תוכן עניינים

תיאור הפרוייקט

מבוא

דמקה הוא משחק לוח אסטרטגי לשני שחקנים. המשחק עתיק מאוד וקיים בצורות שונות ועם חוקים שונים אך צורתו הנפוצה ביותר בעולם המערבי היא הדמקה האנגלית ועליה מתבסס הפרוייקט. הלוח מחולק ל-64 משבצות שחורות ולבנות. כל שחקן מתחיל עם 12 אבנים (חיילים) כאשר השחורים מתחילים. השחקנים מזיזים בתורם את האבנים על גבי המשבצות ומטרת המשחק היא להביא למצב בו לשחקן השני לא נשארו עוד מהלכים חוקיים. חייל רגיל יכול להתקדם למשבצת הסמוכה במידה והיא ריקה או במידה וחייל אויב נמצא על המשבצת והמשבצת שאחריה היא ריקה, לקפוץ מעל החייל למשבצת ולהוריד את החייל מהלוח. כאשר חייל מגיע לשורה האחרונה ('שורת ההמלכה', השורה הראשונה של היריב) הוא "מוכתר" ומקבל את האפשרות לנוע ולאכול אחורה בתור הבא (הכתרה מסיימת את התור גם במצב של אכילה). השחקן אינו יכול לבצע מהלך רגיל אם באפשרותו לבצע מהלך אכילה.

הפרוייקט מציג את המשחק בסביבה תלת מימדית ומאפשר למשתמש לשחק מול המחשב בשלוש דרגות קושי או נגד שחקן אחר על אותו המחשב.

רכיבי התוכנית

- **ממשק:** שני תפריטים מתחלפים (ראשי ומשחק) בתוך חלון, מקשר בין המשתמש למשחק.
- **מנוע גרפי:** מציג את הלוח והאבנים.
- **מנהל פעולות:** מבצע את הפעולות במשחק.
- **רשימת מהלכים:** מחשבת את כל המהלכים והשלכותיהם המיידיות עבור לוח נתון ומחזירה את המהלך האופטימלי עבור צד נבחר.
- **משחק:** מקשר בין המנוע, המנהל והרשימה לממשק

שלבים בעבודה

- יצירת סקיצה של המנוע הגרפי בסביבה מוכרת (OPENGL/C++)
- למידת סביבת WPF
- שכתוב המנוע הגרפי לסביבה החדשה
- יצירת אובייקט המשחק
- חקירה של משחק לדוגמא
- יצירת שחקן מחשב
- יצירת הממשק והצלילים
- כתיבת ספר וביצוע תיקונים אחרונים

סביבת הפיתוח

סביבת הפיתוח שנבחרה היא ה-(Windows Presentation Foundation (WPF של מייקרוסופט. הסביבה מגיעה עם כל המרכיבים הדרושים לפרוייקט: מנוע תלת מימד מבוסס DirectX, נגן מדיה לצלילים, וערכת אובייקטים לחלונות עבור הממשק, כאשר כל הספריות הנדרשות כלולות בסביבות העבודה NET. בגרסא 2 ומעלה הקיימת בתוך כל מערכות חלונות החל מ-Vista.

הסביבה עובדת עם C# ו-XML ומאפשרת ירידה לפרטים הבסיסיים ביותר של כל רכיב, שליטה על תצוגה של רכיבים וקבוצות של רכיבים באמצעות ערכות נושא, מערכת ניהול תהליכים מרכזית המסנכרנת בין התהליכים השונים ועוד. עקב הסיבוכיות הגבוהה של הסביבה, דרישות ממשק נמוכות, תהליך יחיד במשחק ומחסור בזמן, מרבית הפונקציאות הנ"ל אינן מנוצלות ע"י הפרוייקט.

רכיבים בשימוש

- GeometryModel3D: אובייקט המודל.
- MeshGeometry3D: אוסף של נקודות ממופה למשולשים המהווים את שלד האובייקט.
- PerspectiveCamera: מציגה את התמונה הדו-מימדית של החלל התלת מימדי מנקודה בחלל בוקטור מסויים. זויות המצלמה במשחק הן מבט מלמעלה במצב דמו ומזויות הראייה של השחקנים במצב משחק.
- AmbientLight: מאיר את החלל והאובייקטים באור אחיד.
- DirectionalLight: מאיר את האובייקטים באור בעל מקור וכיוון המשתנים יחד עם המצלמה.
- DiffuseMaterial: "חומר" פני המודל.
- Point3DCollection: מערך של נקודות המרכיבות את המודל.
- Int32Collection: מערך של מקומות הנקודות במערך הקודם כאשר כל קבוצה של שלוש מהווה משולש במודל.
- Point3D: מערך של שלושה משתנים עשרוניים המייצג נקודה במישורים Z,Y,X.
- SolidColorBrush אובייקט של צבע עם אפשרות לדרגות שקיפות, התוכנית משתמשת ברשימה אחת של צבעים עבור כל האובייקטים והמודלים.
- DiffuseMaterial החומר המוצג על פני המשולשים במודל, משתמש ברשימת הצבעים של התוכנית.
- Grid מחלק את השטח שלו לתת-משבצות שעל גביהן ניתן להציג אובייקטים.
- CheckBox, RadioButton, TextBlock, Button אובייקטים בממשק המתקשרים עם המשתמש.
- StackPanel קבוצה של אובייקטים המסודרים אחד מעל השני.
- MediaPlayer נגן קבצי מדיה התומך בפורמט mp3.

מודלים, אלגוריתמים ואובייקטים במשחק

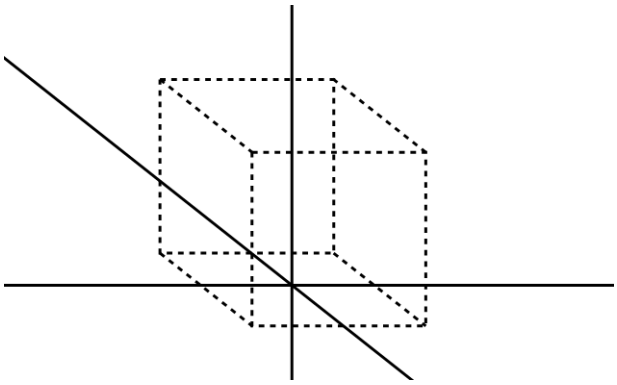
אובייקט המשחק מתקשר עם המשתמש דרך אובייקט Viewport3D המציג חלל תלת מימדי ובו לוח עם 32 משבצות שחורות, 24 אבני משחק שחורים ולבנים, רכיב סימון ושתי נקודות איסוף בצדי הלוח. התקשורת עם החלל נעשית ע"י סימון משבצת עם חייל השחקן ואז סימון משבצת שאליה אותו חייל יכול לעבור בהתאם למצבו על הלוח.

המשחק מבצע פעולות (סימון משבצת, הזזת אבן, היפוך הלוח, חישוב מהלך) דרך מנהל הפעולות כאשר המשתמש והמחשב מתקשרים עם המנוע באותה הצורה, המתפרשת כלחיצות על משבצות. מנגנון הבחירה סורק את המהלכים האפשריים של חייל על משבצת נבחרת וצובע את המשבצות שנסרקו בהתאם, הוא מאפשר הזזה רק למשבצות הצבועות בירוק או צהוב. כאשר חייל נמצא באמצע מהלך והמשתמש בוחר חייל אחר, החייל מוחזר למקומו והסימון שהוא יצר מתבטל לפני שהחייל החדש מסומן.

מודלים

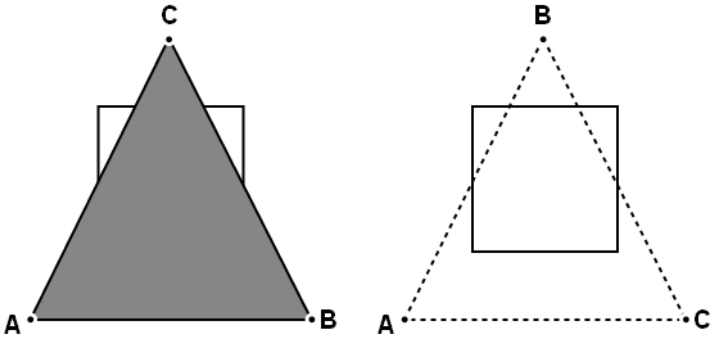
האובייקטים שהשחקן מתקשר איתם במשחק מיוצגים ע"י מודלים תלת מימדיים. נתוני המידות שלהם מתבססים על נתוני מידות הלוח (רוחב של משבצת שווה ל-1/8 מרוחב הלוח פחות רוחב השוליים, רוחב של אבן שווה ל-0.9 של רוחב המשבצת, עובי אובייקט הסימון שווה ל-0.05 מרוחב המשבצת וכד'). ניתן לשנות את צבעם או את מיקומם בחלל של אובייקטים אקטיביים (סמן,אבן,משבצת), מודלים של אובייקטים אקטיביים מקושרים למחלקות אותן הם מייצגים.

הצורות בתכנית מתבססות על שני אובייקטים מסוג Point3D: אחד עבור נק' המוצא ואחד עם ערכי הרוחב, גובה ועומק הצורה. הצורות מצויירות מאמצע צירי הX ו-Z כלפי חוץ ובמעלה ציר ה-Y (למטה עבור צילינדר חיצון). כלומר:



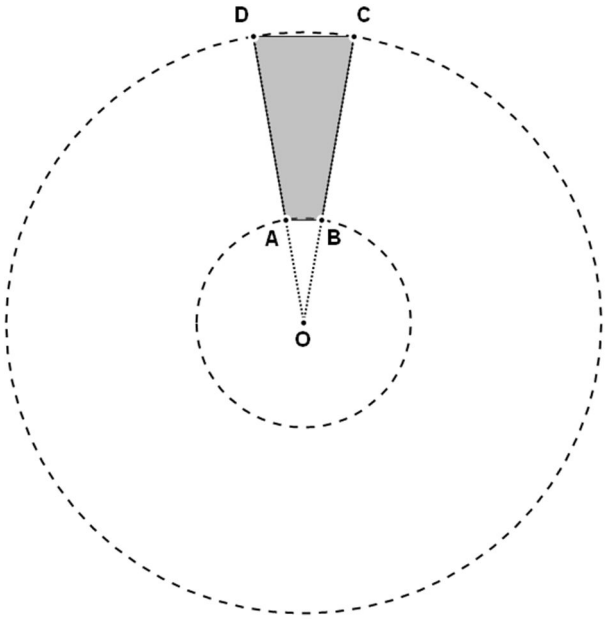
קוביה בנק' 0,0,0 עם מידות 1,1,1 תימצא בין 0.5- ל 0.5 על צירי X\Z ובין 0 ל-1 על ציר Y.

