****

**האוניברסיטה הפתוחה**

**המחלקה למתמטיקה ולמדעי המחשב**

סמינר בבינה מלאכותית

20372

בינה מלאכותית במשחקי מחשב

**העבודה הוכנה על-ידי:** אורי ליבהבר036864122

**בהדרכתה של** **ד"ר** הרמן מעינה

**תאריך ההגשה:** 14.09.14

[מבוא 3](#_Toc398206840)

[גבולות העבודה 4](#_Toc398206841)

[שיטות מחקר 4](#_Toc398206842)

[מוסכמות כתיבה 5](#_Toc398206843)

[הגדרת הסוכן 6](#_Toc398206844)

[מהי בינה מלאכותית 9](#_Toc398206845)

[התנהגות טובה – הקונספט הרציונלי 13](#_Toc398206846)

[סוכן מבוסס מודל 16](#_Toc398206847)

[אבני הבניין של הסוכן 18](#_Toc398206848)

[ארכיטקטורת סוכן בז'אנר משחקי FPS 29](#_Toc398206849)

[סיכום 39](#_Toc398206850)

[ביבליוגרפיה 40](#_Toc398206851)

# מבוא

אנו קוראים לעצמנוHomo Sapiens , משמעות השם בתרגום מלטינית הנו "האדם הנבון". אותה תבונה שמלווה את המין האנושי משחר האנושות עד כדי כך חשובה ומשמעותית עבורנו, שהשקענו אלפי שנים של הרהורים פילוסופים ומחקרים כדי להבין כיצדאנחנו חושבים כלומר, כיצד אוסף סופי וקטן יחסית של חומר ורקמות מסוגל לקלוט, להבין, לצפות ואף לנתב את התהליכים הקורים סביבנו בעולם – אשר הינו הרבה יותר גדול ומורכב מהווייתנו. תחום הבינה המלאכותית, **Artificial Intelligence**(להלן: **"AI**")הולך צעד אחד נוסף קדימה ומנסה להסיק כיצד אנו, יצורים תבונתיים, יכולים לעצב מכונות תבונתיות שישתוו ואף יעפילו על יכולותינו הקוגניטיביות.

רבים מהחוקרים בתחום רואים בחקר התחום ויישומיו כמעין מעשה בריאה - הפחת רוח חיים באוסף של עצמים דוממים, השזורים זה בזה באמצעות רכיבים אלקטרונים שונים. מעבר לאתגר הגדול ולסיפוק הרב שהינו נחלת החוקרים בתחום, להישגים שבו יש השפעות מרחיקות לכת על ירעה רחבה של תחומי מחקר. למשל, אלגוריתמים לניתוח לוגי שפותחו עבור תחום ה-Machine Learning (להלן: "**ML**") שהפכו לכלי עזר רבי עצמה בחקר המתמטיקה ונעשה בהם שימוש בענף לצורך אימות לוגי של הוכחות מתמטיות.

אציין שאין לפיתוחים בתחום מוגבלות לתחום המתמטי בלבד, כיום נעשה שימוש בבינה מלאכותית לכתיבת שירה, נהיגה בטוחה במכונית בדרך המהירה והקצרה ביותר ואפילו לצורך דיאגנוזה של מחלות ופיצוח הגנום האנושי.

אחד היישומים הנפוצים ביותר של AI שצובר תהודה ומודעות בקרב החברה המודרנית הנו השימוש ב-ML בתחום משחקי המחשב. אין עוד מדובר רק בז'אנרים ספציפיים ומוצרי נישה המוגבלים לבידור ושעשוע בלבד אלא גם לתחום הסימולציה למשל בתעשיות הביטחוניות, שמשקיעות מיליוני דולרים בייצור מוצרי הגנה ואבטחה שונים ושפועלים באופן אוטונומי, כמו רכבים בלתי מאוישים וסימולטורים המדמים משחקי מלחמה.

נישת פיתוח הבינה המלאכותית למשחקי מחשב "חיה" (באופן מסוים) בעולם טכני משלה. לחוקרים ולמפתחים העוסקים בדבר יש מילון מונחים מיוחד משלהם, כישורים מיוחדים ואתגרים ייחודיים לתחום. הדבר שהופך את האסכולה למרתקת במיוחד הוא העובדה שלמרות שהפיתוחים בה רבים ומגוונים ישנה עדיין סבירות גבוהה, שחוקר בתחום ימצא את עצמו מיישם פתרונות לבעיות שטרם נחקרו ושיישמו להם פתרונות גנריים.

למרות שנעשו בשנים האחרונות מאמצים רבים ליישר את הקו בין מתודות הפיתוח של התחום לבין תחום פיתוח התוכנה הקלאסי, שני התחומים עודם שונים זה מזה במהותם.

לדוגמה, בפיתוח אלגוריתם להתנהגות של יריב ממוחשב, קיים בדומה לאלגוריתם המפותח לעבודה מול מסד נתונים דגש רב על תחכום, יעילות ביצוע ואופטימליות בפתרון, אך בשונה ממנו, אלגוריתם הבינה המלאכותית עשוי להוסיף עוד שכבות רבות של ניתוח והצלבות ממספר מקורות ועשוי להסתפק לעתים בתשובות חלקיות או לא אופטימלית בכדי לייצר תוצאה פחות שלמה לוגית מתוך כוונה לחקות התנהגות אנושית שהיא לעתים כלל לא רציונלית ואינה מונעת משיקולים לוגיים קרים.

בעבודתי בחרתי להתמקד בתחום הבינה המלאכותית במשחקי מחשב מאחר שתחום זה חווה שינויים רבים במהלך השנים האחרונות ביחס ישיר לשיפורים ולהתקדמות בעולם המחשוב בכללותו. ככל שהמחשבים מתפתחים וצוברים יותר תחכום וכוח חישובי, כך עולות הדרישות למורכבות רבה יותר מתחום זה. הרעיונות וההמצאות שנרקמו עוברים בדרך כלל הסבה מהירה לעולם המסחרי וחברות משחקים נחשבות כחממות של יצירה לתחום הבינה המלאכותית.

בין ההישגים הבולטים בענף ניתן לכלול את פרויקט **"כחול עמוק"**[[1]](#footnote-1)של חברת **IBM** ופיתוח אלגוריתם  **Beta Pruning** **Alpha** ועוד רבים אחרים. ללא ספק בינה מלאכותית רלוונטית למגוון עצום של יישומים אינטלקטואלים ומהווה תחום מרתק ללימוד, לחקר ויישום. אני כולי תקווה שהקורא ימצא בעבודתי עניין שיפתח בפניו צוהר לעולם מרתק ההולך וגדל והמשפיע בהווה ובעתיד על חיינו באופן שלא חלמנו עליו.  
עוד אציע בעבודתי ארכיטקטורה לסוכן במשחקי מחשב מסוג FPS, שתשען על בסיס אלגוריתמי שיפורט בהמשך.

# גבולות העבודה

* העבודה לא תעסוק באלגוריתמים לבחירת אסטרטגיה בתנאי יריבות (שמהווים חלק נכבד מכלל האלגוריתמים המייצגים את התחום) וזאת משום שישנם יישומים רבים של בינה מלאכותית שאינם דורשים התמודדות עם תחרות היות שמרחב המצבים שבהם הם פועלים אינו ידוע מראש. ידיעת מצב העולם במלואו הכרחית ליישום כלל האלגוריתמים הקלסיים המטפלים בתנאי יריבות. כך שלמעשה, היות שלאוסף האלגוריתמים אותם נחקור בעבודתי אין ידע שלם על העולם אלא רק ידע חלקי המבוסס על הסקה לוגית וחיפוש במרחב המצבים, אלגוריתמים אלו אינם רלוונטיים לצורך המחקר שאעלה מעל דפים אלו (למעט אזכורים בהמשך) ולכן לא נעסוק בהם אלא רק נציין את קיומם ונפנה לקריאה נוספת במידת הצורך.
* טפח נוסף בו לא יעסוק המחקר יהיה התאוריה הענפה שעל גבה נבנה תחום הלוגיקה המתמטית היות  
  שקצרה היריעה מלהכיל את כל ההוכחות (שהן במהותן באמת ובתמים מרתקות, אולם, ברובן  
  ארוכות וסוטות מהנושא המרכזי שבבסיס עבודתי). לצורך הבנה מעמיקה יותר של הנקודות שיועלו לדיון בין השורות אספק רקע תאורתי ממוקד שיתמוך בהבנת הטיעונים והנימוקים שאציג.

# שיטות המחקר

**בעבודה זו אציג כל נושא מחקר דרך שני מישורים:**

## המישור הטכני:

מישור זה יעסוק בעיקר בהצגת בעיה חישובית מתחום מדעי המחשב או תחומים ראליים אחרים והפתרון אלגוריתמי של אותה בעיה, עם דגש נרחב על הרקע התיאורית שהוביל לפיתוח אותו פתרון.

## המישור התבונתי:

מישור זה יעסוק בעיקר בהכללה של הבעיה לספקטרום נרחב יותר של דילמות אנושיות וביצירת ההקבלה בין תהליך קבלת ההחלטות מבוסס האינטואיציה האנושית שבבסיס הפתרון האנושי הטבעי לבין זה שמוטמע בבסיס הקוד הלוגי שמוביל לפתרון הטכני.

# מוסכמות כתיבה

## **המוסכמות בהן אעשה שימוש לאורך העבודה**

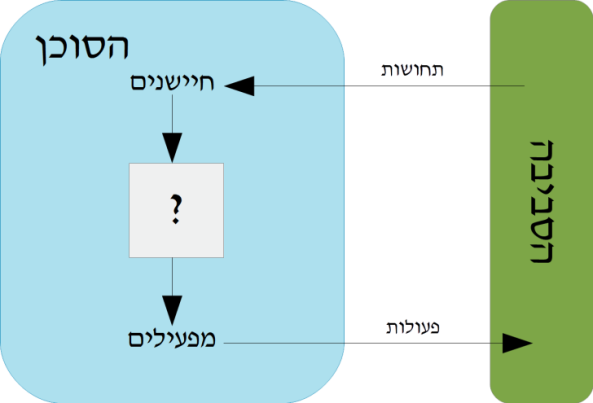
1. בעבודה אעשה שימוש בדיאגרמות UML. ההנחה היא שהקורא בקיא בפירוש האלמנטים השונים שמציגה דיאגרמה כזו.
2. קוד הכנה (**פסאודו קוד**[[2]](#footnote-2)**)** יכתב באנגלית או בעברית כאשר העדיפות תהיה להצגה הברורה ביותר.
3. רקע מתמטי שיהיה רלוונטי לצורך הוכחת נכונות אלגוריתמית או אינטואיטיבית, יינתן באופן  
   המתומצת ביותר וזאת כדי לא לגרוע מפשטות הצגת הפתרון ועל מנת שלא לסטות מהנושא שעל הפרק.
4. כל קיצור כתיב למושג טכני שאעשה בו שימוש, יופיע תחילה בשמו המלא בטרם השימוש הראשוני  
   בו בהמשך הכתיבה, וכמו כן, אם יבחר הקורא לעיין בעבודה בפורמט דיגיטלי  
   (PDF) כל הקלקה על מושג טכני שיצוין, תגלול את המסמך למקום בו אותו מושג טכני הוגדר  
   לראשונה.

# הגדרת הסוכן

בלב ליבו של הדיון בבינה מלאכותית ממוחשבת ניצב מושג הסוכן (Agent). אנו נעשה שימוש נרחב במושג זה ולכן נגדיר אותו היטב ככל הניתן.

## הגדרה: סוכן הוא כל ישות המסוגלת לחוש את סביבתה בעזרת חיישנים ולפעול באותה סביבה בעזרת מפעילים.

### בכדי להמחיש את ההגדרה נביט בשרטוט הבא:



**בשרטוט: הסוכן קולט מידע מהסביבה בעזרת חיישנים ופועל עליה בעזרת מפעילים.**

* נשתמש במושג **תחושה** כדי לתאר את הקלט שהסוכן מקבל מסביבתו בזמן נתון.
* **סדרת תחושות** של הסוכן היא ההיסטוריה השלמה של כל התחושות שקלט מסביבתו.
* פונקציית החלטה מקבלת כקלט את סדרת התחושות, התחושה הנוכחית הנקלטת ואת ההערכה של הסוכן לגבי מצב הסביבה הנוכחי. פלט הפונקציה הוא פעולה שעל הסוכן לבצע.
* הפעולה מועברת אל המפעילים של הסוכן, ואלו מבצעים בתורם את הפעולה על הסביבה.

ניתן להמשיל בקלות את המבנה הבסיסי של הסוכן לאדם, נניח שאותו אדם מעוניין לחצות כביש.

### לאדם חמישה חושים מהם הוא מקבל תחושות, לדוגמה:

* העיניים – מביטות לצדדים בכדי לאתר מכונית קרבה.
* האוזניים – מאזינות לרחש מכונית חולפת.

### המידע מסביבתו של האדם (הכביש) מוזן למוחו ועובר עיבוד נוסף:

* האם המכונית קרובה מידי? ולא לעבור או שמא רחוקה דיה והחציה בטוחה?
* האם במצב של ראות לקויה (לדוגמה בלילה) רחש מכונית מתקרבת עשויי להעיד שהיא  
  נמצאת בקרבת מקום?
* המוח מקבל החלטה לבצע חציית המתבססת על הזיכרון ההיסטורי של האדם (למשל: ניסיונותיו הקודמים לחצות כביש זה) והמידע החדש שנקלט. בשלב הבא, האדם מורה לרגליו (המפעילים שלו) להתחיל לחצות את הכביש. השפעה אפשרית של פעולה זאת על הסביבה עשויה להיות שרכב המתקרב יאט את נסיעתו עד לכדי עצירה, בכך משפיע האדם על מצב סביבתו.

