תיק פרוייקט Lethal Justice

ניתאי לבנון
326343902
תיכון בן גוריון – פתח תקווה

תוכן עניינים

פרק א' - Lethal Justice -ייזום

1. תיאור ראשוני של המערכת

הפרוייקט הוא משחק יריות רב משתמשים, מזווית top down, המטרה היא להרוג כמה שיותר משתמשים ולצבור את הניקוד הגבוה ביותר בתוך כ5 דקות.

החלטתי לבנות פרוייקט זה מתוך חיבה לגרפיקה ממוחשבת, אני לומד את תחום הגרפיקה הממוחשבת גם באוניברסיטה ותמיד הייתה לי משיכה לתחום, לכן רציתי לאתגר את עצמי ולבנות משחק שגם משלב תקשורת בין מחשבים שזה אחד מן התחומים הנלמדים בבית הספר, וגם תחום הגרפיקה שאליו אני נמשך מאוד.

2. הגדרת לקוח

המשחק מיועד למשתמשי מחשב בכל מערכות ההפעלה (windows, linux) ובפרט לחובבי משחקי מחשב, משחק אומנם משחק יריות אך אינו כולל תיאור גרפי שאינו הולם לילדים, לכן הוא מתאים לכל הגילאים.

3. הגדרות יעדים\מטרות

למשחק מטרה ברורה, שהיא בידור והנאת המשתמש, אך כלפיי, למשחק מטרה להרחיב אופקיי ולשפר אותי בתכנות בפרט.

4. בעיות תועלות וחסכונות

בימינו, רוב משחקי היריות כבדים נורא בדרך כלל בשל העובדה שהם משחקי תלת מימד.
אני רציתי לבנות משחק שלא ישקול הרבה, שכל מחשב יוכל להריץ ללא צורך במפרט חזק, ושיהיה חדשני ומהנה, הרוב המוחלט של משחקי היריות בדו מימד שקיימים היום בשוק ישנים מאוד, ובלתי ניתנים ל"שיחוק", לכן החלטתי לנסות לבנות משחק לא כבד, חדשני, ומהנה.

5. האם צפויים קשיים או מגבלות בהגדרת המערכת

השתמשתי בספרייה הגרפית של פייתון, pygame, הטכנולוגיה אינה חדשה והיא מוכרת.
על המשתמשים להיות בעלי חיבור יציב לאינטרנט, **ולהיות על אותה רשת אינטרנט**, ובכך לשחק אחד נגד השני, כמו כן על המחשב צריך להיות מותקן python והספריות בהן נעשה שימוש בפרוייקט ועליהן אפרט בהמשך.

6. תיחום הפרוייקט

הפרוייקט עוסק בתכנות המשחק, ובבניית מערכת שרת לקוח שמחברת מספר שחקנים אינסופי למשחק יחדיו.
הפרוייקט אינו מטפל במערכת ההפעלה עליה המשחק רץ.

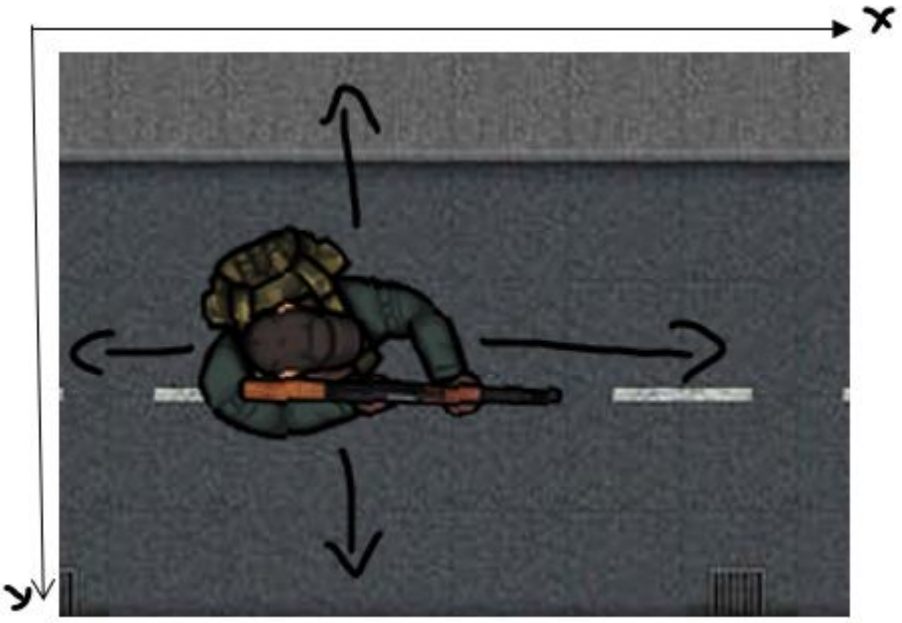
פרק ב' – Lethal Justice - אפיון

א. פירוט המערכת

Lethal Justice הינו משחק יריות דו מימד בזווית top down, המשחק בנוי למספר בלתי מוגבל של משתמשים, המשחק דורש יכולות משחק גבוהות, דיוק, חדות, וקור רוח.
מטרת המשתמש היא להרוג כמה שיותר משתמשים אחרים בזמן הקצוב הנתון, כל הריגה תקנה למשתמש נקודה אחת.
אם המשתמש נהרג אז הוא ישתגר למקום אחר במפה, עם מד חיים מלא, בדרך להמשיך במסעו אל עבר הניצחון.

