石油暨石化產業科技學術合作

八十九年度期末報告完整版

中油公司油品運銷供應鏈管理系統設計

計畫編號：NSC 89-CPC-7-009-003

執行期間：89 年 10 月 1 日至 90 年 9 月 30 日

委託單位：中國石油股份有限公司

計畫主持人：黎漢林 教授

執行單位：國立交通大學資訊管理研究所

中　華　民　國　九　十　年　七　月

目錄

一、緒論

1.1 研究動機

近年來國營事業紛紛民營化，油品的供應也由原本中油的一家獨佔慢慢出現競爭。本研究計畫擬在網路上發展一中油公司油品運銷供應鏈管理系統，使石油油品儲運成本降低、效率提高。此套系統除了可以管理石油公司營運相關的土地、道路、加油站、儲運糟及油罐車路線等相關資訊，尚可結合資料庫及數學模式庫以作儲運管理之最佳化運算。可應用的範圍除了一般的文字功能查詢外，尚可延伸到有關地理資訊的查詢，成為一套完整的營運管理資訊系統並進而引入環球供應鏈(Global Supply Chain)的概念發展石油儲運最佳化的管理模式。

運輸成本為產品總成本的一部份，根據先進國家之統計資料顯示，工商產品的運輸成本約佔產品價值的 20% ~ 30%，其中以中下游之配銷成本所佔比例最高。中油公司已民營化，為落實企業化、進而國際化，降低內部營運成本與提高績效、引進資訊科技輔助經營管理確有其必要性。

由於網路通訊技術的進步，使得企業資源規劃（Entireties Resource Planning ，ERP）系統由單一工廠演進為可以藉由網路整合上、中、下游的大型整合系統。這種包含了廠際間原料採購、生產、

運輸、銷售等功能的系統稱為供應鏈管理系統（Supply Chain Management System，SCM）。

供應鏈中的主要環節可分為原料的供應商（Supplier / Vendor），製造產品的工廠（Facility / Plant）、倉儲中心（Distribution Center / Warehouse）及顧客（Customer）。彼此間以各種交通工具運送物品並透過網路傳遞資訊與金錢。形成環鏈，隨著交通工具及資訊科技的進步，供應鏈早已從小區域、單一國家演變為全球性的規模。因此，全球供應鏈（Global Supply Chain）是由位於不同國家的原料提供者、製造廠商、倉儲中心與顧客所交織而成。

由於生產及資訊網路的快速進步，造成了產品生命週期變短。由於市場的變化迅速、全球性競爭劇烈，企業為尋求生存的利基，需迅速回應消費者的需求，架構供應鏈系統的目的就在使企業間資訊與物品的快速通暢，以提升競爭力。

1.2 研究目的

在國營事業民營化的企求下，中油公司需加強公司的多角化經營與服務水準。本研究目的即為在網際網路上設計一中油公司油品運銷供應鏈管理系統，以輔助經營管理、提高績效、降低內部營運成本並降低油品儲存與運送成本。此系統的功能為：

1.在網頁上以地圖顯示中油現有運輸網如油庫、加油站位置、與運輸路線（包括油罐火車、油罐汽車、 油管、油輪等路線）以使配送作業電腦化，提昇效能。

2.運用最佳化方法 (Optimization)及總體供應鏈 (global supply chain)技術，發展中油公司油品運銷供應鏈管理系統，如現有油品儲運方式之檢討、規劃油庫與加油站間油罐車之最佳配送路線及新增加油站位址規劃評估等。

3.透過網際網路及瀏覽器將中油公司上中下游管理單位資訊作遠程分散式更新處理及保密管理，使資訊即時流通並快速反應在決策上。

二、文獻回顧

目前最常用的供應鏈管理軟體為 i2 與 Paragon。其功能價格與優缺點如下表所示，由於這兩套軟體的缺點甚多，本計畫擬重新設計一中油公司油品運銷供應鏈管理系統。

供應鏈管理(SCM)軟體回顧列表

	i2	Paragon
代理公司	前進國際、華夏科技	源訊、華茂及前進國際
功能	1. 提供 master planning,能同步檢視供應鏈中各個成份是否達到目標 2. 製造規劃能管理單一或多個工廠 3. 能對各種資源作最有效率的排程 4. 需求規劃能根據企業中各種可能的策略,作需求預測及訂單約定管理	1. 提供全球即時 ATP 及 CTP 資訊,包括日期、訂單狀況 2. 以網頁形式管理同步採購、供應商通訊及庫存等 3. 物料及產能規劃 4. 即時事件監測
優點	1. 擴充性及高度客製化,能適應各種產業的需求,並迅速地擴充到整個企業 2. 只需建立單一的規劃模型,就可以同步管理整個供應鏈 3. 採用主從架構,充份運用電腦記憶體,以節省運算時間	1. 提供生產計劃、排程管理 2. 可針對各工廠、發貨中心及倉庫間之產能/物料興現場排程提供快速反應機制 3. 可將生產計劃、排程等資訊整合於單一系統環境下,以快速準確地服務客戶,完成達交
價格	共 10 個模組,每個模組 30 萬美金	10~20 萬美金
缺點	1. 主要用於處理廠內流程，無法處理廠際間流程 2. 資料格式固定,應用程式間資料不易分享 3. 缺少 GIS 介面 4. 使用者無法自己調整最佳化模式 5. 價格過高	

供應鏈整合的概念並非始於現代，古代的生產管理過程中。如漢代鹽、鐵的專賣專銷，就已經有供應鏈整合的具體行為。而進入工業

時代之後，講求專業分工與大規模的製造，加上供應鏈中的每個單位只追求自己的最大利潤，產生了許多問題，其中一個最重要而且被廣泛討論的現象就是「長鞭效應」。在這種效應之下，供應鏈中的每個角色為了避免不確定性，往往得背負大量的存貨來因應。產業的上、中、下游如何合作，使得供應鏈能有效率的運作及整合，降低時間、人力與作業成本，提昇整體產業與個體企業的競爭力是目前管理學界的研究重點。

許多篇文獻對過去提出的供應鏈模式做了不同角度的回顧。

1960 年時，Clark 及 Scarf 提出了在多階層架構下，獲得最佳倉儲策略的模式。雖然它並不是完整的供應鏈模式，但目前供應鏈中多階層的概念，實肇始於本模式。

1974 年時，Geoffrion 及 Graves 利用混合整數規劃法建構一多產品的配銷模式。在此模式中，最主要由工廠、倉儲中心與顧客所組成，工廠會生產多樣的產品，並將這些產品運至倉儲中心；接著，每個顧客的產品需求由一個倉儲中心所提供。在此模式中，利用 Benders' decomposition 求一最佳解，在此最佳解下，會決定那個倉儲中心是否要繼續存在。從這篇文獻中，我們可以知道，整個「鏈」的架構已儼然成形。而 Benders' decomposition 也被廣泛的使用在這類的問題上。

接下來的 80 年代中，整合採購模式、生產模式、運輸模式及倉儲模式之中兩三種模式的文獻相當多。在 Thomas 及 Griffin 的整理中，就列舉了許多整合不同角色間的文獻，並將它們分成：採購者與供應者整合，生產與配銷整合，存貨與配銷整合三種。在 Bhatnager et al.中也用了類似的分類。

Williams 在 1981 年時提出了七種啟發式的演算法用來解決供應鏈中生產排程與配銷過程的問題。目標式為在滿足產品需求的條件下，求解平均存貨成本與固定成本（包含訂購、運輸及設定成本）總和的最小。

Cohen 及 Lee 在 1985 年時提出了兩個模式，其中一個是由原料採購的供應商開始，經過工廠、倉儲中心，最後到達顧客。這個模式類似 1974 年，Geoffrion 及 Graves 所提出的模式，但很顯然的，整個「鏈」的架構又多了原料採購一層，模式則是使用啟發式的演算法求解。另一個則是專注在非線性的生產規模經濟上。

Cohen 及 Lee 在 1988 年時，使用隨機過程方法，提出了另一個整合的模式，在這個模式中，又分為：(1) 原料控制、(2) 生產、(3) 存貨、(4) 配銷四個子模式。每個子模式在給定的需求條件下，可以對成本做最佳化的動作。此外，每個子模式會對其下游的子模式造成影響。對整個模式而言，是一個非線性的問題，處理上幾近不可能。

此外，這個模式最主要是想得到一個長期的運作策略，而非短期的策略。

接著，Cohen 及 Lee 在 1989 年時又提出了一個整數規劃的模式。這個模式最主要討論：在一個全球生產及配銷的體系之下，資源應該如何配置。作者描述了在供應鏈不同階層中，所採行的資源配置策略。此模式的目標是求出稅後純益的最大值，但因為 0-1 變數太多，因此求解也相當耗時。

Cohen 及 Moon 延續 Cohen 及 Lee 一個名為 PILOT 的模式，在 1990 年時發展另一模式。此模式同樣包含供應商、工廠、倉儲中心及顧客並且可以決定(1)那些工廠與倉儲中心必須設立或取消，(2)原物料採購的數量，(3) 那個工廠要生產多少產品，(4) 要由工廠經過倉儲中心最後運到顧客的產品數量。目標式為求解一最小成本，限制式則包含了原料的需求、供給、產能等。在這篇文獻中，作者指出在不同的情況下有許多因素都會影響到供應鏈的成本，其中運輸成本在整個供應鏈的成本中尤其扮演重要的角色。

Towill 在 1991 年時利用模擬的方法來評估在需求變動的情況下，應該對整個供應鏈採取什麼樣的策略。採用的策略有：(1) 儘量消除配銷的階層數，(2) 在整個供應鏈中導入資訊技術，以整合供應鏈中的資訊流。(3) 實現 just-in-time (JIT) 的倉儲策略來減少時間

上的延誤。(4) 改善每次的定購量或定購程序。整個模擬模式的目標在找出那一種策略最能在需求變動的情況下使需求的變異最小。根據模擬的結果，以（3）實現 just-in-time（JIT）的倉儲策略及（1）儘量消除配銷的階層數這兩種策略最有效。

Pyke 及 Cohen 在 1993 年時，使用隨機過程方法，建構一個三層架構的供應鏈模式。這個模式包含了一項產品、一個工廠、一個倉儲中心及一個零售商。此模式的目標式是求總成本的最小值，限制則包含了服務水準、安裝時間、處理時間及補貨前置時間。

Pyke 及 Cohen 在 1994 年時繼續改進就有的方法，將其擴展為多產品的模式。

Arntzen et al.在 1995 年時提出一篇利用混合整數規劃法建構的全球供應鏈模式。在此模式中，目標為求解生產成本，運輸成本，存貨成本等總和的最小值加上活動時間的加權值。只要輸入物料需求單、產品需求量、各項成本及稅制，最後就可以得到解答。此模式相當強調稅制在全球環境下的影響，重新出口的沖退稅，及不同國家地區的稅制都列入考量。另外，此模式為多產品、多階層與多個時期。

三、研究方法與系統架構

本研究之進行可分四個階段：網際地理資訊系統設計、油罐車巡行監控系統設計、供應鏈模式設計、系統介面整合設計。

3.1 網際地理資訊系統設計

（1）資料選取:

本系統將一個瀏覽器(Browser)視窗分格割為三個框架，分割之HTML 檔。地圖欄故名思義是用來放地圖資訊的，配合使用者的點選區域而逐步放大圖幅,如北區->新竹->竹北；主題欄則是存放地圖的主題資訊，如油庫、加油站類、餐飲類等主題；而輔助欄則顯示一些輔助資訊，如目前地圖欄圖幅的縮圖，或是查詢的結果。

為了讓使用者更明確的掌握資訊，本系統在放大地圖的某區域後，都會提供上一層的縮小圖，並且能放大圖幅，同時又能顯示上一層的地圖。

在空間資料分析方面，我們提供了點落面(如列出區域範圍內的加油站)的查詢，這也是我們和其它系統不同的地方。我們跳脫了傳統詢只能以滑鼠點選(Click)的方式，而讓使用者能直接在地圖上拉框，查詢區域範圍內的相關資料。當使用者選好範圍後(Mouse Up 事件發生)，就去比較是否有圖徵落在範圍內，有的話再顯示出來。由

於每一項類別(飯店、學校、觀光點等)的資料都儲存於其相關檔案中，若欲疊合圖層，只須將若干檔組合起來即可完成疊圖的工作。

（2）關連查詢

網際地理資訊系統的資料有一個很重要的特性就是非常繁雜，如何提供使用者一個最具人性化、合理化的查詢就很重要。

透過關連式的查詢，使用過只需要各種分類資訊中選取所需的部份，以查詢某條道路的加油站名稱等細項資料，則只需選加油站，其於道路處填入路名，則系統便經由資料庫查詢語法(SQL)和 CGI 回應該道路所有加油站，再經鏈結的功能由使用者選取所欲查詢的項目，最後系統則出現所有關於該加油站的詳細資料，不僅大大的提升系統的應用層面，更可以改善其查詢界面，讓系統更具親和力，進而提高使用者意願，有效發揮網際地理資訊系統的優點。

（3）搜尋過程與數學模式的結合

一般使用者通常會有一種需求就是希望給予二個位置，而系統會提供該二個位置的最佳通行方法，而一般的 GIS 對於這方面的技術則都比較局部性的，一方面資料量太大，而且資料更新不易，對傳統 GIS 而言資料的即時性是很難去維持的。

就開發中油公司油品運銷供應鏈管理系統而言，因為其架設於網

路上，再加上資料庫的分散性，大部份資料的維護由各相關單位提供。而系統透過使用者的查詢指定，再經由CGI的傳輸加上運用數學模式的分析，如系統模擬法、數學解析法、啟發演算法，來得到一最有效的方式再透過系統回應給使用者，如顯示中油內部油品配送問題，並且其方案可以有多種選擇，提供使用者一較大彈性的決策空間。

（4）網頁設計與數學模式

在網際資訊系統中，為使使用者（user）在系統中查詢時，能夠有較高的滿意度，因此必須要考慮現存傳統式網頁的缺點，這些缺點為大家所熟知的包括：

1、網路速度太慢

2、查詢不方便

3、顧客端（Client）的輸入時間（load time）過長

由上可以看出在網路上做查詢，最大的癥結在於時間；亦即若是查詢時間能夠快速，則使用者滿意度就較高；反之，滿意度就較低了。滿意度高，則工作效率就提高，所以解決上述缺點是一個相當棘手的問題。為解決這些缺點，因此希望在硬體（如伺服器（server）、網路…）與網頁設計間取得一個較佳的設計。如此之設計，可以按照使用者所查詢的頻率調整出一個較佳的網頁設計。因此在做規劃前，我們可先假設每個網頁的查詢頻率，根據此查詢頻率與每個網頁檔案間

的關係，試著去模擬出一個數學模式，此數學模式能夠解決上述的問題外，也能讓設計者知道如何去調整網頁檔案的結構化設計。

然而，我們所要的目標式就是將相關的成本降低；此相關成本包括網路傳輸的成本、伺服器端的存取成本…等。在其目標式中要考慮的限制條件包括了網頁結構的查詢層數、每個查詢層數的檔案數、檔案的連結關係…等等。

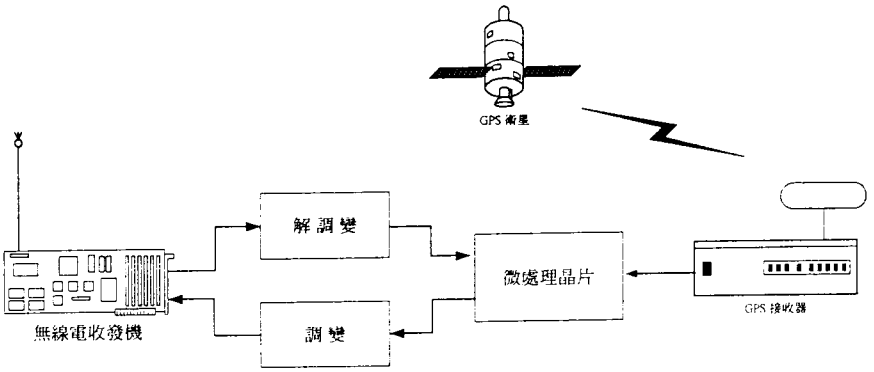
所以在網頁設計前，若能將一個最有效率的網頁設計放在網路上供人查詢，不僅可以節省使用者的查詢時間，也能使網頁的設計有一個較大的彈性調整。

3.2 油罐車巡行監控系統設計

車輛定位與監控功能乃結合網際地理資訊系統、路線管理系統與經由系統通訊路網所傳回的車輛行車資訊所整合而成，說明如下：

（1）車輛定位

本系統可提供目前在地區內所有油罐車之分佈位置及工作概況，使中油公司能確實掌握油罐車目前之分佈及作業狀況。



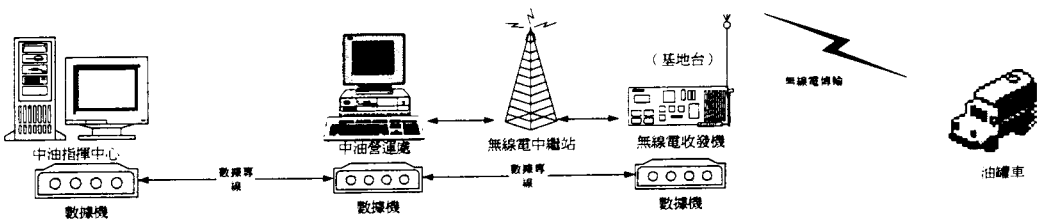
車輛定位系統架構圖

（2）行車與勤務監控

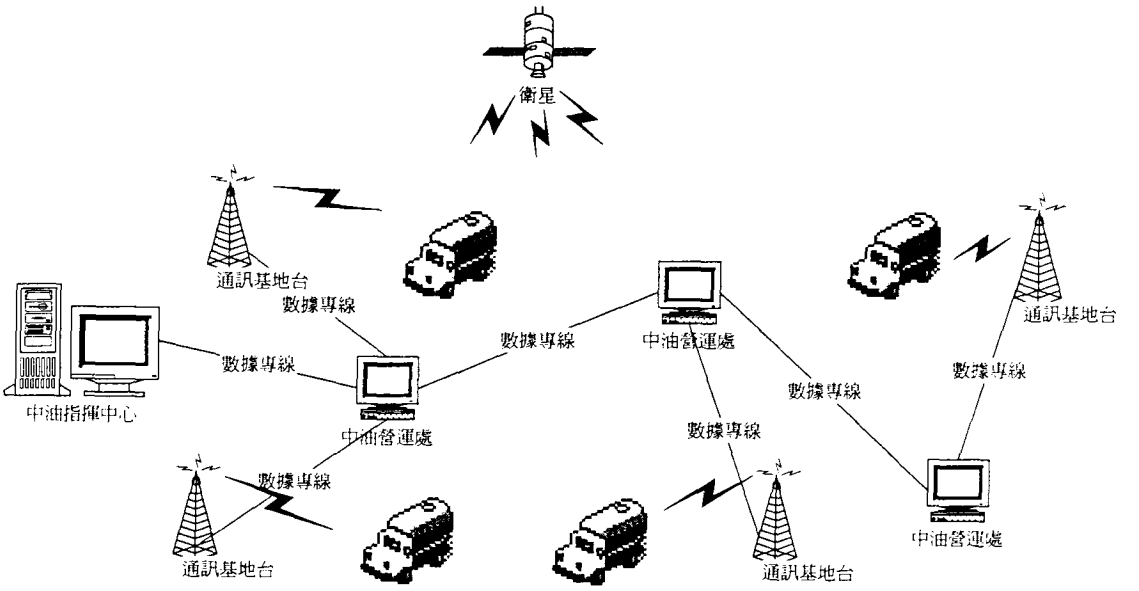
為油罐車之行車路徑、速度、執行任務等相關資訊之即時監控，並結合巡行路線管理系統，查核目前車輛是否依照原先預定之巡行路線執行油品補給任務。司機可透過車上設備的操作，即可將目前作業狀況傳回中油營運處或指揮中心。

（3）油品配送之調度

結合油庫及加油站之存油資料庫，配合油罐車出勤狀況，擬定油品配送最佳化模式，以有效調度油罐車因應加油站需求。



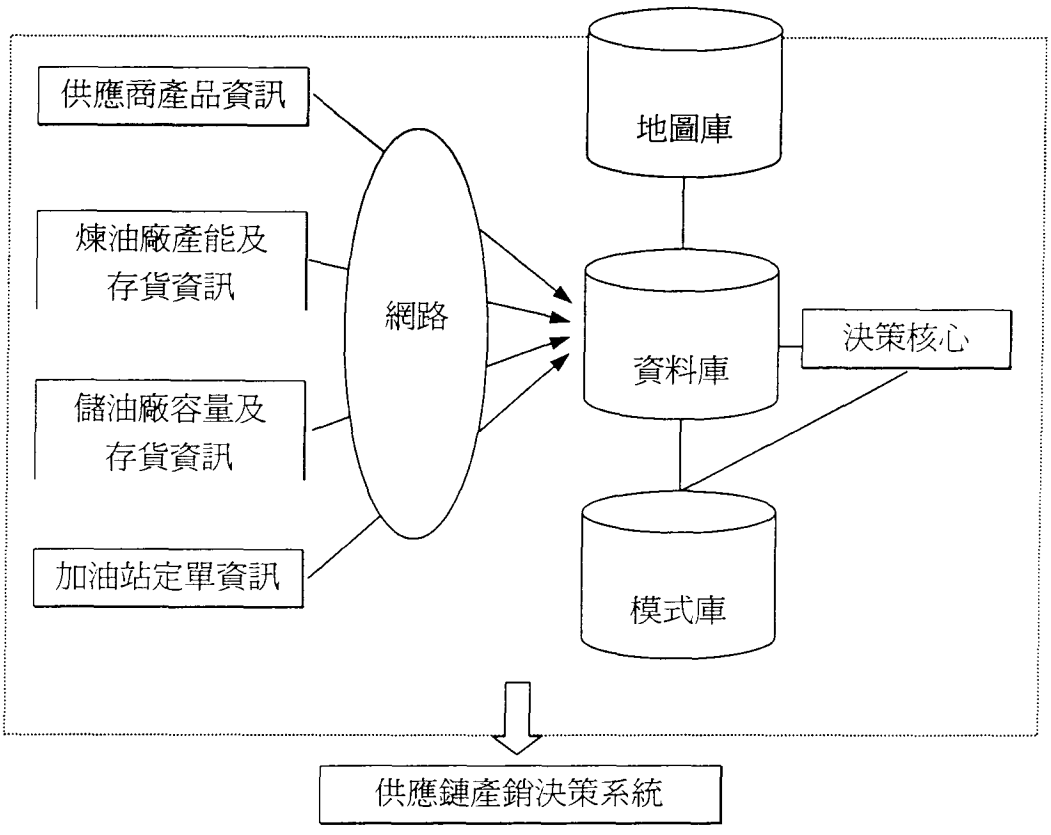
通訊系統架構圖



系統架構圖

3.3 供應鏈模式設計

供應鏈產銷決策系統 = Mathematics Model + Networks + Database + GIS，架構可以下圖表示：



在這個架構之下，共需要五個程式。

1、供應商需要一個程式：這個程式透過網路，使用 TCP/IP 協定，可以將油品供應商的資料寫入到遠端的資料庫，這些資料包含了在每一段時間中，每一種油品的價格及可提供的最大量。

2、煉油廠程式：這個程式要提供遠端資料庫的資料是：每一段時間中煉油廠的每種完成品產能，完成品初始存貨量及原料初始存貨量。

3、儲油廠程式：提供每一段時間中儲油中心對每種完成品的最大容量。

4、加油站程式：類似定單的資料，要告知需要什麼油品，什麼時候要，又需求量為多少就可以。

以上四個程式，性質類似，所用的技術也相同，都是對遠端的資料庫做更新的動作，也是在整個系統中扮演資料提供的角色。

5、管理決策程式：功能有二，一為一般性的資料查詢，另一為決策模擬。

本系統模式建立的數學模式係以 LINDO (1997)為基礎發展的。其特性為：

（1）可與地圖結合，使資料的輸入與輸出均可以地圖或示意圖形式顯示，操作簡便易懂。

（2）可建立模式庫以利應用模式間的串連。

（3）可與由網站上下載的資料相結合後運算。

目前油品供應鏈模式的研究有兩項弱點 ：

（1）建構的數學模式多屬線性（ linear ）。但由於真實世界的成本關係與邏輯關係多屬非線性與整數關係，因此這些模式未盡得反映實際狀況。

（2）建構的資料多未具空間屬性（ spatial attribute ）。但

由於原料、廠商與銷售站的分佈及運送路線均有空間性，因此使用者不易直接看出資料與現實世界的應對關係。

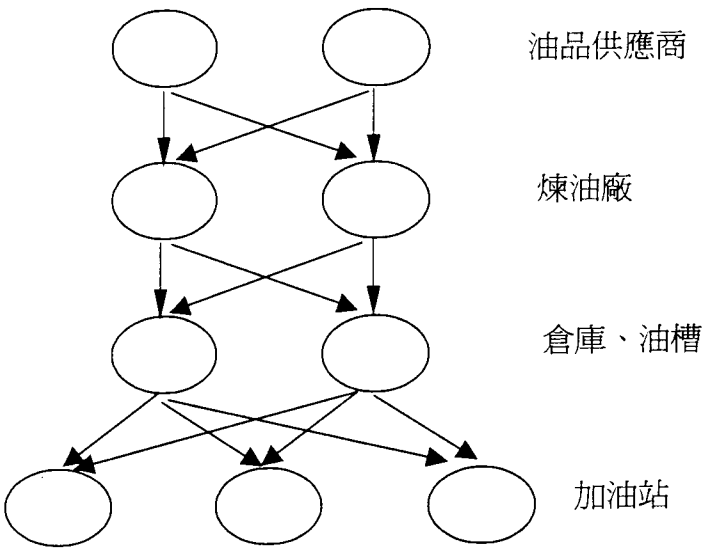
針對以上缺失，本計畫擬發展一供應鏈下的產銷模式，此模式的特點為：

（1）結合實際狀況將各種成本函數與邏輯關係以非線性與整數方式表現。這些數學形式會再轉換為線性整數後求解之。

（2）資料將以地圖與時段方式呈現，並建構在網際網路 Internet 上。此部分將基於目前研發成功的 Internet-GIS 進行之。

（3）所設計之模式將給中油公司提供資料測試其可行性。

在本模式中，目標是求出整個供應鏈的最小成本。為了要求出最小成本，我們首先得定出整個供應鏈的架構，接著才分析在此架構下成本有那些。在本模式中，供應鏈的架構如下:



整個供應鏈中的最上游是油品供應商，它負責提供原料，這些原料經過工廠的加工變成完成品。加工的過程是採取典型 MRP 的例子，也就是完成品是由某些種零件所組成。當工廠生產出完成品之後，它可以把完成品及未使用的原料放在工廠的倉庫中，或是工廠把完成品送至各地的倉儲點，倉儲點最後再把完成品送至顧客。經過這一連串的過程，我們所得到的成本如下

