**פק מן**

**מגיש : עפרי לייפר**

**מורה : אורנה אברך**

**בית ספר : ריאלי בית בירם**

**תוכן עניינים**

|  |  |
| --- | --- |
| **תוכן עניינים** | **2** |
| **הקדמה** | **3** |
| **מבנה המבוך** | **4** |
| **מהלך המשחק** | **5** |
| **סיום משחק** | **6** |
| **קהל יעד** | **7** |
| **נתונים** | **8** |
| **חוקי אתחול ומחיקה** | **9** |
| **חוקים הקשורים בניהול המשחק** | **10** |
| **חוקי עזר כלליים** | **11** |
| **חוקים גרפיים** | **14** |
| **חוקים הקשורים לאלגוריתם** | **15** |
| **האלגוריתם** | **16** |
| **שימוש באלגוריתם** | **18** |
| **גרפיקה וממשק המשתמש** | **20** |
| **הקוד** | **21** |
| **שימוש במערכת והרצתה** | **54** |
| **הצעות לשיפור ולשכלול** | **56** |

**הקדמה**

**פק-מן הוא במקור משחק ארקייד, יצא לאור לראשונה ביפן ב-22 במאי 1980 , המשחק מיועד לשחקן אחד, אשר יתמודד מול המחשב ברמת קושי שבחר, המשחק דורש תגובה בזמן אמת וחשיבה קדימה מהמשחק בה.**

**במשחק לדמויות יש אפשרות לזוז למעלה, למטה, שמאלה או ימינה, תוך כדי כך שהם מוגבלים ממפת המבוך שתוסבר בהמשך, ישנם שני סוגי דמויות :**

**"פק-מן" - משוחק על ידי השחקן, נראה כעיגול צהוב עם פה.**

**"רוח" - משוחק על ידי המחשב, נראה כרוח בצבע ירוק.**

**פק-מן משוחק במפת מבוכים, אשר אין אפשרות לפק-מן או לרוחות לצאת ממנה, במפה יש מעבר באמצע המפה בצד שמאל וימין, מעבר זה מאפשר לפק-מן ולרוחות להגיע מקצה אחד של המעבר לקצהו השני בצד ההופכי של המפה.**

**במשחק פק-מן המטרה היא לאכול את כל הגלולות הרגילות הנמצאים על המפה, או לאכול את כל הרוחות הנמצאים במפה בהיעזר בגלולת הפלא.**

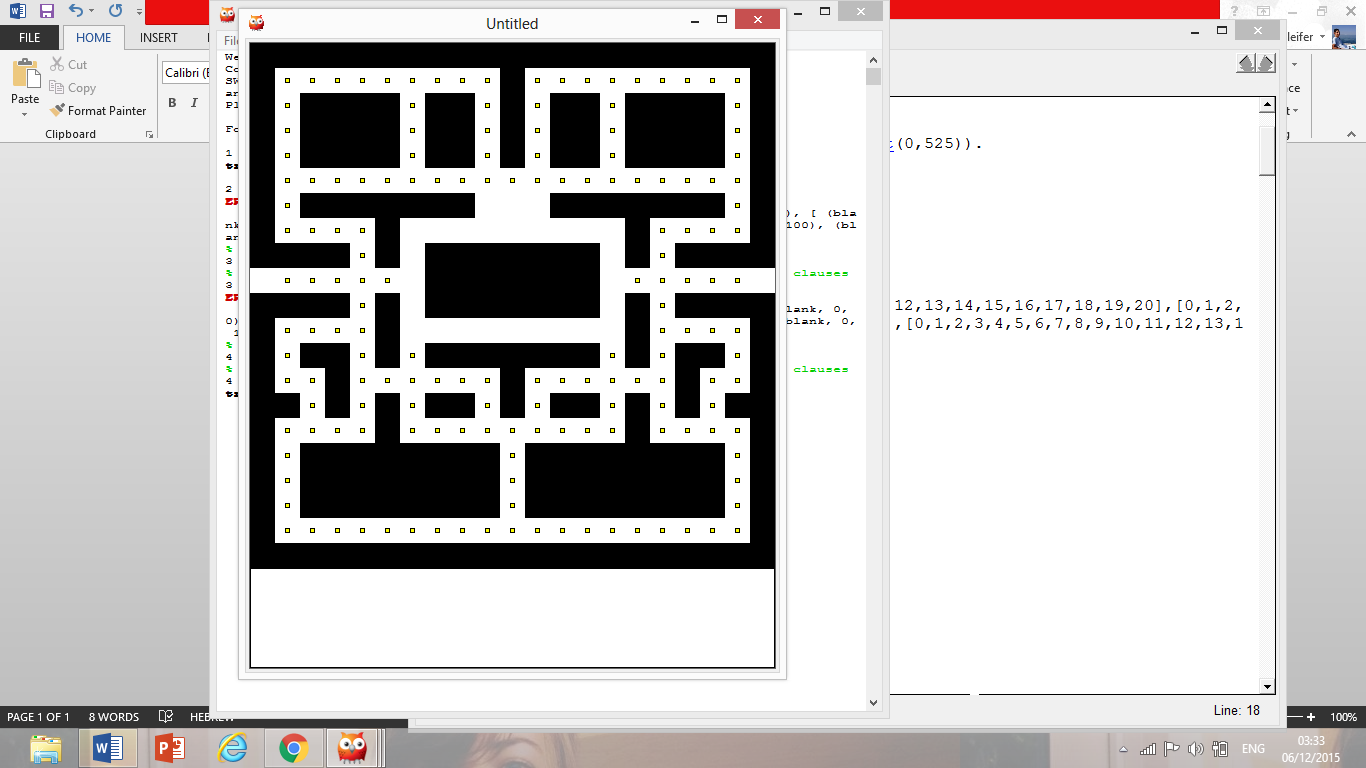
**על השחקן לתמרן את "פק-מן" ולנווט אותו דרך מבוך, בעודו אוכל גלולות, סוגי הגלולות הם :**

**"גלולה רגילה"  - (ריבועים צהובים קטנים) מקנות לו נקודות כאשר הוא אוכל אותם.**

**"גלולת פלא" - (עיגולים אדומים) מקנות לו את האפשרות לאכול את הרוחות לזמן מוגבל, כאשר פק-מן נוטל "גלולת פלא", הצבע של הרוחות הופך לכחול-כהה כסימן לשחקן שביכולתו לאכול גם אותם.**

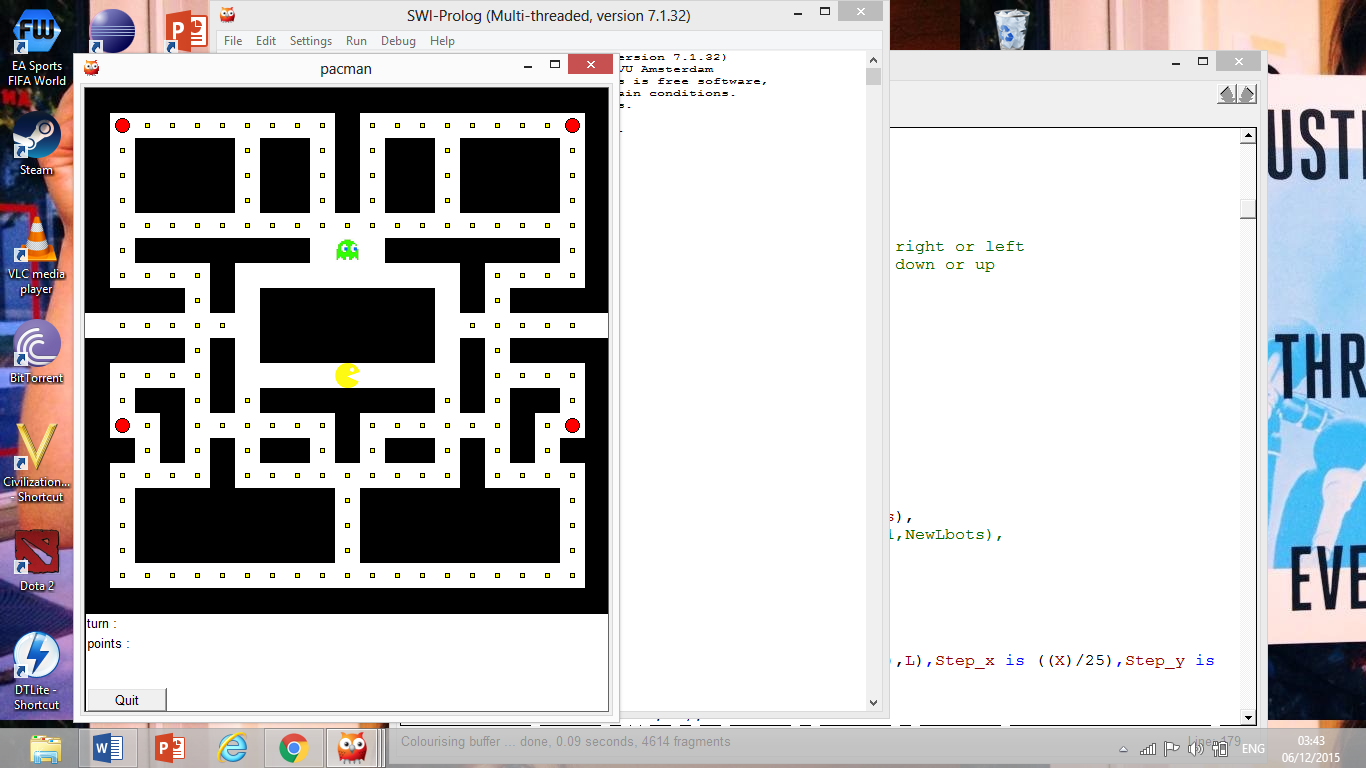
**מבנה המבוך**

**המבוך נראה כך - כל האזורים שצבועים בשחור הם האזורים שבהם הדמויות לא יכולות לעבור, בנוסף אפשר לשים לב לחורים בחלק האמצעי של המפה בצד שמאל וימין (החיצים הכחולים מצביעים עליהם ), החורים מאפשרים מעבר לחור בצד השני של המפה ולהפך.**



