פרויקט סיום - Kirby multiplayer game

מגיש: גיא נבו

בית ספר: תיכון הנדסאים

עיר: הרצליה

מורה מלווה: ערן רוזנק

מועד הגשה: 26.4.20



תוכן עניינים

הערות……………………………………………………………………………3

מבוא……………………………………………………………………………..4

תיאור המוצר המוגמר………………………………………………….…..….….6

מהלך הפיתוח…………………………………………………………...…….....7

מדריך למשתמש………………………………………………………..………..8

* תיאור מסכים…………………………………………………...…...…………………8
* הוראות הפעלה/ משחק………………………....……………………………………..10

סביבת העבודה………………………………………...………………………..13

מבנה/ ארכיטקטורה של הפרויקט…………………….……………...…………..14

* החלקים המרכזיים בקוד………………………..……………………………………...14
* פעולות מרכזיות בקוד………………………………………………………………….18
* האלגוריתם המרכזי…………...……………………………………………………….23

הסברים על הקבצים השונים והמחלקות………….……….……………………...24

מבנה הנתונים…………………………….……………………………………..32

רפלקציה…………………………………………………….…………………..33

ביבליוגרפיה……………………………………...……………………………...35

הערות

* לאורך הספר השתמשתי במילה AI כדי לתאר את הדמות של המחשב.

מבוא:

רקע לפרויקט:

קירבי הוא שמה של סדרת משחקי מחשב שפותחה על ידי HAL Laboratory והוצאה על ידי נינטנדו. משחקו הראשון של קירבי הוא "Kirby's Dream Land" ויצא בשנת 1992. קירבי הוא יצור ורוד בעל מבנה גוף כדורי, יש לו שתי רגליים בצורת כדור פחוס ושתי ידיים כדוריות. הכוח של קירבי הוא שאיבה, כאשר קירבי שואב יצורים מסוימים הוא מקבל את כוחותיהם, כגון אש, קרי, מטרייה, חשמל ועוד. המשחק הספציפי שעליו ביססתי את הפרויקט הוא "kirby Nightmare In Dreamland".

תהליך המחקר:

בתחילת העבודה, למדתי איך לבנות את השרת באופן הרצוי כך שיפעל מול שני משתמשים. השתמשתי בעבודה גם בספרייה threading, לכן חזרתי על החומר בספר של גבהים כדי שאוכל להשתמש בה. בנוסף למדתי הרבה על ספריית pygame ב-python שהשתמשתי בה בכתיבת הפרויקט על מנת ליצור את האנימציות, תזוזת הAI, והגרפיקה באמצעות תמונות.

הסיבה לבחירת הנושא:

אני מבלה הרבה מהזמן הפנוי שלי במשחקי מחשב. לכן החלטתי שלפרויקט סיום במגמה אני רוצה לכתוב משחק מחשב. חשבתי הרבה על איזה סוג של משחק אני רוצה ומה אני יכול להספיק בזמן הנתון, ולבסוף הגעתי למסקנה שליצור משחק מאפס לא יהיה אפשרי, אבל אני כן יכול לכתוב משחק המבוסס על משחק שקיים. החלטתי שהמשחק Kirby מכיל את האלמנטים שאני הייתי רוצה שיהיו בפרויקט שלי אז בחרתי בו. סיבה נוספת לבחירה היא שהמשחק "kirby Nightmare In Dreamland" הוא משחק ששיחקתי בו הרבה בילדות. מאוד אהבתי את המשחק אז חשבתי שיהיה כיף ליצור גרסה משלי למשחק.

תיאור המוצר המוגמר:

שם:

קירבי משחק מרובה משתמשים (kirby multiplayer game).

הסבר על המוצר:

המוצר המוגמר הוא משחק. הוא בנוי משרת ומשני משתמשים, כלומר על מנת לשחק במשחק השרת צריך לפעול ואליו מחוברים שני משתמשים ממחשבים שונים או מאותו מחשב. המשחק מיועד לשני שחקנים. מטרת השחקנים היא להביס את הדמות של המחשב. לכל שחקן כמות חיים מסוימת וכאשר הדמות של המחשב פוגעת בדמות של שחקן מסוים הוא מאבד חיים אחד ולהפך, וכאשר כמות החיים של שחקן מגיעה לאפס הוא לא משחק יותר ורק צופה בשחקן השני. אם לשני השחקנים לא נותרו חיים הם מפסידים, ואם לדמות של המחשב של נותרו חיים השחקנים מנצחים.

מהלך הפיתוח:

תהליך הפיתוח:

תחילה כתבתי את השרת ומחלקה המייצגת שחקן. במחלקת השחקן נמצאות כל הפעולות הקשורות לקבלת קלט מהמשתמש ובניהול התזוזה של הדמות במרחב, כמו למשל: תזוזה שמאלה, תזוזה ימינה, קפיצה והתכופפות. השרת הוא שרת פשוט המקבל מיקום של שחקן אחד ומעביר אותו לשני (רק בשלב ההתחלתי).

יחד עם השרת כתבתי מחלקה נוספת שנקראת network ובה נמצאות פעולות שמטרתן לקשר בין מחלקת השחקן לשרת כמו למשל שליחת וקבלת מידע.

לאחר כתיבת השרת ומחלקת השחקן עברתי לכתיבת הפעולות הקשורות בהדפסת תמונות למסך ולטיפול באנימציה. כל הפעולות האלו נמצאות במסמך נפרד שנקרא kirby. תוך כדי העבודה על המסמך הזה, ערכתי את התמונות שהן האנימציה של הדמות של השחקן (הרבה מאוד תמונות) והעליתי אותן למסמך כך שיהיה ניתן להשתמש בהן.

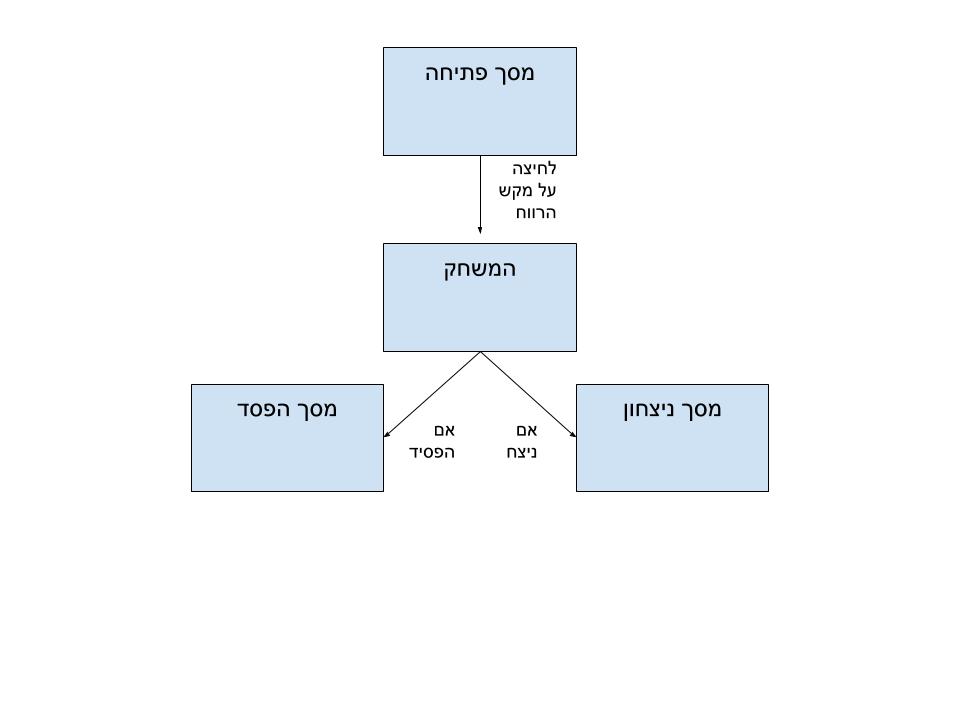
אחרי כמה בדיקות, ראיתי שהכל עובד כמו שצריך אז התחלתי לעבוד על הדמות של המחשב בשלב הראשון (היו שלושה בתכנון, לאחר שהקדימו את מועד ההגשה החלטתי שיהיו שניים). לדמות יש מחלקה נפרדת שנקראת AI ובמחלקה פעולות שנועדו לחשב את מיקום הדמות על המסך בכל רגע וגם לבדוק האם אחד מהשחקנים נפגע מתקיפות הדמות.

