CHƯƠNG 7. GIAO DIỆN NGƯỜI SỬ DỤNG

7.0. Khái niệm về giao diện người sử dụng

Chương trình của người sử dụng không thể nhìn thấy cái gì đang xảy ra trong máy tính.Do đó, điều quan trọng đối với dịch vụ của chương trình tương tác là phải thiết kế trình xuất nhập,nhằm làm cho người sử dụng luôn luôn hiểu rằng,cái gì được anh ta chờ đợi.Kiểu và phương pháp cho phép sử dụng tác động lên các chương trìnhcủa một máy tính, được gọi là giao diện người sử dụng (user interface). Việc nhập vào một sự bày tỏ và xuất ra những dữ liệu đã được sử lý đóng một vai trò quan trọng. Bên cạnh những thiết bị xuất truyền thông như máy in, máy vẽ,....ngày nay, thay vào đó, tất cả các hình vẽ và dữ liệu được thể hiện len màng hình ngay trước mặt.

Tuy nhiên, giao diện người sử dụng tương tác đồ họa ở các máy tính ngày nay đặt ra những yêu cầu rất cao về các ngườn tài nguyên cuả máy tính,nó được phát triển bởi phạm vi mã cũng như bởi việc chi tiêu nguồn tài nguyên của một trong các bộ phận chính của một máy tính cá nhân (CPU,bộ nhớ.....),các bộ phận này phải được gắn chặt trong hệ điều hành để hoạt động một cách có hiệu nghiệm. Do đó, trên góc độ phần cứng, các cạc (card) đồ họa được dự định với các bộ vi xử lý đồ họa hay với các bộ nhớ lặp lại hình nhanh. Ngay cả ở bộ vi xử lý chính, các lệnh máy phải được trợ giúp cho đồ họa (thí dụ máy tính Intel MMX) để tăng tốc đọ chuan đổi chương trình.

Sự trợ giúp của hệ điều hành cho giao diện người sử dụng tương tác đồ họa là một tiếp giáp quan trọng đối với cấu trúc của tất cả các hệ điều hành hiện đại. Bởi lẽ đó, chúng ta sẽ làm việc trong chương trình này với các yêu cầu thực tiễn, các ý tưởng thiết kế và các câu hỏi thực thi về các giao diện người sử dụng trong sự bao quát về các bộ phận của các hệ điều hành truyền thống và các nguồn tài nguyên của máy tính.

7.1. Vấn đề thiết kế giao diện người sử dụng

Giao diện đồ họa người sử dụng ngày nay dựa trên những công việc nền tảng của nhóm nghiêng cứu Xerox về máy tính STAR và bản dự án mang tên Smalltalk 80 vào đầu thập kỉ 80. Họ thiết kế không chỉ một máy tính cá nhân với một màng hình lưới đồ họa (với những tương quan lúc bấy giờ, nó được gọi là một cuộc cách mạng; bởi vậy, máy tính Xerox- STAR và bản sao của nó đã làm cho hệ thống Apple- Lisa trở nên ít hiệu nghiệm); đặc biệt họ đã phát triển một cách có hệ thống ngôn ngữ lập trình hướng đối tượng tổng quát với giao diện người sử dụng đầy tiện dụng.Do đó, họ đã mô hình hóa những vấn đề quản lí văn phòng trên màng hình; họ thu xếp mỗi đối tượng chẳng hạn như một trang giấy, một giỏ rác, một máy in, một tủ chứa nơi đặt hàng... bởi biểu tượng; mà từ đó, người ta nhận biết được ý nghĩa của chúng. Các tác động bình thường (như mở các chiếu đô, thiết đặ

tài liệu...) được mô hình hóa như là các tác động giữa các hình ảnh biểu tượng thực trên một giao diện với một bàn viết ảo.

