ΟΔΗΓΗΣΗ ΓΡΑΦΙΚΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΦΩΝΗΣ ΜΙΑ ΕΦΑΡΜΟΓΗ PYGAME

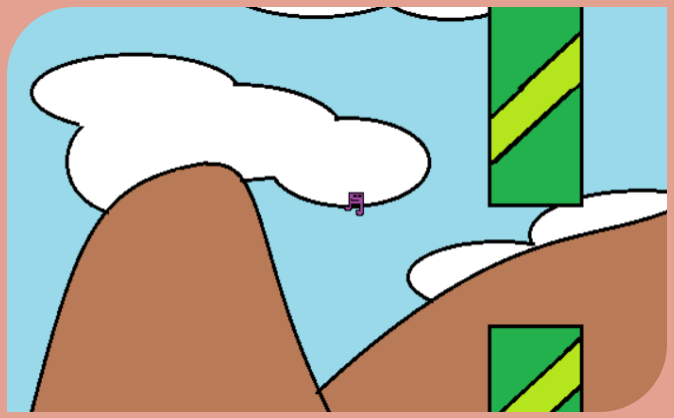
ΕΡΓΑΣΙΑ ΕΑΡΙΝΟΥ ΕΞΑΜΗΝΟΥ 2024 ΣΤΟ ΜΑΘΗΜΑ ΕΠΕΞΕΡΓΑΣΙΑ ΗΧΟΥ ΚΑΙ ΜΟΥΣΙΚΗΣ

ΜΑΡΙΑ ΚΑΡΑΔΗΜΗΤΡΟΠΟΥΛΟΥ 9093202000121

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΠΑΝΑΓΙΩΤΗΣ ΖΕΡΒΑΣ

Το Pygame που θα παρουσιαστεί ονομάζεται ***Flappy Note***. *Το Flappy Note* εμπνεύστηκε από το *Flappy Bird,* ένα διάσημο παιχνίδι που εμφανίστηκε το 2014 και κατάφερε να γίνει από τα διασημότερα που έχουν υπάρξει. Η λογική για να παίξει κανείς είναι πολύ απλή. Ο παίκτης ελέγχει ένα μικρό πουλί και στόχος είναι να το καθοδηγήσει μέσα από μια σειρά από πράσινους σωλήνες που εμφανίζονται σε διαφορετικά ύψη, χωρίς το πουλί να αγγίξει τις σωλήνες ή το έδαφος.

Στη δική μας περίπτωση το παιχνίδι ενεργεί κάπως διαφορετικά. Αρχικά αντί για το διάσημο πουλί τώρα έχουμε μια νότα! Ο λόγος γιατί θα χρειαστείς τη φωνή σου. Όταν η φωνή σου θα γίνεται ψηλή/δυνατή η νότα θα πηγαίνει προς τα πάνω , ενώ όταν η φωνή σου θα γίνεται βαριά/χαμηλή η νότα θα πηγαίνει προς τα πάνω. Σε αυτή την έκδοση δεν θα χρειαστεί να φοβηθείς μήπως ακουμπήσεις το έδαφος διότι όταν δεν θα μιλάς το γραφικό θα συνεχίζει με την ίδια ευθεία πορεία στο ύψος που του έχεις δώσει. Πρόσεχε όμως! Γιατί οι σωλήνες ακόμη υπάρχουν και αν τις ακουμπήσεις τότε το παιχνίδι τελειώνει.



Λειτουργικό κομμάτι του παιχνιδιού

Οι βιβλιοθήκες μας

```
import numpy as np #numerical calculations, data analysis, statistical operations
import math #mathematical computations
import aubio #audio analysis, beat tracking, pitch detection
import pyaudio #recording sounds from a microphone
import pygame #developing 2D games, simulations
from pygametexting import pyg_text #our other code
import random #generates random data
from time import time #measuring the duration of processes
```

Η κλάση του παίχτη μας

```
class player():
    def __init__(self, size, x, w_ysize, frames, gravity, scream_force, color, vol_detection ):
        self.size = size #size of the player
        self.x = x #horizontal position
        self.w_ysize = w_ysize #vertical dimension
        self.y = int(w_ysize / 2) #positioning the player vertically in the middle of the game window
        self.frames = frames #frame rate
        self.gravity = gravity * (1 / frames) #adjusting the gravity
        self.vol_detection = vol_detection
        self.scream_force = scream_force
        self.color = color #for the letters ,PLAY BALANCE etc
        self.box = pygame.image.load("Playing_Player.png").convert_alpha() #loading the image

        # Define pitch thresholds for moving up and down
        self.pitch_threshold_low = 500 #if voice<500 hz goes down
        self.pitch_threshold_high = 700 #if voice>700 hz goes up
        #self.is_active = False
```

Καθορίζουμε κατευθύνσεις, αναγνώριση φωνής (scream), ρυθμό παραθύρων, συχνότητες Hz και εισάγουμε την εικόνα του παίκτη. Ορίζουμε κατώφλια για κίνηση προς τα πάνω ή κάτω.

