```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Nov  2 11:45:12 2020
This software was created as a “proof of concept” for a binaural mixer and
sound control for 3D environments.  The program was written in python3.
The end goal is to have a binaural mixer for real-time artificial and
extended reality application based on head-related sensor data.

Sound elements are placed in the space.   The head’s rotating perspective
is controlled by user input. The distance and angle of the sound-objects
are relative to the orientation of the head.  A mix, at any moment is based
on the heads orientation to the virtual space.  The result is a binaural mix
of sound (a mix intended for headphones only).

The software uses a few classes that I wrote for the "binaural_mixer_class"

@author: d holmes
"""
import pygame as pg
from pygame.locals import *
import os
import sys
import numpy as np
from openal.audio import SoundSink, SoundSource, SoundListener
from openal.loaders import load_wav_file
from binaural_mixer_class import moving_element, stable_element


def screen_size():
    pg.init()
    bubba_screen = pg.display.Info()
    return bubba_screen.current_w, bubba_screen.current_h

def rotate(image, rect, angle):
    """Rotate the image while keeping its center."""
    # Rotate the original image without modifying it.
    new_image = pg.transform.rotate(image, angle)
    # Get a new rect with the center of the old rect.
    rect = new_image.get_rect(center=rect.center)
    return new_image, rect



def main():

    #Set the phase of the circle or elipse
```

```
pa = 0 # Phase use in phase offset
phase = pa/180*np.pi # calculation of phase baced on half circle
radius = 100

#get and set the screen dimensions
sys_screen = screen_size()
pg_width = 640 * 2
pg_height = 480 * 2
#1/2 system screen x - 1/2 pygame window is used to place the pygame window i
dwind_w = ((sys_screen[0]/2) - (int(pg_width/2)))
dwind_h = sys_screen[1] #system screen y is not used
screen = pg.display.set_mode((pg_width, pg_height))
os.environ['SDL_VIDEO_WINDOW_POS'] = "%d,%d" % (dwind_w, 80)

originx = int(pg_width * 0.5)   #set the x value of center screen
originy = int(pg_height * 0.5)  #set the y value of center screen


#Initialize OpenAL related components for ambient soynd
sound_sink_amb = SoundSink()
sound_source_amb = SoundSource()
listener_amb = SoundListener()
sound_sink_amb.activate()
sound_sink_amb._listener = listener_amb
sound_sink_amb._Source = sound_source_amb


source_sound_file_amb = "cricket-ambience.wav"
sound_data_amb = load_wav_file(source_sound_file_amb)
sound_source_amb.queue(sound_data_amb)
sound_source_amb.looping = True


#Initialize OpenAL related components for waterf sound
waterf_listener = SoundListener()
waterf_sink = SoundSink()
waterf_source = SoundSource()
waterf_sink.activate()
waterf_sink._listener = waterf_listener
waterf_sink._Source = waterf_source
waterf_sound_file = "waterf.wav"
waterf_data = load_wav_file(waterf_sound_file)
waterf_source.queue(waterf_data)
waterf_source.looping = True


#Initialize OpenAL related components for ac sound
```

```
ac_listener = SoundListener()
ac_sink = SoundSink()
ac_source = SoundSource()
ac_sink.activate()
ac_sink._listener = ac_listener