משולש: צורה דו מימדית בעלת פנים (השטח המוצג) וגב (שטח ריק). הפנים מוצגות בצד שמצוייר נגד כיוון השעון.



- **מרובע**: שני משולשים מצויירים נגד או עם כיוון השעון בהתאם לצרכי הפונקציה המזמנת. מרכיב את כל הצורות בתכנית למעט הפאות של מעגל סגור (צילינדר ללא רדיוס פנימי).

- **צילינדר**: N פאות שוות של מרובעים או משולשים בין R ל-r. גובה שלילי במתודה מצייר את הצד חיצוני וחיובי את הפנימי. r = 0 מצייר קונוס (או עיגול כאשר הגובה שווה ל-0). N קובע את הרזולוזיה של המעגל (N = 4 יצייר ריבוע, N = 6 משושה וכו', רזולוציה של 18 פאות ("משמונהעשר") נבחרה בתור מספיקה).



שרטוט של פאה 1/18 בתוך צילינדר ששני הרדיוסים שלו שונים ועם אותו ערך Y.

מספר הפאות = $N = 18$

רדיוס מעגל א' = $OC = OD = R$

רדיוס מעגל ב' = $OB = OA = r$

מונה בין 0 ל- $c = N-1$

$$d = \frac{360}{N} * \frac{\pi}{180} = \frac{2\pi}{N}$$

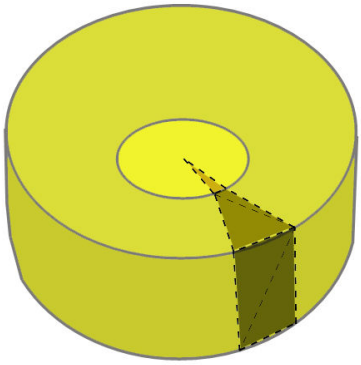
$$\mathrm{A} = \left(X_O + \mathrm{rSin}\big((c+1)d\big), Z_O + \mathrm{rCos}\big((c+1)d\big)\right)$$

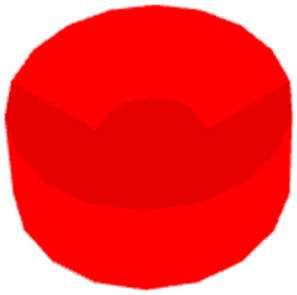
$$B = \left(X_O + \mathrm{rSin}(cd), Z_O + \mathrm{rCos}(cd)\right)$$

$$\mathrm{C} = \left(X_O + \mathrm{RSin}(cd), Z_O + \mathrm{RCos}(cd)\right)$$

$$\mathrm{D} = \left(X_O + \mathrm{RSin}\big((c+1)d\big), Z_O + \mathrm{RCos}\big((c+1)d\big)\right)$$

מודל החייל

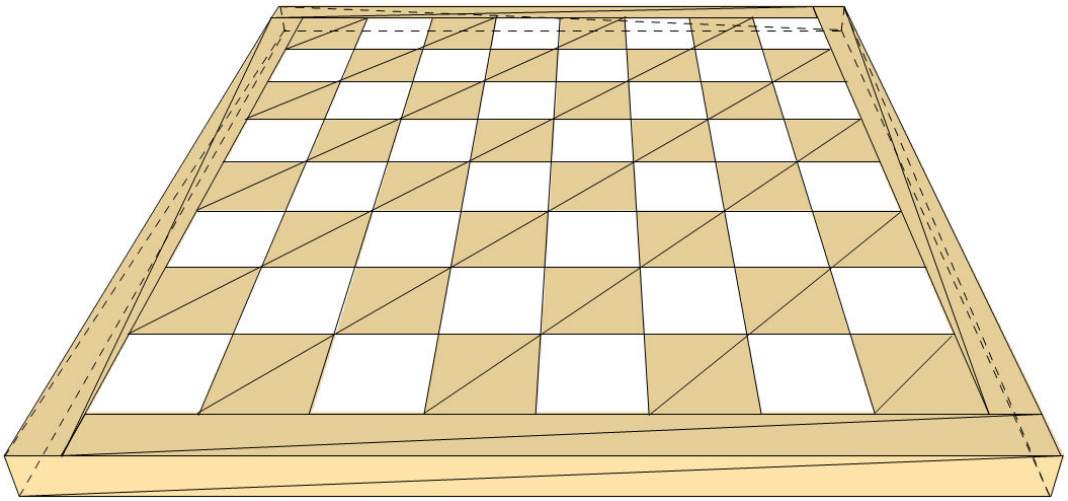




מבט מקרוב על הצורה במשחק ושרטוטה עם פאה ומשולשיה מודגשים ע"ג שלושת הצילינדרים המרכיבים את הצורה.

- צילינדר בן שני מעגלים בני אותו קוטר בגובה X
- צילינדר הפוך עם מעגל פנימי של חצי קוטר בגובה * 0.6
- צילינדר בגובה 0 וקוטר פנימי 0 בחצי קוטר בגובה מקור + גובה * 0.6

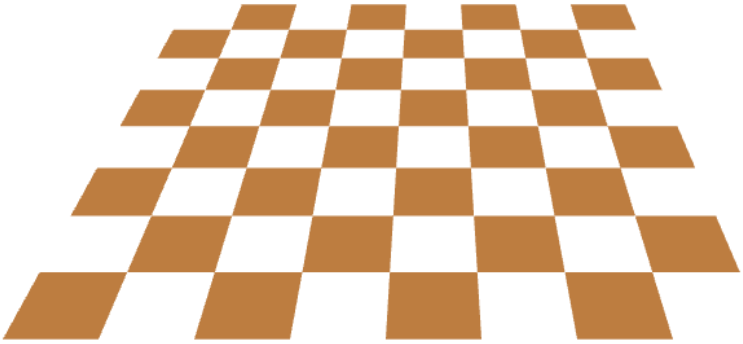
מודל הלוח



הלוח במשחק והמשולשים המרכיבים אותו. החלקים המקווקוים בלתי נראים מנק' מבט זו מכיוון שהם פונים כלפי הצד השני.

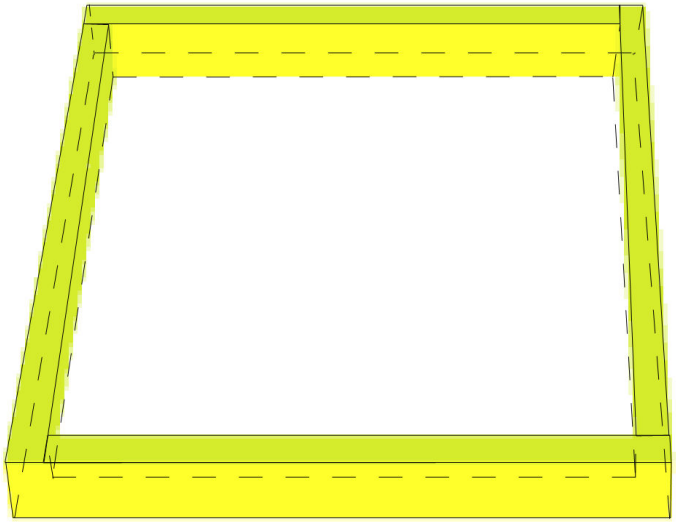
ארבעה מרובעים הפונים כלפי חוץ עבור הקירות, ארבעה נוספים עבור השוליים, 32 מרובעים פאסיביים. האובייקט אינו משתנה לאחר יצירתו.

מודל המשבצות



32 ריבועים במקומות קבועים שמשנים צבע, המודל של כל מרובע מקושר לאובייקט GFXCELL ונשלט דרכו.

מודל הסמן



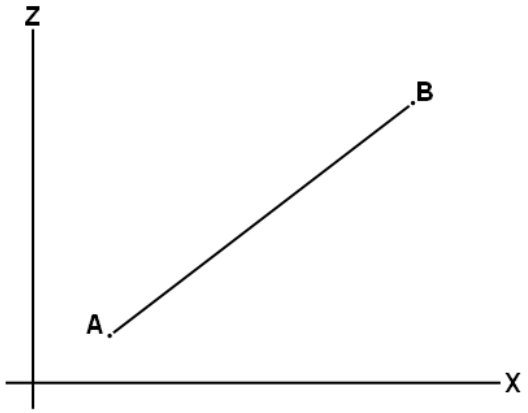
מסמן את משבצת המוצא של מהלך. מקושר לאובייקט רשימת הסימון. במצב המתנה, יושב בנק' 0,0,0 (נק' המוצא של הלוח).

תנועה

התנועה והסימון במשחק מבוצעים ע"י פעולות GFXOp המנוהלות בידי מנהל הפעולות GFXOpMan.

הפעולות הקיימות הן פעולת הזזת סמן שמתבצעת מיידית בעת זימונה ופעולת הזזה של חייל שיכולה להתבצע בבת אחת גם כן, אך בד"כ מתבצעת ב-N שלבים כל M מילי-שניות על מנת ליצור אפקט של אנימציה כאשר N שווה למרחק בין המוצא ליעד על ציר ה-X חלקי קבוע (0.2) ו-M שווה ל 50. הפעולה מזומנת ע"י המנהל באמצעות מתודת MovCall() שמזיזה את האובייקט הנבחר ע"י קידום הנתונים ומרימה דגל כאשר היא הגיעה לסופה. לכל פעולה יש מספר ID ורק פעולות בעלות ה- ID הכי נמוך מתבצעות בכל סיבוב. כאשר כל הפעולות ברשימה הסתיימו, המנהל בודק את התור המשני עבור פעולות אב חדשות כגון איתחול המשחק, ביצוע מהלך נוסף וכו'.