Κλάσεις φωνής και συντεταγμένων παίχτη

```python
def update_based_on_pitch(self, pitch):
    print("Current pitch:", pitch) #Debugging output to check the current pitch
    if pitch > self.pitch_threshold_high:
        print("Moving up")  # Debugging statement
        self.y -= 10  #decrease y position to move up on the screen
    elif pitch < self.pitch_threshold_low:
        print("Moving down")  # Debugging statement
        self.y += 10  #increase y position to move down on the screen

def scream(self, vol):
    #method to react to volume inputs
    pass

def draw(self, window, angle):
    # Draw the player on the screen at the specified rotation angle
    true_box = pygame.transform.rotate(self.box, angle)
    #rotate the player's image
    rotator_pos = true_box.get_rect()
    #get the rectangle that bounds the rotated image
    rotator_pos.center = (self.x, self.y)
    #set the center of the rectangle to the player's position
    window.blit(true_box, rotator_pos)
    #render the rotated image to the screen at the new position
```

Δημιουργούμε 3 μεθόδους. Η 1η προσαρμόζει την κατακόρυφη θέση του παίχτη σύμφωνα με τα Hz της φωνής. Η 2η μέθοδος απλά βοηθάει στην εξασφάλιση της 1ης μεθόδου αλλά και ότι η φωνή μας “ακούστηκε“. Και η 3η μέθοδος στις συντεταγμένες του παίχτη αλλά και να παραμείνει στο κέντρο με τις τρέχουσες συντεταγμένες.

Δημιουργία μεγέθους σωλήνων

```python
class pipe():
    def __init__(self, win_size_x, win_size_y, speed, size, passage_loc, passage_size):

        self.win_size_x = win_size_x #width of the game window
        self.win_size_y = win_size_y #height of the game window
        self.x = win_size_x #initial horizontal position of the pipe, starting at the right edge of the window
        self.speed = speed ##horizontal speed of the pipe, determines how fast it moves across the screen

        self.pipe_img = pygame.image.load("Pipe.png")
        pipe_size = self.pipe_img.get_rect().size[0]
        #get the width of the pipe image

        if size > pipe_size: #set the size of the pip
            self.size = pipe_size
        # Use the image size if the specified size exceeds the image width

        else:
            self.size = size
        #use the specified size if it's within the image width limits

        self.passage_loc = passage_loc
        #vertical position of the top of the gap
        self.passage_size = passage_size
        #height of the gap, controls where the player can pass through
```

Δημιουργούμε πλάτος και ύψος του σωλήνα, εισάγουμε την εικόνα και προσαρμόζουμε το μέγεθός της. Καθορίζουμε την κορυφή του σωλήνα και αλλάζουμε το ύψος του για δυσκολία, ενώ αποθηκεύουμε το σκορ.

Κίνηση σωλήνων

```python
class pipe():

    def pipe_move(self):
        self.x -= self.speed
    #decrease the x position by the speed, moving the pipe to the left

    def draw(self, window):
     #draw the pipe and its gap on the game window
        border = 3
        #set a border width to visually distinguish the pipe edge

        hole = self.passage_loc + self.passage_size
        #calculate the lower boundary of the gap

        pygame.draw.rect(window, (0,0,0), (self.x-border,0,self.size+2*border,self.passage_loc+border))
        #upper part of the pipe
        pygame.draw.rect(window, (0,0,0), (self.x-border,hole-border,self.size+2*border,self.win_size_y-hole+border))
        #lower part of the pipe
        #blit the pipe image to the window
        window.blit(self.pipe_img, (self.x, 0), area = (0, 0, self.size, self.passage_loc))
        window.blit(self.pipe_img, (self.x, hole), area = (0, 0, self.size, self.win_size_y - hole))
# PyAudio object.
#handle real-time audio input
p = pyaudio.PyAudio()
```

```python
p = pyaudio.PyAudio()

RATE = 22100
CHUNK = 1024 #size of each audio buffer

# Open stream.
stream = p.open(format=pyaudio.paFloat32,
    channels=1, rate=RATE, input=True,
    frames_per_buffer=CHUNK) #stream for input (recording)
#number of frames per buffer

pDetection = aubio.pitch("yin", 2048, CHUNK, RATE)
#initialize pitch detection object
pDetection.set_unit("Hz")
#set the unit of pitch to Hertz
pDetection.set_tolerance(0.8)
#set pitch detection tolerance
```

