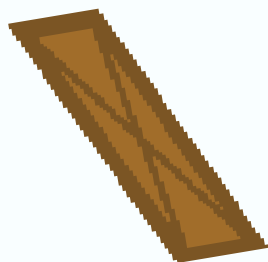


לומגאן

משחק יריות מרובה משתמשים מוצפן



שם המגיש: ליאם ניב

ת.ז: 330780552

מנחה: יהודה אור

שם החלופה: סייבר / מערכות הפעלה

תאריך הגשה: 13.06.2024



הכפר הירוק
HAKFAR HAYAROK

הכפר הירוק ע"ש לוי אשכול

תוכן עניינים

2	תוכן עניינים
4	מבוא
4	ייזום
6	פירוט תיאור המערכת (אפיון)
10	תיאור תחום הידע
10	פירוט וניתוח של יכולות מפרק קודם
14	מבנה / ארכיטקטורה
14	תיאור ארכיטקטורה של מערכת מוצעת
16	תיאור הטכנולוגיה הרלוונטית
17	תיאור זרימת המידע במערכת
19	אלגוריתמים מרכזיים
20	תיאור סביבת הפיתוח
21	פרוטוקול התקשורת
23	תיאור מסכי המערכת
27	תיאור מבני הנתונים
30	סקירת חולשות ואיומים
32	מימוש הפרויקט
32	סקירת ספריות ומחלקות מיובאות
34	סקירת מחלקות שלי
41	סקירת מודולים שלי
42	קטע קוד יפה / פירוט אלגוריתמים
51	מסמך בדיקות
54	מדריך למשתמש
54	עץ קבצים
55	התקנת המערכת
59	אופן הפעלת המערכת
62	רפלקציה
64	ביבליוגרפיה
65	הקוד
65	AES_protocol.py
66	AESHelper.py
68	client.py
85	database_manager.py
88	database.json (דוגמה)
89	my_sha256.py
90	network.py
92	protocol.py

99.....	RSAHelper_client.py
100.....	RSAHelper.py
102.....	server.py

מבוא

ייזום

תיאור ראשוני של המערכת:

פרויקט זה מהווה משחק מחשב המשלב בתוכו ריבוי שחקנים דרך הרשת. בעת הכניסה למשחק, השחקנים ידרשו להזדהות או להירשם למאגר המידע. המשחק הוא משחק יריות דו-מימדי מגוף שלישי, כלומר שנקודת המבט היא מחוץ לדמות השחקן, כאשר המשתמש עוקב אחר השחקן מלמעלה. אל המשחק עד יתחברו שני שחקנים המעוניינים להילחם אחד בשני בקרב יריות בו יזוזו עם מקשי המקלדת ויכוונו עם העכבר שלהם. כאשר שחקן פוגע בשני, כמות החיים של השחקן הנפגע תרד עד שהוא ימות ויחזור חזרה למשחק. כאשר השחקן האחר מת, השחקן ההורג מקבל נקודה והשחקנים חוזרים להילחם שוב.

בחרתי ליצור משחק מחשב כי במשך כל החיים שלי שיחקתי במשחקי מחשב מרובי משתמשים ורציתי ללמוד על היצירה שלהם. אני צופה קושי בהצגה ויצירת גרפיקה מתאימה למסך, פתיחת חלון והצגת גרפיקה היא דבר שלא ניסיתי לבצע לפני העבודה שלי על הפרויקט. בנוסף, גם הצפנת מידע היא תחום שלא התנסיתי בו ולכן אני מצפה שהיא תהווה לי אתגר.

הגדרת הלקוח:

המשחק יכול להתאים לכל הגילאים, חובבי משחקי מחשב. מיועד לכל מי שרוצה לשחק ולהעביר את הזמן עם חברים או אדם אחר שהתחבר לשרת.

הגדרת יעדים:

בפרויקט זה שאפתי בראשונה ליצור משחק מחשב מתפעל הכולל ממשק משתמש ומעוצב לפי אמנות פיקסלים, בצורה שגם שחקן שלא יודע לתכנת יכול להשתמש בו. זאת ועוד, שאפתי להכין את המשחק שלי עם ריבוי משתמשים כדי ששני שחקנים חברים יכולים לחלוק את חווית המשחק שלהם יחד, בתנאי שהם נרשמו למאגר המידע. בנוסף, רציתי לדאוג כמה שיותר שהמידע שיכול להיות רגיש העובר בין המשתמש לשרת יהיה מוצפן ושומר בצורה מוצפנת ובטוחה במקרה של ניסיון התקפה זדוני.

בעיות תועלות וחסכוניות:

בתחילת יצירת הפרויקט, צפיתי קושי ובעיות רבות בניהול המשחק וההצגה הגרפית שלו לכל משתמש. זאת ועוד, יצירת שרת מרובה משתמשים היא גם מכשול מסובך שצפיתי להתמודד איתו. יתר על כן, אתגר נוסף שצפיתי בבניית הפרויקט היא הצפנה ואבטחת המידע שעובד בתקשורת. כל אלו יעילו לחווית המשחק, יהפכו את חווית המשחק למוחשית, חברתית, ובטוחה יותר.

מבחינת המשתמש, הוא יכול לשחק במשחק שלי להנות. הוא יכול להעביר את הזמן שלו במשחק עם אדם אחר.

סקירת פתרונות קיימים:

ישנם המון משחקי יריות, חלקים מרובי משתמשים וחלקם לא, אבל יש הרבה סוגים של משחקי יריות. לעומת זאת, בשנים האחרונות בשל התפתחות החומרה משחקי מחשב רבים לא משתמשים באומנות פיקסלים, שאני חושב שמאוד יפה. לכן, למרות שפורמט היריות מאוד פופולרי, משחק המשלב זאת עם אמנות פיקסלים ומבט על דו-מימדי הוא כבר חווית משחק ייחודית יותר המייחדת את המשחק שלי. (חוץ מזה שהמטרה שלי היא לא באמת להמציא את הפורמט מחדש)

סקירת טכנולוגיות הפרויקט:

בפרויקט שלי בחרתי להשתמש ב-python, בחרתי בשפה הזו כדי לפתח משחקי מחשב ללא קשר לפרויקט הזה בעזרת ספריית פייגיימ (pygame). רציתי ללמוד פייתון בשל האופי החופשי והפשוט שלה ושמעתי שבעזרת pygame אוכל לפתח משחקי מחשב, וכך אוכל להנות יותר מתהליך הלמידה שלי. מאוחר יותר גיליתי גם על היכולות הרחבות של פייתון בעזרת ספריות שונות. ספריות המאפשרות לבנות מערכת שרת ומשתמש או הצפנת מידע דרך hash או AES.

תיחום הפרויקט:

הפרויקט שלי עוסק בנושאים הבאים:

- פייגיימ ויצירת משחקים, כולל שימוש בקלט מקשי מקלדת ועכבר
- תקשורת בין שרת ומספר של לקוחות בעזרת תהליכונים
- פרוטוקול לקליטת סוגים שונים של מידע (בשכבת האפליקציה)
- שימוש בקובץ json לצורך שמירת נתונים
- הצפנה אסימטרית RSA לתיאום הצפנה סימטרית AES
- אבטחה ותיקון חולשות במערכת

לא כללתי בפרויקט שלי:

- שימוש במיקרופון או מצלמה
- בינה מלאכותית למשחק לבד

פירוט תיאור המערכת (אפיון)

תיאור מפורט של המערכת:

כדי להתחיל את המערכת יש להריץ את הקוד של השרת. בעת הרצת קוד השרת, נפתח חלון הדורש את סיסמת המנהל על מנת להתחיל את השרת, זאת כדי למנוע כוונה זדונית של אדם לא מאושר לפתוח שרת משלו וכך להשיג מידע מסווג על שחקנים. על המנהל ללחוץ על תיבת הטקסט, להקליד את הסיסמה שלו, אם הסיסמה נכונה החלון יסגר והשרת יתחיל לפעול ויצג הודעה שמראה שהוא מחכה לכניסה של שחקנים.

עכשיו, על מנת להכנס בתור שחקן, יש להריץ את הקוד של קובץ הלקוח. בעת ההרצה מתבצע תהליך בין השרת ללקוח כדי לסכם בצורה מוצפנת על איזה מפתח AES הם ישתמשו בהמשך התקשורת, המפתח AES מועבר לשרת בעזרת הצפנת RSA. לאחר תיאום זה, נפתח חלון ללקוח הדורש ממנו להירשם למאגר הנתונים. המשתמש יכול להתחבר למשתמש קיים או ליצור משתמש חדש, לאחר תהליך זה הוא יכנס למשחק. במשחק תהיה תקשורת רציפה בין הלקוח לשרת, הלקוח יכול לראות את הדמות שלו ממבט על. הוא יכול להזיז את הדמות שלו לכיוונים שונים בעזרת המקשים w, a, s, d ולהסתכל לכיוונים שונים, השחקן תמיד מסתכל לכיוון של העכבר. כאשר שחקן נוסף מתחבר לשרת, שני השחקנים יכולים ללחוץ מקשי העכבר כדי לירות אחד על השני. במקרה של פגיעה של קליעים בשחקן, מד החיים של השחקן שנמצא מעליו ירד, עד לנקודה שהשחקן ימות. במידה ושחקן מת הוא יחזור עם חיים מלאים לנקודת ההתחלה שלו ומספר ההריגות של השחקן ההורג יעודכן בהתאם. כל המפה, המכשולים והדמויות אני ציירתי בעצמי ב-firealpaca.

פירוט יכולות:

לקוח:

- התחברות לשרת
- ליצור מפתח AES
- להצפין משפתח AES עם הצפנת RSA
- שליחת פרטי משתמש
- שליחת מיקום ומצב משחק
- הצגת משחק

שרת:

- להתחבר לשני לקוחות שונים
- יצירת מפתח RSA ופענוח בעזרתו
- שימוש במפתח AES נתון
- לתקשר ישירות עם מאגר הנתונים ולנהל
- לשמור ולהעביר מידע לגבי המיקום של השחקנים במרחב

פירוט הבדיקות:

- הרצת חלון השרת ובדיקת תקינות.
- הכנסה של סיסמאות שגויות ותווים לא תקינים לחלון השרת.
- הסתכלות על פירוט תהליך ה-RSA שמפורט בהדפסה של שני הצדדים בעת חיבור של שחקן.
- הרצת חלון של לקוח ובדיקת תקינות.
- יצירת חשבון עם מספר תווים קטן מידי או שם שכבר קיים, הכנסת תווים לא תקינים.
- ניסיון יצירת חשבון עם שם תפוס, ניסיון התחברות עם פרטים שגויים.
- התחברות לשרת עם משתמש תקין.
- בדיקה שגם אם השחקן לא זז, השרת מקבל הודעות מוצפנות שונות הודות לשינוי של ה-iv.
- חיבור שני שחקנים, תזוזה במרחב של שני השחקנים.
- הריגה של שחקן אחד כדי לוודא את תקינות מערכת ההריגה.
- התנתקות של שחקן אחד והתחברות עם משתמש אחר.
- התנתקות סופית של שני השחקנים המחוברים והסתכלות על הודעות הניתוק של השרת.

תכנון וניהול לו"ז:

את הפרויקט שלי אני התחלתי בטעות, בפרוץ המלחמה באוקטובר 2023 כאשר בית הספר הפך למקוון החלטתי שאני רוצה להתחיל לפתח משחק מחשב ללא קשר לפרויקט גמר. לאחר פיתוח של שלד של

המשחק בפרויקט זה, בו השחקן פשוט משוטט במפה ויורה קליעים ללא מטרה, החלטתי שאני אקח הפסקה ארוכה ללמוד את שאר השלבים של הפרויקט. בינואר, זמן רב לאחר כתיבת השלד החלטתי שארחיב אותו לפרוייקט מלא. מאותו רגע החלטתי לתכנן לוו"ז (רק של פעולות לעשות) של כל הדברים שאני רוצה לעשות בפרוייקט. אחר הלו"ז הזה עקבתי בדיוק, כאשר שלב אחד נבנה על השני בהתאם לדרישות משרד החינוך. לכן, הלו"ז המתוכנן והמבוצע שלי, פרט להבדלי זמן קטנים, הם בדיוק אותו הדבר ואפרט לגביהם.

לוח זמנים לפרוייקט, לאחר יצירה של משחק מקורי:

- מעבר על כל הקוד שרשמתי והבנתו שוב.
- הוספת משתמש נוסף למפה שיכול להסתובב במפה כמו הראשון.
- הוספת מדד חיים המרחף מעל ראש השחקן וזיהוי פגיעה מקליע.
- נתינת משמעות למוות של שחקן ושיגורו לנקודת ההתחלה שלו עם חיים מלאים.
- פתיחת קובץ json לשימוש כמאגר נתונים ויצירת פונקציות המתנהלות איתו.
- יצירת מסך בית לשחקן עם אפשרות להירשם או להתחבר למאגר נתונים.
- הוספה של שם משתמש מרחף מעל ראש של שחקן.
- יצירת אובייקט ללקוח להצפנה בלבד של RSA על פי מפתח נתון.
- יצירת אובייקט לשרת המייצר מפתחות RSA ויכול להצפין ולפענח.
- יצירת אובייקט לניהול הצפנה ופענוח בעזרת מפתח AES לשרת וללקוח.
- כתיבת תהליך 'לחיצת יד' בין שרת ולקוח בעזרת RSA והמשך תקשורת בעזרת מפתח AES.
- פרטי אבטחה אחרונים כמו טיפול בקלט עם פרוטוקול שגוי, מסך אימות של שרת והצפנת סיסמאות עם sha256.

כפי שציינתי, ביחס ליצירת שלד המשחק המקורי, פעולות אלו לא לקחו הרבה זמן ולכן הצלחתי לבצע את אשר תכננתי לפי סדר לוגי זה.

ניהול סיכונים:

חשש מפריצה למאגר הנתונים:

אחרי שסיימתי לעבוד על הפרויקט ללא ההצפנה, חשבתי על הדוגמה הרעה שאני מהווה לעצמי. הרי אני ועוד הרבה אנשים סובלים בלי אפילו לדעת לפריצה של מאגר נתונים של אתר והפצת הפרטים בתמורה לתשלום. תחילה חשבתי שהעברת הסיסמאות באופן מוצפן בעזרת הצפנת AES מספיקה, אך לא חשבתי על האפשרות שתהיה פריצה למאגר הנתונים עצמו. לכן, אפילו על מאגר הנתונים עצמו לא לכלול את סיסמאות המשתמשים, אלא גם לשמור אותן מוצפנות. לכן ניסיתי לחשוב על דרך משלי

להצפין אותן, אך לבסוף למדתי על אופיין של הצפנות חד כיווניות ופונקציית Hash. ראיתי בהן דרך מצויינת לשמור על הסיסמאות של המשתמשים במשחק והחלטתי להצפין את כל הסיסמאות באתר בעזרת sha256.

פתיחה לא מורשית של שרת:

עוד חולשה שחששתי ממנה בתוכנה שלי היא הפתיחה של שני שרתים כאשר אחד מהם נפתח על ידי בן אדם זדוני שיכול להשיג סיסמאות פרטיות של אחרים. כדי למנוע זאת, תחילה חשבתי על למצוא אתר או שרת חיצוני המאמת את זהותו של השרת, בדומה לפרוטוקול https אבל לבסוף מצאתי דרך פשוטה יותר. הצלחתי להשתמש בעקרון של הצפנה חד כיוונית ולשים את פתיחת השרת מאחורי סיסמה מוצפנת שרק אני יודע אותה. כך רק אני יכול לפתוח את השרת למשחק שלי.

ניהול זמנים:

כדי לבנות פרוייקט, נדרש מאמץ רב ותהליך למידה ארוך. בשל היותי תלמיד כיתה יא' תקופת הכתיבה מוצפת במועדי בגרויות ולחץ רב. אך בעזרת הלו"ז שתכננתי ופירטתי מוקדם יותר, הצלחתי לפרק את הפרויקט לתת-משימות קטנות יותר ולהקדיש תשומת לב לכל אחת מהן. זה עזר לי להישאר ממוקד ועם הרגליים על הקרקע בכל עת, ההתמדה והלו"ז המסודר עזרו לי לסיים את הפרויקט בזמן.

תיאור תחום הידע

פירוט וניתוח של יכולות מפרק קודם

יכולות שרת:

- שם יכולת: חיבור עם שני לקוחות שונים
- מהות: הפונקציה נפתחת כחלק מתהליכון (thread), כלומר היא יכולה להתבצע באותו הזמן עבור שני לקוחות. מקבלת חיבור מלקוח, בודקת אם ואיזה מקום מבין שני המקומות הפנויים למשחק פתוחים ואם כן פותחת את הפונקציה `threaded_client` שמריצה את פרוטוקול לחיצת היד ומתחילה את מעבר הפרטים בין השניים.

- אוסף יכולות נדרשות:

- איתור קשר ראשוני
- פתיחת `socket`

- אובייקטים נחוצים: `socket`.

- שם יכולת: יצירה ופענוח באמצעות מפתח RSA
- מהות: על מנת לבצע תקשורת מוצפנת עם הלקוח, על השרת לסכם עימו על מפתח סימטרי. הם מסכמים על מפתח סימטרי זה באמצעות מערכת מפתחות אסימטרית, שעל השרת ליצור מפתח עבורה והיכולת לפענח מפתח סימטרי הנשלח אליו מהלקוח.

- אוסף יכולות נדרשות:

- יצירת כל המשתנים המספריים לצורך המפתחות
- פענוח באמצעות משוואות מתמטיות

- אובייקטים נחוצים: `RSAHelper`.

- שם יכולת: שימוש במפתח AES
- מהות: לאחר הסכמה על מפתח AES לתקשורת מוצפנת, על השרת להשתמש בו בצורה נכונה כדי לאפשר תקשורת בטוחה.
- אוסף יכולות נדרשות:
 - שמירת מפתחות AES
 - הצפנת AES
 - פענוח AES
- אובייקטים נחוצים: AESHelper.

- שם יכולת: עבודה עם מאגר נתונים
- מהות: לשם הזדהות של לקוח, על השרת לתקשר עם קובץ מאגר נתונים השומר על המשתמשים שהוא יכול להכניס למשחק.
- אוסף יכולות נדרשות:
 - הוספת משתמש
 - זיהוי משתמש וסיסמה קיימים
 - חיפוש אחר שם משתמש (כדי לבדוק ייחודיות)

- שם יכולת: שמירה והעברת מידע
- מהות: קבלת מידע ממשתמש והפצתו לאחר, כדי לאפשר חווית משחק משותפת ורציפה.
- אוסף יכולות נדרשות:
 - הצפנה ופענוח AES
 - קריאה וכתיבה על פי פרוטוקול
 - שמירת נתונים
 - העברת נתונים
- אובייקטים נחוצים: AESHelper, socket, conn, ספריית פרוטוקול (protocol.py)

יכולות לקוח:

- שם יכולת: התחברות לשרת
- מהות: התחברות ראשונית לשרת וביסוס חיבור עם שרת למשך המשך תקשורת
- אוסף יכולות נדרשות:
 - ביסוס חיבור ראשוני עם שרת
 - שליחת פרטים אישיים
- אובייקטים נחוצים: socket
- שם יכולת: יצירת מפתח AES
- מהות: כדי לאפשר שיחה מוצפנת, על כל לקוח ליצור מפתח סימטרי ייחודי לשיחה שישמש אותו בתקשורת עם השרת לאורך כל התקשורת הרציפה שלהם.
- אוסף יכולות נדרשות:
 - יצירת מפתח AES
 - יצירת iv ייחודי לכל הודעה
- אובייקטים נחוצים: AESHelper
- שם יכולת: הצפנת RSA
- כדי לשלוח את מפתח ה-AES בבטחה לשרת, על הלקוח להצפין אותו באמצעות מפתח RSA שהוא קיבל מהשרת.
- אוסף יכולות נדרשות:
 - קליטת מפתח
 - יצירת אובייקט הצפנה
 - שימוש בפעולת הצפנה
- אובייקטים נחוצים: RSAHelper_client

- שם יכולת: שליחת פרטי משתמש
- מהות: שליחה של פרטי משתמש על מנת להתחבר למערכת כמשתמש מוכר או חדש.
- אוסף יכולות נדרשות:
 - איסוף קלט מהמשתמש
 - סינון קלט בלתי תקין
 - שליחה לשרת הודעה על פי פרוטוקול
- אובייקטים נחוצים: Network
- שם יכולת: שליחת מיקום במשחק
- מהות: שליחה של מידע לשרת על מנת לספק פרטים לשחקנים האחרים, לקבל מידע וליישם אותו במשחק בזמן אמת כדי לקבל אות משחקן אחר.
- אוסף יכולות נדרשות:
 - הצפנה סימטרית עם מפתח AES
 - קריאה ויצירה של אותות על פי פרוטוקול
 - שליחת מידע וקבלת מידע לשרת
- אובייקטים נחוצים: socket, AESHelper, ספריית פרוטוקול.
- שם יכולת: הצגת משחק
- מהות: כדי שהמשתמש יוכל לראות את מה שמתרחש מאחורי הקלעים, על הקוד להריץ לו תמונת מצב של מה שקורה בשרת.
- אוסף יכולות נדרשות:
 - שמירת מיקום של דברים
 - הצגה של כל הדברים מסביב למסך
 - עדכון מסך 60 פעמים בשנייה
- אובייקטים נחוצים: player, bullet, button, slash, Rect, display, clock.

מבנה / ארכיטקטורה

תיאור ארכיטקטורה של מערכת מוצעת

תיאור חומרה מרכזית:

מעבד (CPU):

מרכיב מרכזי במחשב המבצע את מירב הפעולות במחשב, אף נרקא לרוב ה"מוח" של המחשב. הוא פועל על פי ההוראות של תוכנות במחשב כדי לקרוא מידע מהזיכרון וביצוע פעולות חשבוניות בין היתר. מעבדים מודרניים מחולקים לחלקים המכונים cores. חילוק המעבד אליהן מאפשר עבודה לצד מספר רב של ישומים וריבוי הפעולות שהוא מבצע.

במקרה של התוכנה שלי, למרות שאני מריץ משחק המשלב בתוכו אלמנטים גרפיים, מירב הפעולות קורות דווקא במעבד. זאת משום שבדרך כלל לתצוגה גרפים משתמשים ברכיב מחשב אחר הנקרא GPU, המתאפיין בחלוקה לcores קטנים יותר משל המעבד, אלו מקנים לו את היכולת לבצע מספר רב יותר של משימות (אך פשוטות יותר) והופכות אותו ליעיל יותר בהצגה גרפית. למרות זאת, בpygame אין שימוש רחב בGPU, משום שהגרפיקה בpygame היא מינימליסטית ולא מכבידה על המעבד, בניגוד למשחקי מחשב מודרניים.

כרטיס רשת (NIC):

רחיב מחשב המאפשר תקשורת של המחשב עם המרשתת. הוא יכול להיות משומש כדי לתקשר דרך מספר דרכי שכבה פיזית, כמו Wi-Fi או כבל ethernet. כרטיס הרשת הופך את רצף הבייטים שנועד לשליחה לרשת לאחד משני פורמטים: במידה ומדובר בחיבור פיזי הוא הופך אותו לפעימות חשמליות, במידה ומדובר ב-Wi-Fi מתקשר באמצעות גלי רדיו. לכל כרטיס רשת יש כתובת שקוראים לה MAC, המשמשת לזיהוי שלו, ולכן היא לא ניתנת לשינוי. זאת ועוד, כרטיס הרשת מעביר את המסרים הנכונים לתהליכים הנכונים דרך כתובת port.

