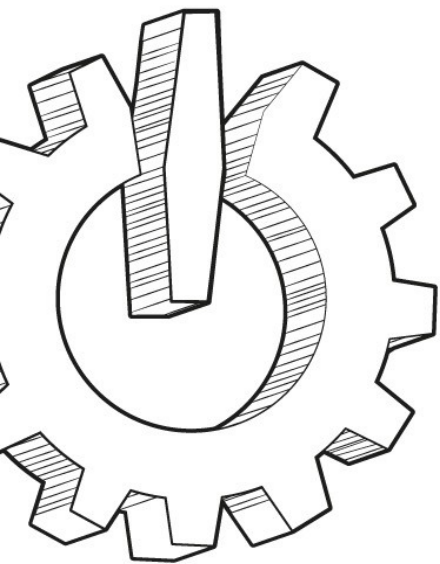




COLLABORA



Writing X11/Wayland agnostic GL applications with Waffle

Emil Velikov

emil.velikov@collabora.com

02/02/2019



FOSDEM ¹⁹

Contents

- What is Waffle
- Why
- Waffle API
 - Vs EGL, GLX, WGL, Getting Native Objects
- Example
- Build system
- Gitlab
- Work ahead
- Questions



What is Waffle

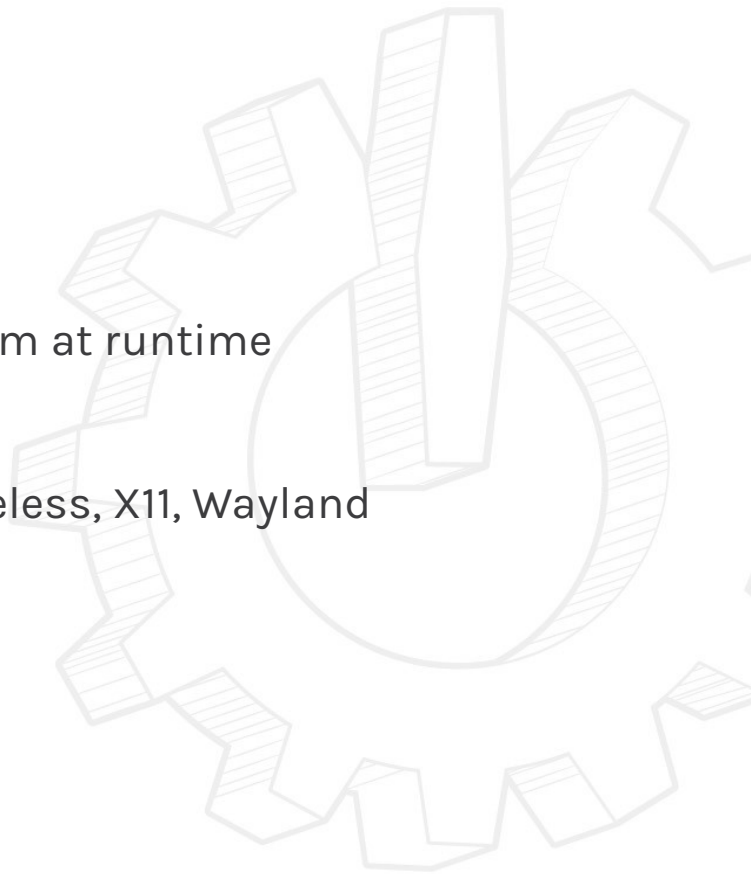


COLLABORA

Open First

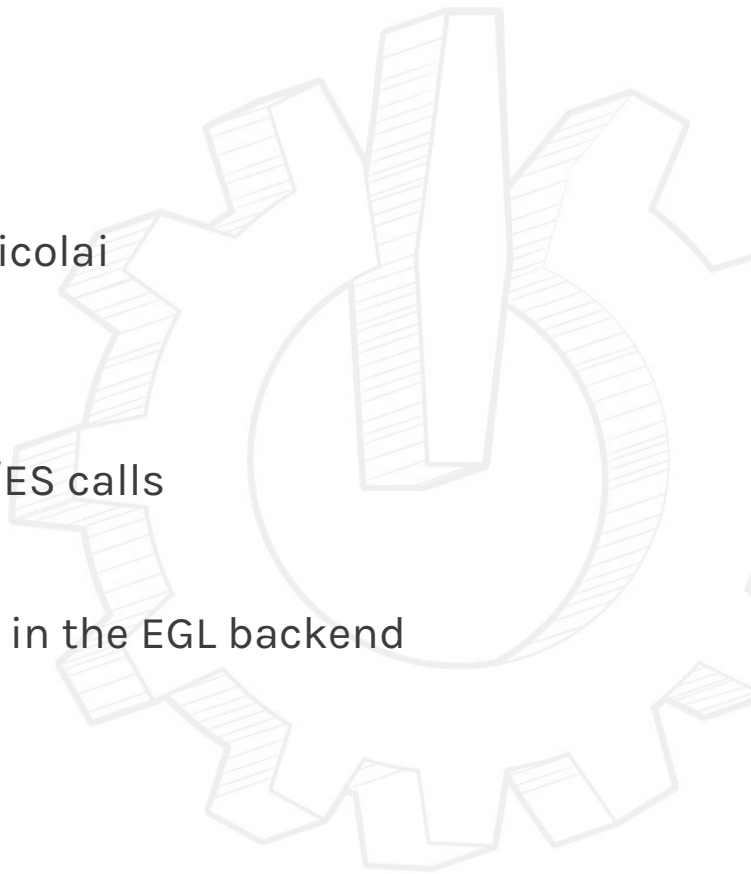
What is Waffle

- Project started in 2012, by Chad Versace
- A library for selecting an OpenGL/ES API and window system at runtime
- Development inspired by Piglit
- Supports CGL, WGL, GLX, NaCl, EGL + Android, GBM, Surfaceless, X11, Wayland
- Simple API and extremely light weight
- Personally involved since 2014, author of the WGL backend



Who uses Waffle

- Piglit
 - Open-source test suite for OpenGL drivers started by Nicolai
 - Later expanded to cover OpenGL ES and OpenCL
- APITrace
 - Open-source tool to trace, replay and inspect OpenGL/ES calls
- Dante: Open-Source Doom 3
