evas filters & masks
Masking

- Just like clipping but with an image
- Alpha layer used as mask
  - R, G, B channels unused
  - Object color is used like with rectangles (multiply)
Masking: HOWTO

- Same as clip:
  - C: `evas_object_clip_set`
  - EDC: `clip_to`

- Valid clippers: Rectangles & Image objects
  - C: `evas_object_image_add`
  - EDC: `RECTANGLE`, `IMAGE`, `PROXY`
Examples
Masks of masks?
Masks of masks?
Masks of masks?
Masks of masks!

- Multiply alpha channels only
Example
Gotchas

- Performance considerations
  - Internal steps to use masking
    - Create mask surface
    - Render (scale) image to surface
    - SW only: extract alpha channel
    - Multiply mask alpha by object pixels
  - GL has one optimization
    - In the proper conditions, performance is great
Performance

Problem:
- Masking performance can be slow
  - Rendering image to draw surface, copy to alpha surface (SW), read from surface

Solution:
- GL is very good at scaling
  - Use original image, scale on-the-fly, read only alpha
Performance

- Conditions
  - GL engine (ELM_ACCEL=gl)
  - Simple image object
  - Filled or fill = image size
  - No border
  - Only one mask (no mask of mask)
- → Scaling on the fly
And Tizen?

- From Tizen 2.3.1
  - Gear S2, used for animations mainly
Filters

Source: http://www.uniflux-filters.com/filtres
Motivations

- Shiny & fancy text effects
  - Replace old SHADOW, GLOW, etc...
  - Animated

- Similar to:
  - HTML/CSS (font-shadow)
  - SVG filters
  - Photoshop
CSS font-shadow

Cool effects, but only series of blurs

Text effects

- Shadow, glow, contour, emboss, etc...
- Various colors
- Various sizes (eg. blur radius)
Design concepts

- Photoshop in real-time (Okay, maybe not all of Photoshop...)
- Multiple layers
- Combined effects
- Scripted
Buffer & operations

input
Buffer & operations

\[ A = \text{buffer}('alpha') \]
\[ \text{grow} \{20, \text{dst} = A\} \]
Buffer & operations

A = buffer({'alpha'})
grow {20, dst = A}
blur {15, src = A, color = "darkblue"}
Buffer & operations

A = buffer['alpha']
grow {20, dst = A}
blur {15, src = A, color = "darkblue"}
blur {15, color = "cyan"}
A = buffer('alpha')
grow {20, dst = A}
blur {15, src = A, color = "darkblue"}
blur {15, color = "cyan"}
blend {color = 'yellow'}
Base functions

- Blend
- Blur
- Color curves
- Grow
- Mask \((A \cdot B)\)
- Bump map
- Displacement map
Base functions

- Blend
- Blur
- Color curves
- Grow
- Mask \((A \cdot B)\)
- Bump map
- Displacement map

Raster + Displacement map
Current status

- **TEXT and IMAGE support**
  - C APIs with EO only: Efl.Gfx.Filter (BETA)
    - filter_program_set()
    - filter_source_set()
  - **Stable** EDC API

- **No TEXTBLOCK support at this point**
  - This can actually be *quite hard* to implement
Performance

- Few ms on desktop for a TEXT object
  - But final output is cached, rarely redrawn
  - Still work to do to optimize redraws
Future plans

- OpenGL acceleration
  - Implement basic filters with shaders
  - Implement buffer system based on FBOs
- Fix API quirks
  - Some filters work only with certain buffer types (alpha vs. rgba)
  - Some functions
  - Add more state information (font geometry)
- Implement GUI editor
  - → @zmike actually started working on one
Future plans

- Implement more base filters
  - For IMAGES
    - Convolutions, color conversions, etc...
  - SVG filters
    - *Vector graphics, anyone?*
- Partial rendering / better caching
Q&A