Troy Johnson (johns4ta)

Simulating the Effects of Food Resource Scavenging on Natural Selection Amongst Neanderthals and Homo sapiens

CPS691: Graduate Seminar

5/4/2015

Abstract

The primary cause of the extinction of the Neanderthal species is a topic of great debate amongst anthropologists, biologists, and all else who study Neanderthal existence. There exist many hypotheses that attempt to explain what led to the extinction of Neanderthals. Hypotheses include causes such as climate, disease, genocidal persecution by Homo sapiens, or that Neanderthals never truly came became extinct. Some researchers believe that Neanderthals never truly became extinct, as several studies have shown that Neanderthals genes are still present in our population's gene pool today, which many believe can be explained as a result of interbreeding between Neanderthals and Homo sapiens. In this paper, I examine how characteristics, such as the strength, intelligence, and speed of Neanderthals and Homo sapiens, may have helped or hindered them as they competed for food resources and interbred. Ultimately, I desire to seek which characteristics may have had the greatest impact on their ability to gather food resources and thus emerged as dominant traits in the population through natural selection. I examine this problem through the development and implementation of a genetic algorithm that attempts to simulate the basic traits of each species as they scavenged for food resources amongst each other 30,000 to 40,000 years ago when they first encountered each

other. Though my software needs a lot of work and feedback, the initial results seem to support the Endurance Running Hypothesis that has been proposed by some researchers.

Problem Domain

This problem lies at the intersection of the domains of computer science, biology, and anthropology. This problem lies in the domain of computer science because knowledge of programming and genetic algorithms is required in order to develop and implement the genetic algorithm and simulation environment required for simulating the gathering of food resources amongst the two species. Knowledge of biology and/or anthropology is required because a basic understanding of the physical differences between Neanderthals and Homo sapiens is required when developing and setting their traits and behaviors in the simulation environment. Background knowledge of what particular traits of each species may have had that most heavily aided in the gathering food resources during the Paleolithic period, approximately 30,000 to 40,000 years ago, is also essential as there are arguably many traits that could have possibly contributed to the traits of Homo sapiens pre-dominantly emerging over the traits of Neanderthals.

Problem Statement

Studies estimate that Neanderthals disappeared around 30,000 to 40,000 years ago [1], with a more recent 2014 study refining that time frame down to approximately 39,000 to 41,000 years ago [2]. The disappearance of Neanderthals is a fiercely debated and studied topic in the

fields of anthropology and biology. Biologists and anthropologists attempt to explain the disappearance of Neanderthals by studying their remains and other aspects of that time period, such as climate and its impact on their survival. Studies indicate that there appears to be an intersection in the time frames between when Neanderthals disappeared and when Homo sapiens first came into contact with Neanderthals approximately 30,000 to 40,000 years ago [3]. This suggests that there may have been some sort of connection between the appearance of Homo sapiens and the disappearance of Neanderthals. As stated earlier, there are many hypotheses for what caused the disappearance of Neanderthals such as climate [4], disease, genocidal persecution by the Homo sapiens [5], or perhaps Neanderthals never truly became extinct. Perhaps, interbreeding just merged the genetics of Homo sapiens and Neanderthals. A recent study showed that Neanderthals genes are still present within today's population [6]. It is estimated that up to 30% of Neanderthal's and Homo sapien's time was spent foraging and hunting for food resources [7]. Thus, the characteristics that led to the success in foraging and hunting by Neanderthals and Homo sapiens, probably had a significant effect on natural selection since the gathering of food resources is vital to survival and such a large proportion of their time was spent attempting to gather food resources. One hypothesis proposed, called the "Endurance Running Hypothesis" [8], proposes that the long distance running ability of Homo sapiens may have contributed to Homo sapiens being more effective at gathering food resources than Neanderthals. This physical difference may have been what ultimately led to the demise of Neanderthals or decline in the dominance of their genes in the gene pool, assuming interbreeding occurred between the two groups. Taking into account all this information, I would like to construct a genetic algorithm and implement it in a simulation environment that attempts to mimic the behavior of natural selection that may have occurred amongst Neanderthals and Homo

sapiens as they competed for food resources amongst each other and reproduced 30,000 to 40,000 years ago.

Physical Traits Simulated

There are many postulated differences in characteristics between Neanderthals and Homo sapiens, too many for me to account for in one simulation. As a result, I tried to focus on the characteristics that I believed would be most effective at helping individuals to gather food resources and those which are generally supported by most researchers as being significantly different amongst the two groups. Most modern research on the characteristics differences between Neanderthals and Homo sapiens seems to agree on many of the main physical differences between the two groups. Researchers typically agree that Neanderthals were physically stronger than Homo sapiens on average, but Homo sapiens were typically more intelligent and better at running longer distances on average [9][10]. As a result, for my simulation, I will be mimicking three characteristics of each species. These three characteristics include intelligence, strength, and speed (long distance running speed).

Desired Solution

The desired solution to this problem involved developing a simulation environment that implements my genetic algorithm for simulating Neanderthals and Homo sapiens gathering resources, while taking into account many of the characteristics of each species that I presented in the problem domain section. My simulation will attempt to simulate the time when

Neanderthals and Homo sapiens first began to co-exist in the same geo locations approximately 30,000 to 40,000 years ago and began to interbreed and live amongst each other. I will assume that interbreeding between the two groups occurs since previous research has shown it likely happened. However, I will not have any control over the interbreeding rate, whoever is selected for breeding based off their fitness value, will breed with their selected mate. There may not be any interbreeding that occurs during the selection phase, or it could turn out by pure chance that interbreeding occurs amongst every pair of parents that are selected for reproduction. Due to interbreeding, after the first generation I no longer classify individuals in the simulation as a Neanderthal or Homo sapien, but rather I just observe how their characteristics behave over time as a whole population. From my research, I was not able to find any hard-set numbers that researchers agree on in terms of the relative strength, intelligence, and speed proportions between Neanderthals and Homo sapiens. As a result, I made these characteristics changeable through sliders that the user can change when first setting up the simulation environment. This will allow the user to set the starting characteristics of each species to whatever they think most accurately reflects the true characteristics of each species.