זיכרון גישה אקראית (RAM):

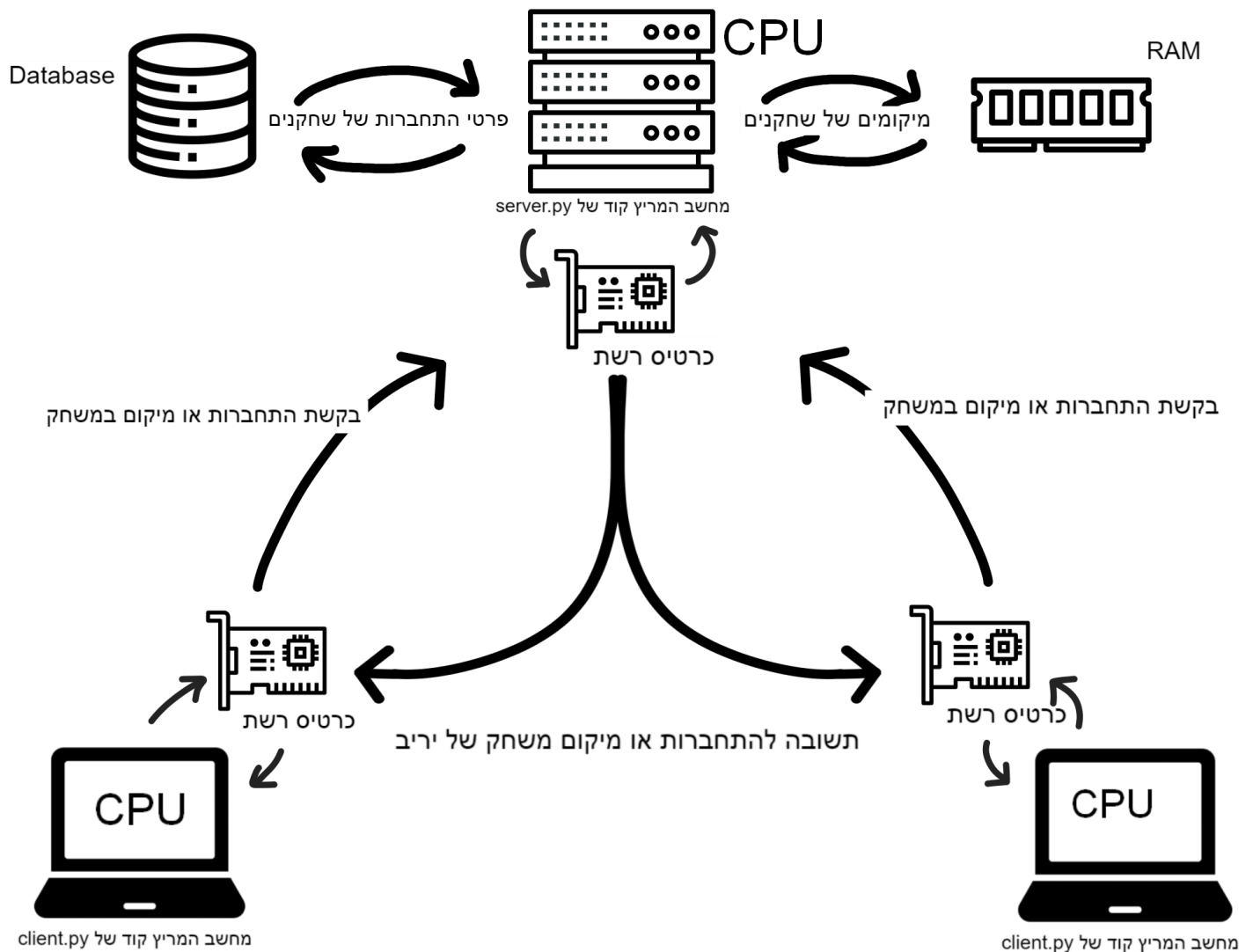
דרך לשמור פרטי מידע בצורה נדיפה, כלומר מחיקה. כאשר כרטיס ה-RAM מתנתק מכלל המערכת, או שהמחשב נכבה הוא לא שומר על נתוניו. הוא משמש לזיכרון פרטים לטווח הקצר ויכול (וחייב) להתעדכן הרבה פעמים בשנייה. לכן, הוא משמש דרך מהירה ונוחה להגיע לפרטי זיכרון זמניים של אפליקציה.

כתוצאה מכך, במשחק שלי פרטי המיקום של השחקנים המשתנים בכל שנייה ורלוונטים אך ורק בזמן הריצה של קוד השרת, הם מאוחסנים במשתנה פשוט בקוד, המשתנה ב-RAM.

זיכרון בלתי נדיף:

שומר מידע של המחשב בצורה לא נדיפה. כלומר, הוא נשמר גם כאשר המחשב נכבה ונדלק. זיכרון בלתי נדיף יכול לבוא בצורות רבות כמו: דיסק קשיח, זיכרון בזק, disk on key ועוד. אופן השמירה הלא נדיף בכרטיס זה מאפשר לשמור נתונים לזמן רב הרלוונטים לקוד מסויים גם בהרצות שונות, כמו פרטי התחברות של משתמשים שונים. משומש במשחק שלי (ובתרשים) כמקום של ה-Database.

תרשים:



תיאור הטכנולוגיה הרלוונטית

את הפרויקט שלי כתבתי לגמרי בפיתון. פיתון היא שפת תכנות עילית, כלומר מיועדת לכתיבה על ידי בני אדם ומאופיינת בתחומים הרבים שהיא מכסה ומידת הקריאות הגבוהה שלה על ידי בני אדם. אחד מהדברים המייחדים את השפה היא השימוש בהזחה כדי להגדרת קטעים שונים של קוד, בניגוד לשפות אחרות הנוהגות להשתמש בסוג של סוגריים.

השפה נוצרה בתחילת שנות התשעים על ידי חידו ואן רוסום. כיום יש לה שתי גרסאות ראשיות: 2 ו-3 כאשר גרסה 2 כבר לא בתהליכי פיתוח.

בפיתון מעבר נתונים המתחלק לשרת לקוח נעשה על ידי ספריית socket. ספרייה זו מאפשרת לייצר יישומים מבוססי שקעים בקלות רבה. השרת מאזין לבקשות והלקוח שולח אות להתחבר לשרת ומשם הם מחוברים על ידי שקע. התקשורת באמצעות שקעים מספקת למפתח גמישות ואבטחה ונחשבת ומאפשרת לו לפתח שלל של יישומים.

בנוסף לספריית socket שעוזרת לי להעביר מידע בין שרת ולקוח, אני משתמש גם במודול pygame. פייגיימס נועדה כדי ליצור משחקי מחשב בעזרת פיתון והיא בדיוק מה שהייתי צריך ליצירת משחק מחשב לפרויקט. היא כוללת דרכים להגדיר עצמים וקבוצות של עצמים, דרך לפתוח מסך נפרד ולהציג עליו תמונות שניתן לשמור כמשתנים. זאת ועוד, בפייגיימס פעולות נוספות המקילות על תכנות משחקי מחשב. לדוגמה, פעולה הבודקת התנגשות של עצמים או קבוצות עצמים, או היכולת ליצור mask סביב תמונה על מנת לבדוק מגע רק בחלקים מוגדרים של תמונה, שהיא מרובעת.

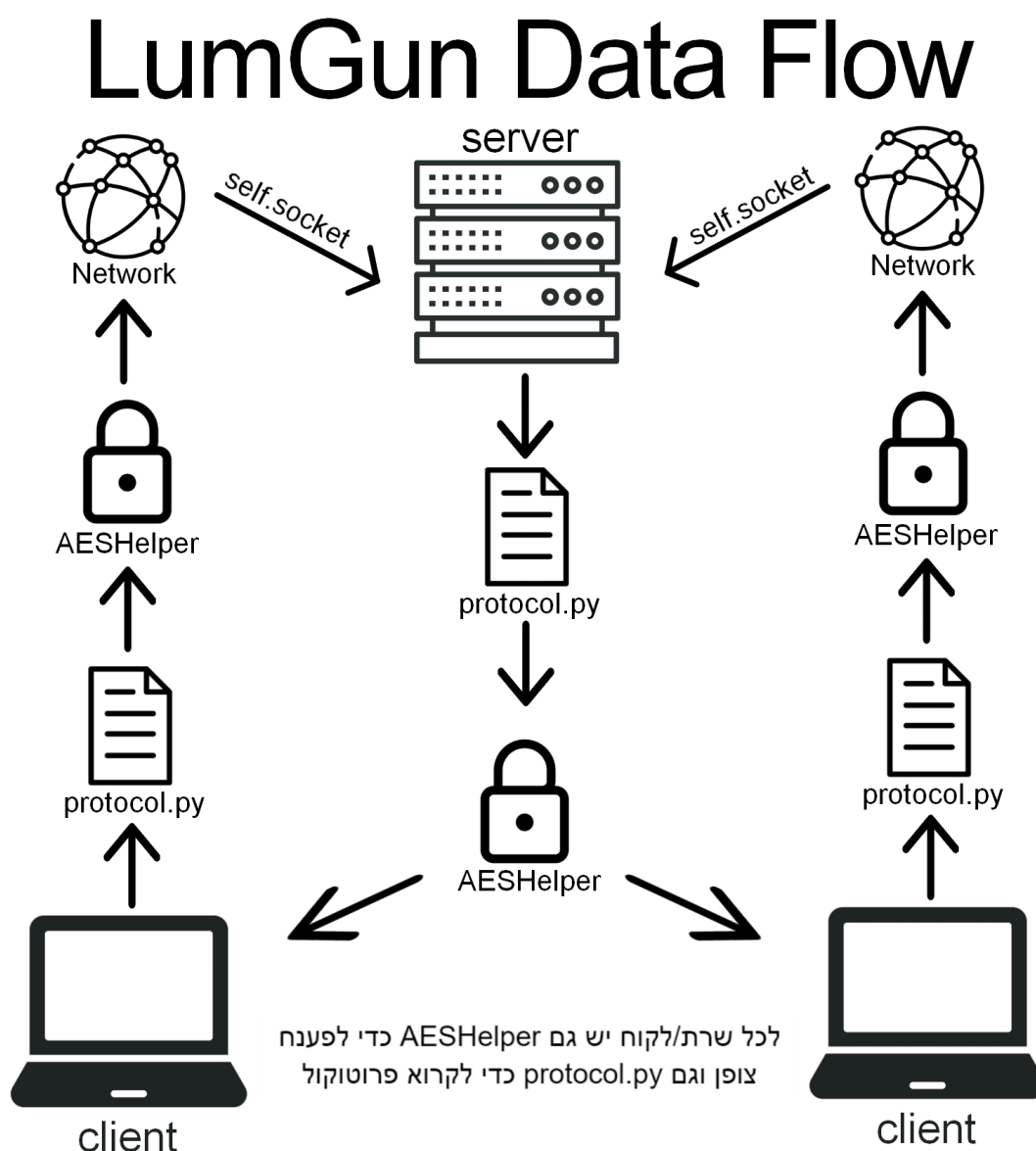
יתר על כך, כדי לאפשר העברה ושמירה בטוחה של נתונים אני משתמש גם בספרייה PyCryptodome. בעזרת ספרייה זו ניתן ליצור אובייקט המצפין ומפענח צופן של AES. אני משתמש בה כדי להעביר מידע בין לקוח לשרת לאחר תהליך תיאום המפתח הראשוני. בנוסף על כן, אני משתמש במודול hashlib כדי להעביר לשרת ולאחסן בו סיסמאות מוצפנות כדי להגן עליהן מפריצה זדונית למאגר הנתונים.

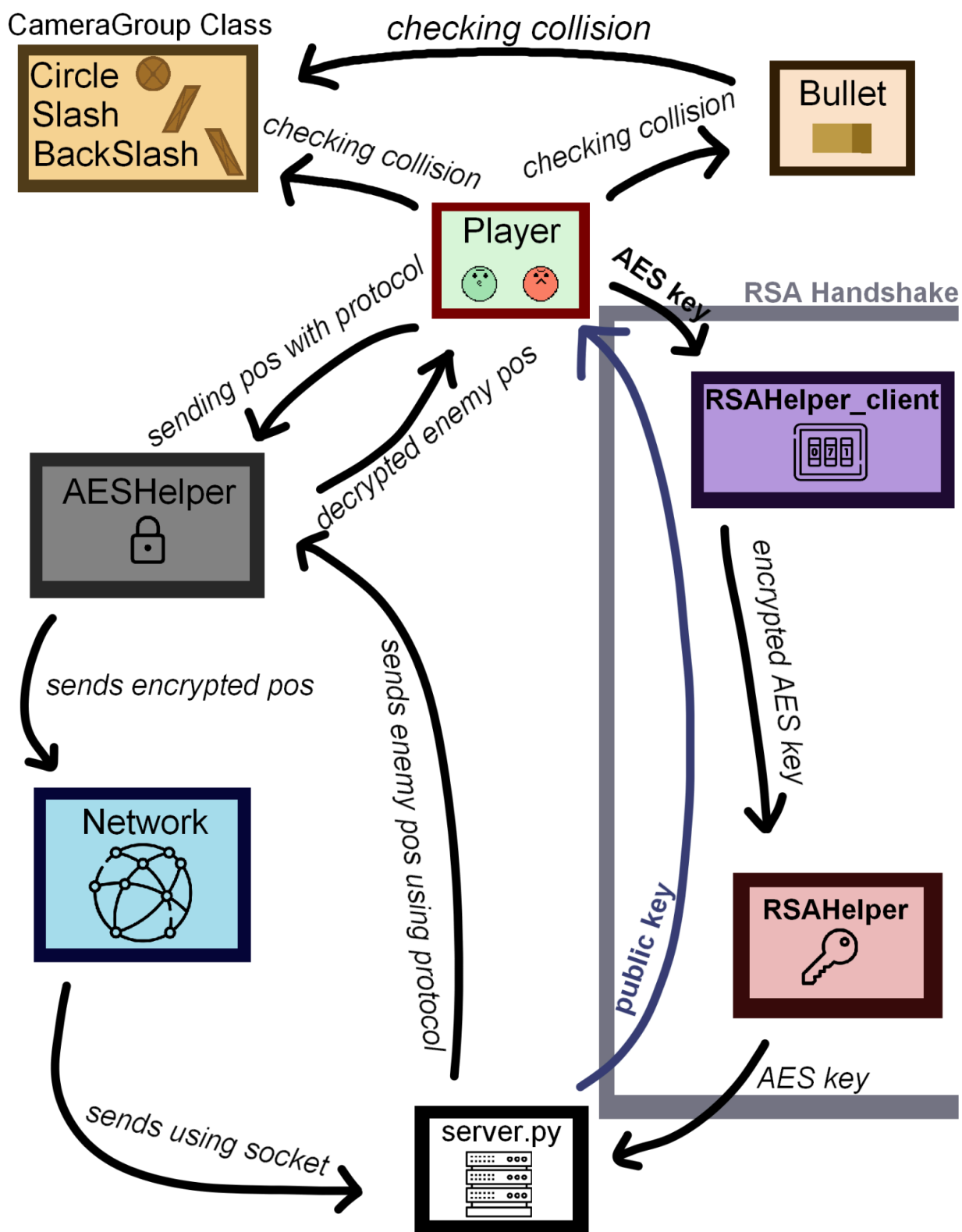
תיאור זרימת המידע במערכת

עבור כל יכולת:

[ישנו פירוט פרוטוקולים העונה על אופן זרימת המידע לפי כל סוג הודעה בהמשך.]

זרימת המידע בפרוייקט נעשית על ידי שקעים, כאשר ללקוח עצם מסוג Network המתווך בינו לבין השרת בתהליך העברת המידע. בנוסף על כך, המידע המועבר עובר תהליך המרה למחרוזת מתאימה לפרוטוקול שהגדרתי ודרך הצפנה של AES כדי להימנע מחשיפה של פרטים אישיים.





אלגוריתמים מרכזיים

לחיצת יד RSA:

בפרויקט התבקשתי לאבטח את המידע המועבר בתקשורת בין השרת ללקוח, כדי למנוע האזנה זדונית לפרטים אישיים שעוברים בתקשורת. לשם כך רציתי שכל התקשורת בין השרת ללקוח תתבצע עם מפתח סימטרי בעזרת מערכת ההצפנה AES-128. אך ברעיון זה הייתה פרצה, איך עלי לשלוח את המפתח הסימטרי בין השרת ללקוח בצורה בטוחה? ראשית רציתי שהלקוח ישלח אל השרת את המפתח AES שהוא רוצה להשתמש בו להמשך התקשורת, אך אם מישהו זדוני מסניף את המפתח המועבר, הוא יכול לפענח את הצופן בעצמו ולגלות פרטים חסויים. לכן פסלתי את הרעיון של העברה בסיסית ולא מוצפנת של המפתח הסימטרי ושאפתי להשתמש במערכת מפתחות פומביים למשתמשים ופרטיים לשרת כדי שהמפתח יעבור מוצפן. בעזרת מפתח פומבי של השרת, המשתמש יצפין את המפתח הסימטרי שלו וכך רק השרת עם המפתח הפרטי שלו יכול לקרוא את המפתח שהעביר אליו הלקוח.

כדי לבצע את תהליך זה יכולתי להשתמש שנית בספריית PyCryptodome שבה השתמשתי כדי להצפין בעזרת מערכת AES-128. במקום זאת, כדי להעשיר את הידע שלי בתחום ההצפנה בחרתי ליצור בעצמי מערכת RSA. המערכת שיצרתי משתמשת בעיקרון מוכר, כל פעם שהשרת רץ מחדש היא מייצרת מפתחות חדשים. היא מייצרת מספרים ארוכים מאוד המשתמשים כמפתחות, וכדי להצפין או לפענח הודעה היא משתמשת במשוואות עם הפלט הרצוי. המפתח הסימטרי מועבר לפורמט של מספר מוצפן ומפוענח משם.

קריאה וכתובה של הודעות ע"פ פרוטוקול:

בפרויקט שלי השרת והלקוח יכולים להעביר סוגים רבים של מידע אחד לשני. כדי לסמן לאיזה מידע לצפות בהודעה מסויימת רציתי להוסיף לכל הודעה משהו שמסמל כבר בתחילתה לאיזה מידע לצפות, סוג של פרוטוקול. הפרוטוקול היחיד שהשתמשתי בו עד לרגע זה היה מה שרציתי להשתמש בו במקור, פרוטוקול שמקציב את שני התווים הראשונים של ההודעה לאורך ההודעה, וזהו. בסופו של דבר לא בחרתי בפרוטוקול זה משתי סיבות שונות. הראשונה, לא הבנתי בתחילת הפרויקט שעלי לשלוח סוגים שונים של הודעות, בין אם זה הודעה למצב משחק, פרטי משתמש או בקשה של מפתח. ולכן, הייתי חייב הרבה סוגים שונים של הודעות עם מערכת רחבה יותר שמתאימה את עצמה לקבלה של המידע המועבר.

בסופו של דבר החלטתי על צירוף ספרה לתחילת ההודעה כדי לסמל את סוג ההודעה. לכל סוג הודעה כתבתי פונקציה בקובץ נפרד שמנסחת אותה ואחרת שקוראת אותה. בנוסף, כדי להתייחס לאי ודאות של קריאת הודעה, יצרתי פעולת read אוניברסלית שמאבחנת את סוג הקריאה המתאים להודעה שמתקבלת.

תיאור סביבת הפיתוח

על מנת לעבוד על הפרוייקט חיפשתי סביבת עבודה מתאימה, כזו שתאפשר לי גם לתכנת וגם להריץ את הקוד שלי. בחרתי להשתמש בסביבת העבודה Visual Studio Code, או בקיצור VS Code. זוהי סביבת עבודה שפותחה על ידי מיקרוסופט, הופצה לציבור בגרסה מלאה ב-2016. VS Code היא סביבת עבודה מאוד פופולרית בשל מספר כלי העזרה לפיתוח הנוכחים בה. התוכנה מדגישה מילים שונות בקוד כדי להפוך אותו לקריא יותר, משלימה פעולות או משתנים בהתאם להקלדה וכוללת ספריה של הרחבות לתוכנה המפותחות על ידי הקהילה. בחרתי לעבוד ב-VS Code הודות לתכונות שאני מאוד אוהב בה.

1. קל מאוד ליצור בה קבצים, כל קובץ מכל סוג, מסוגל לרוץ ולפעול ללא הגדרה קודמת או תהליך של אתחול.

2. התוכנה מעוצבת נפלא, המראה של הקוד מאוד חשוב לי כי הדגשה יפה ומדויקת של מילות מפתח בקוד יכולות לעזור לי לנווט בו בקלות רבה יותר.

3. כשרציתי ללמוד לתכנת ב-pygame ללא קשר לפרוייקט הזה, סרטון ההסבר להתקנה של python ו-pygame השתמש ב-VS Code ואני פשוט רציתי לעקוב אחרי ההוראות.

4. התוכנה מספקת דרך מהירה מאוד להריץ כמה קבצים של קוד בו זמנית, תכונה זו נחוצה כי רבות בגלל שאני יוצר משחק מרובה משתמשים.

5. ההרחבות בה עוזרות מאוד, בכל פעם שאני רוצה פונקציונליות חדשה אני יכול להוריד עוד הרחבה.

6. ה-terminal שנפתח בעת הרצת קוד עוזר מאוד לניהול קבצים באמצעות מערכת git.

פרוטוקול התקשורת

כפי שציינתי לפני שני פרקים, פרוטוקול התקשורת שלי מורכב בסך הכל מתו אחד בתחילת ההודעה המסמל את כוונת המסר. התווים הפותחים המוגדרים על ידי הפרוטוקול שלי הם הספרות 1 עד 8 ולכל אחת תפקיד שונה ללא קשר לסדר לוגי כלשהו, כאשר רציתי להעביר סוג הודעה חדשה הוספתי מספר בעל משמעות שונה. לכל סוג הודעה (פרט לבודדים) יש פונקציות הכנה וקריאה ייחודיות. פונקציות ההכנה מכינה מחרוזת המכילה את כל הפרטים מופרדים על ידי פסיק ובראשה את מספר הפרוטוקול. הפונקציה הקוראת הייחודית מקבלת את המחרוזת ללא המספר המסמל שלה ומפרקת אותה ל-tuple בהתאם לצורך. כדי לענות לצורך לקרוא אפשרויות רבות של הודעות יצרתי גם פונקציית קריאה אוניברסלית המעבירה את הקריאה לפונקציות הקריאה הייחודיות בהתאם לתו הפרוטוקול.

אתאר את משמעות כל שמונה ספרות הפרוטוקול שלי.

1 - בקשה למאגר הנתונים:

הבקשה נשלחת מהלקוח אל השרת במטרה לתקשר בדרך כלשהי עם מאגר הנתונים, מהווה בקשה ליצור משתמש חדש או להתחבר למשתמש קיים. הבקשה כוללת שלושה משתנים המופרדים על ידי פסיק: שם המשתמש, סיסמה, ומשתנה בוליאני העונה על "האם זוהי בקשה להתחבר למשתמש קיים?". במידה והמשתנה הבוליאני בסוף שווה ל-False, אז הבקשה מיועד ליצירת משתנה חדש.

2 - העברת מצב משחק:

ההודעה נשלחת מהלקוח לשרת, כדי להעביר לו את מצב המשחק שלו, מיקום וכו'. ההודעה כוללת שישה משתנים המופרדים על ידי פסיקים:

- מיקום x של דמות השחקן.
- מיקום y של דמות השחקן.
- מיקום x של העכבר השחקן.
- מיקום y של העכבר השחקן.
- ערך בוליאני המסמל האם השחקן יורה קליע.
- מדד חיים, מ0 (מוות) עד 30.

מיקומי העכבר של השחקן נועדו כדי לחשב את זווית ההסתכלות של השחקן, לאן הוא מכוון.

3 - תשובה ממאגר נתונים:

ההודעה נשלחת כתשובה לפרוטוקול 1, המבקש לתקשר עם מאגר הנתונים כחלק מתהליך ההתחברות של הלקוח. בעוד שפרוטוקול 1 מעביר פרטים לגבי רצון ההתחברות, פרוטוקול 3 מהווה ערך בוליאני שמועבר מהשרת ללקוח לגבי סטטוס ההתחברות שלו. אם ההודעה החוזרת היא True אז השחקן התחבר בהצלחה, במידה וההודעה היא False אז משמעות האות היא דחייה והסיבה תלויה בבקשה המקורית. אם הבקשה הייתה להתחבר למשתמש קיים אז למשתמש תוצג הודעה שאומרת שהפרטים שהוא הזין לא מתאימים. אם הבקשה המקורית הייתה ליצור משתמש חדש אז ההודעה שתוצג תהיה ששם המשתמש שהוא בחר כבר תפוס.

4 - העברת מצב משחק + שם:

הודעה זו נשלחת מהשרת ללקוח, מעבירה בדיוק את אותם הפרטים לגבי היריב של המשתמש כמו שהוא שלח לשרת עם תוספת אחת, השם. בנוסף למיקום הדמות, מיקום העכבר, סטטוס ירייה ומד החיים. בהודעה שהשרת מעביר לשחקן לגבי היריב שלו מועבר פרמטר נוסף המהווה מחרוזת של היריב. כל הפרטים מופרדים על ידי פסיקים.

5 - מוות:

הודעה זו נשלחת לשרת מהלקוח, כאשר הוא מזהה שמד החיים שלו נפל ל-0. במחלקה של השחקן, כאשר שחקן מזהה מוות ושהוא השחקן הראשי הוא שולח '5' בלבד לשרת. בעת הקריאה, פונקציית הקריאה לא נפתרת מה-5. לאחר שהשרת מבין שיש מקרה של מוות, הוא מעדכן את מספר ההריגות במאגר נתונים בהתאם ושולח בחזרה את מיקום ההתחלה של השחקן כדי שהוא יוכל להמשיך לשחק.

6 - בקשה למפתח פומבי:

בדומה לפרוטוקול 5, גם זה כולל אך ורק את ספרת הפרוטוקול ('6') ובעת הקריאה ה-6 לא נמחק. הודעה זו נשלחת מהלקוח לשרת, זוהי ההודעה הראשונה שאמורה להישלח לשרת והיא מבקשת ממנו את המפתח הפומבי כדי להתחיל את תהליך לחיצת היד.

