



Spinning Up in Deep Reinforcement Learning (With PyBullet)



Rive Sunder · Follow

Published in Sorta Sota

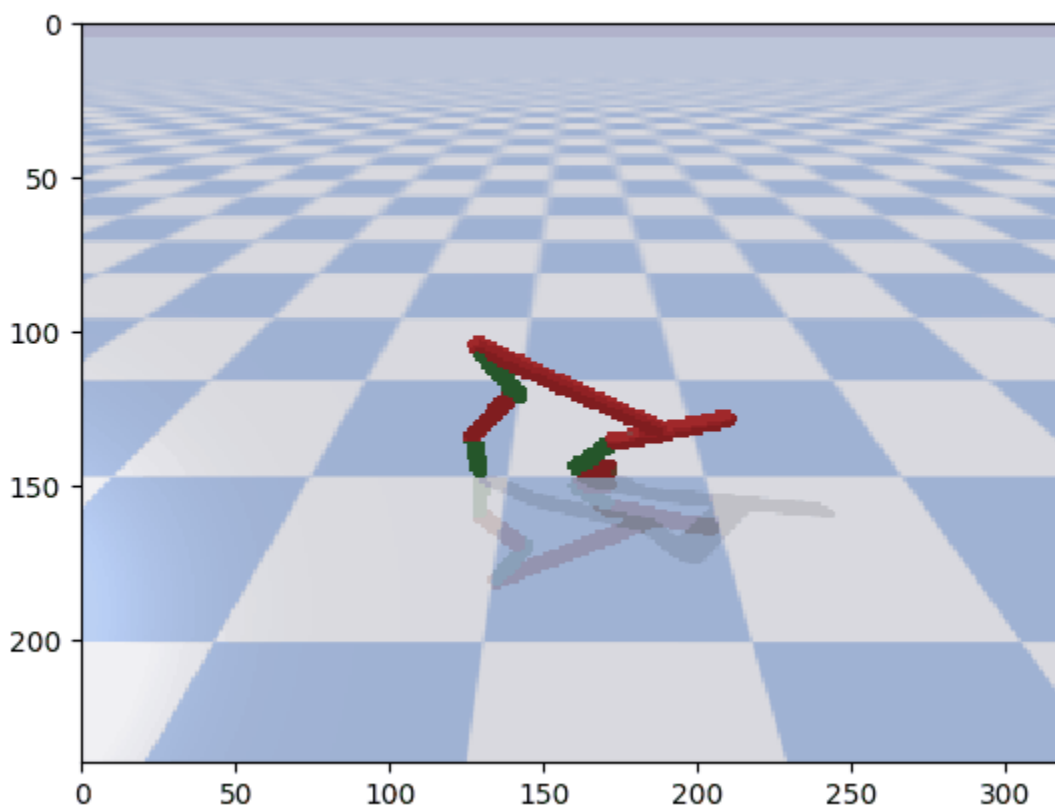
5 min read · Dec 1, 2020



Listen



Share

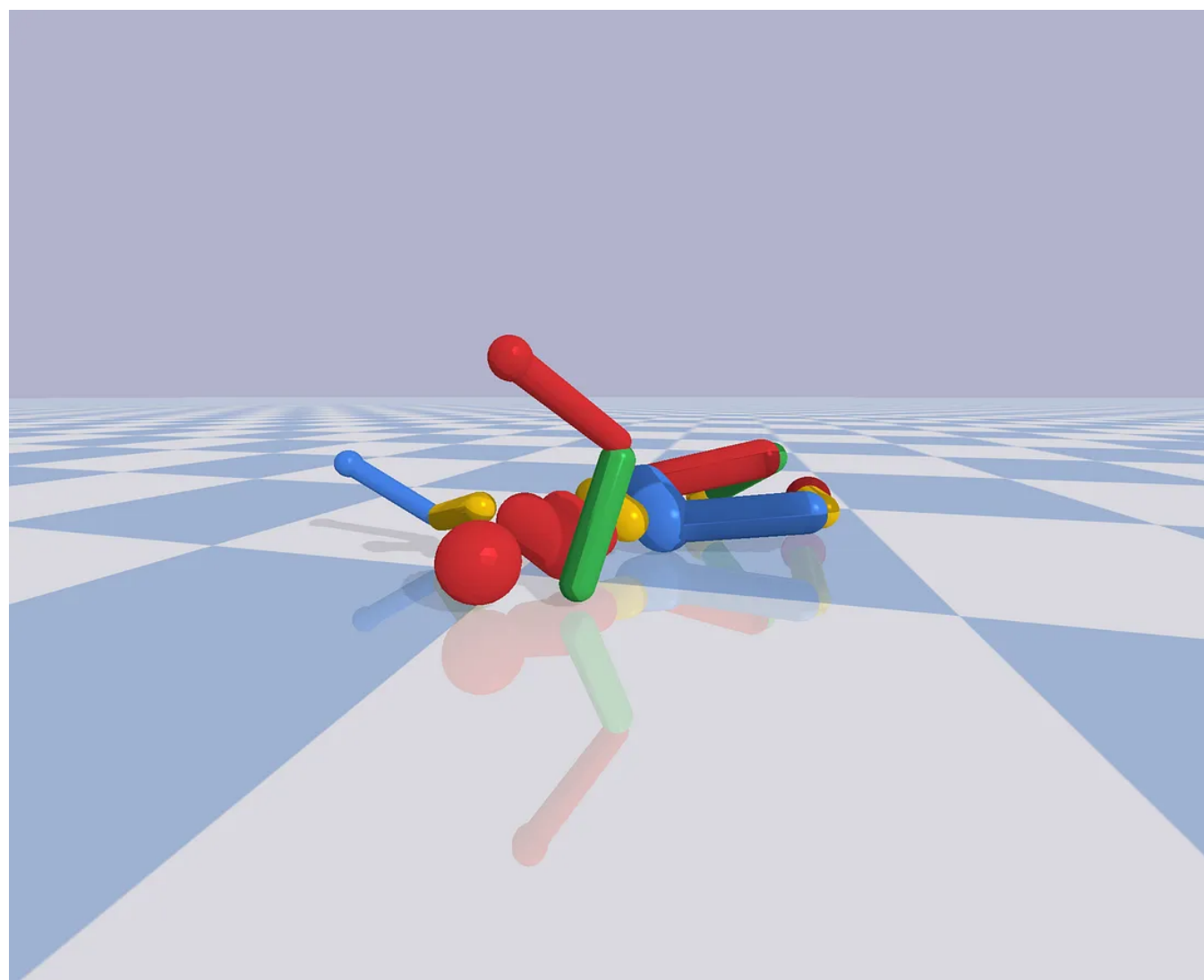


If you want to skip the preamble and get started right away, jump down to Setting Up in Deep RL with PyBullet. You can always come back to read the intro while your experiment

is running. Also, friends fix fading text [here](#).

Perhaps you've decided to study deep reinforcement learning. Maybe you're getting a jump on a New Year's resolution, perhaps it's something you think might help your career, or maybe you just can't find a human that wants to play Go with you.

Whatever your reasons, If you haven't found it already, I'd like to point you toward a useful resource built by Joshua Achiam: [Spinning Up in Deep Reinforcement Learning](#).



The hardest part of using PyBullet for reinforcement learning is getting out of bed.

Spinning Up is great for getting started with running experiments, with implementations and descriptions of several of the most well-known RL algorithms. It also includes several exercises, a bibliography, and an essay of tips and tricks for

conducting good research once you get started. For the most part, the project exudes an open-ness and desire to create an environment conducive to the success of others which I find appealing, or at least that's a general impression I get from this type of project. There's one small problem, and it's one that is increasingly shared by projects originating at OpenAI: the project is built to depend on MuJoCo for continuous control robotics.

