In [1]:

```
%load_ext autoreload
%autoreload 2
```

In [2]:

```
%matplotlib inline
import numpy as np
from rnn.arch import RNN
from vae.arch import VAE
import matplotlib.pyplot as plt
from gym.utils import seeding
from IPython import display
import time
from model import make_model

import config


np.set_printoptions(precision=4, suppress = True)
```

In [3]:

```
def get_mixture_coef(z_pred):

    log_pi, mu, log_sigma = np.split(z_pred, 3, 1)
    log_pi = log_pi - np.log(np.sum(np.exp(log_pi), axis = 1, keepdims = True))

    return log_pi, mu, log_sigma

def get_pi_idx(x, pdf):
    # samples from a categorial distribution
    N = pdf.size
    accumulate = 0
    for i in range(0, N):
        accumulate += pdf[i]
        if (accumulate >= x):
            return i
    random_value = np.random.randint(N)
    #print('error with sampling ensemble, returning random', random_value)
    return random_value

def sample_z(mu, log_sigma):
    z =  mu + (np.exp(log_sigma)) * np.random.randn(*log_sigma.shape)
    return z


def get_z_from_rnn_output(y_pred):
    HIDDEN_UNITS = 256
    GAUSSIAN_MIXTURES = 5
    Z_DIM = 32
    d = GAUSSIAN_MIXTURES * Z_DIM

    z_pred = y_pred[:(3*d)]
    rew_pred = y_pred[-1]

    z_pred = np.reshape(z_pred, [-1, GAUSSIAN_MIXTURES * 3])

    log_pi, mu, log_sigma = get_mixture_coef(z_pred)

    chosen_log_pi = np.zeros(Z_DIM)
    chosen_mu = np.zeros(Z_DIM)
    chosen_log_sigma = np.zeros(Z_DIM)

    # adjust temperatures
    logmix2 = np.copy(log_pi)
    logmix2 -= logmix2.max()
    logmix2 = np.exp(logmix2)
    logmix2 /= logmix2.sum(axis=1).reshape(Z_DIM, 1)


    for j in range(Z_DIM):
        idx = get_pi_idx(np.random.rand(), logmix2[j])
        chosen_log_pi[j] = idx
        chosen_mu[j] = mu[j, idx]
        chosen_log_sigma[j] = log_sigma[j,idx]

    next_z = sample_z(chosen_mu, chosen_log_sigma)

    # print(next_z)
    # print(rew_pred)
    if rew_pred > 0:
        next_reward = 1
    else:
        next_reward = 0

    return next_z, next_reward, chosen_mu
```

In [4]:

```
model = make_model()
model.make_env('car_racing')
```

```
/usr/local/lib/python3.6/dist-packages/gym/logger.py:30: UserWarning: WARN: Box bound precision lowered by casting to float32
  warnings.warn(colorize('%s: %s'%('WARN', msg % args), 'yellow'))
```

In [5]:

```
model.load_model('./controller/car_racing.cma.1.4.best.json')
```

```
loading file ./controller/car_racing.cma.1.4.best.json
```

In [6]:

```
z_weight = model.weight[0][:32,2]
h_weight = model.weight[0][32:,2]
```

In [7]:

```
z_weight
```

Out[7]:

```
array([ 0.0731, -0.3404,  0.3611, -0.7141, -1.5976, -0.2835, -0.174 ,
       -1.5905, -1.1533, -0.6324,  0.9622, -0.2288, -0.1064, -0.9466,
       -0.4954,  0.3949, -1.5045, -1.2485, -0.3165, -1.5292,  0.2247,
        0.5222, -1.4375,  0.1142,  0.4448, -0.5763,  0.0738, -0.5767,
        1.5348,  0.6582,  0.6452,  0.0891])
```