**מהלך המשחק**

**במצב ההתחלתי הפק מן נמצא מתחת לקופסה המרכזית והרוחות נמצאים מעל הקופסה המרכזית (ראה ציור).**



**פק מן – נשלט על ידי המשתמש באמצעות המקשים :**

**W / חץ למעלה - למעלה**

**D / חץ ימינה - ימינה**

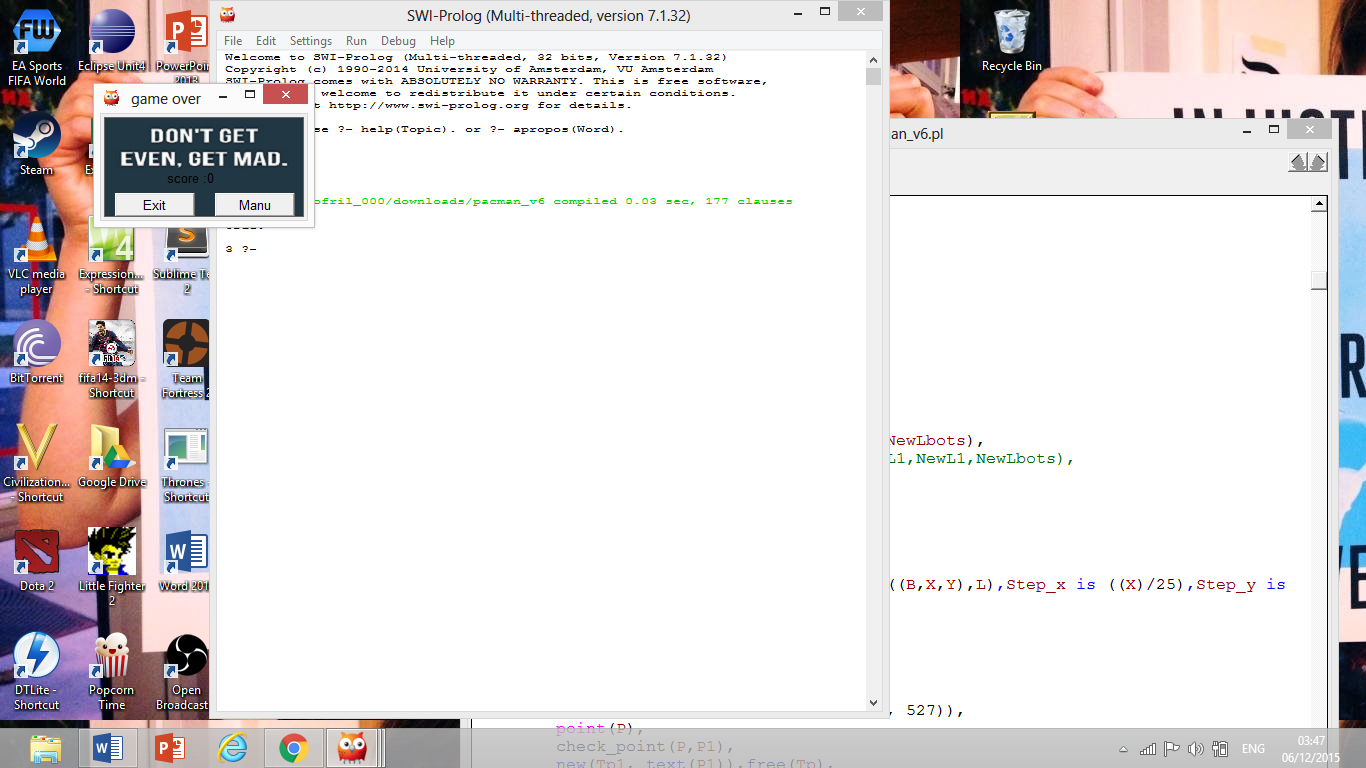
**S / חץ שמאלה - שמאלה**

**X / חץ למטה - למטה**

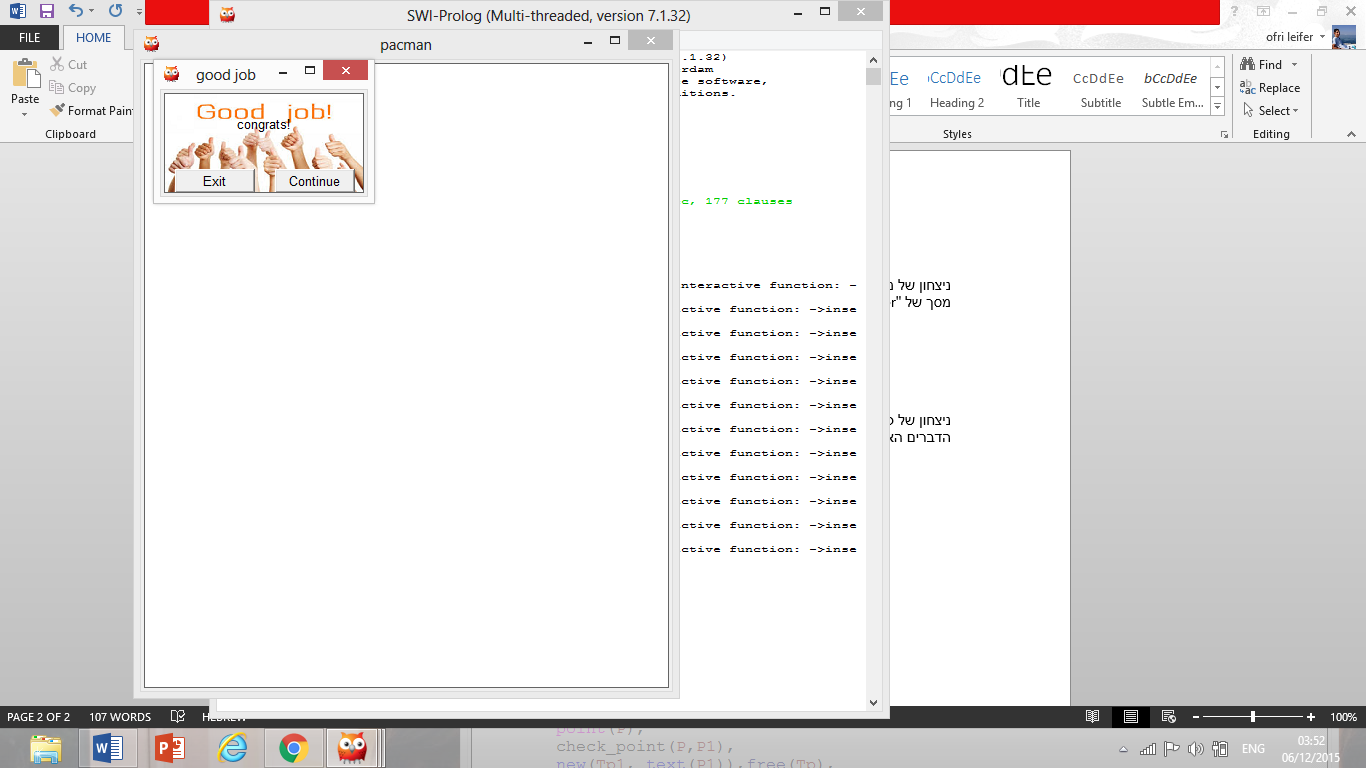
**מטרת הפק מן היא לאכול את כל הגלולות הרגילות הנמצאים על המפה, או לאכול את כל הרוחות הנמצאים במפה בהיעזר בגלולת הפלא**

**סיום המשחק**

**ניצחון של מחשב קורה כאשר הרוחות אכלו את הפק מן, לאחר שיאכלו אותו מסך של "game over" יופיע כמו שבתמונה.**



**ניצחון של פק מן קורה כאשר הפק מן אכל את כול הרוחות או אכל את כל הגלולות הרגילות, לאחר שאחד הדברים האלו מתקיימים מופיע מסך של "good job" כמו שבתמונה.**



**קהל היעד**

העבודה פונה לכל משתמש ולמעשה המשחק מיועד לכל אדם.

כללי המשחק אינם פשוטים אך אחל מהרגע שמבינים אותם, המשחק מיועד לכולם והוא משחק מאתר ומעניין מאוד.

**נתונים**

|  |  |  |
| --- | --- | --- |
| **הסבר** | **פרמטרים** | **שם העובדה** |
| **עובדה שכוללת כיוון והתזוזה ששייכת לו** | **D,Mx,My** | **move** |
| **עובדה דינמית שמקשרת בין catch\_order**  **ל player\_move** | **D** | **order** |
| **עובדה דינמית שמשומשת כתנאי עצירה למשחק כאשר O ניצח** | **O** | **winner** |
| **עובדה דינמית שמקשרת בין התמונה של הפק מן לכל מקום שהיה את הצורך בהפניה שלה** | **B** | **mouse** |
| **עובדה דינמית שמשמשת כדי לעקוב אחרי ניקוד של משחק נוכחי** | **N** | **point** |
| **עובדה דינמית שמשמשת כדי לבחור את הרמה X** | **X** | **user\_level** |
| **עובדה דינמית שמשמשת כדי לעקוב אחרי השפעת גלולת פלא** | **T,N** | **game\_mode** |
| **עובדה דינמית על מנת לקשר בין תמונה של גלולה לפעולה שמוחקת אותה כאשר אוכלים אותה** | **X,Y,B** | **point** |
| **עובדה דינמית על מנת לקשר בין תמונה של גלולה לפעולה שמוחקת אותה כאשר אוכלים אותה** | **X,Y,B** | **spoint** |
| **עובדה דינמית שמאפשרת לאלגוריתם לבחור איזה מהלך לבחור בהתאם לרמת הקושי T** | **T** | **difficulty** |

**חוקי אתחול ומחיקה**

|  |  |  |
| --- | --- | --- |
| **הסבר** | **פרמטרים** | **שם החוק** |
| **מעבר לשלב הבא** | **W** | **next\_level** |
| **סוגר את התוכנה** | **W** | **close\_program** |
| **מעבר להתחלת משחק ברמת קושי קלה** | **W** | **easy** |
| **מעבר להתחלת משחק ברמת קושי בינונית** | **W** | **medium** |
| **מעבר להתחלת משחק ברמת קושי קשה** | **W** | **hard** |
| **סוגר משחק נוכחי** | **-** | **close** |
| **מוחק את כל האיברים הדינמיים שברשימה** | **L** | **delete\_all** |
| **מוחק את כל האיברים הדינמיים על מנת להתחיל משחק חדש** | **-** | **retract\_all** |

**חוקים הקשורים בניהול המשחק**

|  |  |  |
| --- | --- | --- |
| **הסבר** | **פרמטרים** | **שם החוק** |
| **דיאלוג סיום משחק** | **-** | **show\_gameover** |
| **דיאלוג הצלחת שלב** | **-** | **show\_good\_feedback** |
| **דיאלוג תפריט התחלה לאחר משחק אחד לפחות** | **W1** | **start** |
| **דיאלוג תפריט התחלה** | **-** | **start** |
| **מעבר למשחק ברמה X** | **X** | **game** |
| **מתחיל משחק ברמה X** | **X** | **level** |
| **מתחיל תור מסוים במשחק נוכחי** | **Data,Map,Lbots** | **turn** |
| **מחזיר את המהלך הבא של הפק מן** | **M,D,X,Y,ND** | **player\_move** |
| **בודק אם הפק מן ניצח על ידי אכילת כל הגלולות הרגילות** | **P,NP** | **check\_point** |
| **בודק אם הפק מן הפסיד ומוחק את הרוחות שפק מן אכל אותם בעזרת גלולת הפלא** | **L1,(X5,Y5),L1,NL1,NL** | **check\_wincat** |
| **בודק אם הפק מן ניצח על ידי אכילת כל הרוחות** | **L** | **check\_winmouse** |
| **משנה אם הפק מן מושפע על ידי גלולת פלא** | **-** | **check\_mode** |