7 - מפתח פומבי:

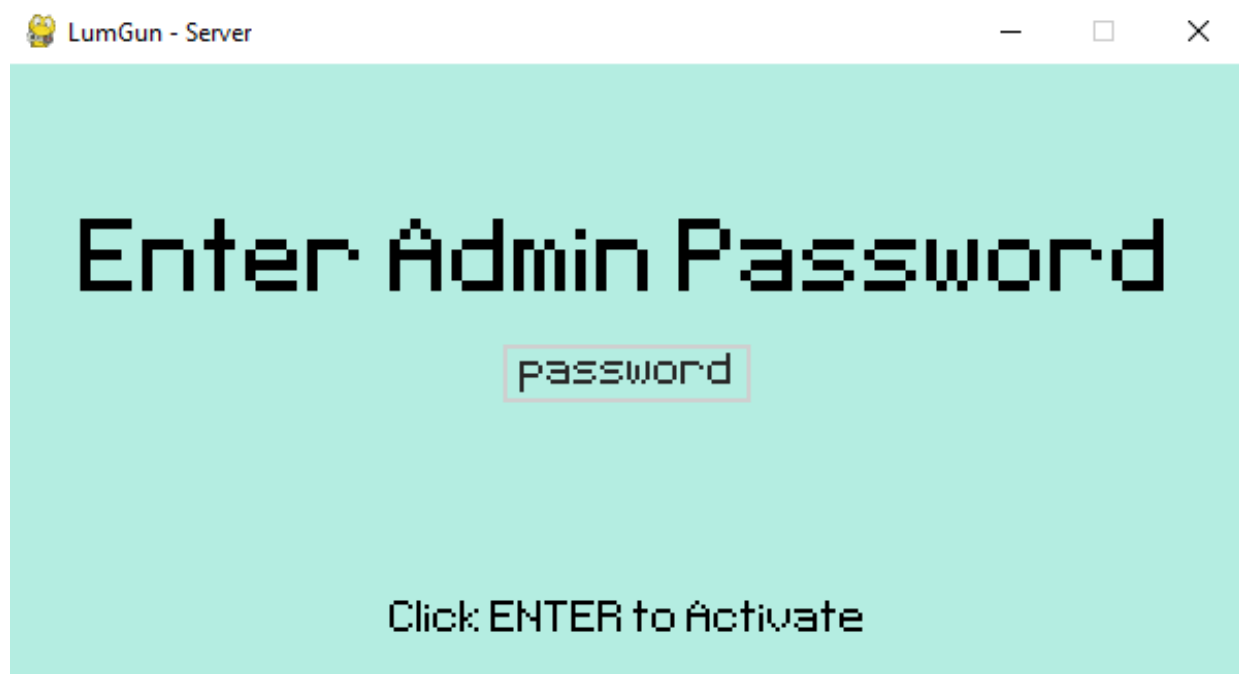
נשלח מהשרת ללקוח, זהו המפתח הפומבי של השרת, כתשובה לפרוטוקול של '6'. המפתח הפומבי כולל בתוכו את e: מספר שלם שתמיד יהיה 65537 שהוא המפתח הפומבי, וגם המספר n הנוכח במשוואת ההצפנה והפענוח שכן משתנה. שני המספרים האלו מופרדים על ידי פסיק.

8 - מפתח סימטרי מוצפן:

הודעה זו מגיעה מהלקוח לשרת, גם כחלק מפרוטוקול לחיצת היד. לאחר שהלקוח קיבל את המפתח הפומבי, הוא לוקח את המפתח הסימטרי שלו, ממיר אותו למספר ארוך ומצפין אותו באמצעות המפתח הפומבי. את המספר הזה הוא שולח בחזרה לשרת כדי שהוא יפענח בחזרה את המפתח הסימטרי המקורי. פרוטוקול 8 מעביר את המפתח הסימטרי המוצפן.

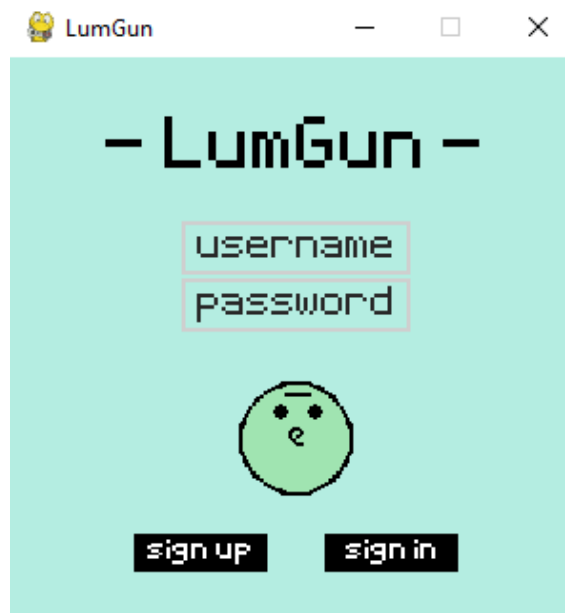
תיאור מסכי המערכת

מסך אימות מנהל:



המסך נפתח בעת הרצת הקוד של השרת, ומטרתו היא לוודא שרק אדם עם סיסמת המנהל יכול לפתוח שרת. על המסך ההוראה "Enter Admin Password", ותיבת טקסט שבה אמורים להזין את הסיסמה. כדי לשלוח את הסיסמה לבדיקה, על המנהל ללחוץ על מקש ה-Enter על פי ההוראה הרשומה למטה. הקוד בודק את הסיסמה על ידי השוואה של ערך ההצפנה שלה דרך sha256, כך שגם מי שמסתכל על הקוד לא יכול לדעת מהי הסיסמה. לאחר הזנה של הסיסמה הנכונה החלון נסגר והשרת נפתח.

מסך כניסה למשתמש:



זהו המסך שהמשתמש רואה לאחר שפרוטוקול לחיצת היד עבר בהצלחה, מאותו רגע כל הפרטים שיעברו בינו לבין השרת יהיו מוצפנים. המסך מאפשר למשתמש להכניס שם משתמש וסיסמה וללחוץ על אחד מהכפתורים למטה כדי להתחבר למשתמש קיים או ליצור אחד חדש. במידה והוא יכניס פרטים לא תקינים תופיע לו לשלוש שניות הודעה אדומה המפרטת על הפגם. לדוגמה:

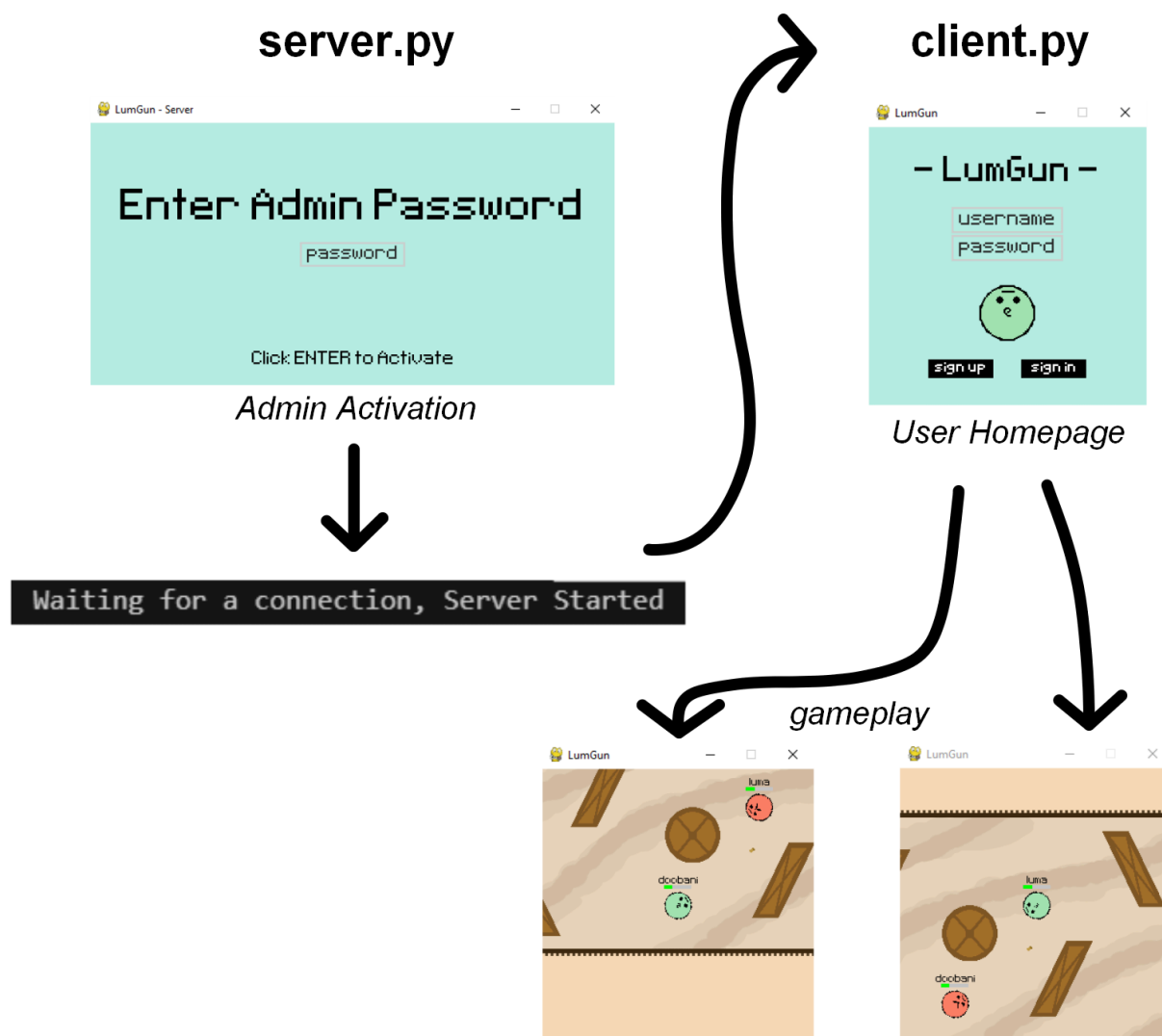


מסך משחק:



זהו המסך העיקרי בכל הפרויקט, מהווה את חווית המשחק עצמה. המשתמש יכול ללכת במפה עם המקשים w, a, s, d ולהמשיך לראות עצמו באמצע המסך. הוא מסתכל תמיד לכיוון העכבר שלו ובעת לחיצה על העכבר הוא יורה קליע על היריב שלו. על המפה מפוזרים באופן סימטרי מכשולים, כאשר העיגול שבתמונה נמצא באמצע.

LumGun - Screen Flow Diagram



תיאור מבני הנתונים

מבני נתונים:

מבנה הנתונים המשומש ביותר אצלי בפרויקט הוא הרשימה. הרשימה עזרה לי הכי הרבה בצד השרת, כאשר אני רוצה לבקר כניסות ויציאות של שחקנים ממנו. בשרת הגדרתי מספר רשימות עם שני ערכים בתוכם, אחד לכל שחקן שנכנס.

יש את רשימת מיקומי ההתחלה (נקרא spawn_pos), הכוללת את שני המיקומים ההתחלתיים האפשריים של שני השחקנים, כאשר מד החיים שלהם מורה על 30. כאשר שחקן מת, השרת מזהה את הפקודה הזו ומיד שולח לו חזרה את המיקום ההתחלתי שמשוייך לו כדי שהוא יוכל לחזור למשחק. הנה קטע הקוד בשרת שמנהל מוות של שחקן:

```
# if the player reports about his own death
if data == '5':
    # changes his stored pos to his spawn pos
    pos[player] = spawn_pos[player]
    # adds a kill to the other player
    addKill(player_usernames[other(player)])
    # sends the dead player his spawn position

conn.sendall(make_cipheriv(aes_lst[player].aes_encrypt(make_pos((pos[player])))))
    continue
```

בנוסף על כך יש את רשימת המיקומים (נקרא pos), המאחסן את המיקום הנוכחי של השחקנים. כאשר שחקן מעביר מידע לגבי המיקום שלו לשרת, השרת בתגובה שומר את המידע הזה ברשימת המיקומים, ומעביר לשחקן השולח את המיקום של השחקן האחר (שמאוחסן בתא השני של הרשימה). השרת עושה את זה עבור כל שחקן וכך הוא מעביר בין השניים את המידע לגבי השני. הנה קטע הקוד שמנהל את רשימה זו ושולח את המיקומים:

```
# if it's not a death the server will continue like normal
else:
    # storing the players position into the pos list
    pos[player] = data

# setting the reply to be the other players position
reply = pos[other(player)]
```

```
# server prints information
print(f"Received from player {player + 1}: ", data)
print(f"Sending to player {player + 1}: ", reply)

# sending information back to the client:
# position and name, using protocol 4
conn.sendall(make_cipheriv(aes_lst[player].aes_encrypt(make_name(*reply,
player_usernames[other(player)]))))
```

עוד רשימה שמאוד חשובה להתנהלות השרת עם שני השחקנים היא הרשימה של החיבורים (נקראת connections), שמאחסנת את השקעים של השחקנים. בעת החיבור של משתמש חדש, לפני פתיחת תהליכון חדש, השרת בודק את הרשימה הזו למקומות פנויים. במידה והוא מוצא מקום פנוי הוא פותח תהליכון חדש ושולח אליו כפרמטר את השקע החדש. במידה והשרת לא מצא מקום פנוי הוא סוגר את השקע החדש שנוצר. הנה קטע הקוד האחראי על זה, מילוי המקום הריק ברשימת השקעים קורה לאחר תהליך לחיצת היד:

```
# accepting connections
conn, addr = s.accept()
print("Connected to:", addr)

# finding player slots
player_slot = -1
for i in range(2):
    if connections[i] is None:
        player_slot = i
        break

# if both the slots are filled
if player_slot == -1:
    # print msg
    print("Server full, cannot handle more connections.")
    # close the new connection
    conn.close()
    continue
```

עוד רשימה חשובה לשרת היא רשימת שמות המשתמשים (נקראת `player_usernames`), בעת אישור ממסד הנתונים לגבי חיבור משתמש, שם המשתמש נשמר בו ונשלח ליריב כדי שייציג אותו מעל ראשו. שימוש ברשימה זו מופיע בקוד שצירפתי להדגמות האחרות.

מסד נתונים:

בפרויקט שלי בחרתי לאכסן את מאגר המשתמשים שלי במסד נתונים מסוג קובץ `json`. בחרתי בקובץ `json` בשל הפשטות שנדרשה כדי לתפעל אותו וגם בשל הפופולריות שלו. כדי לאפשר תקשורת מסודרת עם מסד הנתונים גם הכנתי קובץ פעולות נבחרות על מסד הנתונים.

כדי לאחסן מידע בקלות השתמשתי באחת מדרכי האחסון הנפוצות ביותר, המשלבת שני סוגים של מבני נתונים. בקובץ `json` שלי נמצא מילון, שבו זוג מפתח וערך אחד, המפתח הוא `"users"` והערך הוא רשימה של כל המשתמשים. בכל תא של רשימת המשתמשים יש מילון משל עצמו הכולל כמפתחות את שלושת המאפיינים הנחוצים לפרויקט שלי: שם משתמש, סיסמה (מוצפנת ב-`sha256`) ומספר הריגות. השימוש במילונים מאפשר גישה מהירה לנתונים והאחסון ברשימה מאפשר מעבר זריז על כל המשתמשים לצרכים שונים. הנה דוגמה למסד הנתונים שלי:

```
{
  "users": [
    {
      "username": "luma",
      "password":
"3fc9b689459d738f8c88a3a48aa9e33542016b7a4052e001aaa536fca74813cb",
      "kills": 55
    },
    {
      "username": "doobani",
      "password":
"3fc9b689459d738f8c88a3a48aa9e33542016b7a4052e001aaa536fca74813cb",
      "kills": 27
    }
  ]
}
```

סקירת חולשות ואיומים

עבודה עם בסיס נתונים:

אחת הבעיות המרכזיות עם עבודה מול מאגר נתונים היא השימוש הזדוני בתווים מיוחדים. למזלי, החולשה הזו מעולם לא הייתה בקוד שלי, בגלל אופי העבודה שלי עם json. הדרך שבה אני שומר את הפרטים שהמשתמש מקליד היא דרך הוספה למחרוזת בעזרת unicode:

```
username += event.unicode
```

לאחר בדיקות אישיות, גיליתי שדרך זו מזינה את תו הציון '\ ' לפני תווים בעייתיים ובכך מונעת בלבול של תו " שהוזן על ידי המשתמש לעומת אותו אחד הנמצא בקוד. אבל ליתר ביטחון ולמען שמות תקינים ויפים, אני מסנן עבור הא"ב האנגלי, מספרים ומקף תחתון. רק הם יכולים להירשם כשם משתמש או סיסמה, לפי השורות האלו:

```
allowed_characters =  
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_"  
# check if the character is in the allowed set  
if event.unicode in allowed_characters:  
    username += event.unicode  
username += event.unicode
```

עוד חולשה של עבודה עם מאגר נתונים היא הפחד שאדם זדוני יגיע לאותו מאגר נתונים ויצפה בפרטים אישיים של שחקנים. הפרט היחיד שעלול לפגוע במשתמש שלי במידה וייחשף היא הסיסמה שלו ולכן הייתי חייב להעביר אותה בדרך מיוחדת. כדי לוודא שאף אחד לא יוכל להגיע לסיסמה הייתי חייב לוודא שאפילו אני לא יכול להגיע לסיסמאות המשתמשים שלי, ועשיתי זאת על ידי הצפנה חד כיוונית. בעזרת פונקציית hash מסוג sha256, המשתמש מצפין כל סיסמה שהוא שולח לשרת ולמאגר הנתונים בצורה בלתי הפיכה. כל בדיקה של סיסמה נעשית בהשוואת הסיסמה המוצפנת דרך sha256. כך, גם למאגר הנתונים אין גישה לסיסמה של המשתמש ורק לייצוג שלה, זה מגן עליה מפריצה למאגר נתונים.

ריבוי שרתים:

הקוד שכתבתי לא מובנה לפתיחה של הרבה שרתים, לכן שאפתי למנוע פתיחה של הרבה שרתים על ידי אדם זדוני שיש לו גישה לגרסת exe של הקוד שלי. כאשר הקוד של השרת רץ לראשונה, נפתח חלון המבקש סיסמה. הסיסמה הזו לא נודעת לאף אחד חוץ ממני והתקינות שלה נבדקת על המחשב עצמו. כדי לוודא שאף אחד גם לא יכול לקרוא את הקוד ולפענח את הסיסמה, גם סיסמה זו מוצפנת דרך sha256 בקוד עצמו. כאשר אדם מזין סיסמה היא מוצפנת על ידי sha256 ורק אז נבדקת עם הסיסמה הנכונה שגם היא מוצפנת. כך, יצרתי מערכת שבה אני הוא האדם היחיד שיכול לפתוח שרת של המשחק.

הצפנה:

כפי שכבר ציינתי בפרק של האלגוריתמים המרכזיים, בפרויקט מתבצע תיאום של מפתח AES-128 סימטרי בעזרת פרוטוקול 'לחיצת יד' בין השרת ללקוח. על השרת והלקוח להעביר את מפתח זה בצורה סודית דרך פרוטוקול לחיצת היד:

- השרת מעביר למשתמש מפתח פומבי
- הלקוח מצפין ושולח בחזרה את המפתח הסימטרי שלו בעזרת המפתח הפומבי
- השרת מפענח בחזרה את המפתח הסימטרי באמצעות המפתח הפרטי שלו

הסתקרנתי מאוד לדעת איך המערכת של מפתח פרטי ופומבי עובדת ולכן בחרתי להכין אותה בעצמי. היות וציינתי גם את פרוטוקול לחיצת היד גם בפרק ה'אלגוריתמים מרכזיים', ארחיב רבות בהמשך על מערכת זו (בפרק "קטע קוד יפה / פירוט אלגוריתמים").

מימוש הפרויקט

סקירת ספריות ומחלקות מיובאות

:pygame

ספרייה המאפשרת ליצור מסך חיצוני ולצייר עליו את כלל האלמנטים של המשחק. כולל גם מחלקות ועצמים העוזרים לפיתוח.

:pygame.display

עצם זה מהווה את המסך המציג את המשחק, בעל פעולה של blit המציירת את כל האלמנטים של המשחק אחת ל-60 שניות.

:pygame.image

עצם המחזיק בו תמונה שניתן להעלות מקבצי המחשב, אחד מהפרמטרים הניתנים ליצור על המסך.

:pygame.rect

עצם של מלבן, משמש בקוד שלי על מנת ליצור תיבת טקסט וכפתור, ניתן לעצב בגדלים וצבעים שונים.

:pygame.event

מודול תחת הספרייה של pygame העוזר לקלוט קלט של לחיצות עכבר או מקלדת מהמשתמש.

:pygame.time

מודול תחת הספרייה של pygame המשמש בקוד שלי לשתי פעולות מרכזיות, יצירה של אובייקט הגורם ללולאת המשחק לרוץ 60 פעמים בשנייה ומדידת שלוש שניות להצגה של הודעת שגיאה במסך הבית.

:sys

בתוך כל קובץ המשתמש ב-pygame בדרך כלל מוסיפים את exit מהספרייה הזו, על מנת לסגור את החלון והתוכנה במידה וכפתור האיקס בחלון נלחץ.

:math

הספרייה כוללת מגוון רחב של פעולות מתמטיות המשמשות לחישובים רבים לאורך הפרויקט. כמו לדוגמה: כדי לחשב את זווית הסיבוב של השחקן לעומת העכבר, נאלצתי להשתמש בפונקציה הטריגונומטרית tan כדי לחשב את זווית הסיבוב וגם להמיר את התוצאה למעלות מרדיאנים, בשתי פעולות אלו השתמשתי בספרייה.

```
x_dist = mouse_pos[0] - player_pos[0] + camera_group.offset[0]
y_dist = -(mouse_pos[1] - player_pos[1] + camera_group.offset[1])
self.angle = math.degrees(math.atan2(y_dist, x_dist))
```


:socket

מחלקה זו מאפשר למתכנת ליצור שקע המחובר בין השרת ללקוח, ומשם להעביר דרכו אותות. היא חיונית לפרויקט שלי המבצע תקשורת בין לקוחות לשרת.

:thread

ספרייה זו מאפשרת לפתוח תהליכונים לשרת המתקשר עם מספר רב של לקוחות. אני משתמש בה כדי להריץ בשרת פונקציה מספר פעמים עבור כל משתמש שנכנס.

```
start_new_thread(threaded_client, (conn, player_slot))
```

:PyCryptodome (Crypto)

ספרייה זו כוללת מספר רחב של כלי עזר להצפנה, אני משתמש בה כדי להצפין באמצעות AES-128.

:Crypto.Cipher.AES

אובייקט המקבל מפתח, סוג תת-הצפנה וווקטור התחלתי (iv) המסוגל לפעח ולהצפין.

:Crypto.Random.get_random_bytes

פעולה המקבלת מספר ומייצרת רצף בייטים באורך הפרמטר, רצף בייטים זה ישמש את הלקוח כמפתח וכווקטור התחלתי (iv) משתנה.

:Crypto.Util.Padding.pad, Crypto.Util.Padding.unpad

פעולות עזר החיוניות להצפנה ופענוח על פי AES. כיוון ש-AES מחלקת את ההודעה לטבלה של 4 על 4, כלומר מחלקת את ההודעה לקבוצות של 16, עלינו להתאים את אורך ההודעה להתחלק ב-16. כדי לעשות זאת, pad מצרפת תווים לסוף ההודעה כדי להצפין ו-unpad מסירה אותם לאחר פענוח.

:hashlib

ספרייה זו כוללת מגוון פעולות הצפנה, משמש בה ליצור הצפנת sha256 לסיסמאות.

:sympy

משתמש בה ליצור פעולת nextprime שלה, המאפשרת לי ליצור בקלות מספרים ראשוניים ליצור הצפנת ה-RSA שלי.

:random

כדי להפוך את הצפנת ה-RSA שלי ליחודית כל הרצה, אני משתמש בספרייה זו כדי לקבל מפתחות שונים בכל הרצה.

:json

מאפשרת לי לעבוד מול מסד נתונים מסוג json, לטעון אותו לתוך משתנה שאני יכול לסרוק בפשטות ולהחזירו לקובץ json.

סקירת מחלקות שלי

נתחיל במחלקות שיצרתי עבור הרצה של הלקוח:

:Circle

העיגול שנמצא באמצע המפה.

תכונות:

image - תמונה של העיגול שציירתי.
rect - מלבן סביב התמונה לגודל והצגה.
mask - מסיכה, הופכת את הפגיעה בעיגול לפגיעה בציור שלו ולא במלבן המקיף אותו.

:Slash

מחלקה למכשולים בצורת לוכסנים (/)

תכונות:

image - תמונה של הלוכסן שציירתי.
rect - מלבן סביב התמונה לגודל והצגה.
mask - מסיכה, הופכת את הפגיעה בלוכסן לפגיעה בציור שלו ולא במלבן המקיף אותו.

:Backslash

מחלקה למכשולים בצורת לוכסנים אחוריים (\).

תכונות:

image - תמונה של הלוכסן האחורי שציירתי.

rect - מלבן סביב התמונה לגודל והצגה.

mask - מסיכה, הופכת את הפגיעה בלוכסן האחורי לפגיעה בציור שלו ולא במלבן המקיף אותו.

Bullet:

מחלקה לקליע אחד הנורה משחקן, נע קדימה בזווית ייחודית לו עד שהוא פוגע במשהו.

תכונות:

original_image - התמונה המקורית של הקליע שציירתי, לפני שהיא מסובבת לכיוון התנועה.

rect - ריבוע המקיף את הקליע.

angle - שיפוע של כיוון התנועה.

magnitude - כמות הפיקסלים שהקליע יזוז בכל 1/60 שנייה, עוצמת מהירות.

image - תמונה של הקליע לאחר סיבוב (transform) בהתאם לזווית התנועה.

mask - מסכה, מתחשב בסיבוב של תמונת הכדור למען חישוב פגיעה.

