ספר פרויקט

זירת משחק בין בוטים

שם התלמיד: אנר מור

ת"ז: 329159040

שם המנחה: איתי בן סעדון

תיכון קלעי גבעתיים

יוני 2024

תוכן עניינים

מבוא

ייזום הפרויקט

המערכת מציעה לשחקנים לפתח ולכתוב בוטים משל עצמם ולהתחרות איתם נגד בוטים אחרים כאשר כל שחקן יכול לפתח אסטרטגיות משחק ייחודיות. המערכת מציעה לשחקנים פונקציות רבות ומגוונות, מה שהופך את תהליך כתיבת הבוט לפשוט מאוד ומאפשר לשחקנים להתמקד באסטרטגיה אותה רוצים לממש, במקום בסיבוכיות של כתיבת בוט במצבים אחרים. השחקנים יכולים לצפות במשחקים בזמן אמת ממבט-על ולראות את הניקוד שלהם ושל היריב, את המהלכים של כל אחד ואף סטטיסטיקות המתעדכנות תוך כדי המשחק.

הפרויקט מיועד עבור חובבי משחקי אסטרטגיה ואנשים המתעניינים בתכנות, בין אם הם בעלי ניסיון רב או בעלי הבנה בסיסית. במהלך תכנון הפרויקט היה לי חשוב שמתכנתים מתחילים יוכלו להתחרות באופן שוויוני במתכנתים ותיקים וכי עיקר ההצלחה תתבסס על חשיבה יצירתית ואסטרטגית.

המשחק בו עוסקת המערכת מדמה את דילמת האסיר (**Prisoner's dilemma**) - סוגיה מפורסמת בתורת המשחקים. הדילמה מתארת מצב בו שני אסירים (או במקרה שלנו - השחקנים) צריכים להחליט האם לבטוח או לבגוד זה בזה, כאשר אין להם דרך לתאם אחד עם השני את המהלכים. יש שלושה מצבים: הראשון - אם שניהם יבטחו זה בזה, אזי הם יצליחו במידת מה. השני - אם הם יבגדו זה בזה, אז שניהם ייענשו במידת מה. השלישי - אם האחד יאמין והשני יבגוד, אז הבוגד יזכה בכל והאחר יענש בחומרה. בכל בחירה יש סיכון - שני האסירים רוצים להשיג את התוצאה הטובה ביותר עבור עצמם, אך אם שניהם יבחרו בכך, שניהם יפסידו. כלומר, הדילמה מדגימה מצב בו התנהגות רציונלית לחלוטין, מנקודת מבטו של הפרט, תוביל לתוצאה שאינה אופטימלית לצמד, ואילו תוצאה אופטימלית לצמד תביא לתוצאה בינונית עבור הפרט.

הצורה הבסיסית בה נהוג להציג את הדילמה היא זו: שני עבריינים נעצרו ונלקחו לחקירה במשטרה. כל אחד נחקר בנפרד, אך מתוך חוסר ראיות החוקר מציע לכל אחד מהם להעיד נגד רעהו בתמורה לעונש מופחת. אם שני האסירים יקבלו את הצעת החוקר, ייכנס כל אחד מהם לכלא לחמש שנים, ואם רק אחד מהם יעיד ורעהו ישתוק, העד יצא מיד לחופשי וחברו ייכלא ל-15 שנה. גם כאן ניתן לראות כי על מנת להגיע לתוצאה הטובה ביותר עבור שניהם עליהם לשתף פעולה, אך אם רק אחד מהם יבחר שלא לשתף פעולה הוא ירוויח הרבה יותר.

בעוד שבדוגמה לעיל לא ניתן להצביע על שיטה בה ניתן להגיע בוודאות לתוצאה מיטבית, אילו היה אפשר לחזור על הבחירה פעמים רבות האסירים היו פועלים אחרת, מנסים להבין את דפוס הפעילות של האסיר האחר ואף מנסים להגיע לתוצאה הטובה ביותר עבור עצמם, במקום לתוצאה הטובה ביותר עבור שניהם. למקרה זה קוראים דילמת האסיר האיטרטיבית (**Iterative prisoner's dilemma**). כלומר, במקום בחירה אחת, כעת האסירים צריכים לבחור פעמים רבות האם לשתף פעולה או לא ולהגיע לתוצאה הטובה ביותר. המשחק בו השחקנים בפרויקט שלי ישחקו בו הוא בעצם דילמת האסיר

האיטרטיבית, מכיוון שהוא כולל הרבה סיבובים והשחקנים יכולים לבחור בכל תור בחירה שונה, תוך כדי צבירת נקודות.

על פניו נראה שגם בדילמת האסיר האיטרטיבית אין דרך ברורה להגיע לתוצאה מיטבית ולצבור את הכמות הגבוהה ביותר של נקודות. הרי שמן הסתם שחקן שבוחר לבגוד בכל סיבוב ישיג הרבה נקודות אילו משחק נגד שחקן המשתף פעולה בכל סיבוב, אך תחילה לא ברור עד כמה ניתן להרחיק לכת ולהגיע לאסטרטגיה שלאחר משחקים נגד הרבה שחקנים אחרים תגיע לתוצאה הטובה מכולם.

המחקר המפורסם ביותר בתחום הוא הניסוי שערך פרופסור רוברט אקסלרוד מאוניברסיטת אן ארבור מישיגן בשנת 1980. בניסוי זה התבקשו מומחים ממגוון תחומים (כלכלה, פסיכולוגיה, מדע המדינה, ביולוגיה, מדעי המחשב ועוד) לשלוח תוכניות מחשב שישחקו זו נגד זו בדילמת אסיר איטרטיבית במשך 200 משחקים לכל זוג תוכניות. התוכנית שתוכרז כזוכה היא זו שתזכה ברווח המצטבר הגבוה ביותר בכל 200 המשחקים. בכל סיבוב בין שני שחקנים חולקו נקודות באופן הבא:

- אם שניהם משתפים פעולה כל אחד יקבל 3 נקודות.
- אם שניהם יבגדו יקבל כל אחד נקודה אחת.
- אם אחד משתף פעולה והשני בוגד- הבוגד יקבל 5 נקודות ומשתף הפעולה לא יקבל נקודות בכלל.

חלק מהאסטרטגיות שהציעו המומחים הן:

- בגידה קבועה: לא משנה מה קורה במשחק, התוכנית תבגוד בכל סיבוב
- שיתוף פעולה קבועה: לא משנה מה קורה במשחק התוכנית תשתף פעולה בכל סיבוב
- אקראיות: התוכנית בוחרת אקראית בכל סיבוב האם לשתף פעולה או לבגוד
- נקמה: התוכנית מתחילה בשיתוף פעולה, אך אם היריב בוגד פעם אחת אז התוכנית תבגוד בכל סיבוב עד תום המשחק

ניתן לשייך לכל אסטרטגיה אופי: אם האסטרטגיה נוטה לחזור לשתף פעולה אחרי שהיריב בוגד ניתן לתאר אותה כאסטרטגיה "סלחנית", ואילו אם האסטרטגיה נוטה לבגוד ביריב אחרי שהוא משתף פעולה ניתן לתאר אותה כ"נקמנית". נוסף על כך, אם אסטרטגיה נוטה לבגוד ביריב כאשר אינו מצפה לכך ניתן לתאר אותה כ"תחמנית".

