BLG453E
Homework-3

02.12.2019

Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

Sometimes our desires bang up against obstacles. Somebody else drank that last soft drink in the refrigerator; the formerly all-night grocery store now closes at midnight; my friend's car has a flat tire; the dog ate my homework; the plane just pulled out of the gate thirty seconds ago; the flight has been canceled because of a snowstorm in Saskatoon; we're having computer troubles and we can't seem to make PowerPoint work in here; I left my wallet in my other pair of pants; you misread the final deadline; the reviewer was someone who hates us; she didn't hear about the job until too late; the runner in the next lane is faster than I am; and so on and so forth.

In such cases our will alone, though it pushes us, does not get us what we want. It pushes us in a certain direction, but we are maneuvering inside a hedge maze whose available paths were dictated by the rest of the world, not by our wants. And so we move willy-nilly, but not freewilly-nilly, inside the maze. A combination of pressures, some internal and some external, collectively dictates our pathway in this crazy hedge maze called "life".

I Am a Strange Loop, Douglas Hofstadter

- You should write all your code in Python language.
- Cheating is highly discouraged.
- Ninova only stores files under 20 MB. If you could not upload your results, you can share them with me via Dropbox, or send me private YouTube video links for each part's results.

1 - Part 1: Sobel Filter (10 pts.)

In this homework, we will control a 3D game character given in Figure 1. I created a small game with Unity3D game engine[1] which works with WebGL graphics and can be

[1] https://unity3d.com/unity

played on Firefox web browser. The game can be started by opening **index.html** in Firefox. However, since it is a local file, you should reach "about:config" page on Firefox and change "privacy.file_unique_origin" to "false".

Figure 1: Our main character who will be controlled by us, without his free-will.

Then, a game wicket containing a view for our character and some tiles will be opened as given in Figure 2. The character can be controlled by WASD buttons and Shift button makes him move faster.

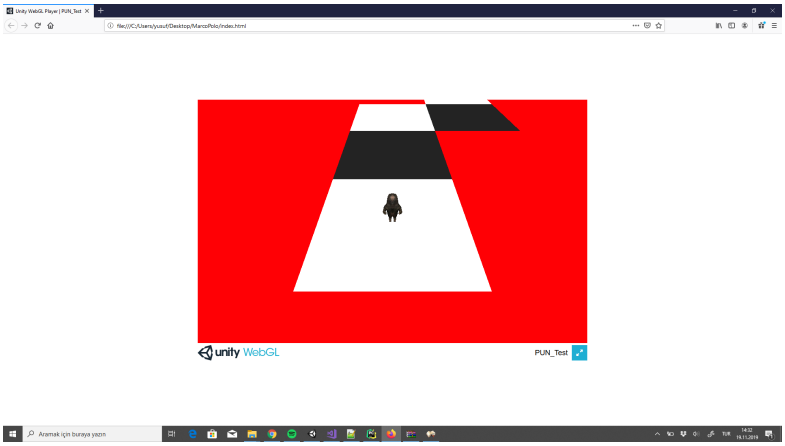

Figure 2: A general view of the game

For this homework, you will benefit from *pyautogui* library which is used to simulate mouse and keyboard interactions with Python. The example script given below first takes a screenshot of the game, then moves the character in different directions. For a better timing, the script should be run after the map inside the game is loaded.

```
1 import pyautogui
  import time
3
  time.sleep(5)
```

```
#In this 5 seconds you should switch to game screen to transfer the
    simulated keyboard inputs to the game.

myScreenshot = pyautogui.screenshot()
myScreenshot.save('test.png')
# An example screenshot is obtained. We will work on screenshots like this
    for this homework.

for i in range(20):
    pyautogui.keyDown('shift')
    if i < 5:
        pyautogui.keyDown('w')
    elif i < 10:
        pyautogui.keyUp('w')
        pyautogui.keyDown('s')
    elif i < 15:
        pyautogui.keyUp('s')
        pyautogui.keyDown('d')
    elif i < 20:
        pyautogui.keyUp('w')
        pyautogui.keyDown('s')

pyautogui.keyUp('s')
pyautogui.keyUp('shift')

#pyautogui.keyUp and pyautogui.keyDown functions are used to simulate
    holding a button. For simple presses, pyautogui.press can be used.
```

To move our character on the dangerous pathway in front of him, we should detect the cliffs(edges) and avoid from them. As a first step, convolve a screenshot image with sobel filters to obtain a clearer look of edges. Instead of filtering whole area, you can predefine a grabbing area containing the game and use only this part of the screenshot.

2 - Part 2: Canny Edge Detector (10 pts.)

Use Canny Edge detection[2] algorithm to obtain an edge map of the game screenshot as given in Figure 3. You can use OpenCV's default detector. Try to find optimum parameters. Also you can use "cv2.findContours" function to find the contours which will ease your job in the last part.

3 - Part 3: Minimum Eigenvalue Corner Detector (30 pts.)

Implement Minimum Eigenvalue corner detector on game screenshots to obtain corners as given in Figure 4.

[2]Canny, John, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 6, 1986, pp. 679-698.

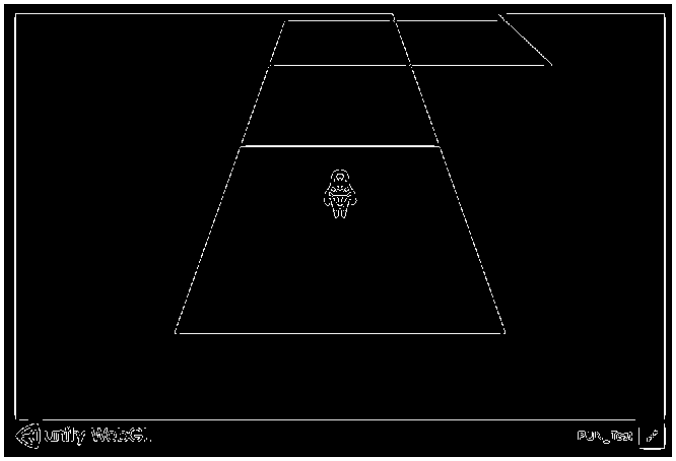

Figure 3: Canny edge detection results of game image.

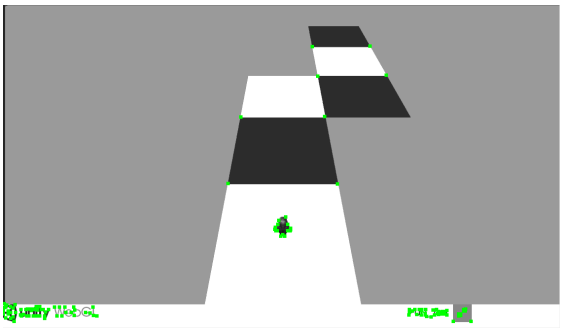

Figure 4: Corners detected on game screenshot

4 - Part 4: "As I walk through the valley of the shadow of death..." (20 pts.)

Make the monster reach the end of the path healthfully. Use your findings in the previous parts.

5 - Part 5: Otsu Thresholding (30 pts.)

In Otsu Thresholding[3], prove that between-class variance can be calculated as

$$\sigma^2_{between}(T) = \sigma^2_{total}(T) - \sigma^2_{within}(T). \tag{0.1}$$

[3]Otsu, N. (1979). A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1), 62-66.