Tennix

Optimizing an open source
game for mobile devices

January 27th, 2009
Effiziente Programme WS08/09
Vienna University of Technology

Stefan Dösinger, Esad Hajdarevic, Thomas Perl

Properties

- Open source game written in C using SDL
  - Different operating systems and processors
- Graphics drawing
  - Not hardware-accelerated (software rendering)
  - (unfortunately) depending on hardware speed
- Using an external library
- Non-deterministic input (obviously – for a game)
  - Making it measurable: Benchmark-Mode (AI vs AI)
- Speed partially subjective (smooth animations)

Before Our Optimizations

- Game runs fluently on commodity hardware
- Mobile devices (Nokia Internet Tablets, N8x0)
  - Less CPU power and RAM
  - Energy consumption (running on batteries!)
  - Multitasking (allow background tasks, e.g. downloads)
- Game does not run fluently on mobile devices
- No profiling has been done yet

Motivation And Goal

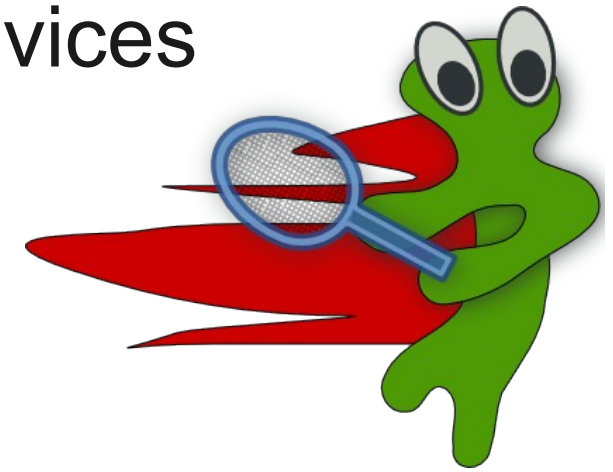
Playable, fluent game on mobile devices

Minimize energy consumption on PCs

Good multitasking citizen (sane FPS-Limit)

Carry out profiling; detect and fix bottlenecks

Goal: Tennix runs fine on N8x0 devices



Adding Benchmark Mode

- “disarm” the Random Number Generator
  - Initialization with known, constant value
- No user interaction (AI versus AI mode)
- Jump directly into the game loop (no menu)
- Automatic exit after fixed game length (timelimit)

New command line switch:

```
./tennix -b
```

Initial Profiling (oprofile)

CPU: Core 2, speed 1000 MHz (estimated)

Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of 0x00 (Unhalted core cycles) count 30000

samples	%	image name	app name	symbol name
1082617	73.4244	vmlinux	vmlinux	idle
330409	**22.4087**	**libSDL-1.2.so.0.11.0**	**libSDL-1.2.so.0.11.0**	**/usr/lib32/libS..**
9800	0.6646	oprofiled	oprofiled	/usr/bin/oprofiled
3515	0.2384	libfb.so	libfb.so	/usr/lib64/xorg/mo...
3371	0.2286	tennix	tennix	font_get_metrics

Font Width Caching

- Dynamic character width
- “Terminating pixels”
- Measure once and cache results

- Reason: Measuring has to lock the SDL surface, which is very costly

Font Width Caching (Results)

```
 sleeping,   0 stopped,   0 zombie
 77.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
sed,   960848k free,    57584k buffers
sed,  3984048k free,   714832k cached

  SHR S %CPU %MEM    TIME+  COMMAND
 2888 S   42  0.8   0:17.18 tennix
  12m S    2  2.8   2:11.97 X
 4028 S    1  1.2   0:34.52 skype
  23m S    1  1.8   0:38.93 pidgin
  944 R    0  0.1   0:00.14 top
  488 S    0  0.0   0:00.31 init
    0 S    0  0.0   0:00.00 kthreadd
```

```
 92.3%id,  0.2%wa,  0.0%hi,  0.2%si,  0.0%st
sed,   957352k free,    57448k buffers
sed,  3984048k free,   714764k cached

  SHR S %CPU %MEM    TIME+  COMMAND
 2884 S    6  0.8   0:01.30 tennix
  12m S    3  3.0   2:09.39 X
 4028 S    2  1.2   0:33.83 skype
  23m S    1  1.8   0:38.38 pidgin
  11m S    1  0.8   0:09.97 kicker
  944 R    0  0.1   0:00.33 top
  15m S    0  0.9   0:00.39 ksnapshot
  488 S    0  0.0   0:00.31 init
```

