תיכון "הדרים" הוד השרון

Puzzles

שם: יואב מלכה

ת.ז: 213339070

שם בית הספר: תיכון הדרים הוד השרון

שם המנחה: ניר סליקטר

תאריך הגשה: 16.05.2020

תוכן עניינים

מבוא : עמוד -3

המבנה של הפרויקט: עמוד 4

מדריך משתמש : עמוד 7

מדריך מתפתח – עמוד 11

רפלקציה : עמוד 35

מבוא

תיאור תכולת הספר: מבוא הכולל רקע לפרויקט תהליך המחקר הצורך עליו הפרויקט עונה ואתגריים מרכזיים. המבנה של הפרויקט שכולל הסבר על כל המחלקות והיחידות בפרויקט, את זרימת המידע בין היחידות השונות, ארכיטקטורת רשת, קשרים בין יחידות ותיאור האלוגריתם הראשי ותיאור אלגוריתמים עיקריים. מדריך למשתמש הכולל דרישות והוראות התקנה, וקבצים נדרשים וכן צילומי מסך של כל המסכים ותיאור והסבר שלהם, הסבר על כל כפתור והודעות למשתמש. בנוסף גם מדריך למפתח שמרחיב על כל קובץ בפרויקט ורפלקציה.

הרקע לפרויקט: בחרתי כפרויקט משחק תורות שמטרתו היא פתירת פאזל משותפת. בחרתי בפרויקט זה מכיוון שמגיל קטן משפחתי ואני הרכבנו פאזלים ביחד ואני חושב שפאזל זה משחק גם מהנה וגם מעורר מחשבה ויצירתיות. בעקבות תקופת הקורונה, לא יכולתי להיות בקרבה פיזית עם החברים והמשפחה הרחוקה וכך לא יכולתי לשחק איתם במשחק זה ולכן חשבתי איך אפשר לקחת את משחק הילדות הזה שכולם מכירים ולהפוך אותו לאלקטרוני וכך לשחק אותו עם המשפחה והחברים גם כשלא נמצאים פיזית ביחד.

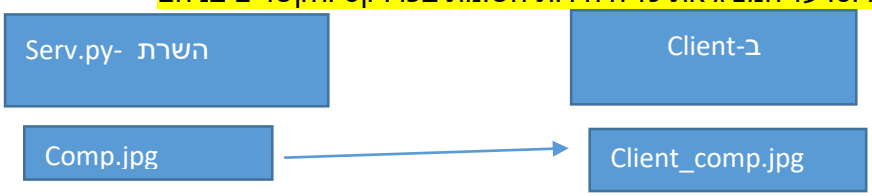
תהליך המחקר: פאזלים הם משחק פיזי וקיימים סוגים רבים בשוק, מפאזלים לקטנטנים עד פאזלים עם מאות חלקים למבוגרים. בדקתי גם על משחקי פאזלים ברשת ומצאתי מספר רב של משחקים אלקטרונים, אך לא מצאתי אף משחק שאפשר להוריד למחשב ולשחק ללא אינטרנט ושגם אפשר לשחק בו מכמה מחשבים בו זמנית, ולפתור פאזל בצורה משותפת, וזהו החידוש שהבאתי לפרויקט שלי ממשחקים אחרים.

אתגרים מרכזיים: האתגר המרכזי איתו התמודדתי הוא החלפת המסכים בהתאם לטכנולוגיה הקיימת בפייתון. Tkinter לא מאפשר כמה חלונות במקביל ולכן נוצרה בעיה עם המסכים. פתרתי זאת באמצעות החלפת המצבים אחרי מעשה. העבודה עם שליחת המידע בצורה היעילה ביותר התאמת הנתונים והצגת המידע בצורה היעילה ביותר שתתאים לפיתוח הפרויקט שהתפתח שתהיה נוחה למשחק וכיף לשימוש בצורה שאוכל להעביר ולסדר את התמונות בצורה טובה וכוללת שתתאים לכל מספרי הפאזל

הצורך עליו הפרויקט עונה: הפריויקט עונה על הצורך לשחק את המשחק המוכר, פאזל, בצורה אלקטרונית וללא אינטרנט.

המבנה של הפרויקט

להתחיל בשרוטו על המציג את כל היחידות השונות בפרויקט והקשרים בניהם



הסבר על איך בנוי הפרויקט

הפרויקט מורכב משני קבצים קובץ השרת(serv.py) וקובץ הלקוח ואנחנו עושים שימוש גם בקובץ התמונה הנמצא אצל השרת בקובץ השרת יש את אובייקט ה server שבעצם עסוק בקבלת הדטא והעברתו מצד לצד כדי לעדכן את שני הצדדים בנוגע הפעולות של השני כל לקוח בעצם מקבל מהשרת את התמונה בתחילת המשחק ואז מקבל את חלקי התמונה הנמצאים אצלו במחסן החלקים ומאז כל לקוח עושה את הפעולה שלו מעדכן את השרת איזה פעולה הוא רצה לשים איפה והשרת מעביר את זה ללקוח השני וחוזר חלילה עד שהשניים יסיימו לפתור את הפאזל באמצעות מיקום כל החלקים במקום הנכון

הקובץ השני הוא קובץ הלקוח(client.py) בעצם הוא אחראי על התצוגה של כל החלקים המוצגים ללקוח ועל כל החלק הוויזואלי הלקוח מקבל את חלקי הפאזל בתחילת המשחק מתוך השרת ואחר כך הוא יוצר חלון תיק אינטר המציג את כל התמונות שבהם הלקוח יכול להשתמש אחר כך המשתמש בוחר באמצעות לחיצה על כפתור שהוא בעצם נמצא באובייקט כפתור(new_but) כדי שיוכל לזכור את המיקום שלו במחסן התמונות ואחרי שהמשתמש בחר תמונה החלון יתחלף לחלון אשר מכיל מקומות בלוח הפאזל כאשר מיקום ריק יוצג בריבוע שחור ומקום מלא יוצג כתמונה הנמצאת כרגע במקום זה ואחרי שיבחר באמצעות לחיצה על המקום בו ירצה לשים את החלק שבחר הלקוח ישלח לשרת איזה חלק הוא בחר לשים באיזה מיקום ואז ינסה לקבל מידע מהשרת אודות פעילות השחקן השני (המשחק **לא פועל בצורת תורות**) השחקן יוכל להמשיך לשחק כשכל סוף פעולה שלו יתעדכן המסך ומיקום החלקים בהתאם לפעילות השחקן השני ולכן יוכלו לחוות תורות של אי סנכרון