（1）煉油廠運送到倉儲點的運輸成本

（2）倉儲點運送到加油站的運輸成本

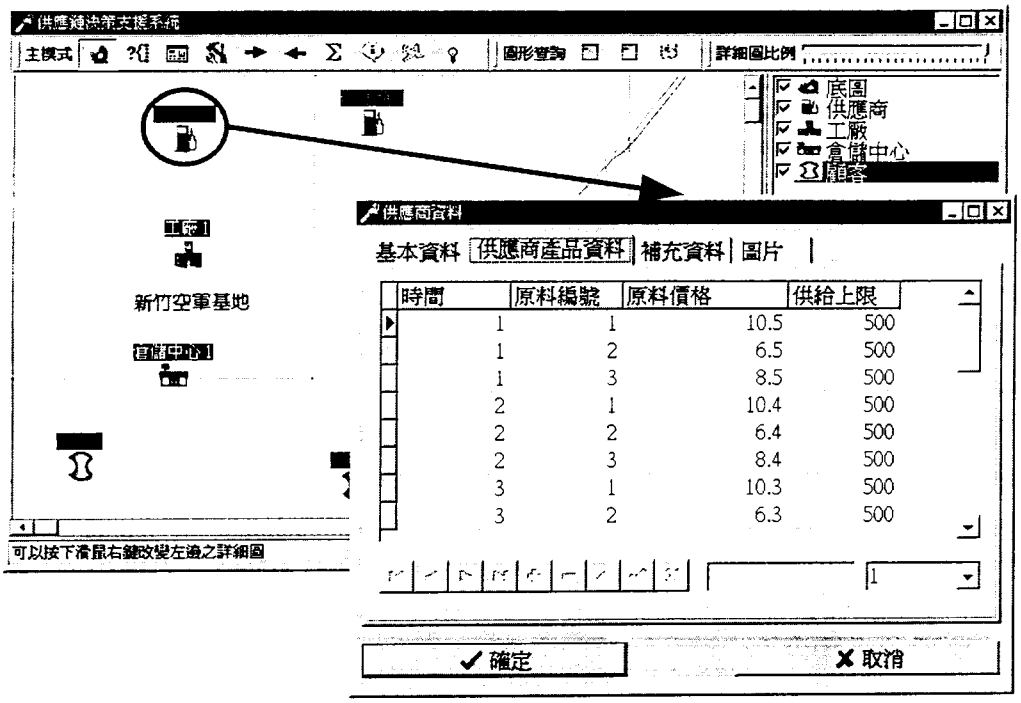
（3）煉油廠原料存貨成本

（4）煉油廠完成品存貨成本

（5）倉儲點完成品存貨成本

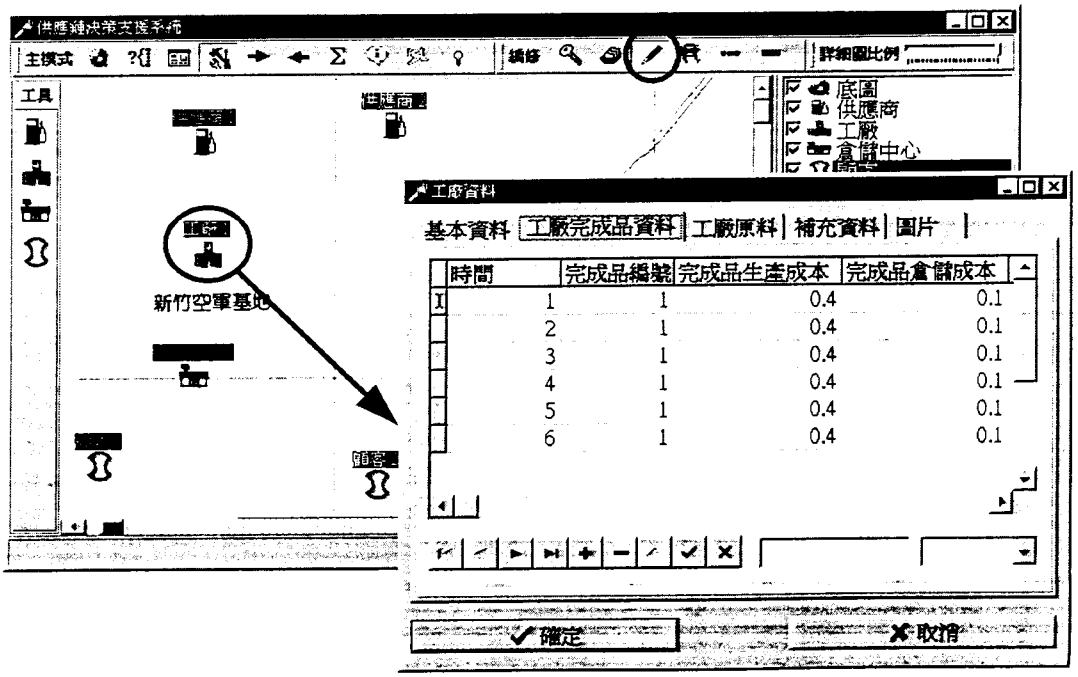
3.4 系統介面整合設計

（1）查詢供應鏈中某個角色的資料



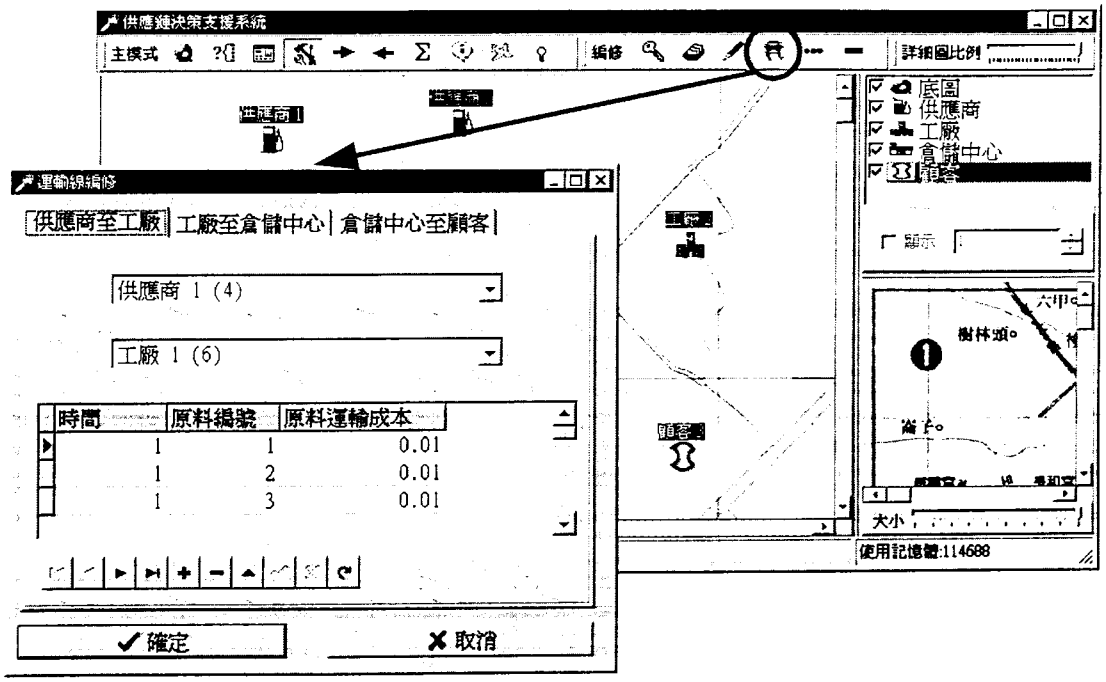
進入系統後，在任一節點上點選，就可以看到這個節點的資料。

（2）修改供應鏈中某個角色的資料



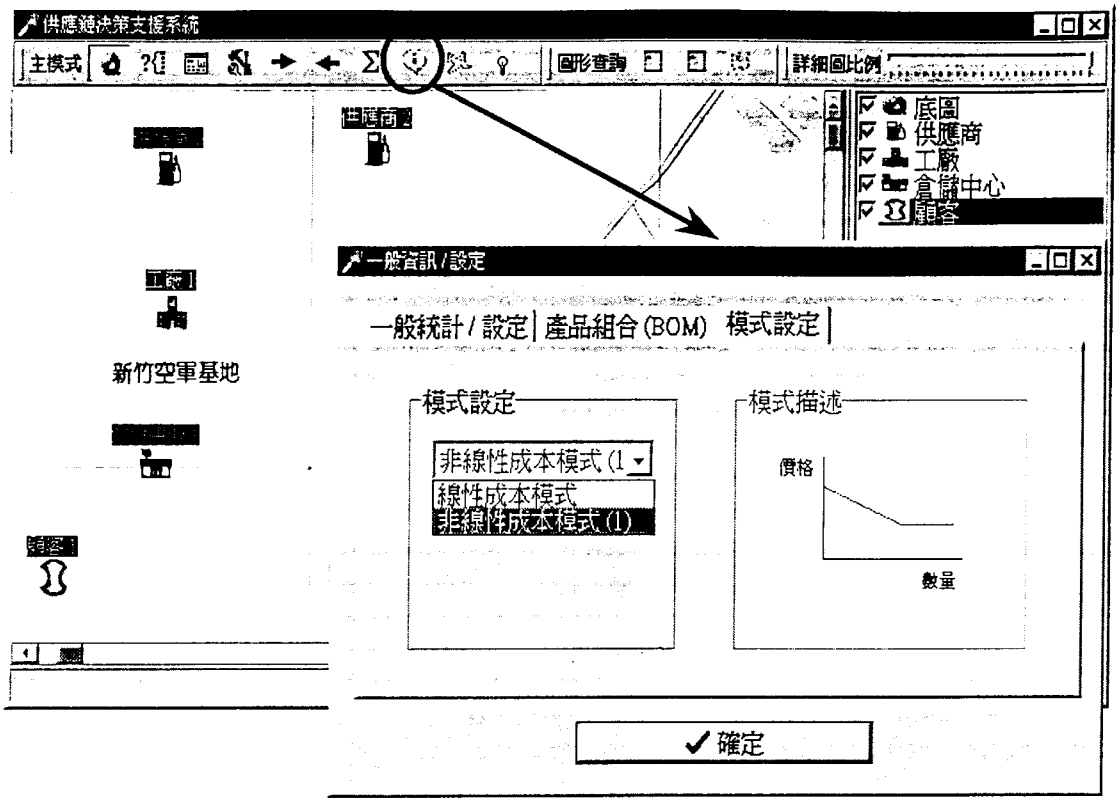
修改時，先按下主模式工具列第四個按鍵，通過身份確認之後，再按下編修工具列第三個按鍵，接著點選一節點，就可修改資料。(底色為黃色代表可以修改)。

(3) 運輸線編修



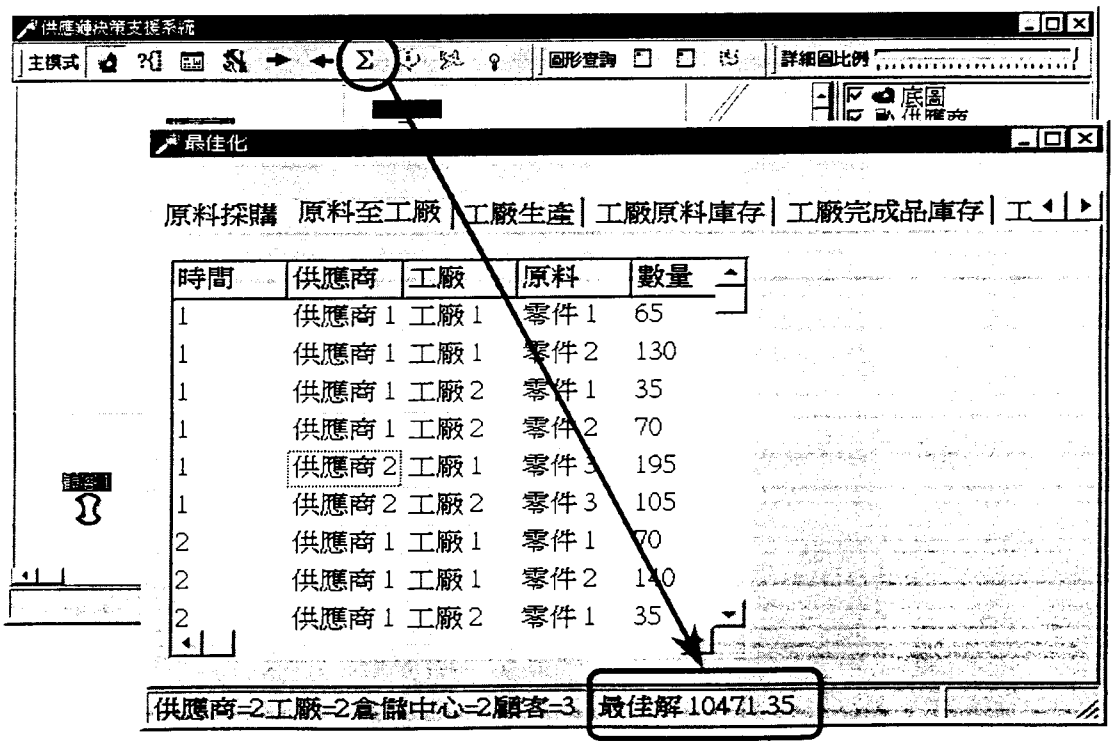
修改時，先按下主模式工具列第四個按鍵，通過身份確認之後，再按下編修工具列第四個按鍵，就可修改運輸線資料。(底色為黃色代表可以修改)。

(4) 模式設定



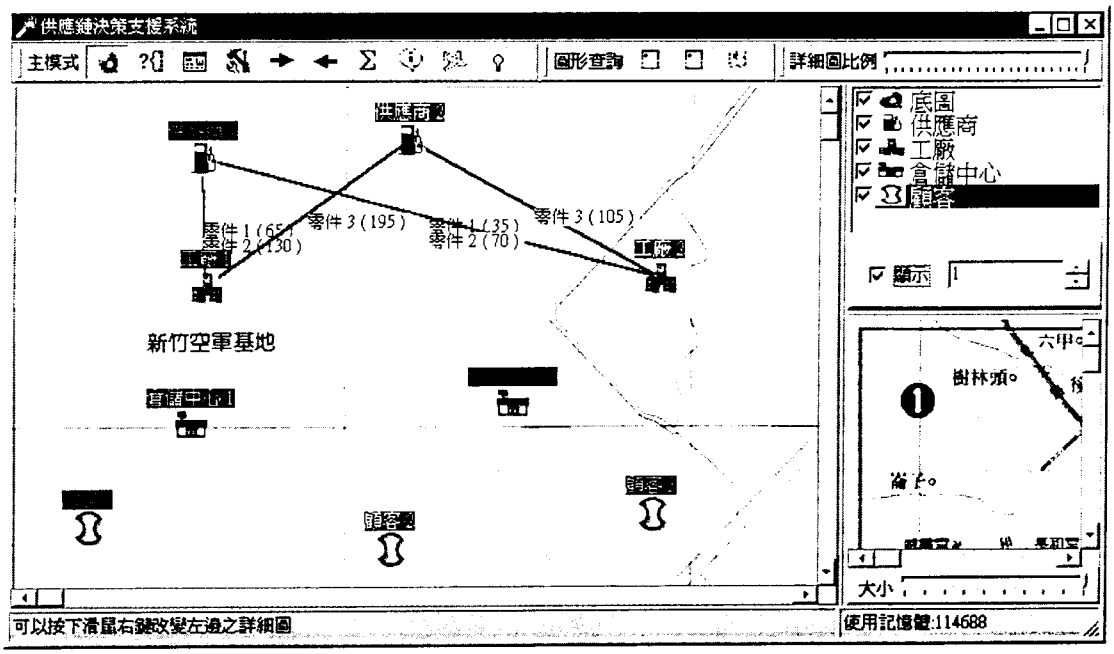
按下主模式工具列第八個按鍵，就可選擇模式及設定產品組合資料。

（5）最佳解運算



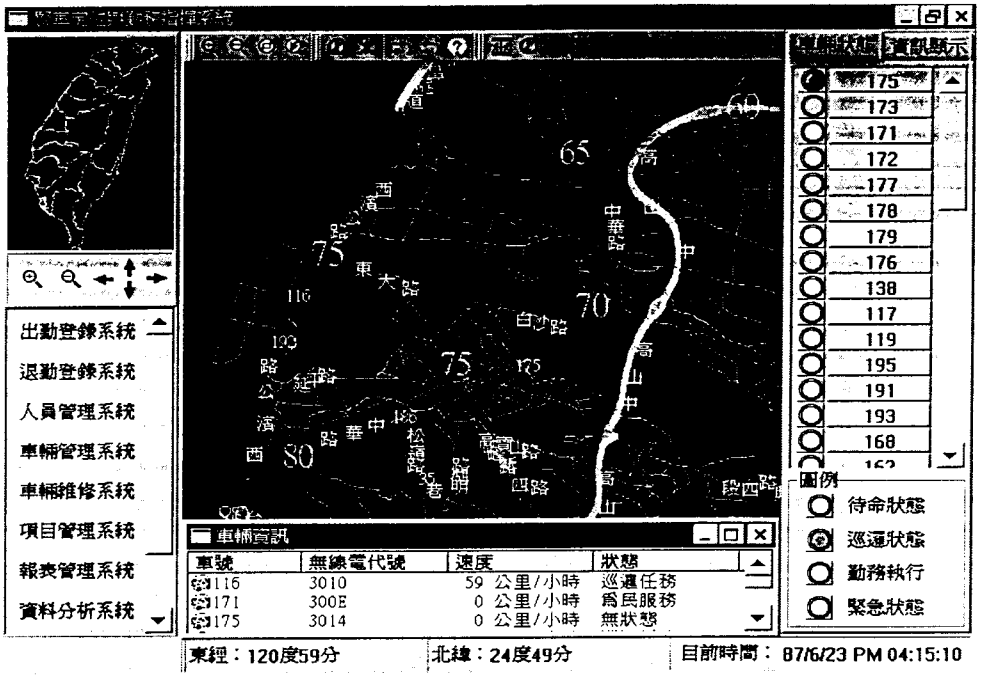
按下主模式工具列第七個按鍵，就可以目前模式及資料算出供應鏈最小成本。

（6）顯示結果



進行任何一次計算之後，將右方顯示的選取方塊打勾，並選擇時期，就可以看到解答的結果。

（7）油罐車巡行監控系統執行畫面



油罐車勤務指派與巡行監控系統執行畫面

四、供應鏈資料庫分析與設計

4.1 資料庫之特性

儲存 GIS 之資料庫又稱為空間資料庫(Spatial database)，而空間資料庫與一般傳統純文字資料庫在考量上複雜許多，因考慮之層面由一維擴展為二維(若再加上高度則成為三維)，而主要考慮之問題有 4 點：

・空間查詢(Location; What is at...?)

查詢圖面上任一地點時之定位問題，而查詢之地點可分為三大類：

- **點資料－油品供應商、煉油廠、儲油槽、加油站等據點。**
- **線資料－道路、河流、地下輸油管線等。**
- **面資料－土壤、地質、地籍、地理位置等。**

空間分佈之分析(Patterns; What spatial patterns exist?)

例如犯罪地點、交通事故地點的集中情形。如癌症病患分佈與環境污染之關係、零售商店空間分佈分析及客戶資料空間分佈分佈等。

屬性查詢(條件查詢) (Condition; Where is it...?)

查詢符合特定條件的地點在那裡(區位查詢)。例如理想的煉油廠、儲油槽、加油站、油罐車等據點。

最佳路徑尋找(Routing; What is the best way...?)

任意兩點或數點之間找出最佳路徑。例如油罐車與加油站的即時服務、儲油槽與加油站之間的最佳路徑之規劃等。

其中 1，2 點可說與資料庫綱目(Schema of Data Base)息息相關，而第 3 點以後則是仰賴知識庫(Knowledge Base)之設計，在此研究中暫且只做到 1，3 點，而第 4 點已完整提出理論架構且實做也正進行中。

4.2 需求分析

通常空間資料(Spacial Data)可視為三部分：

(1) 圖形資料(Graphics) －點、線、面。

(2) 座標資料(Coordinate)－球面座標/平面座標。

(3) 空間關係(Topology)－相鄰/相連/組成。

在此研究中以達成點資料之查詢為主要目的。至於線、面資料之查詢暫不加以討論，故圖形資料中可分為點、面。其中面圖形資料為底圖資料(也就是未做任何疊圖時之地圖)，以網格格式的圖檔為主，而一個底圖可以包含若干個子底圖，當然子底圖更可以包含若干個孫底圖，如此一直延續下去直到最詳細之底圖為止，故底圖之層數為變動式並未固定。而點圖形資料為各個圖層之圖示(Logo)，用以疊在底圖上作為顯示之用。

當某一圖層有據點實際發生時只要將所發生之座標位置記錄下來即可，在此座標資料以平面座標為主，且記錄方式為記錄此據點發生在哪一個底圖之相對位置。雖然使用相對平面座標時，一旦此底圖的比例尺有變動，資料即宣告失效；但好處為簡單、運算快速。

而空間關係就只有地圖與底圖之間的對應(Mapping)問題，也就是說在一張父底圖上包含哪些子底圖，而父底圖上之座標點是座落於哪一張子底圖之上。在此使用下列規則來規定底圖的空間關係(Topology)：「在每一張子底圖上皆紀錄著此資料，此子底圖座落於父底圖的哪一個區間上，而用一連串之點(Point)將此區間圍繞起來」。

4.3 資料庫 E-R Model

將上述需求用 E-R Model 表示如下：

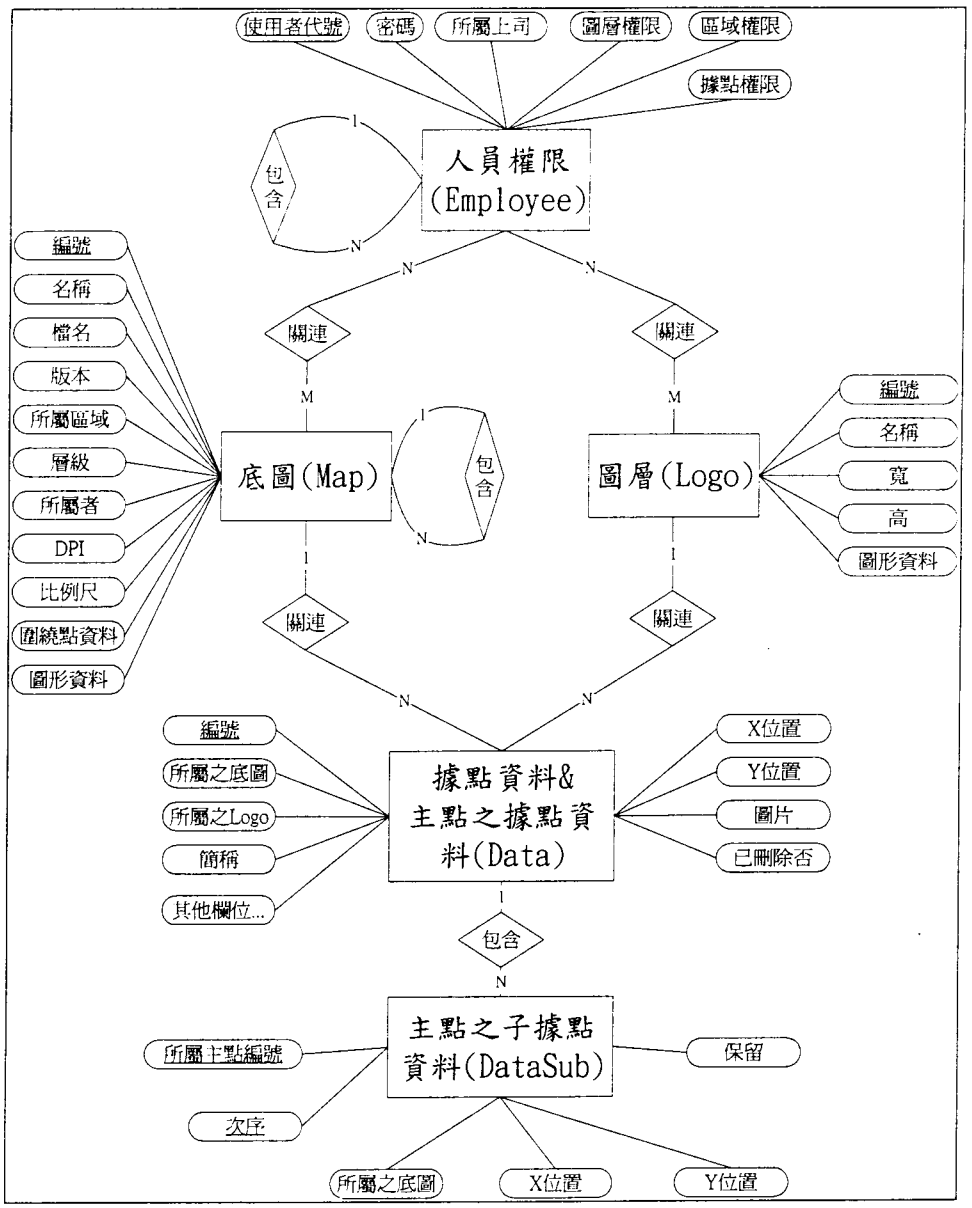


圖 1　本系統資料庫之 E-R Model

下列描述各個 Entity 之功能及其關係：

(4) 人員權限檔(Employee)

記錄所有維護人員之名稱、密碼及權限，且記錄人員中上司與下屬之關係(因此自己具有一對多之關係)，只有

在此檔案中有紀錄之人員才有權限去做 Web GIS 之編修動作。在權限部分共有圖層權限、區域權限及據點權限。其中圖層權限用以記錄是否可以對圖層(Logo)檔編修。區域權限則和記錄是否允許對底圖(Map)檔上每一張底圖做據點的處理，雖然其關係是多對多，但在此系統中有限制底圖(Map)檔最多只能包含 256 張底圖，故不需將多對多關係再遷一個檔案加以記錄。據點權限則是分別記錄各個圖層據點的編修權限，其關係亦為多對多。但在此系統中限制最多只能包含 200 個圖層，亦不需將多對多關係遷出一個檔案記錄。

(5) 底圖檔(Map)

記錄所有底圖之資訊，在此系統中限制底圖檔最多包含 256 張底圖。每張底圖皆可以擁有多張子底圖(父子底圖之空間關係資訊)，所以自己具有一對多之關係。

(6) 圖層檔(Logo)

記錄所有各個圖層之圖示資料。

(7) 單一點之據點資料&主點之據點資料檔(Data)

記錄圖層資料真正發生時之所在位置，故與所屬的底圖檔(Map)與發生之圖層(Logo)具有關連。

(8) 主點之子據點資料檔(DataSub)

作為記錄線性資料，保留作為日後擴充用。

4.4 資料庫之綱目(Schema)

