01b

January 23, 2024

[1]:
```
%pylab inline
import torch
import sys, os
import pystk
device = torch.device('cuda') if torch.cuda.is_available() else torch.
 ↪device('cpu')
print('device = ', device)
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
device =  cuda
```

[2]:
```
class Rollout:
    def __init__(self, screen_width, screen_height, hd=True,␣
 ↪track='lighthouse', render=True):
        # Init supertuxkart
        if not render:
            config = pystk.GraphicsConfig.none()
        elif hd:
            config = pystk.GraphicsConfig.hd()
        else:
            config = pystk.GraphicsConfig.ld()
        config.screen_width = screen_width
        config.screen_height = screen_height
        pystk.init(config)

        self.render = render
        race_config = pystk.RaceConfig(track=track)
        self.race = pystk.Race(race_config)
        self.race.start()

    def __call__(self, agent, n_steps=200):
        self.race.restart()
        self.race.step()
        data = []
        track_info = pystk.Track()
        track_info.update()
        for i in range(n_steps):
            world_info = pystk.WorldState()
            world_info.update()

            # Gather world information
            kart_info = world_info.players[0].kart

            agent_data = {'track_info': track_info, 'kart_info': kart_info}
            if self.render:
                agent_data['image'] = np.array(self.race.render_data[0].image)

            # Act
            action = agent(**agent_data)
            agent_data['action'] = action

            # Take a step in the simulation
            self.race.step(action)

            # Save all the relevant data
            data.append(agent_data)
        return data
```

[3]:
```
def dummy_agent(**kwargs):
    action = pystk.Action()
    action.acceleration = 1
    return action
```

[4]:
```
rollout = Rollout(800, 600)

def rollout_many(many_agents, **kwargs):
    for agent in many_agents:
        yield rollout(agent, **kwargs)

data = rollout(dummy_agent)
```

[5]:
```
def show_video(frames, fps=30):
    import imageio
    from IPython.display import Video, display

    imageio.mimwrite('/tmp/test.mp4', frames, fps=fps, bitrate=10000000)
    display(Video('/tmp/test.mp4', width=800, height=600, embed=True))

show_video([d['image'] for d in data])
```

```
IMAGEIO FFMPEG_WRITER WARNING: input image is not divisible by
macro_block_size=16, resizing from (800, 600) to (800, 608) to ensure video
```

```
compatibility with most codecs and players. To prevent resizing, make your input
image divisible by the macro_block_size or set the macro_block_size to 1
(risking incompatibility).
```

```
<IPython.core.display.Video object>
```

[6]:
```
def show_agent(agent, **kwargs):
    data = rollout(agent, **kwargs)
    show_video([d['image'] for d in data])
```

[7]:
```
def less_dummy_agent(**kwargs):
    action = pystk.Action()
    action.acceleration = 1
    action.steer = -0.6
    return action
show_agent(less_dummy_agent)
```

```
IMAGEIO FFMPEG_WRITER WARNING: input image is not divisible by
macro_block_size=16, resizing from (800, 600) to (800, 608) to ensure video
compatibility with most codecs and players. To prevent resizing, make your input
image divisible by the macro_block_size or set the macro_block_size to 1
(risking incompatibility).
```

```
<IPython.core.display.Video object>
```

[ ]:
```
# Let's load a fancy auto-pilot. You'll write one yourself in your homework.
from _auto_pilot import auto_pilot

show_agent(auto_pilot, n_steps=600)
```

[ ]:
```

```