**עבודת גמר**

לקבלת תואר

טכנאי תוכנה

הנושא: סימולטור נהיגה של מכונית

המגיש: יותם לויט

מנחים: מיכאל צ'רנובילסקי ואלון (להשלים)

מאי 2020 תש"ף

A close up of a sign

Description automatically generated

|  |
| --- |
| תוכן עניינים |
| 1. מבוא |
| * 1. מטרה   2. תיאור המערכת   3. שפת התכנות ופירוט סביבת העבודה והכלים |
| 1. מפרטי תוכנה |
| * 1. תיאור כללי   2. ניסוח וניתוח הבעיה האלגוריתמית   3. פיתוח הפתרון ויישומו   4. תיאור אלגוריתם   5. מבנה נתונים |
| 1. חלוקה למודלים |
| * 1. חלוקה למודלים   2. רשימת הנתונים |
| 1. מדריך למשתמש |
|  |
| 1. ביבליוגרפיה |
|  |
| 1. נספח – קוד מרכזי של הפרויקט |

**1. הקדמה**

1.1 מטרה

*מטרתו של המסמך המוצג להלן היא להציג את הפרויקט בצורה מפושטת על כל חלקיו. המסמך מיועד לכל המעוניין להבין בפשטות את מהותו של הפרויקט, ככל שניתן, ובלי להיכנס לקוד.*

* 1. רקע טיאורטי:
     1. המשחק והאלגוריתם
* AlphaCar – AI *שמטרתו ללמוד לנהוג במכונית על ידי* ML
* *כיום יש חברות העובדות על תוכנה למכונית אוטונומית. חברות אלה בונות מודל אמיתי של*

*מכונית כדי לבדוק את הפיתוח, ועלותו הכללית של המודל היא גדולה. הסימולטור נועד*

*למזער את עלות הפיתוח של מודל המכונית האוטומטית בכך שהוא נותן לחברות סביבה*

*ווירטואלית לבדיקת בינת המכונית.*

*הסימולטור הוא תוכנה המריצה קוד של מכונית אוטונומית בסביבה ממוחשבת, בין אם*

*מדובר בלמידה בין אם מדובר באלגוריתם טכני. התוכנה מאפשרת לחברות לאתר טעויות*

*בקוד או ללמוד במקרה של למידה של המערכת לנהיגה בשלב ראשיתי, קודם לשלב המודלים*

*האמיתיים שאם יתנגשו ויהרסו יאבד הרבה כסף. הודות לסימולטור תוכלנה חברות להריץ*

*את האלגוריתם בסימולטור, לחסוך זמן - א ם מדובר בלמידה, וכסף - אם מדובר בטעות*

*באלגוריתם.*

*בפרויקט זה קוד המכונית יהיה "אלגוריתם מסובך" והמכונית תהיה בעלת חיישנים למדידת*

*מרחק מעצמים.*

* + 1. מהלך הלמידה :

*הלמידה תיעשה בשיטת ה-algorithm Genetic ותיצור מספר רב של מכוניות. לכל אחת יש*

*את המערכת הנוירונים שלה עם ערכים שיוצרים "DNA "ייחודי לכל מכונית. האלגוריתם*

*יבחר את המכוניות בעלות הביצוע הטוב ביותר, ישלב את ה"DNA "שלהם לתוך "DNA "*

*אחד, וייצור מספר רב של "מוטציות" חדשות מתוך השילוב.*

* + 1. הפעלת אלגוריתם המכונית:

*הפעלת האלגוריתם תבוצע במסלולים מתוכננים, אשר אינם מוכרים לאלגוריתם.*

*במהלך הלמידה המכונית תלמד להגיב לאופציות השונות הקיימות בנהיגה.*

* + 1. הגדרת האלגוריתם :

*במסך ההפעלה יוכל המשתמש לתת קובץ המופעל עם "API "של המשחק. המשחק יקבל*

*את הקובץ ובעזרת הAPI יוכל להריצו.*

* 1. מטרות הפרויקט:

*1 .מימוש רשת הנוירונים.*

*2 .מימוש אלגוריתם שנותן ציון לכל "DNA "של מכונית.*

*3 .בניית מערכת שבוחרת את המכוניות המצטיינות, משלבת את ה"DNA "שלהם לתוך*

*"DNA "אחד ויוצרת מספר רב של המוטציות שלו.*

*4 .יצירת ממשק משתמש , אתו יוכל המשתמש לתקשר עם התוכנה.*

*5 .יצירת סביבה תלת-ממדית, שבה האלגוריתם יורץ עם מסלולים ללמידת המכוניות.*

* 1. דרישות מערכת:

*1 .מימוש רשת נוירונים*

*1.1 רשת בעלת 3-4 שכבות*

*2 .מימוש אלגוריתם שנותן ציון לכל "DNA "של מכונית*

*1.2 .הציון יקבע לפי אחוזי התנגשות, שרידות ה"DNA "ומרחק נסיעה*

*3 .יצירת ממשק משתמש , אתו יוכל המשתמש לתקשר עם התוכנה.*

*1.3 .ממשק אינטואיטיבי ונעים לשימוש.*

*4 .יצירת סביבה תלת-ממדית, שבה האלגוריתם יורץ עם מסלולים ללמידת המכוניות.*

*1.4 .מסלולים מתוחכמים ללמידה יעילה יותר*

*2.4 .נדרש כי המסלול לא יהיה סלעי*

* 1. משאבים:
     1. עמדת פיתוח:
* **מערכת הפעלה Windows**
* **סביבת עבודה Visual Studio וכמו כן שימוש בעורך Unity** 
  + 1. עמדת משתמש:
* **מערכת הפעלה Windows**
* **עוד**
  1. קישור למסמכים קודמים

מצורף מסמך אפיון

* 1. הגדרות

GUI - Graphical user interface ממשק משתמש

ML – Machine Learning למידה מלכותית.

AI – Artificial Intelligence *אינטליגנציה מלכותית.*

1. ארכיטקטורת המערכת

A screenshot of a social media post

Description automatically generatedחלק זה כולל את תיאור מבנה המערכת ופירוט המודולים השונים בה

2.1 מבט על הארכיטקטורה

ישנם N(לא ידוע עדיין מספר גמור) קבצי-קוד המרכיבים ומפעילים את הפרויקט

* שני קבצים אחראים על ממשק המשתמש: מסך פתיחה, מסך סימולציה.

אלה שמות הקבצים:

* + open\_window\_AlphaCar
  + Simulator
* קובץ קבצים אחראים על הלוגיקה בין הסימולטור והמסכים לבין המסכים עצמם:
  + ArcadeCar.cs
* 5 קבצים ו -2 ספריות אחראים על הגיאומטריה האנליטית, התנגשות בין אובייקטים בסביבה ועל צורה של האובייקטים בפרויקט:

**הספריות:**

* **NeuralNetworkLibary**
* **GraphLibary**

**כל קבצי מנוע המשחק כגון:**

* ArcadeCar.cs
* LaserSensor.cs
* CarCollision.cs
* MovingBlock.cs
* SuspensionTest.cs
* שלושה קבצים ושתי ספריות האחראים על מנן התזוזה של אובייקטים זזים כמו Car:

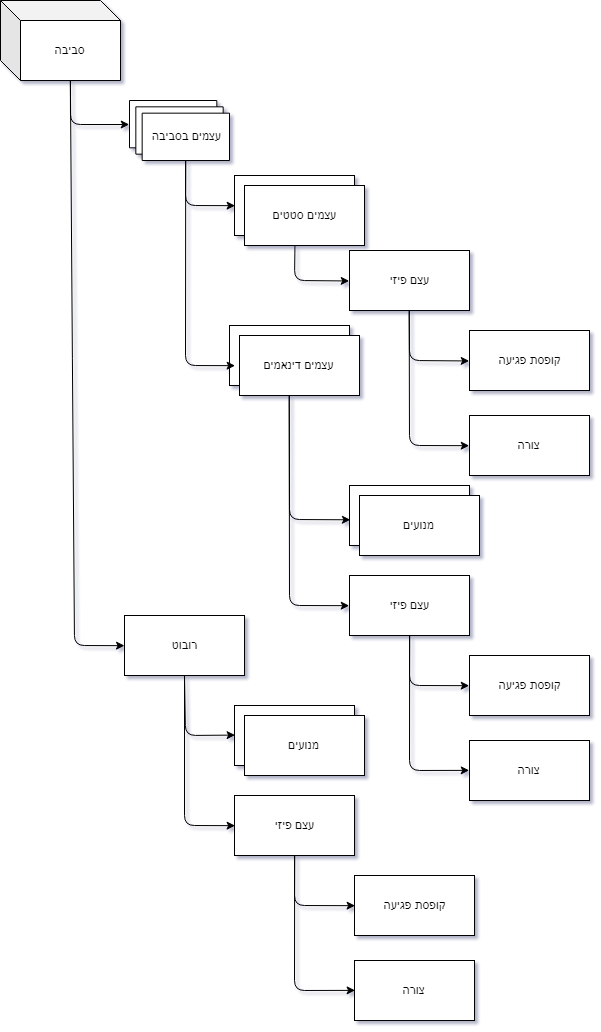
**שלושת הקבצים:**

* + MovingBlock.cs
  + ArcadeCar.cs
  + CarCollision
  + SuspensionTest

**שתי הספריות:**

* **NeuralNetworkLibary**
* **GraphLibary**

2.2 פירוט רכיבי המערכת:

****תיאור מבנה המנוע הלוגי:

מחלקות המנוע:

|  |
| --- |
| ArcadeCar - מחלקה יורשת מMonoBehaviour של המנוע |
| תכונות:   * **Wheel\_Left\_Index – המיקום של הגלגל הקדמי השמאלי** * **Wheel\_Right\_Index - המיקום של הגלגל הקדמי השמאלי** * **Nn – רשת הנוירונים** * **Ls1-ls5 – חיישנים 1-5 מסוג Laser Sensor** * **(Class)WheelData – מחלקה שמכילה את כל המידע על הגלגל** * **(Class)Axle – מחלקה שמכילה מידע על ציר** * **centerOfMass – ווקטור שמייצג מרכז של מסה** * **accelerationCurve\ accelerationCurveReverse- גרף שמייצג את התקדמות האצה** * **steerAngleLimit – גבול זווית פנייה** * **downForceCurve\ downForce – כוח שמושך למטה** * **axles – מערך שמכיל את הצירים** * **isBrake\ isHandBrake\ isAcceleration\ isReverseAcceleration – משתנים כללים לפעולת** **המכונית** |
| פעולות:   * **Reset – מחזיר את המכונית למצב ההתחלתי שלה עם DNA חדש** * **AddEdgesForNN4D – מוסיף Edges לגרף של הNN** * **Start – נקראת פעם אחת בתחילת ריצת הקוד – יוצר את הNN, מרכז המסה ואת הגוף הפח** * **GetHandBrakeK – מחזיר את הכוח של הבלם ים** * **GetSteeringHandBrakeK – מחזיר את הכוח עצירה בפנייה** * **GetAccelerationForceMagnitude – מחזיר כוח האצה** * **GetSpeed – מחזיר את המהירות** * **GetSteerAngleLimitInDeg – מקבלת את המהירות במטר לשנייה ומחזיקה את גבול הפנייה במעלות** * **UpdateInput – עושה את קליטת הצעד הבא** * **FixedUpdate – נקראת כל סיבוב של הסימולטור – עושה את כל הקריאות של הפעולות שאיתן מפעילים את המכונית ואת כל הפיזיקה** * **AddForceAtPosition – מקבלת ווקטור כוח וווקטור מיקום ומסיפה את הכוח באותו מיקום** |

