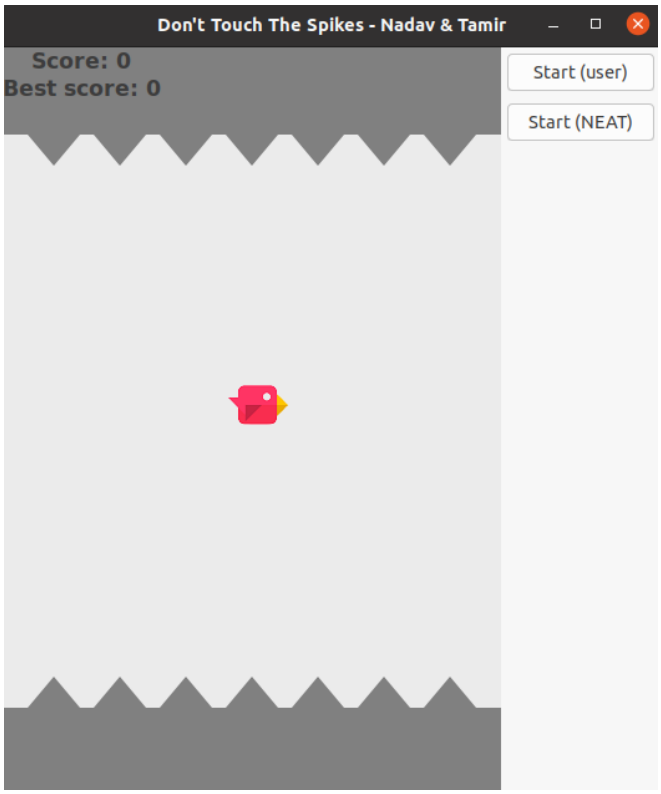
Neuro-Evolutional "Don't Touch the Spikes" with Erlang

Creators: Tamir Cohen and Nadav Hadad.

Course: Functional Programming in Concurrent and Distributed Systems,
Ben Gurion University.

Professor: Yehuda Ben-Shimol.

Teaching Assistant: David Leon.



התכנון שכתבנו בהתחלה

המאסטר:
מחלק עבודה בין 4 המחשבים. למשל אם בכל איטרציה של אוכלוסיה יש 1000 ציפורים, אז השרת שולח לכל אחד מהם 250 ציפורים (ויש 1000 רשתות נוירונים שונות סך הכל, כלומר משקולות שונות). כל מחשב נותן לציפור "לרוץ" ומגדיר לה פונקציית רווח (fitness function) שתהיה שווה לכמות הפריימים שהציפור בחיים.
המחשבים אומרים לשרת מי הם 10% הציפורים הכי טובות שלהם והמאסטר משלב את המידע ומוליד את הדור הבא, שהוא השארה של הציפורים הכי טובות מהדור הקודם וגם מוטציות שלהן (ע"י רינדום קל במשקולות הרשת שלהם).
וכך ממשיכים עוד איטרציות.

- נשלח הודעה im_alive כל שניה ובמקרה והיא לא הגיעה נכריז על המחשב כנעדר ונחלק את העבודה בין שאר המחשבים שעדיין בחיים.

מכונת מצבים של כל ציפור (לכל פרוסס):