מחלקות (תפקיד,קלט,פלט)

GAME: המחלקה הזאת תחילה תקבל את רשימת הסוקטים הפתוחים ואובייקט תמונה ותחילה בפעם הראשונה יתן פלט לכל אחד מהשחקנים אודות איזו תמונות נמצאות במחסנו האישי שלו

ואחר כך במהלך המשחק ישלח להן בצורה של סטרינג את כל אחד מהתמונות הממוקמות על הלוח ואת מיקומה על הלוח בצורה של מחרוזת שאחר כך יפרשו והקלט שהוא יקבל הוא בעצם מחרוזות אותה הוא

מפרש והופך אותה לTUPLE של TUPLES וכך בעצם הוא מוסיף אותו לרשימה שלו ומעביר אותו ללקוח השני

Client– מחלקת הלקוח תחילת הקבלה שלה היא מקבלת אובייקט תמונה את הסוקט שממנו היא תקבל מידע בעצם הסוקט של השרת ומערך של האיברים הנמצאים במחסן שלה ובאמצעותם המשתמש יוכל לשחק במהלך המשחק המחלקה תשלח לשרת את המידע אודות החלק שהיא קיבלה איזה חלק הוא ואת המיקום שהוא ממוקם בלוח הפאזל באמצעות המרה למחרוזת שאותה השרת יפענח והיא מקבלת באותה צורת מחרוזת את המיקום שהצד השני מיקם את חלקיו ומוסיפה אותם למערך המיקומים של הלוח

New_but-מ מחלקה של כפתור המחקבלת בעצם כפתור ואת המיקום שלו בתמונה באמצעות אינדקסים המייצגים את המיקום המחלקה מחזירה בהתאם ללחיצה על הכפתור את מיקום הכפתור כדי שאוכל לייצג את הבחירה של השחקן בחלק שבו הוא רוצה להשתמש ובאיזה מיקום הוא רוצה לשים אותה

הסבר על המערכים/ רשימות שלך

הפרויקט שלי מכיל המון רשימות דו מימדיות שרובן רוב תפקידן הוא להכיל את מיקום החלקים במחסן התמונות ובלוח הפאזל

<u>רשימות הנמצאות בשימוש רב פעמי בSERV :</u>

Open_client_sockets=רשימת צינורות המיד הפתוחים ברגע המשחק משמש כדי לדעת ולזהות את השחקנים במשחק

Self.items= רשימה דו מימדים אשר מכיל בכל אחד מהמיקומים שלה את מיקום החלק(בתמונה) הנמצא בו הרשימה היא רשימה מעורבבת אשר מתבקת מהפעולה MIX אשר מערבבת את החלקים בתמונה

Self.empty- המידע המתקבל מלקוח בעצם מתאר את לוח הפאזל ומכיל בכל מקום את המיקום של איזה חלק בתמונה נמצא כרגע באיזה מיקום על הלוח מכיל tuple of tuples שבעצם בתוך כל תאפל נמצא tupleהמיקום המקורי של החלק בתמונה ו tuple המיקום שלו כרגע על המסך

<u>רשימות הנמצאות בשימוש רב פעמי בClient.py</u>

Items – רשימה דו מימדית המשמת מחסן התמונות לשימוש השחקן מכיל בכל חלק בו את התמונה הנמצאת בו ואת המיקום המקורי שלה בתמונה

Self.forgotten– רשימת החלקים אשר נעשה בהם שימוש במשחק

Anotherone- רשימת החלקים הנמצאים על הלוח בפאזל מכיל תאפל של תאפלים המכילים את המיקום של החלק על המסך ואת המיקום המקורי שלו בתמונה

שאר הרשימות בפרויקט זה הן : mix,allimage,img,panel,banana

ותפקידן הוא תפקיד משותף לעזור לגרפיקה להציג את המידע על המסך כל אחת מכילה ומשמת בחלק אחר בפרויקט חלק מהמרת המידע והצגתו אל המשתמש באמצעות חלון ה tkinter ובאמצעות התמונה המתקבלת מהשרת

זרימת המידע בין היחידות השונות:

המידע עובר בצורה של סטרינג בין השרת ללקוח כאשר הלקוח שולח כל פעם לשרת תאפל של ארבע חלקים אשר הוא מפרק אותו לסטרינג ומפריד בין כל אחד באמצעות "/r/n"

והשרת מעביר את המידע אל הלקוח השני גם באמצעות סטרינג . השרת מוסיף כל פעם את המידע אל תוך רשימה המכיל את כל החלקים הנמצאים על הלוח (empty) ומעביר אותה אחר כך ללקוח עוד פעם באמצעות סטרינג כאשר הוא מפצל כל חתיכת מידע באמצעות "|" וכל ארבע חלקים באמצעות "/r/n" וכך בעצם מעביר את המידע אל הלקוח השני ואחר כך הלקוח מפרש את המידע ומציג אותו על המסך

ארכיטקטורת הרשת:

בעצם צורת הרשת שלי היא צורת רשת של שרת לקוח כאשר השרת בעצם מתייחס לשני הלקוחות ומשמש כאמצעי תקשורת העברת מידע בינהם כאשר הלקוח כל פעם מעביר לשרת מיע והוא מכניס אותו לנתונים שלו ושולח את המידע הכולל לשרת השני

תיאור האלגוריתם הראשי והאלגוריתמים העיקריים

באלגוריתם הראשי בעצם אנחנו לוקחים תמונה ומפרקים אותה ומשתמשים במיקומים שלה בתמונה המקומית כזיהוי התמונה עצמה ובמיקום שלה ואת המיקום הזה אני מעביר דרך השרת מלקוח ללקוח כדי שיוכלו לזהוות את המיקום על המסך