הספריות בשימוש:

אלה הספריות שבהן השתמשתי בכתיבת הפרויקט: pygame, threading, socket, pickle, time.

מדריך למשתמש:

**תיאור מסכים:**

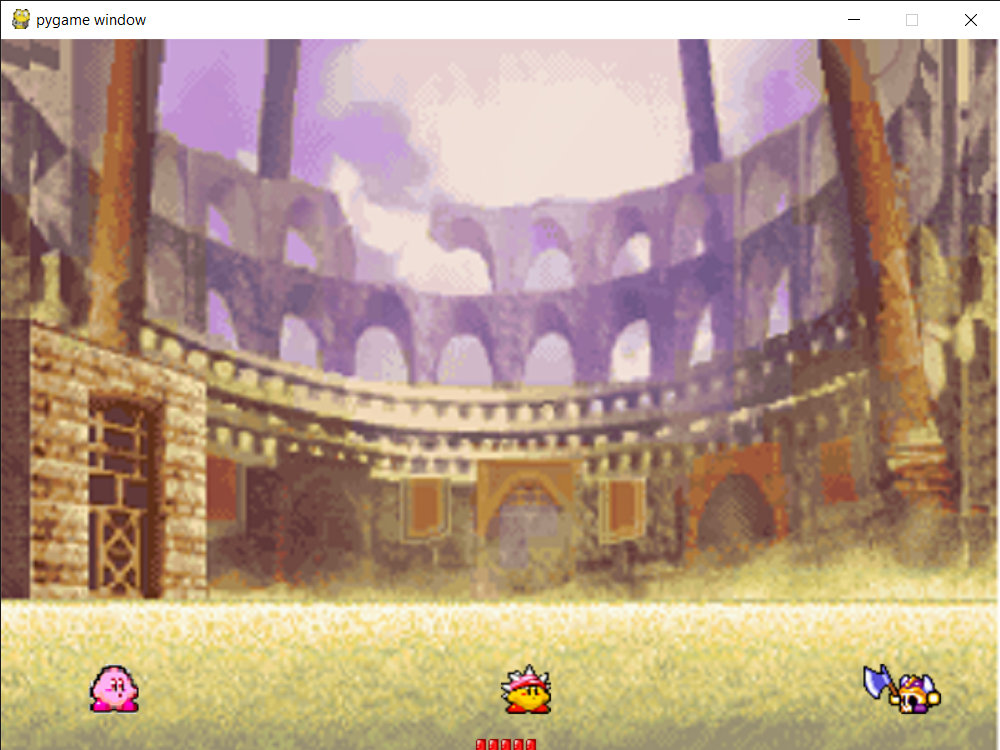


מסך הפתיחה:

נפתח ברגע שמריצים את המשחק. לחיצה על המקש "space" תוביל למסך הבא.

כך נראה מסך הפתיחה (התמונה היא מסך הפתיחה של המשחק המקורי):



מסך המשחק: לאחר מסך הפתיחה המשחק מתחיל. כך נראה חלון המשחק:

כפי שניתן לראות בתמונה, במשחק יש 3 דמויות. השמאלית היא של השחקן הראשון, האמצעית של השני, והימנית של המחשב. הקווים האדומים בתחתית המסך מייצגים את כמות החיים שיש לשחקן.

**מסכי סיום:** בסיום המשחק מופיע אחד ממסכי הסיום.במקרה וניצחת, כלומר לדמות של המחשב אין חיים, מופיע מסך הניצחון:



במקרה והפסדת, כלומר לשני השחקנים לא נותרו חיים, מופיע מסך ההפסד:



לחיצה על רווח/ אסקייפ, תוביל ליציאה מן המשחק.

הוראות הפעלה:

1. את המערכת אפשר להפעיל ממחשבים שונים המחוברים באותה רשת (LAN) או ממחשב אחד (שני לקוחות ושרת המופעלים מאותו מחשב). המחשב צריך להיות מחובר לרשת (אם יותר ממחשב אחד אז כולם).
2. יש לוודא כי כל הקבצים הקשורים לפרויקט, כולל תמונות וקטעי שמע, נמצאים באותה התיקייה, אחרת עלולה להיגרם בעיה בפתיחת התמונות בקוד עצמו.
3. יש להריץ את הקובץ server.py ולוודא שהוא רץ, אם הוא לא ירוץ כאשר מנסים לשחק, המשחק לא יעבוד ויאותחל. לאחר ההרצה יש להכניס את כתובת הIP של המחשב (קלט מהמשתמש).
4. יש להריץ את הקובץ client.py, אפשר להריץ אותו פעמיים מאותו מחשב או דרך מחשבים שונים. לאחר ההרצה, תגיעו למסך הפתיחה. כדי לעבור למסך הבא יש ללחוץ על המקש אנטר. במסך הבא צריך להכניס את כתובת הip של השרת בתוך תיבת הטקסט שעל המסך. לאחר לחיצה על המקש אנטר, המשחק מתחיל. אם כתובת הip שהוכנסה היא לא נכונה, או שהשרת לא פועל, המשתמש יחזור למסך הפתיחה.

הוראות המשחק:

במסך הפתיחה יש להקיש על מקש האנטר כדי להתחיל לשחק.

**מקשי המשחק:** A תזוזה שמאלה, D תזוזה ימינה, SPACE קפיצה, S להתכופף. התכופפות ותזוזה שמאלה יגרמו להחלקה שמאלה וכך גם לצד ימין. פגיעה באוייב תוך כדי החלקה תפגע בו ותוריד לו חיים. לחיצה על המקש W תוך כדי קפיצה תגרום לשחקן לעוף. כאשר הוא עף הוא נופל כל הזמן והמקשים פועלים באופן שונה: A לעוף שמאלה, D לעוף ימינה, W לעוף למעלה, S ליפול מהר יותר. השחקן לא יכול לתקוף תוך כדי שהוא באוויר.

המטרה של שני השחקנים היא לפגוע ביריב 15 פעמים עד שהוא מפסיד ואז המשחק מסתיים. יש לשים לב כי כל מכה עלולה להוריד יותר מחיים אחד, תקף לשני הצדדים.

