Waylandifying Chromium - From Downstream To Shipping

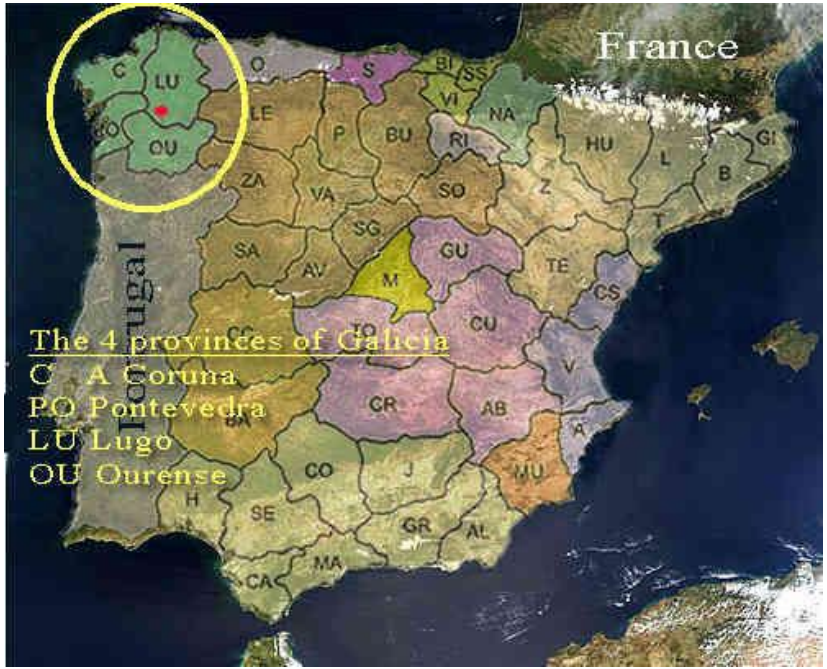
Maksim Sisov - msisov@igalia.com

Agenda

- About Igalia.
- History of the Project.
- What is Ozone?
- How Ozone Integrates Inside Chromium.
- Ozone/Wayland Limitations:
  - Limitation 1 - Graphics Pipeline.
  - Limitation 2 - Tab Dragging Feature.
- Performance of Chromium on Wayland:
  - Performance Comparison - non-Ozone/X11 vs Ozone/Wayland.
- Shipping of Ozone/Wayland.
- Bonus: Migration of X11 to Ozone.

About Igalia

- Worker-owned; employee-run Open Source consultancy company; based in Galicia, Spain.



igalia

About Igalia

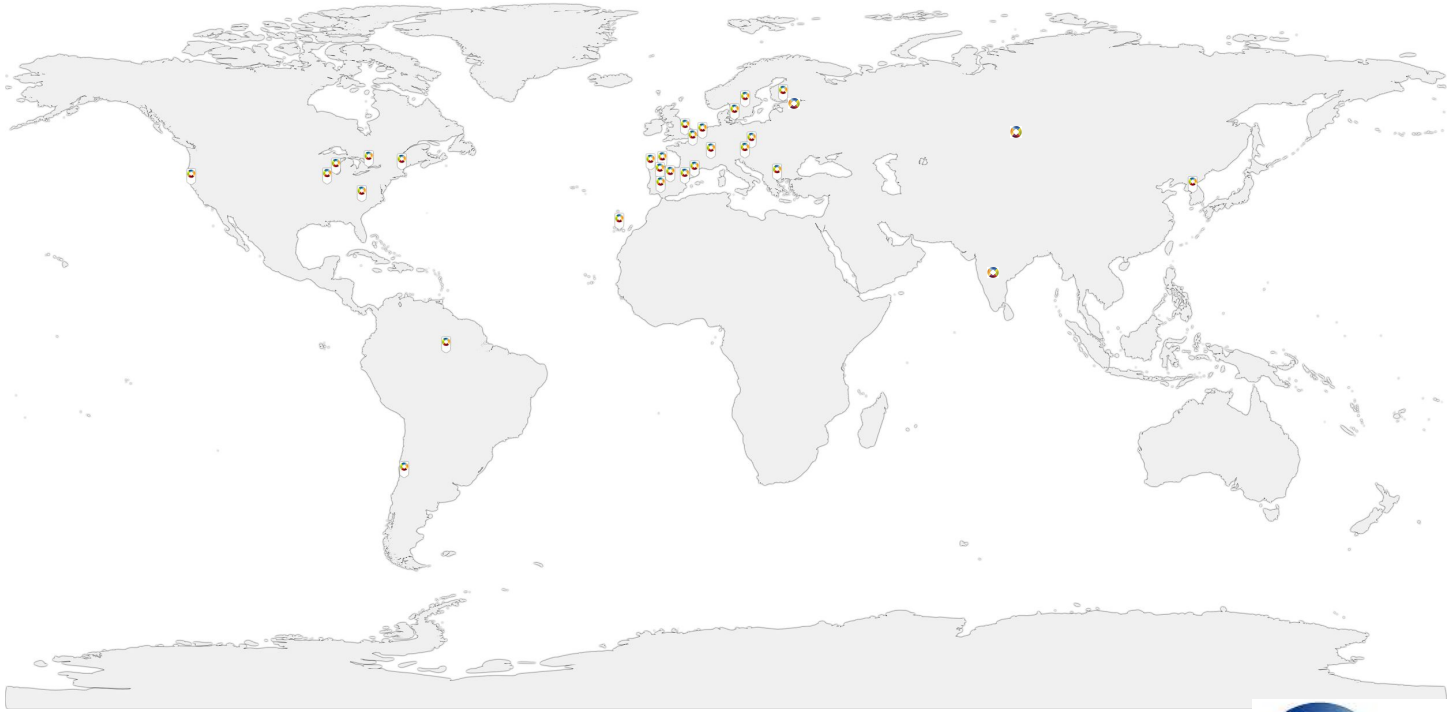
- ~90 employees around the world.
- Areas of expertise:
  - **Browsers and Client-side Web Technologies**
    - *Chromium/Blink, WebKit (WPE & WebKitGTK+), Firefox and Servo*
  - **Graphics Pipeline and Rendering Technologies**
    - *Hardware accelerated 3D APIs, Mesa open source OpenGL (ES), Vulkan drivers, and more*
  - **Compilers and Programming languages**
    - *JavaScript Contributor through TC39*
    - *V8, JavaScriptCore, ChakraCore, SpiderMonkey*
  - **Multimedia**
    - *GStreamer, VA-API, MediaSDK, and more*
  - **Embedded Linux and Device Drivers**
    - *From Graphics to Networking*
  - **Accessibility Tools and Technologies**
    - *Assistive Technologies in Chrome, Firefox or Safari*
  - **Virtualization and Cloud**
    - *QEMU/KVM, CEPH*

igalia

About Igalia

- **World Wide Web Consortium (W3C)**
  - *Co-chairs the W3C's ARIA working group*
- **Linux Foundation**
  - *Also Member of Automotive Grade Linux Steering Committee*
- **Khronos Group**
  - *Participates actively in the Khronos Conformance Testing Working Group focusing on the OpenGL, OpenGL ES, and Vulkan APIs*
- **ECMA**
  - *Chairs the ECMA JavaScript TC39 group*
- **WHATWG**
- **GENIVI® Alliance**
- **Software Freedom Conservancy (SFC), Electronic Frontier Foundation (EFF), AGASOL**