בתום הרצת הניסוי האסטרטגיה המנצחת הייתה "מידה כנגד מידה" (**Tit for tat**). מידה כנגד מידה התחילה בשיתוף פעולה בסיבוב הראשון, ולאחר מכן בחרה בדרך שבה בחר היריב בסיבוב הקודם. בזכות כך שלא בגדה ראשונה מיוזמתה אף פעם הצליחה מידה כנגד מידה להשיג את מספר הנקודות הגבוה ביותר מבין כולם במצטבר לאורך התחרות. אין זה אומר שניצחה בכל משחק, למשל במשחק מול אסטרטגיה של בגידה קבועה היא תפסיד כבר בסיבוב הראשון.

"מידה כנגד מידה" אולי נדמית כאסטרטגיה די סלחנית, אך ניתן לחשוב על אסטרטגיות אף יותר סלחניות. כאשר פרופסור אקסלרוד בחן אסטרטגיה נוספת אשר פועלת כמו "מידה כנגד מידה" אך

בוגדת רק כאשר היריב בוגד פעמיים ברצף, למעשה, "מידה כנגד 2 מידות", גילה שהייתה מנצחת בתחרות אילו הייתה מתחרה.

נקודה למחשבה

בעקבות הניסוי של אקסלרוד חשבתי כך: נגדיר אסטרטגיה של "מידה כנגד N מידות" שהיא "מידה כנגד מידה" (**Tit for tat**) כאשר N שווה ל-1 ו"מידה כנגד 2 מידות" (Tit for two tats) כאשר N = 2. אם במקרה של הטורניר של אקסלרוד בו "מידה כנגד 2 מידות" הייתה מוצלחת יותר, כלומר כאשר N היה גבוה יותר, מדוע לא להגדיל את N אף יותר?

אם ידוע שניתן להגדיל את N ולייצר בכך אסטרטגיה סלחנית יותר, מדוע לא לעשות כך? האם בהכרח נקבל אסטרטגיה מוצלחת יותר? מה ה N האופטימלי?

על מנת לענות על כך נצטרך להתחשב בכמה גורמים. ראשית, נגדיר את מספר הסיבובים במשחק כ R. ככל ש N שואף ל R האסטרטגיה "מידה כנגד N מידות" תהפוך יותר ויותר זהה לאסטרטגיה של שיתוף פעולה קבוע. מכיוון שבניסוי של אקסלרוד אסטרטגיה זו הגיעה לתוצאות בינוניות, ניתן להבין שאי אפשר להגדיל את N ללא הגבלה וכי יש ל N גבול עליון הנמוך מ R, וכנראה רחוק ממנו.

נבחן לעומק יותר את התכונות של האסטרטגיה:

"מידה כנגד מידה" (N = 1)

- נוקמת מיד לאחר בגידה. אפקטיבית בכך ששומרת על שיתוף פעולה הדדי עם היריב ומענישה על בגידה באופן מיידי.
- יוצרת השלכות ברורות לבגידה ובכך מונעת מהיריב להפריז בכך.

"מידה כנגד 2 מידות" (N = 2)

- סלחנית יותר, שומרת על רצף שיתופי פעולה ארוך יותר במקרים בהם היריב מנסה לבגוד פעמים אחדות לאורך המשחק.
- לקיחת סיכון גבוהה יותר מכיוון שניתן לנצל אותה לרעה ולבגוד מספר רב של פעמים לאורך המשחק ללא השלכות.

"מידה כנגד 3 או יותר מידות" (N > 2)

- כאשר N גדל האסטרטגיה נהיית סלחנית יותר, מה שמשאיר מקום לניצול גדול יותר על ידי אסטרטגיות בוגדניות.
- מול אסטרטגיות "נחמדות", אין הרבה משמעות ל N גבוה.

מסקנה

ה N האופטימלי נוטה להיות נמוך. מול אסטרטגיות בוגדניות לא יהיה ניתן להגיב מהר כאשר N גבוה, מה שמשאיר מקום ל"ניצול", ואילו מול אסטרטגיות הנוטות לשתף פעולה יהיה ניתן ליצור רצף ארוך של שיתופי פעולה, אך גם כאן יש מקום לניצול אם היריב בוגד פעמים אחדות על פני סיבובים רבים.
אסטרטגיות נוטות לשנות את האופי שלהן בהתאם למספר הסיבובים ואירועים קודמים במשחק. הצלחה של אסטרטגיה זו או אחרת תלויה מאוד באופי האסטרטגיות בהן מתחרה ולא ניתן להגדיר אסטרטגיה שתהיה המוצלחת ביותר בכל תחרות.

סקירת מערכות דומות אחרות

כיום קיימים מספר משחקים ברשת המציעים למשתמשים לפתח בוטים משל עצמם ולהתחרות איתם, בין אם זה משחקי משחקי קלפים, מלחמה, או משחקי תורות. אחד מהם אף מוצע על ידי משרד החינוך לתלמידי תיכון והתנסיתי בו במהלך הלימודים בכיתה יא, דבר שעזר לי לתכנן את הפרויקט הזה. אולם, הרבה מן המשחקים האלה מצריכים ניסיון מוקדם נרחב בתכנות, או שצריך להתבסס בהם על הרבה קוד קיים שמוצע על ידי המערכת.

אפיון הפרויקט

סקירת הטכנולוגיות בפרויקט

טכנולוגית הפרויקט הינה שילוב של מספר טכנולוגיות אשר קיימות כיון ונעשה בהם שימוש נרחב.
הפרויקט שלי בנוי בשפת התכנות המוכרת Python, על בסיס מערכת הפעלה של Windows. בפרויקט שלי אני משתמש במספר ספריות וכלים כגון:

- **Socket** – תעלות המשמשות בכדי ליצור קשר בין השרת לבין הלקוח
- **arcade** - ספרייה אשר מאפשרת להציג את המשחק למשתמש באופן שוטף ולעדכן אותו בזמן אמת
- **threading** - ספרייה בפייתון אשר מאפשרת הרצה של מספר תהליכים בו זמנית במקום אחד אחרי השני, דבר אשר מייעל מאוד את ניצול המשאבים של המחשב.
- **OOP** - בנייה של הפרויקט לפי עצמים ואובייקטים (תכנות מונחה עצמים)

תיחום הפרויקט

הפרויקט עוסק בתחום הרשתות וכל התקשורת בין השרת ללקוח התבצע באמצעות שימוש במודול socket בpython. כחלק מהשימוש בתחום הרשתות, יש גם לפרויקט שרת המבוסס על socket ומקבל

ושולח נתונים. נוסף על כך, בפרויקט יש שימוש בתכנות מונחה עצמים (OOP) כאשר כל לקוח מהווה אובייקט של שחקן.

