תיק פרויקט

בית חינוך ניסויי ע"ש יצחק רבין תל מונד

שם בית ספר: רבין תל מונד

שם הפרויקט: במבה בממלכה הקסומה

שם התלמיד: יובל בכר

שם החלופה: הנדסת תוכנה

ת"ז התלמיד: 212553242

שם המורה: ניר סליקטר

תאריך הגשה: 31.5.2020

קישור לקוד: https://github.com/ yuvalb21/final-project

תוכן העניינים

מבוא 3
תיאור תכולת הספר 3
הרקע לפרויקט 3
תהליך המחקר 3
אתגרים מרכזיים 3
הצגת פתרונות לבעיה 3
מדריך למשתמש - הוראות 4
הרשמה 4
מגדל 6
מפת המשימה 7
דו קרב 9
כפתור escape 11
תרשים זרימה של המסכים במשחק 12
בסיס נתונים 13
מדריך למפתח 14
הרשמה 14
מגדל 19
משימה 22
דו קרב 31
ספריות 43
רפלקציה אישית 43
ביבליוגרפיה 44

מבוא

תיאור תכולת הספר המסמך מיועד לבוחנים אשר יבדקו את הפרויקט ובעזרת המסמך יוכלו להבין את התהליך המלא של בניית הפרויקט.

הרקע לפרויקט הפרויקט שאני בחרתי לעשות מבוסס על משחק מחשב שנקרא במבה בממלכה הקסומה ורציתי לשחזר אותו כיוון שהוא נמחק לפני שלוש שנים וזה היה המשחק האהוב עלי בזמנו.

תהליך המחקר כפי שציינתי קודם, המשחק מבוסס על משחק שהיה קיים ונמחק, כך שהדרישה שלו בשוק יכולה הייתה להיות רלוונטית, כמו כן אין משחק באותו הסגנון שהיה יכול להתחרות בו. כדי לבנות את הפרויקט היה עלי לעבוד בסביבת עבודה הנקראת unity, שמטרתה היא לפתח משחקים, וכשלמדתי על הסביבה גיליתי שלא מעט משחקים שאני מכיר פותחו בה, כך שידעתי שאוכל לעבוד איתה. לשם העבודה בunity היה עלי לצפות בסרטונים ב-YouTube של אנשים שפיתחו פרויקט בסביבה זו ובעזרת הסרטון לדעת אילו כלים ישנם כדי שאוכל להשתמש בהם בפרויקט שלי. זמן החקירה ערך כשבועיים, כדי שאוכל לדעת באופן בסיסי ויותר את הסביבה ותוך כדי בניית הפרויקט להעמיק את הידע בה.

אתגרים מרכזיים

בעיות שהיו לתלמיד במהלך הפרויקט: האתגר הראשוני היה ללמוד את סביבת העבודה unity כדי להתחיל ליצור את הפרויקט. לאחר מכן היו אתגרים בבניית קטעי קוד מסוימים שלשמם הייתי צריך לשלב מספר לולאות ויצירת משתנים שיעזור לי לבדוק מספר אפשרויות. אתגר זה חזר על עצמו בכמה פונקציות. אתגר נוסף שהיה לי הוא יצירת הקשר, הכנסת והוצאת המשתנים ועדכון טבלת המשתמשים בdatabase והנתונים שיש לשמור לגבי כל שחקן.

הסיבה לבחירת הנושא: המשחק היה בזמנו המשחק האהוב עלי, וכיוון שהוא נמחק רציתי לשחק בו שוב והדרך הכי טובה לעשות משהו היא לעשות אותו בעצמך.

מוטיבציה לעבודה: כמו שציינתי בסעיף מעל, רציתי ליצור שוב את המשחק. בנוסף, נהניתי לעבוד עליו כך שזה הגביר את העבודה שלי.

על איזה צורך הפרויקט עונה: הפרויקט נועד לכיף ולהנאה של המשחקים בו.

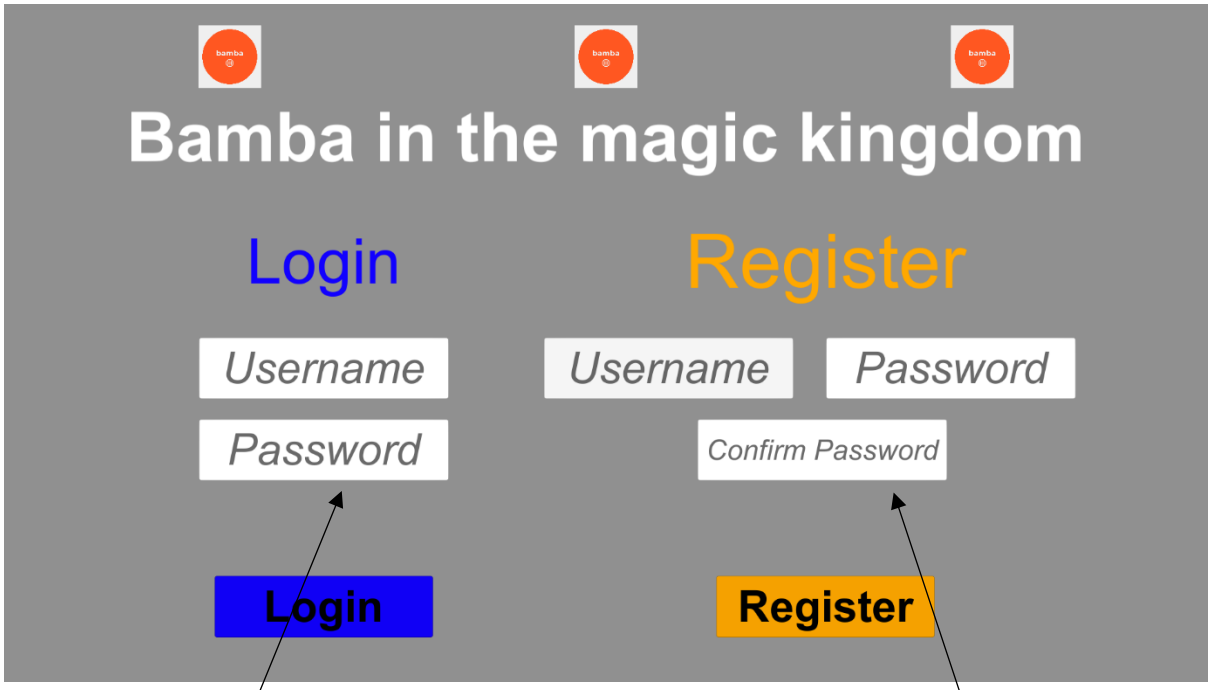
הצגת פתרונות לבעיה כדי לבנות את הפרויקט הייתי צריך ללמוד לעבוד עם unity ולכן צפיתי בפרויקטים לדוגמא שאנשים בנו ובסרטונים בYouTube על תהליך בניית פרויקטים של אנשים. לגבי הבעיה של הפונקציות שבהן הייתי צריך למצוא פתרון לכל אפשרות שיכולה להיגרם כתוצאה ממנו, הייתי צריך לחשוב על פתרונות יצירתיים, וגם נעזרתי בחברים שיעצו לי על אלגוריתמים שהם חשבו

עליהם או השתמשו בהם בפרויקט שלהם. הבעיה של יצירת הקשר וכתיבת וקריאת המידע מהdatabase פתרתי יחד עם חבר שגם הוא עבד עם אותו הdatabase.

מדריך למשתמש - הוראות

הרשמה

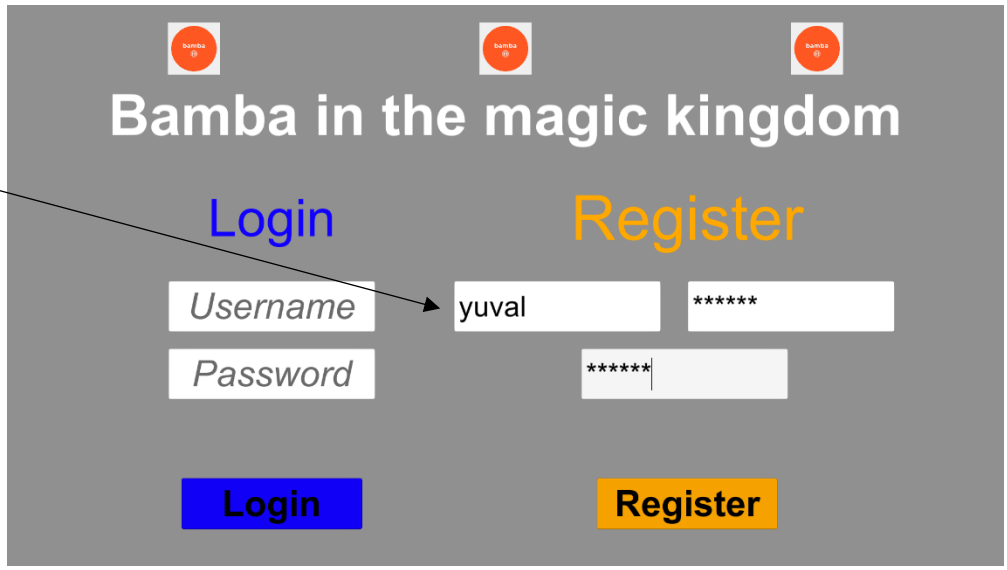
השחקן מתחיל את המשחק במסך הרשמה – גם אם אין לך עדיין משתמש וגם אם יש לך משתמש קיים. אם עדיין המשתמש לא יצר שחקן הוא צריך להירשם – ליצור שם משתמש וסיסמא, ולחזור על הסיסמא פעם נוספת כדי שזאת בוודאות תהיה הסיסמא שהוא בחר. אם למשתמש כבר יש שחקן הוא יצטרך להזין את השם משתמש והסיסמא אותם בחר כשנרשם.



הרשמה לשחקן קיים שבו הוא מזין את השם משתמש והסיסמא שאיתם הוא נרשם ואז עליו ללחוץ על הכפתור הכחול

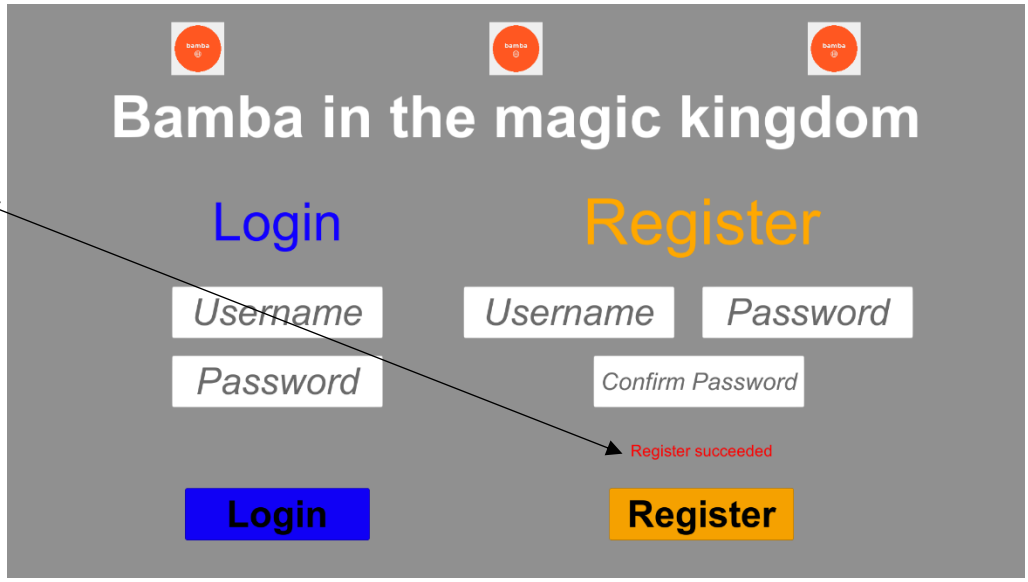
הרשמה לשחקן חדש ולאחר הזנת שם המשתמש, הסיסמא ואז לחזור על הסיסמא סיסמא יש ללחוץ על הכפתור הכתום

<u>דוגמה להרשמה</u>:

