Rock, Paper, Scissors Analogy

New Mexico
Super Computing Challenge
Final Report
April 3, 2012

Team 121
11th Grade
School of Dreams Academy, Los Lunas, NM

Team Members:
Bethany Tanner

Sponsoring Teachers:
Creighton Edington

Project Mentors:
Elizabeth Finley

Executive Summary

When you were a kid I'm sure you played the game Rock, Paper, Scissors with your friends. In this simple, fun loving game, Rock beats Scissors, Scissors, beats Paper, and Paper beats Rock.

I created a program on Netlogo that demonstrated this game electronically using "turtle" figures to make viewing these components easier. In this game though, The Rocks are _______, the Scissors and triangles and the Papers are squares. In my program, I succeeded to make the Rock, Paper, Scissors "combat" when they come in contact with each other. Like the game Rock, Paper, Scissors, Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. There is one difference though. There is an rgb (red, green, blue) value of 255. In this 255 rgb value, the "turtles" have Rock, Paper, and Scissors distributed through them. But, if they have a larger percentage of Rock then Paper and Scissors, they will compete as a Rock. If the other component fighting the Rock, is more Paper then Rock or Scissors, then the "turtle" that will win is the Paper. When there is a fight, the winner produces an offspring of a clone of themselves and the loser dies.
When you start this program, you see a "random" selection the program has made to place and make the Rock, Paper, and Scissors on the plane. When you press "run" the components move around and once they combat, they are dying and cloning at different rates. When I created this program, it was strictly for learning purposes since I have never worked with a computer programming system. It was to see if I create a program, to represent a Rock, Paper, Scissors Analogy, if it will work, and if the Rock, Paper, and

Scissors will distribute equally or one will dominate depending on the amount of turtles that are set on the plane.

Introduction

As my project is now the Rock, Paper, Scissors Analogy, I did not start off with this project at the beginning of the Super Computing Competitions. I started off with a project that had to do with cleaning forest fire debris that would take place after another group project was to figure how forest fires happened and what debris would should up afterwards. After working on this project for about three months, the other group as well as my own team mates dropped out of the competition. I was left with no option but to drop out. I did want to keep on with the Super Computing Project though. I found that it was a good learning opportunity and I ended up asking my teacher Mr. Edington to take over the Rock, Paper, Scissors Analogy project. When I took this project over, the program was already written but it was far to complicated to work with and it seemed to not work the way It was intended to work. After going to The Super Computing Mid-Term and getting fantastic advice from the judges, I was able to re think my project and re write my program to make it function correctly. Only having two months of working with this program was definitely a way of cramming big projects in a short amount of time. Learning how to program and figuring whether the Rock, Paper, Scissors program will distribute evenly or take over completely is worth it.

Problem Statement

What I am testing in this program would be whether the Rock, Paper, and Scissors will distribute evenly or if one of the Rocks, Paper, or Scissors will take over the entire population of the “turtles. In finding this I had to run the program over several times and adjust the population to see if there was a difference in whether there are more or less “turtles” on the plane.

Method

The method I used in testing my project was running it over and over to see if there was any significant circumstances in which the Rock, Paper, or Scissors will take over or if there will be an even distribution of the three through out the plane. As I adjusted the amount of “turtles” on the plane from 0 to 2500, I did not see any significant changes except for the fact that there were instances ever about 1000 ticks where there was an extreme fluctuation of where a Rock, Paper, or Scissors take up the majority of the population. After this extreme fluctuation, there was a steady flow back to the even distribution of the Rock, Paper, and Scissors.

Results

After running through my program several times, I found that it did not matter the number of “turtles” in the program, there was a generally even distribution of the Rock, Paper, and Scissors throughout the plane. There were sometimes very large fluctuations to the point where it almost seems as if one of the Rock, Paper, or Scissors may take over the entire plane but it turns out that it depended on the neighboring “turtles”. The dependency of the neighboring turtles did turn out to be because of the way I programmed my system.

Conclusion

I have concluded from this project that anyone can program if you have the motivation and you give enough effort. I also concluded that in this program, there is no such thing as a “random” selection. This random selection based on the “set up” selection of the program showed me that with the rgb value of 255, you can get it as close to random as can be without it ever being random. I have also concluded that if I were to keep going with this project, I would be able to work with the computer well enough to make the program run more randomly then it already does.