About Igalia



igalia

History of the Project

History of the Project

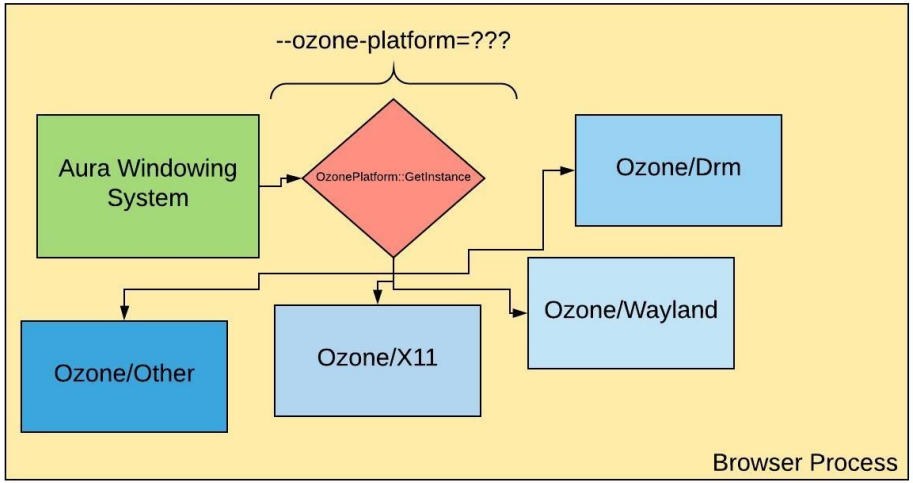
- **Started in May, 2016**
  - Intel's Wayland implementation was not suitable for upstream.
- **Was initially based on Mus+Ash project**
  - UI as a service.
- **Moved to downstream in March, 2017.**
  - Unclear future of the Mus+Ash project.
- **Was finally decided to use a direct Aura integration in April, 2018.**
  - **Aura** - Chromium's own windowing system
  - Moved back to upstream
- **Mus+Ash was finally discontinued in 2019.**
  - Not really related to the current project, but fact.
- **Ozone is part of regular Chrome/Chromium builds since September, 2020.**
  - *Ozone can be enabled with "--enable-features=UseOzonePlatform --ozone-platform=wayland/x11" runtime flag.*
- ***Ozone is default on Linux - H2/2021 (hopefully).***
  - Non-Ozone/X11 (legacy X11) path is removed.

igalia

What is Ozone?

What is Ozone?

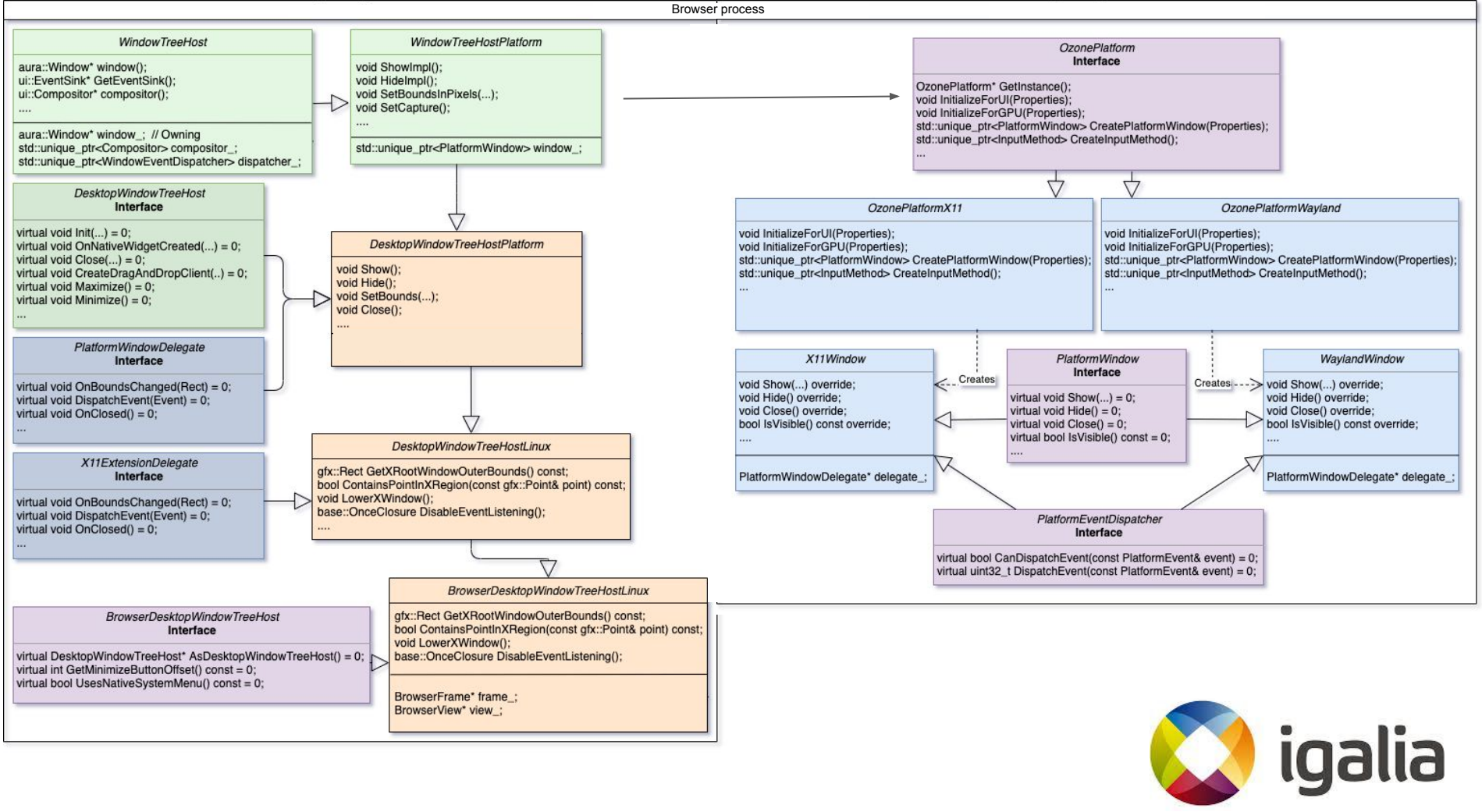
- **Abstraction layer beneath the Aura window system.**
  - Aura - Chromium's window system
- **Goal - make porting of Chromium to other platforms easy:**
  - Interfaces, not ifdefs:
    - ~~#if defined(USE_X11)~~
  - Flexible interfaces:
    - No overly prescriptive interfaces.
  - Runtime binding of platforms:
    - One binary == many backends.
  - Easy out-of-tree platforms:
    - Support for different backends.