In [8]:

```
h_weight
```

Out[8]:

```
array([ 0.7708,  1.6354, -0.026 ,  0.1463, -0.1535,  0.2643, -2.2093,
        0.5426,  0.4684,  1.6411,  0.3211,  0.8327, -0.1384,  0.7367,
       -0.7765, -0.2703, -0.4366,  0.9702,  1.3071,  1.434 , -1.8583,
       -1.0848, -1.2774,  0.798 ,  0.9632, -0.245 ,  1.888 , -0.7669,
       -0.4921,  0.3643,  0.6034, -1.3566, -0.3   ,  0.672 , -0.5353,
       -0.5222, -1.8203, -0.3483,  0.0411,  0.5793,  1.358 ,  0.4963,
       -0.561 , -0.4595, -1.3505, -1.3342,  1.4493,  1.1615, -1.03  ,
       -0.9023,  0.2459, -0.2268,  0.4382, -0.5429, -0.5978, -0.6682,
        0.1689,  0.3315, -0.3384, -0.1408, -0.6581, -1.5985,  0.6023,
        1.165 , -0.9319,  0.1835, -1.5431, -0.811 , -0.2244, -0.6781,
        0.2812,  1.0255,  0.1594, -0.2758, -0.1614, -0.4056, -0.0794,
       -0.6482,  0.2188, -0.0507,  0.3166,  0.2145,  0.3336, -0.885 ,
        0.2917, -0.4975, -0.532 ,  0.6997, -0.1852, -0.608 , -0.2585,
       -0.2187, -1.6553, -0.9485,  0.141 ,  0.6077, -1.1454, -0.7395,
        0.9487, -0.7569, -1.0414,  0.2706,  0.2234,  0.0936,  0.1667,
        0.1819,  0.1624,  1.3433, -0.6247,  0.2671,  0.6688,  0.5441,
        0.3598, -0.3343, -0.454 ,  0.8865,  0.0776,  0.2722,  0.4558,
        0.5903, -0.1236,  1.091 , -0.3669,  1.32  ,  0.5369, -0.9222,
        0.0763, -0.8723, -0.1498, -0.6848, -0.2375, -0.2891,  0.5063,
        0.3236,  1.7393,  1.0349, -0.4014,  0.4054, -0.1571,  1.574 ,
        0.6948,  2.4344,  0.6219, -1.0295, -0.4695, -0.1669, -1.0824,
        0.0037, -0.0938, -1.0715,  0.7284, -0.5315,  0.792 ,  0.7487,
        0.0751,  0.3014, -0.3447, -0.4604, -1.0108,  0.1091, -1.5092,
       -0.6423, -0.9169,  0.3262, -0.6799, -0.9179,  0.3724, -0.1806,
        1.0742,  0.3938,  0.2019,  0.0412, -0.6222,  2.1165, -0.509 ,
        0.2405,  0.2712,  0.5001,  0.0288, -0.1158,  0.2006,  0.696 ,
       -0.667 , -0.4095,  0.5452, -0.3879,  0.3545,  0.4734, -0.11  ,
       -0.2248,  1.2531, -1.3836, -0.2704,  0.1217, -1.7953, -0.179 ,
        0.6782,  0.2202, -0.3277,  1.3844,  1.1085, -1.7758, -0.6746,
        0.0907, -0.0588,  0.009 , -1.0305,  0.2304, -0.1472, -0.5917,
        1.0772, -0.3924, -1.4103, -0.7524,  0.3838,  0.5777,  0.2066,
        1.3378, -0.498 , -0.2556, -0.2855,  0.1986,  0.4392, -0.7434,
        0.0719,  0.0792, -0.7164, -0.7194, -0.4259,  0.287 , -0.1296,
       -0.5445,  0.06  , -1.3068, -0.5268,  0.1093, -0.4393, -0.5988,
        0.4046, -1.1227,  0.769 , -0.874 , -0.111 ,  1.7344, -0.5057,
        0.5589,  0.1008, -0.2948,  0.5114,  0.8845, -0.1265, -1.6181,
        0.9022,  0.6885, -0.9643, -0.8166])
```

In [9]:

```
rollout_files = np.load('./data/rollout/272237.npz')
obs_file = rollout_files['obs']
action_file = rollout_files['action']
reward_file = rollout_files['reward']
done_file = rollout_files['done']

series_files = np.load('./data/series/272237.npz')
mu_file = series_files['mu']
log_var_file = series_files['log_var']
action_2_file = series_files['action']
reward_2_file = series_files['reward']
done_2_file = series_files['done']
```

