­­­Racing AI

פרויקט למידת מכונה

[GitHub](https://github.com/mixelburg/RacingAI)**קישור לקוד:**

Python 3.8.5 **שפת פיתוח:**

PyCharm**סביבת** **פיתוח:**

**ספריות:**

**ספריות עיקריים:**

pygame==1.9.6

neat-python==0.92

colorama==0.4.4

matplotlib==3.3.2

**ספריות עזר:**

certifi==2020.6.20   
cycler==0.10.0   
kiwisolver==1.2.0   
mypy==0.790   
mypy-extensions==0.4.3   
numpy==1.19.2   
Pillow==8.0.0   
pyparsing==2.4.7   
python-dateutil==2.8.1   
six==1.15.0   
typed-ast==1.4.1   
typing-extensions==3.7.4.3

[neat](https://en.wikipedia.org/wiki/Neuroevolution_of_augmenting_topologies) הפרויקט הוא משחק עם מכוניות לומדות לנהוג. הן עושות זאת בעזרת אלגוריתם

**קבצים בפרויקט:**

קובץ קוד ראשי : [**main.py**](https://github.com/mixelburg/RacingAI/blob/master/main.py)

: קובץ עם הגדרות בסיסיות[**main-config.json**](https://github.com/mixelburg/RacingAI/blob/master/main-config.json)

neat : קובץ הגדרות [**config-feedforward.tx**t](https://github.com/mixelburg/RacingAI/blob/master/config-feedforward.txt)

: קובץ תוצאות סימולציות[**results.txt**](https://github.com/mixelburg/RacingAI/blob/master/results.txt)

: תוכנית להפעלת מספר סימולציות[**run.py**](https://github.com/mixelburg/RacingAI/blob/master/run.py)

: תוכנית להצגת תוצאות סימולציות[**display\_resuts.py**](https://github.com/mixelburg/RacingAI/blob/master/display_resuts.py)

המטרה העיקרית של הפרויקט היא להראות כיצד מכונית יכולה ללמוד לנהוג

באמצעות אנלוגי בסיסי של מכ"ם

מדריך למשתמש

**(**[**config-feedforward.tx**t](https://github.com/mixelburg/RacingAI/blob/master/config-feedforward.txt)**,** [**main-config.json**](https://github.com/mixelburg/RacingAI/blob/master/main-config.json)) configהגדר את כל ההגדרות בקבצים 1)

2) ישנן שתי אפשרויות:

1) להריץ סימולציה אחת:

main.pyפשוט הפעל את הקובץ

2) להריץ מספר סימולציות:

כדי להריץ כמה multiprocessing הוא משתמש בrun.pyהפעל את הקובץ

סימולציות באותו זמן.

display\_results.py3) אפשר גם להציג תוצאות בעזרת

DOCכל ההסברים על עבודת של הפונקציות, משתנים... יש ב



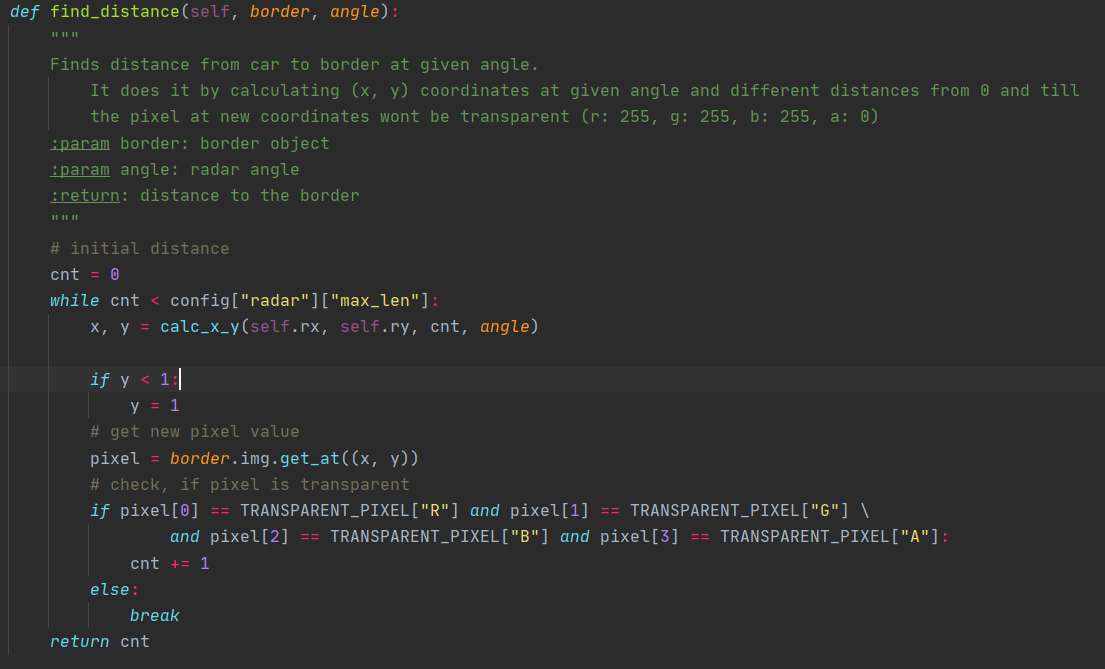
אלגוריתמים מרכזיים:

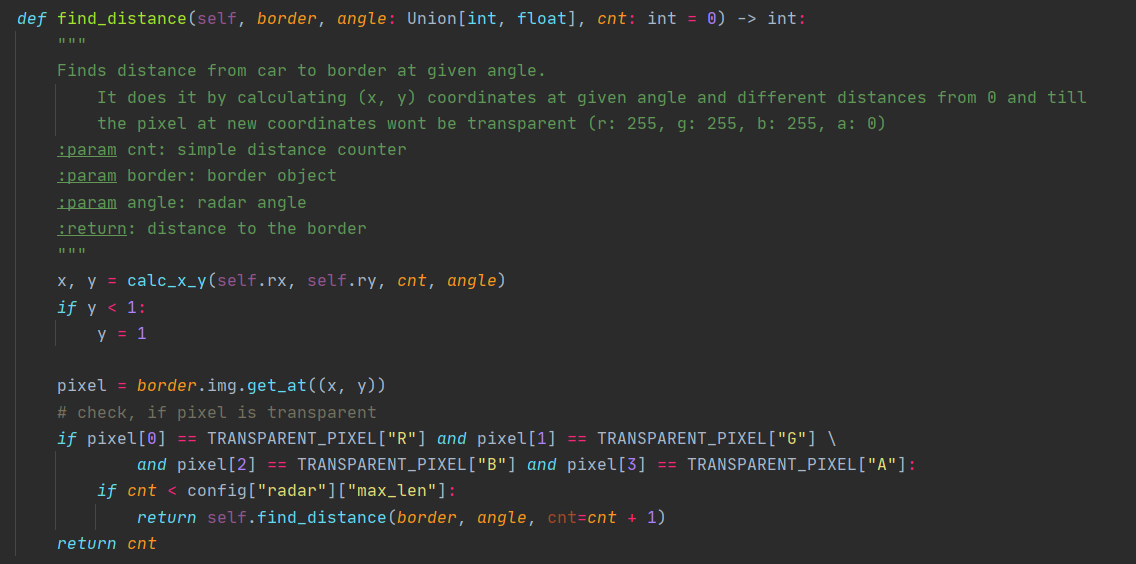
מציאת את המרחק לקיר:

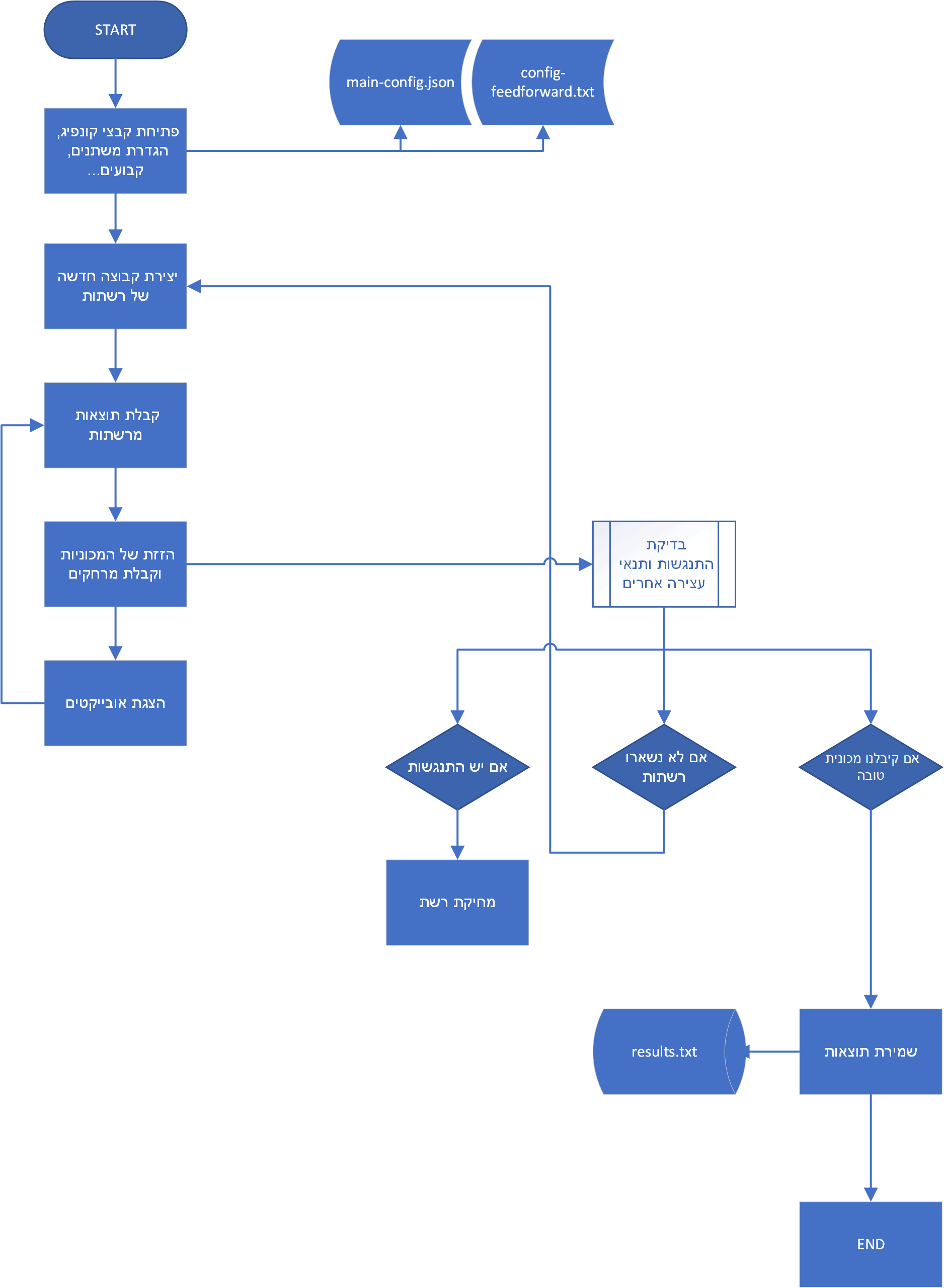
. (255, 255, 255, 0)תמונת הדרך נפרדת מתמונת הרקע. בגלל זה במקום פיקסלים של הדרך יש פיקסלים

.Nאז בתאמצאות לולאה (או רקורסיה) אפשר פשות לבדות את כל הפיקסלים במרחקים מ1 עד

(255, 255, 255, 0)עד שהפיקסל במרחק N לא היהי







:NEATאלגוריתם

הם אקראייםbias, weightאלגוריתם זה יוצר קבוצה של מספר רשתות בה הערכים

הטובות ביותר biasלאחר סימולציה, הוא בוחרת את הרשתות אם ה

ויוצר מהם קבוצה חדשה עם ערכים קצת שונים.

רפלרציה

לבסיף אני יכול לומר שאני מרוצה מהפרויקט שלי. עבדתי על זה הרבה זמן וקיבלתי תוצאה מדהימה. אני לא חושב שצריך לשנות משהו בפרויקט. התמודדתי עם די הרבה קשיים. המון זמן שיחקתי עם הנתונים שאני מהביר לרשתות. לקח לי זמן לכתוב פונקציה מציאת מרחק אבל חוץ מזה הפרויקט היה די קל.

Display.py

בסוף העבודה, עשיתי מחקר קצר. בדקתי, איך מספר סימולציות שצריך לעשות כדי לקבל מכונית "טובה" תלוי בגודל הקבוצה של מכוניות. כדי לעשות זאת, כתבתי תוכנית run.py היא מריצה כמה סימולציות עם גודל קבוצה שונה והרצתי 50 סימולציות לכל גודל קבוצה מ150 ועד 10 (150, 140, 130...). לקח לי זמן, אבל בסוף גיליתי שאין כל כך הבדל בין קבוצות אם 150 ו-50 מכוניות.

תוכן עניינים

1)  [מידע בסיסי](#MainData)

2) [מדריך למשתמש](#Instruction)

2.1) [אלגוריתמים מרכזיים](#MainAlgorithm)

3) [רפלקציה](#Reflection)