igalia

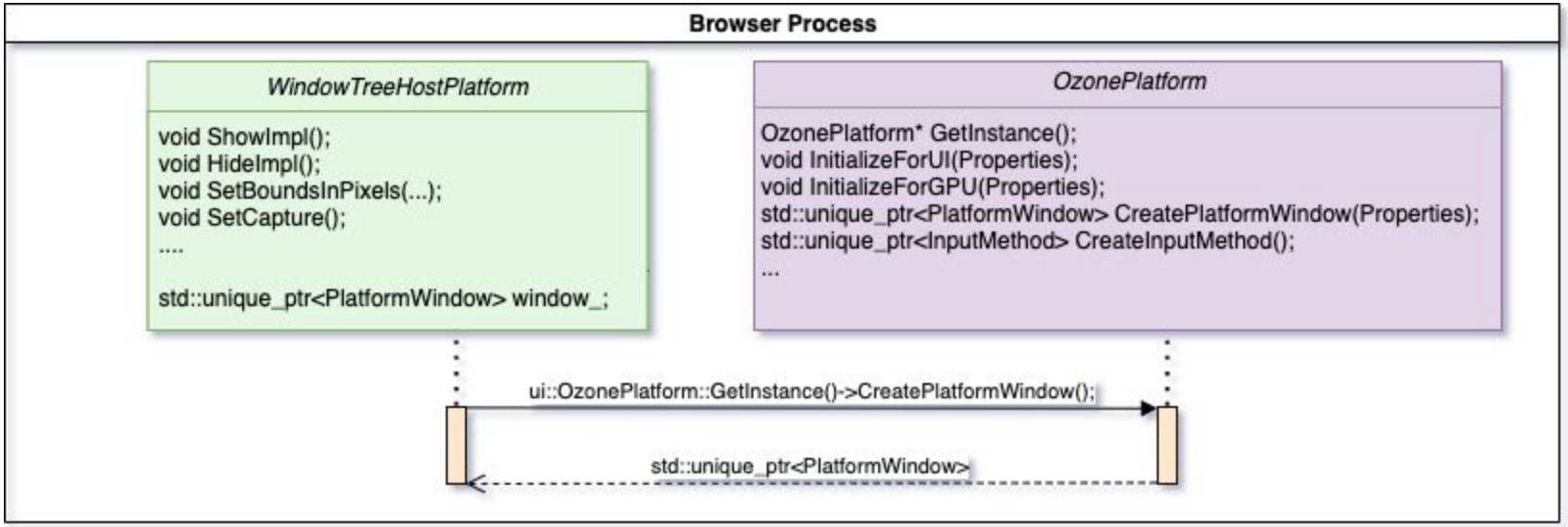
How Ozone Integrates Inside Chromium

Aura/Ozone Design

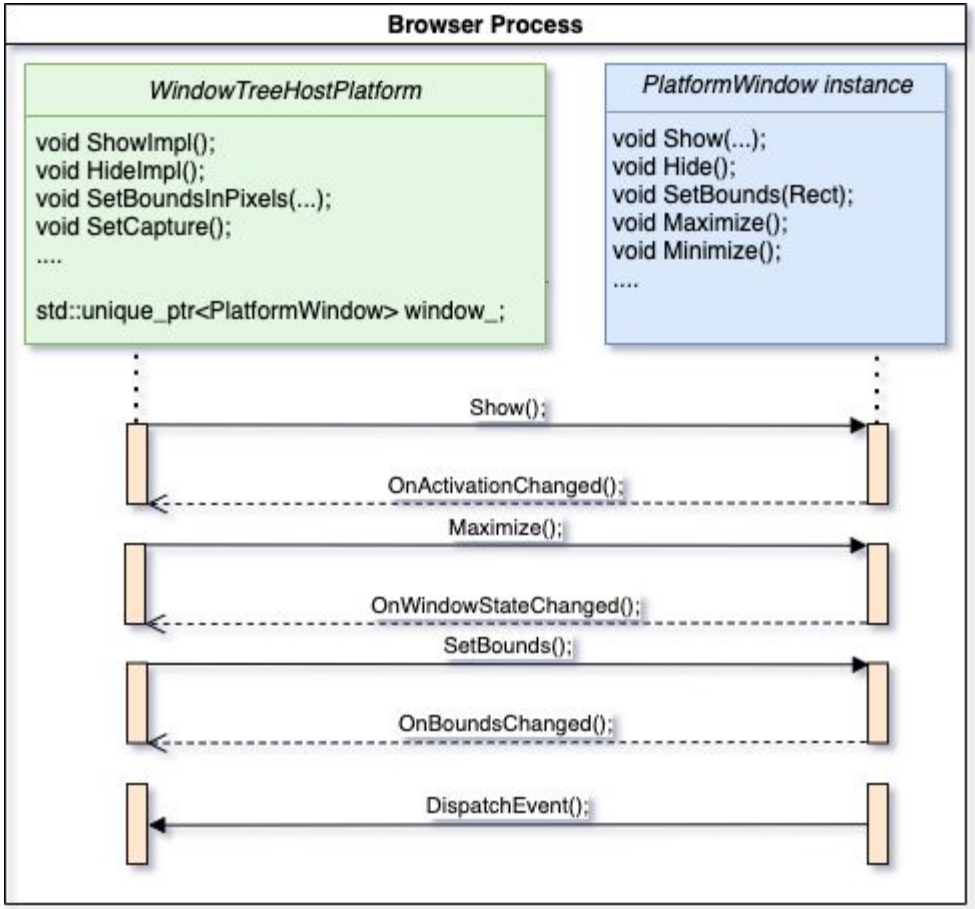

Browser process
WindowTreeHost
aura::Window* window();
ui::EventSink* GetEventSink();
ui::Compositor* compositor();
....
aura::Window* window_; // Owning
std::unique_ptr<Compositor> compositor_;
std::unique_ptr<WindowEventDispatcher> dispatcher_;
WindowTreeHostPlatform
void ShowImpl();
void HideImpl();
void SetBoundsInPixels(...);
void SetCapture();
....
std::unique_ptr<PlatformWindow> window_;
OzonePlatform
Interface
OzonePlatform* GetInstance();
void InitializeForUI(Properties);
void InitializeForGPU(Properties);
std::unique_ptr<PlatformWindow> CreatePlatformWindow(Properties);
std::unique_ptr<InputMethod> CreateInputMethod();
...
DesktopWindowTreeHost
Interface
virtual void Init(...) = 0;
virtual void OnNativeWidgetCreated(...) = 0;
virtual void Close(...) = 0;
virtual void CreateDragAndDropClient(..) = 0;
virtual void Maximize() = 0;
virtual void Minimize() = 0;
...
DesktopWindowTreeHostPlatform
void Show();
void Hide();
void SetBounds(...);
void Close();
....
OzonePlatformX11
void InitializeForUI(Properties);
void InitializeForGPU(Properties);
std::unique_ptr<PlatformWindow> CreatePlatformWindow(Properties);
std::unique_ptr<InputMethod> CreateInputMethod();
...
OzonePlatformWayland
void InitializeForUI(Properties);
void InitializeForGPU(Properties);
std::unique_ptr<PlatformWindow> CreatePlatformWindow(Properties);
std::unique_ptr<InputMethod> CreateInputMethod();
...
PlatformWindowDelegate
Interface
virtual void OnBoundsChanged(Rect) = 0;
virtual void DispatchEvent(Event) = 0;
virtual void OnClosed() = 0;
...
X11Window
void Show(...) override;
void Hide() override;
void Close() override;
bool IsVisible() const override;
....
PlatformWindowDelegate* delegate_;
Creates
PlatformWindow
Interface
virtual void Show(...) = 0;
virtual void Hide() = 0;
virtual void Close() = 0;
virtual bool IsVisible() const = 0;
....
Creates
WaylandWindow
void Show(...) override;
