X11 与 Wayland

Linux GUI 的曲折发展史

Coelacanthus

2022 年 4 月 21 日

X11 的由来

MIT 造的图形界面轮子。全称是 X Window System。
自 1984 年 X1 起步，到 2012 年 X11R7.7 发布。
几个关键点：

- 1987 年 9 月 15 日，现用协议 X11 的第一个版本。
- 1994 年 5 月 16 日，X11R6，迁移到 XFree86 主持开发，xinput 等关键组件被添加。
- 2004 年 9 月 9 日，X11R6.8.0，混成器拓展，已死的 XEvIE 拓展。
- 2007 年 2 月 15 日，X11R7.2，XCB。

栈式窗口管理器

早期栈式窗口管理器的设计特点:

- 使用画家算法绘制图像（或者说，深度缓冲技术?）
- 从而，可以重绘时仅绘制未被覆盖的部分，使得重绘永远不会超过屏幕面积
- 从而可以大幅度节省带宽和内存占用

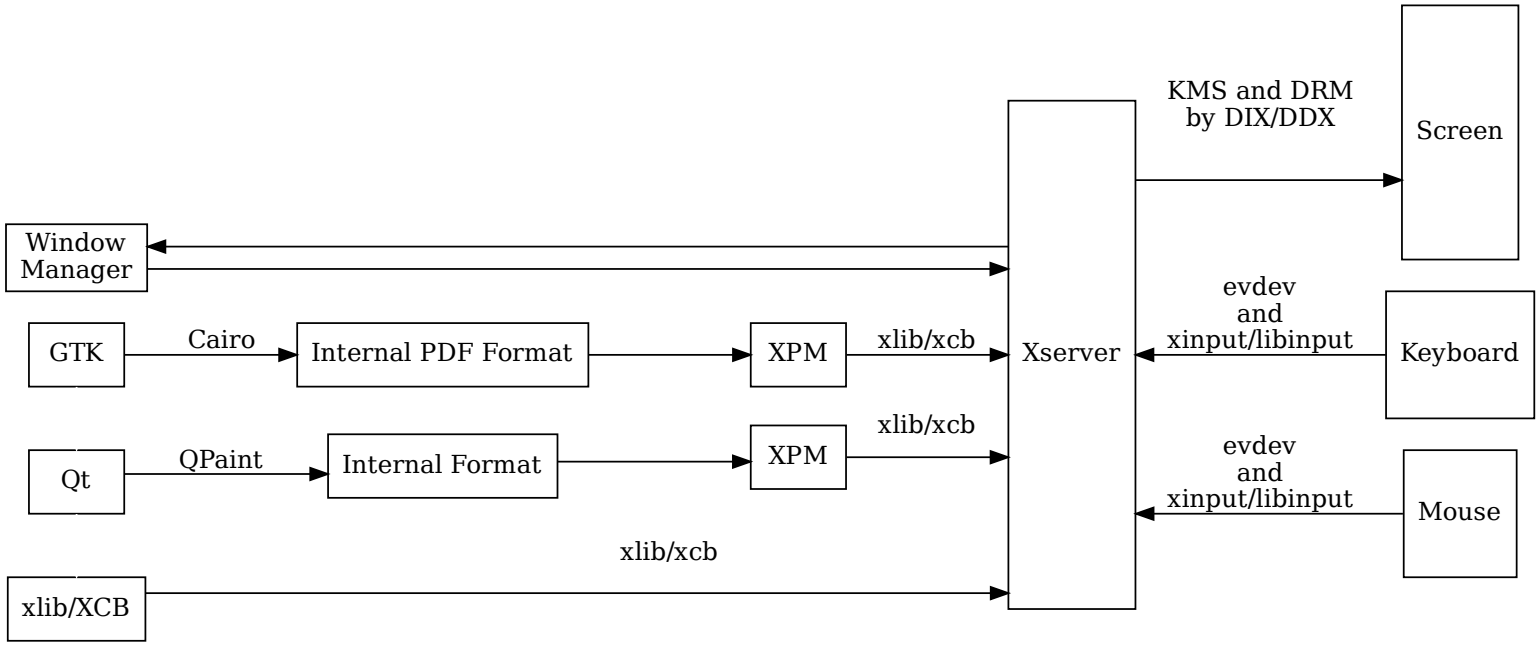
