SWT Vs. Swing Performance Comparison

Document Owner:	Igor Križnar
Status:	***Released***
Creation:	2005-10-05 (Klemen Žagar)
Revision:	1.4

Scope

This document presents the results of a SWT-versus-Swing performance benchmark.

Document History

Revision	*Date*	*Author*	*Section*	*Modification*
1.0	2005-10-05	Klemen Žagar	all	Created.
1.1	2005-11-24	Klemen Žagar	all	Added report on drawing primitive performance.
1.2	2005-11-30	Klemen Žagar	all	Added report on X Windows traffic.
1.3	2005-12-02	Klemen Žagar	all	Added report on X Windows traffic comparison of Java 1.4 and 1.5.
	2005-12-02	Igor Križnar	all	**Released**
1.4	2006-03-03	Klemen Žagar	all	Added report on X Windows in native C.
	2006-03-03	Igor Križnar	all	**Released**

Table of Contents

1. Methodology

1.1 General Approach

Every test was executed continuously in a loop until a pre-defined amount of time (10 seconds) had passed. This way, uniform precision of measurement was assured across platforms and test cases, and hard-coding of appropriate loop counts was avoided. However, a mechanism had to be introduced to take the loop's overhead into account.

Each 10 second test was executed 10 times to measure standard deviation as well.

1.1 Timing Tests

Two groups of tests were performed. One group of tests was concerned with performance of drawing graphics primitivies such as lines and polygons. The tests in this group were:

- **SWT/Swing Rect:** the time required to draw a rectangle.
- **SWT/Swing Rect Fill:** the time required to draw a filled rectangle.
- **SWT/Swing Poly:** the time required to draw a polygon.
- **SWT/Swing Poly Fill:** the time required to draw a filled polygon.
- **SWT/Swing Circle:** the time required to draw a circle.
- **SWT/Swing Circle Fill:** the time required to draw a filled circle.
- **SWT/Swing Line:** the time required to draw a line.

The second group contained tests of widget performance. The following tests were performed:

- **SWT/Swing Table (20,40):** Time required to set the value of 20x40 cells in a table.
- **SWT/Swing Table Scroll (20,100):** Time required to scroll a table with 20x100.
- **Swing Text Fields:** Time required to set the value of 400 text fields with Swing. The text fields are arranged in a 20x20 grid with dimensions of 1024x768 pixels. In this test, function `JTextField.setText()` is being invoked.
- **Swing Text Field (Synch):** Same as Swing Text Field, except that `JTextField.setText()` is performed synchronously in Swing's thread using `SwingUtilities.invokeAndWait()`. This ensures that all requests actually result in a redraw.
- **SWT Text Field:** Same as Swing Text Field, but using SWT. SWT's GUI event loop dispatches events immediately after calling all `Text.setText()` methods.

1.2 Network Traffic Tests

In this suite of tests, an SSH tunnel was established between the SSH client and SSH daemon process on a Linux machine (environment A). On the SSH daemon side, the benchmark was run, which resulted in X Windows traffic being transmitted over an SSH tunnel to the SSH client (X Windows server). The number of packets and bytes

of this traffic was measured for Rect, Poly Fill, Table and Text Field tests, as described in the previous chapter.

1.3 Environment

The benchmarks were tested in the following environments:

ID	CPU	RAM	Graphics	Operating System	Java	X Windows Server
A: reference environment	Intel Centrino 1.6GHz	1GB	ATI Mobility Radeon 9700	Linux Fedora Core 2 Kernel 2.6.10 libc 2.3.3 gtk2 2.4.14	Sun Java 1.5.0_01	Local
B: Java 1.4	Same as A				Sun Java 1.4.2_06-b03	Local
C: Windows	Same as A			Windows XP	Sun Java 1.4.2_08	N/A
D: Windows, II	Intel Pentium 2.4GHz	1GB	ATI Radeon 9200	Windows XP SP 1	Sun Java 1.4.2_04	Local
E: VMware	Same as A			Windows XP SP2 in VMware 4.5.2	Sun Java 1.5.0_03	N/A

2. Results

The following table lists the time taken by an iteration of a particular test.

