

תיק פרויקט



שם התלמיד: רועי אבישר
ת"ז: 211496658
כיתה: יב'5
שנה"ל: התש"פ
שם המורה: קובי שוצמן
תאריך הגשה: 10.6.2020

תוכן עניינים

תוכן עניינים

| | | |
|----|-------|-------------------------------|
| 3 | | מבוא |
| 4 | | מבנה |
| 4 | | מודולים שהשתמשתי בהם |
| 5 | | הלוח והיילים בשרת |
| 6 | | הגרפיקה |
| 6 | | הלוח והשחקנים בלקוח |
| 7 | | קליטת המהלך מהמשתמש |
| 8 | | תזוזה |
| 9 | | תחילת המשחק |
| 10 | | סוף המשחק |
| 11 | | מדריך למשתמש |
| 11 | | סביבת עבודה |
| 11 | | מהלך וחוקי המשחק |
| 12 | | מדריך למפתח |
| 12 | | פעולות בשרת |
| 15 | | פעולות בלקוחות |
| 16 | | תרשים קריאה בין הפעולות בשרת |
| 17 | | תרשים קריאה בין הפעולות בלקוח |
| 18 | | רפלקציה |
| 19 | | ביבליוגרפיה |

מבוא

הפרויקט הינו משחק דמקה בין 2 משתמשים. לפני שהתחלתי את הפרויקט חשבתי על מה לעשות אותו. התלבטתי בין המון אופציות שהיו פתוחות לפניי.

תחילה חשבתי לעשות צ'אט רב משתתפים אבל ישבתי וחשבתי עם עצמי והבנתי שעדיף לי לעשות משהו יותר יצירתי ומשהו שיותר יעניין אותי לעשות, משהו שמתאים לאופי שלי. מכאן החלטתי שעליי לבחור פרויקט או בנושא המשחקים או בנושא הנגינה שמאוד מדבר אליי. תחילה הלכתי יותר לכיוון הנגינה. רציתי ליצור משהו בסגנון של פסנתר/קלידים אבל בגלל שהתקשיתי לחשוב על איך לשלב את נושא הרשתות בפנים ובגלל שעשיתי פרויקט דומה בעברי באסמבלר ויתרתי על הרעיון.

בתוך תחום המשחקים התלבטתי בין דמקה לשש-בש. בסופו של דבר בחרתי בדמקה כי אני פשוט מטורף על המשחק הזה (יהיה לכם מאוד קשה לנצח אותי בדמקה).

לאחר שבחרתי את הנושא פתחתי את ערכת הדמקה בבית שלי ואת הנושא בויקיפדיה והתחלתי לסקר כל מהלך אפשרי שצריך ליישם אותו בפיתון. מיצירת הלוח, איפוס והשמת החיילים דרך תזוזה ואכילה של חייל ומלך עד להמלכה וניצחון.

ציירתי לוח דו ממדי על דף A4 ועל פיו יצרתי את הלוח בפיתון במערך דו מימדי. עשיתי המון עבודת מחקר ולימוד עצמי על פיתון עצמו ורשתות וגרפיקה (ראו ריפלקציה). במהלך הפרויקט עצמו נתקעתי בהמון קשיים ותקלות שחיפשתי עבורן דרכי פיתון. לדוגמה – במהלך בניית הפרויקט קרתה לי תקלה שחזרה על עצמה שהפרויקט היה נופל כל פעם בהרצה. הערת השגיאה שניתנה לי היא שהערכים במשתנים שמועברים בין הלקוח והשרת הם לא בטיפוס שהגדרתי להם.

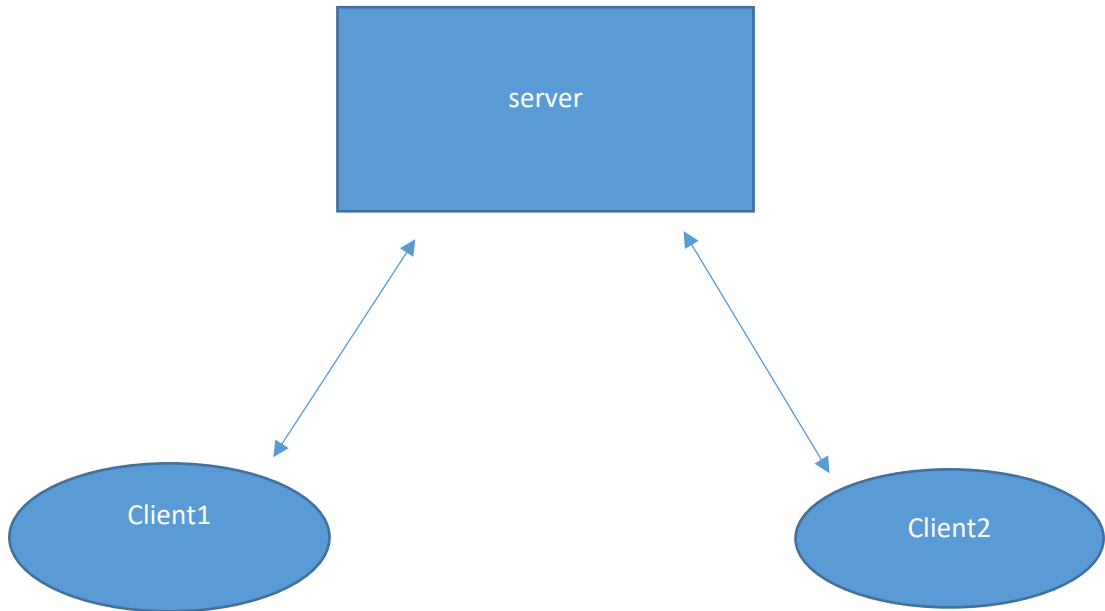
עקבתי אחרי כל הפרויקט וניסיתי להבין איפה הבעיה, איפה יש לי שגיאה בהעברת הנתונים שמשפיע בעקבות כך על כל שאר שליחת הנתונים ולא מצאתי.

