



נושא העבודה - משחק רשת

דין ג'יי מאוי - 324209576

תיכון מקיף יהוד - יהוד מונסון

מורה מלווה: אשר גאווי

תאריך הגשה: 30.3.2019

תוכן עניינים

הערה: תיק הפרויקט לא כולל בדיקות מכיון שנכתב אחרי סיום החלק המעשי ולכן לא הייתה אפשרות לתעד את הבדיקות

3	תיאור המשחק
4	הגדרת יעדים
5	הגדרת הלקוח
5	תיחום הפרויקט
6	בעיות, תועלות וחסכונות
7	אתגרים בהגדרת המערכת
8	פרוט יכולות המערכת
9	תיאור הארכיטקטורה של המערכת המוצעת
11	תיאור הטכנולוגיה הרלוונטית
11	תיאור מודולים בהם נעשה שימוש
13	תיאור סביבת הפיתוח
14	תיאור מבני הנתונים
16	הקוד
20	מדריך למשתמש
25	מבט אישי על העבודה ועל תהליך פיתוחה
26	ביבליוגרפיה

תיאור המשחק

הנושא שבחרתי הוא משחק 2D רב משתתפים. המשחק נבנה בסביבת פיתוח PyCharm בשפה Python עם שימוש בספריון מגוונות כמו cPickle, Pygame ועוד.

המשחק שבחרתי לבנות הוא Draw My Thing מכיוון שהחלטתי שאני לא מעוניין להתמקד בצד הגרפי של המשחק אלא בצד שרת ולהשתמש במשחק כמעין סביבת בדיקה לשרת. Draw My Thing הוא משחק בו כמה שחקנים שנמצאים בלובי משותף רואים את אותו משחק בו זמנית. אחד השחקנים מצייר חפץ שנבחר ע"י השרת באקראיות, שאר השחקנים מנסים לנחש מהו החפץ שהשחקן מנסה לצייר.

כבר קיימת גרסה מוכרת של Draw My Thing ממנה לקחתי את הרעיון הכללי של GUI המשחק במשחק אדם מבצעה את מה שתיארת מלמלע ועל כך ניתן לו ניקוד האדם עם הניקוד הגבוה ביותר מנצח.

הגרסה המוגמרת של המשחק צריכה לתמוך בלובים מרובים ובהם מספר לא מוגבל של שחקנים (תיאורטית).

הגדרת יעדים

● צד לקוח

○ ממשק

- המשחק צריך להכיל תפריט בו השחקן יכול לבחור אם להתחבר ללובי קיים או ליצור לובי חדש
- במשחק צריך להיות צ'אט
- המשחק צריך להציג את הציור שמצויר בזמן אמת
- המשחק צריך להציג מי האדם שניחש ראשון את מה שצויר במקצה הקודם

○ תקשורת

- צד לקוח צריך להתחבר לשרת מרוחק

● צד שרת

- השרת אמור לעבוד במקביל עם כמות לא מוגבלת של לקוחות בצורה יעילה (שימוש בprocessi threds)
- השרת צריך להתייחס ולדעת להפריד כל לובי אחד מהשני
- השרת צריך לגבות את עצמו
- השרת צריך להיות מסוגל להתחיל מאותה נקודה שבה הוא כובה אם נכבה בצורה ראויה או שחזור מאחד הגיבויים
- השרת צריך להכיל \log של המצב בו הוא כובה

הגדרת הלקוח

המשחק מיועד למשתמשי כל מערכות ההפעלה, במערכת יוכל להשתמש כל אדם החפץ לשחק במשחק Draw My Thing.

הלקוח השתמש במערכת ע"י פתיחת הקליינט התחברות לשרת ומשחק משותף דרך הרשת עם חברו.

תיחום הפרויקט

הפרויקט עוסק התחומים הבאים:

● הכרחי

- ממשק דו מימדי, צד לקוח
- תקשורת בסיסית בין שרת ללקוח

● חשוב

- ניהול לובים מרובים, צד שרת (threading)

● רצוי

- שליח יעילה של המידע (buffering)
- שמירת בסיס נתונים בצורה יעילה

בעיות, תועלות וחסכונות

אין בעיה מוגדרת, כלומר הפרויקט שלי לא מנסה לפתור בעיה ממשית שקיימת בעולם אלא אני פיתחתי בעיות לעצמי עם הזמן בכך שהוספתי עוד אלמנטים לפרויקט, מטרת הפרויקט בעיני היא לפתח את הידע שלי ולא לפתור בעיה קיימת הרי הסבירות שפרויקט שאני מכין בעצמי יעלה מעל תוכנות שנבנו ע"י עשרות אנשים בסביבה מקצועית הוא נמוך עד מאוד.

דוגמא לבעיות:

- החלטתי ששרת אחד אמור להחזיק מספר רב של לובים, איך ללמש את זה בצורה אפקטיבית?
 - איך לעשות buffering בלי ספריות?
- ועוד בעיות רבות שצצו על הדרך עם ההתקדמות בפרויקט

שירותים שהמערכת תיתן:

- לפתוח שרת על כל מחשב
- לשחק במשחק עם חברים

השוואה יישומים קיימים:

כמובן שהמשחק כבר קיים, הממשק של המשחק הקיים הרבה יותר מלוטש ויש למשחק עצמו יותר אפשרויות ציור וכו' אך מבחינת שרת המשחק לא מאובטח ואין באפשרות השחקן לבחור להריץ שרת פרטי. המשחק שלי פותר את הבעיות הללו.

אתגרים בהגדרת המערכת

הממשק של המשחק נבנה ע"י שימוש בספריית Pygame ספריה מוכרת ומתועדת היטב
צד שרת נבנה ע"י שימוש בספריית sockets ספריית מובנת לפיתוח ולכן גם היא
מתועדת היטב

בנוסף נעשה שימוש בספריות נוספות המובנות לפיתוח כמו: CPickle, pprint, time, random, sys, threading, select ועוד. רמת התייעוד של הספריות מגוונת ולכן גם הזמן
שנדרש לי ללמוד אותם משתנה.

כמובן שהספריות שנבחרו מגבילות את יכולת התמרות, אך הם נבחרו בקפידה ולכן עונות
על כל הקריטריונים הנדרשים ולא מגבילות ממשית את יישום הפרויקט.

אתגרי הלמידה: כל החומר בפרויקט הוא חומר אשר נלמד אישית ללא עזרה או ליווי של
גורם חיצוני, בנוסף אין חומר לימודי מסודר על הנושאים. לכן הלמידה מאתגרת ודורשת
חיפוש מעיק ברשת של מאגרי מידע ודוגמאות של שימוש באלגוריתמים ובספריות
שנבחר להשתמש בהם.