**הערה:** השחקן של המחשב לא מתחיל לזוז עד ששני השחקנים מתחברים למשחק.

סביבת העבודה:

**שפת התכנות:** python גרסה 3.8.

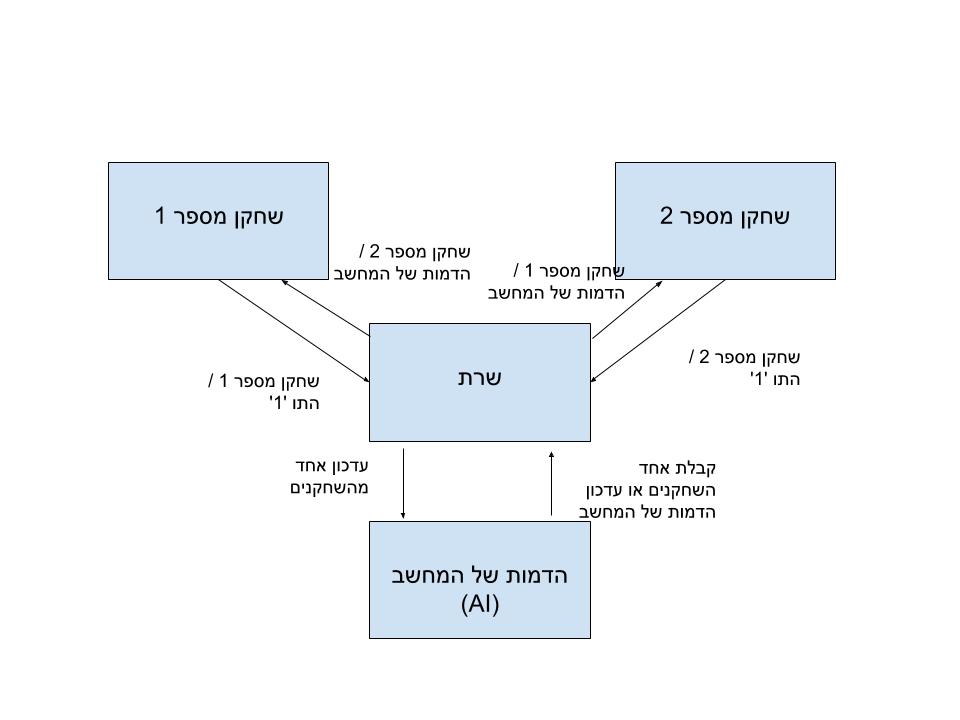
סביבת העבודה לפיתוח היא pycharm. במהלך הפיתוח השתמשתי בספריות הבאות: pygame, socket, time, pickle, threading. בנוסף השתמשתי בתוכנה paint.net כדי לערוך את כל התמונות שהשתמשתי בהן בפרויקט.

מבנה/ ארכיטקטורה של הפרויקט:

החלקים המרכזיים בקוד:

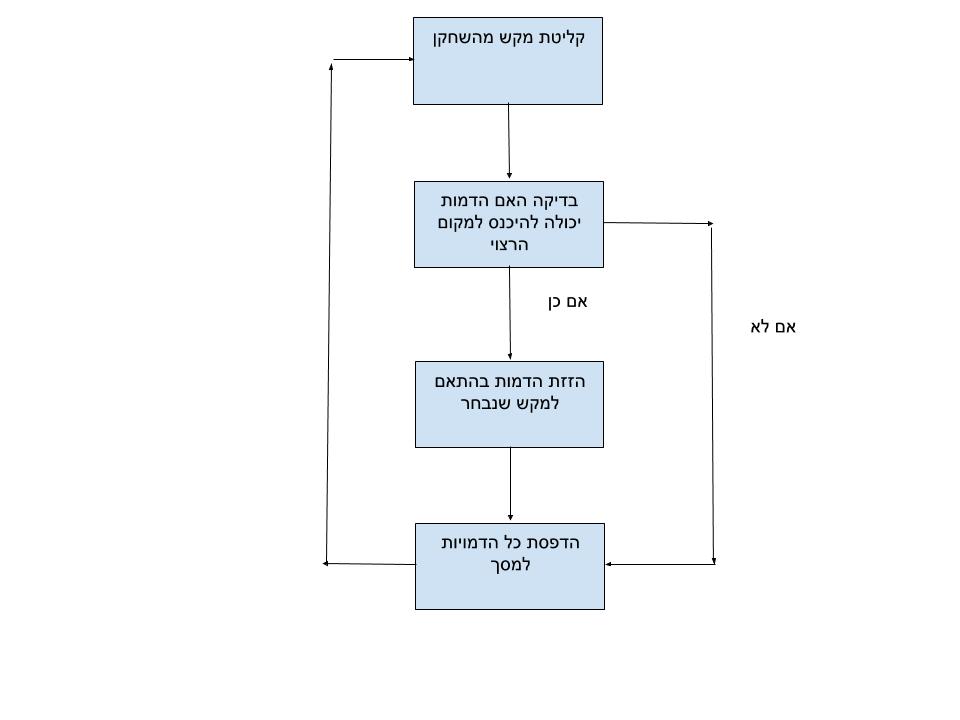
הקוד של המשחק מתחלק לכמה חלקים: שרת, משתמש, AI. בחלק זה אסביר מהו תפקידו של כל אחד מהחלקים ואיך הם קשורים אחד לשני.

שרת: השרת עובד באופן הבא: באמצעות החבילה threading הוא קולט שני משתמשים. לאחר הקליטה כל אחד מהמשתמשים מקבל ממנו את השחקן המתאים עבורו, כלומר המשתמש שהתחבר ראשון מקבל את שחקן מספר 1 (הורוד) והמשתמש השני את השחקן מספר 2 (הצהוב). השרת מעביר מידע למשתמשים בהתאם לבקשתם: הוא ממתין לקלט מהם, ואם הקלט הוא השחקן של המשתמש, הוא יקבל את האוביקט של השחקן השני כדי שיוכל להדפיס אותו. אם הקלט הוא התו '1', המשתמש יקבל את האוביקט של הAI כדי שיוכל להדפיס אותו.



משתמש:הלקוח (client) פועל באופן הבא: לאחר התחלת המשחק (אחרי מסכי הפתיחה) המשתמש בודק האם יש קלט מהמשתמש ופועל בהתאם לקלט. לדוגמא, אם לחץ על המקש A במקלדת, תופעלנה פעולות מתאימות המחשבות את המיקום החדש של הדמות והאנימציה ומדפיסות אותה באופן הבא: ראשית מתקיימת בדיקה האם הדמות יכולה להגיע למקום החדש, כלומר האם אין לה מכשול כמו קצה המסך. אם היא מתאימה, אז המיקום שלה (התכונות x ו-y ) משתנות בהתאם. לאחר מכן מודפס מחדש הרקע כדי שימחקו הדמויות הקודמות, לאחר מכן שולחים לשרת את האובייקט של השחקן כדי שיעדכן אצלו ומקבלים את האובייקט של השחקן השני ומדפיסים את הדמות שלו למסך. עכשיו מקבלים מהשרת את הדמות של המחשב ומדפיסים אותה, ובסוף את הדמות של השחקן הנוכחי.