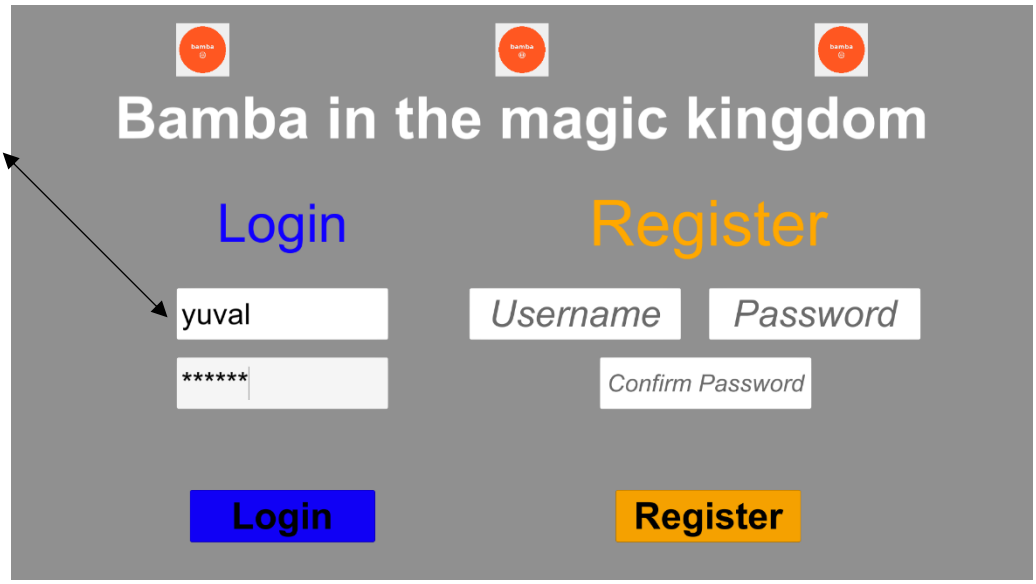


השחקן הכניס את שם המשתמש, הסיסמא ופעם נוספת את הסיסמא

ניתן לראות שההרשמה הצליחה



לאחר ההרשמה השחקן מתחבר לשם משתמש והסיסמא שבחר



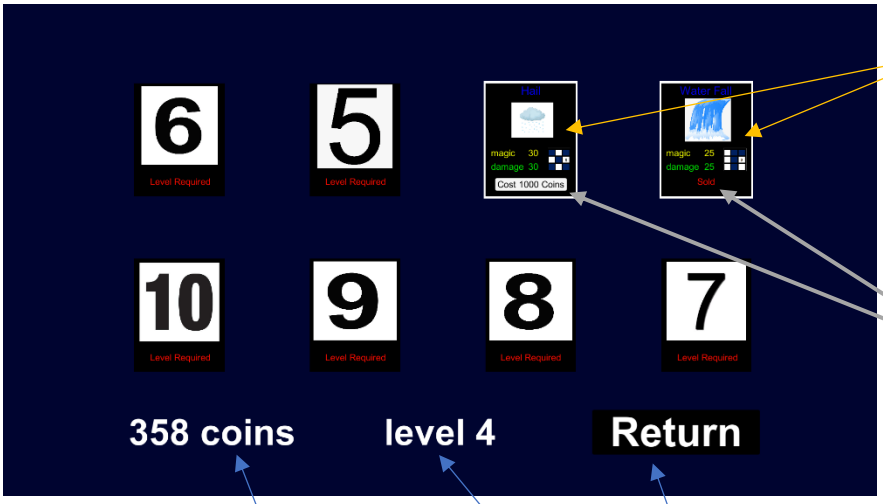
מגדל

לאחר החיבור למשתמש השחקן מגיע למגדל, כאשר ילחץ על הדלת המסומנת בכתום הוא יצא למפת המשימה, וכשילחץ על הקדרה המסומנת בירוק הוא יגיע לחנות הכשפים בה בהתאם לרמתו ולכמות הכסף שברשותו יוכל לקנות כשפים חדשים.

הדלת שבלחיצה עליה יצא השחקן למפת המשימה

הקדרה שבלחיצה עליה יגיע השחקן לחנות הכשפים

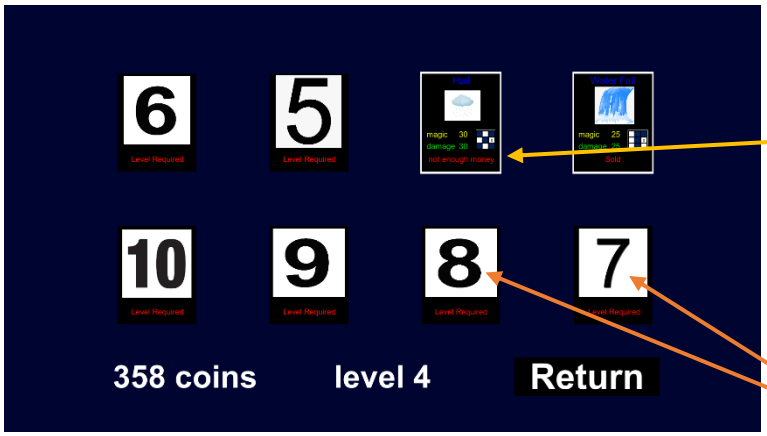
ניתן לראות שהשחקן ברמה 4 ולכן הכשפים שהוא יכול לקנות הם אלו. כל רמה שאתה עולה נפתחת לך אפשרות לקנות כישוף נוסף, בהתאם לכסף שיש לך. המחיר של הכישוף כתוב על כפתור הקנייה שלו, ואם כבר רכשת את הכישוף הכפתור לא יופיע ויהיה כתוב לך שכבר קנית אותו.



הרמה של השחקן והכסף מופיעים בתחתית החנות ויש כפתור חזרה למגדל

כאשר מנסה השחקן לרכוש כישוף ואין לו מספיק כסף הכפתור יעלם ותופיע הודעה שאומרת שאין לו מספיק כסף.

לכשפים שבשבילם צריך להיות ברמה יותר גבוהה משל השחקן לא יהיה כפתור קנייה ויופיע השלב בו השחקן צריך להיות



מפת המשימה

המשימה של השחקן היא להביס את כל האויבים על המפה. תוך כדי המשימה הוא יכול לצבור כסף וניסיון (Exp). השחקן בכל פעם מטיל את הקובייה ויופיעו לו חצים אדומים על המפה מעל המשבצות אליהן הוא יכול ללכת לפי המספר שהטיל בקובייה. כדי ללכת למשבצת שהוא רוצה השחקן לוחץ על החץ שהוא רוצה ללכת אליו. במידה והשחקן לא מרוצה מהמספר שהוטל בקובייה , יש לו מעל הקובייה אפשרויות לגבי המספר שהטיל: הוא יוכל להוסיף לו צעד נוסף, להפחית ממנו צעד או להטיל פעם נוספת. השחקן לא יוכל להשתמש בשני שינויים באותו התור, ובכל משימה יש לו אפשרות לשינוי רק פעם אחת, כלומר לאחר שהשתמש בשינוי מסוים לא יוכל להשתמש בו פעם נוספת.

השחקן יכול במפה לדרוך על אויב, שק הפתעה ותיבת אוצר. הכוונה בלדרוך היא רק לאחר שביצע את כל מספר הצעדים שיצא בהטלת הקובייה.

<u>שק הפתעה</u> – כאשר השחקן דורך על שק הפתעה ישנן שתי אפשרויות:

1) השחקן ירוויח כמות מסוימת של כסף בין 15 ל30.

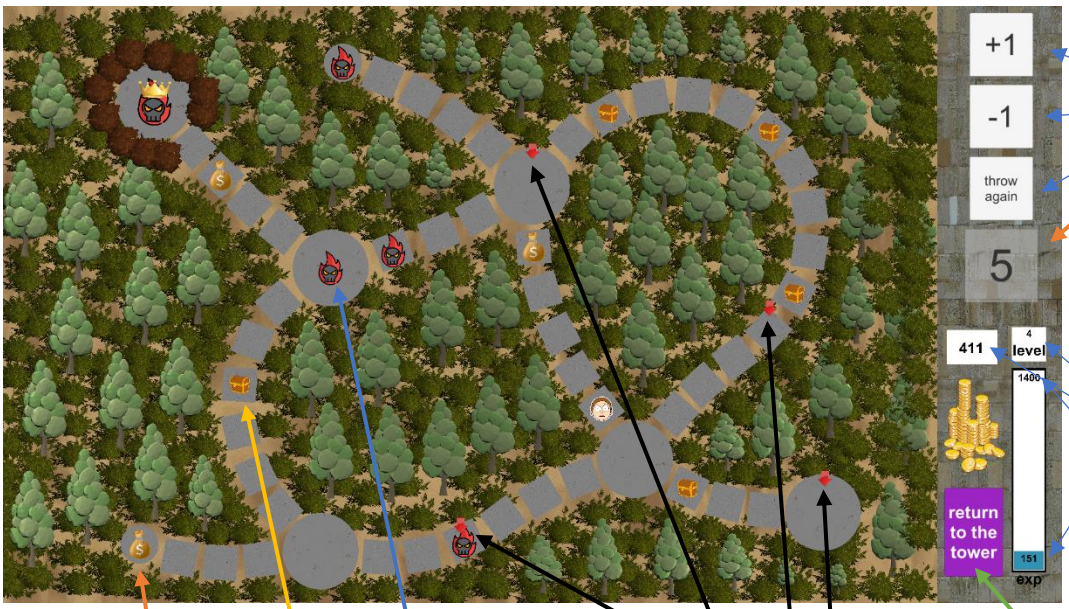
2) השחקן ישוגר למשבצת רנדומלית ריקה על המפה.

<u>תיבת אוצר</u> – כאשר דורך השחקן על תיבת אוצר הוא יקבל כמות מסוימת של כסף בין 15 ל30 וכמות מסוימת של ניסיון בין 15 ל30.

<u>אויב</u> – השחקן ילחם בדו קרב נגד האויב.

השחקן יכול לראות בצד ימין למטה את הנתונים שלו: כמה ניסיון שלו וכמה הוא צריך כדי לעבור רמה וכמה כסף יש לו.

בצד ימין למטה מופיע כפתור סגול שבלחיצה עליו יפתח חלון שבו נשאל השחקן האם הוא בטוח שהוא רוצה ללכת למגדל. אם ילחץ שכן הוא יפסיק את המשימה באמצע ולא ישלים אותה, אך עדיין ישמר לו הכסף והניסיון שהרוויח במהלך המשימה. אפ ילחץ שלא הוא יחזור לנקודה ממנה הפסיק.



הקובייה בכתום והבונוסים בכחול בהם השחקן יכול להשתמש

הנתונים של השחקן: כסף, ניסיון, רמה ולכמה ניסיון הוא צריך להגיע כדי לעלות רמה.

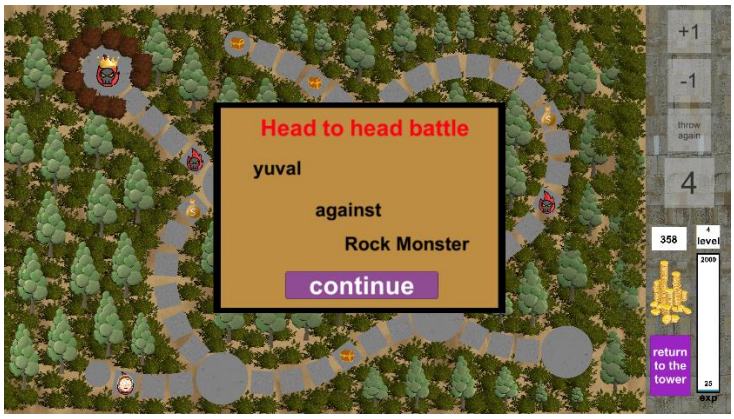
לשחקן יצא 5 בקובייה והחיצים אדומים מראים לו את האפשרויות אליהן הוא יכול ללכת.

אויב

תיבת אוצר

שק הפתעה

כפתור חזרה למגדל



לאחר שדרך על אויב או על שק הפתעה או על תיבת אוצר יופיע לשחקן חלון עם המידע הרלוונטי לגבי כל אחד מהם – תיבת אוצר: כמות הכסף והניסיון שקיבל. שק הפתעה: או שהשחקן משוגר למקום אחר או כמות הכסף שקיבל. אויב: יודיע לשחקן שהוא נלחם בדו קרב נגד אויב.

כאשר השחקן לוחץ על כפתור חזרה למגדל זה החלון שנפתח לו. לחיצה על yes תחזיר את השחקן למגדל ולחיצה על no תחזיר אותו לאיפה שהוא היה.

במפה יש 3 שקי הפתעה, 5 תיבות אוצר, 4 אויבים ואויב בוס. המפה הזו היא היחידה שיש במשחק.

<u>השלמת משימה</u> – כאשר מביס השחקן את כל האויבים הוא משלים את המשימה ומקבל 60 מטבעות (כסף) ו60 ניסיון.

<u>עליית שלב</u> – כאשר השחקן מגיע לכמות ניסיון שהוצבה לו להגיע אליה אז הניסיון מתאפס אך נשאר לו "העודף" משלב שעבר, הניסיון העודף מכמות המטרה, וכמות המטרה שלו גדלה. השחקן יקבל כמות כסף שהיא הכפלת השלב שאליו עלה ב10 והוספת 40 למכפלה.

דו קרב

אל הדו קרב מגע השחקן לאחר שנחת על אויב במפה. לאויב ולשחקן יש מספר זהה של קסם וחיים.

קסם – היחידות בעזרתן השחקן משתמש בכשפים במהלך הדו קרב. למשל כישוף מסוים עולה 20 קסם וכיסוף אחר עולה 40 קסם. כאשר אין לשחקן מספק קסם הוא לא יוכל להשתמש בכשפים שעולים יותר ממה שיש לו. כל כישוף עולה כמות שונה של קסם. הקסם גדל ב15

חיים – כאשר פוגע כישוף בשחקן או שאויב יורד לזה שפגע בו הכישוף חיים. הדו קרב נגמר ברגע שהשחקן או האויב מגיע ל0 חיים, וכך אחד מהם מפסיד. זה של הגיע ל0 חיים מנצח. כל כישוף מוריד כמות שונה של חיים לאחר.

הלוח בנוי מ12 משבצות – 3 שורות ו4 טורים.

* הדו קרב מתנהל כך שהשחקן בוחר שלוש מהלכים בכל תור מבין המהלכים שיש לו:

מהלכים בסיסיים שהשחקן מקבל בעת הרשמתו כשחקן חדש:

תנועה למעלה, למטה, שמאלה, ימינה.

חידוש קסם – השחקן מעלה לעצמו 20 קסם.

מגן – השחקן מגן מפני מתקפת היריב, אם הוא השתמש בכישוף במהלך הזה.

שני כשפים כדי להתקיף את היריב.

הכשפים השחקן קונה במגדל אלו כשפים נוספים שיעזרו לו כנגד האויב בדו קרב.

את המהלכים האלו גם היריב יכול לעשות בדו קרב, אך לו יש כשפים שונים. ליריב אין אפשרות לקנות כשפים חדשים, כלומר יש לו 2 כשפים קבועים.

*בעת בחירת המהלכים השחקן יכול לראות את המיקום שלו ואת המיקום של האויב במפה.

*כדי לבחור מהלך השחקן לוחץ על התמונה של הקלף.