באילו טכנולוגיות נעשה שימוש בפרויקט

בטכנולוגיית הגרפיקה של tkinter בטכנולוגיית התמונה של Pillow שבאמצעותו יכולתי לחתוך את התמונה לחלקים ובטכנולוגיית שרת לקוח וSockets

מדריך משתמש

דרישות התקנה

בשביל להתקין את הפרויקט הדרישות הן פייתון 3.8 והתקנה של ספריית cv2 ו pillow בשביל הגרפיקה והתמונות גם על מחשב השרת וגם על מחשב הלקוח

הוראות התקנה:

מורידים את cv2 ו pillow באמצעות הPIP ומורידים את השרת(serv.py) כ ואת התמונה (comp.jpg) על מחשב אחד באותה התיקייה עלמחשב אחד ומריצים אותו באמצעות פייתון ועל מחשב אחר עושים אותו דבר רק עם קובץ הלקוח(client.py) ומריצים באמצעות פייתון את קובץ הלקוח

הקבצים הנדרשים להורדה :

הקבצים הנדרשים להתקנה הם קובץ הלקוח השרת ותמונת הפאזל :

השחקן צריך להוריד את הקובץ הבא:

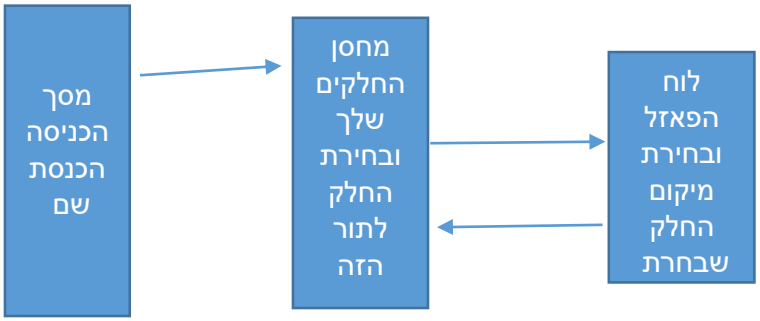
Client.py

והשרת צריך להוריד את הקבצים הבאים :

Comp.jpg

ו serv.py

תרשים המתאר את היררכיית המסכים והמעברים בניהם



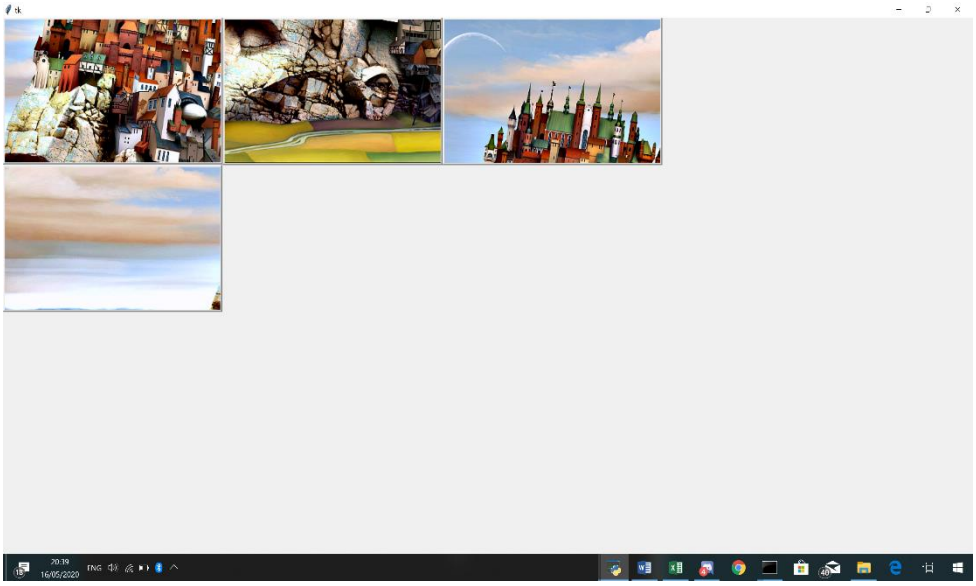
מה תפקיד של כל מסך או חלון עם צילום מסך!

מסך הכניסה והכנסת השם :

```
c:\users\1\desktop\proj>client.py
client begin
what is your name?
yoav

sen



d

sendmesendme
*********************************************now we will start ************************
```

במסך זה השרת מבקש מהלקוח את שמו וכששניהם מוכנים עליו להקליד את המילה sendme וללחוץ על אנטר ויתחיל המשחק ויעבור למסך הבא

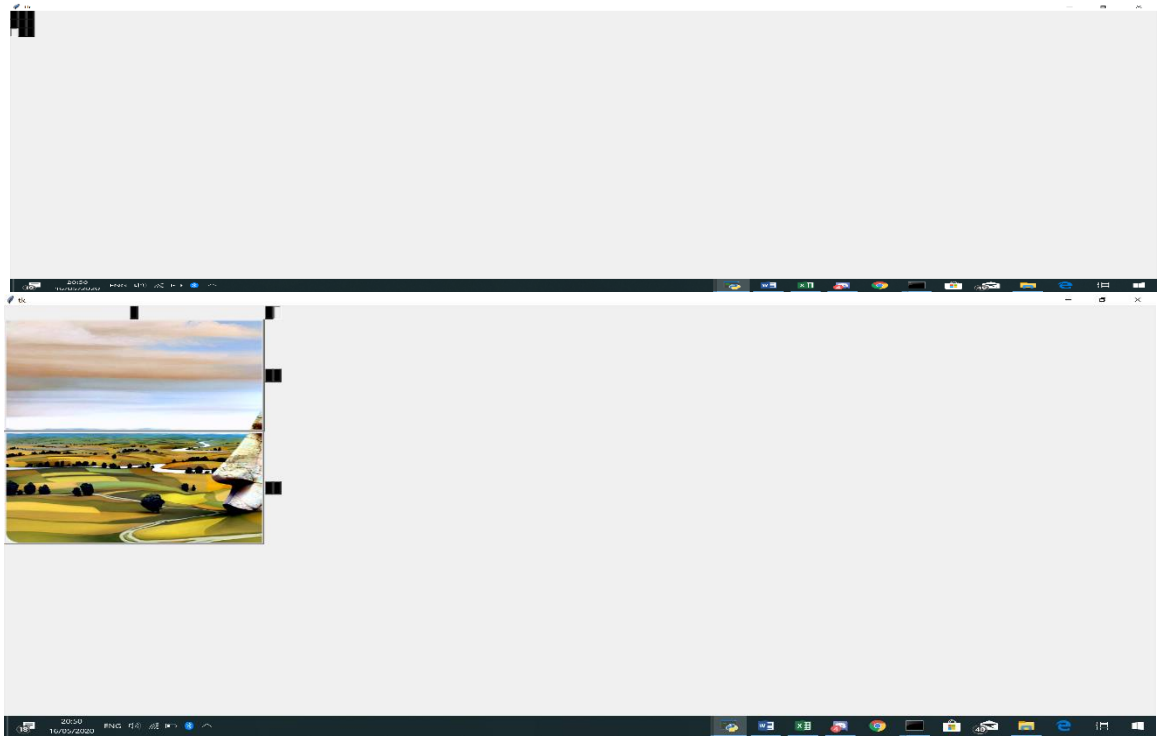
מחסן החלקים שלך ובחירת החלקים לתור הנוכחי :