ואז עלה לי רעיון, רגע לפני שהשתמשתי בערכים שקיבלתי מהשרת עשיתי להם הדפסה. כתוצאה מכך גיליתי שחלק מהערכים מתקבלים במשתנה אחד במקום להתקבל במשתנים שונים. לאחר התייעצות עם חבריי ובן דוד שלי שמכיר תשפה הבנתי שכדי להתמודד עם זה עליי להתשמש במודות של time ולעשות דילאיי בין כל שליחה של מסר בין השרת והלקוחות וכך זה היה שולח אותם בנפרד ולא יוצר את הבאג הזה.

בתיק פרויקט עצמו ניתן למצוא תיאור של מבנה הפרויקט ברשתות ובגרפיקה, תיאור הסינכרון בין הלקוחות בלקוח והשרת, מדריך למשתמש ולמפתח ותרשימי קריאה.

מבנה

הפרויקט עובד במבנה הבא:



2 המשתמשים יכולים לשחק במחשב אחד או ב2 מחשבים שונים. שחקן 1 מזין את הפעולה שהוא רוצה לעשות ללקוח (עם העכבר), הלקוח מקבל את הנתונים ומעביר אותם לשרת. השרת מעבד את הנתונים לפעולות ומעדכן את הלוח בהתאם.

לאחר מכן השרת שולח כמה וכמה נתונים חשובים להמשך המשחק(פירוט בהמשך) אל הלקוחות והם מעדכנים אצלם בלוח הגרפי את המהלך והתור עובר ללקוח הבא. התור לא יכול לעבור למעשה עד שהשחקן לא מזין מהלך חוקי שהוא רוצה לעשות.

מודולים שהשתמשי בהם :

- Time
- Socket
- Pygame

הלוח והחיילים בשרת:

בתחילת המשחק נוצר לוח **בשרת**. לוח זה הוא מבנה נתונים ומורכב באופן הבא:
 הלוח מורכב מרשימה של 8 רשימות. וכך למעשה נוצר אפקט דו מימדי של לוח דמקה.
 הלוח הוא מן מערך דו מימדי כאשר לכל משבצת יש ערך (x,y) משלה וככה פונים אליה.
 בתחילת המשחק הלוח מתאפס (בכל ערכיו משובץ המספר 0) ולאחר מכן מוכנסים החיילים.
 החיילים מקוטלגים כ"1" ו"2".
 בתחילת המשחק ישנה פעולה המכניסה את החיילים למיקומם בלוח.
 מלך מיוצג כ 11 או 22 בהתאמה לחייל שהפך למלך.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 2 | 0 | 2 | 0 | 2 | 0 |
| 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 |
| 2 | 0 | 2 | 0 | 2 | 0 | 2 | 0 |

להלן הלוח שיצרתי על דף בבית:
 מעל ומצד שמאל לטבלה רשומים הערכים של השורות והטורים במערך.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | | 1 | | 1 | | 1 | | 1 |
| 1 | 1 | | 1 | | 1 | | 1 | |
| 2 | | 1 | | 1 | | 1 | | 1 |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | 2 | | 2 | | 2 | | 2 | |
| 6 | | 2 | | 2 | | 2 | | 2 |
| 7 | 2 | | 2 | | 2 | | 2 | |

הגרפיקה

הלוח והחיילים בלקוח:

בנוסף ללוח בשרת, בתחילת המשחק נוצר גם לוח גרפי ב2 בלקוחות שמוצג למשתמש. לוח זה גודלו 800X800 פיקסלים.

כיוון שהלוח מורכב מ8x8 משבצות, נוצא מצב שכל משבצת בלוח הגרפי היא למעשה 100x100 פיקסלים. לכן הקשר בין משבצת בלוח הגרפי למשבצת בלוח מהרשימות הוא חילוק והכפלה ב100.

לדוגמה אם בלוח הגרפי ישנה נקודה ששיעוריה (234, 574) נחלק את שיעורי הנקודה ב100 ונקבל (2, 5) וזוהי למעשה הנקודה בלוח שמורכב מרשימות בשרת. כך למעשה מתקיימת העברת נתוני המיקום של החיילים בין השרת ללקוח.

בתחילת המשחק ישנה פעולה שמציירת את השחקנים על הלוח. השחקנים הם למעשה עיגולים גרפיים בצבעים שונים.

הפעולה מציירת את החיילים בקוטר של 30 פיקסלים וכאשר מרכז העיגול הוא מרכז המשבצת שעליה מצויר החייל.

