p0t0a2 8.00000 p0t1a0 298.8000
setElTmsST eLSz 2996 ss 0 fm 0 minThr 1.00 tminfo t 1 id 233 gtm 0 phs 0 root 1 sz 5 age 1 nOut 15 0 mnOut p0t0a0 927.80916 p0t0a1 1.00000 p0t0a2 5.00000 p0t1a0 18.60000
selTms t 1 Msz 2996 Lsz 13853 mrSz 3290 mSz 0 0 0 eLSz 2996 nDel 294 nOldDel 284 nOldDelPr 0.97
genTms t 2 Msz 3290 Lsz 141733 eLSz 2996 nNTms 294
setElTmsMTA eLSz 2996 ss 210 fm 0 minThr 0.40 tminfo t 2 id 2978 gtm 0 phs 0 root 1 sz 5 age 2 nOut 15 0 mnOut p0t0a0 451.33946 p0t0a1 1.00000 p0t0a2 6.00000 p0t1a0 239.2
setElTmsST eLSz 2996 ss 2 fm 0 minThr 0.40 tminfo t 2 id 3502 gtm 2 phs 0 root 1 sz 5 age 0 nOut 15 0 mnOut p0t0a0 445.85176 p0t0a1 1.00000 p0t0a2 4.00000 p0t1a0 8.00000
setElTmsST eLSz 2996 ss 1 fm 0 minThr 0.00 tminfo t 2 id 1804 gtm 0 phs 0 root 1 sz 5 age 2 nOut 15 0 mnOut p0t0a0 131.78725 p0t0a1 1.00000 p0t0a2 8.00000 p0t1a0 298.8000
setElTmsST eLSz 2996 ss 0 fm 0 minThr 1.00 tminfo t 2 id 233 gtm 0 phs 0 root 1 sz 5 age 2 nOut 15 0 mnOut p0t0a0 927.80916 p0t0a1 1.00000 p0t0a2 5.00000 p0t1a0 18.60000
selTms t 2 Msz 2996 Lsz 12799 mrSz 3290 mSz 0 0 0 eLSz 2996 nDel 294 nOldDel 278 nOldDelPr 0.95
genTms t 3 Msz 3290 Lsz 130973 eLSz 2996 nNTms 294
setElTmsMTA eLSz 2996 ss 210 fm 0 minThr 0.40 tminfo t 3 id 2978 gtm 0 phs 0 root 1 sz 5 age 3 nOut 15 0 mnOut p0t0a0 451.33946 p0t0a1 1.00000 p0t0a2 6.00000 p0t1a0 239.2
setElTmsST eLSz 2996 ss 2 fm 0 minThr 0.40 tminfo t 3 id 3502 gtm 2 phs 0 root 1 sz 5 age 1 nOut 15 0 mnOut p0t0a0 445.85176 p0t0a1 1.00000 p0t0a2 4.00000 p0t1a0 8.00000
setElTmsST eLSz 2996 ss 1 fm 0 minThr 0.00 tminfo t 3 id 1804 gtm 0 phs 0 root 1 sz 5 age 3 nOut 15 0 mnOut p0t0a0 131.78725 p0t0a1 1.00000 p0t0a2 8.00000 p0t1a0 298.8000
setElTmsST eLSz 2996 ss 0 fm 0 minThr 1.00 tminfo t 3 id 233 gtm 0 phs 0 root 1 sz 5 age 3 nOut 15 0 mnOut p0t0a0 927.80916 p0t0a1 1.00000 p0t0a2 5.00000 p0t1a0 18.60000
selTms t 3 Msz 2996 Lsz 11753 mrSz 3290 mSz 0 0 0 eLSz 2996 nDel 294 nOldDel 276 nOldDelPr 0.94
genTms t 4 Msz 3290 Lsz 120383 eLSz 2996 nNTms 294
setElTmsMTA eLSz 2996 ss 210 fm 0 minThr 0.40 tminfo t 4 id 2978 gtm 0 phs 0 root 1 sz 5 age 4 nOut 15 0 mnOut p0t0a0 451.33946 p0t0a1 1.00000 p0t0a2 6.00000 p0t1a0 239.2
setElTmsST eLSz 2996 ss 2 fm 0 minThr 0.40 tminfo t 4 id 3502 gtm 2 phs 0 root 1 sz 5 age 2 nOut 15 0 mnOut p0t0a0 445.85176 p0t0a1 1.00000 p0t0a2 4.00000 p0t1a0 8.00000
setElTmsST eLSz 2996 ss 1 fm 0 minThr 0.00 tminfo t 4 id 1804 gtm 0 phs 0 root 1 sz 5 age 4 nOut 15 0 mnOut p0t0a0 131.78725 p0t0a1 1.00000 p0t0a2 8.00000 p0t1a0 298.8000
setElTmsST eLSz 2996 ss 0 fm 0 minThr 1.00 tminfo t 4 id 233 gtm 0 phs 0 root 1 sz 5 age 4 nOut 15 0 mnOut p0t0a0 927.80916 p0t0a1 1.00000 p0t0a2 5.00000 p0t1a0 18.60000
selTms t 4 Msz 2996 Lsz 10786 mrSz 3290 mSz 0 0 0 eLSz 2996 nDel 294 nOldDel 268 nOldDelPr 0.91
genTms t 5 Msz 3290 Lsz 111033 eLSz 2996 nNTms 294
```

Example **.std** file output in TPG

Pain Point #3

Onboarding Hurdles

Tangled Program Graphs (TPG)

This code reproduces results from the paper:

Stephen Kelly, Tatiana Voegerl, Wolfgang Banzhaf, and Cedric Gondro. Evolving Hierarchical Memory-Prediction Machines in Multi-Task Reinforcement Learning. Genetic Programming and Evolvable Machines, 2021. pdf

Quick Start

1. Setup Linux Server

This code is designed to be used in Linux. If you use Windows, you can use Windows Subsystem for Linux (WSL). You can work with WSL in Visual Studio Code by following this tutorial.