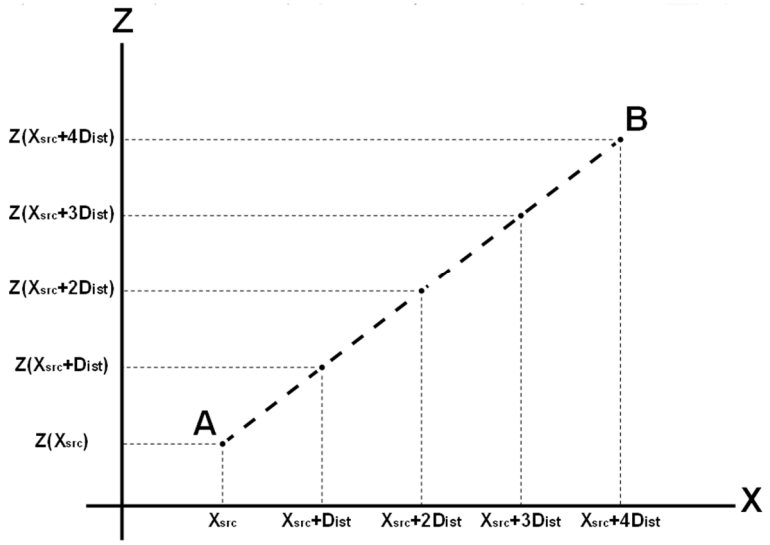
מוצא את נוסחת הישר העובר בנק' A (מוצא) ו-B (יעד)



$$m = \frac{Z_A - Z_B}{X_A - X_B}$$

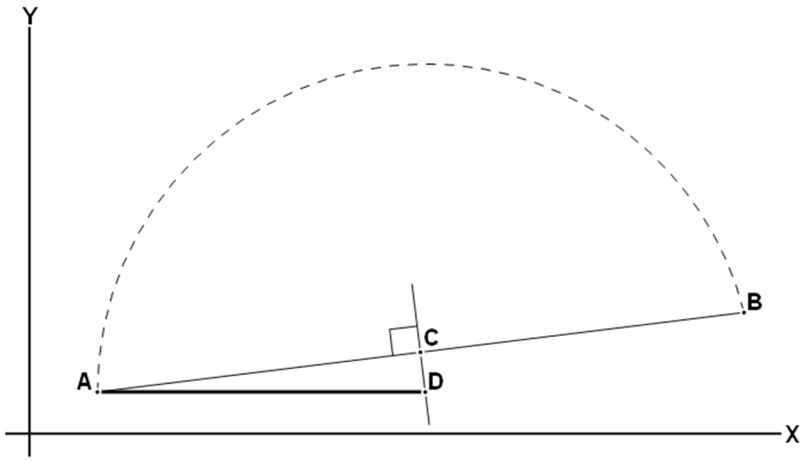
$$a = Z_B - mX_B$$

עבור כל X בין A ל-B



$$Z(X) = mX + a$$

מוסיף את ציר Y לאלגוריתם התנועה: מוצא את מרכז וקוטר המעגל העובר בנק' A ו-B.



הנקודות A ו-B יוצרות קו העובר בתוך מעגל, הקו המאונך לו בנק' האמצע (C) עובר בנק' האמצע של המעגל (D) שהיא נק' המפגש עם הישר היוצא מהנקודה הנמוכה מבין A ו-B (Ylow) והמרחק בינה ל-D על ציר ה-X הוא קוטר המעגל.

$$C = (\frac{X_A + X_B}{2}, \frac{Y_A + Y_B}{2})$$

$$m = \frac{X_B - X_A}{Y_A - Y_B}$$

$$D = \left(\frac{X_C - (Y_C - Y_{low})}{m}, Y_{low}\right)$$

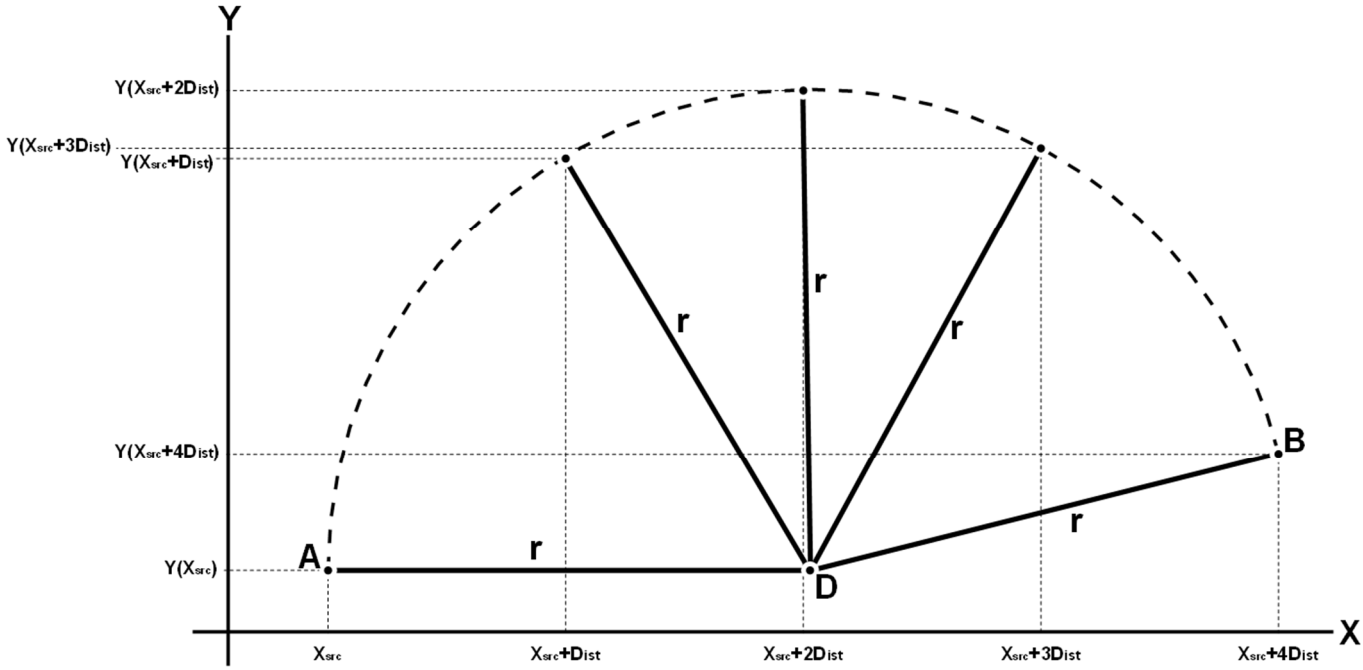
$$r = \left|X_D - X_{Y_{low}}\right|$$

AB = ישר העובר בנקי' A ו-B

C = נקי' האמצע של AB

CD = ישר המאונך ל-AB

D = מרכז המעגל בקוטר AD העובר בנקי' A ו-B



$$r = \sqrt{(X_D - X)^2 + (Y_D - Y)^2}$$

$$\vee$$

$$Y^2 - 2Y_D Y + (X_D - X)^2 + Y_D^2 - r^2 = 0$$

$$\vee$$

$$Y_{1,2} = \frac{2Y_D \pm \sqrt{(-2Y_D)^2 - 4((X_D - X)^2 + Y_D^2 - r^2)}}{2}$$

$$Y = Y_{high}$$

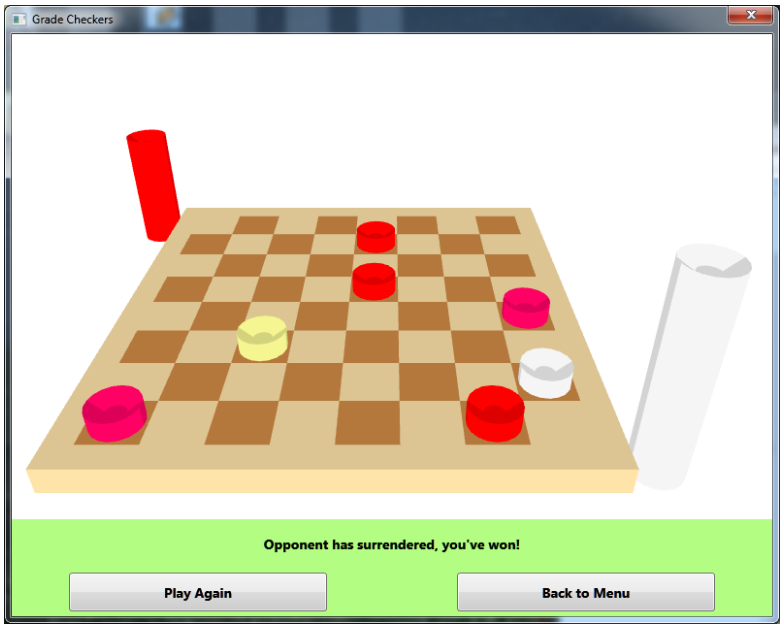
ממשק התוכנית הוא חלון עם שני תפריטים (Grid) מתחלפים החולקים ביניהם את אובייקט המשחק. בעת מעבר בין תפריטים החלון הראשי משנה את גודלו, טוען משחק חדש (או משחק דמו חדש) ומעביר את אובייקט המשחק לתפריט הנטען.

תפריט ראשי

מריץ משחק במצב דמו (מחשב נגד מחשב בדרגות רנדומליות שונות).

על המשתמש לבחור במשחק לשני שחקנים או עם המחשב, לבחור את צבעי האבנים של המחשב ואת רמתו. מאפייני שחקן המחשב אינם זמינים כאשר הצ'קבוקס אינו מסומן.

שחקן לבן (משתמש) באמצע אכילה כפולה. המשבצת הצהובה המסומנת היא משבצת המוצא של החייל, החייל המסומן בירוק קפץ מעל החייל המסומן בכתום והמשבצת הבאה עליה הוא יכול לעבור מסומנת בירוק. בסיום התור, החיילים הלכודים יועברו לערימה מימין.



על המשתמש בתורו להזיז את חייליו על ידי לחיצה על התא עם החייל המבוקש ואז על תא היעד. כאשר חייל מסומן, התכנית סורקת את סביבתו וצובעת את התאים הסמוכים בהתאם למצבם ביחס לחייל המסומן כאשר כל תא צבוע בירוק ניתן לסמן כתא יעד. במצב של אכילה, תא היעד נסרק כמקור משנה ומסמן את התאים הנוספים במידה וניתן להמשיך את המהלך. החיילים שנאכלו נערמים בצידי הלוח (שחורים בצד הלבן ולהיפך) וחוזרים ללוח בעת סידור מחדש (משחק חדש).

השחקן יכול להציע תיקו, להיכנע או לסיים את המשחק ולחזור לתפריט הראשי, המחשב בתורו יכול להציע תיקו או להיכנע בעצמו, במצב כזה השחקן מקבל התראה בשורת התפריט. כמו כן התראות קופצות בעת סיום משחק (עם אפשרות משחק חדש או יציאה), אישור כניעה וכד'.

כאשר המשתמש סיים את תורו, בהתאם לנתוני המשחק התכנית יוצרת ומבצעת מהלך עבור צד המחשב או מסובבת את המצלמה סביב הלוח ב-180 מעלות ומעבירה את התור לשחקן השני. כל התפריטים מצד המשתמש נעולים בעת חישוב וביצוע מהלך המחשב.