## שני הקריטריונים העיקרים שישמשו אותנו לצורך סיווג הסוכנים שנציג בהמשך הינם:

### תפיסת הסביבה של הסוכן/סוג הסביבה שבה פועל הסוכן:

* האם סביבת הסוכן היא עולם סגור וידוע לו באופן מוחלט? קרי, האם הסוכן מודע למצב העולם  
  סביבו בכל שלב של תהליך קבלת ההחלטות?
* או שהוא מודע באופן חלקי - אינו מודע לשינויים במצב העולם שהם מחוץ לטווח ההרגשה שלו, לדוגמה: האדם החוצה את הכביש לא יודע מה מצב התנועה בכל מעבר חצייה בכל כביש בארץ אלא רק את מצב מעבר החצייה אותו הוא מתעתד לחצות.
* האם הסוכן יודע בוודאות מה תהיה תוצאת כל פעולה שיחליט לבצע? (סביבה דטרמיניסטית).  
  או שאולי הסוכן עשוי להיכשל בביצוע הפעולה שבחר היות שאינו יכול לדעת בוודאות את השפעת פעולותיו על הסביבה (סביבה סטוכסטית).

### פונקציית קבלת ההחלטות/האופן שבו מתקבלת החלטה לנוכח התחושה הנוכחית של הסוכן:

* האם הסוכן רק מגיב באופן רפלקסיבי לכל תחושה?
* האם לסוכן יש **מטרת על** שמניעה אותו לבחור בחירה מסוימת לפעולה?
* האם הסוכן מסוגל להסיק מסקנות לגבי מצב העולם לנוכח סדרת התחושות שלו ובכך לייצר מידע שטרם נחשף באופן ישיר?
* מה מידת החופש שיש לסוכן לקבוע את פעולתו הבאה?
* האם בכל שכפול של מצב העולם יחליט הסוכן לבצע את אותה הפעולה?

השילוב בין שני הקריטריונים הללו (תפיסת הסביבה ופונקציית קבלת ההחלטות) הוא שיקבע האם סוכן ממוחשב מפגין התנהגות שאותה נכנה "בינתית".

בפרק הבא נרחיב על הגישות להגדרת המושג "בינה מלאכותית".

# מהי בינה מלאכותית

לכל אורך ההקדמה הרבינו לציין, שתחום הבינה המלאכותית מרתק למחקר ויישום, אולם טרם הגדרנו מהי למעשה "בינה מלאכותית". בהמשך פרק זה נציג את מכלול הדעות וההגדרות הנפוצות ביותר שמנסות להגדירה.

## 1. להתנהג באופן אנושי: גישת "מבחן טיורינג"

מבחן טיורינג (להלן: "**מ.ט**") שהוצע כמסווג על ידי המדען הבריטי אלן טיורינג[[3]](#footnote-3) (1950), עוצב באופן שיספק הגדרה אופרטיבית (התנהגותית) למושג האינטליגנציה המלאכותית. מחשב נחשב תבונתי לפי מ.ט במידה שחוקר אנושי, המציג בפניו מספר שאלות כתובות ובתגובה מקבל תשובות כתובות מהמחשב שנמצא בחדר אחר, אינו מסוגל להבדיל האם התשובות שמתקבלות מגיעות מאדם או ממחשב.

### כדי לצלוח את מבחן טיורינג, המחשב חייב להיות מסוגל ליישם את הדרישות הבאות:

* עיבוד והבנה של שפה טבעית – כדי לתקשר בהצלחה בכל שפה שתבחר לצורך הבחינה.
* ייצוג ידע – כדי לאחסן מידע שהוזן מראש או נתקבל דרך אמצעי הקלט.
* יכולת הסקת מסקנות – כדי להשתמש במידע שנצבר לצורך מענה על שאלות והסקת מסקנות חדשות הנובעות מידע שנרכש.
* למידה ממוכנת ([ML](#Machine Learning)) – על מנת שיוכל להסתגל לנסיבות חדשות ולאתר, לסווג ולנתח תבניות.

[מ.ט](#מבחן טיורינג) מונע במתכוון מגע ישיר בין החוקר האנושי לבין המחשב הנחקר, מאחר שהדמיה אנושית פיזית אינה  
הכרחית לצורך קביעת אינטליגנציה. אולם, קיימת גם גרסה נוספת של [מ.ט](#מבחן טיורינג) הנקראת "מבחן טיורינג השלם" (Total Turing Test) אשר כוללת ערוץ ווידאו בין החוקר האנושי לבין המחשב, בכדי שהחוקר יוכל לבחון את התפיסה החזותית של המחשב. כמו כן גם, כוללת צוהר המאפשר העברת חפצים בין החוקר האנושי לבין המחשב כדי לבחון את יכולותיו המוטוריות.

### בכדי לעבור את מבחן טיורינג השלם, המחשב חייב להיות מסוגל ליישם את הדרישות הבאות:

* ראייה ממוחשבת – כדי לעבד מידע וויזואלי
* רובוטיקה – כדי שיוכל המחשב לבצע מניפולציות על אובייקטים סביבו ולנוע במרחב.

ששת הדיסציפלינות הנתונות מהוות את מרבית אבני הבניין של ה--[AI](#Artificial Intelligence) כך שלמעשה, עיצב [אלן טיורינג](#אלן טיורינג) מבחן שעודו רלוונטי גם 63 שנים אחרי. למרות האמור, כיום חוקרים מקדישים פחות זמן ומשאבים בכדי לעבור  
את [מ.ט](#מבחן טיורינג) ומתמקדים יותר בחקר העקרונות הבסיסיים שמרכיבים בינה מתוך האמונה שאלו עוסקים בסיבתיות שמובילה להתנהגות אינטליגנטית. מוטיבציה לגישה זאת ניתן לשאוב מהעובדה שבמהלך חקר התעופה, הצליחו האחים רייט**[[4]](#footnote-4)** לייצר מטוס ראשון לאחר שזנחו את הניסיון להידמות לבעלי כנף ויישמו ידע באווירודינמיקה. גם כיום תחום התעופה אינו מגדיר מטוס כמכונה שמדמה את מעופו של הברווז באופן כזה שלצופה אנושי לא תהיה האפשרות להבדיל כלל האם זהו ברווז אמתי.

## 2. לחשוב באופן אנושי: גישת המודל הקוגניטיבי

אם ברצוננו להסיק שתוכנה מסוימת "חושבת" כמו בן אדם, חייבת להיות לנו דרך לקבוע כיצד בן אדם חושב.  
אנו נאלץ אם כן, לחקור את התהליכים הפנימים המתרחשים במהלך תהליך קבלת ההחלטות במוח האנושי.

### ישנם שלושה דרכים לבצע זאת:

1. התבוננות פנימית – ננסה לתפוס ולנתח את מחשבותינו בעת קבלת החלטות.
2. ניסויים פסיכולוגים – צפייה ובחינה של התנהגות אנושית בזמן ביצוע.
3. שיקוף מוח – על ידי שימוש במכונת [[5]](#footnote-5)MRI ניתן לצפות בפעילות המוח בזמן אמת ולהתחקות אחרי דפוסי הפעילות שלו.

ברגע שיהיה בידינו מידע מספק על פעילות המוח נוכל לתאר את פעילותו בעזרת תוכנת מחשב.  
אם התכנה תבצע קלט ופלט זהה לזה שמתבצע על ידי המוח אזי יהיה זה אינדיקטור טוב לכך שייתכן שהמנגנונים שהוטמעו באותה תכנית דומים ואולי אף זהים לאלו שמתפקדים במוח האדם. לדוגמה, אלן ניואל והרברט סימון [[6]](#footnote-6) אשר פתחו את מכונת "פותר הבעיות הכלליות" (**General Problem Solver**) בשנת 1961 שמו לעצמם למטרה לא רק לוודא שהמכונה פותרת נכון בעיות אלא נתנו דגש רב יותר לדמיון רב ככל האפשר בין שלבי הפתרון שהציגה המכונה לבין אלו שהציג אדם שנדרש לפתור את אותה הבעיה במקביל.

אציין שתחום מדעי ההכרה האנושית (**Cognitive Science**) מאחד בין מודלים ממוחשבים מעולם ה-[AI](#Artificial Intelligence) לבין טכניקות וניסוים מעולם הפסיכולוגיה, כדי לעצב מודלים מדויקים ככל הניתן שמתארים את המוח האנושי.  
תחום זה צומח כיום בקצב מהיר תודות להתקדמות בשני התחומים המפרים זה את זה.

## 3. לחשוב באופן הגיוני: גישת תורת הלוגיקה

על פי תיעוד, היה זה הפילוסוף היווני אריסטו[[7]](#footnote-7) שניסה לנסח מהו "תהליך נכון של הסקת מסקנות"  
שמשמעותו – יצירה פורמלית של מאין שרשרת מצבי תודעה המורכבת מחוליות של מידע מקדים (הבטחות)  
על עולמו של מסיק המסקנה והמידע המתווסף לעולם התודעה שלו, כתוצר, שתלוי באותו מידע מקדים  
שנכונותו מובטחת (מסקנות).

שרשרת התודעה שתתקבל משמשת להוכחת נכונות של ההליך המחשבתי שהוביל את המצב ההכרתי של מסיק המסקנות מנסיבות מסוימות, להכרה מחודשת של מצב העולם שבו הוא מתנהל, או במילים אחרות הסקת מסקנות "נכונות" עקב מידע נתון.

ההֶקֵּשׁים שנסח אריסטו מספקים תבנית פורמלית שבעזרתה ניתן בהינתן מידע מקדים נכון, תמיד להסיק מסקנות נכונות. לדוגמה: "דוד המלך הוא בן אדם", "כל בני האדם הם בני תמותה" ולכן נסיק ש-"דוד המלך הוא בן תמותה". עבודתו של אריסטו סללה את הדרך לתורת הלוגיקה (Logic).

הלוגיקנים (החוקרים והמיישמים את תורת הלוגיקה) של המאה ה-19, פיתחו כתיב מדויק לגבי שלל סוגים  
של עצמים בעולמם ועל היחסים בניהם וזאת בשונה מהמתמטיקאים הרגילים אשר ניסחו חוקים שהיו מוגבלים לעולם המספרים בלבד.

השפעת תורת הלוגיקה על עולם המחשוב אפשרה עוד בשנת 1965 את כתיבתן של תכניות מחשב המסוגלות (עקרונית) לפתור כל בעיה הניתנת לכתיבה בכתיב לוגי, למרות שבמקרה שאין פתרון הייתה התכנית  
נכנסת ללולאה אין סופית.

הלוגיקנים ה-"מסורתיים" העוסקים בתחום הבינה המלאכותית מקווים שפיתוח תכנות ואלגוריתמים  
המבוססים על העקרונות הלוגיים הטהורים בלבד יאפשרו לבסוף את פיתוחן של מערכות תבונתיות.

### ישנם שני מכשולים עיקריים לגישה זו:

* ראשית, הפיכת מידע שאינו פורמלי או רב משמעי לפורמט פורמלי, נוקשה וחד משמעי הנה משימה קשה יותר מכפי שנדמה ממבט ראשון, וקשה שבעתיים כאשר מדובר במידע שאמינותו אינה מובטחת באופן מוחלט.
* שנית, יש הבדל גדול בין פתרון בעיה "באופן עקרוני" לבין פתרונה בפועל. גם בעיות עם מספר קטן יחסית של עובדות (מאה לדוגמה) יכולות לכלות את כל משאבי הזיכרון של מחשב מודרני, אלא אם, יסופקו למחשב מבעוד מועד רמזים והדרכות לגבי אילו צעדי היגיון עליו לנסות תחילה.

למרות ששני מכשולים אלה חלים על כל ניסיון לבנות מערכות תבונתית חישובית, חשוב לציין שהם הופיעו לראשונה בענף הלוגיקה.

## 4. לחשוב באופן רציונלי: גישת הסוכן הרציונלי

**כל תכנת מחשב היא לבסוף קוד שמבצע פעולה כלשהי. אבל מסוכן בינתי (Agent או סוכן) יש ציפייה לעשות יותר: לפעול באופן אוטונומי, לחוש ולתפוס את הסביבה בה הוא פועל, להתמיד בפעולתו לאורך זמן ממושך, להסתגל לנסיבות משתנות ולחתור להשגת מטרותיו.**

סוכן רציונלי הוא כזה, שפועל כדי להשיג את התוצאה הטובה ביותר או כאשר קיימת אי ודאות, להשיג את התוצאה העדיפה ביותר לאור הנסיבות הקיימות.

החוקרים המאמצים את הגישה הזאת שמים דגש על יצירת סוכנים המסיקים מסקנות נכונות.  
הסקת מסקנות נכונות הינה לב ליבה של הגישה הרציונלית, מאחר שהיא מחקה את התנהגות האדם שפועל  
להשגת מטרותיו מתוך הגיון והסקת מסקנות הנובעות מתוצאת פעולותיו.

מצד השני, הסקת מסקנה נכונה לדרך פעולת הסוכן עשויה להוביל למבוי סתום מאחר שלא תמיד ניתן ללמוד מעובדות נתונות על מצבו האמתי של העולם ועל דרך הפעולה המיטבית. ישנו אם כן אלמנט גדול של חוסר וודאות הנובע מסופיות המידע המקדים והמסקנות הנובעות ממנו.

במצבים מסוימים, אין בהכרח "דבר נכון לעשות", אבל משהו עדיין חייב להיעשות.

ישנן גם דרכים נוספות לפעול באופן רציונלי שאינן בהכרח מערבות הסקה רציונלית, לדוגמה:

רתיעה מכווייה הנגרמת מנר דולק היא פעולת רפלקס שהינה בדרך כלל מוצלח יותר מפעולה איטית יותר המתקבלת לאחר דיון מעמיק.

**מכלול הכישורים הנדרשים למעבר מבחן טיורינג מאפשרים לסוכן לפעול באופן רציונלי. היכולת לייצג ידע והאפשרות לביצוע הסקה רציונלית מאפשרים לסוכנים להגיע להחלטות טובות.**

### ההסברים לכך פשוטים:

* אנחנו צריכים להיות מסוגלים ליצור משפטים המובנים בשפה טבעית כדי להסתדר בחברה מורכבת.
* אנחנו זקוקים לתהליך למידה לא רק בכדי לגבב לעצמנו אוסף עובדות, אלא גם משום שלמידה משפרת את היכולת שלנו לפעול באופן יעיל שתורם להשגת מטרותינו.

