CARRACING-V0

SETUP

1. Instale o Pycharm com Python 3.6
2. Crie um novo projeto em File > New Project
3. Apague o que estiver no arquivo main.py
4. Copie o código fonte do CarRacing-v0 para o arquivo main. Código está disponível em (https://github.com/openai/gym/blob/master/gym/envs/box2d/car_racing.py)
5. Instale as bibliotecas:
    a. gym
    b. pickle-mixin
    c. torch
    d. Box2D
    e. Numpy
    f. copy

- Para fazer esta instalação no PyCharm, acesse File > Settings
- Na janela que se abre, selecione na barra da esquerda a opção Project > Project Interpreter
- Na nova janela, logo acima de “packages” tem um sinal de “+”
- Clique no “+” e procure as bibliotecas listadas acima (de “a” a “f”)
- Uma vez encontradas, as bibliotecas precisam ser instaladas uma a uma clicando em “install package”

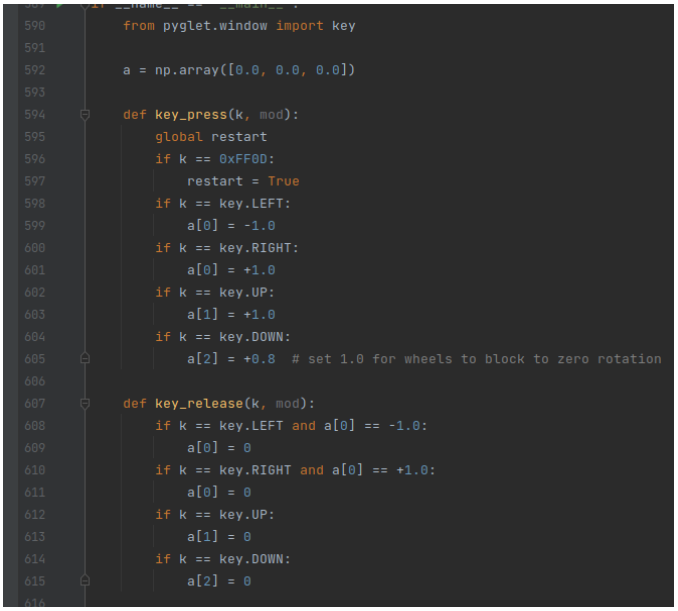
6. No menu “Run” click em “run main”

- Uma janela com o jogo em modo manual deve se abrir. Controle o carro com as teclas direcionais do teclado

NOTAS SOBRE O CÓDIGO FONTE

Manual control

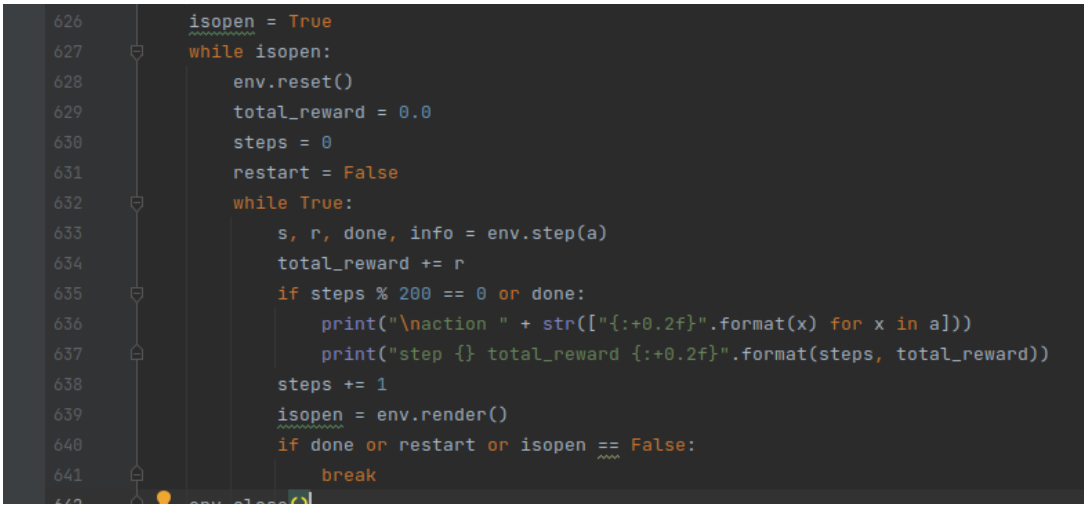
A implementação do controle manual está nas linhas 598 - 615. Esta implementação gera um numpy array com 3 posições chamado **a**, que codifica as ações que serão inputs no loop principal do sistema.



```
590     from pyglet.window import key
591
592     a = np.array([0.0, 0.0, 0.0])
593
594     def key_press(k, mod):
595         global restart
596         if k == 0xFF0D:
597             restart = True
598         if k == key.LEFT:
599             a[0] = -1.0
600         if k == key.RIGHT:
601             a[0] = +1.0
602         if k == key.UP:
603             a[1] = +1.0
604         if k == key.DOWN:
605             a[2] = +0.8  # set 1.0 for wheels to block to zero rotation
606
607     def key_release(k, mod):
608         if k == key.LEFT and a[0] == -1.0:
609             a[0] = 0
610         if k == key.RIGHT and a[0] == +1.0:
611             a[0] = 0
612         if k == key.UP:
613             a[1] = 0
614         if k == key.DOWN:
615             a[2] = 0
```