הפרויקט שלי עושה שימוש בריבוי הליכונים (multithreading) בצד השרת כאשר עבור כל משחק נפתח הליכון (thread) חדש שמטפל בו ושומר את הפרטים הרלוונטים.

מסמך אפיון

1 פונקציונליות

API למימוש הבוטים שיכלול פונקציות לבניית הבוט כגון פונקציות לקבלת המהלכים הקודמים של השחקן עצמו ושל יריבו, כמות הסיבובים שנותרו עד סיום המשחק, סוגי המהלכים ששוחקו על ידו ועל ידי היריב ועוד. פונקציות אלו יהפכו את כתיבת הבוט לפשוטה ואינטואיטיבית. בכך יתאפשר לשחקנים להתמקד באסטרטגיה שרוצים להשתמש בה.

2.2 קהל היעד

קהל היעד של המוצר הוא חובבי משחקי אסטרטגיה ומתכנתים בפייתון בכל הרמות המעוניינים להתנסות בכתיבת בוטים ופיתוח אסטרטגיות משלהם, בנוסף אנשים המתעניינים בתורת המשחקים ורוצים לחקור אותה בעצמם.

2.3 אילוצים עיקריים

המערכת צריכה לתמוך במספר לקוחות בו זמנית בכל טורניר שמקיימת ולדעת לחלק אותם למשחקונים באופן הוגן, כמו כן תצטרך להתמודד עם מצב בו לקוח מתנתק באופן פתאומי או שנופל לו האינטרנט ומקרים בהם נדרש להריץ קוד לא תקין של בוט או לא עומד בדרישות המערכת.

2.4 הנחות ותלויות

- מניחים כי הלקוח מודע למספר הסיבובים אותם נדרש לשחק ולא יפסיק את המשחק בפתאומיות.
- הלקוחות החלטיים ושולחים לשרת מהלך ברור בכל סיבוב.
- מספר הסיבובים בכל משחק הוא סופי וידוע לשרת.
- אין הפרעה לתקשורת בין השרת ללקוח וידוע כי המסר מהלקוח הוא באמת המסר שהתקבל אצל השרת.

3. דרישות מפורטות

דרישות של ממשקים חיצוניים

ממשק API שיכלול פונקציות לכתיבת הבוטים ופונקציות להצגת המשחק, ביניהם:

- ממשק GUI על מנת להציג את המשחק באופן גרפי דרך הלקוח, כך שכל משתמש יוכל לראות את המשחק מנקודת מבטו וממבט על.

- אפשרות לשחק משחק נגד השרת מול בוטים ששמורים אצלו, לבקשת המשתמש.

- הצגת סטטיסטיקות בתום התחרות למשל אחוז הזכיות לפי "אופי" המשחק של השחקנים השונים, מאזן הניצחונות וההפסדים של כל שחקן, ממוצע הנקודות לאחר כל המשחקים, המשחק הטוב ביותר לאורך התחרות ועוד.

- הפסד טכני לשחקן שם בעיות חיבור או שלא מגיב לאחר פרק זמן מסוים.

דרישות לא פונקציונליות

המחשבים שידרשו להריץ את הבוטים יהיו בעלי יכולת להריץ אותם.

דרישות בסיס נתונים

יכולות האחסון והזיכרון הנדרשות תלויות בעיקר בכמות השחקנים ובנתונים שישמרו עבור כל אחד, למשל שם הבוט או המהלכים שביצע. מכיוון שהמערכת לא תצטרך לתמוך בכמות לקוחות גדולה מאוד (למשל, מעל כמה עשרות שחקנים) ניתן להניח כי יכולות האחסון והזיכרון הנדרשות הינן בסיסיות.

תיאור תחום הידע

פירוט היכולות מצד הלקוח

- שם היכולת: **יצירת שחקן חדש**
- מהות: יצירת שחקן חדש עם שם ותכונות והתחברות לשרת
- אוסף יכולות נדרשות:
  - שמירת השם כתכונה
  - שליחת נתונים
  - קבלת תשובה מהשרת והדפסה שהשחקן מחובר לשרת
  - פתיחת החלון הגרפי
- אובייקטים נחוצים: תקשורת, ממשק משתמש

- שם היכולת: **קבלת אות התחלת סיבוב**
- מהות: סימן ברור להתחלת סיבוב וידיעה מתי לשלוח את המהלך לשרת
- אוסף יכולות נדרשות:

- קליטת נתונים
- עדכון תכונות
- בדיקת תקינות
- אובייקטים נחוצים: תקשורת, ממשק משתמש

- שם היכולת: **קבלה ועדכון ניקוד**
- מהות: יש לעדכן את הניקוד שהשחקן קיבל בכל סיבוב
- אוסף יכולת נדרשות:
  - קליטת נתונים
  - פענוח
  - בדיקת תקינות
  - עדכון תכונות
  - הצפנה
  - שליחת נתונים
  - הצגת המידע למשתמש
- אובייקטים נחוצים: תקשורת, פענוח, ממשק משתמש

- שם היכולת: **שליחת הודעה**
- מהות: על מנת לתקשר עם השרת יש לשלוח הודעות ברורות כולל סוג ותוכן
- אוסף יכולת נדרשות:

    - פענוח
    - שליחה לשרת המרכזי
    - שליחת נתונים
    - הצפנה
    - הצגת המידע למשתמש
- אובייקטים נחוצים: תקשורת, פענוח, ממשק משתמש

- שם היכולת: **יצירת גרפיקה**
- מהות: הצגת המשחק למשתמש בצורה קלה להבנה
- אוסף יכולת נדרשות:
    - קליטת נתונים
    - פענוח
    - עדכון תכונות
    - הצגת נתונים טכניים למשתמש
    - שימוש בספרייה ייעודית
    - הצגת המידע למשתמש
- אובייקטים נחוצים: תקשורת, פענוח, ממשק משתמש

פירוט היכולות מצד השרת

- שם היכולת: **חיבור לקוח חדש**
- מהות: חיבור לקוח חדש למערכת
- אוסף יכולת נדרשות:
    - האזנה
    - התחברות המשתמש
    - קליטת כינוי מלקוח
    - פענוח
    - פתיחת Thread טיפול בלקוח
    - שימוש בספרייה ייעודית

- אובייקטים נחוצים: תקשורת, פענוח, מערכת הפעלה

- שם היכולת: **יצירת משחק חדש**
- מהות: יצירת משחק עבור 2 שחקנים
- אוסף יכולת נדרשות:
  - התחברות המשתמש
  - פענוח
  - פתיחת Thread לניהול המשחק
  - שליחה מידע ללקוח
  - קבלת מידע מהלקוח
  - עדכון מידע
  - שימוש בספרייה ייעודית
  - קביעת מנצח
- אובייקטים נחוצים: תקשורת, פענוח, מערכת הפעלה