Qua kinh nghiệm thực tiễn, người ta nhận thấy rằng, chúng được đặ ra những ý tưởng dẫn dắt như sau:

▪ Thay vì phải ghi nhớ và phải gỏ nhiều liên hiệp các nút bàn phím cho các tác động lệnh, người sử dụng có thể chỉ ra trực tiếp nhiệm vụ mà anh ta muốn hoàn thiện. Thuộc cái đó, chúng được phát triển thành một thiết bị hiển thị, được gọi là chuột. Số lượng các nút bấm dẫn ra một tối ưu thực hiện lá hai; một liên hiệp gồm có ba nút thường dành cho những dịch vụ phức tạp; còn một nút bấm thì út dùng.

▪Thay vì phải ghi nhớ và phải gõ tất cả các lệnh, chúng có thể được dẫn ra bởi việc áp dụng danh sách lựa chọn, gọi là menu.Mỗi lựa chọn này còn có thể được cấu trúc trải ra một menu kế tiếp. Kết thúc quá trình lựa chọn là một lệnh nào đó.

▪ Số lượng các lệnh phải ít nhất, vì để đảm bảo tác dụng của chúng vừa vạn năng vừa độc lập với nhau, gọi là tính trực giao (*orthogonal*) của các lệnh. Những lệnh sau đây được coi là đạt được nguyên tắc đó: MOVE (di chuyên), COPY (sao), DELETE (xóa), UNDO (phục hồi), HELP(trợ giúp),và showProperties (các tài nguyên thấy được).Thí dụ, người ta có thể sao chép một dòng text ở trong trình Editor cũng như toàn bộ một tài liệu hay một bức hình với lệnh COPY; tuy nhiên, điều này còn ẩn dụ nhiều chức năng khác nhau. Với lệnh ShowProperties, nó nói về sự mô tả một trang dữ liệu có thể thay đổi tương hổ,mà nó cho phép việc xuất-nhập các thông số ngữ cảnh,thí dụ xê dịch một đại lượng nào đó, thay đổi cú pháp hay thay đổi loại.....tại một đối tượng của bài text.

▪ Những lệnh đa năng đã được thu xếp cho các nút bàn phím chức năng, còn gọi là con đường tắt(shortcut), chúng cũng được chỉ ra trên menu với việc nhấp nút chuột phải.

▪ Tất cả các dữ liệu và tài liệu được gọi chung là tài nguyên; chúng xuất hiện nhờ sự thay đổi các dữ liệu đang tồn tại và nhờ sự làm thích ứng các nhiệm vụ đặc biệt cũng như nhờ cấu trúc di truyền một đối tượng. Do đó, không có lệnh CREATE tồn tại; các tác giả cho rằng cái đó quá phức tạp đối với con người.

▪ Người sử dụng cần phải luôn luôn chỉ ở trên một bình diện một chương trình; mà từ đó, anh ta có thể khởi đầu các tác động của mình, chứ không phải chỉ làm việc cố định trên một môdun chương trình, vì tại đó, anh ta không thể biết được nhiều hơn về:cấu trúc nhập dữ liệu tồn tại ở đâu và mọi chuyện xảy ra ở đây như thế nào(?).

▪ Hình dáng của các tài liệu ở trên màn hình phải phù hợp với cách biểu lộ thực tế. Do đó, màn hình được đặt vào như một tờ giấy, công việc còn lại là chọn font chữ và cỡ chữ.

▪ Việc lựa chọn kiểu chữ qua việc rê và nhấp chuột để lựa chọn các kiểu chữ lớn nhỏ khác nhau được thực hiện đầu tiên.

Vấn đề còn lại là, hình ảnh về một thiết bị vật lý được cảm nhận trên biểu tượng (icon) của nó: cái gì sẽ xảy ra với một tài liệu, nếu người ta rê chuột tới một biểu tượng? Có phải nó không bị thay đổi trang viết khi thực hiện sao chép trên máy in? Điều đó thì không thực chất, nó có bị biến mất khi in không? Sau khi nó bị xóa sạch hay sau khi in thì cái gì sẽ xảy ra?