將上述需求加以正規化後得到五個表格(Table)，為人員權限檔(Employee)、底圖檔(Map)、圖示檔(Logo)、單一點之據點資料&主點之據點資料檔(Data)及主點之子據點資料(DataSub)。地圖資料庫其檔案詳細規格如下：

4.4.1 人員權限檔 : Employee

欄位部份

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
使用者代號	Name	Char	10	N		Y	*
密碼	Password	Char	10	N			
所屬上司	Father	Char	10	Y			若爲局使用者此欄位爲 null
類別	Type	Binary	1	N			0:段使用者 1:處使用者 2:局使用者
圖層權限	Lright	Binary	1	N			
區域權限	Mright	Binary	32	N			每一個 bit 代表是否可編修此區域,最多 256 個區域 (8*32)
據點權限	Pright	Binary	200	N			每一個 Byte 代表是否可編修此據點之地圖資料

註:LRight 及 Pright 欄位每一個 Byte 的結構如下:

Bit 位置	意義
0	0:不可新增 1:可新增
1	0:不可修改 1:可修改
2	0:不可刪除 1:可刪除

4.4.2 底圖檔 : Map

欄位部份:

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
編號	No	Smallint	2	N		Y	* 序號 From 1(爲唯一)
名稱	Name	Char	50	N		Y	此底圖之意義
檔名	FileName	Char	12	N		Y	欲存在 Client 端之檔名
版本	Version	Smallint	2	N			
所屬區域	Type	Smallint	2	N			隸屬於哪一個第一層底圖
層級	Tier	Smallint	2	N			圖層 0,1,2,...(0 爲第一張圖)
所屬者	Father	Smallint	2	N			所隸屬之上一張底圖編號
DPI	DPI	Smallint	2	Y	0		
比例尺	Rate	Int	4	Y			只顯示分母
圍繞點資料	Scope	Text		Y			此圖在上一曾之範圍
圖形資料	Image	Image		Y			

索引部份:

名稱	唯一	組成欄位
No_Map	Y	No
Name_Map	Y	Name
Father_Map	Y	Father+No
Tier_Map	Y	Tier+No

4.4.3 圖示檔 : Logo

欄位部份:

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
編號	No	Smallint	2	Y			序號 From 1(但實際爲唯一)
名稱	Name	Char	16	N		Y	*
寬	Width	Smallint	2	N			
高	Height	Smallint	2	N			
圖形資料	Data	Image		N			

索引部份:

名稱	唯一	組成欄位
No_Logo		No

4.4.4 單一點之據點資料&主點之據點資料檔 ：Data

欄位部份:

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
編號	No	Int	4	Y		Y	序號 From 1
所屬之底圖	MapID	SmallInt	2	N			* 底圖編號 From 0(主點之據點不使用此屬性)
所屬之 Logo	LogoID	Smallint	2	N			* 爲 Image 之 No 欄位
簡稱	Name	Char	16	N			* 作爲秀在底圖上之用(FK)
X 位置	X	Smallint	2	N	0		在 MapName 底圖之位置(主點之據點不使用此屬性)
Y 位置	Y	Smallint	2	N	0		
地點	Address	Varchar	80	Y			
電話	Tel	Varchar	14	Y			
傳真	Fax	Varchar	14	Y			
url 連線位置	URL	Varchar	255	Y			
所在道路	Road	Varchar	40	Y			
內容	Context	Varchar	80	Y			
日期 1	Date1	Date	8	Y			
日期 2	Date2	Date	8	Y			
日期 3	Date3	Date	8	Y			
已刪除否	Del	Smallint	2	N	0		0:否 1:是
描述	Description	Text		Y			
圖片	Picturee	Image		Y			

索引部份:

名稱	唯一	組成欄位
No_Data	Y	No
LogoID_Data		LogoID+MapID+Name
Name_Data		Name
XY		MapID+LogoID+X+Y

4.4.5 主點之子據點資料 : DataSub

欄位部份:

中文意義	欄位名	資料形態	大小	Null	內定值		備註(*:代表主鍵)
所屬主點編號	DataNo	Int	4	N			* 爲 Data 之 No 欄位
次序	No	Smallint	2	N			*
所屬之底圖	MapID	SmallInt	2	N			* 底圖編號 From 0
X 位置	X	Smallint	2	N			在 MapName 底圖之位置(專爲主點之據點用)
Y 位置	Y	Smallint	2	N			
保留	Reverse	Char	20	Y			

索引部份:

名稱	唯一	組成欄位
No_DataSub	Y	DataNo+No

4.5 系統資料庫

決策資料分別由油品供應商、煉油廠、儲油槽、加油站、油罐車等透過網路將資料寫入資料庫中。

油品供應商表格（Vendor）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
供應商編號	Vendor_ID	Int	4	N		Y	會參考到 Data 表格中的 No
原油編號	Parts_ID	Int	4	N		Y	原油編號
原油價格	Parts_Price	Numeric	18	N		N	原油價格
原油供給上限	Parts_Supply_UB	Int	4	N		N	原油供給上限

供應商運送原油至煉油廠表格（VF_Link）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
供應商編號	Vendor_ID	Int	4	N		Y	會參考到 Data 表格中的 No
煉油廠編號	Facility_ID	Int	4	N		Y	會參考到 Data 表格中的 No
原油編號	Parts_ID	Int	4	N		Y	原油編號
原油運輸價格	Parts_Trans_Price	Numeric	18	N		N	原油運輸價格

煉油廠生產完成油品/油品存貨表格（Facility_Prod_Goods）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
煉油廠編號	Facility_ID	Int	4	N		Y	會參考到 Data 表格中的 No
油品編號	Goods_ID	Int	4	N		Y	完成品編號
油品生產價格	Goods_Prod_Price	Numeric	18	N		N	完成品生產價格
油品存貨價格	Goods_Inv_Price	Numeric	18	N		N	完成品存貨價格
油品生產上限	Goods_Prod_UB	Int	4	N		N	完成品生產數量上限
油品生產下限	Goods_Prod_LB	Int	4	N		N	完成品生產數量下限

煉油廠原油存貨表格（Facility_Inv_Parts）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
煉油廠編號	Facility_ID	Int	4	N		Y	會參考到 Data 表格中的 No
原油編號	Parts_ID	Int	4	N		Y	原油編號
原油存貨價格	Parts_Inv_Price	Numeric	18	N		N	原油存貨價格

煉油廠運送完成品至儲油槽中心表格（FW_Link）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
煉油廠編號	Facility_ID	Int	4	N		Y	會參考到 Data 表格中的 No
儲油槽編號	Warehouse_ID	Int	4	N		Y	會參考到 Data 表格中的 No
油品編號	Goods_ID	Int	4	N		Y	完成品編號
油品運輸價格	Goods_Trans_Price	Numeric	18	N		N	完成品運輸價格

儲油槽完成油品存貨表格（Warehouse）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
儲油槽中心編號	Warehouse_ID	Int	4	N		Y	會參考到 Data 表格中的 No
油品編號	Goods_ID	Int	4	N		Y	完成品編號
油品存貨價格	Goods_Trans_Price	Numeric	18	N		N	完成品存貨價格
油品存貨上限	Goods_Inv_UB	Int	4	N		N	完成品存貨數量上限
油品存貨下限	Goods_Inv_LB	Int	4	N		N	完成品存貨數量下限

儲油槽運送油品至加油站表格（WC_Link）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
儲油槽編號	Warehouse_ID	Int	4	N		Y	會參考到 Data 表格中的 No
加油站編號	Customer_ID	Int	4	N		Y	會參考到 Data 表格中的 No
油品編號	Goods_ID	Int	4	N		Y	完成品編號
油品運輸價格	Goods_Trans_Price	Numeric	18	N		N	完成品運輸價格

加油站需求油品表格（Customer）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
時間	Time	Int	4	N		Y	時間編號
加油站編號	Customer_ID	Int	4	N		Y	會參考到 Data 表格中的 No
油品編號	Goods_ID	Int	4	N		Y	完成品編號
顧客需求	Goods_Need	Int	4	N		N	顧客需要完成品的數量

油品供應商運送原油至煉油廠的延遲時間表格（VF_Leadtime）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
供應商編號	Vendor_ID	Int	4	N		Y	會參考到 Data 表格中的 No
煉油廠編號	Facility_ID	Int	4	N		Y	會參考到 Data 表格中的 No
延遲時間	Lead_Time	Int	4	N	(1)	N	延遲時間

煉油廠運送完成品至儲油槽的延遲時間表格（FW_Leadtime）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
煉油廠編號	Facility_ID	Int	4	N		Y	會參考到 Data 表格中的 No
儲油槽編號	Warehouse_ID	Int	4	N		Y	會參考到 Data 表格中的 No
延遲時間	Lead_Time	Int	4	N	(1)	N	延遲時間

儲油槽運送油品至加油站的延遲時間表格（WC_Leadtime）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
儲油槽編號	Warehouse_ID	Int	4	N		Y	會參考到 Data 表格中的 No
加油站編號	Customer_ID	Int	4	N		Y	會參考到 Data 表格中的 No
延遲時間	Lead_Time	Int	4	N	(1)	N	延遲時間

物料需求表格（BOM）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
油品編號	Goods_ID	Int	4	N		Y	完成品編號
原料編號	Parts_ID	Int	4	N		Y	原料編號
需求量	Needs_QTY	Int	4	N		N	一個完成品需要多少的原料

原油表格（Parts）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
原油編號	Parts_ID	Int	4	N		Y	原料編號
原油名	Parts_Name	Varchar	64	N		N	原料名稱
原油描述	Parts_Description	Text		Y		N	原料詳細描述

油品表格（Goods）

中文意義	欄位名	資料形態	大小	Null	內定值	唯一	備註(*:代表主鍵)
油品編號	Goods_ID	Int	4	N		Y	完成品編號
油品名	Goods_Name	Varchar	64	N		N	完成品名稱
油品描述	Goods_Description	Text		Y		N	完成品詳細描述

五、供應鏈地圖庫之建構

此系統中資料可分兩類， Web GIS Server 底圖庫與 Web GIS Server 圖層據點資料庫。Web GIS Server 底圖庫為由網站管理者建構；Web GIS Server 圖層據點資料庫則可透夠過本文提出之遠程分散式方法由被授權者建構。

Web GIS Server 底圖庫必須經過下列手續：底圖取得、底圖掃描、底圖編修及底圖資料庫之操作，分述如下：

5.1 底圖取得

底圖的類別有兩種：

(1)網格式底圖(Grid Map)

直接用光學掃瞄器掃瞄市面上之地圖集。其優缺點如下：

- **優點：圖形漂亮自然、地圖取得容易且成本較低(掃瞄即可得到)、處理單純快速、檔案規格統一、在瀏覽器上顯示無問題。**
- **缺點：檔案龐大不利於 Internet 傳輸、放大縮小時圖形會失真。**

(2)向量格式底圖(Vector Map)

向專業製圖公司購買向量電子圖檔。其優缺點如下：

- **優點：檔案小利於 Internet 傳輸、圖形可做任何比例之放大縮小。**
- **缺點：圖形不夠自然真實、偏遠區塊的地圖取得困難且成本非常高、數化處理複雜速度慢、檔案規格不統一、在瀏覽器上顯示有困難。**

基於台灣製作電子向量圖檔的公司不多，且偏遠地區也找不到相對應電子向量圖檔(一般只有都會區才有電子向量圖檔)，而數化成本高的因素下，本文選擇網格格式之圖檔作為系統之底圖。

而底圖的來源以戶外生活圖書公司所出版的一系列地圖集為主。該套地圖集分為台灣全區圖(1/50 萬)、地區地圖(1/10 萬)與各都會區地圖(1/10000)。此系列地圖是目前民間編製地圖中最翔實、精緻的一套。戶外生活圖書公司已授權交通大學資訊管理研究所可將其出版之部分圖幅上網，只要該類底圖是放在交大伺服主機內且為公益使用皆為合法。

5.2 底圖掃描

圖形的掃瞄並無統一之方式，視其使用的掃瞄器驅動程式與影像處理軟體而定，在此以 UMAX 掃瞄器所附驅動程式及友立公司所出的 PhotoImpact 影像處理軟體作為範例。掃描器準備妥當後(掃描器的操作請參考相關手冊)依下列步驟操作：

(9) 先執行繪圖軟體（例如：PhotoImpact），而此繪圖軟體須支援掃瞄器取得影像功能者，才能使用。

(10) 在「檔案」選單上下拉，點選「取得」再選取「影像」(如下圖 1)。

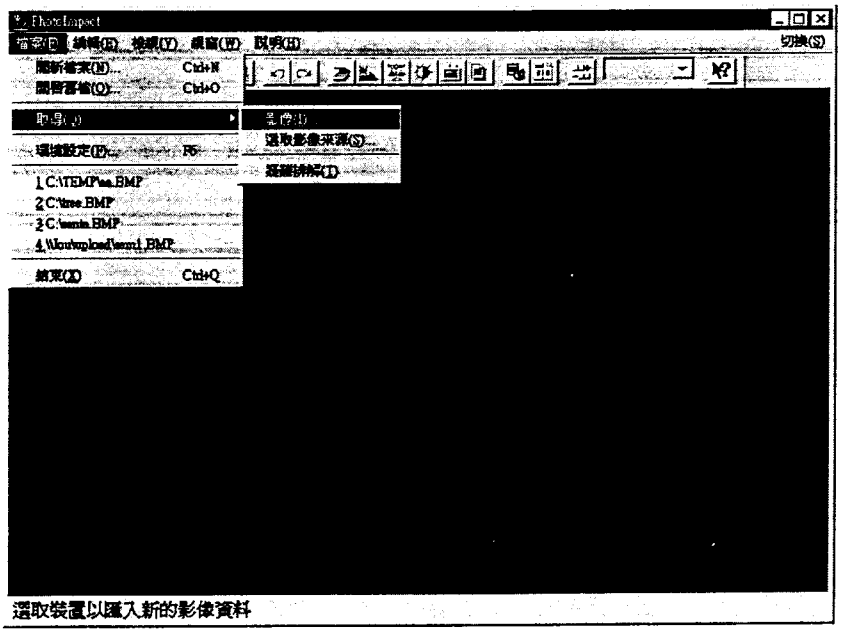


圖 1　PhotoImpact 操作畫面

(11) 接著 PhotoImpact 就會問要將取得的影像，存放成”新影像”、”檔案”、”印表機”或”傳真／郵件”。 基本上我

們是將掃瞄進來的影像存放繪圖程式中，然後檢查有無問題後再存檔，所以請選擇「新影像」再按下「取得」鍵(如下圖 2)。

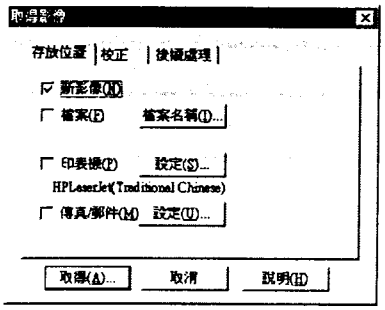


圖 2　PhotoImpact 操作畫面

(12) 在下圖中可以看到掃圖程式的畫面，請選擇影像種類為「全彩」，材質為「反射式」，掃瞄解析度為「150DPI」，改變影像比例為「100%」也就是不改變。再將「自動色彩校正」與「明暗度校正」點選起來，以下的 Gamma、明度、暗度、對比、亮度就可以不必設定。（當然您也可以自行設度，但是要記得最佳的值是多少，以後每次掃瞄地圖時都要設定，才會有一致性）。

(13) 接著請按下「預掃瞄」鍵，就可以在右方看到大概的圖片(如下圖 3)。

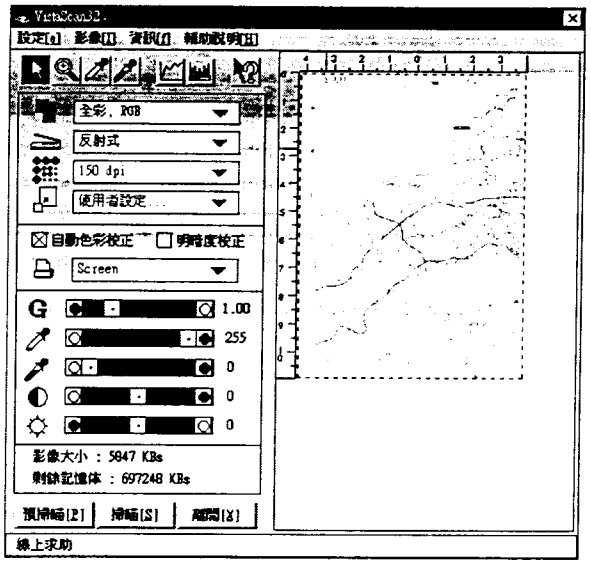


圖 3　PhotoImpact 操作畫面

(14) 預覽掃瞄沒問題後，就可以按下「掃瞄」鍵。圖片掃瞄完畢後，緊接著就會將它存入 PhotoImpact 軟體中(如下圖 4)。

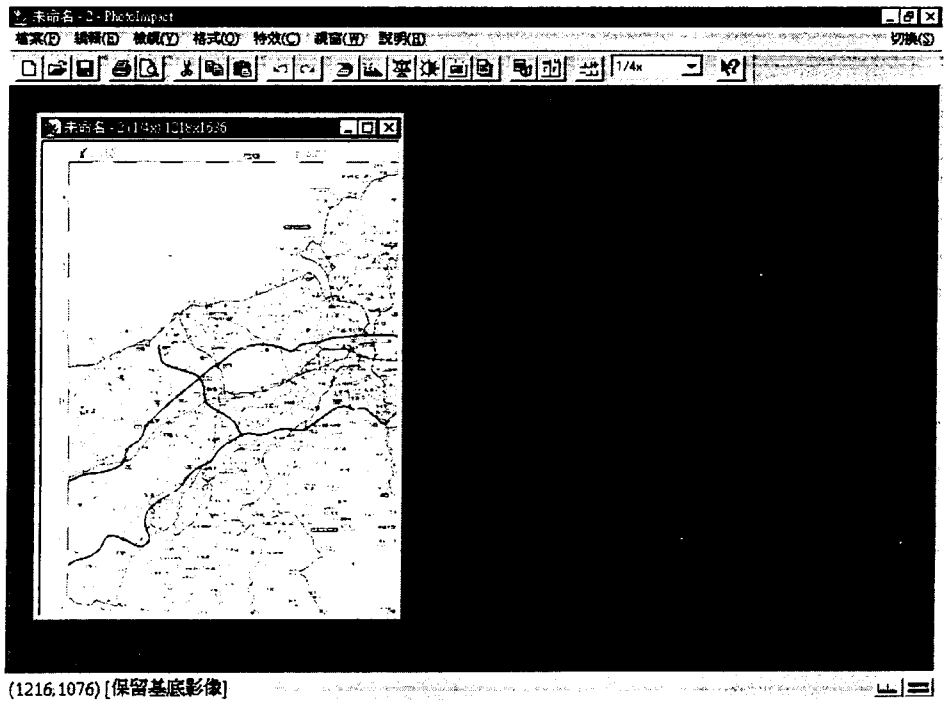


圖 4　PhotoImpact 操作畫面

(15) 或許你會覺得看不清楚或太小了，此時可以選擇「檢視」中的原圖大小就可以看到完整詳細的地圖了(如下圖 5、圖 6)。

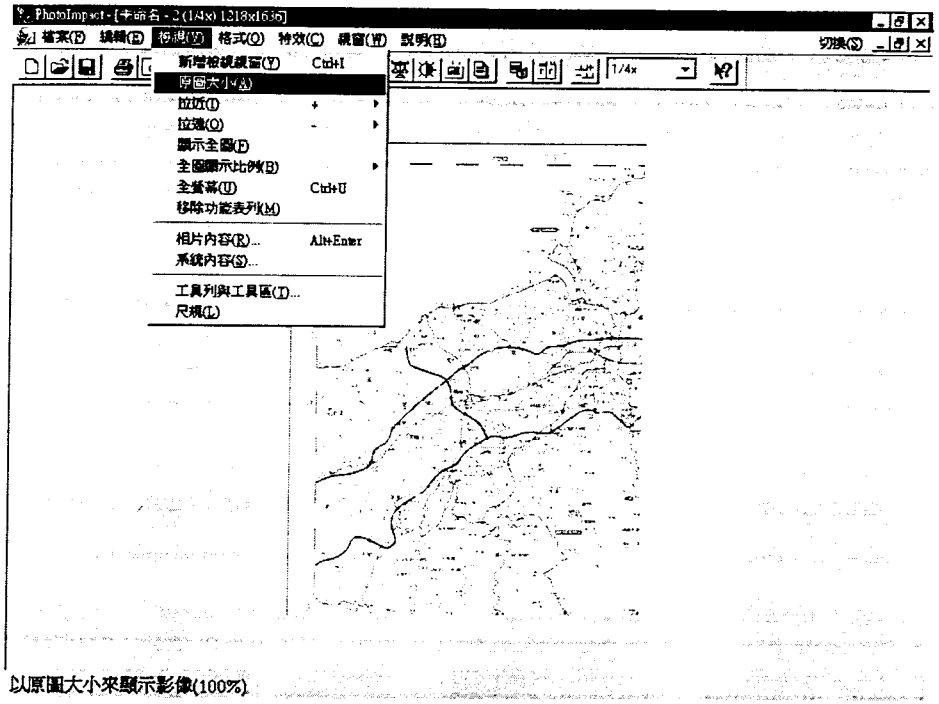


圖 5　PhotoImpact 操作畫面

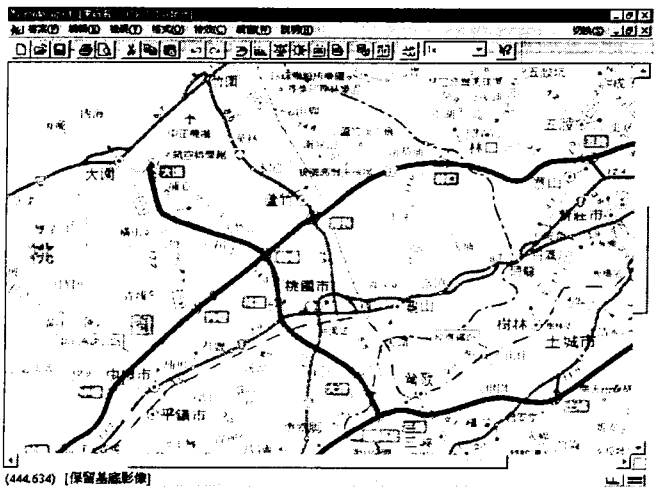


圖 6　PhotoImpact 操作畫面

5.3 底圖校調

圖形掃瞄後，必須加以校調才適合存放於資料庫以利於Internet上傳輸及顯示。其主要的校調事項有：

1. 圖形區塊的裁減

有時在掃瞄的圖形中，可能只需要其中的一小部分，此時就需將需要的部分做區塊的裁減。其操作如下：先用滑鼠將所需要的區域拖曳，在「編輯」選單上下拉，點選「複製(C)」。在「編輯」選單上下拉，點選「貼上 (C)」，再點選「貼成新影像(N)」。最後在此貼上之影像中按滑鼠右鍵，點選「全部合併(A)」，即完成圖形區塊的裁減(如下圖 7)。

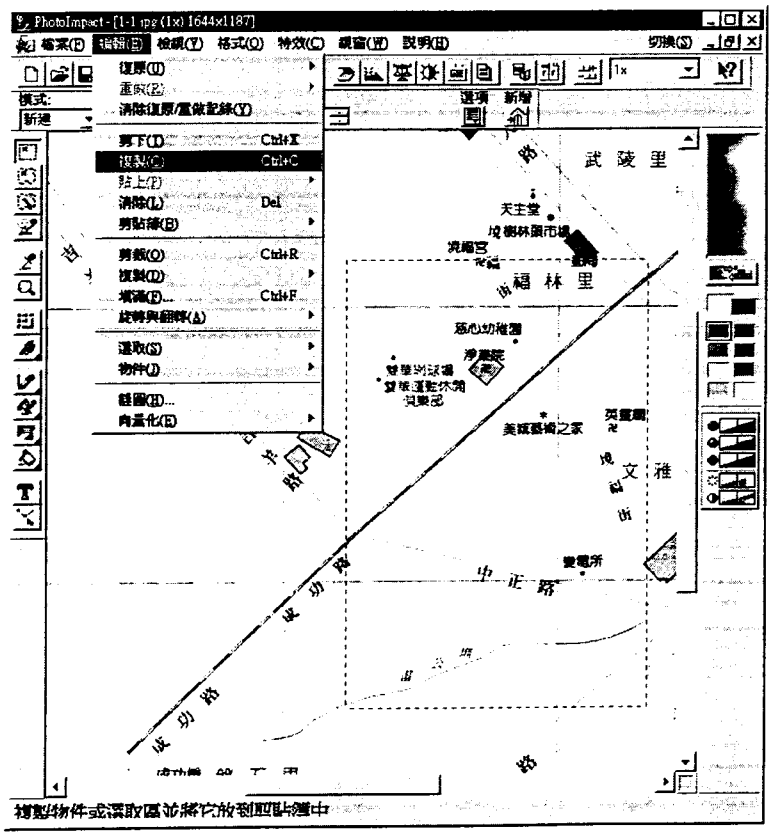


圖 7 PhotoImpact 操作畫面

圖形區塊的縫合

有時在掃瞄的圖形中，需要將兩張圖形合併形成一張，此時就需將做圖形的縫合。其操作如下：先準備兩張圖形準備要縫合，在「編輯」選單上下拉，點選「縫圖 (H)...」。則出現一畫面(如下圖 8)，在此畫面中必須移動兩張圖形直到結合時，按下確定即可做圖形區塊的縫合。