 - The Waffle backed took about 2/3 the amount of code in the EGL backend



Why

- Platform and windowing systems vary a lot
- Makes porting GL applications simpler and easier
- Easier distribution and simpler dependencies
- Aids driver development and validation
- Vulkan 1.0 announced in December 2015, first major update in 2018
 - lower level API, low overhead and explicit developer control
- Lack of driver availability:
 - hardware is missing required functionality
 - unsupported platform



Waffle API

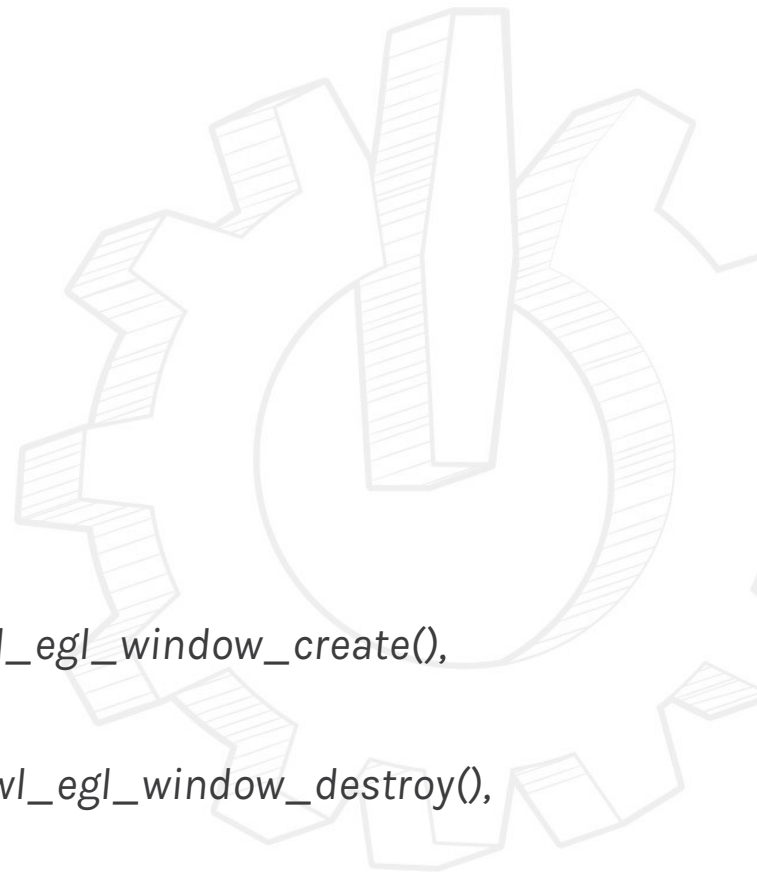
```
struct waffle_display* waffle_display_connect(const char* name);
bool waffle_display_disconnect(struct waffle_display *self);

struct waffle_context* waffle_context_create(struct waffle_config *config,
                                             struct waffle_context *shared_ctx);
bool waffle_context_destroy(struct waffle_context *self);

struct waffle_window*waffle_window_create2( struct waffle_config *config,
                                             const intptr_t attrib_list[]);
bool waffle_window_destroy(struct waffle_window *self);
```