Điều ngạc nhiên là, với một lối dẫn vào, ngày nay người ta đã tìm thấy nhiều phép ẩn dụ của cửa sổ soạn thảo, mà không cần phải lưu ý điều gì cả, lợi ích cụ thể được chứng minh một cách thực nghiệm. Đáng le phải xác định việc điều chỉnh tối ưu đối với các cửa sổ và các biểu tượng (thí dụ về màu sắc, tính tổ hợp.....) trên cơ sở được thiết lập, nó sẽ cho phép người sử dụng tự giải quyết những gì tốt nhất cho mình(thí dụ việc dò tìm giao diện người sử dụng....).

Tuy nhiên, một vài nguyên tắc thiết kế cơ bản đã được đúc kết,chúng được tạo ra như những việc hướng đãn tạo dạng, cũng như, chúng cần được làm đầy cho mỗi giao diện người sử dụng, cụ thể với một yếu tố tồn tại bị thya đổi bản chất của nó, nếu các chức năng mong muốn đã được trợ giúp bởi giao diện. Với Windows 95, người ta có thể dãn tới những thí dụ sau đây:

- Người sử dụng với việc điều khiển (user in control):

Người sử dụng không chỉ đối đáp mà còn phải điều khiển máy tính. Điều ấy có nghĩa là, phải thực hiện điều khiển lối vào công tắc một cách tự động, mà không cần sử dụng trạng thái giao diện sử dụng. Trường hợp không cần thiết phai đi vòng, thì trạng thái này phải rõ ràng và được thoát khỏi một cách dễ dàng.

- Sự phản hồi (feedback):

Đối với người sử dụng,giao diện người sử dụng phai đặt sẵn sàng sự hồi âm khi nhập dữ liệu hay một phản ứng nào đó: thí dụ phải chỉ rõ cho người sử dụng các phương tiện nhìn thấy hay các phương tiện nghe thấy; để từ đó, người sử dụng có thể thực hiện những tác động mong muốn của mình. Nếu không, chúng ta sẽ thấy trống rỗng như một màn hình chết; tức là, nó không phản ảnh cái gì cả về việc nhập đã được thực hiện trước đó. Dưới đây chúng ta nêu một vài ví dụ:

- Con trỏ chuột (mouse pointer)có thể thay đổi hình dáng của nó trong một cửa sổ hay trong khi xử lý.
- Đối tượng có thể thay đổi hình dáng của nó một cách đặc trưng.
- Việc điền vào menu không thể hiện gì cả, nếu ngay khi đó nó bị chặn lại.
- Một dòng trạng thái sẽ thông báo một bước tiến bộ bằng hiển thị số hay hiển thị text .
- Một cửa sổ chuyên dụng sẽ mô tả bước thay đổi của việc xử lý như là một bộ hiển thị của sự thay đổi đó (progress indicator).
- Một cửa sổ thông tin sẽ được nảy ra (directness visual):

Các chương trình xử lý dữ liệu bao giờ cũng chứa đựng một sự tương đương trực giác giữa các dữ liệu và các chức năng cần thiết đã được tạo lập. Do vậy, mục đích là cần phải sử dụng các phép ẩn dụ phổ biến (popular metaphor) cho người sử dụng. Khi đó sự biểu thị của người sử dụng sẽ được thay đổi một cách thuận tiện, anh ta hiểu: anh ta chờ đợi cái gì ở giao diện người sử dụng và với giao diện này

anh ta phải tác động như thế nào(!).Người ta có thể sử dụng các phép ẩn dụ khác nhau, khi người ta muốn mở rộng hay kiến tạo thêm các giao diện này. Thí dụ phép ẩn dụ *work book* (sổ công tác), trong các trang cửa sổ này, người ta có thể đánh số trang, đưa thêm vào hay lấy bớt đi các bức hình hay các bài text; hay thí dụ phép ẩn dụ *pin board* (bảng hiệu) dùng để mô tả các thông tin hay các bài texts giữa mọi người.