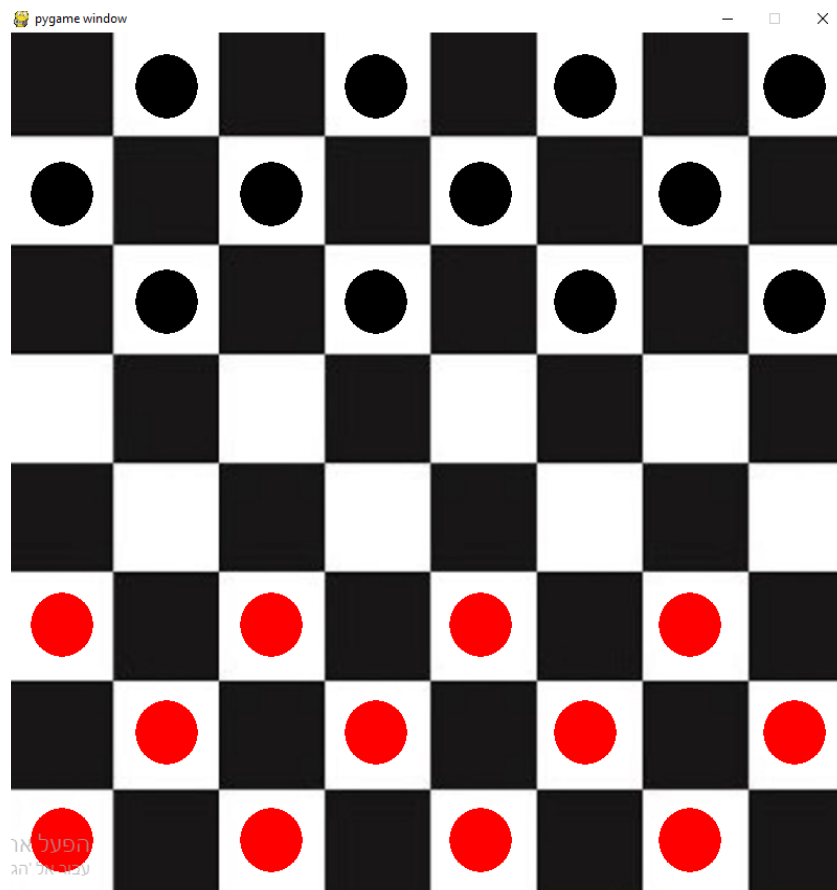
חייל 1 – עיגול שחור

חייל 2 – עיגול אדום

מלך 11 – עיגול אפור

מלך 22 – עיגול בורדו

כך המסך נראה בתחילת המשחק:



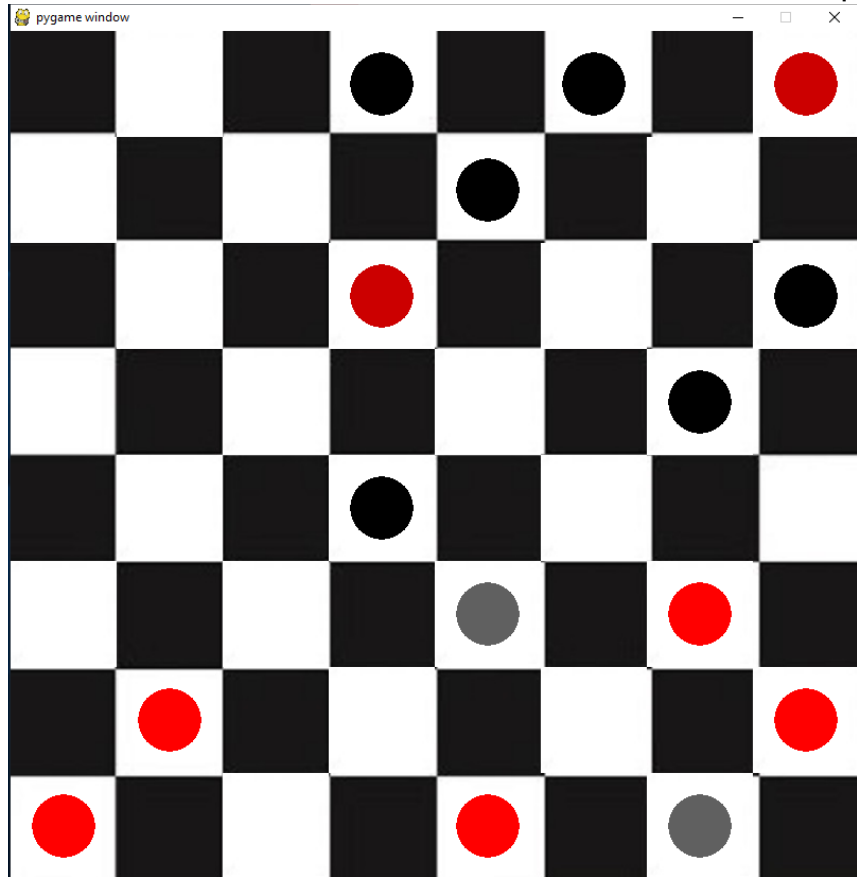
למעשה מה שקורה בתכנית זה שהלוח הגרפי שנמצא בכל אחד מהלקוחות מסונכרן עם הלוח שנמצא בשרת.
הפעולות של המשתמש נקלטות באחד הלקוחות (אסביר בהמשך על איך זה עובד), הן מועברות לשרת. בשרת מתבצעת הפעולה על הלוח שמורכב מרשימות. לאחר מכן מועברים הנתונים אל 2 הלקוחות במקביל:
- אם המהלך שבוצע היה חוקי
- מהו המהלך שבוצע (תזוזה של חייל, אכילה של חייל, תזוזה של מלך או אכילה של מלך)
- במידה והמהלך שבוצע הוא אכילה אז גם את מיקום החייל הנאכל
- במידה והמהלך שבוצע הוא תזוזה או אכילה של חייל אז גם האם נוצר כתוצאה מהמהלך מלך
- לבסוף האם כתוצאה מן המהלך נוצר ניצחון
לאחר קבלת הנתונים בלקוחות הלוח הגרפי מתעדכן גם הוא והתור עובר אל הלקוח הבא.