MuJoCo is a physics simulator that was originally developed with public funding from the NSF and NIH, but is now available to use with licenses ranging from \$500 US for personal use with no commercial application, to as much as \$2,000 US per person for commercial projects. There is also a 30-day trial and a free student version available, the latter of which is so limited as to be nearly useless (you can't do research, and the license is invalid for classwork where more than 50% of the class uses MuJoCo). There is plenty of worthwhile criticism directed toward OpenAI for their insistence on pushing MuJoCo in their projects. But this isn't going to be a pessimistic post about propping up barriers to entry in reinforcement learning and artificial intelligence research; this post is going to show you how you can use a permissively licensed physics simulator for all the canonical robotics tasks often reported in RL literature, free of charge.

PyBullet provides python bindings to the Bullet physics simulator. Released under a permissive zlib license, it's widely used in games and research. Luckily for us, it also comes with a suite of reinforcement learning environments, including the RL robotics tasks you may have seen before. For a full list of RL environments in PyBullet, check out the Quick Start Guide.

There's no need to re-hash a beginner's intro to RL here. For that, head over to Spinning Up. You'll probably also want to check out a few other of my favorite RL resources at some point: the Berkeley/OpenAI Deep RL Bootcamp from 2017 and Sergey Levine's deep RL courses from Berkeley. If you have your own favorites, feel free to mention them in the comments.

Without further ado, let's get started with PyBullet and Spinning Up. It's actually going to be really easy — if you had skipped reading this introductory fluff you would already have an experiment running ;)

Setting Up in Deep RL with PyBullet

Follow instructions from <https://spinningup.openai.com/en/latest/user/installation.html> to set up a virtual environment for spinningup. Joshua Achiam recommends using [Anaconda](#) to manage virtual environments but I prefer [virtualenv](#).

```
virtualenv spinup_env -- python=python3
source spinup_env/bin/activate
```

Spinning up uses openMPI for parallelization, which requires another install. On Ubuntu:

```
sudo apt-get update && sudo apt-get install libopenmpi-dev
```

Spinning Up documentation gives installation on MacOS X as

```
brew install openmpi
```

With that done you'll be ready to clone and install the repository plus dependencies.

```
git clone https://github.com/openai/spinningup.git
cd spinningup
pip install -e .
```

Spinning Up doesn't have a default robotics/physics simulator out of the box, but there are instructions in the docs for MuJoCo. Instead, we'll install pybullet with pip.

```
pip install pybullet
```

Register PyBullet continuous control robotics by adding a single line to ``./spinup/``
``__init__.py``

```
#append to ./spinup/`__init__.py`  
import pybullet_envs
```

You can then start an experiment with a PyBullet environment to see if you were successful.

```
python -m spinup.run ppo --hid "[32,32]" --env Walker2DBulletEnv-v0  
--exp_name pybullet_test
```