早期栈式窗口管理器缺点

缺点：

- 窗口必须为矩形，否则无法处理
- 不支持半透明（毛玻璃爱好者无能狂怒）
- 为效率考虑，移动窗口的过程不会进行重绘

在早期 X，早期 Windows 和早期 mac 上都可以看到这些特点。

X11 的基本架构



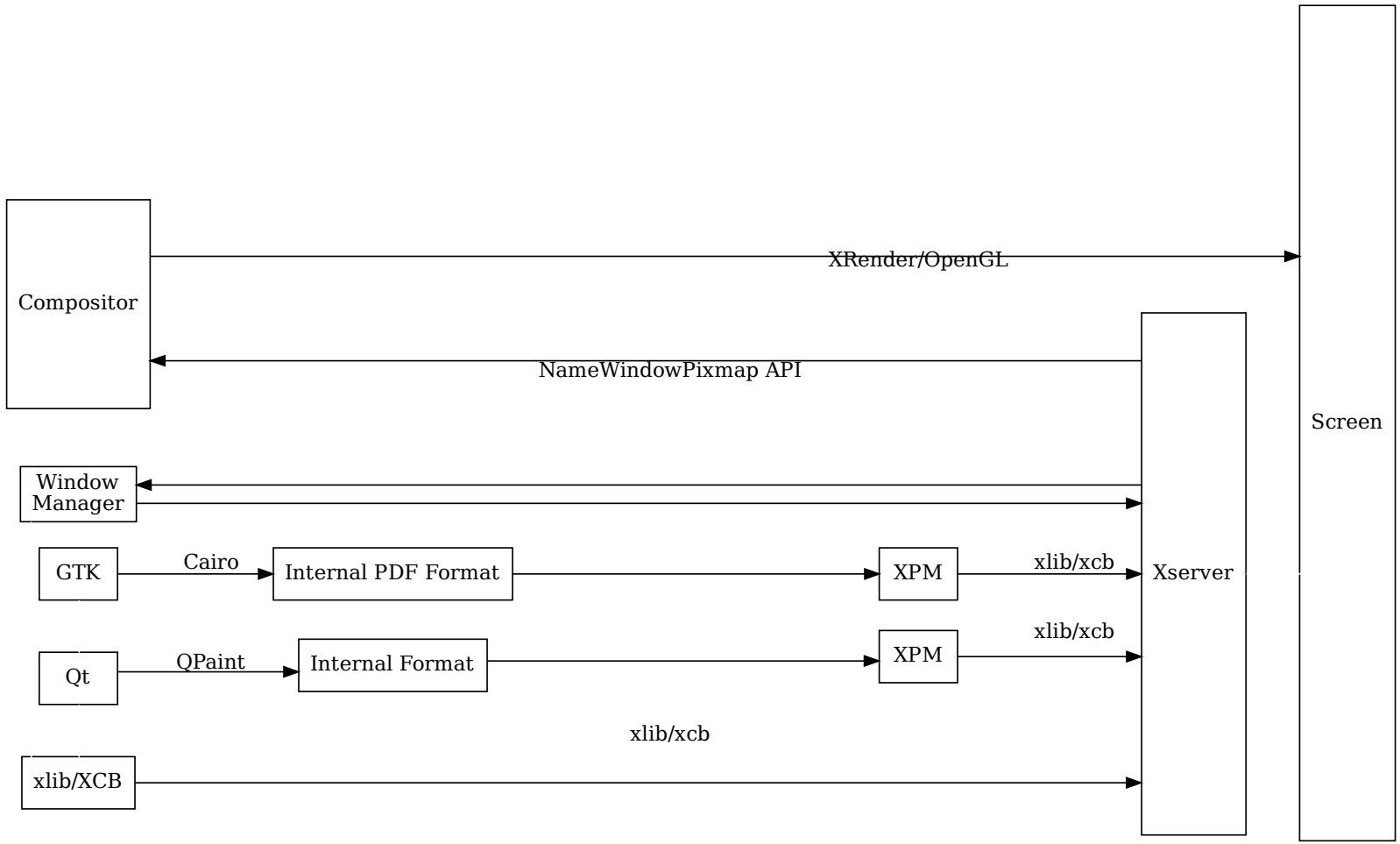
混成器拓展下的 X11 架构

为了提高绘图效率和完善功能，X11R6.8 引入了混成器拓展（Composite Extension）。主要内容是，提供 API，使得程序可以：

- 获取某个窗口的渲染结果（使用 `RedirectSubwindows` 和 `NameWindowPixmap` API）
- 得到一个特殊的绘图窗口，该窗口会覆盖在屏幕最上层（`GetOverlayWindow`）
- 取得窗口裁剪边界（`CreateRegionFromBorderClip`）

混成器拓展下的 X11 架构

这样，X11 的架构在混成器拓展下就变成了下图的样子：



那么输入事件呢?

