Simon Fraser University
CMPT 431 (1097) Assignment 2
Alister Zhao 200124896

Assignment 2 Report

Simon Fraser University
CMPT 431 (1097) Assignment 2
Alister Zhao 200124896

Enhanced Universe Algorithm Approach With Multi-Threads

My algorithm for the Universe with multi-threading is to let the main thread handles the *UpdatePose()* and *controller()* methods, and UpdatePixel() will be multi-threaded. This way, it allows the programs to have any extra procedures for putting barriers and synchronization. Also, it always produces the same result as the original Universe.

In addition, each thread will handle a number of robots, ex. thread 1 will update robots with ID between 0 - 250.

Code Modification

- add a new method called *threadUpdatePixel(* prt)* for multi-threaded calls to *UpdatePixel().*
- re-modified *UpdateAll()* methods to use *threadUpdatePixel(* prt).*

Test Environment

My system specification is:

MacBook Pro 15.4
Mac OS X 10.6.1 Snow Leopard
Intel Core Duo 2.16 GHz
2GB Memory

Observation and Conclusion

- Multi-threaded universe reduces run time by 30~35% when comparing with non-multi-threaded version, and 70% when comparing with the original universe. (see chart on next page)
- Multi-threaded universe produces almost identical result to non-multi-thread universe when 1 core of the CPU is turn off.
- After a numerous runs, I find that when population is small (0~200), both versions of universe produces similar result, so I could not identify which one is truly better than the other. However, when population is bigger (200+), multi-threaded version is truly out performing non-multi-threaded version.
- I also noticed that, when number of threads increases beyond 4, the performance gain of the multi-threading decrease. The reason this happens is because the overhead of switching threads for CPU time is increasing. When I run 1000 threads, the multi-threaded universe runs even slower than original.

Simon Fraser University
CMPT 431 (1097) Assignment 2
Alister Zhao 200124896

Data and Run Time Comparison

Enhance Universe /w Threads (4)	No. of Updates	Population	Mean	Standard Deviation	Run 1	Run 2	Run 3	Run 4	Run 5
	1000	200	1.608	0.0455	1.56	1.58	1.68	1.61	1.61
	1000	400	4.004	0.015	4.01	4.02	4.00	4.01	3.98
	1000	600	8.802	0.212	8.61	8.92	8.93	9.01	8.54
	1000	800	15.052	0.436	14.71	15.21	14.52	15.61	15.21
	1000	1000	23.868	0.254	23.47	23.96	24.12	23.78	24.01
Enhance Universe									
	1000	200	1.76	0.0122	1.78	1.75	1.76	1.75	1.76
	1000	400	6.30	0.280	6.47	6.05	5.94	6.56	6.46
	1000	600	13.0	0.494	13.55	12.64	13.47	12.45	12.86
	1000	800	24.89	0.664	25.68	24.96	23.84	24.91	25.07
	1000	1000	34.38	0.718	35.66	34.04	34.05	34.12	34.02
Original Universe									
	1000	200	3.459	0.0726	3.43	3.41	3.59	3.43	3.44
	1000	400	13.614	0.272	13.26	13.98	13.49	13.75	13.59
	1000	600	30.841	0.654	30.60	31.94	30.29	30.89	30.47
	1000	800	52.085	0.58	51.22	52.19	52.47	51.85	52.70
	1000	1000	80.163	1.52	80.98	79.02	80.94	78.13	81.75

Final Result	Original		Enhanced		Enhanced /w Threads		Reduced Run TIme By % (Enhanced vs Original)	Reduced Run Time By % (Enhanced vs Enhanced /w Threads)	Reduced Run Time By % (Enhanced / w Threads vs Original)
	Population	Run Time	Population	Run Time	Population	Run Time			
	0	0	0	0	0	0			
	200	3.459	200	1.76	200	1.608	49.12%	8.64%	53.51%
	400	13.614	400	6.30	400	4.004	53.75%	36.41%	70.59%
	600	30.841	600	13.0	600	8.802	57.87%	32.26%	71.46%
	800	52.085	800	24.89	800	15.052	52.21%	39.53%	71.10%
	1000	80.163	1000	34.38	1000	23.868	57.11%	30.57%	70.23%