במסך זה בעצם מופיעים החלקים ששייכם לך במשחק הזה (אשר מחולקים באופן רנדומלי מהשרת)

במסך זה השחקן בוחר את אחת התמונות למקם על לוח הפאזל באמצעות לחיצה על התמונה אשר הוא ירצה למקם כאשר ילחץ על התמונה יועבר ללוח הפאזל ושם יוכל לבחור איפה למקם אותה

לוח הפאזל ובחירת מיקום החלק שבחרת :



במסך זה בעצם מופיעות שתי אופציות שבו המסך יראה המסך בתור הראשון (התמונה העליונה) והמסך במהלך המשחק לא בתור הראשון (התמונה השנייה) כאשר הלוח ריק כל מרום על הלוח מסומן כריבוע שחור הממוקם על הלוח ובכל מקום שתרצה להכניס את החלק בלוח תלחץ עליו ובתור הבא התמונה שבחרת תהיה מוצגת על הלוח אצלך ועל הלוח של הלקוח השני

תפקידם של הכפתורים:

בפרויקט שלי הכפתורים הינם בעצם התמונות והריבועים השחורים אשר מייצגים את המיקום של התמונות ובשביל לבחור בשני המקרים את המיקום שאתה רוצה או לקחת או לשים עליו את התמונה עליך ללחוץ על התמונה בעצם כדי לבחור אותה

ם

הודעות למשתמש :

בפרויקט מופיעה רק בתחילת התהליך ההודעה אשר מבקשת מהשחקן להכניס את שמו

מופיעה הודעה למשתמש בהתחלה ששואלת לשמו וזוהי ההודעה היחידה בפרויקט

הוראות המשחק:

אתה במשחק צריך בתורך לבחור חלק מתמונה מהחלקים המופיעים מולך כדי למקם אותו על לוח התמונה הכללי המורכב מלוח של שלושה על שלושה ריבועים שחורים ואתה צריך למקם אותו בתורך במקום שבו אתה חושב שהוא נכון ושבו החתיכה נמצאת בתמונה המלאה אשר תפתח לך בתחילת המשחק ואז יהיה תור השחקן השני שבו הוא יעשה את אותו הדבר ואז יהיה תורך כאשר בתורך תוכל

לראות את הלוח עם החלקים שגם השחקן השני שם ומטרתכם היא ליצור מכל החלקים את התמונה השלמה שהופיעה לכם על המסך

מדריך מפתח

Serv.py /game

תפקיד : השרת בעצם שומר את התמונה הראשונית ותפקידו הוא להעביר אותה אל הלקוחות ואחר כך להעביר להם את החלקים שלהם מתוך התמונה הכוללת ואחר כך לשמש במתווך בין שניהם ולשמור את המצב של הלוח הכולל משני המשתמשים ולשלוח אל שניהם את מצב הלוח העדכני

הסבר על כל המשתנים והפעולות בקובץ :

self.ImageObject=ImageObject-תמונת פתירת הפאזל

self.open_client_sockets=open_client_sockets-רשימת השחקנים המשתתפים

mix=self.mixing_numbers()המיקום של כל חתיכה בפאזל כפי שהיא נשלחת ללקוח

self.items=[[]]-רשימת חלקי הפאזל המעורביכפי שהם קיבלו ברנדומליות במיקס

self.empty=[]-רשימת החלקים המתקבלת כל פעם מהלקוחות

Sender- משתנה המכיל את המידע שישלח ללקוח אודות מחסן החלקים שלו

messages_to_send- רשימה המכילה את ההודעות שיש לשלוח במהלך המשחק ללקוחות

string- המידע המתקבל על ידי הלקוחות

def send_waiting_messages(wlist, messages_to_send): - פעולה המקבלת רשימה של הודעות ושולחת אותן ללקוחות השונים

def send_waiting_messages1(self,wlist, messages_to_send): - פעולה המקבלת רשימה של הודעות ושולחת אותן ללקוחות השונים