* לאחר שהוא בוחר את שלושת המהלכים הוא לוחץ על המשך, אם מתחרט על המהלכים שבחר הוא יכול ללחוץ על כפתור אפס שבלחיצה עליו יבחר שלושה מהלכים חדשים.

* היריב כאמור יכול לעשות אותם מהלכים בסיסיים כמו השחקן ובוחר את המהלכים באופן מסוים.

* התורות בין מהלך למהלך יתנהלו בצורה כזאת: השחקן יעשה את תורו לפני היריב אלא אם השחקן עושה בתורו כשף והיריב לא. במצב הזה היריב יעשה את המהלך שלו קודם ואז השחקן. בכל מצב אחר השחקן יעשה את תורו לפני היריב.

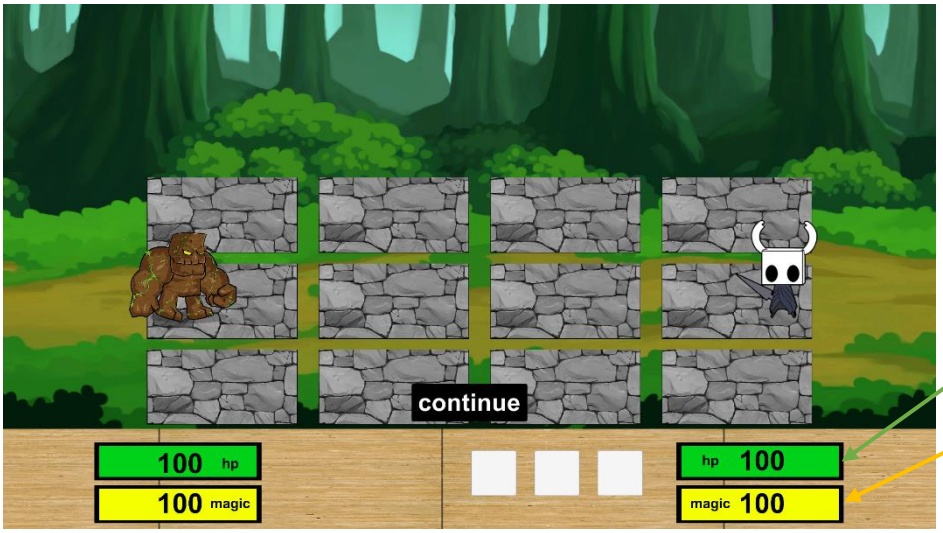
* אם לשחקן אין מספיק קסם, הוא לא יוכל להשתמש בכשפים העולים יותר קסם ממה שיש לו.

*בסוף כל תור (שלושה מהלכים) יתווסף לשחקן ולאויב כמות קסם שווה ל14+הרמה של השחקן.

* המשחק ימשיך כך בצורה הזאת עד שליריב או לשחקן יש 0 חיים. אם השחקן הפסיד האויב ישאר על המפה והשחקן לא יקבל דבר. אם השחקן ניצח האויב יעלם מהמפה ושחקן יקבל כמות מסוימת של כסף בין 25 ל40 וכמות מסוימת של ניסיון בין 40 ל60.

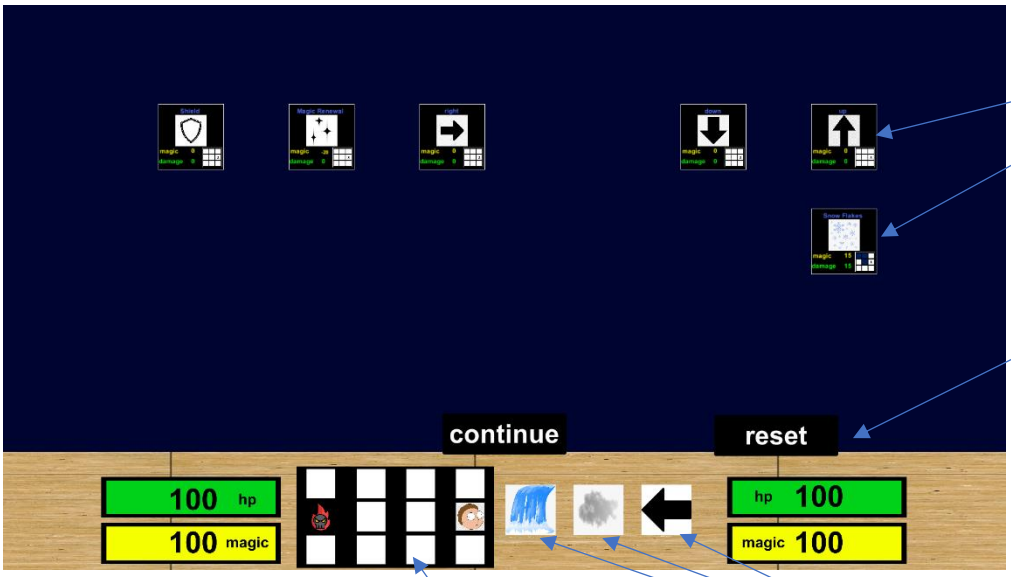
אויב בוס – לבוס יש יותר חיים וקסם מהשחקן והמתקפות שלו מורידות יותר חיים מאשר של אויב רגיל.

הדמות של האויב במשחק היא קבועה ויש רק אחת כזאת.



החיים של השחקן

הקסם של השחקן

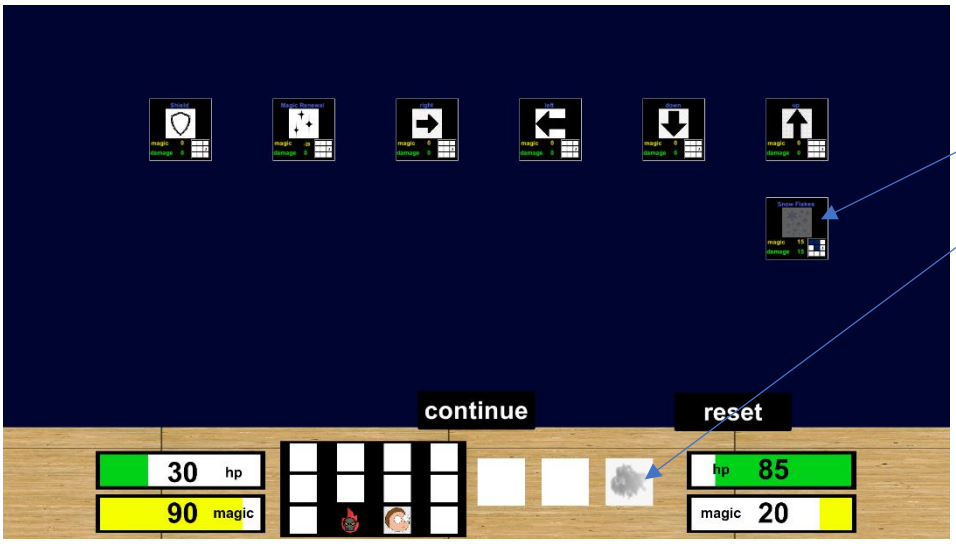


דוגמאות למהלכים של השחקן.

כפתור איפוס המהלכים

המיקומים של השחקן והאויב על המפה

שלושת המהלכים שבחר השחקן



לשחקן יש 20 קסם אך כיוון שהוא בחר להשתמש בכישוף במהלך הראשון אין לו מספיק קסם לכישוף השני ולכן לא יוכל להשתמש בו כרגע.

כפתור escape

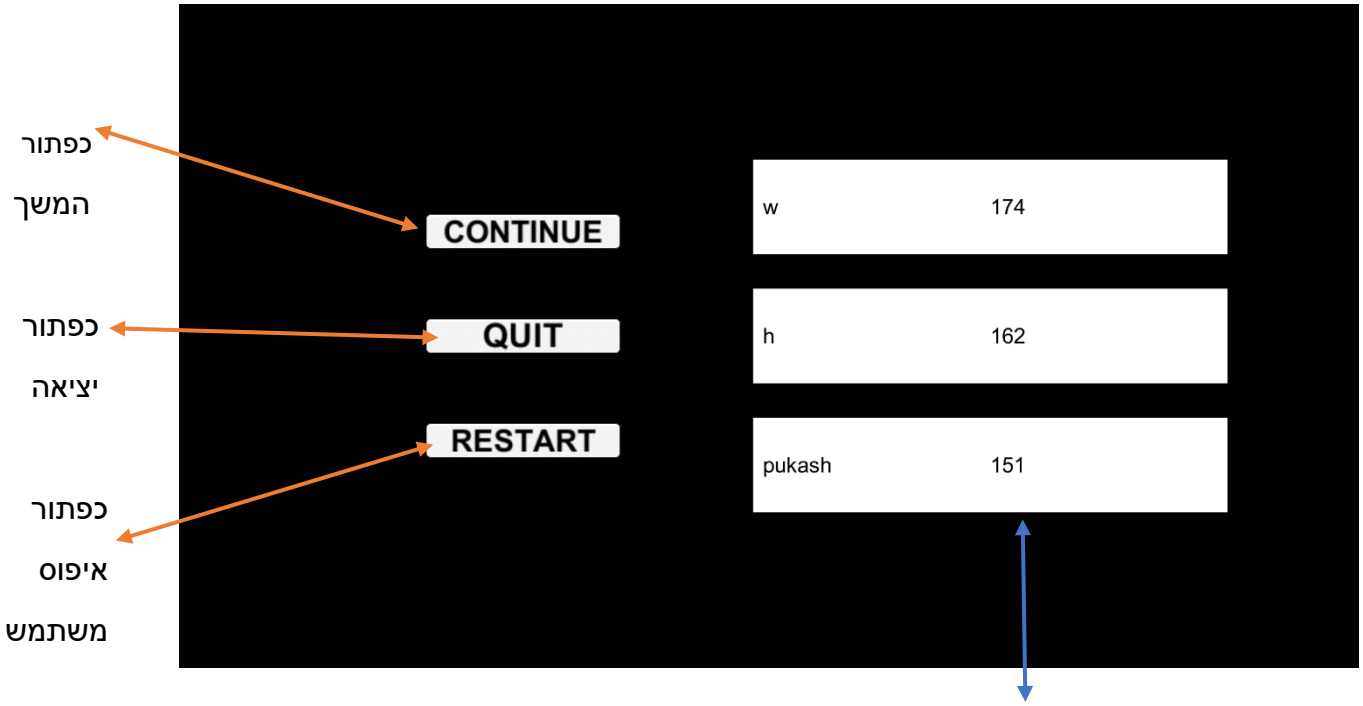
כאשר לוחץ השחקן כשהוא במפה או במגדל על כפתור escape במקלדת הוא יעבור למסך בו הוא יוכל לעשות מספר דברים:

כפתור המשך – בלחיצה עליו הוא יחזור למקום בו הוא עצר במשחק.

כפתור יציאה – בלחיצה עליו הוא יצא מהמשחק וההישגים שצבר ישמרו.

כפתור איפוס משתמש – בלחיצה עליו המשתמש מאפס את הנתונים שלו, כלומר הוא חוזר לרמה 1 כשברשותו 0 ניסיון ו0 כסף. זה לא כפתור מחיקת משתמש ולכן השם משתמש והסיסמא שלו נשארים כשהיו.

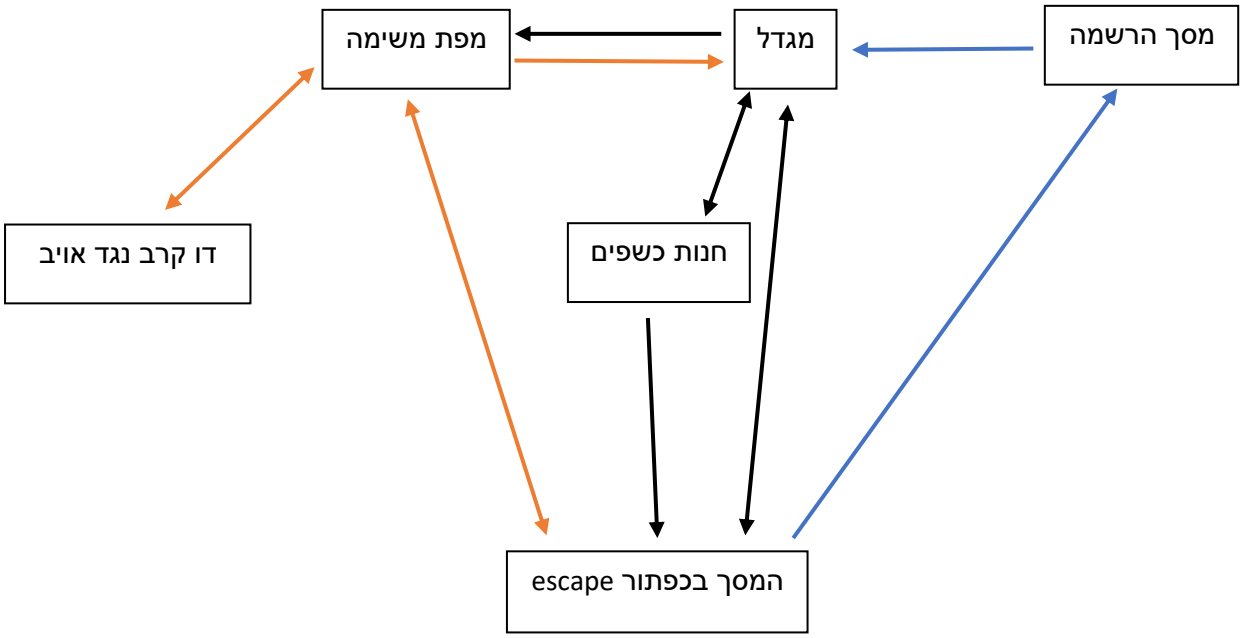
טבלת הניקוד הגבוה ביותר – הטבלה מציגה את שמותיהם ואת כמות הניסיון הכוללת שצברו של שלושת המשתמשים שצברו את כמות הניסיון הכוללת הגבוהה ביותר, כלומר כל הניסיון מכל הרמות הקודמות שהיו. למשל אם שחקן נמצא ברמה 5 וכרגע יש לו רק 20 ניסיון, אז הכוונה היא שיש לו גם את הניסיון שצבר בכל השלבים הקודמים ולא רק 20.



