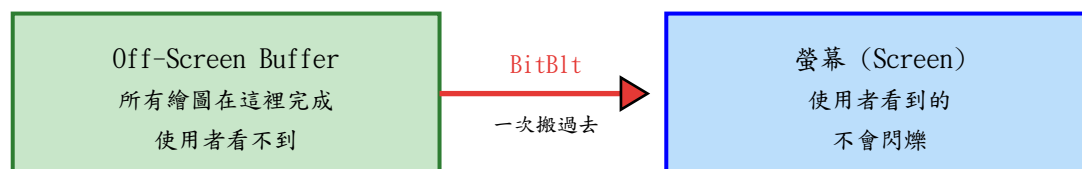


GTWVT Technical Glossary

GTWVT 技術名詞圖解

一、核心概念

Double Buffer (雙緩衝)



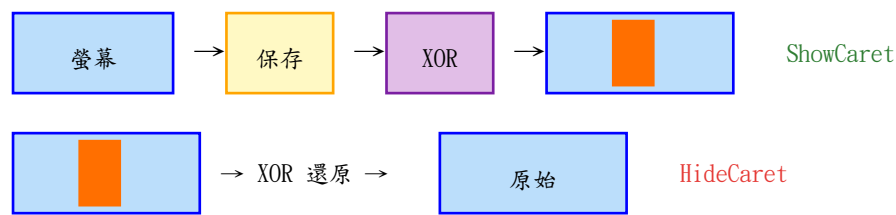
好處：畫到一半的圖不會出現在螢幕上

圖 1: Double Buffer — 所有繪圖先畫在 off-screen buffer，完成後一次 BitBlt 到螢幕

名詞	英文	說明
BitBlt	Bit Block Transfer	將一塊像素從一個 DC 複製到另一個 DC。雙緩衝的核心操作—buffer → 螢幕的最後一步
DC	Device Context	Windows GDI 繪圖介面。螢幕有自己的 DC，buffer 也有自己的 DC。所有繪圖操作都對 DC 執行
pWVT	PHB_GTWVT	GTWVT 的「物件」—包含視窗、字型、顏色、游標、paint buffer 等所有狀態。一個視窗一個 pWVT
WndProc	Window Procedure	Windows 視窗的訊息處理函式。每個 WM_* 訊息（按鍵、滑鼠、重繪等）都由它處理
Subclass	Window Subclassing	用 SetWindowLongPtr 替換視窗的 WndProc，放入自己的處理函式。usrfont 用這個技術放入 UDA/ASCFONT 渲染

二、游標 (Caret)

Caret (游標) XOR 機制



HideCaret/ShowCaret 用計數器管理：每個 Hide 需要一個 Show 對應
圖 2: Caret 用 XOR 畫在螢幕上，HideCaret 用 XOR 還原

名詞	說明
CreateCaret	建立新的 caret (隱藏狀態)。會銷毀舊的 caret。重置 blink timer
DestroyCaret	銷毀 caret。重置 blink timer。避免在每次 WM_PAINT 呼叫 (會導致游標延遲 ~1 秒)
ShowCaret	遞減隱藏計數器。計數=0 時 caret 可見。每個 HideCaret 需要一個 ShowCaret 對應
HideCaret	遞增隱藏計數器。計數>0 時 caret 不可見。可累加 (呼叫 3 次 Hide 就需 3 次 Show)
BeginPaint	開始 WM_PAINT 處理。若 caret 在 dirty rect 內，自動呼叫 HideCaret。EndPaint 不會自動 ShowCaret
2xShowCaret	我們的解法：BitBlt 前 HideCaret，BitBlt 後 2xShowCaret (平衡自己的 + BeginPaint 的)