**לסוכן הגישה הרציונלית שני יתרונות מהותיים על פני סוכני הגישות האחרות:**

ראשית, הוא יותר כללי מאשר "סוכני תורת הלוגיקה" מאחר שהסקה לוגית היא רק אחת מיכולותיו!  
היקש נכון הוא רק אחד מכמה מנגנונים המאפשרים השגת רציונליות.

שנית, זה נוח יותר לעבד פיתוח מדעי/לוגי מאשר גישות המבוססות על התנהגות אנושית או מחשבה אנושית. נוסחה רציונלית סטנדרטית היא ייצור מתמטי מוגדר היטב וכללי לחלוטין.

סוכנים המדמים התנהגות אנושית, לעומת זאת, מותאמים היטב לסביבה אחת ספציפית ומוגדרת היטב כך שבמידה ונרצה לעשות בהם שימוש מחוץ לסביבתם המקורית, נאלץ להתקין בהם **מכלול ידע שלם** על סביבתם החדשה.

**הסוכן הרציונלי יזדקק רק לספר מצומצם של עובדות כלליות על עולמו בכדי להסיק מסקנות על כל סביבה שבה יפעל וגם יוכל לבצע פעולות שאינן בהכרח נגזרות של הסקת מסקנות מושלמת.**

# התנהגות טובה – הקונספט הרציונלי

**סוכן רציונלי תמיד בוחר לבצע את הפעולה הנכונה לנוכח המצב הנתון.**

מבחינה טכנית, המשמעות היא שסוכן תמיד יודע לבחור את הפעולה הטובה ביותר מבין שלל האופציות העומדות לפניו. בוודאי שלבצע את הדבר הנכון יותר טוב מלבצע את הדבר הלא נכון, אך מה זה אומר בעצם לבחור בפעולה הנכונה ביותר?

### **שאלה עתיקת יומין זו תענה בתשובה עתיקת יומין:**

**על ידי שקילת ההשלכות של ביצוע הפעולה הנבחרת.**

## השלכות:

מהן ההשלכות של פעולת הסוכן על סביבתו?

אחלק את ההשלכות לשני חלקים:

1. **השלכות ידועות**
2. **השלכות שאינן ידועות**

### **השלכות ידועות:**

כאשר הסוכן פועל בסביבה נראית לחלוטין, תוצאת הפעולה שבחר תהיה ידועה מראש והשינוי שיחול בסביבה יהיה צפוי וקבוע. לדוגמה: במשחק שחמט, תזוזת הכלים קבוע ומצב הלוח לאחר כל מהלך של השחקן צפוי להיות קבוע גם הוא.

### השלכות שאינן ידועות:

כאשר הסוכן פועל בסביבה הנראית חלקית, אין ערבון לכך שהפעולה שבחר בה הסוכן תניב בהכרח את  
התוצאה לה הוא ציפה. לדוגמה: במשחק מרוץ, לחיצה על הבלמים בזמן נסיעה על כביש מושלג עשויה לעצור את הרכב או לגרום לו להחליק ללא שליטה אל תעלה סמוכה. כלומר, אין לדעת בוודאות מה יהיה מצב הסביבה החדש שינבע מביצוע פעולת הסוכן.

## שקילת ההשלכות:

כיצד נתעדף תוצאת פעולה אחת על פני האחרת?

### נגדיר לסוכן מדד ביצועים:

מדד הביצועים היא פונקציה מתמטית שהקלט שלה מורכב מסדרת המצבים של הסביבה עד לנקודה נתונה.  
הפלט שלה הוא מספר המייצג את טיב סדרת המצבים הנתונה. סוכן רציונלי פועל כך שימקסם את תוחלת הערך של מדד הביצועים, בהינתן סדרת התחושות שקיבל עד כה.

כיצד נבחרת פעולה?

### נגדיר פונקציית בחירה:

פונקציית בחירה נבחרת לפי התאמתה משתי גישות דומיננטיות בעולם המשחקים:

1. **פונקציה עבור סוכנים מבוססי מטרה.**
2. **פונקציה עבור סוכנים מבוססי תועלת.**

### פונקציה עבור סוכנים מבוססי מטרה:

קביעה בינרית של כן ולא לשאלה, האם המצב העתידי הנובע מביצוע הפעולה הנבחרת הוא מצב המטרה?  
סוג זה של פונקציות נפוץ בעיקר במשחקים בהם הסביבה נראית לחלוטין והסוכן צריך לתכנן או לחפש  
את סדרת הפעולות שתביא לסדרת המצבים שבה המצב האחרון הוא מצב המטרה.

### פונקציה עבור סוכנים מבוססים תועלת:

פונקציות אלו מתאפיינות בהעמקה למשמעות התועלת המופקת מביצוע פעולה. הסוכן התועלת מעריך את הקרבה של המצב העתידי למצב המטרה וגם מתחשב בעלות מול תועלת לגבי מטרות הביניים שלו מבחינת חשיבות וסדר עדיפויות.

אמחיש שימוש אפשרי בעזרת הדוגמה הבאה:

נביט במשחק יריות פשוט. נגדיר מטרה נאיבית לאויב בינתי ממוחשב (להלן: "**א.ב.מ**") להיתקל בשחקן, להרוג אותו ולא להיהרג". לצורך כך נדרשות מספר יריות מרובה הלייזר שברשות ה-א.ב.מ. נניח שאין משמעות לתחמושת ושהרובה יכול לירות תמיד. הסוכן נע בסביבה במטרה להיתקל בשחקן. בהיתקלו בשחקן, ינסה ה-א.ב.מ לירות לכיוונו ללא לאות בכדי להרגו, אולם, השחק המיומן מצליח לירות ולפגוע באופן אנוש ב-א.ב.מ.

מאחר ומטרת ה-א.ב.מ היא נאיבית ה-א.ב.מ ימשיך לנסות ולפגוע בשחקן היות שהוא משוכנע שפגיעה  
נוספת בו תוביל אותו למצב המטרה הנכסף! אולם, מאחר שהשחקן מיומן יותר, הוא מצליח לפגוע ב-א.ב.מ, להורגו ולהביאו למצב שנכשל במשימתו. לעומת זאת, באם נגדיר פונקציית תועלת ל-א.ב.מ, הוא יוכל להבחין בקלות שהתועלת שלו ממצב של עימות חזיתי מול השחקן תהיה נמוכה ותוביל למותו.

לכן, יעדיף ה-א.ב.מ להיתקל בשחקן בנסיבות בהן ידו על העליונה, כשמצב בריאותו אינו קריטי.

## בחירת מדד הביצועים:

כיצד נקבעת פונקציית מדד הביצועים?

התשובה לכך תלויה מאוד בהגדרת מרחב המצבים של הסוכן. לדוגמה, עבור סוכן שהוא "רוח רפאים" במשחק "פאקמן" ההגדרה של ביצועים טובים עשויה להיות, מספר הנקודות שנשארו במסך לאחר שפאקמן נפגש עם הסוכן ונפסל. או אולי מספר הנקודות שנשארו במסך חלקי סה"כ מספר הפעמים שנאכל הסוכן על ידי פאקמן.

באופן כללי, רצוי לקבוע לסוכן מדד ביצועים שמתווה את מצב הסביבה האופטימלי למטרתו. באופן זה, הסוכן ישאף תמיד למקסם את הרווח שלו מסדרת המצבים שנובעת מסדרת פעולותיו.



**התמונה נלקחה מהמשחק F.E.A.R פיתוח של חברת Monolith Productions.**

**בתמונה הקטנה מימין למעלה**: לפני ביצוע סדרת פעולות התקיפה בידי הסוכנים.

**בתמונה הגדולה**: לאחר התקיפה, מצבו של השחקן הפך לפחות אופטימלי ביחס ישר למצב עולמו של הסוכן.

# סוכן מבוסס מודל

**בכדי לענות על הדרישות העולות מהגישה הרציונלית ובפרט לטפל בתמונה חלקית של העולם, יש לעקוב אחר חלקי העולם שאינם גלויים לסוכן ברגע מסוים.**

הסוכן נדרש לתחזק ולשמור על מצב פנימי שתלוי בהיסטוריית התחושות שלו שנאספו עד כה ומשקפות לפחות  
חלק מהמאפיינים הלא גלויים של המצב הנוכחי .

למשל, במשחק נהיגה, כאשר המכונית הנוסעת לפני הסוכן בולמת לפתע, הסוכן יצטרך להבחין בכך ולבלום גם הוא. המצב הפנימי שעשוי לספק את הסוכן הוא שמירה על מצב נורות הבלימה של הרכב שנוסע לפניו, כך שיוכל להבחין בקלות מתי נורות הבלימה נדלקו ולבלום בתגובה.

או למשל, אם הסוכן מעוניין להחליף נתיב, די לו שישמור על מיקומם של המכוניות שנראו עד אותו רגע במראה האחורית שלו בכדי לתמוך בהשערה שבסבירות גבוהה יוכל להחליף נתיב בבטחה.

### בכדי לדעת כיצד לעדכן את המצב הפנימי כחלק מן התכנית של הסוכן נדרש ידע על שני גורמים:

* כיצד העולם החיצוני מתפתח בלי תלות בפעולותיו של הסוכן.

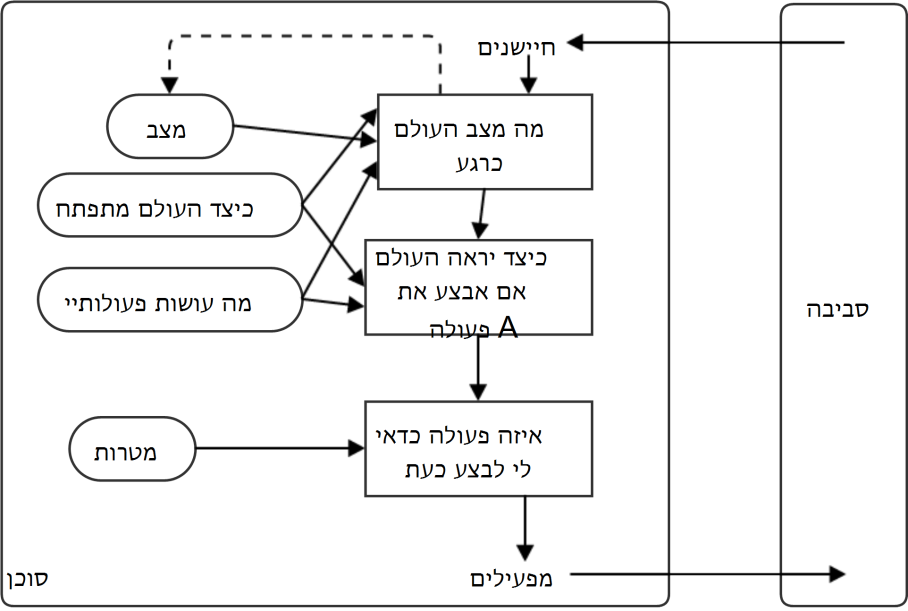
לדוגמה : רכב מהיר יותר שנוסע אחרי רכבו של הסוכן, יהיה קרוב אליו יותר ויותר בכל רגע נתון.

* כיצד הפעולות של הסוכן משפיעות על מצב העולם?

לדוגמה : אם הסוכן מסובב את ההגה עם כיוון השעון , המכונית נעה ימינה .

* הידע על האופן שבו העולם פועל נקרא **המודל של העולם.**
* סוכנים המכילים את המנגנון שתיארנו נקראים **סוכנים מבוססי-מודל**.

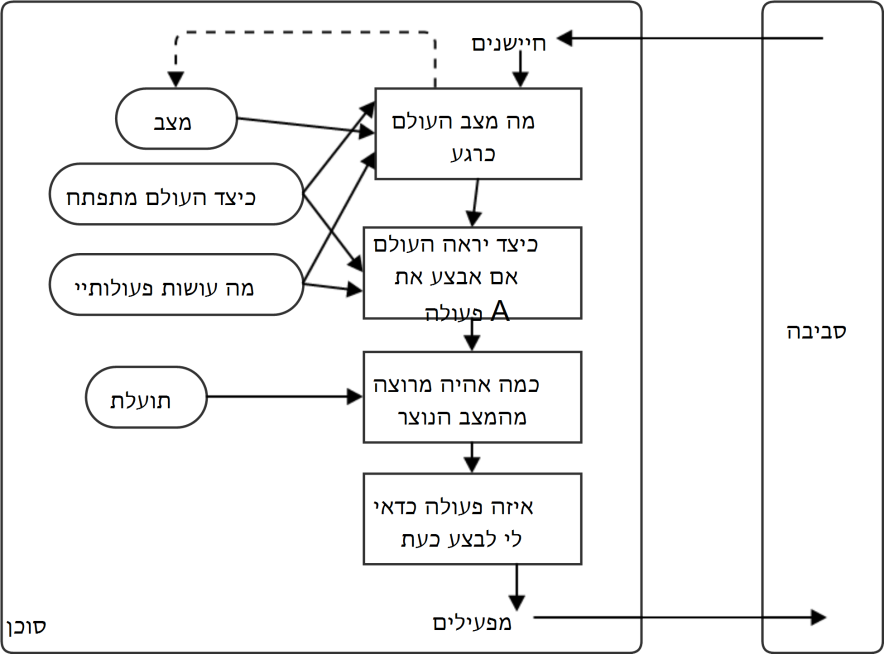
### מבנה הסוכן הרציונלי מבוסס המודל ומונחה מטרה, ניתן לתיאור סכמתי כך:



### מספר הבהרות הנוגעות לשרטוט:

1. **מצב:** משמעותו, מה המצב הפנימי של הסוכן בכל רגע נתון. לדוגמה, מצב הרוח של הסוכן: אגרסיבי, נינוח, פוחד...
2. **כיצד העולם מתפתח**: מה עשויות להיות ההשלכות של פעולות שחקנים אחרים על מצב העולם.
3. **מה עושות פעולותיי**: מה השינויים הצפויים במצב העולם עקב ביצוע פעולות הסוכן.
4. **מטרות**: מהן המטרות אותן שואף הסוכן להשיג.
5. **איזה פעולה כדי לי לבצע כעת**: פונקציית בחירת הפעולה.