◆ Tính cố định(consistency):

Giao diện người sử dụng không cần phải phản ứng một cách đột ngột trước các tình huống tương tự đã quen thuộc. Các tác vụ tương tự cần thiết phải có một sự mô phỏng tương tự và một quá trình tương tự. Sau đây, người ta nêu lên vài ví dụ:

▫ Một menu mà người ta không thể lựa chọn được nữa, thì không cần thiết phải loại bỏ; đặc biệt nó cần thiết được chỉ dẫn thêm bởi việc tạo lập cố định.

▫ Nếu tại các vụ lưu an toàn các dữ liệu ứng với một tiện dụng, các dữ liệu được lưu trữ ngay lập tức; khi đó; một tệp tin phải được đưa vào mục chọn lựa tệp tin của menu; do đó, nó không còn cố định nữa.

▫ Việc nhấp chuột có thể lựa chọn một đối tượng hay một tọa độ riêng lẻ ở trong tất vả các chương trình. Nếu vừa bấm nút chuột vừa rê chuột, do đó, một khoảng sẽ được chọn; thí dụ trong một text hay trong trường đồ họa.

◆ Tính đơn giản (simplicity):

Khi mô tả trực quan, một sự giàn xếp phải luôn luôn được tìm thấy, nhằm trình diễn không quá nhiều mà cũng không quá ít. Về cái đó, có những công nghệ sau đây được kể tới:

▫ Vài lệnh hay vài thông báo ngắn và súc tích được sử dụng.

▫ Hiển thị tiến bộ (progress dislosure): Khi các thông tin được dẫn ra, nếu những thông tin này là cần thiết, thì đầu tiên, các hàm tương quan được phô bày một cách thô kệchóau đó, do nhu cầu, nó được phô bày một cách thanh nhã hơn hay một cách chi tiết hơn.

◆ Tính thẩm mỹ (aesthetic):

Một giao diện không chỉ đáp ứng đầy đủ các chức năng tác động; đối với người sử dụng, những quan điểm trình bày thẩm mỹ cũng rất được coi trọng:một màn hình màu mè thì nhìn đẹp hơn một màn hình xám xịt. Tuy nhiên, về điều này, người ta phải lưu ý: không được từ bỏ tính đơn giản. Vì các kiến trúc rườm rãe làm hỗn độn màn hình. Các yếu tố đồ họa cần phải được tổ chức một cách khái quát và có mục đích. Vì tất cả các yếu tố này phải được chọn lọc trong sự quan tâm của người sử dụng, do đó, người ta phải suy nghĩ thât tốt: cái gì người ta muốn hướng tới và tại sao vậy(?).

7.2. Cấu trúc giao diện người sử dụng

Một giao diện đồ họa phức tạp có thể làm cho người sử dụng thực hành một cách đơn giản hơn để phục vụ cho một chương trình ứng dụng; nó có thể trình bày một hàng rào bổ sung hay làm phức tạp thêm việc dịch vụ. Trên cơ sở đó, điều

quan trọng là phải thiết kế giao diện người sử dụng một cách kỹ lưỡng, và mặt khác, không được thay đổi giao diện đã được dẫn ra.

Những yêu cầu này, được thực hiện trước nhất, không phải mỗi chương trình được tạo ra một giao diện được lập trình một cách tỷ mẩn và độc đáo, sao cho nó không chỉ tiện dụng trong quá khứ, mà một giao diện người sử dụng sẵn sàng được sử dụng bởi một hệ điều hành. Sự tác động cần thiết sẽ trợ giúp nhiều ứng dụng; thí dụ bao gồm việc trợ giúp xuất- nhập nói chung như các nút bấm chức năng, chuột, cửa sổ, các mặt nạ xuất- nhập..... Những chức năng này được đúc kết lại trong một giao diện, gọi là giao diện đồ họa người sử dụng (graphical user interfice:GUI); các giao diện này được người lập trình làm phù hợp bởi một giao diện chuẩn, gọi là giao diện lập trình ứng dụng (application programming interfice: API). Ở hình 7.1 chỉ ra một sơ đồ khái quát về dấu hiệu lớp đối với việc sắp xếp một giao diện người sử dụng như thế.