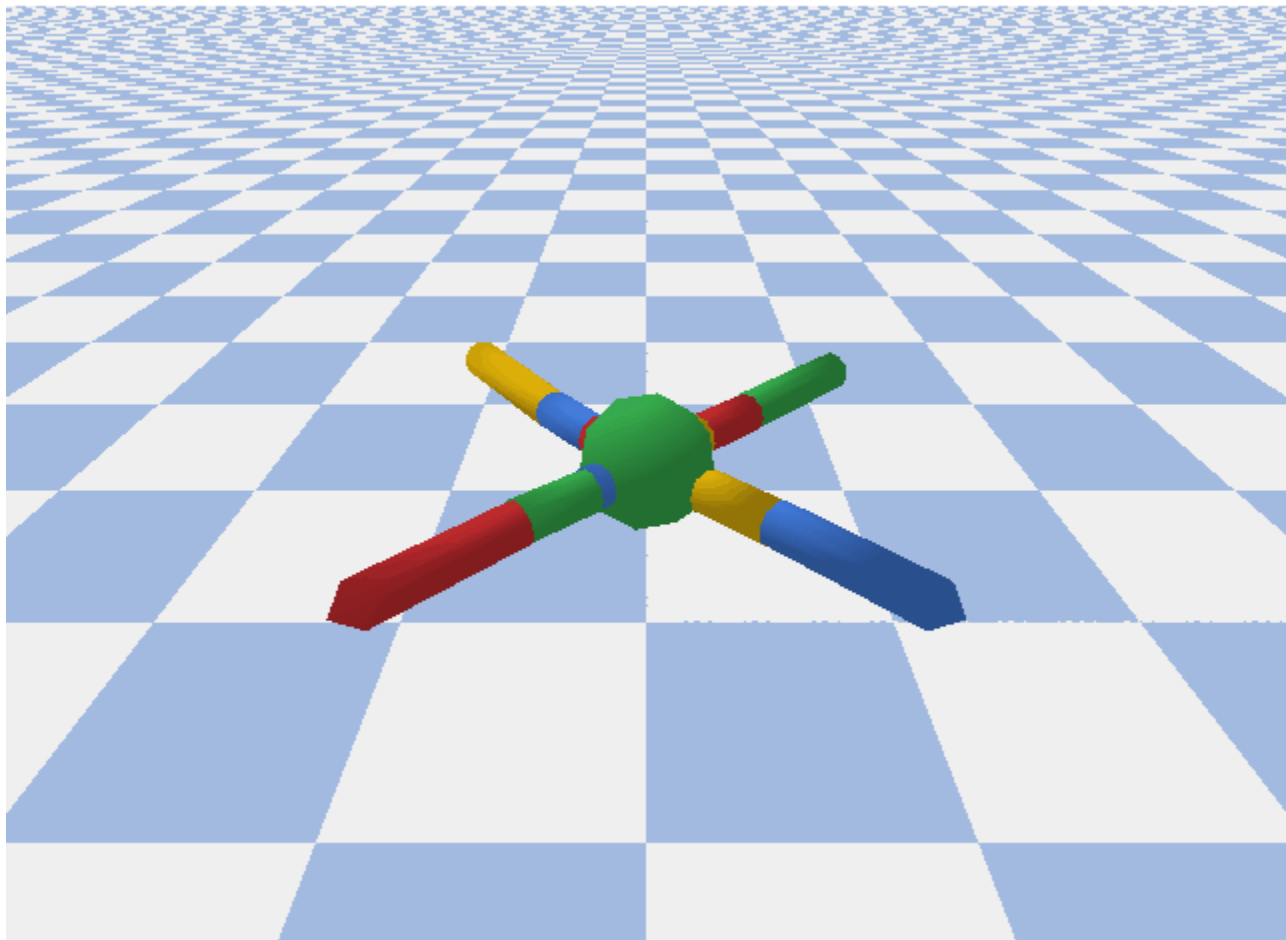
Now, if you want to watch your agent after training, and I think that you will, you'll have to make a few more modifications to the ``test_policy`` utility in Spinning Up. The way PyBullet handles rendering is a little different than other RL environments based on the gym API, *i.e.* you have to call ``env.render()`` *before* calling ``env.reset()``, and you only have to do so once. PyBullet environments also don't play nicely with the way saving and loading environments is implemented in Spinning Up. My workaround is to add a few lines to ``test_policy``. These lines can be placed directly above the line in ``run_policy`` where `env.reset()` is first called.

```
# add after the line 'logger = EpochLogger()'

if "BulletEnv" in env.spec.id
    env = gym.make(env.spec.id)
    if render:
        # pybullet envs have to call env.render before env.reset
        env.render()

# next line should be:
# o, r, d, ep_ret, ep_len, n = env.reset(), 0, False, 0, 0, 0
```

After all this, you should be able to enjoy the efforts of your trained agent. Here's an example of a policy for controlling the PyBullet Ant I trained with Spinning Up's PPO implementation:



Bonus crazy aliasing in the background.