פרוט יכולות המערכת

ממשק לקוח:

- אפשרות תמרון במשחק והתחברות ע"י ממשק גרפי אל השרת.
- צ'אט המורץ בצד ובוא השחקן יכול להקליד את הניחושים שלו ולראות ניחושים של יריבים
- הקנבס עליו האדם שמצייר יכול לצייר ואותו רואים שאר השחקנים בלובי הממשק רץ על pygame ומשתמש בsurface המוגדר מראש ועליו מרונדרים האובייקטים השונים, הציור מרונדר כאוסף עיגולים על המסך

ממשק שרת:

- השר, רץ עצמאית מרגע הפעלתו, המשתמש יכול להכניס פקודות לממשק טקסט
- ע"י הפקודות המשתמש יכול לגבות את השרת לכבות אותו ועוד.
- הממשק מודיע על התחברות התנתקות וכו'

רשת לקוח:

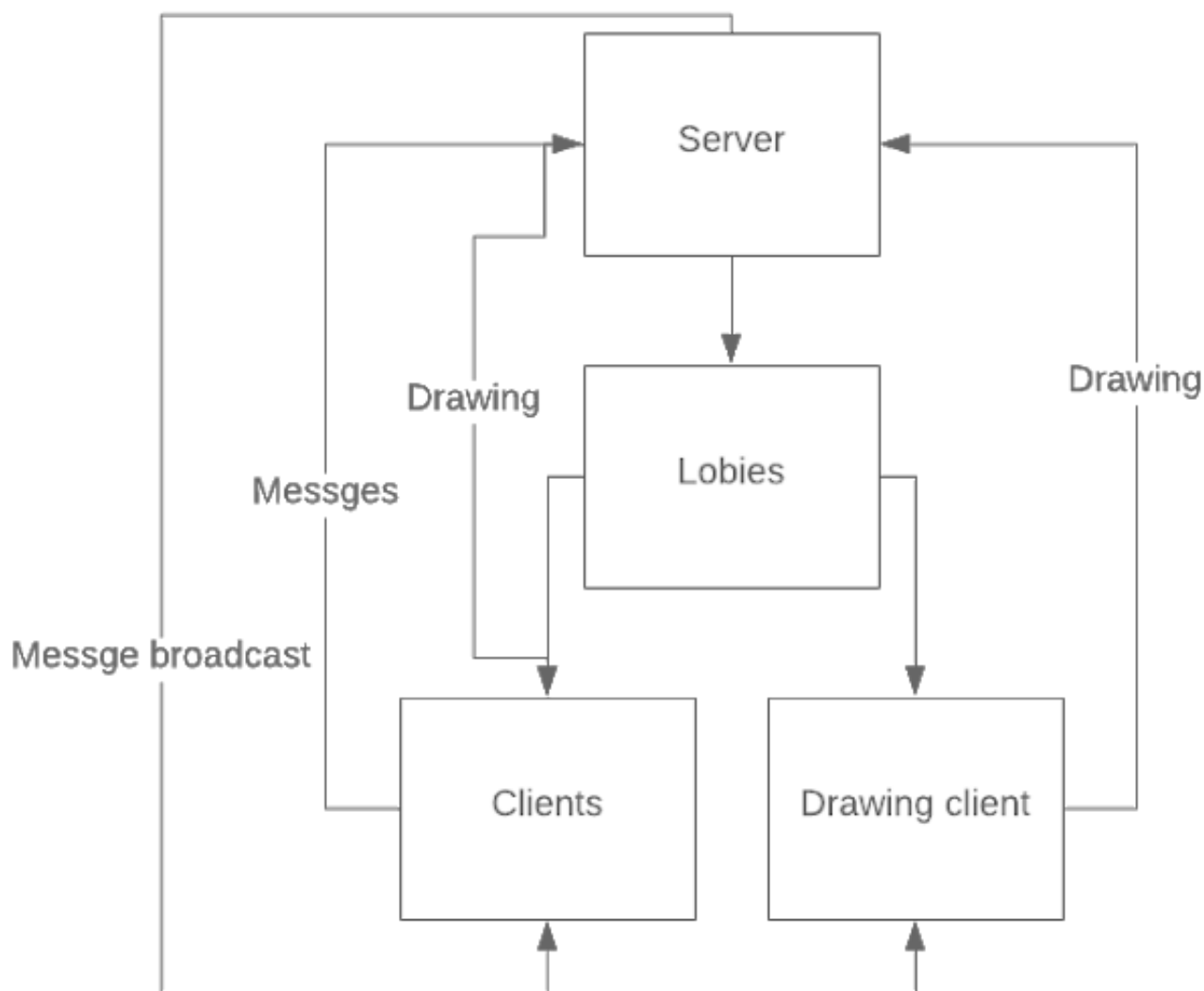
- הלקוח מתחבר לשרת ע"י סוקטים ומשם מתקשר ללא הפסקה עם השרת, המטרה היא שליחת הציור מקצה לקצה ועדכון הצ'אט.

רשת שרת:

- השרת אמור להתחבר למספר לא מוגבל של משתתפים ולקשר ביניהם בצורה שנבחר
- השרת שומר את המידע בבסיס נתונים
- השרת מקבל סוקטים נכנסים ומשם מתקשר ללא הפסקה עם הלקוחות, המטרה היא שליחת כל המידע בצורה מוגנת ללא הפסקה בצורה אפקטיבית בין כל הלקוחות.