Waffle API vs EGL

- `waffle_display_connect()` → `eglGetDisplay()`
- `waffle_display_disconnect()` → `eglTerminate()`
- `waffle_context_create()` → `eglCreateContext()`
- `waffle_context_destroy()` → `eglDestroyContext()`
- `waffle_window_create2()` → `xcb_create_window_checked()`, `wl_egl_window_create()`,
`gbm_surface_create()`, ...
- `waffle_window_destroy()` → `xcb_destroy_window_checked()`, `wl_egl_window_destroy()`,
`gbm_surface_teardown()`, ...



Waffle API vs GLX

- `waffle_display_connect()` → `XOpenDisplay()`
- `waffle_display_disconnect()` → `XCloseDisplay()`

- `waffle_context_create()` → `glXCreateNewContext()`
- `waffle_context_destroy()` → `glXDestroyContext()`

- `waffle_window_create2()` → `xcb_create_window_checked()`
- `waffle_window_destroy()` → `xcb_destroy_window_checked()`



Waffle API vs WGL

- `waffle_display_connect()` → `CreateWindow()`, `GetDC()`
- `waffle_display_disconnect()` → `ReleaseDC()`, `DestroyWindow()`

- `waffle_context_create()` → `wglCreateContext()`
- `waffle_context_destroy()` → `wglDestroyContext()`

- `waffle_window_create2()` → `CreateWindow()`
- `waffle_window_destroy()` → `DestroyWindow()`



Getting native objects

```
struct waffle_window *window = waffle_window_create(...);
```

```
struct waffle_glx_window *n_window = waffle_window_get_native(window)-> glx;
```

```
struct waffle_x11_egl_window *n_window = waffle_window_get_native(window)->x11_egl;
```

```
Display *xlib_dpy = n_window->display.xlib_display;
```

```
EGLDisplay *egl_dpy = n_window->display.egl_display;
```

```
Window xlib_window = n_window->xlib_window;
```

```
EGLSurface egl_surface = n_window->egl_surface;
```



Example

```
const int32_t init_attrib_list[] = {  
    WAFFLE_PLATFORM, WAFFLE_PLATFORM_GLX,  
    WAFFLE_NONE,  
};  
waffle_init(init_attrib_list);  
struct waffle_display *dpy = waffle_display_connect(NULL);
```

```
const int32_t config_attrib_list[] = {  
    WAFFLE_CONTEXT_API, WAFFLE_CONTEXT_OPENGL,  
    WAFFLE_CONTEXT_PROFILE, WAFFLE_CONTEXT_CORE_PROFILE,  
    WAFFLE_CONTEXT_MAJOR_VERSION, 3,  
    WAFFLE_CONTEXT_MINOR_VERSION, 3,  
    WAFFLE_NONE,  
};
```