That's it, you're ready to go. Good luck in your explorations with reinforcement learning! Also, note that I've forked Spinning Up and made the changes described in this post, so you can save a lot of time by cloning and installing from my fork if you like.

```
git clone https://github.com/rivesunder/spinningup.git
cd spinningup
```

```
# optional: checkout the commit for this post
#git checkout 017ecbffe5edc72b6b26b73d5d9c989bbf1280d3
```



```
install -e .
```

```
m spinup.run ppo --hid "[32,32]" --env Walker2DBu17
xp_name pybullet_test
```

Follow

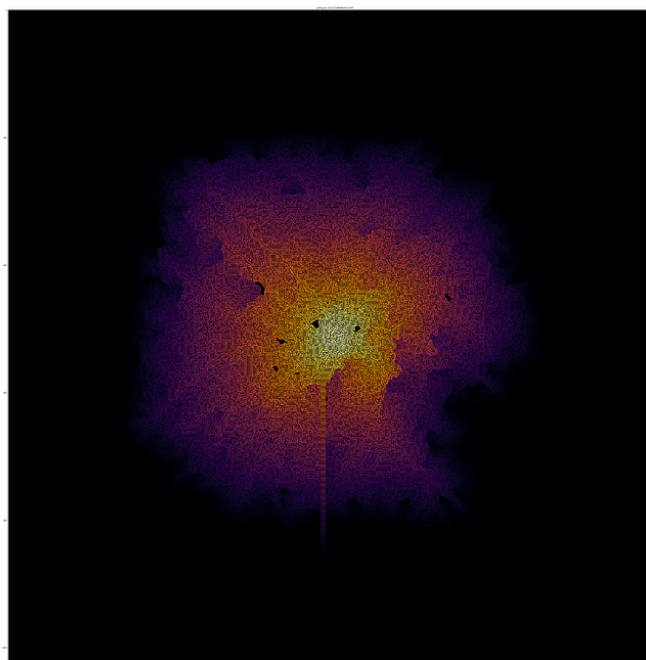
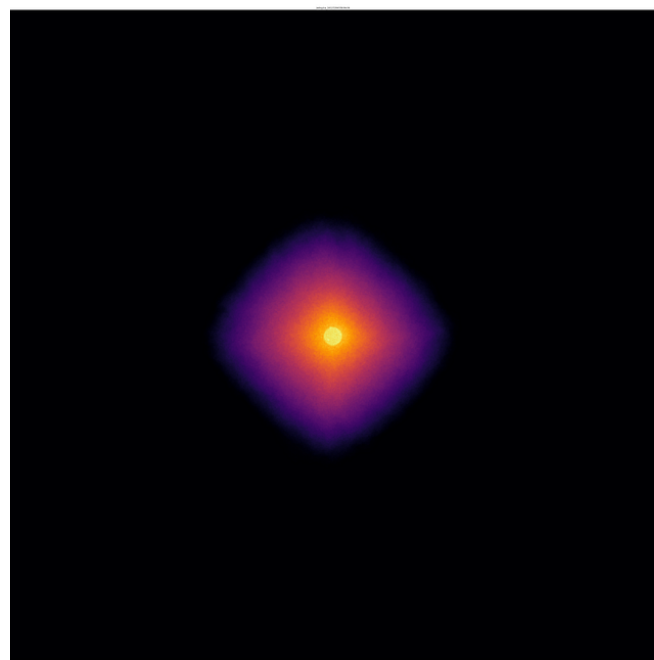



Written by Rive Sunder

30 Followers · Editor for Sorta Sota

Rive Sunder here. I'm an independent scientist and writer working on evolutionary and developmental machine learning, among other things.