קליטת המהלך מהמשתמש : הממשק עובד עם העכבר, בדרך מאוד פשוטה כאשר המשתמש רוצה לבצע מהלך הוא מקיש עם העכבר על מיקום החייל שהוא רוצה להזיז ועל המיקום הסופי שאליו הוא רוצה להגיע. כתוצאה מהקשות העכבר נפלטים צלילים שמעידים אם המהלך שבוצע שגוי או לא.
לאחר שהמשתמש עושה את המהלך שלו מתקבלת בתוכנית הנקודות בפיקסלים שעליהם הקיש המשתמש. ומכאן התוכנית ממשיכה לרוץ לפי איך שהסברתי קודם.

תזוזה - תזוזה של חייל או מלך קורית כך:

- התוכנית מדביקה תמונה של ריבוע לבן בגודל 100x100 פיקסלים במקום ההתחלתי של החייל.
- במידה והתזוזה כוללת אכילה היא גם מדביקה את הריבוע הלבן במקום של החייל שנאכל.
- לאחר מכן הפעולה מציירת עיגול חדש במיקום הסופי שהוגדר לחייל להגיע אליו.
- כך למעשה נוצר מן אפקט של תזוזה או אכילה בתוך הממשק הגרפי.

כך נראה הלוח לאחר כמה מהלכים שכוללים בתוכם אכילות והמלכות:



תחילת המשחק :

כאשר רק לקוח אחד מתחבר מוצג מסך שרושם שמחכים להתחברות של הלקוח השני. לאחר שהמשתמש השני התחבר מוצג מסך שרושם לכל אחד איזה צבע הוא.



המשחק יתחיל בקרוב

אתה החיילים השחורים

אתה מתחיל

בהצלחה!

סוף המשחק :

בסוף המשחק מוצגים בהתאמה למשתמשים מסך "you win" ו "you lose".



מדריך למשתמש:

סביבת עבודה:

סביבת העבודה שדרושה למשתמש היא pycharm 2.7 שם הקוד רץ. המשתמשים מקבלים את הקוד כקבצי python. מהמשתמש נדרש לשמור 5 קבצי תמונות ו4 קבצי אודיו על שולחן העבודה בשמן המקורי.

מהלך וחוקי המשחק:

הלוח מונח כך שהמשבצת הימנית ביותר בשורה הקרובה לכל שחקן תהיה שחורה. האבנים מונחות על המשבצות הלבנות של הלוח וההתקדמות היא רק על משבצות אלה באלכסון.

בתחילת המשחק אבני השחקן האחד מונחות בשלוש השורות הראשונות של הלוח ואילו אבני השחקן השני מונחות באותו אופן בצד שלו. לפי המוסכם, השחור מבצע את המהלך הראשון והלבן(בפרויקט – אדום) משיב עליו. שתי הפעולות גם יחד נחשבות כמסע אחד.

חיילים:

תנועה:

כל שחקן מניע בתורו אבן-משחק באלכסון, ממשבצת לבנה אחת למשבצת לבנה סמוכה בכיוון היריב. על המשבצת להיות פנויה מכלים, כלומר לכל אבן יש שתי אפשרויות תנועה – לכיוון היריב ימינה ולכיוון היריב שמאלה – וכל אחת מהן עשויה להיות חסומה.

אכילה:

אכילה או דילוג מתבצעת כאשר אבן משחק מונחת במשבצת סמוכה לאבן היריב, ומעבר לאבן היריב יש מקום פנוי. אכילה מבוצעת על ידי הנחת האבן במקום הפנוי שמעבר לאבן היריב והסרת אבן היריב מן הלוח.

מלכים:

תנועה:

כשאבן משחק מגיעה לשורה האחרונה, היא הופכת להיות "מלך" (בעברית יש הקוראים לכך "מלכה"), כאשר בדרך כלל מייצגים מלך על ידי שתי אבני משחק מונחות אחת על השנייה (בפרויקט הם מיוצגים בצבעים שונים). מלך, בניגוד לאבן רגילה, יכול לנוע לכל הכיוונים באלכסון (כלומר גם אחורה), ללא הגבלה על כמות המשבצות בדרך, וכמו כן הוא יכול לבצע אכילות בכל כיוון.

אכילה:

המלך יכול לאכול כל אבן בודדה של היריב שנמצאת באחת מארבעת דרכיו האפשריות. על המלך לעצור משבצת אחת אחרי המשבצת של החייל שנאכל.