- שם היכולת: **חישוב הניקוד**
- מהות: חישוב הניקוד עבור כל שחקן על פי נתונים בכל סיבוב
- אוסף יכולת נדרשות:
  - חישוב
  - שימוש במבני נתונים
  - שמירת מידע
- אובייקטים נחוצים: מערכת הפעלה

- שם היכולת: **בדיקת חוקיות של הודעה**

- מהות: בדיקת החוקיות של הודעה מסוימת וטיפול בה בהתאם
- אוסף יכולת נדרשות:
  - האזנה
  - בדיקת תקינות מול בסיס הנתונים
  - שליחת מידע ללקוח
  - שימוש בSocket
  - פענוח
- אובייקטים נחוצים: תקשורת, פענוח, מערכת הפעלה

- שם היכולת: **בניית טורניר**
- מהות: בניית טורניר בין כל השחקנים על פי מבנה של "ליגה"
- אוסף יכולת נדרשות:
  - שימוש באלגוריתם
  - התחברות המשתמש
  - שמירת נתונים
  - שליחת מידע ללקוח
  - פתיחת Thread טיפול בלקוח
- אובייקטים נחוצים: מערכת הפעלה

ארכיטקטורת המערכת

עיצוב

תיאור הטכנולוגיה הרלוונטית: הטכנולוגיה הרלוונטית עבור הפרויקט שלי היא:

- **מערכת הפעלה Windows**. הפרויקט שלי בנוי כך שיפעל במערכת ההפעלה Windows.
- **שפת תכנות Python** - כל הקוד שלי כתוב בשפת התכנות פייתון. פייתון היא שפה פשוטה למדי אשר מאפשרת כתיבת קוד נוחה וקלה להבנה.
- **תקשורת Socket** - התקשורת בין השרת ללקוח בפרויקט שלי מתבצעת על גבי סוקטים אשר מחוברים לכתובת IP ספציפית ול-PORT ספציפי במחשב.
- **גרפיקת משתמש arcade -** זהו ממשק GUI אשר מאפשר להציג בצורה נוחה את נתוני המידע כמו לוח המשחק וטבלת הניקוד.

הצגת זרימת תהליך הרצת המשחק

⚠

תיאור האלגוריתמים המרכזיים בפרויקט

יצירת פורמט "ליגה" עבור הבוטים להתחרות בה: על מנת לבחון את היכולת של כל בוט, מוטב לתת להם להתחרות בפורמט שבו כולם ישחקו נגד כולם במהלך התחרות, בדומה לאופן שבו התבצע הניסוי של אקסלרוד. לכן, לאחר שכל הבוטים התחברו לשרת עליו לצוות צמדים של שחקנים למשחקים בצורה שכל שחקן ישחק נגד כל השחקנים האחרים.

האלגוריתם האחראי לכך מתבצע באופן הבא:

- יצירת רשימה של כל השחקנים
- עבור כל שחקן ברשימה השרת שומר משחק שלו מול כל אחד מהבוטים שאחריו ברשימה
- חזרה על השלב הקודם עבור השחקן הבא ברשימה

לאלגוריתם הזה קוראים "Round robin" והוא הנפוץ ביותר ליצירת תחרות בה רוצים שכל שחקן ישחק נגד כל שחקן אחר לפחות פעם אחת.

חישוב הנקודות על פי קבלת המהלך של כל אחד מהשחקנים: בכל סיבוב הבוטים שולחים לשרת את המהלכים שלהם; שיתוף פעולה או בגידה. השרת צריך לחשב על פי כך את הניקוד שמגיע לכל שחקן, לשמור ולשלוח את הנתון הזה.

האלגוריתם האחראי לכך מתבצע באופן הבא:

- קבלת שני המהלכים
- בדיקה אם שניהם זהים

  אם כן:

  - אם שניהם שיתפו פעולה: נתינה לכל אחד 3 נקודות
  - אם שניהם בגדו: נתינה לכל אחד נקודה אחת

אם לא:

- עבור כל שחקן:
  - אם שיתף פעולה: נתינה של 0 נקודות, אם בגד: נתינה של 5 נקודות

תיאור סביבת הפיתוח

Python: הפרויקט נכתב בשפת התכנות Python, יחד עם הספריות והמודולים הנדרשים לפרויקט שלי. שפה זו היא הנוחה והיעילה ביותר לביצוע ומימוש פרויקט זה.

סביבת הרצה: Windows היא מערכת הפעלה בשימוש נרחב (OS) המספקת סביבה מתאימה לפיתוח תוכנה. מערת זו מציעה קבוצה נרחבת של תוכנות וכלים לפיתוח תוכנה, מה שהופך אותה לבחירה טובה עבור הפרויקט שלי.

כלי דיאגרמה: כדי להמחיש את הארכיטקטורה והזרימה של הפרויקט, אני צריך להשתמש בכלי דיאגרמות. השתמשתי בכלים אלה כדי ליצור דיאגרמות המייצגות את המבנה הכולל, זרימת הנתונים וארכיטקטורת הרשת של הפרויקט שלי.

תיאור פרוטוקול התקשורת

פרוטוקול התקשורת בפרויקט שלי בנוי על פי פורמט של סוג הודעה ותוכן הודעה. כל הודעה שנשלחת בין הלקוח לשרת או בין השרת ללקוח בנויה על פי הפרוטוקול הזה. ישנם קבועים בתוכנית אשר שומרים סוג ספציפי של הודעות, למשל, שליחת מהלך, שליחת ניקוד, הצטרפות למשחק ועוד. כל אחד מהם מכיל ערך מספרי מסוים שמופיע בתחילת כל הודעה ובעזרתו ניתן לדעת את סוג ההודעה. על מנת לממש את הפרוטוקול בניתי פונקציה המקבלת סוג הודעה ותוכן הודעה כפרמטרים, משרשרת אותם ושולחת אותם.

⚠

ישנו פרוטוקול דומה גם עבור שליחה וקבלה של מספרים, אך לכך יש פונקציות אחרות.

מכיוון שמקבל ההודעה מקבל שרשור של סוג ההודעה והתוכן שלה, עליו להפריד ביניהם על מנת להגיע לתוכן. ניתן לעשות זאת באמצעות גישה ל byte הרצוי בהודעה המתקבלת. באופן הזה:

תיאור מסכי המערכת

מסך התחברות: מופיע עד התחברות כל השחקנים והתחלת התחרות.