ב. מה היכולות שהיא תעניק למשתמש, פירוט היכולות:

i. תנועה של הדמות בדו מימד במפה במתואר בתמונה, בעזרת הכפתורים w, a, s, d כאשר W גורם לדמות לעלות למעלה (במורד ציר הy), S גורם לדמות לרדת למטה, A גורם לדמות לזוז שמאלה (במורד ציר הx) וD גורם לדמות לזוז ימינה.



ii. הדמות תסתכל על העכבר כמומחש בתמונה.

iii. הדמות תירה בכיוון אליה היא מסתכל מהנשק בלחיצה על רווח.

iv. לאחר ריקון המחסנית (30 כדורים) על הדמות להטעין את מחסנית, לחיצה על R תבצע זאת ותחדש את מלאי הכדורים במחסנית. (ניתן לעשות זאת גם אם המחסנית אינה מרוקנת)

ג. תכנון לוח זמנים לפרוייקט

פעילות	זמן התחלה מתוכנן	זמן סיום מתוכנן	זמן התחלה בפועל	זמן סיום בפועל	הערות
כתיבת קוד המשחק	**1.11.21**	**15.12.21**	**1.11.21**	**1.12.21**	**כתיבת קוד המשחק עצמו הייתה מהירה מהצפוי, השקעתי עליה זמן והנבתי התקדמות יפה**
כתיבת הקוד לצד הלקוח	**1.1.22**	**1.3.22**	**1.5.22**	**27.5.22** **אלו תאריכים משוערים, רוב הקוד גמור אך דרוש קצת שיפור**	**העיכוב הרב נבע ממקרים אישיים במשפחה (בפרט פטירת אימי)**
כתיבת קוד השרת	**1.1.22**	**1.3.22**	**1.5.22**	**27.5.22**	**ההערה המפורטת לעיל תקפה גם פה**
שיפור האסטטיקה והתפעול	**15.3.22**	**31.3.22**	**25.5.22**	**27.5.22**	
בדיקות הפרוייקט	**1.4.22**	**15.4.22**	**20.5.22**	**24.5.22**	
סיום ספר הפרוייקט	**1.5.22**	**15.5.22**	**20.5.22**	**24.5.22**	

פרק ג' – Lethal Justice – מסמך ניתוח

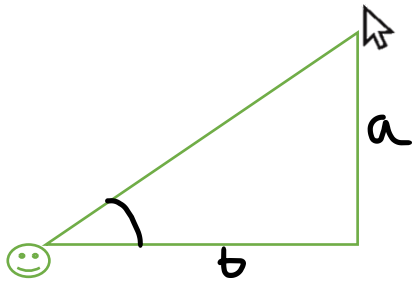
א. פירוט יכולות המערכת ההוצגו בפרק הקודם

תזוזת המשתמש:

היכולת מאפשרת למשתמש לזוז ברחבי המפה בחופשיות בעולם דו מימדי, כאשר יש קליטת אחד ממקשי התנועה, יש קריאה לתהליך ()movex כאשר x הוא אחד מכיווני התנועה בהתאמה, הפעולה שייכת לאווביקט מסוג sprite, המשוייך למשתמש, הפעולה מעדכנת את ערכי הx והy החדשים שלו לאחר התנועה בכיוון בהתאם לתנועה. לכן היכולת משתמשת באוביקט sprite ובתכונות sprite.x ו-sprite.y כמו כן הפעולה תעלה את המשתנה של האוביקט sprite.walkcount ששולט באנימציית התזוזה.

הסתכלות על העכבר:

יכולת זאת יותר מסובכת למימוש, עבורה השתמשתי בספרייה math על מנת להשתמש בפונקציות טריגונומטריות.
על מנת לסובב את התמונה השתמשתי בתהליך biltrotate שמדפיס על האוביקט screen, שהוא המסך בpygame, תמונה. כמו כן התהליך מקבל פרמטר angle שמסובב את התמונה בזווית נתונה. לכן נשאר לחשב את הזווית בין הדמות לעכבר, ולהדפיס את דמות המשתמש בעזרת התהליך blitrotate, בכך שנותנים לתהליך את הזווית שחישבנו.



אם הסמיילי הוא הדמות, אז הזווית בין העכבר לדמות היא tan(a/b) ובעזרת הספרייה math נחשב זאת ונשמור את הזווית בתכונה sprite.angle של המשתמש.
תהליך חישוב זה נמצא באוביקט sprite וקוראים לו ()calculateAngle, וקוראים לתהליך זה בכל פעם בgameloop
ובביצוע התהליך blitsprites באוביקט screen מתבצעת הדפסה על המסך של כל הsprites ושם קוראים לפעולה blitrotate עם הזווית המתאימה של כל sprite של כל משתמש בנפרד.

יריה בנשק

היריה בנשק מאוד פשוטה לביצוע, קיים אוביקט weapon, אשר יש לו רשימה של אוביקטים מסוג bullet, לכל אוביקט bullet יש תכונה של x,y.
לכן בכל פעם שהמשתמש לוחץ על מקש הרווח לרשימת הbulletsיתווסף bullet חדש עם x,y התחלתיים הממוקמים בקצה הקנה*, ונוריד לאוביקט weapon 1 מהתכונה weapon.ammo שמייצג את תחמושת הנשק

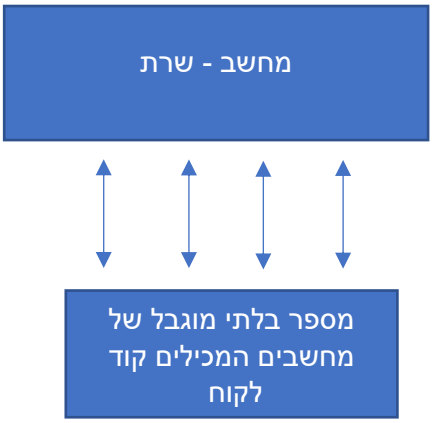
הדפסת הכדורים מתבצעת בblitprojectile כך שזווית ההדפסה של הכדור היא בדיוק הזווית של הדמות וגם שם יש שימוש בblitrotate, כמו כן מקדמים את הערכים x,y של הכדור בכיוון התנועה שלהם באמצעות טריגונומטריה ובעזרת התכונה vel של האובייקט weapon שמייצג את מהירות הכדור.

<u>הטענת המחסנית:</u>
בלחיצה על הכפתור R, תחילה משנים את התכונה הבוליאנית של האובייקט sprite-sprite.reload לTrue, שכן מתבצעת פעולת טעינה, כמו כן מדפיסים את אנימציית הטעינה של הsprite, ומגדירים מחדש את כמות הכדורים במחסנית השמור באובייקט weapon.ammo להיות weapon.mag שזה כמות הכדורים המקסימלית במחסנית.