פעולות:

collision_sprite - לא מקבל פרמטרים, מחזיר בוליאן של האם הקליע יצא מהמפה או פגע במשהו.

update - לא מקבל ערכים, מקדם את הקליע. בודק אם collision_sprite יצא אמת, אם כן אז הפעולה מעלימה את הקליע.

Button:

מחלקת כפתורי הרשמה והתחברות במסך הבית של הלקוח. לכפתור אין ניהול של לחיצה, הוא ריבוע עם טקסט והוא משנה צבע כאשר העכבר מעליו.

תכונות:

button_rect - המקום והגודל של הכפתור, כולל בתוכו x, y, width, height שכולם מוזנים בפעולת ה-init.

color_passive - הצבע של הכפתור כאשר העכבר לא נמצא עליו.

color_active - הצבע של הכפתור כאשר העכבר כן נמצא עליו.

`color` - צבע נוכחי של כפתור, מוגדר בהתחלה כ `color_passives` אבל מתעדכן בהתאם.
`button_text_surf` - המשטח (מה שמצויר על המסך) של הטקסט שעל הכפתור, מוגדר בעזרת טקסט, צבע טקסט וגודל טקסט שנתונים בפרמטרים של `init`.
`button_text_rect` - ריבוע המוגדר לגודל של הטקסט באמצע של `button_rect`, מהווה מציין מקום לאיפה שמציירים את הטקסט.

פעולות:

`update` - לא מקבל פרמטרים, בודק אם מיקום העכבר על הכפתור ומשתנה את `color` בהתאם.
`draw` - מצייר קודם את `button_rect` ואז מעליו את `button_text_surf`.

CameraGroup

קבוצה (`pygame.sprite.Group`) המחזיקה בה את המפה, והמכשולים בה. מקבלת בפעולת הציור שלה את השחקן ואת היריב. זאת משום שהיא זו שמציירת את הכל עם `offset` בהתאם לתזוזה של השחקן כדי ליצור את האשלייה של מעקב מלמעלה אחרי השחקן.

תכונות:

`display_surface` - המסך שעליו מציירים את הכל, שמור בתור תכונה.
`offset` - ערך וקטורי המראה איך לצייר את הסביבה ביחד לשחקן כדי שהוא ישאר באמצע.
`half_w` - חצי מהרוחב של המסך, צריכים להתחשב בזה כי אנחנו רוצים להשאיר את השחקן באמצע
`half_h` - חצי מהגובה של המסך, צריכים להתחשב בזה כי אנחנו רוצים להשאיר את השחקן באמצע
`ground_surf` - המשטח (תמונה) של הרצפה
`ground_rect` - ריבוע המקיף את המשטח של הרצפה, כדי לסמן מיקום לציור

פעולות:

`center_target_camera` - מקבלת את השחקן כפרמטר, מחשבת את וקטור `offset`, על ידי בדיקה של המיקום של השחקן.
`custom_draw` - מציירת (`blitting`) את הרצפה, כל המכשולים, הקליעים והשחקן והיריב הניתנים כפרמטרים על פי המיקום שלהם פחות הערך הווקטורי `offset`, כדי להשאיר את השחקן באמצע.

Player:

מחלקת שחקן, משמש גם לשחקן הראשי וגם את ליריב, יכול לזוז, להסתכל, לירות, לאבד חיים עד מוות.

תכונות:

isUser - ניתן כפרמטר ל-init, ערך בוליאני המתאר האם השחקן הוא האחד הראשי (אם 'שקר' אז הוא היריב).

original_image - תמונה מקורית שאני ציירתי של השחקן, לפני סיבוב לפי הסתכלות, תהיה שונה בהתאם לאם השחקן הוא יריב או ראשי (ערכו של isUser).

image - תמונה של השחקן, מסובבת לפי ההסתכלות של השחקן.

rect - ריבוע מסביב לשחקן, נועד לצורך מעקב אחר השחקן וציון מיקומו.

direction - ערך וקטורי המסמל על כיוון ההליכה של השחקן, משתנה בהתאם ללחיצה על מקשי התנועה.

speed - מהירות השחקן, כמה פיקסלים הוא זז כל 1/60 שנייה.

angle - זווית הסיבוב של השחקן, מחושב לפי מיקום העכבר.

mask - מסכה, מאפשר לעקוב אחר התנגשות של התמונה של השחקן עם הסביבה.

health - כמות חיים שיש לשחקן, יורדת בעת פגיעה (יש גם עוד משתנים הקשורים לזה כדי להציג את מד החיים בצורה ויזואלית, לא נכנס לזה אלו פשוט צבעים וריבועים, משתנים בהתאם לחיים, לא מעניין).

username - שם משתמש של השחקן.

username_surf - משטח של הטקסט שעליו רשום השם משתמש.

username_rect - מלבן סביב שם משתמש, נועד לציון מיקום בעת ציור, מורם מעל השחקן.

פעולות:

set_username - פעולה המקבלת שם משתמש ומשנה את self.username בהתאם.

face_mouse - מקבלת את מיקום העכבר כפרמטר, משתמשת בו כדי לחשב את זווית הסיבוב של השחקן ולסובב את תמונת דמותו בהתאם.

create_bullet - מקבלת את מיקום העכבר לצורך בדיקת זווית, יוצרת קליע חדש בעת ירייה.

input - משנה את הגודל הווקטורי direction בהתאם ללחיצות על מקשי התנועה.

border_collision - בודקת האם השחקן יוצא מגבולות המפה, אם כן היא מחזירה אותו אחורה.

obstacle_collision - בודק אם השחקן מתלכד עם אחד מהמכשולים על המפה, מרחיק אותו ממנו בהתאם.

bullet_collision - בודק אם קליע מתלכד עם השחקן, אם כן הוא דואג להעלים את הקליע ולהוריד נקודה אחת ממד החיים.

die - פונקציה מופעלת כאשר שחקן ראשי מת, שולחת לשרת אות למוות, מקבלת בחזרה אות של מיקום התחלתי ומשנה את המיקום של השחקן בהתאם.

update - מריצה את כל הפונקציות הנ"ל באחת, מזיזה את השחקן לפי הוקטור כיוון, מפעילה את פונקציות die אם השחקן מת.

ועכשיו ניתן לעבור גם על מחלקות ההצפנה שיצרתי:

AESHelper:

מחלקה המשמשת ליצור או להשתמש במפתח AES-128 על מנת להצפין ולפענח הודעות על פי AES.

תכונות:

key - מפתח AES-128, מורכב מ-16 בייטים (לכן נקרא 128, כי זה כמו 128 ביטים), נועד להצפנה ולפענוח.

פעולות:

__init__ - מגדירה את המפתח, יכולה לקבל פרמטר שישמש כמפתח, עוזר להגדרה של העצם בשרת לאחר שהועבר אליו מפתח. יכולה לקבל גם לא לקבל פרמטרים והיא תיצור מפתח משל עצמה, משמש את הלקוח בקביעת מפתח.

get_key - מחזירה את המפתח.

aes_encrypt - פונקציית ההצפנה, מקבלת את הטקסט המיועד להצפנה ומגרילה iv שעל פיו להצפין יחד עם המפתח. מחזיקה tuple עם הטקסט המוצפן וה-iv.

aes_decrypt - פונקציית הפענוח, מקבלת טקסט מוצפן ואת ה-iv שהצפין אותו. בעזרת פרטים אלו היא מחזירה את ההודעה המקורית.

RSAHelper:

מחלקה המשמשת את השרת ליצור מפתח פומבי ופרטי ולהשתמש בהם לצורך הצפנה ופענוח בעזרת מספרים.

תכונות:

public_key - המפתח הפומבי בפורמט של tuple, מוצג כך (e, n).
private_key - המפתח הפרטי בפורמט של tuple, מוצג כך (d, n).

פעולות:

__init__ - פעולת האתחול, לא מקבלת פרמטרים, יוצרת את שני המפתחות.
get_public_key - מחזירה את המפתח הפומבי.
rsa_encrypt - מצפינה את המספר שהועבר כפרמטר בעזרת המפתח הפומבי.
rsa_decrypt - מפענחת את המספר שהועבר כפרמטר בעזרת המפתח הפרטי.

RSAHelper_client:

מחלקה נחותה מRSAHelper, נועדה ללקוח, לא מייצרת אלא מקבלת מפתח פומבי ויכולה רק להצפין בעזרתו.

__init__ - פעולת האתחול, מקבלת מפתח פומבי ושומרת אותו.
get_public_key - מחזירה את המפתח הפומבי.
rsa_encrypt - מצפינה את המספר שהועבר כפרמטר בעזרת המפתח הפומבי.

ועכשיו נעבור למחלקה המשמשת לתקשורת של הלקוח:

Network:

מחלקה המפשטת את תהליך התקשורת של הלקוח עם השרת. השרת פונה למחלקה זו כדי לתקשר בפשטות עם השרת.

תכונות:

client - שקע המחובר בין הלקוח לשרת.

server - ה-ip של השרת.

port - ה-port של התקשורת.

addr - צמד של ה-ip וה-port של השרת.

pos - מיקום התחלתי של הלקוח.

פעולות:

getPos - פעולה המחזירה את המיקום ההתחלתי של הלקוח, משומש לאתחול לקוח.

connect - פעולה המחקרת את השקע לשרת.

send_before - פעולה המפשטת את תהליך שליחה וקבלה של הודעות לפעולה אחת שמנהלת שליחה וקבלה, ה-before אומר לפני סיום תהליך לחיצת היד ולכן בגלל שלא צריך להשתמש ב-AES יש המרה מבייטים.

send - פעולה זהה לקודמת רק שאין המרה מבייטים בעת קבלת מידע, כי ה-AES משתמש בבייטים.

send_no_answer - רק שליחה של מידע לשרת, ללא קבלת תשובה, משומש כדי לשלוח מפתח סימטרי מוצפן לשרת.

סקירת מודולים שלי

:my_sha256

מחלקה המבצעת את הצפנת ה-sha256 על מחרוזות.

:AES_protocol

הפרדה וצימוד של הטקסט המוצפן יחד עם ה-iv כדי ששניהם יועברו רצף בייטים אחד דרך השרת, חשוב להצפנה ולפענוח של AES.

:database_manager

איחוד של פעולות המתקשרות מול מסד הנתונים, משמש על ידי השרת לקבלת מידע על תהליכים במסד נתונים. כולל פעולות כמו בדיקה של קיימות שם משתמש, או בדיקה של נכונות של שם משתמש וסיסמה.

:protocol

כוללת פעולות כתיבה וקריאה עבור על תו פרוטוקול משלו. בעלת פעולת קריאה אוניברסלית המשייכת כל הודעה לפעולת קריאה ייחודית.

קטע קוד יפה / פירוט אלגוריתמים

פרוטוקול העברת מפתחות מוצפן RSA:

בפרק ה'אלגוריתמים מרכזיים' ציינתי את השיקולים שלי לבחירה בהצפנת RSA כדי להעביר בין שרת ללקוח את המפתח הסימטרי שהם חולקים. סוג ההצפנה הזה חשוב מאוד לבטיחות של מעבר נתונים בין השרת ללקוח ופה אפרט על איך הוא עובד.

מערכת ההצפנה/פענוח RSA שהכנתי פועלת על פי השלבים האלו, שרובם קורים בהגדרת המחלקה RSAHelper שנפתחת בשרת לאחר אימות סיסמת מנהל. אחרי כל שלב אציג את קטע הקוד האחראי לבצעו, קטעי הקוד המוצגים הם למעשה קטע אחד רצוף המקוטע על ידי ההסברים שלי:

- נגדיר שני מספרים ראשוניים p, q , מאוד גדולים ולא שווים זה לזה.

```
# generate p - large prime
p = sympy.nextprime(randint(2**1018, 2**1023))

# generate q - large prime
q = sympy.nextprime(randint(2**1018, 2**1023))

# regenerate q if needed
while q == p:
    q = sympy.nextprime(randint(2**1018, 2**1023))
```

נגדיר את n , שווה למכפלה של p ו- q , ישמש אותנו למשוואת הצפנה ופענוח.

```
# constant component of public/private keys
n = p * q
```

נגדיר את ϕ , שווה ל- $(q - 1) * (p - 1)$ יעזור להכין להגדיר את המפתח הפומבי שלנו.

```
# helps create e, public key
phi = (p - 1) * (q - 1)
```

נגדיר את e , המפתח הפומבי, מספר זר ל- ϕ (המחלק המשותף הגדול ביותר שלהם הוא 1)

```
# very common coprime to phi
e = 65537

# if e is coprime with phi, exit the loop, we have what we need
```

```
# if it's not coprime, q and p will be regenerated
if gcd(e, phi) == 1:
    break
```

כל הקוד של יצירת המפתחות נמצא בלולאה `while True`. ה-`break` מהווה סיום תהליך החיפוש.

נמצא את d , המפתח הפרטי שלנו על ידי הנוסחה $d = (1/e) \bmod \phi$ (ϕ זה `phi`)

```
# d = (1/e)(mod phi)
d = pow(e, -1, phi)
```

וכעת, בגלל שאנחנו צריכים את n לנוסאות ההצפנה והפענוח אנחנו צריכים לצרף אותו למפתחות, לכן המפתח הפומבי הוא (e, n) והפרטי הוא (d, n)

```
self.public_key = (e, n)
self.private_key = (d, n)
```

נסמן הודעה לא מוצפנת ב- m והודעה מוצפנת ב- c

נוכל להצפין הודעות כך:

$$c = m^e \bmod n$$

```
def rsa_encrypt(self, message):
    """
    this function is in charge of encrypting the AES key,
    the parameter is the binary formatted key and this function
    turns it from a binary to int and then encrypts it using RSA
    """
    # separating the public key to e and n
    e, n = self.public_key
    # converting the AES key to int
    message_as_int = int.from_bytes(message, byteorder='big')
    # encrypting {c = (m^e)(mod n)}
    encrypted_data = pow(message_as_int, e, n)
    return encrypted_data
```

ולפענח אותן כך:

$$m = c^d \bmod n$$

```
def rsa_decrypt(self, encrypted_message):
    """
    this function is in charge of receiving the encrypted as int
    AES key and decrypting it back to the original binary AES key
    """
    # splitting the private key to d and n
    d, n = self.private_key
    # decrypting {m = (c^d)(mod n)}
    decrypted_int = pow(encrypted_message, d, n)
    # returning msg to bytes
    # adding 7 and floor dividing by 8 the bit length to calculate the length
of the decrypted msg
    decrypted_message = decrypted_int.to_bytes((decrypted_int.bit_length() +
7) // 8, byteorder='big')
    return decrypted_message
```

ההצפנה מתבצעת על ידי מחלקה בלקוח שיכולה רק להצפין על ידי קבלה של מפתח פומבי (RSAHelper_client), והפענוח מתבצע אצל השרת במחלקה שיכולה ליצור מפתחות, להצפין ולפענח (RSAHelper).

מאותה נקודה לשרת וללקוח מפתח AES-128 והם ימשיכו להשתמש בו בלבד, הצפנה סימטרית זו נחשבת לבטוחה ביותר כל עוד המפתח לא נודע לאחרים.

מעקב "מצלמה" אחר שחקן:

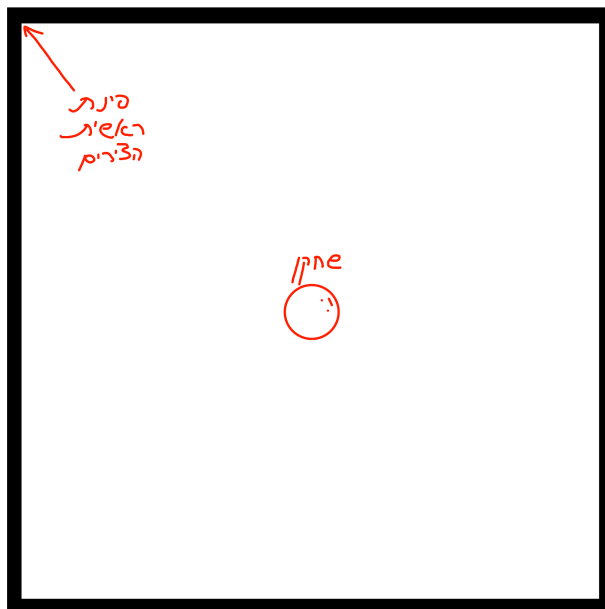
כאשר דמיינתי את המשחק שלי לראשונה, דמיינתי אותו דומה למשחקים מנקודת מבט של גוף שלישי אחרים, שבהם המצלמה עוקבת אחר השחקן. לכן, רציתי שבמשחק שלי השחקן תמיד ישאר באמצע המסך.

לרוב שאנשים מפתחים משחקים הם מתכנתים בתוך מרחב שהם יכולים לעצב עם אובייקט של מצלמה. מנגד, בתכנות בpygame אין "מצלמה", ניתן לצייר דברים במיקומים שונים במסך ואפילו מחוצה לו. כדי לכתוב מערכת שתגרום לשחקן שלי להיות באמצע המסך פניתי ליוטיוב למצוא פתרון, ולאחר סינון של מערכות מעקב לא רצויות, מצאתי את האחת שאני חיפשתי.

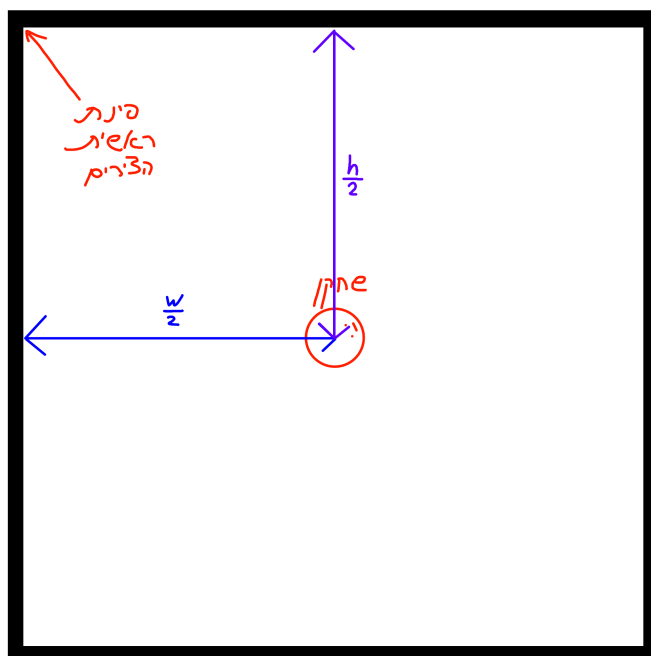
הלוגיקה של המערכת הזו בונה כולה על האשלייה שכדי להזיז מצלמה עוקבת, ניתן להשאיר את השחקן במקום ולהזיז את הסביבה של בכיוון המנוגד, אבל בשיטה זו יש כשל לוגי. כיוון שאני באמת רוצה שלשחקן יהיה מיקום בסביבה שאני אוכל לשלוח לשחקן האחר, אני צריך למצוא דרך לשלב שני דברים שנראים מנוגדים: איך אני בלוגיקה של המשחק מזיז את השחקן במפה אך השחקן רואה את המפה זזה תחתיו?

כדי שאוכל לענות על השאלה הזו צריך קודם להבין את אופי הציור של pygame. pygame הנורמה היא שכל דמות במשחק תהיה מורכבת ממלבן כלשהו ותמונה, וכדי לצייר את הדמות עלינו לקרוא לפעולה blit. בתוך הפעולה הזו ניתן להזין את התמונה של הדמות כפרמטר של "מה לצייר?" ואת המלבן המקיף את התמונה כפרמטר ל"איפה לצייר?". אבל כדי גם לצייר לשחקן שהרצפה היא זו שזזה במסך שלו ועדיין לשמור על מיקום לוגי של שחקן שזז עלינו להפריד בין התמונה למלבן המקיף אותה. במערכת שלי, כאשר דמות זזה המיקום של המלבן שלה משתנה וכך כאשר אני משתמש במיקום המלבן כדי לצייר אותה על המסך זה נראה שהתמונה היא זו שזזה. למרות שהתמונה נראית כאילו היא זזה, הדבר החשוב קורה בכלל מאחורי הקלעים, המלבן זז. וזה חשוב שהמלבן הוא זה שזז כי לפי המלבן נבדקות התנגשויות בין שחקן לקיר או לקליע. כך, ניתן למצוא פתרון כלל המרצה את שתי הדרישות: נוכל להשאיר את כל לוגיקת המשחק שנבנית על המלבנים שזזים כפי שהיא ורק בעת הציור להשתמש במיקום אחר מזה של המלבן כדי לצייר למשתמש את הדברים. ככה שהמשתמש יראה בעצם בבואה ממקום אחר על המפה של מה שקורה.

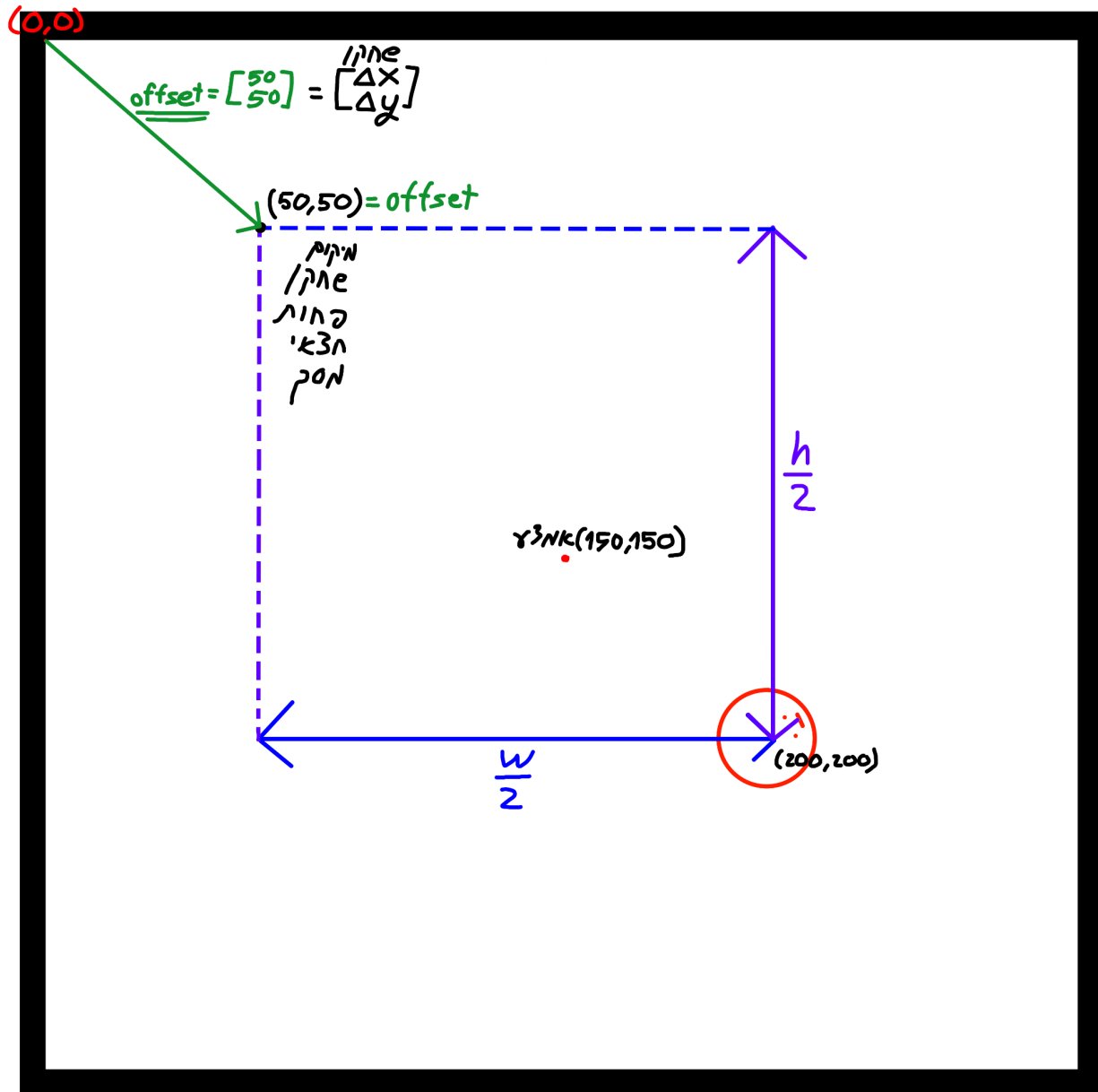
עכשיו, רק ניתן לענות על השאלה "איך נוכל לקבוע את המיקום המצויר של דברים?". דבר ראשון, נוכל לקבוע שאנחנו רוצים להשאיר את כל הדברים על המפה יחסית אחד לשני באותו מיקום, לכן נוכל פשוט לקחת את המיקום שלהם ולהפעיל עליהם סטייה כלשהי. הסטייה הזו יכולה לבוא באופן של ערך וקטורי, שבמקרה של הקוד שלי נקרא **offset**. כדי לעדכן את **offset**, אני יכול לאחסן את ההעתק של פינת המסך הרצויה (150 פיקסלים מעל ושמאלה מהשחקן בכל רגע) מראשית הצירים. ואז כאשר יש לי את העתק זה במשתנה, בעת הציור אוכל לצייר את השחקן באמצע המסך **ולאחסן** את **offset** מהמיקום המקורי של הסביבה שלו. זה עונה בצורה מושלמת על העיקרון של "תנועה לכיוון אחד של דמות השחקן, המשתמש יראה תנועה מנוגדת מהסביבה שלו". לפני כל ציור של אלמנט על המסך, עלי להכין **offset_pos** של האלמנט ששווה למיקום האמיתי שלו (המלבן שלו) פחות וקטור **offset**. כל תנועה של השחקן שתזיז את הווקטור לכיוון אחד, כיוון שאנחנו מחסרים וקטורית זה יהיה לכיוון השני. עכשיו, איך נוכל למצוא את ההעתק של הפינה הרצויה כדי לענות על שאלה זו נוכל להשתמש בפינה הזו:



הפינה הזו בפתיחת החלון (אם השחקן מוצב ב-150, 150) נחשבת כ- $(0, 0)$ שזה בדיוק מה שאנחנו מחפשים, בעת הגדרת שני השחקנים, נוכל לדעת כמה הם קרובים / רחוקים מהפינה $(0, 0)$ האוניברסלית. במידה והשחקן עומד ב- $(150, 150)$ נוכל לחשב את הפינה הזו על ידי החסרה של חצי מאורך המסך וחצי רוחב המסך. אנחנו בעצם מחפשים את ההעתק של הפינה הרצויה המדומה של המסך (כלומר ביחס לשחקן שימצא באמצע המסך זה יראה כמו ראשית הצירים) מ- $(0, 0)$.