Problem Instance

A problem instance for the use of this software could be a scientist, most likely in the fields of biology or anthropology, trying to figure out how food resource gathering may have affected natural selection amongst Neanderthals and Homo sapiens. Using this software, they could set all the initial characteristics of each species to what they believe the values should be, or what the most current research indicates they should be. After setting these initial values, the

researcher runs the simulation and observes how the characteristics of the population of Neanderthals and Homo sapiens behaves over time as they breed and compete for resources amongst each other. After the researcher starts to see the characteristics plateauing, or the desired number of generations has been reached, the researcher can stop the simulation and make their own conclusions based on the results, run the simulation with other values for the characteristics and see if they achieve any different results or conclusions, or keep running the simulation for as many more generations as they would like.

Proposed Algorithm

Below is brief overview of my algorithm, I go over the algorithm in greater detail below in the possible solution section.

```
setUp()
generation → 1
While generation < targetGeneration:
    While animals and plants left:
        Make Animals Wander
        For each person:
            If atTarget():
                actOnTarget()
                determineTarget()
            Else:
                Move towards target
    breedNewGeneration()
    calcStatistics()
    spawnNewFoodResources()
```

Increment generation

Possible Solution

The function setUp() is responsible for populating the world with the first generation of Neanderthals and Homo sapiens, along with the plant and animal food resources. The specific number of Neanderthals, Homo sapiens, plant food resources, and animal food resources that are populated into the world is based upon the number set by the user using the sliders. Once the set up function is run, the user clicks the button labeled "go" and the simulation will run until the user decides to stops it. I assume that users will want to run the simulation for a set number of generations and stop it once the simulation gets to the generation they desire or they can stop it at any other point along the way if they feel the characteristics are plateauing and the simulation does not need to keep running. For each generation, I simulate each individual competing against each other as they attempt to gather as many food resources as possible until there are no food resources left. Each person in the list of persons is iterated over and they must make a decision; if the person is at their food resource target, then they must attempt to harvest the food resource. If they are not at their food resource target, then they move closer towards their target. How a person goes about harvesting their resource depends on the type of food resource they are attempting to harvest. If it is a plant food resource, then the person harvests the resource with no problems and the total food they accumulated thus far is incremented based on the plant food value, which is controllable by the user through a slider. If the food resource target is an animal, then whether or not the person is successful in harvesting the animal food resource is based upon their strength value. I based the success of their hunt off their strength value since a person with higher strength may be able to throw a spear harder and further. In addition, they may be able to

physically overpower an animal much easier than a person with less strength. The strength value for a person is calculated by taking the sum of the bit array of ones and zeroes that is used to represent the person's strength genes. This summed strength value will end up being a value between 0 and 100 since there are 100 bits in the bit array representing their strength genes. Next, a random number between 0 and 100 is be generated, if this random value is less than the summed strength value for the person, the person is successful with their hunt and they are able to harvest the animal and collect the animal food value from it (the animal food value is also controllable through a slider). Individuals also get a food bonus based on their intelligence value. An individual's intelligence is determined by the sum of their intelligence bit array. This summed intelligence value will be a value between 0 and 100 since there are 100 bits in the bit array representing their intelligence genes. This summed intelligence value is then divided by 100 (resulting in a value between 0 and 1) and multiplied by the food gathered value from the food resource to determine the intelligence food bonus the person gets from this food resource. This food bonus is then added to their total food gathered for the current generation along with the food amount they gathered from the food resource target. Individuals get this intelligence based food bonus for plant food resources and animal food resources. If the person was not successful in their attempt to harvest the animal food resource, then they will attempt to do so again the next time they must make a decision, unless the animal has wandered too far away. In the event the animal has wandered too far away, the person must once again move towards the animal until they are close enough. Once an individual successfully harvests a food resource, they must make a decision on whether or not their next target is going to be a plant or animal food resource. This decision is based off from their strength value. I assume an individual with a higher strength value would most likely have a higher muscle mass and require more calorically

dense foods in order to meet their daily caloric needs. Therefore, to determine what type of food resource an individual sets as their target, a number between 0 and 100 is randomly drawn. If the random number is less than the individual's strength value (calculated by summing the bit array representing their strength), the individual's target is set to a randomly chosen animal food resource. Otherwise, the individual's food target is set to a randomly chosen plant food resource. If no animal food resources are left, then the individual just sets their target to a randomly chosen plant food resource. Likewise, if no plant resources are left, then the individual just sets their target to a randomly chosen animal food resource. If the person was not even at their target resource yet in the first place, then the decision they make is to move towards their target. The distance they move towards their target is based off from their speed and strength values. Their speed and strength traits are represented by bit arrays of 1's and 0's and the speed and strength values are determined based off from the sum of the corresponding bit arrays. To determine the distance the person moves forward each time, the quotient between the two sums of the bit arrays is taken. In other words, the sum of the speed bit array is divided by the sum of the strength bit array. This creates a sort of balancing relationship between the speed of an individual and their strength. This relationship can be explained physiologically because as an individual achieves a higher strength level, they likely pack on more muscle mass and are thus slower due to an assumed increase in mass. When an individual harvests a food resource, the food resource disappears from the world. Once all food resources are gone from the world, individuals are selected for breeding based off from their fitness value and crossover (reproduction) takes place. The fitness value is a measure of how well a person performed, which in this case is the total amount of food gathered. Each section below covers a stage of the process for reproduction.

Selection

Selection is the process through which individuals are selected to be the parents for reproduction (crossover). Tournament selection is the method I use to determine who is selected for crossover. Tournament selection works by first randomly selecting n individuals from the population with n being the tournament size. In the case of my software project, I use a tournament size of four. From those four individuals, the individual with the highest fitness value is selected as a parent for crossover. This is repeated until the desired number of parents needed to breed the desired number of new individuals is met. The number of individuals that are to be bred is based off from the percent the user sets the crossover rate slider too. For example, if there are one-hundred people and the crossover rate is set to twenty; twenty new people will be bred. Once the parents have been selected, crossover takes place.