In [10]:

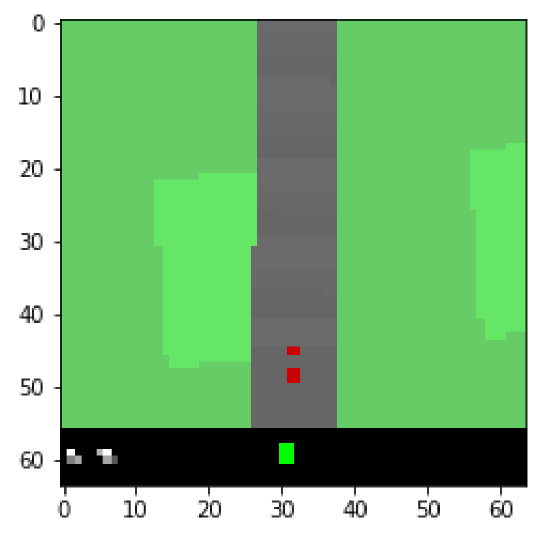
```
obs = obs_file[4]
action = [0,0,0]
reward = 0
model.reset()
```

In [11]:

```
plt.imshow(obs)
```

Out[11]:

```
<matplotlib.image.AxesImage at 0x7f6efc232d30>
```



In [12]:

```
vae_encoded_obs = model.update(obs, 0)
```

In [13]:

```
vae_encoded_obs
```

Out[13]:

```
array([-0.0956, -1.8537, -0.9818, -0.4553, -0.223 ,  1.2784,  0.8178,
       -1.0334, -0.5426,  0.047 , -0.7834, -3.2447, -0.0163,  0.0817,
       -0.1257,  0.6326,  1.1279,  1.5259,  1.0971,  1.3797, -0.5222,
        1.2767, -1.5914, -0.1917,  2.4134,  0.4496,  1.1854, -1.2209,
        0.197 ,  0.1581, -0.5397, -0.8014], dtype=float32)
```

In [14]:

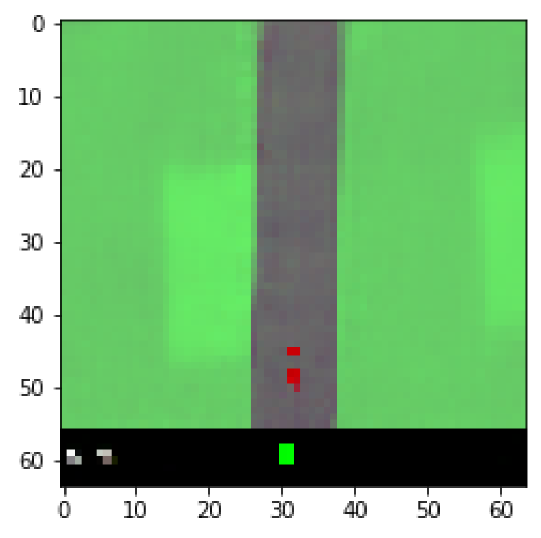
```
recon = model.vae.decoder.predict(np.array([vae_encoded_obs]))[0]
```