### מבנה הסוכן הרציונלי מבוסס המודל ומונחה תועלת, ניתן לתיאור סכמתי כך:



מספר הבהרות הנוגעות לשרטוט:

יש לשים לב שהתווספו לסרטוט הקודם שני אלמנטים:

1. **תועלת**: פונקציית התועלת של הסוכן.
2. **כמה אהיה מרוצה מהמצב הנוצר**: תוחלת התועלת של סדרת המצבים הקיימת של הסביבה בתוספת המצב הנשקל להוספה.

**גישת הסוכן התועלתי נמצאת בשימוש נרחב בתעשיית משחקי המחשב, נביט בתמונת מסך של מבט מפתח הבינה המלאכותית במשחק "Kill Zone" של חברת "Guerrilla Games":**



כפי שניתן לראות בברור בתמונה השמאלית, כדי לקבוע את מסלול התנועה של סוכן ה-AI (החץ הכתום) מנוע המשחק מעריך את התועלת במעבר הסוכן שלו לכל משבצת **שנראית עבורו** במסך ובכך מתקבלת תנועה מבוקרת ורציונלית של הסוכן על פי מפת המשחק. החץ הירוק מצביע על המיקום בעל ערך התועלת הגבוהה ביותר, לשם יבחר הסוכן לנוע. התמונה הימנית מציגה את נקודת מבטו של השחקן האנושי על המתרחש במסך, הרחק ממערכת השיקולים של מנוע הבינה המלאכותית.

# אבני הבניין של הסוכן

### בכדי לספק בסיס לדיון, נציג תחילה מספר בעיות קלסיות מתחום הבינה המלאכותית ונתאר את פתרונן:

## בעיית מציאת המסלול הקצר ביותר

**תיאור הבעיה:** נתון גרף G המורכב מקבוצה של צמתים V וקבוצה של קשתות E המחברות בין זוגות צמתים  
מ-V, כך שלכל קשת ערך מספרי המציין את העלות של בחירת אותה קשת למסלול כלשהו בגרף.

בהינתן צומת מקור וצומת מטרה, נתבקש למצוא את המסלול ה"קצר ביותר" המחבר (אם בכלל קיים מסלול כזה) בין זוג הצמתים. ערכו של מסלול מוגדר כסכום ערכי הקשתות המרכיבות אותו. המסלול הקצר ביותר בין זוג צמתים מוגדר ככזה שערכו הוא הנמוך ביותר מבין כל המסלולים האפשריים בין אותו זוג הצמתים בגרף.

* אין עלות שלילית לקשת בגרף.
* ייתכנו בגרף מעגלים.
* אין הכרח שקיים מסלול בין כל זוג צמתים בגרף.

### מוטיבציה:

קל לראות שפתרון לבעיה ישים למגוון בעיות אנושיות יום יומיות, כגון:

* **מציאת מסלול הנסיעה הקצר ביותר או המהיר ביותר מהבית לעבודה:**

משקל הקשתות משול לזמן הנסיעה בכביש מסוים במערכת הכבישים העירונית.

כל צומת בגרף משולה לצמתים המחברים בין כבישים.

* **מציאת מסלול ההליכה השלו ביותר בפארק:**

משקל הקשתות נמוך יותר ככל שמספר הצועדים בו נמוך יותר.

כל צומת מייצגת פרשת דרכים בין שבילי הפארק.

**פתרון:**

אלגוריתם A\*:

אלגוריתם A\* (להלן: "**A\***") הוא אלגוריתם מסוג **Best-First search** ומוצא את המסלול הזול ביותר מצומת המקור לאחד מצמתי היעד. כאשר A\* עובר על הגרף הוא מנסה לפתח בכל פעם את המסלול שנראה הזול ביותר. **הוא ממש זאת בעזרת שימוש בתור עדיפויות ממוי**ן. האלגוריתם משתמש בפונקציית עלות המשלבת היוריסטיקה בצירוף מידע ידוע בכדי לבחור איזה צומת לפתח. פונקציית העלות היא הסכום של שתי הפונקציות:

כדי שהאלגוריתם יחזיר פתרון אופטימלי, פונקציית ההיוריסטיקה[[8]](#footnote-8), , צריכה להיות אדמיסיבלית (קבילה), כלומר לא יותר יקרה מהמרחק האמתי לצומת היעד. בניתוב היא יכולה להיות מרחק אווירי, מכיוון שהוא מהווה את המרחק המינימלי בין כל שני צמתים. בנוסף אם ההיוריסטיקה מונוטונית (קונסיסטנטיות), כלומר מקיימת את התנאי לכל קשת (x,y) בגרף (d מסמן את אורך הקשת), האלגוריתם ירוץ מהר יותר מכיוון שאין צורך לבדוק בשנית צמתים בהם האלגוריתם כבר עבר.)

אפשר להסתכל על האלגוריתם כהרחבה של אלגוריתם דייקסטרה[[9]](#footnote-9) עם היוריסטיקה (בדייקסטרה הפונקציה ההיוריסטית תהיה תמיד שווה ל-0).

**נביט בפסאודוקוד של האלגוריתם:**

**מספר הבהרות:**

* פונקציית העזר **reconstruct\_path** משמשת לבנייה של המסלול הסופי מהצומת ההתחלתית אל צומת המטרה. היא מבצעת זאת באופן רקורסיבי על ידי שימוש ב-**טבלת גיבוב** [[10]](#footnote-10) **came\_from**, שמאפשרת לאחזר את מצב העולם הקודם לכל מצב נתון מבין המצבים שנחקרו.

כל שימוש במפה נעשה בסיבוכיות זמן O(1).

* זמן הריצה הכללי עבור האלגוריתם הוא O(|E|) כאשר |E| מסמן את מספר הקשתות בגרף החיפוש.

**נביט בדוגמת ריצה של האלגוריתם על גרף פשוט:**

הערך היוריסטי y של צומת x מסומן בשרטוט כ- . צומת ההתחלה צבועה בירוק, צומת המטרה צבועה בכחול. חץ כתום יסמן את הצומת current שנחקרת בשלב הנתון.

|  |  |  |
| --- | --- | --- |
| **מס"ד** | **גרף** | **תיאור הגרף** |
| 1 |  | F(a) הוא הנמוך ביותר ולכן נבחר להרחיבו |
| 2 |  | F(b) הוא הנמוך ביותר ולכן יורחב |
| 3 |  | F(d) הוא הנמוך ביותר בopen\_list ולכן נבחר להרחיב אותו על פני c (שנכנס גם לopen\_list) |
| 4 |  | F(e) הוא הערך הנמוך ביותר בopen\_list ולכן הוא יבחר להפתח ויוביל לבסוף לצומת המטרה |

האלגוריתם A\* בגרסתו הגנרית כפי שהובא כאן, משמש כאבן יסוד לפתרון בעיות מתחום הAI על ידי חיפוש במרחב מצבים.

## פתרון בעיות על ידי חיפוש במרחב המצבים

המתכון הכללי להפעלת A\* כאלגוריתם חיפוש במרחב מצבים מתבסס על ההנחה שמרחב המצבים הינו  
סגור ובדיד וכמו כן, שעולמו של הסוכן דטרמיניסטי.

נדגים את השימוש ב-A\* בהתייחס לבעיה הידוע, "בעיית הקניבלים והמיסיונרים":

על גדה אחת של נהר האמזונס נמצאים שלושה מסיונריים ושלושה קניבלים. כל השישה חייבים לחצות את הנהר ולהגיע לגדה השנייה בבטחה. ברשותם סירה אחת בלבד שיכולה להשיט בין הגדות הלוך ושוב לכל היותר שני אנשים. לכל הפחות אדם אחד חייב להיות נוכח על הסירה בכדי שתשוט.

במידה ומספר הקניבלים בגדה כלשהיא גדול ממספר המסיונריים, הקניבלים יאכלו את המיסיונרים באותה הגדה.

מהו סדר הפעלות שיש לבצע בכדי שכל ששת האנשים יחצו את הנהר מבלי שיאכלו מסיונריים כלל?

### הגדרת מרחב המצבים:

בכדי לייצג נאמנה את מצב העולם הגלוי בפני הסוכן, נעזר בייצוג ווקטורים (N'יה סדורה).  
מצב העולם מתואר כך:

**[Ml,Cl,B,Mr,Cr]**

* Ml – מספר המסיונריים בגדה השמאלית, תחום הערכים: 0-3.
* Cl – מספר הקניבלים בגדה השמאלית, תחום הערכים: 0-3.
* B – מיקום הסירה, תחום הערכים: L – הסירה בגדה השמאלית, R- הסירה בגדה הימנית.
* Mr – מספר המסיונריים בגדה הימנית, תחום הערכים: 0-3
* Cr – מספר הקניבלים בגדה הימנית, תחום הערכים: 0-3.

### פונקציית מטרה:

נגדיר פונקציית מטרה G שהקלט שלה הוא ווקטור מצב עולם לבעיה נתונה, כפונקציה בוליאנית שמחזירה True אם מצב העולם הנתון הוא מצב מטרה וFalse אחרת. מצב x הוא מצב מטרה עם **פונקציית המטרה G**, מקיימת .

### פונקציית מעברים:

פונקציית מעברים היא פונקציה שהקלט שלה הוא מצב עולם בעיה נתון (כלומר הווקטור שהוגדר לעיל), והפלט שלה הינו קבוצת מצבים עולם אפשריים שהם עוקבים למצב הנתון כקלט.

לדוגמה, עבור פונקציית המעברים T(X) לבעיית המסיונריים והקניבלים, והקלט: [3,3,L,0,0]  
מתקיים:

יש לשים לב שהושמט מראש כל מצב שאינו אפשרי מבחינת הגדרת הבעיה קרי כל מצב שבו שטו בסירה יותר מ-2 אנשים ופחות מאיש אחד, ומצבים בהם יש על גדה כלשהי יותר קניבלים ממסיונריים.

נגדיר את **מצב ההתחלה** של הבעיה, כמצב הראשוני הניתן לסוכן המחפש שממנו יש לייצר את **סדרת מצבי העולם העוקבים** שמהווים את פתרון הבעיה. **עבור שני מצבי עולם a,b, b נקרא מצב עוקב למצב a ביחס לפונקציית מעברים T**, אם מתקיים ש-. **סדרת מצבים עוקבים נקראת פתרון** לבעיה אם ורק אם המצב האחרון בסדרה הוא מצב מטרה.

### **נגדיר רדוקציה בכדי להתאים את אלגוריתם A\* לבעיית חיפוש במרחב מצבים:**

* כל צומת מתאימה באופן חד חד ערכי ועל למצב עולם.
* x הוא שכן של y אם ורק אם .
* לאלגוריתם מסופקת פונקציית מטרה שמגדירה את קבוצת מצבי המטרה.
* לאלגוריתם מסופקת פונקציית מעברים לקביעת מצבים עוקבים למצב נתון.

**נבצע את ההתאמות הנדרשות על A\*:**

**פונקציה:** neighbor\_nodes(node)  
הפונקציה מקבלת צומת ומחשבת בעזרת פונקציית המעברים רשימה של כל המצבים העוקבים האפשריים.

**פונקציה:** heuristic\_cost\_estimate(node,goal)  
הפונקציה מחשבת את הערך היוריסטי שמשקף את העלות המשוערת של סידרת הפתרון שזנבה הוא ה-node הנתון.

**טכניקה להגדרת היוריסטיקה קבילה:**

אין זה ברור כ"כ מהגדרת הבעיה כיצד לבחור יוריסטיקה שתהיה טובה מספיק וקבילה. מאחר שלא הגדרנו משקל שונה לפעולות בבעיה, משקלי כל הקשתות בגרף ההולך ונבנה יהיו זהים ושווים ל-1.

**כיצד אם כן, ניתן לבחור יוריסטיקה שתהווה ערך שובר שוויון לצורך הערכת המסלול המיטבי?**

טכניקה ידועה לקבלת יוריסטיקה קבילה לבעיה שממבט ראשון אין לה יוריסטיקה אוטומטית (כגון **מרחק מנהטן**[[11]](#footnote-11) או **מרחק אווירי**) היא לרכך את אילוצי הבעיה. כלומר, להתעלם מאילוצים החלים על עולם הבעיה, בכדי להעריך בקירוב מה יהיה משקל סדרת הפתרון שעשויה להבנות אילו היה מצב העולם כפי שניתן בבעיה המקורית, ניתן גם לבעיה דומה השונה במקצת.

**אציע שתי אופציות קבילות הנעזרות בטכניקה זאת:**

1. במקרה של בעיית המסיונריים והקניבלים, ניתן לדוגמה להגדיר בעיה חלשה יותר שבה אין חשש לחיי המסיונריים באם אלו נמצאים במיעוט לעומת הקניבלים באחת מהגדות. אם כן, הערך היוריסטי של מצב העולם הבא: עשוי להיות 8, נדגים את סדרת מצבים האפשריים בבעיה המרוככת:  
   כפי שניתן לראות, אורך סדרת הפתרון לבעיה הרכה החל מהמצב הנתון הוא 8.

ניתן לראות בנקל שהתעלמנו באלגנטיות מהאילוץ החל על הבעיה המקורית. לראייה, המצב הסגול בלתי אפשרי בבעיה המקורית: המסיונרי הבודד בגדה הימנית היה הופך במהירות לארוחה דשנה עבור שני הקניבלים הרעבים שבמחיצתו.

1. ניתן לרכך את הבעיה המקורית עוד יותר, לבעיה הרכה , ופונקציה יוריסטית מתאימה .  
   למשל, נתעלם מהאילוץ שהסירה חייבת להכיל לפחות אדם אחד בכדי לשוט בין גדות הנהר.

נביט בסדרת הוכחה אפשרית למצב ששימש אותנו ב-א':

כפי שניתן לראות, אורך סדרת הפתרון לבעיה הרכה החל מהמצב הנתון הוא 4.

1. מקבילות שני ערכי היוריסטיקה הנובעים מהבעיות המרוככות, ניתן לקבל התנהגות טובה עוד יותר על ידי שילוב ביניהם! נבצע זאת על ידי כך שנבחר את הערך היוריסטי הסופי המתאים למצב x **כמקסימאלי מבין שני הערכים**. כלומר, הפעלת הפונקציה היוריסטית הסופית H על הצומת x, תקיים:  
    .