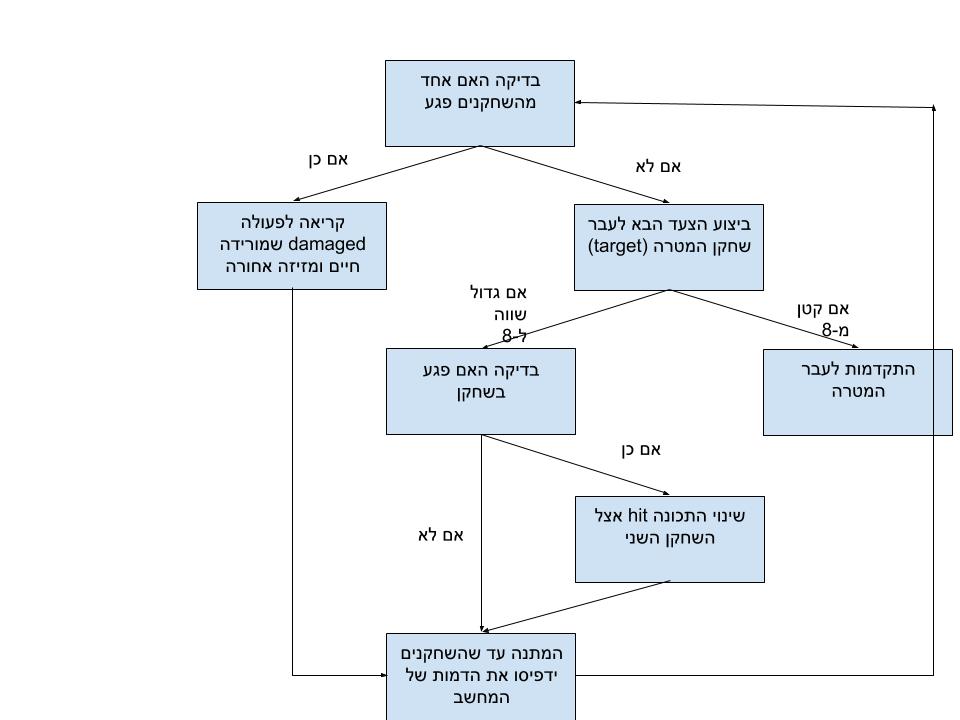
להלן תרשים זרימה המתאר את אופן פעולת הלקוח:



שני האובייקטים של השחקנים נוצרים בתוך הקובץ של הAI ולא בתוך הקובץ של השרת וזאת מכיוון שהאלגוריתם של הAI צריך גישה לשחקנים גם כן והשרת מייבא את הקובץ של הAI. כדי לא לבצע ייבוא מעגלי העדפתי שהשחקנים ישמרו בקובץ של הAI.

המחשב: הAI פועל באופן הבא: הלולאה המרכזית רצה במקביל לשרת באותו המחשב של השרת. בתחילת הלולאה מתקיימת בדיקה, האם נפגע הAI על ידי אחד מהשחקנים. אם כן מופעלות פעולות מתאימות שמשנות את המיקום שלו, החיים שלו ואת האנימציה שלו שייראה כאילו הוא עף אחורה ונפגע. (יש לציין שהדפסות לא מתקיימות בקוד של הAI מכיוון שהוא לא מקושר למסך כלשהו. במקום זה, המיקום והאנימציה משתנים והאובייקט של הדמות שלו נשלח לשחקנים והם מדפיסים אותו) אם הוא לא נפגע, באמצעות התכונה target הוא עוקב אחרי השחקן שהוא מנסה לפגוע בו ומתקדם צעד אחד לעברו. ב7 הצעדים הראשונים הוא רק מתקדם, בשמיני ובתשיעי הוא מתכונן לזריקת הנשק. בצעד העשירי הוא זורק את הנשק לעבר השחקן, ועד הצעד ה15 הנשק עף וחוזר אליו. בכל צעד החל מה-8 ועד ה-15 מתקיימת בדיקה האם הAI פגע באחד מהשחקנים, כלומר האם המיקום של אחד מהם הוא במיקום של הנשק שלו. אם הוא פגע באחד מהם, הוא מעדכן את התכונה hit אצל השחקן השני. (הסיבה לכך היא שהאובייקט של שחקן מועבר לשני בעת ההדפסה. שם אפשר לבדוק האם נפגע שחקן במקום לבצע שליחה נוספת של מידע לשרת וממנו ללקוחות). אם השחקן שהוא פגע בו הוא השחקן שניסה לפגוע בו (target), אז עכשיו הוא מתחיל לעקוב אחרי השחקן השני ולנסות לפגוע בו. לאחר ביצוע הצעד והבדיקות, האלגוריתם ממתין בלולאה המרכזית עד ששני השחקנים ידפיסו למסך את הדמות של המחשב, ואז מתקדם לצעד הבא.

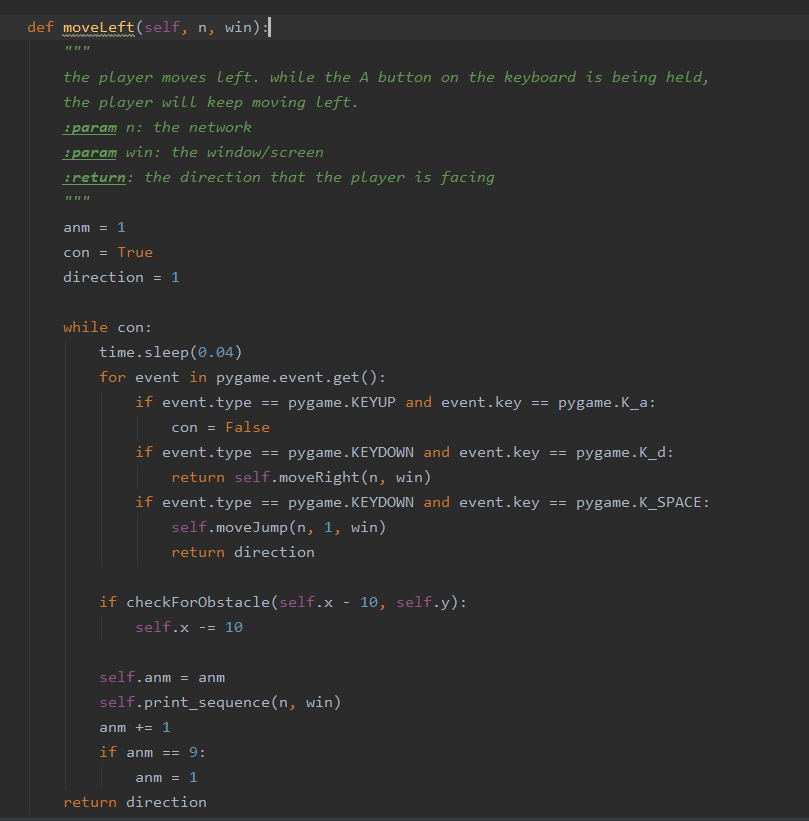
להלן תרשים זרימה הממחיש את אופן פעולת הAI:



פעולות מרכזיות בקוד:

בקטע זה אסביר על אופן הפעולה של הפעולות המרכזיות בקוד, תזוזה שמאלה (כל פעולות התזוזה דומות אחת לשניה עם מעט הבדלים ביניהן), שליחה וקבלת מידע (מצד השרת) והלולאה המרכזית של הAI.