פרק ד' – Lethal Justice - עיצוב

א. תיאור האכיטקטורה של המערכת המוצעת

החמרה בה יהיה שימור היא מחשב בעל מערכת הפעלה כלשהי (ללא חשיבות) הספריות הדרושות וקוד הלקוח, כמו כן מחשב עליו יושב השרת שמחבר בין הלקוחות. כל המכשירים האלו נמצאים על אותה רשת אינטרנט.



ב. תיאור הטכנולוגיה הרלוונטית

מחשב לכתיבת קוד המשחק, השרת והלקוח, מחשב להרצת השרת ומחשב בעל קוד לקוח להרצת משתמש.
המודלים בהם נעשה שימוש:

ג.תיאור המודלים בהם נעשה שימוש

i. מודלים מיובאים:
pygame הספרייה הגרפית שבעזרתה הדפסנו תמונות למסך ויצרנו משחק.
Math אשר נעשה בו שימוש בפאן הגרפי בשימוש בפונקציות טריגונומטריות
pickle אשר נעשה בו שימוש בקוד הלקוח, המודל פיקל מסוגל להמיר אובייקטים לסטרינג בינארי, וההפך, ובכך מאפשר להעביר סרבר אובייקטים ולהקל על עבודת התקשורת.
socket אפשר תקשורת בין משתמשים שונים, בעזרת סרבר והעברת פקטות מידע בין sockets.
Threading אפשר יצירת תהליכונים בסרבר ובקליינט, כך שתהליכים שונים יוכלו גם להאזין ולחכות לקבל מידע מהשרת (או מהקליינטים השונים, במקרה של השרת) וגם לבצע את פעולות המשחק הדרושות ולשלוח הודעות בין הקליינט לסרבר.
Time הספרייה הבסיסית time מאפשרת לקפוא ולחכות זמן מסויים ללא כל מעש, למשל לפני תחילת המשחק.

ii. מודלים שאני כתבתי:

Server:

תכונות:

server.server_socket הסוקט של הסרבר

server.conn_counter משתנה רץ הסופר את כמות המשתמשים

()run - רץ בלולאה אינסופית המקבלת חיבור קליינטים, וישר קוראת לתהליכון עם הקליינט שהתקבל לולאה אינסופית אחרת הבודקת האם התקבלה הודעה מאותו קליינט, במידה והתקבלה, היא קוראת לפעולה sendAll עם ההודעה שהתקבלה

(sendAll(message - פעולה השולחת לכל הקליינטים המחוברים לשרת את ההודעה message אשר התקבלה ממשתמש כלשהו, לכן שולחת זאת לכל המשתמשים למעט המשתמש אשר שלח את ההודעה.

```
def send_all(self, message):
global clients_list
for client in clients_list:
if client != self.client:
client.send(message)
```

Client:

התכונות:

self.cllient_socket הסוקט של המשתמש

self.screen השומרת את האובייקט סקרין (מסך הpygame)

self.id מספר סידורי שמקבלת מהסרבר לפי המשתנה הרץ

()run פעולת תהליכון, שרצה בלולאה אינסופית אשר ממתינה לקבלת מידע מהסרבר, כאשר מתקבל מידע ממירים אותו חזרה לאובייקט בעזרת pickle. המידע אשר היא מקבלת מהסרבר זה אובייקט sprite של משתמש אחר שנע במרחב, לכן יש להדפיס אותו על המסך, ומשם קוראים לפעולה של האובייקט client.screen אותו שמרנו אשר מוסיפה אותו לרשימת הsprites (()adsprite) – על הפעולה נפרט בהמשך

(send_message(message שולח הודעה לשרת, שם יש שימוש בפיקל, ההודעה היא האובייקט sprite שלנו, בעזרת pickle אנחנו ממירים אותו לטקסט בינארי ושולחים אותו לשרת.

main:

()main הפעולה הראשית בה יש קריאות לפעולות אחרות

()gameloop הלולאה האינסופית של המשחק, רצה עד להתנתקות המשתמש.
בgameloop יש קריאה לפונקציה ()check_key_input וכמו כן פונקציות הבודקות את זווית דמות המשתמש והדפסת כדורים אשר נורו על המסך כפי שפירטתי בעבר.

()check_key_input הפעולה בודקת מה המקשים שנלחצו ובהתאם לכך מבצעת פעולה מתאימה (מזיזה את השחקן או יורה כדור), לאחר ביצוע הפעולה אם נלחץ כפתור אז יש שינוי כלשהו בתמונות המודפסות על המסך, אז יש לעדכן את שאר המשתמשים ולעדכן את האובייקטים השמורים שם, ובכך שולחים לסרבר את האובייקטים המעודכנים בעזרת הפעולה ()send_message

screen:

תכונות:
width רוחב המסך
height אורך המסך
screen.screen אובייקט מסך הפייגיים עצמו.
screen.sprites רשימת הsprites של כל המשתמשים השונים.

פעולות:
()addsprite מוסיף לרשימת הsprites דמות חדשה, למשל בעת קבלת sprite של משתמש אחר מהserver.

()blitsprite עובר על רשימת הsprites ומדפיס את כולם על המסך.

(blitrotate(image, pos, originPos, angle מדפיס תמונה נתונה בזווית מסויימת angle, יש לשלוח לפעולה את התמונה הרצויה, את המיקום שלה ואת מיקום האמצע שלה

()updateScreen מבצע פעולת ריפרוש למסך בעזרת הפעולה display.flip וscreen.fill שהן פעולות של פייגיים אשר נועדו לעדכן את המסך לאחר שינוי גרפי כלשהו.