נוכל למצוא את העתק פינת המסך הרצויה מ(0,0) על ידי חישוב מיקומו של השחקן, פחות חצאי אורכי המסך. אדגים לפי דוגמה שבה גודל המסך הוא (300, 300) והשחקן זז 50 פיקסלים לכיוון החיובי של כל ציר (ציר x מוצב ימינה וציר y מוצב מטה).



כך חישבנו את העתק פינת המסך המדומה ביחס לראשית הצירים האוניברסלית. נוכל להחסיר את offset מכל מיקום של מכשול במפה לפני הציור על המסך שלו. ועכשיו, אחרי שהסברתי את הלוגיקה מאחורי המעקב, אכנס גם ליישום של מערכת זו. כדי לצייר את כל הפריטים על המסך על פי אותו משתנה offset, בחרתי ליצור מחלקה המציירת עבורה את כל הדברים על פי אותה סטייה, היא נקראת camera_group וכבר דיברתי עליה, והקוד שנמצא בה הוא קטע הקוד היפה, הפותר בעיה אלגוריתמית חשובה.

המחלקה camera_group היא בעצם קבוצת ספרייטים (pygame.sprite.Group), שבתוכה נכללים כל המכשולים על המפה.

```
# initializing sprites in camera group
camera_group = CameraGroup()
b1 = Backslash((320, 320), camera_group)
b2 = Backslash((680, 180), camera_group)
s1 = Slash((600, 300), camera_group)
s2 = Slash((400, 200), camera_group)
c1 = Circle((500, 250), camera_group)
```

בעוד שרק המכשולים הם חלק מהקבוצה, המחלקה מציירת גם את שני השחקנים ואת האדמה.

האדמה היא תכונה של המחלקה.

השחקנים ניתנים לפעולת הציור של המחלקה כפרמטרים.

נמשיך בפעולה center_target_camera, המעדכנת את offset (שבמקרה שלנו הוא תכונה של המחלקה), המקבלת כפרמטר במקרה שלנו את השחקן.

```
def center_target_camera(self, target):
    """
    this method is in charge of updating the camera offset,
    in which objects around the map need to be offsetted by
    in order to look like they are the ones moving around the player,
    looks at the offset while target being the player in the center
    """
    self.offset.x = target.sprite.rect.centerx - self.half_w
    self.offset.y = target.sprite.rect.centery - self.half_h
```

ועכשיו, הנה פעולת הציור של המחלקה, שימו לב לחישוב של offset בעזרת הפעולה

center_target_camera. ועבור כל אלמנט בתמונה היא מייצרת עבורו offset_pos חדש שמותאם לפי offset, ורק אז מציירת אותו.

```
def custom_draw(self, player, enemy):
    """
    this method uses the offset in order to create the illusion of
    a camera following the player around, by moving everything around him
    while blitting them into the screen, uses the player as the main "target"
    """
```



```
# calculating the offset (players displacement)
self.center_target_camera(player)

# blitting ground
ground_offset = self.ground_rect.topleft - self.offset
self.display_surface.blit(self.ground_surf, ground_offset)

# blitting all of the bullets in the bullets group
for sprite in bullets.sprites():
    bullet_offset_pos = sprite.rect.topleft - self.offset
    self.display_surface.blit(sprite.image, bullet_offset_pos)

# blitting player
player_offset = player.sprite.rect.topleft - self.offset
self.display_surface.blit(player.sprite.image, player_offset)

# blitting enemy
enemy_offset = enemy.sprite.rect.topleft - self.offset
self.display_surface.blit(enemy.sprite.image, enemy_offset)

# blitting all of the obstacles on the map
for sprite in self.sprites():
    offset_pos = sprite.rect.topleft - self.offset
    self.display_surface.blit(sprite.image, offset_pos)
```

ומכאן ואילך, הפעולה גם מציירת את מדד החיים שמעל השחקנים וגם את השם של השחקן מעליו
(פחות משמעותי):

```
# blitting player health
player_center_offset = player.sprite.rect.center - self.offset
self.display_surface.blit(player.sprite.health_bar_background,
                           (player_center_offset[0] - 15, player_center_offset[1] - 22))
self.display_surface.blit(player.sprite.health_bar_health,
                           (player_center_offset[0] - 15, player_center_offset[1] - 22))

# blitting enemy health
enemy_center_offset = enemy.sprite.rect.center - self.offset
self.display_surface.blit(enemy.sprite.health_bar_background,
```

```
        (enemy_center_offset[0] - 15, enemy_center_offset[1] - 22))
self.display_surface.blit(enemy.sprite.health_bar_health,
        (enemy_center_offset[0] - 15, enemy_center_offset[1] - 22))

# blitting player username
self.display_surface.blit(player.sprite.username_surf, (player_center_offset[0] -
player.sprite.username_rect.w//2, player_center_offset[1] - 32))

# blitting enemy username
self.display_surface.blit(enemy.sprite.username_surf, (enemy_center_offset[0] -
enemy.sprite.username_rect.w//2, enemy_center_offset[1] - 32))
```

מסמך בדיקות

הרצת חלון השרת ובדיקת תקינות:

אני מריץ את קוד השרת ומקווה לראות חלון המבקש סיסמת מנהל, כל הטקסט המורה מה לעשות קיים ותיבת ההקלדה עובדת. מוודא ש-pygame עובד ולמנהל גישה פוטנציאלית לפתוח שרת. תוצאות: החלון נפתח ללא בעיה, תיבת טקסט מתפקדת כראוי.

הכנסה של סיסמאות שגויות ותווים לא תקינים לחלון השרת:

אני בודק את תקינות העבודה שלי בבדיקת סיסמת המנהל. בודק שלא ניתן להקליד תווים מיוחדים ושבימדה והסיסמה היא לא האחת הנכונה, נכתבת הודעה מפורטת על כך. תוצאות: בגלל שלקחתי את תיבת הסיסמה מהלקוח, היא הציגה הודעת שגיאה לא נכונה כאשר הקלדתי סיסמה לא נכונה. פתרון: התאמה זריזה של תיבת הטקסט למטרה החדשה שלה עברה בהצלחה.

הסתכלות על פירוט תהליך ה-RSA שמפורט בהדפסה של שני הצדדים בעת חיבור של שחקן:

אני מפעיל את צד הלקוח ומסתכל על הודעות שמפיק הלקוח ושמפיק השרת. ההודעות ממוספרות ואמרות להתאים אחת לשנייה, כל מידע שעובד מתקבל בצד השני ומפתח ה-AES עובד מוצפן ובהצלחה. תוצאות: אם נעבור על מספרי ההודעות בסדר כרונולוגי נוכל לראות תיאור מדויק של תהליך העברת המפתח, עבר בהצלחה.

הרצת חלון של לקוח ובדיקת תקינות:

לאחר שהעברת המפתח עברה בהצלחה, על חלון ההרשמה של השחקן להיפתח כדי לאפשר לו להתחבר למשחק. תוצאות: פעמים רבות החלון לא נפתח, בדרך כלל כי פשוט שכחתי להריץ את השרת.

יצירת חשבון עם מספר תווים קטן מידי או שם שכבר קיים, הכנסת תווים לא תקינים:

בדיקה מעמיקה של הבדיקות המקומיות בחלון הכניסה. הבדיקות המקומיות כוללות בדיקות של קלט הנתונים מהמשתמש לפני התקשורת עם מאגר הנתונים. הבדיקות המקומיות מוודאות שתו לא תקין לא מוקלד ושם המשתמש מנסה ליצור חשבון חדש, ששם המשתמש והסיסמה שלו לא יהיו קצרים מידי. תוצאות: פעמים רבות הקלדתי תווים לא תקינים ואף תווים בעברית והם נקלטו במערכת, זה לא מצב אידיאלי.

פתרון: בדיקה של תקינות כל תו שנכתב, לוודא שהוא או אות באנגלית, מספר או מקף תחתון.

ניסיון יצירת חשבון עם שם תפוס, ניסיון התחברות עם פרטים שגויים:

בדיקה של שתי השגיאות האפשריות המגיעות מתקשורת עם מאגר הנתונים. מוודא שהתקשורת איתו עובדת בצורה חלקה והודעות השגיאה הן רלוונטיות. תוצאות: מאותו רגע שכתבתי את פעולות התקשורת עם מאגר הנתונים לא היו לי איתן בעיות בכלל, התקשורת עם מאגר הנתונים מצויינת ומציגה פלט כראוי.

התחברות לשרת עם משתמש תקין:

מוודא שיש אפשרות לשחקן שרוצה לשחק כראוי להתחבר גם למערכת. תוצאות: ברגע שהשחקן מקליד ומזין את הפרטים הנכונים הוא נכנס ישירות לזירת הקרב.

בדיקה שגם אם השחקן לא זז, השרת מקבל הודעות מוצפנות שונות הודות לשינוי של ה-iv:

בדיקת ההודעות שהשרת שולח על מנת לוודא שהצפנת ה-AES עובדת בצורה תקינה. בהצפנה זו כל הזמן משתנה ה-iv שנוסף לפענוח בנוסף למפתח ולכן, גם עבור פענוח דומה השרת אמור לקבל הודעות מוצפנות שונות. תוצאות: למרות שהקלט הקריא שהתקבל מהלקוח (שלא זז) זהה, ההודעה שהשרת שולח המציינת את המידע המוצפן שונה.

חיבור שני שחקנים, תזוזה במרחב של שני השחקנים:

חיבור שחקן שני למשתמש ותזוזה במרחב. לבדוק ששני השחקנים רואים אחד את השני זזים במרחב, ויכולים לירות אחד בשני.

תוצאות: בשל טעות לוגית, במקום למסור פשוט זווית סיבוב של השחקן אני בחרתי לשלוח שני פרמטרים המסמלים על מיקום העכבר שלו. זה הוביל להמון זמן שמערכת הכיוון לא הייתה מתואמת בין שני השחקנים.

פתרון: לא לשלוח את המיקום המקורי של העכבר שנשלח מהאויב אלא לשנות גם אותו בהתאם ליריב כדי שיהיה מותאם אליו.

הריגה של שחקן אחד כדי לוודא את תקינות מערכת ההריגה:

לירות בשחקן אחד עד שהוא מת, ולבדוק אם מספר ההריגות של ההורג עולה ואם השחקן המת חוזר לנקודת ההתחלה שלו.

תוצאות: הייתה לי בעיה שבעת הריגה, השחקן ההורג קיבל שתי הריגות במאגר הנתונים במקום אחת.

פתרון: הסתבר שהאות מוות נשלח לשרת פעמיים, פעם אחת מהלקוח שמת כי הוא מת, ופעם אחרת מהלקוח ההורג כי היריב שגם מוגדר כשחקן מת. כדי להגביל את מספר השליחות ל1, עשיתי כך שרק אם השחקן שמת הוא הראשי ולא היריב הוא יכול לשלוח הודעה המדווחת על המוות שלו.

התנתקות של שחקן אחד והתחברות עם משתמש אחר:

לוודא שהשרת יכול להמשיך כרגיל אם שחקנים יוצאים ונכנסים ממנו למרות שהשרת לא עובר אתחול ונשאר פתוח.

תוצאות: בהתחלה אחרי שהייתי מנסה להתחבר עם לקוח שלילי השרת היה קורס.

פתרון: להשתמש ברשימה של connections כפי שציינתי כבר בפרק על מבני נתונים, מתמלא בהתאם לכמות השחקנים וכאשר שחקן חדש רוצה להצטרף הרשימה נסרקה עבור מקום פנוי.

התנתקות סופית של שני השחקנים המחוברים והסתכלות על הודעות הניתוק של השרת:

לוודא שהשרת לא קורס גם כשאין לו שני שחקנים ומוודא לשלוח הודעות התנתקות מתאימות. תוצאה: השרת מזהה שלא נשלחת פקטה חזרה, סוגר את התהליכון ומדפיס הודעה בהתאם.

מדריך למשתמש

עץ קבצים

```
:C
AESHelper.py |
AES_protocol.py |
client.py |
database.json |
database_manager.py |
my_sha256.py |
network.py |
protocol.py |
RSAHelper.py |
RSAHelper_client.py |
server.py |
|
vscode.———|
launch.json |
settings.json |
|
font———|
Pixeltype.ttf |
|
graphics———|
Background_new.png |
Backslash.png |
Bullet.png |
Circle.png |
Enemy.png |
Player.png |
Slash.png |
```

```
|
|
| project_book——|
| ניב ליאם pdf.330780552 |
|
|
| __pycache__——L
|
| AESHelper.cpython-311.pyc
| AES_protocol.cpython-311.pyc
| database_manager.cpython-311.pyc
| my_sha256.cpython-311.pyc
| network.cpython-310.pyc
| network.cpython-311.pyc
| protocol.cpython-311.pyc
| RSAHelper.cpython-311.pyc
| RSAHelper_client.cpython-311.pyc
| Valumant.cpython-311.pyc
```

התקנת המערכת

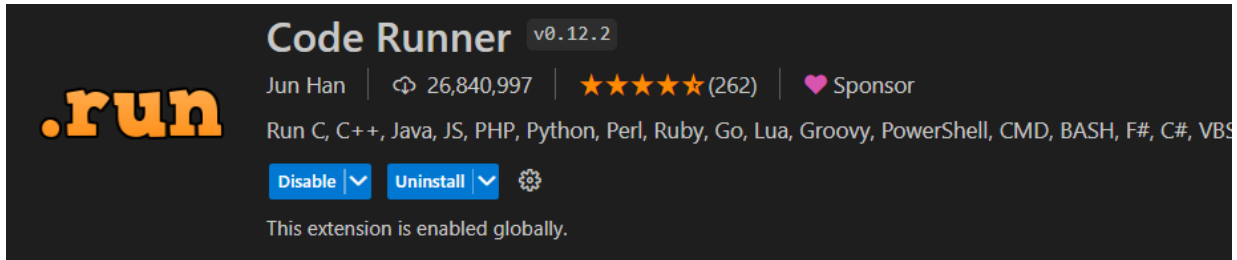
סביבה:

את תיקיית הפרויקט שלי, אני ממליץ לפתוח ב-visual studio code, זוהי התוכנה שאני פיתחתי עליה את המשחק ועליה אני אדגים איך להריץ את הפרויקט.

כלים:

כמובן על המשתמש להתקין את הגרסה המעודכנת ביותר של python 3, כי זו שפת התכנות שאני השתמשתי בה.

בנוסף לתוכנה הבסיסית של vs code, כדי להריץ את הפרויקט בנוחות אני ממליץ להוסיף את ההרחבה המאוד פופולרית Code Runner, המאפשרת להריץ קטעי קוד בצורה נגישה יותר.



בנוסף להרחבה זו, על מנת להשתמש בכלל הספריות החיצוניות שאני משתמש בהן, על המתקין להכנס ל-cmd, ושם להקליד את הפקודות הבאות:

```
pip install pygame
```

```
pip install sympy
```

```
pip install pycryptodome
```

קבצים:

- בתוך תיקיית הפרויקט שלי מספר תיקיות וקבצים, אעבור על מה התיקייה כוללת.
- כל קבצי הקוד והמחלקות נמצאים מחוץ לתיקיות, יש להריץ את הקבצים הנכונים לפי הצורך שעליו אפרט בהמשך.
- בתיקייה vscode יש את ההגדרות שלי לסביבת העבודה.
- בתיקייה __pycache__ יש בייטקוד של הקוד שרשמתי, לזמן הרצה מהיר יותר.
- בתיקייה code_illustrations יש ציורים שחלקם הגדול נמצאים בתיק זה המסבירים על המודל של הפרויקט.
- בתיקייה font שמור הפונט שאני משתמש בו כדי להציג טקסט על המסך.
- בתיקייה graphics יש את כל התמונות (גרפיקה) שנכללות במשחק, שצירתי.

נתונים התחלתיים:

יש להזין בקוד בקטע הגדרת כתובת ה-ip את הכתובת הנחפצת.
במידה ורוצים להריץ מקרה של לקוח, על המשתמש לשנות את כתובת ה-ip שבמחלקת network לזה של השרת.

```
class Network:
    def __init__(self):
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = "192.168.10.128"
        self.port = 5555
        self.addr = (self.server, self.port)
        self.pos = self.connect()
```


כדי לפענח מה כתובת ה-ip של השרת, במחשב השרת יש לכתוב בטרמינל או בשורת הפקודה את הפקודה "ipconfig", ומשם לבחור בכתובת השנייה.

```
Ethernet adapter | Ethernet:

Connection-specific DNS Suffix . : 
Link-local IPv6 Address . . . . . : fe80::da47:d3bc:1a9d:a05b%6
IPv4 Address. . . . . : 192.168.10.128
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.10.1
```

בנוסף, בקובץ מצד השרת, יש לוודא כי המשתנה server מקבל את הכתובת "0.0.0.0" על מנת להקשיב לכל פנייה.

```
192 server = "0.0.0.0"
193 port = 5555
194
```

בנוסף (למקרה שלקוח חיצוני לא מצליח להתחבר):

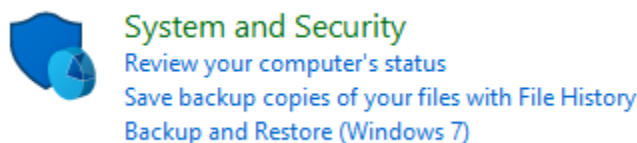
בנוסף, למקרה שה-firewall של השרת יחסום בקשות התחברות ותקשורת משחקנים ממחשבים אחרים יש לכבות את ה-firewall של windows:

ללחוץ win + R

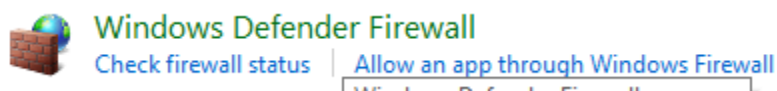
לכתוב בתיבת הטקסט "control"

ללחוץ ENTER

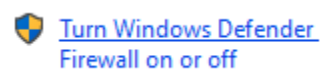
משם להגיע לcontrol panel ולהגיע להגדרות של האבטחה



שם ללחוץ על הגדרות ה-firewall



ושם, בצד ללחוץ על האפשרות להדליק ולכבות את ה-firewall



יש ללחוץ על שני סימוני הכיבוי

☐ Turn off Windows Defender Firewall (not recommended)

ולאחר מכן על אישור

OK

לאחר השימוש במכשיר כשרת יש להחזיר את ה-firewall כדי להגן על המחשב

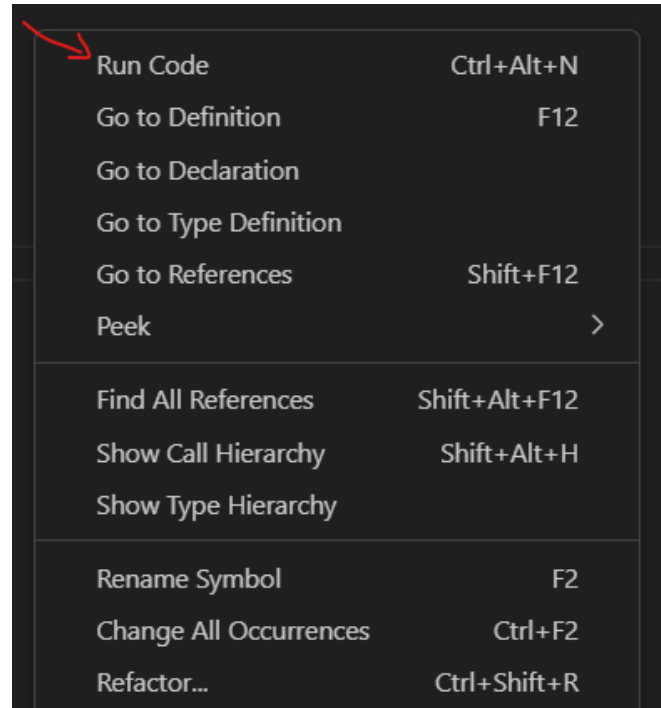
אכטקטורה מינימלית:

- מערכת הפעלה: windows 8.1, macOS 10.10 לפחות.
- מעבד: 2.5GHz או גבוה יותר.
- זיכרון גישה אקראית (RAM): לפחות 4GB.
- כרטיס גרפי: יכול להיות משולב בתוך המעבד.
- אחסון: 10GB לפחות.
- גרסת python: מומלצת הגרסה העדכנית ביותר של python 3.
- חיבור לרשת: אמין וקבוע.

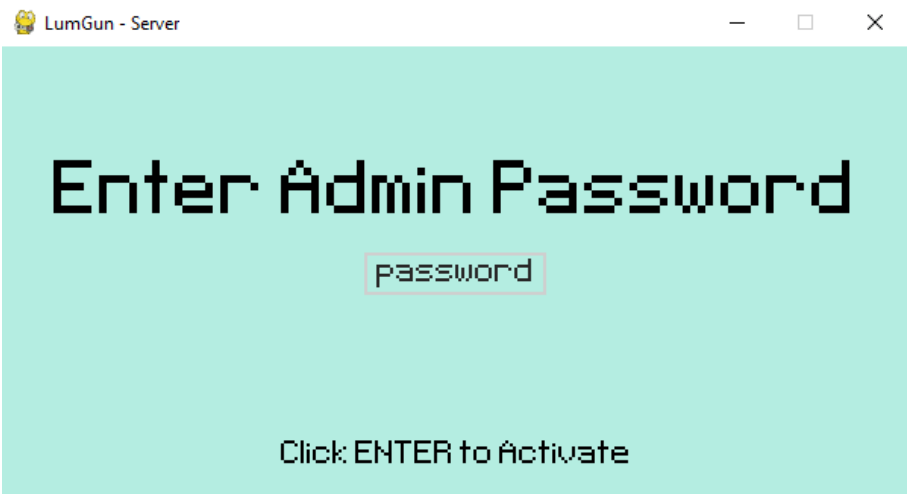
אופן הפעלת המערכת

שרת:

יש להריץ את הקוד של השרת בעזרת לחיצה ימנית על קטע הקוד, ולאחר מכן ללחוץ על "Run Code".



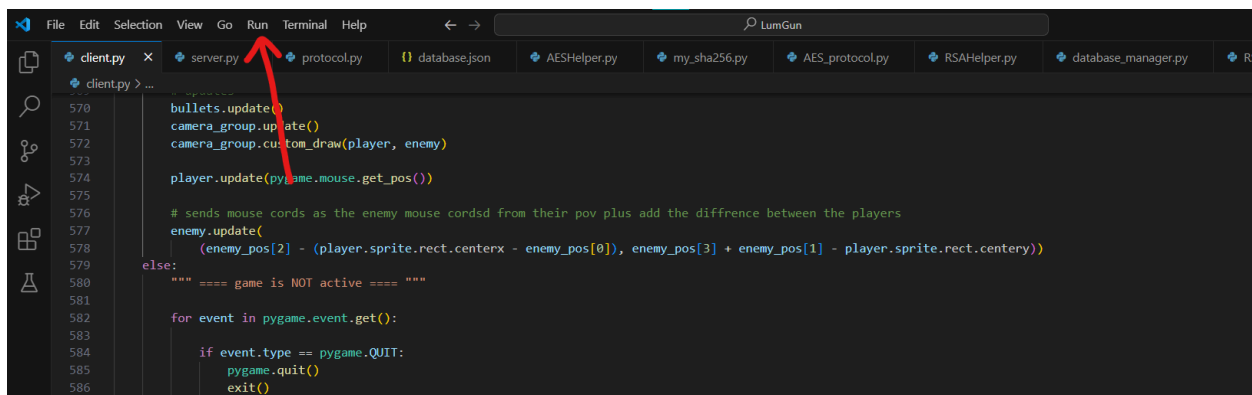
לאחר מכן, יפתח חלון המאפשר למשתמש להקליד סיסמה, ניתן ללחוץ על תיבת הטקסט ולהקליד. אם הוא אינו יודע את הסיסמה הוא אינו רשאי לפתוח שרת. עליו ללחוץ על מקש ה-ENTER כדי להזין את הסיסמה, אם היא נכונה יפתח השרת והודעה מתאימה תופיע ב-console. שהמשחק יהיה בפורמט קובץ מורץ, סיסמה זו תחסום פתיחה זדונית של שרת.