More from Rive Sunder and Sorta Sota

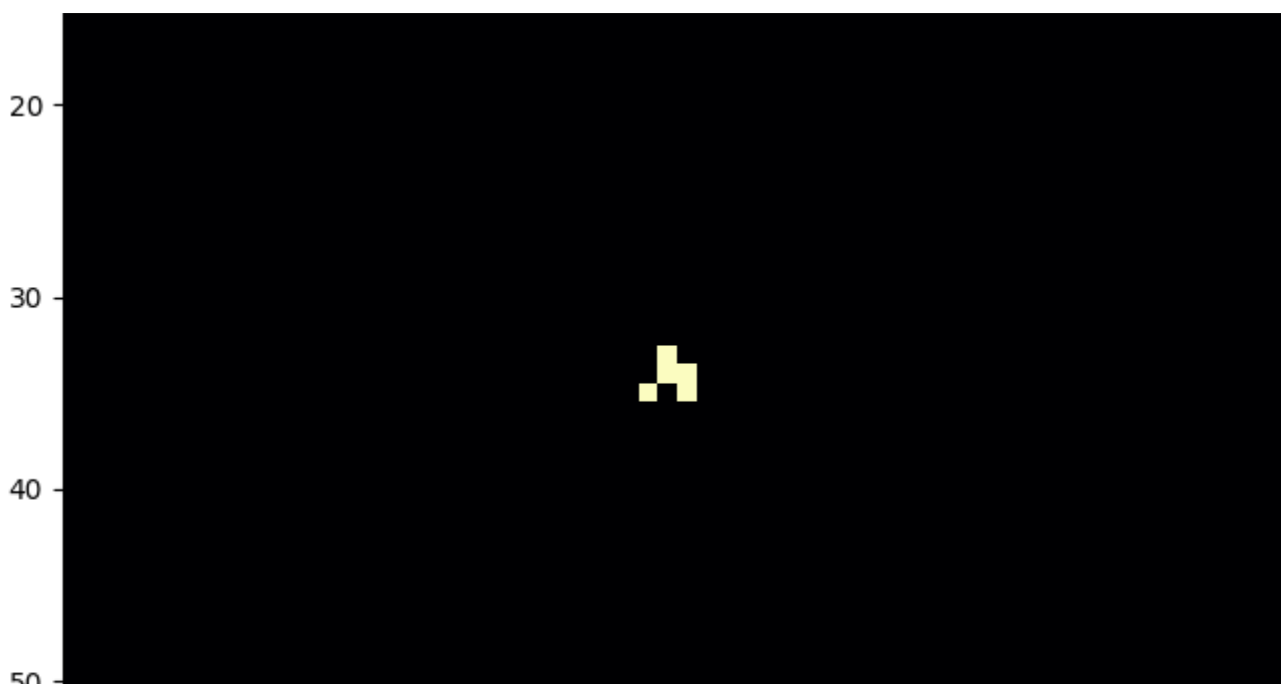


 Rive Sunder in the Scinder

Evaluating Open-Ended Exploration and Creation

I've been working on a reinforcement learning (RL) environment for machine exploration and creativity using Life-like cellular automata...


Feb 11, 2021  73



 Rive Sunder in Sorta Sota

Life-Like Execution Speeds with Julia and PyTorch

Originally published at https://rivesunder.github.io/SortaSota/2021/08/24/life_in_julia.html on August 24, 2021.

