**CSE 423  
Moinul Hossain Bhuiyan  
ID: 20301002   
Lab Section: 01**

**Assignment 02**

**BRAC UNIVERSITY**

!pip install tensorflow==1.15.0

!pip install -q lucid>=0.2.3

!pip install -q moviepy

!pip install imageio==2.4.1

import numpy as np

import json

import moviepy.editor as mvp

from google.colab import files

import tensorflow.compat.v1.gfile as gfile

import lucid.misc.io.showing as show

from lucid.misc.gl.glcontext import create\_opengl\_context

# Now it's safe to import OpenGL and EGL functions

import OpenGL.GL as gl

from OpenGL.GLU import \*

# create\_opengl\_context() creates GL context that is attached to an

# offscreen surface of specified size. Note that rendering to buffers

# of different size and format is still possible with OpenGL Framebuffers.

#

# Users are expected to directly use EGL calls in case more advanced

# context management is required.

WIDTH, HEIGHT = 800,600

create\_opengl\_context((WIDTH, HEIGHT))

# OpenGL context is available here.

print(gl.glGetString(gl.GL\_VERSION))

print(gl.glGetString(gl.GL\_VENDOR))

#print(gl.glGetString(gl.GL\_EXTENSIONS))

gl.glClear(gl.GL\_COLOR\_BUFFER\_BIT)

gl.glColor3f(.5,.3,.7)

gl.glPointSize(5)

def findZone(x1,y1,x2,y2):

    dx = x2-x1

    dy = y2-y1

    if abs(dx)>=abs(dy):    #Modulus

        if dx>=0:

          if dy>=0:

            return "Z0"

        if dx<0:

          if dy>=0:

            return "Z3"

        if dx<0:

          if dy<0:

            return "Z4"

        if dx>=0:

          if dy<0:

            return "Z7"

    else:

        if dx>=0:

          if dy>=0:

            return "Z1"

        if dx<0:

          if dy>=0:

            return "Z2"

        if dx<0:

           if dy<0:

            return "Z5"

        if dx>=0:

          if dy<0:

            return "Z6"

def convert0(x1,y1,x2,y2,zone):

    if zone=="Z0":

        x1,y1=x1,y1

        x2,y2=x2,y2

        converted=[x1,y1,x2,y2]

        return converted

    if zone=="Z1":

        x1,y1=y1,x1

        x2,y2=y2,x2

        converted = [x1, y1, x2, y2]

        return converted

    if zone=="Z2":

        x1,y1=y1,-x1

        x2,y2=y2,-x2

        converted = [x1, y1, x2, y2]

        return converted

    if zone=="Z3":

        x1,y1=-x1,y1

        x2,y2=-x2,y2

        converted = [x1, y1, x2, y2]

        return converted

    if zone=="Z4":

        x1,y1=-x1,-y1

        x2,y2=-x2,-y2

        converted = [x1, y1, x2, y2]

        return converted

    if zone=="Z5":

        x1,y1=-y1,-x1

        x2,y2=-y2,-x2

        converted = [x1, y1, x2, y2]

        return converted

    if zone=="Z6":

        x1,y1=-y1,x1

        x2,y2=-y2,x2

        converted = [x1, y1, x2, y2]

        return converted

    if zone=="Z7":

        x1,y1=x1,-y1

        x2,y2=x2,-y2

        converted = [x1, y1, x2, y2]

        return converted

def convertbacktoOriginal(x,y,zone):

    if zone=="Z0":

        x,y=x,y

        return [x,y]

    if zone=="Z1":

        x,y=y,x

        return [x, y]

    if zone=="Z2":

        x,y=-y,x

        return [x, y]

    if zone=="Z3":

        x,y=-x,y

        return [x, y]

    if zone=="Z4":

        x,y=-x,-y

        return [x, y]

    if zone=="Z5":

        x,y=-y,-x

        return [x, y]

    if zone=="Z6":

        x,y=y,-x

        return [x, y]

    if zone=="Z7":

        x,y=x,-y

        return [x, y]

def plotpoints(x,y):

  rd1 = x/100

  rd2 = y/100

  gl.glColor3f(0,.5,0)

  gl.glBegin(gl.GL\_POINTS)

  gl.glVertex2f(rd1,rd2)

  gl.glEnd()