שחקן המחשב

האלגוריתם עובר על משבצות של לוח נתון ומחלק את המשבצות התפוסות בין ארבע רשימות (מהלכים ויעדים למהלכים עבור כל צבע). מהלכי אכילה עוברים עיבוד מיידי (נוצר עבורם אובייקט PCPMOVE) ומתווספים לרשימת המהלכים לפי צבע החייל, מהלכים רגילים (משבצות המוצא והיעד) מתווספים לרשימת היעדים (אובייקט PCPNEXT). בתום סריקת הלוח ובמידה ורשימת מהלכים כלשהי ריקה, המחשב מעבד את רשימת היעדים למהלכים, שם אותם ברשימת המהלכים ויוצר רשימת ציונים עבור כל אחד ממהלכיו בלבד (לא נוצרת רשימה עבור השחקן השני).

במידה ורמת האינטליגנציה של השחקן היא 2 ומעלה, המחשב יוצר וממלא את רשימת ההישארויות (PCPSTAY) עבור המהלכים שלו ומחסר את ציונן מציוני המהלכים ברשימת הציונים. אם הישארות מסויימת מגינה על משבצת אחרת (כלומר אם החייל זז אז חייל אחר ייאכל), ההשלכות של הפעולה מחושבות בתוך הציון של ההישארות (לוח חדש נוצר עבור המהלך וציוני ההישארות החדשים של המשבצות המוגנות לשעבר מתווספים לציון).

ברמה 3 המחשב עובר על שתי רשימות המהלכים ומוסיף את המהלכים המושפעים (מהלכים המשתמשים באותן משבצות) והישארויותיהם לרשימות חדשות, לאחר מכן הוא עובר על כל מהלך שלו, יוצר לוח חדש עבור אותו המהלך, מחשב מחדש את המהלכים מהרשימה עבור אותו לוח ומוסיף את הפרש הציונים בין הלוח החדש לנוכחי עבור אותם מהלכים חלקי סך המהלכים ברשימה לציון של המהלך.

בתום הפעולות הנ"ל המחשב בוחר את המהלך עם הציון הגבוה ביותר ומבצע אותו.

אובייקטים בשימוש:

- **מהלך:** סורק את משבצות הלוח הרלבנטיות למהלך ונותן ציון בהתאם למצבים שהוא יוצר. ציונים של אכילות המשך מחושבים ע"י יצירת מהלך חדש והוספה\הורדה של ציונו לציון מהלך האב. המשבצות שנסרקו ע"י המהלך (והמהלכים שנוספים שהוא בדק) מתווספות לרשימות של משבצות שעברו שינוי ומשבצות שלא.

- **הישארות:** פועלת על עקרון זהה למהלך למעט חלק מהמצבים ושדה הסריקה הראשוני. משבצות המוגנות ע"י החייל שבמשבצת הנבחרת מתווספות לרשימה.

- **התנגשות:** סורקת את משבצות המהלך הנבחר מול משבצות של מהלכים מתוך שתי רשימות (מהלכי הצד המחושב וצד הנגד). מהלכים שמתנגשים ביניהם מתווספים לרשימת ההתנגשויות.

המשחק מזהה שלושה גורמים עבור שחקן, עם "אינטרסים" ייחודיים להם:

- **השחקן** שואף להביא את הלוח למצב של ניצחון ע"י בחירה של מהלכים עם הציון הגבוה ביותר. כאשר מצב הלוח נוטה לרעתו הוא מעוניין לסיים את המשחק כמה שיותר מהר (ע"י הצעת תיקו או כניעה, בהתאם לחומרת הסיטואציה). ציוני המהלכים שהשחקן בוחר נובעים ממצבם והאינטרסים של ה**חיילים** ו**המלכות** שברשותו.

- **החיילים** שואפים להישאר בשורה הראשונה או בצדי הלוח, אדישים לשורה השנייה והשלישית ורוצים להתקדם מהמרכז לכל שורה שאחרי לכיוון הפינות עם נטייה להישאר בפינות אליהן הגיעו כדי לאפשר לאחרים להתקדם. ביחס ליחידות, החיילים שואפים להגן ולהיות מוגנים, ליצור ולשמור על מצבי סכנה אך עם העדפה חד משמעית של יצירת מצבי אכילה חיובית (כאשר השחקן לוכד יותר אבנים משחקן היריב בתום שרשרת המהלכים החזויה הקרובה).

- **המלכות** שואפות להתרחק מקצוות הלוח שהגיעו אליהן ולרדוף יחידות אויב עם העדפה כלפי תקיפת חיילים מאחורה מבלי לייחס חשיבות לשורה, שאר השיקולים דומים לשל חיילים.

אופן חישוב הציון

ערך המהלכים נקבע ע"פ ערך **משבצת** היעד וערך סכום **המצבים** ביחס אליה.

משבצות:

- עבור **חייל**:
  - **פינה \ לפני פינה:** 0.5 נק' או 0.25 בהתאם. למעשה, כל משבצת הנמצאת בקצה הלוח.
  - **שורה:** נק' 1 עבור השורה הראשונה, 0 עבור שתי השורות שאחריה ו-1, 2, 3 ו-4 בהתאם לשורות שמעבר.
- עבור **מלכה**:
  - **פינה \ לפני פינה:** מינוס חצי (או פלוס רבע) מהציון.

מצבים:

- **מוגן\מגן** (+/-0.5) : המשבצת מאחורי האבן תפוסה או שהאבן נמצאת בשורה הראשונה. במצב שהמשבצת שמאחורי האבן ריקה, אך יש אבן במשבצת שמאחוריה המצב מוגדר כ"חצי מוגן" ובהתאם מחזיר חצי מהניקוד.
- **חוסם\נחסם** (+/-0.25) : מעבר למשבצת זו יביא לחסימת דרכו של חייל אחר.
- **אזור סכנה** (+/-0.5) : משבצת ריקה בין שני חיילים יריבים או יותר (כלומר, התקדמות נוספת של אי אחד מהחיילים תביא ללכידתו האוטומטית) במצב של אבן מול שתי אבני אויב או שתי אבנים מול שלוש, הניקוד מוכפל ב-2- ("אזור סכנה רע").
- **לוכד\נלכד** (+/-10/20) : מצב של מהלך אכילה (אבן מול אבן יריב עם משבצת ריקה מאחוריה, 10 נק' עבור חייל, 20 עבור מלכה). במצב שבו המהלך מביא לסיטאוציה של אכילה במהלך שאחריו מתקבלות 5 נקודות בלבד (כי החייל יכול לזוז, להפוך להיות מוגן ע"י חייל אחר וכד'). גובה הציון ביחס לשאר דורס כל צירוף אפשרי של שיקולי נגד (הקרבת חייל שגוררת אכילה של מלכת אויב או שני חיילי אויב מוסיפה עשר נקודות [20+10- או 10+10+10-], אכילה של שני חיילי אויב ע"י מלכה או אכילה הדדית מניבות ציון מצטבר של 0 וכן הלאה).

כל אחד מהמצבים יכול להופיע כמה פעמים במצב הכולל וניקודם מצטבר בהתאם.

אופן הסריקה

האלגוריתם סורק את המשבצות הסובבות למשבצת בה יימצא החייל ומנקד את המהלך ע"פ נתוני המצבים שנוצרים בה. על מנת לחסוך במס' המשבצות והאפשרויות הנבחנות, הסביבה מחושבת בשלבים כאשר משבצות נוספות שייסרקו נקבעות ע"פ מצב המשבצות שכבר נסרקו (לדוגמא, במידה והאבן מגיעה למשבצת בה היא תיאכל בתור הבא, רק מהלך האכילה ונתיבו ייחושבו).

אופן סימון ייאוש המשבצות:

0. משבצת לא קיימת (חורגת מהלוח)
1. מלכת אויב
2. חייל אויב
3. משבצת ריקה
4. חייל
5. מלכה

- **בדיקה ראשונית** (ממומשת ע"י הפונ' `LandCheck`)

0. משבצת היעד (מהלך) או הנוכחות (הישארות) של החייל
1. "שמאל" המשבצת הבאה בוקטור התנועה של החייל
2. "ימין"
3. "שמאל אחורה"
4. משבצת המוצא של החייל במהלך (לא נבדקת, מוגדרת כריקה) או "ימין אחורה" עבור הישארות*

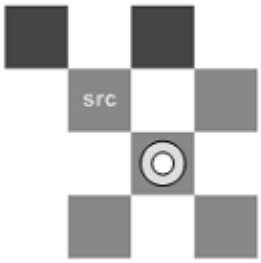
** במהלך מסוג אכילה, האבן הנלכדת יושבת על מש' 4 ומשבצת המוצא של האבן נמצאת מחוץ לשטח הנבדק ע"י הבדיקה הראשונית*

בודקת את הסביבה המיידית של האבן. במידה ומש' 1 ו-2 ריקות וקיימות, מבצעת בדיקה על שלושת המשבצות שמעבר להן, אם המשבצות מאויישות או מאויישות חלקית, בוחנת את המשבצת בנפרד. במידה והמשבצת האחורית קיימת, מבצעת בדיקה אחורית. חייל אויב במש' 1, חייל אויב במש' 2 ומש' 3 ריקה או מלכת אויב ב-3 וכלום ב-2 הם מצבי אכילת נגד ונבחנים ע"י יצירת מהלך נפרד עבורם. תוצאות כל הבדיקות, מהלכי נגד והמשך מתווספות לסכום הציון המוחזר.

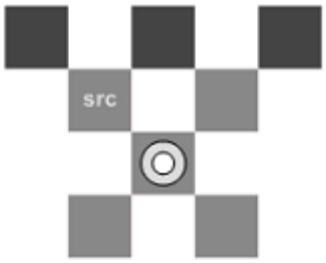
- **בדיקה קדמית** (ממומשת ע"י הפונ' `FrontSideCheck` ו-`DZCheckThree`)

בדיקה קדמית רגילה וכפולה

משמשת לבדיקת אזורי סכנה וחצי הגנה במצב בו מש' ה SRC ריקה, בדיקת אפשרות אכילה עתידית, אכילת המשך או אכילת נגד אם היא מאויישת ע"י אבן אויב או הגנה אם זאת אבן של השחקן.