Hình 7.1.Giao diện người sử dụng và tổng quan cấu trúc hệ thống

Một vấn đề đặc biệt nữa cũng được đặt ra, nếu người ta muốn giới thiệu một giao diện người sử dụng định hướng đồ họa nhờ một thiết bị đầu cuối với khả năng đồ họa giới hạn.Điều này được thực nghiệm với dự án trọng điểm của các công ty chế tạo thiết bị trên thế giới; với dự án này, giao diện người sử dụng được trợ giúp không chỉ bởi một vị trí làm việc với màng hình đồ họa màu; ma f còn với một thiết bị đàu cuối trắng đen theo tiêu chuẩn ASCII. Các bộ điều khiển trình diễn đặt biệt đảm nhận việc mô tả những chức năng xuất (dữ liệu) như việc tạo lập hay việc thực hiện một dòng trạng thái.

7.2.1. Việc nhập (dữ liệu)

Để chuyển vận các bài text hay các lệnh ở trong một máy tính, người ta sử dụng phương pháp chuyển tải qua các nút bàn phím. Nói chung, bên cạnh các chữ cái alphabet thuần khiết, còn tồn tại các nút phím bổ sung để thực hiện những chức năng điều khiển đặc biệt. Bên cạnh các nút phím di chuyển ( ) và các điểm lưu ý về con trỏ (cursor) làm nhiệm vụ nhập dữ liệu, còn có các nút phím để chuyển đổi chữ cái lớn/nhỏ (Shift và Caps LOCK), các phím để di chuyển các dong chữ lui/tới và những nút phím khác để thực hiện lệnh nhập dữ liệu.

Người ta nhận thấy rằng, tổng số liên hiệp các nút phím được tạo ra và đượ mô phỏng bởi bộ vi xử lý bàn phím với bộ mã số nội bộ ; bộ mã này được chuyển đổi bởi bộ kích tạo thiết bị của hệ điều hành thành bộ mã số được chuẩn hóa quốc tế chính thức. Cõ lẽ do nguồn gốc của chiếc máy tính đầu tiên, cho nên bộ chữ cái ở trong máy tínhngày nay dựa theo alphabet của Mỹ, chúng bao gồm 128 kí tự đã được chuẩn hóa. Bộ mã số được mô tả với 7 Bit, nó là bộ mã chuẩn ASCII (American standard Code for Information Inter change), xem hình 7.2 ở dưới đây.

Hình 7.2. Bộ mã hóa các kí tự theo chuẩn ASCII

Ngoài bảng này ra, còn có bộ mã hóa ssó được ghi chép bằng phương pháp mô tả hệ đếm thập lục phân (hexa- decimal). Gía trị của 4 Bit đầu tiên được điền vào ở cuối bên trái theo cột thẳng; còn giá trị của 4 Bit tiếp theo ở phía bên trên theo hàng ngang. Hai hàng đầu tiên của bảng này chứa đựng những kí tự điều khiển, như kết thúc việc nhập EOT, bộ ghi hàng ngang HT, quay trở lại hàng CR...Bây giờ, thế giới không chỉ bao gồm Bắc Mỹ, do đó, điều cần thiết là phải kết hợp các kí tự của Châu Âu, như u,a,x và ê vào trong bộ mã đang tồn tại. Điều đó buộc dẫn tới một bộ mã 8 Bit với chuẩn ANSI do tổ chức ISO (International Standard Organisation) đưa ra.

Tuy nhiên, thế giới không chỉ dừng lại ở châu Âu; điều cần thiết được dẫn tới là phải mô ta các gói phần mềm khác nhau cho thị trường châu Á, như Việt Nam, Trung Quốc, Ấn Độ, Arập....; do đó, người ta phải mở rộng việc mã hóa cacá kí tự nhiều hơn 8 Bit.

Một cuộc thử nghiệm quan trọng theo hướng này là việc phát triển một bộ mã đa năng (unicode), ở trong đó chứa đựng toàn bộ các kí tự của các thứ tiếng của các quốc gia trên thế giới, goi là UNI1997. Ở trong bảng này còn chừa các chỗ trống, do đó rất tiện lợi cho việc mở rộng tiếp theo. Hình 7.3 chỉ ra một sự dẫn giải về bộ mã đa năng này: bắt đầu với mã 16 Bit 000_H và kết thúc với $FFFF_H$.