void Hide() override;
void Close() override;
bool IsVisible() const override;
....
PlatformWindowDelegate* delegate_;
X11ExtensionDelegate
Interface
virtual void OnBoundsChanged(Rect) = 0;
virtual void DispatchEvent(Event) = 0;
virtual void OnClosed() = 0;
...
DesktopWindowTreeHostLinux
gfx::Rect GetXRootWindowOuterBounds() const;
bool ContainsPointInXRegion(const gfx::Point& point) const;
void LowerXWindow();
base::OnceClosure DisableEventListening();
....
PlatformEventDispatcher
Interface
virtual bool CanDispatchEvent(const PlatformEvent& event) = 0;
virtual uint32_t DispatchEvent(const PlatformEvent& event) = 0;
BrowserDesktopWindowTreeHost
Interface
virtual DesktopWindowTreeHost* AsDesktopWindowTreeHost() = 0;
virtual int GetMinimizeButtonOffset() const = 0;
virtual bool UsesNativeSystemMenu() const = 0;
BrowserDesktopWindowTreeHostLinux
gfx::Rect GetXRootWindowOuterBounds() const;
bool ContainsPointInXRegion(const gfx::Point& point) const;
void LowerXWindow();
base::OnceClosure DisableEventListening();
BrowserFrame* frame_;
BrowserView* view_;

igalia

Aura/Ozone Interaction

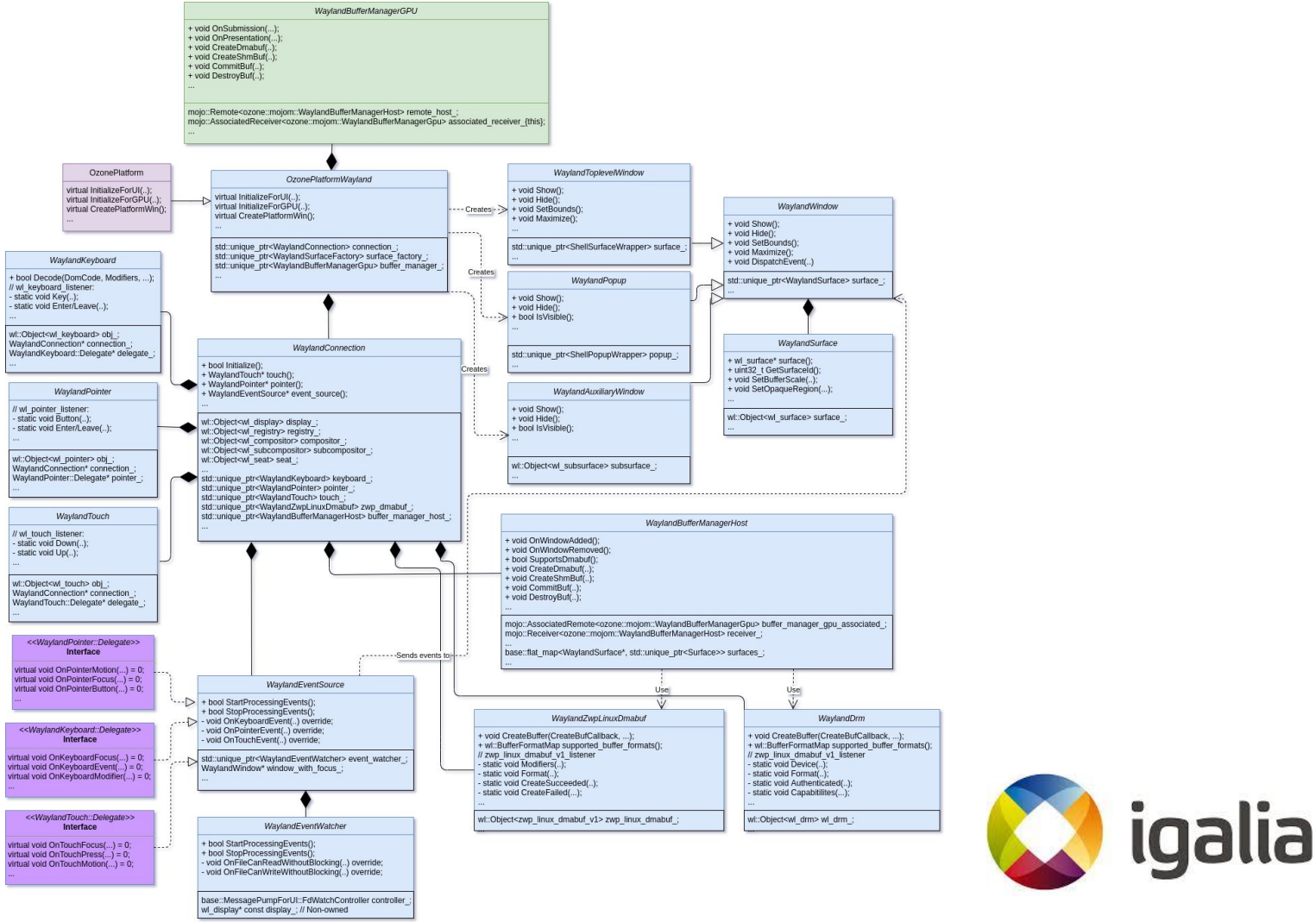

Browser Process
WindowTreeHostPlatform
void ShowImpl();
void HideImpl();
void SetBoundsInPixels(...);
void SetCapture();
....
std::unique_ptr<PlatformWindow> window_;
OzonePlatform
OzonePlatform* GetInstance();
void InitializeForUI(Properties);
void InitializeForGPU(Properties);
std::unique_ptr<PlatformWindow> CreatePlatformWindow(Properties);
std::unique_ptr<InputMethod> CreateInputMethod();
...
ui::OzonePlatform::GetInstance()->CreatePlatformWindow();
std::unique_ptr<PlatformWindow>

igalia


Browser Process
WindowTreeHostPlatform
void ShowImpl();
void HideImpl();
void SetBoundsInPixels(...);
void SetCapture();
....
std::unique_ptr<PlatformWindow> window_;
PlatformWindow instance
void Show(...);
void Hide();
void SetBounds(Rect);
void Maximize();
void Minimize();
....
Show();
OnActivationChanged();
Maximize();
OnWindowStateChanged();
SetBounds();
OnBoundsChanged();
DispatchEvent();