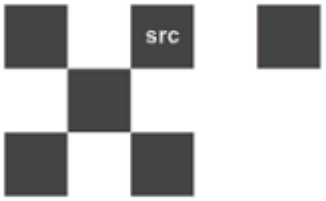


בדיקה קדמית עבור משבצת אחת + משבצות שכבר נבדקו

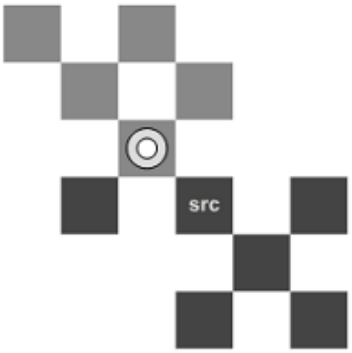


בדיקה קדמית עבור שתי משבצות + משבצות שכבר נבדקו

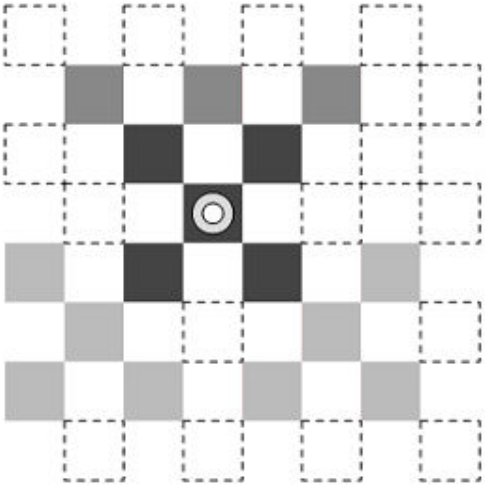
- **בדיקה אחורית**



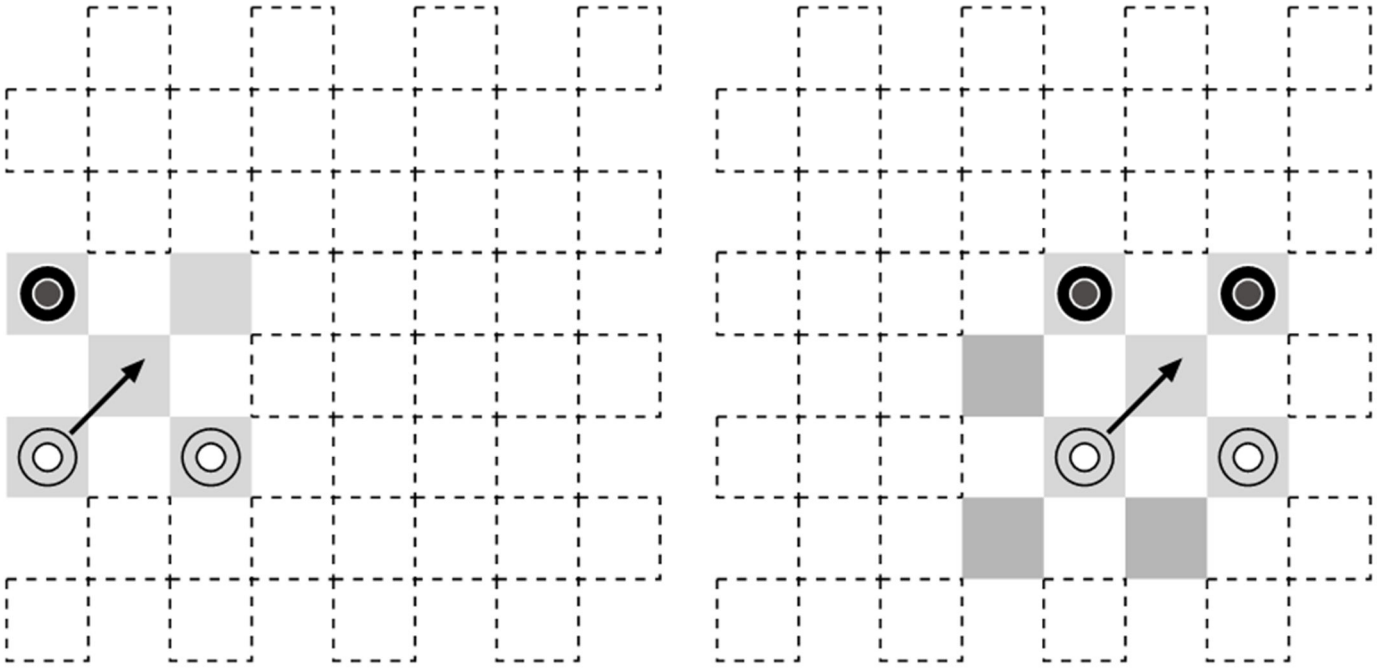
מתבצעת על משב' 3 במקרה של מהלך רגיל ושהיא קיימת, או על משב' 3 ו-4 במקרה של מלכה שהולכת אחורה. בודקת אם האבן מוגנת, האם במקרה של אכילה מצד האויב תהיה אכילת נגד, אכילת המשך או חסימה בחלק שכבר נסרק.



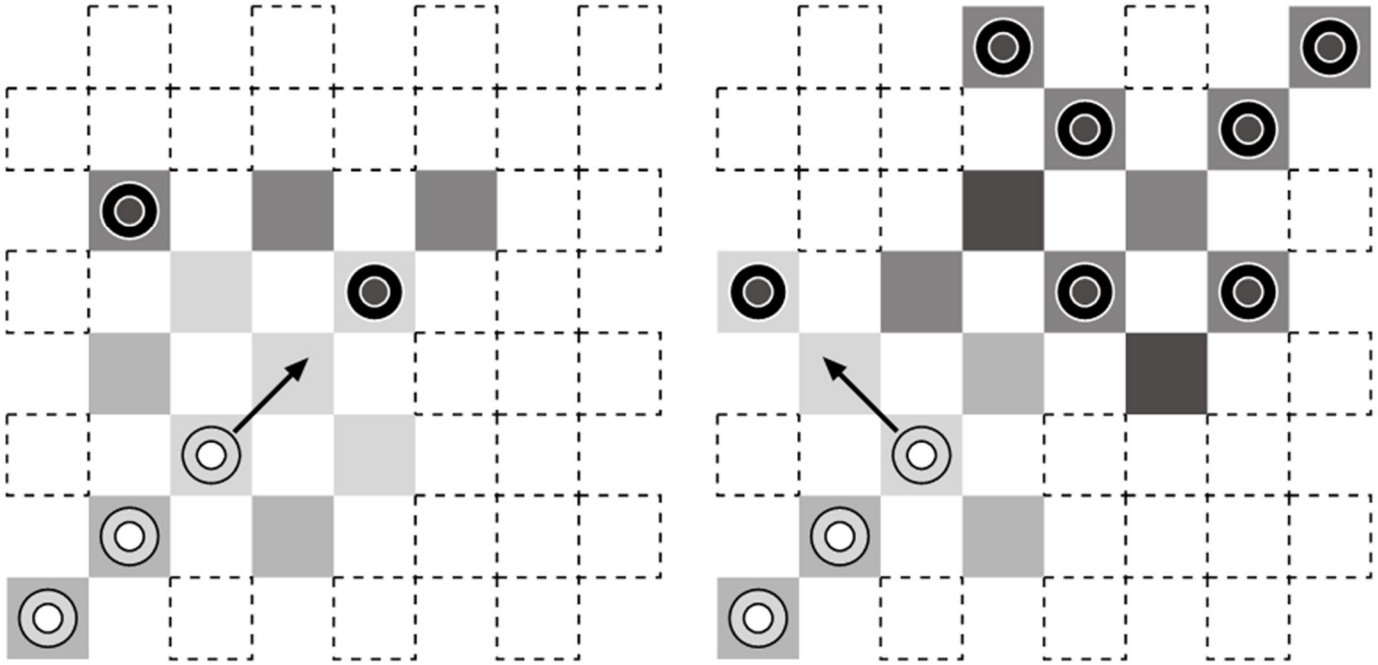
בדיקה אחורית עבור כיוון אחד + משבצות שכבר נבדקו



פריסה מקסימלית על לוח עבור מהלך בודד ללא מהלכי המשך



שני מהלכים מאותו לוח ומשבצותיהם. המהלך משמאל מקבל בונוס מנעילת החייל השחור בפינה ועלייה בשורה ליצירת אזור סכנה הנועל חייל נוסף, בעוד שהמהלך מימין מאבד עשר נקודות ע"ח אכילת הנגד שתבוא במהלך הבא . משבצות כהות יותר מוספות ע"י מהלך הבן.



שני מהלכים נוספים מאותו הלוח (עבור אותו חייל, לשני כיוונים שונים). המהלך משמאל גורר אכילה ע"י השחורים שתגרור אכילה נוספת ע"י הלבנים (תיקו) כאשר החייל הלבן נוחת בעמדת "אזור סכנה X 1". המהלך מימין גורר אכילה ע"י השחורים, אכילת נגד כפולה של הלבנים ואכילת נגד נוספת (תיקו). משבצות כהות יותר מוספות ע"י מהלכי הבנים.

צלילים במשחק

במשחק קיימים ארבעה סוגים של צלילים:

- **גרירה של חייל לבן על גבי הלוח**: צליל "יורד"
- **גרירה של חייל שחור**: צליל "עולה"
- **נחיתה של חייל על הלוח**
- **נחיתה של חייל על ערימת הלכודים**

לכל סוג ארבעה צלילים שונים הנבחרים באקראי בעת הפלייבק על מנת לא להמאיס על האוזן. צלילים נוספים כגון הרמת חייל או קטעי מוסיקה עבור אירועים במשחק בוטלו כדי לא ליצור אנדרלמוסיה רצופה של צלילים ורעשים. הצלילים מתנגנים באובייקט הנגן הקיים בכל אובייקט של חייל, בסיבוב הראשון של פעולה (גרירה במהלך רגיל) או בסופה (נחיתה בתום קפיצה). נתוני סוגי ומועדי הצלילים נקבעים בתוך אובייקט הפעולה, כאשר צליל ההתחלה מזומן ע"י הפעולה וצליל הסוף ע"י מנהל הפעולות (המונע מפעולות אחרות להשמיע צלילים באותו סיבוב על מנת למנוע תופעה של צלילים שמתלבשים אחד על השני במצב של סידור הכלים מחדש על הלוח).

מחלקת ה-DEBUG

מחלקה בתוך התוכנית המדפיסה טקסט מהמתודות לתוך קבצים ע"פ דרישות המפתח כאשר כל זימון של המתודה מדפיס שורה נפרדת יחד עם שם המחלקה המזמנת ולאחריה מספר משתנה של טאבים ע"פ עומק המתודה, על מנת להקל על קריאת הפלט (כמה מאות אלפי שורות עבור משחק ממוצע עם שחקן מחשב ברמה 3). המחלקה אינה פעילה בגרסת המשתמש הסופית אך עדיין קיימת בתוך הקוד.