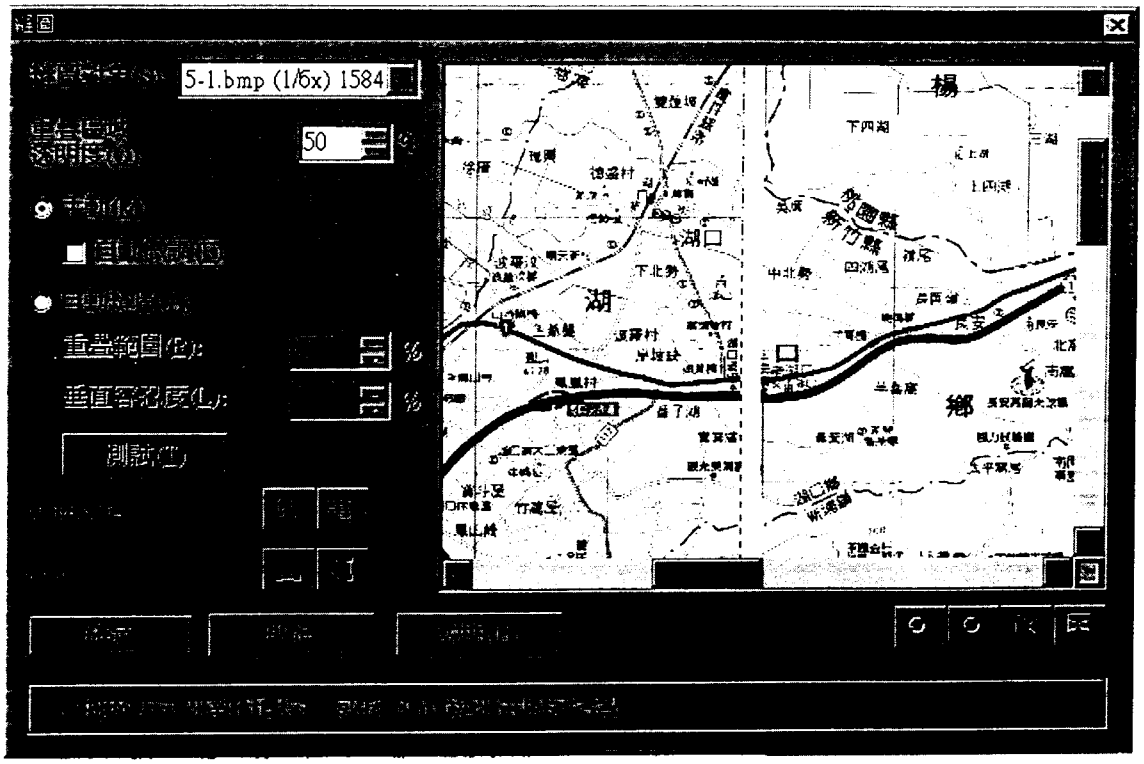


圖 8　PhotoImpact 操作畫面

圖形參數的調整

圖形剪裁及合併完成後，需將圖形做一些參數的調整，如此才能在顯示觀感上得到最大效益。其需要調整的參數有：

(1)去網紋

凡是印刷品經過掃瞄後，接會產生印刷網紋，而造成圖形品質下降。故必須將圖形作去網紋處理，其步驟如下：在「特效」選單上下拉，點選「雜點(N)」後再選擇「移除印刷紋路 (R)」。(如下圖 9)

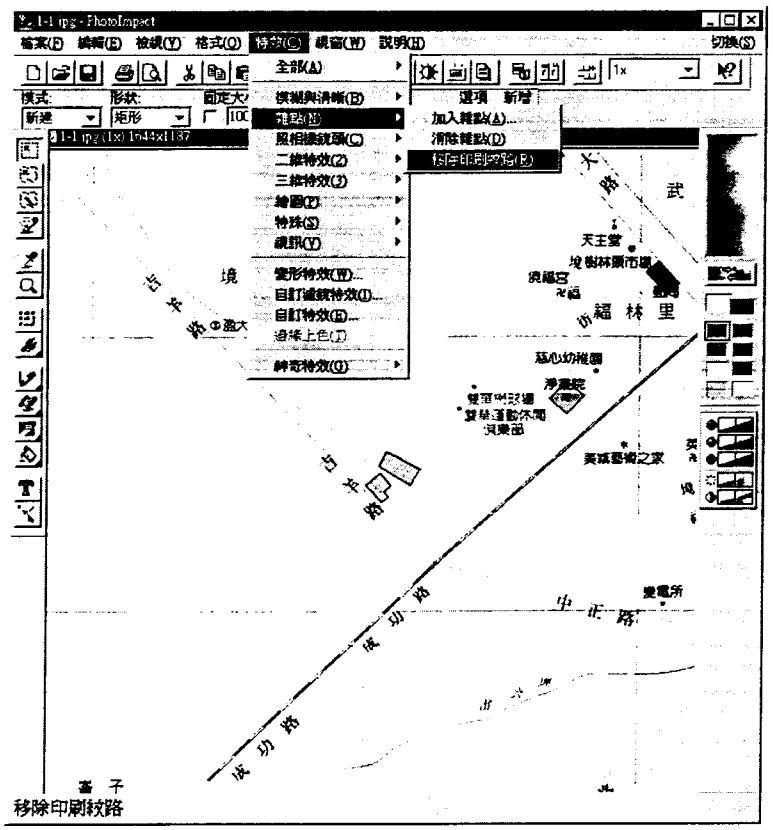


圖 9　PhotoImpact 操作畫面

(2)亮度與對比

一張圖形往往會亮度與對比不對，而使品質不佳。其調整方法為：在「格式」選單上下拉，點選「亮度與對比(B)... Ctrl-B」則會出一畫面(如下圖 10)，讓您自由調整亮度與對比。

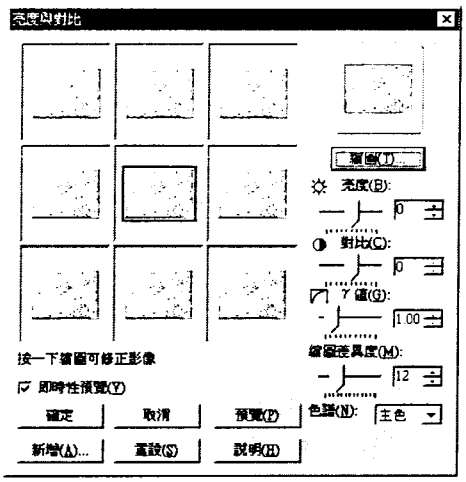


圖 10　PhotoImpact 操作畫面

(3)色彩飽和度

有時色彩大鮮豔時會使人眼睛不舒服，其調整方式為：在「格式」選單上下拉，點選「色相與飽和度(H)...」則會出一畫面(如下圖 11)，讓你自由調整色彩飽和度。

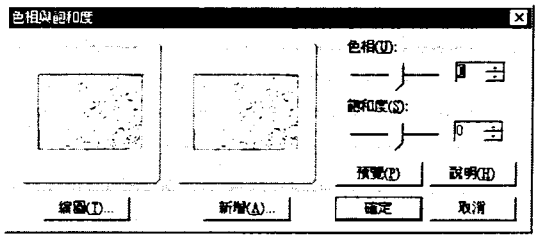


圖 11　PhotoImpact 操作畫面

(4)色調調整

圖形中有最暗點、最亮點、中間值。其調整可以有效增加圖形之清晰度，而數值之調整通常要依靠經驗，其調整方式為：在「格式」選單上下拉，點選「調整色調(T)... F8」則會出現一畫面(如下圖 12)，後再點選上方的「高亮度、中間值、陰影」頁次，讓你自由調整最暗點、最亮點、中間值。

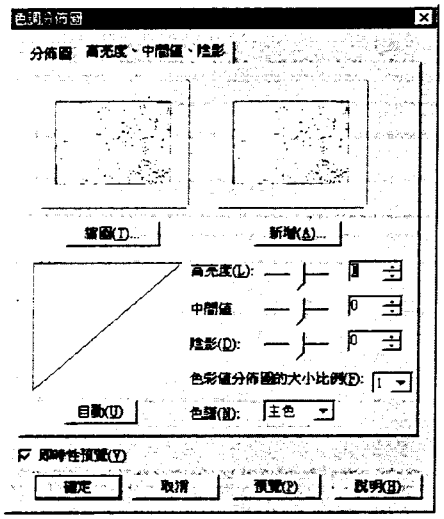


圖 12　PhotoImpact 操作畫面

(5)降低色階

圖形校調完畢最就要降低色階，以最少的空間儲存所需的資訊。因底圖檔的顏色有 6 色(6 色套印)，而掃瞄進來的圖形皆為 True Color，在此只要將圖形色階為 256 色即可(圖形品質不被破壞且有間省記憶空間)。其操作方式為：在「格式」選單上下拉，點選「影像類型(P)」再點選「最佳化 256 色(2)(8-bit)」　即可將 True Color 轉換為 256 色。(如下圖 13)

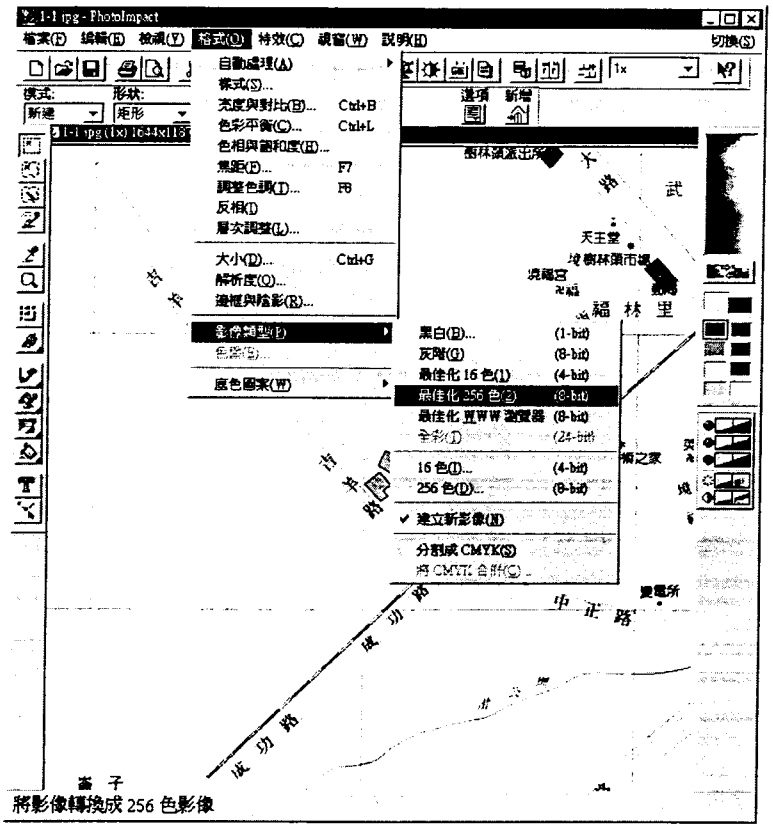


圖 13　PhotoImpact 操作畫面

(6)壓縮及存檔

為節省儲存空間，必須要將圖形檔做適當的壓縮。在此選擇以不失真壓縮為主，其壓縮檔案格式在此選擇 GIF。因 GIF 檔剛好是用儲存 256 色的圖形，高壓縮率且不失真，為 Internet 上 Web 用已顯示不失真圖形的最佳選擇。其操作方式為：在「檔案」選單上下拉，點選「另存新檔(A)...」則會出一畫面(如下圖 14)，在此畫面的「存檔

類型(T)」選擇 GIF(圖形交換格式)，最後在「存檔名稱(N)」中輸入名稱，按下「儲存檔案(S)」即可。

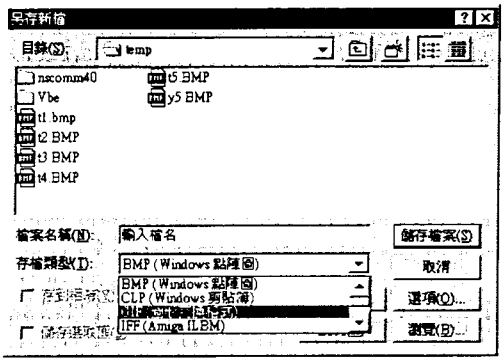


圖 14 PhotoImpact 操作畫面

5.4 底圖資料庫之建構

當所有圖形檔取得且處理完畢後，即可開始底圖資料庫之建構。此時需要執行後端底圖資料庫建立程式(其執行畫面如圖 15)。在底圖結構上就如同樹狀結構般，最開始只有一張圖(如台灣省縮圖)，而只要在此圖上按下新增鍵即可在此主圖上新增一個子圖(如北、中、南台灣省縮圖)，圖時在所編輸入相對應之資料。此底圖可以一層層新增下去直到最詳細之比例尺為止(在圖 15 一共有 5 層)。而修改及刪除只要先選取欲邊修之底圖，再壓下相對應按鍵即可完成工作。左下角為此圖形的基本資訊，在此可以輸入名稱、版本、DPI、比例尺及圖繞點。其中圖繞點用以構成一個封閉多邊型描述此圖位於上一張底圖位置的區域，用以表示圖形間從屬及相鄰資訊。

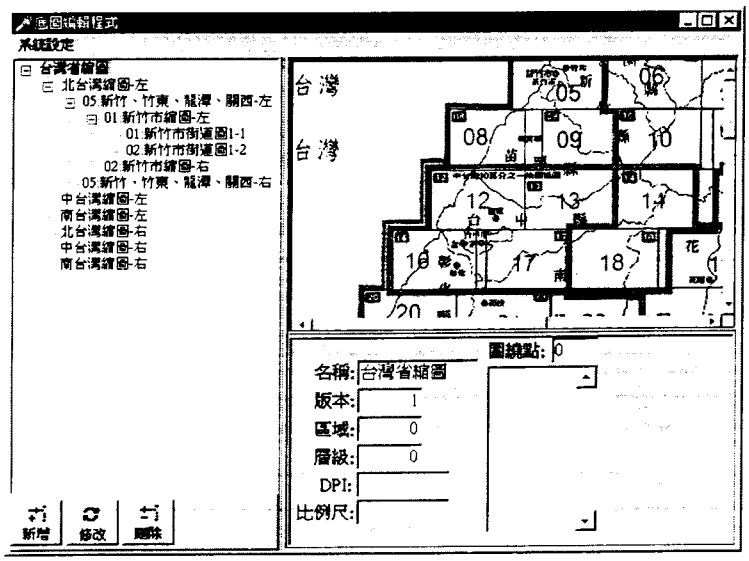


圖 15 底圖資料庫建構之執行畫面

六、油品運銷供應鏈系統設計說明

本系統模式建立的數學模式係以 LINDO (1997)為基礎發展的。其特性為：

1、 與地圖結合，使資料的輸入與輸出均可以地圖或示意圖形式顯示，操作簡便易懂。

2、 建立模式庫以利應用模式間的串連。

3、 與由網站上下載的資料相結合後運算。

擬建立的應用模式如：

1、 管佈設最佳模式。

2、 車最佳巡行模式。

3、 加油站選址模式。

將台灣省的街道圖製成底圖之後，並運用疊圖、點圖的技術，將各據點的位置繪製於底圖上，並且與資料庫相結合。經由瀏覽器透過網際網路連上本系統之後，經由其查詢條件過濾之後，得到相關的資訊點，發展成為一完整的網際決策資訊系統。

6.1 查詢底圖及圖層資料

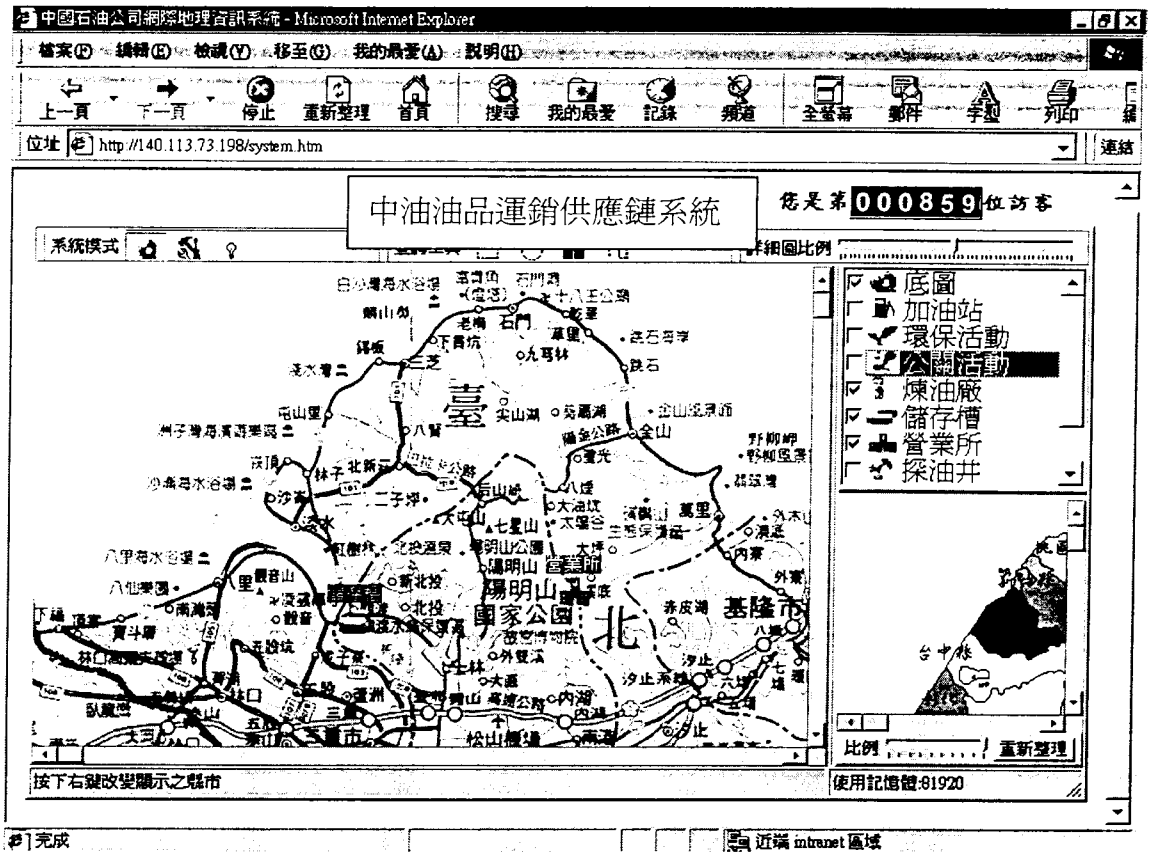


圖 6-1 中油油品運銷供應鏈管理系統

圖 6-1 為中油油品運銷供應鏈管理系統之主畫面。上方之工具列(Tool Bar)為本系統之所有查詢功能。右上方做為顯示圖層之控制視窗，只要在方框內打勾就會動態的將據點類別(主題層)疊合於縣市底圖上。右下方為縮圖區在本例中圖 6-1 為台灣省之縮圖，用來選取詳細顯示的縣市，而縮圖本身可以由下方之 Track Bar 控制顯示比例。左半部為詳細地圖顯示區，用以顯示詳細的各個縣市及主題層之疊合地圖，大小可以由右上方之 Track Bar 控制顯示比例，目前所顯示之

區域為台北縣。而縮圖區及詳細地圖顯示區均可以用滑鼠在上面拖曳以便上下左右平移。

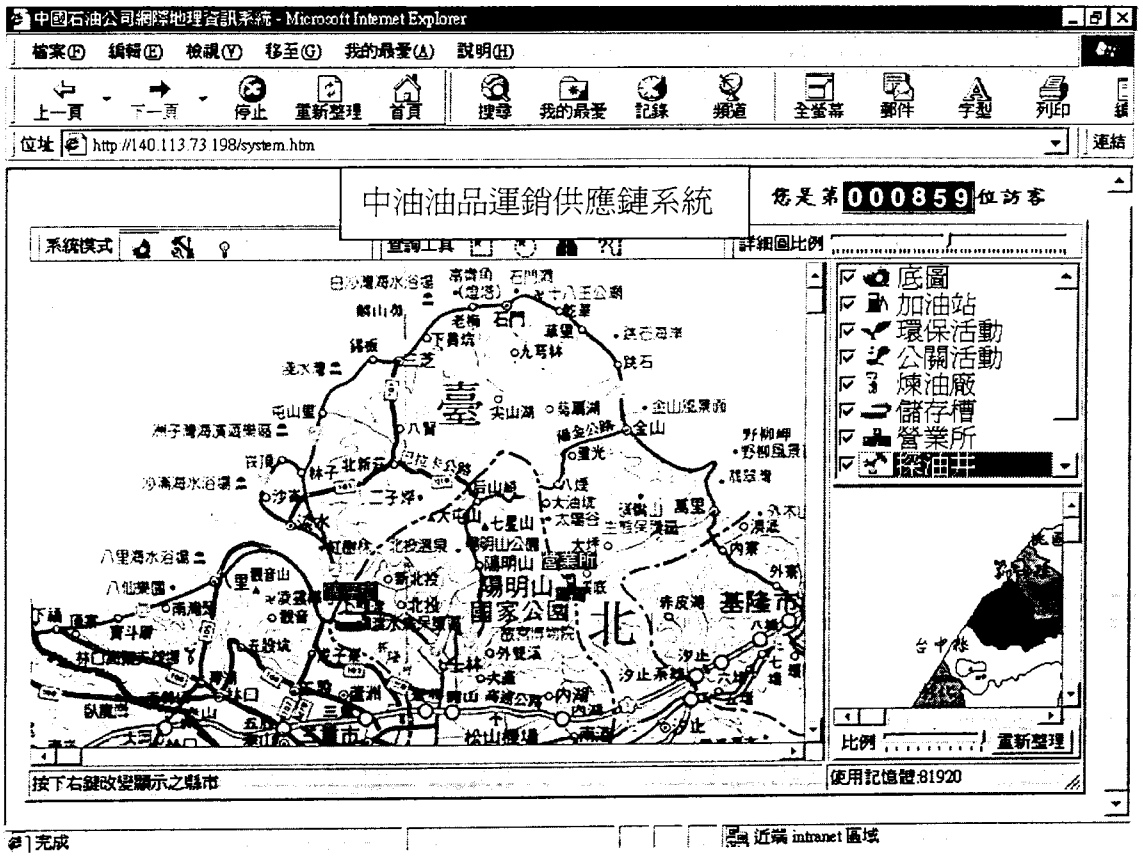


圖 6-2 中油油品運銷供應鏈管理系統

只要在右上方圖層控制視窗內的方框打勾就會動態的將主題層疊合於底圖上，圖 6-2 是將有關圖層打勾，所以左半部之台北縣地圖顯示區就將相關主題層(煉油廠、儲油槽、加油站等)疊合，且共同顯示於詳細地圖顯示區。只要在圖 6-2 上的任一據點上方按兩下滑鼠左鍵即可查詢據點之詳細資料之畫面，如圖 6-3。

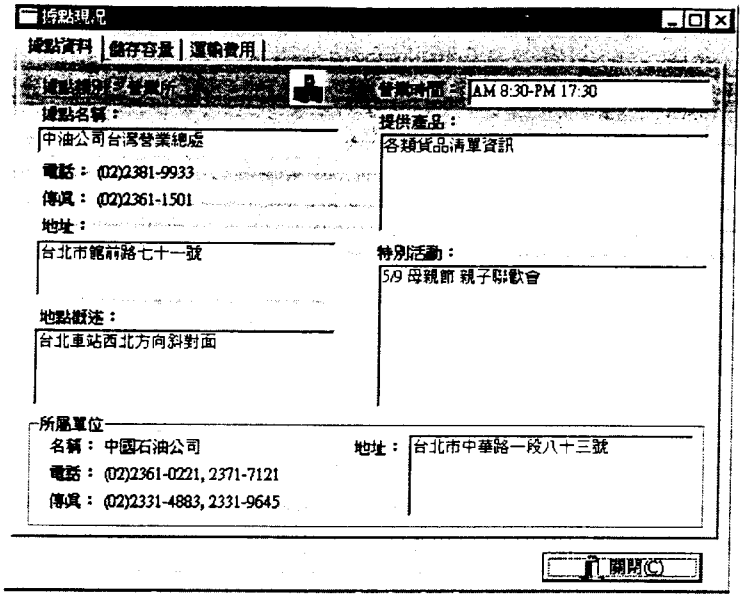


圖 6-3　中油油品運銷供應鏈管理系統

6.2 以圖形區塊查詢

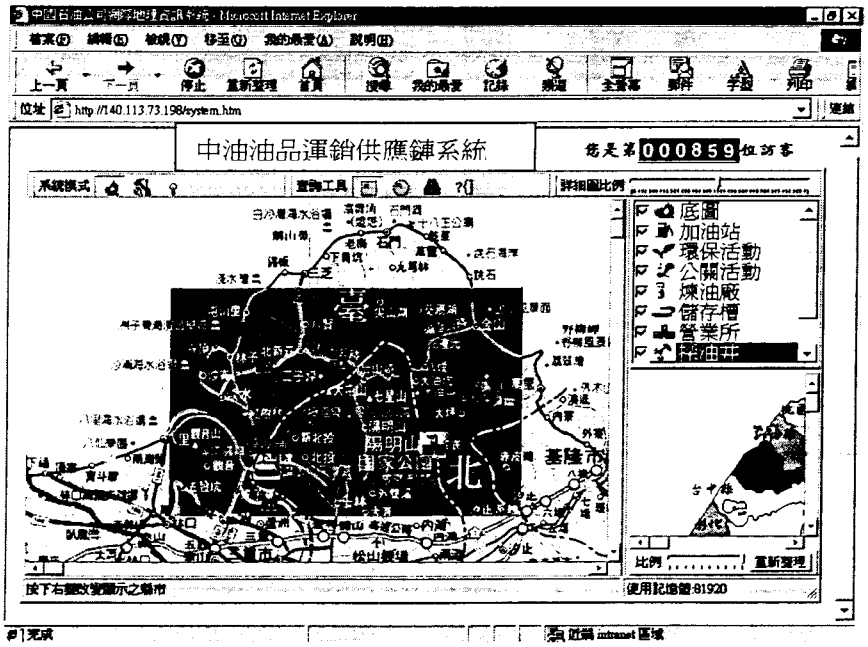


圖 6-4　中油油品運銷供應鏈管理系統

要使用圖形區塊查詢功能，只要將查詢工具中的方形或圓形圖樣（Icon）壓下。圖 4-4 分別是壓下矩形查詢後，用滑鼠在詳細地圖顯示區拖曳所形成之反白區塊。此時只要放開滑鼠則在此反白區塊內所有被勾選之圖層資訊(右上方圖層控制視窗內的方框中有打勾者)，皆會被搜尋到而產生圖 4-5 之報表。在圖 4-5 之報表中的任一列用滑鼠點一下(Click)，會將詳細地圖顯示區域自動載入有包含所點選據點之底圖。