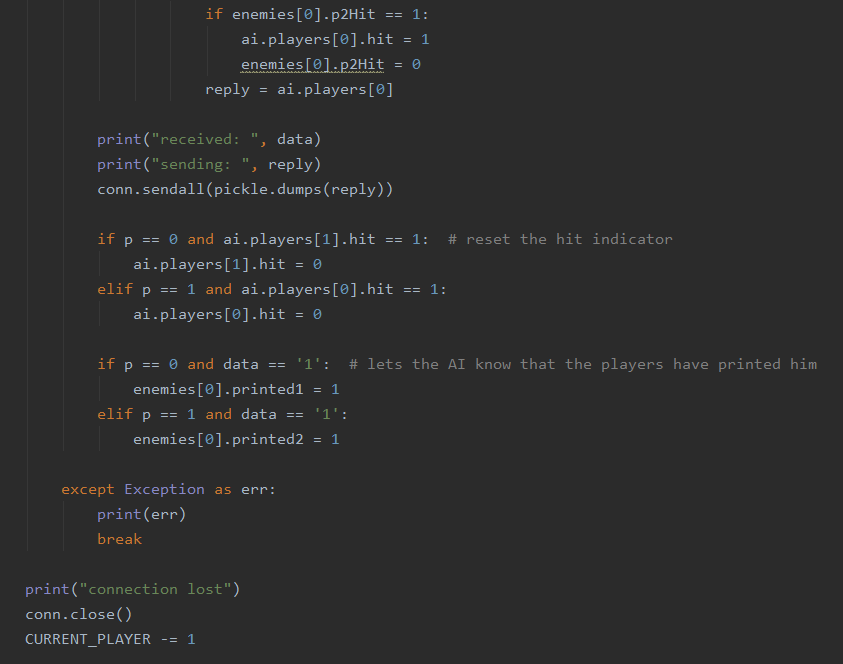
תזוזה שמאלה:להלן הקוד של הפעולה:



הפעולה שייכת למחלקה Player (הסבר עליה בהמשך). הפעולה פועלת באופן הבא: הלולאה רצה כל עוד לא שוחרר המקש A. אם נלחץ המקש D, נקראת הפעולה לתזוזה ימינה ואם נלחץ מקש הרווח, נקראת הפעולה לקפיצה. הסיבה לכך היא שהתנועות של הדמות הופכות להיות הרבה יותר חלקות. במקום שהדמות תעצור ורק אז תקפוץ, היא יכולה לקפוץ תוך כדי תזוזה או לשנות כיוון. לפני ההדפסה נקראת הפעולה checkForObstacle שמטרתה לבדוק האם הדמות הגיע לדפנות המסך או לא. אם לא אז מיקום ה-x של הדמות משתנה. לאחר מכן משנים את האנימציה של הדמות כך שזה יראה כאילו היא הולכת. לבסוף מדפיסים את כל הדמויות למסך ומגדילים את האנימציה כל פעם כדי שהתמונה תשתנה. אנימצית התזוזה שמאלה היא מ-1 עד 8, לכן התנאי בסוף נועד להחזיר את האנימציה ל-1 אם היא עלתה מעל 8. הפעולה מחזירה את הכיוון שאליו הדמות מסתכלת, 0 עבור ימינה ו-1 עבור שמאלה.

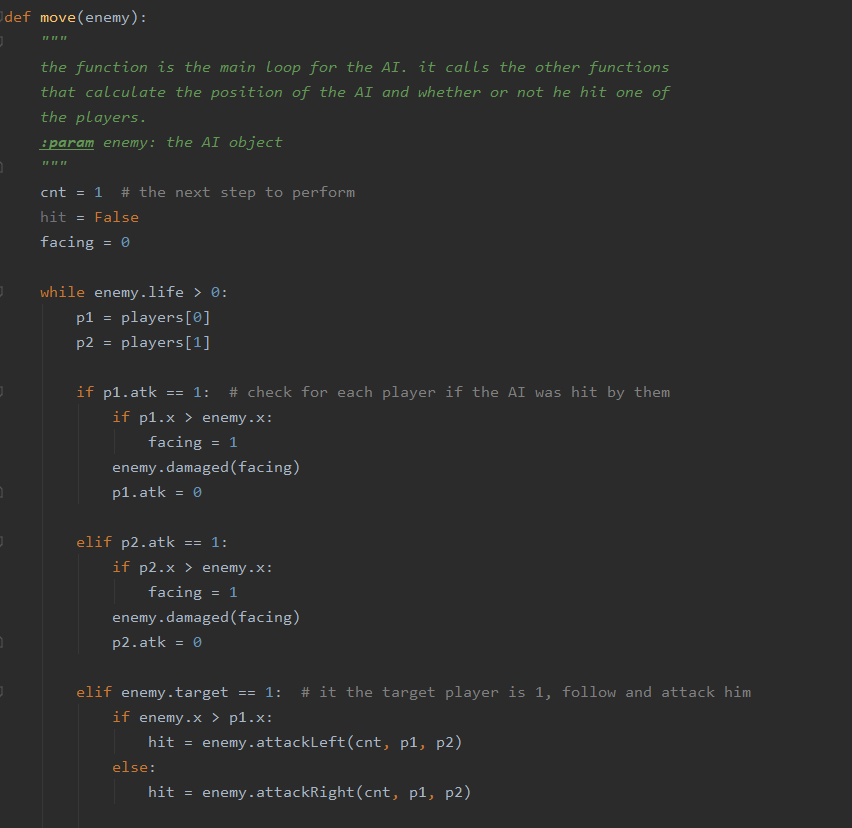
**שליחה וקבלת מידע (שרת):** להלן הקוד של הפעולה:

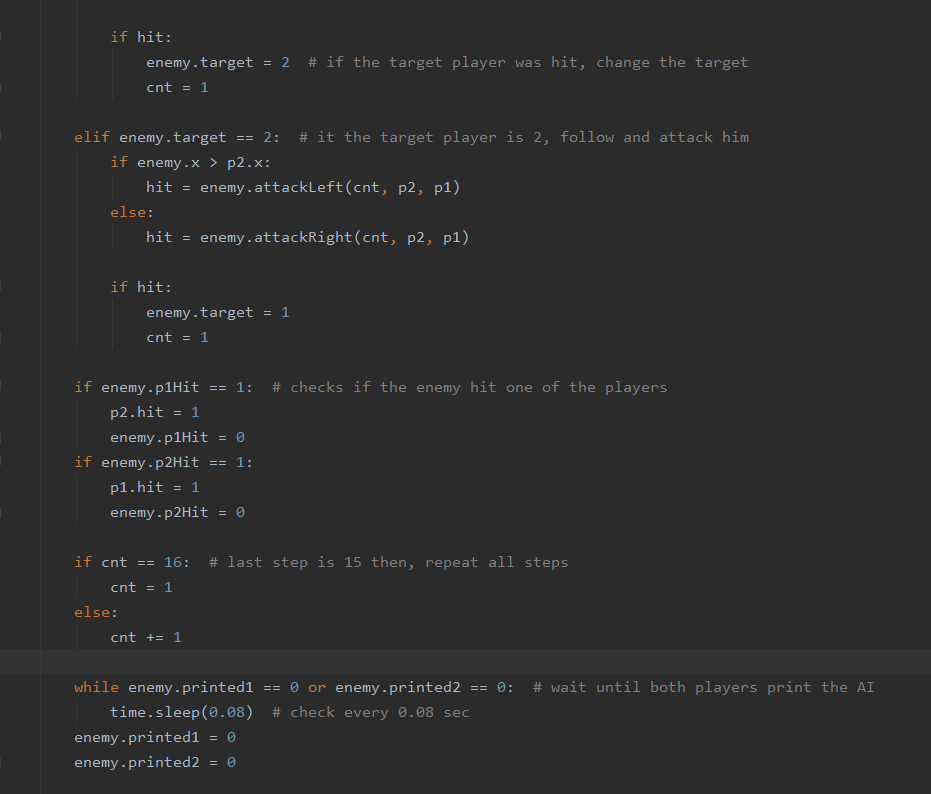




הפעולה פועלת באופן הבא: הפעולה פועלת ברגע ששחקן מתחבר לשרת. ברגע ששחקן מתחבר לשרת, הוא מחכה לקלט ממנו. הקלט הוא האובייקט של השחקן ולכן בהתחלה (לפני הלולאה) מתבצעת שליחה של האובייקט. בלולאה עצמה, השרת קולט מידע מהשחקן. אם המידע שקיבל הוא התו '1' השרת יודע להחזיר את האובייקט של הAI. אם המידע שקיבל הוא שונה מהתו '1', השרת יודע שהוא קיבל את האובייקט של השחקן, לכן הוא מעדכן אותו ברשימה שנמצאת בקובץ ai.py ומחזיר לשולח את הקובץ של השחקן השני. אם השרת לא קיבל מידע, זה אומר שהשחקן התנתק ויש לסגור את החיבור.בעת סגירת החיבור, כמות השחקנים המחוברים יורדת ב-1. לפני השליחה, מתקיימת בדיקה האם הAI פגע בשחקן השולח. אם כן, כאשר השרת אמור לשלוח לו בחזרה את האובייקט של השחקן השני, הוא מעדכן אצלו את התכונה hit כך שהשולח יוכל לבדוק האם נפגע. התכונה תתאפס ברגע שהשחקן השני ישלח את האובייקט של עצמו והוא יתעדכן ברשימה. לאחר השליחה, אם האובייקט של הAI נשלח, השרת מעדכן אצלו שאותו שחקן שקיבל את האובייקט, הדפיס את הדמות למסך, כדי שהAI יוכל להמשיך לצעד הבא שלו.

**הלולאה המרכזית של הAI (הפעולה move):** להלן הקוד של הפעולה:





הפעולה פועלת באופן הבא: הלולאה המרכזית רצה כל עוד הדמות של המחשב נמצאת בחיים. בתחילת הלולאה יש בדיקה האם הדמות נפגעה על ידי אחד מהשחקנים. אם כן, קוראים לפעולה damaged שמזיזה את הדמות אחורה ומורידה לו חיים. אם לא אז ממשיכים בלולאה. לאחר מכן יש בדיקה אחרי איזה שחקן צריך לעקוב ומה המיקום שלו ביחס לדמות של המחשב (שמאלית או ימנית לה). לאחר הבדיקה נקראת הפעולה attackLeft או attackRight בהתאם למיקום השחקן. הפעולות מבצעות את הצעד הבא ברצף ההתקפה ומחזירות האם הדמות פגעה בשחקן המטרה. אם כן אז המטרה משתנה לשחקן השני. עבור כל אחד מהשחקנים, מתרחשת בדיקה האם הדמות של המחשב פגעה בהם. אם כן אז התכונה hit מעודכנת אצל השחקן השני כאינדיקציה שהנוכחי נפגע.

בסוף המשתנה cnt שמונה את הצעדים בהתקפה של הAI גדל ב-1 והקוד ממתין עד ששני השחקנים ידפיסו את הדמות של המחשב לפני שהלולאה הראשית חוזרת על עצמה. כפי שניתן לראות בלולאת ההמתנה יש עיכוב של 0.08 שניות. הסיבה לכך היא שכאשר לא היה עיכוב זה תקע את כל המשחק והוא רץ ממש לאט, כנראה בגלל שהלולאה יצרה עומס על המערכת.

האלגוריתם המרכזי:

**אופן הפעולה של הAI:** הAI מתחיל בכך שהוא עוקב אחרי השחקן הראשון. מתקיימת השוואה בין המיקומים של הAI ושל השחקן, אם הוא נמצא שמאלית אליו, הAI יזוז שמאלה, ואם ימנית, ימינה. לאחר שהAI זז לכיוון השחקן מספר פעמים, הוא מנסה לתקוף אותו בכך שהוא זורק את הנשק לעברו. אם המיקום של הנשק יהיה אותו המיקום של השחקן, תתרחש פגיעה, כלומר השחקן מאבד חיים ונהדף לאחור. מכיוון שגם הנשק וגם השחקן הם בעלי תיבת פגיעה ריבועית, מספיק שלשני הריבועים תהיה נקודה אחת משותפת כדי שתהיה פגיעה. לאחר שפגע בשחקן הראשון, הAI יעקוב אחרי השחקן השני ומנסה לפגוע בו באותו האופן. התהליך חוזר על עצמו עד שהשחקנים מפסידים או עד שהAI מפסיד.

**אופן הפגיעה של השחקן בדמות של המחשב:** כאשר השחקן מחליק (המקשים S + A/D במקלדת) לאחד הכיוונים, מתקיימת בדיקה האם המיקום שלו שווה למיקום של הAI. אם כן, תעודכן אצלו התכונה atk שמסמלת שהוא פגע בAI. האובייקט של השחקן יועבר בשרת ולאחר מכן, הAI בודק האם אחד מהשחקנים פגע בו. אם כן, תופעל פעולה מתאימה שמורידה לו חיים ומזיזה אותו אחורה.

**אופן הפגיעה של הAI בשחקן:** לאחר שהנשק פוגע בשחקן, תעודכן אצל השחקן השני (האחד שלא נפגע) התכונה hit. כאשר השחקן מדפיס למסך את כל הדמויות, הוא מקבל מהשרת את האובייקט של השחקן השני כדי שיוכל להדפיס אותו. בזמן הזה יש בדיקה האם הוא נפגע, ואם כן, ירד לו חיים אחד וההדפסה של הדמות שלו תשתנה כך שהוא נהדף לאחור.

הסברים על הקבצים השונים:

**הצגת החלוקה לקבצים:** את הפרויקט החלטתי לחלק לקבצים הבאים: player, kirby, network, client, server, ai. חילקתי את הקבצים באופן שהיה נראה לי הכי הגיוני, כלומר פעולות הדומות בתפקידן יהיו באותו הקובץ. להלן הסברים על כל הקבצים השונות, ותפקידן.

kirby:

בקובץ זה קיימות הפעולות הקשורות בהדפסת תמונות למסך ופלט למסך. בין הפעולות: פעולה המקבלת אנימציה המיוצגת על ידי מספר וקוראת לפעולה נוספת שמדפיסה תמונה לפי האנימציה שקיבלה, פעולה המדפיסה את מסך הרקע לפי מספר שהיא מקבלת. בנוסף ביחידה זו נפתחות כל התמונות שנעשה בהן שימוש.