三、CJK 字元處理

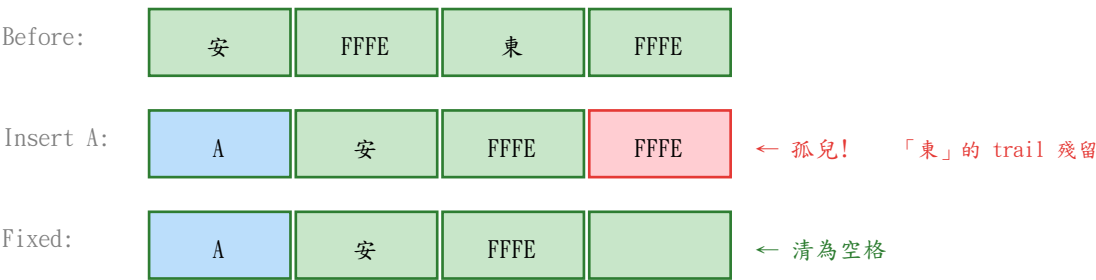
CJK Trail Marker (0xFFFF)



CJK 字佔 2 格: lead(字元) + trail(0xFFFF 佔位標記)

圖 3: CJK 字元在螢幕緩衝區的儲存方式

CJK 孤兒鏈 (Orphan Chain)



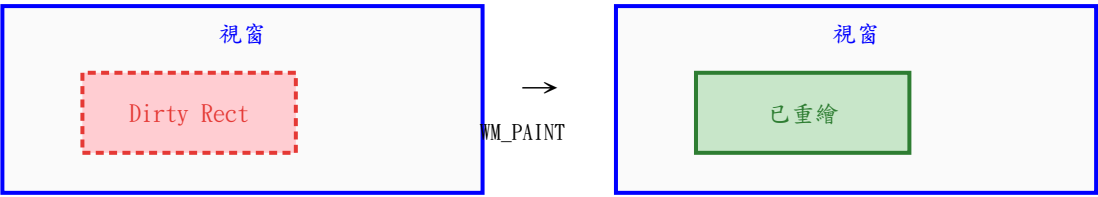
寫 trail 到 col+1 前，先檢查 col+1 是否有 CJK lead，有的話清 col+2

圖 4: Insert 時的連鏈孤兒問題與修正

名詞	說明
Trail Marker	0xFFFE。CJK 字元佔 2 格，第 1 格存字元 (lead)，第 2 格存 0xFFFE (trail)。渲染時 trail 寬度=0
孤兒 (Orphan)	單獨存在的 lead (沒有 trail) 或單獨的 trail (沒有 lead)。顯示為亂碼。PutChar 的清理邏輯負責防止
連鎖孤兒	Insert 時 CJK 字元右移，trail marker 覆蓋下一個 CJK lead，產生新的孤兒。修正：寫 trail 前先檢查 col+1
isWideChar	判斷 Unicode 字元是否全形 (2 cells)。涵蓋 CJK、Hangul、Kana、PUA (UDA E000-F8FF) 等
Redraw ±1	Redraw 無條件擴展左右各 1 cell。因為 PutChar 清理後，Redraw 看到的是新內容 (空格)，不知道舊內容是 CJK

四、程式結構

InvalidateRect / Dirty Rect (失效區域)



InvalidateRect 標記「需要重繪」的區域，Windows 在空閒時產生 WM_PAINT
圖 5: InvalidateRect 標記需要重繪的區域

名詞	說明
extern	宣告「這個變數定義在別的檔案」。用於跨檔案存取全域變數。應放在檔案頂端統一宣告，禁止用 compound statement
static (func)	函式內的 static 變數：值在呼叫間保留，但外面看不到。應搬進 pWVT 結構體統一管理
Compound Statement	{ } 包起來的程式區塊。C89 要求變數宣告在 { 開頭。禁止用於局部 extern/static 宣告（造成多層巢狀）
InvalidateRect	標記視窗中一塊區域為「需要重繪」。Windows 在空閒時產生 WM_PAINT。NULL = 全畫面（效能差，避免）
WM_PAINT	Windows 的重繪訊息。由 InvalidateRect 觸發，優先權最低（所有其他訊息處理完才執行）
Refresh	Harbour GT 的同步點。收集所有 dirty cells → 呼叫 Redraw → InvalidateRect。在 Inkey() 迴圈中執行
UDA / PUA	User Defined Area / Private Use Area。Unicode E000-F8FF。用於造字區（USRFONT）的自訂字元
ASCFONT	華康自訂框線字型。用控制字元 (< 0x20) 對應 Unicode 框線字元，由 ScanAndDrawUda 渲染

