TRUE Double Buffer + Caret Unification
GTWVT 雙緩衝重構與游標統一管理

日期	2026-05-05
專案	HBSTK / Stk32 (Harbour GTWVT Console)
檔案	gtwvt.c, usrfont_c.c, usrfont_scan.c, usrfont_wndproc.c

一、訴求與現象

GTWVT 的「偽 double buffer」(shared paint v3) 架構導致三個嚴重問題：

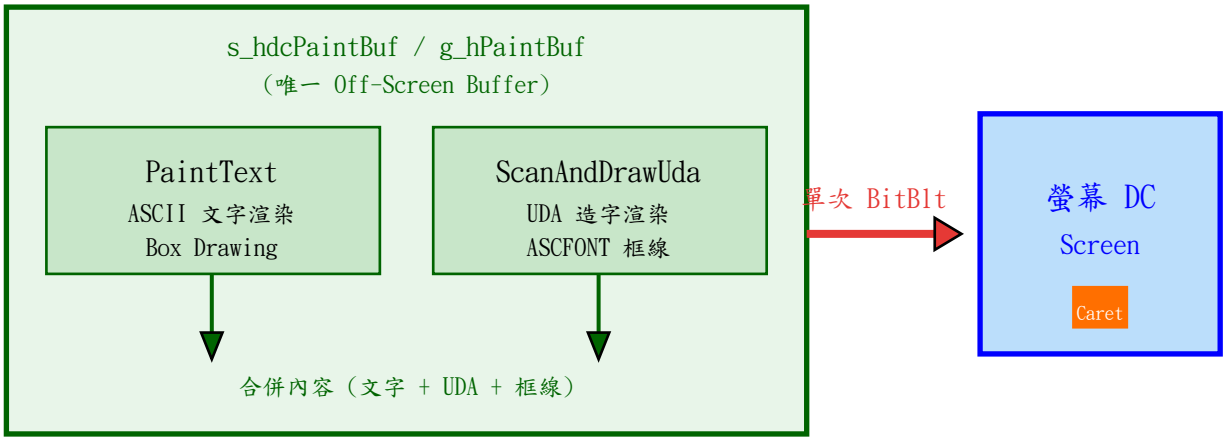
1. BUYSELL 單據修改時游標消失（實體鍵盤 ENTER 看不到游標）
2. winform() 游標半形　全形來回閃爍
3. 游標移動後消失約 1 秒才重新出現

二、根因分析

問題	根因	影響
游標消失	UpdateCaret 永遠用半形寬度建立 caret，ScanAndDrawUda 再用全形寬度重建，兩者搶控制權	BUYSELL 游標完全不見
游標閃爍	UpdateCaret(半形) → ScanAndDrawUda(全形) → UpdateCaret(半形) 循環	winform 游標閃爍不停
移動延遲	DestroyCaret/CreateCaret 每次 WM_PAINT 都重置 blink timer；InvalidateRect(NULL) 觸發全畫面重繪	游標移動後消失 ~1 秒
2個 Buffer	s_hdcPaintBuf (PaintText) + s_hdcDblBuf (ScanAndDrawUda) 各自管理	記憶體浪費、程式複雜

三、TRUE Double Buffer 架構

TRUE Double Buffer 架構圖



流程：

BeginPaint → HideCaret
PaintText 畫 ASCII 到 buffer
ScanAndDrawUda 畫 UDA 到同一 buffer
HideCaret → BitBlt → 2×ShowCaret

圖 1：單一 off-screen buffer 架構—所有渲染都在 buffer 上完成，單次 BitBlt 到螢幕

舊架構 vs 新架構對比

舊版（Shared Paint v3）

UpdateCaret: CreateCaret(半形)
BeginPaint: HideCaret
PaintText → s_hdcPaintBuf
EndPaint + ShowCaret ← 過早!
ScanAndDrawUda:
HideCaret
畫 UDA → s_hdcDblBuf ← 第2個 buffer
BitBlt s_hdcDblBuf→螢幕
DestroyCaret
CreateCaret(全形) ← 衝突!
ShowCaret

