Tron

Summer Project, 2011

Programming Club

Science and Technology Council

IIT Kanpur

By:

Anshu Avinash

Pankaj Prateek

Rishabh Nigam

Mentor: Shubham Tulsiani

1.Tron

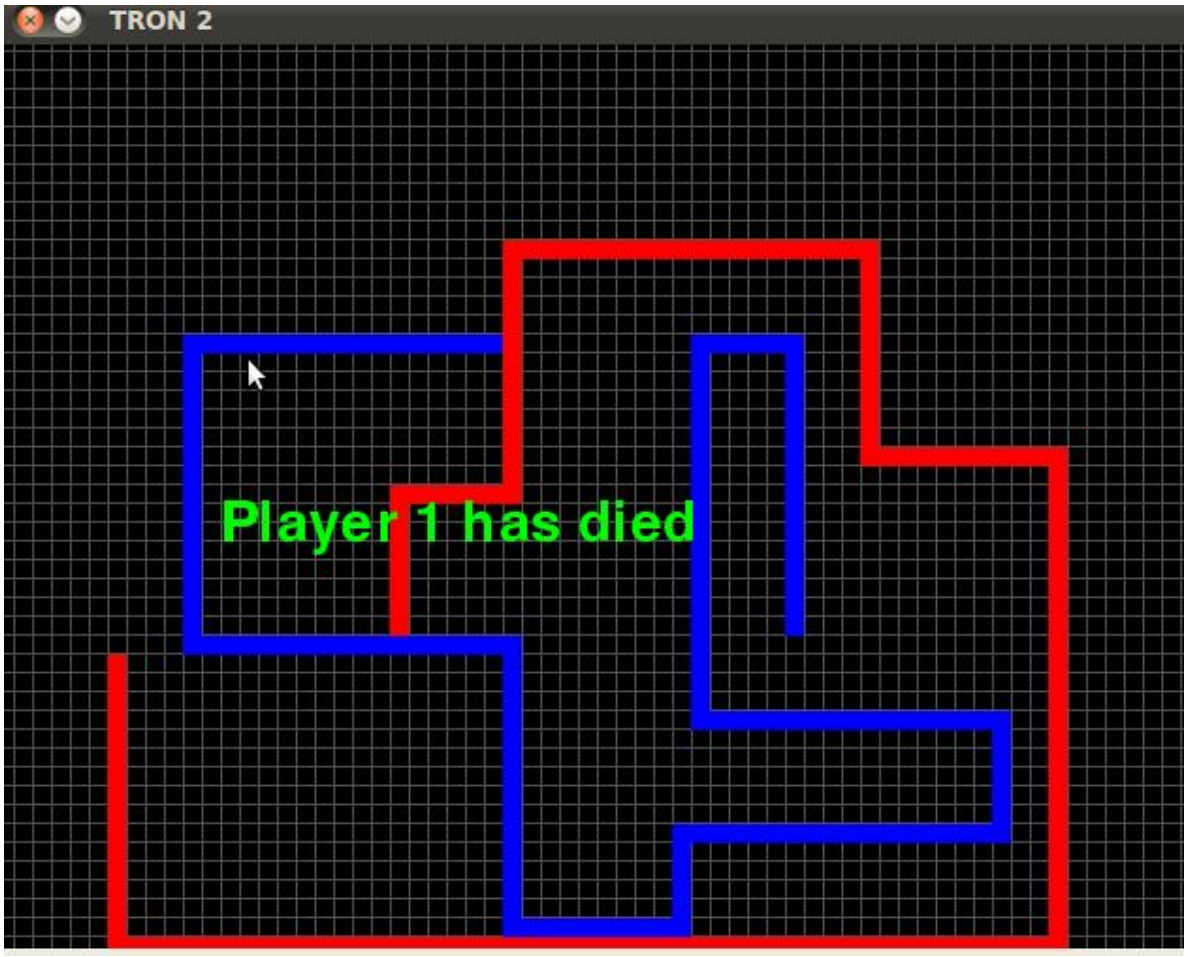
Tron (or light cycles) is a game of pursuit and evasion played on a computer screen. It is based on the Walt Disney Productions motion picture Tron released in 1982. The movie was about a programmer who gets sucked into his computer and its electronic world. It was centered on a game where the players had to cut each other off using motor-bikes that left a line behind them.