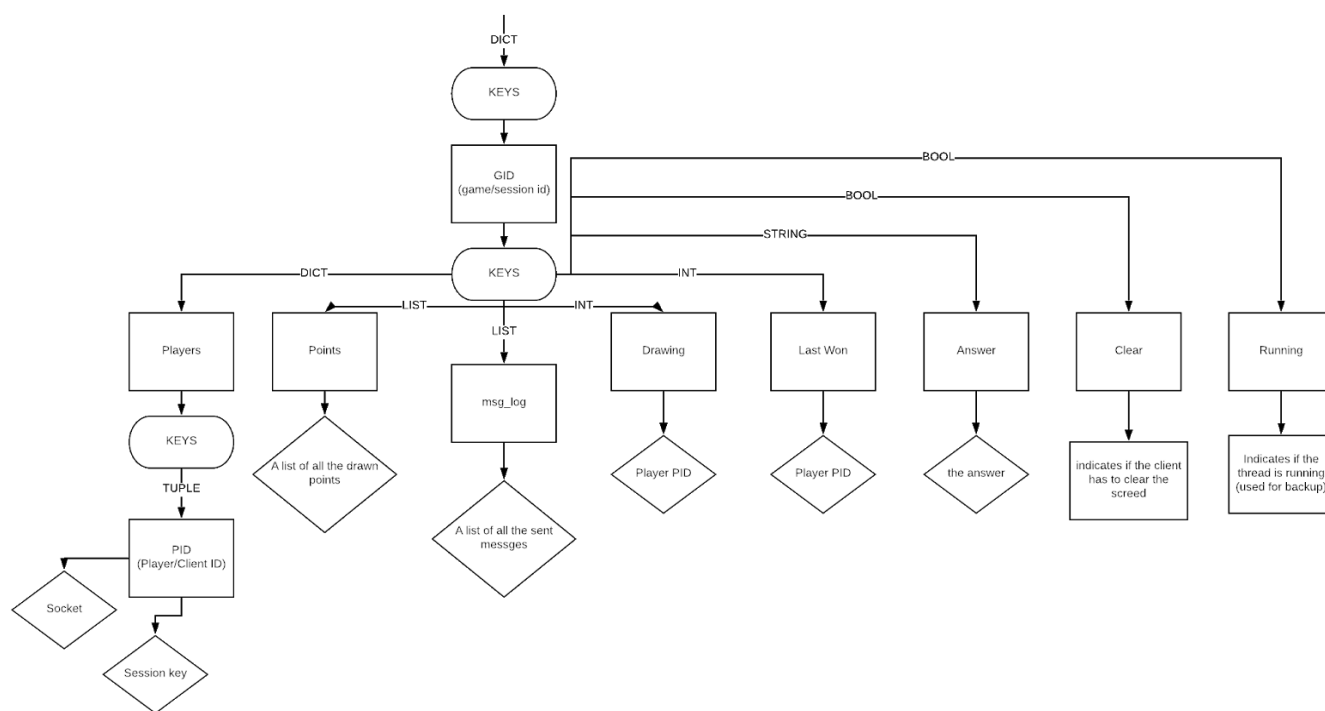
תיאור הארכיטקטורה של המערכת המוצעת

השרת והלקוח יכולים לרוץ על כל מערכת הפעלה כל עוד היא תומכת פייתון. המערכת כוללת לקוח ושרת, לקוח שולח מידע לשרת והשרת מפיץ אותו לשאר הלקוחות בצורה משוקללת ובטוחה.



בסיס הנתונים:

DATABASE



תיאור הטכנולוגיה הרלוונטית

צד לקוח: Python, SocketIO, CPickle, Pygame, Threading

צד שרת: Python, SocketIO, CPickle, Threading

הספריות נבחרו בגלל התיעוד הרחב שלהם וההוכחה המקצועית בהם כספריות טובות,

תיאור מודולים בהם נעשה שימוש

Pygame - היא ספריית גרפיקה מוכרת בפיתוח הספרייה מאפשרת רינדור של צורות ותמונות על המסך בתור "משחק" למשעה כל הGUI נבנה ע"י הספרייה הזאת הנוסף המשחק עצמו מרונדר על ידי הספרייה הזאת.

SocketIO - הספרייה הבסיסית של פייתון לתקשורת ברשת הספרייה מאפשרת יצירת חיבור דרכו אפשר להעביר ביטים ובכך נגמרות האפשרויות שלה, זו היא ספרייה ברמה נמוכה ולכן רוב העבודה נעשית על ידי מודולים אחרים או על ידי המשתמש עצמו

Select - הוא כלי המאפשר לקבל את כל הסוקטים המוכנים לכתיבה או קריאה הוא נדרש אם יש צורך לעבוד עם מספר גדול של סוקטים בלי לחסום סוקטים מקבילים.

Cpickle - פיקל היא ספריה מובנת לפייתון המאפשרת סריאליזציה של אובייקטים בדומה לJson הספריה נבחרה במקום Json מכיון שJson לא היה יציב במהירויות גבוהות.

Pprinter - הספריה מאפשרת הדפסה "יפה" מפורטת של אובייקטים היא נדרשת לכתובת log

Time - ספריה מובנת ומוכרת בפייתון המאפשרת עבודה עם זמן בפרויקט הזה היא משומשת כדי לשים רווחים בין פקודות ולבדוק מתי הגיע הזמן לבנות בק-אפ של ה database

Threading - ספריה זו מאפשרת הרצת מספר קטעי קוד במקביל חשוב לציין היא לא מריצה אותם באותו הזמן אלא מקציבה זמן הרצה מתאים לכל אחד בשונה מProcess שמריצה קוד במקביל. היא נבחרה במקומה משום שהיא מאפשרת עבודה עם מאגר נתונים משותף

Random - ספריה מובנת ומוכרת המאפשרת לבצע בחירות אקראיות בפרויקט הזה היא משומשת למספר דברים ביניהם בחירת משתמש אקראי שמצייר או בחירה אקראית של האובייקט אותו הוא מצייר

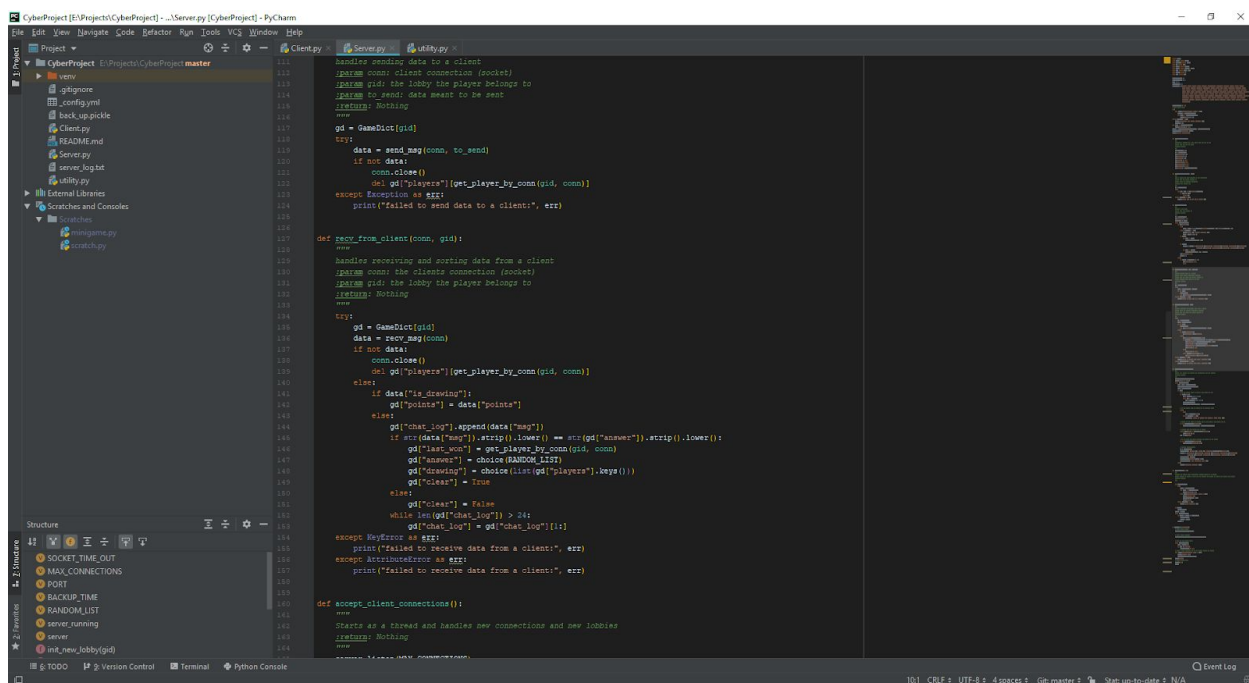
Sys - ספריה המאפשרת תקשורת עם המערכת השימוש היחיד בתוך הפרויט הוא בשביל לסגור את החלון בסוף המשחק או בסוף השרת