|  |
| --- |
| **LaserSensor –** מחלקה יורשת מMonoBehaviour של המנוע |
| תכונות:   * **Range – המרחק המקסימלי של החיישן** * **Lr – דרך להראות את הלייזר עם LineRenderer של Unity** * **Distance – המרחק כרגע שהחיישן מחזיר** |
| * **Start – נקראת פעם אחת בתחילת ריצת הקוד – יוצרת את ה LineRenderer** * **FixedUpdate – נקראת כל סיבוב של הסימולטור – עושה את כל חישובי המרחק ואת שינויי הפיזיקה והלייזר. כמו כן מעדכן את המרחק שהחיישן מחזיר (Distance)** |

|  |
| --- |
| **CarCollision -** מחלקה יורשת מMonoBehaviour של המנוע |
| תכונות:   * **Movement – אובייקט מסוג ArcadeCar שמכיל את המכונית שתתנגש במחסומים כשתטעה** |
| פעולות:   * OnCollisionEnter – פעולה שקוראת רק כאשר ישנו שילוב במשוואות הגופים במרחב של המכונית ומחסום – הפעולה בודקת אם מדובר במחסור ואיזה מחסור |

|  |
| --- |
| **MovingBlock -** מחלקה יורשת מMonoBehaviour של המנוע |
| תכונות:   * **Amplitude – משתנה שמכיל את השינוי בכיוונים במרחק** * **Speed – מהירות השינוי Amplitude** * **initialTime – זמן התחלתי** * **originalPos – המירום של העצם שזז** * **time -** |
| פעולות:   * **Start – נקראת פעם אחת בתחילת ריצת הקוד – נותנת ערכים לתכונות** * **Update – קוראת כל סיבוב של סימולטור – משנה את הערכים של המיקום של** **dynamic** |

|  |
| --- |
| **SuspensionTest –** מחלקה יורשת מMonoBehaviour של המנוע |
| תכונות:   * **Amplitude – משתנה שמכיל את השינוי בתנודות המטלים** * **originalPos – ווקטור המיקום של הגוף עם המטלים** * **dynamic – עצם של הגוף שאליו מחובר המטלים** * **cameraObject – המצלמה שעוקבת אחרי dynamic** * **speed – מיהירות השינוי של** |
| פעולות:   * **Start – נקראת פעם אחת בתחילת ריצת הקוד – נותנת ערכים לתכונות** * **Update – קוראת כל סיבוב של סימולטור – משנה את הערכים של המיקום של** **dynamic** |

**מתוך הספריות:**

**ספריית GraphLibary:**

|  |
| --- |
| **Parallelogram – יורש מ-Object** |
| * **self.relative\_angle – זווית בין הצלעות** * **self.main\_line – צלע מנחה מרכזית בצורה כיוונה ככיוון התנועה כמו כן נקודת ההתחלה של הקו היא נקודת ההתחלה של הצורה** * **self.relative\_line –צלע מנחה קו בזווית self.relative\_angle ביחס לצלע המרכזית** * **self.main\_line\_2 – קו שמקביל ל self.main\_line** * **self.relative\_line\_2 – קו שמקביל ל self.relative\_line** |
| * **get\_lines – מחזיר את הקווים שמרכיבים את הצורה** * **change\_relative\_angle – מקבל זווית שתהווה כערך חדש לself.relative\_angle** * **move\_by\_units – מזיז את הצורה לפי יחידות** * **get\_middle\_point – מחזיר את נקודת המרכז הגיאומטרי של הצורה** * **change\_pos – מקבל נקודה שאליה נקודת ההתחלה של הצורה תזוז** * **change\_rotation – מקבל זווית ווקטור נקודתי שמייצג את מרכז הכבידה ומסתובב סביבו את המעלות** * **change\_size – מקבל שתי אורכים ומציב אותם בקווים המנחים של הצורה** * **get\_points – מחזיר את הקדקודים של הצורה כאובייקט נקודה ברשימה** * **get\_equation – מחזיר את המשוואות של הקווים שמרכיבים את הצורה** * **get\_point\_value – מחזיר את הערכים של הנקודות בתוך רשימה של tuples** * **draw – מצייר את הצורה** * **get\_edge\_points – מחזיר את שתי הנקודות הכי קיצוניות כtuple של עצם נקודה** * **convert\_shape\_to\_txt – ממיר את הצורה למחרוזת בפורמט xml** * **is\_Colliding – מקבל צורה ובודק אם ישנה התנגשות** * **parallelogram\_from\_file - ממיר את מחרוזת בפורמט xml לצורה** |