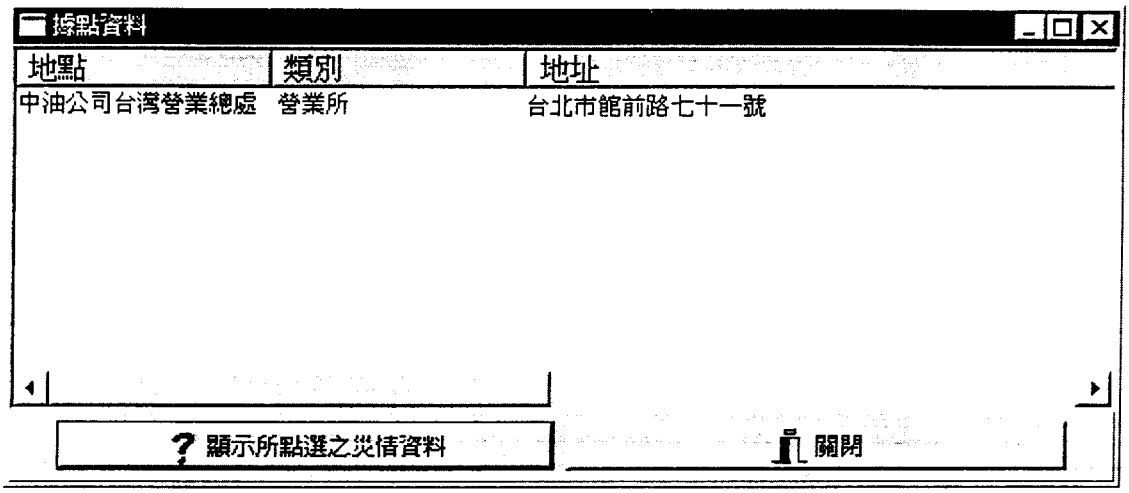


圖 6-5 中油油品運銷供應鏈管理系統

6.3 以保留字查詢

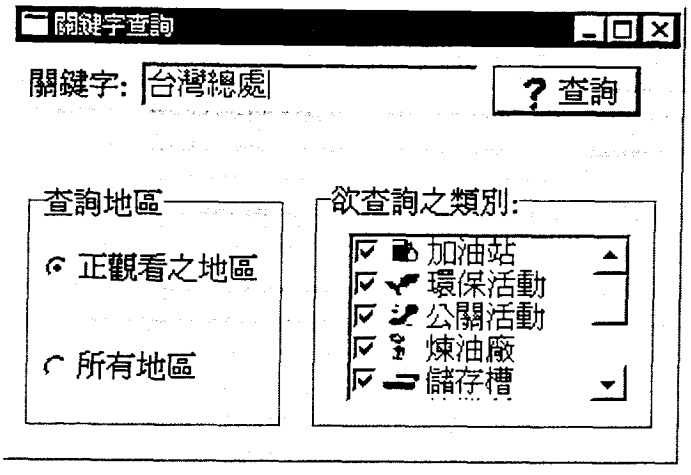


圖 6-6　中油油品運銷供應鏈管理系統

查詢方式除了上述圖形查詢法，也可以用傳統保留字查詢法。只要在主畫面上的查詢工具列壓下最右邊的圖樣即出現圖 6-6 之畫面。其中關鍵字欄位只要輸入欲查詢之關鍵字(查詢範圍可包含名稱、地址及描述)，其查詢之範圍可包含正在觀看之地區(主畫面左半部詳細地圖顯示區正顯示之區域)或所有地區，而查詢之圖層更可包含所選擇之圖層。一旦查詢完畢，即出現產生圖 6-5 之報表。在圖 6-5 之報表中的任一列用滑鼠點一下(Click)，會將詳細地圖顯示區域自動載入有包含所點選據點之底圖。

6.4 遠程分散式編修模式

要進入編修模式前，首先要通過身分確認，以決定編修之權限。首先在主畫面上方左邊的系統模式列第二個按鍵壓下(如圖 6-8)即出現圖 6-9 之畫面，在上面輸入正確之使用者代號及密碼可以通過身分確認，並依帳號不同取得不同編修之權限。一旦通過身分確認後，主畫面上方中間的查詢工具列即會被編修工具列取代，如圖 6-7。

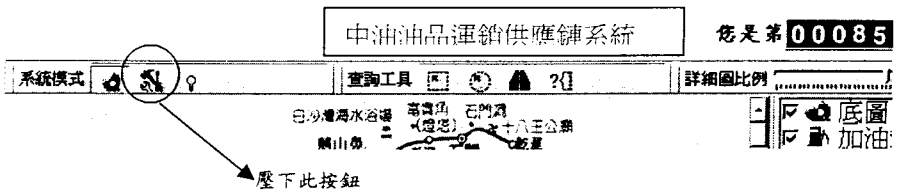


圖 6-7 中油油品運銷供應鏈管理系統

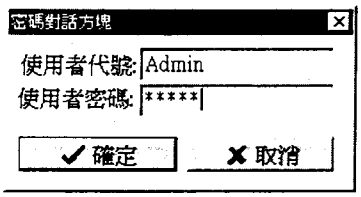


圖 6-8 中油油品運銷供應鏈管理系統

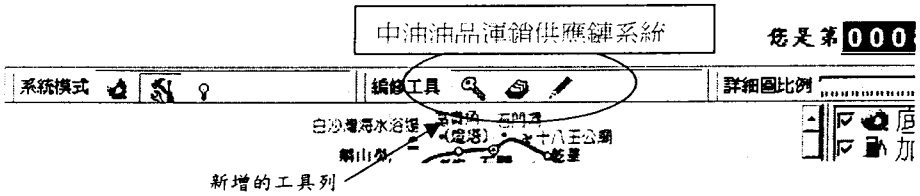


圖 6-9 中油油品運銷供應鏈管理系統

6.5人員權限設定

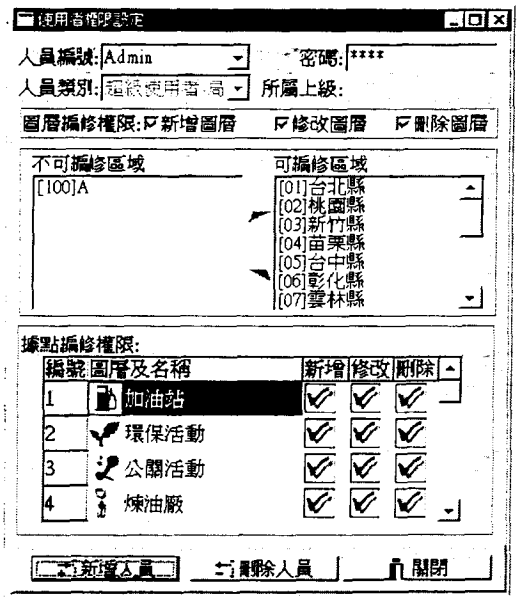


圖 6-10　中油油品運銷供應鏈管理系統

要設定每一個編修人員之權限，則是在通過身分確認的情況下，按下編修工具列的第一個按鍵壓下即出現圖 6-10 之畫面。此畫面用以對每一個編修人員之權限進行設定。此畫面可以設定編修人員是否可編修圖層，決定哪些底圖(縣市)可編修，及各個圖層是否可以新增、修改或刪除據點。再進入此畫面時，只能查詢到權限比自己小之使用人員。

6.6圖層編修

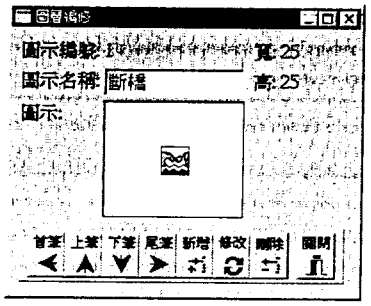


圖 6-11　中油油品運銷供應鏈管理系統

要編修圖層的種類，則是按下編修工具列的第二個按鍵，此時會出現圖 6-11 之畫面。此畫面用以新增、修改及刪除每一個圖層之資料。

6.7 據點編修

要新增、修改或刪除個別據點，則須先按下編修工具列的第三個按鍵(如圖 6-12)，以進入據點編修模式。此時主畫面左邊會出現一個由圖層圖示所組成之據點工具列。若主圖層控制視窗中的圖示方框內有打勾，左邊工具列之圖示即會顯示出來(如圖 6-13)。而據點編修可分為新增、修改及刪除其說明如下：

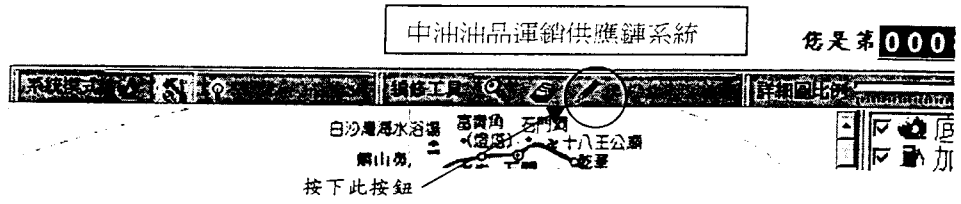


圖 6-12 中油油品運銷供應鏈管理系統

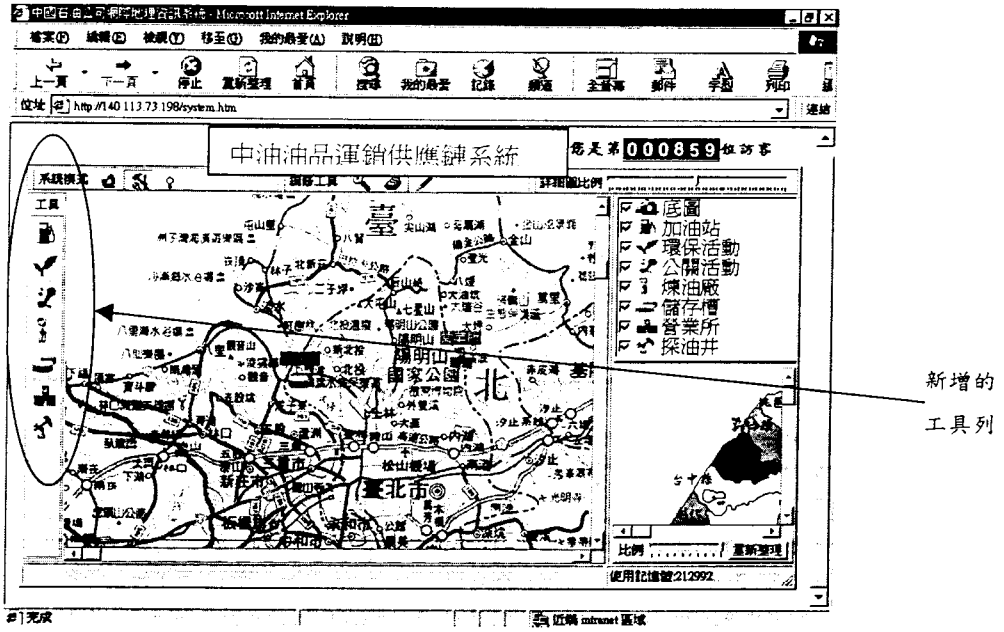


圖 6-13 中油油品運銷供應鏈管理系統

6.8 新增據點

一旦左邊工具列之圖示(淡紫色)顯示出來，即表示此圖層據點可以新增，而新增之方式為「壓下此圖示拖曳到詳細地圖顯示區的正確位置然後放開滑鼠的按鍵」，新的據點在拖到正確位置後，即會出現圖 6-14 之畫面，用以確認是否要新增據點，確認後便會出現圖 6-15 之畫面(可編修欄位為黃色)即可開始新增作業。其中

所屬單位一欄會先提供選單以供選擇，若所屬機關不在選單內，則可按下新增鍵以新增一個上級機關（如圖 6-16）。

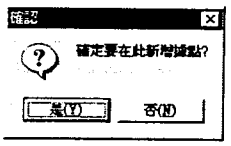


圖 6-14 中油油品運銷供應鏈管理系統

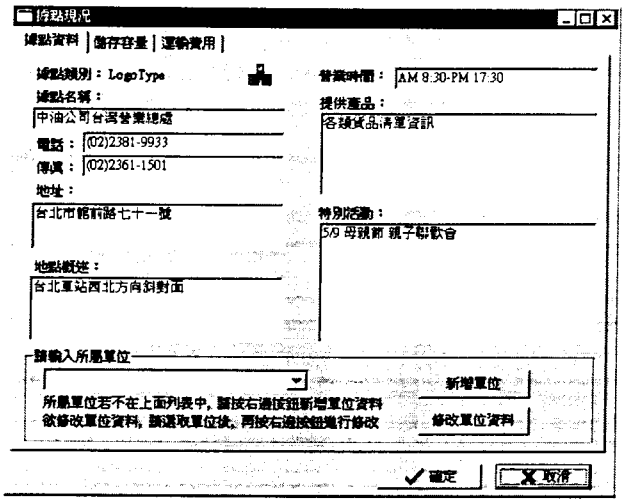


圖 6-15 中油油品運銷供應鏈管理系統

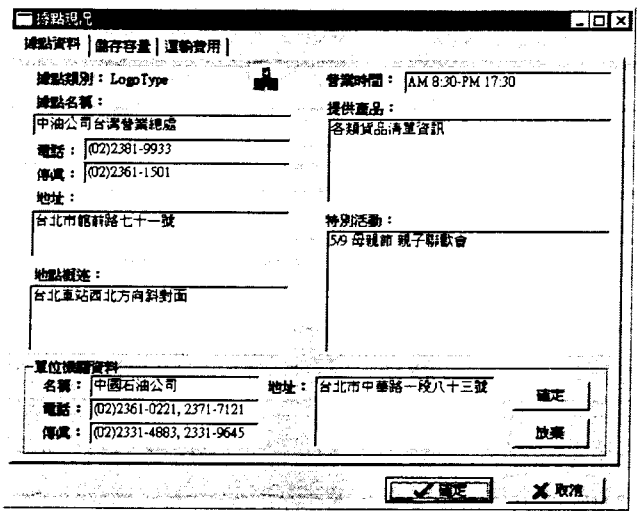


圖 6-16 中油油品運銷供應鏈管理系統

6.9修改據點

欲移動據點位置時，只要在詳細地圖顯示區上的據點圖示壓下且拖曳，直到目的地才放開滑鼠。即會顯示圖畫面用以確認是否將據點移動到此，一旦確定後表示移動到此新位置。欲修改據點內容時，只要在詳細地圖顯示區上的據點圖示壓下滑鼠右鍵，即會顯示圖 6-17 之畫面。此時選擇修改即出現圖 6-18 之畫面，用以進行修改。

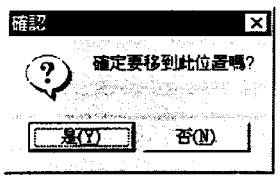


圖 6-17、6-18 中油油品運銷供應鏈管理系統

6.10 刪除據點

欲刪除據點時，只要在詳細地圖顯示區上的據點圖示壓下滑鼠右鍵，即會顯示圖 6-19 之畫面。此時選擇刪除即完成刪除作業。

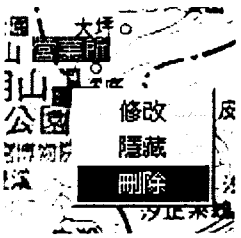


圖 6-19 中油油品運銷供應鏈管理系統

七、結論與討論

本研究計畫達到的成果為：

1.發展中油公司油品運銷供應鏈管理系統。

2.將供應鏈數學模式、網際地理資訊系統、全球定位系統、多階層分散式網路技術與資料庫結合。

3.將發展的成果架設於網站上，供中油公司各相關單位使用。

目前發展瀏覽器的廠商皆在開發功能更強，應用範圍更廣的瀏覽器。而超文件建構語言(HTML)標準的版本也由 1.0 提升至 3.0，甚至有廠商推出動態式超文件建構語言(Dynamic HTML)，加強客戶端瀏覽器的功能基本上已成為各方的共識。

雖然我們解決了技術上的問題，但若要發展商業用途的運銷供應鏈系統的過程中，除了上述問題外，尚需探討以下課題：

（1）底圖使用

底圖來源:需取得準確又詳細的各種比例尺地圖。

底圖授權:可探政府印行地圖或民間印行地圖二種。但政府發行之地圖，迄今尚未准上網路；民間有部份底圖品質甚佳，可能經其授權後上網。

底圖更新:需有單位實際管理及更新底圖。但電子地圖的建立通常需

要很多時間，如何有效掌控其時效，也是有待討論。

(2)經營管理

系統經營方式：系統伺服器(SERVER)的經營方式有三:1.政府管理；2.民間自辦；3.協會代辦。初期可採取資訊使用者付費方式經營，再逐步擴展為會員年金制，向會員收取固定費用。

Inter 與 Intra：在 SCM 未來的發展，也應考慮在一般網路與在公司內部網路的使用情形，尤其在公司內有許多資訊是具有保密性，不允許一般人任意取得，其安全分級的資料保護則日益重要。

伺服端的管理：SCM 可只維護基本的底圖，而大部份加值圖的資料及超連結(HYPERLINK)則由一般公司提供，而這樣的架構除了減少伺服器維護成本，更可以達到資料即時更新的效果。

運銷供應鏈管理系統的應用發展除了需提昇資訊技術及開發不同領域的應用外，擴大使用層次及使用對象將促進油品運銷 SCM 應用普及化及多樣化。為達此目的，油品運銷供應鏈管理系統將成為中油公司推動的主要方向。在國內目前由中油機構大力推動下，未來將有更多商業機構及其它相關企業逐漸將 GIS 與 SCM 系統予以整合，一方面滿足業務上圖形資料應用管理需求外，更可經由網際網路創造新的合作契機。例如國外油品供應商與中油之間的 B2B。訊系統應用所能

帶來之效益。於此期盼政府機構能優先建立 SCM 基礎環境及資料外，也展望藉由 SCM 使各行各業以至於一般人皆可充分的享有便捷低廉之 SCM 使用環境。

八、參考文獻

[1] 黎漢林等，供應鏈管理與決策，儒林圖書，2000 年 9 月。

[2] 黎漢林等，網際地理資訊系統，儒林圖書，1999 年 2 月。

[2] CAD 與自動化，1995 年 2 月。

[3] 周建成，從三層架構談 Internet 發展區趨勢，物件導向雜誌 p16-28，1995 年 5/6 月。

[4] 施保旭，1995，地理資訊系統，儒林

[5] Rober Laurini and Derek Thompson, 1994, Fundamentals of spatial information systems, Academic Press.

[6] URL:http://tiger.census.gov
[7] URL:http://www.xerox.com/map
[8] URL:http://www.esd.ornl.gov
[9] URL:http://ingis.can.purdue.edu/INGIS/mission.html
[10] URL:http://pasture.ecn.purdue.edu
[11] URL:http://www.regis.berkeley.edu
[12] URL:http://www.regis.berkeley.edu/grasslinks
[13] URL:http://epawww.ciesin.org
[14] URL:http://alexandria.sdc.ucsb.edu
[15] URL:http://www.bp.ntu.edu.tw
[16] URL:http://www.ttw.com.tw
[17] http://www.cpc.com.tw
[18] Wayne L. Winston, 1987, Operations Research: Applications and Algorithms, PWS-KENT.