def send_waiting_messages(wlist, messages_to_send): - פעולה המקבלת רשימה של הודעות ושולחת אותן ללקוחות השונים

```
import socket

import select

import sys

import PIL

from PIL import Image,ImageTk

import tkinter as tk
```

```
import random
import time
import msvcrt as m

server_socket=socket.socket()
server_socket.bind(('0.0.0.0',8820))
server_socket.listen(5)
open_client_sockets=[]
messages_to_send=[]

class game:
    def __init__(self,ImageObject,open_client_sockets):
        #self.ImageObject=ImageObjectתמונת פתירת הפאזל
        #self.open_client_sockets=open_client_socketsרשימת השחקנים המשתתפים
        #mix=self.mixing_numbers()המיקום של כל חתיכה בפאזל כפי שהיא נשלחת ללקוח
        #self.items=[[]]רשימת חלקי הפאזל המעורבים כפי שהם קיבלו ברנדומליות במיקס
        #self.empty=[]רשימת החלקים המתקבלת כל  פעם מהלקוחות

        #מכניסים את הערכים של מיקס לתוך אייטמס
        for i in range (len(mix)):
            for t in range (len(mix[i])):
                self.items[i].append(mix[i][t])
            self.items.append([])

        #פעולת ההתחלה בה השרת שולח לכל אחד מהלקוחות מה הם החלקים שלהם
        def start1(self):

            wlist=self.open_client_sockets
```

```
for g in range (len(wlist)):
sender=""
if wlist[0]!=wlist[1]:
mix=self.to_send(g)
for i in range (len(mix)):
for t in range(len(mix[i])):
if t!=(len(mix[i])-1):
sender=sender+str(mix[i][t][0])+","+str(mix[i][t][1])+"/r/n"
else:
sender=sender+str(mix[i][t][0])+","+str(mix[i][t][1])
if i!=(len(mix)-1):
sender=sender+"|||"
wlist[g].send(sender.encode())
```

#תפעולת המשחק עצמו השרת יקבל כל תור את המידע מאחד מהלקוחות וימיר אותו לשורת טקסט בצורה שבה הלקוח יבין וישלח את המידע לשני הלקוחות

```
def play(self):

global server_socket

rlist,wlist,xlist=select.select([server_socket]+self.open_client_sockets,self.open_client_sockets,[]
)
messages_to_send=[]
chachaw=False
for socket in rlist:

while (True):#
string=socket.recv(1024).decode()
if (string!=''):
print ("recveddddd")
arr=string.split("/r/n")
```

```
x=arr[0]
y=arr[1]
x1=arr[2]
y1=arr[3]
tupl=(x,y,x1,y1)
for ttt in self.empty:
if x1 in ttt and y1 in ttt:
chachaw=True
if chachaw==False:
self.empty.append(tupl)
break#
string1=""
for tup in self.empty:


string1=string1+"/r/n"+tup[2]+","+tup[3]+"|"+tup[1]+","+tup[0]
messages_to_send.append((socket,string1))
for my_socket in open_client_sockets: #
if my_socket!=socket: #
my_socket.send(string1.encode()) #
self.send_waiting_messages1(self.open_client_sockets,messages_to_send)
```

פעולת ההכנה לשליחה של חלקי הפאזל ללקוח שולחת לשחקן הראשון חצי ולשחקן השני חצי מהחלקים המעורבבים במיקס

```
def to_send(self,name):
if (name==0):


mix=[[]]
for i in range (1):
for t in range (len(self.items[i])):
```

```
mix[i].append(self.items[i][t])
mix.append([])
mix.append([])
for t in range (1):
mix[1].append(self.items[1][t])
return mix
elif(name==1):
mix1=[[]]
for i in range (2,3):
for t in range (len(self.items[i])):
mix1[i-2].append(self.items[i][t])
mix1.append([])
mix1.append([])
for k in range (1,3):

popai=self.items[1][k]
mix1[1].append(popai)

return mix1
#פעולה היוצרת עירבוב של מיקומים במערך דו מימדי
def mixing_numbers(all_image):
list1=[[]]
list1.append([])
for i in range (3):
for t in range(3):
list1[i].append(t)
list1.append([])
list2=[0,1,2]
mix=[[]]
```

```
Flag = False
for i in range (3):
for t in range (3):
mix[i].append(None)
mix .append([])
for i in range(3):
for f in range (3):

while (Flag==False):
y=random.choice(list2)
m=random.choice(list1[y])

list1[y].remove(m)

if(len(list1[y])==0):
list2.remove(y)

if mix[i][f]==None:

mix[i][f]=(y,m)
Flag=True
else:
print ("byyyyyy")

Flag=False
```

```
return mix


#פעולה המקבלת רשימה של הודעות  ושולחת אותן ללקוחות השונים
def send_waiting_messages1(self,wlist, messages_to_send):
for message in messages_to_send:
(client_socket, data) = message

if it possible to write to the socket #
for my_socket in wlist:
if my_socket!=client_socket:
client_socket.send(data.encode())
print ("sent")
messages_to_send.remove(message)
#פעולה המקבלת רשימה של הודעות  ושולחת אותן ללקוחות השונים
def send_waiting_messages(wlist, messages_to_send):
for message in messages_to_send:
(client_socket, data) = message

if it possible to write to the socket #
if client_socket in wlist:
client_socket.send(data.encode())
messages_to_send.remove(message)
sent=False
preperd=[]
waiting=False
while (not sent):
rlist,wlist,xlist=select.select([server_socket]+open_client_sockets,open_client_sockets,[])
```

```
for current_socket in rlist:
# אם לקוח הוא לקוח חדש יצטרף לתוך רדשימת הלקוחות הפתוחים
if current_socket is server_socket:
(new_socket,adress)=server_socket.accept()
if new_socket not in open_client_sockets:
open_client_sockets.append(new_socket)

else:

# read the data from the client
data = current_socket.recv(1024)

#אם אחד הלקוחותישלח את מילת הקוד אז השרת ישלח לו אותה חזרה
if (data.decode()=="sendme" and len(open_client_sockets)==2):
for ha_socket in open_client_sockets:
ha_socket.send("sendme".encode())
#אחרי שהלקוחקיבלאת מילת הקוד הוא ישלח שהוא מוכן לקבל את התמונה ואז השרת ישלח לו את תמונת הפאזל
if data.decode()=="ready":
preperd.append(current_socket)
waiting=True

if(len(preperd)==len(open_client_sockets)):
with open("comp.jpg", 'rb') as filesent:
data = filesent.read()
file_len = len(data)

msg_data1 = file_len
```

```
msg = str(msg_data1)

msg = msg.encode('latin-1')
for socket in open_client_sockets:
socket.send(msg)
data = data.encode('latin-1')#
for socket in open_client_sockets:
if socket in wlist:
socket.sendall(data)

wait here untill finish to send the whole fil#
sent=Trueממן שהשליחהקרתה ומפסיק את הלולאה ויתחיל את המשחק#

elif(waiting==False):
for the_socket in open_client_sockets:

add the message from the client to the list of the messages that weren't sent #
messages_to_send.append((the_socket, data.decode()))

send_waiting_messages(wlist,messages_to_send)
imageObject=Image.open("comp.jpg")
```

```
g1=game(imageObject,open_client_sockets)
#שליחה ללקוחות את רשימת החלקים שלהם
g1.start1()
while True:
#תחילת המשחק
    g1.play()
```