1. Install required software

From the tpg directory run:

2. Manually install dependencies

```
sudo xargs --arg-file requirements.txt apt install
```

2. Set environment variables

In order to easily access tpg scripts, we must add appropriate folders to the $PATH environment variable. To do so, add the following to *~/.profile*

```
export TPG=<YOUR_PATH_HERE>/tpg
export PATH=$PATH:$TPG/scripts/plot
export PATH=$PATH:$TPG/scripts/run
```

3. Set env vars

Then run:

```
source ~/.profile
```

3. Compile

From the tpg directory run:

```
scons --opt
```

4. Build project

```
markcruz@DESKTOP-BBOM7S8:~/tpg/src$ cmake -B build -S . -DCMAKE_BUILD_TYPE=Release
 --build build --config Release

-- The C compiler identification is GNU 13.1.0
-- The CXX compiler identification is GNU 13.1.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found MPI_C: /usr/lib/x86_64-linux-gnu/openmpi/lib/libmpi.so (found version "3.1")
-- Found MPI_CXX: /usr/lib/x86_64-linux-gnu/openmpi/lib/libmpi_cxx.so (found version "3.1")
-- Found MPI: TRUE (found version "3.1")
-- Found Boost: /usr/lib/x86_64-linux-gnu/cmake/Boost-1.74.0/BoostConfig.cmake (found version "1.74.0") found compon
ents: system thread mpi serialization iostreams
-- Found OpenCV: /usr (found version "4.5.4")
-- Found CURL: /usr/lib/x86_64-linux-gnu/libcurl.so (found version "7.81.0")
-- Found OpenGL: /usr/lib/x86_64-linux-gnu/libOpenGL.so
-- Found GLEW: /usr/include (found version "2.2.0")
-- Performing Test HAVE_FLAG__ffile_prefix_map__home_markcruz_tpg_src_build__deps_catch2_src__
-- Performing Test HAVE_FLAG__ffile_prefix_map__home_markcruz_tpg_src_build__deps_catch2_src__ - Success
-- Configuring done
-- Generating done
-- Build files have been written to: /home/markcruz/tpg/src/build
markcruz@DESKTOP-BBOM7S8:~/tpg/src$ cmake --build build --config Release
[  0%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/benchmark/catch_chronometer.cpp.o
[  1%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/benchmark/detail/catch_benchmark_func
tion.cpp.o
[  1%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/benchmark/detail/catch_run_for_at_lea
st.cpp.o
[  2%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/benchmark/detail/catch_stats.cpp.o