Η μέθοδος pipe_move μετακινεί τον σωλήνα αριστερά, δημιουργώντας την αίσθηση ότι ο παίκτης προχωρά. Η μέθοδος draw σχεδιάζει τον σωλήνα και το κενό που πρέπει να περάσει ο παίκτης, ορίζοντας περίγραμμα για διακριτικότητα και σχεδιάζοντας τα δύο μέρη του σωλήνα.
Αρχικοποιούμε το pyaudio για εγγραφή ήχου σε πραγματικό χρόνο.

Βασικά στοιχεία εκκίνησης και διαχείρισης παιχνιδιού

```python
pygame.init()
#window size for the game
win_sizes = (600,600)

wind = pygame.display.set_mode(win_sizes)
 #create a game window with dimensions 600x600 pixels
pygame.display.set_caption("Flappy Voice")

txt = pyg_text(20,(0,0,0),"comicsansms", Win = wind)
#create a text object for on-screen text

clock = pygame.time.Clock()
#setup game clock for timing control
clock_time = 30
#the game clock to tick at 30 times per second
run = True #game loop
pipe_hole = 300
```

```python
#vertical position of the hole in the pipe
hole_size = 100
#size of the gap in the pipe through which the player must pass
pipe_speed = 8
#speed at which the pipes move leftward
pipe_size = 75
# Width of the pipes
pipe_x_limit = int(win_sizes[0]*(3/5))
 #horizontal limit for spawning new pipes
hole_limit = 50
#minimum vertical limit for the pipe gap
score = 0
```

```python
score = 0
violation = False
#flag to check if scoring is possible
scoring = False
score_detection = False
#flag to detect scoring events
BG = pygame.image.load("BG.png")
you_died_image = pygame.image.load("you_died.png").convert_alpha()

#game states
Menu = True
#show the menu
Play = False
#control gameplay state
Replay = False
#enable replay option
about_to_start = False
#flag to check if the game is about to start
transition = [False,0]
#array to handle transition states and timing
vol_limit = 100
#volume limit for voice input
```

Το Pyaudio χρησιμοποιείται για την αρχικοποίηση του ηχητικού stream, επιτρέποντας την εγγραφή ήχου από μικρόφωνο με ρυθμίσεις όπως format (float32), αριθμός καναλιών (μονοφωνικό), συχνότητα δειγματοληψίας και μέγεθος buffer (CHUNK). Το Aubio ανιχνεύει τον τόνο (pitch) του εισερχόμενου ήχου.
Αρχικοποιούμε το pygame και δημιουργούμε παράθυρο παιχνιδιού 600x600 pixels. Ορίζουμε μεταβλητές για καταστάσεις παιχνιδιού (Menu, Play, Replay), σκορ, ανίχνευση παραβάσεων και έλεγχο σκορ. Προσδιορίζουμε τοποθεσίες, μεγέθη και ταχύτητα των σωλήνων και παραμέτρους για τον υπολογισμό των νέων θέσεών τους.

Κύριος βρόγχος

```python
while run:
    clock.tick(clock_time)
    #control the game update rate (frames per second)
    for event in pygame.event.get():
        ##check for the QUIT event to stop the game
        if event.type == pygame.QUIT:
            run = False
            #check for the QUIT event to stop the game

    if Menu or Replay:

        wind.fill(pygame.Color("blue"))
        #fill the screen with blue color to indicate menu or replay state
        if time()-transition[1]>=0.5:
        #check if half a second has passed to reset transition
            transition = [False,0]

        if about_to_start:
            #create a player object
            pp = player(20, 300, 600, clock_time, 10, 3, (0,0,255), vol_limit)
            pipe_hole = 300
            #initialize pipes with the specified dimensions and speed
            pipes = [pipe(win_sizes[0], win_sizes[1], pipe_speed, pipe_size, pipe_hole, hole_size)]
            score = 0 #reset the score for a new game
            #reset flags to transition into the game play state
```

```python
about_to_start = False
Play = True
Menu = False
Replay = False
```

