Lachlan Sinclair

sinclala@oregonstate.edu

5/1/2020

CS 475 Spring

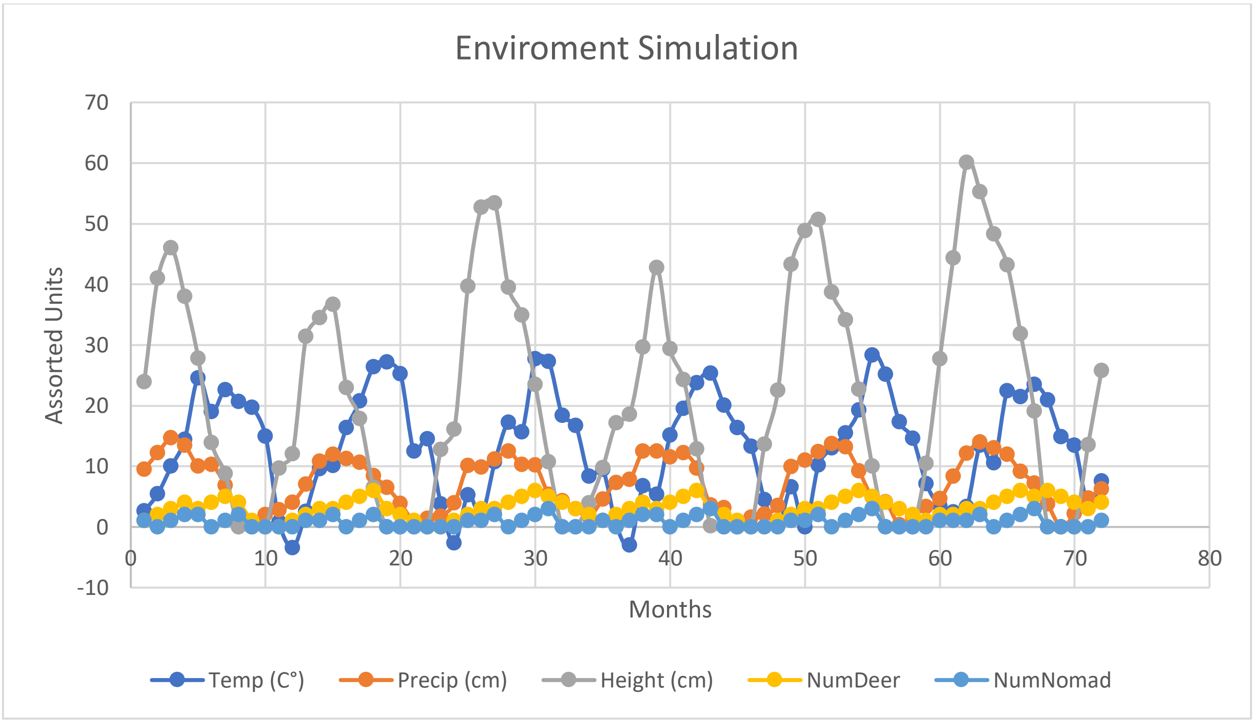
Project #3: Functional Decomposition

System: I used my personal computer for this project which uses a Windows OS. The program was developed in Visual Studios.

Results:

Month	Temperature (C)	precipitation (cm)	height of grain (cm)	NumDeer	NumNomad
1	2.600197	9.50259	23.963411	1	1
2	5.475964	12.192316	41.00115	2	0
3	9.980511	14.710103	45.998738	3	1
4	14.450268	13.439312	37.989193	4	2
5	24.552587	10.061757	27.829244	3	2
6	18.962517	10.28606	13.886701	4	0
7	22.640678	6.874314	8.807206	5	1
8	20.680466	1.729801	0	4	2
9	19.736752	0.474869	0	1	0
10	14.969949	1.998598	0	0	0
11	0.742885	2.797216	9.698954	0	0
12	-3.476718	4.071618	12.039444	1	0
13	2.530683	6.990794	31.374398	2	1
14	9.583077	10.795789	34.482609	3	1
15	10.136155	11.989439	36.679384	3	2
16	16.336651	11.272376	22.965124	4	0
17	20.73235	10.612311	17.889804	5	1
18	26.41819	8.400139	6.459808	6	2
19	27.208956	6.525915	0	3	0
20	25.257564	3.900522	0	2	0
21	12.461478	0.949249	0	1	0
22	14.506745	1.400069	0	0	0
23	3.728612	1.780949	12.712923	0	0
24	-2.687984	3.936482	16.096215	1	0
25	5.279045	10.067671	39.658258	2	1
26	1.929713	9.807843	52.724863	3	1
27	10.712276	11.182407	53.388883	3	2
28	17.268143	12.445386	39.535	4	0
29	15.670344	10.285397	34.882784	5	1