Hình 7.3. Sự dẫn giải về bộ mã đa năng(UNI- code)

Người ta nhận thấy rằng, bộ mã đa năng đã được thiết kế như là sự mở rộng bộ mã ASCII, nhằm tạo nên một sự hài hòa với các hệ thông chuẩn đang tồn tại. Tuy nhiên, có rất nhiều kiểu chữ (fonts); cho nên, ở bộ mã đa năng 16 Bit, người ta có thể đảm bảo đầy đủ các font chữ, vì các thông tin được tách chia một cách mạnh mẽ, bất kì khi nào và cách sử dụng như thế nào đều chứa đựng trong bộ mã này cả. Tất cả các thông tin tạo dạng (font chữ, kiểu trình bày, đậm/nghiêng....) đều được bộ mã kí tự quản lí một cách tách biệt; chúng là các kiểu dùng cho trình soạn thảo. Bộ mã đa năng đã được chuẩn hóa(ISO 10646) và không ngừng được phát triển.

Bộ mã quan trọng nữa là bộ mã Unix nâng cao 32 Bit (Extended Unix Code: EUC);nó giới thiệu một sự mở rộng của kiểu kí tự theo chuẩn ASCII dùng cho hệ điều hành Unix. Các kí tự châu Á và các kiểu phức hợp khác được dẫn ra như là kết quả của nhiều chữ cái alphabet theo chuẩn ASCII trên bàn phím thông thường và chúng cũng được lưu trữ như chuẩn EUC của các nước đặt biệt.

Thí dụ về bộ mã kí tự nhập ở Unix:

Từ buổi đầu, ở hệ điều hành Unix cũng như POSIX, kiểu kí tự chuẩn US-ASCII được dùng một cách rộng rãi. Gần đây, trong các ấn bảng mới hơn, các kiểu này đã bao hàm các đặt điểm riêng biệt của từng quốc gia. Thí dụ, công ty Hewlett- Packard đã tạo lập được một sự đặt biệt, gọi là sự trợ giúp ngôn ngữ thiểu số(native language support:NLS). Nó bao gồm một dãy các biến số ngoại vi (LANG, LC_XX), nếu chúng tôn taịi trong phạm vi tiến trình người sử dụng và nếu chúng được thiết đặt một cách hợp lí; khi đó, chúng điều khiển phần cục bộ các lập thức thư viện (library routine)và các lệnh cảu hệ điều hành(ed,grep...).Ở đây, chúng không chỉ bao hàm các đặc điểm ngôn ngữ đặt biệt như việc sử dụng đúng đắn bộ mã 32 Bit khi so sánh chuỗi có tự hay nhận dạng mã chữ cái(chữ cái lớn/nhỏ, chữ số, kí tự điều khiển.....), mà còn bao hàm cả việc ưngs dụng hợp lí đối với từng quốc gia cho điểm và dấu phẩy khi biểu diễn số cũng như đăc điểm tiền tệ của từng nước(thí dụ biễu diễn đồng đô-la tiền tệ của Mỹ:1.568,74US$).

Thí dụ về bộ mã kí tự ở Windows NT:

Ngược với hệ điều hành Unix, ở Windows NT, khi thiết kế bộ mã kí tự nhiều quốc gia trên thế giới, đối với hệ điều hành, người ta khẳng định rằng, đó là một bộ mã đa năng. Điều này có nghĩa là tất cả các chuỗi kí tự,tên đối tượng, tên đường dẫn.... đều phải dùng bộ mã kí tự đa năng.

Các đăc trưng của từng quốc gia như múi giờ, ký hiệu tiền tệ, tiếng nói....được khẳng định một cách đọc lập với cái đó khi tạo lập hệ thống và được lưu trữ tập trung.