[  2%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/generators/catch_generator_exception.
cpp.o
[  3%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/generators/catch_generators.cpp.o
[  4%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/generators/catch_generators_random.cp
p.o
[  4%] Building CXX object _deps/catch2-build/src/CMakeFiles/Catch2.dir/catch2/reporters/catch_reporter_automake.cpp
.o
```

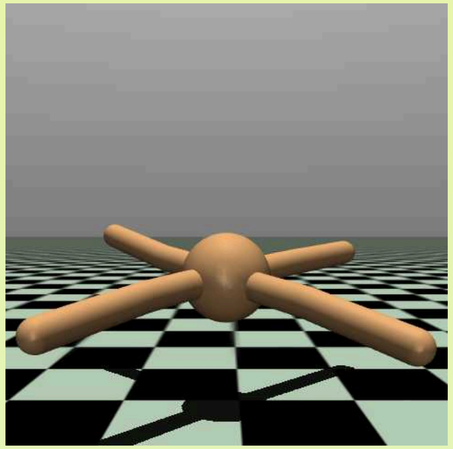
Process of setting up TPG environment

Live Demo

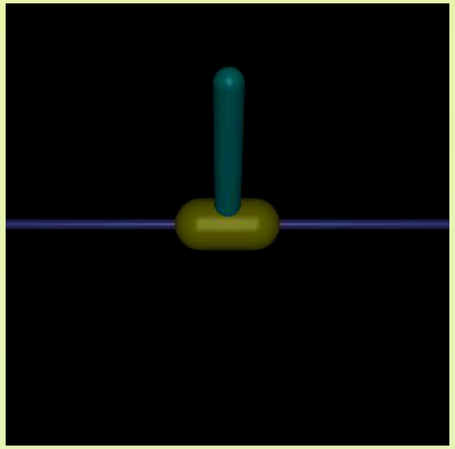
*gifs taken from MuJoCo Documentation

Integrated MuJoCo Environments

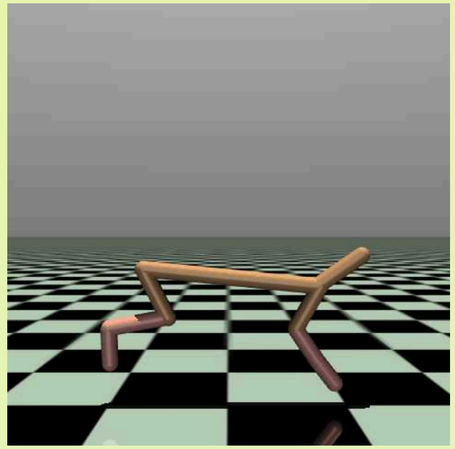
Ant



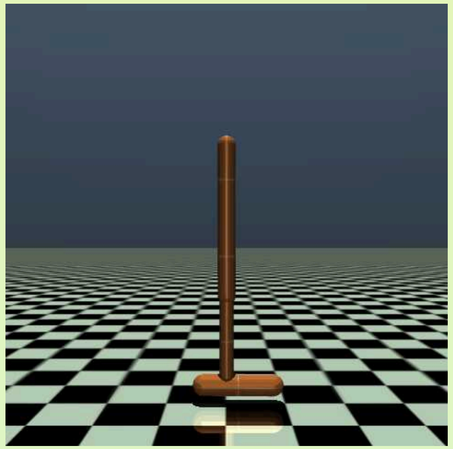
Inverted Pendulum



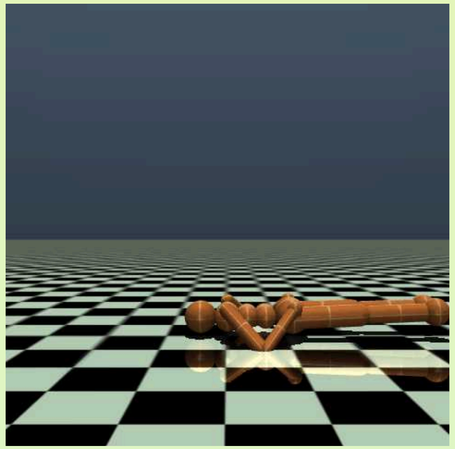
Half Cheetah



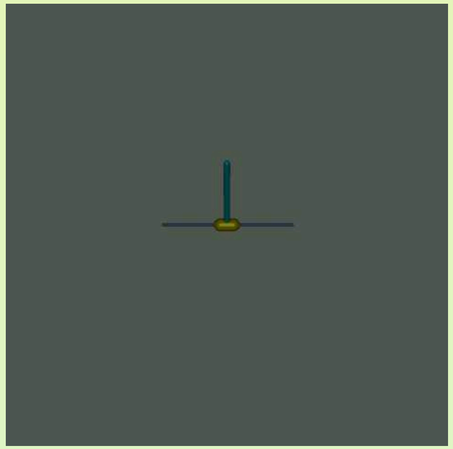
Hopper

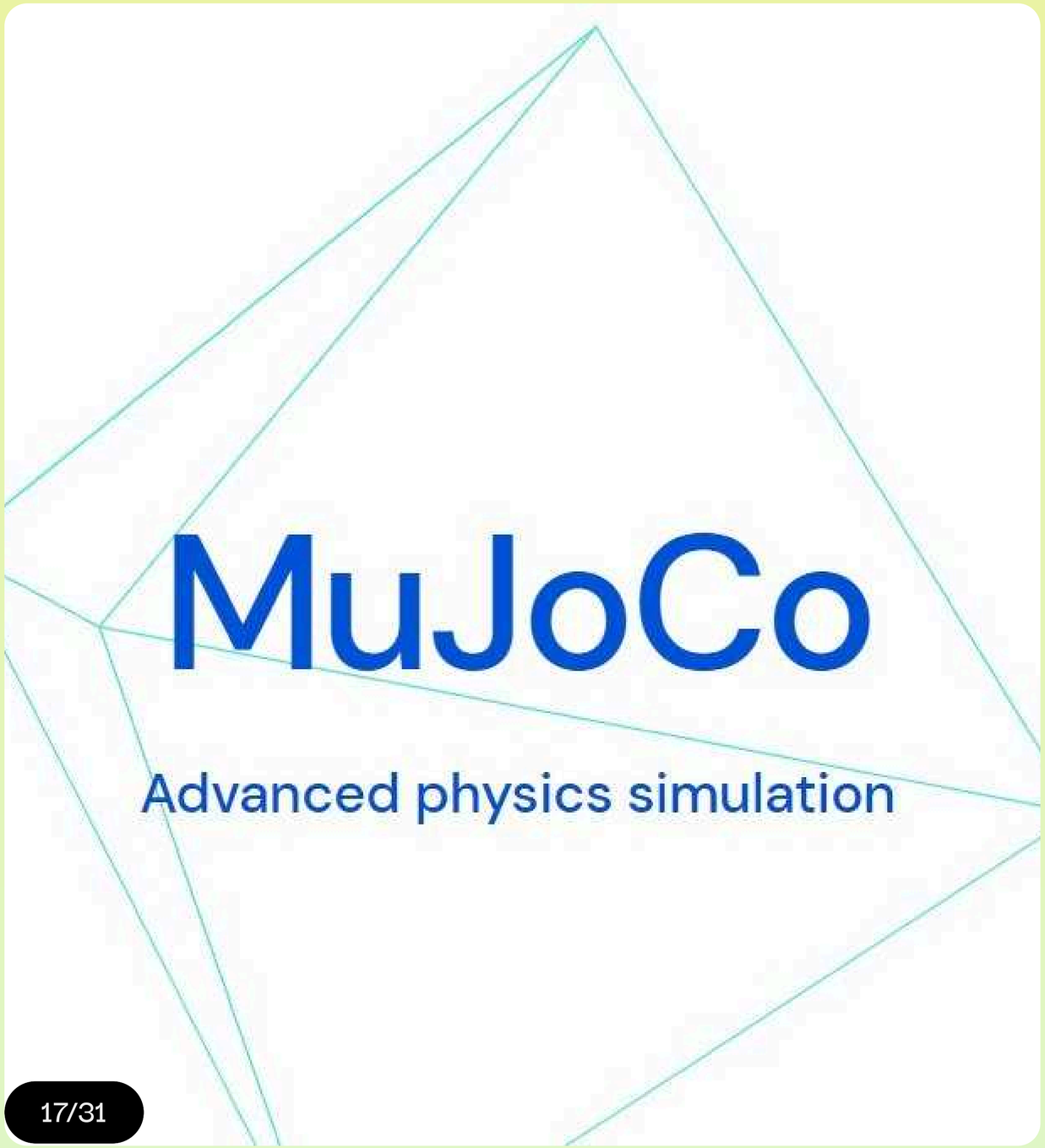


Humanoid Standup



Inverted Double Pendulum





MuJoCo Benefits

1. Open-Source and Actively Maintained
2. Widely Used in Robotics Research
3. Closer to Real-World Robotics Challenges