Aug 23, 2021  74




```
grid2 = grid

if size(grid2) != size(kernel)
    padded_kernel = pad_to_2d(kernel, size(grid2))
else
    padded_kernel = kernel
end

abs.(FFTW.ifftshift(FFTW.ifft(FFTW.fft(FFTW.fftshift(grid))
    .* FFTW.fft(padded_kernel) ) ))


convolved = round.(FFTW.ifftshift(abs.(FFTW.ifft(
    FFTW.fft(FFTW.fftshift(grid2)) .*
    FFTW.fft(FFTW.fftshift(padded_kernel)) ))) )

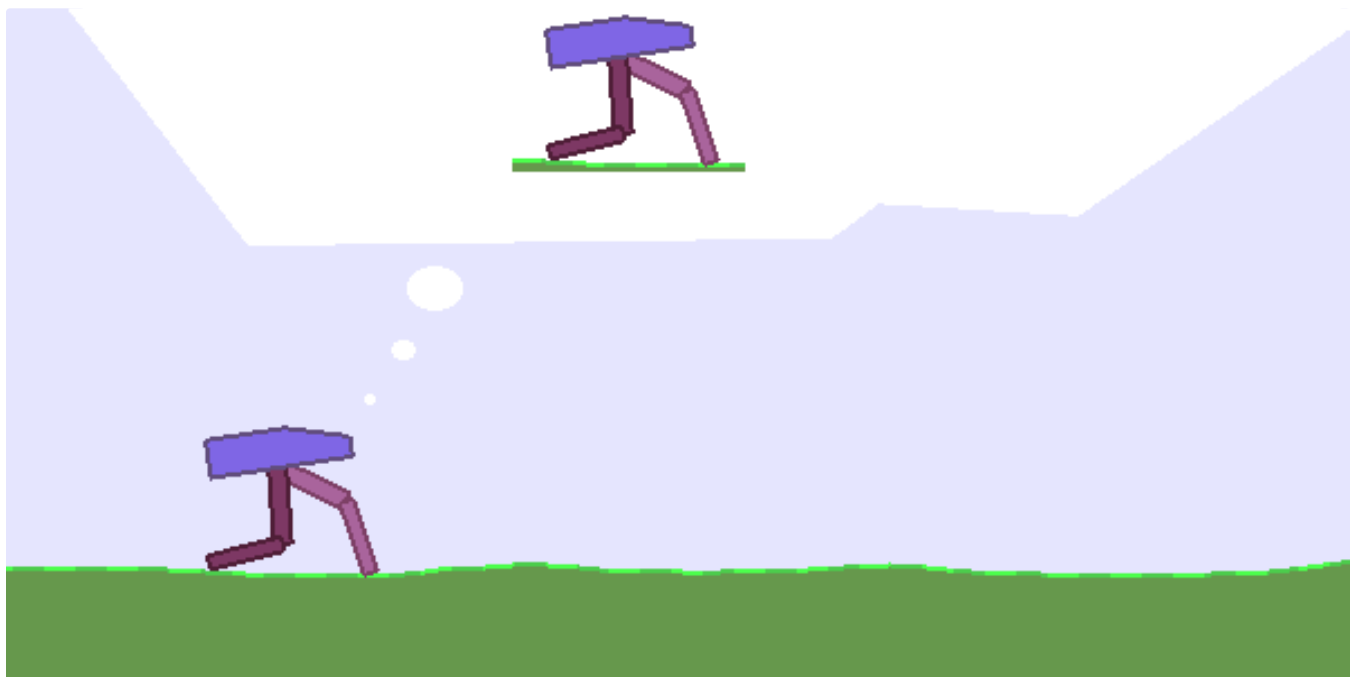
return convolved
```


 Rive Sunder in Sorta Sota

Simulation Speeds for Conway's Game of Life in NumPy vs. Julia

Originally published at https://rivesunder.github.io/SortaSota/2021/08/29/living_julia.html on August 29, 2021


Aug 28, 2021  144



 Rive Sunder in the Scinder

The RL Contest: Threadripper vs. the Cloud, Setup and Benchmarks

This is the second in a series of posts on how it's possible to do interesting and meaningful work in reinforcement or evolutionary...


Apr 16, 2020  52



See all from Rive Sunder

Recommended from Medium See all from Sorta Sota



 d*classified in d*classified

Training Dogfighting Agents with Multi-Agent Reinforcement Learning (Part 1 of 2)

Hey there! Welcome to Part 1 of my Multi-Agent Reinforcement Learning (MARL) series. I'm Ryan Quek, a final-year Industrial Systems...

Oct 9  3  1



Amazon.com

Software Development Engineer

Seattle, WA

Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by \$25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay



Alexander Nguyen in Level Up Coding

The resume that got a software engineer a \$300,000 job at Google.

1-page. Well-formatted.



Jun 1



25K



511



Lists



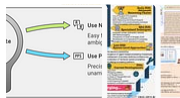
data science and AI

40 stories · 286 saves



Icon Design

36 stories · 445 saves



Natural Language Processing

1816 stories · 1430 saves



Visual Storytellers Playlist

61 stories · 547 saves