**חוקי עזר כלליים**

|  |  |  |
| --- | --- | --- |
| **הסבר** | **פרמטרים** | **שם החוק** |
| **יוצר רשימה של N רוחות** | **N,List** | **bots** |
| **מקשר בין רשימה של אובייקטים גראפיים לרשימה של שמות** | **L,L,TB** | **get\_new\_board** |
| **יוצר רשימה לאובייקטים גראפיים שהיא רשימה של שמות** | **L,L1** | **change\_board\_term\_to\_string** |
| **מעבר לפעולה**  **minimax** | **L,L1,CB,M,FL,X,T,(V,A,B)** | **find\_minimax\_for\_list** |
| **יוצר לוח חדש בהתאם למאורעות בלוח הקודם** | **B,OB,NB** | **edit** |
| **מעבר לחוק**  **edit** | **B,B1** | **edit\_board** |
| **מוצא לוחות אפשריים כאשר זהו התור של המחשב לפי לוח מסוים**  **B** | **M,B,NB** | **find\_possible\_boards\_for\_board** |
| **מוצא לוחות אפשריים כאשר זהו התור של הפק מן לפי לוח מסוים**  **B** | **M,B,NB** | **find\_possible\_boards\_for\_pacman** |
| **מוצא אם המהלך לכיוון D**  **אפשרי מהמקום**  **X,Y** | **M,X,Y,D** | **possible** |
| **מקשר בין לוח ישן ללוח אפשרי לו** | **L,B** | **get\_board** |
| **מקשר בין תזוזה מסוימת ממקום מסוים למקום הסופי שלו** | **(Cx,Cy),(Mx,My),D,X,Y** | **final\_position** |
| **מחזיר רשימה של מטרות בציר ה x שחוסמות את הפק מן לפי מקומו** | **M,Cx,T,L** | **find\_list\_of\_line\_x** |
| **מחזיר רשימה של מטרות בציר ה y שחוסמות את הפק מן לפי מקומו** | **M,Cy,T,L** | **find\_list\_of\_line\_y** |
| **מקשר בין לוח אפשרי לרשימה של תנועות שמובילות ללוח** | **L,L1** | **find\_moves** |
| **מעבר לניקוד של הלוח לפי קרבה לנקודות המטרה שלהם** | **FL,B,P** | **get\_distance\_points** |
| **מחזיר את המרחק בין שתי מקומות** | **(Cx,Cy),(Cx1,Cy1),D** | **distance\_between\_place\_and\_object** |
| **מחזיר תוצאה של פיתגורס** | **X,Y,R** | **pitaguras** |
| **מוודא שלא יותר מרוח אחת תתקדם לעבר אותה מטרה** | **(Cx,Cy),FL,List** | **findall\_closer** |
| **מחזיר את מיקומו של הפק מן לאחר ביצוע המהלך** | **(X5,Y5),(X,Y)** | **position\_of\_mouse** |
| **שולח את ההוראה שנובעת מלחיצה על מקש** | **B** | **catch\_order** |
| **בודק אם המהלך אפשרי** | **M,B,D** | **possible** |
| **מעבר בין הוראה ליצירת איבר דינמי שהקרא לאחר מכן ב player\_move** | **D** | **change\_o** |
| **מעביר לחוקים שמחזירים ומראים את המפה** | **L** | **board\_map1** |
| **מעבר לחוק שמחזיר את המפה כרשימה** | **L** | **places\_map1** |
| **מחזיר את המפה כרשימה** | **Lx,Ly,Li,FL** | **places\_map1** |

**חוקים גרפיים**

|  |  |  |
| --- | --- | --- |
| הסבר | **פרמטרים** | **שם החוק** |
| **מבצע אנימציה לרשימה L שכוללת בתוכה איברים שכוללים אובייקט גראפי, הצעד שלו בציר ה y והצעד שלו בציר ה x** | **L,L,Frames** | **anilist** |
| **בודק אם**  **הפק מן תחת השפעת גלולת פלא ומשנה את Lbots בהתאם** | **Lbots,NewLbots** | **change\_ghost** |
| **בודק את כיוונו של הפק מן ומשנה את תמונתו בהתאם** | **X5,Y5** | **changemouse** |
| **משנה את המקום של דמות במשחק אם הם עברו במעבר האמצעי המפה** | **B,D** | **change\_place** |
| **משנה את המקום של כל הדמויות המשוחקות על ידי המחשב** | **L** | **change\_place\_for\_bots** |
| **מראה את המפה** | **-** | **display\_map1** |
| **מראה קופסה לפי גודלה ומיקומה** | **Sw,Sh,Px,Py** | **box\_style** |

**חוקים הקשורים באלגוריתם**

|  |  |  |
| --- | --- | --- |
| **הסבר** | **פרמטרים** | **שם החוק** |
| **מעבר לאלגוריתם בהתאם לרמת הקושי של המשחק הנוכחי** | **Map,Lbots,(X5,Y5),L1** | **bot\_move** |
| **מחזיר מהלך אקראי לרשימה Lbots ב L1** | **Map,Lbots,L1** | **random\_move** |
| **מחזיר מהלך חכם בהתאם לרמת הקושי לרשימה Lbots ב L1** | **M,L,(X5,Y5),L1** | **smart\_move** |
| **מחזיר לוח אפשרי שהוא הכי טוב בהתאם לרמת הקושי** | **M,(Cx,Cy),CB,TB** | **get\_chosen\_board** |
| **מימוש אלגוריתם של minimax , ראה הסבר נוסף בפרק "האלגוריתם"** | **OB,M,FL,CB,(B,P),X,T,(Alpha,Beta)** | **mini\_max** |
| **מחזיר רשימה של מטרות למחשב בהתאם לרמת הקושי ומיקומו של הפק מן – או במילים אחרות לפי האסטרטגיה** | **M,(Cx,Cy),L** | **target\_for\_bots** |
| **מחזיר רשימה של מטרות שחוסמות את הפק מן לפי מקומו** | **(Cx,Cy),FLx,FLy,FL** | **get\_final\_spots** |
| **מחזיר נקודות ללוח מסוים לפי פונקציה היוריסטית** | **B,FL,P** | **board\_points** |
| **מחזיר ניקוד של לוח לפי מצב של ניצחון/הפסד** | **B,P** | **massive\_points** |

**האלגוריתם**

הבינה המלאכותית במשחק נכתבה באמצעות אלגוריתם ה MINMAX.

**MINMAX**

אלגוריתם ה MINMAX הוא אלגוריתם החלטה למשחק סכום אפס דו-שחקני.

בכל שלב במשחק בו האלגוריתם מופעל עבור שחקן א' , נפרסות כל האפשרויות שיש לשחקן א' בתור הנוכחי, לאחר מכן נפרסות כל האפשרויות של שחקן ב' בתור הבא וכן הלאה. בצורה הזאת נפרס עץ מהלכים שנקרא 'עץ המינמקס' . גובה/עומק העץ הוא פרמטר שנקבע לפני תחילת המשחק ועלי העץ הם מצבים שנגיע אליהם לאחר רצף מסוים של מהלכים.

לכל מצב בעלה מתאימים ציון ע"י פונקציה שנקראת 'הפונקציה ההיוריסטית' , הציון משקף כמה המצב עבור שחקן א' . כעת סורקים את החל בעלים, דרך הצמתים שבעץ עד השורש. הציון שניתן לכל צומת הוא הערך הגבוה ביותר של הצמתים/עלים שתחתיו אם הרמה התחתונה היא מהלך של שחקן א' ואחרת (הרמה התחתונה היא מהלך של שחקן ב') מותאם הערך הנמוך ביותר של הצמתים/עלים שתחתיו. כאשר מגיעים לשורש, בוחרים את הקודקוד שמתחת לשורש עם הציון הטוב ביותר.

בצורה הזאת, לא מובטח לנו שנגיע למצב הטוב ביותר במשחק אך מובטח לנו שאנחנו עושים את המהלך הטוב ביותר שאנחנו יכולים כתלות בעומק העץ שלנו.

כפי שנאמר עומק העץ הוא פרמטר קבוע שנקבע בתחילת המשחק. באופן אידיאלי יש לבחור את עומק העץ גדול ככל שיותר (בצורה הזאת מקבלים עץ גדול יותר ויש לנו יותר מידע שניתן להשתמש בו על מנת לבור מהלך הטוב ביותר), אך כאשר מגדילים את העומק הזיכרון הדרוש גדל באופן אקספוננציאלי ולכן יש חסם לעומק המקסימלי שאנחנו יכולים לבחור.

**AlphaBeta pruning**

על מנת להקטין את הסיכון של לעבור את כמות הזיכרון המקסימלית שהממשק יכול להכיל, יש להשתמש בשיטה הקרויה AlphaBeta pruning , השיטה כוללת אלגוריתם במטרתו לקצץ את החיפוש של עץ ה MINMAX וכתוצאה מכך קיצור זמן הריצה וצמצום הזיכרון שנדרש בהרצה.

הרעיון מאחורי האלגוריתם הוא בדיקת האם יש צורך להמשיך לחפש צומת מסוימת תוך שימוש בשני משתנים :

Alpha : הערך המקסימלי שכבר נמצא בדרך לצומת הנוכחית

Beta : הערך המינימלי שכבר נמצא בדרך לצומת הנוכחית

בעזרתם נוכל לדעת האם כדאי להמשיך לחקור צומת מסוימת או לא.