(blitProjectile(gun gun הוא אובייקט weapon שלו יש מערך של הכדורים שנורו עם המיקומים שלהם, blitProjectile מחשב את הx, y שיש להדפיס את הכדור שנורה ביחס לזווית השחקן ומיקום הקנה שלו, ומחשב זאת בעזרת טריגונומטריה כמו כן לאחר מכן קורה לפעולה bitRotate כדי להדפיס כל כדור בנפרד בזווית הנכונה.

sprite:

תכונות:
x ערך הx של הדמות
y ערך הy של הדמות
vel מהירות התזוזה של הדמות

walkcount משתנה רץ אשר נועד לעזור להדפסת אנימציות התזוזה והטעינה
angle זווית הדמות אשר משתנה בהתאם לעכבר
gun אובייקט הweapon אליו הוא קשור
id מספר הסידורי של הקליינט של האובייקט
reloaded משתנה בוליאני שקובע האם הsprite כרגע טוען את המחסנית או לא

calculateAngle(mouse_x, mouse_y) בעזרת מיקומי העכבר, ובעזרת מתמטיקה, מחושבת הזווית של הדמות כפי שתואר כבר למעלה.

right, left, up, down() כל אלה פעולות שונות אשר משנות את ערך הx והy של הדמות בהתאמה, כמו כן מזיזות את ערך המשתנה הרץ walkcount אשר נועד להדפסת אנימציית התזוזה.

addBullet() פעולה אשר מוסיפה לתכונת הgun עוד כדור, היא קוראת לפקודה addBullet ושולחת לפקודה גם את מיקום הקנה של הנשק במידה והsprite בזווית ישרה, וגם שולחת את זווית הדמות כדי לחשב את מיקום הכדור האמיתי.

Weapon:

תכונות:

mag כמות הכדורים המקסימלית במחסנית הנשק
ammo כמות הכדורים הקיימת כרגע במחסנית
vel מהירות תנועת הכדור
bullets רשימה של כדורים אשר נורו

addBullet(x, y, angle) בודקת תחילה אם יש תחמושת, אם אין אז חוזרת ללא כלום, אם יש תחמושת אז היא מבצעת את הפעולה rotateBullet שהמימוש שלה דומה מאוד לblitRotate, רק שבמקום להדפיס את הכדור בסוף הפעולה, היא מחזירה את ערכי הx והy המסובבים שלה, בעזרת ערכים אלו היא יוצרת אובייקט bullet חדש ומוסיפה אותו לרשימה bullets.

rorateBullet(image, pos, originPos, angle) מבצעת בדיוק מה שתיארתי לעיל.

reloadAmmo() מגדירה את התכונה ammo להיות mag, כלומר טוענת מחדש את המחסנית

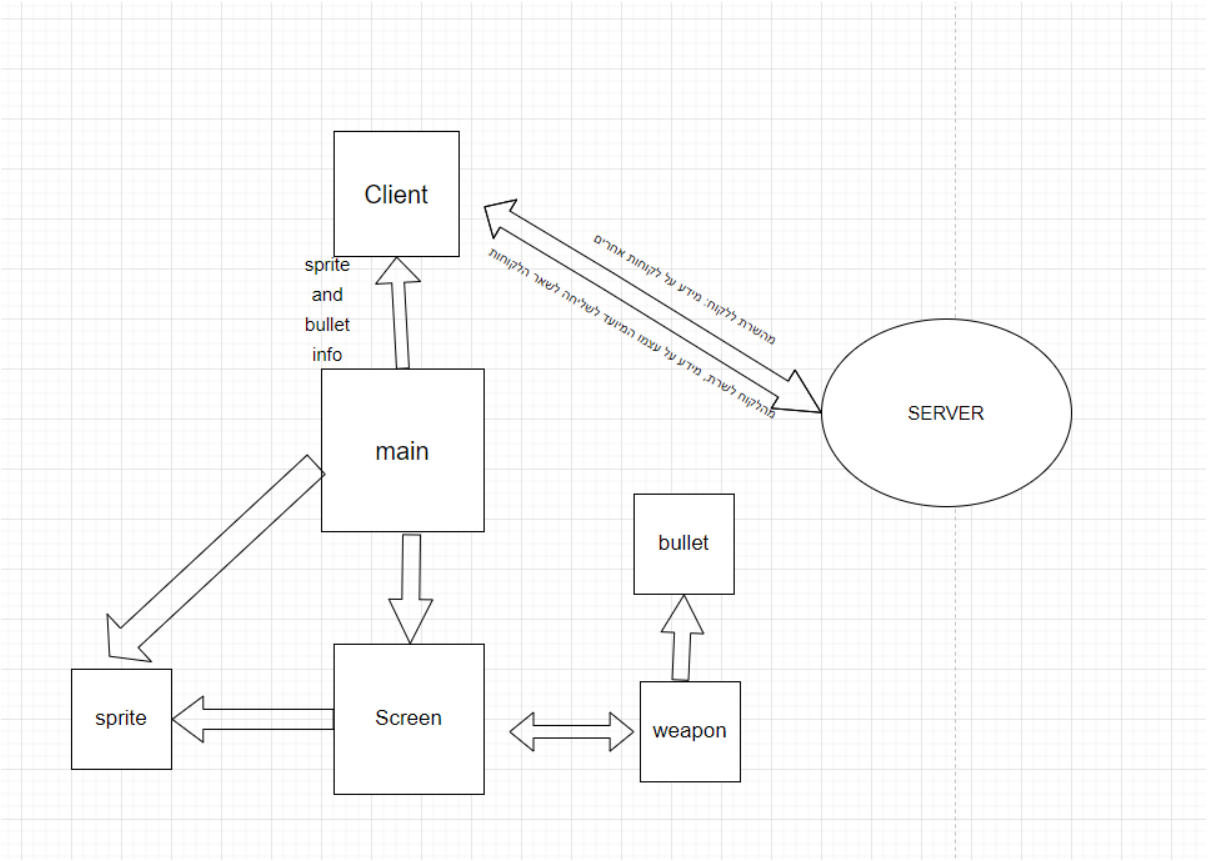
Bullet:

תכונות

x ערך הx של הכדור
y ערך הy של הכדור
angle זווית יריית הכדור

למחלקה זאת יש רק גטרס וסטרס

ובתרשים זרימה מתומצת:



ד. תיאור סביבת הפיתוח

python – השפה בה כתבתי את הפרוייקט היא פייתון, בה אני הכי שולט והיה לי מאוד נוח לעבוד איתה

Pycharm - הIDE בו השתמשתי, הוא היה מאוד נוח, יכולתי דרכו להוריד מודולים שונים ולגשת לconsole באופן מובנה דרך הכתבן.