4. Unlocks New Research Opportunities

Software Engineering Enhancements

1

Command Line Interface (CLI)

2

Revamped Logging System

3

Docker Containerization

Command Line Interface

```
vscode → /workspaces/tpg/src (main) $ cd experiments/inverted_pendulum/
vscode → /workspaces/tpg/src/experiments/inverted_pendulum (main) $ tpg-run-mpi.sh -n 4
vscode → /workspaces/tpg/src/experiments/inverted_pendulum (main) $ tpg-plot-stats.sh
vscode → /workspaces/tpg/src/experiments/inverted_pendulum (main) $ tpg-run-mpi.sh -m 1
vscode → /workspaces/tpg/src/experiments/inverted_pendulum (main) $ cd ../ant
vscode → /workspaces/tpg/src/experiments/ant (main) $ tpg-run-mpi.sh -n 4
vscode → /workspaces/tpg/src/experiments/ant (main) $ tpg-plot-stat.sh
vscode → /workspaces/tpg/src/experiments/ant (main) $ tpg-run-mpi.sh -m 1
vscode → /workspaces/tpg/src/experiments/ant (main) $ cd ../..
vscode → /workspaces/tpg/src/ (main) $
```

VS

```
vscode → /workspaces/tpg/src/ (main) $ tpg evolve inverted_pendulum
vscode → /workspaces/tpg/src/ (main) $ tpg replay inverted_pendulum