Ορίζουμε το ρολόι του παιχνιδιού για σταθερό αριθμό καρέ ανά δευτερόλεπτο και ελέγχουμε τα συμβάντα QUIT. Δημιουργούμε το παράθυρο του παιχνιδιού, το γεμίζουμε με μπλε κατά το menu ή replay, και ρυθμίζουμε μηχανισμό μετάβασης κατάστασης κάθε μισό δευτερόλεπτο. Αρχικοποιούμε παίκτη και εμπόδια με θέσεις, ταχύτητες και μεγέθη. Προσθέτουμε λειτουργία ρύθμισης σκορ όταν ο παίκτης χάνει, και διαχειριζόμαστε καταστάσεις παιχνιδιού, μηδενίζοντας το σκορ και αλλάζοντας κατάσταση όταν χρειάζεται.

Εμφάνιση κουμπιών Play Balance, Mic Sensitivity και θέση εμφάνισης τίτλου

```
if Menu:
    #title centered at the top of the screen with colours
    txt.screen_text_centerpos("Flappy Note", int(win_sizes[0]/2), 150, size = 40, color = (255,255,0))
    #display a button and check if it is clicked to start the game
    if txt.screen_button_centerpos("Play Balance", int(win_sizes[0]/2), 300, transition[0],size = 30, color = (255,255,255)) and not transition[0]:
        pygame.time.delay(100)
        #delay to prevent multiple rapid clicks

        about_to_start = True #indicates that the game is about to start

    #display current microphone sensitivity setting
    txt.screen_text_centerpos("Mic Sensitivity: {}".format(vol_limit),int(win_sizes[0]/2), 450, color = (224,224,224))
    #display a button to increase microphone sensitivity and check if it is clicked
    if txt.screen_button_centerpos("More", int(win_sizes[0]/2)-50, 500, transition[0], color = (224,224,224)) and not transition[0]:
        pygame.time.delay(100)

        vol_limit += 100 #increase volume limit by 100

        if vol_limit > 1000: #cap the volume limit at 1000
            vol_limit = 1000

    if txt.screen_button_centerpos("Less", int(win_sizes[0]/2)+50, 500, transition[0], color = (224,224,224)) and not transition[0]:
        pygame.time.delay(100)  #delay to prevent multiple rapid clicks

        vol_limit -= 100

        if vol_limit < 100: #set a minimum volume limit of 100
            vol_limit = 100
```

Ο τίτλος του παιχνιδιού "Flappy Note" εμφανίζεται στο κέντρο, πάνω μέρος της οθόνης, με γραμματοσειρά 40 και κίτρινο χρώμα. Στο κέντρο της οθόνης εμφανίζεται το "Play Balance" και, όταν πατηθεί, θέτει το flag (about_to_start) σε True για έναρξη παιχνιδιού. Υπάρχουν κουμπιά "More" και "Less" κάτω από το "Mic Sensitivity" για ρύθμιση ευαισθησίας μικροφώνου (100-1000). Προσθέτουμε καθυστέρηση 100 ms για να αποτρέψουμε πολλαπλά κλικ και να δώσουμε χρόνο στον χρήστη να δει τις αλλαγές στο UI.

Interface για κατάσταση «Replay» (αφού ο παίκτης έχει χάσει)

```
if Replay: #check if the game is in the 'Replay' state, which indicates the player has lost
    #display a 'game over' image at the center of the window
    image_rect = you_died_image.get_rect(center=(win_sizes[0] / 2, win_sizes[1] /2))
    wind.blit(you_died_image, image_rect)
    #display the player's score
    txt.screen_text_centerpos("Score: {0}".format(score),int(win_sizes[0]/2), 50, size=40,  color = (255,255,255))
    #check if the 'Play Again' button is pressed
    if txt.screen_button_centerpos("Play Again", int(win_sizes[0]*3/4), 550, size=40 , color = (255,255,255)):
        pygame.time.delay(100)  #add a short delay to prevent accidental rapid button presses
        transition = [True,time()] #set transition to True and record the current time

        about_to_start = True #set flag to start a new game
        #check if the 'Back to Menu' button is pressed
    if txt.screen_button_centerpos("Back to Menu", int(win_sizes[0]/4), 550, transition[0], color = (255,255,255)):
        pygame.time.delay(100)
        transition = [True,time()]

        Menu = True  #switch to the Menu state

        Replay = False #turn off the Replay state
```