לא נוכיח כאן באופן פורמלי שהיוריסטיקות שהצעתי קבילות, נסתפק באינטואיציה פשוטה האומרת:  
שכל סדרת פתרון לבעיה הקשה היא סדרת פתרון לבעיה הרכה, אך לא כל סדרת פתרון לבעיה הרכה היא סדרת פתרון לבעיה הקשה, כלומר, אם נגדיר כ-S את קבוצת כל סדרות הפתרון של הבעיה הרכה ונגדיר כ-H את קבוצת כל סדרות הפתרון לבעיה הקשה, נראה בוודאי שמחשבון עוצמות מתקיים (לפי ההבחנה האינטואיטיבית שציינו לעיל) ואפילו נעז ונקבע ש-, אזי אם נביט בסדרת הפתרון הקלה ביותר ב-H, אזי היא בוודאי נמצאת גם ב-S (ואולי אפילו ניתן לנפות ממנה פעולות שהן נגזרות של האילוץ המספרי החל בין המסיונריים לקניבלים או כזה שחל לגבי תפעול הסירה ולקבל סדרה קצרה עוד יותר שבוודאי נמצאת ב-S).

מנכונות ושלמות אלגוריתם A\*, הרי ש-A\* מוצא את סדרת ההוכחה הקלה ביותר תמיד (תחת ההנחות הנכונות על עולמו של הסוכן). ולכן משקלה של כל סדרת פתרון הקצרה ביותרP לבעיה הרכה שהסיפא שלה מתחילה מצומת x שהיא צומת פנימית בסדרת הפתרון לבעיה הקשה, תקיים תמיד ש- כש- ו מסמן את משקל הסיפא בסדרות הפתרון הקשה והרכה בהתאמה.

## כמה מילים על היוריסטיקה

בעמודים הקודמים הרבינו להשתמש במינוח היוריסטיקה מבלי שדנו בחשיבותו לתכנון אלגוריתמים לסוכנים בינתיים.

היוריסטיקה היא כלל חשיבה המבוסס על הגיון פשוט או אינטואיציה.

היוריסטיקה מציע דרך קלה ומהירה לקבלת החלטות במחיר של הקרבת רמת הדיוק של עלות ההשלכות.  
היוריסטיקה נעזרת לעיתים בידע קודם שנרכש בעבר בכדי להסיק מסקנות על מצבים היפותטיים בעתיד.

בני אדם משתמשים באינטואיציה ככלי יום יומי לקבלת החלטות, לדוגמה:

* ילד קטן יחפוץ לטעום פריט מזון מסוים אם יבחין שהוריו אוכלים אותו בהנאה למרות שמעולם  
  לא טעם אותו בעבר. זאת עפ"י מסקנה שאם אנשים שהוא סומך עליהם צורכים דבר מזון   
  מסוים, הרי שהסיכוי גבוהה שפריט המזון טעים ומזין עבורו.
* נהג מונית יבחר דרכים מסוימות לנסיעה בהתאם לשעה בה הוא נוסע, זאת מאחר והוא מעריך על סמך ניסיון קודם, שבשעות עומס, כביש כזה או אחר עשויים להיות פקוקים.

היות שלסוכן אין אינטואיציה אנושית, היוריסטיקה משמשת אותו כמנגנון תואם לצרכי קבלת החלטות.  
פונקציה היוריסטית טובה מאפשרת לסוכן לפעול באופן "נכון" ויעיל יותר לנוכח מצבי עולם שאינם דטרמיניסטיים.

**לדוגמה, חישוב היוריסטיקה של הכדאיות בחיפוש נשק על ידי הסוכן:**

k - מוגדר כקבוע שמשמש להתאמת התוצאה לתחום המספרים הרצוי.

WeaponStrength - הינו מספר בין 0-1 שמייצג את כמות נקודות החיים שמופחתת מהשחקן עקב פגיעה מהנשק.

Health - הינה כמות נקודות החיים של הסוכן ברגע נתון, נכניס אותו למשוואה משום שייתכן שהסוכן עשוי להינזק מירי השחקן במהלך הניסיון שלו להשיג את הנשק. ככל שבריאותו של הסוכן ירודה יותר, כך רצונו למצוא נשק פוחתת בהשוואה לרצונות למצוא ערכת עזרה ראשונה למשל.

DistanceToWeapon - הינו המרחק המוערך בין הסוכן לנשק.

**דוגמה לחישוב היוריסטיקה של הכדאיות בחיפוש תיק עזרה ראשונה על ידי הסוכן:**

## k משמש באותו תפקיד. ככל שבריאותו של הסוכן נמוכה יותר, ערך המונה יהיה גדול יותר והכדאיות תעלה. במקרה שהמרחק לתיק העזרה הראשונה גדול יותר, הכדאיות תפחת עקב הגדלת המכנה.

## בעיית התכנון

**תיאור הבעיה:**

בהינתן מערכת עם מצב התחלתי, מצב מטרה (תיאור המצב של העולם הרצוי) וקבוצת פעולות דטרמיניסטיות  
המאפשרות לסוכן לשנות את מצב העולם, יש למצוא סדרת פעולות שתביא את העולם מהמצב ההתחלתי למצב המספק את המטרה.

### **מוטיבציה:**

פתרון לבעיה ישים למגוון בעיות, כגון:

* **תכנון סדר הרכבה של רכב בפס ייצור:**

הרכבת רכב היא תהליך מובנה שכל שלב בו תלוי בתוצאת השלב שקודם לו.

לדוגמה, לא ניתן להרכיב דלת לרכב אם טרם נבנתה השלדה, או להתקין חלון בטרם נבנתה הדלת.

* **תזמון המראות ונחיתות מטוסים:**

לא ניתן להנחית מטוס על מסלול שמשמש מטוס אחר באותו הזמן להמראה.

מטוס לא יורשה להמריא בטרם נבדק ונמצא תקין על ידי צוות הקרקע.

**הבהרה:**

תכנון (Planning) וחיפוש (Search) הן גישות שונות לפתרון בעיות, למרות שבהרבה מקרים ניתן להשתמש בשתיהן כדי לפתור את אותן הבעיות.

ברם, כאשר מדובר בבעיות "גדולות", כלומר, כאלו שמרחב המצבים שלהן גדול, השיטות המשמשות לתכנון מאפשרות לפתור באופן יעיל בעיות שלא ניתן לפתור בעזרת שיטות חיפוש.

**נשתמש בשפה לייצוג בעיות תכנון קלאסי הנקראת:**

**P**lanning **D**omain **D**efinition **L**anguage = PDDL

**הרכיבים של משימת תכנון ב-PDDL הם:**

* **אובייקטים:** עצמים בעולם הרלבנטיים לבעיה.
* **פרדיקטים:** תכונות האובייקטים שבהם יש לנו עניין. ערכם אמת או שקר.
* **מצב התחלתי:** מצב העולם שבו מתחילים, מסומן S0.
* **מצב המטרה:** מה רוצים שיתקיים כמטרה (בהגדרה זו נציין רק את מה שמעניין אותנו), מסומן G.
* **פעולות/אופרטורים:** דרכים לשינוי מצב העולם, התוצאות של הפעלת הפעולות והאופרטורים.
* כל מצב מיוצג כקוניונקציה של אטומים קרקעיים (חיוביים) ואין בו פונקציות.
* כל מה שידוע בעולם כלול בתיאור המצב.  
  מה שלא ידוע (לא מוזכר) בעולם כאמת, מניחים שהוא שקר.
* פעולה משנה את מצב העולם:  
  PDDL מתארת את תוצאת ביצוע הפעולה בעזרת השינויים שנגרמו.  
  כל מה שנותר ללא שינוי, אינו מצוין בתוצאה.
* פתרון לבעיה הוא תכנית, סדרת פעולות שתביא אותנו מהמצב ההתחלתי למצב מטרה.
* **כל פעולה תיוצג ב-PDDL על-ידי סכימת פעולה הכוללת 3 חלקים:** 
  1. שם הפעולה והמשתנים (הפרמטרים) שבהם נעשה שימוש בסכימה.
  2. **התנאי המקדים PreCondition:**  
     זהו משפט המגדיר את התנאים שצריכים להתקיים (מה צריך להיות נכון בעולם) כדי  
     שהפעולה תוכל להתבצע. התנאי המקדים הוא קוניונקציה של ליטרלים חיוביים.  
     נאמר שהפעולה a יכולה להתבצע במצב s רק אם התנאי המקדים מסופק על- ידיs .
  3. **התוצאה Effect:**  
     התוצאה של ביצוע פעולה היא משפט המתאר באיזה אופן השתנה העולם כתוצאה  
     מביצוע הפעולה (מה מתקיים אחרי ביצוע הפעולה) ולו שני חלקים:
     + הפרדיקטים שיתווספו לעולם לאחר ביצוע הפעולה - **Added List**.
     + הפרדיקטים שיוסרו מהעולם לאחר ביצוע הפעולה - **Delete List**.
     + כלומר, התוצאה מגדירה מצב חדש המתקבל על- ידי מחיקת התנאים של   
       Delete List מ-s והוספת אלה שב-Add List ל-s.
     + Delete List, Add List והתוצאה הם **קוניונקציה של ליטרלים.**

כל משתנה המופיע בתוצאה חייב להופיע גם בתנאי המקדים.

**לדוגמה, בעיית התכנון הידועה "Dock Worker Crane" מנוסחת ב-PDDL:**

מפעיל מנוף העובד בנמל מתבקש לערום שלוש מכולות זאת על גבי זאת, כמגדל לפי סדר גודל. כאשר הגדולה ביותר בתחתית, הבינונית מעליה והקטנה ביותר בראש הערימה.

|  |  |
| --- | --- |
| מצב התחלה | מצב מטרה |
| C2  C1  C3 | C2  C1  C3 |

### **מצבים:**

### **פעולות:**

**מספר הבהרות:**

* אנו מניחים שלסוכן קיים מנגנון **"האחדה" (Unification)** בכדי שתתאפשר לו לבצע התאמות לשמות המשתנים שמופיעים בכותרות הפעולות.
* היינו יכולים להסתפק גם בשתי הפעולות Unstack,Stack ולהגדירן באופן כללי יותר ללא שימוש בליטרל הסופי ground כתוצאה של הפעולה stack, ולהחליפו במשתנה y.

הסיבה העיקרית להפרדה בין הפעולות היא כדי שסדרת הפעולות שבפלט מנגנון התכנון תהיה בעלת משמעות סינטקטית ברורה יותר.

**פתרון:**

## אלגוריתם חיפוש קדימה

**מספר הבהרות:**

* פעולה a נקראת applicable למצב s אם ורק אם כל התנאים המקדימים ב Precondition(a) הם נכונים במצב s.
* היא פונקציית מעברים במודל העולם, המשמעות של הביטוי: הוא: מצב העולם החדש הנובע מהפעלת הפעולה a על מצב s.
* קיים ונמצא בשימוש נרחב **אלגוריתם חיפוש אחורה (Recursive Backwards Chaining)** שמשמש גם הוא לתכנון ודומה במבנהו לאלגוריתם שהוצג. האלגוריתם חיפוש קדימה נבחר בשל אופן פעולתו הברור והאינטואיטיבי. אציין רק שבאלגוריתם החיפוש אחורה מתחילים ממצבי המטרה למצב ההתחלה ונעזרים בפונקציית המעברים ההופכית בכדי לקבל את מצבי העולם הקודמים האפשריים למצב s מסוים. בכך נמנע האלגוריתם מ"פיתוח" מצבי עולם שאינם רלוונטיים למטרתו.
* באלגוריתם נעשית בחירה שאינה דטרמיניסטית של פעולה a מתוך applicable.

מימוש הבחירה יכול להיעשות באמצעות מחולל רנדומלי של סביבת התוכנה או פשוט על ידי קביעה שרירותית תלוית מימוש של מבנה הנתונים בו נעשה שימוש למימוש הקבוצה applicable.

# ארכיטקטורת סוכן בז'אנר משחקי FPS

עד כה דנו באופן מופשט בקווים המנחים המשפיעים על עיצובו של הסוכן. הדגמנו בעיות קלאסיות מתחום הבינה המלאכותית והצגנו אלגוריתמים שמהווים את הבסיס המקובל לפתרונן. כעת, כשאנו מצוידים בכלים תאורטיים נאותים, נדון בארכיטקטורה הפרקטית של סוכן בז'אנר משחקי FPS.

נציג תחילה מודל פשוט של א.ב.מ, שאותו נשפר בהדרגה.