Achievements

I have achieved the ability to work with the program Netlogo through this Super Computing Competition. It gave me opportunity to meet and work with people that would teach me more things then I would if I were to work on this alone. I also learned to think more about statistics and be more specific in writing and programming. From taking on this new challenge, a new motivation and purpose was sparked.

Netlogo Program

```
;; off-spring is the same too many times

turtles-own [
  rock paper scissors ;; each turtle will be designated rock paper or
scissors
]
```

```
globals [
  count-rock count-paper count-scissors
]


to setup
  clear-all
  ;hstogram [color] of turtles ;; (not working)
  ask patches [
    set pcolor white  ;; make the background white
  ]
  create-turtles population [
    setup-turtle
  ]
  reset-ticks
end

to setup-turtle
  set rock random-float 1
  set paper random-float 1
  set scissors random-float 1
  let total rock + paper + scissors
  set rock rock / total
  set paper paper / total
  set scissors scissors / total
  update-breed
  setxy random-xcor random-ycor
end

to update-breed
  set color (list (floor (rock * 255.999999)) (floor (paper *
255.999999)) (floor (scissors * 255.999999)))
  let max-value max (list rock paper scissors)
  if max-value = rock [
    set shape "pentagon"
    set count-rock count-rock + 1
  ]
  if max-value = paper [
    set shape "square"
    set count-paper count-paper + 1
  ]
  if max-value = scissors [
    set shape "triangle"
    set count-scissors count-scissors + 1
  ]
end

to decrease-breed
  let max-value max (list rock paper scissors)
  if max-value = rock [
    set shape "pentagon"
    set count-rock count-rock - 1
  ]
  if max-value = paper [
    set shape "square"
    set count-paper count-paper - 1
  ]
  if max-value = scissors [
    set shape "triangle"
    set count-scissors count-scissors - 1
  ]
end

to go
  wiggle-walk
  combat
;  change-shape
;  update-data
  tick
end

to wiggle-walk
  ask turtles [
    left random 30 ;; changes turtles heading to the left by 0 or 1
degree
    right random 30
    forward 1
  ]
end

to combat
  ask turtles [
    let possible-opponents other (turtles in-radius 1)
    if any? possible-opponents [
      let opponent one-of possible-opponents
      let strategy random-strategy
      let opponent-strategy [random-strategy] of opponent
      if strategy != opponent-strategy [
        ifelse ((strategy - 1) mod 3) = opponent-strategy [
          record-victory opponent strategy
        ]
        [
          ask opponent [
            record-victory myself opponent-strategy
          ]
        ]
      ]
    ]
  ]
end

to record-victory [opponent strategy]
  hatch 1 [
    mutate strategy
    set heading random-float 360
    update-breed
  ]
  ask opponent [
    decrease-breed
    die
  ]
end

to mutate [strategy]
```

```
  ifelse mutation-favors-victor? [
    let mutation-amount random-float max-mutation
    if strategy = 0 [
      set rock rock + mutation-amount
      set paper paper - mutation-amount / 2
      set scissors scissors - mutation-amount / 2
    ]
    if strategy = 1 [
      set rock rock - mutation-amount / 2
      set paper paper + mutation-amount
      set scissors scissors - mutation-amount / 2
    ]
    if strategy = 2 [
      set rock rock - mutation-amount / 2
      set paper paper - mutation-amount / 2
      set scissors scissors + mutation-amount
    ]
    set rock max list 0 (min list 1 rock)
    set paper max list 0 (min list 1 paper)
    set scissors max list 0 (min list 1 scissors)
  ]
  [
    set rock rock + random-float max-mutation
    set paper paper + random-float max-mutation
    set scissors scissors + random-float max-mutation
  ]
  let total rock + paper + scissors
  set rock rock / total
  set paper paper / total
  set scissors scissors / total
end

to-report random-strategy
  let strategy 0
  let selector random-float 1
  ifelse selector < rock [
    set strategy 0
  ]
  [
    set selector selector - rock
    ifelse selector < paper [
      set strategy 1
    ]
    [
      set strategy 2
    ]
  ]
  report strategy
end

to-report dominant-strategy
  let strategy 0
  let max-value max (list rock paper scissors)
  if max-value = rock [
    set strategy 0
  ]