**שימוש באלגוריתם**

**במשחק יש לשחקן אפשרות לבחור את רמת המחשב מולו הוא משחק.**

**ברמת מחשב קלה (easy) – המהלך יהיה רנדומלי**

**ברמת מחשב בינונית (medium) – עומק העץ שנפרס באלגוריתם הוא 1 .**

**ברמת מחשב קשה (hard) – עומק העץ שנפרס באלגוריתם הוא 3.**

**על מנת להפעיל את האלגוריתם, יש לבחור את האסטרטגיה שהכי טובה בהתאם לרמת הקושי, ולאחר מכן את הפונקציה היוריסטית שגם היא בהתאם לרמת הקושי.**

**האסטרטגיות נבחרות כדלהן :**

* **רמת קושי – קל המהלך יהיה רנדומלי ללא אסטרטגיה כלשהי.**
* **רמת קושי – בינונית המהלך יהיה לפי השטח שבו הפאקמאן נמצא, לכל אזור אחר בלוח/מפה יש אסטרטגיה שונה, להלן האסטרטגיות :**
* **רמת קושי – קשה המהלך יהיה לפי המצב של המשחק, לכל מצב יש אסטרטגיה שונה ואם אף אחד מהמצבים לא מתקיים אז המהלך יהיה לפי רמת קושי בינוני, האמצעי, אסטרטגיית רדיפה, אסטרטגיית לוכד הפינות .**

**להלן פירוט האסטרטגיות :**

**אסטרטגיית הפיתיון - האסטרטגיה תתקיים ברמת קושי קשה כאשר הפקמן אכל גלולת פלא וביכולתו לאכול את הרוחות, ע"פ האסטרטגיה על הרוחות להגיע למקומות האפשריים שהפקמן יכול לזוז במהלך הבא.**

**אסטרטגיית אובייקטיבית - האסטרטגיה תתקיים ברמת קושי קשה כאשר נשאר לפק מן עוד 10 כדורים צהובים, 10 נקודות אשר אם ישיג אותם הפק מן ינצח, ע"פ האסטרטגיה על הרוחות להגיע למקומות של הנקודות על מנת שהפק מן לא יוכל לעבור את השלב.**

**אסטרטגיית לוכד הפינות - האסטרטגיה תתקיים ברמת קושי בינוני או קשה כאשר הפק מן נמצא קרוב לפינת הלוח/מפה, ע"פ האסטרטגיה על הרוחות להגיע למקומות החוסמים את הפק מן מלצאת מהפינה.**

**אסטרטגיית רדיפה - האסטרטגיה תתקיים ברמת קושי בינוני או קשה כאשר הפק מן נמצא בחלק האמצעי של הלוח/מפה, ע"פ האסטרטגיה על הרוחות לרדוף אחרי הפק מן במטרה לאכול אותו.**

**אסטרטגיית המעבר האמצעי - האסטרטגיה תתקיים ברמת קושי בינוני או קשה כאשר הפק מן נמצא במעבר האמצעי של הלוח/מפה המאפשר לא לעבור לקצה השני של המפה במהלך אחד, ע"פ האסטרטגיה על הרוחות להגיע למקומות החוסמים את המעבר האמצעי וכתוצאה מכך לחסום גם כן את הפק מן.**

**הנוסחא היוריסטית נבחרת כדלהלן**

**רמת קושי – קלה difficulty (easy) המהלך יהיה רנדומלי ללא שום נוסחא היוריסטית.**

**רמת קושי – בינונית difficulty (medium) אם הפק מן חסר(נאכל) 8000 נק', בנוסף למרחק מנקודות המטרה (שנקבעות לפי האסטרטגיה) , המרחק כפול 100 נכלל בסיכום הנקודות.**

**רמת קושי – קשה difficulty (hard) אם הפק מן חסר(נאכל) מוסיף 8000 נק', בנוסף למרחק מנקודות המטרה (שנקבעות לפי האסטרטגיה) , המרחק כפול 100 נכלל בסיכום הנקודות, בנוסף לכך כל רוח שחסרה מורידה 8000 נק' .**

**גרפיקה וממשק המשתמש**

* מסך הפתיחה - במסך הפתיחה השחקן מתבקש לבחור את רמת הקושי על ידי לחיצת כפתור .
* מסך המשחק - במסך הפתיחה השחקן מתבקש לוודא שהעכבר נמצא בחלון ולא מחוץ לחלון, לאחר שזה נעשה השחקן מתבקש לשחק על ידי לחיצה על המקשים המתוארים בפירוש בהקדמה, יש לשחקן אפשרות גם לפרוש מהמשחק בלחיצה על הכפתור בפינה התחתונה שמאלית של החלון .
* מסך סיום - מסך הסיום יכול להיות שני מסכים אפשריים שהם :

א. ניצחון לפק מן : מסך המברך את השחקן על

הצלחתו בשלב והשחקן מתבקש לבחור אם

להמשיך לשלב הבא או לצאת מהמשחק על

ידי לחיצת כפתור .

ב. ניצחון למחשב : מסך המתגרה בשחקן על

כישלונו בשלב והשחקן מתבקש לבחור אם

לחזור למסך פתיחה או לצאת מהמשחק על

ידי לחיצת כפתור .

**הקוד**

:-dynamic order/1.

:-dynamic winner/1.

:-dynamic mouse/1.

:-dynamic point/1.

:-dynamic user\_level/1.

:-dynamic game\_mode/2.

:-dynamic point/3.

:-dynamic spoint/3.

:-dynamic difficulty/1.

show\_gameover:-

new(W,window('game over',size(200,100))),

send(W,background,bitmap('mad.bmp')),

send(W,open),

point(P),

retractall(point(\_)),

send(W, display,text('score : '), point(62, 52)),

send(W, display,text(P), point(102, 52)),

new(B1, button(exit,message(@prolog, close\_program,W))),

send(W,display,B1,point(10,75)),

new(B2, button(manu,message(@prolog, start,W))),

send(W,display,B2,point(110,75)).

show\_good\_feedback:-

new(W,window('good job',size(200,100))),

send(W,background,bitmap('goodjob.bmp')),

send(W,open),

send(W, display,text('congrats! '), point(72, 22)),

new(B1, button(exit,message(@prolog, close\_program,W))),

send(W,display,B1,point(10,75)),

new(B2, button(continue,message(@prolog, next\_level,W))),

send(W,display,B2,point(110,75)).

next\_level(W):-

free(W),

user\_level(X),

X1 is X+1,

level(X1).

start(W1):-

free(W1),

retractall(difficulty(\_)),

new(W,window('pacman',size(525,625))),

send(W,open),

send(W,background,colour(black)),

send(W,display,new(\_,bitmap('title.bmp')),point(125,100)),

new(B1, button(exit,message(@prolog, close\_program,W))),

send(W,display,B1,point(225,325)),

new(B2, button(easy,message(@prolog, easy,W))),

send(W,display,B2,point(225,175)),

new(B3, button(medium,message(@prolog,medium,W))),

send(W,display,B3,point(225,225)),

new(B4, button(hard,message(@prolog, hard,W))),

send(W,display,B4,point(225,275)).

start:-

retractall(difficulty(\_)),

new(W,window('pacman',size(525,625))),

send(W,open),

send(W,background,colour(black)),

send(W,display,new(\_,bitmap('title.bmp')),point(125,100)),

new(B1, button(exit,message(@prolog, close\_program,W))),

send(W,display,B1,point(225,325)),

new(B2, button(easy,message(@prolog, easy,W))),

send(W,display,B2,point(225,175)),

new(B3, button(medium,message(@prolog,medium,W))),

send(W,display,B3,point(225,225)),

new(B4, button(hard,message(@prolog, hard,W))),

send(W,display,B4,point(225,275)).

close\_program(W):-

free(@window),

free(W).

close\_program(W):-

free(W).

easy(W):-

assert(difficulty(easy)),

free(W),

game(1).

medium(W):-

assert(difficulty(medium)),

free(W),

game(1).

hard(W):-

assert(difficulty(hard)),

free(W),

game(1).

close:-

assert(winner(ghost)).

game(X):-

(object( @window),!,free(@window);true),

retractall(winner(\_)),

retractall(point(\_)),

new(@window,window('pacman',size(525,625))),

send(@window,open),

level(X).

level(X):-

send(@window,clear),

retractall,

board\_map1(Map),

send(@window,display,new(\_,box(525,100)),point(0,525)),

new(B,bitmap('pacman\_right.bmp')),

send(@window,display,B,point(250,275)),

new(B1, button(quit,message(@prolog, close))),

send(@window,display,B1,point(2,600)),

assert(mouse(B)),

bots(X,L),

sleep(2),

assert(game\_mode(normal,0)),

assert(order(right)),

assert(user\_level(X)),

assert(point(0)),

send(@window, display,text('turn : '), point(2, 527)),

send(@window, display,text('points : '), point(2, 547)),

turn((\_,0,\_,\_),Map,L).

bots(0,[]).

bots(N,List):-

new( B,bitmap('ghost\_eating.bmp')), %creates cat

send( @window,display,B,point(250,150)), %shows Object

N1 is N-1,

bots(N1,L),

append([B],L,List).

anilist([],\_,1).

anilist([],L,Frames):-

Delay is 1/150,

FramesLeft is Frames - 1,

new(B1,box(525,625)),

send(@window,display,B1,point(0,0)),

catch\_order(B1),

sleep(Delay),

free(B1),

anilist(L,L,FramesLeft).

anilist([(B,Step\_x,Step\_y)|T],L,Frames):-

get(B, x,Cx),

get(B, y,Cy),

send( B,x(Cx+Step\_x)), %moves box to the right or left

send( B,y(Cy+Step\_y)), %moves box to the down or up

send( @window,flush), %flushes

repeat,

anilist(T,L,Frames).

turn((Tp,N,T1,D),Map,Lbots):-

not((winner(\_))),

player\_move(Map,D,X5,Y5,Dnew),

check\_mode,

bot\_move(Map,Lbots,(X5,Y5),L1),

check\_wincat(L1,(X5,Y5),L1,NewL1,NewLbots),

changemouse(X5,Y5),

mouse(B1),

append(NewL1,[(B1,X5,Y5)],L),

check\_winmouse(NewLbots),

findall((B,Step\_x,Step\_y),(member((B,X,Y),L),Step\_x is ((X)/25),Step\_y is ((Y)/25)),List),

anilist(List,List,25),

free(T1),

new(T, text(N)),N1 is N+1,

send(@window, display,T, point(32, 527)),

point(P),

check\_point(P,P1),

new(Tp1, text(P1)),free(Tp),

send(@window, display,Tp1, point(47, 547)),

retractall(point(\_)),

assert(point(P1)),

change\_ghost(NewLbots,NewLbots1),

turn((Tp1,N1,T,Dnew),Map,NewLbots1).

turn(\_,\_,\_):-

winner(ghost),

retractall(winner(\_)),

free(@window),

show\_gameover.

turn(\_,\_,\_):-

winner(pacman),

send(@window,clear),

show\_good\_feedback.

change\_ghost([],[]).

change\_ghost([B|T1],NewLbots):-

game\_mode(normal,X),X=<1,

new(B1,bitmap('ghost\_eating.bmp')),

get(B,x,Cx),

get(B,y,Cy),

send(@window,display,B1,point(Cx,Cy)),

free(B),

change\_ghost(T1,OldLbots),

append([B1],OldLbots,NewLbots).

change\_ghost([B|T1],NewLbots):-

game\_mode(normal,X),X>1,

new(B1,bitmap('ghost\_eatable.bmp')),

get(B,x,Cx),

get(B,y,Cy),

send(@window,display,B1,point(Cx,Cy)),

free(B),

change\_ghost(T1,OldLbots),

append([B1],OldLbots,NewLbots).

changemouse(0,0).

changemouse(X5,Y5):-

mouse(B),

retractall(mouse(B)),

move(D,X5,Y5),

term\_string(D,S),

string\_concat('pacman\_',S,Ns1),

string\_concat(Ns1,'.bmp',Ns),

new(B1,bitmap(Ns)),

get(B,x,Cx),

get(B,y,Cy),

send(@window,display,B1,point(Cx,Cy)),

free(B),

assert(mouse(B1)).

bot\_move(Map,Lbots,\_,L1):-

difficulty(easy),

random\_move(Map,Lbots,L1).

bot\_move(Map,Lbots,(X5,Y5),L1):-

smart\_move(Map,Lbots,(X5,Y5),L1).