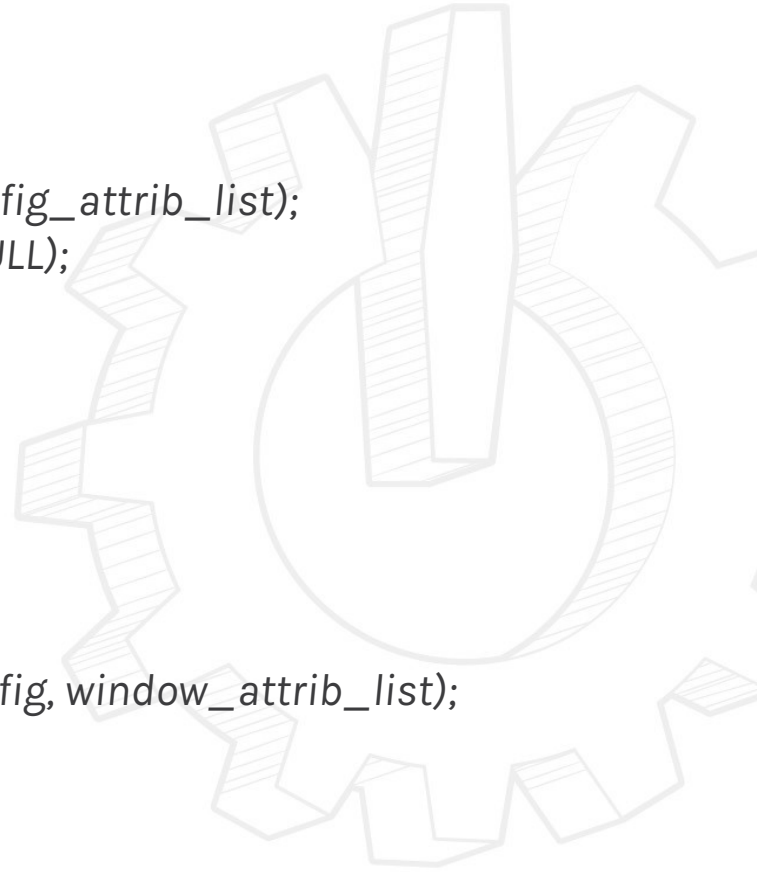


Example (2)

```
struct waffle_config *config = waffle_config_choose(dpy, config_attrib_list);  
struct waffle_context *ctx = waffle_context_create(config, NULL);
```

```
const intptr_t window_attrib_list[] = {  
    WAFFLE_WINDOW_WIDTH, 800,  
    WAFFLE_WINDOW_HEIGHT, 600,  
    WAFFLE_NONE,  
};
```

```
struct waffle_window *window = waffle_window_create2(config, window_attrib_list);  
waffle_make_current(dpy, window, ctx);  
draw(window);
```



Recent changes



COLLABORA

Open First

Build system

- Meson support introduced
- Simpler more intuitive than existing CMake
- Better overall integration and handling of tooling



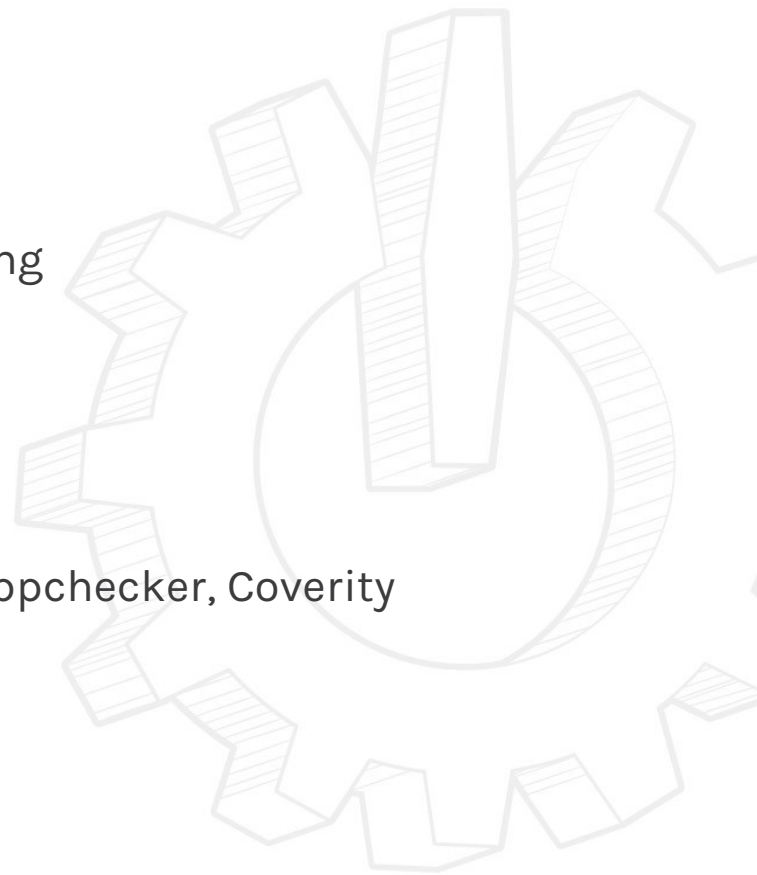
Gitlab

- Not quite official, although everyone is in favor
- Trendy, fully open source
- CI is a step away – simpler build and deployment
- Better integration with tooling like static analyzers
- Updating and deploying the website becomes trivial
- Coverity Scan integration becomes more complex
- Proprietary platforms – Windows, MacOS



Work ahead

- Finalize/formalize the transition to gitlab
- Improve workflow documentation - MR, reviewing, releasing
- Convert existing ML patch series to gitlab MR
- Build Android support against the SDK
- Polish and merge the CI
- Add analyzers to the CI pipelines - gcov, clang-analyser, cppchecker, Coverity
- Your feature



Questions?

Psst...
We're hiring!



COLLABORA