  if max-value = paper [
    set strategy 1
  ]
  if max-value = scissors [
    set strategy 2
  ]
  report strategy
end

;
;
;
;to make-new-turtle
;     hatch 1
;
;
;      [
;        forward 2
;        set heading random 360
;
;        let randomization-range random 4 ;; random 4 equals a random
value equals 0 to 3 to set number (eventually to 0 to 6 (I think)
;
;       let randomization-sign random 1
;       if(randomization-sign = 0)
;       [
;         let randomisation-sign -1
;       ]
;
;       let randomization-number randomization-range * randomization-
sign
;
;
;        ;; fight as ROCK
;       if(combat-type = 1) ;; fight-as-rock
;       [
;          ifelse r + randomization-number >= 0
;          [
;           set r r + randomization-number ;; sets rock to plus or
minus randomization-number amount
;          ]
;          [
;           set r 0 ;; else part of if/else
;          ]
;
;          let g1 g - (randomization-number / 2)
;          ifelse(g1 >= 0)
;          [
;          set g g1
;          ]
;          [
;            set g 0 ;; else part
;          ]
;
;          let b1 255 - (r + g)
;          ifelse(b1 >= 0)
;          [
;          set b b1
;           ]
;           [
;             set b 0
;           ]
;
;           set color ( list r g b )
;           set rock int ((20 / 51) * r)
;            set paper int ((20 / 51) * b)
;            set scissors int ((20 / 51) * g)
;            set rock-range rock
;
;    set scissors-range rock-range + paper
;
;        ]
;
;
;
;        ;; fight as PAPER
;        if(combat-type = 2) ;;fight-as-paper)
;        [
;           ifelse g + randomization-number >= 0
;           [
;            set g  g + randomization-number ;; sets rock to plus or
minus randomization-number amount
;           ]
;           [
;            set g 0 ;; else part of if/else
;           ]
;
;           let b1 g - (randomization-number / 2)
;           ifelse(b1 >= 0)
;           [
;           set b b1
;           ]
;           [
;             set b 0
;           ]
;
;           let r1 255 - (g + b)
;           ifelse(r1 >= 0)
;           [
;           set r r1
;           ]
;           [
;             set r 0
;           ]
;
;           set color ( list r g b )
;           set rock int ((20 / 51) * r)
;            set paper int ((20 / 51) * b)
;            set scissors int ((20 / 51) * g)
;            set rock-range rock
;
;    set scissors-range rock-range + paper
;        ]
;
;
```

```
;       if(combat-type = 3) ;;fight-as-scissors)
;        [
;           ifelse(b + randomization-number >= 0)
;           [
;            set b  b + randomization-number ;; sets rock to plus or
minus randomization-number amount
;           ]
;           [
;            set b 0 ;; else part of if/else
;           ]
;
;           let r1 r - (randomization-number / 2)
;           ifelse(r1 >= 0)
;           [
;           set r r1
;           ]
;           [
;             set r 0
;           ]
;           let g1 255 - (b + r)
;           set g 11
;
;           set color ( list r g b )
;
;            set rock int ((20 / 51) * r)
;            set paper int ((20 / 51) * b)
;            set scissors int ((20 / 51) * g)
;
;          set rock-range rock
;
;   set scissors-range rock-range + paper
;        ]
;
;       ]
;
;
;
;
;
;
;       set combat-type 0
;end
;
;
;
;
;
;to update-data
;
; ;set rock-graph ask turtles
;
;
;end
;
;
;
;to change-shape
```

```
;  ask turtles
;  [
;    if (rock > paper) and (rock > scissors)
;    [
;      set shape "pentagon"
;    ]
;    if (paper > rock) and (paper > scissors)
;     [
;       set shape "square"
;     ]
;     if (scissors > rock) and (scissors > paper)
;     [
;       set shape "triangle 2"
;     ]
;   ]
;
;end
```

References

Netlogo 5.0

http://www.randomizer.org/form.htm, 2008, Social Physiology Network

(http://www.theorie.physik.uni-

muenchen.de/lsfrey/members/group_leaders/erwin_frey/index.html), Erwin Frey, 2007

Bacteria Research, Jon Brown

E. Frey, U. C. Täuber,

Two-loop renormalization-group analysis of the Burgers–Kardar-Parisi-Zhang equation,

Phys. Rev. E **50**, 1024 (1994).

Acknowledgements

Thank you to:

Sponsoring Teachers:
Creighton Edington

Project Mentors:
Elizabeth Finley

For helping and guiding me through this entire project.