טבלת הניסיון הכולל הגבוה ביותר, ניתן לראות כי המשתמש "w" הוא בעל הניסיון הכולל הגבוה ביותר בסך 174, שני הוא המשתמש "h" שצבר 162 נקודות ניסיון ושלישי המשתמש "pukash" שצבר 151 נקודות ניסיון.

תרשים זרימה של המסכים במשחק

לאחר כל ההוראות אציג את הדרך בה ניתן להגיע לכל מסך ולאיזה מסכים כל מסך מוביל



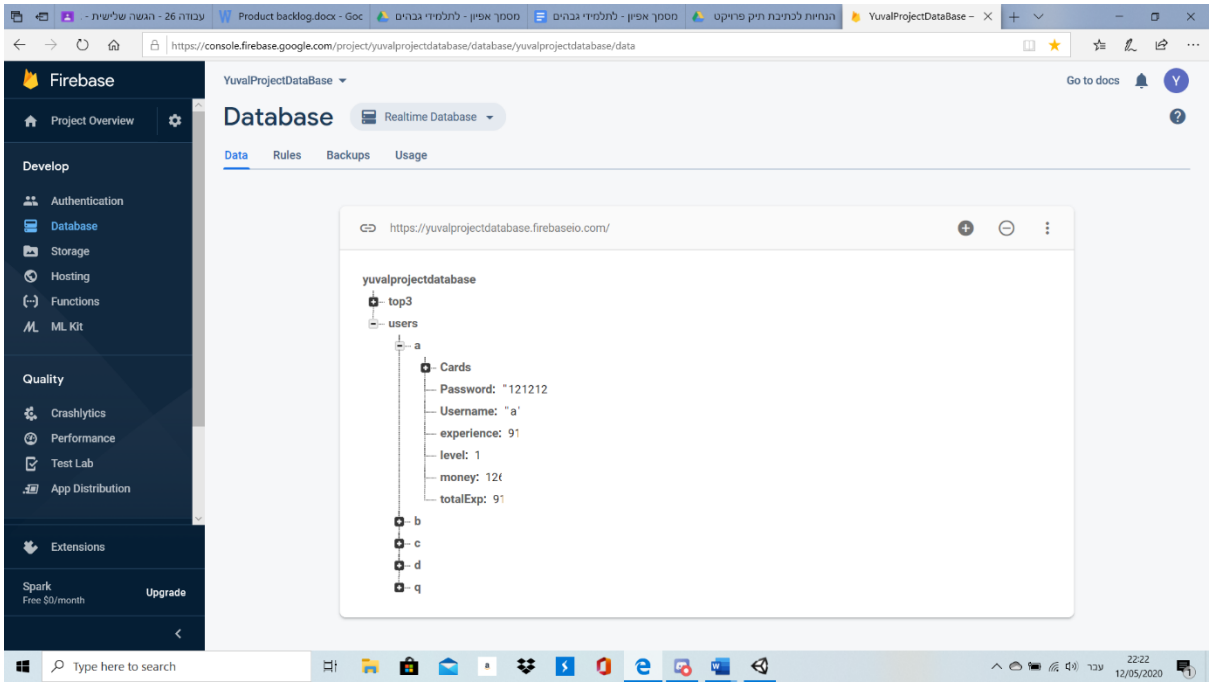
כל צבע מציג מצב שונה. אם אתה נכנס למסך escape כשאתה במפת המשימה, לא תוכל להגיע למגדל, ואם נכנס למסך escape כשאתה בחנות הקסמים לא תוכל להגיע למפת המשימה.

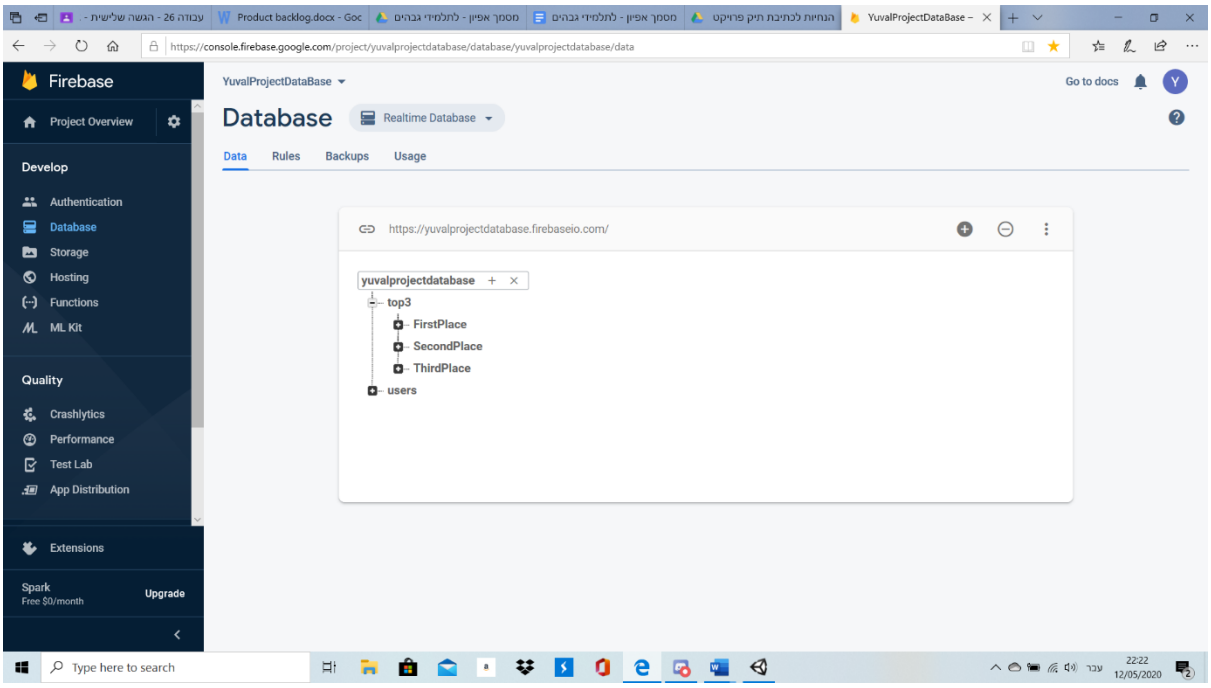
בסיס נתונים

אני השתמשתי בפרויקט בבסיס הנתונים firebase ואני שומר בו את הנתונים שצריך לשמור לגבי כל שחקן:

שם משתמש, סיסמא, כמה כסף יש לו, כמה ניסיון יש לו, באיזה שלב הוא ומערך בוליאני של מספרים המייצגים קלפי כסף שאם השחקן רכש אותם אז הם יהיו true, ואם לא אז false. השחקן מתחיל את המשחק כאשר לרשותו שתי מתקפות.

בנוסף אני שומר את שלושת השחקנים שיש להם את הניסיון הגבוה ביותר לפי סדר ואלו מוצגים בטבלה בעמוד 7.





מדריך למפתח

הפרויקט שלי כולל 15 קובצי קוד בשפה סי שארפ (c#) שכתבתי בסביבת העבודה visual studio שכל אחד מהקבצים דואג בנפרד או יחד עם קבצים אחרים למשהו מסוים בפרויקט:

אציג לגבי כל קובץ פונקציה אחת או כמה מרכזיות שמהוות חלק משמעותי מהקוד.

הרשמה

Register – קוד הבודק את הקלט של המשתמש שמזין שם משתמש, סיסמא וחזרה על הסיסמא. הקוד יכתוב את השחקן בdatabase רק אם השם לא תפוס על ידי משתמש אחר, אם הסיסמא גדולה מ6 תווים כדי שתהיה ביטחונית, ואם הסיסמא בשתי השדות של הסיסמא שוות כדי שלא יקרה מצב שבו המשתמש הקליד משהו אחר ממה שהוא חשב שהייתה הסיסמא. בעת ההרשמה השחקן מקבל נתונים התחלתיים – רמה 1, 0 כסף ו0 ניסיון, ואת הקלפים ההתחלתיים שהוצגו בהוראות תחת כותרת המשנה דו קרב.

Login – קוד הבודק קלט של שם משתמש וסיסמא של שחקן שכבר קיים במערכת וכשהוא מוצא אותו בdatabase הוא לוקח את הנתונים שלו לקוד סטטי אחר ששומר על הנתונים. הקוד כולל בנוסף פונקציה שמסדרת את טבלת הניסיון הגדול ביותר. אם אין עדיין שחקנים בכל התאים בטבלת הניסיון הגדול ביותר אז הקוד יכניס משתמשים ריקים לdatabase שיעודכנו כאשר יהיו שחקנים חדשים.

User – קוד היוצר אובייקט חדש מטיפוס User שהוא יוכנס לdatabase.

PlayerData – הקוד שמקשר בין המידע שבdatabase לבין שאר קבצי הקוד כשאר הם נדרשים לדעת את המידע הרלוונטי להם על השחקן. הקוד מכיל מערך של שמות הקלפים וגם פונקציה שבה נתוני השחקן מתקבלים לפי הרמה שלו: החיים שלו בקרב, הקסם שלו בקרב והמטרה הבאה שלו בכמות הקסם על מנת לעלות שלב.

```
private void PostToDataBase()
    {
        User user = new User();
        RestClient.Put(databaseURL + UsernameText + ".json", user);
        usernameRegister.GetComponent<InputField>().text = "";
        passwordRegister.GetComponent<InputField>().text = "";
        confPasswordRegister.GetComponent<InputField>().text = "";
        comments.GetComponent<Text>().text = "Register succeeded";
    }
```

פונקציית PostToDataBase בקובץ Register מטפלת בהכנסת המידע לתוך הdatabase באמצעות בניית אובייקט user שמכיל את המידע הרלוונטי למשתמש חדש.

```
public void OnSubmit()
    {
        namecheck = true;
        PasswordText = passwordRegister.GetComponent<InputField>().text;
        UsernameText = usernameRegister.GetComponent<InputField>().text;
        ConfPasswordText = confPasswordRegister.GetComponent<InputField>().text;
        level = 1;
        money = 0f;
        experience = 0f;
        totalExp = 0f;
        for(int i=0; i<Cards.Length; i++)
        {
            if (i <= 1 || i == 11 || i == 10)
                Cards[i] = true;
            else
                Cards[i] = false;
        }


        if (UsernameText == "")
        {
            comments.GetComponent<Text>().text = "input an username";
            return;
        }
        if (PasswordText == "")
        {
            comments.GetComponent<Text>().text = "input a password";
            return;
        }

        if (ConfPasswordText == "")
        {
            comments.GetComponent<Text>().text = "confirm your password";
            return;
        }

        if(PasswordText.Length < 6)
        {
            comments.GetComponent<Text>().text = "Use 6 characters or more for your
password";
            return;
        }

        if (UsernameText.Length > 10)
        {
            comments.GetComponent<Text>().text = "Use 10 characters or less for your
username";
            return;
        }

        if (PasswordText.Length > 10)
        {
            comments.GetComponent<Text>().text = "Use 10 characters or less for your
password";
            return;
        }


        RestClient.Get<User>(databaseURL + UsernameText + ".json").Then(response =>
        {
```

```
            if (response.Username == UsernameText)
            {
                namecheck = false;
                usernameRegister.GetComponent<InputField>().text = "";
                passwordRegister.GetComponent<InputField>().text = "";
                confPasswordRegister.GetComponent<InputField>().text = "";
                PasswordText = "";
                UsernameText = "";
                ConfPasswordText = "";
                namecheck = true;
                comments.GetComponent<Text>().text = "That username is taken. Try
another";
            }

        }).Catch(error =>
        {
            Debug.Log("good usesrname");
            if (namecheck)
            {
                if (string.Equals(PasswordText, ConfPasswordText))
                {
                    PostToDataBase();
                }
                else
                    comments.GetComponent<Text>().text = "Those passwords didn't
match. Try again.";
            }
        });
    }
```

פונקציית OnSubmit בקובץ Register בודקת שכל הערכים שהמשתמש הכניס חוקיים ונכונים. ההגדרה לחוקיים נמצאת בדף מעל תחת הכותרת Register.

```
public void OnSubmit()
    {
        Debug.Log("try to login");
        Username = usernameText.GetComponent<InputField>().text; ;
        Password = passwordText.GetComponent<InputField>().text; ;
        if (Password == null || Password == "")
        {
            comments.GetComponent<Text>().text = "input a password!";
            return;
        }
        if (Username == null || Username == "")
        {
            comments.GetComponent<Text>().text = "input a username!";
            return;
        }
        RestClient.Get<User>(databaseURL + Username + ".json").Then(response =>
        {
            Debug.Log("1" + response.Username + " 2 " + Username + " 3 " +
response.Password + " 4 " + Password);
            if (string.Equals(response.Username, Username) &&
string.Equals(response.Password, Password))
            {
                Debug.Log("work");
                //input data to the playerData
                PlayerData.startt(response.level, response.experience, response.money,
response.Cards, top3Names[0], top3Names[1], top3Names[2],
```

```
                top3TotalExp[0], top3TotalExp[1], top3TotalExp[2], Username);
            FirstPlace();
            SecondPlace();
            ThirdPlace();
            comments.GetComponent<Text>().text = "";
            Debug.Log(PlayerData.firstPlaceName);

            usernameText.GetComponent<InputField>().text = "";
            passwordText.GetComponent<InputField>().text = "";
            SceneManager.LoadScene("towerScene");
        }
        else
            comments.GetComponent<Text>().text = "Incorrect username or password";
    }).Catch(error =>
    {
        comments.GetComponent<Text>().text = "fail";
    });

}
```