Crossover

Crossover is the process of merging the genetic information from two parents to create a new individual. The method of crossover is used is called single-point crossover, I will explain how my implementation of single-point crossover works below. For the sake of following example, I will refer to parents as P1 and P2 and children as C1 and C2 in my explanation for how crossover in my implementation works. Single point crossover works by first selecting a random split point along which to split the bit array of each parent. Once the split values are determined, C1 and C2 are assigned the values in their bit arrays based upon the split point. C1 receives the contents to the left of the split point in P1's bit array along with the contents to the right of the split point in the bit array of P2. Similarly, C2 receives the contents to the right of

the split point in P1's bit array along with the contents to the left of the split point in the bit array of P2. Once crossover takes place, mutation then takes place on the newly generated individual's bit arrays.

Mutation

Mutation accounts for the random deformation of genes that can occur during reproduction. The mutation rate percent is controllable by the user through a slider. Mutation rate controls the probability that any given bit in a newly bred individual's bit array will encounter a mutation (be changed from a zero to a one or vice versa).

Replacement Strategy

Replacement strategy is the strategy used for incorporating the newly bred individuals back into the population, while eliminating some of the individuals from the previous generation. First of all, the newly bred children constitute the base of the new generation. Next, I used a method similar to the tournament selection I used during the selection phase to determine who will constitute the rest of the population. Four individuals are selected at random from the population and the individual with the highest fitness level out of those four individuals is cloned from the old generation into the new generation. This is repeated until the desired number of individuals in the new generation is met. Once the new generation is created, basic statistics are calculated about the population such as the average strength, average speed, and average

intelligence. These values are then plotted after each generation finishes so the user can see how the traits behave over time and see if any patterns crop up.

Results

The results from most runs of my implementation seem to support the Endurance Running Hypothesis that I previously mentioned in the problem statement section. It seems that almost no matter what I set the initial slider values too, speed always seems to rise above intelligence and strength as the dominant characteristic. An example of this output can be seen below in Figure 1 from a simulation run where I ran it for 500 generations. However, I believe there is still much work and discussion that needs to be done on this software in order to deem this simulation as evidence to support the Endurance Running Hypothesis. Perhaps the way my simulation is set up is not completely fair and speed gives individuals a bigger advantage than what there actually was at that time. I will discuss these possible problems, improvements, and future work in the next section.

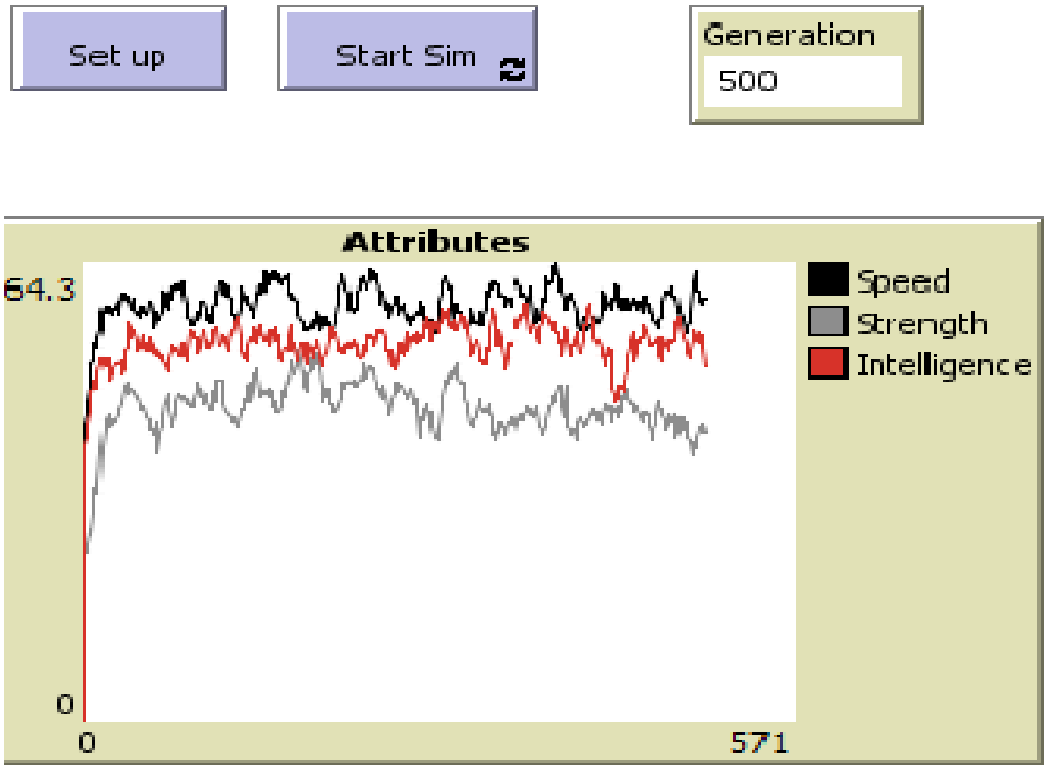

Figure 1: Sample output from running simulation for 500 generations.

Future Work and Improvements

I believe there is a lot of room for improvements to my software project. At the beginning of the semester, I met with Dr. Rachel Caspari, a professor at Central Michigan University in the anthropology department, after some research indicated to me that she might be a good resource to start with for gathering information. Dr. Caspari is considered in high regard in the field of Neanderthal studies and it sounded like she is still actively involved in research efforts based off her academic background. Dr. Caspari was a great resource for starting my project as she was able to suggest some great resources and things to look into, but she could not give me much advice on the genetic algorithm side of things. She mentioned that she could not think of anything like this that has been developed before and that it might be a very difficult

thing to do due to the sheer number of factors that are believed to have had an impact on Neanderthal disappearance. Therefore, I think the next best step would be to present my project to a computer scientist with expertise in this field or an evolutionary biologist who has experience in genetic algorithms or related field and see what they would have to suggest. I would like to see if they had any suggestions to build upon the simple relationship I have between speed and strength in my simulation as well as any other relationships that they would recommend considering.