חוקים נוספים:

- לא קיימת חובת אכילה כאשר היא אפשרית
- אי אפשר לאכול יותר מחייל אחד בתור

מטרת המשחק היא להוריד מהלוח ("לאכול") את כל האבנים של היריב. שחקן שנשאר ללא אבנים מוכרז כמפסיד.

מדריך למפתח

פעולות בשרת:

1) set_board():

הפעולה מאתחלת את הלוח ומכניסה לכל ערכיו 0. הלוח הינו משתנה גלובלי.

2) insert_players():

הפעולה מכניסה אל תוך הלוח את השחקנים 1 ו 2

3) move(firstx, firsty, lastx, lasty, type):

הפעולה מקבלת את המיקום ההתחלתי של חייל, מיקום היעד שלו ואת הסוג שלו. הפעולה בודקת אם תחילה את אמינות הנתונים. לאחר מכן אם התזוזה אפשרית על הלוח על ידי חיסור ערכי היעד מערכי המיקום הנוכחי. אם המנה עונה על חוקיות מסוימת שאסביר בהמשך הפעולה מזיזה את החייל ממיקום ההתחלה שלו למיקום היעד שלו. ושולחת מהשרת אל 2 הלקוחות את הפרטים הבאים:

1) "False" כדי לציין שהערכים שהוזנו הם חוקיים ואפשר להתקדם בתוכנית
2) "moving" כדי להעביר ללקוח שהפעולה שנעשה היא תזוזה של חייל ושימשיך בתוכנית לפי ערך זה.

הפעולה מחזירה אמת אם התזוזה בוצעה בהצלחה ושקר אם אחרת.

החוקיות:

לחייל מסוג "1":

מנת ה-x: 1

מנת ה-y: 1 או 1-

לחייל מסוג "2":

מנת ה-x: -1

מנת ה-y: 1 או 1-

4) eating(firstx, firsty, lastx, lasty, type):

הפעולה מקבלת את המיקום ההתחלתי של חייל, מיקום היעד שלו ואת הסוג שלו. הפעולה בודקת אם תחילה את אמינות הנתונים. לאחר מכן אם האכילה אפשרית על הלוח על ידי בדיקת ערך ומיקום החייל המיועד לאכילה וחיסור ערכי היעד מערכי המיקום הנוכחי. אם המנה עונה על חוקיות מסוימת שאסביר בהמשך הפעולה מזיזה את החייל ממיקום ההתחלה שלו למיקום היעד שלו ואוכלת את החייל שבדרך. ושולחת מהשרת אל 2 הלקוחות את הפרטים הבאים:

1) "False" כדי לציין שהערכים שהוזנו הם חוקיים ואפשר להתקדם בתוכנית
2) "eating" כדי להעביר ללקוח שהפעולה שנעשה היא אכילה של חייל ושימשיך בתוכנית לפי ערך זה.

- (3) את הערך הא של החייל שנאכל (מתקבל בלקוח כפרמטר בשם eatedx)
 (4) את הערך הy של החייל שנאכל (מתקבל בלקוח כפרמטר בשם eatedy)

הפעולה מחזירה אמת אם האכילה בוצעה בהצלחה ושקר אם אחרת.

החוקיות:

לחייל מסוג "1":

מנת הא: 2

מנת y: 2 או 2-

לחייל מסוג "2":

מנת הא: 2-

מנת y: 2 או 2-

5) king_move(firstx, firsty, lastx, lasty, type):

הפעולה מקבלת את המיקום ההתחלתי של המלך, מיקום היעד שלו ואת הסוג שלו. הפעולה בודקת אם תחילה את אמינות הנתונים. לאחר מכן אם התזוזה אפשרית על הלוח על ידי בדיקה שאין שום חייל אשר מפריע בדרך וחיסור ערכי היעד מערכי המיקום הנוכחי. אם המנה עונה על חוקיות מסוימת שאסביר בהמשך הפעולה מזיזה את המלך ממקום ההתחלה שלו למקום היעד שלו. ושולחת מהשרת אל 2 הלקוחות את הפרטים הבאים:
 (1) "False" כדי לציין שהערכים שהוזנו הם חוקיים ואפשר להתקדם בתוכנית
 (2) "king moving" כדי להעביר ללקוח שהפעולה שנעשה היא תזוזה של מלך ושימשיך בתוכנית לפי ערך זה
 הפעולה מחזירה אמת אם התזוזה בוצעה בהצלחה ושקר אם אחרת.

החוקיות:

לכל מלך:

מנת הא שווה למנת y

או

מנת הא שווה למינוס מנת הy

6) king_eating(firstx, firsty, lastx, lasty, type):

הפעולה מקבלת את המיקום ההתחלתי של המלך, מיקום היעד שלו ואת הסוג שלו. הפעולה בודקת אם תחילה את אמינות הנתונים. לאחר מכן אם האכילה אפשרית על הלוח על ידי בדיקת ערך ומיקום החייל המיועד לאכילה, בדיקה ששום חייל אחר לא מפריע באמצע הדרך וחיסור ערכי היעד מערכי המיקום הנוכחי.
 חשוב לציין שמלך יכול לאכול בתנאי שהוא עוצר משבצת אחת לאחר המשבצת של השחקן הנאכל.
 אם המנה עונה על חוקיות מסוימת שאסביר בהמשך הפעולה מזיזה את המלך ממקום ההתחלה שלו למקום היעד שלו ואוכלת את החייל שבדרך. ושולחת מהשרת אל 2 הלקוחות את הפרטים הבאים:

- (1) "False" כדי לציין שהערכים שהוזנו הם חוקיים ואפשר להתקדם בתוכנית
- (2) "king eating" כדי להעביר ללקוח שהפעולה שנעשה היא אכילה של מלך ושימשיך בתוכנית לפי ערך זה.
- (3) את הערך הא של החייל שנאכל (מתקבל בלקוח כפרמטר בשם eatedx)
- (4) את הערך הע של החייל שנאכל (מתקבל בלקוח כפרמטר בשם eatedy)

הפעולה מחזירה אמת אם האכילה בוצעה בהצלחה ושקר אם אחרת.