Όταν ο παίκτης χάσει, εμφανίζεται στο κέντρο του παραθύρου η εικόνα με την πρόταση "Game Over". Πάνω από την εικόνα εμφανίζεται το σκορ και κάτω από αυτήν οι επιλογές "Back to Menu" και "Play Again". Αυτές επιτρέπουν στον παίκτη να ρυθμίσει την ευαισθησία του μικροφώνου ή να ξαναπαίξει με την ίδια ρύθμιση. Οι παράμετροι περιλαμβάνουν θέση (x, y), μέγεθος και χρώμα. Προσθέτουμε καθυστερήσεις για να εξασφαλίσουμε ομαλή λειτουργία του παιχνιδιού και ικανοποιητική εμπειρία χρήστη.

Η διαδικασία του "Play"

```
if Play:
    #blit the background image to the window
    wind.blit(BG, (0, 0))
    #read audio data from the stream
    data = stream.read(CHUNK)
    #convert audio bytes to numpy array for processing
    samples = np.frombuffer(data, dtype=np.float32)
    #detect pitch using the Aubio library
    pitch = pDetection(samples)[0]
    #update the player's position based on the pitch
    if pitch>0:  # If there's a valid pitch detected
        #update the player's position based on the pitch
        pp.update_based_on_pitch(pitch)
    else:
```

```
    else:
        pass
    # Compute the energy (volume) of the
    # current frame
    volume = np.sum(samples**2)/len(samples)
    if volume == 0: #stay at the current position
        pass
    else:
        volume = int(math.log(volume)*10**2)
    #calculate a modified volume used for some game logic
    y_vol = int(volume/2)
    #move each pipe object in the list of pipes
    for i in pipes:
        i.pipe_move()
```

Όταν ο παίκτης πατήσει "Play", φορτώνεται το background και συλλέγονται δεδομένα φωνής (Hz) του χρήστη. Αν αναγνωριστεί pitch άνω των 100 Hz, ενημερώνεται η θέση του παίκτη. Η ένταση του ήχου (volume) υπολογίζεται ως η μέση τετραγωνική τιμή της ενέργειας για κάθε καρέ. Αν η ένταση δεν είναι μηδενική, μετατρέπεται σε λογαριθμική κλίμακα.

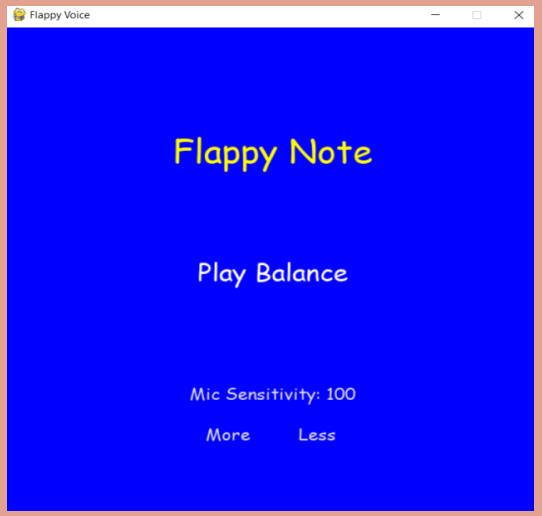
```
if pipes[len(pipes)-1].x <= pipe_x_limit:

    if random.choice([False,True]):
        pipe_hole += random.randint(0,100)
        if pipe_hole > win_sizes[1] - hole_size - hole_limit:
            pipe_hole = win_sizes[1] - hole_size - hole_limit
    else:
        pipe_hole -= random.randint(0,100)
        if pipe_hole < hole_limit:
            pipe_hole = hole_limit
    #append a new pipe to the list with updated position
    pipes.append(pipe(win_sizes[0], win_sizes[1], pipe_speed, pipe_size, pipe_hole, hole_size))

if pipes[0].x < -pipes[0].size:
    pipes.pop(0)
#trigger any special actions if the player screams (based on volume
pp.scream(volume)
scoring = True
#collision detection with pipes
for i in pipes:
    #define half the size of the player for collision calculations
    half_size = int(pp.size/2)
    #check if the player is within the horizontal boundaries of the pipe
    x_cond_1 = (i.x <= pp.x + half_size <= i.x + i.size)
    x_cond_2 = (i.x <= pp.x - half_size <= i.x + i.size)
```