ac_sink._Source = ac_source
ac_sound_file = "ac.wav"
ac_data = load_wav_file(ac_sound_file)
ac_source.queue(ac_data)
ac_source.looping = True

#Initialize OpenAL related components for TV sound
tv_listener = SoundListener()
tv_sink = SoundSink()
tv_source = SoundSource()
tv_sink.activate()
tv_sink._listener = tv_listener
tv_sink._Source = tv_source
tv_sound_file = "tv.wav"
tv_data = load_wav_file(tv_sound_file)
tv_source.queue(tv_data)
tv_source.looping = True

#Initialize OpenAL related components for moving sound
bug1_listener = SoundListener()
bug1_sink = SoundSink()
bug1_source = SoundSource()
bug1_sink.activate()
bug1_sink._listener = bug1_listener
bug1_sink._Source = bug1_source
bug1_sound_file = "bug1.wav"
bug1_data = load_wav_file(bug1_sound_file)
bug1_source.queue(bug1_data)
bug1_source.looping = True

bug1 = moving_element(originx, originy, 200, phase = 0, color = (0,255,0))
bug1.image = pg.Surface((10, 10))
bug1.image.fill((bug1.color))

bug2_listener = SoundListener()
bug2_sink = SoundSink()
bug2_source = SoundSource()
bug2_sink.activate()
bug2_sink._listener = bug2_listener
bug2_sink._Source = bug2_source
bug2_sound_file = "bug2.wav"
bug2_data = load_wav_file(bug2_sound_file)
```

```
bug2_source.queue(bug2_data)
bug2_source.looping = True

bug2 = moving_element(originx, originy, 100, color = (0,100,205))
bug2.image = pg.Surface((10, 10))
bug2.image.fill((bug2.color))

bug3_listener = SoundListener()
bug3_sink = SoundSink()
bug3_source = SoundSource()
bug3_sink.activate()
bug3_sink._listener = bug3_listener
bug3_sink._Source = bug3_source
bug3_sound_file = "bug3.wav"
bug3_data = load_wav_file(bug3_sound_file)
bug3_source.queue(bug3_data)
bug3_source.looping = True

bug3 = moving_element(originx, originy, 150, phase = 90, color = (50, 150, 15(
bug3.image = pg.Surface((10, 10))
bug3.image.fill((bug3.color))

bug4_listener = SoundListener()
bug4_sink = SoundSink()
bug4_source = SoundSource()
bug4_sink.activate()
bug4_sink._listener = bug4_listener
bug4_sink._Source = bug4_source
bug4_sound_file = "bug4.wav"
bug4_data = load_wav_file(bug4_sound_file)
bug4_source.queue(bug4_data)
bug4_source.looping = True

bug4 = moving_element(originx, originy, 100, phase = 180, color = (150, 150, (
bug4.image = pg.Surface((10, 10))
bug4.image.fill((bug4.color))

bug5_listener = SoundListener()
bug5_sink = SoundSink()
bug5_source = SoundSource()
bug5_sink.activate()
bug5_sink._listener = bug5_listener
bug5_sink._Source = bug5_source
bug5_sound_file = "bug5.wav"
bug5_data = load_wav_file(bug5_sound_file)
bug5_source.queue(bug5_data)
bug5_source.looping = True
```

```
bug5 = moving_element(originx, originy, 200, phase = 270,
                      color = (0, 150, 150))
bug5.image = pg.Surface((10, 10))
bug5.image.fill((bug4.color))

pg.init()

bug1_sink.play(bug1_source)
bug2_sink.play(bug2_source)
bug3_sink.play(bug3_source)
bug4_sink.play(bug4_source)
bug5_sink.play(bug5_source)
tv_sink.play(tv_source)
ac_sink.play(ac_source)
waterf_sink.play(waterf_source)
sound_sink_amb.play(sound_source_amb)

gray = (55, 55, 55)
clock = pg.time.Clock()
apple_size = 5
apple_pos = (100, 100)
apple_image = pg.Surface((apple_size, apple_size))
apple_image.fill((255, 0, 0))  # Apple should be red.