半形 全形閃爍
2個 buffer 浪費記憶體

新版（TRUE Double Buffer）

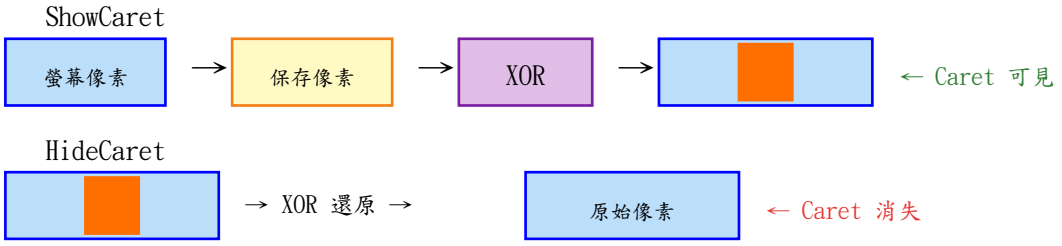
UpdateCaret: CreateCaret(正確寬度!)
BeginPaint: HideCaret
PaintText → g_hPaintBuf
EndPaint（跳過 ShowCaret）
ScanAndDrawUda:
畫 UDA → g_hPaintBuf ← 同一 buffer
HideCaret
BitBlt g_hPaintBuf→螢幕
2×ShowCaret

單一 caret 管理者
1個 buffer—無閃爍
Blink timer 不重置
游標即時反應

圖 2：舊版 vs 新版架構對比—消除雙 buffer、雙 caret 管理者的衝突

四、Caret XOR 機制與解決方案

Windows Caret XOR 機制



錯誤：BitBlt 沒有先 HideCaret

BitBlt 覆蓋螢幕像素 → Caret 被蓋掉 → Windows 不知道
→ 下次 blink HideCaret XOR 還原「舊像素」→ 畫面錯誤 → 需 2 個 blink cycle（~1秒）才修正

正確：HideCaret → BitBlt → 2×ShowCaret

HideCaret：先保存+移除 caret → BitBlt：寫入新內容 → ShowCaret：在新內容上畫 caret

圖 3：Windows Caret XOR 機制—BitBlt 前必須 HideCaret 保存像素

HideCaret/ShowCaret 計數平衡圖

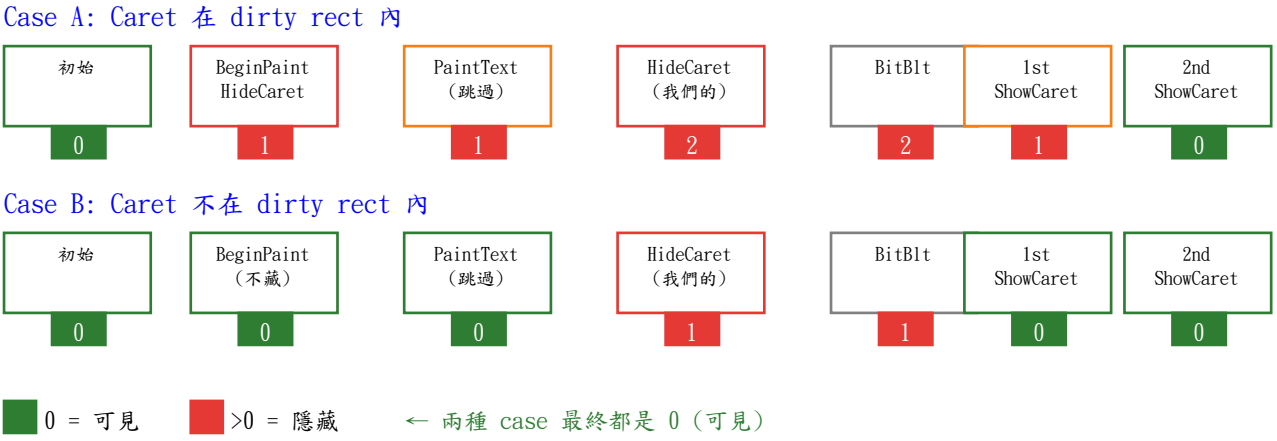


圖 4：HideCaret/ShowCaret 計數平衡—兩種情況最終都歸零

五、解決方法

修改項目	內容	效果
gtwvt.c UpdateCaret	加入 isWideChar() 判斷全形字元，CreateCaret 使用正確寬度 (1x 或 2x PTEXTSIZE.x)；處理 trail marker (0xFFFE) col-1 fallback；移除 InvalidateRect(NULL)	消除半形　全形閃爍
usrfont_scan.c ScanAndDrawUda	移除 DestroyCaret/CreateCaret/SetCaretPos，改用 HideCaret + BitBlt + 2×ShowCaret；移除 s_hdcDblBuf (第二 buffer)；移除重複 pre-scan；SetCaretPos 加 nOffX/nOffY (最大化修正)	Blink timer 不重置，游標即時反應
usrfont_c.c	g_bSharedPaint → g_bHookPaint；移除 s_hdcDblBuf/s_hBmpDblBuf/s_nDblBufW/s_nDblBufH	精簡代碼、減少記憶體
usrfont_wndproc.c	g_bSharedPaint → g_bHookPaint	語意更清楚
gtwvt.c PaintText	g_bHookPaint=1 時跳過 ShowCaret (讓 ScanAndDrawUda 統一管理)	避免 PaintText 過早 ShowCaret