תיאור סביבת הפיתוח

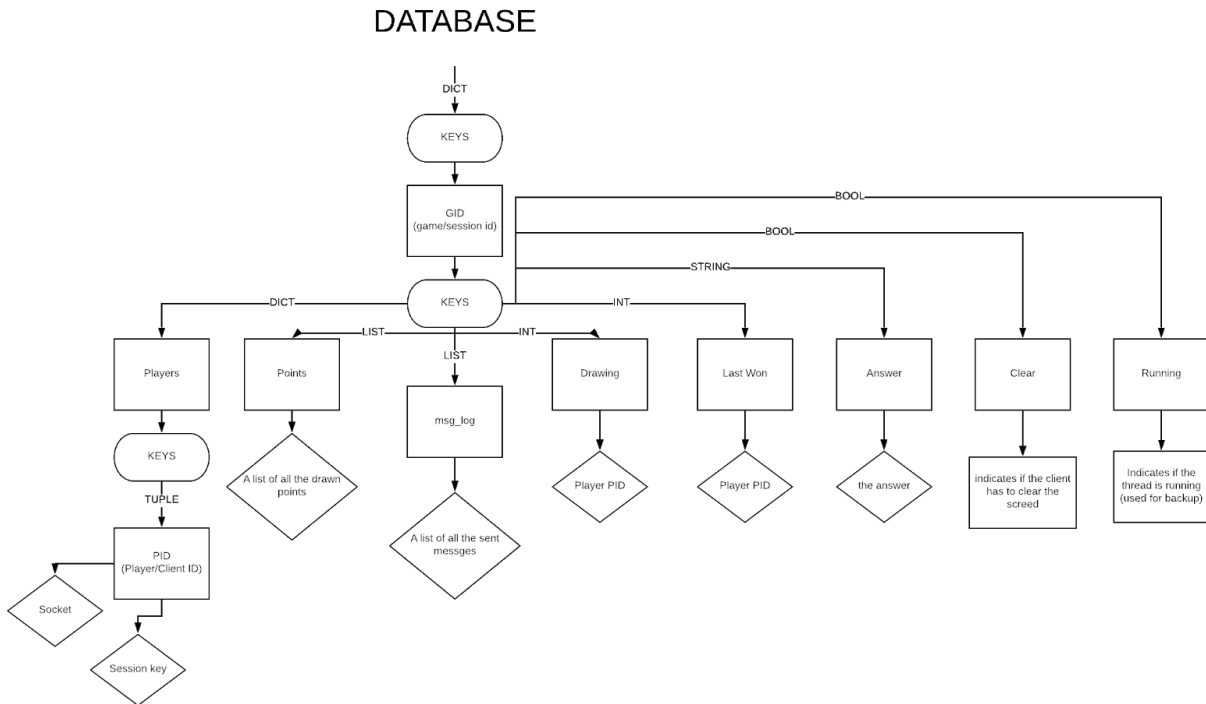
שפת התכנות שנבחרה לכתיבת הפרויקט היא פייתון בלבד מכיוון שהיא שפה אוניברסלית המקובלת בעולם הסייבר.

כלי הפיתוח הנדרשים לפיתוח הפרויקט הם text editor במקרה שלי בחרתי להשתמש IDE PyCharm מכיוון שהוא מאפשר עבודה בפורמט פרויקט ומאפשר פרמוט ועבודה נוחה עם הקוד. בנוסף השתמשתי בgit בשביל VVC וסנכרון בין מחשבים. בנוסף כמובן שצריך python מותקן על המחשב, בנוסף מומלץ להשתמש בpip כדי להתקין את הספריות החיצוניות.

הסביבה והכלים הנדרשים לבדיקות הם בתוכנות עצמן מכיון שזהו פרויקט סגור לא צריך כלים חיצוניים כדי לבדוק אותו אפשר רק להריץ אותו לראות אם יש שגיאות, כמובן שלדיבוג אפשר להשתמש בתוכנות חיצוניות, לדוגמא אני השתמשתי בPyCharm באפשרויות המובנות לתוכו וכך פתרתי בעיות שהופיעו בפרויקט בנוסף קצת Stack Overflow לא הזיק לאף אחד.



תיאור מבני הנתונים



מכיון שרציתי לעבוד בפנימית עם הנתונים לא רציתי לעשות outsource למבנה הנתונים שלי כדי שאוכל לעבוד איתו ישירות ולעשות ניסיונות שונים עם pool, threads ו process כלומר הייתי חייב לעבוד עם משהו פנימי לכן בחרתי לבנות את כל מאגר הנתונים על DICTIONARY שך פייתון למרות כל האי נוחות שזה מביא מעל שימוש ב MySQL בנוסף שימוש ב MySQL היה פותר מעבודה עם סוקטים, בכללי הפרויקט הזה היה חסר משמעות עם פשוט הייתי משתמש בספרייה ל SQL וגומר עניין.