tv1 = stable_element(originx, originy, radius = 486, deg = 4.748,
                     name = "oldtv.gif")
tv1.rect = (92, 451, 125, 89)
tv = pg.image.load(tv1.name)
tv_rect = tv1.rect
#tv_rect =  tv.get_rect()
#tv1.rect = tv_rect
#print("tv rect is ", tv_rect)
tv_mask = pg.Surface((tv1.rect[2], tv1.rect[3]))
tv_mask.fill((255, 0, 0))  #  should be red/tranparent.
tv_mask.set_alpha(80)
tv1.position()

ac1 = stable_element(originx, originy, radius = 601, deg = 1.06465,
                     name = "airconditioner.gif")
ac1.rect = (1093, 694, 150, 150)
ac = pg.image.load(ac1.name)
ac_rect = ac1.rect
#ac_rect =  ac.get_rect()
#ac1.rect = ac_rect
#print("ac rect is ", ac_rect)
```

```
ac_mask = pg.Surface((ac1.rect[2], ac1.rect[3]))
ac_mask.fill((255, 0, 0))  #  should be red/tranparent.
ac_mask.set_alpha(80)
ac1.position()

waterf1 = stable_element(originx, originy, radius = 455, deg = 2.705,
                         name = "water-fountain.gif")
waterf1.rect = (734, 10, 200, 116)
waterf = pg.image.load(waterf1.name)
waterf_rect = waterf1.rect
#waterf_rect =  waterf.get_rect()
#waterf1.rect = waterf_rect
#print("waterf rect is ", waterf_rect)
waterf_mask = pg.Surface((waterf1.rect[2], waterf1.rect[3]))
waterf_mask.fill((255, 0, 0))  #  should be red/tranparent.
waterf_mask.set_alpha(80)
waterf1.position()



image = pg.image.load("head.gif")
orig_image = image
rect = image.get_rect(center=(int(pg_width/2), int(pg_height/2)))

angle = 0
deg, deg1, deg2, deg3, deg4, deg5 = 0,0,0,0,0,0

done = False
kright = False
kleft = False
apple_sauce = [0,0]



while not done:

    bug5.clockwize(deg5, offsetx = 100, offsety = 0)
    bug5.image.fill((bug5.behind_head(deg5, deg, 90)))
    bug5_source.position = (round(bug5.snd_pos[0], 2), 0,
                            round(bug5.snd_pos[1], 2))
    bug5_source.gain = bug5.gain

    bug4.counter_clockwize(deg4, offsetx = 0, offsety = 100)
    bug4.image.fill((bug4.behind_head(deg4, deg, 90)))
    bug4_source.position = (round(bug4.snd_pos[0], 2), 0,
                            round(bug4.snd_pos[1], 2))
    bug4_source.gain = bug4.gain
```

```
bug3.clockwize(deg3, offsetx = 50, offsety = 0)
bug3.image.fill((bug3.behind_head(deg3, deg, 90)))
bug3_source.position = (round(bug3.snd_pos[0], 2), 0,
                        round(bug3.snd_pos[1], 2))
bug3_source.gain = bug3.gain

bug2.counter_clockwize(deg2, offsetx = 50, offsety = 0)
bug2.image.fill((bug2.behind_head(deg2, deg, 90)))
bug2_source.position = (round(bug2.snd_pos[0], 2), 0,
                        round(bug2.snd_pos[1], 2))
bug2_source.gain = bug2.gain

bug1.clockwize(deg1)
bug1.image.fill((bug1.behind_head(deg1, deg, 90)))
bug1_source.position = (round(bug1.snd_pos[0], 2), 0,
                        round(bug1.snd_pos[1], 2))
bug1_source.gain = bug1.gain


tv_source.position = (round(tv1.snd_pos[0], 2), 0,
                      round(tv1.snd_pos[1], 2))
tv_source.gain = tv1.gain


ac_source.position = (round(ac1.snd_pos[0], 2), 0,
                      round(ac1.snd_pos[1], 2))
ac_source.gain = ac1.gain

waterf_source.position = (round(waterf1.snd_pos[0], 2), 0,
                          round(waterf1.snd_pos[1], 2))
waterf_source.gain = waterf1.gain

tv1.behind_head(deg, 90)
ac1.behind_head(deg, 90)
waterf1.behind_head(deg, 90)

deg1 += 0.005
deg2 += 0.0055
deg3 += 0.006
deg4 += 0.005