六、踩坑與陷阱

坑	症狀	解法
只呼叫 ShowCaret	全部游標消失—BeginPaint 可能不 HideCaret，BitBlt 覆蓋 caret 像素但 Windows 不知道	BitBlt 前加 HideCaret，後加 2×ShowCaret
DestroyCaret/ CreateCaret	游標移動後消失 ~1 秒—每次 WM_PAINT 重置 blink timer	不用 DestroyCaret/CreateCaret，改用 HideCaret/ShowCaret
InvalidateRect (NULL)	全畫面重繪導致游標閃爍 + 性能問題	移除；INSERT 切換用 timer 9988 (50ms)
UpdateCaret 寬度固定	永遠用半形寬度，與 ScanAndDrawUda 的全形寬度衝突	加 isWideChar() 判斷，建立正確寬度

七、精簡成果

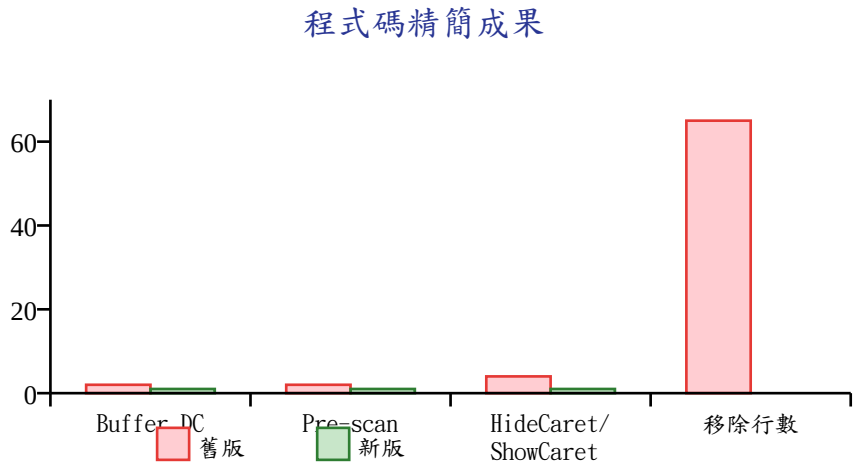


圖 5: 程式碼精簡對比—Buffer DC、1 pre-scan、輕量級 caret 管理

八、最終結果

測試項目	結果
BUYSELL 游標	正常顯示，實體 ENTER 可見
winform 游標	不再半形　全形閃爍
P32 游標	正常
游標移動速度	即時反應，無延遲
ASCFONT 框線	正常顯示
Blink 閃爍	正常 blink，timer 不被重置

九、GTWVT 精簡重構

TRUE Double Buffer 完成後，繼續精簡 gtwvt.c 的結構，讓未來加東西不再受限制。

階段	內容	效果
Phase 1	PaintText 5 個 static → pWVT 物件管理，New() 初始化、Free() 清理	函式內不再有隱藏狀態
Phase 2A/2B	5 處 { extern } compound statements 消除，統一到檔案頂端	4 層巢狀 → 1 層扁平
Phase 3A	WM_CHAR/WM_IME_CHAR ~60 行重複 → HandleWideChar()	修 bug 只要改一處
Phase 3B	WriteCon/WriteConW ~130 行重複 → WriteConChar() + 修 TAB bug	修 bug 只要改一處
Phase 4	移除 PutChar debug fopen 殘留	清除產品碼中的 debug I/O
Phase 5A	Info (876行) → InfoFont + InfoCustom + 主體	加自訂功能碼只需改 InfoCustom
Phase 5B	DefineBoxChar (900行) → RC + Std + dispatcher	兩大怪獸馴服為子函式群

表 6: GTWVT 精簡重構 7 個階段—每步獨立可測試、可回退

十、整天總結

指標	修改前	修改後
Buffer DC 數量	2 (s_hdcPaintBuf + s_hdcDblBuf)	1 (pWVT->hdcPaintBuf)
Caret 管理者	2 (UpdateCaret + ScanAndDrawUda)	1 (UpdateCaret 唯一建立者)
Compound statements	5 處 { extern/static }	0
重複程式碼	~190 行	0 (合併為 helper)
最大函式	DefineBoxChar 900行 / Info 876行	各拆為 3 個子函式
PaintText static	5 個隱藏在函式內	統一由 pWVT 物件管理
Debug 殘留	fopen debug_trail.log	已移除

表 7: 整天修改前後對比—GTWVT 從「補丁疊補丁」變成「穩定基礎」