**QUIZ 3 NOTES – BOONE TISON**

**Introduction to NetLogo:**

GUI – anything other than plain text, includes visual indicators, buttons, etc. Convenient for experts but needed to support usability.

Commands – short piece of code, when we type a command, the program calls the piece of code

User-defined Procedures – go, setup, etc. Command created by the user

Comments – start with ;; illustrates the user’s thought process for others. Good comments: summarize a large piece of code, translate specific bits in human language, explain decisions. Bad comments: are misleading, repeat the code, how explains how not why, makes little sense for anyone but the person who wrote them

All models have two main parts: 1. Initialization (setup), assigns the baseline value of all entities. 2. Computes simulation results (go), updates entities values.

Topology – formed by the set of cells and their connections, depends on whether your neighbors with cells sharing a side or corner. The concept of neighbors is flexible.

Neighborhoods – sides only (Von Neumann), sides and corners (Moore), 2-radial (sides x 2), 2-axial (corners x 2)

Periodic Boundaries – cells wrap up horizontally and vertically. Looks like a grid but is actually a torus

Direct neighbors can be based on: Closed or periodic boundaries. Whether cells share sides only or sides and corners. The shape of cells. The distance.

States – Describe states -> encode states -> visualize states, NetLogo bypasses this. Proper process: tell NetLogo that cells have states -> assign these states -> map states to colors

How does the computer store states? – everything is stored as 0 and 1. Transistors: current on or current off

How do you encode numbers using only 0s and 1s? – only have two signs, position stands for a power of 2. 0-1, 1-2, 2-4, 3-8, 4-16, 5-32, 6-64, 7-128, 8-256, 9-512, 10-1024. If you are given a decimal number: subtract the largest powers of 2 OR repeatedly divide the number by 2

How do we store binary? – computer storage is abstracted as linear; the storage is always 1D. To map the linear storage organization to actual data, you need to know how many rows (**horizontal**) and columns (**vertical**) you have. First two bits tell you your columns and rows and then the numbers follow.

History – Jacquard Loom (1801): a machine to make textiles/tissues. Punched card controls the pattern. Only two possible values, hole or solid.

Babbage’s Analytic Engine (1834): used punch cards to represent instructions and also has punch cards for variables, has a memory.

Hollerith (1890): won a prize to use punch cards for the purpose of computation, he created IBM. Was an electro-mechanical machine that used relays (open or closed). 3 operations per second

Central Processing Unit: made very small due to microprocessors that can fit lots of transistors on a very small space

Hard drives: when you look at one part, it is magnetized in one direction or the other. Only have two different values on a hard drive.

**Controlling Entities:**

If command – block of code is within []. Can write if within if. *Random 100 < 5*, *ticks >= 20*.

*Can-move?* Distance – can the agents keep moving the distance.

*Any?* Condition – is there any agent satisfying the condition.

*All?* Condition – are all the agents satisfying the condition.

Halting condition – how the code is stopped

Variables – names that are associated with a value. Its value may change.

Constants – names associated to the same value. Could be a universal constant such as pi or set within NetLogo

Integers – for calculations, 79, transform: + \* / -, compare: < <= >= >

Floating Points – for calculations, 17.158, transform: + \* / -, compare: < <= >= >

Booleans – for conditions, true or false, transform: or and, compare: == !=

Strings – for names, “Hello”, transform: concatenate, compare: == !=

Variable commands:

Integers - NetLogo completes it by assuming you want to *show* the result, Operators have a priority (PEMDAS), Space matters, the command can fail without proper spacing

Strings - When you have typed a complete string, it shows up in brown, Operator overloading: the behavior of an operator changes based on the types you give it (Not done in NetLogo), Use the wordcommand to combine strings, use substringto take from first number place to second number place - 1

Where do variables come from: Variables are created for a purpose and die after serving. In between we can use them. That’s their scope.

Globals – general simulation parameters

Turtles-own – tracking agent characteristics

Patches-own – tracking cell characteristics

Variable are created for parts of the model. Is it a characteristic of a cell, agent, or world?

The *let* keyword makes a local variable, the scope is limited to the block of code where it’s made

When using the GUI, it creates global variables:

Slider – produces a number within the range set in the “slider” dialogue

Switch – produces a Boolean

Chooser – can produce any data type, depending on the list of values entered in the “chooser” dialogue

Input – can produce a number, string, or color, depending on the set data type given in “input” dialogue

Randomness:

Set height 165 + random 30 – 15, Set weight 72 + random 70 – 35

Percent: (patches of choice / total patches) \* 100

Ifelse condition:

[do if condition is satisfied]

[do if condition is not satisfied]