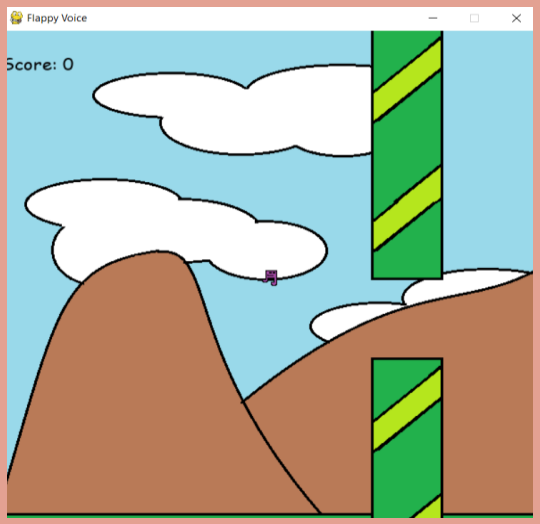
Οι σωλήνες έχουν τυχαία κατανομή ύψους για να δυσκολεύει το παιχνίδι. Αν ο παίκτης περάσει έναν σωλήνα, αυτός κινείται προς την άκρη του παραθύρου για διαγραφή. Για τις συγκρούσεις, υπολογίζεται το μισό μέγεθος του παίκτη για απλοποιημένες συγκρίσεις θέσεων. Οι συνθήκες x_cond_1 και x_cond_2 ελέγχουν αν ο παίκτης βρίσκεται οριζόντια εντός των ορίων του σωλήνα. Αν ισχύουν οι συνθήκες σύγκρουσης, το σκοράρισμα απενεργοποιείται, αποτρέποντας την κερδοφορία πόντων κατά τη διάρκεια του καρέ.

```
if pipes[0].x < -pipes[0].size:
    pipes.pop(0)
 #trigger any special actions if the player screams (based on volume
pp.scream(volume)
scoring = True
#collision detection with pipes
for i in pipes:
    #define half the size of the player for collision calculations
    half_size = int(pp.size/2)
    #check if the player is within the horizontal boundaries of the pipe
    x_cond_1 = (i.x <= pp.x + half_size <= i.x + i.size)
    x_cond_2 = (i.x <= pp.x - half_size <= i.x + i.size)

    if x_cond_1 or x_cond_2:
        scoring = False
        score_detection = True
        y_cond_1 = (i.passage_loc <= pp.y + half_size <= i.passage_loc+ i.passage_size)
        y_cond_2 = (i.passage_loc <= pp.y - half_size <= i.passage_loc+ i.passage_size)

        if  y_cond_1 and y_cond_2:
            pass
        else:  #set the player to have violated the game rules, triggering game over
            #i.color = (255,0,0)
            violation = True
            Play = False
            Replay = True
        break
#handle scoring based on successful passing through pipes
if scoring and score_detection:
    if not violation:
        score += 1
    violation = False
    score_detection = False
```

```
        for i in pipes:  #draw pipes and player
            i.draw(wind)
        pp.draw(wind, 0)
        #display the score
        txt.screen_text_initpos("Score: {0}".format(score),0,25)

    pygame.display.update()

stream.stop_stream() #Stops processing of audio stream
stream.close()
p.terminate() #terminates PyAudio
pygame.quit() #terminate pygame
```

Το ίδιο και για τις συνθήκες y_cond_1 και y_cond_2 μόνο που τώρα ο έλεγχος γίνεται κατακόρυφα. Τέλος προτού κλείσουμε τον κώδικα προσθέτουμε άλλο ένα for η οποία είναι υπεύθυνη για την απεικόνιση των σωλήνων στο παράθυρο παιχνιδιού σε συνάρτηση με τον παίχτη.