Future work could also incorporate new traits into the simulation or delve into other factors such as disease and climate, which studies indicate may also been factors that led to the extinction of Neanderthals. Dr. Caspari also mentioned she believes that adaption to the climate changes and utilization of resources may have played large roles in Neanderthal extinction and I believe these may tie into intelligence somehow. In my current simulation, intelligence does tie into utilization of resources since individuals get a bonus for the food resources they gather and this bonus is based on their intelligence value. This was because I assumed that if they are more intelligent they make be able to better utilize the resources they gather and get value more out of them. Something else that I would like to do is try and figure out some way to balance intelligence with another attribute such as there is with the relationship between strength and speed in my current simulation. An additional possibility that I would like to add in is to allow users to automate their simulation to run as many times as they would like. For example, a user would be able to script their simulation to run for 100 times for 4000 generations each time and the results from each run would be saved to a text file for analyzing later. This would allow users to determine how statistically significant their results are. NetLogo provides built in support for this feature, but I ran out of time to implement it unfortunately.

Results Comparison

I was not able to find any other attempts at developing software to attempt what I was trying to attempt, so I do not have any other results to compare with.

Software and Hardware Platforms

The software that I used to develop my program with is called NetLogo and I used version 5.1.0 of NetLogo. NetLogo is a free tool offered by Northwestern University's Center for Connected Learning and Computer-Based Modeling and is available for download at https://ccl.northwestern.edu/netlogo/. NetLogo can be installed and run on most Mac, Windows, or Linux machines. The specific minimum hardware requirements are not provided by NetLogo, but it should be runnable on most modern day desktop and laptop computers. NetLogo was built from Java and Scala and thus requires that a Java virtual machine be installed in order to develop and run any simulations.

Implementation Details

To use the simulation environment I developed, a user must first download and install NetLogo and Java virtual machine. Once both are downloaded and installed, my NetLogo project be viewed by launching NetLogo, navigating to the file menu in the upper left hand corner, and clicking on the option named "open" to open up the source code file provided (.nlogo file). Once the environment loads, there should be a several sliders visible along the right-hand

side of the screen that the user can use for setting specific details for the simulation. Please refer to Table 1 for a description of the purpose of each slider.

Slider Label	Description	Number in Figure 2
cross-over rate	Sets the percent of people that are chosen for reproduction each generation.	1
mutation-rate	Specifies the mutation rate or percent chance any arbitrary gene will be mutated after crossover occurs.	2
animal-food-value	Specifies the amount of food value an individual receives from harvesting an animal resource.	3
plant-food-value	Specifies the amount of food value an individual receives from harvesting a plant resource.	4
number-of-plants	Specifies the number of plant food resources that are placed in the world.	5
number-of-animals	Specifies the number of animal food resources that are placed in the world.	6
init-number-neanderthals	Specifies the number of Neanderthals that are placed on the map.	7
starting-neanderthal-strength	Specifies the starting value for the average Neanderthal strength.	8
starting-neanderthal-speed	Specifies the starting value for the average Neanderthal speed.	9
starting-neanderthal-intelligence	Specifies the starting value for the average Neanderthal intelligence.	10
initial-number-homosapiens	Specifies the number of Homo Sapiens that are placed on the map.	11

starting-homosapien-strength	Specifies the starting value for the average Homo Sapien strength.	12
starting-homosapien-speed	Specifies the starting value for the average Homo Sapien speed.	13
starting-homosapien-intelligence	Specifies the starting value for the average Homo Sapien intelligence.	14

Table 1: Description of each sliders purpose.

