Update on EFL performance

Cedric Bail
Senior Open Source Engineer
Samsung Open Source Group
cedric@osg.samsung.com
The plan

- Benchmark memory usage, frame rate and time to first frame
- One week should be enough, right?
- Use usual tool and experiment with a few new one
  - Expedite
  - Massif
  - Smem
- Walk in a park!
Not quite…

Find memory leak

Find massive memory usage increase

Find massive performance regression

Barely enough time to fix them!
Easiest one, the one I create!

- Massive leak of Cairo context in expedite
- 132MB of leaked data at the end of the test
- Kind of make any other peak impossible to see!
- That one is now fixed
Easiest one, the one I create!
Weird increase in PSS

- Write a script to measure PSS
- Gathered the data
- Look at it in the plane
- 8-O
- Need to review the script first …
- That's not a fixed problem!
Weird increase in PSS
Run expedite

- Instead of the usual raw benchmark, try to find the reliable test
- Run all expedite test twice on every version since 1.8
- That's 18 runs! More than 2000 tests...
- Start process them in the plane...
- 8-O
- See major slow down!
Run expedite
Run expedite

• Let's blame Tom, because he is not here! :-) Is he?
• Then run expedite under callgrind with 1.8 and git
• Issue seems to be related to raster change in region_add
  – 1.8: 40% of the benchmark, while drawing was 20%
  – 1.16: 80% of the benchmark, while drawing is 5%
• Going to be fun to fix...
So what about the rest?

- No time for time to first frame benchmarking!
- No time for power consumption benchmarking!
- Memory testing still need fixing
What did I learn

- Automating expedite benchmark is doable if we focus only on a subset
- Expedite does catch regression if we use it…
- How to integrate it with Jenkins and our infrastructure?
- Maybe put more logic in it to benchmark memory by processing /proc/$PID/maps
- Still no test of elementary/edje at all

- If we don't see the issue when we have tools, then what about when we don't!
Opinions ? Questions ?
Thank you.