פונקציית OnSubmit בקובץ Login בודקת שכל הערכים שהמשתמש הכניס חוקיים ונכונים. ההגדרה לחוקיים נמצאת בדף מעל תחת הכותרת Login. בתוך הפונקציה OnSubmit יש אתחול לאובייקט הסטטי playerData על ידי קריאה לפונקציית startt והכנסת הערכים הדרושים אליה. בנוסף יש קריאה לפונקציות שלוקחות את המידע מהdatabase על שלושת המקומות הראשונים בטבלת הניסיון.

```
private static void FirstPlace()
{
    RestClient.Get<User>(databaseURLTop3 + first).Then(response =>
    {
        top3Names[0] = response.Username;
        top3TotalExp[0] = response.totalExp;
        PlayerData.firstPlaceName = response.Username;
        PlayerData.firstPlaceTotalExp = response.totalExp;
    }).Catch(error =>{
        User user = new User();
        user.Username = "";
        user.Password = "";
        user.totalExp = 0f;
        top3Names[0] = "";
        top3TotalExp[0] = 0;
        RestClient.Put(databaseURLTop3 + first, user);});
}
```

הפונקציה שלוקחת את המידע לגבי המקום הראשון בטבלה, כך גם האחרות לגבי מקום שני ושלישי.
הפונקציות מופרדות כדי שלא יהיה עומס פניות ללקיחת מידע מהdatabase ואז המידע לא יקלט בקוד.

```
public static void startt(int _level,float expreiencee, float mony, bool[] _cards,
string name1, string name2, string name3, float exp1, float exp2, float exp3, string
name)
    {
        Level = _level;
        experience = expreiencee;
        money = mony;
        expGoal = 500;

        PlayerName = name;
```

```
        Hp = 100f + PermanentNum * (Level-1);
        Stamina = 100f+ PermanentNum * (Level-1);

        firstPlaceName = name1;
        secondPlaceName = name2;
        thirdPlaceName = name3;

        firstPlaceTotalExp = exp1;
        secondPlaceTotalExp = exp2;
        thirdPlaceTotalExp = exp3;

        NamesArray[0] = "SnowFlakesAttack";
        NamesArray[1] = "FogAttack";
        NamesArray[2] = "WaterFallAttack";
        NamesArray[3] = "HailAttack";
        NamesArray[4] = "IciclesAttack";
        NamesArray[5] = "IceBallAttack";
        NamesArray[6] = "IceFieldAttack";
        NamesArray[7] = "WhirlpoolAttack";
        NamesArray[8] = "AvalancheAttack";
        NamesArray[9] = "TsunamiAttack";
        NamesArray[10] = "MagicRenewal";
        NamesArray[11] = "Shield";
        //NamesArray[12] = "Heal";

        for(int i=0; i<_cards.Length; i++)
        {
                IsCardBought[i] = _cards[i];
        }

        for (int i = 0; i < SpellsLevelArray.Length; i++)
        {
                SpellsLevelArray[i] = 0;
        }

        for (int j = 1; j < Level; j++)
                expGoal += 300;
}
```

פונקציית startt שנמצאת באובייקט playerData שמקבלת את המידע מקוד login שמחובר לdatabase ומעדכנת את משתניה כך שקבצי קוד אחרים יוכלו להשתמש במידע זה.

מגדל

TowerManager – קוד השולח את השחקן למשימה כשהוא לוחץ על הדלת וכשהוא לוחץ על הקדרה הוא נכנס לחנות בא הוא יכול לקנות קלפים בהתאם לכסף שלו ולרמה בה הוא נמצא.

```
public void BuyCards()
    {
        foreach (Transform a in doorCanvas.transform)
        {
            a.gameObject.SetActive(false);
            if (a.gameObject.name == "return")
                a.gameObject.SetActive(true);
        }
        background.SetActive(false);
        CardsCanvas.SetActive(true);
        money.GetComponent<Text>().text = PlayerData.money + " coins";
        level.GetComponent<Text>().text = "level "+PlayerData.Level;
        for (int i=2; i<10; i++)
        {
            if(PlayerData.Level >= i+1)
            {
                if(PlayerData.IsCardBought[i] == true)
                {
                    foreach (Transform card in CardsCanvas.transform)
                    {
                        card.gameObject.SetActive(true);
                        if(card.gameObject.name.Contains((i+1).ToString()))
                        {
                            foreach (Transform component in card.transform)
                            {
                                if (component.gameObject.name == "Canvas")
                                {
                                    component.gameObject.SetActive(true);
                                    foreach (Transform CanvasComponent in
component.transform)
                                    {
                                        if (CanvasComponent.gameObject.name ==
"LevelRequiredText")
                                        {
                                            CanvasComponent.gameObject.SetActive(true);
                                            CanvasComponent.GetComponent<Text>().text
= "Sold";
                                        }
                                        if (CanvasComponent.gameObject.name ==
"BuyCard")
                                        {
                                            CanvasComponent.gameObject.SetActive(false);
                                        }
                                    }
                                }
                                if (component.gameObject.name == "LevelRequired")
                                {
                                    component.gameObject.SetActive(false);
                                }
                            }
```

```
                        }
                    }
                }
                else
                {
                    foreach (Transform card in CardsCanvas.transform)
                    {
                        card.gameObject.SetActive(true);
                        if (card.gameObject.name.Contains((i + 1).ToString()))
                        {
                            foreach (Transform component in card.transform)
                            {
                                if (component.gameObject.name == "Canvas")
                                {
                                    component.gameObject.SetActive(true);
                                    foreach (Transform CanvasComponent in
component.transform)
                                    {
                                        if (CanvasComponent.gameObject.name ==
"LevelRequiredText")
                                        {

CanvasComponent.gameObject.SetActive(false);
                                        }
                                        if (CanvasComponent.gameObject.name ==
"BuyCard")
                                        {

CanvasComponent.gameObject.SetActive(true);
                                        }
                                    }
                                }
                                if (component.gameObject.name == "LevelRequired")
                                {
                                    component.gameObject.SetActive(false);
                                }
                            }
                        }
                    }
                }
            }
            else
            {
                foreach (Transform card in CardsCanvas.transform)
                {
                    card.gameObject.SetActive(true);
                    if (card.gameObject.name.Contains((i + 1).ToString()))
                    {
                        foreach (Transform component in card.transform)
                        {
                            component.gameObject.SetActive(false);
                            if (component.gameObject.name == "Canvas")
                            {
                                component.gameObject.SetActive(true);
                                foreach (Transform CanvasComponent in
component.transform)
                                {
                                    CanvasComponent.gameObject.SetActive(false);
                                    if (CanvasComponent.gameObject.name ==
"LevelRequiredText")
                                    {
                                        CanvasComponent.gameObject.SetActive(true);
```

```
                                    }
                                }
                            }
                            if (component.gameObject.name == "LevelRequired")
                            {
                                component.gameObject.SetActive(true);
                            }
                        }
                    }
                }
            }
        }
    }
```

הקוד BuyCards הוא הקוד שממיין את הקלפים בחנות לפי הרמה של השחקן האם יש לו האפשרות לקנות אותם או שלא, האם הוא כבר רכש אותם ואז אין צורך שירכוש אתם שוב.

```
public void Pay(float payment)
    {
        CardName = (int)(payment/500+2);
        if(PlayerData.money >= payment)
        {
            PlayerData.money -= payment;
            PlayerData.IsCardBought[CardName - 1] = true;
            Login.UpdatePurchaseCardAndMoney(CardName - 1, PlayerData.money);
            BuyCards();
        }
        else
        {
            foreach (Transform card in CardsCanvas.transform)
            {
                card.gameObject.SetActive(true);
                if (card.gameObject.name.Contains(CardName.ToString()))
                {
                    foreach (Transform component in card.transform)
                    {
                        if (component.gameObject.name == "Canvas")
                        {
                            component.gameObject.SetActive(true);
                            foreach (Transform CanvasComponent in component.transform)
                            {
                                if (CanvasComponent.gameObject.name ==
"LevelRequiredText")
                                {
                                    CanvasComponent.gameObject.SetActive(true);
                                    CanvasComponent.GetComponent<Text>().text = "not
enough money";
                                }
                                if (CanvasComponent.gameObject.name == "BuyCard")
                                {
                                    CanvasComponent.gameObject.SetActive(false);
                                }
                            }
                        }
                        if (component.gameObject.name == "LevelRequired")
                        {
                            component.gameObject.SetActive(false);
                        }
                    }
                }
```

```
        }
    }
```

הקוד Pay הוא הקוד שבודק האם שחקן יכול לקנות קלף, כלומר האם יש לו כסף כדי לרכוש אותו. אם יש לו הוא יוסיף ויעדכן את הdatabase שהשחקן רכש את הקלף, אך אם אין לו מספיק כסף אז יכתב לו שעליו לצבור עוד כסף על מנת לקנות אותו.

```
public void GoToMission()
{SceneManager.LoadScene("mapScene");}
```

הפונקציה GoToMission היא הפונקציה שפועלת כשהשחקן לוחץ על הדלת והיא כוללת רק שורת קוד אחת שמעבירה את השחקן למסך מפת המשימה.

משימה

orderSquares – קוד שבו המפה מסודרת באופן "רנדומלי", דואגת לתנועת השחקן ולתנועת האויבים, שזזים צעד אחד בכל פעם לאחר שהשחקן זז, דואגת לקבלת המידע חזרה לאחר שהשחקן חוזר מדו קרב ולסדר את המפה לפי המידע ששמרה, כיוון שברגע שעוברים לסצנת הדו קרב המידע נמחק, אז לפני שהשחקן נכנס לדו קרב אז המידע נשמר בקוד סטטי. בנוסף הקוד מראה בכל פעם לאחר הטלת קובייה את החיצים שעליהם השחקן לוחץ כדי לזוז על המפה. הקוד גם אחראי לבונוסים בהטלת הקובייה שכוללים הוספת צעד מהמספר שיצא בקובייה, הפחתת צעד והטלה חדשה. כל המידע לגבי המפה נשמר בקוד WinOrLose.

```
private void Start()
    {
        if (winOrLose.isFirstDuelOver == false)
        {
            winOrLose.start();
            //PlayerData.startt();
            CurrentPlayerLocation = 0;

            situationss = new situationss(CurrentPlayerLocation, squaresArray,
windowCanvas, window, cube, bar, barCanvas, backToTowerButton);
            situationss.setBarAndData();

            int treasures = 5;
            int enemies = 4;
            int surpriseBag = 3;
            bool isEnemy = true;
            bool isTreasure = true;
            bool isSurpriseBag = true;
            int Ienemy = 0;
            int Itreasure = 0;
            int IsurpriseBag = 0;
            int sum = treasures + enemies + surpriseBag;
```

```
            plus1Cube.gameObject.GetComponent<Button>().interactable = false;
            minus1Cube.gameObject.GetComponent<Button>().interactable = false;
            ThrowAgain.gameObject.GetComponent<Button>().interactable = false;

            while (sum != 0)
            {
                for (int j = 1; j < 55; j++)
                {
                    resett(j);
                }
                Ienemy = 0;
                Itreasure = 0;
                IsurpriseBag = 0;
                treasures = 5;
                enemies = 4;
                surpriseBag = 3;
                isEnemy = true;
                isTreasure = true;
                isSurpriseBag = true;

                for (int i = 1; i < 55; i++)
                {
                    int num = (int)(Random.Range(0f, 13f));
                    GameObject block = pickSquare(i);
                    foreach (Transform Square in block.transform)
                    {
                        if (num == 0 && enemies > 0 && isEnemy == true && (Ienemy + 1
!= i) && (IsurpriseBag + 1 != i) && (Itreasure + 1 != i) &&
                            (Ienemy + 2 != i) && (IsurpriseBag + 2 != i) && (Itreasure
+ 2 != i))
                        {
                            if (Square.gameObject.name == "Enemy")
                            {
                                enemies--;
                                isEnemy = false;
                                Ienemy = i;
                                Square.gameObject.SetActive(true);
                            }
                        }
                        else if (num == 4 && treasures > 0 && isTreasure == true &&
(Itreasure + 1 != i) && (IsurpriseBag + 1 != i) &&
                            (Ienemy + 1 != i) && (Itreasure + 2 != i) && (IsurpriseBag
+ 2 != i) && (Ienemy + 2 != i))
                        {
                            if (Square.gameObject.name == "TreasureBox")
                            {
                                treasures--;
                                isTreasure = false;
                                Square.gameObject.SetActive(true);
                                Itreasure = i;
                            }
                        }
                        else if (num == 8 && surpriseBag > 0 && isSurpriseBag == true
&& (IsurpriseBag + 1 != i) && (Itreasure + 1 != i) && (Ienemy + 1 != i) &&
                            (IsurpriseBag + 2 != i) && (Itreasure + 2 != i) && (Ienemy
+ 2 != i))
                        {
                            if (Square.gameObject.name == "SurpriseBag")
                            {
                                surpriseBag--;
                                isSurpriseBag = false;
                                Square.gameObject.SetActive(true);
```

```
                                IsurpriseBag = i;
                            }
                        }
                    }
                    isEnemy = true;
                    isTreasure = true;
                    isSurpriseBag = true;
                }
                sum = treasures + enemies + surpriseBag;
                //Debug.Log("sum: " + sum + " enemies: " + enemies + " treasures: " +
treasures + " surprisebags: " + surpriseBag);
            }
        }
        else
        {
            situationss = new situationss(winOrLose.playerLocation, squaresArray,
windowCanvas, window, cube, bar, barCanvas, backToTowerButton);
            situationss.setBarAndData();

            ThrowAgain.gameObject.SetActive(winOrLose.throwAgain);
            plus1Cube.gameObject.SetActive(winOrLose.plus1);
            minus1Cube.gameObject.SetActive(winOrLose.minus1);

            GameObject b = pickSquare(55);
            foreach (Transform Square in b.transform)
            {
                if (Square.gameObject.activeSelf == true)
                    Square.gameObject.SetActive(false);
            }
            b = pickSquare(0);
            foreach (Transform Square in b.transform)
            {
                if (Square.gameObject.activeSelf == true)
                    Square.gameObject.SetActive(false);
            }
            for (int i=0; i<StringSquareArray.Length; i++)
            {
                b = pickSquare(i);
                foreach (Transform Square in b.transform)
                {
                    for(int f=0; f<winOrLose.treasureBoxesArray.Length; f++)
                    {
                        if(i == winOrLose.playerLocation)
                        {
                            if (Square.gameObject.name == "player")
                                Square.gameObject.SetActive(true);
                        }
                        if(i == winOrLose.treasureBoxesArray[f] && i != 0)
                        {
                            if (Square.gameObject.name == "TreasureBox")
                                Square.gameObject.SetActive(true);
                        }
                        if (i == winOrLose.enemiesArray[f] && i != 0)
                        {
                            if (Square.gameObject.name == "Enemy")
                                Square.gameObject.SetActive(true);
                        }
                        if(f < 3 && i == winOrLose.surpriseBagsArray[f] && i != 0)
                        {
                            if (Square.gameObject.name == "SurpriseBag")
                                Square.gameObject.SetActive(true);
                        }
```

```
                }
            }
        }
        if (winOrLose.win == true)
        {
            Debug.Log("player win player win");
            PlayerData.button = false;
            situationss.AfterWin();
        }

    }
}
```