המשתנים העיקריים והפונקציות של הלקוח :

self.image-התמונה הנמצאתבכפתור

self.tup- מיקום הכפתור במערך הכפתורים

self.okVar=okVar-משתנה המודיע על לחיצת הכפתור

xxx1-שורת הכפתור הלחוץ כרגע

yyy1-תור הכפתור הלחוץ כרגע

self.name-שם המשתמש

self.items-מערך דו מימדי המכיל את כל החלקים

self.imageObject-שם תמונת הפאזל

self.all_image-התמונה הראשונית מחלוקת לחלקים

self.items-מחסן המשתמשים

self.root-חלון הגרפיקה

self.forgotten-רשימת החלקים שבהם עשינו שימוש במהלך המשחק

self.anotherone-רשימת החלקים המופיע על לוח הפאזל ואיזה חלקים אלו

tuple_Array-מערך של כל אחד מהחלקים במיקומים כל פעם שהשרת שולח

panel- מערך הכפתורים הנמצאים על המסך

self.server_socket- השרת שממנו מתקבל המידע

file_len- אורך הקובץ הנשלח

msg11- ההודעה שהתקבלה מהשרת בעצם נתוני התמונה

Filesent- הקובץ שנשלח

Order- הסדר של האיברים הנמצאים במחסן

פעולות :

במחלקה : new_but

def ret_but(self)- משתנה המחזיר את

def ret_img(self)- מחזיר את תהמונה הכפתור

def ret_tup(self,okVar,xxx,yyy)- מחזיר את מיקום הכפתור

def start_game(string)- פעולת קבלת הסדר של המחסן בתחילת המשחק

כלליות :

:def start_game(string)- פעולת קבלת הסדר של המחסן בתחילת המשחק

פעולות ב: client

def image_crop(self,x1,y1,imageObject)- פעולה הלוקחת תמונה וחותכת אותה לחלק לפי נתונים

def handling_server_msg(self,string)- פעולה המפענחת את המיגע המתקבל מהלקוח ומעדכנת את רשימת הנוכחים על המסך

:def create_panel(self)- הפעולה היוצרת את הלוח הראשוני ומוסיפה אליו את התמונות ואת הכפתורים יוצרת רשימה של כפתורים ואת מסך המחסן

def check(self,string)- פעולה הבודקת האם המשתתפים סיימו את הפאזל

def crop_Array(self,imageObject)- פעולה היוצרת מערך של תמונות חתוכות אשר משלימות את התמונה

def show_items(self)- פעולה המחזירה את רשימת ה items

def game(self)- הפעולה שבה בעצם קורה המשחק מחכה ללחיצת השחקן על התמונה ואז משנה את המסך למסך לוח הפאזל ומחכה לכך שיבחר מיקום על המסך ואחרי הבחירה שלו מחזירה למסך המחסן ושולחת את המידע על התזוזה לשרת

הקוד :

```
import socket

import msvcrt

import select

import PIL

from PIL import Image,ImageTk

import tkinter as tk

import random

import time

import msvcrt as m
```

```
print ("client begin")
my_socket = socket.socket()
my_socket.connect(("127.0.0.1", 8820))
LARGE_FONT= ("Verdana", 12)
sent1=False
xxx1=-1
yyy1=-1
class new_but:
    def __init__(self,image1,root,g,b,okVar,xxx,yyy):
        self.Butt=tk.Button(root, image = image1,command=lambda:self.ret_tup(okVar,xxx,yyy))
        #self.image=image1התמונה הנמצאתבכפתור
        #self.tup=(g,b)מיקום כל כפתור
        #self.okVar=okVarמשתנה המודיע על לחיצת הכפתור
    # משתנה המחזיר את הכפתור
    def ret_but(self):
        return self.Butt
    #מחזיר את תהמונה הכפתור
    def ret_img(self):
        return self.image
    # מחזיר את מיקום הכפתור
    def ret_tup(self,okVar,xxx,yyy):
        okVar.set(1)
        print (self.tup)
        #global xxx1מיקום הכתור הלחוץ כרגע

        #global yyy1מיקום הכפתור הלחוץ כרגע
        xxx1=self.tup[0]
        yyy1=self.tup[1]
        return self.tup
```

```
# פעולת קבלת הסדר של המחסן בתחילת המשחק
def start_game(string):
    order=[[]]
    arr=string.split("|||")

    for i in range(len(arr)):
        tup=arr[i].split("/r/n")

        for t in range (len(tup)):
            if tup[t]!="":
                x=tup[t].split(",")[0]
                print ("x",x)#
                y=tup[t].split(",")[1]

                print ("y:",y)#
                x=int(x)
                y=int(y)
                order[i].append((x,y))
            order.append([])
    return order

class client:
    def __init__(self,name,imageObject,order,server_socket):
        self.name=name#שם המשתמש
        self.items=[[]]#מערך דו מימדי המכיל את כל החלקים
        self.imageObject=imageObject#שם תמונת הפאזל
        self.all_image=self.crop_Array(imageObject)#התמונה הראשונית מחלוקת לחלקים
        self.items=[[]]#מחסן המשתמשים
        self.server_socket=server_socket
```

```
#self.root=tk.Tk()חלון הגרפיקה
self.lastdata=""
#self.forgotten=[]רשימת החלקים שבהם עשינו שימוש במהלך המשחק
#self.anotherone=[]רשימת  החלקים המופיע על לוח הפאזל ואיזה חלקים אלו
for i in range(len(order)):
    for t in range (len(order[i])):
        x1,y1=order[i][t]
        self.items[i].append((self.all_image[x1][y1],(x1,y1)))
    self.items.append([])
#פעולה הלוקחת תמונה וחותכת אותה לחלק
def image_crop(self,x1,y1,imageObject):
    cropped = imageObject.crop((x1,y1,x1+(imageObject.size[0]/3),y1+imageObject.size[1]/3))
    return cropped
#פעולה המפענחת את המיגע המתקבל מהלקוח ומעדכנת את רשימת הנוכחים על המסך
def handling_server_msg(self,string):
    self.anotherone=[]
    images=[[]]
    for i in range (3):
        images.append([])
        for t in range (3):
            images[i].append("x")

    if(self.check(string)):
        print ("you wonnnnnnnn")
        tuple_Array=string.split("/r/n")

        print (self.anotherone)
        for tup in tuple_Array:
            if(tup!=""):
```

```
print ("tup:"+tup)
arr=tup.split("|")
x=arr[0].split(",")[0]
y=arr[0].split(",")[1]
x1=arr[1].split(",")[0]
y1=arr[1].split(",")[1]
if (len(x)>1):
print("innnnnnnnnn",x[0])
x=x[1]
if (len(y)>1):
print("innnnnnnnnn")
y=y[1]
print
if (len(y1)>1):
print("innnnnnnnnn",y1[0])
y1=y1[1]
if(len(x1)>3):
print("innnnnnnnnn",x1[0])
x1=x1[1]
if int(x)<3 and int(y)<3 and int(x1)<3 and int(y1)<3:
x=int(x)
y=int(y)
x1=int(x1)
y1=int(y1)
images[x][y]=(x1,y1)
tapir=((x,y),(x1,y1))
if tapir not in self.anotherone:
self.anotherone.append(tapir)
```