### סוכן מבוסס מכונת מצבים (להלן: "סכמה א'")

AGENT

NAV

KB

FSM

WORLD

**הסבר:**

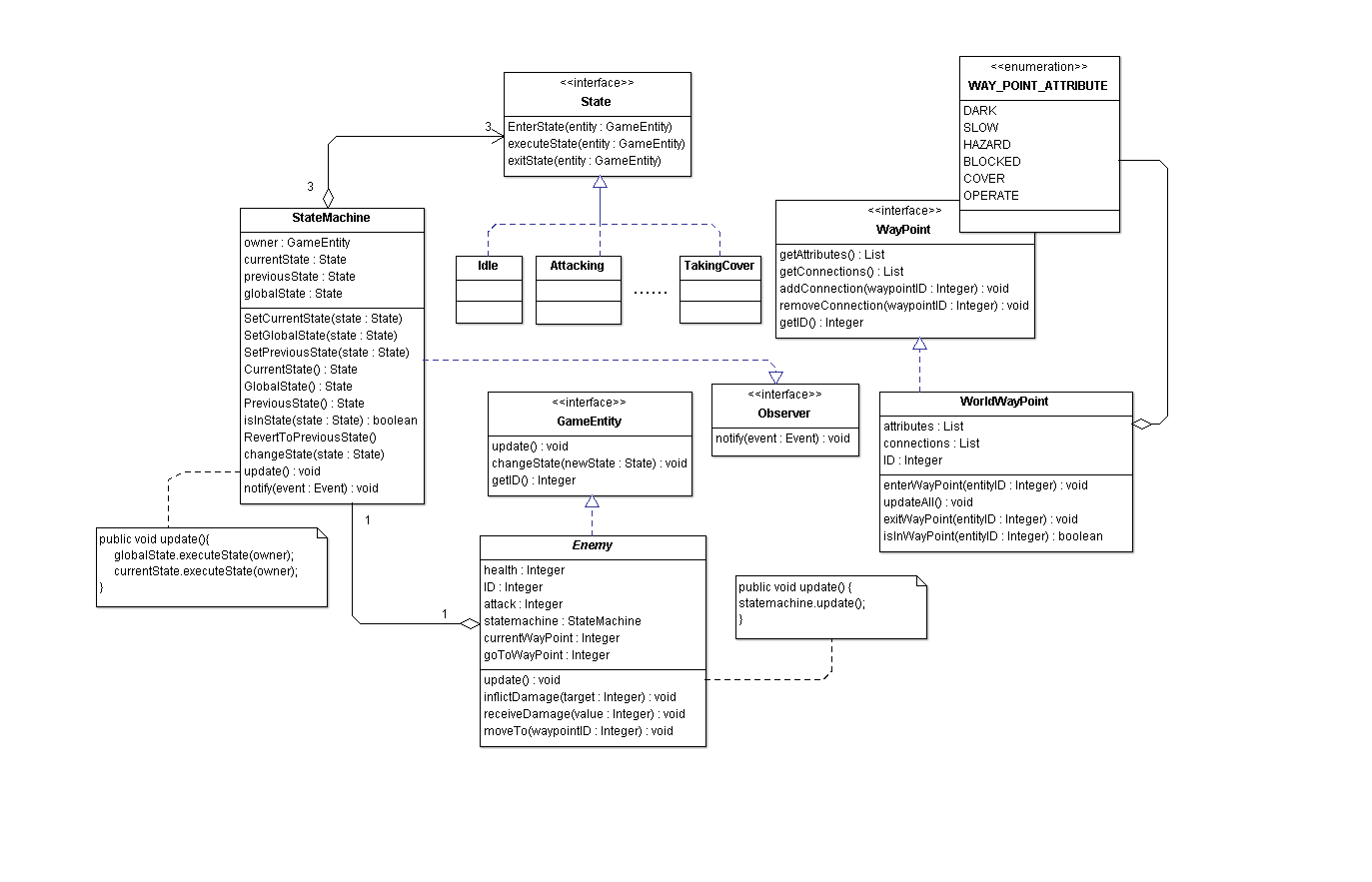
מבנה הסוכן מבוסס על מספר תבניות עיצוב (Design Patterns[[12]](#footnote-12)):

* Observer – משמש לקבלת עדכונים הקשורים לשינויים במצב העולם. העדכונים מסופקים לסוכן מהרכיב הלוגי האחראי לתחזוקת מצב העולם (WORLD).
* FSM - Finite State Machine, מכונת מצבים משמשת לעדכון מצבו הפנימי של הסוכן בכל רגע נתון.

הסוכן מכיל מכונת מצבים שמתארת בכל רגע נתון את מצבו הפנימי של הסוכן. מכונת המצבים מאחזרת מידע מרכיב ה-KB ((Knowledge Base שמכיל מידע מצטבר שיש לסוכן על העולם וכמו כן את הנתונים הפנימיים שקשורים לפעולת הסוכן.

רכיב נוסף הינו רכיב הניווט – NAV – Navigation, רכיב זה אחראי על תזוזת הסוכן בעולם.

### תרשים UML כללי של מבנה הסוכן מסכמה א':



**פרטים נוספים:**

כפי שניתן לראות מתרשים הUML כל סוכן במשחק יממש את המחלקה האבסטרקטית **Enemy** שמכילה מתודות כלליות להתקפה (**inflictDamage**) ופגיע (**recieveDamage**). לסוכן יש מתודה update שמבצעת delegation למתודה בעלת אותו שם במכונת המצבים של הסוכן.

**ה-State Machine של הסוכן מחזיק שלושה מצבים:**

1. **currentState** – המצב הנוכחי של הסוכן: מה הפעולה שהסוכן מבצע בהווה.
2. **previousState** – המצב הקודם של הסוכן: מה הפעולה האחרונה שביצע הסוכן לפני הפעולה הנוכחית.
3. **globalState** – המצב הגלובלי של הסוכן: מה הפעולה הכללית שהסוכן צריך לבצע בלי קשר לפעולתו  
   הנוכחית. הפעולה הגלובלית יכולה להיות מטרה כללית של הסוכן, כגון, לחפש את השחקן.

**כל State אפשרי של הסוכן מכיל שלוש מתודות:**

1. **enterState** – מה הסוכן צריך לעשות כשהוא עובר למצב הנתון. לדוגמה, אם הסוכן היה במצב **IDLE** והוא עובר למצב **TakingCover** אז הסוכן צריך להפעיל את אנימציית ההתכופפות שלו או לחילופין לחפש מסלול לWayPoint הקרוב ביותר שהוא מסוג **COVER**.
2. **executeState** – פעולה שוטפת שהסוכן צריך לבצע בהיותו במצב הנוכחי, לדוגמה, אם הסוכן נמצא בWayPoint מסוג **HAZARD** אז בכל עדכון של מצבו הוא צריך להפעיל את מתודת **ה-receiveDamage** שלו ולהפחית ממשתנה הhealth שלו את כמות הנזק שהוא סופג.
3. **exiteState** – כאשר הסוכן עובר ממצב הנוכחי למצב החדש, הוא צריך לעדכן את המשתנים שלו בהתאם או לבצע פעולות מסוימות. לדוגמה, אם הסוכן יוצא ממצב Idle ונכנס למצב Attacking אז ראשית הוא צריך לשלוף את נשקו.

מערכת הניווט של הסוכן ממומשת על ידי המתודה **moveTo**. בכדי להבין את אופן פעולתה, נרחיב כעת אודות המבנה של עולם המשחק וכיצד ניתן להתאימו לניווט יעיל של הסוכן. אנו נתמקד בשיטת הניווט השכיחה ביותר כיום: **ניווט בעזרת גרף way points.**

## ניווט מבוסס Way Points

כידוע, עולם המשחק התלת ממדי בנוי מ- **Polygons**ואלו בתורם מורכבים מ-**Faces** שתחומים בעזרת **Edges** שבקצותיהם **Vertices.**

**נמחיש את הרעיון בעזרת הסרטוט הבא:**

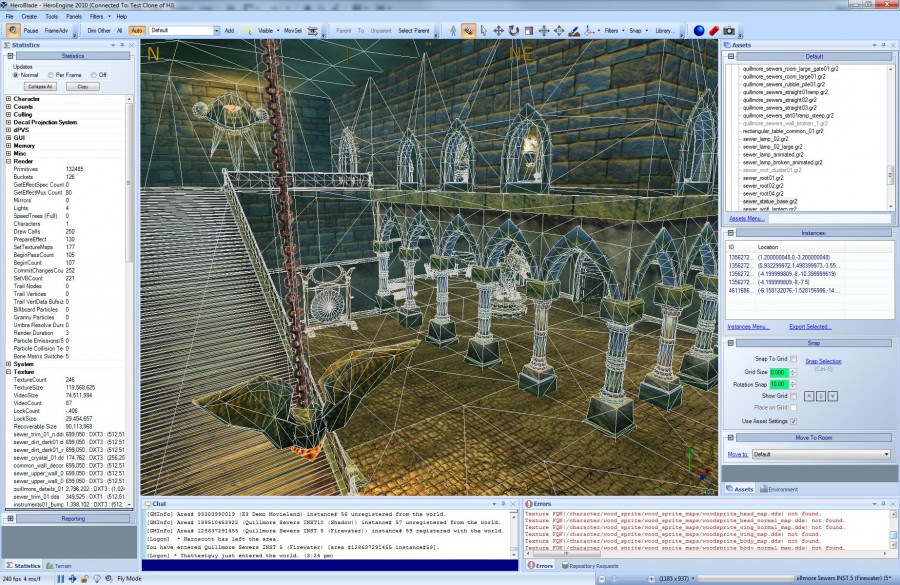
Vertex

Edge

Face

### בשרטוט ניתן לראות את יחידת המידה הקטנה ביותר במודל התלת ממדי שמוגדר להיות הקרקע בעולם המשחק.

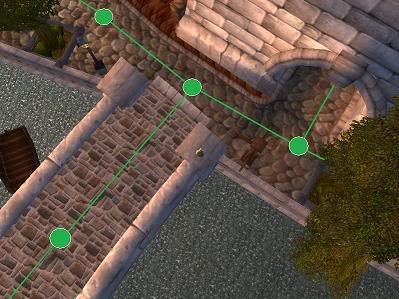
### **צירוף של אוסף יחידות כאלו מהווה משטח בעולם המשחק:**



Faces, Vertices and Edges

ניווט לפי Faces מורכב מאוד וכולל חישובים מתמטיים יקרים מבחינת זמן ריצה שעשויים להשפיע לרעה על זמן התגובה של המשחק ושל הסוכן. לעומת זאת, אילו היה הסוכן צריך לנווט רק בעזרת מעבר על Edges וVertices, הייתה משימתו פשוטה יותר וקלה לחישוב. בעת בניית המודל התלת ממדי של עולם המשחק, מכניס המפתח גרף ניווט עבור הסוכנים השונים שיאכלסו את עולם המשחק. כל צומת בגרף הניווט יקרא אז Way Point ויתאים במימושו לאובייקט Way Point כפי שהגדרנו בסכמה א'.

**המחשה:**



בתמונה ניתן לראות שבמפת המשחק משובצים Way Points (להלן: "**wp**"), שמשמשים לניווט של סוכנים.  
אזי, מאחר ולכל wp יש **attributes** שונים שמגדירים את סוגו, ניתן להתאים בנקל את מצב הסוכן לסוג הסביבה שבה הוא נמצא. לדוגמה, סוכן שנכנס ל-wp שמכיל attribute **DARK**, מכונת המצבים שלו תעבור למצב **TURN\_ON\_LIGHT** שיגרום לסוכן להדליק את הפנס שברשותו. ניווט בגרף ה-wp יכול להיעשות בעזרת שימוש באלגוריתם A\*, כאשר ניתן לאלץ את הסוכן להעדיף מסלולים מסוימים על ידי מניפולציה של מחירי הקשתות שמרכיבות את המסלול. סוכן שמותקף על ידי השחקן ושבריאותו נמוכה, יגרום למכונת המצבים שלו לעבור למצב **RUN\_FOR\_COVER** שיפעיל את מתודת **moveTo** עם wp מטרה שמכיל attribute מסוג **COVER**.



COVER location

Agent location

## מנגנון האירועים

מכונת המצבים של הסוכן מממשת את ממשק **Observer**.  
ממשק זה מאפשר לה לקבל אירועים או התראות מסביבת המשחק כגון:

* האם השחקן נמצא בטווח שמיעה?
* האם השחקן גלוי לסוכן?
* האם השחקן בטווח ירי?
* האם מצב העולם השתנה באופן כזה שמשנה את הglobalState של הסוכן? לדוגמה, השחקן פוצץ גשר ולכן כל הסוכנים בעולם המשחק יודעים כעת את מיקומו של השחקן וצרכים להגיע אליו ולהילחם בו.

כל GameEntity שנוצר, ירשם אם כן לסוגי ההתראות שהוא מעוניין לקבל מעולם המשחק, וכאשר זה האחרון יבצע שינוי שעשויי לעניין את הסוכנים הרשומים אצלו, הוא ישגר לכולם event עם הפרטים המתאימים.

עד כה ראינו שעיצוב הסוכן המוצע עונה על צרכינו, אולם זהו השלב לציין מספר בעיות מהותיות בעיצוב הנ"ל:

1. מבחינת ארכיטקטורה, העיצוב המוצע קשה לתחזוקה:
   1. ישנה תלות (coupling) חזקה בין מצבו הפנימי של הסוכן לבין הפעולות שהוא מבצע.  
      כלומר, בכדי להכניס התנהגויות חדשות של הסוכן, יש צורך לעדכן ולתחזק את כל המעברים האפשריים בין כל המצבים שקיימים כבר לבין כל המצבים החדשים!
   2. תלות זאת מסרבלת מאוד את תהליך הפיתוח ומגבילה את הסוכן לאוסף התנהגויות מובנות מראש ומותאמות רק למצבים מסוימים שנצפו על ידי המפתח.
   3. הסוכן שעיצבנו אינו מסוגל להתמודד עם מצבים חדשים שלא היו ידועים לו ומקודדים באופן קשיח  
      במכונת המצבים שלו.
2. הסוכן מגיב באופן עיוור לכל שינוי במצבו, לעיתים, אוסף הפעולות שמוכתב לו לביצוע על ידי מכונת המצבים יוביל אותו להפעיל עד תום פעולה חסרת תועלת או כזאת, שבוודאות אינה יכולה להתבצע כראוי.  
   סוכן עשויי להימצא במצב בו הוא השלים את כל סט הפעולות שהוכתבו לו על ידי מכונת המצבים שלו מבלי שיש לו פעולה או מטרה להמשיך אליה. במצב כזה לדוגמה, הסוכן יכול לעבור למצב Idle ולחבל בלוגיקת המשחק באופן בוטה ומגוחך:



Hello Player! Please don’t mind me!

I’m just standing here pointlessly.

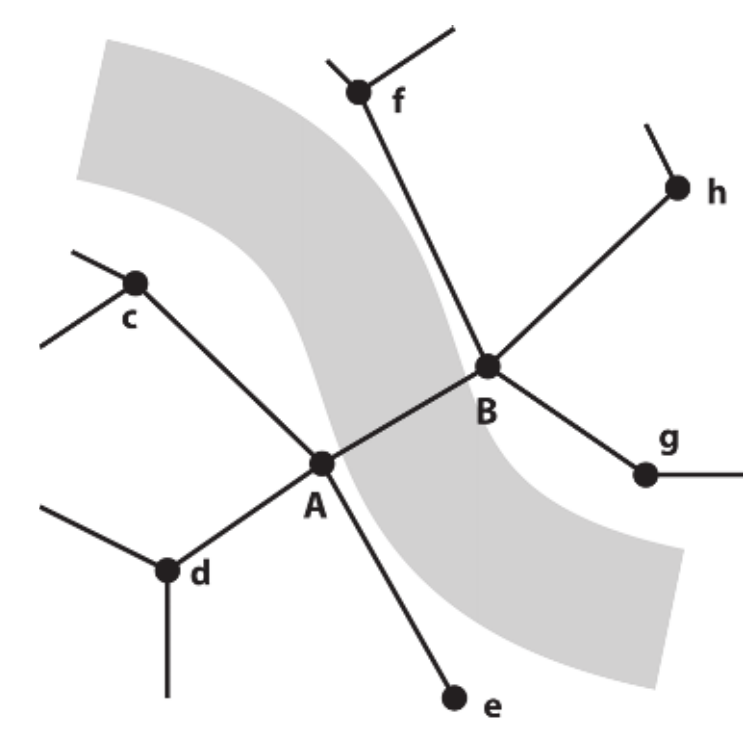
**תמונת מסך נלקחה מהמשחק Thief II: The Metal Age של חברת Looking Glass Studios.**

1. האופן בו מנווט הסוכן בעולם מאוד מוגבל וגורם ליצירת "נקודות מתות" בעולם המשחק. נקודה בעולם המשחק נחשבת "נקודה מתה" אם אין הסוכן מסוגל להגיב לשינויים שמתחוללים בה, לדוגמה:



ללא הכוונה מובנת בקוד מצד המפתח, הסוכן יתעלם לחלוטין מכל מה שקורה בנקודות הכתומות.