פלט לדוגמא עבור מהלך של מחשב:

```
PCPMove  PCPMove(12, 17): { -2, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, -1, 0, 0, -1, 0, -1, 0, -1, -1, 0, 0, 0, 0, 0, -1, -1, -1 }
PCPMove  LandCheck(12,17,True, False, False:)
PCPMove          Dest Score == 0.5
PCPMove          323
PCPMove          FrontSideCheck(sourceid 17, sqf 3, sqfid 21, sqotherf 2, sqb 3, sqbid 12, isKing False, capt False, captid -1, left False,
                  whiteTurn True,skipBack False)
PCPMove                  NextTwoCheck(21,False,True, True, False:)
PCPMove                          free x 2
PCPMove          FrontSideCheck(sourceid 17, sqf 2, sqfid 20, sqotherf 3, sqb 3, sqbid 13, isKing False, capt False, captid -1, left True,
                  whiteTurn True,skipBack False)
PCPMove                  blocked x 1 -0.25
PCPMove                  :2COUNTER CAPT 20 to 13
PCPMove          BackSideCheck(sourceid 17, sqb 3, sqbid 13, sqf 2, sqfid 20, sqff 3, isKing False, capt False, captid -1, left False, whiteTurn True
                 ,skipBack False)
PCPMove                  NextTwoCheck(13,False,True, False, True:)
PCPMove                          semi defended x 2 +0.5
PCPMove          PRENEXT SCORE: 0.75
PCPMove          cont=0 capt=1
PCPMove                  NextCaptCheck(20,13,17):
PCPMove                          PCPMove(20, 17, 13): { -2, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, -1, 0, 1, -1, 0, -1, 0, -1, -1, 0, 0, 0, 0, 0, -1, -1, -1 }
PCPMove                          LandCheck(20,13,True, True, True:)
PCPMove                                  Dest Score == 0.5
PCPMove                                  224
PCPMove                                  FrontSideCheck(sourceid 13, sqf 2, sqfid 10, sqotherf 2, sqb 3, sqbid 17, isKing False, capt True,
                                          captid 17, left True, whiteTurn False,skipBack False)
PCPMove                                  blocked x 1 -0.25
PCPMove                                  :2COUNTER CAPT 10 to 17
PCPMove                                  FrontSideCheck(sourceid 13, sqf 2, sqfid 9, sqotherf 2, sqb 4, sqbid 18, isKing False, capt True,
                                          captid 17, left False, whiteTurn False,skipBack False)
PCPMove                                  :0CONT CAPT 13 to 4
PCPBoard                                          BalanceCheck: tmp: -11 piecesBlack 9 piecesWhite 4
```

```
                                                                                pieceBalance -6 piecesTotal 13
PCPBoard                                                                        3BALANCE = 1
PCPMove                                                         blocked x 1 -0.25
PCPMove                                                 BackSideCheck(sourceid 13, sqb 4, sqbid 18, sqf 2, sqfid 9, sqff 2, isKing False, capt True,
                                                                 captid 17, left True, whiteTurn False,skipBack True)
PCPMove                                                 PRENEXT SCORE: 0
PCPMove                                                 cont=1 capt=1
PCPMove                                                         NextCaptCheck(13,4,9):
PCPMove                                                                 PCPMove(13, 9, 4): { -2, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, -1, 0, -1, 0, 0,
                                                                                         -1, 0, 0, 0, -1, -1, 0, 0, 0, 0, 0, -1, -1, -1 }
PCPMove                                                                         LandCheck(13,4,True, False, True:)
PCPMove                                                                                 Dest Score == 1.5
PCPMove                                                                                 533
PCPMove                                                                                 FrontSideCheck(sourceid 4, sqf 5, sqfid 0,
                                                                                                 sqotherf 3, sqb 3, sqbid 9, isKing False,
                                                                                                 capt True, captid 9, left False,
                                                                                                 whiteTurn False,skipBack False)
PCPMove                                                                                          blocked x 1 -0.25
PCPMove                                                                                          blocks x 1 -0.25
PCPMove                                                                                 FrontSideCheck(sourceid 4, sqf 3, sqfid 1,
                                                                                                 sqotherf 5, sqb 3, sqbid 8, isKing False,
                                                                                                 capt True, captid 9, left True,
                                                                                                 whiteTurn False,skipBack False)
PCPMove                                                                                          precrown
PCPMove                                                                                 BackSideCheck(sourceid 4, sqb 3, sqbid 8, sqf 3,
                                                                                                 sqfid 1, sqff 5, isKing False, capt True,
                                                                                                 captid 9, left False, whiteTurn
                                                                                                 False,skipBack False)
PCPMove                                                                                          NextTwoCheck(8,False,False,
                                                                                                         False, True:)
PCPMove                                                                                 PRENEXT SCORE: 1
PCPMove                                                                                 cont=0 capt=0
PCPMove                                                                         LandCheck res: 1
PCPMove                                                                 final score: 11
PCPMove                                                         MergeNext(): 3 3
PCPMove                                                                 before:
                                                                         ch: 20 17 13
                                                                         af: 10 9 18 6 4
                                                                         after:
                                                                         ch: 20 17 13 9 4
                                                                          af: 10 18 6 0 1 8
                                                                          CaptMoves: 4
PCPMove                                         LandCheck res: 11
PCPMove                                 final score: 21
PCPMove  LandCheck res: -21
PCPMove  final score: -21
```

במהלך הפרוייקט נעזרתי בספר WPF Control Development Unleashed (P.Podila, K.Hoffman) ובמדריכים הבאים:

WPFTutorial.net (Christian Moser)

https://www.wpftutorial.net/Home.html

Drawing a Cube in WPF (Sean Sexton)

https://stuff.seans.com/2008/08/13/drawing-a-cube-in-wpf/

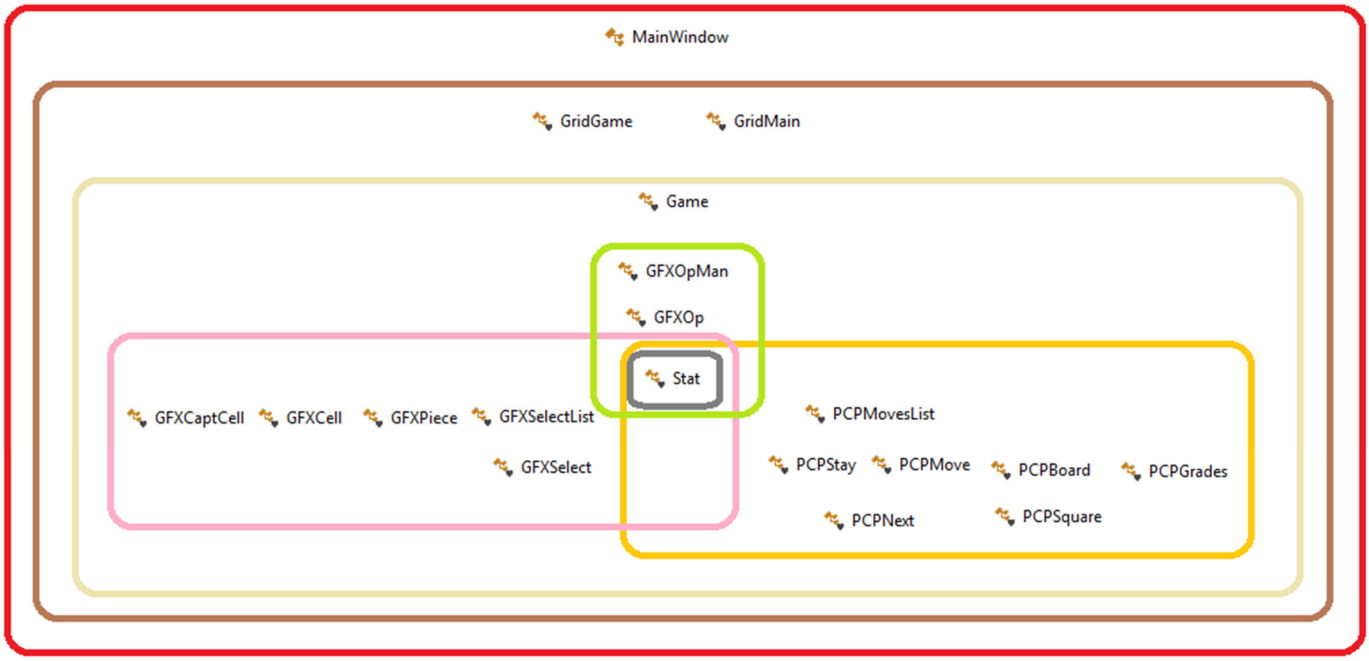
Rotate a 3D cube using XAML and C# (Rod Stephens)

http://csharphelper.com/blog/2014/10/rotate-a-3d-cube-using-xaml-and-c/

Automatic 3D Normal Vector Calculation in C# WPF Applications (Michael Hall)

http://xoax.net/blog/automatic-3d-normal-vector-calculation-in-c-wpf-applications/

רשימת מחלקות:



STAT

מחלקה סטטית המכילה פונקציות המשומשות ע"י מחלקות מרובות בתכנית (כגון פונ' היוצרות את המודלים ופונ' הממירות בין מערכי האובייקטים השונים.

```
//adds points for a square, boolean flips displayed side
public static void AddSquarePts(Point3DCollection points, Point3D s, Point3D ms, Boolean high)
//all the passive bits of the board
public static GeometryModel3D NewBoardModel(Point3D s, Point3D ms, double bdRim, DiffuseMaterial[]
colours)
//selection object: source point, measurements, rim width, colours array
public static GeometryModel3D NewSelectModel(Point3D s, Point3D ms, double rim, DiffuseMaterial[] colours)
//draws cylinder, returns num of points added
public static int AddCylPts(Point3DCollection qPoints, Point3D s, double height, double rLarge, double rSmall, int
n)
// board square (active)
public static GeometryModel3D NewSquareModel(Point3D s, Point3D ms, DiffuseMaterial[] colours)
// two cylinders and a circle
public static GeometryModel3D NewPieceModel(Point3D s, Point3D ms, int clrIndex, int resolution,
DiffuseMaterial[] colours)
// converts a board code (00-77) to square index (0-31) e.g. rctoi(77) == 31
public static int RCtoI(int r, int c)
// the opposite of the above. itorc(31) == 77
public static int ItoRC(int i)
//check if next square is within board bounds
public static bool BoundCheck(int r, int c, int e, bool left, bool up)
//next row index
```

```
public static int RowNext(int r, int e, bool up)
//next column index
public static int ColNext(int c, int e, bool left)
//next square index
public static int NextSq(int id, int levels, Boolean white,Boolean up,Boolean left)
//convert board array to pieces array
public static void BoardToPieces(int[] board, int[] bl, int[] wh)
//vice versa
public static PCPBoard PiecesToBoard(int[] bl, int[] wh)
```