Διαχείριση Στοιχείων Κειμένου και Διαδραστικών Κουμπιών

```python
class pyg_text():

    def __init__(self,Size,Color,TXTFont,Win=None,Color_over=(0,100,200),Color_press=(0,100,100)):

        self.Win = Win #παραθυρο κειμενου
        self.Size = Size #μεγευος γραμματοσειρας
        self.Color = Color
        self.TXTFont = TXTFont
        self.Color_over = Color_over #χρώμα
        self.Color_press = Color_press #χρωμα οταν το κειμενο πατιεται
        if Win!=None:
            self.Win = Win
    #συνάρτηση για το κέντραρισμα κειμένου σε μια συγκεκριμένη θέση στο παράθυρο
    def screen_text_centerpos(self,text, x_pos, y_pos, size=None, color=None, txtfont=None, win=None):
        import pygame
        import sys
        #χρησιμοποίηση προκαθορισμένων τιμών αν δεν δοθούν νέες
        if win==None:
            win=self.Win
        if size==None:
            size=self.Size
        if color==None:
            color=self.Color
        if txtfont==None:
            txtfont=self.TXTFont
        #δημιουργία αντικειμένου γραμματοσειράς και απεικόνιση του κειμένου
        text_font = pygame.font.SysFont(txtfont, size)
        rendered_text = text_font.render(str(text), True, color)
        textRect = rendered_text.get_rect()
        textRect.center = (x_pos, y_pos)
        win.blit(rendered_text, textRect) # Τοποθέτηση του κειμένου στο παράθυρο
```

```python
def screen_text_initpos(self,text, x_pos, y_pos, size=None, color=None, txtfont=None, win=None):
    import pygame
    import sys
    # If no window is specified, use the default window set during initialization
    if win==None:
        win=self.Win
    # Use default size if none is specified
    if size==None:
        size=self.Size
    # Use default color if none is specifie
    if color==None:
        color=self.Color
    # Use default font if none is specified
    if txtfont==None:
        txtfont=self.TXTFont

    # Create a font object from the system's fonts
    text_font = pygame.font.SysFont(txtfont, size)
    # Render the text as a surface object with anti-aliasing and the specified color
    rendered_text = text_font.render(str(text), True, color)
    # Get the rectangle bounding box for the rendered text
    textRect = rendered_text.get_rect()
    # Set the position of the text rectangle
    textRect.x = x_pos
    textRect.y = y_pos
    # Draw the text surface onto the window at the specified rectangle position
    win.blit(rendered_text, textRect)
```

Η συνάρτηση λοιπόν χρησιμοποιείται για να τοποθετήσει κείμενο σε συγκεκριμένη θέση στο παράθυρο του παιχνιδιού. Αρχικά, ελέγχει αν δόθηκαν συγκεκριμένες τιμές για το παράθυρο, το μέγεθος της γραμματοσειράς, το χρώμα και το είδος της γραμματοσειράς. Αν δεν έχουν δοθεί, χρησιμοποιεί τις προκαθορισμένες τιμές. Στη συνέχεια, δημιουργεί ένα αντικείμενο γραμματοσειράς με τη συγκεκριμένη γραμματοσειρά και μέγεθος, και δημιουργεί το κείμενο ως γραφικό στοιχείο (surface), το οποίο απεικονίζει με το δοθέν χρώμα. Αφού παραχθεί το κείμενο, καθορίζεται τη θέση του αντικειμένου κειμένου στις συντεταγμένες x και y που έχουν δοθεί και τοποθετείται το γραφικό στοιχείο στο παράθυρο. Επομένως παρέχεται μεγάλη ευελιξία για τη διαχείριση γραφικών και διεπαφών χρήστη σε εφαρμογές και παιχνίδια.

```
def screen_button_centerpos(self,text, x_pos, y_pos, over_effect=False, size=None, color=None,
    color_press=None, txtfont=None, win=None):
    import pygame
    import sys
    # Use default window if none provided
    if win==None:
        win=self.Win
    # Use default size if none provided
    if size==None:
        size=self.Size
    # Use default color if none provided
    if color==None:
        color=self.Color
    # Use default color for press if none provided
    if color_press==None:
        color_press=self.Color_press
    # Use default font if none provided
    if txtfont==None:
        txtfont=self.TXTFont
    text_font = pygame.font.SysFont(txtfont, size)
    rendered_text = text_font.render(str(text), True, (0,0,0))
    textRect = rendered_text.get_rect()
    textlist=list(textRect[2:4])
    # Center of the text for positioning
    x_center = x_pos
    y_center = y_pos
    # Calculate initial and limit positions for interactive behavior
    total_x=int(textlist[0])
    total_y=int(textlist[1])
    x_init=int(x_center-total_x/2)
    y_init=int(y_center-total_y/2)
    x_limit=int(x_center+total_x/2)
    y_limit=int(y_center+total_y/2)
```

```
    n_actions = 0
    # Draw the text at the center position
    self.screen_text_centerpos(text,x_center,y_center, size=size, color = color, txtfont=txtfont, win=win)
    # Check if mouse is over the text and handle interactions
    if x_init<pygame.mouse.get_pos()[0]<x_limit and y_init<pygame.mouse.get_pos()[1]<y_limit:
        n_actions = 1
        self.screen_text_centerpos(text,x_center,y_center, size=size, txtfont=txtfont, win=win)
        if pygame.mouse.get_pressed()[0]:
            n_actions = 2
            self.screen_text_centerpos(text,x_center,y_center, size=size, color = color_press, txtfont=txtfont, win=win)
            pygame.display.update((x_init,y_init),(total_x,total_y))
    if over_effect:
        return n_actions
    # το ποντίκι βρίσκεται πάνω από το κουμπί και έχει γίνει κλικ σε αυτό
    elif n_actions == 2:
        return True # Return True if button was pressed
```