有了混成器，我们现在可以重定向输出事件。
那么，输入事件呢?
输入事件可以分为两类，键盘事件，和鼠标事件。
对于键盘事件，如果要处理，需要截取所有键盘事件，然后重新派发，水很深，所以大多数混成器都直接放弃了重定向键盘事件。
对于鼠标事件，就更复杂了，所以混成器一般使用 XFixes 拓展提供的功能将上面提到的 OverlayWindow 设置成对鼠标事件透明，从而使得点击直接透传到下面的原有窗口。
但是这也造成了严重的问题，就是一旦混成器绘制的图像与原有的不符，就会造成错误的结果。

那么输入事件呢?

所以混成器一般会有两个状态，一个状态保证窗口大小位置形状完全一致，也就是通常的状态。
如果需要的话（例如有些桌面有一个应用切换界面），可以发生不一致，但是这时候所有鼠标事件都会被拦截，只保留混成器专门留下的（比如切换模式下点击窗口，或者点击专门的关闭窗口按钮）。
注意这里只能有专门的关闭按钮，因为这时候鼠标输入是无法传递给窗口的。

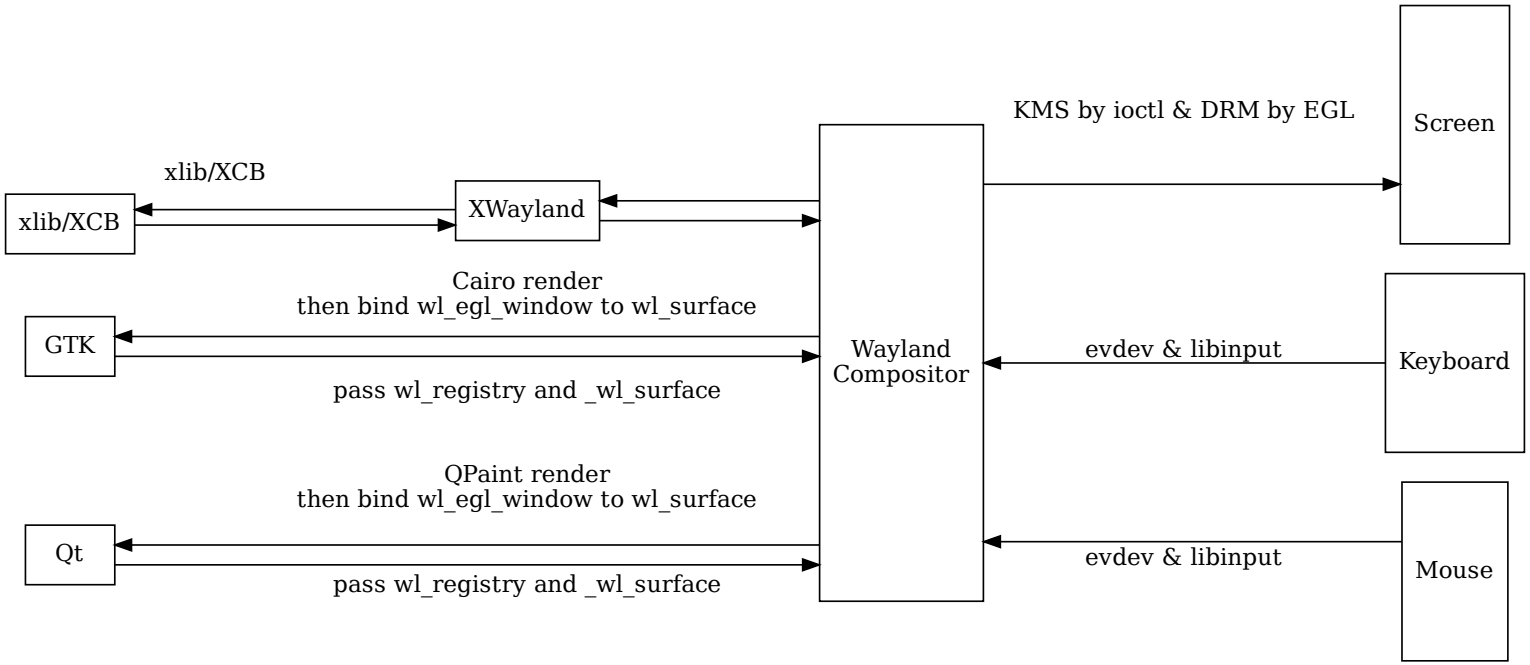
我们为什么要有 Wayland

面对这些问题，有人提出了一个观点：
既然 X11 因为基础设计原因有着这样那样的问题，我们为什么不推倒重来呢，重新设计，避开 X11 所遇到的问题，从头开始。
这就是 Wayland 了。
那么，Wayland 都解决了什么问题呢?