Bên cạnh bàn phím dùng để nhập các kí tự, ngày nay còn có nhiều khả năng nhập khác nữa. Sau đây, người ta nêu ra một vài phương pháp.

- **Các phím chức năng** (function key):

Việc mở rộng đơn giản dẫn tới khả năng, để sử dụng các nút phím cho các chức năng đăc biệt. Đó là một bước quan trọngđối với việc thiết kế giao diện người sử dụng, vì ở đây, bên cạnh việc nhập các dữ liệu bằng số và chữ cái, việc làm thích ứng các chức năng cũng được tạo điều kiện. Do đó, một cách lôgic, điều quan trọng là phải giữ vững các dữ liệu và các chức năng được tách biệt trong một chương trình tại môi trường nhập, và phải tránh mọi sự nhầm lẫn. Chỉ có một giao diện người sử dụng tồi mới pha tạp hai chức năng, thí dụ việc ngừng xuất dữ liệu phải ấn hai phím CTRL+S, nhiều khi còn tệ hơn, phải điều khiển chương trình bằng nhiều phím kí tự riêng lẽ.

- **Thiết bị chỉ thị**(pointer device):

Có một bước quan trọng để nhập tương tự, đó là việc dẫn vào các thiết bị chỉ thị như chuột, banh dò (track ball). Ở đay, vị trí của con trỏ chỉ thị trên màn hình có thể được điều khiển một cách nhe nhàng hơn ấn các nút phím chức năng (cursortaste).

◈ **Các mâm đò họa** (graphic tablet):

Một sự trợ giúp to lớn đem lại việc xác định vị trí của một bút chì trên một bảng nhập điện tử đăc biệt. Tuy nhiên, chức năng của nó là một thiết bị chỉ thị, do đó, người ta có thể truyền đạt các hình dạng vào máy tính một cách trực tiếp; thí dụ các dữ liệu cảu các dự án kiến tạo hay các mẫu y khoa, các hàm chức năng và các biểu đồ hay có thể sử dụng các chữ kí để kiểm duyệt ngân phiếu.

◈ **Thiết bị quét hình**:

Trong vài năm lại đây, các bảng đồ họa có được nhờ việc áp dụng các thiết bị dò quang điện với độ phân giải cao, thiết bị này gọi là máy quét (scaner) hình hay đồ họa. Tuy nhiên, cùng với các khuynh hướng tân tiến khác, các kết quả quét đồ họa không chỉ được lưu trữ trên giấy, chúng còn được biến đổi thành tín hiệu điện để phục vụ những yêu cầu của nghành công nghệ thông tin và các lĩnh vực khoa học khác.

7.2.2. Mành đồ họa và sự phân giải

Đối với việc xuất ra một bảng phác thảo đồ họa thường được sử dụng một mô hình đặt biệt, gọi là mô hình mành đồ họa. Người ta sử dụng một hệ thông tọa đọ cho các dấu vẽ phác; chúng xuất hiện như một mành màn hình. Thật vậy, người ta gọi điểm ra(0,0) của tọa độ(x, y) là ở góc trên bên trái; ở đó các dấu vết của các điểm ảnh được thu xếp giống như một ma trận điểm. Đối với màn hình có kích cỡ 1024x786 điểm, thì ma trận tọa độ các điểm được dẫn ra như trong hình 7.4 ở dưới đây.

Hình 7.4. Tọa độ các điểm của mành đồ họa

Mô hình mành đò hạo thì khác với phương pháp vec-tơ đồ họa; phương pháp vec-tơ đồ họa chỉ quan tâm tới các tọa độ của các điểm hình học (thí dụ điểm đầu và điểm cuối của một đường thẳng). Ngược lại, phương pháp mành đồ họa phân bổ mỗi điểm một giá trị màu đã được định nghĩa, các giá trị này được lưu trữ như những con số. Khi nhớ lặp lại hình kiểu đồ họa mành phải dự định trước một đơn vị bộ nhớ; còn trương hợp vec-tơ đồ hạo chỉ cần nhớ từng điểm ảnh của màn hình đồ họa. Cho nên trước đây, người ta ưa chuộng phương pháp vec-tơ đồ họa, vì phương pháp này sử dụng it không gian bộ nhớ. Tuy nhiên, ở việc làm tươi (refresh) các đồ họa phức tạp, phương pháp vec-tơ phải làm tươi tất cả các phần tử đồ họa tong danh sách, do đó, quá trình này xảy ra rất chậm; ngoài ra nó còn dẫn tới những điểm lổ chỗ trên bức đồ họa vừa làm tươi.