החוקיות:

לכל מלך:

מנת הא שווה למנת y

או

מנת הא שווה למינוס מנת הע

7) wining():

הפעולה סורק את כל הלוח ובודקת אם נוצר מצב של ניצחון (1 מסוגי החיילים לא נמצא יותר על הלוח) אם כן היא מדפיסה את השחקן שניצח ומחזירה אמת אם לא מדפיסה שאף אחד לא ניצח מחזירה שקר.

8) check_crowning(type, x, y):

הפעולה מקבלת מיקום של חייל ואת הסוג שלו.
הפעולה בודקת אם החייל נמצא במיקום שהופך אותו למלך. אם כן היא משנה את ערכו בלוח לערכו של המלך המתאים (1 ל 11 ו 2 ל 22).
פעולה זו נקראת רק על ידי eating move.

9) broadcast(message):

הפעולה מקבלת מסר, מעבירה אותה למחרוזת ושולחת אל 2 הלקוחות.
בתוך הפעולה יש דיילאיי של 0.1 כדי כשכל פעם כשקוראים לפעולה ברצף המסרים לא יתחברו.

10) onclient():

מקבלת את הנתונים למהלך במשחק של אחד הלקוחות בזמן שהלקוח השני מחכה לקבל את הלוח המעודכן. הפעולה בודקת האם מדובר בחייל או במלך וקוראת בהתאם לפעולות האפשריות. אם חוזר אמת הפעולה שולחת את הלוחות המעודכנים אל הלקוחות ועוברת אל המהלך של הלקוח השני בזמן שהלקוח הראשון מחכה לקבל את הלוח המעודכן וכן הלאה.
במעבר בין המהלכים של הלקוחות הפעולה בודקת האם קיים ניצחון על ידי קריאה לפעולה המתאימה. אם קיים ניצחון הפעולה מדיפסה את השחקן המנצח ויוצאת מהמשחק.

פעולות בלקוחות:

1) draw_players():

הפעולה מציירת בתחילת המשחק את החיילים על הלוח הגרפי.

2) place_of_piece():

הפעולה מחכה עד שהמשתמש יקיש על משבצת והיא מחזירה את ערכי ה x ו y בפיקסלים של הנקודה שהמשתמש לחץ עליה.

3) update_screen_moving(firstx, firsty, lastx, lasty, color):

הפעולה מקבלת ערכי תזוזה של חייל/מלך והצבע שלו ולפיכך מעדכנת את המסך הגרפי. כלומר מוחקת את החייל מהמיקום הראשוני שלו ומציירת אותו מחדש במיקום הסופי שלו.

4) update_screen_eating (firstx, firsty, lastx, lasty, color, eatedx, eatedy):

הפעולה מקבלת ערכי אכילה של חייל/מלך וצבעו ומוחקת את החייל/מלך במיקום הנאכל ולאחר מכן קוראת לפעולה update_screen_moving.

5) the_end (who_win):

הפעולה מחרוזת של איזה שחקן ניצח ומעדכנת את המסך בהתאם. אם השחקן הפסיד תציג שהוא הפסיד. אם הוא ניצח תציג שהוא ניצח.