מבנה הנתונים מתחלק ברמה הכי גבוה ל:

● לובים

○ שחקנים

■ סוקט

■ מפת

○ נקודות

■ רשימת נקודות הציור

○ צ'אט לוג

■ רשימת הודעות

○ מצייר

■ שם האדם שמצייר ברגע זה

○ נצח בפעם האחרונה

■ שם האדם שניצח בפעם האחרונה

○ תשובה

■ התשובה הנכונה

○ ניקוי

■ משתנה בוליאני שמודיע לשחקנים על כך שצריך לנקות את המסך

○ רץ

■ משתנה שנועד כדי לבדוק אם הלובי כבר רץ או שרק הרגע שוחזר

מגיבוי

הקוד

שתי קטעי הקוד הבאים הם בעצם פונקציות שליחה וקבלה שבהם משתמש השרת והלקוח הפונקציות עושות בפרינג למידע כלומר לא משנה איזה אומר הודעה אתה שולח אתה לא תתקל בבעיות. זאת לא הדרך הכי אפקטיבית לעשות זאת אבל אני עדיין חושב שאלו קטעי קוד מעניינים. היה מעניין להבין איך לעשות שזה יפעל ולא להשתמש בספריות לשם שינוי.

```
def send_msg(conn, msg):
    """
    sends a buffered message to the designated connection
    :param conn: the designated socket
    :param msg: the data
    :return: True if sent scornfully Else False
    """
    msg = dumps(msg)
    msg = bytes(f'{hex(len(msg))}<{HEADERSIZE}}', "utf8") + msg
    total_sent = 0
    msglen = len(msg)
    while total_sent < msglen:
        try:
            total_sent += conn.send(msg[total_sent:])
        except ConnectionError as err:
            print(err)
            return False
        except EOFError as err:
            print(err)
    return True
```

```
def recv_msg(conn):
    """
    receives buffered data from the designated connection
    :param conn: the connection socket
    :return: the received data
    """
    try:
        msg = conn.recv(64)
    except:
        return False
    msglen = int(msg[:HEADERSIZE], 16)
    full_msg = b'' + msg
    while len(full_msg) < msglen+HEADERSIZE:
        try:
            full_msg += conn.recv(1024)
            print("recv:", len(full_msg), "left:", msglen-len(full_msg))
        except ConnectionError as err:
            print(err)
            return False
        except EOFError as err:
            print(err)
    msg = loads(full_msg[HEADERSIZE:msglen+HEADERSIZE])
    return msg
```


הקוד הבא הוא בעצם פונקציית הmain של השרת הסיבה שהחלטתי להראות את הפונקציה כאן היא בגלל שניתן לראות את אופן החלוקה של threading בשרת. כלומר ניתן לראות שכל חלק בשרת שבעצם מחקה לIO רץ על thread משלו וכל ברגע שאחד מהם יצטרך לפעול זה יקרה מיידית ולא תהיה חסימה מצד פונקציה או פקודה אחרת. לדוגמא ללא threading לא הייתי יכול גם לחקות לפקודות מהממשק של השרת וגם להקשים לחיבורים נכנסים.

```
if __name__ == "__main__":
    # start backup thread
    Thread(target=backup).start()

    # start client handling
    Thread(target=accept_client_connections).start()

    # look out for client input
    while server_running:
        cmd = input(">")
        if cmd == "close":
            server_running = False
    # writes a human readable log if the client closes the server
    with open("server_log.txt", "w") as file:
        print("writing log")
        file.write(pformat(GameDict))
    print("done")
    # backup before exit
    backup(True)
    exit()
```

קטע הקוד הבא מורץ בthread נפרד כל פעם שנפתח לובי חדש, אפשר לראות איך אני מתייחס לכל סוקט בנפרד באיזה סדר אני רץ עליהם, בשביל שסוקטים לא יחסמו אחד את השני אני משתמש בselect חשבתי על להריץ הכל ב pool אחד וככה לקבל את כל המידע במכה אבל החלטתי שאני מעדיף לעבור עליהם ברצף ולא במקביל בשביל לא ליצור התקליויות במאגר הנתונים ולכן הוא לא מורצים בpool במקביל

```
def handle_lobby(gid):
    """
    handles a new lobby
    :param gid: the lobbies ID
    :return: Nothing
    """
    gd = GameDict[gid]
    gd["running"] = True
    start_time = time()
    while server_running:
        if gd["players"]:
            try:
                readl, writel, _ = select(gd["players"].values(), gd["players"].values(), [])
            except Exception as err:
                print("gid:", gid, "select failed:", err)
                readl, writel = [], []
            if readl:
                for conn in readl:
                    recv_from_client(conn, gid)

            if writel:
                to_send = {"drawing": gd["drawing"], "points": gd["points"], "chat_log": gd["chat_log"],
                           "answer": gd["answer"], "last_won": gd["last_won"], "clear": gd["clear"]}
                for conn in writel:
                    send_to_client(conn, gid, to_send)
            gd["clear"] = False
            sleep(0.1)
        else:
            if time() - start_time > 200:
                gd["running"] = False
                break
```