/\* random move \*/

random\_move(\_,[],[]).

random\_move(M,[B|T],List):-

findall((D,X,Y),(move(D,X,Y),possible(M,B,D)),L),

random\_member((D1,X1,Y1),L),

change\_place(B,D1),

random\_move(M,T,L1),

append([(B,X1,Y1)],L1,List).

/\* smart move \*/

smart\_move(M,L,(X5,Y5),List\_of\_Moves):-

findall((B3,Cx1,Cy1),(member(B3,L),get(B3,x,Cx1),get(B3,y,Cy1)),B),

position\_of\_mouse((X5,Y5),(Cx\_Mouse,Cy\_Mouse)),

get\_chosen\_board(M,(Cx\_Mouse,Cy\_Mouse),B,The\_Board),

find\_moves(The\_Board,List\_of\_Moves),

change\_place\_for\_bots(List\_of\_Moves).

get\_chosen\_board(M,(Cx\_Mouse,Cy\_Mouse),CurrentBoard,The\_Board):-

difficulty(medium),

target\_for\_bots(M,(Cx\_Mouse,Cy\_Mouse),Final\_List),

change\_board\_term\_to\_string(CurrentBoard,CurrentBoard1),

append([(pacman,Cx\_Mouse,Cy\_Mouse)],CurrentBoard1,CurrentBoard2),

mini\_max(CurrentBoard2,M,Final\_List,CurrentBoard2,(The\_Board1,\_),1,max,(-1000000,1000000)),

delete(The\_Board1,(pacman,\_,\_),Board),

get\_new\_board(CurrentBoard,Board,The\_Board).

get\_chosen\_board(M,(Cx\_Mouse,Cy\_Mouse),CurrentBoard,The\_Board):-

difficulty(hard),

target\_for\_bots(M,(Cx\_Mouse,Cy\_Mouse),Final\_List),

change\_board\_term\_to\_string(CurrentBoard,CurrentBoard1),

append([(pacman,Cx\_Mouse,Cy\_Mouse)],CurrentBoard1,CurrentBoard2),

mini\_max(CurrentBoard2,M,Final\_List,CurrentBoard2,(The\_Board1,\_),3,max,(-1000000,1000000)),

delete(The\_Board1,(pacman,\_,\_),Board),

get\_new\_board(CurrentBoard,Board,The\_Board).

get\_new\_board([],\_,[]).

get\_new\_board([(B,\_,\_)|T],L,The\_Board):-

term\_string(B,S),

member((B1,X,Y),L),

term\_string(B1,S1),

string\_concat('@',S1,S),

get\_new\_board(T,L,Old\_Board),

append([(B,X,Y)],Old\_Board,The\_Board).

change\_board\_term\_to\_string([],[]).

change\_board\_term\_to\_string([(B,P1,P2)|T],[(B1,P1,P2)|T1]):-

term\_string(B,S),

string\_concat('@',S1,S),

term\_string(B1,S1),

change\_board\_term\_to\_string(T,T1).

mini\_max(\_,\_,Final\_List,B,(B,P),0,\_,\_):-

board\_points(B,Final\_List,P).

mini\_max(\_,M,Final\_List,CurrentBoard,(Board,P),X,max,(Alpha,Beta)):-

dif(X,0),

find\_possible\_boards\_for\_board(M,CurrentBoard,List\_of\_Boards),

X1 is X-1,

find\_minimax\_for\_list(List\_of\_Boards,List,CurrentBoard,M,Final\_List,X1,min,(-1000000,Alpha,Beta)),

max\_member((P,Board),List).

mini\_max(\_,M,Final\_List,CurrentBoard,(Board,P),X,min,(Alpha,Beta)):-

dif(X,0),

find\_possible\_boards\_for\_pacman(M,CurrentBoard,List),

X1 is X-1,

find\_minimax\_for\_list(List,List1,CurrentBoard,M,Final\_List,X1,max,(1000000,Alpha,Beta)),

min\_member((P,Board),List1).

find\_minimax\_for\_list([],[],\_,\_,\_,\_,\_,\_).

find\_minimax\_for\_list(\_,[],\_,\_,\_,\_,min,(Value,\_,Beta)):-

Value>Beta.

find\_minimax\_for\_list(\_,[],\_,\_,\_,\_,max,(Value,Alpha,\_)):-

Value<Alpha.

find\_minimax\_for\_list([B|T],L,CurrentBoard,M,Final\_List,X1,max,(Value,Alpha,Beta)):-

edit\_board(B,B2),

edit(B2,CurrentBoard,B3),

mini\_max(CurrentBoard,M,Final\_List,B3,(\_,V1),X1,max,(Alpha,Beta)),

min\_member(V2,[Value,V1]),

min\_member(NewBeta,[Beta,V1]),

find\_minimax\_for\_list(T,L1,CurrentBoard,M,Final\_List,X1,max,(V2,Alpha,NewBeta)),

append([(V1,B3)],L1,L).

find\_minimax\_for\_list([B|T],L,CurrentBoard,M,Final\_List,X1,min,(Value,Alpha,Beta)):-

edit\_board(B,B2),

edit(B2,CurrentBoard,B3),

mini\_max(CurrentBoard,M,Final\_List,B3,(\_,V1),X1,min,(Alpha,Beta)),

max\_member(V2,[Value,V1]),

max\_member(NewAlpha,[Alpha,V1]),

find\_minimax\_for\_list(T,L1,CurrentBoard,M,Final\_List,X1,min,(V2,NewAlpha,Beta)),

append([(V1,B3)],L1,L).

edit\_board(B,B2):-

member((pacman,X,Y),B),

delete(B,(pacman,\_,\_),B1),

append(B1,[(pacman,X,Y)],B2).

edit\_board(B,B):-

not((member((pacman,\_,\_),B))).

edit(L,\_,L):-

not((member((pacman,\_,\_),L))).

edit([(pacman,X,Y)],\_,[(pacman,X,Y)]).

edit([(B,X,Y)|T],O,NewBoard2):-

game\_mode(normal,0),

dif(B,pacman),

member((pacman,X,Y),T),

edit(T,O,NewBoard),

delete(NewBoard,(pacman,\_,\_),NewBoard1),

append([(B,X,Y)],NewBoard1,NewBoard2).

edit([(B,X,Y)|T],OldBoard,NewBoard2):-

game\_mode(normal,0),

dif(B,pacman),

member((pacman,X1,Y1),T),

member((B,X1,Y1),OldBoard),

member((pacman,X,Y),OldBoard),

edit(T,OldBoard,NewBoard),

delete(NewBoard,(pacman,\_,\_),NewBoard1),

append([(B,X,Y)],NewBoard1,NewBoard2).

edit([(B,X,Y)|T],OldBoard,NewBoard1):-

game\_mode(normal,N),N>0,

dif(B,pacman),

member((pacman,X,Y),T),

edit(T,OldBoard,NewBoard1).

edit([(B,X,Y)|T],OldBoard,NewBoard1):-

game\_mode(normal,N),N>0,

dif(B,pacman),

member((pacman,X1,Y1),T),

member((B,X1,Y1),OldBoard),

member((pacman,X,Y),OldBoard),

edit(T,OldBoard,NewBoard1).

edit([(B,X,Y)|T],O,NewBoard):-

not((dif(B,pacman),member((pacman,X,Y),T))),

not((dif(B,pacman),member((pacman,X1,Y1),T),member((B,X1,Y1),O),

member((pacman,X,Y),O))),

edit(T,O,NewBoard1),

append([(B,X,Y)],NewBoard1,NewBoard).

find\_possible\_boards\_for\_pacman(\_,Board,[Board]):-

not((member((pacman,\_,\_),Board))).

find\_possible\_boards\_for\_pacman(M,Board,SBoard2):-

member((pacman,Cx,Cy),Board),

findall((pacman,(Cx,Cy),(X1,Y1)),(

(move(D,Move\_X,Move\_Y),possible(M,Cx,Cy,D),final\_position((Cx,Cy),(Move\_X,Move\_Y),D,X1,Y1)))

,List1),

delete(Board,(pacman,\_,\_),Rest),

findall(F,(get\_board(List1,Board1),append(Rest,Board1,F)),Board2),

sort(Board2,SBoard2).

find\_possible\_boards\_for\_board(\_,Board,[Board]):-

not((member((pacman,\_,\_),Board))).

find\_possible\_boards\_for\_board(\_,Board,[Board]):-

member((pacman,\_,\_),Board),

delete(Board,(pacman,\_,\_),[]).

find\_possible\_boards\_for\_board(M,Board,SBoard2):-

member((pacman,X,Y),Board),

findall((B1,(Cx,Cy),(X1,Y1)),(

member((B1,Cx,Cy),Board),dif(B1,pacman),(move(D,Move\_X,Move\_Y),possible(M,Cx,Cy,D),final\_position((Cx,Cy),(Move\_X,Move\_Y),D,X1,Y1)))

,List),

findall(F,(get\_board(List,Board1),append([(pacman,X,Y)],Board1,F)),Board2),

sort(Board2,SBoard2).

possible(\_,X,Y,left):-

X is 25,

Y is 225.

possible(\_,X,Y,right):-

X is 475,

Y is 225.

possible(Map,X,Y,Direction):-

not((compare(=,X,25),compare(=,Y,225),compare(=,Direction,left))),

not((compare(=,X,475),compare(=,Y,225),compare(=,Direction,right))),

move(Direction,X1,Y1),

Fx is (X+X1),Fy is (Y+Y1),

not((member((blank,Fx,Fy),Map))).

get\_board([],[]).

get\_board(L,Board):-

member((B,(0,225),(475,225)),L),

delete(L,(B,\_,\_),L1),

get\_board(L1,OldBoard),

append(OldBoard,[(B,475,225)],Board).

get\_board(L,Board):-

member((B,(500,225),(25,225)),L),

delete(L,(B,\_,\_),L1),

get\_board(L1,OldBoard),

append(OldBoard,[(B,25,225)],Board).

get\_board(L,Board):-

member((B,\_,(X,Y)),L),

delete(L,(B,\_,\_),L1),

get\_board(L1,OldBoard),

append(OldBoard,[(B,X,Y)],Board).

final\_position((X,Y),(0,0),\_,X,Y).

final\_position((X,Y),(Move\_x,Move\_y),Direction,X1,Y1):-

move(Direction,Move\_x,Move\_y),X is 475,Y is 225,compare(=,Direction,right),X1 is 25,Y1 is 225/\*,writeln('c1')\*/.

final\_position((X,Y),(Move\_x,Move\_y),Direction,X1,Y1):-

move(Direction,Move\_x,Move\_y),X is 25,Y is 225,compare(=,Direction,left),X1 is 475,Y1 is 225/\*,writeln('c2')\*/.

final\_position((Cx,Cy),(Move\_X,Move\_Y),Direction,X1,Y1):-

move(Direction,Move\_X,Move\_Y),not((compare(=,X,25),compare(=,Y,225),compare(=,Direction,left))),not((compare(=,X,475),compare(=,Y,225),compare(=,Direction,right))),X1 is (Cx+Move\_X),Y1 is (Cy+Move\_Y).