[19] 莊志諒，1988，配送網路之設計研究，國立交通大學運輸研究所碩士論文。

[20] 陳昭堯，1988，石油產品最適儲運體系之研究，國立交通大學運輸研究所碩士論文。

附錄

供應鏈系統設計程式碼

```
constructor TGIFItem.Create(GIFImage: TGIFImage);
begin
  inherited Create;

  FGIFImage := GIFImage;
end;

procedure TGIFItem.Warning(Severity: TGIFSeverity; Message: string);
begin
  FGIFImage.Warning(self, Severity, Message);
end;

function TGIFItem.GetVersion: TGIFVersion;
begin
  Result := gv87a;
end;

procedure TGIFItem.LoadFromFile(const Filename: string);
var
  Stream: TStream;
begin
  Stream := TFileStream.Create(Filename, fmOpenRead OR fmShareDenyWrite);
  try
    LoadFromStream(Stream);
  finally
    Stream.Free;
  end;
end;

procedure TGIFItem.SaveToFile(const Filename: string);
var
  Stream: TStream;
begin
  Stream := TFileStream.Create(Filename, fmCreate);
  try
    SaveToStream(Stream);
  finally
    Stream.Free;
  end;
end;

////////////////////////////////////////////////////////////////////////////////
//
//                       TGIFList
//
////////////////////////////////////////////////////////////////////////////////
constructor TGIFList.Create(Image: TGIFImage);
begin
  inherited Create;
  FImage := Image;
  FItems := TList.Create;
end;

destructor TGIFList.Destroy;
begin
  Clear;
  FItems.Free;
  inherited Destroy;
end;

function TGIFList.GetItem(Index: Integer): TGIFItem;
begin
  Result := TGIFItem(FItems[Index]);
end;

procedure TGIFList.SetItem(Index: Integer; Item: TGIFItem);
begin
  FItems[Index] := Item;
end;

function TGIFList.GetCount: Integer;
begin
  Result := FItems.Count;
end;

function TGIFList.Add(Item: TGIFItem): Integer;
begin
```

```
    Result := FItems.Add(Item);
  end;

  procedure TGIFList.Clear;
  begin
    while (FItems.Count > 0) do
      Delete(0);
  end;

  procedure TGIFList.Delete(Index: Integer);
  var
    Item                    : TGIFItem;
  begin
    Item := TGIFItem(FItems[Index]);
    // Delete before item is destroyed to avoid recursion
    FItems.Delete(Index);
    Item.Free;
  end;

  procedure TGIFList.Exchange(Index1, Index2: Integer);
  begin
    FItems.Exchange(Index1, Index2);
  end;

  function TGIFList.First: TGIFItem;
  begin
    Result := TGIFItem(FItems.First);
  end;

  function TGIFList.IndexOf(Item: TGIFItem): Integer;
  begin
    Result := FItems.IndexOf(Item);
  end;

  procedure TGIFList.Insert(Index: Integer; Item: TGIFItem);
  begin
    FItems.Insert(Index, Item);
  end;

  function TGIFList.Last: TGIFItem;
  begin
    Result := TGIFItem(FItems.Last);
  end;

  procedure TGIFList.Move(CurIndex, NewIndex: Integer);
  begin
    FItems.Move(CurIndex, NewIndex);
  end;

  function TGIFList.Remove(Item: TGIFItem): Integer;
  begin
    // Note: TGIFList.Remove must not destroy item
    Result := FItems.Remove(Item);
  end;

  procedure TGIFList.SaveToStream(Stream: TStream);
  var
    i                       : integer;
  begin
    for i := 0 to FItems.Count-1 do
      TGIFItem(FItems[i]).SaveToStream(Stream);
  end;

  procedure TGIFList.Warning(Severity: TGIFSeverity; Message: string);
  begin
    Image.Warning(self, Severity, Message);
  end;

  ////////////////////////////////////////////////////////////////////////////////
  //
  //                       TGIFGlobalColorMap
  //
  ////////////////////////////////////////////////////////////////////////////////
  type
    TGIFGlobalColorMap = class(TGIFColorMap)
    private
      FHeader      : TGIFHeader;
```

```
    protected
      procedure Warning(Severity: TGIFSeverity; Message: string); override;
      procedure BuildHistogram(var Histogram: TColormapHistogram); override;
      procedure MapImages(var Map: TColormapReverse); override;
    public
      constructor Create(HeaderItem: TGIFHeader);
      function Optimize: boolean; override;
      procedure Changed; override;
    end;

  constructor TGIFGlobalColorMap.Create(HeaderItem: TGIFHeader);
  begin
    Inherited Create;
    FHeader := HeaderItem;
  end;

  procedure TGIFGlobalColorMap.Warning(Severity: TGIFSeverity; Message: string);
  begin
    FHeader.Image.Warning(self, Severity, Message);
  end;

  procedure TGIFGlobalColorMap.BuildHistogram(var Histogram: TColormapHistogram);
  var
    Pixel                 ,
    LastPixel             : PChar;
    i                     : integer;
  begin
    (*
    ** Init histogram
    *)
    for i := 0 to Count-1 do
    begin
      Histogram[i].Index := i;
      Histogram[i].Count := 0;
    end;

    for i := 0 to FHeader.Image.Images.Count-1 do
      if (FHeader.Image.Images[i].ActiveColorMap = self) then
      begin
        Pixel := FHeader.Image.Images[i].Data;
        LastPixel := Pixel + FHeader.Image.Images[i].Width * FHeader.Image.Images[i].Height;

        (*
        ** Sum up usage count for each color
        *)
        while (Pixel < LastPixel) do
        begin
          inc(Histogram[ord(Pixel^)].Count);
          inc(Pixel);
        end;
      end;
  end;

  procedure TGIFGlobalColorMap.MapImages(var Map: TColormapReverse);
  var
    Pixel                 ,
    LastPixel             : PChar;
    i                     : integer;
  begin
    for i := 0 to FHeader.Image.Images.Count-1 do
      if (FHeader.Image.Images[i].ActiveColorMap = self) then
      begin
        Pixel := FHeader.Image.Images[i].Data;
        LastPixel := Pixel + FHeader.Image.Images[i].Width * FHeader.Image.Images[i].Height;

        (*
        **  Reorder all pixel to new map
        *)
        while (Pixel < LastPixel) do
        begin
          Pixel^ := chr(Map[ord(Pixel^)]);
          inc(Pixel);
        end;

        (*
        **  Reorder transparent colors
        *)
```

```
        if (FHeader.Image.Images[i].Transparent) then
          FHeader.Image.Images[i].GraphicControlExtension.TransparentColorIndex :=
            Map[FHeader.Image.Images[i].GraphicControlExtension.TransparentColorIndex];
      end;
  end;

  function TGIFGlobalColorMap.Optimize: boolean;
  begin
    { Optimize with first image, Remove unused colors if only one image }
    if (FHeader.Image.Images.Count > 0) then
      Result := DoOptimize
    else
      Result := False;
  end;

  procedure TGIFGlobalColorMap.Changed;
  begin
    FHeader.Image.Palette := 0;
  end;

  ////////////////////////////////////////////////////////////////////////////////
  //
  //                       TGIFHeader
  //
  ////////////////////////////////////////////////////////////////////////////////
  constructor TGIFHeader.Create(GIFImage: TGIFImage);
  begin
    inherited Create(GIFImage);
    FColorMap := TGIFGlobalColorMap.Create(self);
    Clear;
  end;

  destructor TGIFHeader.Destroy;
  begin
    FColorMap.Free;
    inherited Destroy;
  end;

  procedure TGIFHeader.Clear;
  begin
    FColorMap.Clear;
    FLogicalScreenDescriptor.ScreenWidth := 0;
    FLogicalScreenDescriptor.ScreenHeight := 0;
    FLogicalScreenDescriptor.PackedFields := 0;
    FLogicalScreenDescriptor.BackgroundColorIndex := 0;
    FLogicalScreenDescriptor.AspectRatio := 0;
  end;

  procedure TGIFHeader.Assign(Source: TPersistent);
  begin
    if (Source is TGIFHeader) then
    begin
      ColorMap.Assign(TGIFHeader(Source).ColorMap);
      FLogicalScreenDescriptor := TGIFHeader(Source).FLogicalScreenDescriptor;
    end else
    if (Source is TGIFColorMap) then
    begin
      Clear;
      ColorMap.Assign(TGIFColorMap(Source));
    end else
      inherited Assign(Source);
  end;

  type
    TGIFHeaderRec = packed record
      Signature: array[0..2] of char; { contains 'GIF' }
      Version: TGIFVersionRec;    { '87a' or '89a' }
    end;

  const
    { logical screen descriptor packed field masks }
    lsdGlobalColorTable   = $80;          { set if global color table follows L.S.D. }
    lsdColorResolution    = $70;          { Color resolution - 3 bits }
    lsdSort               = $08;          { set if global color table is sorted - 1 bit }
    lsdColorTableSize     = $07;          { size of global color table - 3 bits }
                                          { Actual size = 2^value+1    - value is 3 bits }
  procedure TGIFHeader.Prepare;
```

```
var
  pack                  : BYTE;
begin
  Pack := $00;
  if (ColorMap.Count > 0) then
  begin
    Pack := lsdGlobalColorTable;
    if (ColorMap.Optimized) then
      Pack := Pack OR lsdSort;
  end;
  // Note: The SHL below was SHL 5 in the original source, but that looks wrong
  Pack := Pack OR ((Image.ColorResolution SHL 4) AND lsdColorResolution);
  Pack := Pack OR ((Image.BitsPerPixel-1) AND lsdColorTableSize);
  FLogicalScreenDescriptor.PackedFields := Pack;
end;

procedure TGIFHeader.SaveToStream(Stream: TStream);
var
  GifHeader             : TGIFHeaderRec;
  v                     : TGIFVersion;
begin
  v := Image.Version;
  if (v = gvUnknown) then
    Error(sBadVersion);

  GifHeader.Signature := 'GIF';
  GifHeader.Version := GIFVersions[v];

  Prepare;
  Stream.Write(GifHeader, sizeof(GifHeader));
  Stream.Write(FLogicalScreenDescriptor, sizeof(FLogicalScreenDescriptor));
  if (FLogicalScreenDescriptor.PackedFields AND lsdGlobalColorTable = lsdGlobalColorTable) ther
    ColorMap.SaveToStream(Stream);
end;

procedure TGIFHeader.LoadFromStream(Stream: TStream);
var
  GifHeader             : TGIFHeaderRec;
  ColorCount            : integer;
  Position              : integer;
begin
  Position := Stream.Position;

  ReadCheck(Stream, GifHeader, sizeof(GifHeader));
  if (uppercase(GifHeader.Signature) <> 'GIF') then
  begin
    // Attempt recovery in case we are reading a GIF stored in a form by rxLib
    Stream.Position := Position;
    // Seek past size stored in stream
    Stream.Seek(sizeof(longInt), soFromCurrent);
    // Attempt to read signature again
    ReadCheck(Stream, GifHeader, sizeof(GifHeader));
    if (uppercase(GifHeader.Signature) <> 'GIF') then
      Error(sBadSignature);
  end;

  ReadCheck(Stream, FLogicalScreenDescriptor, sizeof(FLogicalScreenDescriptor));

  if (FLogicalScreenDescriptor.PackedFields AND lsdGlobalColorTable = lsdGlobalColorTable) ther
  begin
    ColorCount := 2 SHL (FLogicalScreenDescriptor.PackedFields AND lsdColorTableSize);
    if (ColorCount < 2) or (ColorCount > 256) then
      Error(sScreenBadColorSize);
    ColorMap.LoadFromStream(Stream, ColorCount)
  end else
    ColorMap.Clear;
end;

function TGIFHeader.GetVersion: TGIFVersion;
begin
  if (FColorMap.Optimized) or (AspectRatio <> 0) then
    Result := gv89a
  else
    Result := inherited GetVersion;
end;

function TGIFHeader.GetBackgroundColor: TColor;
```

```
begin
  Result := FColorMap[BackgroundColorIndex];
end;

procedure TGIFHeader.SetBackgroundColor(Color: TColor);
begin
  BackgroundColorIndex := FColorMap.AddUnique(Color);
end;

procedure TGIFHeader.SetBackgroundColorIndex(Index: BYTE);
begin
  if ((Index >= FColorMap.Count) and (FColorMap.Count > 0)) then
  begin
    Warning(gsWarning, sBadColorIndex);
    Index := 0;
  end;
  FLogicalScreenDescriptor.BackgroundColorIndex := Index;
end;

function TGIFHeader.GetBitsPerPixel: integer;
begin
  Result := FColorMap.BitsPerPixel;
end;

function TGIFHeader.GetColorResolution: integer;
begin
  Result := FColorMap.BitsPerPixel-1;
end;

////////////////////////////////////////////////////////////////////////////////
//
//                      TGIFLocalColorMap
//
////////////////////////////////////////////////////////////////////////////////
type
  TGIFLocalColorMap = class(TGIFColorMap)
  private
    FSubImage           : TGIFSubImage;
  protected
    procedure Warning(Severity: TGIFSeverity; Message: string); override;
    procedure BuildHistogram(var Histogram: TColormapHistogram); override;
    procedure MapImages(var Map: TColormapReverse); override;
  public
    constructor Create(SubImage: TGIFSubImage);
    function Optimize: boolean; override;
    procedure Changed; override;
  end;

constructor TGIFLocalColorMap.Create(SubImage: TGIFSubImage);
begin
  Inherited Create;
  FSubImage := SubImage;
end;

procedure TGIFLocalColorMap.Warning(Severity: TGIFSeverity; Message: string);
begin
  FSubImage.Image.Warning(self, Severity, Message);
end;

procedure TGIFLocalColorMap.BuildHistogram(var Histogram: TColormapHistogram);
var
  Pixel                ,
  LastPixel            : PChar;
  i                    : integer;
begin
  Pixel := FSubImage.Data;
  LastPixel := Pixel + FSubImage.Width * FSubImage.Height;

  (*
  ** Init histogram
  *)
  for i := 0 to Count-1 do
  begin
    Histogram[i].Index := i;
    Histogram[i].Count := 0;
  end;
```

```
    (*
    ** Sum up usage count for each color
    *)
    while (Pixel < LastPixel) do
    begin
      inc(Histogram[ord(Pixel^)].Count);
      inc(Pixel);
    end;
  end;

  procedure TGIFLocalColorMap.MapImages(var Map: TColormapReverse);
  var
    Pixel                ,
    LastPixel            : PChar;
  begin
    Pixel := FSubImage.Data;
    LastPixel := Pixel + FSubImage.Width * FSubImage.Height;

    (*
    **  Reorder all pixel to new map
    *)
    while (Pixel < LastPixel) do
    begin
      Pixel^ := chr(Map[ord(Pixel^)]);
      inc(Pixel);
    end;

    (*
    **  Reorder transparent colors
    *)
    if (FSubImage.Transparent) then
      FSubImage.GraphicControlExtension.TransparentColorIndex :=
        Map[FSubImage.GraphicControlExtension.TransparentColorIndex];
  end;

  function TGIFLocalColorMap.Optimize: boolean;
  begin
    Result := DoOptimize;
  end;

  procedure TGIFLocalColorMap.Changed;
  begin
    FSubImage.Palette := 0;
  end;


  ////////////////////////////////////////////////////////////////////////////////
  //
  //                      LZW Decoder
  //
  ////////////////////////////////////////////////////////////////////////////////
  const
    GIFCodeBits          = 12;                    // Max number of bits per GIF token code
    GIFCodeMax           = (1 SHL GIFCodeBits)-1;// Max GIF token code
                                                  // 12 bits = 4095
    StackSize            = (2 SHL GIFCodeBits);  // Size of decompression stack
    TableSize            = (1 SHL GIFCodeBits);  // Size of decompression table

  procedure TGIFSubImage.Decompress(Stream: TStream);
  var
    table0               : array[0..TableSize-1] of integer;
    table1               : array[0..TableSize-1] of integer;
    firstcode, oldcode   : integer;
    buf                  : array[0..257] of BYTE;

    Dest                 : PChar;
    v                    ,
    xpos, ypos, pass     : integer;

    stack                : array[0..StackSize-1] of integer;
    Source                     : ^integer;
    BitsPerCode          : integer;              // number of CodeTableBits/code
    InitialBitsPerCode   : BYTE;

    MaxCode              : integer;              // maximum code, given BitsPerCode
    MaxCodeSize          : integer;
    ClearCode            : integer;              // Special code to signal "Clear table"
```

```
  EOFCode               : integer;          // Special code to signal EOF
  step                  : integer;
  i                     : integer;

  StartBit              ,                   // Index of bit buffer start
  LastBit               ,                   // Index of last bit in buffer
  LastByte              : integer;          // Index of last byte in buffer
  get_done              ,
  return_clear          ,
  ZeroBlock             : boolean;
  ClearValue            : BYTE;
{$ifdef DEBUG DECOMPRESSPERFORMANCE}
  TimeStartDecompress   ,
  TimeStopDecompress    : DWORD;
{$endif}

  function nextCode(BitsPerCode: integer): integer;
  const
    masks: array[0..15] of integer =
      ($0000, $0001, $0003, $0007,
       $000f, $001f, $003f, $007f,
       $00ff, $01ff, $03ff, $07ff,
       $0fff, $1fff, $3fff, $7fff);
  var
    StartIndex, EndIndex              : integer;
    ret                 : integer;
    EndBit              : integer;
    count               : BYTE;
  begin
    if (return_clear) then
    begin
      return_clear := False;
      Result := ClearCode;
      exit;
    end;

    EndBit := StartBit + BitsPerCode;

    if (EndBit >= LastBit) then
    begin
      if (get_done) then
      begin
        if (StartBit >= LastBit) then
          Warning(gsWarning, sDecodeTooFewBits);
        Result := -1;
        exit;
      end;
      buf[0] := buf[LastByte-2];
      buf[1] := buf[LastByte-1];

      if (Stream.Read(count, 1) <> 1) then
      begin
        Result := -1;
        exit;
      end;
      if (count = 0) then
      begin
        ZeroBlock := True;
        get_done := TRUE;
      end else
      begin
        // Handle premature end of file
        if (Stream.Size - Stream.Position < Count) then
        begin
          Warning(gsWarning, sOutOfData);
          // Not enough data left - Just read as much as we can get
          Count := Stream.Size - Stream.Position;
        end;
        if (Count <> 0) then
          ReadCheck(Stream, Buf[2], Count);
      end;

      LastByte := 2 + count;
      StartBit := (StartBit - LastBit) + 16;
      LastBit := LastByte * 8;

      EndBit := StartBit + BitsPerCode;
```

```
  end;

  EndIndex := EndBit DIV 8;
  StartIndex := StartBit DIV 8;

  ASSERT(StartIndex <= high(buf), 'StartIndex too large');
  if (StartIndex = EndIndex) then
    ret := buf[StartIndex]
  else
    if (StartIndex + 1 = EndIndex) then
      ret := buf[StartIndex] OR (buf[StartIndex+1] SHL 8)
    else
      ret := buf[StartIndex] OR (buf[StartIndex+1] SHL 8) OR (buf[StartIndex+2] SHL 16);

  ret := (ret SHR (StartBit AND $0007)) AND masks[BitsPerCode];

  Inc(StartBit, BitsPerCode);

  Result := ret;
end;

function NextLZW: integer;
var
  code, incode        : integer;
  i                   : integer;
  b                   : BYTE;
begin
  code := nextCode(BitsPerCode);
  while (code >= 0) do
  begin
    if (code = ClearCode) then
    begin
      ASSERT(ClearCode < TableSize, 'ClearCode too large');
      for i := 0 to ClearCode-1 do
      begin
        table0[i] := 0;
        table1[i] := i;
      end;
      for i := ClearCode to TableSize-1 do
      begin
        table0[i] := 0;
        table1[i] := 0;
      end;
      BitsPerCode := InitialBitsPerCode+1;
      MaxCodeSize := 2 * ClearCode;
      MaxCode := ClearCode + 2;
      Source := @stack;
      repeat
        firstcode := nextCode(BitsPerCode);
        oldcode := firstcode;
      until (firstcode <> ClearCode);

      Result := firstcode;
      exit;
    end;
    if (code = EOFCode) then
    begin
      Result := -2;
      if (ZeroBlock) then
        exit;
      // Eat rest of data blocks
      if (Stream.Read(b, 1) <> 1) then
        exit;
      while (b <> 0) do
      begin
        Stream.Seek(b, soFromCurrent);
        if (Stream.Read(b, 1) <> 1) then
          exit;
      end;
      exit;
    end;

    incode := code;

    if (code >= MaxCode) then
    begin
      Source^ := firstcode;
```

```
          Inc(Source);
          code := oldcode;
        end;

        ASSERT(Code < TableSize, 'Code too large');
        while (code >= ClearCode) do
        begin
          Source^ := table1[code];
          Inc(Source);
          if (code = table0[code]) then
            Error(sDecodeCircular);
          code := table0[code];
          ASSERT(Code < TableSize, 'Code too large');
        end;

        firstcode := table1[code];
        Source^ := firstcode;
        Inc(Source);

        code := MaxCode;
        if (code <= GIFCodeMax) then
        begin
          table0[code] := oldcode;
          table1[code] := firstcode;
          Inc(MaxCode);
          if ((MaxCode >= MaxCodeSize) and (MaxCodeSize <= GIFCodeMax)) then
          begin
            MaxCodeSize := MaxCodeSize * 2;
            Inc(BitsPerCode);
          end;
        end;

        oldcode := incode;

        if (longInt(Source) > longInt(@stack)) then
        begin
          Dec(Source);
          Result := Source^;
          exit;
        end
      end;
      Result := code;
    end;

    function readLZW: integer;
    begin
      if (longInt(Source) > longInt(@stack)) then
      begin
        Dec(Source);
        Result := Source^;
      end else
        Result := NextLZW;
    end;

  begin
    NewImage;

    // Clear image data in case decompress doesn't complete
    if (Transparent) then
      // Clear to transparent color
      ClearValue := GraphicControlExtension.GetTransparentColorIndex
    else
      // Clear to first color
      ClearValue := 0;

    FillChar(FData^, FDataSize, ClearValue);

  {$ifdef DEBUG DECOMPRESSPERFORMANCE}
    TimeStartDecompress := timeGetTime;
  {$endif}

    (*
    ** Read initial code size in bits from stream
    *)
    if (Stream.Read(InitialBitsPerCode, 1) <> 1) then
      exit;
```

```
(*
**  Initialize the Compression routines
*)
BitsPerCode := InitialBitsPerCode + 1;
ClearCode := 1 SHL InitialBitsPerCode;
EOFCode := ClearCode + 1;
MaxCodeSize := 2 * ClearCode;
MaxCode := ClearCode + 2;

StartBit := 0;
LastBit := 0;
LastByte := 2;

ZeroBlock := False;
get_done := False;
return_clear := TRUE;

Source := @stack;

try
  if (Interlaced) then
  begin
    ypos := 0;
    pass := 0;
    step := 8;

    for i := 0 to Height-1 do
    begin
      Dest := FData + Width * ypos;
      for xpos := 0 to width-1 do
      begin
        v := readLZW;
        if (v < 0) then
          exit;
        Dest^ := char(v);
        Inc(Dest);
      end;
      Inc(ypos, step);
      if (ypos >= height) then
        repeat
          if (pass > 0) then
            step := step DIV 2;
          Inc(pass);
          ypos := step DIV 2;
        until (ypos < height);
    end;
  end else
  begin
    Dest := FData;
    for ypos := 0 to (height * width)-1 do
    begin
      v := readLZW;
      if (v < 0) then
        exit;
      Dest^ := char(v);
      Inc(Dest);
    end;
  end;
finally
  if (readLZW >= 0) then
    ;
```

```
unit IGIS_ClientTLB_TLB;

// ************************************************************************ //
// WARNING
// -------
// The types declared in this file were generated from data read from a
// Type Library. If this type library is explicitly or indirectly (via
// another type library referring to this type library) re-imported, or the
// 'Refresh' command of the Type Library Editor activated while editing the
// Type Library, the contents of this file will be regenerated and all
// manual modifications will be lost.
// ************************************************************************ //

// PASTLWTR : $Revision:   1.88.1.0.1.0  $
// File generated on 2000/5/14 PM 01:29:31 from Type Library described below.

// ************************************************************************ //
// Type Lib: C:\IGIS Source\IGIS Client\IGIS Client.tlb (1)
// IID\LCID: {04FF1040-8170-11D1-BB48-0080C8583129}\0
// Helpfile:
// DepndLst:
//   (1) v1.0 stdole, (D:\WINNT\System32\stdole32.tlb)
//   (2) v2.0 StdType, (D:\WINNT\System32\OLEPRO32.DLL)
//   (3) v1.0 StdVCL, (D:\WINNT\System32\STDVCL32.DLL)
// ************************************************************************ //
{$TYPEDADDRESS OFF} // Unit must be compiled without type-checked pointers.
interface

uses Windows, ActiveX, Classes, Graphics, OleServer, OleCtrls, StdVCL;

// *********************************************************************//
// GUIDS declared in the TypeLibrary. Following prefixes are used:
//   Type Libraries     : LIBID xxxx
//   CoClasses          : CLASS xxxx
//   DISPInterfaces     : DIID xxxx
//   Non-DISP interfaces: IID xxxx
// *********************************************************************//
const
  // TypeLibrary Major and minor versions
  IGIS_ClientTLBMajorVersion = 1;
  IGIS_ClientTLBMinorVersion = 0;

  LIBID_IGIS_ClientTLB: TGUID = '{04FF1040-8170-11D1-BB48-0080C8583129}';

  IID_IIGIS_Client: TGUID = '{04FF1041-8170-11D1-BB48-0080C8583129}';
  DIID_IIGIS_Client_Events: TGUID = '{04FF1042-8170-11D1-BB48-0080C8583129}';
  CLASS_IGIS_Client_Form: TGUID = '{04FF1043-8170-11D1-BB48-0080C8583129}';

// *********************************************************************//
// Declaration of Enumerations defined in Type Library
// *********************************************************************//
// Constants for enum TxActiveFormBorderStyle
type
  TxActiveFormBorderStyle = TOleEnum;
const
  afbNone = $00000000;
  afbSingle = $00000001;
  afbSunken = $00000002;
  afbRaised = $00000003;

// Constants for enum TxPrintScale
type
  TxPrintScale = TOleEnum;
const
  poNone = $00000000;
  poProportional = $00000001;
  poPrintToFit = $00000002;

// Constants for enum TxMouseButton
type
  TxMouseButton = TOleEnum;
const
  mbLeft = $00000000;
  mbRight = $00000001;
  mbMiddle = $00000002;

// Constants for enum TxWindowState
```

```
type
  TxWindowState = TOleEnum;
const
  wsNormal = $00000000;
  wsMinimized = $00000001;
  wsMaximized = $00000002;

type

// *********************************************************************//
// Forward declaration of types defined in TypeLibrary
// *********************************************************************//
  IIGIS_Client = interface;
  IIGIS_ClientDisp = dispinterface;
  IIGIS_Client_Events = dispinterface;

// *********************************************************************//
// Declaration of CoClasses defined in Type Library
// (NOTE: Here we map each CoClass to its Default Interface)
// *********************************************************************//
  IGIS_Client_Form = IIGIS_Client;


// *********************************************************************//
// Interface: IIGIS_Client
// Flags:     (4432) Hidden Dual OleAutomation Dispatchable
// GUID:      {04FF1041-8170-11D1-BB48-0080C8583129}
// *********************************************************************//
  IIGIS_Client = interface(IDispatch)
    ['{04FF1041-8170-11D1-BB48-0080C8583129}']
    function  Get_AutoScroll: WordBool; safecall;
    procedure Set_AutoScroll(AutoScroll: WordBool); safecall;
    function  Get_AxBorderStyle: TxActiveFormBorderStyle; safecall;
    procedure Set_AxBorderStyle(AxBorderStyle: TxActiveFormBorderStyle); safecall;
    function  Get_Caption: WideString; safecall;
    procedure Set_Caption(const Caption: WideString); safecall;
    function  Get_Color: OLE_COLOR; safecall;
    procedure Set_Color(Color: OLE_COLOR); safecall;
    function  Get_Font: IFontDisp; safecall;
    procedure Set_Font(const Font: IFontDisp); safecall;
    function  Get_KeyPreview: WordBool; safecall;
    procedure Set_KeyPreview(KeyPreview: WordBool); safecall;
    function  Get_PixelsPerInch: Integer; safecall;
    procedure Set_PixelsPerInch(PixelsPerInch: Integer); safecall;
    function  Get_PrintScale: TxPrintScale; safecall;
    procedure Set_PrintScale(PrintScale: TxPrintScale); safecall;
    function  Get_Scaled: WordBool; safecall;