五、命名規則

前綴	範例	意義
s_	s_wvtWindows	檔案範圍 static 變數
g_	g_nCellW	usrfont 全域變數 (gtwvt 用 extern 存取)
pWVT->	pWVT->hdcPaintBuf	物件欄位（統一管理的狀態）
hb_gt_wvt_	hb_gt_wvt_PaintText	GTWVT 的 static 函式
HB_GTSELF_	HB_GTSELF_PUTCHAR	呼叫自己的 GT 虛擬函式（含 CJK patch）
HB_GTSUPER_	HB_GTSUPER_PUTCHAR	呼叫父層 GT 的函式（跳過 CJK patch）

六、UDA / PUA / CJK 與造字專章

這些名詞在系統中經常出現，理解它們的關係是維護 GTWVT 的基礎。

Unicode 字元區域分布



isWideChar 涵蓋: CJK(4E00-9FFF) + PUA(E000-F8FF) + Hangul + Kana + 全形符號

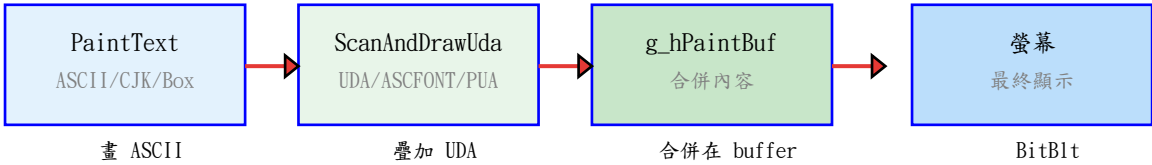
圖 6: Unicode 字元區域分布與 GTWVT 的關係

名詞	全名	範圍	說明
CJK	Chinese Japanese Korean	U+4E00-9FFF	中日韓統一漢字。全形 (2 cells)。GTWVT 原生支援，不需要 USRFONT
PUA	Private Use Area	U+E000-F8FF	Unicode 私人使用區。任何人都可以定義自己的字元。全形 (2 cells)
UDA	User Defined Area (造字區)	BIG5 FA40-FEFE → U+E000-E310	BIG5 的造字區，對應 PUA 的 E000-E310。USRFONT.24M 字型檔提供這些字的 bitmap
USRFONT	User Font (自訂字型)	.24M 檔	24x24 bitmap 字型檔，包含 UDA 造字的圖形。ScanAndDrawUda 用它來渲染 UDA 字元
ASCFONT	ASCII Font (華康框線)	U+0001-001F + U+007F	控制字元對應華康自訂框線字型。半形 (1 cell)。由 ASCFONT.UMAP 對應表控制
DynUnicode	Dynamic Unicode (動態分配)	U+E311-F8FF	IME 輸入的稀有字 (超出 BIG5 對照表)。動態分配 UDA slot，用 TrueType 渲染
ScanAndDrawUda	主渲染函式	usrfont_scan.c	掃描全畫面，找出 UDA/ASCFONT/PUA 字元，用對應的字型資料渲染到 buffer

表: UDA/PUA/CJK/造字相關名詞完整對照

七、字元渲染管線

字元渲染流程



PaintText 處理標準字元，ScanAndDrawUda 處理造字/框線，兩者畫在同一個 buffer
UpdateCaret 負責游標建立 (唯一管理者)，ScanAndDrawUda 只做 HideCaret + 2xShowCaret

圖 7: 字元渲染管線—PaintText + ScanAndDrawUda 協作流程