deg5 += 0.007
if deg1 >= np.pi*2:
        deg1 = 0
if deg2 >= np.pi*2:
        deg2 = 0
if deg3 >= np.pi*2:
```

```
        deg3 = 0
if deg4 >= np.pi*2:
        deg4 = 0
if deg5 >= np.pi*2:
        deg5 = 0

screen.fill((gray))
screen.blit(apple_image, apple_pos)
screen.blit(bug1.image, bug1.pos)
screen.blit(bug2.image, bug2.pos)
screen.blit(bug3.image, bug3.pos)
screen.blit(bug4.image, bug4.pos)
screen.blit(bug5.image, bug5.pos)
screen.blit(image, rect)
screen.blit(tv, tv1.rect)

if tv1.highlight == True:
    screen.blit(tv_mask, tv1.rect)
screen.blit(ac, ac1.rect)
if ac1.highlight == True:
    screen.blit(ac_mask, ac1.rect)
screen.blit(waterf, waterf1.rect)
if waterf1.highlight == True:
    screen.blit(waterf_mask, waterf1.rect)

bug5_sink.update()
bug4_sink.update()
bug3_sink.update()
bug2_sink.update()
bug1_sink.update()
tv_sink.update()
ac_sink.update()
waterf_sink.update()
sound_sink_amb.update()
pg.display.flip()
pg.display.update()

for event in pg.event.get():
    if event.type == pg.QUIT:
        pg.quit()
        sys.exit()
        done = True

    if (event.type == KEYDOWN):
        if event.key == K_q:
            pg.quit()
            sys.exit()
```

```
        if event.key == K_RIGHT and kright == False:
            kright = True
            st_clock = pg.time.get_ticks()
            break

        if event.key == K_LEFT and kleft == False:
            kleft = True
            st_clock = pg.time.get_ticks()
            break

    if (event.type == KEYUP):
        if event.key == K_RIGHT:
            kright = False
            end_clock = pg.time.get_ticks()
            move = round((end_clock - st_clock)*0.1)
            angle -= move
            image, rect = rotate(orig_image, rect, angle)

        if event.key == K_LEFT:
            kleft = False
            end_clock = pg.time.get_ticks()
            move = round((end_clock - st_clock)*0.1)
            angle += move
            image, rect = rotate(orig_image, rect, angle)

    if (event.type == KEYDOWN) and (event.key == K_1):
        key = 1
    if (event.type == KEYUP) and (event.key == K_1):
        key = 0
    if (event.type == MOUSEBUTTONDOWN) and key == 1:
        cursor = pg.mouse.get_pos()[0], pg.mouse.get_pos()[1]
        #apple_sauce = waterf1.get_radius(cursor)
        #apple_sauce = ac1.get_radius(cursor)
        apple_sauce = tv1.get_radius(cursor)

    if abs(angle) >= 360:
        angle = 0
    into_one = (1/360)

    deg = (angle * into_one * (np.pi*2)) % (np.pi*2)

    hx = (originx + np.sin(apple_sauce[0])*(apple_sauce[1]))
    hy = (originy + np.cos(apple_sauce[0])*(apple_sauce[1]))

    xx = (0 - np.sin(deg-(phase))*1)  #head position (0, 0, XX)
    xy = (0 - np.cos(deg-(phase))*1)  #head position (xy, 0, 0)
```

```
            apple_pos = (hx,hy)


            bug1_listener.orientation = [xx, 0, xy, 0, 1, 0]
            bug2_listener.orientation = [xx, 0, xy, 0, 1, 0]
            bug3_listener.orientation = [xx, 0, xy, 0, 1, 0]
            bug4_listener.orientation = [xx, 0, xy, 0, 1, 0]
            bug5_listener.orientation = [xx, 0, xy, 0, 1, 0]
            tv_listener.orientation = [xx, 0, xy, 0, 1, 0]
            ac_listener.orientation = [xx, 0, xy, 0, 1, 0]
            waterf_listener.orientation = [xx, 0, xy, 0, 1, 0]


if __name__ == '__main__':
    pg.init()
    main()
    pg.quit()
```