/\* the bait \*/

target\_for\_bots(Map,(Cx,Cy),List):-

difficulty(hard),

game\_mode(normal,X),X>0,

findall((X1,Y1),(move(D,Move\_X,Move\_Y),final\_position((Cx,Cy),(Move\_X,Move\_Y),D,X1,Y1),not((member((blank,X1,Y1),Map)))),Final\_List),

findall\_closer((Cx,Cy),Final\_List,List).

/\* objective gaming \*/

target\_for\_bots(\_,\_,Final\_List):-

difficulty(hard),

findall((X1,Y1),(point(X1,Y1,\_)),L1),

length(L1,Length),Length=<10,append([],L1,Final\_List).

/\* the one way hallway\*/

target\_for\_bots(\_,(Cx,225),List):-

( ( between(0,225,Cx),T is 1);(between(375,500,Cx),T is 1);(between(225,375,Cx),T is 2);true),

compare(=,T,1),dif(T,2),

append([],[(100,225),(400,225)],Final\_List),

findall\_closer((Cx,225),Final\_List,List).

/\* mid strategy\*/

target\_for\_bots(\_,(Cx,Cy),[]):-

between(150,350,Cx),between(100,350,Cy).

/\* the corner trapper\*/

target\_for\_bots(Map,(Cx,Cy),List):-

( (Cx<250,T1 is 1);(Cx>=250,T1 is 2)),

find\_list\_of\_line\_x(Map,Cx,T1,Lx),

( (Cy<225,T2 is 1);(Cy>=225,T2 is 2)),

find\_list\_of\_line\_y(Map,Cy,T2,Ly),

findall(N1,(member(L1,Lx),length(L1,N1)),Lx\_of\_N),

findall(N2,(member(L2,Ly),length(L2,N2)),Ly\_of\_N),

min\_list(Lx\_of\_N,Nx),

min\_list(Ly\_of\_N,Ny),

findall(L3,(member(L3,Lx),length(L3,Nx)),Final\_Lx),

findall(L4,(member(L4,Ly),length(L4,Ny)),Final\_Ly),

get\_final\_spots((Cx,Cy),Final\_Lx,Final\_Ly,Final\_List),

findall\_closer((Cx,Cy),Final\_List,List).

get\_final\_spots((Cx,Cy),Final\_Lx,Final\_Ly,Final\_List):-

findall(X1,member([(X1,\_)|\_],Final\_Lx),List\_of\_X),

findall(Y1,member([(\_,Y1)|\_],Final\_Ly),List\_of\_Y),

( (Cx<250,max\_list(List\_of\_X,X2));(Cx>=250,min\_list(List\_of\_X,X2))),

( (Cy<225,max\_list(List\_of\_Y,Y3));(Cy>=225,min\_list(List\_of\_Y,Y3))),

member([(X2,Y2)|T1],Final\_Lx),member([(X3,Y3)|T2],Final\_Ly),

append([(X2,Y2)],T1,Line\_x),append([(X3,Y3)],T2,Line\_y),

findall((X4,Y4),(member((X4,Y4),Line\_x),( (Cy<225,Y4=<Y3);(Cy>=225,Y4>=Y3))),Final\_Line\_X),

findall((X5,Y5),(member((X5,Y5),Line\_y),( (Cx<250,X5=<X2);(Cx>=250,X5>=X2))),Final\_Line\_Y),

append(Final\_Line\_X,Final\_Line\_Y,Final\_List0),

sort(Final\_List0,Final\_List).

find\_list\_of\_line\_x(Map,Cx,1,L2):-

findall((X1,Y1),(member((\_,X1,Y1),Map),not((member((blank,X1,Y1),Map))),X1>=Cx,X1=<250),L1),

findall(X5,member((X5,\_),L1),L5),

sort(L5,L3),

findall(L4,(member(X2,L3),findall((X2,Y2),(member((X2,Y2),L1)),L4)),L2).

find\_list\_of\_line\_x(Map,Cx,2,L2):-

findall((X1,Y1),(member((\_,X1,Y1),Map),not((member((blank,X1,Y1),Map))),X1=<Cx,X1>=250),L1),

findall(X5,member((X5,\_),L1),L5),

sort(L5,L3),

findall(L4,(member(X2,L3),findall((X2,Y2),(member((X2,Y2),L1)),L4)),L2).

find\_list\_of\_line\_y(Map,Cy,1,L2):-

findall((X1,Y1),(member((\_,X1,Y1),Map),not((member((blank,X1,Y1),Map))),Y1>=Cy,Y1=<225),L1),

findall(Y5,member((\_,Y5),L1),L5),

sort(L5,L3),

findall(L4,(member(Y2,L3),findall((X2,Y2),(member((X2,Y2),L1)),L4)),L2).

find\_list\_of\_line\_y(Map,Cy,2,L2):-

findall((X1,Y1),(member((\_,X1,Y1),Map),not((member((blank,X1,Y1),Map))),Y1=<Cy,Y1>=225),L1),

findall(Y5,member((\_,Y5),L1),L5),

sort(L5,L3),

findall(L4,(member(Y2,L3),findall((X2,Y2),(member((X2,Y2),L1)),L4)),L2).

find\_moves([],[]).

find\_moves([(B,X,Y)|T],L):-

get(B,x,Cx),

get(B,y,Cy),

final\_position((Cx,Cy),(Move\_x,Move\_y),\_,X,Y),

find\_moves(T,L1),

append(L1,[(B,Move\_x,Move\_y)],L).

board\_points(Board,Final\_List,Points):-

get\_distance\_points(Final\_List,Board,P1),

massive\_points(Board,P2),

Points is ((P1\*10)+P2).

massive\_points(Board,P2):-

not((member((pacman,\_,\_),Board))),

P2 is 8000.

massive\_points(Board,P2):-

difficulty(hard),

delete(Board,(pacman,\_,\_),F),

length(F,L),

user\_level(L1),

L<L1,

P2 is -8000.

massive\_points(\_,0).

get\_distance\_points(\_,Board,0.8):-

not((member((pacman,\_,\_),Board))).

get\_distance\_points(Final\_List,Board,P):-

findall((X,Y),(member((B,X,Y),Board),dif(B,pacman)),Places),

member((pacman,Cx,Cy),Board),

get\_distance\_points(Final\_List,Places,(Cx,Cy),P).

get\_distance\_points(\_,[],\_,0).

get\_distance\_points([],[Place|T],(Cx,Cy),P):-

distance\_between\_place\_and\_object((Cx,Cy),Place,D),

FD is D+1,delete(T,Place,NewBoard),P1 is (1/FD),

get\_distance\_points([],NewBoard,(Cx,Cy),OldP),

P is OldP+P1.

get\_distance\_points([(Cx,Cy)|T],Board,(Cx1,Cy1),P1):-

findall(D1,(member((X1,Y1),Board),distance\_between\_place\_and\_object((Cx,Cy),(X1,Y1),D1)),List\_of\_Distances),

min\_list(List\_of\_Distances,Min\_D),

member(Place,Board),distance\_between\_place\_and\_object((Cx,Cy),Place,Min\_D),FMin\_D is (Min\_D+1),

delete(Board,Place,NewBoard),delete(T,(Cx,Cy),T1),

Points is (100/FMin\_D),

get\_distance\_points(T1,NewBoard,(Cx1,Cy1),OldP),

P1 is OldP+Points.

distance\_between\_place\_and\_object((Cx,Cy),(Cx1,Cy1),Distance):-

X\_d1 is abs(Cx1-Cx),Y\_d1 is abs(Cy1-Cy),

pitaguras(X\_d1,Y\_d1,Distance1),

X\_d2\_1 is abs(Cx1),Y\_d2\_1 is abs(Cy1-225),X\_d2\_2 is abs(Cx-500),Y\_d2\_2 is abs(Cy-225),

pitaguras(X\_d2\_1,Y\_d2\_1,Distance2\_1),pitaguras(X\_d2\_2,Y\_d2\_2,Distance2\_2),sum\_list([Distance2\_1,Distance2\_2],Distance2),

X\_d3\_1 is abs(Cx),Y\_d3\_1 is abs(Cy-225),X\_d3\_2 is abs(Cx1-500),Y\_d3\_2 is abs(Cy1-225),

pitaguras(X\_d3\_1,Y\_d3\_1,Distance3\_1),pitaguras(X\_d3\_2,Y\_d3\_2,Distance3\_2),sum\_list([Distance3\_1,Distance3\_2],Distance3),

min\_list([Distance1,Distance2,Distance3],Distance).

pitaguras(X,Y,Result):-

X1 is (X\*X),Y1 is (Y\*Y),Result is sqrt(X1+Y1).

findall\_closer(\_,[],[]).

findall\_closer((Cx,Cy),Final\_List,List):-

findall(Distance1,(member((Cx11,Cy11),Final\_List),distance\_between\_place\_and\_object((Cx,Cy),(Cx11,Cy11),Distance1)),L1),

min\_list(L1,Min),

findall((Distance,Cx1,Cy1),(member((Cx1,Cy1),Final\_List),distance\_between\_place\_and\_object((Cx,Cy),(Cx1,Cy1),Distance)),L2),

member((Min,X,Y),L2),

delete(Final\_List,(X,Y),L3),

findall\_closer((Cx,Cy),L3,OldList),

append([(X,Y)],OldList,List).

position\_of\_mouse((X5,Y5),(X,Y)):-

mouse(B),

get(B,x,Cx),

get(B,y,Cy),

X is (Cx+X5),Y is (Cy+Y5).