    procedure Set_Scaled(Scaled: WordBool); safecall;
    function  Get_Active: WordBool; safecall;
    function  Get_DropTarget: WordBool; safecall;
    procedure Set_DropTarget(DropTarget: WordBool); safecall;
    function  Get_HelpFile: WideString; safecall;
    procedure Set_HelpFile(const HelpFile: WideString); safecall;
    function  Get_WindowState: TxWindowState; safecall;
    procedure Set_WindowState(WindowState: TxWindowState); safecall;
    function  Get_Visible: WordBool; safecall;
    procedure Set_Visible(Visible: WordBool); safecall;
    function  Get_Enabled: WordBool; safecall;
    procedure Set_Enabled(Enabled: WordBool); safecall;
    function  Get_Cursor: Smallint; safecall;
    procedure Set_Cursor(Cursor: Smallint); safecall;
    procedure AboutBox; safecall;
    property AutoScroll: WordBool read Get_AutoScroll write Set_AutoScroll;
    property AxBorderStyle: TxActiveFormBorderStyle read Get_AxBorderStyle write
Set_AxBorderStyle;
    property Caption: WideString read Get_Caption write Set_Caption;
    property Color: OLE_COLOR read Get_Color write Set_Color;
    property Font: IFontDisp read Get_Font write Set_Font;
    property KeyPreview: WordBool read Get_KeyPreview write Set_KeyPreview;
    property PixelsPerInch: Integer read Get_PixelsPerInch write Set_PixelsPerInch;
    property PrintScale: TxPrintScale read Get_PrintScale write Set_PrintScale;
    property Scaled: WordBool read Get_Scaled write Set_Scaled;
    property Active: WordBool read Get_Active;
    property DropTarget: WordBool read Get_DropTarget write Set_DropTarget;
    property HelpFile: WideString read Get_HelpFile write Set_HelpFile;
    property WindowState: TxWindowState read Get_WindowState write Set_WindowState;
    property Visible: WordBool read Get_Visible write Set_Visible;
```

```
    property Enabled: WordBool read Get_Enabled write Set_Enabled;
    property Cursor: Smallint read Get_Cursor write Set_Cursor;
  end;

// ******************************************************************//
// DispIntf:  IIGIS ClientDisp
// Flags:     (4432) Hidden Dual OleAutomation Dispatchable
// GUID:      {04FF1041-8170-11D1-BB48-0080C8583129}
// ******************************************************************//
  IIGIS_ClientDisp = dispinterface
    ['{04FF1041-8170-11D1-BB48-0080C8583129}']
    property AutoScroll: WordBool dispid 1;
    property AxBorderStyle: TxActiveFormBorderStyle dispid 2;
    property Caption: WideString dispid 3;
    property Color: OLE_COLOR dispid 4;
    property Font: IFontDisp dispid 5;
    property KeyPreview: WordBool dispid 6;
    property PixelsPerInch: Integer dispid 7;
    property PrintScale: TxPrintScale dispid 8;
    property Scaled: WordBool dispid 9;
    property Active: WordBool readonly dispid 10;
    property DropTarget: WordBool dispid 11;
    property HelpFile: WideString dispid 12;
    property WindowState: TxWindowState dispid 13;
    property Visible: WordBool dispid 14;
    property Enabled: WordBool dispid 15;
    property Cursor: Smallint dispid 16;
    procedure AboutBox; dispid -552;
  end;

// ******************************************************************//
// DispIntf:  IIGIS Client Events
// Flags:     (4096) Dispatchable
// GUID:      {04FF1042-8170-11D1-BB48-0080C8583129}
// ******************************************************************//
  IIGIS_Client_Events = dispinterface
    ['{04FF1042-8170-11D1-BB48-0080C8583129}']
    procedure OnActivate; dispid 1;
    procedure OnClick; dispid 2;
    procedure OnCreate; dispid 3;
    procedure OnDblClick; dispid 4;
    procedure OnDestroy; dispid 5;
    procedure OnDeactivate; dispid 6;
    procedure OnKeyPress(var Key: Smallint); dispid 7;
    procedure OnPaint; dispid 8;
  end;


// ******************************************************************//
// OLE Control Proxy class declaration
// Control Name     : TIGIS Client Form
// Help String      : IGIS1Control
// Default Interface: IIGIS Client
// Def. Intf. DISP? : No
// Event   Interface: IIGIS Client Events
// TypeFlags        : (34) CanCreate Control
// ******************************************************************//
  TIGIS_Client_FormOnKeyPress = procedure(Sender: TObject; var Key: Smallint) of object;

  TIGIS_Client_Form = class(TOleControl)
  private
    FOnActivate: TNotifyEvent;
    FOnClick: TNotifyEvent;
    FOnCreate: TNotifyEvent;
    FOnDblClick: TNotifyEvent;
    FOnDestroy: TNotifyEvent;
    FOnDeactivate: TNotifyEvent;
    FOnKeyPress: TIGIS_Client_FormOnKeyPress;
    FOnPaint: TNotifyEvent;
    FIntf: IIGIS_Client;
    function  GetControlInterface: IIGIS_Client;
  protected
    procedure CreateControl;
    procedure InitControlData; override;
  public
    procedure AboutBox;
    property  ControlInterface: IIGIS_Client read GetControlInterface;
```

```
    property  DefaultInterface: IIGIS_Client read GetControlInterface;
    property Active: WordBool index 10 read GetWordBoolProp;
  published
    property  Align;
    property  DragCursor;
    property  DragMode;
    property  ParentShowHint;
    property  PopupMenu;
    property  ShowHint;
    property  TabOrder;
    property  OnDragDrop;
    property  OnDragOver;
    property  OnEndDrag;
    property  OnEnter;
    property  OnExit;
    property  OnStartDrag;
    property AutoScroll: WordBool index 1 read GetWordBoolProp write SetWordBoolProp stored
  False;
    property AxBorderStyle: TOleEnum index 2 read GetTOleEnumProp write SetTOleEnumProp stored
  False;
    property Caption: WideString index 3 read GetWideStringProp write SetWideStringProp stored
  False;
    property Color: TColor index 4 read GetTColorProp write SetTColorProp stored False;
    property Font: TFont index 5 read GetTFontProp write SetTFontProp stored False;
    property KeyPreview: WordBool index 6 read GetWordBoolProp write SetWordBoolProp stored
  False;
    property PixelsPerInch: Integer index 7 read GetIntegerProp write SetIntegerProp stored
  False;
    property PrintScale: TOleEnum index 8 read GetTOleEnumProp write SetTOleEnumProp stored
  False;
    property Scaled: WordBool index 9 read GetWordBoolProp write SetWordBoolProp stored False;
    property DropTarget: WordBool index 11 read GetWordBoolProp write SetWordBoolProp stored
  False;
    property HelpFile: WideString index 12 read GetWideStringProp write SetWideStringProp
  stored False;
    property WindowState: TOleEnum index 13 read GetTOleEnumProp write SetTOleEnumProp stored
  False;
    property Visible: WordBool index 14 read GetWordBoolProp write SetWordBoolProp stored
  False;
    property Enabled: WordBool index 15 read GetWordBoolProp write SetWordBoolProp stored
  False;
    property Cursor: Smallint index 16 read GetSmallintProp write SetSmallintProp stored False;
    property OnActivate: TNotifyEvent read FOnActivate write FOnActivate;
    property OnClick: TNotifyEvent read FOnClick write FOnClick;
    property OnCreate: TNotifyEvent read FOnCreate write FOnCreate;
    property OnDblClick: TNotifyEvent read FOnDblClick write FOnDblClick;
    property OnDestroy: TNotifyEvent read FOnDestroy write FOnDestroy;
    property OnDeactivate: TNotifyEvent read FOnDeactivate write FOnDeactivate;
    property OnKeyPress: TIGIS_Client_FormOnKeyPress read FOnKeyPress write FOnKeyPress;
    property OnPaint: TNotifyEvent read FOnPaint write FOnPaint;
  end;

procedure Register;

implementation

uses ComObj;

procedure TIGIS_Client_Form.InitControlData;
const
  CEventDispIDs: array [0..7] of DWORD = (
    $00000001, $00000002, $00000003, $00000004, $00000005, $00000006,
    $00000007, $00000008);
  CTFontIDs: array [0..0] of DWORD = (
    $00000005);
  CControlData: TControlData2 = (
    ClassID: '{04FF1043-8170-11D1-BB48-0080C8583129}';
    EventIID: '{04FF1042-8170-11D1-BB48-0080C8583129}';
    EventCount: 8;
    EventDispIDs: @CEventDispIDs;
    LicenseKey: nil (*HR:$00000000*);
    Flags: $00000000;
    Version: 401;
    FontCount: 1;
    FontIDs: @CTFontIDs);
begin
  ControlData := @CControlData;
```

```
  TControlData2(CControlData).FirstEventOfs := Cardinal(@@FOnActivate) - Cardinal(Self);
end;

procedure TIGIS_Client_Form.CreateControl;

  procedure DoCreate;
  begin
    FIntf := IUnknown(OleObject) as IIGIS_Client;
  end;

begin
  if FIntf = nil then DoCreate;
end;

function TIGIS_Client_Form.GetControlInterface: IIGIS_Client;
begin
  CreateControl;
  Result := FIntf;
end;

procedure TIGIS_Client_Form.AboutBox;
begin
  DefaultInterface.AboutBox;
end;

procedure Register;
begin
  RegisterComponents('ActiveX',[TIGIS_Client_Form]);
end;

end.
```

```
constructor TSteveArcheDitherer.Create(AWidth: integer; Lookup: TColorLookup);
begin
  inherited Create(AWidth, Lookup);

  GetMem(ErrorsR0, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsG0, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsB0, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsR1, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsG1, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsB1, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsR2, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsG2, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsB2, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsR3, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsG3, sizeof(TErrorTerm)*(Width+6));
  GetMem(ErrorsB3, sizeof(TErrorTerm)*(Width+6));
  FillChar(ErrorsR0^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsG0^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsB0^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsR1^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsG1^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsB1^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsR2^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsG2^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsB2^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsR3^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsG3^, sizeof(TErrorTerm)*(Width+6), 0);
  FillChar(ErrorsB3^, sizeof(TErrorTerm)*(Width+6), 0);

  FDirection2 := 2 * Direction;
  FDirection3 := 3 * Direction;

  ErrorR0 := PErrors(longInt(ErrorsR0)+3*sizeof(TErrorTerm));
  ErrorG0 := PErrors(longInt(ErrorsG0)+3*sizeof(TErrorTerm));
  ErrorB0 := PErrors(longInt(ErrorsB0)+3*sizeof(TErrorTerm));
  ErrorR1 := PErrors(longInt(ErrorsR1)+3*sizeof(TErrorTerm));
  ErrorG1 := PErrors(longInt(ErrorsG1)+3*sizeof(TErrorTerm));
  ErrorB1 := PErrors(longInt(ErrorsB1)+3*sizeof(TErrorTerm));
  ErrorR2 := PErrors(longInt(ErrorsR2)+3*sizeof(TErrorTerm));
  ErrorG2 := PErrors(longInt(ErrorsG2)+3*sizeof(TErrorTerm));
  ErrorB2 := PErrors(longInt(ErrorsB2)+3*sizeof(TErrorTerm));
  ErrorR3 := PErrors(longInt(ErrorsR3)+3*sizeof(TErrorTerm));
  ErrorG3 := PErrors(longInt(ErrorsG3)+3*sizeof(TErrorTerm));
  ErrorB3 := PErrors(longInt(ErrorsB3)+3*sizeof(TErrorTerm));
end;

destructor TSteveArcheDitherer.Destroy;
begin
  FreeMem(ErrorsR0);
  FreeMem(ErrorsG0);
  FreeMem(ErrorsB0);
  FreeMem(ErrorsR1);
  FreeMem(ErrorsG1);
  FreeMem(ErrorsB1);
  FreeMem(ErrorsR2);
  FreeMem(ErrorsG2);
  FreeMem(ErrorsB2);
  FreeMem(ErrorsR3);
  FreeMem(ErrorsG3);
  FreeMem(ErrorsB3);
  inherited Destroy;
end;

{$IFOPT R+}
  {$DEFINE R_PLUS}
  {$RANGECHECKS OFF}
{$ENDIF}
function TSteveArcheDitherer.Dither(Red, Green, Blue: BYTE; var R, G, B: BYTE): char;
var
  ColorR                ,
  ColorG                ,
  ColorB                : integer; // Error for current pixel

  // Propagate Stevenson & Arche error terms:
  //     ...      ...      ...      (here)   ...      32/200   ...
  //     12/200   ...      26/200   ...      30/200   ...      16/200
  //     ...      12/200   ...      26/200   ...      12/200   ...
```

```
    //    5/200   ...     12/200  ...     12/200  ...     5/200
    procedure Propagate(Errors0, Errors1, Errors2, Errors3: PErrors; Error: integer);
    var
      TempError           : integer;
    begin
      if (Error = 0) then
        exit;
      TempError := 5 * Error;
      inc(Errors3[FDirection3], TempError);       // Error * 5
      inc(Errors3[-FDirection3], TempError);      // Error * 5

      TempError := 12 * Error;
      inc(Errors1[-FDirection3], TempError);      // Error * 12
      inc(Errors2[-FDirection2], TempError);      // Error * 12
      inc(Errors2[FDirection2], TempError);       // Error * 12
      inc(Errors3[-Direction], TempError);        // Error * 12
      inc(Errors3[Direction], TempError);         // Error * 12

      inc(Errors1[FDirection3], 16 * TempError);  // Error * 16

      TempError := 26 * Error;
      inc(Errors1[-Direction], TempError);        // Error * 26
      inc(Errors2[0], TempError);                 // Error * 26

      inc(Errors1[Direction], 30 * Error);        // Error * 30

      inc(Errors0[FDirection2], 32 * Error);      // Error * 32
    end;

  begin
    // Apply red component error correction
    ColorR := Red + (ErrorR0[0] + 100) DIV 200;
    if (ColorR < 0) then
      ColorR := 0
    else if (ColorR > 255) then
      ColorR := 255;

    // Apply green component error correction
    ColorG := Green + (ErrorG0[0] + 100) DIV 200;
    if (ColorG < 0) then
      ColorG := 0
    else if (ColorG > 255) then
      ColorG := 255;

    // Apply blue component error correction
    ColorB := Blue + (ErrorB0[0] + 100) DIV 200;
    if (ColorB < 0) then
      ColorB := 0
    else if (ColorB > 255) then
      ColorB := 255;

    // Map color to palette
    Result := inherited Dither(ColorR, ColorG, ColorB, R, G, B);

    // Propagate red component error
    Propagate(ErrorR0, ErrorR1, ErrorR2, ErrorR3, ColorR - R);
    // Propagate green component error
    Propagate(ErrorG0, ErrorG1, ErrorG2, ErrorG3, ColorG - G);
    // Propagate blue component error
    Propagate(ErrorB0, ErrorB1, ErrorB2, ErrorB3, ColorB - B);

    // Move on to next column
    if (Direction = 1) then
    begin
      inc(longInt(ErrorR0), sizeof(TErrorTerm));
      inc(longInt(ErrorG0), sizeof(TErrorTerm));
      inc(longInt(ErrorB0), sizeof(TErrorTerm));
      inc(longInt(ErrorR1), sizeof(TErrorTerm));
      inc(longInt(ErrorG1), sizeof(TErrorTerm));
      inc(longInt(ErrorB1), sizeof(TErrorTerm));
      inc(longInt(ErrorR2), sizeof(TErrorTerm));
      inc(longInt(ErrorG2), sizeof(TErrorTerm));
      inc(longInt(ErrorB2), sizeof(TErrorTerm));
      inc(longInt(ErrorR3), sizeof(TErrorTerm));
      inc(longInt(ErrorG3), sizeof(TErrorTerm));
      inc(longInt(ErrorB3), sizeof(TErrorTerm));
    end else
```

```
    begin
      dec(longInt(ErrorR0), sizeof(TErrorTerm));
      dec(longInt(ErrorG0), sizeof(TErrorTerm));
      dec(longInt(ErrorB0), sizeof(TErrorTerm));
      dec(longInt(ErrorR1), sizeof(TErrorTerm));
      dec(longInt(ErrorG1), sizeof(TErrorTerm));
      dec(longInt(ErrorB1), sizeof(TErrorTerm));
      dec(longInt(ErrorR2), sizeof(TErrorTerm));
      dec(longInt(ErrorG2), sizeof(TErrorTerm));
      dec(longInt(ErrorB2), sizeof(TErrorTerm));
      dec(longInt(ErrorR3), sizeof(TErrorTerm));
      dec(longInt(ErrorG3), sizeof(TErrorTerm));
      dec(longInt(ErrorB3), sizeof(TErrorTerm));
    end;
  end;
  {$IFDEF R_PLUS}
    {$RANGECHECKS ON}
    {$UNDEF R_PLUS}
  {$ENDIF}

  {$IFOPT R+}
    {$DEFINE R_PLUS}
    {$RANGECHECKS OFF}
  {$ENDIF}
  procedure TSteveArcheDitherer.NextLine;
  var
    TempErrors            : PErrors;
  begin
    FillChar(ErrorsR0^, sizeof(TErrorTerm)*(Width+6), 0);
    FillChar(ErrorsG0^, sizeof(TErrorTerm)*(Width+6), 0);
    FillChar(ErrorsB0^, sizeof(TErrorTerm)*(Width+6), 0);

    // Swap lines
    TempErrors := ErrorsR0;
    ErrorsR0 := ErrorsR1;
    ErrorsR1 := ErrorsR2;
    ErrorsR2 := ErrorsR3;
    ErrorsR3 := TempErrors;

    TempErrors := ErrorsG0;
    ErrorsG0 := ErrorsG1;
    ErrorsG1 := ErrorsG2;
    ErrorsG2 := ErrorsG3;
    ErrorsG3 := TempErrors;

    TempErrors := ErrorsB0;
    ErrorsB0 := ErrorsB1;
    ErrorsB1 := ErrorsB2;
    ErrorsB2 := ErrorsB3;
    ErrorsB3 := TempErrors;

    inherited NextLine;

    FDirection2 := 2 * Direction;
    FDirection3 := 3 * Direction;

    if (Direction = 1) then
    begin
      // ErrorsR0[1] gives compiler error, so we
      // use PErrors(longInt(ErrorsR0)+sizeof(TErrorTerm)) instead...
      ErrorR0 := PErrors(longInt(ErrorsR0)+3*sizeof(TErrorTerm));
      ErrorG0 := PErrors(longInt(ErrorsG0)+3*sizeof(TErrorTerm));
      ErrorB0 := PErrors(longInt(ErrorsB0)+3*sizeof(TErrorTerm));
      ErrorR1 := PErrors(longInt(ErrorsR1)+3*sizeof(TErrorTerm));
      ErrorG1 := PErrors(longInt(ErrorsG1)+3*sizeof(TErrorTerm));
      ErrorB1 := PErrors(longInt(ErrorsB1)+3*sizeof(TErrorTerm));
      ErrorR2 := PErrors(longInt(ErrorsR2)+3*sizeof(TErrorTerm));
      ErrorG2 := PErrors(longInt(ErrorsG2)+3*sizeof(TErrorTerm));
      ErrorB2 := PErrors(longInt(ErrorsB2)+3*sizeof(TErrorTerm));
      ErrorR3 := PErrors(longInt(ErrorsR3)+3*sizeof(TErrorTerm));
      ErrorG3 := PErrors(longInt(ErrorsG3)+3*sizeof(TErrorTerm));
      ErrorB3 := PErrors(longInt(ErrorsB3)+3*sizeof(TErrorTerm));
    end else
    begin
      ErrorR0 := @ErrorsR0[Width+2];
      ErrorG0 := @ErrorsG0[Width+2];
      ErrorB0 := @ErrorsB0[Width+2];
```

```
      ErrorR1 := @ErrorsR1[Width+2];
      ErrorG1 := @ErrorsG1[Width+2];
      ErrorB1 := @ErrorsB1[Width+2];
      ErrorR2 := @ErrorsR2[Width+2];
      ErrorG2 := @ErrorsG2[Width+2];
      ErrorB2 := @ErrorsB2[Width+2];
      ErrorR3 := @ErrorsR2[Width+2];
      ErrorG3 := @ErrorsG2[Width+2];
      ErrorB3 := @ErrorsB2[Width+2];
    end;
  end;
  {$IFDEF R_PLUS}
    {$RANGECHECKS ON}
    {$UNDEF R_PLUS}
  {$ENDIF}

  ////////////////////////////////////////////////////////////////////////////
  //      TBurkesDitherer
  constructor TBurkesDitherer.Create(AWidth: integer; Lookup: TColorLookup);
  begin
    inherited Create(AWidth, Lookup);

    GetMem(ErrorsR0, sizeof(TErrorTerm)*(Width+4));
    GetMem(ErrorsG0, sizeof(TErrorTerm)*(Width+4));
    GetMem(ErrorsB0, sizeof(TErrorTerm)*(Width+4));
    GetMem(ErrorsR1, sizeof(TErrorTerm)*(Width+4));
    GetMem(ErrorsG1, sizeof(TErrorTerm)*(Width+4));
    GetMem(ErrorsB1, sizeof(TErrorTerm)*(Width+4));
    FillChar(ErrorsR0^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsG0^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsB0^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsR1^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsG1^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsB1^, sizeof(TErrorTerm)*(Width+4), 0);

    FDirection2 := 2 * Direction;
    ErrorR0 := PErrors(longInt(ErrorsR0)+2*sizeof(TErrorTerm));
    ErrorG0 := PErrors(longInt(ErrorsG0)+2*sizeof(TErrorTerm));
    ErrorB0 := PErrors(longInt(ErrorsB0)+2*sizeof(TErrorTerm));
    ErrorR1 := PErrors(longInt(ErrorsR1)+2*sizeof(TErrorTerm));
    ErrorG1 := PErrors(longInt(ErrorsG1)+2*sizeof(TErrorTerm));
    ErrorB1 := PErrors(longInt(ErrorsB1)+2*sizeof(TErrorTerm));
  end;

  destructor TBurkesDitherer.Destroy;
  begin
    FreeMem(ErrorsR0);
    FreeMem(ErrorsG0);
    FreeMem(ErrorsB0);
    FreeMem(ErrorsR1);
    FreeMem(ErrorsG1);
    FreeMem(ErrorsB1);
    inherited Destroy;
  end;

  {$IFOPT R+}
    {$DEFINE R_PLUS}
    {$RANGECHECKS OFF}
  {$ENDIF}
  function TBurkesDitherer.Dither(Red, Green, Blue: BYTE; var R, G, B: BYTE): char;
  var
    ErrorR                ,
    ErrorG                ,
    ErrorB                : integer; // Error for current pixel

    // Propagate Burkes error terms:
    //    ...     ...     (here)  8/32    4/32
    //    2/32    4/32    8/32    4/32    2/32
    procedure Propagate(Errors0, Errors1: PErrors; Error: integer);
    begin
      if (Error = 0) then
        exit;
      inc(Error, Error);
      inc(Errors1[FDirection2], Error);   // Error * 2
      inc(Errors1[-FDirection2], Error);  // Error * 2

      inc(Error, Error);
```

```
      inc(Errors0[FDirection2], Error);    // Error * 4
      inc(Errors1[-Direction], Error);     // Error * 4
      inc(Errors1[Direction], Error);      // Error * 4

      inc(Error, Error);
      inc(Errors0[Direction], Error);      // Error * 8
      inc(Errors1[0], Error);              // Error * 8
    end;

 begin
    // Apply red component error correction
    ErrorR := Red + (ErrorR0[0] + 16) DIV 32;
    if (ErrorR < 0) then
      ErrorR := 0
    else if (ErrorR > 255) then
      ErrorR := 255;

    // Apply green component error correction
    ErrorG := Green + (ErrorG0[0] + 16) DIV 32;
    if (ErrorG < 0) then
      ErrorG := 0
    else if (ErrorG > 255) then
      ErrorG := 255;

    // Apply blue component error correction
    ErrorB := Blue + (ErrorB0[0] + 16) DIV 32;
    if (ErrorB < 0) then
      ErrorB := 0
    else if (ErrorB > 255) then
      ErrorB := 255;

    // Map color to palette
    Result := inherited Dither(ErrorR, ErrorG, ErrorB, R, G, B);

    // Propagate red component error
    Propagate(ErrorR0, ErrorR1, ErrorR - R);
    // Propagate green component error
    Propagate(ErrorG0, ErrorG1, ErrorG - G);
    // Propagate blue component error
    Propagate(ErrorB0, ErrorB1, ErrorB - B);

    // Move on to next column
    if (Direction = 1) then
    begin
      inc(longInt(ErrorR0), sizeof(TErrorTerm));
      inc(longInt(ErrorG0), sizeof(TErrorTerm));
      inc(longInt(ErrorB0), sizeof(TErrorTerm));
      inc(longInt(ErrorR1), sizeof(TErrorTerm));
      inc(longInt(ErrorG1), sizeof(TErrorTerm));
      inc(longInt(ErrorB1), sizeof(TErrorTerm));
    end else
    begin
      dec(longInt(ErrorR0), sizeof(TErrorTerm));
      dec(longInt(ErrorG0), sizeof(TErrorTerm));
      dec(longInt(ErrorB0), sizeof(TErrorTerm));
      dec(longInt(ErrorR1), sizeof(TErrorTerm));
      dec(longInt(ErrorG1), sizeof(TErrorTerm));
      dec(longInt(ErrorB1), sizeof(TErrorTerm));
    end;
  end;
  {$IFDEF R_PLUS}
    {$RANGECHECKS ON}
    {$UNDEF R_PLUS}
  {$ENDIF}

  {$IFOPT R+}
    {$DEFINE R_PLUS}
    {$RANGECHECKS OFF}
  {$ENDIF}
  procedure TBurkesDitherer.NextLine;
  var
    TempErrors            : PErrors;
  begin
    FillChar(ErrorsR0^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsG0^, sizeof(TErrorTerm)*(Width+4), 0);
    FillChar(ErrorsB0^, sizeof(TErrorTerm)*(Width+4), 0);
```

```
    // Swap lines
    TempErrors := ErrorsR0;
    ErrorsR0 := ErrorsR1;
    ErrorsR1 := TempErrors;

    TempErrors := ErrorsG0;
    ErrorsG0 := ErrorsG1;
    ErrorsG1 := TempErrors;

    TempErrors := ErrorsB0;
    ErrorsB0 := ErrorsB1;
    ErrorsB1 := TempErrors;

    inherited NextLine;

    FDirection2 := 2 * Direction;
    if (Direction = 1) then
    begin
      // ErrorsR0[1] gives compiler error, so we
      // use PErrors(longInt(ErrorsR0)+sizeof(TErrorTerm)) instead...
      ErrorR0 := PErrors(longInt(ErrorsR0)+2*sizeof(TErrorTerm));
      ErrorG0 := PErrors(longInt(ErrorsG0)+2*sizeof(TErrorTerm));
      ErrorB0 := PErrors(longInt(ErrorsB0)+2*sizeof(TErrorTerm));
      ErrorR1 := PErrors(longInt(ErrorsR1)+2*sizeof(TErrorTerm));
      ErrorG1 := PErrors(longInt(ErrorsG1)+2*sizeof(TErrorTerm));
      ErrorB1 := PErrors(longInt(ErrorsB1)+2*sizeof(TErrorTerm));
    end else
    begin
      ErrorR0 := @ErrorsR0[Width+1];
      ErrorG0 := @ErrorsG0[Width+1];
      ErrorB0 := @ErrorsB0[Width+1];
      ErrorR1 := @ErrorsR1[Width+1];
      ErrorG1 := @ErrorsG1[Width+1];
      ErrorB1 := @ErrorsB1[Width+1];
    end;
  end;
  {$IFDEF R_PLUS}
    {$RANGECHECKS ON}
    {$UNDEF R_PLUS}
  {$ENDIF}

  ////////////////////////////////////////////////////////////////////////////
  //
  //                      Octree Color Quantization Engine
  //
  ////////////////////////////////////////////////////////////////////////////
  //  Adapted from Earl F. Glynn's ColorQuantizationLibrary, March 1998
  ////////////////////////////////////////////////////////////////////////////
  type
    TOctreeNode = class;  // Forward definition so TReducibleNodes can be declared

    TReducibleNodes = array[0..7] of TOctreeNode;

    TOctreeNode = Class(TObject)
    public
      IsLeaf              : Boolean;
      PixelCount          : integer;
      RedSum              : integer;
      GreenSum            : integer;
      BlueSum             : integer;
      Next                : TOctreeNode;
      Child               : TReducibleNodes;

      constructor Create(Level: integer; ColorBits: integer; var LeafCount: integer;
        var ReducibleNodes: TReducibleNodes);
      destructor Destroy; override;
    end;

    TColorQuantizer = class(TObject)
    private
      FTree               : TOctreeNode;
      FLeafCount          : integer;
      FReducibleNodes     : TReducibleNodes;
      FMaxColors          : integer;
      FColorBits          : integer;

    protected
```

```
      procedure AddColor(var Node: TOctreeNode; r, g, b: byte; ColorBits: integer;
        Level: integer; var LeafCount: integer; var ReducibleNodes: TReducibleNodes);
      procedure DeleteTree(var Node: TOctreeNode);
      procedure GetPaletteColors(const Node: TOctreeNode;
        var RGBQuadArray: TRGBQuadArray; var Index: integer);
      procedure ReduceTree(ColorBits: integer; var LeafCount: integer;
        var ReducibleNodes: TReducibleNodes);

    public
      constructor Create(MaxColors: integer; ColorBits: integer);
      destructor Destroy; override;

      procedure GetColorTable(var RGBQuadArray: TRGBQuadArray);
      function ProcessImage(const DIB: TDIBReader): boolean;

      property ColorCount: integer read FLeafCount;
    end;

  constructor TOctreeNode.Create(Level: integer; ColorBits: integer;
    var LeafCount: integer; var ReducibleNodes: TReducibleNodes);
  var
    i                     : integer;
  begin
    PixelCount := 0;
    RedSum := 0;
    GreenSum := 0;
    BlueSum := 0;
    for i := Low(Child) to High(Child) do
      Child[i] := nil;

    IsLeaf := (Level = ColorBits);
    if (IsLeaf) then
    begin
      Next := nil;
      inc(LeafCount);
    end else
    begin
      Next := ReducibleNodes[Level];
      ReducibleNodes[Level] := self;
    end;
  end;

  destructor TOctreeNode.Destroy;
  var
    i                     : integer;
  begin
    for i := High(Child) downto Low(Child) do
      Child[i].Free;
  end;

  constructor TColorQuantizer.Create(MaxColors: integer; ColorBits: integer);
  var
    i                     : integer;
  begin
    ASSERT(ColorBits <= 8, 'ColorBits must be 8 or less');

    FTree := nil;
    FLeafCount := 0;

    // Initialize all nodes even though only ColorBits+1 of them are needed
    for i := Low(FReducibleNodes) to High(FReducibleNodes) do
      FReducibleNodes[i] := nil;

    FMaxColors := MaxColors;
    FColorBits := ColorBits;
  end;

  destructor TColorQuantizer.Destroy;
  begin
    if (FTree <> nil) then
      DeleteTree(FTree);
  end;

  procedure TColorQuantizer.GetColorTable(var RGBQuadArray: TRGBQuadArray);