#Equations For Zone

def rvn(x1,y1,x2,y2,zone):

    dx=x2-x1

    dy=y2-y1

    d=2\*dy-dx

    e=2\*dy

    ne=2\*(dy-dx)

    x=x1

    y=y1

    while x<x2:

        x=x+1

        if d>0:

            d=d+ne

            y=y+1

        else:

            d=d+e

        values=convertbacktoOriginal(x,y,zone)

        plotpoints(values[0],values[1])

#Point Plot

def plotting(x1,y1,x2,y2):

  zone=findZone(x1,y1,x2,y2)

  values=convert0(x1,y1,x2,y2,zone)

  rvn(values[0],values[1],values[2],values[3],zone)

#LEFT

def up():

  axis=[0,50,-50,50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def upright():

  axis=[0,50,0,0]

  plotting(axis[0],axis[1],axis[2],axis[3])

def upleft():

  axis=[-50,50,-50,0]

  plotting(axis[0],axis[1],axis[2],axis[3])

def mid():

  axis=[0,0,-50,0]

  plotting(axis[0],axis[1],axis[2],axis[3])

def downright():

  axis=[0,0,0,-50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def downleft():

  axis=[-50,0,-50,-50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def down():

  axis=[-50,-50,0,-50]

  plotting(axis[0],axis[1],axis[2],axis[3])

#RIGHT

def up1():

  axis=[20,50,70,50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def upright1():

  axis=[70,50,70,0]

  plotting(axis[0],axis[1],axis[2],axis[3])

def upleft1():

  axis=[20,50,20,0]

  plotting(axis[0],axis[1],axis[2],axis[3])

def mid1():

  axis=[20,0,70,0]

  plotting(axis[0],axis[1],axis[2],axis[3])

def downright1():

  axis=[70,0,70,-50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def downleft1():

  axis=[20,0,20,-50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def down1():

  axis=[20,-50,70,-50]

  plotting(axis[0],axis[1],axis[2],axis[3])

def left(axis):

  if axis==0:

    up()

    upleft()

    downleft()

    down()

    downright()

    upright()

  if axis==1:

    upright()

    downright()

  if axis==2:

    up()

    upright()

    mid()

    downleft()

    down()

  if axis==3:

    up()

    upright()

    mid()

    downright()

    down()

  if axis==4:

    upleft()

    mid()

    upright()

    downright()

  if axis==5:

    up()

    upleft()

    mid()

    downright()

    down()

  if axis==6:

    up()

    upleft()

    downleft()

    down()

    downright()

    mid()

  if axis==7:

    up()

    upright()

    downright()

  if axis==8:

    up()

    upleft()

    mid()

    upright()

    downright()

    down()

    downleft()

  if axis==9:

    up()

    upleft()

    mid()

    upright()

    downright()

    down()

def right(axis):

  if axis==0:

    up1()

    upleft1()

    downleft1()

    down1()

    downright1()

    upright1()

  if axis==1:

    upright1()

    downright1()

  if axis==2:

    up1()

    upright1()

    mid1()

    downleft1()

    down1()

  if axis==3:

    up1()

    upright1()

    mid1()

    downright1()

    down1()

  if axis==4:

    upleft1()

    mid1()

    upright1()

    downright1()

  if axis==5:

    up1()

    upleft1()

    mid1()

    downright1()

    down1()

  if axis==6:

    up1()

    upleft1()

    downleft1()

    down1()

    downright1()

    mid1()

  if axis==7:

    up1()

    upright1()

    downright1()

  if axis==8:

    up1()

    upleft1()

    mid1()

    upright1()

    downright1()

    down1()

    downleft1()

  if axis==9:

    up1()

    upleft1()

    mid1()

    upright1()

    downright1()

    down1()

id=(input())

left(int(id[-2]))

right(int(id[-1]))

img\_buf = gl.glReadPixelsub(0, 0, WIDTH, HEIGHT, gl.GL\_RGB, gl.GL\_UNSIGNED\_BYTE)

img = np.frombuffer(img\_buf, np.uint8).reshape(HEIGHT, WIDTH, 3)[::-1]

show.image(img/255.0)