igalia

Design of Ozone/Wayland


WaylandBufferManagerGPU
+ void OnSubmission(...);
+ void OnPresentation(...);
+ void CreateDmabuf(..);
+ void CreateShmBuf(..);
+ void CommitBuf(..);
+ void DestroyBuf(..);
...
mojo::Remote<ozone::mojom::WaylandBufferManagerHost> remote_host_;
mojo::AssociatedReceiver<ozone::mojom::WaylandBufferManagerGpu> associated_receiver_{this};
...
OzonePlatform
virtual InitializeForUI(..);
virtual InitializeForGPU(..);
virtual CreatePlatformWin();
...
OzonePlatformWayland
virtual InitializeForUI(..);
virtual InitializeForGPU(..);
virtual CreatePlatformWin();
...
std::unique_ptr<WaylandConnection> connection_;
std::unique_ptr<WaylandSurfaceFactory> surface_factory_;
std::unique_ptr<WaylandBufferManagerGpu> buffer_manager_;
...
Creates
WaylandToplevelWindow
+ void Show();
+ void Hide();
+ void SetBounds();
+ void Maximize();
...
std::unique_ptr<ShellSurfaceWrapper> surface_;
...
WaylandWindow
+ void Show();
+ void Hide();
+ void SetBounds();
+ void Maximize();
+ void DispatchEvent(..)
std::unique_ptr<WaylandSurface> surface_;
...
WaylandPopup
+ void Show();
+ void Hide();
+ bool IsVisible();
...
std::unique_ptr<ShellPopupWrapper> popup_;
...
WaylandAuxiliaryWindow
+ void Show();
+ void Hide();
+ bool IsVisible();
...
wl::Object<wl_subsurface> subsurface_;
...
WaylandSurface
+ wl_surface* surface();
+ uint32_t GetSurfaceId();
+ void SetBufferScale(..);
+ void SetOpaqueRegion(...);
...
wl::Object<wl_surface> surface_;
...
WaylandKeyboard
+ bool Decode(DomCode, Modifiers, ...);
// wl_keyboard_listener:
- static void Key(..);
- static void Enter/Leave(..);
...
wl::Object<wl_keyboard> obj_;
WaylandConnection* connection_;
WaylandKeyboard::Delegate* delegate_;
...
WaylandConnection
+ bool Initialize();
+ WaylandTouch* touch();
+ WaylandPointer* pointer();
+ WaylandEventSource* event_source();
...
wl::Object<wl_display> display_;
wl::Object<wl_registry> registry_;
wl::Object<wl_compositor> compositor_;
wl::Object<wl_subcompositor> subcompositor_;
wl::Object<wl_seat> seat_;
...
std::unique_ptr<WaylandKeyboard> keyboard_;
std::unique_ptr<WaylandPointer> pointer_;
std::unique_ptr<WaylandTouch> touch_;
std::unique_ptr<WaylandZwpLinuxDmabuf> zwp_dmabuf_;
std::unique_ptr<WaylandBufferManagerHost> buffer_manager_host_;
...
WaylandPointer
// wl_pointer_listener:
- static void Button(..);
- static void Enter/Leave(..);
...
wl::Object<wl_pointer> obj_;
WaylandConnection* connection_;
WaylandPointer::Delegate* pointer_;
...
WaylandTouch
// wl_touch_listener:
- static void Down(..);
- static void Up(..);
...
wl::Object<wl_touch> obj_;
WaylandConnection* connection_;
WaylandTouch::Delegate* delegate_;
...
WaylandBufferManagerHost
+ void OnWindowAdded();
+ void OnWindowRemoved();
+ bool SupportsDmabuf();
+ void CreateDmabuf(..);
+ void CreateShmBuf(..);
+ void CommitBuf(..);
+ void DestroyBuf(..);
...
mojo::AssociatedRemote<ozone::mojom::WaylandBufferManagerGpu> buffer_manager_gpu_associated_;
mojo::Receiver<ozone::mojom::WaylandBufferManagerHost> receiver_;
...
base::flat_map<WaylandSurface*, std::unique_ptr<Surface>> surfaces_;
...
<<WaylandPointer::Delegate>>
Interface
virtual void OnPointerMotion(...) = 0;
virtual void OnPointerFocus(...) = 0;
virtual void OnPointerButton(...) = 0;
...
<<WaylandKeyboard::Delegate>>
Interface
virtual void OnKeyboardFocus(...) = 0;
virtual void OnKeyboardEvent(...) = 0;
virtual void OnKeyboardModifier(...) = 0;
...
<<WaylandTouch::Delegate>>
Interface
virtual void OnTouchFocus(...) = 0;
virtual void OnTouchPress(...) = 0;
virtual void OnTouchMotion(...) = 0;
...
Sends events to
WaylandEventSource
+ bool StartProcessingEvents();
+ bool StopProcessingEvents();
- void OnKeyboardEvent(..) override;
- void OnPointerEvent(..) override;
- void OnTouchEvent(..) override;
std::unique_ptr<WaylandEventWatcher> event_watcher_;
WaylandWindow* window_with_focus_;
...
WaylandEventWatcher
+ bool StartProcessingEvents();
+ bool StopProcessingEvents();
- void OnFileCanReadWithoutBlocking(..) override;
- void OnFileCanWriteWithoutBlocking(..) override;
base::MessagePumpForUI::FdWatchController controller_;
wl_display* const display_; // Non-owned
Use
WaylandZwpLinuxDmabuf
+ void CreateBuffer(CreateBufCallback, ...);
+ wl::BufferFormatMap supported_buffer_formats();
// zwp_linux_dmabuf_v1_listener
- static void Modifiers(..);
- static void Format(..);
- static void CreateSucceeded(..);
- static void CreateFailed(...);
...
wl::Object<zwp_linux_dmabuf_v1> zwp_linux_dmabuf_;
Use
WaylandDrm
+ void CreateBuffer(CreateBufCallback, ...);
+ wl::BufferFormatMap supported_buffer_formats();
// zwp_linux_dmabuf_v1_listener
- static void Device(..);
- static void Format(..);
- static void Authenticated(..);
- static void Capabilities(..);
...
wl::Object<wl_drm> wl_drm_;
igalia

