

Full Double Buffer 完整研究報告

2026-04-30 ~ 05-01 專案：HBSTK 21 輪研究 + 4 次實作

一、研究歷程

輪次	焦點	關鍵發現
v1	現況概覽	2 buffer、4 次螢幕操作、6 個待解問題
v2	深入剖析	10 個隱藏問題 (empty rcPaint、resize stale、UDA remnants)
v3	Data Flow	PaintText capture 在穩定狀態下是冗餘的
v4	DC 生命週期	s_hdcDblBuf 浪費 3MB，可跳過建立
v5	重入分析	PaintText/ScanAndDrawUda 不重入，boolean flag 足夠
v6	Caret 分析	ScanAndDrawUda 管理 caret 最安全
v7	Resize	全螢幕 dirty 保證 resize 後完整重繪
v8	★ v1/v2 失敗根因	推測 ShowCaret 循環 (後被 v19 推翻)
v9	理想架構	修正 caret 時序即可實現 (理論)
v10	總結	演進路線圖
v11	v5 可行性	跳過 capture 安全 (理論) + 早期返回 Bug
v12	逐幀模擬	5 場景全部安全 (理論)
v13	壓力測試	7 極端情境全部安全 (理論)
v14	實作規格	14 行修改
v15	v5 失敗分析	畫面切換時 buffer 殘留舊內容
v16	v5 深入	RestScreen + 選單重繪分兩次 WM_PAINT
v17	v6 架構	ShowCaret 用 XOR 不觸發 WM_PAINT
v18	v6 藍圖	需驗證 v1 是否循環
v19	v6 Phase1 結果	★ 非循環、bUseMemDC 判斷錯誤、畫面停留
v20	v4 = 最佳	v4 full 已達成一個 buffer、零複製
v21	★ 二合一瓶頸	PaintText 只渲染 dirty column + capture 成本趨近零

二、4 次實作嘗試

版本	架構	結果	失敗原因
v1/v2	WndProc 管新 buffer + 螢幕寫入	無法輸入	PaintText final BitBlt 仍寫 stale s_hdcPaintBuf (bUseMemDC 判斷錯誤)
v5	跳過 screen capture	畫面切換閃爍	buffer 殘留舊畫面，TextOut 只覆蓋 dirty column
v6	WndProc 管新 buffer + bUseMemDC 修正	畫面停留	dirty region 不完整，非 dirty column 保留初始 capture 內容

三、最終架構 (v4 full)

經 21 輪研究確認，v4 full 是安全可達的最佳架構：

指標	原始	v4 full
Buffer 數量	2 (s_hdcPaintBuf + s_hdcDblBuf)	1 (s_hdcPaintBuf, s_hdcDblBuf 不建立)
Buffer 間複製	1 次 (PaintBuf → DblBuf capture)	0 次
螢幕讀取	2 次 (PaintText + ScanAndDrawUda)	1 次 (PaintText capture)
螢幕寫入	2 次 (PaintText + ScanAndDrawUda)	1 次 (ScanAndDrawUda writeback)
總螢幕操作	4 次	2 次 (1R + 1W)
記憶體	~6MB	~3MB
中間狀態可見	是 (ASCII 先顯示、UDA 後疊加)	否 (一次 writeback)

四、★ 核心發現

4.1 capture 不能跳過的根本原因

PaintText 只渲染 ps.rcPaint 範圍的 column (rcRect.left ~ rcRect.right)，不是整行。沒有 capture，非渲染的 column 在 buffer 中保留舊內容。跳過 capture 後的 writeback 會顯示混合的新+舊內容。

4.2 capture 的效能成本趨近零

穩定狀態下 buffer = screen (上一幀 writeback 過)。capture = BitBlt(buffer ← screen) = 自己抄自己 = NOP。Windows DWM 可能做 copy-on-write 優化。即使實際複製，dirty region 通常很小 (< 4KB) → <

0.01ms。

4.3 v1/v2「無法輸入」的真正原因

不是 ShowCaret 循環 (paint count 正常)。是 PaintText 的 final BitBlt 判斷錯誤：bUseMemDC=2 (shared mode) 但 g_bSharedPaint=0 → 走原始路徑 → 寫 stale s_hdcPaintBuf 到螢幕。修正 bUseMemDC==2 判斷後可以輸入，但畫面不持續更新 (dirty region 問題)。

4.4 二合一的結論

收益 0.01ms/幀 (capture 成本趨近零)。風險 = 畫面切換閃爍 + 畫面停留。不值得。

五、修改檔案 (v4 full)

檔案	修改
usrfont_c.c	g_bSharedPaint + g_hPaintBuf 全域變數
usrfont_wndproc.c	WM_PAINT: g_bSharedPaint=1/0 (2 行)
gtwvt.c PaintText	g_bSharedPaint 時跳過 final BitBlt + 匯出 buffer + SetStretchBltMode
usrfont_scan.c	g_hPaintBuf 時直接畫 (跳過 capture/s_hdcDblBuf) + 早期返回 writeback

六、研究報告索引

21 份研究報告保存於 doc/DoubleBuffer_Research_v1.md ~ v21.md

規劃文件：doc/FullDoubleBuffer_Plan_20260430.pdf