1.1 Why Tron

Tron was chosen as a domain to investigate machine learning because it is a simple one yet presents several features that are known difficulties for machine learning: A dynamic, real-time environment where a perfect strategy is not known. Human and machine intelligence can show their particular weaknesses and strengths chasing each other in this competitive arena.

1.2 Rules

Tron is a contest between a human and a computer. Two opponents move at a constant speed. They leave a trail behind, which is an impassable wall. The red trail is controlled by the computer and the blue is controlled by the human using the arrow keys. The first player to crash against a wall loses. Only right angle turns are allowed. The screen is **NOT** wrap-around.



A Tron game as seen on the computer screen: Two players, blue and red, started near the middle of the screen and chased each other. The game finished when the blue player crashed while blue was alive nearby

2.Machine Learning

Machine learning, a branch of artificial intelligence, is a scientific discipline concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. A learner can take advantage of examples (data: In our case, the 'experience' from previous games played) to capture characteristics of interest of their unknown underlying probability distribution. Data can be seen as examples that illustrate relations between observed variables. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new cases.

3. Programming Language : Python

3.1 Why Python

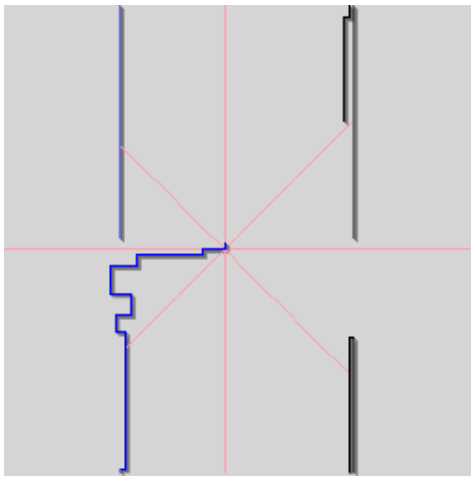
Python is a 'scripting' language, an interpretive language with its own built-in memory management and good facilities for calling and cooperating with other programs. It is flexible, easy and powerful in a way that cannot be matched by other mainstream languages. Also it has in-built support for graphics and GUI development (Pygame and Tkinter), which makes it preferable.

4. Representation

The computer player has been provided 8 radial distance sensors. The sensor value is the distance of the nearest obstacle in the direction of that sensor.

The behavior of the player is controlled by a 'genotype', here, the hypothesis variable, consisting of 32 integer numbers. Each value represents the relative weight that a sensor has in each of the four possible directions that can be chosen.

At every step, the player will go to the direction with a highest score in terms of the current sensor readings and its individual weights.



Sensor model: 8 sensors are the source of information for the player.

5. Machine Learning Implementation

1. Parameter Adjustment : When the player dies

Our first implementation involved learning through practice, giving the computer a hint. When a collision happens, the algorithm raises the weight value of the sensor that pointed in the crash direction by a fixed amount. This creates a successful learning paradigm. Initially the player played blindly, but after a few games, it could play comparatively better against the human or some other AI.

But this implementation did not work in the way we wanted, so we were left with looking for other possibilities and scenarios for parameter adjustment.

2. Parameter Adjustment : Based on when the player traps

When the player traps in a closed loop, there is a decrease in the area available for movement. We tried to use this fact as a method for parameter adjustment by implementing a few functions to detect when the player gets trapped and changing the parameters accordingly.

This method seemed to have a comparatively higher success rate.

6. Implementation of Files to store results of previous games

File handling was implemented to store the results of previous games which was lost initially.

7. Main Classes and Functions

Classes:

snake and snake1: Represent the two players

Functions: __init__ (): initialize the two players

Update(): To update the two players according to the direction they were moving in.

Functions:

checkclose(): To check whether the snake has closed onto itself

hypoadj(): To adjust the values of the weights in different directions

sensorvalue() and **evaluate():** To find out the sensor values and evaluate the best move based on the weighed sum of the sensor values with the weights (hypothesis variable)

readfile() and **writefile():** Used for file handling purposes, instead of initializing the values of the weights at the beginning of each game with random values, the values from previous games are read from the file ‘data.txt’ and the modified values after each game are rewritten into it