vscode → /workspaces/tpg/src/ (main) $ tpg plot inverted_pendulum
vscode → /workspaces/tpg/src/ (main) $ tpg evolve ant
vscode → /workspaces/tpg/src/ (main) $ tpg replay ant
vscode → /workspaces/tpg/src/ (main) $ tpg plot ant
```

WHAT WE IMPROVED ↓

Versatile

Able to evolve, plot, and replay all with one tool

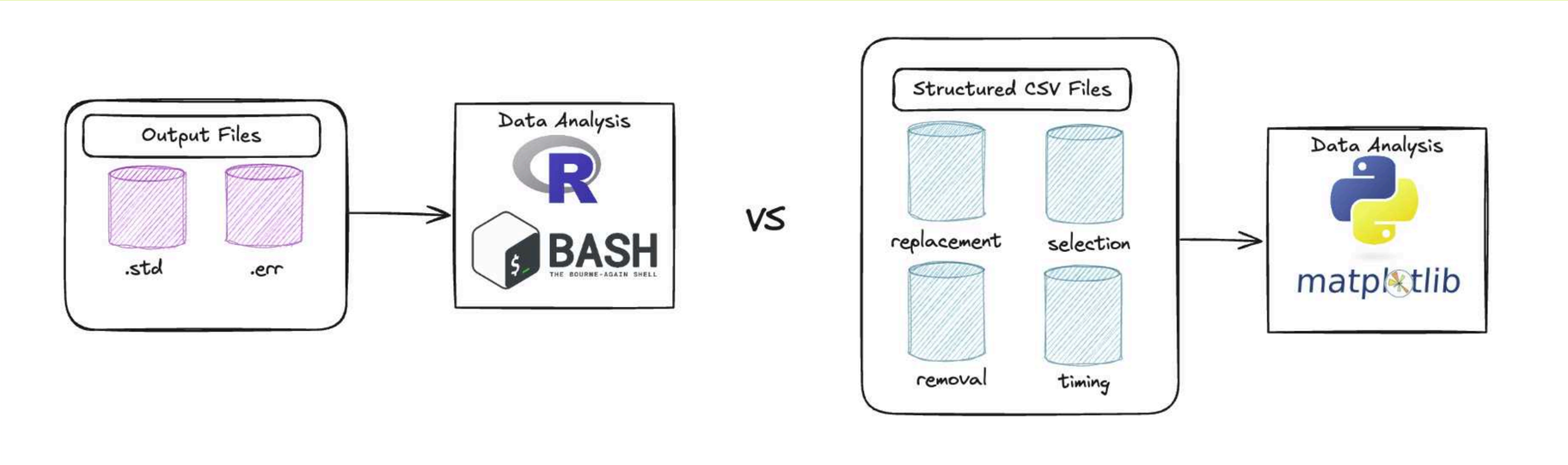
Intuitive

Commands are more descriptive at a glance

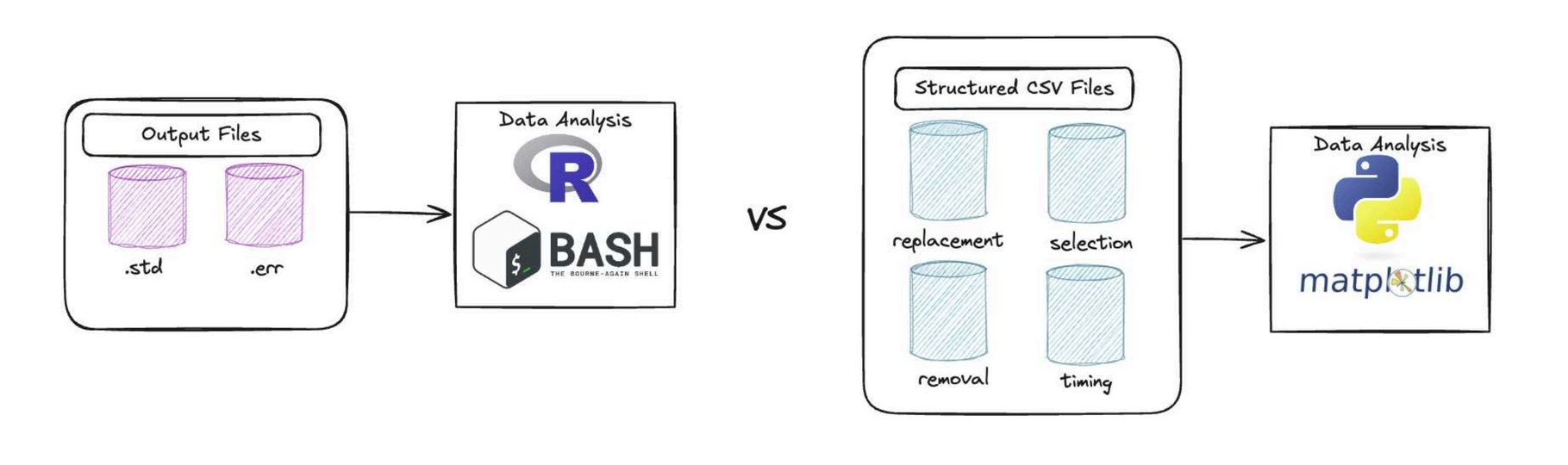
Efficient

Decreased number of commands by ~50%

Revamped Logging System



Revamped Logging System



1 **Clarity**: Researchers can pinpoint key metrics in seconds, eliminating hours of sifting through unstructured .std files

2 **Flexible**: Developers can use Python for advanced analysis instead of being locked into rigid R scripts or shell commands

Containerization

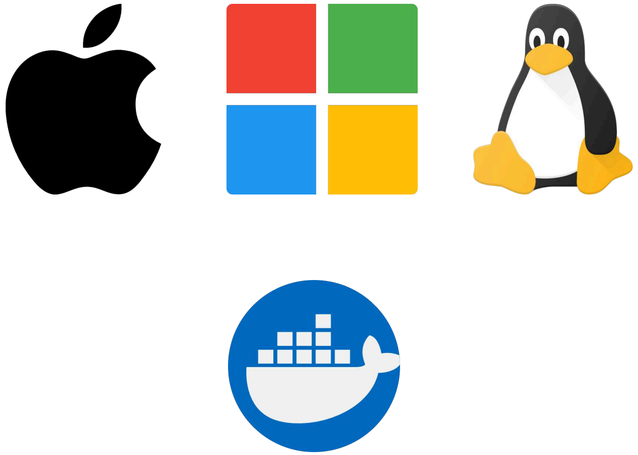
Onboarding Speed

1 week

↓

10 minutes

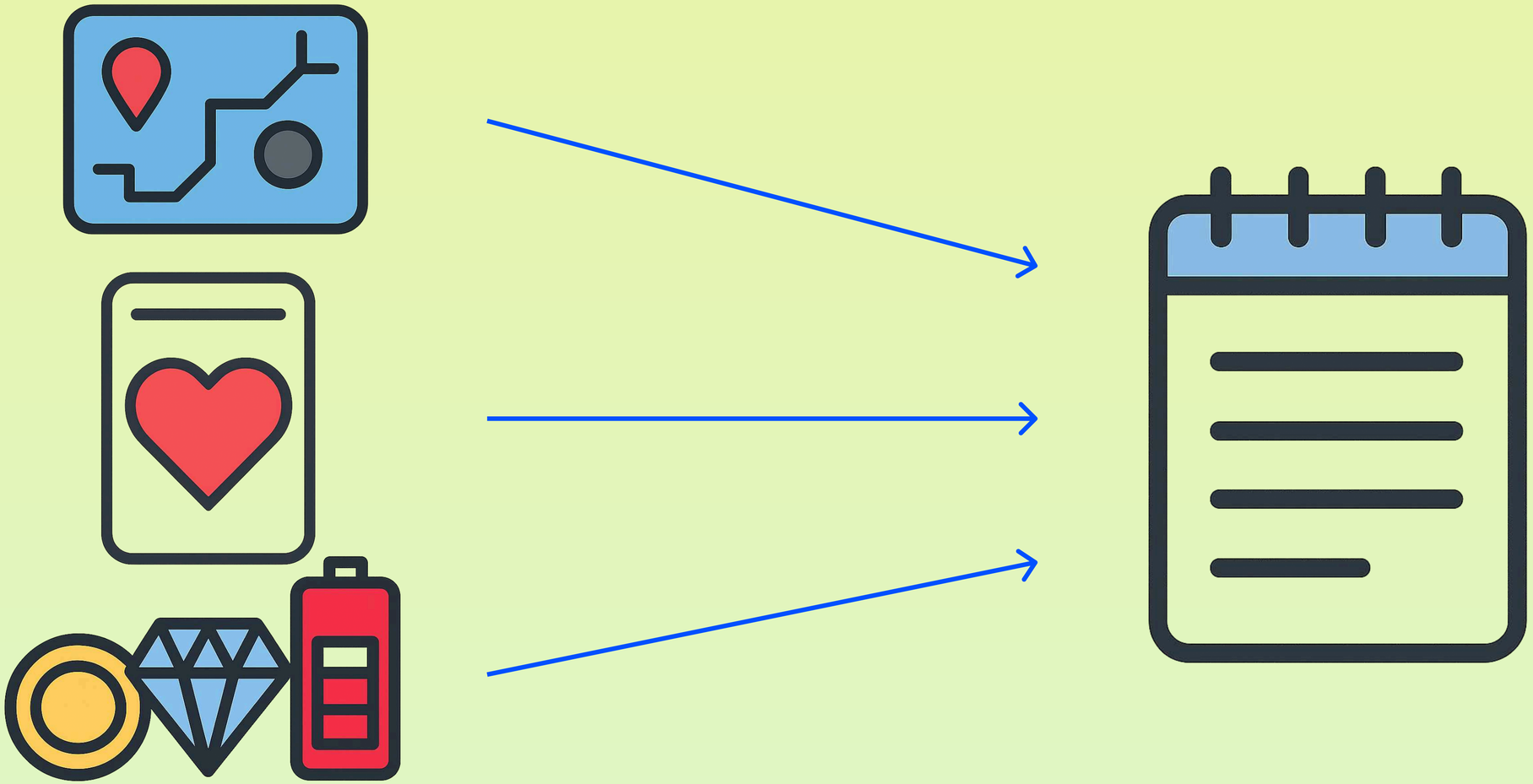
Uniformity



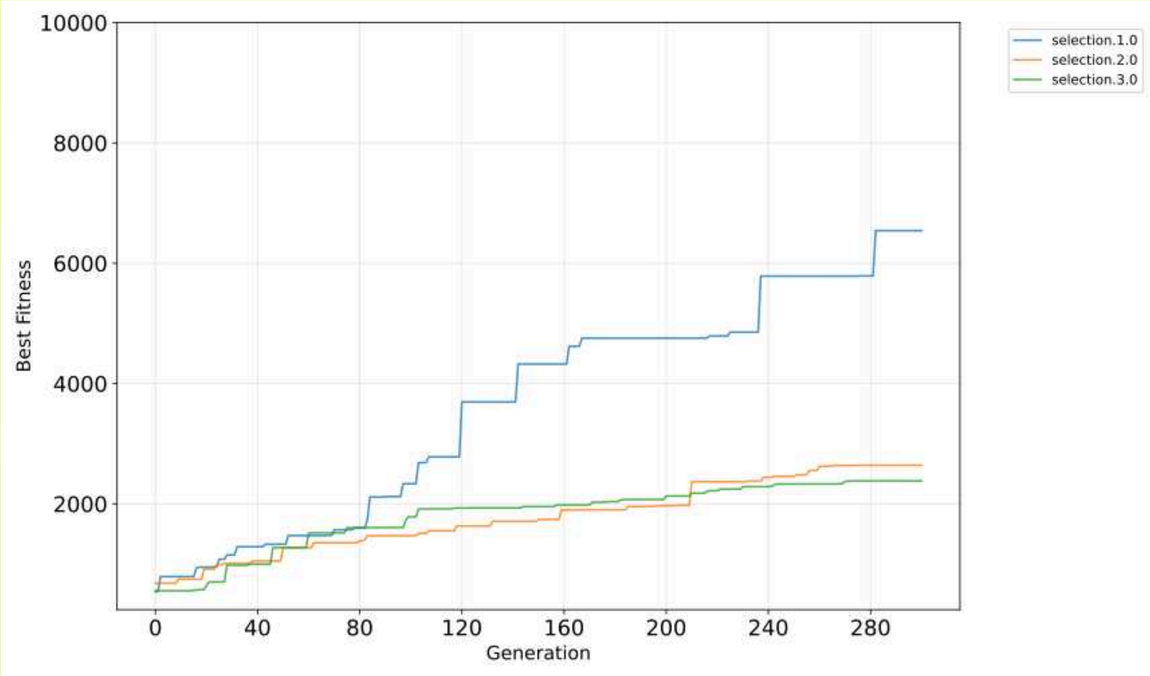
Research Problem

Can we enhance TPG's performance in these complex MuJoCo tasks by using Dynamic Memory?