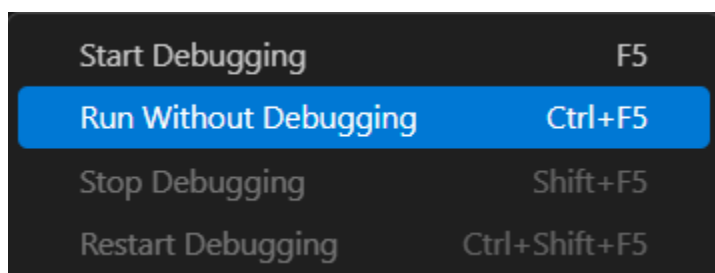
לאחר שהשרת נפתח, כל המידע הרלוונטי מוצג בהודעות ב-console.

לומגאן - משחק יריות מרובה משתמשים | ליאם ניב

במידה והמשתמש שפותח שרת רוצה גם לפתח לקוח, בזמן שהוא מסתכל על הקוד של הלקוח הוא יכול ללחוץ על "Run" בשמאל למעלה של המסך:



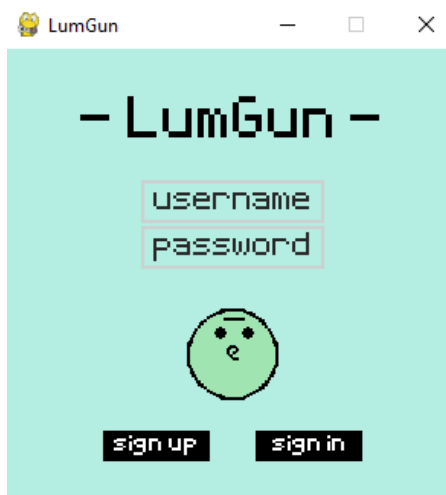
ולאחר מכן ללחוץ על "Run Without Debugging":



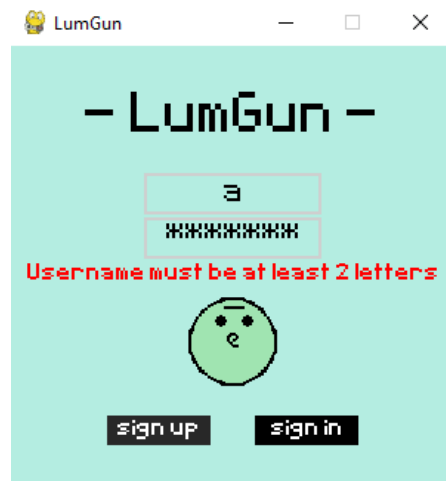
זוהי יאפשר לו לפתוח הרצה שנייה ומשם לעקוב אחר ההוראות ללקוח.

לקוח:

המשחק נפתח במסך הכניסה. במסך זה ניתן להזין שם משתמש וסיסמה, ובעזרת הכפתורים למטה או להתחבר למשתמש קיים או ליצור אחד חדש. לאחר שהמשתמש יזין פרטים תקינים הוא ייכנס מיידית למשחק.



אם פרטי המשתמש לא יהיו תקינים תוצג הודעה מתאימה, לדוגמה "Username must be at least 2 letters":



בעת החיבור למשחק, השחקן יכול לזוז במרחב עם המקשים:

W - קדימה

A - שמאלה

S - למטה

D - ימינה

ראש השחקן תמיד יכוון אל העכבר, ככה שהמשתמש שולט על המבט של הדמות בעזרת העכבר. מעל ראש השחקן את השם שלו ומדד חיים, כאשר הקו הירוק נגמר, השחקן מת. מול השחקן יוצב יריב, השחקן יכול לירות עליו קליעים בעזרת שני מקשי העבר (ימין ושמאל) במטרה לפגוע ולהרוג אותו.



לאחר ששחקן מת, הוא יחזור לנקודת ההתחלה שלו עם מד חיים מלא, מוכן לשחק שוב.

רפלקציה

תהליך העבודה:

העבודה על פרויקט זה עברה הרבה יותר בקלות מאשר דמיינתי. חשבתי שאצטרך להתגמש לכל הכיוונים ולרדוף אחר דרישות הפרויקט. בפועל, חווית כתיבה הקוד הייתה מאוד מהנה, למדתי דברים שלא תכננתי ללמוד וכל זה במטרה שהיא מעבר לפרוייקט: לראות משחק שדמיינתי חודשים רץ מול עיני ובצורה מאובטחת. אני שמח מאוד שבחרתי לאמץ משחק שיצרתי בעבר, כי זה הרגיש כאילו אני משיב לו חיים ויוצר את החזון שהיה לי. בנוסף, מירב האתגרים שלי בפיתוח הפרויקט התרחשו כאשר רק התחלתי לתכנת ב-python, וזה היה בתקופה שרק הכנתי את המשחק של הלקוח, לפני חודשים רבים וללא כוונה להפיכה לפרוייקט מלא. בגלל שרוב התקלות שקרו לי היו בתקופה שתכננתי רק בשביל הכיף, זה גרם לא להתבאס מטעויות ולנסות לפתור אותן כחלק מאתגר עצמי.

תהליך הלמידה:

את הלמידה שלי התחלתי עם המון סרטוני YouTube על ספריית pygame. פיתחתי משחק זהה לצד יוצר תוכן שאני מאוד אוהב, תוך כדי הכתיבה למדתי רבות על pygame וגם על python ככלל, למדתי את שני המרכיבים הללו בו זמנית. אחרי שהתחלתי לבנות את המשחק שלי, גיליתי שעליי לפתור המון בעיות שעדיין אין לי את הכלים לבצע, ולכן על כל שאלה שהייתה לי פניתי שוב ל-YouTube לסרטוני הסבר (דברים כמו מצלמה עוקבת אחרי שחקן, כפתורים, קופסאות טקסט ועוד). גם כאשר התחלתי לפתח שרת מרובה משתמשים, פניתי לסדרת סרטונים המלמדת על תכנות pygame מרובה משתמשים. את כל הסרטונים החשובים שאני זוכר שמהם למדתי אצרף לביבליוגרפיה. גם עבודה עם קובץ json והצפנת RSA למדתי מ-YouTube, בעזרת צפייה הדוקה בסרטונים לצד כתיבה עם נסייה וטעייה.

כלים להמשך:

אחרי שכתבתי את הפרויקט שלי, למדתי בעיקר על החשיבות של כתיבת קוד מסודרת ונאותה. זו הפעם הראשונה שאני עובד עם כמות קוד בסדר גודל שכזה, ולפעמים אם לא שמרתי על סדר כמו שצריך, הייתי יכול לחזור אחורה ואפילו לא להבין מה כתבתי. למדתי על החשיבות של מתן שמות משמעותיים למשתנים, תיעוד של מחלקות, פעולות ושורות חשובות. מעכשיו אשים לב יותר למידת הקריאות של הקוד שלי, כי אין דבר יותר מתסכל מלחזור על קוד שרשמת ולא להבין אותו.

תובנות מהתהליך:

למדתי גם על החשיבות של הכוונה. לדוגמה, למרות שלמדתי על הצפנה RSA לבד, אני לא הגעתי אליה לבד. המורה שלי למדעי המחשב, אמר לי לחקור על סוג הצפנה זה כדרך ליצירת מערכת מפתחות אסימטריים. למרות שבעידן של היום ניתן ללמוד על הכל לבד, חשוב גם לדעת על מה ללמוד, ולכן הכוונה מאדם עם יותר ניסיון היא מאוד חשובה.

יישום חלקים באופן שונה:

במבט לאחור, הייתי מיישם הרבה דברים אחרת בפרויקט שלי. דבר ראשון, הייתי מציע לעבוד על מסך יותר גדול מ-300 על 300. אני עדיין הייתי רוצה לשמר את האסתטיקה של אומנות הפיקסלים, ולכן במבט לאחור הייתי מציע מסך של 400 על 400. בנוסף, אחת מהטעויות הגדולות ביותר שעשיתי היא במקום לשלוח לשרת את זווית הסיבוב של השחקן, בחרתי לשלוח את מיקום העכבר שלו ושהלקוח השני יחשב אותה. חוסר חשיבה זה עלה לי בכמה נסיונות של תיקון מערכת הסיבוב ובחוסר יעילות שליחת נתונים.

שיפור בעזרת משאבים נוספים:

- במידה והייתי מוסיף עוד דברים לפרויקט שלי, והייתי יכול לדעת איך ליישם אותם, הייתי מוסיף: מערכת ניקוד שהמשתמשים רואים, הם מתחרים להגיע למספר הריגות מסויים.
- פרסים ועיצוב שונה לדמות שחקן כתגמול למשחק.
- מערכת המאפשרת לפתוח חדרי משחק ולנהל הרבה משחקים בו זמנית.
- יותר מפות.
- צ'אט דיבור עם מיקרופונים.

תודות:

אני רוצה להודות למנחה שלי, יהודה אור. הוא מהשיעור הראשון שלנו נקט במדיניות שאם מישהו לא מבין משהו בשיעור, עליו לא להפסיק לשאול שאלות עד להבנה מוחלטת. ובמקרים רבים אני הייתי האדם שלא הבין, והוא תמיד עזר וידא שאני לא אאבד בחומר של השיעור. יהודה עזר לי מאוד להבין את הכיוון שלי בפרויקט ולאן לשאוף בשעת התכנון, הוא נתן לי את ההכוונה שהייתי צריך כדי ללמוד על נושאים נכונים.

ביבליוגרפיה

References

- Chris. [Clear Code]. (2021, July 11). *The ultimate introduction to Pygame* [video]. YouTube. <https://youtu.be/AY9MnQ4x3zk>
- Chris. [Clear Code]. (2022, February 12). *Cameras in Pygame* [video]. YouTube. <https://youtu.be/u7LPRqrzry8>
- Chris. [Clear Code]. (2020, May 14). *Python / Pygame tutorial: Getting text input* [video]. YouTube. <https://youtu.be/Rvcyf4HsWiw>
- Mattingly, W. J. B. [Python Tutorials for Digital Humanities]. *What are JSON Files and How do we use them in Python (Python and JSON Tutorial 01)* [video]. YouTube. https://youtu.be/B8AvyqCBdE?list=PL2VXyKi-KpYs_f1gu30AGqy0H6x97Vomf
- Mattingly, W. J. B. [Python Tutorials for Digital Humanities]. *How to Read a JSON file in Python (Python and JSON Tutorial 02)* [video]. YouTube. https://youtu.be/rNVII7NRA0k?list=PL2VXyKi-KpYs_f1gu30AGqy0H6x97Vomf
- Mattingly, W. J. B. [Python Tutorials for Digital Humanities]. *How to Create a JSON file in Python (Python and JSON Tutorial 03)* [video]. YouTube. https://youtu.be/uGzZuVaUvdU?list=PL2VXyKi-KpYs_f1gu30AGqy0H6x97Vomf
- Mattingly, W. J. B. [Python Tutorials for Digital Humanities]. *How to Append JSON files in Python* [video]. YouTube. https://youtu.be/QRcZmDaO_I?list=PL2VXyKi-KpYs_f1gu30AGqy0H6x97Vomf
- Crawford, T. [Tom Rocks Maths]. *How does RSA Cryptography work?* [video]. YouTube. <https://youtu.be/qph77bTKJTM>
- Dedov, F. [NeuralNine]. *RSA Encryption From Scratch - Math & Python Code* [video]. YouTube. https://youtu.be/D_PfV_IcUdA
- Yu, B. [Spanning Tree]. *AES: How to Design Secure Encryption* [video]. YouTube. <https://youtu.be/C4ATDMlz5wc>
- Ventures, M. [Matthew Ventures]. *AES Encryption: What's the difference between the IV and Key? Why do we need an IV?* [video]. YouTube. <https://youtu.be/uWEPEBmFBHw>

הקוד -

כולו, וכל ההערות בו נכתבו על ידי (כתבתי הרבה הערות)

AES_protocol.py

```
1  """
2  this file is contains the functions that enable sending out
3  encrypted AES ciphertext along with the iv
4
5  when sending information from or to the server, not related to the handshake
6  these functions are used. the ciphertext that is connected to the iv is
7  send as bytes to the server/client, when then its separated for decryption
8  use
9  """
10
11 def make_cipheriv(data):
12     """forms the sent combination of both the ciphertext and the iv
13
14     Args:
15         data (tuple): ciphertext, iv
16
17     Returns:
18         bytes: iv attached to the end of the ciphertext
19     """
20     return data[0] + data[1]
21
22
23 def read_cipheriv(data):
24     """splits the received combination of the ciphertext and the iv
25
26     Args:
27         data (bytes): long string of bytes with the ciphertext and the iv at
the end
28
29     Returns:
30         tuple: containing the ciphertext and the iv, this tuple will be
unpacked for decryption
31     """
32     # ciphertext being first until the last 16 bytes
33     ciphertext = data[:-16]
34     # iv being the last 16 bytes
35     iv = data[-16:]
36     return ciphertext, iv
37
```

AESHelper.py

```
1  from Crypto.Cipher import AES
2  from Crypto.Random import get_random_bytes
3  from Crypto.Util.Padding import pad, unpad
4
5
6  class AESHelper:
7      """Class made for AES encryption and decryption for both
8      the client and server
9      """
10
11     def __init__(self, key=None):
12         """making an AES-128 key, can either be generated
13         using random 16 bytes or a parameter
14
15         Args:
16             key (bytes, optional): AES-128 key. Defaults to None.
17         """
18         if key is None:
19             # 16 bytes for AES-128
20             self.key = get_random_bytes(16)
21         else:
22             self.key = key
23
24     def get_key(self):
25         """getting the AES key
26
27         Returns:
28             bytes: the AES key
29         """
30         return self.key
31
32     def aes_encrypt(self, plaintext):
33         """encrypts using AES
34
35         Args:
36             plaintext (string): the string being encrypts
37
38         Returns:
39             tuple: the encrypted text (ciphertext, bytes) and the
40             initialization vector (bytes)
41         """
42         # setting the unique initialization vector
43         iv = get_random_bytes(16)
44
45         # make new AES cipher object for encryption
46         # using Cipher Block Chaining (CBC)
47         cipher = AES.new(self.key, AES.MODE_CBC, iv)
48
49         # pad plaintext to be a multiple of the block size
50         padded_plaintext = pad(plaintext.encode(), AES.block_size)
```

```

50
51     # encrypt the padded plaintext
52     ciphertext = cipher.encrypt(padded_plaintext)
53
54     return ciphertext, iv
55
56 def aes_decrypt(self, ciphertext, iv):
57     """decrypts using AES
58
59     Args:
60         ciphertext (bytes): the encrypted info
61         iv (bytes): initialization vector
62
63     Returns:
64         string: the decrypted string
65     """
66     # make new AES cipher object for decryption
67     cipher = AES.new(self.key, AES.MODE_CBC, iv)
68
69     # decrypt ciphertext
70     decrypted_padded_data = cipher.decrypt(ciphertext)
71
72     # unpad decrypted data to get the original plaintext
73     decrypted_data = unpad(decrypted_padded_data, AES.block_size)
74
75     return decrypted_data.decode()
76

```

client.py

```
1 import math
2 from sys import exit
3 import pygame
4 from network import Network
5 from protocol import *
6 from AESHelper import AESHelper
7 from RSAHelper_client import RSAHelper_client
8 from AES_protocol import *
9 import my_sha256
10
11
12 class Circle(pygame.sprite.Sprite):
13     """The big circle obstacle in the middle of the map
14
15     Args:
16         pygame (class): the pygame sprite class
17     """
18
19     def __init__(self, pos, group):
20         super().__init__(group)
21         self.image = pygame.image.load('graphics/Circle.png').convert_alpha()
22         self.rect = self.image.get_rect(center=pos)
23         self.mask = pygame.mask.from_surface(self.image)
24
25
26 class Slash(pygame.sprite.Sprite):
27     """The forward facing slashes obstacles around the map
28
29     Args:
30         pygame (class): the pygame sprite class
31     """
32
33     def __init__(self, pos, group):
34         super().__init__(group)
35         self.image = pygame.image.load('graphics/Slash.png').convert_alpha()
36         self.rect = self.image.get_rect(center=pos)
37         self.mask = pygame.mask.from_surface(self.image)
38
39
40 class Backslash(pygame.sprite.Sprite):
41     """The backwards facing slashes obstacles around the map
42
43     Args:
44         pygame (class): the pygame sprite class
45     """
46
47     def __init__(self, pos, group):
48         super().__init__(group)
49         self.image = pygame.image.load(
```

```

50         'graphics/Backslash.png').convert_alpha()
51     self.rect = self.image.get_rect(center=pos)
52     self.mask = pygame.mask.from_surface(self.image)
53
54
55 class Player(pygame.sprite.Sprite):
56     """a player, can move around the map and shoot,
57     either the main player or the enemy
58
59     Args:
60         pygame (class): the pygame sprite class
61     """
62
63     def __init__(self, pos, isUser, username=''):
64
65         super().__init__()
66
67         self.isUser = isUser
68
69         # decides what skin to give player based on if hes an enemy or not
70         if isUser:
71             self.original_image = pygame.image.load(
72                 'graphics/Player.png').convert_alpha()
73         else:
74             self.original_image = pygame.image.load(
75                 'graphics/Enemy.png').convert_alpha()
76
77         # setting up basic parameters for the player
78         self.image = self.original_image
79         self.rect = self.image.get_rect(center=pos)
80         self.direction = pygame.math.Vector2()
81         self.speed = 2
82         self.angle = 0
83         self.mask = pygame.mask.from_surface(self.original_image)
84
85         # health bar- gray
86         self.health_bar_background = pygame.Surface((30, 4))
87         self.health_bar_background.fill((199, 199, 199))
88         # actual health variable
89         self.health = 30
90         self.health_bar_health = pygame.Surface((self.health, 4))
91         self.health_bar_health.fill((0, 255, 0))
92         self.health_bar_rect = self.health_bar_background.get_rect(
93             center=(self.rect.centerx, self.rect.centery - 22))
94
95         # username
96         self.username = username
97         self.username_font = pygame.font.Font('font/Pixeltype.ttf', 18)
98         self.username_surf = self.username_font.render(
99             self.username, False, 'Black')
100         self.username_rect = self.username_surf.get_rect(

```

```

101         center=(self.rect.centerx, self.rect.centery - 30))
102
103     def set_username(self, username):
104         """sets the username to be what is requested
105
106         Args:
107             username (string): the new username
108         """
109         self.username = username
110
111     def face_mouse(self, mouse_pos):
112         """calculates the difference between the players position and the mc
113         and applies the rotation needed for the player to face the mouse.
114
115         Args:
116             mouse_pos (tuple): containing the mouse x and y position
117         """
118
119         player_pos = self.rect.center
120         # distance between the player and the mouse
121         x_dist = mouse_pos[0] - player_pos[0] + camera_group.offset[0]
122         y_dist = -(mouse_pos[1] - player_pos[1] + camera_group.offset[1])
123         # updating the angle of the player, using tan
124         self.angle = math.degrees(math.atan2(y_dist, x_dist))
125         # rotating the image
126         self.image = pygame.transform.rotate(
127             self.original_image, self.angle - 90)
128         self.rect = self.image.get_rect(center=(player_pos[0], player_pos[1])
129         self.mask = pygame.mask.from_surface(self.image)
130
131     def create_bullet(self):
132         """shoots a bullet from the player
133
134         Args:
135             mouse_pos (tuple): the position of the mouse, pass to the bullet
136
137         Returns:
138             Bullet: the bullet being fired
139         """
140         return Bullet(self.rect.center, self)
141
142     def input(self):
143         """is in charge of changing the players direction,
144         which is a 2d vector attribute of the player
145         """
146
147         keys = pygame.key.get_pressed()
148
149         if keys[pygame.K_w] and keys[pygame.K_s]:
150             self.direction.x = 0
151         elif keys[pygame.K_w]:

```

```

152         self.direction.y = -1
153     elif keys[pygame.K_s]:
154         self.direction.y = 1
155     else:
156         self.direction.y = 0
157
158     if keys[pygame.K_d] and keys[pygame.K_a]:
159         self.direction.x = 0
160     elif keys[pygame.K_d]:
161         self.direction.x = 1
162     elif keys[pygame.K_a]:
163         self.direction.x = -1
164     else:
165         self.direction.x = 0
166
167     def border_collision(self):
168         """in charge making sure the player doesn't exist the fighting arena
169         by detecting if he is about to head outside and quickly change his p
170         """
171         # half is the player's radius, so the method can detect if the playe
172         # touching the border with his outline.
173         half = 15
174         if self.rect.center[0] < 120 + half:
175             self.rect.center = (120 + half, self.rect.center[1])
176         if self.rect.center[0] > 880 - half:
177             self.rect.center = (880 - half, self.rect.center[1])
178         if self.rect.center[1] < 120 + half:
179             self.rect.center = (self.rect.center[0], 120 + half)
180         if self.rect.center[1] > 380 - half:
181             self.rect.center = (self.rect.center[0], 380 - half)
182
183     def obstacle_collision(self):
184         """handles the player colliding with the obstacles
185         in the map, simply pushing them the opposite way when they
186         try to move into them to simulate a wall
187         """
188         keys = pygame.key.get_pressed()
189         if pygame.sprite.spritecollide(self, camera_group, False, pygame.spr
190             if keys[pygame.K_w]:
191                 self.rect.centery += 2
192             elif keys[pygame.K_s]:
193                 self.rect.centery -= 2
194
195             if keys[pygame.K_d]:
196                 self.rect.centerx -= 2
197             elif keys[pygame.K_a]:
198                 self.rect.centerx += 2
199
200     def bullet_collision(self, bullet_group):
201         """checks for collisions of a bullet with a player,
202         kills the bullet (third parameter in if) and changes the health

```

```

203
204     Args:
205         bullet_group (Group): the sprite group containing all of the bul
206     """
207     # makes sure the player being hit is only the main one, enemy health
208     if pygame.sprite.spritecollide(self, bullet_group, True, pygame.spr
209         if self.health != 0:
210             self.health -= 1
211
212 def die(self):
213     """in charge of sending the info to the server when a player dies
214     can only be activated if the main player, not enemy dies, if the ene
215     he will trigger this method in his own code and the info will pass t
216     """
217
218     # sending the sever a message meaning death
219     # and receiving the cords to re-spawn
220     spawn_pos = read(aes_helper.aes_decrypt(
221         *read_cipheriv(n.send(make_cipheriv(aes_helper.aes_encrypt('5'))
222
223     self.rect.centerx = spawn_pos[0]
224     self.rect.centery = spawn_pos[1]
225     self.health = spawn_pos[5]
226
227 def update(self, mouse_pos):
228     """updates the player using all of the necessary methods
229
230     Args:
231         mouse_pos (tuple): the mouse position
232     """
233
234     self.input()
235     self.rect.center += self.direction * self.speed
236     self.face_mouse(mouse_pos)
237     self.border_collision()
238     self.obstacle_collision()
239     self.bullet_collision(bullets)
240
241     if self.health == 0 and self.isUser:
242         self.die()
243
244     # health bar
245     self.health_bar_health = pygame.Surface((self.health, 4))
246     self.health_bar_health.fill((0, 255, 0))
247
248     # username
249     self.username_surf = self.username_font.render(
250         self.username, False, 'Black')
251     self.username_rect = self.username_surf.get_rect(
252         center=(self.rect.centerx, self.rect.centery - 32))
253

```



```

254
255 class Bullet(pygame.sprite.Sprite):
256     """class for a single bullet
257
258     Args:
259         pygame (class): pygame sprite class
260     """
261
262     def __init__(self, pos, player):
263         super().__init__()
264         # loading the image before its turned to the face the trajectory
265         self.original_image = pygame.image.load('graphics/Bullet.png')
266
267         self.rect = self.original_image.get_rect(center=(pos))
268
269         self.angle = player.angle
270
271         self.magnitude = 10
272
273         # rotates the image
274         self.image = pygame.transform.rotate(
275             self.original_image, self.angle - 90)
276
277         # positions the bullet right outside the player body, so it doesn't
278         self.rect.center += pygame.Vector2(self.magnitude * math.cos(math.radians(self.angle)),
279                                             -self.magnitude * math.sin(math.radians(self.angle)))
280
281         # creates mask
282         self.mask = pygame.mask.from_surface(self.image)
283
284     def collision_sprite(self):
285         """checks for collision with the edges of the maps or the map's obstacles
286
287         Returns:
288             bool: boolean statement for "is the bullet hitting something?"
289         """
290         if pygame.sprite.spritecollide(self, camera_group, False, pygame.sprite.collide_mask):
291             return True
292         if self.rect.centerx < 124 or self.rect.centerx > 876 or self.rect.centery < 50 or self.rect.centery > 1000:
293             return True
294         return False
295
296     def update(self):
297         """updates the bullet, moving it forward and killing it if it touches an obstacle
298
299         # bullet travel
300         self.rect.center += pygame.Vector2(self.magnitude * math.cos(math.radians(self.angle)),
301                                             -self.magnitude * math.sin(math.radians(self.angle)))
302         # if bullet collides with obstacle it self kills
303         if self.collision_sprite():
304             self.kill()