Ozone/Wayland Limitations

Ozone/Wayland Limitations

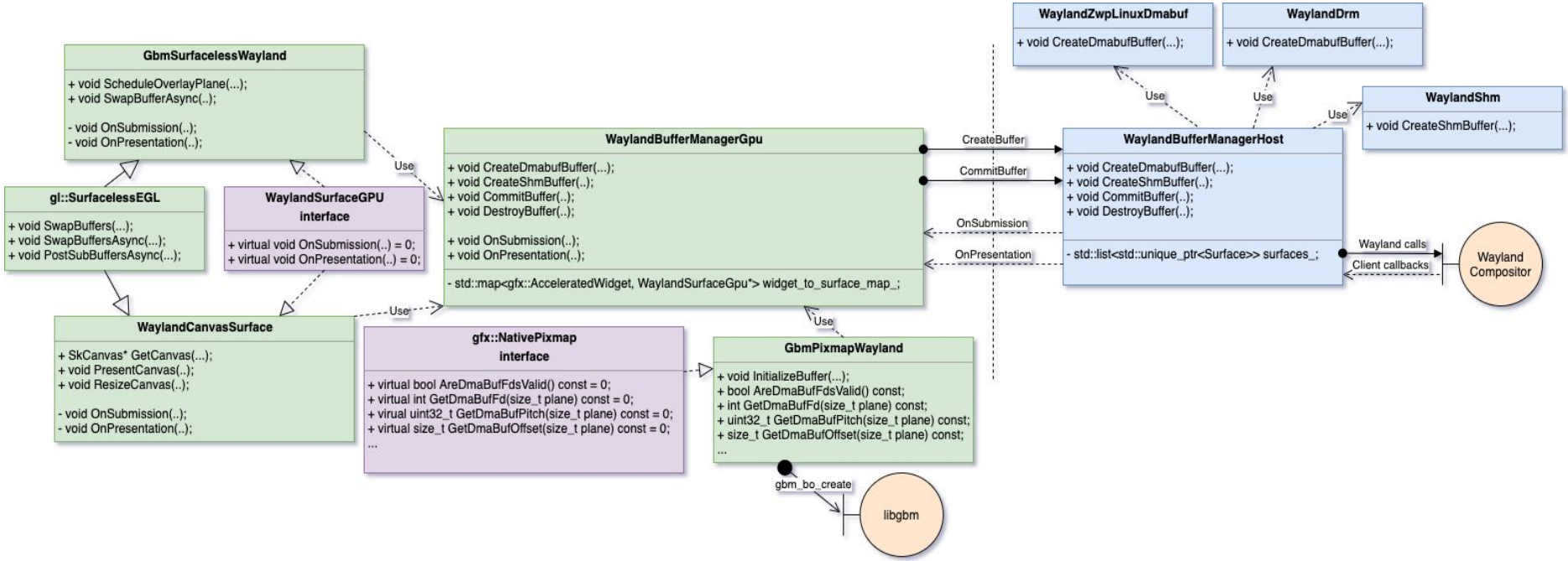
- **Wayland EGL**
  - Required access to WaylandConnection.
  - Could only be used with “in-process-gpu”.

- **Tab Dragging Feature**
  - Required additional extension.
  - Wasn’t possible to use original Drag&Drop extension because of some Chromium’s internal assumptions.

Limitation 1 - Graphics Pipeline

Graphics Design

- **Consists of GPU and Browser process side objects**
  - WaylandBufferManagerHost
  - WaylandBufferManagerGpu
- **Uses surfaceless drawing**
  - SurfacelessEGL
- **Uses Mojo for communication**
  - Associated pipe that ensures order of messages
- **Uses libgbm to create dmabuf**
  - Reused existing implementation from Ozone/Drm
- **Allows to reuse GpuMemoryBuffers framework without any modifications**
  - Ozone/Drm is the first user