Major improvements

More a bug than just slow: Expensive SDL calls should not be put in the main loop if avoidable

Font Width Caching (oprofile)

```
CPU: Core 2, speed 1000 MHz (estimated)

Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of
  0x00 (Unhalted core cycles) count 30000

Samples    %        image name               app name                 symbol name
1919481  94.2060  vmlinux                  vmlinux                  idle
45567    2.2364   libSDL-1.2.so.0.11.0     libSDL-1.2.so.0.11.0/usr/lib32/libS...
15018    0.7371   oprofiled                oprofiled                /usr/bin/oprofiled
4742     0.2327   libfb.so                 libfb.so
  /usr/lib64/xorg..
3646     0.1789   libqt-mt.so.3.3.8        libqt-mt.so.3.3.8        /usr/qt/3/lib...
3605     0.1769   Xorg                     Xorg                     /usr/bin/Xorg
3290     0.1615   oprofile                 oprofile                 /oprofile
```

Avoiding Redraws

- Some parts of the game screen are unchanged:
  - Referee, score display
  - Racket only changes when the user moves it
- Optimization by avoiding redraws and re-using as many already-drawn graphics as possible

Avoiding Redraws (Results)

```
CPU: Core 2, speed 1000 MHz (estimated)

Counted CPU_CLK_UNHALTED events (Clock cycles when not halted) with a unit mask of
  0x00 (Unhalted core cycles) count 30000

samples  %         image name               app name                 symbol name
2150031  94.9117   vmlinux                  vmlinux                  idle
42187      1.8623   libSDL-1.2.so.0.11.0     libSDL-1.2.so.0.11.0
  /usr/lib32/libSDL..
16723      0.7382   oprofiled                oprofiled
  /usr/bin/oprofiled
3602       0.1590   libqt-mt.so.3.3.8        libqt-mt.so.3.3.8        /usr/qt/3/lib64
3592       0.1586   libfb.so                 libfb.so
  /usr/lib64/xorg..
3580       0.1580   oprofile                 oprofile                 /oprofile
3435       0.1516   Xorg                     Xorg                     /usr/bin/Xorg
```

Compiler Options

Known-good CFLAGS for TI OMAP 2420:

```
-mfpu=vfp -mfloat-abi=softfp
-mcpu=arm1136j-s
```

Future Improvements

- Micro-optimizations (as with `mastermind.c`)
  - but: Most of the CPU time in SDL → Library calls!
- Threading
  - Increases code complexity without much improvement; most useful on multi-core CPUs, which handhelds are not (yet?)
- pixel doubling (XOMAP X Server feature)
  - Would change the appearance of the game; only possible on suitable hardware (N8x0, etc...)

Energy Consumption

- Rough measurements with `powertop`, but reproduceable
- Less CPU time used, CPU is longer in C3
- About 2 W savings, according to ACPI
- Can we measure this on N8x0 devices too?

```
C0 (cpu running)          (22.7%)
polling              0.0ms ( 0.0%)
C1 halt              0.0ms ( 0.0%)
C2                   0.0ms ( 0.0%)
C3                   3.8ms (77.3%)

Wakeups-from-idle per second : 259.
Power usage (ACPI estimate): 20.0W

Top causes for wakeups:
  18.1% ( 57.0)                tennix :
  17.6% ( 55.3)      <interrupt> :
```

```
Cn                   Avg residency
C0 (cpu running)          ( 7.9%)
polling              0.0ms ( 0.0%)
C1 halt              0.0ms ( 0.0%)
C2                   0.2ms ( 0.0%)
C3                   3.8ms (92.8%)

Wakeups-from-idle per second : 309.
Power usage (ACPI estimate): 17.9W

Top causes for wakeups:
  21.4% ( 93.5)                tennix :
```