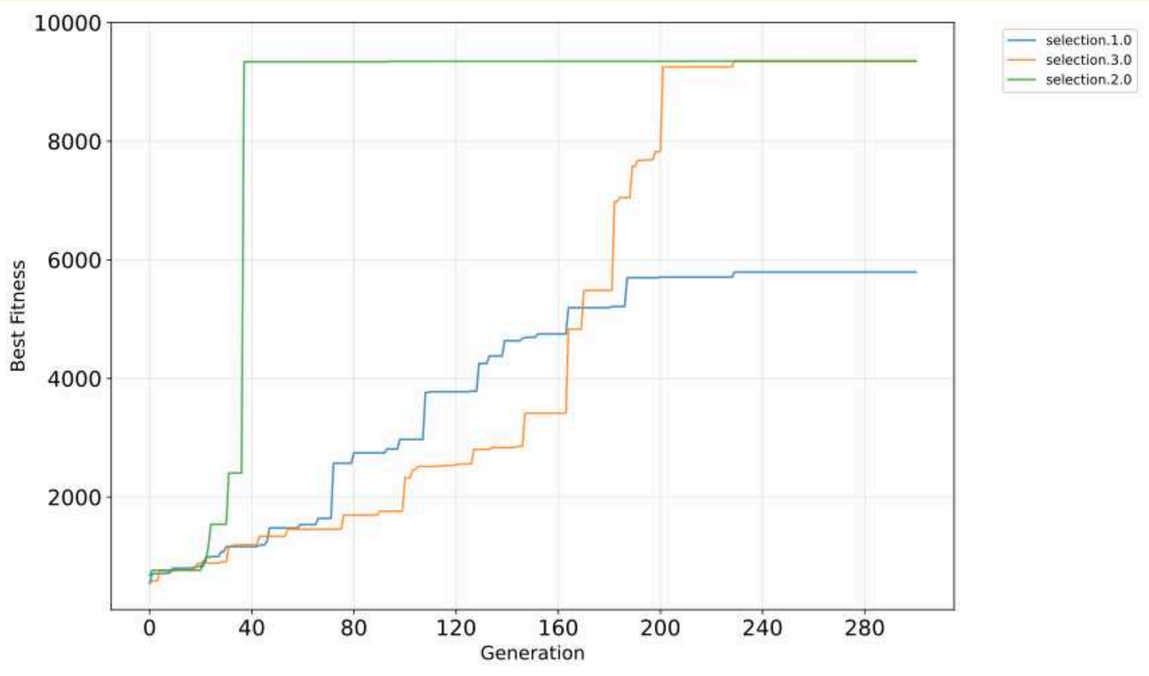
Notepad analogy for Dynamic Memory



Inverted Double Pendulum

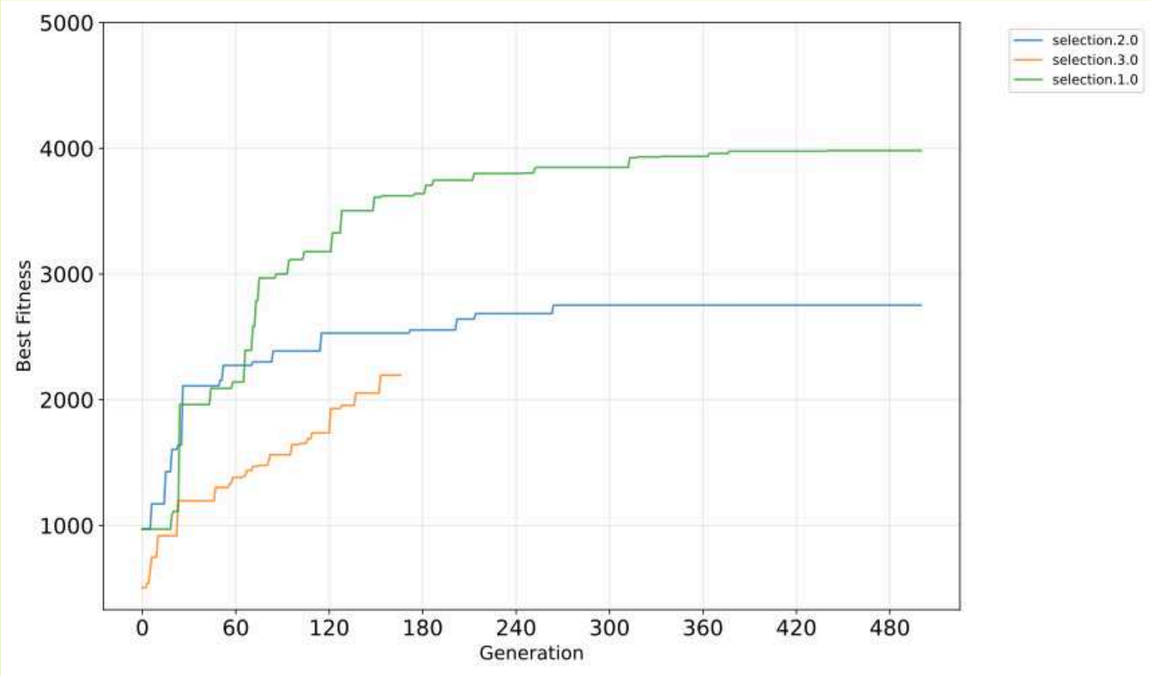


Static Memory

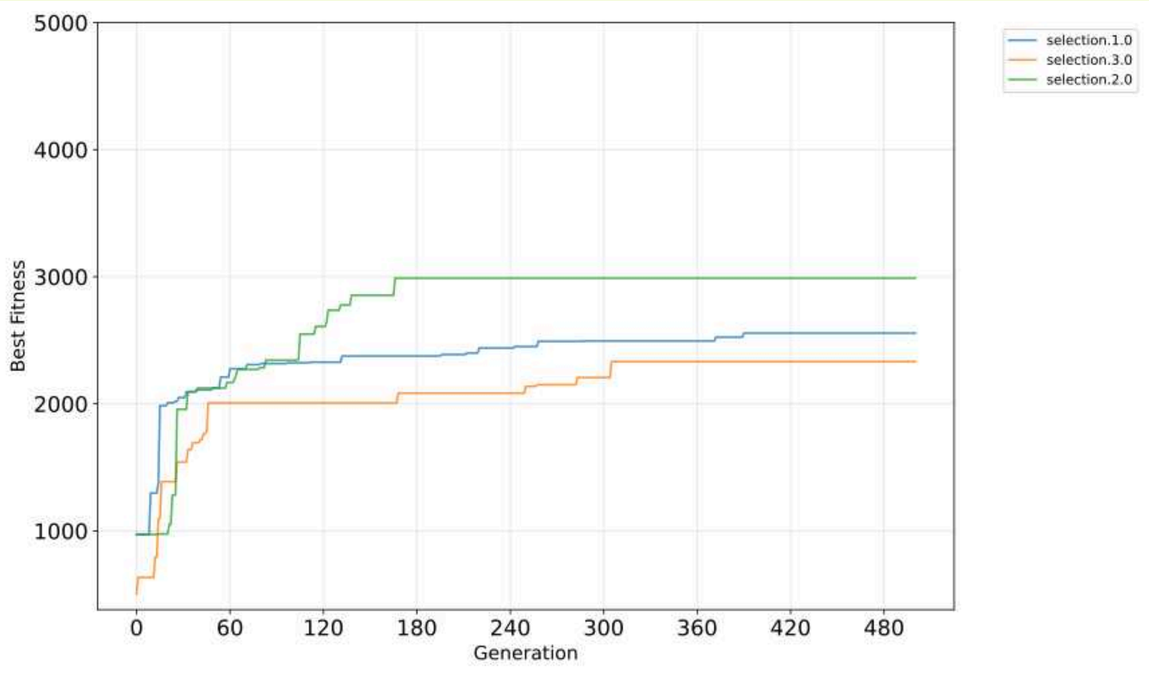


Dynamic Memory

Half Cheetah



Static Memory

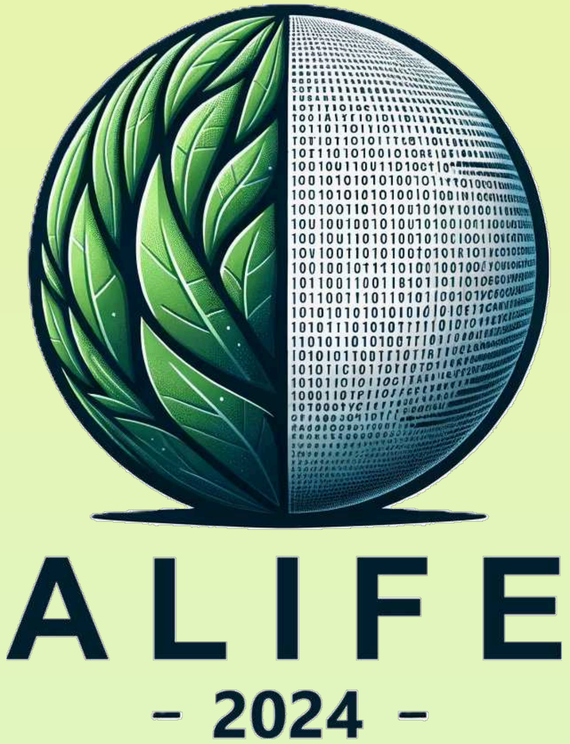


Dynamic Memory

Next Steps

TPG @ GECCO and ALIFE Conferences