המשתנים בקובץ:

כל המשתנים בקובץ הם למעשה תמונות. שמות המשתנים מתארים את התמונה שהם מכילים. למשל, KML1 זה קיצור ל kirby move left 1, ו E1ML1 זה קיצור ל enemy 1 move left 1.

הפעולות בקובץ:

* פעולה למיון האנימציה עבור הדמויות של השחקנים.
* פעולה למיון האנימציה עבור הדמות של המחשב.
* פעולה להדפסת תמונה למסך.
* פעולה להדפסת הרקע למסך.
* פעולה להדפסת הנשק של הדמות של המחשב.
* פעולה להדפסת מסך הפתיחה.

player:

בקובץ זה קיימת המחלקה Player המייצגת שחקן. הקובץ מכיל גם את כל הפעולות הקשורות בהתנהלות מול הקלט מהמשתמש והפעולות הקשורות למיקומים על המסך (השוואה בין מיקומים, בדיקה האם דמות נכנסת למקום מסוים). רוב הפעולות ביחידה שייכות למחלקה Player.

המשתנים בקובץ:

המשתנה LIFE - כמות החיים שיש לשחקן (5).

בתוך הפעולות יש שימוש במשתנים: anm - האנימציה הנוכחית של הדמות, direction/ dir/ facing - הכיוון שאליו הדמות מסתכלת (0 - ימינה, 1 - שמאלה). כל שימוש ב-x ו-y הכוונה למיקום על המסך. חלק מהפעולות מקבלות גם את הפרמטרים n ו-win שהם הnetwork שדרכו מתקשרים עם השרת והמסך שאליו מדפיסים.

הפעולות בקובץ:

* פעולה להשוואת מיקומים של דמויות על המסך.
* פעולה לבדיקה האם דמות יכולה להגיע למקום מסוים, כלומר האם היא הגיע לקצה המסך או לא.
* פעולה הבודקת האם דמות נמצאת באוויר.
* פעולה הקולטת מקש מהמקלדת וקוראת לפעולות בהתאם למקש שנקלט (לדוגמא עבור המקש A נקראת פעולה שמזיזה את הדמות שמאלה).

\*שאר הפעולות בקובץ שייכות למחלקה Player

**Player -** המחלקה מייצגת שחקן. היא מכילה את כל הפעולות הקשורות לעבודה מול המשתמש. התכונות שלה הן:

**מיקום על ציר X ומיקום על ציר Y:** ביחד הן המיקום של דמות השחקן על המפה.

**אורך הדמות (בפיקסלים):** הדמות היא ריבועית, אורך הדמות הוא אורך הצלע בפיקסלים.

**מספר:** מספר השחקן (1 או 2) חשוב לדעת להבדיל בין השחקנים כי דמות השחקן הראשון ורודה והשניה צהובה.

**חיים:** החיים של הדמות, מתחיל מ5 ויורד כל פעם שהיא נפגעת. החיים מופיעים בתחתית המסך בעת המשחק.

**אנימציה:** האנימציה של הדמות בכל רגע ורגע. האנימציה מיוצגת על ידי מספר שפעולה אחרת מתאימה אותו לתמונה. חשוב לשמור את האנימציה כי היא צריכה לעבור גם לשחקן השני כדי שידפיס אותה.

**באוויר:** תכונה זו שווה ל1 אם הדמות נמצאת באוויר.

**נפגע:** תכונה זו שווה ל1 אם השחקן השני נפגע מהדמות של המחשב.

**תוסף:** התוסף יהיה שווה ל0 אם מדובר בשחקן מספר 1 ול42 אם מדובר בשחקן מספר 2. התוסף מתווסף לאנימציה בכל הדפסה כדי שהשחקן השני יהיה צהוב ולא ורוד כמו השחקן הראשון.

**הפעולות במחלקה:** תזוזה לשני הכיוונים, קפיצה, התכופפות, החלקה לשני הכיוונים, תעופה, פגיעה (פגעו בשחקן).

client:

כאן נמצאת הלולאה המרכזית שרצה בקוד. בהתחלה יש הגדרה של כל המשתנים שהמערכת צריכה כדי לעבוד כמו אורך ורוחב המסך, הדפסה של מסך הפתיחה (קריאה לפעולה המדפיסה אותו). בהמשך נמצאת לולאה שמחכה לקלט של התו רווח מהמשתמש, ואחריה הלולאה המרכזית. היחידה משמשת כפעולה הראשית שמפעילה את כל הפעולות ביחידות האחרות (למעט server ו-ai).

המשתנים בקובץ:

menu screen - מכיל את התמונה של מסך הפתיחה.

display\_height - הגובה של חלון המשחק.

display\_width - הרוחב של חלון המשחק.

game\_display - האובייקט של חלון המשחק.

clock - אובייקט של שעון שבאמצעותו ניתן להגדיר את ה FPS.

n - אובייקט מסוג Network, דרכו מתקשרים עם השרת.

p1 - האובייקט של השחקן. מקבלים אותו דרך השרת.

network:

הקובץ מכיל את המחלקה Network בלבד, למעט שני משתנים נוספים, IP ו - PORT שהמחלקה משתמשת בהם כדי להתחבר לשרת.

**Network -** המחלקה מכילה את הsocket שמתחבר לשרת והיא בעצם הדרך של השחקנים לתקשר איתו. התכונות שלה הן:

**לקוח (client):** כאן יושב הsocket שמתחבר לשרת.

**שרת (server):** התכונה מכילה את הכתובת של השרת.

**פורט:** התכונה מכילה את הפורט שאליו יש להתחבר.

**כתובת:** מכיל את שתי התכונות שרת ופורט כדי ליצור כתובת שניתן להתחבר אליה.

**פי (האות פי באנגלית):** התכונה הזאת תהיה שווה לשחקן הראשון (לאובייקט ממחלקת שחקן) בעת ההתחברות לשרת.

**הפעולות במחלקה:** התחברות לשרת, שליחת מידע לשרת וקבלת מידע מתאים ממנו בחזרה (בפעולה אחת).

ai:

בקובץ נמצאת המחלקה AI, והפעולה שמריצה את הדמות של המחשב. הפעולה הזאת היא הלולאה המרכזית של הAI והיא פועלת כל עוד משחקים. הפעולה מקבלת את האוביקט של הAI (מצביע לאוביקט) ומבצעת עליו פעולות שונות כמו למשל תזוזה שמאלה וימינה, תקיפה ופגיעה.

משתנים בקובץ:

LIFE - כמות החיים של היריב (15).

players - רשימה המכילה שני אובייקטים מסוג Player. הרשימה מתעדכנת כל פעם ששחקן מבצע תנועה או שינוי כלשהו באנימציה ודרכה ניתן להעביר לשחקן את המידע הדרוש כדי להדפיס את השני.

בתוך הפעולות יש שימוש במשתנים: anm - האנימציה הנוכחית של הדמות, direction/ dir/ facing - הכיוון שאליו הדמות מסתכלת (0 - ימינה, 1 - שמאלה). כל שימוש ב-x ו-y הכוונה למיקום על המסך. step/cnt - נועד למנות את צעדי התקיפה של הAI כדי שהתמונה תשתנה בהתאם. hit - במקרה והדמות של המחשב פגעה בשחקן שהיא עוקבת אחריו, ערך המשתנה יהיה 1 וזו אינדיקציה לכך שעכשיו יש לתקוף את השחקן השני.