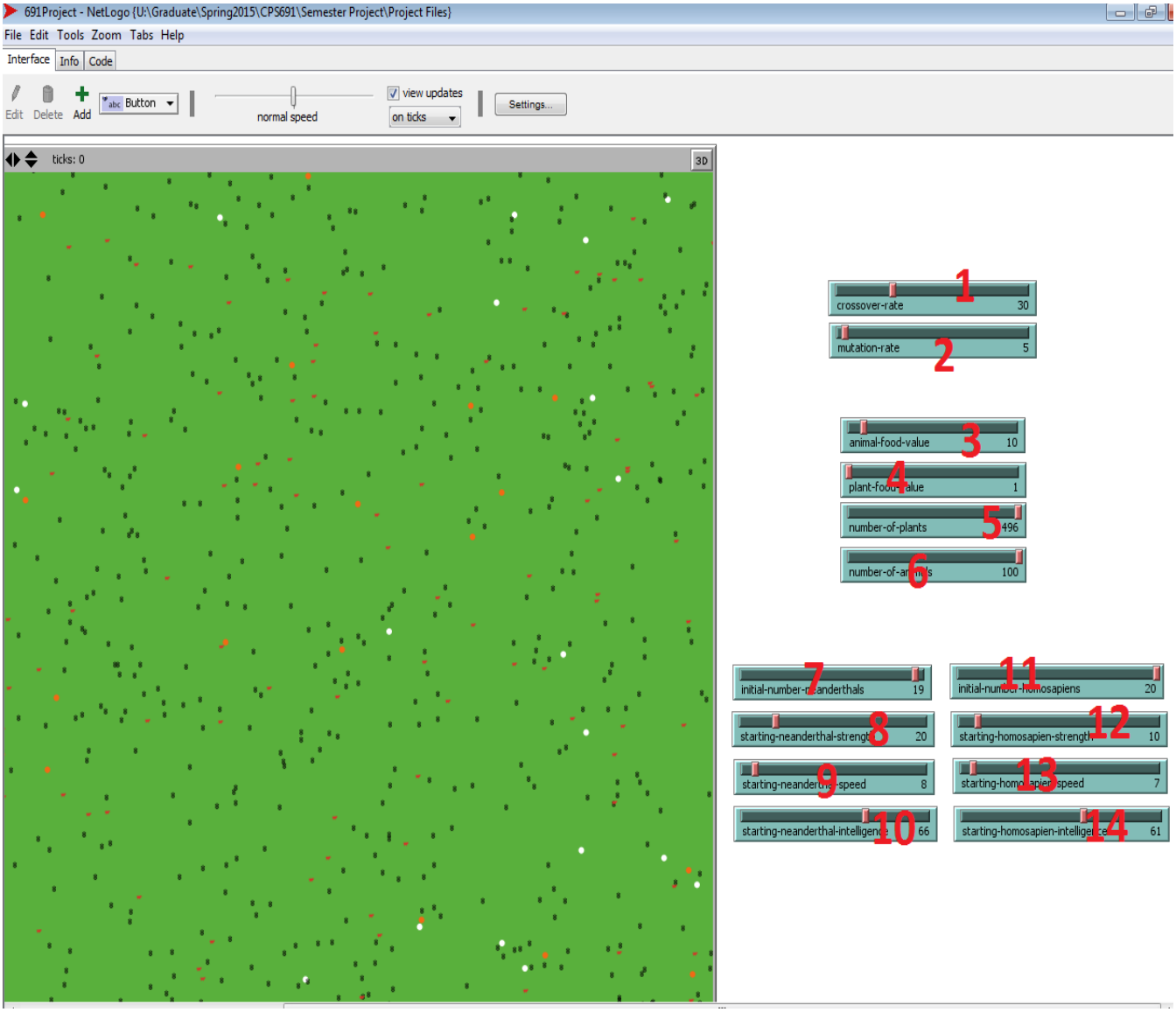

Figure 2: View of the set up sliders

Once each setting has been set to what the user desires (the settings do not have to be changed, the defaults can be used), the button labeled “Set up” must be clicked to populate the initial world with food resources, Neanderthals, and Homo sapiens as specified by the values set by the sliders. The Neanderthals are represented by the white circles, the Homo sapiens are represented by the orange circles, the red cows represent the animal food resources, and the black plants represent the plant food resources. Please note that after the first round of the simulation, interbreeding between Neanderthals and Homo sapiens can occur and thus I do not classify the shapes as orange (Homo sapiens) or white (Neanderthals), I just set their colors to yellow. At any point during the simulation, a user can right click on an individual and click “Inspect Person “ followed by their number, and view all the individual characteristics for that person. The set up button is indicated in Figure 3 by the button with labeled with a red “1”. Once the set up button has been clicked, the map should visibly populate. Note, if the world (the world indicated in Figure 3 with a red “5”) is populating very slowly, there is a slider at the top that can speed up the process. This slider also controls the speed at which the simulation runs at and this slider is indicated in Figure 3 labeled with a red “3”. Once the world has been populated, the simulation can be started by pressing the button labeled “Start Sim”, this button is indicated in Figure 3 as the button with a red “2” printed over it. Once the simulation starts, each individual starts gathering resources using the methodology described in the possible solution section. Once all the food resources are gathered, selection, crossover, mutation, etc. take place and a new generation is produced. To track what generation the simulation is currently on, the user can refer to the monitor labeled “Generation” that is indicated in Figure 3 with the red “6”. As each generation comes and goes, the averages of their attributes (strength, intelligence, and speed) are computed and plotted in the plot labeled “attributes” and indicated by a red “4” in

Figure 3. The speed at which the simulation runs can be controlled using the slider indicated by a red "3" in Figure 3. Also, the speed of the simulation can additionally be sped up by disabling visual updates, which can done by unchecking the box indicated by a red "7" in Figure 3.

