**Introduction**

One popular tiling window managers for X Window System (X11) is i3. It comes with a simple screen locker i3lock. i3lock display a solid color or a user specified image. Someone had made a simple wrapper script, which
- takes a screenshot using scrot,
- blurs it using convert from imagemagick,
- uses blurred image with i3lock.

The goal of this project is to improve i3lock so that the background blurring would be done in real time.

**Wrong estimate**

At first this seemed like a one weekend project. Just fork i3lock and add some open source code for Gaussian blurring. But it turned out that X11 is complicated and the goal was achieved many weekends later.

Some of the difficulties encountered
- Getting the desktop contents behind lock window.
- When to redraw screen.
- Gaussian blurring on CPU is not fast enough.

Getting the desktop contents behind lock screen was solved by using an overlay window for the lock window by using X11 Composite extension.

X11 extension XDamage was used to know when one of the background windows updates something, so that i3lock-blur, would know when to redraw the screen.

**Gaussian blurring on GPU**

After a little research it was certain, that Gaussian blur is a simple operation that can be implemented by using OpenGL shaders and this avoids turning to more low-level APIs as CUDA or OpenCL.

### Blurring in two passes

Usually Gaussian blurring with a 9x9 kernel requires 81 texel reads. However Gaussian blur can also be done in two passes, first in one direction, and then in other direction.

Getting the desktop contents behind lock screen was solved by using an overlay window for the lock window by using X11 Composite extension.

X11 extension XDamage was used to know when one of the background windows updates something, so that i3lock-blur, would know when to redraw the screen.

### Optimizing the number of texel reads

When the kernel diameter is 9, then 9 texel reads are still needed for calculating the output texel for one pass. It turns out that this number can be reduced, without losing quality thanks to the interpolation built into GPUs.

**Conclusion**

- Blurring is fast enough to watch a movie through it.
- Actual speed depends on the blur radius. Larger radius means more texel lookups.
- Some people, besides the author, actually use it.

Source code at: [http://github.com/karulont/i3lock-blur](http://github.com/karulont/i3lock-blur)

Mail: karulont@ut.ee