ה. תיאור האלגוריתמים המרכזיים בפרוייקט:

1. בעיה – חיבור כל השחקנים לonline

פתרונות אפשריים:

- שימוש בספרייה מוכנה לכך הכתובה בשפה אחרת
- יצירת שרת בפייתון בשימוש בספריות socket, threads

בחרתי לבנות את השרת בפייתון בשימוש בספריות אלו, מאחר שזה יאפשר לי יותר גמישות בכתיבת הקוד, כמו כן יעמיק את ידיעותיי בנושא שסביבו סובב הפרוייקט.

2. בעיה – העברת המידע בין משתמשים

פתרונות אפשריים:

- העברת ערכי x,y דרך שרת
- העברת אובייקטים שלמים כקובץ בינארי בשימוש עם המודל pickle

בחרתי באפשרות השניה שבמקרה שלי מקלה על כתיבת הקוד, על אף חוסר יעילות pickle וזמן הריצה הניכר שהיא גוזלת מהתוכנית.

3. בעיה – התנתקות שחקנים

פתרונות אפשריים:

- כתיבת קוד אשר תסיים את המשחק שהתחיל, ותחכה להתחלת משחק חדש
- כתיבת קוד מסובך יותר, אשר תנתק את המשתמש ותמחק אותו מהמשחק ותמחק את האובייקטים שלו, אך תאפשר המשך משחק רציף של שאר המשתמשים שלא נותקו.

בחרתי באפשרות השניה על מנת לאפשר למשחק להשאר רציף כפי שהוא

4. בעיה – אנימציית הטעינה

פתרונות אפשריים

-מגוון תמונות שכאשר רצים ישר יוצרים מאין סטופ מושן של טעינ
-גיף או סרטון קצר של טעינת הדמות

בחרתי כמובן באפשרות הראשונה, לטעון גיף או סרטון קצר ירוץ מאוד לאט ויתקע את כל המשחק, לכן כאשר הדמות רוצה לטעון הרצתי תמונות של טעינה שיוצרות אפקט יפה מאוד של טעינת הנשק.