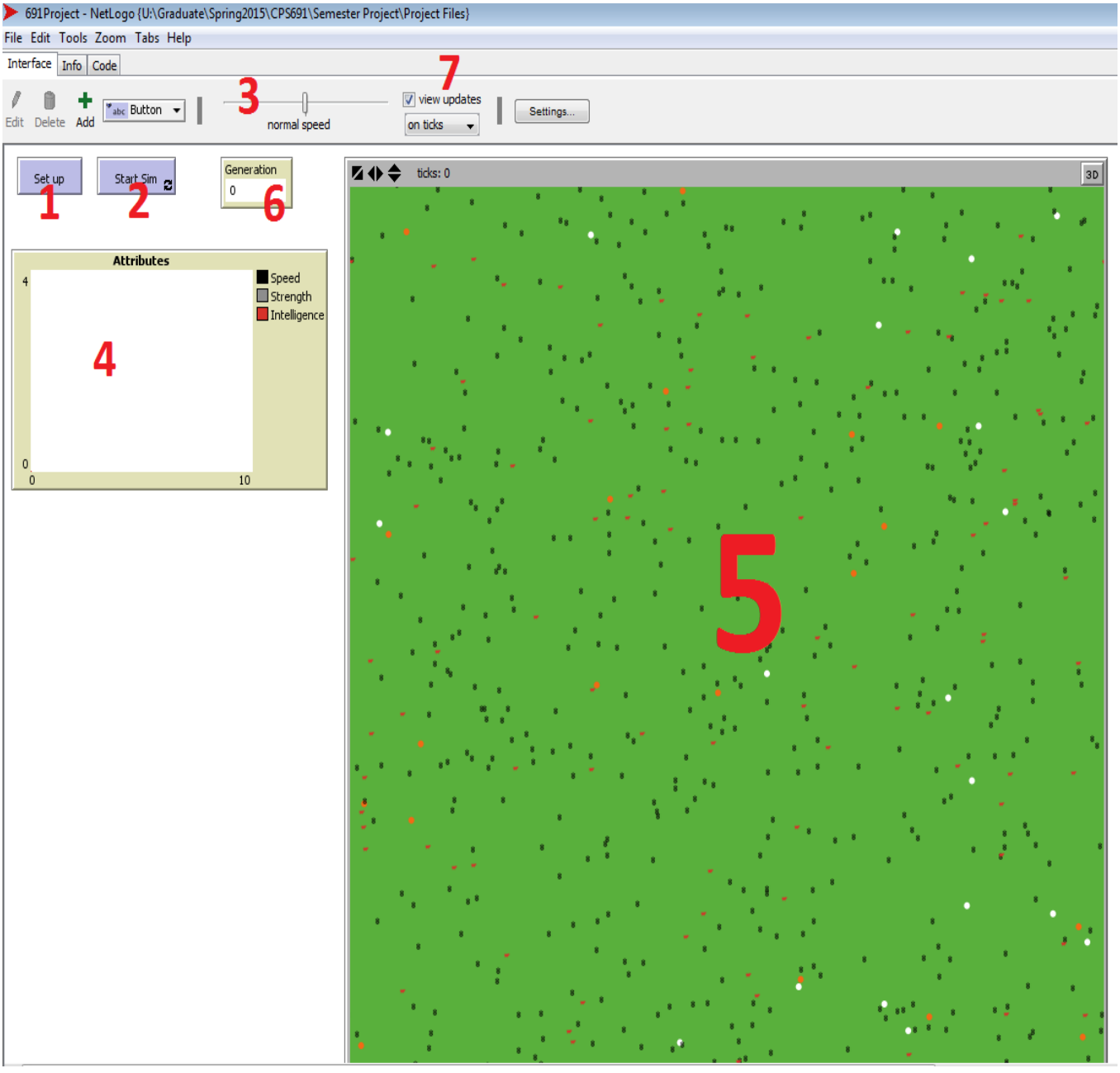

Figure 3: View of simulation environment once set up has been run.

Source Code

```
breed[people person]

breed[plants plant]

breed [animals animal]


globals[GENES_LENGTH currentGeneration numPeople meanSpeedLevel meanStrengthLevel
meanIntelligenceLevel]

people-own[intelligenceLevel intelligenceGenes speedLevel speedGenes speed strengthLevel
strengthGenes target foodGathered]

plants-own[ foodCapacity ]

animals-own[ foodCapacity ]



;Once the user presses go to start the simulation, this function is called and the contents of the
function are looped over until the

;user presses the go button again to stop the simulation. First, each animal is asked to wander.
Next, each person is iterated over

;and if they are at their target, then actOnTarget is called for them to act on the target and then a
new target is determined. If

;a person is not at their target yet, then they move towards their target, with the distance they
move towards their target being

;determined by their speed trait. If no plants or animals are left, then the current generation ends,
the new generation is created,

;new resources are populated into the world, and average values of traits are calculated for the
purpose of plotting.


to go

  ask animals

  [

    wander
```

```
]
ask people
[
  if isPlantsLeft or isAnimalsLeft
  [
    ifelse target != nobody
    [
      ifelse not atTarget
      [
        set heading towards target
        jump speed
      ]
      [
        actOnTarget
      ]
    ]
    [
      determineTarget
    ]
  ]
]

tick

if not isPlantsLeft and not isAnimalsLeft
[
  breedNewGeneration
```

```
    populateResources
    set currentGeneration currentGeneration + 1
    calculateAverages
  ]
end



;Sets up the initial world by clearing anything currently on the map, resetting the ticks to 0, and then setting all the patch colors to green. Next,
;the default shapes are set for each type of turtle (person, plant, and animal) and populateResources is called to populate the resources into the world.
;Lastly, the initial number of Neanderthals and Homosapiens are created and place on the map.

to setup

  clear-all
  reset-ticks
  ask patches
  [
    set pcolor green
  ]

  set-default-shape people "circle"
  set-default-shape plants "plant"
  set-default-shape animals "cow"

  set GENES_LENGTH 100
  set numPeople initial-number-neanderthals + initial-number-homosapiens
```

```
  populateResources
  create-people initial-number-neanderthals [ populateNeanderthal]
  create-people initial-number-homosapiens[ populateHomosapien]
end



;Determines the new target food resource for a person based on their strength level. The higher the strength level, the more likely a person is to want
;to target an animal. To determine the type of food resource target, a random number between 0 and 100 is drawn, if it less than the strength value for
;the person, then the person sets their target to an animal. If no animals are left, then a person sets their target to a plant resource. If the randomly
;drawn value is greater than the strength value for the person, then the person sets their target to a plant resource. If no plant resources are left, then
;the person sets their target to an animal resource. Once the target has been determine, the individuals heading is set towards their target so when they move,
;they move in the direction of the target.


to determineTarget
  ifelse (random 100) < strengthLevel
  [
    ifelse isAnimalsLeft
    [
      set target one-of animals
    ]
    [
      if isPlantsLeft
      [
        set target one-of plants
```

```
      ]
    ]
  ]
  [
    ifelse isPlantsLeft
    [
      set target one-of plants
    ]
    [
      if isAnimalsLeft
      [
        set target one-of animals
      ]
    ]
  ]
  set heading towards target
end
```

```
;Called on by persons to act on their target when they are within close enough range. If the target is an animal, then a random number is drawn between

;0 and a 100. If the randomly drawn value is less than the strength value of the person, then the food resource is harvested and the target resource

;dies and disappears from the world. If the target is an animal, then the plant is harvested, the person accumulates the food gained from the plant resource,

;and the plant dies and disappears from the world. If a harvest is successful, then the person also gets a food bonus based on their intelligence level

;added to their food gained. Their intelligence level is divided by a 100 (resulting in a number between 0 and 1) and multiplied by the food gained from the
```