  var
    Index                 : integer;
  begin
```

```
  Index := 0;
  GetPaletteColors(FTree, RGBQuadArray, Index);
end;

// Handles passed to ProcessImage should refer to DIB sections, not DDBs.
// In certain cases, specifically when it's called upon to process 1, 4, or
// 8-bit per pixel images on systems with palettized display adapters,
// ProcessImage can produce incorrect results if it's passed a handle to a
// DDB.
function TColorQuantizer.ProcessImage(const DIB: TDIBReader): boolean;
var
  i                     ,
  j                     : integer;
  ScanLine              : pointer;
  Pixel                 : PRGBTriple;
begin
  Result := True;

  for j := 0 to DIB.Bitmap.Height-1 do
  begin
    Scanline := DIB.Scanline[j];
    Pixel := ScanLine;
    for i := 0 to DIB.Bitmap.Width-1 do
    begin
      with Pixel^ do
        AddColor(FTree, rgbtRed, rgbtGreen, rgbtBlue,
                 FColorBits, 0, FLeafCount, FReducibleNodes);

      while FLeafCount > FMaxColors do
        ReduceTree(FColorbits, FLeafCount, FReducibleNodes);
      inc(Pixel);
    end;
  end;
end;

procedure TColorQuantizer.AddColor(var Node: TOctreeNode; r,g,b: byte;
  ColorBits: integer; Level: integer; var LeafCount: integer;
  var ReducibleNodes: TReducibleNodes);
const
  Mask:  array[0..7] of BYTE = ($80, $40, $20, $10, $08, $04, $02, $01);
var
  Index                 : integer;
  Shift                 : integer;
begin
  // If the node doesn't exist, create it.
  if (Node = nil) then
    Node := TOctreeNode.Create(Level, ColorBits, LeafCount, ReducibleNodes);

  if (Node.IsLeaf) then
  begin
    inc(Node.PixelCount);
    inc(Node.RedSum, r);
    inc(Node.GreenSum, g);
    inc(Node.BlueSum, b);
  end else
  begin
    // Recurse a level deeper if the node is not a leaf.
    Shift := 7 - Level;

    Index := (((r and mask[Level]) SHR Shift) SHL 2)  or
             (((g and mask[Level]) SHR Shift) SHL 1)  or
              ((b and mask[Level]) SHR Shift);
    AddColor(Node.Child[Index], r, g, b, ColorBits, Level+1, LeafCount, ReducibleNodes);
  end;
end;

procedure TColorQuantizer.DeleteTree(var Node: TOctreeNode);
var
  i                     : integer;
begin
  for i := High(TReducibleNodes) downto Low(TReducibleNodes) do
    if (Node.Child[i] <> nil) then
      DeleteTree(Node.Child[i]);

  Node.Free;
  Node := nil;
end;
```

```
procedure TColorQuantizer.GetPaletteColors(const Node: TOctreeNode;
  var RGBQuadArray: TRGBQuadArray; var Index: integer);
var
  i                     : integer;
begin
  if (Node.IsLeaf) then
  begin
    with RGBQuadArray[Index] do
    begin
      if (Node.PixelCount <> 0) then
      begin
        rgbRed   := BYTE(Node.RedSum   DIV Node.PixelCount);
        rgbGreen := BYTE(Node.GreenSum DIV Node.PixelCount);
        rgbBlue  := BYTE(Node.BlueSum  DIV Node.PixelCount);
      end else
      begin
        rgbRed := 0;
        rgbGreen := 0;
        rgbBlue := 0;
      end;
      rgbReserved := 0;
    end;
    inc(Index);
  end else
  begin
    for i := Low(Node.Child) to High(Node.Child) do
      if (Node.Child[i] <> nil) then
        GetPaletteColors(Node.Child[i], RGBQuadArray, Index);
  end;
end;

procedure TColorQuantizer.ReduceTree(ColorBits: integer; var LeafCount: integer;
  var ReducibleNodes: TReducibleNodes);
var
  RedSum                ,
  GreenSum              ,
  BlueSum               : integer;
  Children              : integer;
  i                     : integer;
  Node                  : TOctreeNode;
begin
  // Find the deepest level containing at least one reducible node
  i := Colorbits - 1;
  while (i > 0) and (ReducibleNodes[i] = nil) do
    dec(i);

  // Reduce the node most recently added to the list at level i.
  Node := ReducibleNodes[i];
  ReducibleNodes[i] := Node.Next;

  RedSum   := 0;
  GreenSum := 0;
  BlueSum  := 0;
  Children := 0;

  for i := Low(ReducibleNodes) to High(ReducibleNodes) do
    if (Node.Child[i] <> nil) then
    begin
      inc(RedSum, Node.Child[i].RedSum);
      inc(GreenSum, Node.Child[i].GreenSum);
      inc(BlueSum, Node.Child[i].BlueSum);
      inc(Node.PixelCount, Node.Child[i].PixelCount);
      Node.Child[i].Free;
      Node.Child[i] := nil;
      inc(Children);
    end;

  Node.IsLeaf := TRUE;
  Node.RedSum := RedSum;
  Node.GreenSum := GreenSum;
  Node.BlueSum := BlueSum;
  dec(LeafCount, Children-1);
end;

////////////////////////////////////////////////////////////////////////////////
//
```

```
//                    Octree Color Quantization Wrapper
//
////////////////////////////////////////////////////////////////////////////////
//      Adapted from Earl F. Glynn's PaletteLibrary, March 1998
////////////////////////////////////////////////////////////////////////////////

// Wrapper for internal use - uses TDIBReader for bitmap access
function doCreateOptimizedPaletteFromSingleBitmap(const DIB: TDIBReader;
  Colors, ColorBits: integer; Windows: boolean): hPalette;
var
  SystemPalette         : HPalette;
  ColorQuantizer        : TColorQuantizer;
  i                     : integer;
  LogicalPalette        : TMaxLogPalette;
  RGBQuadArray          : TRGBQuadArray;
  Offset                : integer;
begin
  LogicalPalette.palVersion := $0300;
  LogicalPalette.palNumEntries := Colors;

  if (Windows) then
  begin
    // Get the windows 20 color system palette
    SystemPalette := GetStockObject(DEFAULT_PALETTE);
    GetPaletteEntries(SystemPalette, 0, 10, LogicalPalette.palPalEntry[0]);
    GetPaletteEntries(SystemPalette, 10, 10, LogicalPalette.palPalEntry[245]);
    Colors := 236;
    Offset := 10;
    LogicalPalette.palNumEntries := 256;
  end else
    Offset := 0;

  // Normally for 24-bit images, use ColorBits of 5 or 6.  For 8-bit images
  // use ColorBits = 8.
  ColorQuantizer := TColorQuantizer.Create(Colors, ColorBits);
  try
    ColorQuantizer.ProcessImage(DIB);
    ColorQuantizer.GetColorTable(RGBQuadArray);
  finally
    ColorQuantizer.Free;
  end;

  for i := 0 to Colors-1 do
    with LogicalPalette.palPalEntry[i+Offset] do
    begin
      peRed   := RGBQuadArray[i].rgbRed;
      peGreen := RGBQuadArray[i].rgbGreen;
      peBlue  := RGBQuadArray[i].rgbBlue;
      peFlags := RGBQuadArray[i].rgbReserved;
    end;
  Result := CreatePalette(pLogPalette(@LogicalPalette)^);
end;

function CreateOptimizedPaletteFromSingleBitmap(const Bitmap: TBitmap;
  Colors, ColorBits: integer; Windows: boolean): hPalette;
var
  DIB                   : TDIBReader;
begin
  DIB := TDIBReader.Create(Bitmap, pf24bit);
  try
    Result := doCreateOptimizedPaletteFromSingleBitmap(DIB, Colors, ColorBits, Windows);
  finally
    DIB.Free;
  end;
end;

function CreateOptimizedPaletteFromManyBitmaps(Bitmaps: TList; Colors, ColorBits: integer;
  Windows: boolean): hPalette;
var
  SystemPalette         : HPalette;
  ColorQuantizer        : TColorQuantizer;
  i                     : integer;
  LogicalPalette        : TMaxLogPalette;
  RGBQuadArray          : TRGBQuadArray;
  Offset                : integer;
  DIB                   : TDIBReader;
begin
```

```
  if (Bitmaps = nil) or (Bitmaps.Count = 0) then
    Error(sInvalidBitmapList);

  LogicalPalette.palVersion := $0300;
  LogicalPalette.palNumEntries := Colors;

  if (Windows) then
  begin
    // Get the windows 20 color system palette
    SystemPalette := GetStockObject(DEFAULT_PALETTE);
    GetPaletteEntries(SystemPalette, 0, 10, LogicalPalette.palPalEntry[0]);
    GetPaletteEntries(SystemPalette, 10, 10, LogicalPalette.palPalEntry[245]);
    Colors := 236;
    Offset := 10;
    LogicalPalette.palNumEntries := 256;
  end else
    Offset := 0;

  // Normally for 24-bit images, use ColorBits of 5 or 6.  For 8-bit images
  // use ColorBits = 8.
  ColorQuantizer := TColorQuantizer.Create(Colors, ColorBits);
  try
    for i := 0 to Bitmaps.Count-1 do
    begin
      DIB := TDIBReader.Create(TBitmap(Bitmaps[i]), pf24bit);
      try
        ColorQuantizer.ProcessImage(DIB);
      finally
        DIB.Free;
      end;
    end;
    ColorQuantizer.GetColorTable(RGBQuadArray);
  finally
    ColorQuantizer.Free;
  end;

  for i := 0 to Colors-1 do
    with LogicalPalette.palPalEntry[i+Offset] do
    begin
      peRed   := RGBQuadArray[i].rgbRed;
      peGreen := RGBQuadArray[i].rgbGreen;
      peBlue  := RGBQuadArray[i].rgbBlue;
      peFlags := RGBQuadArray[i].rgbReserved;
    end;
  Result := CreatePalette(pLogPalette(@LogicalPalette)^);
end;

////////////////////////////////////////////////////////////////////////////////
//
//                      Color reduction
//
////////////////////////////////////////////////////////////////////////////////
{$IFOPT R+}
  {$DEFINE R_PLUS}
  {$RANGECHECKS OFF}
{$ENDIF}
//: Reduces the color depth of a bitmap using color quantization and dithering.
function ReduceColors(Bitmap: TBitmap; ColorReduction: TColorReduction;
  DitherMode: TDitherMode; ReductionBits: integer; CustomPalette: hPalette): TBitmap;
var
  Palette               : hPalette;
  ColorLookup           : TColorLookup;
  Ditherer              : TDitherEngine;
  Row                   : Integer;
  DIBResult             : TDIBWriter;
  DIBSource             : TDIBReader;
  SrcScanLine           ,
  Src                   : PRGBTriple;
  DstScanLine           ,
  Dst                   : PChar;
  BGR                   : TRGBTriple;
{$ifdef DEBUG DITHERPERFORMANCE}
  TimeStart             ,
  TimeStop              : DWORD;
{$endif}

  function GrayScalePalette: hPalette;
```

```
var
  i                   : integer;
  Pal                 : TMaxLogPalette;
begin
  Pal.palVersion := $0300;
  Pal.palNumEntries := 256;
  for i := 0 to 255 do
  begin
    with (Pal.palPalEntry[i]) do
    begin
      peRed := i;
      peGreen := i;
      peBlue  := i;
      peFlags := PC_NOCOLLAPSE;
    end;
  end;
  Result := CreatePalette(pLogPalette(@Pal)^);
end;

function MonochromePalette: hPalette;
var
  i                   : integer;
  Pal                 : TMaxLogPalette;
const
  Values              : array[0..1] of byte
                      = (0, 255);
begin
  Pal.palVersion := $0300;
  Pal.palNumEntries := 2;
  for i := 0 to 1 do
  begin
    with (Pal.palPalEntry[i]) do
    begin
      peRed := Values[i];
      peGreen := Values[i];
      peBlue  := Values[i];
      peFlags := PC_NOCOLLAPSE;
    end;
  end;
  Result := CreatePalette(pLogPalette(@Pal)^);
end;

function WindowsGrayScalePalette: hPalette;
var
  i                   : integer;
  Pal                 : TMaxLogPalette;
const
  Values              : array[0..3] of byte
                      = (0, 128, 192, 255);
begin
  Pal.palVersion := $0300;
  Pal.palNumEntries := 4;
  for i := 0 to 3 do
  begin
    with (Pal.palPalEntry[i]) do
    begin
      peRed := Values[i];
      peGreen := Values[i];
      peBlue  := Values[i];
      peFlags := PC_NOCOLLAPSE;
    end;
  end;
  Result := CreatePalette(pLogPalette(@Pal)^);
end;

function WindowsHalftonePalette: hPalette;
var
  DC                  : HDC;
begin
  DC := GDICheck(GetDC(0));
  try
    Result := CreateHalfTonePalette(DC);
  finally
    ReleaseDC(0, DC);
  end;
end;
```

```
begin
{$ifdef DEBUG DITHERPERFORMANCE}
  timeBeginPeriod(5);
  TimeStart := timeGetTime;
{$endif}

  Result := TBitmap.Create;
  try

    if (ColorReduction = rmNone) then
    begin
      Result.Assign(Bitmap);
{$ifndef VER9x}
      SetPixelFormat(Result, pf24bit);
{$endif}
      exit;
    end;

{$IFNDEF VER9x}
    if (Bitmap.Width*Bitmap.Height > BitmapAllocationThreshold) then
      SetPixelFormat(Result, pflbit); // To reduce resource consumption of resize
{$ENDIF}

    ColorLookup := nil;
    Ditherer := nil;
    DIBResult := nil;
    DIBSource := nil;
    Palette := 0;
    try // Protect above resources

      // Dithering and color mapper only supports 24 bit bitmaps,
      // so we have convert the source bitmap to the appropiate format.
      DIBSource := TDIBReader.Create(Bitmap, pf24bit);

      // Create a palette based on current options
      case (ColorReduction) of
        rmQuantize:
          Palette := doCreateOptimizedPaletteFromSingleBitmap(DIBSource, 1 SHL ReductionBits,
8, False);
        rmQuantizeWindows:
          Palette := CreateOptimizedPaletteFromSingleBitmap(Bitmap, 256, 8, True);
        rmNetscape:
          Palette := WebPalette;
        rmGrayScale:
          Palette := GrayScalePalette;
        rmMonochrome:
          Palette := MonochromePalette;
        rmWindowsGray:
          Palette := WindowsGrayScalePalette;
        rmWindows20:
          Palette := GetStockObject(DEFAULT_PALETTE);
        rmWindows256:
          Palette := WindowsHalftonePalette;
        rmPalette:
          Palette := CopyPalette(CustomPalette);
      else
        exit;
      end;

      { TODO -oanme -cImprovement : Gray scale conversion should be done prior to
dithering/mapping. Otherwise corrected values will be converted multiple times. }

      // Create a color mapper based on current options
      case (ColorReduction) of
        // For some strange reason my fast and dirty color lookup
        // is more precise that Windows GetNearestPaletteIndex...
        // rmWindows20:
        //  ColorLookup := TSlowColorLookup.Create(Palette);
        // rmWindowsGray:
        //  ColorLookup := TGrayWindowsLookup.Create(Palette);
        rmQuantize:
          ColorLookup := TFastColorLookup.Create(Palette);
        rmNetscape:
          ColorLookup := TNetscapeColorLookup.Create(Palette);
        rmGrayScale:
          ColorLookup := TGrayScaleLookup.Create(Palette);
        rmMonochrome:
```

```
          ColorLookup := TMonochromeLookup.Create(Palette);
        else
          ColorLookup := TFastColorLookup.Create(Palette);
        end;

        // Nothing to do if palette doesn't contain any colors
        if (ColorLookup.Colors = 0) then
          exit;

        // Create a ditherer based on current options
        case (DitherMode) of
          dmNearest:
            Ditherer := TDitherEngine.Create(Bitmap.Width, ColorLookup);
          dmFloydSteinberg:
            Ditherer := TFloydSteinbergDitherer.Create(Bitmap.Width, ColorLookup);
          dmStucki:
            Ditherer := TStuckiDitherer.Create(Bitmap.Width, ColorLookup);
          dmSierra:
            Ditherer := TSierraDitherer.Create(Bitmap.Width, ColorLookup);
          dmJaJuNI:
            Ditherer := TJaJuNIDitherer.Create(Bitmap.Width, ColorLookup);
          dmSteveArche:
            Ditherer := TSteveArcheDitherer.Create(Bitmap.Width, ColorLookup);
          dmBurkes:
            Ditherer := TBurkesDitherer.Create(Bitmap.Width, ColorLookup);
        else
          exit;
        end;

        // The processed bitmap is returned in pf8bit format
        DIBResult := TDIBWriter.Create(Result, pf8bit, Bitmap.Width, Bitmap.Height,
          Palette);

        // Process the image
        Row := 0;
        while (Row < Bitmap.Height) do
        begin
          SrcScanline := DIBSource.ScanLine[Row];
          DstScanline := DIBResult.ScanLine[Row];
          Src := pointer(longInt(SrcScanLine) + Ditherer.Column*sizeof(TRGBTriple));
          Dst := pointer(longInt(DstScanLine) + Ditherer.Column);

          while (Ditherer.Column < Ditherer.Width) and (Ditherer.Column >= 0) do
          begin
            BGR := Src^;
            // Dither and map a single pixel
            Dst^ := Ditherer.Dither(BGR.rgbtRed, BGR.rgbtGreen, BGR.rgbtBlue,
              BGR.rgbtRed, BGR.rgbtGreen, BGR.rgbtBlue);

            inc(Src, Ditherer.Direction);
            inc(Dst, Ditherer.Direction);
          end;

          Inc(Row);
          Ditherer.NextLine;
        end;
      finally
        if (ColorLookup <> nil) then
          ColorLookup.Free;
        if (Ditherer <> nil) then
          Ditherer.Free;
        if (DIBResult <> nil) then
          DIBResult.Free;
        if (DIBSource <> nil) then
          DIBSource.Free;
        // Must delete palette after TDIBWriter since TDIBWriter uses palette
        if (Palette <> 0) then
          DeleteObject(Palette);
      end;
    except
      Result.Free;
      raise;
    end;

{$ifdef DEBUG DITHERPERFORMANCE}
    TimeStop := timeGetTime;
    ShowMessage(format('Dithered %d pixels in %d mS, Rate %d pixels/mS (%d pixels/S)',
```

```
      [Bitmap.Height*Bitmap.Width, TimeStop-TimeStart,
      MulDiv(Bitmap.Height, Bitmap.Width, TimeStop-TimeStart+1),
      MulDiv(Bitmap.Height, Bitmap.Width * 1000, TimeStop-TimeStart+1)]));
    timeEndPeriod(5);
  {$endif}
  end;
  {$IFDEF R PLUS}
    {$RANGECHECKS ON}
    {$UNDEF R PLUS}
  {$ENDIF}

  ////////////////////////////////////////////////////////////////////////////////
  //
  //                        TGIFColorMap
  //
  ////////////////////////////////////////////////////////////////////////////////
  const
    InitColorMapSize = 16;
    DeltaColorMapSize = 32;

  //: Creates an instance of a TGIFColorMap object.
  constructor TGIFColorMap.Create;
  begin
    inherited Create;
    FColorMap := nil;
    FCapacity := 0;
    FCount := 0;
    FOptimized := False;
  end;

  //: Destroys an instance of a TGIFColorMap object.
  destructor TGIFColorMap.Destroy;
  begin
    Clear;
    Changed;
    inherited Destroy;
  end;

  //: Empties the color map.
  procedure TGIFColorMap.Clear;
  begin
    if (FColorMap <> nil) then
      FreeMem(FColorMap);
    FColorMap := nil;
    FCapacity := 0;
    FCount := 0;
    FOptimized := False;
  end;

  //: Converts a Windows color value to a RGB value.
  class function TGIFColorMap.Color2RGB(Color: TColor): TGIFColor;
  begin
    Result.Blue := (Color shr 16) and $FF;
    Result.Green := (Color shr 8) and $FF;
    Result.Red  := Color and $FF;
  end;

  //: Converts a RGB value to a Windows color value.
  class function TGIFColorMap.RGB2Color(Color: TGIFColor): TColor;
  begin
    Result := (Color.Blue SHL 16) OR (Color.Green SHL 8) OR Color.Red;
  end;

  //: Saves the color map to a stream.
  procedure TGIFColorMap.SaveToStream(Stream: TStream);
  var
    Dummies               : integer;
    Dummy                 : TGIFColor;
  begin
    if (FCount = 0) then
      exit;
    Stream.WriteBuffer(FColorMap^, FCount*sizeof(TGIFColor));
    Dummies := (1 SHL BitsPerPixel)-FCount;
    Dummy.Red := 0;
    Dummy.Green := 0;
    Dummy.Blue := 0;
    while (Dummies > 0) do
```

```
    begin
      Stream.WriteBuffer(Dummy, sizeof(TGIFColor));
      dec(Dummies);
    end;
  end;

  //: Loads the color map from a stream.
  procedure TGIFColorMap.LoadFromStream(Stream: TStream; Count: integer);
  begin
    Clear;
    SetCapacity(Count);
    ReadCheck(Stream, FColorMap^, Count*sizeof(TGIFColor));
    FCount := Count;
  end;

  //: Returns the position of a color in the color map.
  function TGIFColorMap.IndexOf(Color: TColor): integer;
  var
    RGB                   : TGIFColor;
  begin
    RGB := Color2RGB(Color);
    if (FOptimized) then
    begin
      // Optimized palette has most frequently occuring entries first
      Result := 0;
      // Reverse search to (hopefully) check latest colors first
      while (Result < FCount) do
        with (FColorMap^[Result]) do
        begin
          if (RGB.Red = Red) and (RGB.Green = Green) and (RGB.Blue = Blue) then
            exit;
          Inc(Result);
        end;
      Result := -1;
    end else
    begin
      Result := FCount-1;
      // Reverse search to (hopefully) check latest colors first
      while (Result >= 0) do
        with (FColorMap^[Result]) do
        begin
          if (RGB.Red = Red) and (RGB.Green = Green) and (RGB.Blue = Blue) then
            exit;
          Dec(Result);
        end;
    end;
  end;

  procedure TGIFColorMap.SetCapacity(Size: integer);
  begin
    if (Size >= FCapacity) then
    begin
      if (Size <= InitColorMapSize) then
        FCapacity := InitColorMapSize
      else
        FCapacity := (Size + DeltaColorMapSize - 1) DIV DeltaColorMapSize * DeltaColorMapSize;
      if (FCapacity > GIFMaxColors) then
        FCapacity := GIFMaxColors;
      ReallocMem(FColorMap, FCapacity * sizeof(TGIFColor));
    end;
  end;

  //: Imports a Windows palette into the color map.
  procedure TGIFColorMap.ImportPalette(Palette: HPalette);
  type
    PalArray =  array[byte] of TPaletteEntry;
  var
    Pal                   : PalArray;
    NewCount              : integer;
    i                     : integer;
  begin
    Clear;
    NewCount := GetPaletteEntries(Palette, 0, 256, pal);
    if (NewCount = 0) then
      exit;
    SetCapacity(NewCount);
    for i := 0 to NewCount-1 do
```

```
      with FColorMap[i], Pal[i] do
      begin
        Red := peRed;
        Green := peGreen;
        Blue := peBlue;
      end;
    FCount := NewCount;
    Changed;
  end;

  //: Imports a color map structure into the color map.
  procedure TGIFColorMap.ImportColorMap(Map: TColorMap; Count: integer);
  begin
    Clear;
    if (Count = 0) then
      exit;
    SetCapacity(Count);
    FCount := Count;

    System.Move(Map, FColorMap^, FCount * sizeof(TGIFColor));

    Changed;
  end;

  //: Imports a Windows palette structure into the color map.
  procedure TGIFColorMap.ImportColorTable(Pal: pointer; Count: integer);
  var
    i                    : integer;
  begin
    Clear;
    if (Count = 0) then
      exit;
    SetCapacity(Count);
    for i := 0 to Count-1 do
      with FColorMap[i], PRGBQuadArray(Pal)[i] do
      begin
        Red := rgbRed;
        Green := rgbGreen;
        Blue := rgbBlue;
      end;
    FCount := Count;
    Changed;
  end;

  //: Imports the color table of a DIB into the color map.
  procedure TGIFColorMap.ImportDIBColors(Handle: HDC);
  var
    Pal                  : Pointer;
    NewCount             : integer;
  begin
    Clear;
    GetMem(Pal, sizeof(TRGBQuad) * 256);
    try
      NewCount := GetDIBColorTable(Handle, 0, 256, Pal^);
      ImportColorTable(Pal, NewCount);
    finally
      FreeMem(Pal);
    end;
    Changed;
  end;

  //: Creates a Windows palette from the color map.
  function TGIFColorMap.ExportPalette: HPalette;
  var
    Pal                  : TMaxLogPalette;
    i                    : Integer;
  begin
    if (Count = 0) then
    begin
      Result := 0;
      exit;
    end;
    Pal.palVersion := $300;
    Pal.palNumEntries := Count;
    for i := 0 to Count-1 do
      with FColorMap[i], Pal.palPalEntry[i] do
      begin
```

```
        peRed := Red;
        peGreen := Green;
        peBlue := Blue;
        peFlags := PC_NOCOLLAPSE; { TODO -oanme -cImprovement : Verify that PC NOCOLLAPSE is the
  correct value to use. }
      end;
    Result := CreatePalette(PLogPalette(@Pal)^);
 end;

 //: Adds a color to the color map.
 function TGIFColorMap.Add(Color: TColor): integer;
 begin
   if (FCount >= GIFMaxColors) then
     // Color map full
     Error(sTooManyColors);

   Result := FCount;
   if (Result >= FCapacity) then
     SetCapacity(FCount+1);
   FColorMap^[FCount] := Color2RGB(Color);
   inc(FCount);
   FOptimized := False;
   Changed;
 end;

 function TGIFColorMap.AddUnique(Color: TColor): integer;
 begin
   // Look up color before add (same as IndexOf)
   Result := IndexOf(Color);
   if (Result >= 0) then
     // Color already in map
     exit;

   Result := Add(Color);
 end;

 //: Removes a color from the color map.
 procedure TGIFColorMap.Delete(Index: integer);
 begin
   if (Index < 0) or (Index >= FCount) then
     // Color index out of range
     Error(sBadColorIndex);
   dec(FCount);
   if (Index < FCount) then
     System.Move(FColorMap^[Index + 1], FColorMap^[Index], (FCount - Index)* sizeof(TGIFColor));
   FOptimized := False;
   Changed;
 end;

 function TGIFColorMap.GetColor(Index: integer): TColor;
 begin
   if (Index < 0) or (Index >= FCount) then
   begin
     // Color index out of range
     Warning(gsWarning, sBadColorIndex);
     // Raise an exception if the color map is empty
     if (FCount = 0) then
       Error(sEmptyColorMap);
     // Default to color index 0
     Index := 0;
   end;
   Result := RGB2Color(FColorMap^[Index]);
 end;

 procedure TGIFColorMap.SetColor(Index: integer; Value: TColor);
 begin
   if (Index < 0) or (Index >= FCount) then
     // Color index out of range
     Error(sBadColorIndex);
   FColorMap^[Index] := Color2RGB(Value);
   Changed;
 end;

 function TGIFColorMap.DoOptimize: boolean;
 var
   Usage                : TColormapHistogram;
   Remap                : array[0..255] of TGIFColor;
```

```
  ReverseMap            : TColormapReverse;
  i                     : integer;
  LastFound             : boolean;
  NewCount              : integer;
  T                     : TUsageCount;
  Pivot                 : integer;

  procedure QuickSort(iLo, iHi: Integer);
  var
    Lo, Hi: Integer;
  begin
    repeat
      Lo := iLo;
      Hi := iHi;
      Pivot := Usage[(iLo + iHi) SHR 1].Count;
      repeat
        while (Usage[Lo].Count - Pivot > 0) do inc(Lo);
        while (Usage[Hi].Count - Pivot < 0) do dec(Hi);
        if (Lo <= Hi) then
        begin
          T := Usage[Lo];
          Usage[Lo] := Usage[Hi];
          Usage[Hi] := T;
          inc(Lo);
          dec(Hi);
        end;
      until (Lo > Hi);
      if (iLo < Hi) then
        QuickSort(iLo, Hi);
      iLo := Lo;
    until (Lo >= iHi);
  end;

begin
  if (FCount <= 1) then
  begin
    Result := False;
    exit;
  end;

  FOptimized := True;
  Result := True;

  BuildHistogram(Usage);

  (*
  **  Sort according to usage count
  *)
  QuickSort(0, FCount-1);

  (*
  ** Test for table already sorted
  *)
  for i := 0 to FCount-1 do
    if (Usage[i].Index <> i) then
      break;
  if (i = FCount) then
    exit;

  (*
  ** Build old to new map
  *)
  for i := 0 to FCount-1 do
    ReverseMap[Usage[i].Index] := i;


  MapImages(ReverseMap);

  (*
  **  Reorder colormap
  *)
  LastFound := False;
  NewCount := FCount;
  Move(FColorMap^, TempMap, FCount * sizeof(TGIFColor));
  for i := 0 to FCount-1 do
  begin
    FColorMap^[ReverseMap[i]] := TempMap[i];
```

```
    // Find last used color index
    if (Usage[i].Count = 0) and not(LastFound) then
    begin
      LastFound := True;
      NewCount := i;
    end;
  end;

  FCount := NewCount;

  Changed;
end;

function TGIFColorMap.GetBitsPerPixel: integer;
begin
  Result := Colors2bpp(FCount);
end;

//: Copies one color map to another.
procedure TGIFColorMap.Assign(Source: TPersistent);
begin
  if (Source is TGIFColorMap) then
  begin
    Clear;
    FCapacity := TGIFColorMap(Source).FCapacity;
    FCount := TGIFColorMap(Source).FCount;
    FOptimized := TGIFColorMap(Source).FOptimized;
    FColorMap := AllocMem(FCapacity * sizeof(TGIFColor));
    System.Move(TGIFColorMap(Source).FColorMap^, FColorMap^, FCount * sizeof(TGIFColor));
    Changed;
  end else
    inherited Assign(Source);
end;
```