Hi all,

First of all, as I'm sure many of you have noticed, it can be cumbersome to get pystk to work on various machines. Here are some tips on how to get set up:

Colab setup:

You must update NVIDIA drivers to get pystk to run properly. Add and run the following 4 lines in your Colab notebook. You will have to answer two questions during the update.

```
!sudo apt-get purge nvidia*
!sudo add-apt-repository ppa:graphics-drivers/ppa -y
!sudo apt-get update
!sudo apt install nvidia-driver-530 nvidia-dkms-530 -y
```

How to install pystk permanently on Colab (from Dustin Hayden):

The lectures for reinforcement learning (and HW5/the final if you don't have a GPU) are going to need you to pip install PySuperTuxKart for every runtime session. That's annoying.

But you can actually download the required documents directly into your google drive and just import from there.

First create a notebook. Put this in the first cell:

```
import os, sys
from google.colab import drive
drive.mount('/content/drive')
nb_path = '/content/notebooks'
path_to_RL = '/content/drive/MyDrive/Colab Notebooks/Deep Learning/Week - 07'
os.symlink(path_to_RL, nb_path)
sys.path.insert(0, nb_path)
```

Run it and allow it to access your drive. Then run this cell:

```
!pip install --target=$nb_path PySuperTuxKart
```

After a few minutes you'll see that, within "/content/drive/MyDrive/Colab Notebooks/Deep Learning/Week - 07" of your google drive and the "/content/notebooks" folder of the current google colab session there will be a ton of packages and what not.

In a new google colab notebook, run that first cell again (to get google colab to know where to look for those packages).

Now you can just run

```
import pystk
```

and not have to pip install it every time.

How to run pystk on an Arm Mac (from Son Quach):

Here I lay out every step I took in order to make it work:

1) Clone the Python SuperTuxKart repository into a directory of choice in your Mac and move to the project root:

git clone https://github.com/philkr/pystk.git cd pystk

2) Open the following file in your favorite text editor:

lib/angelscript/projects/cmake/CMakeLists.txt

3) Comment out line 112

111 if (NOT IOS) 112 #set_property(SOURCE ../../source/as_callfunc_arm_gcc.S APPEND PROPERTY COMPILE_FLAGS " -Wa,-mimplicit-it=always") 113 endif()

4) Open the following file in your favorite text editor:

lib/libpng/pngpriv.h

5) Edit line 174, to change it from:

#if PNG_ARM_NEON_OPT > 0

to:

#if PNG_ARM_NEON_OPT > 2

6) Create a build directory (at top level), navigate into it and compile the sources with the following commands:

mkdir build cd build cmake .. make

Note: you might need to install cmake/make on your Mac if you don't have them already. You can do it with homebrew or other package manager you like.

7) Navigate to the pystk_data directory and fetch the data assets of the game:

cd pystk_data python setup.py fetch_data

8) After doing steps 6 and 7, navigate back to the top-level directory. You will notice a new file, similar to the following:

pystk.cpython-39-darwin.so

You must copy that file into your **site-packages** directory so it becomes accesible by your virtual environment. An example from my system (using anaconda):

/Users/ander/opt/anaconda3/envs/DL/lib/python3.9/site-packages

Similarly, copy the entire pystk_data directory into the site-packages directory.

9) Install a couple auxiliary libraries required to visualize the videos generated by the game:

pip install imageio imageio_ffmpeg

10) Done! Now you can import and use pystk on your Mac M1

How to run pystk on an Arm Mac (alternative):

Use a Rosetta terminal and run everything from there

https://dev.to/courier/tips-and-tricks-to-setup-your-apple-m1-for-development-547g

https://stackoverflow.com/questions/74198234/duplication-of-terminal-in-macos-ventura

Pytorch versions:

Some students have trouble using newer versions of pytorch than the grader. The remote (canvas) grader uses Pytorch version '1.9.0+cu102' and Python 3.9.6. The extra submissions granted for this assignment can also be used to account for the need to downgrade your pytorch version. Please note here that only a few students reported needing to do this last semester, so it may not affect you.

More Homework 5 Tips:

HW5 is almost all about the controller, having a robust controller that can complete the tracks will give you more accurate data so that your model can better predict aim points. The controller can be implemented entirely without Deep Learning/Machine Learning knowledge. Think about how you would design a basic controller given ground truth aim points to get a basic working controller. Then review your videos of your controller and find where it needs improvement, taking into consideration things like:

- Is the kart turning too early/late?
- Is the kart drifting appropriately (i.e. too late/too early/not at all/over drifting)?
- Is the kart going fast enough? Can it go faster without impacting performance?
- Is the kart turning too hard/not hard enough?