In [15]:

```
plt.imshow(recon)
```

Out[15]:

```
<matplotlib.image.AxesImage at 0x7f6efc01e160>
```



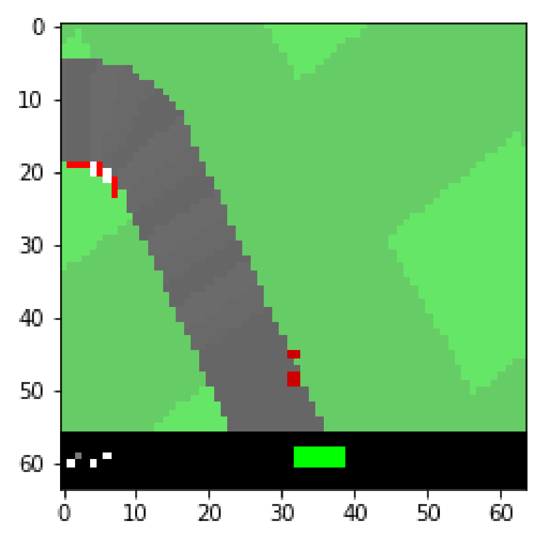
In [16]:

```
# obs = obs_file[0]
action = [0,1,0]
reward = 0
model.reset()
total_reward = 0
total_pseudo_reward = 0
t = 0

obs = model.env.reset()

model.env.render('rgb_array')

actions0 = []
actions1 = []
actions2 = []
```

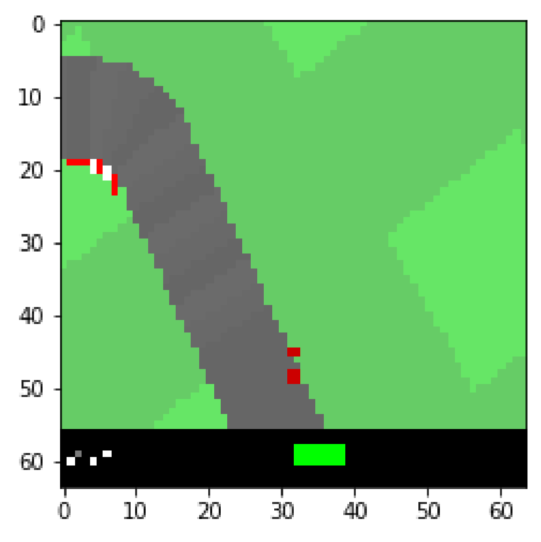
In [17]:

```
# while(1):
from PIL import Image, ImageDraw
imagelist = []
```

```
for i in range(500):
    ######

    obs = config.adjust_obs(obs)
    reward = config.adjust_reward(reward)

    total_pseudo_reward+= reward

    vae_encoded_obs = model.update(obs, 0)

    recon = model.vae.decoder.predict(np.array([vae_encoded_obs]))[0]

#     input_to_rnn = [np.array([[np.concatenate([vae_encoded_obs, action, [reward]])]]),np.zeros(shape=(1,256)),np.zeros(shape=(1,256))]
#     input_to_rnn = [np.array([[np.concatenate([np.zeros(32), action, [reward]])]]),np.array([model.hidden]),np.array([model.cell_values])]


    input_to_rnn = [np.array([[np.concatenate([vae_encoded_obs, action, [reward]])]]),np.array([model.hidden]),np.array([model.cell_values])]

    #     print(np.array([[np.concatenate([vae_encoded_obs, action, [reward]])]]).shape)
    #     print(np.array([model.hidden]).shape)
    #     print(np.array([model.cell_values]).shape)

    out = model.rnn.forward.predict(input_to_rnn)

    y_pred = out[0][0][0]
    h = out[1][0]
    c = out[2][0]

    model.hidden = h
    model.cell_values = c

    next_z, next_reward, chosen_mu = get_z_from_rnn_output(y_pred)

    recon_next_z = model.vae.decoder.predict(np.array([next_z]))[0]

    controller_obs = np.concatenate([vae_encoded_obs,model.hidden])
    action = model.get_action(controller_obs, t=0, add_noise=0)
    #     actions0.append(action[0])
    #     actions1.append(action[1])
    #     actions2.append(action[2])

    #     action = model.activations(action)
#     action = [1,1,0]
    obs, reward, done, _ = model.env.step(action)

    total_reward+= reward
    img = Image.fromarray(obs)
    imagelist.append(img)

    plt.gca().cla()
    plt.imshow( obs)

    display.clear_output(wait=True)
    display.display(plt.gcf())

    print(total_reward)
    print(total_pseudo_reward)
    print(t)
    print(action)
    t += 1
imagelist[0].save('out.gif', save_all=True, append_images=imagelist[1:])

    #     print(action)
```



```
-27.900552486188044
7
499
[1.      0.993  0.1209]
```



```
In [ ]:
```