Subject: WebViewCtrl using native engine
Posted by forlano on Sun, 24 May 2026 13:45:14 GMT
View Forum Message <> Reply to Message

Hello,

here is a crossplatform WebViewCtrl that uses the native browser engine in each OS. More info in the report-notes.txt. For windows a webview2 dll (included in the package) should be next to the executable.
Screenshot are here https://www.ultimatepp.org/forums/index.php?t=msg&th=125 51&start=0&

I developed it with the precious help of my friend Claudio Antropico. It was a challeging project made of continuos web research, parsing U++ source code, and debug but at the end we successed.

It would be nice if more expert U++ user could test it and add in designer as a standard Ctrl.

The Linux side took much more time for lack of documentation and not optimal design from GTK. At the end the only way was to embed webkitGTK on a popup that follow its U++ container. This is not an optimal solution but still quite effective.

Claudio come up with some proposal but that was even more challenging and required to modify the U++ library in some point (make a Cairo clip on a drawingarea to produce transparency). So we limited ourself to the previous solution. Anyway, I asked him to make a report of the problem encountered and of a partial solution that could help the future integration of GTK widget in U++. It requires just a little change of GtkCtrl.h (1 line) and few line in GtkCreate.cpp. At bottom I add his suggestion to be evaluated by our experts.

Thanks and best regards,

Luigi

```
  ============================================================
====================
  Subject: Proposal — GtkOverlay patch in GtkCreate.cpp for native GtkWidget
           embedding in U++ Ctrl on Linux
  ============================================================
====================

BACKGROUND
----------
We built a cross-platform browser engine Ctrl for U++ called WebViewCtrl,
embedding WebView2 on Windows, WKWebView on macOS, and WebKitGTK on Linux.
On Windows and macOS we achieved true native embedding — the browser engine
lives inside the U++ Ctrl as a real child widget with correct z-order,
resize, and focus handling.
```

On Linux we hit a fundamental obstacle.

THE PROBLEM

WebKitWebView is a GtkWidget and can be added to any GtkContainer via gtk_container_add() or gtk_fixed_put(). However, after studying the U++ source (GtkCreate.cpp, GtkCtrl.cpp), we discovered that U++ does NOT use a GtkContainer for child Ctrl layout on Linux. The actual widget tree is:

GtkWindow

(with CSD):
GtkWindow

GtkDrawingArea is NOT a GtkContainer, so gtk_container_add() on it fails. This means there is no standard GTK way to embed a third-party GtkWidget as a sibling of U++ Ctrl content.

As a workaround we are currently using a separate GTK_WINDOW_TOPLEVEL window with GDK_WINDOW_TYPE_HINT_UTILITY, positioned over the U++ Ctrl and synchronized via configure-event/focus-in/focus-out signals. It works but is not a true embedded solution.

THE PROPOSED PATCH

Insert a GtkOverlay between GtkWindow and GtkDrawingArea in GtkCreate.cpp. This is a minimal, non-invasive change:

1) Add field to Win struct in GtkCtrl.h:

```
struct Win : Moveable<Win> {
    GtkWidget   *gtk;
    GdkWindow   *gdk;
    GtkWidget   *drawing_area;

    Ptr<Ctrl>    ctrl;
    ...
};
```

2) In GtkCreate.cpp, replace the CSD branch:

```
// BEFORE:
if (top->header) {
```

```
        gtk_container_add(GTK_CONTAINER(top->window), top->drawing_area);
        gtk_widget_show_all(top->window);
    }

    // AFTER:
    if (top->header) {
        // Insert GtkOverlay between GtkWindow and GtkDrawingArea.
        // Allows embedding third-party GtkWidgets (e.g. WebKitWebView)
        // via gtk_overlay_add_overlay(). U++ GtkDrawingArea remains the
        // primary child and renders normally. Third-party widgets are
        // added as overlay children with explicit positioning via the
        // "get-child-position" signal.
        GtkWidget* overlay = gtk_overlay_new();
        gtk_container_add(GTK_CONTAINER(top->window), overlay);
        gtk_container_add(GTK_CONTAINER(overlay), top->drawing_area);
        w.overlay = overlay;
        gtk_widget_show_all(top->window);
    }
```

3) Initialize in Win setup:

```
    w.overlay = nullptr;
```

The non-CSD branch (plain X11, where drawing_area == window) would need a separate approach — see "What is missing" below.

WHY IT DID NOT FULLY WORK FOR US

We applied the patch and successfully found the GtkOverlay at runtime. We added WebKitWebView as an overlay child using gtk_overlay_add_overlay() and positioned it correctly using the "get-child-position" signal callback.

The browser was positioned correctly but was NOT VISIBLE because:

In GtkOverlay, overlay children are drawn ON TOP of the primary child. Our setup was:

GtkOverlay

However, U++ redraws the entire GtkDrawingArea surface on every frame with an opaque Cairo fill, covering the area where WebKit should show through. The GtkDrawingArea is fully opaque and paints over everything.

We attempted to invert the order (WebKit as primary, GtkDrawingArea as overlay) but then U++ Ctrl rendering was broken.

WHAT IS MISSING TO COMPLETE THE SOLUTION

Two things are needed from the U++ side:

1) IN THE CSD BRANCH — Cairo clip region:
   The GtkDraw callback (GtkDraw in GtkEvent.cpp or similar) needs to
   exclude the area occupied by overlay widgets from the Cairo clip region,
   so U++ does not paint over the embedded third-party widget.

   Something like:
    // Before U++ draws with Cairo, subtract overlay widget regions
    for each overlay_widget in overlay:
        cairo_rectangle(cr, ox, oy, ow, oh);
        cairo_clip(cr);   // U++ skips this area

   OR: set the GtkDrawingArea background to transparent in those regions.

2) IN THE NON-CSD BRANCH (plain X11):
   On X11, top->drawing_area == top->window (same widget). There is no
   separate GtkDrawingArea to wrap. A possible approach:
    - Create a real GtkDrawingArea separate from the GtkWindow
    - Wrap both in GtkOverlay as in the CSD case
   This is more invasive but would give true embedding on X11 as well.

CURRENT WORKAROUND (WORKING)

We use a GTK_WINDOW_TOPLEVEL window with GDK_WINDOW_TYPE_HINT_UTILITY:
 - Positioned over the U++ Ctrl using absolute screen coordinates
 - Synchronized with configure-event (move/resize)
 - Z-order managed with focus-in-event/focus-out-event:

 - Print dialog: popup hidden before WebKitPrintOperation, shown after

Note: GTK_WINDOW_POPUP does NOT work for z-order management because
popup windows bypass the window manager entirely and ignore
gdk_window_raise/lower. GTK_WINDOW_TOPLEVEL with UTILITY hint is
required for WM-managed z-order.

The workaround works correctly on both X11 and Wayland (XWayland).

We believe this is worth completing in U++ core, as it would enable a whole
class of native GTK widget embedding that is currently impossible.

  ==========
==========

File Attachments

1) WVupp.zip, downloaded 115 times