/\* end of smart move \*/

player\_move(M,\_,X,Y,D):-

order(D),

mouse(B),

possible(M,B,D),

change\_place(B,D),

move(D,X,Y).

player\_move(M,Direction,X,Y,Direction):-

mouse(B),

possible(M,B,Direction),

change\_place(B,Direction),

move(Direction,X,Y).

player\_move(\_,D,0,0,D).

catch\_order(B):-

new(Kw,key\_binding('w')),

send(Kw,function,'w',message(@prolog, change\_o, up)),

send(B,recogniser,Kw),

new(Kw1,key\_binding('cursor\_up')),

send(Kw1,function,'cursor\_up',message(@prolog, change\_o, up)),

send(B,recogniser,Kw1),

new(Ks,key\_binding('s')),

send(Ks,function,'s',message(@prolog, change\_o,down)),

send(B,recogniser,Ks),

new(Ks1,key\_binding('cursor\_down')),

send(Ks1,function,'cursor\_down',message(@prolog, change\_o, down)),

send(B,recogniser,Ks1),

new(Kd,key\_binding('d')),

send(Kd,function,'d',message(@prolog, change\_o,right)),

send(B,recogniser,Kd),

new(Kd1,key\_binding('cursor\_right')),

send(Kd1,function,'cursor\_up',message(@prolog, change\_o, right)),

send(B,recogniser,Kd1),

new(Ka,key\_binding('a')),

send(Ka,function,'a',message(@prolog, change\_o,left)),

send(B,recogniser,Ka),

new(Ka1,key\_binding('cursor\_left')),

send(Ka1,function,'cursor\_left',message(@prolog, change\_o, left)),

send(B,recogniser,Ka1).

move(left,-25,0).

move(right,25,0).

move(down,0,25).

move(up,0,-25).

possible(\_,B,left):-

get(B,x,X),X is 25,

get(B,y,Y),Y is 225.

possible(\_,B,right):-

get(B,x,X),X is 475,

get(B,y,Y),Y is 225.

possible(Map,B,Direction):-

get(B,x,X),

get(B,y,Y),

not((compare(=,X,25),compare(=,Y,225),compare(=,Direction,left))),not((compare(=,X,475),compare(=,Y,225),compare(=,Direction,right))),

move(Direction,X1,Y1),

Fx is (X+X1),Fy is (Y+Y1),

not((member((blank,Fx,Fy),Map))).

change\_o(D):-

retractall(order(\_)),

assert(order(D)).

check\_point(P,NP):-

mouse(B1),

get(B1,x,X),

get(B1,y,Y),

point(X,Y,\_),

findall(B,point(X,Y,B),L),

delete\_all(L),

NP is P+1.

check\_point(P,P):-

mouse(B1),

get(B1,x,X),

get(B1,y,Y),

spoint(X,Y,\_),

findall(B,spoint(X,Y,B),L),

delete\_all(L),

retractall(game\_mode(\_,0)),

assert(game\_mode(normal,20)).

check\_point(P,P).

delete\_all([]).

delete\_all([B|T]):-

retractall(point(\_,\_,B)),

retractall(spoint(\_,\_,B)),

free(B),

delete\_all(T).

check\_wincat([],\_,L1,L1,NewLbots):-

findall(B,member((B,\_,\_),L1),NewLbots).

check\_wincat([(H,\_,\_)|\_],\_,\_,\_,\_):-

game\_mode(normal,0),

get(H, x,Cx1),

get(H, y,Cy1),

mouse(B),

get(B, x,Cx),

get(B, y,Cy),

Cx1 is Cx, Cy1 is Cy,

assert(winner(ghost)).

check\_wincat([(H,X,Y)|\_],(X5,Y5),\_,\_,\_):-

game\_mode(normal,0),

get(H, x,Cx1),

get(H, y,Cy1),

mouse(B),

get(B, x,Cx),

get(B, y,Cy),

Cx1 is (Cx+X5), Cy1 is (Cy+Y5),

Cx is (Cx1+X), Cy is (Cy1+Y),

assert(winner(ghost)).

check\_wincat([(H,\_,\_)|\_],\_,L1,NewL1,NewLbots):-

game\_mode(normal,\_),

get(H, x,Cx1),

get(H, y,Cy1),

mouse(B),

get(B, x,Cx),

get(B, y,Cy),

Cx1 is Cx, Cy1 is Cy,

delete(L1,(H,\_,\_),NewL1),

findall(B1,(member((B1,\_,\_),NewL1),dif(H,B1)),NewLbots),

free(H).

check\_wincat([(H,X,Y)|\_],(X5,Y5),L1,NewL1,NewLbots):-

game\_mode(normal,\_),

get(H, x,Cx1),

get(H, y,Cy1),

mouse(B),

get(B, x,Cx),

get(B, y,Cy),

Cx1 is (Cx+X5), Cy1 is (Cy+Y5),

Cx is (Cx1+X), Cy is (Cy1+Y),

delete(L1,(H,\_,\_),NewL1),

findall(B1,(member((B1,\_,\_),NewL1),dif(H,B1)),NewLbots),

free(H).

check\_wincat([\_|T],(X5,Y5),L1,NewL1,NewLbots):-

check\_wincat(T,(X5,Y5),L1,NewL1,NewLbots).

check\_winmouse(\_):-

findall(B,point(\_,\_,B),[]),

assert(winner(pacman)).

check\_winmouse([]):-

assert(winner(pacman)).

check\_winmouse(\_).

check\_mode:-

game\_mode(normal,0).

check\_mode:-

game\_mode(normal,N),

retractall(game\_mode(normal,N)),

N1 is N-1,

assert(game\_mode(normal,N1)).

retractall:-

retractall(order(\_)),

retractall(winner(\_)),

retractall(mouse(\_)),

retractall(game\_mode(\_,\_)),

retractall(user\_level(\_)),

retractall(point(\_,\_,\_)),

retractall(spoint(\_,\_,\_)).

change\_place(B,left):-

get(B,x,X),X is 25,

get(B,y,Y),Y is 225,

send(B,x(500)).

change\_place(B,right):-

get(B,x,X),X is 475,

get(B,y,Y),Y is 225,

send(B,x(0)).

change\_place(\_,\_).

change\_place\_for\_bots([]).

change\_place\_for\_bots([(B,X,Y)|T]):-

move(D,X,Y),

change\_place(B,D),

change\_place\_for\_bots(T).

/\* from here its only on the map/board \*/

board\_map1(L):-

places\_map1(L),

display\_map1.

places\_map1(L):-

places\_map1([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],L).

places\_map1([],\_,\_,[]).

places\_map1([\_|T],[],L,List):-

places\_map1(T,L,L,List).

/\* s\_points \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1 = 1,H2 is 1,

X is H1\*25,

Y is H2\*25,

X1 is X+5,Y1 is Y+5,

places\_map1([H1|T1],T2,L,List),

append([(spoint,X,Y)],List,NList),

new(B,circle(15)),

send(B,fill\_pattern,colour(red)),

send(@window,display,B,point(X1,Y1)),

assert(spoint(X,Y,B)).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1 is 19,H2 is 1,

X is H1\*25,

Y is H2\*25,

X1 is X+5,Y1 is Y+5,

places\_map1([H1|T1],T2,L,List),

append([(spoint,X,Y)],List,NList),

new(B,circle(15)),

send(B,fill\_pattern,colour(red)),

send(@window,display,B,point(X1,Y1)),

assert(spoint(X,Y,B)).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1 is 19,H2 is 13,

X is H1\*25,

Y is H2\*25,

X1 is X+5,Y1 is Y+5,

places\_map1([H1|T1],T2,L,List),

append([(spoint,X,Y)],List,NList),

new(B,circle(15)),

send(B,fill\_pattern,colour(red)),

send(@window,display,B,point(X1,Y1)),

assert(spoint(X,Y,B)).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1 is 1,H2 is 13,

X is H1\*25,

Y is H2\*25,

X1 is X+5,Y1 is Y+5,

places\_map1([H1|T1],T2,L,List),

append([(spoint,X,Y)],List,NList),

new(B,circle(15)),

send(B,fill\_pattern,colour(red)),

send(@window,display,B,point(X1,Y1)),

assert(spoint(X,Y,B)).

/\* special blank boxes \*/

/\* 0 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=2,H1=<5,

H2>=2,H2=<4,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=7,H1=<8,

H2>=2,H2=<4,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=10,H1=<10,

H2>=1,H2=<4,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=15,H1=<18,

H2>=2,H2=<4,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=12,H1=<13,

H2>=2,H2=<4,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 1 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=5,H1=<5,

H2>=10,H2=<12,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=3,H1=<3,

H2>=12,H2=<14,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=15,H1=<15,

H2>=10,H2=<12,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=17,H1=<17,

H2>=12,H2=<14,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 2 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=1,H1=<3,

H2>=8,H2=<8,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=17,H1=<19,

H2>=8,H2=<8,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=1,H1=<3,

H2>=10,H2=<10,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=17,H1=<19,

H2>=10,H2=<10,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 3 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=5,H1=<5,

H2>=7,H2=<8,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=5,H1=<5,

H2>=14,H2=<15,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=15,H1=<15,

H2>=7,H2=<8,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=15,H1=<15,

H2>=14,H2=<15,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=10,H1=<10,

H2>=13,H2=<14,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 4 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=2,H1=<8,

H2>=6,H2=<6,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=12,H1=<18,

H2>=6,H2=<6,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=7,H1=<13,

H2>=12,H2=<12,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 5 \*/

places\_map1([1|T1],[14|T2],L,NList):-

X is 1\*25,

Y is 14\*25,

places\_map1([1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=2,H1=<2,

H2>=12,H2=<12,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=18,H1=<18,

H2>=12,H2=<12,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=19,H1=<19,

H2>=14,H2=<14,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 6 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=7,H1=<8,

H2>=14,H2=<14,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=12,H1=<13,

H2>=14,H2=<14,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 7 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=2,H1=<9,

H2>=16,H2=<18,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=11,H1=<18,

H2>=16,H2=<18,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* 8 \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=7,H1=<13,

H2>=8,H2=<10,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* borders \*/

places\_map1([0|T1],[H2|T2],L,NList):-

X is 0,

Y is H2\*25,

places\_map1([0|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[0|T2],L,NList):-

X is H1\*25,

Y is 0,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([20|T1],[H2|T2],L,NList):-

X is 500,

Y is H2\*25,

places\_map1([20|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

places\_map1([H1|T1],[20|T2],L,NList):-

X is H1\*25,

Y is 500,

places\_map1([H1|T1],T2,L,List),

append([(blank,X,Y)],List,NList).

/\* valid ways without points \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=9,H1=<11,

H2>=6,H2=<6,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(nopoint,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=6,H1=<14,

H2>=7,H2=<7,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(nopoint,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=6,H1=<14,

H2>=11,H2=<11,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(nopoint,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=6,H1=<6,

H2>=8,H2=<10,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(nopoint,X,Y)],List,NList).

places\_map1([H1|T1],[H2|T2],L,NList):-

H1>=14,H1=<14,

H2>=8,H2=<10,

X is H1\*25,

Y is H2\*25,

places\_map1([H1|T1],T2,L,List),

append([(nopoint,X,Y)],List,NList).