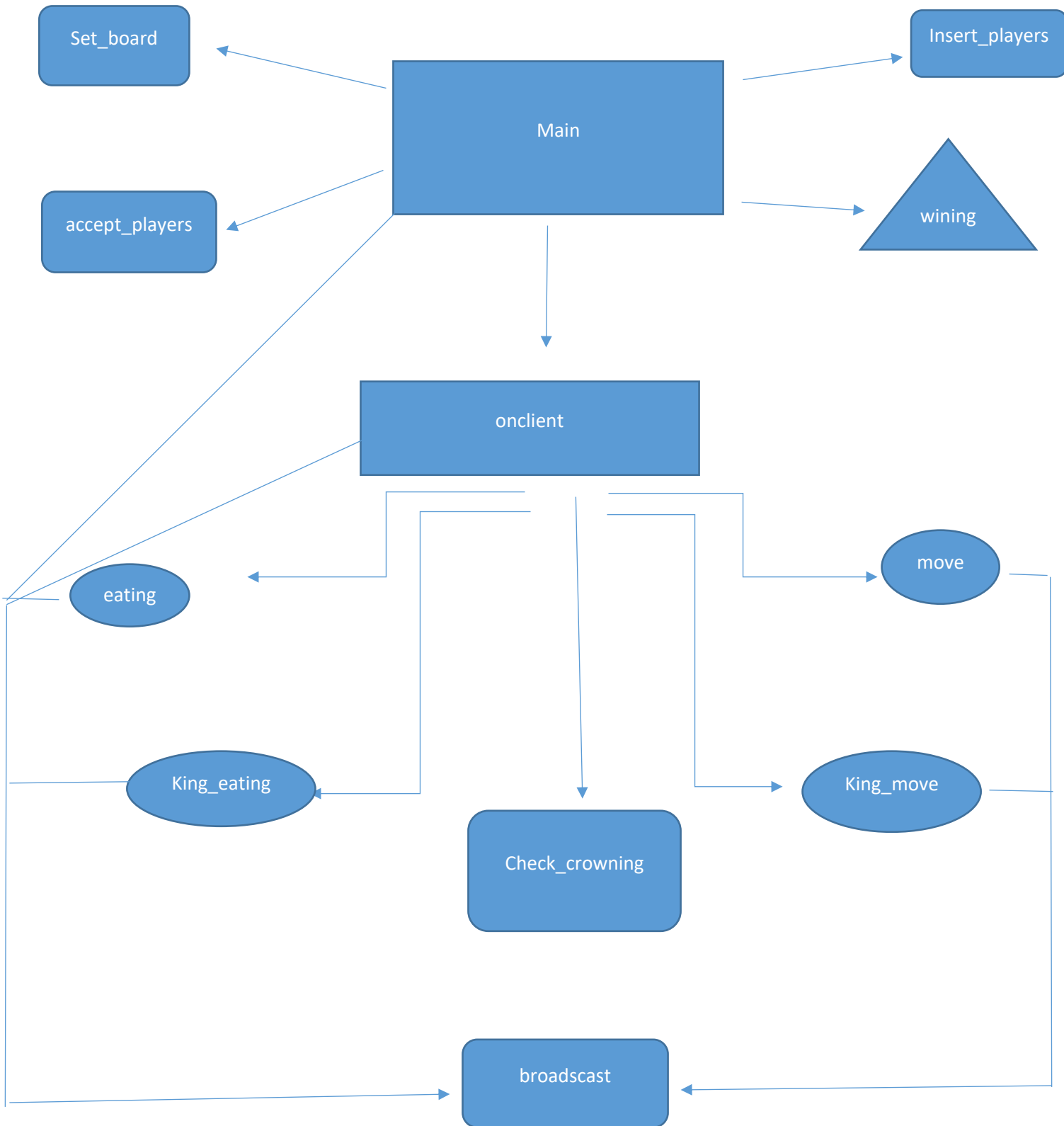
6) play(sound):

הפעולה מקבלת צליל, טוענת אותו ומשמיעה אותו.

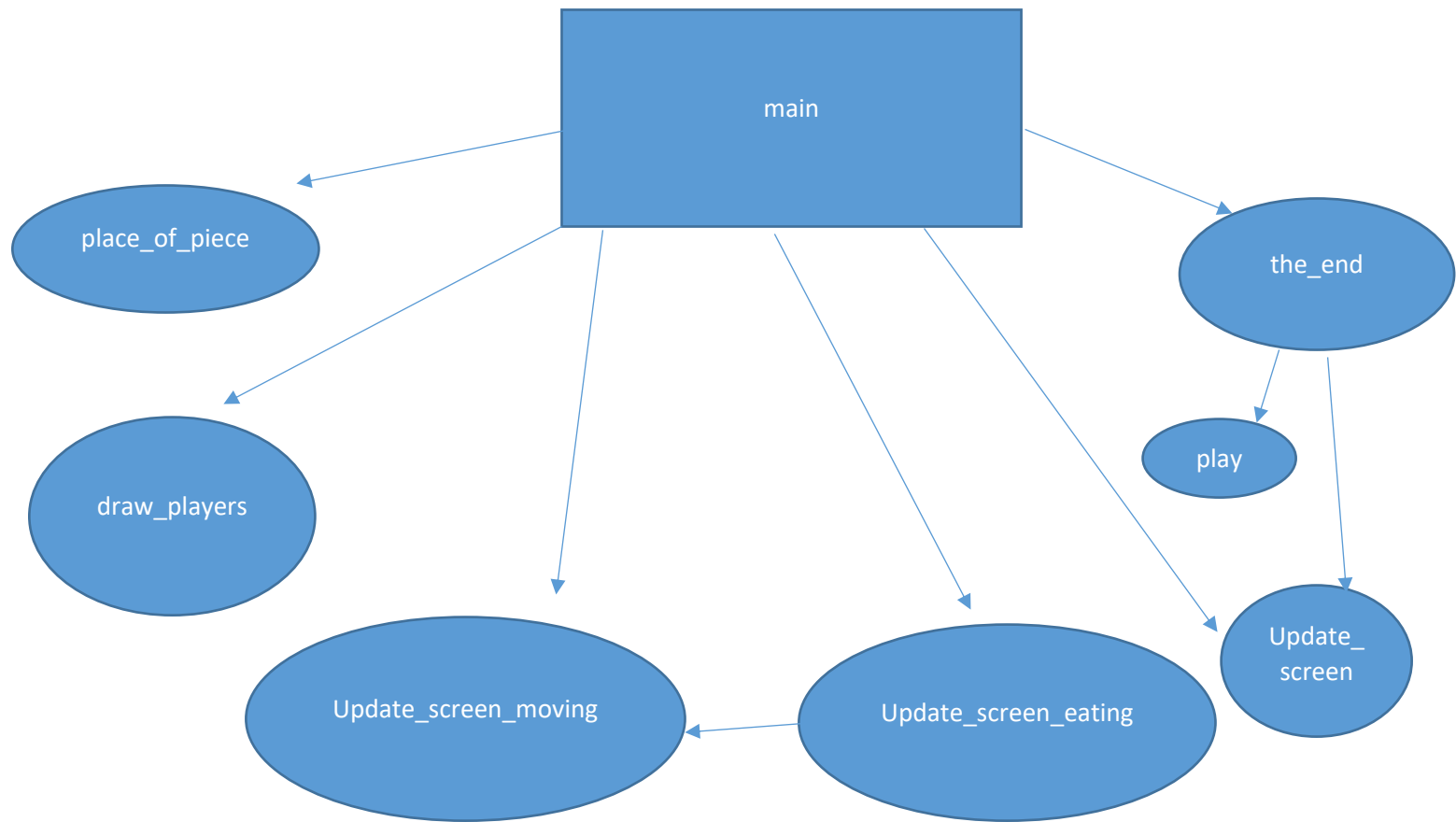
7) update screen(image):