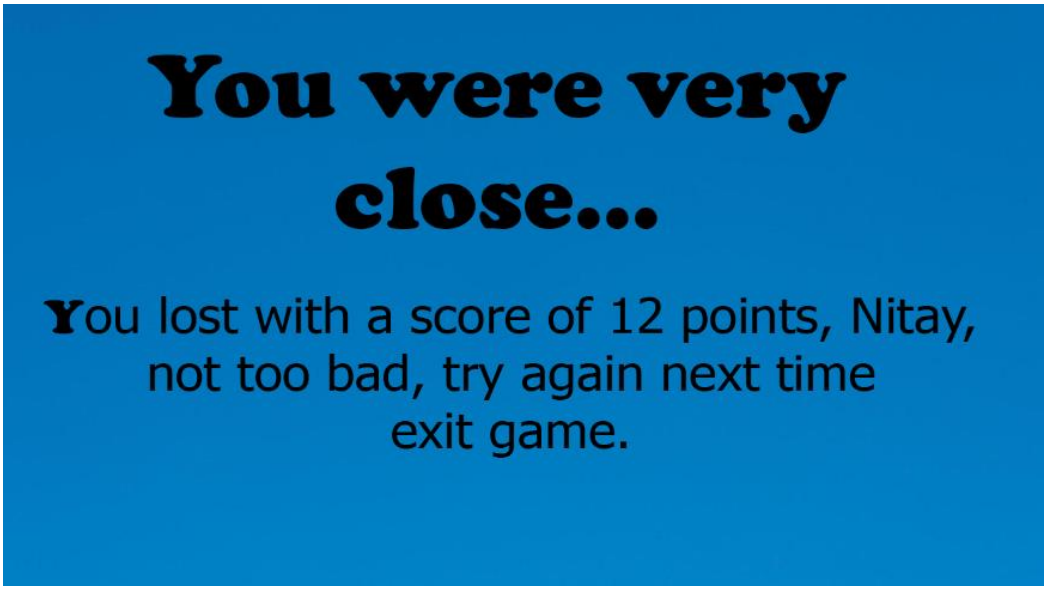
ו. תיאור מסכי הפרוייקט

מסך הכניסה למשחק: על המשתמש להקליד את שמו וללחוץ אנטר

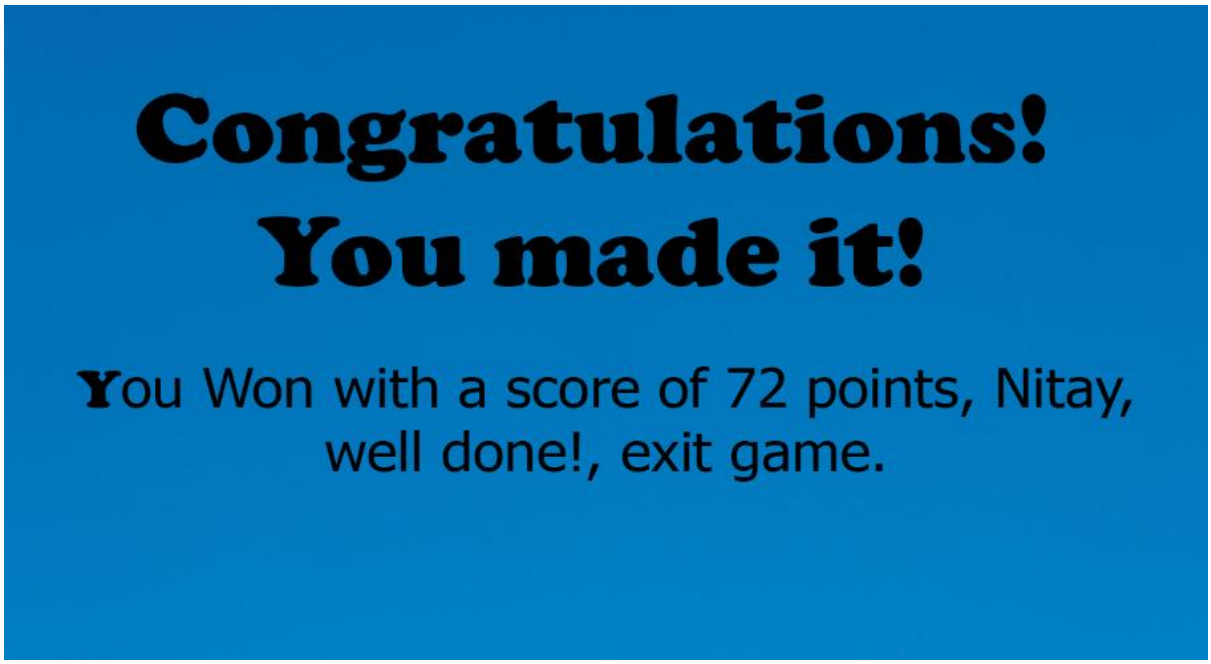
מסך בזמן שמחכים שהסרבר יתחיל את המשחק:

מסך המשחק:

מסך הפסד משחק:



מסך ניצחון משחק:



ז. תיאור מבני נתונים

את כל מבני הנתונים השונים של הפרוייקט תיארתי כבר למעלה בהסבר על המחלקות ועל הפונקציות, ולא נשמרים בפרוייקט זה נתונים בין משחקים שונים ובין כניסות שונות ולכן אין ממסדי נתונים.

פרק ה' – Lethal Justice – הקוד

קטע קוד 1:

קטע הקוד של הסרבר שמקבל מספר בלתי מוגבל של קליינטס ומנהל את כל קבלת ההודעות מהקליינטים השונים במקביל.

```
class Server(threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.server_socket = socket.socket()
        self.flag = True
        self.conn_counter = 1 # counter of the connections

    def startup(self):
        try:
            self.server_socket.bind((IP, PORT))
            self.server_socket.listen()
        except Exception as e:
            print(e)

    def run(self): # waiting for connection and adding it to clients_list
        global clients_list
        while self.flag:
            client_socket, client_address = self.server_socket.accept()
            print(f"{client_address} connected")
            message = str(self.conn_counter)
            client_socket.send(message.encode())
            clients_list.append(client_socket)
            self.conn_counter += 1
            ProducerThread(client_socket, client_address).start()


class ProducerThread(threading.Thread):
    def __init__(self, client, address):
        threading.Thread.__init__(self)
        self.client = client
        self.address = address

    def run(self):
        while True: # A thread waiting to receive message
            message = self.client.recv(8192)
            self.send_all(message)

    def send_all(self, message): # sending all clients The message
        global clients_list
        for client in clients_list:
            if client != self.client:
                client.send(message)
```

קטע קוד 2:

קטע הקוד הלקוח מתוך המחלקה screen אשר אחראי להדפיס גם את הsprites וגם והדפיס את הprojectiles שהם כדורי הנשק.

```
def blitsprite(self):
    for sprite in self.sprites:
        sprite_images, walk_count = sprite.getimage()
        sprite_images = pygame.transform.scale(sprite_images[walk_count],
(156, 100))
        self.blitRotate(sprite_images,(sprite.getx() + 78, sprite.gety() +
50) , (78, 50), sprite.angle) #(79, 50) is the middle of the sprite image

def blitRotate(self, image, pos, originPos, angle):
    # offset from pivot to center
    image_rect = image.get_rect(topleft=(pos[0] - originPos[0], pos[1] -
originPos[1]))
    offset_center_to_pivot = pygame.math.Vector2(pos) - image_rect.center
    # roatated offset from pivot to center
    rotated_offset = offset_center_to_pivot.rotate(-angle)
    # roatetd image center
    rotated_image_center = (pos[0] - rotated_offset.x, pos[1] -
rotated_offset.y)
    # get a rotated image
    rotated_image = pygame.transform.rotate(image, angle)
    rotated_image_rect =
rotated_image.get_rect(center=rotated_image_center)
    self.screen.blit(rotated_image, rotated_image_rect)



def blitProjectile(self, gun):
    for bullet in gun.bullets:
        x, y, angle = bullet.getBulletCoordinations()
        x = x + (gun.vel * math.cos(math.radians(angle))) # calculating the
right x, y of the rifle, and also moooving it forward
        y = y - (gun.vel * math.sin(math.radians(angle)))
        if x < 0 or x > 1000 or y < 0 or y > 600: # if a bullet is out of
range, remove it from the screen
            gun.bullets.pop(gun.bullets.index(bullet))
        else:
            bullet.setBulletCoordinations(x, y)
        self.blitRotate(bullet_pic, (x + 8, y + 3), (16, 6), angle)
```

קטע קוד 3:

עוד קטע קוד הלקוח מהמחלקה screen, הפעולה מוסיפה sprite חדש למאגר הsprites, אם הsprite כבר קיים אז צריך להחליף אותו ב sprite עם המיקום המעודכן אך אין צורך להוסיף אחד חדש למאגר כדי לא ליצור עומס של מספר רב של sprites שונים

```
def addsprite(self, new_sprite):
    flag = True
    for sprite in self.sprites: # iterating through sprites
        if sprite.id == new_sprite.id: # if sprite does exist, replace old
one
            print("replacing")
            self.sprites[self.sprites.index(sprite)] = new_sprite
            flag = False
    if flag:
        self.sprites.append(new_sprite)
```

קטע קוד 4:

קטע קוד זה לקוח מהclient, והוא קטע הקוד שיוצר תהליכון וממתין למידע מהserver לגבי הsprites השונים

```
class ThreadRecv(threading.Thread):
    def __init__(self, client_socket, screen):
        threading.Thread.__init__(self)
        self.client_socket = client_socket
        self.screen = screen

    def run(self):
        while True:
            try:
                print("waiting for message")
                message = pickle.loads(self.client_socket.recv(8192))
                print("sprite added")
                self.screen.addsprite(message) # Adding the sprite that was
sent
            except Exception as e:
                print(e)


class ClientGame(object):
    def __init__(self, screen):
        try:
            self.client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
            self.client_socket.connect((SERVER_IP, PORT))
            self.screen = screen
            self.id = self.client_socket.recv(1024).decode() # Getting the
client id
            ThreadRecv(self.client_socket, self.screen).start() # Creating
thread waiting for messages
        except Exception as e:
            print(e)
```

קטע קוד 5:
לקוח מהmain, קטע הקוד של הפעולה שבודקת ומנהלת את הinput מהמקלדת

```
def check_for_input()
    global player1, keys, client, shootloop
    if keys[pygame.K_d]: #If any of these keys was pressed, sent the
updated sprite to server
        player1.right()
        client.send_message(player1)
    if keys[pygame.K_a]:
        player1.left()
        client.send_message(player1)
    if keys[pygame.K_w]:
        player1.up()
        client.send_message(player1)
    if keys[pygame.K_s]:
        player1.down()
        client.send_message(player1)

    if shootloop > 0: # delaying the bullets
        shootloop -= 1

    if keys[pygame.K_SPACE] and shootloop == 0:
        player1.addBullet()
        shootloop = 80

    if keys[pygame.K_r] and player1.reloaded == False: # If a player is not
reloading and R is pressed
        player1.reload()
        client.send_message(player1)
```

קטע קוד 6:
קטע הלקוח מהמחלקה sprite, אשר מנהלות את מיקום השחקן וכיוון הסתכלותו
הפעולה הראשונה מחשבת את הזווית בין מרכז השחקן לבין העכבר, ושאר הפעולות נועדו להזיז את השחקן לכל אחד מצדדי המסך במידת הצורך, ולהזיז את המשתנה הרץ אשר נועד להדפיס אנימציית תנועה, אלא אם המשתמש טוען את נשקו ואז אין לשנות את המשתנה הרץ.

```
def calculateAngle(self, mouse_x, mouse_y):
    angle_x = mouse_x - (self.getx() + 76) # The distance between the mouse
and the middle of the sprite
    angle_y = (self.gety() + 50) - mouse_y
    self.angle = math.degrees(math.atan2(angle_y, angle_x))

def right(self):
    self.x += self.vel
    if not self.reloaded:
        self.walkcount += 1


def left(self):
    self.x -= self.vel
    if not self.reloaded:
        self.walkcount += 1
```

```
def up(self):
    self.y -= self.vel
    if not self.reloaded:
        self.walkcount += 1

def down(self):
    self.y += self.vel
    if not self.reloaded:
        self.walkcount += 1
```

קטע קוד 7:

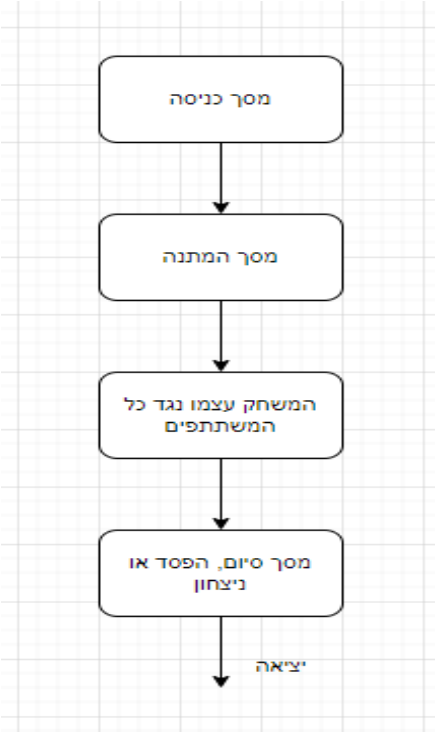
קטע הקוד זה לקוח גם הוא מהמחלקה sprites, קטע זה נועד לשימוש עבור המחלקה screen כאשר היא מדפיסה למסך את ההדמות. הפעולה מסתכלת על מצב הדמות (אם היא מטעינה או לא) ומחזירה למבקש גם את רשימת התמונות בה אחת מן התמונות נועדה להדפסה וגם את האינדקס של התמונה הנכונה באנימציית ה"סטופ מושן" של תנועת הדמות או של טעינת הרובה

```
def getimage(self):
    global images, reloadImages
    if self.walkcount // 20 == len(images) - 1: # walkcount surpassed the
limit, reseting it back to 0
        self.walkcount = 0
        if self.reloaded: # finished reloading
            self.reloaded = False
            self.gun.reloadAmmo()

    if not self.reloaded:
        return images, self.walkcount // 20

    else:
        self.walkcount += 1
        return reloadImages, self.walkcount // 20
```

פרק ו' – Lethal Justice – מדריך למשתמש

יש לוודא שעל המחשב מותקנים קבצי הלקוח, python (רצוי מגרסת 3.7 ומעלה), הספריות בהן נעשה שימוש, ושהמחשב נמצא על אותה רשת אינטרנט כמו השרת.

להלן תרשים הזרימה של המשחק, וכל התהליכים שעוברים מהרצת המשחק עד לסיומו:



לאחר שהרצתם את המשחק מופיעה בפניכם אפשרות להכניס את שמכם, לאחר הכנסת השם תועברו למסך הבא:[1]

על מנהל השרת (שזה אני), להתחיל את המשחק, בזמן זה תמתינו, אבל אל תהיו שאננים, כי המשחק המותח עומד להתחיל ואתם רוצים להיות מרוכזים מאוד.

כאשר תתחילו את המשחק תשוגרו למפה עם שחקנים אחרים, בזמן הקצוב (5:00) עליכם להרוג כמה שיותר שחקנים, כל הריגה תזכה אתכם בנקודה, נהרגתם? לא נורא, זהו לא סופכם, אתם תשוגרו למקום אקראי במפה בו תוכלו להמשיך במסע ההרג שלכם.