הפעולה היוצרת את הלוח הראשוני ומוסיפה אליו את התמונות ואת הכפתורים יוצרת רשימה של כפתורים ואת מסך המחסן

```
def create_panel(self):
    img=[[]]
    mix=self.items

    for label in self.root.grid_slaves():

        label.config(image="",background="black")



    all_image=[[]]
    for i in range (len(mix)):
        for t in range (len(mix[i])):
            all_image[i].append(mix[i][t][0])

    all_image.append([])
    for i in range (len(all_image)):
        for t in range (len(all_image[i])):

            img[i].append(ImageTk.PhotoImage(all_image[i][t]))



    img.append([])

    global okVar
```

```
xxx1=-1
yyy1=-1
okVar = tk.IntVar()
panel=[[]]
for g in range(len(img)):
for b in range (len(img[g])):
print(g,b)

panel[g].append(new_but(img[g][b],self.root,g,b,okVar,xxx1,yyy1))

panel[g][b].ret_but().grid(row=g,column=b)
panel.append([])
for label in self.root.grid_slaves():
kkk=(int(label.grid_info()["row"]),int(label.grid_info()["column"]))
if (kkk in self.forgotten):
flag=True
label.config(image='',background="black")#


self.root.after(10,self.game)
self.root.mainloop()
#פעולה הבודקת האם המשתתפים סיימו את הפאזל
def check(self,string):
tuple_Array=string.split("/r/n")

for tup in tuple_Array:
if(tup!=""):
print ("tup:"+tup)
arr=tup.split("|")
```

```
x=arr[0].split(",")[0]
y=arr[0].split(",")[1]
x1=arr[1].split(",")[0]
y1=arr[1].split(",")[1]
if (x,y)!=(x1,y1):
return False
if (len(tuple_Array)<9):
return False
return True

#פעולה היוצרת מערך של תמונות חתוכות אשר משלימות את התמונה
def crop_Array(self,imageObject):
x1=0
y1=0
all_image=[[]]
for i in range (3):
for t in range (3):

all_image[i].append(self.image_crop(x1,y1,imageObject))
y1=y1+(imageObject.size[1]/3)
y1=0
x1=x1+(imageObject.size[0]/3)
all_image.append([])
return all_image
#פעולה המחזירה את רשימת ה items
def show_items(self):

return self.items
```

#הפעולה שבה בעצם קורה המשחק מחכה ללחיצת השחקן על התמונה ואז משנה את המסך למסך לוח הפאזל ומחכה לכך שיבחר מיקום על המסך ואחרי הבחירה שלו מחזירה למסך המחסן