בתמונה: מסך ההתחברות שיראה הבוט "COOPERATE BOT"

*הערה: תצוגת המסך יכולה להשתנות עד סיום כתיבת הפרויקט ותהיה מעט שונה, אך התכלית נשארת זהה

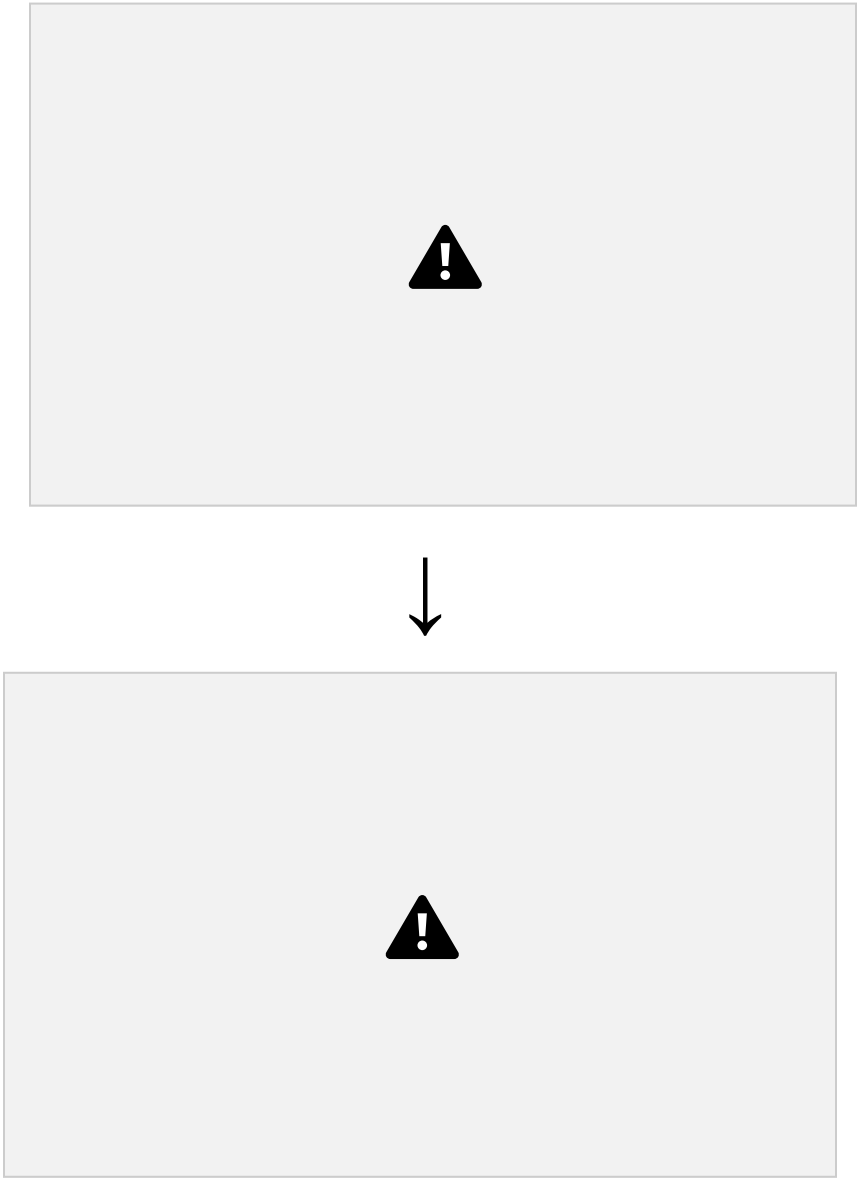
מסך המשחק: המסך שהשחקנים יראו במהלך המשחק. מתעדכן בכל תור לפי המהלכים של השחקנים. עיגול ירוק מסמל שיתוף פעולה ועיגול אדום מסמל בגידה. בצד שמאל מופיעים השמות של הבוטים ובצד ימין הניקוד של כל אחד.

בתמונה: משחק מנקודת המבט של הבוט "ALTERNATE"

*הערה: תצוגת המסך יכולה להשתנות עד סיום כתיבת הפרויקט ותהיה מעט שונה, אך התכלית נשארת זהה

מסך טבלת התחרות:

תרשים זרימה של מסכי המערכת



מימוש הפרויקט

סקירת המודולים והמחלקות המרכיבים את המערכת

קוד הפרויקט מורכב משרת ומ-API לכתיבת הבוטים. בתוכו נמצא ה class בשם gameState והוא מייצג אובייקט של בוט. בתוך ה class נמצא הבנאי שיוצר את האובייקט ומאתחל את התכונות שלו ואת הפעולות שדואגות לחיבור לשרת ותקשורת מולו, מבצעות חישוב ועדכון לנקודות ומייצרות את הגרפיקה. גם השרת משתמש ב-API עבור יישום פרוטוקול התקשורת.

תכונות __init__ של GameState:

פירוט:

• client socket כתובת הסוקט של הבוט. כל התקשורת מול השרת מתבצעת דרכו והוא מאותחל כNone. מקבל לתוכו ערך רק בהמשך התוכנית דרך הפונקציה connect_to_server, עליה אפרט בהמשך.

• round מספר הסיבוב של המשחק. מאותחל כ-0 ומתעדכן עם סיום כל סיבוב.

• my_total_score סכום הנקודות המצטבר של הבוט במהלך המשחק. מאותחל כ-0, ומתעדכן בכל סיבוב עם קבלת הניקוד.

• opponent_total_score סכום הנקודות המצטבר של היריב במהלך המשחק. מאותחל כ-0, ומתעדכן בכל סיבוב עם קבלת הניקוד.

• opponent_last_decision המהלך של היריב בתור הקודם. מאותחל כמחרוזת ריקה ומתעדכן עם סיום כל סיבוב.

• my_last_decision המהלך של הבוט בתור הקודם. מאותחל כמחרוזת ריקה ומתעדכן עם סיום כל סיבוב.

• opponent_decisions_list רשימה השומרת את המהלכים של היריב. מאותחלת כרשימה ריקה ומתעדכנת בסיום כל סיבוב.

• my_decisions_list רשימה השומרת את המהלכים של הבוט. מאותחלת כרשימה ריקה ומתעדכנת בסיום כל סיבוב.

• opponent_decisions_list רשימה השומרת את המהלכים של היריב. מאותחלת כרשימה ריקה ומתעדכנת בסיום כל סיבוב.

• my_name השם של הבוט. שומר בתוכו את הפרמטר של ה class המייצג את השם ולא משתנה במהלך המשחק.

• graphics משמש את השחקן על מנת להחליט אם הוא רוצה לראות את המשחק בצורה גרפית. משמש בעיקר לבנייה של המשחק על מנת להריץ בדיקות ולתקן שגיאות. שווה ל True כברירת מחדל אלא אם השחקן בוחר אחרת.

• is_game_over האם המשחק נגמר. מאותחל כ-False ומתחלף ל- True אם נגמרו הסיבובים, ובכך מסתיים המשחק.

- num_rounds מספר הסיבובים הצפוי למשחק. מאותחל לפי קבוע השומר את הנתון.

מודולים/ מחלקות מיובאות:

• **socket** - מודול הסוקט מספק פונקציונליות רשת ברמה נמוכה, ומאפשר ליצור ולקיים אינטראקציה עם סוקטים. סוקטים משמשים לתקשורת בין מכונות ברשת, ומאפשרים תקשורת רשת באמצעות פרוטוקולים כמו TCP ו- UDP.