```

```

305
306
307 class CameraGroup(pygame.sprite.Group):
308     """sprite group for all of the obstacles around the map
309     ground also in it but not as a sprite
310     makes the illusion of the camera following the player
311     using a custom blitting function with an offset
312
313
314     Args:
315         pygame (class): the python group class
316     """
317
318     def __init__(self):
319         super().__init__()
320         self.display_surface = pygame.display.get_surface()
321
322         # camera offset
323         # serves to keep the player's view on himself
324         self.offset = pygame.math.Vector2()
325         self.half_w = self.display_surface.get_size()[0] // 2
326         self.half_h = self.display_surface.get_size()[1] // 2
327
328         # ground
329         self.ground_surf = pygame.image.load(
330             'graphics/Background_new.png').convert_alpha()
331         self.ground_rect = self.ground_surf.get_rect(topleft=(0, 0))
332
333     def center_target_camera(self, target):
334         """in charge of updating the camera offset,
335         in which objects around the map need to be offset by
336         in order to look like they are the ones moving around the player.
337
338         Args:
339             target (Player): the main player
340         """
341
342         self.offset.x = target.sprite.rect.centerx - self.half_w
343         self.offset.y = target.sprite.rect.centery - self.half_h
344
345     def custom_draw(self, player, enemy):
346         """
347         this method uses the offset in order to create the illusion of
348         a camera following the player around, by moving everything around hi
349         while blitting them into the screen, uses the player as the main "t
350         """
351
352         # calculating the offset (players displacement)
353         self.center_target_camera(player)
354
355         # blitting ground

```

```

356     ground_offset = self.ground_rect.topleft - self.offset
357     self.display_surface.blit(self.ground_surf, ground_offset)
358
359     # blitting all of the bullets in the bullets group
360     for sprite in bullets.sprites():
361         bullet_offset_pos = sprite.rect.topleft - self.offset
362         self.display_surface.blit(sprite.image, bullet_offset_pos)
363
364     # blitting player
365     player_offset = player.sprite.rect.topleft - self.offset
366     self.display_surface.blit(player.sprite.image, player_offset)
367
368     # blitting enemy
369     enemy_offset = enemy.sprite.rect.topleft - self.offset
370     self.display_surface.blit(enemy.sprite.image, enemy_offset)
371
372     # blitting all of the obstacles on the map
373     for sprite in self.sprites():
374         offset_pos = sprite.rect.topleft - self.offset
375         self.display_surface.blit(sprite.image, offset_pos)
376
377     # blitting player health
378     player_center_offset = player.sprite.rect.center - self.offset
379     self.display_surface.blit(player.sprite.health_bar_background,
380                             (player_center_offset[0] - 15, player_cent
381     self.display_surface.blit(player.sprite.health_bar_health,
382                             (player_center_offset[0] - 15, player_cent
383
384     # blitting enemy health
385     enemy_center_offset = enemy.sprite.rect.center - self.offset
386     self.display_surface.blit(enemy.sprite.health_bar_background,
387                             (enemy_center_offset[0] - 15, enemy_center
388     self.display_surface.blit(enemy.sprite.health_bar_health,
389                             (enemy_center_offset[0] - 15, enemy_center
390
391     # blitting player username
392     self.display_surface.blit(player.sprite.username_surf, (
393         player_center_offset[0] - player.sprite.username_rect.w//2, play
394
395     # blitting enemy username
396     self.display_surface.blit(enemy.sprite.username_surf, (
397         enemy_center_offset[0] - enemy.sprite.username_rect.w//2, enemy_
398
399
400 class Button(pygame.sprite.Sprite):
401     """button class for the homepage "sign up / in" buttons
402     doesn't handle actually being pressed, changes color if
403     mouse is hovering over it
404
405     Args:
406         pygame (class): the pygame sprite class

```

```

407     """
408
409     def __init__(self, x, y, width, height, text, button_color_passive=(0, 0, 0),
410 text_color=(255, 255, 255), text_size=25):
411         super().__init__()
412         # setting up the square
413         self.button_rect = pygame.rect.Rect(0, 0, width, height)
414         self.button_rect.center = (x, y)
415
416         # colors for rect
417
418         self.color_passive = button_color_passive
419         self.color_active = button_color_active
420         self.color = color_passive
421
422         # setting up the text
423         button_text_font = pygame.font.Font('font/Pixeltype.ttf', text_size)
424         self.button_text_surf = button_text_font.render(
425             text, False, text_color)
426         self.button_text_rect = self.button_text_surf.get_rect(
427             center=self.button_rect.center)
428
429     def update(self):
430         """if the mouse is on top of the button, it changes the current color"""
431         if self.button_rect.collidepoint(pygame.mouse.get_pos()):
432             self.color = self.color_active
433         else:
434             self.color = self.color_passive
435
436     def draw(self):
437         """drawing the button on screen using its current color"""
438
439         pygame.draw.rect(screen, self.color, self.button_rect)
440         screen.blit(self.button_text_surf, self.button_text_rect)
441
442
443 # object with methods including basic communication with the server
444 n = Network()
445
446
447 pygame.init()
448 screen = pygame.display.set_mode((300, 300)) # screen size
449 pygame.display.set_caption('LumGun') # give window title
450 clock = pygame.time.Clock() # setting clock
451
452 # bool for game activeness
453 game_active = False
454
455 # initializing sprites in camera group
456 camera_group = CameraGroup()

```

```

457 b1 = Backslash((320, 320), camera_group)
458 b2 = Backslash((680, 180), camera_group)
459 s1 = Slash((600, 300), camera_group)
460 s2 = Slash((400, 200), camera_group)
461 c1 = Circle((500, 250), camera_group)
462
463 # initializing player
464 player = pygame.sprite.GroupSingle()
465 start_pos1 = read(n.getPos())
466 player.add(Player((start_pos1[0], start_pos1[1]), True))
467 click = False
468
469 # initializing enemy
470 enemy = pygame.sprite.GroupSingle()
471 start_pos2 = read(n.getPos())
472 enemy.add(Player((start_pos2[0], start_pos2[1]), False))
473
474 # initializing bullet group
475 bullets = pygame.sprite.Group()
476
477
478 """ ==== menu screen ==== """
479
480
481 # static elements - text
482 # fonts
483 pixel_font_small = pygame.font.Font('font/Pixeltype.ttf', 25)
484 pixel_font_mid = pygame.font.Font('font/Pixeltype.ttf', 35)
485 pixel_font_big = pygame.font.Font('font/Pixeltype.ttf', 60)
486
487 # big "LumGun" title
488 menu_text_top_surf = pixel_font_big.render("- LumGun -", False, 'Black')
489 menu_text_top_rect = menu_text_top_surf.get_rect(center=(150, 50))
490
491 # static elements - player pic
492 menu_player_surf = pygame.image.load('graphics/Player.png').convert_alpha()
493 menu_player_surf = pygame.transform.scale2x(menu_player_surf)
494 menu_player_rect = menu_player_surf.get_rect(center=(150, 200))
495
496
497 """ username """
498
499 username_box_height = 100
500
501 # the "username" text that shows up if text box is passive and empty
502 username_placeholder_surf = pixel_font_mid.render(
503     "username", False, (40, 40, 40))
504 username_placeholder_rect = username_placeholder_surf.get_rect(
505     center=(150, username_box_height))
506
507 # var that stores the username

```

```

508 username = ''
509
510 # the visible border around the username text box
511 username_border_rect = pygame.Rect(0, 0, 120, 28)
512 username_border_rect.center = (150, username_box_height)
513
514 # colors for the username text box border depending on activity
515 color_active = pygame.Color(240, 240, 240)
516 color_passive = pygame.Color(207, 207, 207)
517 username_border_rect_color = color_passive
518
519 # bool for activeness of username text box
520 username_box_active = False
521
522
523 """ password """
524
525 password_box_height = 130
526
527 # the "password" text that shows up if text box is passive and empty
528 password_placeholder_surf = pixel_font_mid.render(
529     "password", False, (40, 40, 40))
530 password_placeholder_rect = password_placeholder_surf.get_rect(
531     center=(150, password_box_height))
532
533 # var that stores the username
534 password = ''
535 censored_password = ''
536
537 # the visible border around the username text box
538 password_border_rect = pygame.Rect(0, 0, 120, 28)
539 password_border_rect.center = (150, password_box_height)
540
541 # colors for the password text box border depending on activity
542 password_border_rect_color = color_passive
543
544 # bool for activeness of username text box
545 password_box_active = False
546
547
548 """ error message """
549
550 error_height = 155
551
552 msg = ""
553 show_error = False
554
555 # time from the start of program until error msg needs to be presented
556 start_time = pygame.time.get_ticks()
557
558 # tim from the start of program until current time (when msg is up)

```

```

559 current_time = pygame.time.get_ticks()
560
561 # the time the msg is presented, when it gets too big, the msg hides
562 delta_time = start_time - current_time
563
564
565 """ buttons """
566
567
568 sign_in_button = pygame.sprite.GroupSingle()
569 sign_in_button.add(Button(200, 260, 70, 20, 'sign in'))
570
571 sign_up_button = pygame.sprite.GroupSingle()
572 sign_up_button.add(Button(100, 260, 70, 20, 'sign up'))
573
574
575 """ RSA HANDSHAKE """
576
577 print('=== client ===')
578 # contains methods to communicate using aes
579 aes_helper = AESHelper()
580
581 # sending request for the public key
582 print('1. sending \'6\' to get public key')
583 public_key = read(n.send_before('6'))
584
585 print('4. got the public key: ' + make_public_key(public_key)[1:])
586 # making encryptor using the public key
587 rsa_helper_client = RSAHelper_client(public_key)
588
589 print('5. AES key: ' + str(aes_helper.get_key()))
590
591 # making an encrypted aes key
592 encrypted_aes_key = rsa_helper_client.rsa_encrypt(aes_helper.get_key())
593 print('6. made encrypted AES key, sending it: ' + str(encrypted_aes_key))
594
595 # sending the encrypted aes key
596 n.send_no_answer(make_encrypted_key(str(encrypted_aes_key)))
597
598
599 # game loop
600 while True:
601     if (game_active):
602         """ ==== game is active ==== """
603
604         # bool var for shooting
605         click = False
606
607         # for loop for checking for input to shut program or to shoot
608         for event in pygame.event.get():
609             if event.type == pygame.QUIT:

```

```

610         pygame.quit()
611         exit()
612
613     if event.type == pygame.MOUSEBUTTONDOWN:
614         bullets.add(player.sprite.create_bullet())
615         click = True
616     else:
617         click = False
618
619     # sending and receiving info from/to enemy
620     # cant send if the player is dead
621     if player.sprite.health != 0:
622         enemy_pos =
623         read(aes_helper.aes_decrypt(*read_cipheriv(n.send(make_cipheriv(aes_helper.a
624         player.sprite.rect.centery,
625         pygame.mouse.get_pos()[0], pygame.mouse.get_pos()[
626         1],
627         click, player.sprite.health))))))
628
629     # updating enemy's position and health
630     enemy.sprite.rect.centerx = enemy_pos[0]
631     enemy.sprite.rect.centery = enemy_pos[1]
632     enemy.sprite.health = enemy_pos[5]
633
634     # if information is delivered about an enemy's name, save it
635     if len(enemy_pos) > 6:
636         enemy.sprite.username = enemy_pos[6]
637
638     # general color to fill the screen
639     screen.fill('#F7D9B4')
640
641     # check whether to add bullets depending on shooting status of enemy
642     if enemy_pos[4]:
643         bullets.add(enemy.sprite.create_bullet())
644
645     # updates
646     bullets.update()
647     camera_group.update()
648     camera_group.custom_draw(player, enemy)
649
650     player.update(pygame.mouse.get_pos())
651
652     # sends mouse cords as the enemy mouse cords from their pov plus adc
653     enemy.update(
654         (enemy_pos[2] - (player.sprite.rect.centerx - enemy_pos[0]), ene
655         player.sprite.rect.centery))
656
657     else:
658         """ ==== game is NOT active ==== """

```



```

656     for event in pygame.event.get():
657
658         if event.type == pygame.QUIT:
659             pygame.quit()
660             exit()
661
662         # if there is a click it decides which textbox is active
663         if event.type == pygame.MOUSEBUTTONDOWN:
664             # changes the state of text boxes
665             username_box_active = username_border_rect.collidepoint(
666                 event.pos)
667             password_box_active = password_border_rect.collidepoint(
668                 event.pos)
669
670             # if the sign in button was clicked
671             if sign_in_button.sprite.button_rect.collidepoint(event.pos)
672                 # than it would send the details to the server and check
673                 if
674 read(aes_helper.aes_decrypt(*read_cipheriv(n.send(make_cipheriv(aes_helper.a
675 my_sha256.hash_string(password), True)))))):
676                 # changes game state to on because details got appr
677                 game_active = True
678                 enemy_pos =
679 read(aes_helper.aes_decrypt(*read_cipheriv(n.send(make_cipheriv(aes_helper.a
680 player.sprite.rect.centery,
681 pygame.mouse.get_pos()[0], pygame.mouse.get_pos()[
682 1],
683 click, player.sprite.health))))))
684                 player.sprite.username = username
685                 # if the details are not good
686                 else:
687                     msg = 'Details don\'t match with any users'
688                     show_error = True
689                     start_time = pygame.time.get_ticks()
690                 # if the sign up button was clicked
691                 if sign_up_button.sprite.button_rect.collidepoint(event.pos)
692
693                     # checking for min username and password length, locally
694                     if len(username) <= 1:
695                         msg = 'Username must be at least 2 letters'
696                         show_error = True
697                         start_time = pygame.time.get_ticks()
698                     elif len(password) <= 5:
699                         msg = 'Password must be at least 6 letters'
700                         show_error = True
701                         start_time = pygame.time.get_ticks()
702                 # checking for servers response
703                 # if the sign up request is positive
704                 elif
705 read(aes_helper.aes_decrypt(*read_cipheriv(n.send(make_cipheriv(aes_helper.a

```

```

my_sha256.hash_string(password), False))))))):
701         # changes game state to on because details got apprc
702         game_active = True
703         enemy_pos =
read(aes_helper.aes_decrypt(*read_cipheriv(n.send(make_cipheriv(aes_helper.a
player.sprite.rect.centery,
704 pygame.mouse.get_pos()[0], pygame.mouse.get_pos()[
705 1],
706 click, player.sprite.health)))))))))
707         player.sprite.username = username
708         # sign up request is negative
709         else:
710             # change the msg
711             msg = 'Username already taken'
712             # starting the 3000 millisecond timer to show msg
713             show_error = True
714             start_time = pygame.time.get_ticks()
715
716     if event.type == pygame.KEYDOWN:
717         # makes sure enter doesn't count
718         if event.key != pygame.K_RETURN:
719
720             if username_box_active:
721                 # if the backspace is pressed
722                 if event.key == pygame.K_BACKSPACE:
723                     # than the last char in username is deleted
724                     username = username[0:-1]
725                 # limits username length to 12
726                 elif len(username) <= 12:
727                     allowed_characters = "abcdefghijklmnopqrstuvwxyz
728                     # check if the character is in the allowed set
729                     if event.unicode in allowed_characters:
730                         username += event.unicode
731             elif password_box_active:
732                 # if the backspace is pressed
733                 if event.key == pygame.K_BACKSPACE:
734                     # than the last char in password is deleted
735                     password = password[0:-1]
736                     censored_password = censored_password[0:-1]
737                 # limits username length to 18
738                 elif len(password) <= 18:
739                     allowed_characters = "abcdefghijklmnopqrstuvwxyz
740                     # check if the character is in the allowed set
741                     if event.unicode in allowed_characters:
742                         password += event.unicode
743                         censored_password = '*' * len(password)
744
745     # static menu elements
746     screen.fill('#b4ede1')

```

```

747 screen.blit(menu_text_top_surf, menu_text_top_rect)
748 screen.blit(menu_player_surf, menu_player_rect)
749
750 """ username textbox """
751
752 # changes the color of the username text box outline based on active
753 if username_box_active:
754     username_border_rect_color = color_active
755 else:
756     username_border_rect_color = color_passive
757
758 # draw the username border rect
759 pygame.draw.rect(screen, username_border_rect_color,
760                  username_border_rect, 2)
761
762 # the surface on which the username is written on
763 username_surf = pixel_font_mid.render(username, False, 'Black')
764 # the invisible rect around that username
765 username_rect = username_surf.get_rect(
766     center=(150, username_box_height))
767
768 # changes the width of the username text box outline based on text l
769 username_border_rect.w = max(username_rect.w + 8, 120)
770
771 # makes sure the username border rect is in the center
772 username_border_rect.center = (150, username_box_height)
773
774 # decides whether "username" text placeholder shows or actual usern
775 if len(username) > 0 or username_box_active:
776     screen.blit(username_surf, username_rect)
777 else:
778     screen.blit(username_placeholder_surf, username_placeholder_rect)
779
780 """ password textbox """
781
782 # changes the color of the password text box outline based on active
783 if password_box_active:
784     password_border_rect_color = color_active
785 else:
786     password_border_rect_color = color_passive
787
788 # draw the password border rect
789 pygame.draw.rect(screen, password_border_rect_color,
790                  password_border_rect, 2)
791
792 # the surface on which the password is written on
793 password_surf = pixel_font_mid.render(
794     censored_password, False, 'Black')
795 # the invisible rect around that password
796 password_rect = password_surf.get_rect(
797     center=(150, password_box_height))

```

```

798
799     # changes the width of the password text box outline based on text l
800     password_border_rect.w = max(password_rect.w + 8, 120)
801
802     # makes sure the password border rect is in the center
803     password_border_rect.center = (150, password_box_height)
804
805     # decides whether "password" text placeholder shows or actual passw
806     if len(password) > 0 or password_box_active:
807         screen.blit(password_surf, password_rect)
808     else:
809         screen.blit(password_placeholder_surf, password_placeholder_rect
810
811     """ error msg """
812
813     if show_error:
814         error_surf = pixel_font_small.render(msg, False, 'Red')
815         error_rect = error_surf.get_rect(center=(150, error_height))
816         current_time = pygame.time.get_ticks()
817         delta_time = current_time - start_time
818
819         # makes sure the error displays for only 3000 milliseconds
820         if delta_time < 3000:
821             screen.blit(error_surf, error_rect)
822         else:
823             show_error = False
824             msg = ''
825
826     """ buttons """
827
828     sign_in_button.sprite.update()
829     sign_in_button.sprite.draw()
830
831     sign_up_button.sprite.update()
832     sign_up_button.sprite.draw()
833
834     # draw all our elements
835     # update everything
836     pygame.display.update()
837     clock.tick(60) # 60 tick per second
838

```

database_manager.py

```
1 import json
2
3
4 def isValid(username, password):
5     """checks the validity of a username a password
6
7     Args:
8         username (string): inputted username
9         password (string): inputted password
10
11     Returns:
12         bool: boolean for "are these details saved in the database?"
13     """
14     with open("database.json") as f:
15         data = json.load(f)
16         user_list = data["users"]
17         for user in user_list:
18             if user["username"] == username and user["password"] ==
password:
19                 return True
20         return False
21
22
23 def isUnique(username):
24     """checks if a username is already in the database to check if
25     it would be unique for a player trying to sign up
26
27     Args:
28         username (string): the username being checked
29
30     Returns:
31         bool: boolean for "will this username be unique?"
32     """
33     with open("database.json") as f:
34         data = json.load(f)
35         user_list = data["users"]
36         for user in user_list:
37             if user["username"] == username:
38                 return False
39         return True
40
41
42 def addUser(username, password):
43     """adding a new user to the database with 0 kills
44
45     Args:
46         username (string): username of the new player
47         password (string): password of the new player
48     """
49     # reading the file
```

```

50     with open('database.json') as f:
51         data = json.load(f)
52         # current list of dicts
53         current_users = data["users"]
54         # defining new dict for the new player
55         new_user = {"username": username, "password": password, "kills": 0}
56         # adding the new player to the list of dicts
57         current_users.append(new_user)
58
59     # writing the updated list back into the json file
60     with open('database.json', "w") as f:
61         json.dump(data, f, indent=4)
62
63
64 def signUp(username, password):
65     """full function for the sign up process, check if a user
66     is unique, if its unique it adds it to the database. if
67     its not unique than it doesn't
68
69     Args:
70         username (string): username
71         password (string): password
72
73     Returns:
74         bool: boolean for "was the process successful?"
75     """
76     if isUnique(username):
77         addUser(username, password)
78         return True
79     return False
80
81
82 def addKill(username):
83     """adds a kill for the specified player
84
85     Args:
86         username (string): username of player that has got another kill
87
88     Returns:
89         bool: return whether the addition was successful
90     """
91     did_work = False
92     with open("database.json") as f:
93         data = json.load(f)
94         user_list = data["users"]
95         for user in user_list:
96             if user["username"] == username:
97                 user["kills"] = user["kills"] + 1
98                 did_work = True
99
100     # writing the updated list back into the json file
101     with open('database.json', "w") as f:

```

```

102         json.dump(data, f, indent=4)
103
104     return did_work
105
106
107 def getKills(username):
108     """gets the amount of kills a player has
109
110     Args:
111         username (string): the username of the player
112
113     Returns:
114         int: the number of kills
115     """
116     with open("database.json") as f:
117         data = json.load(f)
118         user_list = data["users"]
119         for user in user_list:
120             if user["username"] == username:
121                 return user["kills"]
122

```

database.json

```
1  {
2      "users": [
3          {
4              "username": "luma",
5              "password": "3fc9b689459d738f8c88a3a48aa9e33542016b7a
6              4052e001aaa536fca74813cb",
7              "kills": 59
8          },
9          {
10             "username": "doobani",
11             "password": "3fc9b689459d738f8c88a3a48aa9e33542016b7a
12             4052e001aaa536fca74813cb",
13             "kills": 29
14         },
15         {
16             "username": "dizzy",
17             "password": "3fc9b689459d738f8c88a3a48aa9e33542016b7a
18             4052e001aaa536fca74813cb",
19             "kills": 13
20         }
21     ]
22 }
```


my_sha256.py

```
1 import hashlib
2
3
4 def hash_string(input_string):
5     """hashing function using sha256 but on strings
6
7     Args:
8         input_string (string): the data being encrypted
9
10    Returns:
11        string: hashed string
12    """
13    # making hash object
14    hash_object = hashlib.sha256()
15
16    # feeding the string into the object as bytes
17    hash_object.update(input_string.encode())
18
19    # digest the data as hexadecimal string of the hash
20    # return it
21    return hash_object.hexdigest()
22
```

network.py

```
1 import socket
2 from protocol import *
3 from AESHelper import AESHelper
4 from protocol import *
5
6 # class meant to handle the communication with the server by a player
7
8
9 class Network:
10     def __init__(self):
11         self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12         self.server = "192.168.10.128"
13         self.port = 5555
14         self.addr = (self.server, self.port)
15         self.pos = self.connect()
16
17     def getPos(self):
18         """method in charge of getting the original spawn position
19
20         Returns:
21             list: the spawn position of the player
22         """
23         return self.pos
24
25     def connect(self):
26         """connecting to the server and receiving first information
27
28         Returns:
29             list: spawn pos
30         """
31         try:
32             self.client.connect(self.addr)
33             return self.client.recv(1000).decode()
34         except socket.error as e:
35             print(e)
36
37     def send(self, data):
38         """
39         method that simplifies information flow to one
40         function that send and returns what is received
41         """
42         try:
43             # check is the data is already bytes and if not it converts it
44             if not isinstance(data, bytes):
45                 data = data.encode()
46
47             # send data
48             self.client.send(data)
49
```

```

50         return self.client.recv(1000)
51     except socket.error as e:
52         print(e)
53
54     def send_before(self, data):
55         """
56         method that simplifies information flow to one
57         function that send and returns what is received
58         """
59         try:
60             # check is the data is already bytes and if not it converts it
61             if not isinstance(data, bytes):
62                 data = data.encode()
63
64             # send data
65             self.client.send(data)
66
67             return self.client.recv(1000).decode()
68         except socket.error as e:
69             print(e)
70
71     def send_no_answer(self, data):
72         """
73         simple method that sends info
74         """
75         try:
76             # check is the data is already bytes and if not it converts it
77             if not isinstance(data, bytes):
78                 data = data.encode()
79
80             # send data
81             self.client.send(data)
82         except socket.error as e:
83             print(e)
84