1. הצמדה של attribute ל-Way Point עלולה לגרום לסוכן לא לפעול כפי שציפינו לדוגמה:



אם בנקודה A יוגדר attribute מסוג SWIM אזי במידה והסוכן ינסה לנווט מהנקודה c לנקודה e דרך A

אנו עשויים למצוא את הסוכן שוחה לאורך הקשת (A,e) במקום לצעוד בה.

נציג כעת את הסכמה המתוקנת.

### **סוכן מתכנן ומכוון תועלת (להלן: "סכמה ב'")**

AGENT

PLANNER

KB

WORLD

NAV

**GOALS**

**FSM**

**הסבר:**

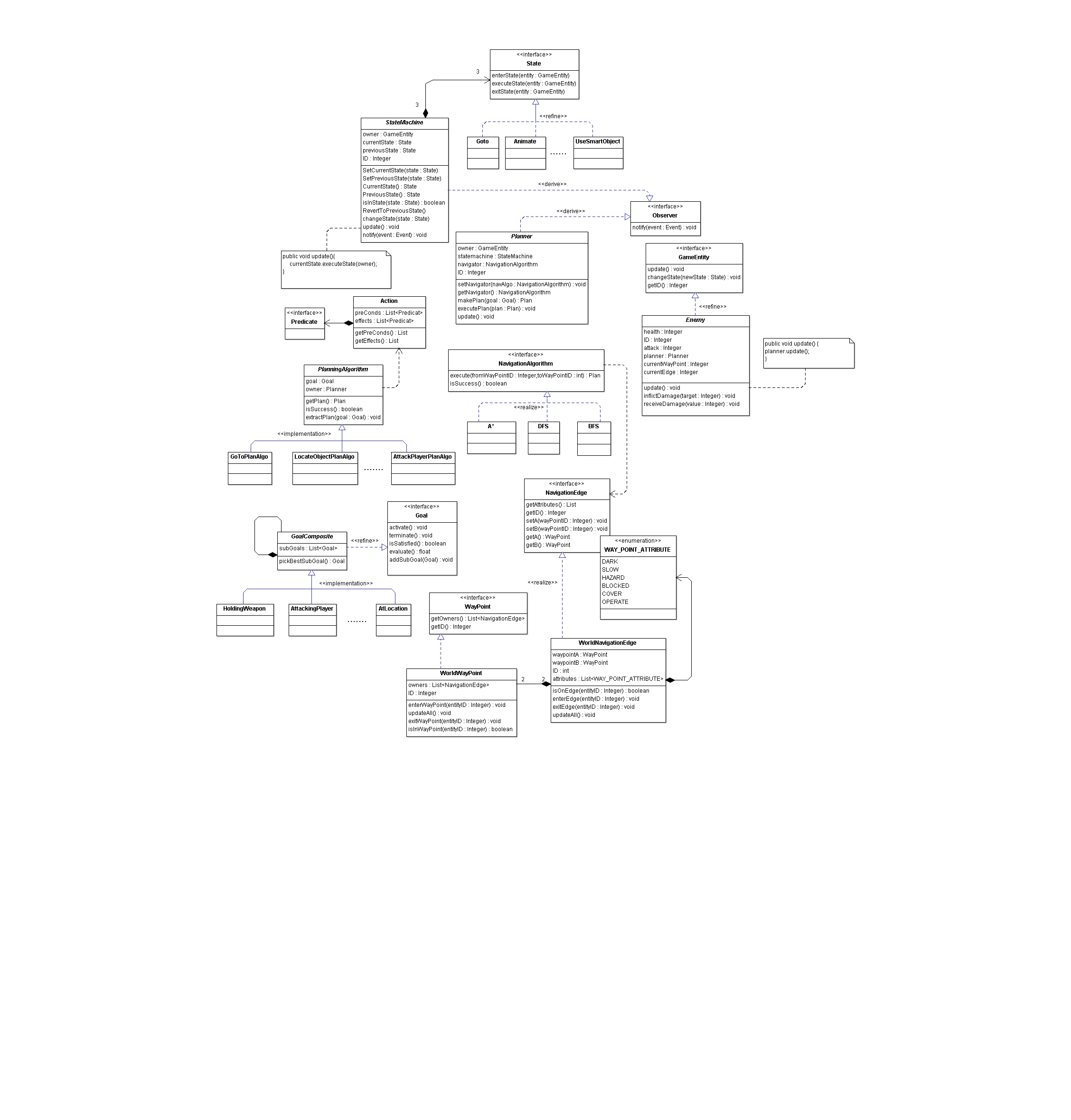
בדומה לסכמה א' KB מבצע את תפקידו באופן דומה.

הוספנו לסוכן את האפשרות לתכן את פעולותיו בהתאם למטרות הנתונות לו ולהגדיר לעצמו מטרות חדשות ותת מטרות באופן דינאמי.  
בהינתן מטרה:

1. המתכנן יפעיל את אלגוריתם התכנון ויקבל סדרה של תת מטרות שמרכיבות שהשגתן תאפשר השגת המטרה הראשית.
2. עבור כל תת מטרה יבצע המתכנן את 1 עד שיתקבלו פעולות אטומיות שניתן להעביר אל מכונת המצבים ומנגנון הניווט.
3. מכונת המצבים ומנגנון הניווט יפעלו בהתאם לפעולה האטומית ויחזירו הצלחה או כישלון בהתאם.

נמשיך להשתמש במכונת מצבים בשביל להפעיל "פעולות אטומיות", כלומר, פעולות שחייבות להתבצע באופן מידי. לדוגמה, במידה והסוכן היה בדרכו לנקודה כלשהי בעולם המשחק ונורה במפתיע על ידי רובה הצלפים של השחקן, הסוכן חייב להפעיל את אנימציית המוות שלו.

**תרשים UML כללי של מבנה הסוכן מסכמה ב':**



**הסבר:**

נציין את השינויים העיקריים:

1. הסוכן כבר אינו מכיל מופע יחיד של מכונת המצבים שלו, מופע של האובייקט **Planner** משמש לתיאום ושליטה על פעולת מכונת המצבים ומנגנון הניווט. ה**Planner** אחראי להפעיל אלגוריתמים של תכנון המתאימים לסוג המטרה שקיבל הסוכן. אלגוריתם תכנון מחזיר תכנית פעולה לשם השגת מטרה נתונה.  
   האלגוריתם מותאם באופן ספציפי למטרה נתונה ויחזיר פתרון מתאים עבורה. הסיבה שבגינה לא נשתמש באלגוריתם תכנון כללי היא מטעמי יעילות זמן ריצה והיכולת לבצע אופטימציות מקומיות.

מכונת המצבים מכילה רק 3 מצבים אפשריים:

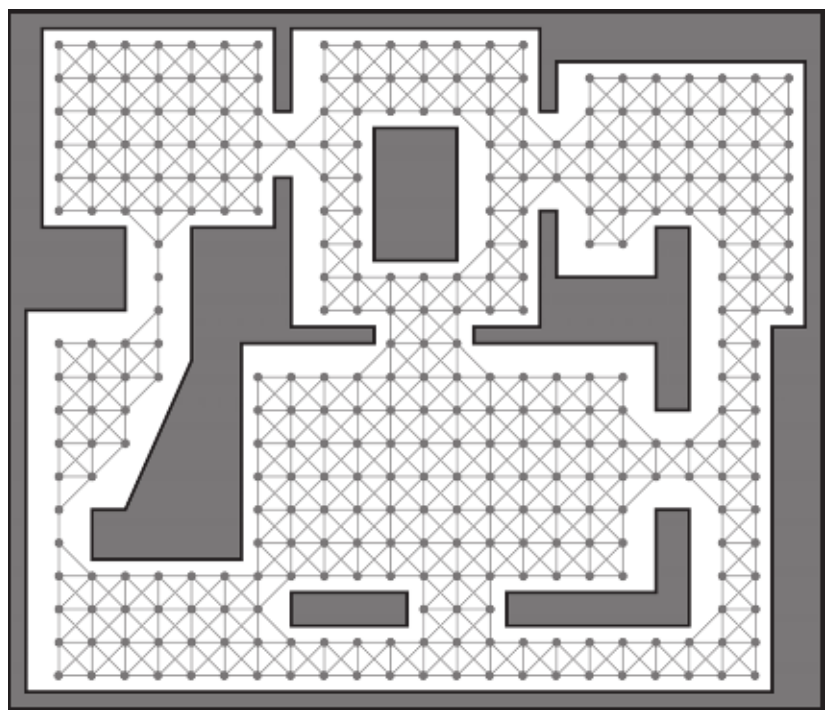
* 1. מצב **GoTo** – הסוכן נע ל-wp נתון.
  2. **Animate** – הסוכן מבצע אנימציה מסוימת (לכרוע ברך, לירות, לעמוד וכו...)
  3. **UseSmartObject** – זאת למעשה ספציפיקציה של מצב ה-Animate שמתאפיינת בכך שבהינתן אובייקט בעולם מסוג SmartObject הסוכן יפעיל את האנימציה שתסופק לו מתוך האובייקט.  
     נשים לב לכך שלמעשה, מכונת המצבים שלנו מורכבת מ-2 מצבים בלבד!

מצב זה הגיוני כי מבחינת הסוכן, כל פעולה שהוא מבצע הרי מורכבת מהפעלה של אנימציה או תזוזה למקום מסויים בעולם במשחק, לדוגמה: הפעולה RunForCover מורכבת למעשה מתזוזה של הסוכן למיקום שמוגדר כ-COVER והפעלת אנימציה DUCK.

1. כל מטרה אפשרית בעולם המשחק, מכילה אימפלמנטציה מותאמת של אלגוריתם הערכה (**Evaluator**)  
   שנעדר מהתרשים מטעמי נוחות הצגה. אותו אלגוריתם אחראי לחשב את ערך התועלת הנובעת מהשגת מטרה מסוימת. מטרה מורכבת מאוסף של אובייקטים מסוג פרדיקט. הגדרנו מטרה כמבנה מורכב (**Composite**), לכל מטרה ייתכנו אוסף של תת-מטרות הדרושות לצורך השגתה. מטרה מורכבת היא אוסף של מטרות, הסוכן יבחר במטרה בעלת התועלת הגבוהה ביותר עבורו. הוא משיג זאת בעזרת הפעלת המתודה **evaluate** על המטרה המורכבת. זאת בתורה, ממיינת את רשימת המטרות שהיא מכילה ובוחרת להפעיל את המטרה בעלת התועלת הגבוהה ביותר.
2. ה-Planner מכיל מחסנית של מטרות שממוינת לפי התועלת של כל מטרה. סדר המופעים של המטרות במחסנית מכתיב את סדר התוכניות שה-Planner מייצר ומבצע בהתאם. אלגוריתם לבניית תכנית עבור מטרה, עושה שימוש באובייקטים מסוג Action בכדי להגדיר את הפלט. בכל הפעלה של מתודת **update** ה-**Planner** בודק האם הושגה המטרה שבראש המחסנית:
   1. אם המטרה אינה בעלת השגה, אזי המתכנן מנסה לתכנן תכנית חדשה עבורה. אם הוא נכשל בכך המטרה מוסרת מהמחסנית ונבחרת המטרה הבאה אחריה. אם הוא מצליח, אז התוכנית המתקבלת עוברת לביצוע.
   2. במידה ומתקבלת תחושה (אירוע/**Event**) מהעולם שדורשת טיפול מידי, המתכנן מיצר מטרה מתאימה ודוחף אותה לראש המחסנית. באופן זה, כשתופעל המתודה **update** הסוכן יכול לחדול לביצוע התוכנית הנוכחית ולאחר שטיפל בתוכנית שמתקבלת מהמטרה בראש המחסנית, לחזור ולתכנן בשנית פתרון למטרה שהופסקה. בכך סיפקנו מענה לבעיה 1 של סכמה א'.

|  |  |  |
| --- | --- | --- |
| הסוכן מבצע תכנית למטרה Patrol | השחקן יורה על הסוכן, כתוצאה מכך נוצרת מטרה חדשה: ReturnFire שנדחפת לראש המחסנית  **PUSH** | הסוכן סיים להחזיר אש. כתוצאה מכך המטרה ReturnFire נשלפת (POP) מראש המחסנית, והסוכן חוזר לתכנן ולבצע את Patrol  **POP** |
| Patrol | ReturnFire  Patrol | ReturnFire  Patrol |

1. הניווט של הסוכן הפך דינאמי יותר. הסוכן יכול כעת לבחור את אלגוריתם החיפוש המועדף אליו בכדי למצוא מסלולים בגרף הניווט של עולם המשחק. נשים לב שבכדי לפתור את בעיה 4 של סכמה א', ה-wp אינם מכילים יותר attribute. תכונה זאת עברה לאובייקט **WorldEdge** שמאפשר לסוכן לבצע בזמן ריצה (**update**) בדיקה על התכונות של הקשת עליה הוא מטייל ולפעול בהתאם.
2. פתרנו את בעיה 2 של סכמה א' בכך שנספק לסוכן תמיד מטרת בסיס שתהיה ראשונה להיכנס למחסנית המטרות ושלא ניתן להשלימה כל עוד השחקן לא נפסל, כגון, LookForPlayer, שבמידה ונמצא השחקן תתחלף במטרות כגון, AttackPlayer או RunForCover. באופן זה, הסוכן תמיד יוכל לתכנן את מהלכיו הבאים ולעולם לא ימצא במצב שהוא נטול מטרה.
3. בכדי למזער למינימום האפשרי נקודות מתות בעולם המשחק נשתמש בגרף ניווט צפוף ככל הניתן:



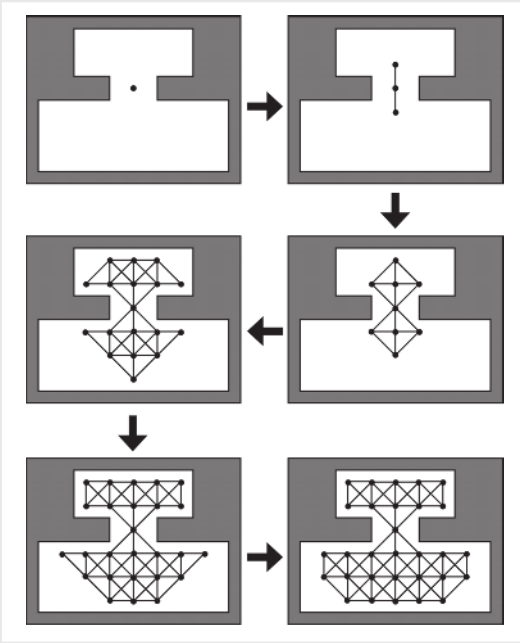
ניתן להשיג מילוי טוב של עולם המשחק, בעזרת אלגוריתם FloodFill, שמקורו מתחום הגרפיקה הממוחשבת, נתאר בקצרה את אופן פעולתו:

אלגוריתם FloodFill:

**קלט:** גרף שמתאים למודל התלת ממדי של עולם המשחק (צמתים הם vertices, קשתות הן edges), צומת התחלה.  
**פלט:** גרף ניווט לסוכן.