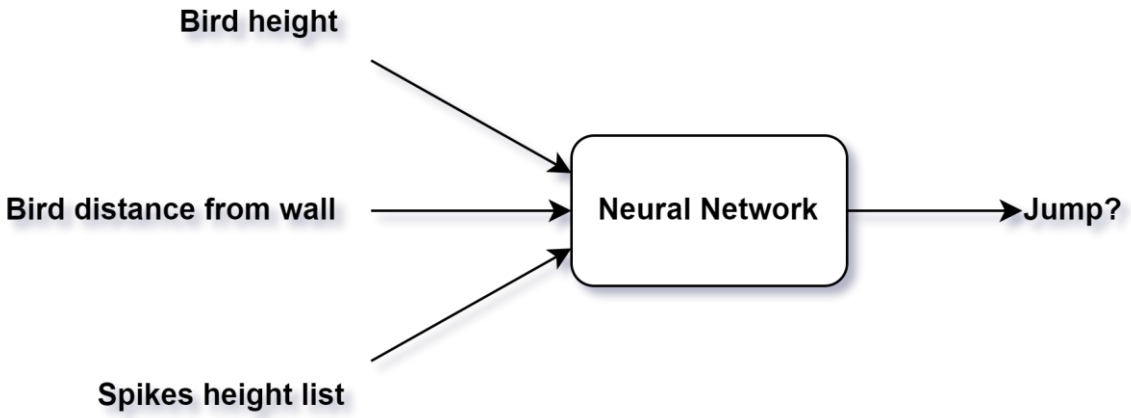
1. המתנה ללחיצה על הכפתור של המשחק או לקבלת רשת נוירונים לתפעול של הציפור. (idle)
2. התחלת ריצה של המשחק (בעזרת הרשת שקיבל) עד פסילה (נגיעה בקוץ). (simulation)

קלט לרשת נוירונים:

1. גובה הציפור.
2. מרחק הציפור מהקיר.
3. רשימת הקוצים שבקיר ממול.

הפלט מהרשת הוא האם לבצע קפיצה.

כמו באיור:

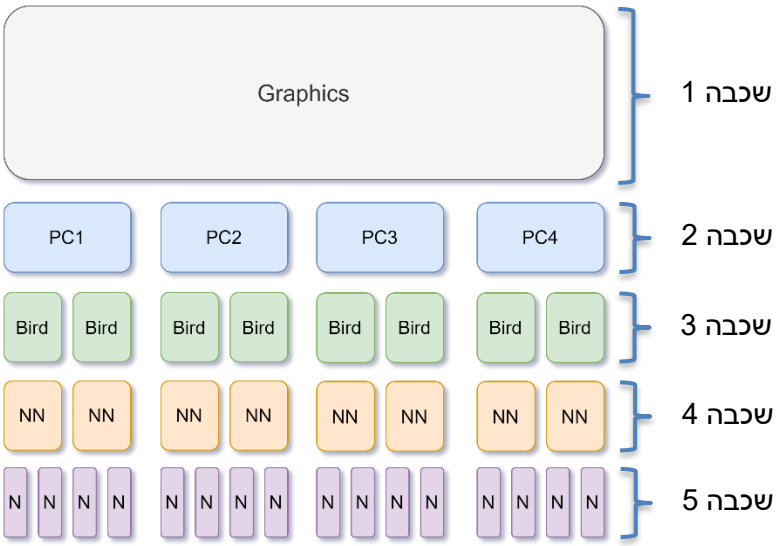


שלבי עבודה בסיסיים:

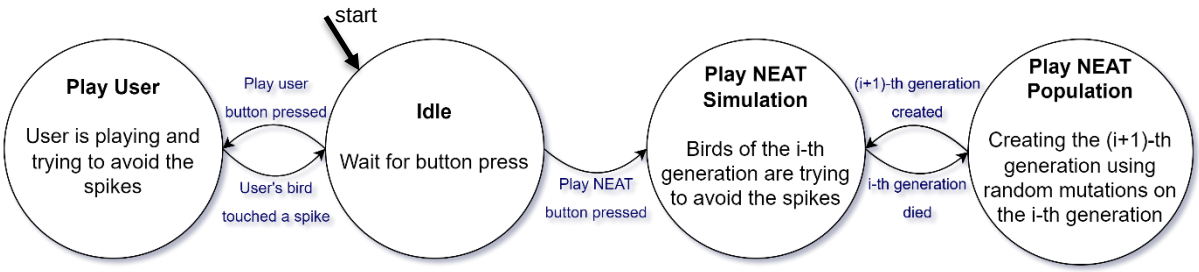
- תחילה נכין את המשחק שיעבוד אינטרקטיבית למשתמש.
- לאחר מכן נשנה אותו כך שאת הבחירה האם לקפוץ תבצע רשת הנוירונים ואז נוסיף המון ציפורים.
- לאחר מכן נפצל את העבודה בין טרמינלים שונים ולבסוף בין מחשבים שונים תוך תמיכה בנפילות.