הפונקציה start היא הפונקציה שמסדרת את המפה באופן שבו יהיו 5 תיבות אוצר 4 אויבים 3 שקי הפתעה ובוס, והם יהיו מופרדים אחד מן השני. הדבר השני שהיא מטפלת בו זה סידור המפה לאחר דו קרב, היא לוקחת את המידע מהקובץ winOrLose שעודכן לפני הדו קרב וממיינת את המפה לפי מידע זה.

```
public void MoveOptions(int situation)
    {
        counter = 0;
        SArray();

        for (int i = 0; i < StringSquareArray.Length; i++)
        {
            GameObject go = pickSquare(i);
            foreach (Transform x in go.transform)
            {
                if (x.gameObject.name == "player" && x.gameObject.activeSelf == true)
                {
                    CurrentPlayerLocation = i;
                }
            }
        }
        string[] TotalPossibleRoutes =
StringSquareArray[CurrentPlayerLocation].Split(',');
        //number = 5;

        if (situation == 1)
        {
            number++;
            plus1Cube.gameObject.SetActive(false);
            if(minus1Cube.gameObject.activeSelf == true)
                minus1Cube.gameObject.GetComponent<Button>().interactable = false;
            if (ThrowAgain.gameObject.activeSelf == true)
                ThrowAgain.gameObject.GetComponent<Button>().interactable = false;
        }
        if (situation == 2)
        {
            number--;
            minus1Cube.gameObject.SetActive(false);
            if (plus1Cube.gameObject.activeSelf == true)
                plus1Cube.gameObject.GetComponent<Button>().interactable = false;
            if (ThrowAgain.gameObject.activeSelf == true)
                ThrowAgain.gameObject.GetComponent<Button>().interactable = false;
        }
        if(situation == 3)
        {
            number = (int)(Random.Range(1f, 7f));
            ThrowAgain.gameObject.SetActive(false);
            if (plus1Cube.gameObject.activeSelf == true)
```

```
                plus1Cube.gameObject.GetComponent<Button>().interactable = false;
            if (minus1Cube.gameObject.activeSelf == true)
                minus1Cube.gameObject.GetComponent<Button>().interactable = false;
        }
        if(situation != 0)
        {
            foreach (Transform x in ArrowsArray.transform)
            {
                x.gameObject.SetActive(false);
            }
        }
        if (situation == 0 || situation == 3)
        {
            number = (int)(Random.Range(1f, 7f));
            cubeText.GetComponent<Text>().text = number.ToString();
        }
        if(situation == 0)
        {
            if (plus1Cube.gameObject.activeSelf == true)
                plus1Cube.gameObject.GetComponent<Button>().interactable = true;

            if (minus1Cube.gameObject.activeSelf == true)
                minus1Cube.gameObject.GetComponent<Button>().interactable = true;

            if(number == 1)
                minus1Cube.gameObject.GetComponent<Button>().interactable = false;

            if (ThrowAgain.gameObject.activeSelf == true)
                ThrowAgain.gameObject.GetComponent<Button>().interactable = true;
        }

        cube.gameObject.GetComponent<Button>().interactable = false;

        for (int y = 0; y < constant; y++)
        {
            for (int e = 0; e < constant; e++)
            {
                possibleRouts[y, e] = "";
            }
        }

        for (int i = 0; i < TotalPossibleRoutes.Length; i++)
        {
            if (i != 0)
                counter++;
            possibleRouts[counter, 0] = TotalPossibleRoutes[i];
            Debug.Log("counter: " + counter);
            TheOptions(int.Parse(TotalPossibleRoutes[i]), CurrentPlayerLocation);
        }
    }
```

הקוד moveOptions אחראי על הטלת הקובייה ועל הבונוסים של הקובייה ובנוסף היא מתחילה את ניהול אפשרויות חצי התנועה שממשיך בפונקציה theOptions.

```
private void TheOptions(int NextLocation, int _last)
    {
        int cpl = NextLocation;
        int lastLocation = _last;
        bool SkipCell = false;
        routesNumber = 1;
```

```
for(int j=0; j<routesNumber; j++)
{
    if (j > 0)
    {
        possibleRouts[counter,0] = possibleRouts[counter - 1,0];
    }
    for (int i = 1; i < number; i++)
    {
        string[] a = StringSquareArray[cpl].Split(',');
        if (a.Length == 1)
        {
            lastLocation = cpl;
            cpl = int.Parse(a[0]);
            possibleRouts[counter, i] = cpl.ToString();
        }
        if (a.Length == 2)
        {
            if (a[0] == lastLocation.ToString())
            {
                lastLocation = cpl;
                cpl = int.Parse(a[1]);
                possibleRouts[counter, i] = cpl.ToString();
            }
            else
            {
                lastLocation = cpl;
                cpl = int.Parse(a[0]);
                possibleRouts[counter, i] = cpl.ToString();
            }

        }
        if (a.Length > 2)
        {
            routesNumber = a.Length - 1;
            if (a[j] == lastLocation.ToString())
            {
                SkipCell = true;
                lastLocation = cpl;
                cpl = int.Parse(a[j + 1]);
                possibleRouts[counter, i] = cpl.ToString();
            }
            else
            {
                if (SkipCell == true)
                {
                    lastLocation = cpl;
                    cpl = int.Parse(a[j + 1]);
                    possibleRouts[counter, i] = cpl.ToString();
                }
                else
                {
                    lastLocation = cpl;
                    cpl = int.Parse(a[j]);
                    possibleRouts[counter, i] = cpl.ToString();
                }
            }
        }
    }
    cpl = NextLocation;
    lastLocation =_last ;
    if(j+1 < routesNumber)
        counter++;
```

```
        }
        for(int u=0; u< constant; u++)
        {
            for(int y=0; y< constant; y++)
            {
                if(possibleRouts[u,y] == "")
                {
                    if (y == 0)
                        y = constant;
                    else
                    {
                        ActiveArrow(possibleRouts[u, y - 1]);
                    }
                }
                else
                {
                    if (y == constant-1)
                    {
                        ActiveArrow(possibleRouts[u, y]);
                    }
                }

            }
        }
        for (int u = 0; u < constant; u++)
        {
            for (int y = 0; y < constant; y++)
            {
                Debug.Log("row: " + u + " coulmn: " + y + " value: " +
possibleRouts[u, y]);

            }
        }
    }
```

הפוקנציה theOptions אחראית על ניהול החצים שעליהם לוחץ השחקן כדי לזוז, היא עושה זאת על ידי תזוזה תיאורית לשכניה באורך המספר שהוטל בקובייה.

Situations – אובייקט שמנהל את מה שקורה לאחר שהשחקן מגיע למשבצת החדשה לאחר הטלת הקובייה. אם הוא נוחת על פיצ'ר מסוים אז נפתח חלון שמודיע לשחקן את המידע שמקבל, לדוגמא עם הוא נוחת על תיבת אוצר אז בחלון יהיה כתוב כמה כסף וניסיון הוא קיבל. האובייקט אחראי גם על בדיקת המידע של השחקן לגבי המשימה – אם סיים אותה הוא מודיע לו וכמו בדוגמה עם תיבת אוצר, מודיע לו כמה כסף וניסיון הוא קיבל. כך גם לגבי עליית רמה.

```
private void enableWindow(int num)
    {
        situation = num;
        Debug.Log("situation: " + situation);
        window.gameObject.SetActive(true);
        foreach (Transform s in windowCanvas.transform)
        {
            if (situation == 0 && s.gameObject.name == "treasureBoxText")
            {
                exp = (int)(Random.Range(15f, 31f));
                PlayerData.experience += (float)exp;

                money = (int)(Random.Range(15f, 31f));
                PlayerData.money += (float)money;
                Login.UpdateExpAndMoney(money, exp);
                Debug.Log("money: " + money);
```

```
                s.gameObject.SetActive(true);
                foreach (Transform a in s.gameObject.transform)
                {
                    if (a.gameObject.name == "expAmount")
                    {
                        a.GetComponent<Text>().text = exp.ToString();
                    }
                    if (a.gameObject.name == "moneyAmount")
                    {
                        a.GetComponent<Text>().text = money.ToString();
                    }
                    if (a.gameObject.name == "youGet")
                    {
                        a.GetComponent<Text>().text = PlayerData.PlayerName.ToString()
+ " get:";
                    }
                }
            }
            else if (situation == 1 && s.gameObject.name == "surpriseBagMoneyText")
            {
                money = (int)(Random.Range(15f, 31f));
                PlayerData.money += (float)money;
                Login.UpdateMoney(PlayerData.money);

                s.gameObject.SetActive(true);
                foreach (Transform a in s.gameObject.transform)
                {
                    if (a.gameObject.name == "moneyAmount")
                    {
                        a.GetComponent<Text>().text = money.ToString();
                    }
                    if (a.gameObject.name == "youGet")
                    {
                        a.GetComponent<Text>().text = PlayerData.PlayerName.ToString()
+ " get:";
                    }
                }
            }
            else if (situation == 2 && s.gameObject.name == "surpriseBagMovementText")
            {
                s.gameObject.SetActive(true);
            }
            else if (situation == 3 && s.gameObject.name == "enemyText")
            {
                foreach (Transform a in s.gameObject.transform)
                {
                    if (a.gameObject.name == "name")
                    {
                        a.GetComponent<Text>().text =
PlayerData.PlayerName.ToString();
                    }
                }
                s.gameObject.SetActive(true);
            }
        }
    }
```

הפונקציה enableWindow היא פונקציה שפותחת חלון על המידע הרלוונטי לגבי הפיצ'ר עליו דרך השחקן.

```
public void AfterContinueButton()
    {
        foreach (Transform s in windowCanvas.transform)
        {
            if(s.gameObject.name != "Button")
                s.gameObject.SetActive(false);
        }
        window.gameObject.SetActive(false);
        deleteCurrentSquarsFiture();
        if (situation == 0 || situation == 1)
            updateExpAndMoney();
        if (situation == 2)
            SurpriseBagTransform();
        if (situation == 3)
            Duel();
        if(situation == 5)
        {
            levelUp();
        }
        if(situation == 7)
        {
            missionAccomplished();
        }

        if(situation == 9)
        {
            if (PlayerData.experience < PlayerData.expGoal)
            {
                bar.transform.localScale = new Vector3(1f, -(PlayerData.experience /
PlayerData.expGoal), 1f);
                cube.gameObject.GetComponent<Button>().interactable = true;
                backToTowerButton.gameObject.SetActive(true);
            }
            else
            {
                levelUp();
            }
        }
    }
```

הפונקציה נקראת לאחר הלחיצה על כפתור המשך בחלון שנוצר בפונקציה enableWindow, והיא ממיינת את את הקיראה לפונקציה שאחראית לכל פיצ'ר, בהתאם לזה שדרך עליו. למשל אם השחקן דרך על תיבת אוצר אז בפונקציה enableWindow יפתח חלון עם המידע של כמה ניסיון וכסף הוא קיבל, ואז לאחר לחיצה על הכפתור המשך הפונקציה AfterContinueButton תקרא לפונקציה updateMoneyAndExp שתעדכן את המידע על השחקן בdatabase.

```
private void updateExpAndMoney()
    {
        foreach (Transform a in barCanvas.transform)
        {
            if (a.gameObject.name == "currentExp")
            {
                a.GetComponent<Text>().text = PlayerData.experience.ToString();
            }
            if (a.gameObject.name == "money")
            {
                a.GetComponent<Text>().text = PlayerData.money.ToString();
            }
        }
        if (PlayerData.experience < PlayerData.expGoal)
        {
```

```
            bar.transform.localScale = new Vector3(1f, -(PlayerData.experience /
PlayerData.expGoal), 1f);
            cube.gameObject.GetComponent<Button>().interactable = true;
        }
        else
        {
            levelUp();
        }
```

WinOrLose – class סטטי שמנהל את המידע שבמפה כשהשחקן יוצא וחוזר מדו קרבים, ומסדר את המפה כמו שהייתה כשעבר לסצנת הדו קרב. שומר את המידע לגבי המפה, הבונוסים של הקובייה והמיקומים של כל הפיצ'רים – תיבות אוצר, אויבים ושקי הפתעה

```
public static void start()
    {
        for (int i = 0; i < enemiesArray.Length; i++)
        {
            enemiesArray[i] = 0;
            treasureBoxesArray[i] = 0;
        }
        for (int i = 0; i < surpriseBagsArray.Length; i++)
        {
            surpriseBagsArray[i] = 0;
        }
        plus1 = true;
        minus1 = true;
        throwAgain = true;
```

הפונקציה זוכרת את המצב ההתחלתי בו מיקום השחקן הוא 0 ועדיין לא השתמשו כלל בבונוסים של הקובייה. כשהשחקן דורך על פיצ'ר המידע לגבי מיקומי התיבות אוצר, שקי הפתעה, אויבים מיקום השחקן ושימושים בבונוסים של הקובייה נשמר בקוד orderSquares.