זו היא אחת הפונקציות הארוכות ביותר בקוד המטרה שלה היא לקבל לקוחות חדשים לאתחל הצפנה למיין אותם ללובי שלהם או ליצור לובי חדש בהתאם לבקשה ולשלוח מסר ראשוני לפני שהם נכנסים למערכת לטיפול ע"י `handle_lobby` בנוסף אם הלקוח מנסה להכנס ללובי ישן השרת מזהה את זה ופותר את הלובי עם אותם הגדרות ואותו מצב מהפעם האחרונה שהוא נסגר

```
def accept_client_connections():
    """
    Starts as a thread and handles new connections and new lobbies
    :return: Nothing
    """
    server.listen(MAX_CONNECTIONS)
    print("Server is open")
    while server_running:
        conn, addr = server.accept()
        conn.settimeout(SOCKET_TIME_OUT)
        data = recv_msg(conn)
        if data:
            # if the client asked to build a new lobby for him
            if data["new"]:
                gid = randint(10000, 99999)
                while gid in GameDict:
                    gid = randint(10000, 99999)
                init_new_lobby(gid)
                Thread(target=handle_lobby, args=(gid)).start()

            # if the client asks to connect to an existing lobby
            else:
                try:
                    gid = int(data["gid"])
                except Exception as err:
                    print(addr, "tried to connect but entered a false gid", err)

                # checks if the lobby is running atm if not opens a new thread for it
                if not GameDict[gid]["running"]:
                    Thread(target=handle_lobby, args=(gid)).start()

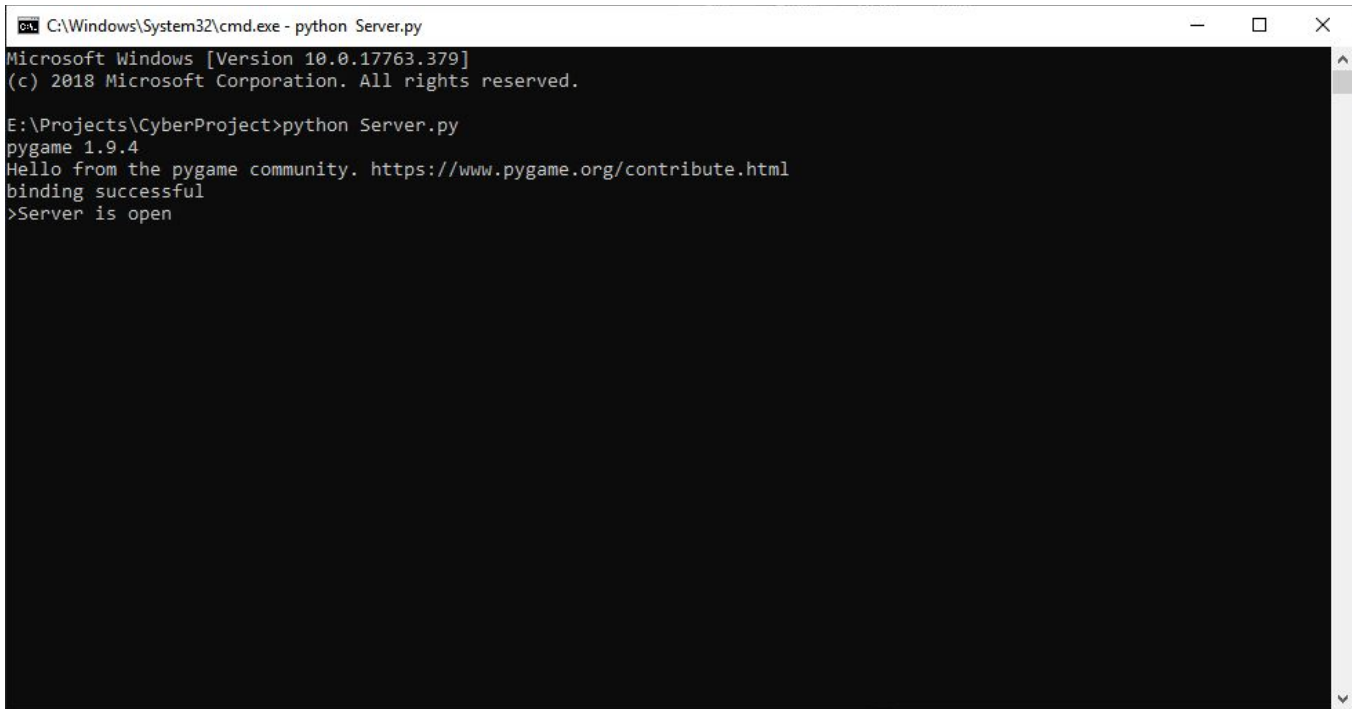
                # ensures that the client pid is unique
                while data["pid"] in GameDict[gid]["players"]:
                    data["pid"] += "1"
                pid = data["pid"]

                # if the lobby was empty entitles the player as the drawer
                if not GameDict[gid]["players"]:
                    GameDict[gid]["drawing"] = pid

                # initial communication
                gd = GameDict[gid]
                send_msg(conn, {"gid": gid, "pid": pid, "points": GameDict[gid]["points"]})
                to_send = {"drawing": gd["drawing"], "points": gd["points"], "chat_log": gd["chat_log"],
                           "answer": gd["answer"], "last_won": gd["last_won"], "clear": gd["clear"]}
                send_msg(conn, to_send)
                GameDict[gid]["players"][pid] = conn
                print(str(addr), "has connected, as:", pid)
            else:
                print("connection failed:", addr)
```

מדריך למשתמש

כדי להריץ את השרת יש להריץ את הסקריפט הוא יפתח חלון שמחכה ל ויתחיל לקבל חיבורים



```
C:\Windows\System32\cmd.exe - python Server.py
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

E:\Projects\CyberProject>python Server.py
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
binding successful
>Server is open
```

לנווט לתיקיה בא נמצא הקובץ בדרך המתאימה למערכת ההפעלה:

`cd C:\project\dict`

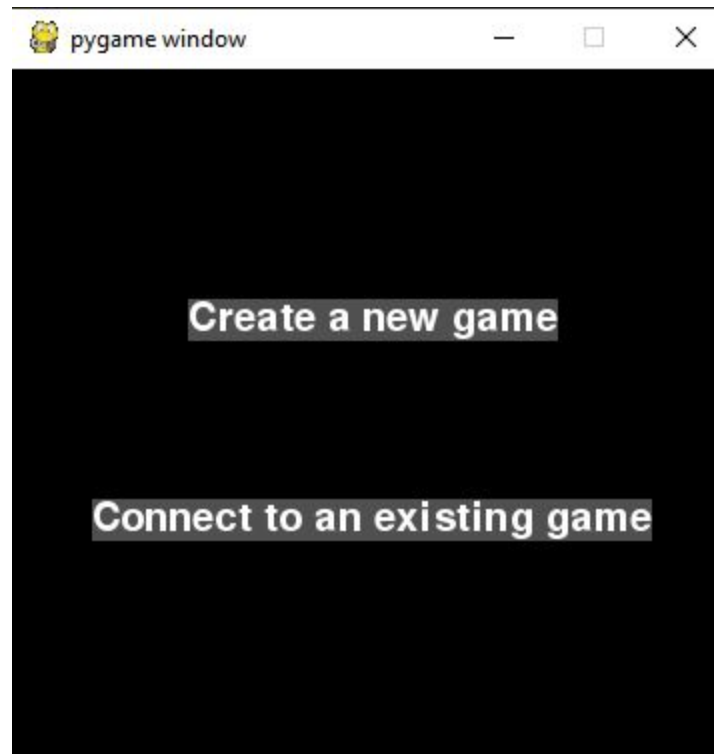
ואז בהתאם למערכת ההפעלה

`python Server.py` או `./Server.py`

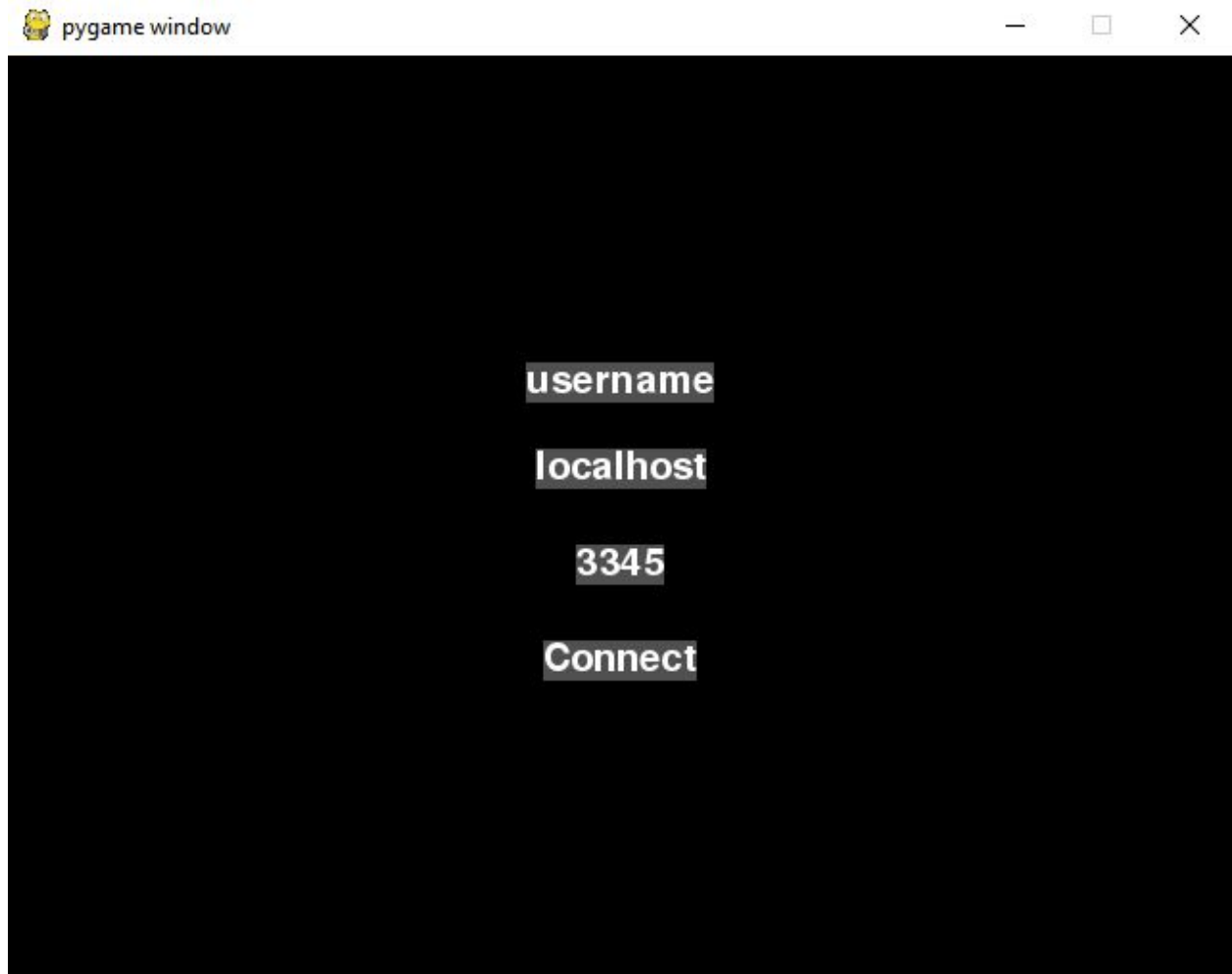
דרך הפעלת הקליינט זהה הדבר היחיד שנדרש לשנות הוא לכתוב `Client.py` במקום `Server.py`

שימוש בממשק לקוח גרפי:

ברגע שפתחת את Client ה Client יפתח מולך חלון טפריט בעל שתי אפשרויות אחת היא להתחבר למשחק קיים והשניה ליצור משחק חדש

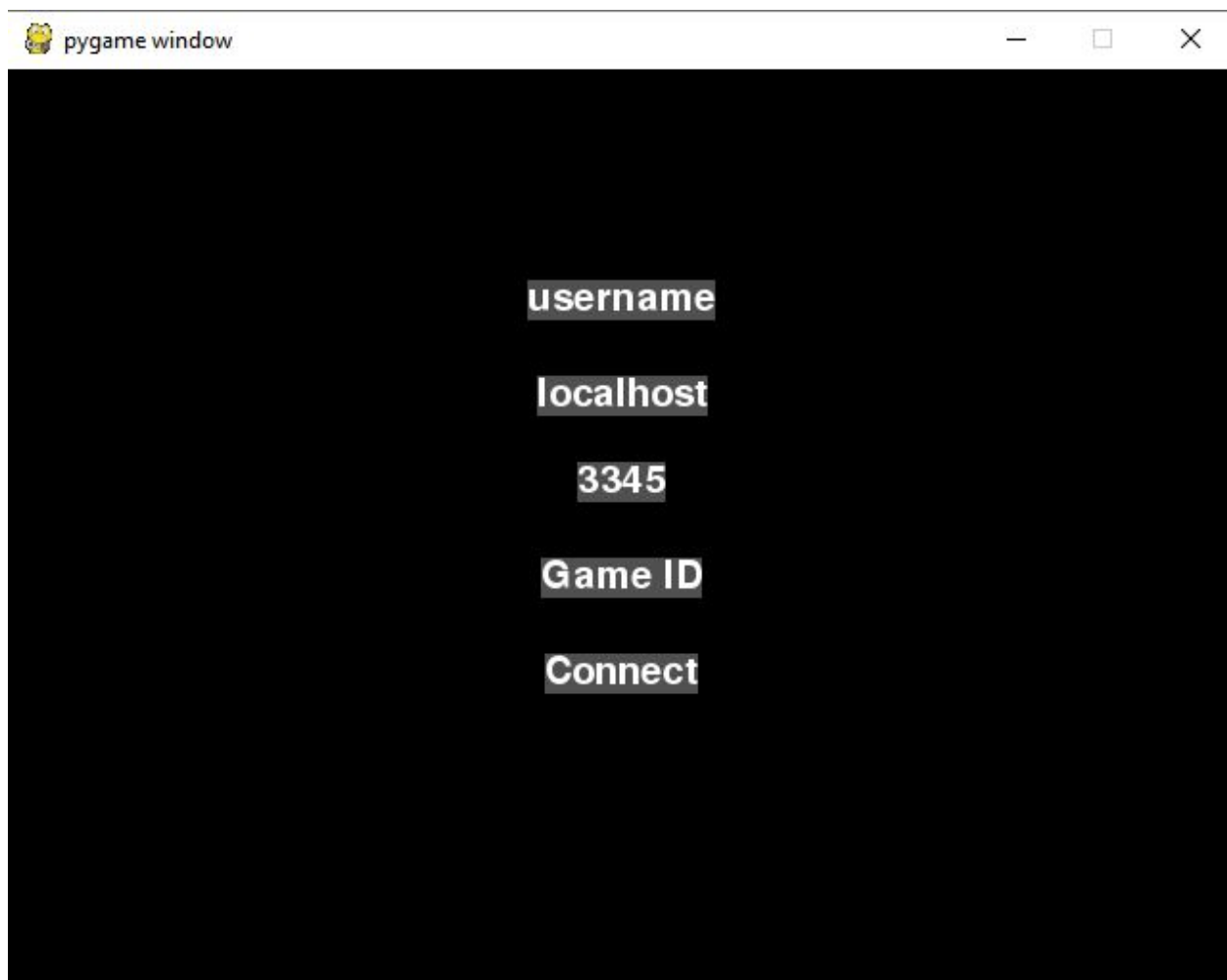


אם בחרתה ליצור משחק חדש יפתח החלון הבא :