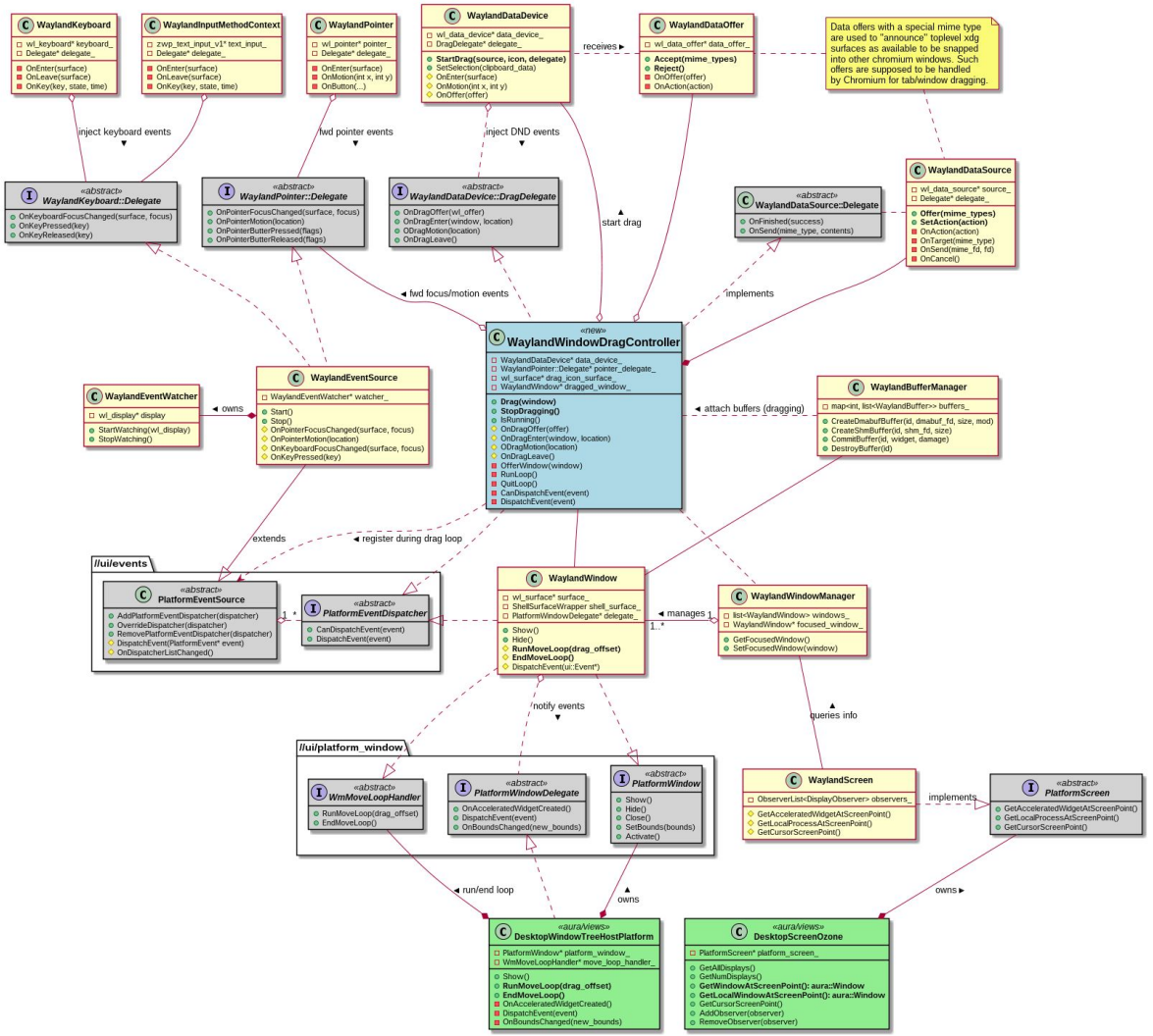

GbmSurfacelessWayland
+ void ScheduleOverlayPlane(...);
+ void SwapBufferAsync(..);
- void OnSubmission(..);
- void OnPresentation(..);
gl::SurfacelessEGL
+ void SwapBuffers(...);
+ void SwapBuffersAsync(...);
+ void PostSubBuffersAsync(...);
WaylandSurfaceGPU
interface
+ virtual void OnSubmission(..) = 0;
+ virtual void OnPresentation(..) = 0;
Use
WaylandBufferManagerGpu
+ void CreateDmabufBuffer(...);
+ void CreateShmBuffer(..);
+ void CommitBuffer(..);
+ void DestroyBuffer(..);
+ void OnSubmission(..);
+ void OnPresentation(..);
- std::map<gfx::AcceleratedWidget, WaylandSurfaceGpu*> widget_to_surface_map_;
CreateBuffer
CommitBuffer
OnSubmission
OnPresentation
WaylandBufferManagerHost
+ void CreateDmabufBuffer(...);
+ void CreateShmBuffer(..);
+ void CommitBuffer(..);
+ void DestroyBuffer(..);
- std::list<std::unique_ptr<Surface>> surfaces_;
WaylandZwpLinuxDmabuf
+ void CreateDmabufBuffer(...);
WaylandDrm
+ void CreateDmabufBuffer(...);
WaylandShm
+ void CreateShmBuffer(...);
Use
Wayland calls
Client callbacks
Wayland
Compositor
WaylandCanvasSurface
+ SkCanvas* GetCanvas(...);
+ void PresentCanvas(..);
+ void ResizeCanvas(..);
- void OnSubmission(..);
- void OnPresentation(..);
gfx::NativePixmap
interface
+ virtual bool AreDmaBufFdsValid() const = 0;
+ virtual int GetDmaBufFd(size_t plane) const = 0;
+ virtual uint32_t GetDmaBufPitch(size_t plane) const = 0;
+ virtual size_t GetDmaBufOffset(size_t plane) const = 0;
...
GbmPixmapWayland
+ void InitializeBuffer(...);
+ bool AreDmaBufFdsValid() const;
+ int GetDmaBufFd(size_t plane) const;
+ uint32_t GetDmaBufPitch(size_t plane) const;
+ size_t GetDmaBufOffset(size_t plane) const;
...
gbm_bo_create
libgbm

igalia

Limitation 2 -
Tab Dragging Feature

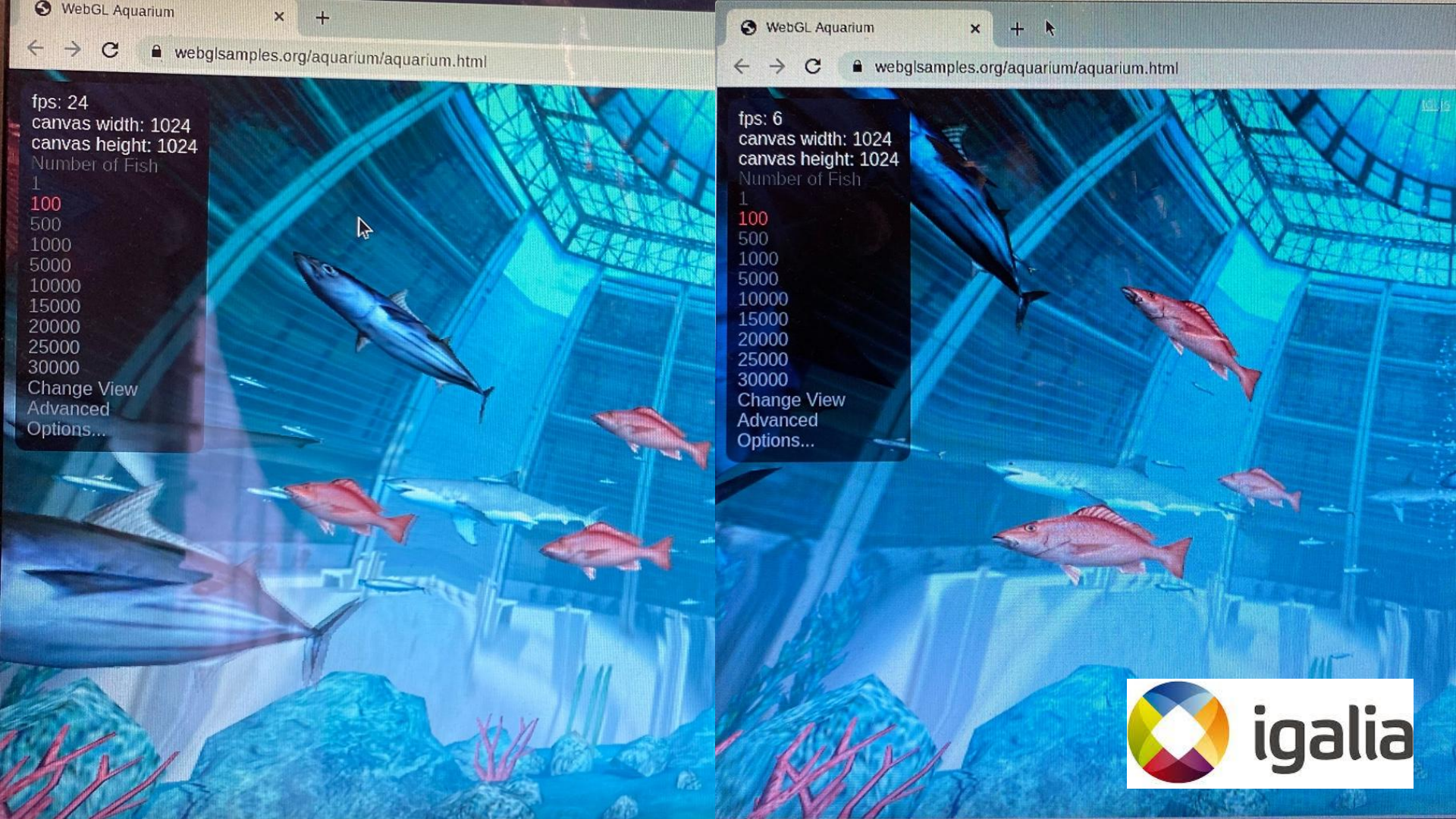
Limitation 2 - Tab Dragging Feature

- **Problem**
  - Cannot properly “swallow” and “unswallow” a surface.
  - Chromium uses a so-called “preview mode” for windows that are being dragged
  - Existing DnD protocol has limited functionality
- **Solution**
  - Implement protocol extension that would allow to reuse existing surface as a drag icon and allow a further reuse.
- **More details -**
  https://docs.google.com/document/d/1s6OwTi_WC-pS21WLGQYI39yw2m42ZlVolUXBcIljXB4/edit#heading=h.gjdgxs