/\* valid ways with points \*/

places\_map1([H1|T1],[H2|T2],L,NList):-

X is H1\*25,

Y is H2\*25,

X1 is X+10,Y1 is Y+10,

places\_map1([H1|T1],T2,L,List),

append([(point,X,Y)],List,NList),

new(B,box(5,5)),

send(B,fill\_pattern,colour(yellow)),

send(@window,display,B,point(X1,Y1)),

assert(point(X,Y,B)).

display\_map1:-

/\* border \*/

box\_style(25,225,0,0),

box\_style(25,275,0,250),

box\_style(25,225,500,0),

box\_style(25,275,500,250),

box\_style(475,25,25,0),

box\_style(475,25,25,500),

/\* 0 \*/

box\_style(100,75,50,50),

box\_style(50,75,175,50),

box\_style(25,100,250,25),

box\_style(100,75,375,50),

box\_style(50,75,300,50),

/\* 1 \*/

box\_style(25,75,75,300),

box\_style(25,75,125,250),

box\_style(25,75,375,250),

box\_style(25,75,425,300),

/\* 2 \*/

box\_style(75,25,25,200),

box\_style(75,25,25,250),

box\_style(75,25,425,200),

box\_style(75,25,425,250),

/\* 3 \*/

box\_style(25,50,125,175),

box\_style(25,50,125,350),

box\_style(25,50,375,175),

box\_style(25,50,375,350),

box\_style(25,50,250,325),

/\* 4 \*/

box\_style(175,25,50,150),

box\_style(175,25,300,150),

box\_style(175,25,175,300),

/\* 5 \*/

box\_style(25,25,25,350),

box\_style(25,25,50,300),

box\_style(25,25,475,350),

box\_style(25,25,450,300),

/\* 6 \*/

box\_style(50,25,175,350),

box\_style(50,25,300,350),

/\* 7 \*/

box\_style(200,75,50,400),

box\_style(200,75,275,400),

/\* 8 \*/

box\_style(175,75,175,200)

.

box\_style(Size\_w,Size\_h,Point\_x,Point\_y):-

new(B,box(Size\_w,Size\_h)),

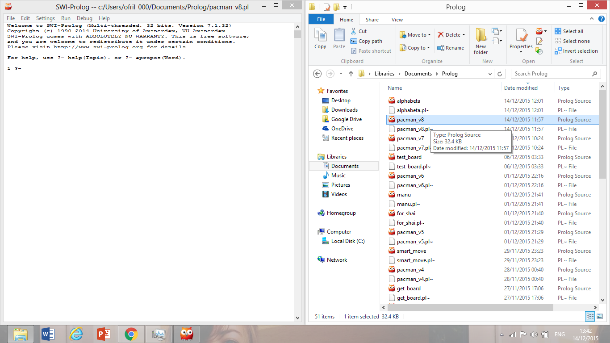
send(B,fill\_pattern,colour(black)),

send(@window,display,B,point(Point\_x,Point\_y)).

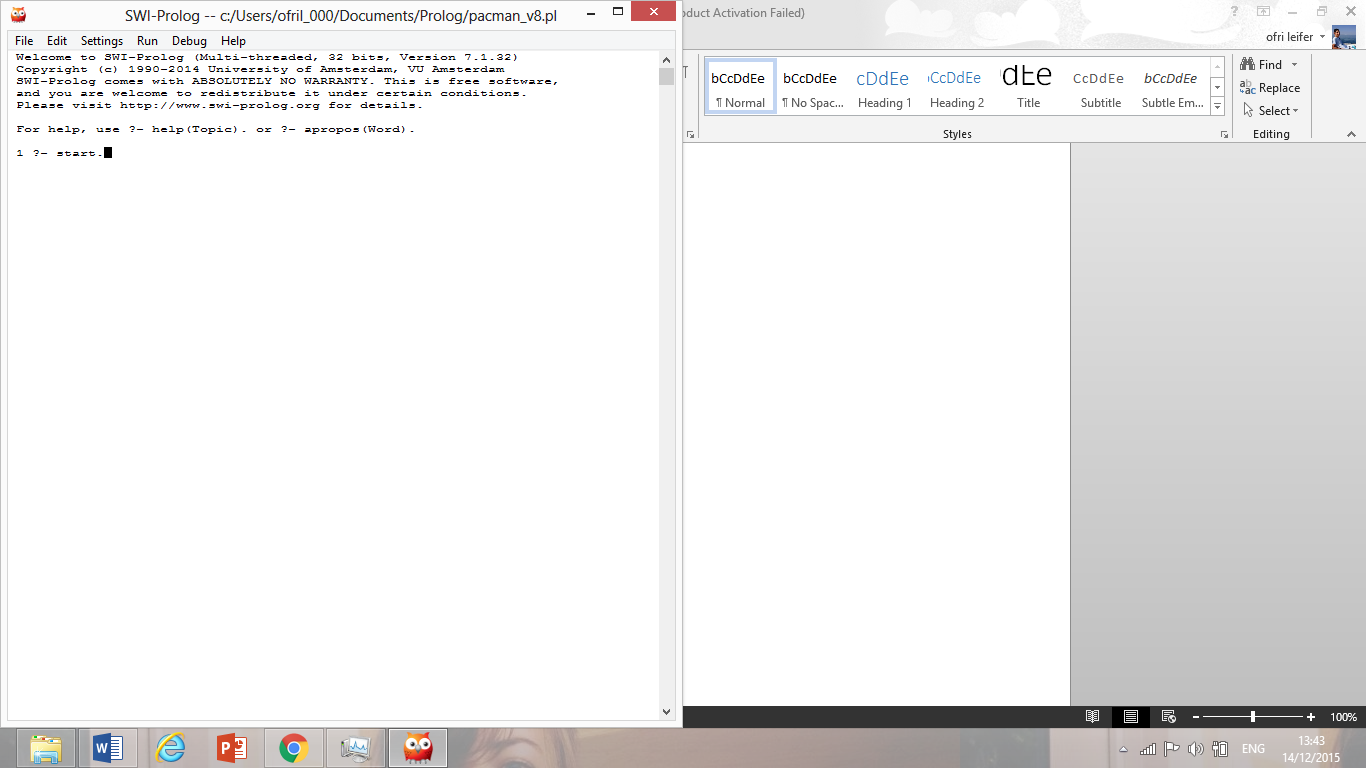
/\* The End \*/

**שימוש במערכת והרצתה**

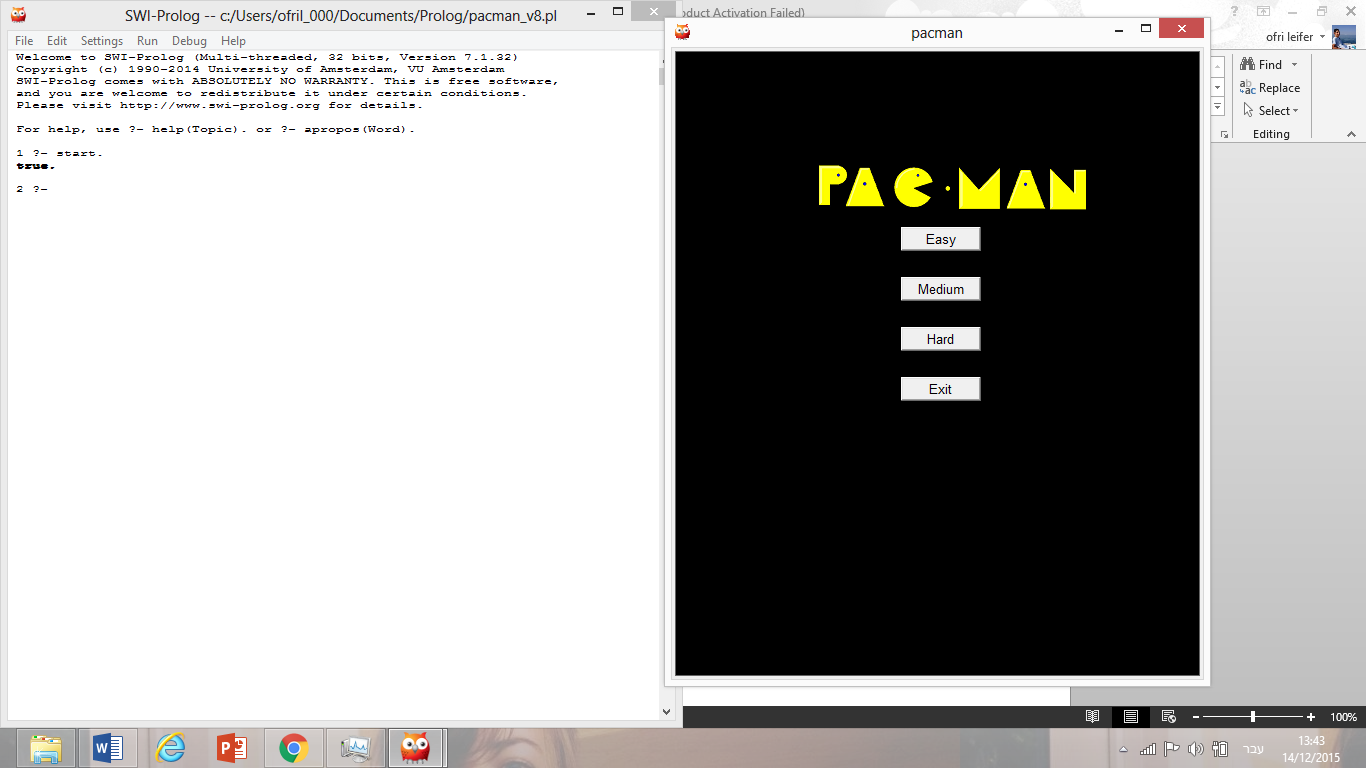
ראשית יש לטעון את הקובץ pacman\_v8.pl .



כעת יש להקליד start.



ללחוץ ENTER .

כעת נפתח מסך הכניסה, בו יש לך את האפשרות לבחור את רמת קושי הרצויה או לצאת מהמשחק.

לאחר בחירת רמת הקושי, המשחק מתחיל, יש לוודות שהעכבר נמצא על החלון אחרת לא תוכל לשלוט בפק מן. 

תהנה!

**הצעות לשיפור ולשכלול**

ישנן אפשרויות רבות לשכלול המשחק, הן מבחינת שיפור המשחק הקיים ומבחינת רעיונות למשחקים חדשים המבוססים על המשחק הקיים.

שיפור :

* גרפיקה : ליצור גרפיקה שמתאימה את עצמה כתלות בגודל המסך.

שכלול :

* מפות/מבוכים שונים
* יותר רמות קושי
* יותר סוגי נקודות (כמו במשחק המקורי כגון דובדבן).
* להוסיף את גברת פקמן
* להוסיף אנימציית רקע למסך פתיחה
* לשפר את מסך הניצחון וההפסד