הפעולות בקובץ:

אין פעולות נוספות למעט הפעולות של המחלקה AI והלולאה המרכזית שנמצאת בפעולה move.

**AI -** המחלקה מייצגת את הדמות של המחשב ומכילה את כל הפעולות הקשורות בחישוב המיקום והאנימציה שלו וגם האם הוא פגע בשחקן. תכונות המחלקה הן:

**מיקום על ציר X ומיקום על ציר Y:** ביחד הן המיקום של דמות השחקן על המפה.

**אורך הדמות (בפיקסלים):** הדמות היא ריבועית, אורך הדמות הוא אורך הצלע בפיקסלים.

**חיים:** החיים של הדמות, מתחיל מ5 ויורד כל פעם שהיא נפגעת. החיים מופיעים בתחתית המסך בעת המשחק.

**אנימציה:** האנימציה של הדמות בכל רגע ורגע. האנימציה מיוצגת על ידי מספר שפעולה אחרת מתאימה אותו לתמונה. חשוב לשמור את האנימציה כי היא צריכה לעבור לשני השחקנים כדי שידפיסו אותה.

**מטרה:** איזה שחקן הוא תוקף (1 או 2)

**מיקום הנשק:** (1 - 5) המרחק של הנשק מהדמות, 1,5 המרחק הוא 50, 2,4 המרחק הוא 100, 3 המרחק הוא 150.

**הודפס על ידי השחקנים:** (2 תכונות, אחת עבור כל שחקן) התכונה שווה ל1 אם השחקנים הדפיסו את הדמות. תכונה זו חשובה מכיוון שהAI רץ במקביל לשחקנים וכדי שהוא לא יעקוף אותם וידלג על הדפסות, כל הרצה של הלולאה המרכזית הוא מחכה שהשחקנים ידפיסו אותו למסכים ורק אז ממשיך.

**פגע באחד השחקנים:** (2 תכונות, אחת עבור כל שחקן) התכונה שווה ל1 אם הדמות של המחשב פגעה באחד מהשחקנים.

**הפעולות במחלקה:** ביצוע הצעד הבא שמאלה (התקדמות/ תקיפה שמאלה), ביצוע הצעד הבא ימינה (התקדמות/ תקיפה ימינה), פגיעה (ב-AI).

server:

בקובץ נמצא הקוד של השרת. הקובץ הזה צריך לרוץ ראשון כדי שהמשחק יעבוד. ברגע ההרצה, נוצר הsocket שממתין להתחברות משני השחקנים. באמצעות החבילה threading השרת מתחבר לשני השחקנים במקביל וקורה לפעולה המשמשת כלולאה המרכזית של השרת. בפעולה השרת מחכה לקלט מהלקוח ומחזיר לו פלט בהתאם.

משתנים בקובץ:

SERVER - כתובת הip של השרת.

PORT - הפורט של השרת.

enemies - רשימה המכילה את האובייקט של הAI.

s - ה-socket מצד השרת.

CURRENT\_PLAYER - כמות השחקנים הנוכחית.

פעולות בקובץ:

* פעולה הפועלת בתהליכון (thread) נפרד עבור כל שחקן. הפעולה משמשת כלולאה המרכזית של השרת ודרכה עובר המידע אל השרת וממנו.

מבנה הנתונים:

בפרויקט אין נתונים שצריכים להישמר לטווח הארוך, אבל יש נתונים זמניים שצריכים להישמר לאורך המשחק. הנתונים הנשמרים במהלך המשחק הם האובייקטים של שני השחקנים ושל הAI (סה"כ שלושה אובייקטים). האובייקטים נשמרים בשתי רשימות (לא מקושרות, אלא רשימות רגילות, list) אחת לשחקנים ואחת לAI.

רפלקציה:

איך הייתה העבודה עבורי:

אני נהניתי לעבוד על הפרויקט, במיוחד שהכל הלך כמו שתכננתי. אהבתי את העובדה שנתנו לנו חופש פעולה לכתוב פרויקט באיזה נושא שנבחר. בתחילת העבודה, כשהתבקשנו לבחור נושא לפרויקט, לא לקח לי הרבה זמן לבחור, כבר ידעתי בדיוק מה אני רוצה לעשות. בשלבים ההתחלתיים התקדמתי הרבה בכתיבת הפרויקט, אז הרשיתי לעצמי להאט את הקצב ולעבוד עליו פעם או פעמיים בשבוע. אחד החלקים שלקחו לי הרבה זמן בתחילת העבודה הוא למצוא תמונות של הדמות ולערוך אותן כדי שיתאימו לי. בפרויקט יש שימוש בכ-100 תמונות. לאחר כל ההתעסקות עם התמונות, יכולתי להתחיל לבדוק את הקוד ולראות אם מה שכתבתי עובד. להפתעתי הכל עבד בדיוק כמו שרציתי (אולי היו כמה טעויות קטנות) ויכולתי להמשיך בעבודה.

כלים שאני לוקח איתי להמשך:

בשונה מפרויקטים אחרים שעשיתי בעבר, הפעם החלטתי שאני אתכנן יותר לפני שאני כותב, במקום לכתוב ולקוות לטוב. בזכות השיטה הזאת כמעט ולא התמודדתי עם שגיאות, אבל תהליך הכתיבה היה איטי יותר.

אתגרים שעמדו בפניי:

במהלך הכתיבה עמדתי בלא מעט אתגרים שונים. חלקם היו בכתיבת הקוד ולאחרים אין קשר לכך. קושי ראשון שהיה לי הוא שילוב כל המערכת של הAI בתוך הפרויקט. בכתיבת הפרויקט, עבדתי קודם כל על השחקנים, כלומר המחלקות של השחקנים, כל הפעולות הקשורות להם ולגרום לכך ששני שחקנים יוכלו לשחק יחד באותו משחק. לאחר שסיימתי עם החלק הזה ובדקתי שהכל עובד כמו שצריך, כתבתי את כל מה שקשור לAI וניסיתי לשלב אותו בפרויקט.

אתגר שני שהתמודדתי איתו היו בעיות טכניות במחשב הפרטי שלי. עד תקופה הבידוד, עבדתי על הפרויקט במחשבים של בית הספר אבל לאחר שלא יכולנו לצאת מהבית, עבדתי על הפרויקט במחשב שלי. בערך בתחילת חודש אפריל הייתה לי בעיה במחשב. המחשב קרס כמה וכמה פעמים ברצף ופחדתי שאני לא אצליח לפתוח אותו ושכל העבודה שלי תימחק. לבסוף סידרתי את הבעיה וחזרתי לעבוד כרגיל.

מסקנות:

לא להשאיר את כל העבודה לרגע האחרון ולתכנן לוח זמנים מסודר ולעמוד בו. בעבודה על הפרויקט תכננתי נכון את הזמנים ובכך מנעתי מצבי לחץ מיותרים ותהליך העבודה היה מהנה ופשוט יותר.

ביבליוגרפיה

* ספר גבהים פייתון.
* ספר גבהים רשתות
* סרטון ביוטיוב: מדריך לכתיבת משחק מולטיפלייר בפייתון <https://www.youtube.com/watch?v=McoDjOCb2Zo&list=LLHuQ4old88wxK2mDDN2L6cQ&index=4&t=0s>
* האתר שממנו הבאתי את כל התמונות (האתר לא קיים יותר)