2.1 Quantitive Observations

Test	A: Java 1.5, Linux	B: Java 1.4, Linux	C: Java 1.4, Windows	D: Java 1.4, Windows, II	E: VMware
Drawing primitive tests (times in microseconds)					
SWT Rect	9.9±0.02	11.0±0.07	11.3±0.07	9.0±0.18	89.4±1.27
Swing Rect	8.4±0.1	8.4±0.4	11.9±0.4	8.9±0.6	67.8±0.4
SWT Rect Fill	57.8±0.5	60.6±0.8	35.3±14.5	103.0±3.6	71.5±14.2
Swing Rect Fill	49.6±0.1	55.2±5.3	43.4±0.8	103.0±10.0	36.2±8.8
SWT Poly	13.8±0.03	15.1±0.03	251.0±66.97	20.0±0.1	133.8±54
Swing Poly	13.4±0.03	13.5±0.03	6.6±0.34	19.9±0.1	141.1±37
SWT Poly Fill	102.6±1.0	104.8±0.3	98.3±0.2	82.5±3.3	1071.2±350
Swing Poly Fill	95.7±0.2	94.6±1.1	45.8±3.1	82.1±7.9	887.0±445
SWT Circle	53.2±0.1	54.7±0.7	160.7±0.1	49.0±3.6	148.5±1.8
Swing Circle	51.3±0.3	51.6±0.4	56.1±4.8	45.8±5.0	130.6±0.5
SWT Circle Fill	58.7±0.1	61.4±0.1	158.0±0.3	101.0±2.3	953.1±6.6
Swing Circle Fill	49.9±0.4	50.6±0.3	128.4±0.3	100.0±0.8	931.7±7.6
SWT Line	5.9±0.03	6.5±0.02	34.9±0.13	5.5±0.3	657.9±320
Swing Line	5.1±0.05	5.7±0.01	10.2±0.03	6.8±0.03	808.7±7.3
Widget tests (times in milliseconds)					
SWT Table (20,40)	578±5.6	584±60.9	98±0.4	30±1.0	576±4.9
Swing Table (20,40)	30±8	141±43	9.4±0.9	6±1.0	29±4.2
Swing Table - synch	127.1±2.0	100.8±0.6	101.3±0.5	56.0±3.0	88.9±8.0

Test	*A: Java 1.5, Linux*	*B: Java 1.4, Linux*	*C: Java 1.4, Windows*	*D: Java 1.4, Windows, II*	*E: VMware*
(20,40)					
SWT Table Scroll (20,100)	12.2±0.2	12.1±0.05	10.0±0.04	2.0±0.0	15.5±0.6
Swing Table Scroll (20,100)	4.3±0.1	33.1±0.1	31.6±2.4	75.0±6.0	9.2±0.3
SWT Text Fields (20,20)	276±0.6	319±44.7	68±0.1	33±3.0	341±1.8
Swing Text Fields (20,20)	84.2±4.7	78.0±53.0	37.8±1.6	35.0±21.0	131.0±18.0
Swing Text Fields - synch(20,20)	133±0.5	166±3.6	120±0.6	66±1.0	302±6.7

Table 1: *Performance of tests in various environments. Note that the measurement refers to the time required to perform one iteration of the test.*

Network bandwidth utilization of the X Windows protocol tunneled over SSH is shown in the table below. The tests were performed in environment A (Java 1.5). SSH daemon and SSH client were residing on the same computer and were communicating via the loopback network interface.

Test	*To daemon (packets)*	*To daemon (bytes)*	*From daemon (packets)*	*From daemon (bytes)*
SWT Rect	2.97±0.25*	0.18±0.02	4.93±0.42*	40.34±3.42
Swing Rect	2.07±0.10*	0.13±0.01	4.16±0.18*	40.80±2.02
SWT Poly Fill	5.01±0.09*	0.30±0.00	8.13±0.06*	56.65±0.00
Swing Poly Fill	3.65±0.10*	0.22±0.01	5.82±0.13*	54.14±1.36
SWT Table (20,40)	9.71±0.08	577.45±5.50	14.53±0.05	110,135.16±3.59
Swing Table (20,40)	1.21±0.09	73.23±4.80	1.78±0.08	14,929.09±519.86
Swing Table - synch (20,40)	10.04±0.28	705.29±23.70	16.76±0.24	133,368.47±1,522.04
SWT Text Fields (20,20)	9.73±0.04	1,066.94±2.01	10.05±0.04	95,445.19±5.37
Swing Text Fields (20,20)	3.56±0.29	195.80±16.13	3.69±0.30	15,247.65±1,146.43
Swing Text Fields - synch(20,20)	7.90±0.10	546.09±5.51	10.94±0.14	82,578.85±292.69

Table 2: *Network traffic generated by X Windows while drawing over an SSH tunnel.*

* Packet counts for Rect and Poly Fill cases are per 1000 drawn items.