WaylandKeyboard
wl_keyboard* keyboard_
Delegate* delegate_
OnEnter(surface)
OnLeave(surface)
OnKey(key, state, time)
WaylandInputMethodContext
zwp_text_input_v1* text_input_
Delegate* delegate_
OnEnter(surface)
OnLeave(surface)
OnKey(key, state, time)
WaylandPointer
wl_pointer* pointer_
Delegate* delegate_
OnEnter(surface)
OnMotion(int x, int y)
OnButton(...)
WaylandDataDevice
wl_data_device* data_device_
DragDelegate* delegate_
StartDrag(source, icon, delegate)
SetSelection(clipboard_data)
OnEnter(surface)
OnMotion(int x, int y)
OnOffer(offer)
receives
WaylandDataOffer
wl_data_offer* data_offer_
Accept(mime_types)
Reject()
OnOffer(offer)
OnAction(action)
Data offers with a special mime type are used to "announce" toplevel xdg surfaces as available to be snapped into other chromium windows. Such offers are supposed to be handled by Chromium for tab/window dragging.
inject keyboard events
fwd pointer events
inject DND events
«abstract»
WaylandKeyboard::Delegate
OnKeyboardFocusChanged(surface, focus)
OnKeyPressed(key)
OnKeyReleased(key)
«abstract»
WaylandPointer::Delegate
OnPointerFocusChanged(surface, focus)
OnPointerMotion(location)
OnPointerButtonPressed(flags)
OnPointerButtonReleased(flags)
«abstract»
WaylandDataDevice::DragDelegate
OnDragOffer(wl_offer)
OnDragEnter(window, location)
ODragMotion(location)
OnDragLeave()
start drag
«abstract»
WaylandDataSource::Delegate
OnFinished(success)
OnSend(mime_type, contents)
WaylandDataSource
wl_data_source* source_
Delegate* delegate_
Offer(mime_types)
SetAction(action)
OnAction(action)
OnTarget(mime_type)
OnSend(mime_fd, fd)
OnCancel()
fwd focus/motion events
implements
«new»
WaylandWindowDragController
WaylandDataDevice* data_device_
WaylandPointer::Delegate* pointer_delegate_
wl_surface* drag_icon_surface_
WaylandWindow* dragged_window_
Drag(window)
StopDragging()
IsRunning()
OnDragOffer(offer)
OnDragEnter(window, location)
ODragMotion(location)
OnDragLeave()
OfferWindow(window)
RunLoop()
QuitLoop()
CanDispatchEvent(event)
DispatchEvent(event)
WaylandEventWatcher
wl_display* display
StartWatching(wl_display)
StopWatching()
owns
WaylandEventSource
WaylandEventWatcher* watcher_
Start()
Stop()
OnPointerFocusChanged(surface, focus)
OnPointerMotion(location)
OnKeyboardFocusChanged(surface, focus)
OnKeyPressed(key)
attach buffers (dragging)
WaylandBufferManager
map<int, list<WaylandBuffer>> buffers_
CreateDmabufBuffer(id, dmabuf_fd, size, mod)
CreateShmBuffer(id, shm_fd, size)
CommitBuffer(id, widget, damage)
DestroyBuffer(id)
extends
register during drag loop
//ui/events
«abstract»
PlatformEventSource
AddPlatformEventDispatcher(dispatcher)
OverrideDispatcher(dispatcher)
RemovePlatformEventDispatcher(dispatcher)
DispatchEvent(PlatformEvent* event)
OnDispatcherListChanged()
«abstract»
PlatformEventDispatcher
CanDispatchEvent(event)
DispatchEvent(event)
1
*
WaylandWindow
wl_surface* surface_
ShellSurfaceWrapper shell_surface_
PlatformWindowDelegate* delegate_
Show()
Hide()
RunMoveLoop(drag_offset)
EndMoveLoop()
DispatchEvent(ui::Event*)
manages
1..*
WaylandWindowManager
list<WaylandWindow> windows_
WaylandWindow* focused_window_
GetFocusedWindow()
SetFocusedWindow(window)
notify events
queries info
//ui/platform_window
«abstract»
WmMoveLoopHandler
RunMoveLoop(drag_offset)
EndMoveLoop()
«abstract»
PlatformWindowDelegate
OnAcceleratedWidgetCreated()
DispatchEvent(event)
OnBoundsChanged(new_bounds)
«abstract»
PlatformWindow
Show()
Hide()
Close()
SetBounds(bounds)
Activate()
WaylandScreen
ObserverList<DisplayObserver> observers_
GetAcceleratedWidgetAtScreenPoint()
GetLocalProcessAtScreenPoint()
GetCursorScreenPoint()
implements
«abstract»
PlatformScreen
GetAcceleratedWidgetAtScreenPoint()
GetLocalProcessAtScreenPoint()
GetCursorScreenPoint()
run/end loop
owns
owns
«aura/views»
DesktopWindowTreeHostPlatform
PlatformWindow* platform_window_
WmMoveLoopHandler* move_loop_handler_
Show()
RunMoveLoop(drag_offset)
EndMoveLoop()
OnAcceleratedWidgetCreated()
DispatchEvent(event)
OnBoundsChanged(new_bounds)
«aura/views»
DesktopScreenOzone
PlatformScreen* platform_screen_
GetAllDisplays()
GetNumDisplays()
GetWindowAtScreenPoint(): aura::Window
GetLocalWindowAtScreenPoint(): aura::Window
GetCursorScreenPoint()
AddObserver(observer)
RemoveObserver(observer)

igalia

Performance Comparison - non-Ozone/X11 vs Ozone/Wayland Raspberry Pi3 B+


WebGL Aquarium
webglsamples.org/aquarium/aquarium.html
fps: 24
canvas width: 1024
canvas height: 1024
Number of Fish
1
100
500
1000
5000
10000
15000
20000
25000
30000
Change View
Advanced
Options...
WebGL Aquarium
webglsamples.org/aquarium/aquarium.html
fps: 6
canvas width: 1024
canvas height: 1024
Number of Fish
1
100
500
1000
5000
10000
15000
20000
25000
30000
Change View
Advanced
Options...
igalia

Shipping of Ozone/Wayland

Shipping of Ozone/Wayland

- **Mostly all the features are in place except tab drag.**

- **Most test suites are exercised with ozone/wayland backend:**
  - Various browser, content browser tests, and unittests,
  - More test suites are to be enabled including gpu tests,
  - See linux-ozone-rel bot.

- **Ozone is part of Chrome releases since M87.**
  - Can be used with *“--enable-features=UseOzonePlatform --ozone-platform=wayland/x11”* runtime flag.

- **Ozone/Wayland cannot be shipped because of X11.**

igalia

Bonus:
Migration of X11 to Ozone

Migration of X11 to Ozone

- **Started around May/June, 2019.**
- **Shared most of the non-Ozone/X11 implementation with Ozone/X11**
  - Including non-X11 specific bits that Ozone/Wayland could reuse
    - Like StatusIconDbus.
- **Switched Ozone from compile-time to runtime switch.**
  - Happened in September, 2020,
  - Can be tried with *“--enable-features=UseOzonePlatform --ozone-platform=wayland/x11”.*
- **Can be tracked at https://crbug.com/789065**

Thank you

Maksim Sisov - msisov@igalia.com
Nick Diego Yamane - nickdiego@igalia.com
Alexander Dunaev - adunaev@igalia.com
Antonio Gomes - agomes@igalia.com