MAINWINDOW

המחלקה הראשית: יוצרת את האובייקטים השונים (תפריטים, מודלים, משחק וכו') ומתאמת ביניהם.

```
//creates the brushes, the materials, the grades, the game, the screens, loads the main screen with a demo game
public MainWindow()
//swaps between grids in window
public void LoadGrid(int grid)//1 = main, 2 = game
//loads prompt in game grid, op
public void LoadPrompt(int op, string message)
//start new game
public void Init(Boolean playerWhite, Boolean comPlayer, Boolean demo,int intel)
//returns the cpu's intelligence level
public int GetGameInteLevel()
//offer draw to cpu
public void OffDraw()
//restart comp moves
public void ResumeTurn()
```

GRIDMAIN

מציגה את התפריט הראשי ושולחת הוראת שיגור של משחק על פי האפשרויות.

```
//creates the main menu grid; logo, demo game, start button, settings panel
public GridMain(MainWindow main, SolidColorBrush[] brushes)
//switches to game grid and inits a new game using settings from the settings grid
private void StartGame(object sender, EventArgs e)
//disables / enables the settings menu depending on the checkbox
private void ChkClick(object sender, EventArgs e)
```

GRIDGAME

מציגה את המשחק, את תפריט המשתמש ומציגה הודעות מהמשחק.

```
//creates the game grid; game, menu/prompt panel
public GridGame(MainWindow main, SolidColorBrush[] brushes)
//creates the objects
private Grid CreateGameGrid()
```

//creates the grid for the game menu panel
private Grid CreateGameManGrid()
// creates the prompt panel grid
private Grid CreatePromptGrid()
// loads prompt of type OP with MESSAGE
//-1 none (menu)
// 1 player won (replay/menu prompt)
// 2 draw
// 3 player lost
// 4 draw off (accept/reject prompt)
// 5 confirm player surrender
// 6 cpu rejected draw offer (surrender/continue prompt)
public void LoadPrompt(int op, string message)
//left button pressed
private void PrmptBtnA(object sender, EventArgs e)
//right button pressed
private void PrmptBtnB(object sender, EventArgs e)
//draw button pressed, sends draw offer to cpu
private void DrawBtn(object sender, EventArgs e)
//surrender button pressed, prompts to confirm
private void SurrBtn(object sender, EventArgs e)
// menu button pressed, loads main grid and starts a demo game
private void BackBtn(object sender, EventArgs e)

GAME

מחלקת המשחק הראשית. יוצרת ומנהלת את האובייקטים בחלון המשחק ואת המשחק עצמו. משנה את זוית המצלמה, מזיזה ומסדרת את הכלים, מחליפה בין תורים, מעבדת את הקלט מהמשתמש ומחלקות שחקן המחשב לפעולות.

public Viewport3D GetViewport()
public PCPGrades GetGrades()
//change camera: 1 = top view, 2 = side view
public void SetCamPos(int mode)
// queues a new game command in the op manager
public void QueueInit(Boolean playerWhite, Boolean comPlayer, Boolean demo,int intel)
// creates the models, game board and grades and sets the board in full captured position, if DEMO is true then starts a demo game
public Game(MainWindow main,DiffuseMaterial[] clrs, Boolean playerWhite, Boolean comPlayer, Boolean demo,PCPGrades grades)
//creates the moves list, nabs the squares from the top graded move and clicks them
public void CreateMoves()
//finds where the piece is (-1 == captured), takes three-digit pid ((2 pid)(1 b/w))
public int GetPieceCell(int pieceId)
//reset board and start new game
public void Init(Boolean playerWhite, Boolean comPlayer, Boolean demo, ArrayList move, PCPGrades grades, int intel)
//Capture all pieces
public void CaptureAll(Boolean white,Boolean atOnce)
//Move pieces from capt spot to board according to the map array

public void RestoreOnBoard(int[] pieCells, Boolean white)
//set turn for side
public void FlipSide(Boolean playerWhite)
//rotate board 180 degrees
private void FlipBoard(Boolean whiteSide)
//flip turn
public void FlipTurn()
//process draw offer from player
public void OffDraw()
//if it's been 5 turns since last check, checks board balance and decides if to offer draw, surrender (returns true) or continue (returns false)
private Boolean CheckBalance()
//processes mouse click and checks for correlation with one of the active squares
private void SelCell(object sender, MouseButtonEventArgs e)
//processes a square click
public void SelCell(int index)
//outputs the board data in a format that's copypastable into a new int[32] (for debugging)
public string GetBoard()

GFXOPMAN

מנהל הפעולות של המשחק. מוסיף ומוחק את ההוראות ברשימה, ומריץ את הפעולות.

//creates a timer dispatcher that performs ThreadCycle every millisecTimer ms
public GFXOpMan(Game game)
//returns the op id
public int GetOpId()
//goes over the operations in the list, sending a MovCall to the one(s) with the lowest opId, if movement queue is empty checks if a game reset or cpu turn are queued
private void ThreadCycle(object sender, EventArgs e)
//for cpu to play the turn
public void QueueSelect(int select)
//move selection obj to cell
public void Select(GFXSelectList select, GFXCell dest)
//hide selection obj
public void UnSelect(GFXSelectList select,Boolean flipturn)
//move piece from cell to cell
public void Move(GFXPiece piece, GFXCell source, GFXCell dest, Boolean jump)
//move piece from cell to cell with (un)crowning (for move reset)
public void Move(GFXPiece piece, GFXCell source, GFXCell dest, Boolean jump, Boolean crown)
//queue new game (for init)
public void AddToQueue(int[] whites, int[] blacks, Boolean whiteTurn, Boolean playerWhite, Boolean comPlayer, Boolean demo,ArrayList move)
public void QueueInit(Boolean playerWhite, Boolean comPlayer, Boolean demo, int GmInt)
//queue next cpu turn
public void QueueMoves()
//move piece from capture spot to cell

public void Restore(GFXPiece[] pieces, int piece, GFXCell dest, Boolean jump, GFXCaptCell captSq,GFXOpMan oplist,Boolean white)
//moves GamePiece from GameCell to GameCaptCell
public void Eat(GFXPiece piece, GFXCell source, GFXCaptCell dest, int opid,Boolean atOnce)
//reshuffle in the captured spot
public void ReEat(GFXPiece piece, Point3D dstPt)
// calculate num of cycles for movement
private int GetAnimTime(Point3D src, Point3D dst)
//remove op from list
public Boolean RemOp(int index)
//return op from index
public GFXOp GetOp(int index)

GFXCAPTCELL

אובייקט "מחנה שבויים", מחזיר X ו-Z קבועים ו-Y המשתנה בהתאם למספר החיילים שעליו.

//collection spot for captured pieces
public GFXCaptCell(Point3D sPt)
//adds a piece to the stack
public void AddPiece(GFXPiece piece)
//removes a piece from the stack
public void Restore(GFXPiece[] pieces,int piece,GFXOpMan opList)
// returns the center-top point
public Point3D GetPoint()

GFXCELL

אובייקט משבצת משחק. מיוצג ע"י ריבוע, מחליף צבע ומצב (ריק, חייל\מלך שחור\לבן) ומחזיר את קודי הצבע והאבן שעליו (או -1).

//active game square
public GFXCell(Point3D sPt, Point3D msPt, int colour, int pieceIndex, DiffuseMaterial[] colours)
//change colour
public void Select(int colour,bool visible)
//change back to default colour
public void Select()
//remove piece from square
public void RemPc()
//add piece to square
public void AddPc(int pieceIndex)

GFXPIECE

אובייקט האבן. מיוצג ע"י מודל (אבן). מחזיר ומעדכן את מקומו בחלל המשחק, את צבעו ואת מצבו (חייל\מלך, האם הוכתר במסגרת המהלך הנוכחי), מנגן צלילים.

```
//game piece
public GFXPiece(int id,Point3D sPt, Point3D msPt, int colourIndex, GeometryModel3D model3D, DiffuseMaterial[] colours)
//play sound
//1 drag
//2 drop
//5 stack
public void Sound(int type)
//is the piece crowned?
public Boolean Crowned()
//has the piece been crowned in this turn?
public Boolean JustCrowned()
//complete the crowning
public void UnJustCrowned()
//update source point
public void SetPt(Point3D pt)
//recalculate source-origin vector
public void GetVect(Vector3D vect)
//return a recalculated copy of the vector
public Vector3D GetVect()
//change to select colour
public void Select(int colour,bool visible)
//reset colour
public void Select()
//initiate crowning
public void Crown(Boolean initSetup)
//uncrown
public void UnCrown()
```

GFXSELECTLIST

מנהל את סימון המשבצות והכלים במשחק, את שינוי מצבם ואת תנועות הביניים במצב של אכילות מרובות.

```
public GFXSelectList(Point3D spt, GeometryModel3D model3D, DiffuseMaterial[] colours,
GFXCell[] gameCl, GFXPiece[] whitePc, GFXPiece[] blackPc, GFXCaptCell whiteCpt, GFXCaptCell blackCpt,
Boolean whiteTurn)
//selected piece
public int GetSelPiece()
//move status for main selection
public int GetSelRes()
//returns current selected square
public int GetCurrentCell()
//returns the source square
public int GetPrimaryCell()
```

```
//returns distance vector for model
public Vector3D GetVect()
//update current location
public void SetPt(Point3D pt)
//returns a selection object
public GFXSelect GetSelection(int index)
//new primary selection
public void Select(int cellIndex, bool visible, int moveType, int[] turnStatus)
//new secondary selection
public void SubSelect(int cellIndex, bool visible)
//clear all selections
public void Select()
//finds and returns the id of piece that's captured in a move
public int GetCaptId(int cellId)
```

GFXSELECT

מחלקה בת של GFXSELECTLIST, ענף (קבוצה של משבצות) לכיוון אחד (למטה\למעלה, שמאלה\ימינה) ממשבצת מקור כלשהי.

```
//primary select
public GFXSelect(GFXCell[] gameCl,
GFXPiece[] whitePc, GFXPiece[] blackPc, GFXCaptCell whiteCpt, GFXCaptCell blackCpt,
Boolean whiteTurn, int cellIndex, Boolean visible, Boolean clear, int exclude, Boolean enabled)
//subselect
public GFXSelect(GFXSelect parent, GFXCell[] gameCl,
GFXPiece[] whitePc, GFXPiece[] blackPc, GFXCaptCell whiteCpt, GFXCaptCell blackCpt,
Boolean whiteTurn, int cellIndex, Boolean visible,int exclude)
//searches the four paths and returns the id of the cell with the captured piece
public int GetCaptId(int cellId)
//searches one path for the captured piece, returns cell id or -1
int GetCapt(int[] path, int cellId)
//clears selections in a path if it's uncapturable
void ClearPath(int[] path)
//clears selection
public void ClearPrevSelect()
//calculates path and sets colours if visible

void calcPath(GFXPiece selPiece, int[] path, bool left, bool up, bool forward, bool visible,int exclude,Boolean
enabled)

//colour the source square
public void FillMainSelect(int colour, Boolean visible)
//colour all the squares in the paths
public void FillSubSelect(int colour,Boolean visible)
```

