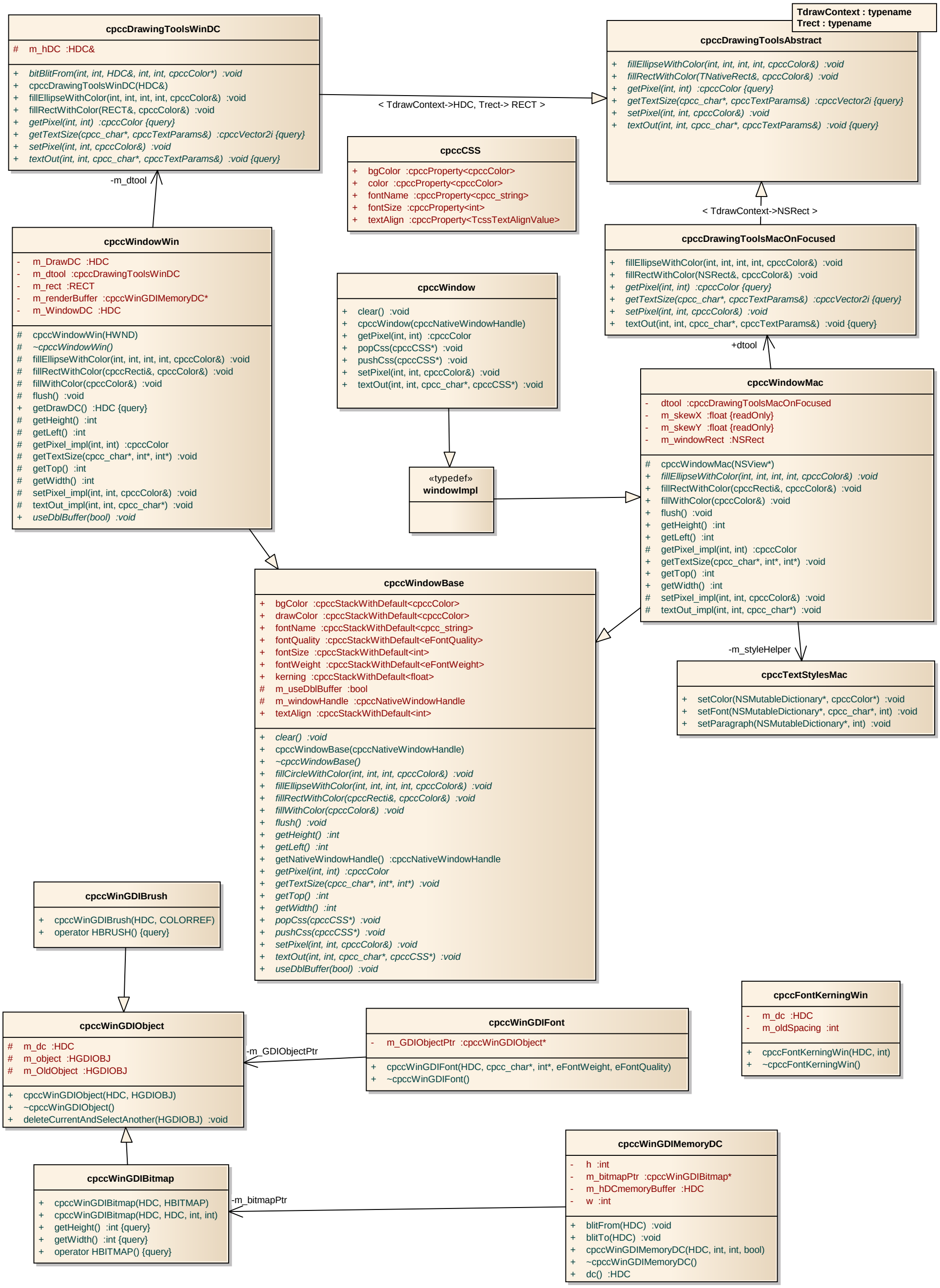
cpccDrawingToolsWinDC
m_hDC :HDC&
+ bitBlitFrom(int, int, HDC&, int, int, cpccColor*) :void
+ cpccDrawingToolsWinDC(HDC&)
+ fillEllipseWithColor(int, int, int, int, cpccColor&) :void
+ fillRectWithColor(RECT&, cpccColor&) :void
+ getPixel(int, int) :cpccColor {query}
+ getTextSize(cpcc_char*, cpccTextParams&) :cpccVector2i {query}
+ setPixel(int, int, cpccColor&) :void
+ textOut(int, int, cpcc_char*, cpccTextParams&) :void {query}
< TdrawContext->HDC, Trect-> RECT >
TdrawContext : typename
Trect : typename
cpccDrawingToolsAbstract
+ fillEllipseWithColor(int, int, int, int, cpccColor&) :void
+ fillRectWithColor(TNativeRect&, cpccColor&) :void
+ getPixel(int, int) :cpccColor {query}
+ getTextSize(cpcc_char*, cpccTextParams&) :cpccVector2i {query}
+ setPixel(int, int, cpccColor&) :void
+ textOut(int, int, cpcc_char*, cpccTextParams&) :void {query}
-m_dtool
cpccCSS
+ bgColor :cpccProperty<cpccColor>
+ color :cpccProperty<cpccColor>
+ fontName :cpccProperty<cpcc_string>
+ fontSize :cpccProperty<int>
+ textAlign :cpccProperty<TcssTextAlignValue>
< TdrawContext->NSRect >
cpccWindowWin
- m_DrawDC :HDC
- m_dtool :cpccDrawingToolsWinDC
- m_rect :RECT
- m_renderBuffer :cpccWinGDIMemoryDC*
- m_WindowDC :HDC
cpccWindowWin(HWND)
~cpccWindowWin()
fillEllipseWithColor(int, int, int, int, cpccColor&) :void
fillRectWithColor(cpccRecti&, cpccColor&) :void
fillWithColor(cpccColor&) :void
flush() :void
+ getDrawDC() :HDC {query}
# getHeight() :int
# getLeft() :int
# getPixel_impl(int, int) :cpccColor
# getTextSize(cpcc_char*, int*, int*) :void
# getTop() :int
# getWidth() :int
# setPixel_impl(int, int, cpccColor&) :void
# textOut_impl(int, int, cpcc_char*) :void
+ useDblBuffer(bool) :void
cpccWindow
+ clear() :void
+ cpccWindow(cpccNativeWindowHandle)
+ getPixel(int, int) :cpccColor
+ popCss(cpccCSS*) :void
+ pushCss(cpccCSS*) :void
+ setPixel(int, int, cpccColor&) :void
+ textOut(int, int, cpcc_char*, cpccCSS*) :void
cpccDrawingToolsMacOnFocused
+ fillEllipseWithColor(int, int, int, int, cpccColor&) :void
+ fillRectWithColor(NSRect&, cpccColor&) :void
+ getPixel(int, int) :cpccColor {query}
+ getTextSize(cpcc_char*, cpccTextParams&) :cpccVector2i {query}
+ setPixel(int, int, cpccColor&) :void
+ textOut(int, int, cpcc_char*, cpccTextParams&) :void {query}
+dtool
«typedef»
windowImpl
cpccWindowMac
- dtool :cpccDrawingToolsMacOnFocused
- m_skewX :float {readOnly}
- m_skewY :float {readOnly}
- m_windowRect :NSRect
cpccWindowMac(NSView*)
+ fillEllipseWithColor(int, int, int, int, cpccColor&) :void