The same tests were performed in environment B (Java 1.4):

Test	*To daemon (packets)*	*To daemon (bytes)*	*From daemon (packets)*	*From daemon (bytes)*
SWT Rect	2.97±0.25	0.18±0.02	4.93±0.42	40.34±3.42
Swing Rect	1.92±0.13	0.12±0.01	3.89±0.25	38.41±2.40
SWT Poly Fill	4.60±0.07	0.27±0.00	7.86±0.05	56.63±0.00
Swing Poly Fill	3.35±0.08	0.20±0.00	5.39±0.07	50.26±0.38
SWT Table (20,40)	9.65±0.85	573.43±50.46	14.50±1.27	110,133.27±9,716.19

Test	To daemon (packets)	To daemon (bytes)	From daemon (packets)	From daemon (bytes)
Swing Table (20,40)	1.87±0.36	113.61±22.51	3.07±0.64	24,058.36±5,282.78
Swing Table - synch (20,40)	10.36±0.13	706.26±7.42	17.28±0.25	132,109.61±1,298.19
SWT Text Fields (20,20)	9.73±0.05	1,067.35±2.68	10.03±0.04	95,445.78±4.46
Swing Text Fields (20,20)	9.51±1.14	1,239.54±150.89	9.44±1.14	1,091.87±132.51
Swing Text Fields - synch(20,20)	404.33±29.59	50,333.12±3,782.46	393.19±29.20	135,379.37±9,741.58

Ratio of resource consumption of Java 1.4 and Java 1.5 is as follows:

Test	To daemon (packets)	To daemon (bytes)	From daemon (packets)	From daemon (bytes)
SWT Rect	1	1	1	1
Swing Rect	0.93	0.93	0.93	0.94
SWT Poly Fill	0.92	0.93	0.97	1
Swing Poly Fill	0.92	0.92	0.93	0.93
SWT Table (20,40)	0.99	0.99	1	1
Swing Table (20,40)	1.55	1.55	1.73	1.61
Swing Table - synch (20,40)	1.03	1	1.03	0.99
SWT Text Fields (20,20)	1	1	1	1
Swing Text Fields (20,20)	2.67	6.33	2.56	0.07
Swing Text Fields - synch(20,20)	51.21	92.17	35.93	1.64

Table 3: *Network resource consumption (packets/bytes) ratio of Java 1.4 and Java 1.5. Greater number means that Java 1.5 is more efficient by the given factor.*

2.2 SWT Vs. C++

On Linux, a test has also been performed in C++ using the X library directly. Performance for drawing of rectangles was compared. For the particular test case, Java SWT test took 44.5 microseconds per rectangle, whereas C++ took 41.8 microseconds (6% faster).

2.3 Qualitative Observations

Apart from the quantitative measurements presented above, the following observations were made:

1. Swing Text Field redrawing is not deterministic. For example, the order in which the fields were initially populated and subsequently updated varied significantly. Thus, sometimes the fields in the middle of the 20x40 grid were populated before those at the top-left corner of the grid, though their `setText` methods were invoked later.
2. Swing Text Field may skip updates. If several `setText` invocations are made in a short succession, only the latest one may take effect.

3. Interpretation

By observing the figures in the above tables, the following inferences may be made:

- Swing avoids performing unnecessary redrawing. This is noticeable by comparing results of tests Swing Text Field and Swing Text Field (sync), where in the latter case Swing forced a redraw upon every `setText` invocation.
- On Linux, Swing widgets outperform SWT's by a significant factor (e.g., compare Swing Table and SWT Table tests).
- Standard deviation of Swing's performance is higher than standard deviation of SWT's. Standard deviation decreases significantly if redrawing is performed in the Swing's thread. This is likely due to non-determinism involved in communication between Swing's and application's thread.
- In general, Java 1.5 is faster than Java 1.4.
- For rendering of text fields, Java 1.5 offers a significant improvement of performance when using X Windows over network.
- In general, Swing generates less network traffic with X Windows over SSH than SWT.
- Code written in C/C++ using the X library performs somewhat faster than code written in Java using SWT, with the speed-up being from 5% to 10%.

4. Conclusion

It is hard to give a rule-of-thumb where SWT would outperform Swing, or vice versa. In some environments (e.g., Windows), SWT is a winner. In others (Linux, VMware hosting Windows), Swing and its redraw optimization outperform SWT significantly. Differences in performance are significant: factors of 2 and more are common, in either direction.

It is questionable how often Swing's capability of suppressing successive redraws is effective. It is very likely that in practice this is not a frequent occurence, as repeated update requests in a short period of time may be uncommon. In this case, results of the Swing Text Field (sync) test should be considered with greater credibility than those of the Swing Text Field test, as they approximate the resource usage more closely.

Initial expectation before performing this benchmark was to find SWT outperform Swing. This expectation stemmed from greater responsiveness of SWT-based Java applications (e.g., Eclipse IDE) compared to Swing-based applications. However, this expectation could not be quantitatively confirmed. It is possible that perceived responsiveness is a consequence of smaller time required to respond to user interaction, which involves not only drawing, but also detecting and responding to the user's action. Also, determinism (smaller standard deviation) could result in greater perceived responsiveness.