GFXOP

מחלקה בת של GFXOPMAN, מחשבת את מסלול התנועה של חייל מנק' מוצא ליעד בשניים או בשלושה מימדים, פורסת אותו למספר פריימים מוגדר ויוצרת פונקציה המזיזה בכל קריאה את האובייקט בפריים אחד ע"פ הנוסחה שחושבה.

```
//is the operation finished
public Boolean IsFinished()
//should turn be flipped once the operation is finished
public Boolean FlipTurn()
//op id
public int GetId()
// returns op params in a string (for debugging)
public override string ToString()
//move piece from its current position to dstPt in either a straight line or a jump
// type: 1=move select, 4 = move piece in one go
public GFXOp(GFXPiece piece, Point3D dstPt, Boolean jump, int animTimes, int type,int opId,int gameId,int
soundStart,int soundEnd)
//move select object
public GFXOp(GFXSelectList select, Point3D dstPt, Boolean flipTurn, int opId, int gameId)
//calculates the center X of the circle
double XYCalcCentX(double x1, double y1, double x2, double y2)
//calculates the Y position for X relative to center X and center Y
double XYCalcY(double ccx, double ccy, double rad, double x)
//calculates M for the X-Z equation
double XZCalcM(double x1, double y1, double x2, double y2)
//calculates A for the X-Z equation
double XZCalcA(double x, double y, double m)
//calculates Z position for X
double XZCalcZ(double x, double m, double a)
//Operation cycle
public void MovCall()
//play end sound if one has been selected
public void PlayEndSound()
```

PCPBOARD

מחלקה המייצגת לוח מבחינתו של שחקן המחשב, מכילה מידע על משבצות, תכולתן ומאזן הכלים במשחק. המחלקה יכולה לבצע שינויים בלוח הקיים או ליצור ולהחזיר העתק של עצמה עם הלוח המעודכן.

```
// creates the board, the squares get their grades from their id number (i.e. their positions on the board)
public PCPBoard(int[] board)
// checks balance and returns a recommendation code (-2 to 2 or 66)
public int BalanceCheck()
//moves the OCP from SRC to DST and 0 to SRC. if DST == -1 (capture) then just does the 0
public void Move(int src, int dst)
//creates amd returns a copy of itself moving SRC to DST in the process.
public PCPBoard Copy(int src, int dst)
//returns square by index
```

public PCPSquare GetSq(int id)
//returns square occupation status (-2 to 2)
public int GetSqOcp(int id)
//returns the grade of a piece by id
public double GetSqGrade(int id, Boolean white, PCPGrades grades, Boolean isKing)
//returns the board values in a string formatted to be easily pastable into an int[] (for debugging)
public override string ToString()

PCPSQUARE

מחלקה בת של PCPBOARD, מכילה מידע על תכולתה של משבצת ונתונים קבועים המשפיעים על ערכה על הלוח (פינה, אחת לפני פינה). יכולה לבצע שינויים או ליצור ולהחזיר העתק של עצמה עם העדכון.

public PCPSquare(int id, int occupied,Boolean corner,Boolean preCorner)
//calculate and return the square's grade with relative row grade
public double GetScore(Boolean white, PCPGrades grades,Boolean isKing)
//square id
public int GetId()
//square's occupation status (-2 to 2)
public int GetOcp()
//set occupation status
public void SetOcp(int val)
//return a copy of the square with the updates OCP
public PCPSquare Copy(int ocp)
//return a copy of itself
public PCPSquare Copy()

PCPMOVESLIST

מחשבת את המהלך האופטימלי עבור צד כלשהו מתוך רשימה של מהלכים אפשריים על לוח מסויים.

//creates the moveslist object; fills up the black and white moves lists (int 1) then sets the top choice index
public PCPMovesList(PCPGrades grades, PCPBoard board, ArrayList prevCompMoves, Boolean whiteTurn)
// depending on intLevel performs or skips creating a stays list for cpu (int 2) and creating an interferences list for the moves (int 3)
// afterwards, finds the top graded move in the cpu list and returns its place in the list
private int CalcTop(ArrayList cpuMoves, ArrayList plrMoves, Boolean whiteTurn)
//check how the move affects other moves
private void InterCheck(double[] InterfScores, ArrayList cpuMoves, ArrayList plrMoves)
//check one move against the two lists and return the balance between their old grades and their grades should the move be performed
private double InterfsWith(PCPMove move, ArrayList cpuMoves, ArrayList plrMoves)
//finds (or not) the stay with the identical movId
private int GetStayListId(int movId)
//add move to list providing it's not already there
private void AddToMoves(PCPMove move, ArrayList list)
//same but with stays
private void AddToStays(PCPStay stay, ArrayList list)

//goes over movList to see if any of the moves overlap with move. if they do, adds them and their stays to movListNew and stayListNew respectively

public void InterfLoad(PCPMove move, ArrayList movList,ArrayList movListNew,ArrayList stayListNew,Boolean cpu)
//returns selection object
public PCPMove GetSelection(int index,Boolean white)
//fills the moves array list with all of the side's possible moves
private double ReCalcMoves(PCPBoard newBoard, ArrayList movesList, ArrayList staysList)
//goes over all the squares checking to see if they're occupied and if the pieces that are on them can move
//if they can, either adds them to list (in case of capture moves) or queues them for the end
//at the end, if there were no capture moves, adds the queued moves to the list
private ArrayList CalcMoves(Boolean white)
//what it says on the tin
private Boolean IsFoe(Boolean whiteTurn, int val)
//return move from list by index
private PCPMove ListGet(ArrayList list, int id)

PCPMOVE

מחלקה בת של PCPMOVESLIST. ממפה, מנתחת ונותנת ציון להשלכות של מהלך כלשהו על לוח מסויים.

//regular move
public PCPMove(PCPGrades grades, PCPBoard board, int src, int dst, int movId)
//capture move
public PCPMove(PCPGrades grades, PCPBoard board, int src, int capt, int dst, int movId)
//for multi captures
public int GetCaptMove(int id)
//how many captures are there
public int GetCaptMove()
//source square
public int GetSrc()
//destination square (-1 for captured)
public int GetDst()
//calculate the move's grade
private double LandCheck(int src,int dest,Boolean whiteTurn, Boolean up, Boolean left,Boolean capt)
// check the grade for the back branches (level 2)
private double BackSideCheck(int sourceid, int sqb, int sqbid, int sqf, int sqfid, int sqff, ArrayList contCapts, ArrayList counterCapts, Boolean isKing, Boolean capt, int captid, Boolean left, Boolean whiteTurn,Boolean skipBack)
// check the grade for the front branches (level 2)
private double FrontSideCheck(int dest, int sqf, int sqfid, int sqotherf, int sqb, int sqbid, ArrayList contCapts, ArrayList counterCapts, Boolean isKing, Boolean capt, int captid, Boolean left, Boolean whiteTurn)
// check a single sub branch (level 2.5)
private double NextTwoCheck(int srcid, Boolean isKing, Boolean whiteTurn, Boolean up, Boolean left)
// check next capture move's grade
private PCPMove NextCaptCheck(PCPBoard board,int src, int dst, int capt)
// process occupation value from -2 - 2 to 0-5 and add to affected squares list if it's not already there
private int ProCell(int id, Boolean whiteTurn)

//add changes/affected squares from move to this move (for multiple captures)
private double MergeNext(PCPMove move)
// check a double sub branch (level 2.5)
public double DZCheckThree(int src, Boolean whiteTurn, Boolean up, Boolean left, Boolean isKing)
// check for protection status in case of hostiles in level 2 (level 3)
public double SubNextCheck(int src, Boolean whiteTurn, Boolean up, Boolean left)
//return the moves' grade
public double GetScore()

PCPSTAY

public PCPStay(PCPGrades grades, PCPBoard board, int dst,int movId)
//how many friendlies does the stay protect?
public int GetProt()
//get a protected square by index
public int GetProtId(int i)
//source square index num
public int GetSrc()
//square's occupation status
private int GetSqOcp(int id)
//calculate the stay's grade
private double StayCheck(int dest, Boolean whiteTurn)
// check the grade for the back branches (level 2)
private double BackSideCheck(int sourceid, int sqb, int sqbid, int sqf, int sqfid, int sqff, ArrayList counterCapts, Boolean isKing, Boolean left, Boolean whiteTurn)
// check the grade for the front branches (level 2)
private double FrontSideCheck(int dest, int sqf, int sqfid, int sqotherf, int sqb, int sqbid, ArrayList counterCapts, Boolean isKing, Boolean left, Boolean whiteTurn)
// check a single sub branch (level 2.5)
private double NextTwoCheck(int srcid, Boolean isKing, Boolean whiteTurn, Boolean up, Boolean left)
// check next capture move's grade
private PCPMove NextCaptCheck(PCPBoard board, int src, int dst, int capt)
// process occupation value from -2 - 2 to 0-5 and add to affected squares list if it's not already there
private int ProCell(int id, Boolean whiteTurn)
// check a double sub branch (level 2.5)
public double DZCheckThree(int src, Boolean whiteTurn, Boolean up, Boolean left, Boolean isKing)
// check for protection status in case of hostiles in level 2 (level 3)
public double SubNextCheck(int src, Boolean whiteTurn, Boolean up, Boolean left)
//return the moves' grade
public double GetScore()

DEBUG

מתעלת את הנתונים לתוך קבצי טקסט עם אינדוקציה.

```
//tab down
public static void UnTab()
//pad string with spaces (for source titles)
private static string FormatSource(string str)
//print message in 1-3 files (debug, selected and thread) and pop it up in a messagebox
/* tab:   -3   tabs down, prints, tabs up
          -2   prints and tabs down
          -1   tabs down and prints
          0nothing
          1prints then tabs up
          3tabs up, prints, tabs down */
public static void Post(string source, string text, Boolean popup, Boolean output_to_selected, int tab)
```

PCPGRADES

מחלקת אריזה עבור סט הציונים ששחקן המחשב מתבסס עליהם.

```
//def,block,dz,alley,capture,edible,corner,row,free

public PCPGrades(double[] vals)
```

PCPNEXT

מחלקת אריזה זמנית עבור מהלך על לוח (כל הנתונים הדרושים ליצירת PCPMOVE) .

```
//def,block,dz,alley,capture,edible,corner,row,free

public PCPNext(PCPBoard board, int src, int dst, int capt)
```