הביצוע בפועל

הפרויקט שלנו מורכב **מ-5 שכבות**:



1. השכבה הראשונה היא **Graphics**. שכבה זו תנהל את הגרפיקה שהמשתמש מתנהל איתה (GUI). קיים רק מופע אחד של המודול הזה, שירוץ על המחשב הראשי, ויהיה אחראי על יצירה ואחזקה של 4 תהליכי PC. הוא ממומש לפי behaviour של **WxWidget** ופועל לפי ה-FSM הבא:



במכונת המצבים הזו קיימים 4 מצבים:

1.1. **Idle** – המתנה ללחיצה על אחד משני הכפתורים: 

זהו בנוסף המצב ההתחלתי של המערכת. בעת לחיצה על "Start (User)", המערכת תיכנס למצב "Play User". בעת לחיצה על "Start (NEAT)", המערכת תיכנס למצב "play NEAT Simulation".

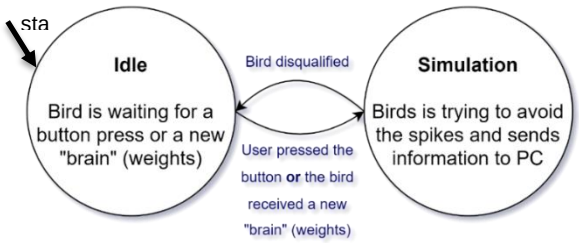
1.2. **Play User** – במצב זה המשתמש יכול בעצמו לשחק במשחק עם ציפור אחת על המסך. המשתמש יכול לגרום לציפור לקפוץ על ידי לחיצה על הכפתור "Jump". כאשר יש פסילה (נגיעה באחד מהקוצים), המערכת חוזרת למצב ההתחלתי "idle".

1.3. **Play NEAT Simulation** – בכניסה הראשונה למצב זה המערכת מיד מתחילה להריץ את דור הציפורים הראשון בו כל המשקולות שברשתות של הציפורים הם רנדומליות לחלוטין. כל הציפורים מחולקות בצורה שווה בין כל 4 ה-PC השונים. כאשר כולן נפסלות, המערכת עוברת למצב "Play NEAT Population", כך שבחזרה מהמצב הזה המערכת יכולה להתחיל את ההרצה עבור דור הציפורים הבא.

1.4. **Play NEAT Population** – במצב זה מחלקים בצורה שווה בין המחשבים את 20% הציפורים הכי טובות מהדור האחרון, כלומר 5% לכל מחשב. כל מחשב ישמור לדור הבא את הציפורים שקיבל ובנוסף ייצור 4 מוטציות לכל אחת מהן. כלומר, כל מחשב קיבל 5% מסך כל הציפורים במערכת ויצר 20% חדשות. סך הכל, כל מחשב יחזיק 25% מסך כל הציפורים של הדור הבא.

כאשר כל המחשבים סיימו את יצירת המוטציות של הדור הבא, הם מודיעים זאת לגרפיקה והמערכת עוברת למצב "Play NEAT Simulation".

2. השכבה השנייה היא **PC**. כל מופע מהשכבה הזו מקבל הוראות מהגרפיקה שמעליו על מנת ליצור ולתחזק כמות גדולה של ציפורים. הוראה לדוגמא יכולה להיות simulate_frame שקוראת כל 80ms, ומשמעותה היא שכל הציפורים שלחו את מיקום ה-Y שלהן וניתן לעבור לבצע את הפריים הבא. מודול זה ממומש לפי behaviour של **gen_server**.
3. השכבה השלישית היא **ציפור**. מודול זה ממומש לפי behaviour של **gen_statem** ופועל לפי ה-FSM הבא:

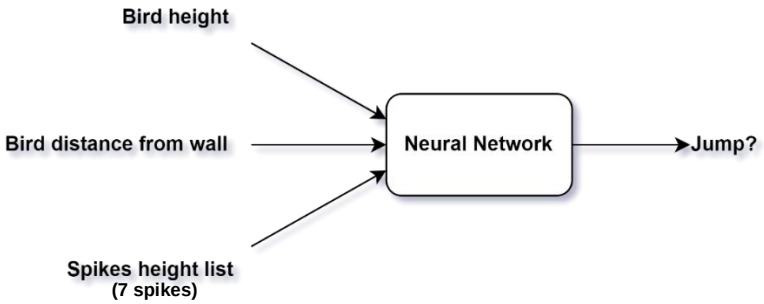


במכונת המצבים הזו קיימים 2 מצבים שמשותפים גם לציפור של המשתמש וגם לציפור של הרשת נוירונים:

3.1. **Idle** – המתנה ללחיצה על אחד משני הכפתורים Start (user) Start (NEAT) או לקבלת מוח חדש מהמחשב לאחר סיום של הדור הנוכחי.

3.2. **Simulation** – הציפור קיבלה הוראה לעבור למצב זה ולהתחיל סימולציה. בכל הודעת simulate_frame היא תתקדם פריים בודד ותשלח את המיקום שלה או הודעת פסילה אל ה-PC שלה. במידה וזו ציפור של NEAT היא גם תדבר עם הרשת נוירונים.

4. השכבה הרביעית היא **רשת הנוירונים** של הציפור. היא מעין "שרת" שיש לכל ציפור וניתן אף לתאר אותה כ"מוח" של הציפור. בכל פריים חמישי (מדלגים על ארבעה) הציפור שואלת את הרשת האם לקפוץ, וכתלות בקלטים הנוכחיים הרשת תחזיר תשובה אם יש צורך לבצע קפיצה. הרשת מתואר באיור:



גודל השכבות של הרשת שבחרנו לבנות היא [9, 6, 6, 6, 1], כאשר ה-9 הראשון הוא גודל ה-input layer, וה-1 שבסוף הוא ה-output layer. כל רשת נוירונים כזאת יוצרת ומתחזקת נוירונים (מהשכבה הבאה).

<u>**הערה:**</u> לאחר ההצגה במעבדה ביצענו שינויים במבנה הרשת ובפונקציית האקטיבציה והשתפרו ביצועי המערכת:

5. השכבה החמישית היא **נוירון** בודד. מודול זה ממומש עם לולאה אינסופית שבה הנוירון מקבל הודעות מהנוירונים שבשכבה שלפניו, מבצע חישובים ולבסוף שולח את התשובה לנוירונים הבאים בשכבה שאחריו.

נפילת מחשב

כפי שתיכננו בהתחלה, כל 0.5 שניות הגרפיקה שולחת הודעת "{are_you_alive}" אל 4 המחשבים, והם צריכים להגיב תוך 0.5 שניות בחזרה "{im_alive, PC_Name}". המחשב מסמן אצלו מי שלח לו את ההודעה הזו, וכל 0.5 שניות רשימת המחשבים שבחיים מתעדכנת להיות רשימת המחשבים ששלחו תשובה. בכל דור שעובר, הגרפיקה מחלקת את העבודה החדשה בין המחשבים שבחיים ולכן התוכנית לא תיתקע. כל התהליך מתרחש ברקע ואינו דורש מצב במכונת המצבים של הגרפיקה, ועל כן מאיץ את ביצועי המערכת.