* אתחל מחסנית ריקה.
* אתחל רשימה ריקה של צמתים שנחקרו.
* אתחל גרף פלט ריק.
  1. הכנס את צומת ההתחלה למחסנית.
  2. כל עד המחסנית **לא** ריקה:
     1. שלוף את הצומת שבראש המחסנית.
     2. הכנס את הצומת לרשימת הצמתים שנחקרו.
     3. עבור כל צומת שיש לה קשת משותפת עם הצומת הנוכחית ושהיא מסומנת כ"קרקע":  
        אם הצומת לא ברשימת הצמתים שנחקרו – הכנס אותה לראש מחסנית והכנס אתה ואת  
        הקשת שמחברת בינה לבין הצומת הנוכחית לגרף הפלט.
     4. חזור ל-2.
  3. החזר את גרף הפלט.

**להלן המחשה של פעולת האלגוריתם:**



1. נשים לב שאובייקט מסוג פעולה מכיל את המשתנים **PreConds** **וeffects** שהם רשימה של פרדיקטים שמהווים תנאים מקדימים ותוצאות של ביצוע הפעולה בהתאמה. למעשה, פעולה תואמת מבחינה רעיונית לפעולה המנוסחת בשפת **PDDL** ולכן, ניתן להיעזר **באלגוריתמים לתכנון** כגון **חיפוש קדימה** בכדי לייצר תכנית פעולה עבור הסוכן.

# סיכום

בעבודתי נסקרו הגישות השונות להגדרת בינה מלאכותית. צוינו והוסברו דרך פעולתם של אלגוריתמים שמהווים אבני יסוד למרבית המימושים הקיימים של סוכנים בינתיים במשחקי מחשב. דרך הפיתוח וההצגה של ארכיטקטורה שמוצגת תחילה בסכמה א', והשיפורים וההרחבות שבוצעו בסכמה ב' הכרנו לעומק את התרומה העמוקה של האלגוריתמים שהוצגו לפיתוח סוכן בינתי במשחק FPS. בתכנון סכמה ב', נשארנו נאמנים לקווים המנחים שבבסיס הגישה הרציונלית כגון, התמודדות עם חוסר הוודאות של עולם המשחק באמצעות שימוש בתכנון ובתועלת בכדי לדמות אינטואיציה אנושית.

**התוצאה שהתקבלה הינה עיצוב משופר לסוכן בינתי, הניתן בקלות לתחזוקה והרחבה.**

בעתיד סוכנים שפועלים בסביבה הטכנולוגית שאופפת אותנו יהפכו לחלק מחיינו:

* ינהלו עבורנו יומני פגישות.
* ינקו את ביתנו.
* ינטרו את מצב בריאותינו ויתריעו על כל חשד לתסמין חולי.
* יבצעו הערכה פסיכולוגית של מצבנו הנפשי ועוד...

עם התפתחות הטכנולוגיה והחומרה הייעודית שתיטיב עם ביצועי האלגוריתמים שאת חלקם הכרנו בעבודתי יהפוך תהליך הלמידה הממוחשבת של הסוכנים למהיר ויעיל יותר. מהרגע שבו יעשה תהליך הלמידה של הסוכנים למהיר ויעיל כשל בני האדם, ניתן יהיה רק לדמיין ולחלום אילו אופקים חדשים יפתחו בפני האנושות, שלא תהיה עוד מוגבלת ללמוד ולהסיק לבדה על מסתורי היקום.

אני כולי תקווה שבזכות עבודתי זאת, הצלחתי להצית בלב הקורא ייצר סקרנות וחקרנות לתחום המרתק של הבינה המלאכותית.

# ביבליוגרפיה

## ספרים

* BEHAVIORAL MATHEMATICS FOR GAME AI – Dave Mark,   
  ISBN-10: 1584506849, 2009
* AI Game Engine Programming 2 nd edition- Brian Schwab,   
  ISBN-10: 1584505729, 2009
* AI Techniques For Game Programming – Mat Buckland (Classic),  
  ISBN-10: 193184108, 2002
* Artificial Intelligence for Games - Ian Millington,   
  ISBN-10: 0123747317, 2009
* Artificial Intelligence A Modern Approach 3rd Edition – Stuart Russell,  
  ISBN-10: 0132071487, 2010
* Programming Game AI by Example - Mat Buckland,  
  ISBN-10: 1556220782, 2005
* Automated Planning: Theory & Practice - Malik Ghallab,
* ISBN-10: 1558608567, 2004

## מאמרים

* Representing First-Order Causal Theories by Logic Programs - Paolo Ferraris (Google, Inc.), 2011
* Three States and a Plan: The A.I. of F.E.A.R. 3 Game Developers Conference 2006  
  By Jeff Orkin Monolith Productions / M.I.T. Media Lab, Cognitive Machines Group <http://www.jorkin.com>
* Killzone’s AI: dynamic procedural combat tactics, an article by: Remco Straatman (Guerrilla Games), William van der Sterren (CGF-AI) and Arjen Beij (Guerrilla Games), 2005.

1. פרויקט "כחול עמוק" של חברת IBM או בשמו המקורי Deep Blue נחשב לאחד ההישגים הבולטים בתחום עד היום, מטרת הפרויקט הייתה לייצר מחשב המסוגל לשחק שחמט ולנצח את אלוף העולם דאז בשנת 1997 גרי קספרוב. המחשב נצח את קספרוב לאחר סדרה של ששה משחקים. [↑](#footnote-ref-1)
2. פסאודו-קוד (מאנגלית Pseudo-Code ): הוא תיאור מצומצם ולא רשמי לאלגוריתם של תוכנת מחשב. פסאודו-קוד משתמש במונחים הלקוחים משפות תכנות, אך מיועד לקריאה של בני אדם ולא לקריאה על ידי מחשב. פסאודו-קוד מהווה שפה משותפת לכל המתכנתים, ובעזרתו מתכנתים בשפות תכנות שונות יכולים להבין זה את זה. כיוון שהוא מיועד לבני אדם, נהוג לזנוח סימני תחביר ופרטים נוספים, כמו הצהרת משתנים, שלא חיוניים להבנת הקוד ואף מפריעים לקריאות שלו. [↑](#footnote-ref-2)
3. אלן מת'יסון טיורינג( באנגלית: Alan Mathison Turing;‏ 23 ביוני 1912 – 7 ביוני 1954) היה מתמטיקאי בריטי, ממניחי היסודות למדעי המחשב. הגיע להישגים יוצאי דופן בצד התאורטי ובצד המעשי של מדעי המחשב. [↑](#footnote-ref-3)
4. האחים אורוויל (19 באוגוסט 1871 - 30 בינואר 1948) ווילבור (16 באפריל 1867 - 30 במאי 1912) רייט (Wright) היו הראשונים שביצעו טיסה בכלי טיס כבד ומאויש. [↑](#footnote-ref-4)
5. דימות תהודה מגנטית(באנגלית Magnetic Resonance Imaging בראשי תיבות MRI )הוא סוג של סריקה לא-פולשנית המשמשת להמחשת איברים פנימיים בגוף ללא שימוש בקרני רנטגן, למטרות אבחון רפואי, חקר המוח, פסיכיאטריה, מחקר ביולוגי ועוד. [↑](#footnote-ref-5)
6. אלן ניואל ( מרס 1927 - 19 יולי 1992 ) היה חוקר של תחום מדעי המחשב ופסיכולוגיה קוגניטיבית בתאגידRAND ובבית הספר של אוניברסיטת Carnegie Mellon למדעי מחשב, בית הספרTepper למנהל עסקים, והמחלקה לפסיכולוגיה. הוא תרם לשתיים מתוכנות הAI המוקדמות: מכנות הלוגיקה התאורטית (1956) ופותר הבעיות הכללית בשנת (1957) עם הרברט סיימון. הוענק לו פרס טיורינג, יחד עם הרברט סיימון ב-1975 על התרומה הבסיסית שלהם לתחומי הבינה המלאכותית והפסיכולוגיה של ההכרה אנושית. [↑](#footnote-ref-6)
7. אריסטו היה פילוסוף יווני, מבכירי הפילוסופים של העת העתיקה, ומאבות הפילוסופיה המערבית. מייסד האסכולה האריסטוטלית בפילוסופיה, מהאסכולות המשפיעות ביותר בפילוסופיה עד היום. [↑](#footnote-ref-7)
8. היוריסטיקה (Heuristic) היא כלל חשיבה פשוט, מעין כלל אצבע המבוסס על הגיון פשוט או אינטואיציה, המציע דרך קלה ומהירה לקבלת החלטות, ללא התעמקות  
    ובמחיר דיוק נמוך. [↑](#footnote-ref-8)
9. אלגוריתם דייקסטרה, פותר את בעיית מציאת המסלול הקצר ביותר מנקודה [בגרף](http://he.wikipedia.org/wiki/%D7%92%D7%A8%D7%A3_(%D7%AA%D7%95%D7%A8%D7%AA_%D7%94%D7%92%D7%A8%D7%A4%D7%99%D7%9D)) ליעד. מכיוון שניתן למצוא באמצעות [אלגוריתם](http://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D) זה, בזמן זהה, את המסלולים המהירים לכל הנקודות בגרף, בעיה זאת נקראת לעתים מציאת המסלולים הקצרים מנקודה. האלגוריתם עובד על גרף נתון, מכוון או לא מכוון, בעל משקולות אי-שליליות על הקשתות. המשקולות בגרף מסמלות מרחק. משמעותו של המסלול הקצר ביותר בין שתי נקודות היא המסלול בעל סכום המשקולות הנמוך ביותר בין שתי הנקודות. [↑](#footnote-ref-9)
10. טבלת גיבוב (Hash table), היא מבנה נתונים שמקשר מפתחות עם ערכים. הפעולה העיקרית שבה הוא תומך ביעילות היא אחזור מידע מתוך מבנה הנתונים: בהינתן מפתח נתון (למשל שם של אדם), מצא את הערך המתאים (למשל מספר הטלפון של אותו אדם). המבנה עובד על ידי הפיכת המפתח באמצעות פונקציית גיבוב למספר סידורי המייצג למעשה את האינדקס של המערך. ניתן לומר, אם כן, כי טבלת גיבוב היא מעין מערך שאליו ניתן לגשת באמצעות מפתחות, ולא אינדקסים. [↑](#footnote-ref-10)
11. מקור השם מרחק מנהטן (הידועה גם בשם "גאומטריית מנהטן") בתנועתו של נהג הנוסע בעיר הבנויה כולה מגושי בניינים מלבניים (כמנהטן) שכל כבישיה מאונכים ומקבילים אלה לאלה. מאחר שלא יוכל לעבור דרך הבניינים, יאלץ הנהג לנסוע תמיד מצפון לדרום או ממזרח למערב.  
    לכן, אורך המסלול אותו יעבור נהג כזה יהיה שווה בדיוק לסכום המרחקים אותם עבר בנסיעה מערבה ואלה אותם עבר בתנועתו צפונה.  
    האנלוגיה למנהטן מטעה במקצת, משום שבמנהטן הנהג יכול להגיע רק לנקודות מסוימות, אלה הנמצאות על הכביש. בגאומטריית מנהטן ניתן להגיע לכל נקודה במישור, וההגבלה היחידה היא שהתנועה היא במקביל לאחד משני הצירים.

    |  |
    | --- |
    | http://upload.wikimedia.org/wikipedia/commons/thumb/d/de/Manhattan_distance_bgiu.png/300px-Manhattan_distance_bgiu.png |
    | **מרחק מנהטן** |

    [↑](#footnote-ref-11)
12. בהנדסת תוכנה, תבנית עיצוב (באנגלית: Design pattern) היא פתרון כללי לבעיה שכיחה בעיצוב תוכנה. תבנית עיצוב אינה עיצוב סופי שניתן להעבירו הישר לקוד, אלא תיאור או תבנית לדרך לפתרון בעיה, שעשויה להיות שימושית במצבים רבים. תבניות עיצוב מונחות עצמים מציגות לרוב יחסים וקשרי גומלין בין מחלקות או אובייקטים, בלי לפרט את המחלקות או האובייקטים הסופיים המעורבים ביישום. [↑](#footnote-ref-12)