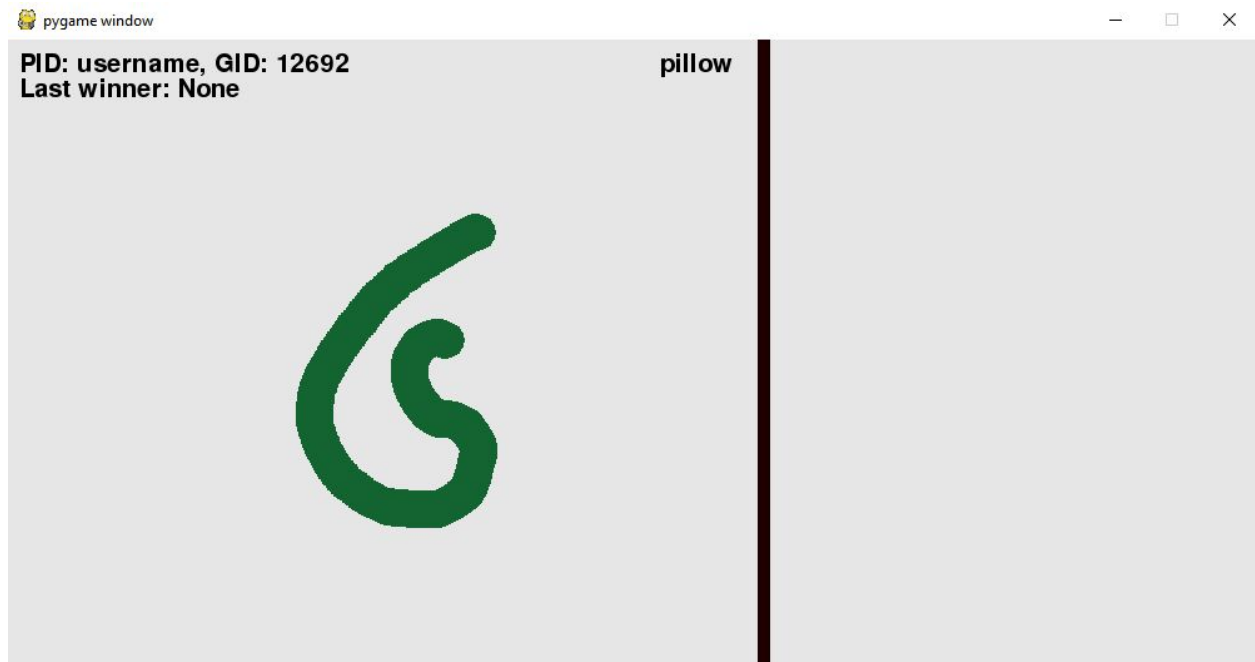
כאשר username הוא שם המשתמש והPID שלך, localhost זה IP של השרת, ו3345 הוא הפורט הנתונים כאן הם דוגמא (והנתונים שאני השתמשתי באת בדיקת המערכת) לבסוף תתבקש לנחוץ Connect ותתבצעה התחברות לשרת

אם בחרתה להתחבר למשחק קיים יפתח החלון הבא :



כאשר username הוא שם המשתמש והPID שלך, localhost זה IP של השרת, ו3345 הוא הפורט בשונה מיצירת משחק חדש תתבקש להכניס GID שינתן ע"י חבר שכבר פתח שרת, הנתונים כאן הם דוגמא (והנתונים שאני השתמשתי באת בדיקת המערכת) לבסוף תתבקש לנחוץ Connect ותבצעה התחברות לשרת

בתור הצייר המילה תהיה חשופה לעיניך ואתה תוכל לצייר



לעומת זאת בתור אחד השחקנים האחרים בלובי המילה תהיה מוצפנת ואתה תוכל לשלוח צ'אט לנסות לנחש מה מצוייר



מבט אישי על העבודה ועל תהליך פיתוחה

מטרתי בעבודה הזאת הייתה ללמוד כמה שיותר נושאים חדשים בזמן המוגבל שהיה לי (לצערי אני הייתי בתקופה של הרבה תחרויות ולכן הזמן שהייתי יכול להשקיע בפרויקט היה מוגבל)

בכל זאת למדתי הרבה דברים גם אם הם לא נמצאים בפרויקט הסופי כמו שימוש ב JSON איך לעשות PADDING, הרחבתי את הידע בthreading לדוגמא בכלל לא ידעתי שקיים דבר כמו process שלא מוגבל על ידי global interpreter lock למדתי להשתמש בselect בצורה נכונה גיליתי את הספריה Cpickle ועוד הרבה דברים אחרים. גם אם אני לא חושב שאני סיימתי את הפרויקט במאת האחוזים ואני חושב שיש עוד מה לשפר (וחד משמעית יש) אני מרוצה ממה שלמדתי גם אם לא בהכרח יצא לי ליישם את זה. אני למדתי הרבה על הצפנות ועל הפעולה שלהם מבחינה מתמטית.

אני חושב שאני עשיתי את רוב המטרות שהצבתי לעצמי למרות שיש מה לשפר.

במהלך הפיתוח היו בעיות רבות חלקם אני עדיין לא מבין לדוגמא JSON לא עבד טוב ונאלצתי לחפש אלטרנטיבה וככה גיליתי על קיומו של Pickle. בעצם כל הקשיים שנתקלתי בהם עברתי אותם בדרך כזאת או אחרת או שהחלפתי כלי שאני משתמש בו או קראתי עוד דוקומנטציה/עברתי על עוד דוגמאות/פרויקטים עד שהבנתי את דרך השימוש

בעתיד אני חושב על להעביר את המשחק לרשת כנראה בJS כי אני לא יודע canvas והיה מעניין ללמוד דבר כל כך בסיסי בweb development הפרויקט הזה לא הכי מלוטש אבל הוא בסיס טוב ללימודים וזה הדבר העיקרי שעשיתי, למדתי. גם אם הצפנה למשחק מטופש עם GUI לא מושקע זה רעיון מפגר וגם אם buffering מאט פי כמה וכמה את מהירות התגובה המטרה של הפרויקט הייתה ללמוד על נושאים חדשים ולא ליצור משחק פרקטי לכן אני חושב שהמטרה הושגה במלואה.

ביבליוגרפיה

https://en.wikipedia.org/wiki/Main_Page

<https://realpython.com/async-io-python/>

<https://pycryptodome.readthedocs.io/en/latest/index.html#>

<https://stackoverflow.com/>

<https://www.pygame.org/docs/>