דו קרב

StaminaBar – קוד אובייקט שמסדר את בר החיים והקסם מתוך הכמות ההתחלתית. הקוד משומש גם במפה בבר הניסיון. בכל פעם שהשחקן משתמש בכשף הקוד מחלק את הבר ביחס שבין הכמות שהייתה קודם פחות הקסם שהכישוף עלה לבין כמות הקסם ההתחלתית שיש לשחקן. כנ"ל גם לגבי החיים והפגיעה בשחקן.

```
public void TakeStamina(float stamina, float totalStamina)
    {
        if (currentStamina - stamina > totalStamina)
        {
            this.currentStamina = totalStamina;
        }
        else
        {
            this.currentStamina = this.currentStamina - stamina;
        }

        if (this.currentStamina <= 0)
```

```
        {
            transform.localScale = new Vector3(0f, 1f, 1f);
            this.currentStamina = 0;
        }
        else
            transform.localScale = new Vector3((currentStamina / totalStamina), 1f,
1f);
        StaminaTextbox.GetComponent<Text>().text = currentStamina.ToString();
                                                                                }
```

הפונקציה שמעדכנת את הבר בהתאם לנתונים העדכניים (חיים, קסם, ניסיון)

Attacks – משנה את צבע המשבצות לצבע השחקן כשהוא מתקיף, משנה את צבע המשבצת כשהשחקן ממלא קסם או מפעיל מגן ומזיז את השחקנים אם בחרו בקלף תזוזה. שולחת משתנה בוליאני האם השחקן פגע באויב כשהשתמש בכישוף או לא ולהפך לגבי פגיעת האויב בשחקן.

```
private void ListOfBlocks(string posisions, bool playe)
    {
        string[] a = posisions.Split(',');
        isEnemyHited = false;
        isPlayerHited = false;
        foreach (Transform SquareFloor in BlocksArray.transform)
        {
            for (int i = 0; i < a.Length; i++)
            {
                if (SquareFloor.gameObject.name == a[i])
                {
                    if (playe == true)//player attack
                    {
                        SquareFloor.gameObject.GetComponent<SpriteRenderer>().color =
new Color(0, 0, 40);
                        if (a[i] == "Floor" + EnemyCrntRow.ToString() +
EnemyCrntClmns.ToString())
                        {
                            isEnemyHited = true;
                        }
                    }
                    else//enemy attack
                    {
                        SquareFloor.gameObject.GetComponent<SpriteRenderer>().color =
new Color(0.5490196f, 0.2745098f, 0.07843138f, 1f);
                        if (a[i] == "Floor" + CrntRow.ToString() +
CrntClmns.ToString())
                        {
                            isPlayerHited = true;
                        }
                    }
                }
            }
        }
```

הפונקציה "צובעת" את משבצות הרצפה בצבע השחקן כשהוא משתמש בכישוף, בנוסף בודקת האם האויב פגע בשחקן כשעשה כישוף והאם השחקן פגע באויב כשעשה כישוף.

```
public void up()
    {
        if (CrntRow != 0)
        {
            player.transform.Translate(0, Yunits, 0);
            Debug.Log("row: " + CrntRow + " clmns: " + CrntClmns);
            PlayerSquareMove(1);
            Debug.Log("up");
        }
        else
        {
            Debug.Log("cant move up " + CrntRow);
        }
        isEnemyHited = false;
```

דוגמה לפונקציית תנועה שבה השחקן זז מספר יחידות קבוע שנמדד מראש לכיוון מעלה.

```
public void SnowFlakesAttack()
    {
        //in this function will be the damage and the stamina of the player when he
uses SnowFlakes attack
        if (direction)
        {
            string posisions = "Floor" + CrntRow.ToString() + (CrntClmns -
1).ToString() + ",";
            posisions += "Floor" + (CrntRow - 1).ToString() + (CrntClmns -
1).ToString() + ",";
            posisions += "Floor" + (CrntRow - 1).ToString() + (CrntClmns -
2).ToString() + ",";
            ListOfBlocks(posisions, true);
        }
        else
        {
            string posisions = "Floor" + CrntRow.ToString() + (CrntClmns +
1).ToString() + ",";
            posisions += "Floor" + (CrntRow - 1).ToString() + (CrntClmns +
1).ToString() + ",";
            posisions += "Floor" + (CrntRow - 1).ToString() + (CrntClmns +
2).ToString() + ",";
            ListOfBlocks(posisions, true);
        }
    }
```

דוגמה לפונקציית כישוף שבה בהתאם למשבצות שהמהלך יפגע בהם שנקבעו מראש יתווספו לרצף של string שמופרד באמצעות "," ולאחר מכן ישלחו לפונקציה ListOfBlocks.

EnemyManger – קוד שמחליט על שלושת המהלכים של האויב לפי סיכויים שנקבעו מראש ולפי תנאים המונעים ממנו לבצע מהלכים כביכול מטופשים או לא יעילים.

```
public void pickAttacks(int rowPlayer, int coulmnPlayer, int _rowEnemy, int
_coulmnEnemy)
       {
              rowEnemy = _rowEnemy;
              coulmnEnemy = _coulmnEnemy;
              int counter = 0;
```

```
            bool continu = false;


            while(continu == false)
            {
                    bool staminafull = false;
                    bool RemoteAttackOrToMuchStamina = false;
                    bool doubleMovement = false;


                    int num = (int)(Random.Range(0f, 501f));

                    if (num <= 40)
                            number = 0;
                    else if (num <= 80 && num > 40)
                            number = 1;
                    else if (num <= 120 && num > 80)
                            number = 2;
                    else if (num <= 160 && num > 120)
                            number = 3;
                    else if (num <= 260 && num > 160)
                            number = 4;
                    else if (num <= 340 && num > 260)
                            number = 5;
                    else if (num <= 420 && num > 340)
                            number = 6;
                    else if (num <= 500 && num > 420)
                            number = 7;


                    Debug.Log("num: " + number);
                    moves[counter] = movesArray[number];

                    if (this.TotalStamina == this.Stamina && moves[counter] ==
attacks[2].getNameSpell())//if the stamina full and the card is magic renewal
                    {
                            staminafull = true;
                    }

                    if((counter >= 1 && (moves[1] == moves[0] || moves[1] == moves[2]
|| moves[0] == moves[2]) && moves[counter].Contains("Attack") == false))
                    {
                            //if there is movement to any direction twice
                            doubleMovement = true;
                    }

                    if(moves[counter].Contains("Attack") == true)
                    {
                            //if the attack's stamina required is more than the
current stamina or the enemy is too far from the player
                            SpellClass attack = whichAtttack(moves[counter]);
                            if((attack.getStaminaSpell() > this.TotalStamina ||
(coulmnPlayer - coulmnEnemy == 3) || (coulmnPlayer - coulmnEnemy == -3) || (rowPlayer
- rowEnemy == 2) || (rowPlayer - rowEnemy == -2)) && turn == 0)
                                    RemoteAttackOrToMuchStamina = true;
                    }

                    if (doubleMovement == false && RemoteAttackOrToMuchStamina ==
false && staminafull == false)
                    {
                            for (int i = 0; i < attacks.Length; i++)
                            {
```

```
                                    if (moves[counter] == attacks[i].getNameSpell())
                                    {
                                            this.TotalStamina -=
attacks[i].getStaminaSpell();
                                            staminaMoves[counter] =
attacks[i].getStaminaSpell();
                                            damageMoves[counter] =
attacks[i].getDamage();
                                            Debug.Log(this.TotalStamina + " " +
attacks[i].getNameSpell() +" "+ attacks[i].getStaminaSpell());
                                    }
                            }
                            counter++;
                            if (counter > 2)
                                    continu = true;
                    }
                    movement();
            }
            if ((!moves[0].Contains("Attack")) && (!moves[1].Contains("Attack")) &&
(!moves[2].Contains("Attack")) && turn > 0)
            {
                    if (this.TotalStamina >= attacks[0].getStaminaSpell() &&
this.TotalStamina >= attacks[1].getStaminaSpell())
                    {
                            Debug.Log("no attack");
                            int numMoves = (int)(Random.Range(0f ,3f));
                            int atack = (int)(Random.Range(0f, 2f));
                            Debug.Log(atack + " pppppppppp " + numMoves);
                            moves[numMoves] = attacks[atack].getNameSpell();

                            this.TotalStamina -= attacks[atack].getStaminaSpell();
                            staminaMoves[numMoves] = attacks[atack].getStaminaSpell();
                            damageMoves[numMoves] = attacks[atack].getDamage();
                    }
            }



            this.TotalStamina += 15;
            if (this.TotalStamina > this.Stamina)
                    this.TotalStamina = this.Stamina;
            for (int i = 0; i < moves.Length; i++)
            {
                    Debug.Log("000099999999 " + moves[i]);
                    Debug.Log("Llllllllllll " + staminaMoves[i] + " " +i);
            }

            Debug.Log(this.TotalStamina + " jjjjjjjj");
            turn++;
    }
```

הפונקציה pickAttacks בוחרת את מהלכי היריב לפי תנאים המונעים ממנה לעשות דברים לא חוקיים כגון שימוש בקסם שדורש יותר קסם ממנה שיש ליריב ברגע זה, או דברים מטופשים כמו שימוש במתקפות כשהיריב כלל לא קרוב אליו.

PlayerClass – אובייקט המנהל את הקסם והחיים של השחקן במהלך הדו קרב, ובונה אובייקטים של קלפים.

```
public PlayerClass(int level, float hp, float stamina)
	{
		this.Level = level;
		this.Hp = hp;
		this.TotalStamina = stamina;
		this.Stamina = stamina;
		this.TotalHp = hp;
		this.StaminaBonus = 14 + Level;


		NamesArray[0] = "SnowFlakesAttack";
		NamesArray[1] = "FogAttack";
		NamesArray[2] = "WaterFallAttack";
		NamesArray[3] = "HailAttack";
		NamesArray[4] = "IciclesAttack";
		NamesArray[5] = "IceBallAttack";
		NamesArray[6] = "IceFieldAttack";
		NamesArray[7] = "WhirlpoolAttack";
		NamesArray[8] = "AvalancheAttack";
		NamesArray[9] = "TsunamiAttack";
		NamesArray[10] = "MagicRenewal";
		NamesArray[11] = "Shield";
		NamesArray[12] = "Heal";

		for (int i = 0; i<spellsArray.Length; i++)
		{
			if (i <= 9)
			{
				spellsArray[i] = new SpellClass(NamesArray[i], (num + i *
5), (num + i * 5));
			}
			if(i == 10)
				spellsArray[i] = new SpellClass(NamesArray[i], -20f, 0f);


			if (i == 12)
				spellsArray[i] = new SpellClass(NamesArray[i], 20f, 20f);


			if (i == 11)
				spellsArray[i] = new SpellClass(NamesArray[i], 0f, 0f);

		}
	}
```

הפונקציה PlayerClass היא פונקציית הבנאי של האובייקט והיא משתמשת במשתנים שמכילים את הקסם והחיים של השחקן, ובונה אובייקטים של הקלפים של השחקן, חלק מהקלפים בנפרד כי יש להם נתונים שונים כמו קלף חידוש הקסם או קלף המגן.

spellsClass – אובייקט של קלף קסם שבו יש את הנתונים כמה חיים הקלף מוריד לאויב, כמה קסם הוא "עולה" לשחקן ומה השם שלו.

```
public class SpellClass
{
	private string name;
	private float stamina;
	private float damage;
	private int stars;
	public SpellClass(string name, float stamina, float damage)
	{
		this.name = name;
		this.stamina = stamina;
		this.damage = damage;
	}

	public string getNameSpell()
	{
		return this.name;
	}

	public float getStaminaSpell()
	{
		return this.stamina;
	}

	public float getDamage()
	{
		return this.damage;
	}
}
```

ScreenManager – קוד אובייקט המנהל את המסך כך שבכל פעם שהשחקן לוחץ המשך הוא "מעלים" ו"מראה" אובייקטים בunity בהתאם לסיטואציה. למשל לאחר שהשחקן בחר שלושה מהלכים אז המסך יחזור מבחירת הקלפים למסך המלחמה, וכשהאויב והשחקן יסיימו את שלושת המהלכים שלהם, אז הכפתור יחזור למסך בחירת הקלפים. בנוסף אחראי על הופעת הקלפים אותם השחקן רכש בלבד.

```
public void isScreen()
    {
        screen.SetActive(!screen.activeSelf);
        player.SetActive(!player.activeSelf);
        enemy.SetActive(!enemy.activeSelf);
        BlocksArray.SetActive(!BlocksArray.activeSelf);
        courtPositionsArray.SetActive(!courtPositionsArray.activeSelf);
        foreach (Transform ChosenButton in continueButton.transform)
        {
            if (ChosenButton.gameObject.name == "Disappear")
            {

ChosenButton.gameObject.SetActive(!ChosenButton.gameObject.activeSelf);
                SortCards();
                a = ChosenButton.gameObject.activeSelf;
            }
        }
```

```
        foreach (Transform ChosenButton in continueButton.transform)
        {
            if (ChosenButton.gameObject.name == "ContinueButton")
                ChosenButton.gameObject.SetActive(a);
        }
    }
```

הקוד האחראי על ניהול המסך בהתאם לבחירת הקלפים ובעת הדו קרב.

```
private void SortCards()
    {
        foreach (Transform ChosenButton in continueButton.transform)
        {
            if (ChosenButton.gameObject.name == "Disappear")
            {
                foreach (Transform x in ChosenButton.gameObject.transform)
                {
                    for(int i=0; i<PlayerData.IsCardBought.Length; i++)
                    {
                        if(x.gameObject.name == PlayerData.GetCardName(i))
                        {
                            x.gameObject.SetActive(PlayerData.IsCardBought[i]);
                        }
                    }
                }
            }
        }
    }
```

הפונקציה האחראית על הופעת הקלפים שאותם השחקן רכש בלבד.