Genetic and Evolutionary Computation Conference

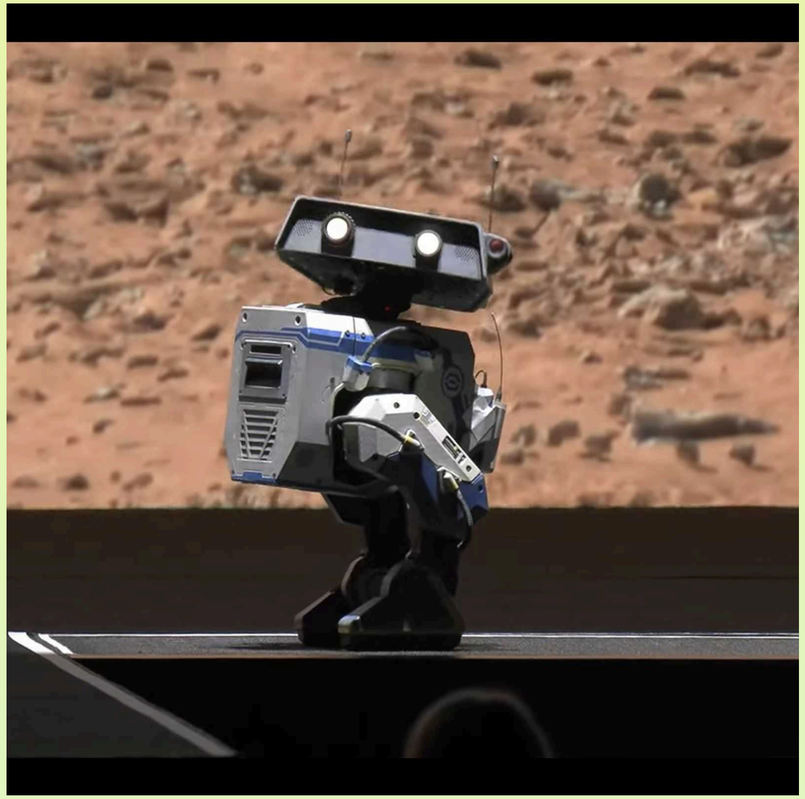


Conference on Artificial Life

Open-Source TPG

GitHub logo (left) and GEGELATI [OS RL framework] logo (right)

Big Vision: Integration to Robotics



NVIDIA's Blue (left) and Figure's Humanoid Robots (right)

Thank you

ANY QUESTIONS?