Simulation

O loop principal do jogo está nas linhas 627-641:



```
626         isopen = True
627         while isopen:
628             env.reset()
629             total_reward = 0.0
630             steps = 0
631             restart = False
632             while True:
633                 s, r, done, info = env.step(a)
634                 total_reward += r
635                 if steps % 200 == 0 or done:
636                     print("\naction " + str(["{:+0.2f}".format(x) for x in a]))
637                     print("step {} total_reward {:+0.2f}".format(steps, total_reward))
638                 steps += 1
639                 isopen = env.render()
640                 if done or restart or isopen == False:
641                     break
```

Linha 628 ⇒ reseta o jogo e cria o estado inicial (s_0)
[*Hint:* fazendo s_prev = env.reset() é possível armazenar o estado inicial]

Linha 633 ⇒ Com base no estado s_t o jogador toma uma ação a_t
A ação entra no ambiente com env.step(a) e gera novo estado (s), dentre outros

Trajectories saving

- Para armazenar as trajetórias que servirão como conjunto de treinamento para o modelo é possível fazer, por exemplo, uma lista de pares de estado/ação.
- Esta lista pode ser elaborada editando diretamente o código fonte.
- Para cada passo da simulação, adicione o estado s_prev e a ação tomada naquele estado
- *Se tiver problemas salvando a ação, i.e, todas as ações salvas são iguais, considere fazer uma deep copy de **a** e depois salvar (https://docs.python.org/3/library/copy.html)*
- Terminado o episódio, esta lista pode ser salva em um arquivo no computador.
- Para salvar este arquivo, pode ser utilizado Pickle
- https://stackoverflow.com/questions/52444921/save-numpy-array-using-pickle

Miscellaneous

- Os controles manuais (direction, throttle, brake) estão inicialmente ajustados para o máximo. Valores menores podem auxiliar no controle do carro para gerar as trajetórias manualmente
- O loop original é infinito, mas pode ser editado para salvar uma trajetória por arquivo, por exemplo
- Uma vez colhidos os dados de treinamento, é necessário definir uma rede neural que será o agente

NOTAS SOBRE O TREINAMENTO DA REDE NEURAL EM PYTORCH

- PyTorch dispõem de alguns exemplos de como treinar um modelo para openAI gym, como por exemplo: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
- Este exemplo é útil como um starting point para a rede neural
- Uma vez que a rede neural estiver treinada utilize o pytorch para salvar os parâmetros da rede neural em um arquivo

NOTAS SOBRE O DEPLOY DA REDE NEURAL

- Crie um novo arquivo baseado no código fonte oficial do CarRacing
- Substitua os comandos manuais pela rede neural, isto é, use o estado (s_0) gerado por s_prev = env.reset() como input da rede neural, que vai ter como output um vetor de ação **a**
- Este vetor ação vai agora para o simulador em env.step(a) e gera o novo estado s
- O novo estado entra na rede neural e gera uma nova ação.
- Assim o loop se repete e o carro consegue dar uma volta completa no circuito.