#ושולחת את המידע על התזוזה לשרת

```
def game(self):
socket=self.server_socket
flag=False
global okVar
num=0
banana=[[]]
self.root.wait_variable(okVar)
del okVar
okVar=tk.IntVar()
print (self.anotherone)
while (flag==False):
i=xxx1
t=yyy1
for a in range(len(self.all_image)):
for n in range(len(self.all_image[a])):
banana[a].append(self.all_image[a][n])
banana.append([])


for y in range (len(banana)):
for m in range (len(banana[y])):
for choice in self.anotherone:
if (y,m)==choice[0]:
(s,h)=choice[1]

banana[y][m]=banana[s][h]
```

```
                    banana[s][h]=self.all_image[y][m]

                                img=[[]]
                              for o in range (3):
                            for c in range (3):

            img[o].append(ImageTk.PhotoImage(banana[o][c]))
                                 img.append([])

                                 for g in range(3):
                            for b in range (3):

        panel = new_but(img[g][b],self.root,b,g,okVar,xxx1,yyy1)

                        panel.ret_but().grid(row=b,column=g)
                          for label in self.root.grid_slaves():
                                   flagy=False
(s,h)= (int(label.grid_info()["row"]),int(label.grid_info()["column"]))

                        for choice1 in self.anotherone:
                              if (s,h)==choice1[0]:
                                   flagy=True
                                  if flagy==False:
                 label.config(image="" ,background="black")

                          self.root.wait_variable(okVar)
                                   del okVar
                                    x1=xxx1
                                    y1=yyy1
```

```
self.forgotten.append((i,t))
try:
print ("$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$")
for label in self.root.grid_slaves():
kkk=(int(label.grid_info()["row"]),int(label.grid_info()["column"]))
if (kkk in self.forgotten):
flag=True
label.config(image="",background="black")##
if (kkk==(i,t)):

str1=(str(self.items[i][t][1][0])+"/r/n"+str(self.items[i][t][1][1])+"/r/n"+str(x1)+"/r/n"+str(y1))
socket.send(str1.encode())
print ("sennttttttttttttttttttttttttttttttttt")


if flag==False:
print ("your numbers were wrong")
except:
print ("your numbers were wronggggggggggggggggggggggggggggggggggggggg")




while True:#
print ("reciving")
data=socket.recv(1024)
if data.decode()!='':
print (data)
self.handling_server_msg(data.decode())
print ("outtttttttttttttttttttttttttt")
break#
```

```
self.lastdata=self.lastdata+data.decode()

self.create_panel()



print ('what is your name?')
name = input()
f = open('torecv.jpg','wb')#

while(sent1==False):
select #
rlist, wlist, xlist = select.select([my_socket], [my_socket], [])

word = ('')
send = False
waiting for a key press #
while msvcrt.kbhit() and sent1==False:
input the command #
char=msvcrt.getch()
word=char.decode()
input the message #
while ord(char) != 13 and sent1==False:
char = msvcrt.getch()

if ord(char) != 13:
word = word + char.decode()
print (word)#
```

```
send = True
print ('')

if send:
val =  word
if my_socket in wlist:
my_socket.send(val.encode())

if my_socket in rlist and sent1==False:

data = my_socket.recv(1024)
#תהליך קבלת התמונה מהשרת
print (data.decode())
if (data.decode()=="sendmesendme"):

my_socket.send("ready".encode())
sent1=True
file_len=my_socket.recv(1024)
file_len=int(file_len)
cnt = 0
msg11 = b""
while cnt < file_len:
```

```
data = my_socket.recv(file_len)  #in some computers we need to collect the whole
                                                                    data in lot of packets
                                                        data = data.decode('latin-1')#

                                  print ("client_rcv_data cnt:{0} len:{1}".format(cnt,len(data)))#

                                                                    print (data)#
                                                            msg11 = msg11 + data
                                                              cnt = cnt + len(data)
                                                                  data = msg11
                                                                    sent1=True
                                                  with open("client_File.jpg" , 'wb') as filesent:
                                                              filesent.write(data)
                                                                    sent1=True

                  print ("**************************************************now we will start
                                                                  *************************")
                                                                    while True:
                                                              data=my_socket.recv(1024)
                                                                  if(data.decode()!=""):
                                                        order=start_game(data.decode())
                                                                    break
                                                    imageObject=Image.open("client_File.jpg")
                                                  c1=client(name,imageObject,order,my_socket)
                                                                  c1.create_panel(
```

רפלקציה

איך הייתה העבודה על הפרויקט: העבודה על הפרויקט הייתה מאתגרת אך מהנה, זוהי הייתה עבודה קשה בסדר גודל שלא הכרתי עד היום ובנושא שרבות ממנו היה חדש לי. זהו הפרויקט הראשון במסגרת הלימודים בו אני מתבקש ללמוד חומר לבדי ולבצע פרויקט בסדר גודל של 5 יחידות לימוד. יחד עם זאת למדתי הרבה על תכנות בפייתון ועל רשתות והחומר היה מעניין מאוד ולכן נהנתי מהלמידה.

הכלים שלקחתי להמשך: למדתי איך לעשות פרויקט ברמה כזו. למדתי איך לומדים כמות גדולה של חומר מסוג זה באופן עצמאי (כמובן עם עזרה מהמורה אך לא בשיעור פרונטלי) ואיך מיישמים אותו. למדתי איך לעשות פרויקט אמיתי בתכנות ברמה גבוהה ועם מטרה לחיי היום יום ולא רק לשם הפרויקט.

הקשים והאתגרים היוא העברת המידע מכיוון שמדובר במידע מסובך יחסית מכיוון שלכל מידע יש בעצם ארבע פרמטרים שניים המיועדים על המיקוף הנוכחי ש החלק ושניים על המיקום הראשוני שלו בתמונה המקומית וכדי להעביר את המידע הזה בצורה טובה ולשמור על עקביות הייתי צריך לעבוד בזהירות ולדאוג לא לפגוע בו ולא לפרש אותו לא נכון מכיוון שהמון חלקים של הקובץ תלוים בפירוש המידע הזה ואם חלק אחד לא יפרש נכון הכל ישתנה בנוסף מערכת הטיקאינטר שאני עובד איתה הייתה מיושנת ומסובכל יחסית ועליי היה להבין כיצד לעשות ולהשתמש בהצגת תמונה וגם בנוסף בתקשורת עם המשתמש בצורה הכי טובה שאפשר בהתאם למגבלות המערכת

מה הייתי ועשה אחרת אם הייתי מתחיל היום: הייתי מסדר את הזמן שלי יותר טוב ופורס את העבודה על משך תקופה ארוכה יותר. בנוסף העבודה התפתחה לכיוונים שונים שלא תכננתי בהתחלה, ואם הייתי יודע זאת מראש הייתי יכול לייעל את הזמן שלי יותר למרות זאת אני שמח שלא ידעתי מראש כי על אף שלא השתמשתי בידע זה בפרויקט למדתי הרבה דברים חדשים ומעניינים שגם יכולים לשמש אותי בעתיד, וכן גם למדתי איך להתמודד עם מצבים כאלה שעלולים לקרות לי גם בעתיד. בנוסף הייתי צריך להסתכל על כל התמונה הגדולה ולתכנן יותר את הפרויקט מראש כי שינויים גם אם הם טובים מאוד קשים להחיל על פרוקיט עובד מכיוון שמערכת שכבר פועלת אינה כל כך גמישה ולכן היו דברים שהייתי מעדיף לעשות יותר טוב אבל כדי שלא לפרק את הפרויקט הנוכחי לא יישמתי אתה

מה היה גורם לעבודה להיות יעילה יותר: אם הייתי מציב מטרות יותר ברורות ופורס את העבודה לאורך יותר זמן. בנוסף הייתי מוודא מראש מה צריך להיות לי בפרויקט ולכך לא הייתי צריך לגלוש לכיוונים אחרים באמצע העבודה אלה להתמקד רק במה שאני זקוק לו להשלמת הפרויקט.