Εδώ δημιουργείται ένα κουμπί με κείμενο στο κέντρο μιας δεδομένης θέσης στο παράθυρο του παιχνιδιού. Αρχικά, ελέγχει αν οι παράμετροι (όπως μέγεθος, χρώμα, γραμματοσειρά και παράθυρο) έχουν δοθεί και, αν όχι, χρησιμοποιεί προκαθορισμένες τιμές. Στη συνέχεια, δημιουργεί το κείμενο και το τοποθετεί στο κέντρο της συγκεκριμένης θέσης. Ελέγχει αν ο δείκτης του ποντικιού είναι πάνω από το κείμενο και αν το κουμπί έχει πατηθεί, αλλάζοντας χρώμα ανάλογα με την αλληλεπίδραση του χρήστη και επιστρέφει ανάλογες ενδείξεις.

```
def screen_button_initpos(self,text, x_pos, y_pos, over_effect=False, size=None, color=None,
    color_press=None, txtfont=None, win=None):
    import pygame
    import sys
    if win==None:
        win=self.Win
    if size==None:
        size=self.Size
    if color==None:
        color=self.Color
    if color_press==None:
        color_press=self.Color_press
    if txtfont==None:
        txtfont=self.TXTFont
    text_font = pygame.font.SysFont(txtfont, size)
    rendered_text = text_font.render(str(text), True, (0,0,0))
    textRect = rendered_text.get_rect()
    textlist=list(textRect[2:4])
    x_init = x_pos
    y_init = y_pos
    total_x=int(textlist[0])
    total_y=int(textlist[1])
    x_center=int(x_init+total_x/2)
    y_center=int(y_init+total_y/2)
    x_limit=int(x_init+total_x)
    y_limit=int(y_init+total_y)
    n_actions = 0
    self.screen_text_initpos(text,x_init,y_init, size=size, color = color, txtfont=txtfont, win=win)
    if x_init<pygame.mouse.get_pos()[0]<x_limit and y_init<pygame.mouse.get_pos()[1]<y_limit:
        n_actions = 1
        self.screen_text_initpos(text,x_init,y_init, size=size, txtfont=txtfont, win=win)
        if pygame.mouse.get_pressed()[0]:
            n_actions = 2
            self.screen_text_initpos(text,x_init,y_init, size=size, color = color_press, txtfont=txtfont, win=win)
            pygame.display.update((x_init,y_init),(total_x,total_y))
    if over_effect:
        return n_actions
    elif n_actions == 2:
        return True
pygame.quit()
```

Η συνάρτηση αυτή απεικονίζει ένα κείμενο στο παράθυρο σε μία αρχική θέση και ελέγχει για ενέργειες του ποντικιού.Ανάλογα με την αλληλεπίδραση του χρήστη, το κείμενο μπορεί να αλλάξει χρώμα, δείχνοντας ότι το κουμπί έχει ενεργοποιηθεί ή πατηθεί, και μπορεί να ενημερώνει την οθόνη για να αντικατοπτρίσει αυτές τις αλλαγές.
Η screen_button_centerpos τοποθετεί ένα κουμπί κειμένου στο κέντρο της θέσης που ορίζετε με τις συντεταγμένες x_pos και y_pos. Το κέντρο του κειμένου είναι ακριβώς σε αυτές τις συντεταγμένες, εξασφαλίζοντας απόλυτο κεντράρισμα στην οθόνη.Αντίστοιχα, η screen_button_initpos τοποθετεί το κουμπί κειμένου χρησιμοποιώντας τις συντεταγμένες x_pos και y_pos ως την αρχική άκρη του κειμένου στην οθόνη.

Ευχαριστώ για την προσοχή σας

ΚΑΛΟ ΚΑΛΟΚΑΙΡΙ