• **threading** - מודול ההליכונים מספק יכולות ריבוי פעולות ברמה גבוהה בפייתון. זה מאפשר ליצור ולנהל thread-ים, שהם יחידות ביצוע עצמאיות שיכולות לפעול במקביל במסגרת תוכנית. דבר זה שימושי לטיפול במשימות במקביל ולשיפור ביצועי התוכנית.

• **struct** - ספריית ה-struct מספקת פונקציות להמרה בין ערכי פייתון לבין ייצוגי נתונים בינאריים בסגנון C, מה שמאפשר אריזה ופריקה יעילה של נתונים אל בתים וממנה.

• **arcade** - ספרייה המספקת פונקציות עבור יצירה והדמיה של משחקים דו-מימדיים בצורה פשוטה.

• **time** - ספריית הזמן של פייתון מאפשרת מדידה של זמן, מדידת משך זמן של פעולה ויישום של עיכובים מבוססי זמן.

הפעולות במחלקה GameState

1. שם: connect_to_server

תפקיד ושימוש: חיבור הלקוח והשרת, התחלת מסך פתיחה

טענת כניסה: תכונות המחלקה (self)

החזרה: הsocket של הלקוח

2. שם: get_round_start_signal

תפקיד ושימוש: קבלת סימן לתחילת הסיבוב על מנת לדעת מתי לשלוח את המהלך

טענת כניסה: תכונות המחלקה

החזרה: אין

3. שם: send_decision

תפקיד ושימוש: שליחה של המהלך לשרת

טענת כניסה: תכונות המחלקה, המהלך של הבוט

החזרה: אין

4. שם: update_score

תפקיד ושימוש: קבלת הניקוד מהשרת עבור הסיבוב שנגמר ועדכון תכונות המחלקה הדורשות את זה

טענת כניסה: תכונות המחלקה

טענת יציאה: אין

5. שם: calculate_op_decision

תפקיד ושימוש: מציאת המהלך של היריב בסיבוב הנוכחי

טענת כניסה: תכונות המחלקה, הניקוד שהשחקן קיבל בסיבוב הנוכחי

טענת יציאה: המהלך של היריב בסיבוב הנוכחי (שיתוף פעולה או בגידה)

6. שם: calculate_op_score

תפקיד ושימוש: חישוב ומציאת הניקוד שהיריב קיבל בסיבוב הנוכחי

טענת כניסה: תכונות המחלקה, הניקוד שהשחקן קיבל בסיבוב הנוכחי

טענת יציאה: הניקוד של היריב בסיבוב הנוכחי

7. שם: send_message

תפקיד ושימוש: שליחה של הודעה בשרשור עם הסוג שלה

טענת כניסה: תכונות המחלקה, סוג ההודעה (קבוע המכיל מספר), תוכן ההודעה

טענת יציאה: שליחת ההודעה

8. שם: get_average_score

תפקיד ושימוש: החזרה של הניקוד הממוצע ברגע נתון של משחק

טענת כניסה: תכונות המחלקה

טענת יציאה: החזרת הניקוד הממוצע

9. שם: loading_screen

תפקיד ושימוש: הצגה של מסך ההמתנה עד לחיבור של כל השחקנים. מתחלף למסך המשחק כאשר כל השחקנים מחוברים

טענת כניסה: תכונות המחלקה

טענת יציאה: הצגת מסך ההמתנה

10. שם: create_graphics

תפקיד ושימוש: יצירת הגרפיקה של המשחק - לוח המשחק, שמות השחקנים והמהלכים של כל אחד, מספר הסיבוב והניקוד של כל שחקן. הלוח מתעדכן בכל סיבוב עם הקריאה לפונקציה.

טענת כניסה: תכונות המחלקה

טענת יציאה: הצגת המשחק

הצגת לוח המשחק:

הצגת המהלכים של כל שחקן:

הצגת שמות השחקנים והניקוד שלהם

11. שם: draw_circle

תפקיד ושימוש: ציור עיגול בצבע מסוים. משמש את הפונקציה הקודמת create_graphics על מנת המחשת המהלכים של כל שחקן

טענת כניסה: תכונות המחלקה, מספר הסיבוב, מהלך(מאותחל כ-None) משתנה 'whos_round' המורה לפונקציה אם לצייר את העיגול בשורה של השחקן הצופה או בשורה של היריב, מיוצג על ידי מספר 0 או 1.

טענת יציאה: ציור עיגול על לוח המשחק

בתוך הקובץ bot_api בו נמצאת המחלקה 'GameState' נמצאות עוד פונקציות מחוץ למחלקה המשמשות גם את השרת.

שם: send_int

תפקיד ושימוש: שליחת מספר וסוג מספר על פי פרוטוקול התקשורת

טענת כניסה: סוג המספר, הסוקט של הלקוח, המספר אותו רוצים לשלוח

טענת יציאה: שליחת המספר בשרשור עם הסוג שלו

שם: recieve_integer

תפקיד ושימוש: קבלת המספר והפרדה שלו מהסוג שלו

טענת כניסה: הסוקט של השרת

טענת יציאה: החזרת המספר המקובל

קוד השרת

השרת אחראי לניהול המשחק. לאחר שנפתח השרת מאזין לחיבורים ומחבר שחקנים. לאחר שכמות השחקנים הנדרשת מתחברים מתחיל בהקצאת המשחקים לכל שחקן ומתחיל להריץ את התחרות. עבור כל משחק נפתח תהליכון נפרד שמנהל אותו, כשלבסוף השחקנים ממוינים לטבלה על פי הניקוד שקיבלו לאורך התחרות וניתן להכריז על המנצח.