;harvested food resource to compute the food bonus, which is added to the food gained from the harvested food resource.

```
to actOnTarget

  let strength strengthLevel
  let intelligence intelligenceLevel
  let foodGained 0

  ask target
  [
    ifelse breed = animals
    [
     if (random-float 100.0 < strength)
     [
       set foodGained (animal-food-value * (1 + intelligence / 100))
       die
     ]
    ]
    [
      set foodGained (plant-food-value * (1 + intelligence / 100))
      die
    ]
  ]
  set foodGathered foodGathered + foodGained
end
```

```
;Breeds the next generation of people. Based on the crossover rate, the number of parents needed is calculated. Then using tournament selection

;with a tournament size of 4, parents are selected by selecting the person with the highest food gathered value as a parent. Next, crossover takes

;place by calling geneticCrossover for each bit array to determine a split point and split the arrays based on that split point. Next, parts of the

;bit arrays are combined from each parent to form the children and mutation takes place on the new children. A similar method of tournament selection

;is used to clone people from the old generation to the new generation until the desired population level is reached and then the people from the old

;generation who didn't make it to the new generation die off.


to breedNewGeneration

  let oldPeople people with [true]

  let crossover-count  (floor (numPeople * crossover-rate / 100 / 2))


  repeat crossover-count

  [

    let parent1 max-one-of (n-of 4 oldPeople) [foodgathered]

    let parent2 max-one-of (n-of 4 oldPeople) [foodgathered]


    let newSpeedGenes geneticCrossover ([speedGenes] of parent1) ([speedGenes] of parent2)

    let newIntelligenceGenes geneticCrossover ([intelligenceGenes] of parent1) ([intelligenceGenes] of parent2)

    let newStrengthGenes geneticCrossover ([strengthGenes] of parent1) ([strengthGenes] of parent2)


    ask parent1

     [

       hatch 1
```

```
      [
        set speedGenes item 0 newSpeedGenes
        set intelligenceGenes item 0 newIntelligenceGenes
        set strengthGenes item 0 newStrengthGenes
      ]
    ]
   ask parent2
    [
      hatch 1
       [
         set speedGenes item 1 newSpeedGenes
         set intelligenceGenes item 1 newIntelligenceGenes
         set strengthGenes item 1 newStrengthGenes
       ]
    ]
]

repeat (numPeople - crossover-count * 2)
[
  ask max-one-of (n-of 4 oldPeople) [foodgathered]
  [
    hatch 1
  ]
]

ask oldPeople
[
```

```
    die
  ]

  ask people
  [
    mutation
    set foodGathered 0
    setxy random-xcor random-ycor
    set color yellow
    calculateAttributes
  ]
end



;Iterate through the bit arrays holdsin the values for speed, strength, and intelligence and for each value in the bit array, generate a random

;number, the the random number is less than the mutation-rate number set by the slider, then flip the bit. This accounts for random mutations

;that can occur.


to mutation


  set speedGenes map
  [
    ifelse-value (random-float 100.0 < mutation-rate)
    [
      1 - ?
    ]
```

```
  [
    ?
  ]
]
speedGenes

set strengthGenes map
[
  ifelse-value (random-float 100.0 < mutation-rate)
 [
   1 - ?
 ]
 [
   ?
 ]
]
strengthGenes

set intelligenceGenes map
[
  ifelse-value (random-float 100.0 < mutation-rate)
   [
     1 - ?
   ]
   [
     ?
   ]
```

```
  ]
  intelligenceGenes
end



;Generate a random number between 0 and the length of the bit array to serve as the splitting point. Split the bit
;array based on this value. Return a list of the sublists to the calling function.
to-report geneticCrossover [p1 p2]
  let split-point 1 + random (length p1 - 1)
  report list (sentence (sublist p1 0 split-point)
                        (sublist p2 split-point length p2))
              (sentence (sublist p2 0 split-point)
                        (sublist p1 split-point length p1))
end



;Fill the Homo sapien's bit arrays that represent their based on the values set in the sliders for Homo sapiens.

to setInitialHomoSapienGenes
    let genes []
    let x 0

    while[x < GENES_LENGTH]
    [
      ifelse (random 100) < starting-homosapien-strength
```

```
  [
    set genes lput 1 genes
  ]
  [
    set genes lput 0 genes
  ]
  set x x + 1
]
set strengthGenes genes

set genes []
set x 0
while[x < GENES_LENGTH]
[
  ifelse (random 100) < starting-homosapien-speed
   [
     set genes lput 1 genes
    ]
    [
      set genes lput 0 genes
     ]
 set x x + 1
]
set speedGenes genes

set genes []
set x 0
```

```
  while[x < GENES_LENGTH]
  [
    ifelse (random 100) < starting-homosapien-intelligence
     [
       set genes lput 1 genes
     ]
     [
       set genes lput 0 genes
     ]
   set x x + 1
  ]
  set intelligenceGenes genes
end



;Fill the Neaderthal's bit arrays that represent their based on the values set in the sliders for Neanderthals.


to setInitialNeanderthalGenes
  let genes []
  let x 0


  while[x < GENES_LENGTH]
  [
    ifelse (random 100) < starting-neanderthal-strength
     [
       set genes lput 1 genes
     ]
```

```
    [
      set genes lput 0 genes
    ]
  set x x + 1
]
set strengthGenes genes

set genes []
set x 0
while[x < GENES_LENGTH]
 [
    ifelse (random 100) < starting-neanderthal-speed
     [
       set genes lput 1 genes
     ]
     [
       set genes lput 0 genes
     ]
  set x x + 1
]
set speedGenes genes

set genes []
set x 0
while[x < GENES_LENGTH]
 [
    ifelse (random 100) < starting-neanderthal-intelligence
```

```
        [
          set genes lput 1 genes
        ]
        [
          set genes lput 0 genes
         ]
       set x x + 1
    ]
   set intelligenceGenes genes
end


;Calculates the characteristics for a person based of their gene arrays. A person's speed is set to the sum of their speed bit array
;divided by the sum of their strength bit array. A person's strength level is set to the sum of their strength bit array and a person's
;intelligence level is set to the sum of their intelligence bit array.


to calculateAttributes
  set speedLevel sum SpeedGenes
  set speed speedLevel / (sum strengthGenes)
  set strengthLevel sum strengthGenes
  set intelligenceLevel sum intelligenceGenes
end



;Calculates the average traits amongst the populations for plotting purposes. Averages of the speed, strength and intelligence levels
;are all taken.
```

```
to calculateAverages
  set meanSpeedLevel mean[speedLevel] of people
  set meanStrengthLevel mean[strengthLevel] of people
  set meanIntelligenceLevel mean[intelligenceLevel] of people
end


;Populates the world with the initial animal and plant resources. Each animal resource is placed at a random xy coordinate in the world
;and their color is set to red. Each plant resource is placed at a random xy coordinate in the world and their color is set to black.

to populateResources
   create-animals number-of-animals
  [
    set color red
    setxy random-xcor random-ycor
  ]


  create-plants number-of-plants
  [
    set color black
    setxy random-xcor random-ycor
  ]
end
```

```
;Creates each Neanderthal and sets their position to a random x-coordinate and random y-coordinate. Next, setInitialNeanderthalGenes is called

;to form their genes by filling in their bit array values. Finally, calculateAttributes is called to calculate their traits and set them, and

;determineTarget is called to determine what the inital target the Neanderthal will head towards is.


to populateNeanderthal

   set color white

   setxy random-xcor random-ycor

   setInitialNeanderthalGenes

   calculateAttributes

   determineTarget

end



;Creates each Homo sapien and sets their position to a random x-coordinate and random y-coordinate. Next, setInitialHomoSapienGenes is called

;to form their genes by filling in their bit array values. Finally, calculateAttributes is called to calculate their traits and set them, and

;determineTarget is called to determine what the inital target the homosapien will head towards is.


to populateHomosapien

   set color orange

   setxy random-xcor random-ycor

   setInitialHomoSapienGenes

   calculateAttributes

   determineTarget
```

```
end


;Determines if person is within one speed length of their target, if they are, the person's xy coordinates are set to that of their target otherwise
;the person will often continually jump around their target until their coordinates match exactly that of their target. If the person is at their
;target, true is returned, otherwise false is returned.


to-report atTarget
  let target-xcor 0
  let target-ycor 0

  ask target
  [
   set target-xcor xcor
   set target-ycor ycor
  ]

  if distance target < speed
  [
    set xcor target-xcor
    set ycor target-ycor
    report true
  ]
  report false
end
```

```
;Check if any animal food sources are left. Return true if any are left, otherwise false is returned.

to-report isAnimalsLeft
  ifelse any? animals
   [
     report true
   ]
   [
     report false
   ]
end


;Check if any plant food sources are left. Return true if any are left, otherwise false is returned.

to-report isPlantsLeft
  ifelse any? plants
   [
     report true
   ]
   [
     report false
   ]
end
```

```
;Tells the person or animal to wander by turning right a random number of degrees between 0 and 90, turning left a random number of degrees

;between 0 and 90, and then heading forward 0.1 patches.


to wander

  rt random 90

  lt random 90

  fd 0.1

end
```