- 同样的位图从 client 发送到 xserver，再从 xserver 发送到混成器，这是无谓的开销，正如 Daniels 所说：
  and what's the X server? really bad IPC[1]
- 只有像素的绘图原语，难以实现分数缩放和由 display server 控制的缩放，以及子像素抗锯齿
- 如上文所述，基本可以认为是没有的输入事件重定向

[1] https://people.freedesktop.org/~daniels/lca2013-wayland-x11.pdf

Wayland 的架构



What’s new?

- No Xserver, No bad IPC!

What’s new?

- No Xserver, No bad IPC!
- No render API!

What’s new?

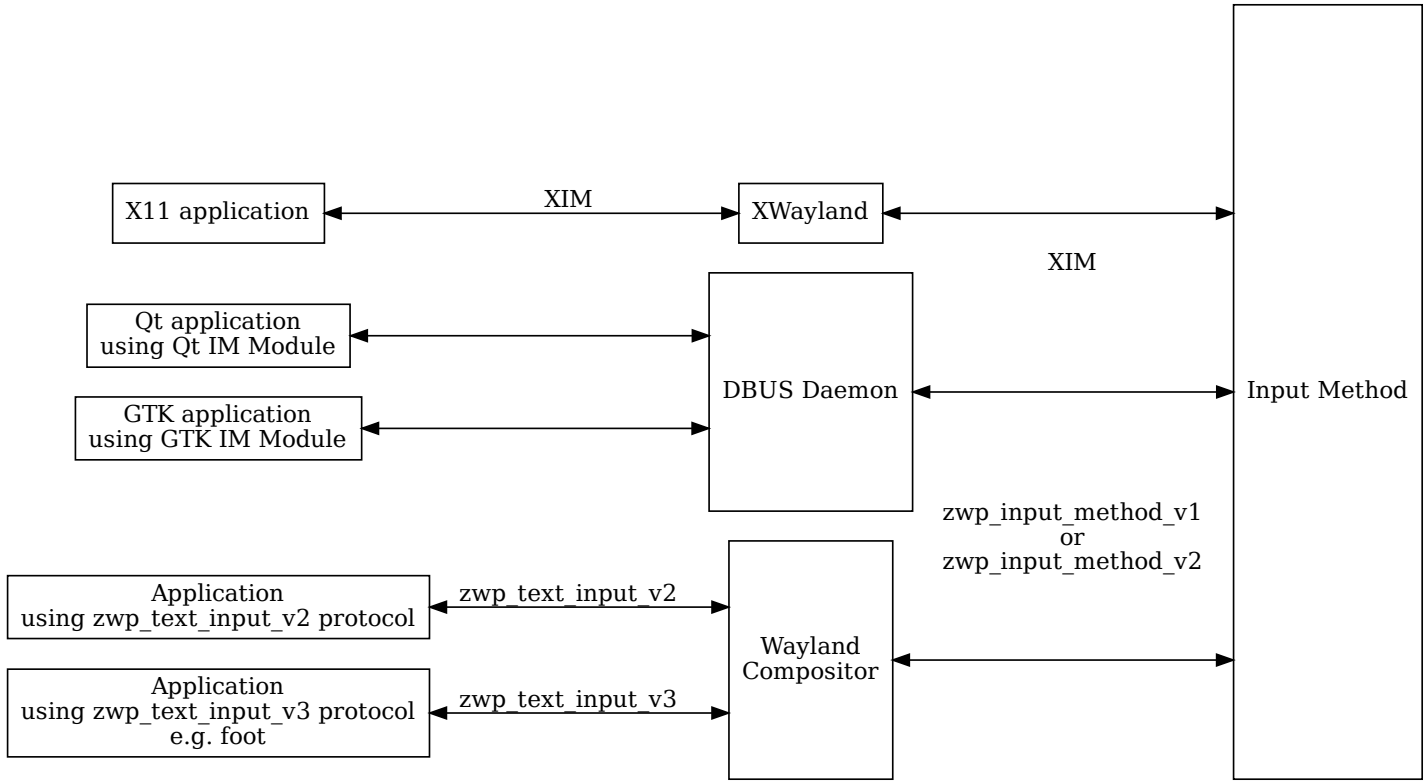
- No Xserver, No bad IPC!
- No render API!
- No alone WM and Compositor, they are same now!