|  |
| --- |
| **Sensor – מחלקה** |
| **Self.line\_of\_sight – ישר המראה את קוו הראייה של החיישן ואת כיוון הראייה** |
| **Get\_dis\_for\_sensor – מחזיר את המרחק בין החיישן לעצם שנכנס שקוו הראייה שלו** |

דיאגרמה לתיאור מפת הGUI:

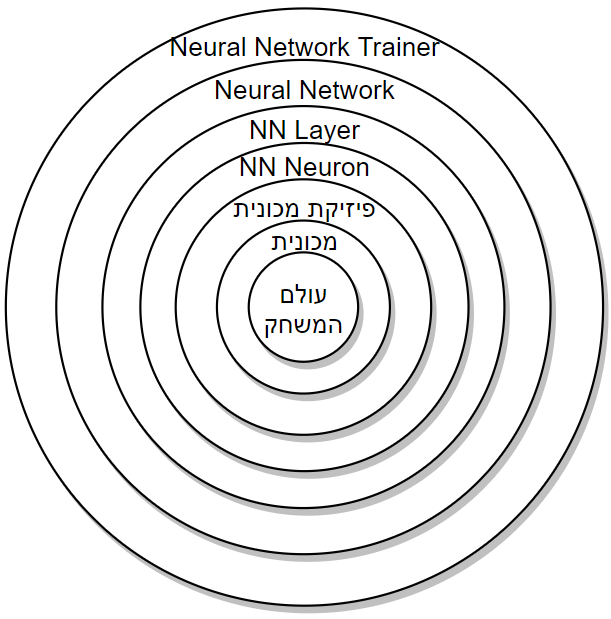
A screenshot of a cell phone

Description automatically generated

2.3 דיון בנושא העיצוב הנבחר

* שפת תכנות: בחרתי ב-C# לאורך כל הקוד שלי משום ששפה זו מחולקת בשיטת OOP כמו כן למנוע המשחק Unity יש תמיכה בשפה. דברים זה מאפשר גמישות בשיטות הפעולה ומתן אפשרויות רבות להרחבת הפרויקט.
* חלוקה למחלקות בשיטת OOP: שיטה זו יוצרת מדרגיות ושינויים בה קלים יותר. בנוסף OOP מספקת פשטות וארגון במהלך התכנות.

מעגלי פיתוח:



1. **חישובים מתמטיים ואלגוריתמיקה**
2. **האלגוריתם:**
   1. מהלך הלמידה :

*הלמידה תיעשה בשיטת ה-algorithm Genetic ותיצור מספר רב של מכוניות. לכל אחת יש*

*את המערכת הנוירונים שלה עם ערכים שיוצרים "DNA "ייחודי לכל מכונית. האלגוריתם*

*יבחר את המכוניות בעלות הביצוע הטוב ביותר, ישלב את ה"DNA "שלהם לתוך "DNA "*

*אחד, וייצור מספר רב של "מוטציות" חדשות מתוך השילוב.*

1. **הסימולטור**:

הסימולטור:

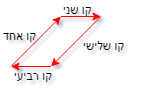
מבט על:

A close up of a map

Description automatically generated

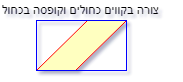
הסימולטור הוא המרכז את הGUI ומנוע הסימולציה. הפונקציה מריצה את המנוע, בודקת התנגשויות ומשנה את הGUI כך שיהיה תצורה גרפית בזמן ההרצה

הרכב של צורה:

כל עצם מורכב מקווים המתוארים בפונקציה מתמטית וכמו כן כווקטור המגביל את גודל הקו כל קו מתחיל בנקודת הסיום של הקו לפניו חוץ מהקו הראשון שהוא מתקבל ביצירת הצורה. כך נוצר פוליגון שמהווה גוף לעצם. דוגמה לפוליגון שיכול להיות:

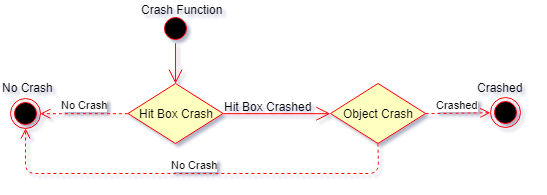
קופסת פגיעה – Hit Box:

כל עצם בסביבה יש מלבן או "קופסה" סביבו המאונך לצירי x z ו-y של המסך. לקופסה קוראים "Hit Box ".

קצוות הקופסה נמצאים בערכים הכי קיצוניים של העצם שעליו הם נמצאים. מטרת הקופסה היא להקל על החישובים של התנגשות בכך שהיא מורכבת רק מארבעה ישרים וכך החישוב קצר יותר כמו כן חישובים של פונקציות שמאונכות לצירים יותר פשוטות .דוגמה לקופסה:

בדיקת התנגשות בין עצמים:

כאשר הסימולטור בודק אם יש התנגשות בין שתי עצמים הוא קודם בודק אם קורת התנגשות בין Hit Box יש עצם אחד עם Hit Box של עצם אחר. המחשה וויזואלית של הבדיקה:



איך בודקים אם שתי צורות מתנגשות:

על צורה של עצם יש קווים ממנה היא מורכבת וכך גם עצם Hit Box מורכב מצוהר שבנויה מישרים. כל ישר מתואר בעזרת פונקציה מתמטית על המישור ,z x, y ווקטור, כדי לבדוק התנגשות צריך

* לבדוק אם לקוו מצורה של עצם אחד יש נקודה משותפת עם קוו מצורה של עצם אחר. כלומר צריך להשוות בין משוואות הישרים
* אם יוצא נקודה משותפת משמע הצורות מתנגשות. לכן:
  + בשלב ראשון בודקים את ה-Hit Box משום שמשוואות הישרים שלו מקבילים לצירים
  + אם יש התנגשות בודקים את הצורה עצמה שיכולה להיות מורכבת ממספר רב של ישרים בזוויות שונות.

1. **ממשק משתמש (לא סופי)**

"ממשק משתמש" של הפרויקט מורכב מאלה המסכים:

1. A picture containing web

   Description automatically generatedמסך פתיחה – החלון הראשון העולה בפתיחת התוכנה. החלון מעוצב מלוגו הפרויקט ומשתי אפשרויות בחירה: האחד, להמשיך להפעלת התוכנית; השני, ליציאה מהתוכנה.
2. A picture containing outdoor, road, mountain, sitting

   Description automatically generatedמסך הסימולציה – חלון הסימולציה מעוצב משורת תפריט: כפתור הפעלה, כפתור עצירה וכפתור יציאה. כפתור הפעלה מפעיל את הסימולציה; כפתור עצירה עוצר את הסימולציה; כפתור יציאה מוציא ממסך הסימולציה למסך פתיחה. נוסף על מסך הסימולציה עומדים לרשות המשתמש אלה השניים: מחוון שליטה על מהירות ההפעלה ותווית להצגת מצבם של הNN והפלט ממנה.