Conclusions

Most effective: fixing the “font width bug”

Avoiding redraws brings performance gains, but makes code more complex

External libraries make profiling harder

Biggest potential for optimization here: drawing on the surfaces (takes most of the time)

Small code changes could make a great impact on game performance on N8x0 devices

Used / Suggested Tools

codeviz (creating codegraphs)

ncc (Source Code Analyzer, “replaces” gcc)

oprofile (Linux Kernel Profiler)

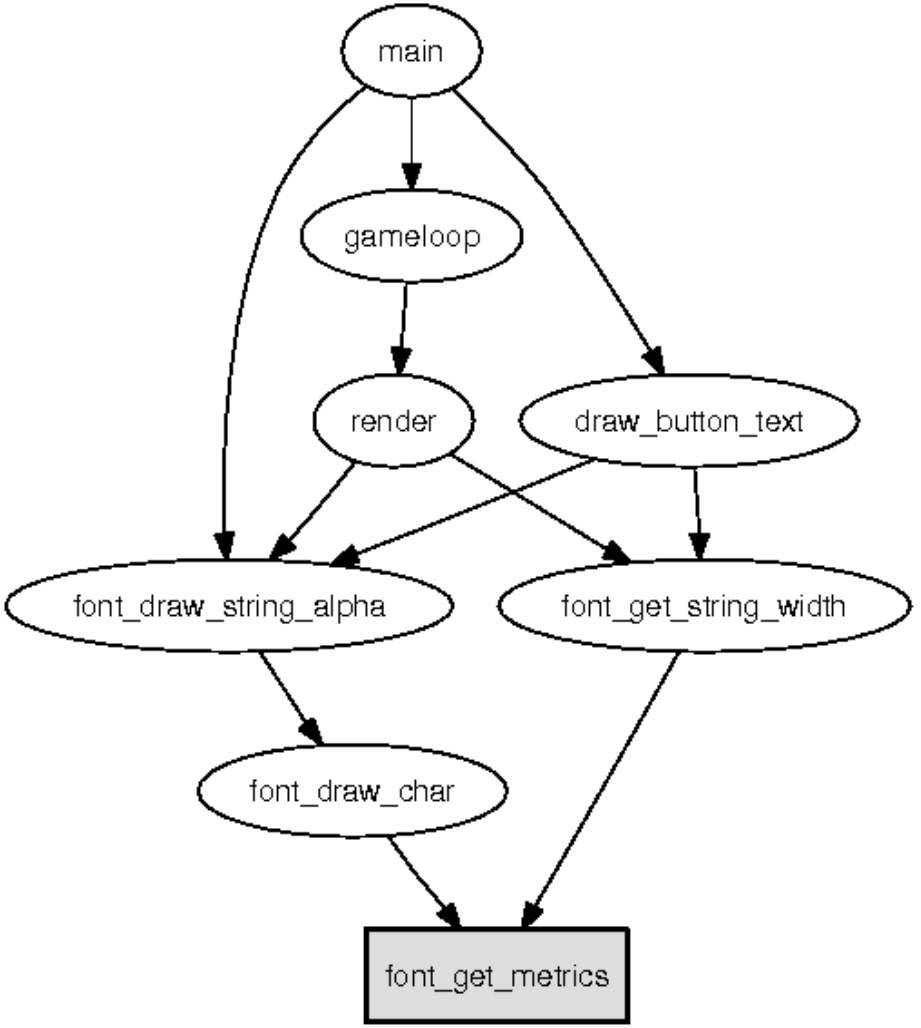
powertop, top, time

gprof

qprof

git

Example Callgraph



Presentation available at
http://icculus.org/tennix/files/effprog_200901.pdf

Tennix website and download
http://icculus.org/tennix/