סטטיסטיקות

אספנו כמה סוגי סטטיסטיקות:

- שמנו לב כי מספר ההודעות הנשלחות בכל פריים הן (פריים קורה כל 80ms):

$$4 + 400 + 400 + 400 = 1{,}204$$

Simulate frame (graphics to PC)

Simulate frame (PC to birds)

Bird sends location or disqualified (bird to PC)

PC sends location or disqualified (PC to graphics)

פעם ב-5 פריימים, כל הציפורים החיות שולחות בקשה לרשת נוירונים לבצע חישוב האם לקפוץ או לא. עבור ציפור בודדת, זוהי כמות ההודעות הנוספות שנשלחות:

$$1 + 9 + \underbrace{9 \cdot 6 + 6 \cdot 6 + 6 \cdot 6 + 6 \cdot 1}_{} + 1 + 1 = 144$$

Ask NN to decide whether to jump (bird to NN)

Feeding the input layer (NN to neurons)

Neurons messages - Feed Forward (neuron to neuron)

Send to NN the answer (neuron to NN)

NN tells the bird to jump (**only if needed**)

לכן, עבור כל הציפורים ביחד, במקרה שבו כל 400 הציפורים עדיין חיות, כמות ההודעות המקסימלית שתשלח (אם כולן קופצות) היא:

$$144 \cdot 400 = 57{,}600$$

ולכן **כמות ההודעות הממוצעת לפריים אחד** היא (עבור 400 ציפורים שעדיין לא נפסלו):

$$\frac{1{,}204 \cdot 5 + 57{,}600}{5} = \mathbf{12{,}724}$$

בנוסף לכל אלה יש גם את ההודעות של הבדיקה האם PC בחיים, שזה בסך הכל 8 הודעות (4 לPC ו-4 תשובה לגרפיקה) כל חצי שנייה.

הודעות חשובות (מסודרות לפי מבנה המערכת)

מוצא	יעד	מבנה ההודעה	תיאור
גרפיקה	PC	{are_you_alive}	הגרפיקה שולחת כל 0.5 שניות ומצפה לתשובה
PC	גרפיקה	{im_alive, PC_Name}	ה-PC עונה לגרפיקה שהוא בחיים
גרפיקה	PC	{start_bird_FSM}	הוראה למחשב ליצור תהליכים של ציפורים
גרפיקה	PC	{start_simulation}	התחל סימולציה (העברת הציפורים למצב סימולציה)
PC	ציפור	{start_simulation}	התחל סימולציה (העברת הציפור למצב סימולציה)
גרפיקה	PC	{spikes_list, SpikesList}	עדכון רשימת הקוצים של הציפורים
PC	ציפור	{spikes_list, SpikesList}	עדכון רשימת הקוצים של הציפור
גרפיקה	PC	{simulate_frame}	הודעה שנשלחת כל timer event שאומרת שניתן לקדם את הציפורים בפריים בודד
PC	ציפור	{simulate_frame}	הודעה שנשלחת כל timer event שאומרת שניתן לקדם את הציפור בפריים בודד
ציפור	PC	{neat_bird_location, Y}	שליחת גובה הציפור אל המחשב
PC	גרפיקה	{neat_bird_location, Y}	שליחת גובה הציפור אל הגרפיקה
ציפור	PC	{bird_disqualified, BirdPID, FrameCount, WeightsList}	ציפור נפסלה. המחשב מוסיף אותה לרשימה ממוינת לפי כמות הפריימים שהצליחה לשרוד
PC	גרפיקה	{pc_finished_simulation, CandBirds}	כל הציפורים של המחשב נפסלו והמחשב שולח את הציפורים הכי טובות שלו אל הגרפיקה
גרפיקה	PC	{populate_next_gen, BestBrains}	שליחה של המוחות (רשימת משקולות) הכי טובים, כדי שכל PC יבצע מוטציות
PC	ציפור	{replace_genes, NewBrain}	המחשב שולח לציפור מוח חדש והיא שולחת אותו אל הרשת נוירונים שלה
PC	גרפיקה	{pc_finished_population}	המחשב סיים לבצע מוטציות ומוכן להתחיל שוב סימולציה

קישורים

קישור לגיט:

https://github.com/Tamir-Co/NeuroEvolutional-Dont-Touch-The-Spikes-with-Erlang

קישור ליוטיוב:

https://www.youtube.com/watch?v=tJf9ZkzeMAI