30	27.725737	10.189378	23.452784	6	2
31	27.243885	5.394777	10.752784	5	3
32	18.406838	4.299826	0	4	0
33	16.721861	2.950897	0	3	0
34	8.388801	1.772883	3.987608	2	0
35	9.478298	4.553495	9.754615	1	1
36	0.356068	7.331361	17.167462	2	0
37	-3.04437	7.789158	18.558107	3	1
38	6.780461	12.507999	29.654316	4	2
39	5.346447	12.519485	42.711908	3	2
40	15.106415	11.502852	29.366289	4	0
41	19.543482	12.200373	24.301281	5	1
42	23.759689	9.694002	12.871424	6	2
43	25.316811	3.562323	0.171436	3	3
44	20.064672	3.131822	0	2	0
45	16.362421	0	0	1	0
46	13.292448	1.590382	0	0	0
47	4.454274	2.094435	13.595877	0	0
48	0.051899	3.541573	22.55342	1	0
49	6.5615	9.945719	43.249632	2	1
50	-0.026184	10.974138	48.796406	3	1
51	10.182273	12.425243	50.688355	3	2
52	13.04099	13.733755	38.734041	4	0
53	15.488252	13.177697	34.095392	5	1
54	19.269401	9.269306	22.685808	6	2
55	28.320173	4.945056	9.985808	5	3
56	25.156852	4.111099	0	4	0
57	17.366142	0.351697	0	3	0
58	14.570156	1.656546	0	2	0
59	7.087682	3.358634	10.490627	1	0
60	3.504221	4.62913	27.718294	2	1
61	2.461847	8.309804	44.371187	2	1
62	3.324409	12.131157	60.056601	3	1
63	13.47861	13.939938	55.251802	3	2
64	10.520162	12.96884	48.314599	4	0
65	22.408833	11.911545	43.235302	5	1
66	21.446724	9.132339	31.807458	6	2
67	23.463321	7.215389	19.107649	5	3
68	20.916816	3.758916	0	6	0
69	14.822593	0	0	5	0
70	13.428425	2.152777	0	4	0
71	3.495992	4.697786	13.54433	3	0
72	7.56488	6.228056	25.804922	4	1



Analysis:

For the additional influence on the grain and the deer I implemented a quantity called Nomads. It represents Nomads who move into the area when there is enough grain and deer to supply their needs. The capacity of nomads is half the number of deer and half the height of the grain in inches. If the number of nomads is less than half of both numbers, their size increases by one. If the number of nomads exceeds half the number of deer or half the height of the grain, they travel to a new environment that can handle their increased size, this causes the number of nomads to instantly drop to 0. If either the number of deer or height of the grain is exactly two times the number of nomads and the other is two times or higher than the number of nomads, the number of nomads will stay constant. The nomads diet follows a two-month pattern, one month they eat a diet of strictly grain, consuming 2 inches (defined as a constant) per nomad, the following month they follow an omnivore diet where every nomad consumes 1 inch of grain and 1 deer.

The nomads diet has an interesting effect on the height of the grain and the number of deer. Once the grain is high enough to begin support a population of deer, the number of nomads begins to increase shortly after. The two-month cycle of their diet can be seen in both the grain height and deer number lines, it is seen as a "step" in the line. This occurs while the height and number of deer are increasing and decreasing. However, as the nomad's numbers grow their impact on the other two entities also grows in magnitude. I had to play with the constants a bit to ensure the grain was able to grow enough to support this new height sink. The most noticeable change is the deer population, it no longer follows a linear increase and decrease as exactly when compared to running the simulation without the nomads.

As for the other values, their behaviors are as expected. The temperature and precipitation follow a cyclical pattern that is dependent on the month. They are slightly out of phase because one is using sin and other using cos to scale the amplitude, one also adds that scaled amplitude to the average and the other subtracts it. These two values also have a bit of random noise added in, all of which can be seen in the graphs and data. The grain can be seen growing when the temperature and precipitation are suitable and stops growing when they are not. The decline in its height is due to the nomads and deer continuing to eat the grain after it has stopped growing. As previously mentioned, the nomads affect the deer population, however the number of deer's is also affected by the height of the grain. If there is more than enough grain the population increases by 1, if there isn't enough it decreases by one, if there is just enough it stays the same. All these relationships are expressed in the attached graph and table.

Explanation:

The code works by creating a thread pool size of four, it then uses those four threads to run the four separate sections. Three of these sections are dedicated to the quantities. Those quantities are the grain, deer and nomads. These three sections follow the same pattern, they all loop until the global year factor reaches 2026. In each loop these sections first calculate what their next values will be based off the current values of the environment and other quantities, they wait at a barrier for all sections to finish calculating. After that barrier these three sections write their calculated values into the global "now" variables, then they wait at another barrier. After this second barrier they immediately wait at a third barrier. The fourth section is the watcher, it too loops until the year 2026. It only acts between the barrier after the other threads are done assigning values and the last barrier. It prints the "variables" along with environmental variables and then calculates the new environmental values.

Before the sections are ran, the environmental variables are calculated and the "now" variables are set to starting values. Since I ran this using Visual Studios, I had issues with the rand_r function. The professor mentioned on Piazza that we could use the random number generation method from the Monte Carlo simulation. I used the TimeOfDaySeed function from that assignment to call the srand function. This had to be done before the initial setup of the environment variables and then again once in the watcher thread before the while loop.