+ fillRectWithColor(cpccRecti&, cpccColor&) :void
+ fillWithColor(cpccColor&) :void
+ flush() :void
+ getHeight() :int
+ getLeft() :int
# getPixel_impl(int, int) :cpccColor
+ getTextSize(cpcc_char*, int*, int*) :void
+ getTop() :int
+ getWidth() :int
# setPixel_impl(int, int, cpccColor&) :void
# textOut_impl(int, int, cpcc_char*) :void
cpccWindowBase
+ bgColor :cpccStackWithDefault<cpccColor>
+ drawColor :cpccStackWithDefault<cpccColor>
+ fontName :cpccStackWithDefault<cpcc_string>
+ fontQuality :cpccStackWithDefault<eFontQuality>
+ fontSize :cpccStackWithDefault<int>
+ fontWeight :cpccStackWithDefault<eFontWeight>
+ kerning :cpccStackWithDefault<float>
# m_useDblBuffer :bool
# m_windowHandle :cpccNativeWindowHandle
+ textAlign :cpccStackWithDefault<int>
+ clear() :void
+ cpccWindowBase(cpccNativeWindowHandle)
+ ~cpccWindowBase()
+ fillCircleWithColor(int, int, int, cpccColor&) :void
+ fillEllipseWithColor(int, int, int, int, cpccColor&) :void
+ fillRectWithColor(cpccRecti&, cpccColor&) :void
+ fillWithColor(cpccColor&) :void
+ flush() :void
+ getHeight() :int
+ getLeft() :int
+ getNativeWindowHandle() :cpccNativeWindowHandle
+ getPixel(int, int) :cpccColor
+ getTextSize(cpcc_char*, int*, int*) :void
+ getTop() :int
+ getWidth() :int
+ popCss(cpccCSS*) :void
+ pushCss(cpccCSS*) :void
+ setPixel(int, int, cpccColor&) :void
+ textOut(int, int, cpcc_char*, cpccCSS*) :void
+ useDblBuffer(bool) :void
-m_styleHelper
cpccTextStylesMac
+ setColor(NSMutableDictionary*, cpccColor*) :void
+ setFont(NSMutableDictionary*, cpcc_char*, int) :void
+ setParagraph(NSMutableDictionary*, int) :void
cpccWinGDIBrush
+ cpccWinGDIBrush(HDC, COLORREF)
+ operator HBRUSH() {query}
cpccFontKerningWin
- m_dc :HDC
- m_oldSpacing :int
+ cpccFontKerningWin(HDC, int)
+ ~cpccFontKerningWin()
cpccWinGDIObject
m_dc :HDC
m_object :HGDIOBJ
m_OldObject :HGDIOBJ
+ cpccWinGDIObject(HDC, HGDIOBJ)
+ ~cpccWinGDIObject()
+ deleteCurrentAndSelectAnother(HGDIOBJ) :void
-m_GDIObjectPtr
cpccWinGDIFont
- m_GDIObjectPtr :cpccWinGDIObject*
+ cpccWinGDIFont(HDC, cpcc_char*, int*, eFontWeight, eFontQuality)
+ ~cpccWinGDIFont()
cpccWinGDIBitmap
+ cpccWinGDIBitmap(HDC, HBITMAP)
+ cpccWinGDIBitmap(HDC, HDC, int, int)
+ getHeight() :int {query}
+ getWidth() :int {query}
+ operator HBITMAP() {query}
-m_bitmapPtr
cpccWinGDIMemoryDC
- h :int
- m_bitmapPtr :cpccWinGDIBitmap*
- m_hDCmemoryBuffer :HDC
- w :int
+ blitFrom(HDC) :void
+ blitTo(HDC) :void
+ cpccWinGDIMemoryDC(HDC, int, int, bool)
+ ~cpccWinGDIMemoryDC()
+ dc() :HDC