Wayland 的罪与罚

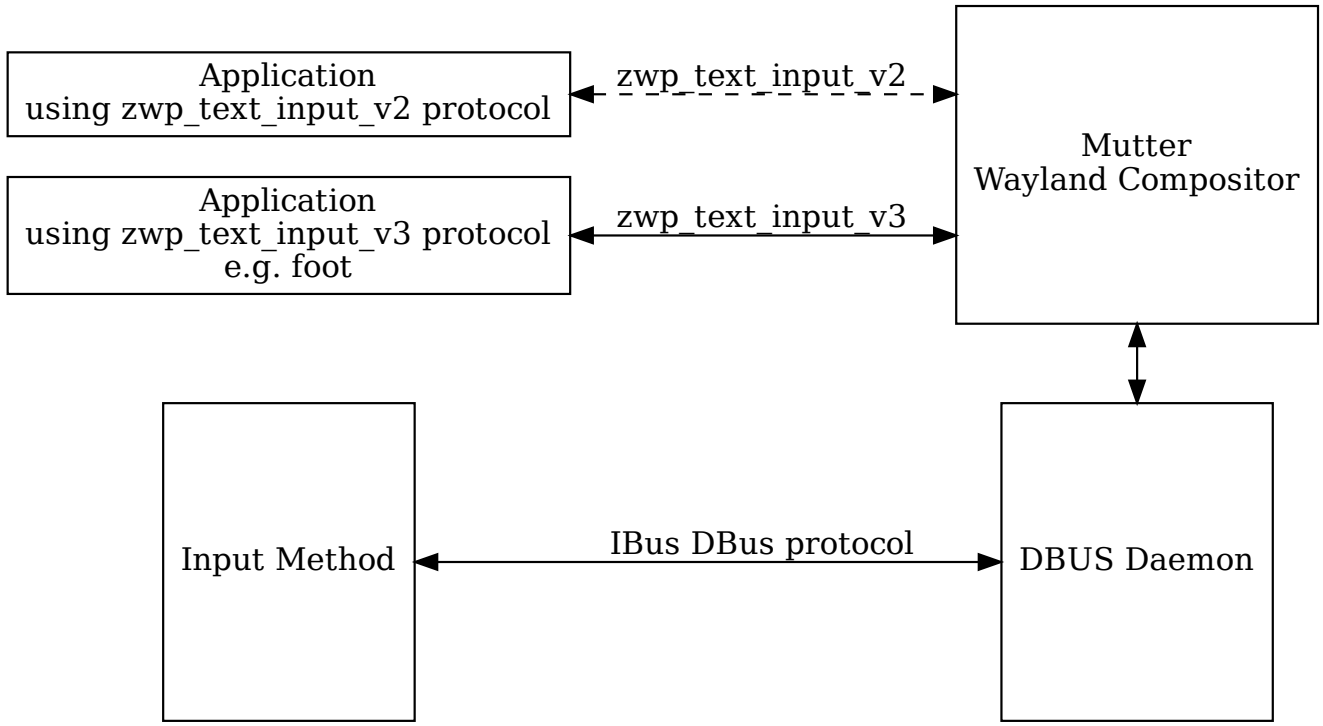
- 首先是老生常谈的，有许多应用尚不支持 Native Wayland，而且他们中的很多也不可能支持 Wayland 了
- 在 X11 下 Window 有一个 Class，可以用于识别窗口，由此可以实现很多有用的功能，比如窗口自动归类和对不同应用使用不同的输入法（Rime）和输入法状态（Fcitx5）
- 因为 Wayland 的协议设计缺陷，输入法不能获得准确的光标位置来绘制候选框，同样的，不能移动候选框位置
- 同样的，因为 Wayland 输入法协议有多个版本，如果应用，输入法和 Wayland 混成器支持的协议没有交集，会直接无法使用输入法[2]
- 同时，Wayland 输入法协议要求输入法进程被 Compositor 启动

[2] https://gitlab.freedesktop.org/wayland/wayland-protocols/-/issues/39#implementation-matrix

Wayland 下的输入法架构



害群之马



讨论时间

谢谢大家