הפעולה מקבלת תמונה, טוענת אותה ומדביקה אותה על המסך.

תרשים קריאה בין פעולות בשרת:



תרשים קריאה בין פעולות בלקוח:



רפלקציה

מבחינתי להגיע לחלק הזה בעבודה שאני צריך לסכם זה הישג אדיר. במהלך השנתיים האלה מצאתי את עצמי (מישהו שמאוד אוהב את המקצוע מדעי המחשב ומתחבר לזה) נופל לתוך תהום. ממש התקשיתי ללמוד את הנושא פייתון ורשתות למרות שטכנית הוא היה אמור להיות לי ממש קל להבנה. כמה שניסיתי ללמוד ושהסבירו לי לא הצלחתי לתפוס את הנושא והחשיבה. עם יד על הלב חשבתי שאני מבזבז את הזמן שלי בשיעורי סייבר כי פשוט לא האמנתי שאני אצליח לעשות פרויקט ואף הייתי על סף פרישה.

במהלך השנה הנוכחית כשהגיע הזמן לבחור נושא לפרויקט ולעבוד עליו הרגשתי שאני לא יכול לוותר לעצמי בכזאת קלות. ידעתי שזה לא מתאים לאופי שלי והצבתי לעצמי מטרה לסיים את השנה עם פרויקט גמר שאני שלם איתו. קיבלתי המון פרטני מהמורה בבית ספר. ישבתי וחקרתי ולמדתי שעות על גבי שעות. התניסיתי כל פעם בכתיבה של תוכניות קטנות ושמתי דגש על לא רק לכתוב את התוכניות אלא גם להבין מה עשיתי.

הזמן לא שיחק לטובתי והייתי חייב להתחיל לעבוד על הפרויקט בשביל לסיים אותו בזמן מבלי שעדיין למדתי והבנתי הכל.

התחלתי לעבוד על הפרויקט ותוך כדי ללמוד את מה שאני עושה (עם המון תמיכה ודחיפה מבן דוד שלי שיודע פייתון ושחפרתי לו בשאלות).

תחילה חשבתי על להקל ראש וללכת לכיוון של פרויקט שיהיה לי קל לעשות ולא אצטרך להשקיע יותר מדי. רבל שוב ידעתי זה לא האופי שלי וכשאת עושה משהו אני אעשה אותו עד הסוף בצורה הכי טובה שלי. לכן בחרתי דמקה. מבחינתי להצליח לעשות דמקה אז באותה תקופה היה כמו חלום.

ישבתי ימים ולילות (הקורונה עזרה בזה) על הפרויקט תוך כדי שאני לומד איך לעשות אותו. ונתקלתי בהמון אבל המון תקלות. וכך כל התקדמות הכי קטנה שלי עם הפרויקט ממש ריגשה אותי כמו ילד קטן.

הרגשתי שעם ההתקדמות בפרויקט אני מבין יותר ויותר ולא רק את השפה אלא גם כישורים שיעזרו לי להתמודד עם עוד שפות תיכנות כמו איתור תקלות ובאגים בפרויקט, חשיבה מחוץ לקופסה ליישום של הפרויקט, כתיבת אלגוריתם, לימוד עצמי של חומר מאינטרנט, תכנון זמן ויכולת לשבת ולהתרכז רק במשהו אחד.

הרחבתי את הידע שלי בפייתון ברמה מטורפת בפרק זמן שלא היה נראה אפשרי וזאת בזכות התמדה, השקעה והמון כוח רצון.

אז הגענו לסוף שנה והפרויקט שלי מוכן. אני ממש שלם איתו וגאה בו וממש מרוצה ממה שיצא לי ומרוצה ממעצמי שהצלחתי לעמוד במטרה שלי מבלי לנסות לחפף.

ביבליוגרפיה

https://www.w3schools.com/python/python_examples.asp

https://data.cyber.org.il/python/python_book.pdf

<https://www.fxp.co.il/showthread.php?t=9888729&goto=nextnewest>

<https://realpython.com/pygame-a-primer/>

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

<https://he.wikipedia.org/wiki/%D7%93%D7%9E%D7%A7%D7%94>