המקשים כלל לא מסובכים, בעזרת W,A,S,D תזיזו את דמותכם במרחב הדו מימדי, כמו כן היא תסתכל על דמותכם בכל עת. בעזרת הכפתור Space אתם תירו בנשקכם, שימו לב שיש לכם מספר כדורים מוגבל – 30, כאשר הוא אוזל לא תוכלו לירות עד שלא תטעינו את נשקכם בכפתור R. בזמן זה לא תוכלו לירות, לכן חשוב לשמור את הכדורים לזמן החשוב ביותר.

זהו, פשוט לא? זה כל מה שאתם צריכים לדעת במסעכם לתת צדק קטלני לכלל האויבים, קדימה תשחקו!

[1] – אם זה לא חיבר אתכם לשרת, תתקשרו למספר 055-662-4022

פרק ז' – Lethal Justice – מבט אישי

כאשר קיבלתי את המשימה לעשות פרוייקט – ישר ידעתי שאעשה משחק, נפלא! אמרתי לעצמי, מטלה בנושא שאני אוהב, וכך באמת היה! ישר עלה לי רעיון למשחק, והתחלתי לבצע אותו, כבר בתחילת השנה הפרוייקט שלי הראה התקדמות מופלאה, והה נראה שעתיידו מזהיר, אך מאוד מהר הבנתי שזה לא ככה, אמא שלי נפטרה והפסקתי לעבוד על הפרוייקט, הזנחתי אותו והשארתי אותו במגירה, מה שגרם לעיכוב רצינ ישל הפרוייקט וסחיבתו עד מאוחר.

בתיכנון המקורי הפרוייקט היה אמור להכיל עוד המון אלמנטים כמו רכבים, מסוקים, עוד המון נשקים וכו', אך בגלל האירוע האישי, כל התיכנונים האלה לא יצאו לפועל.
הקושי העיקרי שלי היה המנטלי – הייתי צריך להתגבר עליו ועל אף המצב רוח הירוד, לעשות את הפרוייקט ולהצליח בו, וכך היה.
מצאתי הנאה וניחום בפרוייקט הזה, שהוציא אותי ממצב הרוח המדוכדך ונתן לי דרייב להצלחה, ההשקעתי בו מאוד, דבר שהסיח את דעתי רבות מהמצב העגום ותרם לי רבות.

המוצר הגמור אומנם לא תאם לכל תיכנוניי ושאיפותיי השונות, אך בסופו של דבר, הצלחתי להרים משחק מתפקד, פועל ומהנה, ויותר חשוב מכך, למדתי רבות על התחום והחכמתי בו.

נתקלתי בפיתוח בעוד בעיות רבות, בעיקר פיתוח צד השרת וכל התקשורת, להבין לעומק כיצד תהליכונים עובדים וכיצד לבנות תקשורת טובה ונכונה בין שרתים שונים היה דבר קשוח מאוד להבנה והכנה, נעזרתי רבות באתרים ופלטפורמות רבות כמו:
-youtube
-fxp
-stackoverflow
על מנת להכין את שרת והתקשורת לקוח בצורה היעילה והנכונה ביותר, ויותר חשוב מכך, להבין את הקוד עד לעומקו.

כמו כן חלק מהאלגוריתמים של המשחק לא היו פשוטים, לקח לי זמן לחשוב על דרך יצירתית להדפיס את כל המשתמשים על המסך- כיצד לעדכן כל sprite לאחר תזוזתו, מה שגרם לי לחשוב על הפתרון היצירתי של מספר סידורי לכל אחד מהקליינטים.
או לחשוב על אלגוריתם הסיבוב של הדמות בשימוש של טריגונומטריה ווקטורים מתמטים.

על עף שאני מרוצה מפרוייקט זה, יש לו עוד המון דברים שהייתי רוצה להוסיף שהיו יכולים לשפר מאוד את חווית המשחק.

דבר ראשון, יהיה מערכת כניסה עם שם משתמש וסיסמה, ובכך משתמשים שונים יוכלו לשמור את השיא שלהם, כמות הניצחונות, ועוד סטטיסטיקות שונות, את פרטי שם המשתמש והסיסמה אף אצפין בשימוש במפתח כמו md5.

דבר שני, הייתי רוצה להרחיב את סוגי הנשקים, שכל נשק יהיו תכונות מיוחדות משלהם, יורידו מספר חיים שונה בפגיעה, מהירות כדור שונה, כמות כדורים במחסנית, ועוד תכונות שונות ומגוונות,
אולי גם אוכל לאפשר שימוש בכמה נשקים בו זמנית, או סכין למשל.

כמו כן, הייתי מוסיף רכבים ואולי גם כלי טיס כמו הליקופטר,
הייתי מגדיל את המפה ומתאים אותה למשחק עם כלי רכב, ששחקנים שונים יוכלו להתנייד במהירות בעזרת מכוניות שונות ולדרוס משתמשים אחרים, או אולי לטוס בהליקופטר ולהפציץ מהשמיים דמויות שוונות.

לבסוף, אני מרוצה מהפרוייקט שלי, בהתחשב בנסיבות, יצא לי פרוייקט מוצלח, מהנה, והכי חשוב, חוויתי, למדתי ממנו רבות, והוא גרם לי לחשוב בחיוב על העתיד ועל הפרוייקטים שצפונים לי בעתיד.

פרק ח' – Lethal Justice - ביבליוגרפיה

שימוש נרחב מאוד ב:stackoverflow.com
למשל באשכול הבא: https://stackoverflow.com/questions/35879096/pickle-unpicklingerror-could-not-find-mark

ויקיפדיה:
https://he.wikipedia.org/wiki/TCP/IP

https://he.wikipedia.org/wiki/%D7%AA%D7%94%D7%9C%D7%99%D7%9B%D7%95%D7%9F

דווקומנטציה של ספריות פייתון:
https://docs.python.org/3/library/pickle.html

https://docs.python.org/3/library/socket.html