References

[1] Herrera KJ, Somarelli JA, Lowery RK, Herrera RJ (2009) To what extent did Neanderthals and modern humans interact? Biol Rev Camb Philos Soc 84(2):245–257

[2] Higham, T., Douka, K., Wood, R., Bronk Ramsey, C., Brock, F., Basell, L., Camps, M., Arrizabalaga, A., Baena, J., Barroso-Ruíz, C., Bergman, C., Boitard, C., Boscato, P., Caparros, M., Conard, N.J., Draily, C., Froment, A., Galv an, B., Gambassini, P., Garcia-Moreno, A., Grimaldi, S., Haesaerts, P., Holt, B., Iriarte-Chiapusso, M.J., Jelinek, A., Jorda Pardo, J.F., Maíllo-Fern andez, J.M., Marom, A., Maroto, J., Menendez, M., Metz, L., Morin, E., Moroni, A., Negrino, F., Panagopoulou, E., Peresani, M., Pirson, S., de la Rasilla, M., Riel-Salvatore, J., Ronchitelli, A., Santamaria, D., Semal, P., Slimak, L., Soler, J., Soler, N., Villaluenga, A., Pinhasi, R., Jacobi, R., 2014. The timing and spatiotemporal patterning of Neanderthal disappearance. Nature 512, 306-309.

[3] First genocide of human beings occurred 30,000 years ago. Pravda. 24 October 2007. 18 May 2009.

[4] Francisco J. Jiménez-Espejo, Francisca Martínez-Ruiz, Clive Finlayson, Adina Paytan, Tatsuhiko Sakamoto, Miguel Ortega-Huertas, Geraldine Finlayson, Koichi Iijima, David Gallego-Torres, Darren Fa, Climate forcing and Neanderthal extinction in Southern Iberia: insights from a multiproxy marine record, Quaternary Science Reviews, Volume 26, Issues 7–8, April 2007, Pages 836-852.

[5] Policarp Hortolà, Bienvenido Martínez-Navarro, The Quaternary megafaunal extinction and the fate of Neanderthals: An integrative working hypothesis, Quaternary International, Volume 295, 8 May 2013, Pages 69-72.

[6] S. Sankararaman, S. Mallick, M. Dannemann, K. Prufer, J. Kelso, S. Paabo,N. Patterson, D. Reich The genomic landscape of Neanderthal ancestry in present-day humans Nature, 507 (2014), pp. 354–357.

[7] Than, K. (2010). Neanderthals, humans interbred - first solid DNA evidence. National Geographic Daily News. Retrieved March 2, 2015.

[8] Bramble, Dennis; Lieberman, Daniel (November 2004). "Endurance running and the evolution of Homo". Nature 432.

[9] Domínguez-Rodrigo, M., 2002. Hunting and scavenging by early humans: the state of the debate. Journal of World Prehistory 16, 1–54.

[10] Lieberman D.E., Bramble D.M., Raichlen D.A., Shea J.J. (2009) Brains, brawn and the evolution of human endurance running capabilities. In The First Humans: Origin and

Early Evolution of the Genus Homo. Grine F.E., Fleagle J.G., Leakey R.E., eds. New York: Springer, pp. 77-98.