פונקציות השרת

1. שם: wait_for_players

תפקיד ושימוש: חיבור מספר השחקנים הרצוי והחזרת רשימה שלהם

טענת כניסה: מספר השחקנים הרצוי, הsocket של השרת

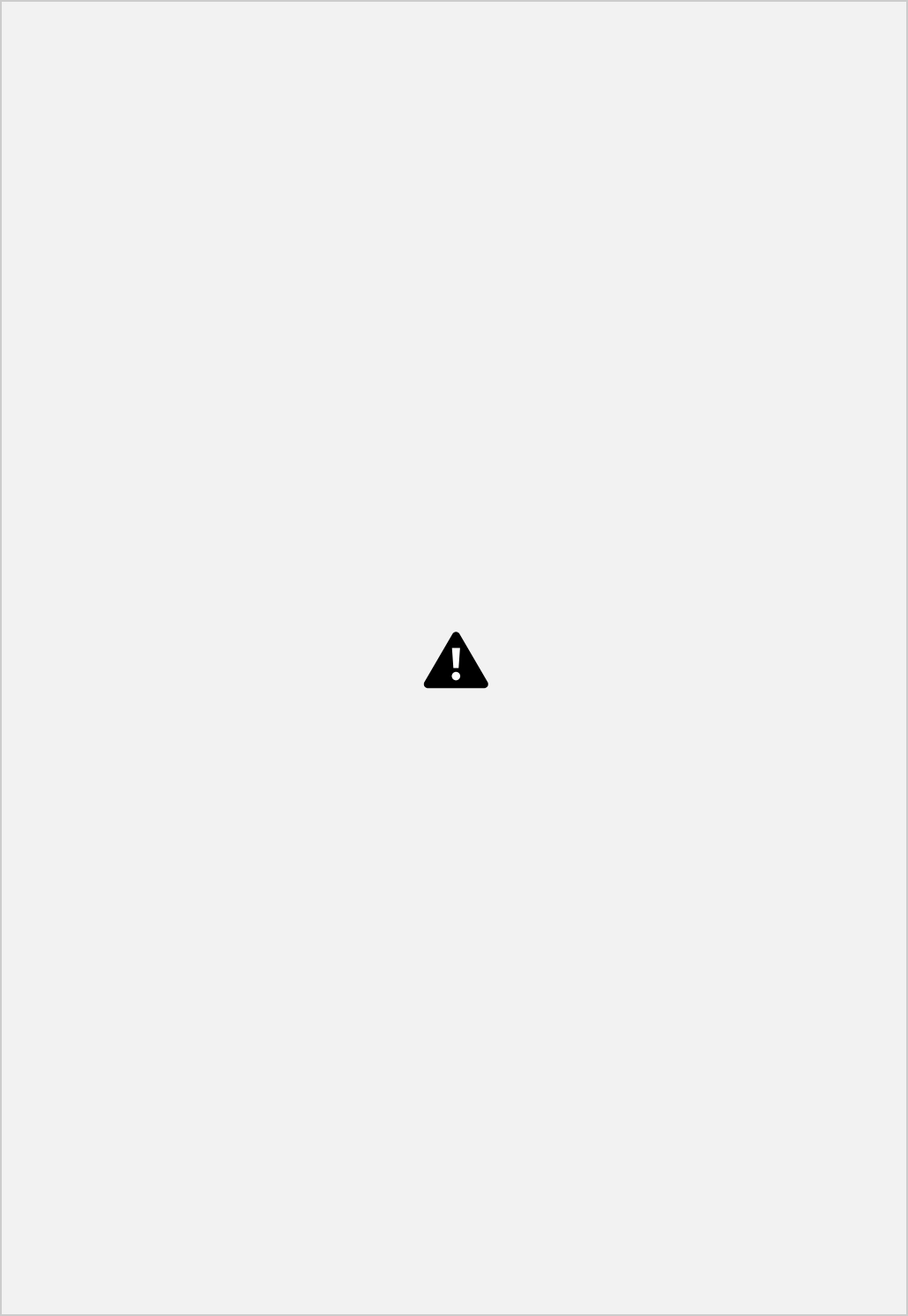
טענת יציאה: החזרת רשימת אובייקטים מסוג tuple כאשר כל איבר ברשימה מצביע על סוקט של שחקן והשם שלו

2. שם: calculate_scores

תפקיד ושימוש: חישוב הניקוד עבור כל שחקן על פי המהלכים שלהם

טענת כניסה: המהלכים של השחקנים כאובייקט של מילון (מהלך של שחקן א: מהלך של שחקן ב)

טענת יציאה: הנקודות עבור כל שחקן כאובייקט מילון (ניקוד של שחקן א: ניקוד של שחקן ב)



3. שם: check_legal_decision

תפקיד ושימוש: בדיקה האם מהלך של שחקן הוא מהלך חוקי, כלומר האם הוא אחד משני המהלכים האפשריים

טענת כניסה: מהלך של שחקן

טענת יציאה: מחזירה את המהלך אם הוא חוקי, אחרת מחזירה None

4. שם: check_legal_message

תפקיד ושימוש: בדיקה האם הודעה שהתקבלה היא אכן ההודעה המצופה

טענת כניסה: סוקט של לקוח, סוג ההודעה המצופה, אורך ההודעה המצופה

טענת יציאה: מחזירה את ההודעה אם הבדיקה עברה בהצלחה, אחרת מחזירה None

5. שם: play_round

תפקיד ושימוש: לדאוג לכל הרצה של סיבוב. הפונקציה שולחת לבוטים הודעה על התחלת סיבוב, מקבלת את המהלכים שלהם, בודקת האם הם חוקיים ושולחת לבוטים את הניקוד שקיבלו.

טענת כניסה: מילון המכיל סוקט של שחקן ושמו

טענת יציאה: מילון המכיל את הניקוד של כל שחקן

6. שם: create_new_game
תפקיד ושימוש: הפונקציה אחראית להרצת המשחק ומציאת המנצח
טענת כניסה: אובייקט clients המכיל סוקט ושם של שחקן
טענת יציאה: הסוקט והשם של הבוט המנצח

7. שם: handle_game
תפקיד ושימוש: טיפול במשחק - מההתחלה ועד הכרזה על מנצח וניתוק הסוקטים
טענת כניסה: הסוקטים והשמות של השחקנים
טענת יציאה: ניתוק הסוקטים של הבוטים

8. שם: end_connections
תפקיד ושימוש: ניתוק הסוקטים של הלקוחות
טענת כניסה: הסוקטים של הלקוחות
טענת יציאה: ניתוק הסוקטים

מדריך למשתמש

התקנת המערכת

- הפרויקט תוכנן עבור מערכת הפעלה מסוג **Windows** לכן יש לוודא כי המחשב עליו מריצים את המערכת עובד עם Windows

- זיכרון (זיכרון RAM) : זיכרון מתאים לתמיכה במערכת ההפעלה ולהפעלת היישום בצורה חלקה. הסכום הספציפי תלוי במורכבות המערכת, אך בדרך כלל מומלץ לפחות 4GB

- אחסון: חובה שיהיה למשתמש שטח אחסון מספיק להתקנת היישום ולאחסון כל הנתונים המקומיים או העדפות משתמש. כמות האחסון הינה בערך 20 KB, רק לקבצים של המערכת, אך ייתכן שהמשתמש יזדקק ליותר אחסון פנוי, תלוי בבוט שאותו הוא בונה.

- על מנת להריץ את המערכת נדרש ראשית התוכנה pycharm – תוכנת עריכת קוד python, עדיף גרסה עדכנית.

- נדרש שבמחשב יותקן python בגרסה עדכנית מאחר ויש modules שלא פועלים בגרסאות נמוכות.

- נדרש שכל המחשבים במערכת, כלומר השרת והלקוחות יהיו מחוברים לאותה רשת.

- נדרש להתקין את הספריות הבאות. ניתן להתקין אותן בעזרת pip:
    - socket

- threading
- struct
- time
- arcade

משתמשי המערכת

יש להפעיל את הקובץ bot_api בתוכנת pycharm בכל המחשבים, גם אם משמשים שחקנים וגם אם המחשב הוא השרת.