Remote vs Local Grader
The remote canvas grader contains an additional level that the local grader does NOT test - therefore, your grades may differ.

Q&As from Summer 2021
Q: Is there a way to make the kart just not require any action for a while? It seems like all actions are implemented immediately back to back. How can I just get this penguin to chill out for a bit?

A: Yes! You can tell the kart to *not* accelerate and/or tell the kart to break

Q: How to find the current state (i.e. point) of the player in the "control" method to figure out what action I need to take according to the input "aim_point".

A: The goal is to steer your kart towards the aim-point.
You tell the controller how to do this by setting the Action
Ex.
if current_vel > target_vel
  action.acceleration = 0

Q:Are we only writing train.py to train for the aim-point detector? We're not using train.py to train the controller (i.e. with gradient free descent)?
A: Yes - that's correct

Lastly, attached is a quick hack from Christopher Hahn to produce a video from the controller in Colab. Change the rollout function in utils.py to the attached code (Make sure to change it back before submitting your solution!). That will generate a video test.mp4 in the current directory when you run the visualization command. Please note that I haven't tested this recently - if you find errors feel free to let the course staff know so that we can update the code for everyone.

```
  def rollout(self, track, controller, planner=None, max_frames=1000,
verbose=False, data_callback=None):
        """
        Play a level (track) for a single round.
        :param track: Name of the track
        :param controller: low-level controller, see controller.py
        :param planner: high-level planner, see planner.py
        :param max_frames: Maximum number of frames to play for
        :param verbose: Should we use matplotlib to show the agent drive?
        :param data_callback: Rollout calls data_callback(time_step, image,
2d_aim_point) every step, used to store the
                                data
        :return: Number of steps played
        """

        import io

        if self.k is not None and self.k.config.track == track:
            self.k.restart()
            self.k.step()
```

```
        else:
            if self.k is not None:
                self.k.stop()
                del self.k
            config = pystk.RaceConfig(num_kart=1, laps=1, track=track)
            config.players[0].controller =
pystk.PlayerConfig.Controller.PLAYER_CONTROL

            self.k = pystk.Race(config)
            self.k.start()
            self.k.step()

        state = pystk.WorldState()
        track = pystk.Track()

        last_rescue = 0

        if verbose:
            import matplotlib.pyplot as plt
            fig, ax = plt.subplots(1, 1)

        frames = []

        for t in range(max_frames):

            state.update()
            track.update()

            kart = state.players[0].kart

            if np.isclose(kart.overall_distance / track.length, 1.0, atol=2e-
3):
                if verbose:
                    print("Finished at t=%d" % t)
                break

            proj = np.array(state.players[0].camera.projection).T
            view = np.array(state.players[0].camera.view).T

            aim_point_world =
self._point_on_track(kart.distance_down_track+TRACK_OFFSET, track)
            aim_point_image = self._to_image(aim_point_world, proj, view)
            if data_callback is not None:
                data_callback(t, np.array(self.k.render_data[0].image),
aim_point_image)

            if planner:
                image = np.array(self.k.render_data[0].image)
                aim_point_image =
planner(TF.to_tensor(image)[None]).squeeze(0).cpu().detach().numpy()

            current_vel = np.linalg.norm(kart.velocity)
            action = controller(aim_point_image, current_vel)
```

```
            if current_vel < 1.0 and t - last_rescue > RESCUE_TIMEOUT:
                last_rescue = t
                action.rescue = True

            if verbose:
                ax.clear()
                ax.imshow(self.k.render_data[0].image)
                WH2 = np.array([self.config.screen_width,
self.config.screen_height]) / 2
                ax.add_artist(plt.Circle(WH2*(1+self._to_image(kart.location,
proj, view)), 2, ec='b', fill=False, lw=1.5))
                ax.add_artist(plt.Circle(WH2*(1+self._to_image(aim_point_worl
d, proj, view)), 2, ec='r', fill=False, lw=1.5))
                if planner:
                    ap = self._point_on_track(kart.distance_down_track +
TRACK_OFFSET, track)
                    ax.add_artist(plt.Circle(WH2*(1+aim_point_image), 2,
ec='g', fill=False, lw=1.5))
                plt.pause(1e-3)

                with io.BytesIO() as buff:
                    fig.savefig(buff, format='raw')
                    buff.seek(0)
                    data = np.frombuffer(buff.getvalue(), dtype=np.uint8)
                w, h = fig.canvas.get_width_height()
                im = data.reshape((int(h), int(w), -1))

                frames.append(im)

            self.k.step(action)
            t += 1

        if verbose:
            import imageio

            imageio.mimwrite("test.mp4", frames, fps=30, bitrate=1000000)
            return t, kart.overall_distance / track.length
```