```

protocol.py

```
1  """
2  general protocol headers
3  1: request to log in or sign up (username, password, isTryingToLogIn)
4  2: player game status (pos_x, pos_y, mouse_pos_x, mouse_pos_y, isShooting,
   health)
5  3: an answer to a request to log in (isValid)
6  4: player game status (like 2) + enemy's name
7  5: this digit simply marks the player requesting his original spawn position
   from the server, will not be stripped, only '5' will be sent
8  6: handshake - client requesting to get the server's public key, not stripped
9  7: public key being sent from the server to the client
10 8: encrypted aes key being sent back to the server
11 """
12
13
14 # reads data in a form of a string and uses the other functions
15 # also strips the protocol header
16 def read(data):
17     try:
18         first_letter = data[0]
19         data = data[1:]
20         if first_letter == '1':
21             return read_details(data)
22         elif first_letter == '2':
23             return read_pos(data)
24         elif first_letter == '3':
25             return read_ans(data)
26         elif first_letter == '4':
27             return read_name(data)
28         elif first_letter == '5':
29             return '5'
30         elif first_letter == '6':
31             return '6'
32         elif first_letter == '7':
33             return read_public_key(data)
34         elif first_letter == '8':
35             return read_encrypted_key(data)
36         else:
37             print('invalid protocol head-number: ' + first_letter)
38             return 'Protocol Fail'
39     except Exception as e:
40         print(f"An error occurred: {e}")
41         return 'Protocol Fail'
42
43
44 """ ==== 1 ==== """
45
46
47 def read_details(data):
48     """splits data requesting to log in
49
```

```

50     Args:
51         data (string): all of the parameters split with a ","
52
53     Returns:
54         tuple: (string) username, (string) password, (bool) isTryingToLogIg
55     """
56     try:
57         data = data.split(',')
58         if data[2] == 'True':
59             return str(data[0]), str(data[1]), True
60         else:
61             return str(data[0]), str(data[1]), False
62     except Exception as e:
63         print(f"An error occurred: {e}")
64         return 'Protocol Fail'
65
66
67 def make_details(data):
68     """makes a protocol string from a sign in / up request
69
70     Args:
71         tuple: (string) username, (string) password, (bool) isTryingToLogIg
72
73     Returns:
74         data (string): all of the parameters split with a "," + number "1" at
the start
75     """
76
77     try:
78         if data[2]:
79             return '1' + str(data[0]) + ',' + str(data[1]) + ',' + 'True'
80         else:
81             return '1' + str(data[0]) + ',' + str(data[1]) + ',' + 'False'
82     except Exception as e:
83         print(f"An error occurred: {e}")
84         return 'Protocol Fail'
85
86
87 """ ==== 2 ==== """
88
89 # translates string data of player's game position and status
90 # parameters: pos_x, pos_y, mouse_pos_x, mouse_pos_y, isShooting, health
91 # (mouse pos referring to one of their own screen ranging from 0 to screen
dimensions)
92
93
94 def read_pos(data):
95     """converts string data about the players position to individual variables
96
97     Args:
98         data (string): all of the players position attributes divided by ","
99
100     Returns:

```

```

101         tuple: (int) pos_x, (int) pos_y, (int) mouse_pos_x, (int) mouse_pos_y,
102         (bool) isShooting, (int) health
103         """
104         try:
105             data = data.split(",")
106             if data[4] == "True":
107                 return int(data[0]), int(data[1]), int(data[2]), int(data[3]),
108                 True, int(data[5])
109             else:
110                 return int(data[0]), int(data[1]), int(data[2]), int(data[3]),
111                 False, int(data[5])
112         except Exception as e:
113             print(f"An error occurred: {e}")
114             return 'Protocol Fail'
115
116 # makes data of player's game position and status from string
117 # parameters: pos_x, pos_y, mouse_pos_x, mouse_pos_y, isShooting, health
118 # (mouse pos referring to one of their own screen ranging from 0 to screen
119 # dimensions)
120
121 def make_pos(data):
122     """converts data in tuple about the players position to a protocol string
123
124     Args:
125         tuple: (int) pos_x, (int) pos_y, (int) mouse_pos_x, (int) mouse_pos_y,
126         (bool) isShooting, (int) health
127
128     Returns:
129         data (string): all of the players position attributes divided by "," +
130         a "2" on the start
131     """
132     try:
133         if data[4]:
134             return '2' + str(data[0]) + "," + str(data[1]) + "," + str(data[2])
135             + "," + str(data[3]) + "," + "True" + ',' + str(data[5])
136         else:
137             return '2' + str(data[0]) + "," + str(data[1]) + "," + str(data[2])
138             + "," + str(data[3]) + "," + "False" + ',' + str(data[5])
139         except Exception as e:
140             print(f"An error occurred: {e}")
141             return 'Protocol Fail'
142
143 """ ==== 3 ==== """
144
145 # reading string data indicating log in status
146
147 def read_ans(data):
148     """reading string data from the server regarding a log in / sign up
149     request
150
151     Args:

```

```

147         data (string): as string representation of True or False
148
149     Returns:
150         bool: the original answer from the server
151     """
152     try:
153         if data == "True":
154             return True
155         else:
156             return False
157     except Exception as e:
158         print(f"An error occurred: {e}")
159         return 'Protocol Fail'
160
161 # make string from data indicating log in status
162
163
164 def make_ans(data):
165     """making a protocol following string data from the
166     server regarding a log in / sign up request
167
168     Args:
169         bool: the original answer from the server
170
171     Returns:
172         data (string): as string representation of True or False + a "3" on
the start
173     """
174
175     try:
176         if data:
177             return '3' + 'True'
178         else:
179             return '3' + 'False'
180     except Exception as e:
181         print(f"An error occurred: {e}")
182         return 'Protocol Fail'
183
184
185 """ ==== 4 ==== """
186
187 # reading enemy's name and game status from string
188
189
190 def read_name(data):
191     """converts string data about the players position and username to
individual variables
192
193     Args:
194         data (string): all of the players position attributes divided by "," +
his username
195
196     Returns:
197         tuple: (int) pos_x, (int) pos_y, (int) mouse_pos_x, (int) mouse_pos_y,

```

```

198         (bool) isShooting, (int) health, (string) username
199     """
200     try:
201         data = data.split(",")
202         if data[4] == "True":
203             return int(data[0]), int(data[1]), int(data[2]), int(data[3]),
True, int(data[5]), data[6]
204         else:
205             return int(data[0]), int(data[1]), int(data[2]), int(data[3]),
False, int(data[5]), data[6]
206     except Exception as e:
207         print(f"An error occurred: {e}")
208         return 'Protocol Fail'
209
210 # making string of other persons name + game status
211
212
213 def make_name(data):
214     """converts data in tuple about the players position and username to a
protocol string
215
216     Args:
217         tuple: (int) pos_x, (int) pos_y, (int) mouse_pos_x, (int) mouse_pos_y,
(bool) isShooting, (int) health, (string) username
218
219     Returns:
220         data (string): all of the players position attributes and username
divided by "," + a "4" on the start
221     """
222     try:
223         if data[4]:
224             ret = f"4{data[0]},{data[1]},{data[2]},{data[3]},True,{data[5]},
{data[6]}"
225             return ret
226         else:
227             ret = f"4{data[0]},{data[1]},{data[2]},{data[3]},False,{data[5]},
{data[6]}"
228             return ret
229     except Exception as e:
230         print(f"An error occurred: {e}")
231         return 'Protocol Fail'
232
233
234 """ ==== 7 ===== """
235
236 # reads the public key being sent to the client from string
237
238
239 def read_public_key(data):
240     """reads the public key being sent from the server to the client,
the key includes an e and an n
241
242
243     Args:
244         data (string): both attributes of the key divided by a ','

```



```

245
246 Returns:
247     tuple: (string) public key, (string) n
248     ""
249 try:
250     data = data.split(",")
251     return data[0], data[1]
252 except Exception as e:
253     print(f"An error occurred: {e}")
254     return 'Protocol Fail'
255
256
257 def make_public_key(data):
258     """makes a string for the public key being set from the
259     server to the client
260
261     Args:
262         tuple: (string) public key, (string) n
263
264     Returns:
265         data (string): both attributes of the key divided by a ','
266         ""
267
268     try:
269         return f"7{data[0]},{data[1]}"
270     except Exception as e:
271         print(f"An error occurred: {e}")
272         return 'Protocol Fail'
273
274
275     "" == 8 == ""
276
277
278 def read_encrypted_key(data):
279     """retrieving the int value from a string representing the encrypted AES
280     key
281
282     Args:
283         data (string): the encrypted AES key
284
285     Returns:
286         (int): the encrypted AES key
287         ""
288
289     try:
290         return int(data)
291     except Exception as e:
292         print(f"An error occurred: {e}")
293         return 'Protocol Fail'
294
295
296 def make_encrypted_key(data):
297     """generating a protocol string representing the encrypted AES key

```

```
297 | Args:
298 |     data (int): encrypted AES key
299 |
300 | Returns:
301 |     string: "8" + string of the encrypted AES key
302 |     ""
303 | try:
304 |     return f"8{data}"
305 | except Exception as e:
306 |     print(f"An error occurred: {e}")
307 |     return 'Protocol Fail'
308 |
```

RSAHelper_client.py

```
1  from random import randint
2  from math import gcd
3
4
5  class RSAHelper_client:
6      """class meant for RSA encryption in only the client side,
7      incapable of generating keys or holding a private key, cant decrypt data
8      """
9
10     def __init__(self, public_key):
11         self.public_key = (int(public_key[0]), int(public_key[1]))
12
13     def get_public_key(self):
14         return self.public_key
15
16     def rsa_encrypt(self, message):
17         """in charge of encrypting the AES key,
18         turns the key from a binary to int and then encrypts it using RSA
19
20         Args:
21             message (bytes): AES key being encrypted
22
23         Returns:
24             int: encrypted AES key
25         """
26         # separating the public key to e and n
27         e, n = self.public_key
28         # converting the AES key from bytes to int
29         message_as_int = int.from_bytes(message, byteorder='big')
30         # encrypting {c = (m^e)(mod n)}
31         encrypted_data = pow(message_as_int, e, n)
32         return encrypted_data
33
```

RSAHelper.py

```
1 import sympy
2 from random import randint
3 from math import gcd
4
5 # Class for RSA encryption made by the server
6
7
8 class RSAHelper:
9     """class for all RSA encryption needs, can generate keys,
10     encrypt and decrypt
11     """
12
13     def __init__(self):
14         """goes through the process of creating keys
15         """
16         while True:
17             # generate p - large prime
18             p = sympy.nextprime(randint(2**1018, 2**1023))
19
20             # generate q - large prime
21             q = sympy.nextprime(randint(2**1018, 2**1023))
22
23             # regenerate q if needed
24             while q == p:
25                 q = sympy.nextprime(randint(2**1018, 2**1023))
26
27             # constant component of public/private keys
28             n = p * q
29
30             # helps create e, public key
31             phi = (p - 1) * (q - 1)
32
33             # very common co-prime to phi
34             e = 65537
35
36             # if e is co-prime with phi, exit the loop, we have what we need
37             # if its not co-prime, q and p will be regenerated
38             if gcd(e, phi) == 1:
39                 break
40
41             # d = (1/e)(mod phi)
42             d = pow(e, -1, phi)
43
44             self.public_key = (e, n)
45             self.private_key = (d, n)
46
47     def get_public_key(self):
48         return self.public_key
49
```

```

50 def rsa_decrypt(self, encrypted_message):
51     """decrypts the encrypted AES key
52
53     Args:
54         encrypted_message (int): encrypted AES key
55
56     Returns:
57         bytes: AES key
58     """
59     # splitting the private key to d and n
60     d, n = self.private_key
61     # decrypting {m = (c^d)(mod n)}
62     decrypted_int = pow(encrypted_message, d, n)
63     # returning msg to bytes
64     # adding 7 and floor dividing by 8 the bit length to calculate the
length of the decrypted msg
65     decrypted_message = decrypted_int.to_bytes(
66         (decrypted_int.bit_length() + 7) // 8, byteorder='big')
67     return decrypted_message
68
69 def rsa_encrypt(self, message):
70     """in charge of encrypting the AES key,
71     turns the key from a binary to int and then encrypts it using RSA
72
73     Args:
74         message (bytes): AES key being encrypted
75
76     Returns:
77         int: encrypted AES key
78     """
79     # separating the public key to e and n
80     e, n = self.public_key
81     # converting the AES key to int
82     message_as_int = int.from_bytes(message, byteorder='big')
83     # encrypting {c = (m^e)(mod n)}
84     encrypted_data = pow(message_as_int, e, n)
85     return encrypted_data
86

```

server.py

```
1 import socket
2 from _thread import *
3 from protocol import *
4 from database_manager import *
5 from RSAHelper import RSAHelper
6 from AESHelper import AESHelper
7 from AES_protocol import *
8 import pygame
9 import my_sha256
10
11
12 """
13 =====
14 =====
15
16 ADMIN PASSWORD
17
18 =====
19 =====
20 """
21
22
23 pygame.init()
24
25 screen = pygame.display.set_mode((600, 300)) # screen size
26 pygame.display.set_caption('LumGun - Server') # give window title
27 clock = pygame.time.Clock() # setting clock
28
29 give_access = False
30 pixel_font_small = pygame.font.Font('font/Pixeltype.ttf', 25)
31 pixel_font_mid = pygame.font.Font('font/Pixeltype.ttf', 35)
32 pixel_font_giant = pygame.font.Font('font/Pixeltype.ttf', 80)
33
34 # top text
35 menu_text_top_surf = pixel_font_giant.render(
36     "Enter Admin Password", False, 'Black')
37 menu_text_top_rect = menu_text_top_surf.get_rect(center=(300, 100))
38
39 # "click ENTER to activate"
40 menu_text_bottom_surf = pixel_font_mid.render(
41     "Click ENTER to Activate", False, 'Black')
42 menu_text_bottom_rect = menu_text_bottom_surf.get_rect(center=(300, 270))
43
44 """ password """
45
46 password_box_height = 150
47
48 color_active = pygame.Color(240, 240, 240)
49 color_passive = pygame.Color(207, 207, 207)
```

```

50
51 # the "password" text that shows up if text box is passive and empty
52 password_placeholder_surf = pixel_font_mid.render(
53     "password", False, (40, 40, 40))
54 password_placeholder_rect = password_placeholder_surf.get_rect(
55     center=(300, password_box_height))
56
57 # var that stores the username
58 password = ''
59 censored_password = ''
60
61 # the visible border around the username text box
62 password_border_rect = pygame.Rect(0, 0, 120, 28)
63 password_border_rect.center = (300, password_box_height)
64
65 # colors for the password text box border depending on activity
66 password_border_rect_color = color_passive
67
68 # bool for activeness of username text box
69 password_box_active = False
70
71
72 """ error message """
73
74 error_height = 180
75
76 msg = ""
77 show_error = False
78
79 # time from the start of program until error msg needs to be presented
80 start_time = pygame.time.get_ticks()
81
82 # tim from the start of program until current time (when msg is up)
83 current_time = pygame.time.get_ticks()
84
85 # the time the msg is presented, when it gets too big, the msg hides
86 delta_time = start_time - current_time
87
88 while not give_access:
89     for event in pygame.event.get():
90
91         if event.type == pygame.QUIT:
92             pygame.quit()
93             exit()
94
95         # if there is a click it decides if password textbox is active
96         if event.type == pygame.MOUSEBUTTONDOWN:
97             password_box_active =
password_border_rect.collidepoint(event.pos)
98
99         if event.type == pygame.KEYDOWN:
100             # if the enter key is not pressed

```

```

101         if event.key != pygame.K_RETURN:
102             if password_box_active:
103                 # if the backspace is pressed
104                 if event.key == pygame.K_BACKSPACE:
105                     # than the last char in password is deleted
106                     password = password[0:-1]
107                     censored_password = censored_password[0:-1]
108                 # limits username length to 18
109                 elif len(password) <= 18:
110                     allowed_characters = "
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789_"
111                     # check if the character is in the allowed set
112                     if event.unicode in allowed_characters:
113                         password += event.unicode
114                         censored_password = '*' * len(password)
115                 else:
116                     if my_sha256.hash_string(password) == "
8e0bdd994d9c0b4a093fa172c049788cfdcdea8a0133bad4abc511ac6189fc48":
117                         give_access = True
118                     else:
119                         # change the msg
120                         msg = 'Incorrect Password'
121                         # starting the 3000 millisecond timer to show msg
122                         show_error = True
123                         start_time = pygame.time.get_ticks()
124
125             # static elements
126             screen.fill('#b4ede1')
127             screen.blit(menu_text_top_surf, menu_text_top_rect)
128             screen.blit(menu_text_bottom_surf, menu_text_bottom_rect)
129
130             """ password textbox"""
131
132             # changes the color of the password text box outline based on active
status
133             if password_box_active:
134                 password_border_rect_color = color_active
135             else:
136                 password_border_rect_color = color_passive
137
138             # draw the password border rect
139             pygame.draw.rect(screen, password_border_rect_color,
140                             password_border_rect, 2)
141
142             # the surface on which the password is written on
143             password_surf = pixel_font_mid.render(censored_password, False, 'Black')
144             # the invisible rect around that password
145             password_rect = password_surf.get_rect(center=(300, password_box_height)
)
146
147             # changes the width of the password text box outline based on text
length
148             password_border_rect.w = max(password_rect.w + 8, 120)

```



```

149
150 # makes sure the password border rect is in the center
151 password_border_rect.center = (300, password_box_height)
152
153 # decides whether "password" text placeholder shows or actual password
154 if len(password) > 0 or password_box_active:
155     screen.blit(password_surf, password_rect)
156 else:
157     screen.blit(password_placeholder_surf, password_placeholder_rect)
158
159 """ error msg """
160
161 # if the error msg is supposed to show
162 if show_error:
163     error_surf = pixel_font_small.render(msg, False, 'Red')
164     error_rect = error_surf.get_rect(center=(300, error_height))
165     current_time = pygame.time.get_ticks()
166     delta_time = current_time - start_time
167
168     # makes sure the error displays for only 3000 milliseconds
169     if delta_time < 3000:
170         screen.blit(error_surf, error_rect)
171     else:
172         show_error = False
173         msg = ''
174
175 # draw all our elements
176 # update everything
177 pygame.display.update()
178 clock.tick(60) # 60 tick per second
179 pygame.display.quit()
180
181
182 """
183 =====
184 =====
185
186 Server working - functioning as server
187
188 =====
189 =====
190 """
191
192 accept_all_connections_ip = "0.0.0.0"
193 port = 5555
194
195 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
196
197
198 def other(p):
199     """returns the value of the other player
200

```

```

201     Args:
202         p (int): 0 or 1, the number of a player
203
204     Returns:
205         int: 0 or 1, whatever p isn't
206     """
207     if p == 0:
208         return 1
209     else:
210         return 0
211
212
213 try:
214     s.bind((accept_all_connections_ip, port))
215 except socket.error as e:
216     str(e)
217
218 s.listen(2)
219 print("Waiting for a connection, Server Started")
220
221 rsa_helper = RSAHelper()
222
223 players_online = 0
224 # list for the original spawn positions
225 spawn_pos = [(200, 250, 200, 200, False, 30), (800, 250, 200, 300, False,
226 30)]
227 # list for storing the current player positions
228 pos = [spawn_pos[0], spawn_pos[1]]
229 # list for storing the player's usernames
230 player_usernames = ['', '']
231 # list for tracking how many shots a player shot,
232 # have to use it in case two messages are received
233 # for the same player in a row
234 bullets_shot = [0, 0]
235
236 # tracking the connections
237 connections = [None, None]
238
239 # saving the AES encryption object per player
240 aes_lst = [None, None]
241
242 def threaded_client(conn, player):
243
244     # sending non sensitive spawn pos cords
245     conn.sendall(str.encode(make_pos(spawn_pos[player])))
246
247     """ RSA HANDSHAKE """
248
249     print(f'==== Handshake player {[player + 1]} ===')
250
251     # receiving request for public key

```

```

252     data = read(conn.recv(120).decode())
253
254     if data == '6':
255         print('2. received request for public key')
256         # sending the public key
257         print('3. sending public key: ' +
258             make_public_key(rsa_helper.get_public_key())[1:])
259         conn.sendall(str.encode(make_public_key(rsa_helper.get_public_key())
260 ))
261
262         # receiving the encrypted aes key
263         encrypted_aes_key = read(conn.recv(1000).decode())
264         print('7. received encrypted AES key: ' + str(encrypted_aes_key))
265
266         # decrypting the encrypted aes key
267         aes_key = rsa_helper.rsa_decrypt(encrypted_aes_key)
268
269         print("8. decrypted AES key: " + str(aes_key))
270
271         # adding to the aes object the one transferred
272         aes_lst[player] = AESHelper(aes_key)
273
274         # adding the connection
275         connections[player] = conn
276
277         # variable that turns to True if there is a lost connection in the home
278         # screen,
279         # tells to skip the while True loop for the game in order to shut down
280         # the connection
281         skip_game_loop = False
282
283         """ SIGN IN / SIGN UP """
284         while True:
285             data = conn.recv(120)
286
287             # if the there if no data, disconnect the client
288             if not data:
289                 print(f"player {player + 1}: No data received")
290                 skip_game_loop = True
291                 break
292
293             print("==== Menu ====")
294             print(f"Received from player {player + 1} [encrypted]: ", data)
295
296             data = read(aes_lst[player].aes_decrypt(*read_cipheriv(data)))
297
298             # if the data received doesn't follow protocol, than disconnect him
299             if data == 'Protocol Fail':
300                 print(
301                     f"player {player + 1}: Decrypted data doesn't follow
302 protocol")
303                 skip_game_loop = True
304                 break

```

```

301
302     print(f"Received from player {player + 1}: ", data)
303
304     # if the request is for signing in
305     if data[2]:
306         # if log in was valid
307         if isValid(data[0], data[1]):
308             # send True back
309             conn.sendall(make_cipheriv(
310                 aes_lst[player].aes_encrypt(make_ans(True))))
311             # go to game loop
312             print(f"Sending to player {player + 1}: True")
313             break
314         # if log in was NOT valid
315         else:
316             # send False
317             conn.sendall(make_cipheriv(
318                 aes_lst[player].aes_encrypt(make_ans(False))))
319             # redo the loop
320             print(f"Sending to player {player + 1}: False")
321             continue
322     # if the request is to sign up
323     else:
324         # if signing up was successful
325         if signUp(data[0], data[1]):
326             conn.sendall(make_cipheriv(
327                 aes_lst[player].aes_encrypt(make_ans(True))))
328             # go to game loop
329             print(f"Sending to player {player + 1}: True")
330             break
331         else:
332             conn.sendall(make_cipheriv(
333                 aes_lst[player].aes_encrypt(make_ans(False))))
334             # redo the loop
335             print(f"Sending to player {player + 1}: False")
336             continue
337
338     # if there wasn't a problem with the login process
339     if not skip_game_loop:
340         # save the connected name
341         player_usernames[player] = data[0]
342
343     """ GAME """
344
345     reply = ""
346     while True:
347         try:
348
349             # if data wasn't received during sign in
350             if skip_game_loop:
351                 # leave game loop
352                 break

```

```

353
354     data = conn.recv(120)
355
356     # if there's no data being received
357     if not data:
358         # print correspondent message
359         print(f"player {player + 1}: No data received")
360         # exit the game loop
361         break
362
363     print("==== Game ====")
364     print(f"Received from player {player + 1} [encrypted]: ", data)
365
366     data = read(aes_lst[player].aes_decrypt(*read_cipheriv(data)))
367
368     # if the data received doesn't follow protocol
369     if data == 'Protocol Fail':
370         # print correspondent message
371         print(
372             f"player {player + 1}: Decrypted data doesn't follow
protocol")
373         # leave game loop
374         break
375
376     # if the player reports about his own death
377     if data == '5':
378         # changes his stored pos to his spawn pos
379         pos[player] = spawn_pos[player]
380         # adds a kill to the other player
381         addKill(player_usernames[other(player)])
382         # sends the dead player his spawn position
383         conn.sendall(make_cipheriv(
384             aes_lst[player].aes_encrypt(make_pos((pos[player])))))
385         continue
386     # if its not a death the server will continue like normal
387     else:
388         # storing the players position into the pos list
389         pos[player] = data
390
391     # if a player shot a shot, add one to his shots meant for
transfer
392     if data[4]:
393         bullets_shot[player] += 1
394
395     # setting the reply to be the other players position
396     # initializing variable for "should the other player be
shooting?"
397     other_player_shooting = False
398
399     # updating the isShooting variable for the other player
400     if bullets_shot[other(player)] > 0:
401         other_player_shooting = True
402         bullets_shot[other(player)] -= 1

```

```

403
404         # sending a response with all of the detail, inserting the
shooting bool in the middle
405         reply = (*pos[other(player)][0:4],
406                 other_player_shooting, pos[other(player)][5])
407
408         # server prints information
409         print(f"Received from player {player + 1}: ", data)
410         print(f"Sending to player {player + 1}: ", reply)
411
412         # sending information back to the client:
413         # position and name, using protocol 4
414         conn.sendall(make_cipheriv(aes_lst[player].aes_encrypt(
415             make_name((*reply, player_usernames[other(player)]))))))
416     except error as e:
417         print(e)
418
419     # the server lost connection with the user
420
421     print(f"player {player + 1}: Lost connection")
422     # resetting position to spawn position
423     pos[player] = spawn_pos[player]
424     # resetting the username
425     player_usernames[player] = ''
426     # resetting the count of the player
427     bullets_shot[player] = 0
428     # clearing connection
429     connections[player] = None
430     aes_lst[player] = None
431     conn.close()
432
433
434 while True:
435     # accepting connections
436     conn, addr = s.accept()
437     print("Connected to:", addr)
438
439     # finding player slots
440     player_slot = -1
441     for i in range(2):
442         if connections[i] is None:
443             player_slot = i
444             break
445
446     # if both the slots are filled
447     if player_slot == -1:
448         # print msg
449         print("Server full, cannot handle more connections.")
450         # close the new connection
451         conn.close()
452         continue
453

```

```
454 | start_new_thread(threaded_client, (conn, player_slot))
455 |
456 | players_online = players_online + 1
457 |
```