לאחר מכן יש להפעיל את הקובץ Server_for_bots בתוכנת pycharm במחשב בו רוצים שיהיה השרת הראשי.

עבור כתיבת בוט על פי הכללים הנדרשים יש לפתוח קובץ פייתון חדש על פי קובץ הבוט הבסיסי שמצורף ולקרוא לו בשם של הבוט או בכל שם אחר. לאחר מכן צריך לבצע import עבור הקובץ bot_api והספריה time וכל ספריה אחרת בה תרצו להשתמש עבור כתיבת הבוט.

קובץ הבוט הבסיסי:

```
basic bot template #
this is the code that must be in the bot file for it to work #
the playing style of the bot will be created by the player #

import time
import bot_api
```

```
Client setup #
NUM_ROUNDS = 10

:()def main

('my_bot = bot_api.GameState('type your name here
Client setup #

Game loop #

:(for _ in range(NUM_ROUNDS
Receive round start signal #
()my_bot.get_round_start_signal

.determine here wether to cooperate or defect #
the decision can depend on a certain strategy, your score, the #
opponent decisions or anything else


decision = #select your decision here (bot_api.COOPERATE  or
(bot_api.DEFECT
(my_bot.send_decision(decision

Receive and print score #
()my_bot.update_score

("print(f"my total score: {my_bot.my_total_score}\n

(time.sleep(5

Close connection #
()my.client_socket.close


:"__if __name__ == "__main
()main
```

לאחר מכן יש לפעול על פי ההוראות שבקוד. יש להכניס את שם הבוט במקום הייעודי, היכן שכתוב "type your name here", ובשורות שלאחר מכן להחליט האם לשתף פעולה או לבגוד ולשלוח את המהלך.

לאחר שכל הבוטים מוכנים יש להריץ את השרת ולאחר מכן את הבוטים. מיד אחר כך יופיע מסך הפתיחה ואחריו מסך המשחק, בו יהיה ניתן לצפות בכל המשחקים של הבוט, כשלבסוף תופיע טבלה עם דירוגי הבוטים.

סיכום אישי / רפלקציה

בהתחלה היה לי קשה לחשוב על רעיון לפרויקט. עלו לי רעיונות שהיה לי קשה לתכנן כיצד אממש אותם, מה שעיכב את התחלת העבודה שלי. הרעיון הגיע לי כאשר צפיתי בסרטון על תורת המשחקים והניסוי של אקסלרוד. הכרתי את דילמת האסיר עוד לפני, אך הצפיה בסרטון גרמה לי להבין את העומק והסיבוכים שנמצאים בה. הנושא ריתק אותי והבנתי שהפרויקט שאצור יעסוק בו.

במהלך הכתיבה למדתי הרבה מאוד על תחומים שונים כמו sockets, multithreading, עבודה עם ממשק גרפי ואפילו כתיבת מאמרים. בהתחלה נרתעתי מלכתוב הרבה קוד בעצמי ונעזרתי הרבה במורה ובמקורות מידע, אך עם התקדמות הפרויקט צברתי ביטחון עם הידע שלי והכתיבה נעשתה הרבה יותר קלה.

מבחינתי הפרויקט היה הדבר הכי משמעותי שעשיתי בתור מתכנת. הלמידה והסקרנות שהעבודה גרמה לי לא משתווים לשום דבר אחר שעשיתי לפני כן בתחום. במבט לאחור אני מאוד שלם עם הדרך בה בחרתי לבצע את הפרויקט ואני בטוח שהידע שצברתי במהלכו יעזור לי בעתיד.

ביבליוגרפיה

דילמת האסיר What Game Theory Reveals About Life, The Universe, and Everything

משרד החינוך: הנחיות לכתיבת תיק פרויקט בחלופה הגנת סייבר ומערכות הפעלה
במסגרת לימודי ההתמחות בהנדסת תוכנה 883589

-https://meyda.education.gov.il/files/CSIT/Cyber-protection-and-operating
systems-Project.pdf

Geeksforgeeks: Socket Programming with Multithreading in Python
-https://www.geeksforgeeks.org/socket-programming-multi-threading
python/amp/

נספחים

טכנולוגיה בה נעשה שימוש

פרוטוקול רשת

לאורך הפרויקט ישנו שימוש בהעברת מידע דרך sockets שמתבצע בפרוטוקול TCP.

פרוטוקול TCP הוא פרוטוקול הפועל בשכבת התעבורה של מודול חמשת השכבות, פרוטוקול זה מבטיח העברה אמינה של נתונים בין שתי תחנות ברשת מחשבים.
הפרוטוקול מוודא את הנתונים שהועברו באמצעות IP, מוודא את הנכונות שלהם, מאשר קבלה של הנתונים במלואם ובמידת הצורך מבקש שליחה חזרה של נתונים שאינם הגיעו בצורה תקינה.

בעת הקמת החיבור בין התחנות, יישנם שלושה שלבים (handshake way three):
1. syn – תחנת המקור שולחת הודעה לפתיחת קשר.
2. ack-syn – תחנת היעד מקבלת את ההודעה ושולחת בתגובה הודעת אישור – מייצג קבלה ואישור פתיחת קשר מצידה.
3. ack – תחנת המקור מייעדת את תחנת היעד על סיום מיסוד הקשר.

לאחר מיסוד הקשר: כל ההודעות בין שני המכשירים נשלחות באופן ממוספר אחת לשנייה, ורק כאשר מתברר כי ההודעה הקודמת הגיעה ליעדה באופן תקין. שליחת הנתונים בין מחשב הלקוח לשרת, מתבצע באמצעות sockets על בסיס פרוטוקול זה.

threading

לאורך הפרויקט יש שימוש במודול threading למטרת ביצוע כמה פעולות בו זמנית.

כדי להבין threading אסביר לפני כן על המעבד וליבותיו.
המעבד היא היחידה הראשית במחשב המבצעת פקודות ומקבלת אותות. למשל כל כפתור שנלחץ בעכבר מועבר כאות למעבד והמעבד מחליט מה להריץ ואם האות חשוב מספיק כדי לתת לו עדיפות.

thread או תהליכון הוא אובייקט של מערכת ההפעלה הנותן לנו אפשרות להריץ קוד באחת מהליבות של מערכת ההפעלה. הקוד ירוץ במקביל.

המודול threading בpython נותן לנו כלים שונים כדי להשתמש בתהליכונים.

משתמשים בthreading על מנת להריץ קודים שונים בו זמנית על מנת שקוד אחד לא יחכה לשני. למשל: כאשר יוצרים שרת השרת מאזין ללקוחות, זוהי פעולה חוסמת, פעולה אשר לא מבצעת כלום עד שמישהו לא מתחבר לשרת. במקרה זה נרצה ליצור thread היריץ את השרת ואותו הזמן יהיה ניתן לפעול במקביל על פעולות אחרות.