PlayerMovment – הקוד המרכזי בדו קרב, שמנהל את המלחמה בזמן אמת ובודק האם השחקן פגע באויב ולהפך, את התנועות שלהם, את המיקומים שלהם, האם הם מתחלפים בכיוון וכדומה.

הפונקציה המרכזית היא פונקציה שנקראת IEnumerator שהיא פונקציה שבאפשרותה לבצע פסקי זמן תוך כדי. אני משתמש בפסקי הזמן האלה בין תור של השחקן לתור של היריב ולהפך, ול"צביעת" משבצות המגרש בהתאם להתקפות והחזרתן לצבען הרגיל.

```
public void CellMove(string MoveCard)
    {
        if (isFull == false)
        {
            Debug.Log("playnum: " + PlayNum);
            foreach (Transform ChosenCard in ButtonsCanvas.transform.GetChild(0))
            {
                if (ChosenCard.gameObject.name == MoveCard)
                {

ChosenCard.gameObject.GetComponent<Button>().gameObject.SetActive(false);
                }
            }

            bool isAttack = false;
            for (int i = 0; i < playerManager.spellsArray.Length; i++)
            {
                if (playerManager.spellsArray[i].getNameSpell() == MoveCard)
                {
```

```
                    isAttack = true; // this is an attack

playerManager.setTotalStamina(playerManager.getTotalStaminaPlayer() -
playerManager.spellsArray[i].getStaminaSpell());

                    Debug.Log("kkkkkkkkkkkkkkkk " +
playerManager.getTotalStaminaPlayer() + " " +
playerManager.spellsArray[i].getStaminaSpell() + " " + MoveCard);

                    StaminaArray[PlayNum] =
playerManager.spellsArray[i].getStaminaSpell();
                    DamageArray[PlayNum] = playerManager.spellsArray[i].getDamage();

                }
            }

            if (isAttack == false)
            {
                StaminaArray[PlayNum] = 0;
                DamageArray[PlayNum] = 0;
            }
            //bool s = false;

            if (StaminaArray[PlayNum] < 0) //if the card is MagicRenewal
            {
                if (playerManager.getTotalStaminaPlayer() - StaminaArray[PlayNum] >
playerManager.getStaminaPlayer())
                {
                    playerManager.setTotalStamina(playerManager.getStaminaPlayer());
                }
                MoreStamina();
            }

            for (int i = 0; i < playerManager.spellsArray.Length; i++)
            {
                DontHaveEnoughStamina(playerManager.spellsArray[i]);
            }

            MoveOrderArray[PlayNum] = MoveCard;
            ListOfMoves();
        }
    }
```

הפונקציה cellMove מקבלת ערך string של הקלף שהשחקן בחר ומכניסה לתוך מערך של קסם באורך 3 כמה הקסם עלה, ולמערך של פגיעה כמה הקלף מוריד לאויב, ומוסיפה את זה למערך שמות המהלכים של אותו תור. בנוסף היא "מעלימה" אותו כדי שהשחקן לא יוכל לבחור בו פעם נוספת.

```
IEnumerator Continue()
    {
        screenManager.isScreen();
        yield return new WaitForSeconds(2);
        setActiveCards();

        enemyManager.pickAttacks(attack.getCrntRow(), attack.getCrntClmns(),
attack.getEnemyCrntRow(), attack.getEnemyCrntClmns());

        for(int i=0; i<MoveOrderArray.Length; i++)
        {
            if (PlayerBeforeEnemy(i) == true)
            {
```

```
                WhichMove(MoveOrderArray[i]);
                direction();
                isenemyHited(i);
                staiminabarPlayer.TakeStamina(StaminaArray[i], PlayerData.Stamina);
                yield return new WaitForSeconds(2);
                DeleteImageButton(i.ToString());

                if(MoveOrderArray[i] != "Shield")
                {
                    foreach (Transform SquareFloor in BlocksArray.transform)
                        SquareFloor.gameObject.GetComponent<SpriteRenderer>().color =
new Color(255, 255, 255);
                    yield return new WaitForSeconds(1);
                }

                if (enemyManager.getTotalHpEnemy() <= 0)
                {
                    Debug.Log("you won");
                    i = 2;
                    stop = true;
                    Destroy(enemy);
                    winOrLoseWindow(true);
                }

                if (stop == false)
                {
                    WhichMove(enemyManager.getCurrentMove(i));
                    direction();
                    isplayerHited(i);

staiminabarEnemy.TakeStamina(enemyManager.getCurrentstaminaMove(i),
enemyManager.getStaminaEnemy());
                    yield return new WaitForSeconds(2);
                }
            }
            else
            {
                WhichMove(enemyManager.getCurrentMove(i));
                direction();
                isplayerHited(i);
                staiminabarEnemy.TakeStamina(enemyManager.getCurrentstaminaMove(i),
enemyManager.getStaminaEnemy());
                yield return new WaitForSeconds(2);

                if(enemyManager.getCurrentMove(i) != "ShieldEnemy")
                {
                    foreach (Transform SquareFloor in BlocksArray.transform)
                        SquareFloor.gameObject.GetComponent<SpriteRenderer>().color =
new Color(255, 255, 255);
                    yield return new WaitForSeconds(1);

                }
                if (playerManager.getTotalHpPlayer() <= 0)
                {
                    Debug.Log("you lost");
                    i = 2;
                    stop = true;
                    Destroy(player);
                    winOrLoseWindow(false);
                }
                if (stop == false)
                {
```

```
                    WhichMove(MoveOrderArray[i]);
                    //hpbarPlayer.TakeStamina(-ifHeal, playerManager.getHpPlayer());
                    //ifHeal = 0;
                    direction();
                    isenemyHited(i);
                    staiminabarPlayer.TakeStamina(StaminaArray[i],
PlayerData.Stamina);
                    yield return new WaitForSeconds(2);
                    DeleteImageButton(i.ToString());
                }
            }
            if (playerManager.getTotalHpPlayer() <= 0)
            {
                Debug.Log("you lost");
                i = 2;
                stop = true;
                Destroy(player);
                winOrLoseWindow(false);
            }
            if (enemyManager.getTotalHpEnemy() <= 0)
            {
                Debug.Log("you won");
                i = 2;
                stop = true;
                Destroy(enemy);
                winOrLoseWindow(true);
            }

            foreach (Transform SquareFloor in BlocksArray.transform)
                SquareFloor.gameObject.GetComponent<SpriteRenderer>().color = new
Color(255, 255, 255);
            yield return new WaitForSeconds(1);
        }

        if(stop == false)
        {
            PlayNum = 0;
            isFull = false;

            for (int i = 0; i < MoveOrderArray.Length; i++)
                MoveOrderArray[i] = "";

            if (playerManager.getTotalStaminaPlayer() +
playerManager.getStaminaBonus() > 100)
            {
                playerManager.setTotalStamina(100);
            }

            else
            {
                playerManager.setTotalStamina(playerManager.getTotalStaminaPlayer() +
playerManager.getStaminaBonus());
            }

            staiminabarPlayer.TakeStamina(-playerManager.getStaminaBonus(),
PlayerData.Stamina);
            staiminabarPlayer.resetStats(playerManager.getTotalStaminaPlayer());

            staiminabarEnemy.TakeStamina(-enemyManager.getStaminaBonus(),
enemyManager.getStaminaEnemy());
            staiminabarEnemy.resetStats(enemyManager.getTotalStaminaEnemy());
```

```
            for (int i = 0; i < StaminaArray.Length; i++)
            {
                //Debug.Log("stamina array " + i + " " + StaminaArray[i]);
                StaminaArray[i] = 0f;
            }

            MoreStamina();

            screenManager.isContinueB();
            enemyManager.ResetEnemy();
            CourtPositions();
            Debug.Log("choose moves again");
        }
    }
```

הפונקציה אחראית לניהול הדו קרב כאשר היא מנהלת את סדר התורות, שליחת מידע לפונקציות הרלוונטיות המנהלות את ברי החיים והקסם, את צבע המשבצות ואת תזוזת השחקן והאויב.

```
private void DontHaveEnoughStamina(SpellClass spell)
    {
        foreach (Transform ChosenCard in ButtonsCanvas.transform.GetChild(0))
        {
            if (ChosenCard.gameObject.name == spell.getNameSpell() &&
spell.getNameSpell() != "MagicRenewal")
            {
                float sum = playerManager.getTotalStaminaPlayer() -
spell.getStaminaSpell();
                if (sum < 0)
                {
                    ChosenCard.gameObject.GetComponent<Button>().interactable = false;
                }
            }
        }
    }
```

פונקציה הנקראת לאחר שהשחקן בחר קלף שבודקת לגבי שאר הקלפים שהשחקן יכול לבחור האם יש לו מספיק קסם כדי להשתמש בהם, ואם לא אז הוא לא יוכל לבחור אותם.

ספריות

- **UnityEngine**: הכוונה היא שהכותב משתמש בגרסת **C #** של **unity**. שימוש: כתיבת ב**c#** ב**unity**.
- **System.collections**: מרחב שמות של אוספים מכיל ממשקים ומעמדות המגדירים אוספים שונים של חפצים, כגון רשימות, תורים, מערכי סיביות, טבלאות חשיש ומילונים. שימוש: בשביל להשתמש ברשימות ובעוד אוספים שונים של חפצים בקוד.
- **UnityEngine.UI**: מכיל קבועים הקשורים ל**ui** של **unity**. שימוש: במטרה לשנות או לעבוד עם ה **ui** בעזרת קוד.
- **System.collections.Generic**:מרחב שמות גנרי מכיל ממשקים וקטגוריות המגדירות אוספים גנריים, המאפשרים למשתמשים ליצור אוספים שהוקלדו בצורה חזקה המספקים בטיחות וביצועים טובים יותר מאשר אוספים לא כלליים שהוקלדו בצורה חזקה. שימוש: מאפשר לי ליצור אוסף של גינרי בקוד.
- **Proyecto26**: **asset** שמאפשר לנו לתקשר עם **firebase**. שימוש: במטרה לשלוח לבסיס נתונים.
- **UnityEngine.SceneManagement**: ניהול סצנות בזמן ריצה. שימוש: לעבור בין סצנות.
- **unityEngine.serialization**: מציין שניתן לסדר את הכיתה או המבנה.
  כדי להפעיל את ההמשכים, החל את התכונה [**Serializable**] למידע נוסף על הסדרת ההמשכים, ראה. שימוש: במטרה להפוך משתנים ל**Serializable** .

רפלקציה אישית

הפרויקט עבורי היווה מעין "תהליך התבגרות" בכתיבת קוד – התמודדתי בעצמי עם ניהול תיק פרויקט ומספר לא קטן של קבצים, ניהול זמנים באופן שאספיק לעמוד ביעדי כמתוכנן ולפי ההגשות, פתירת בעיות וחשיבה על אלגוריתמים, חקירה באינטרנט לגבי טכנולוגיות הקיימות בסביבת העבודה וכתיבת קוד מסודר.

אני חושב שהדבר שבזכותו הפרויקט שלי הוא כמו שהוא עכשיו זה הרצון והמוטיבציה שיש לי לבנות אותו, ועצם העובדה שהנושא שבחרתי הוא משהו אישי ומקורי גרם לי אפילו יותר לעבוד עליו.

המסקנות שאני לוקח איתי מבניית הפרויקט זה רוב מה שציינתי עד כה, ובעיקר הניסיון שצברתי. אני מאוד מקווה שניסיון זה יעזור לי בעתיד, שכן בזכות הפרויקט אזכה להתפתח ולהתקדם בתחום הזה.

במידה והייתי מתחיל את הפרויקט היום אני בטוח שהוא היה הרבה יותר מסודר וקצר, הייתי מתכנן מראש את סדר העבודה שלי וכך לא הייתי מתרכז בדברים שבדיעבד היו שוליים ובזבזתי עליהם זמן, הייתי צריך פחות לחקור כיוון שהידע שיש לי עכשיו כפול ואפילו משולש משהיה לי בתחילת הפרויקט, וכמובן שהזמן שבו הייתי עובד היה הרבה יותר ממוקד ויעיל ולא הייתי צריך המון זמן כמו שקרה.

ביבליוגרפיה

https://stackoverflow.com/

https://learn.unity.com/tutorials

https://www.youtube.com/

שלושת מקורות המידע האלו היו המרכזיים שבהם השתמשתי, כמו כן בהם היה את המידע השימושי ביותר עבורי ותוצאות החיפוש שלי כשחקרתי נושא מסוים לרוב הביאו אותי לאתרים האלה.

אני רוצה להודות לחברי דניאל טל שעזר לי מאוד בבניית הפרויקט.