Công nghệ hiện đại của các màn hình mành kết nối với kiểu dáng màn hình do người lập trình tự kiến tạo, đã tạo nên chỗ đứng vững vàng cho công nghệ thiết kế các phần mềm dồ họa. Ở đây, mỗi điểm ảnh được mô tả thành các màu sắc và

được biểu diễn bởi một giá trị màu từ các Bit, gọi là các Bit màu(b_0,b_1,.....,b_n). Đại đa số các Bit có thể được kết hợp với từng chỉ số giông nhau của bộ điều khiển trình diễn (display controler) tới một bức ảnh đồ họa. Một bức ảnh như vậyđược biểu thị một mức bình diện nào đó.

Thí dụ về các mức màn hình:

Người ta thấy rằng, mỗi điểm ảnh của bức đồ họa thường chứa đựng trong 8 Bit (ứng với một byte). Nếu chúng ta phân đoàn 8 Bit này thành 6+2 Bit; với chủ định, đối với một phần mềm đồ họa, điều cần thiết là phải thiết đặt một mặt trước (front) khoảng 2^6 màu (cho hình ảnh), còn mặt nền(underground) koảng 2^2màu. Về các lệnh phần cứng, chúng ta có thể di chuan các dữ liệu của một mức (đối với mặt trước); cho nên, một hình(figure) đồ họa có thể xuất hiện và di dich trên một nền phía sau cố định(tương tự như một phong cảnh: xe cộ chuan động, núi non đứng yên). Khi trình diễn các phần mềm trò chơi (gamé) trên máy tính, người ta thấy các hình ảnh chuan động chính là những hình vuông nhỏ tạo dạng được điều hưởng trên màn hình.

Các màu sắc được pha trộn từ ba màu cơ bản là đỏ (rose), xanh lá cây(Green) và xanh da trời (Blue). Vì cự kết hợp cường độ ba màu (mmỗi màu có 8 Bit)thành một tổng duy nhất (24 Bit) thì cần dùng quá nhiều Bit; cho nên, đối với một điểm ảnh cần một số lượng đề chỉ cường độ ba màu cơ bản (R,G,B) thì không cần thiết; mà người ta chỉ viết chỉ số của ba màu, gọi là địa chỉ màu(color address) vào trong một bảng đăc biệt, gọi là bảng chiếu màu(color lookup table:CLUT). Sơ đồ truy cập đối với các giá tị màu của một điểm ảnh khi phô diễn thì chuan thành địa chỉ của một mức, tương tự như đã trình bày trong mục 3.3 ở chương trước. Ở hình 7.5 là một ví dụ, nó chỉ ra dngf dữ liệu cho một điểm ảnh có màu tím sáng(bright violet) với giá trị màu (R, G, B) = (215,175,240) ở vị ttrí (x, y), Người ta lưu ý, mỗi điểm ảnh chỉ dùng 3 Bit của không gian bộ nhớ; nhưng về độ chính xác mau phải dùng tới 24 Bit. Sự hạn chế của việc mô tả các điểm ảnh trên 3 Bit không chỉ tác đông tới độ chính xác màu(color accuracy), mà còn tác đông tới chỉ số lớn nhất trong bảng chiếu màu CLUT; vì vậy, nó cũng ảnh hưởng tới số lượng các màu có thể được trình bày đông thời.

Để trình bày các giá trị màu (R, G, B) ở dạng số thành các giá trị màu tương đương ở dạng tương tự; khi đó, người ta dùng một bộ biến đổi điển tử số/ tương tự (D/A converter); thí dụ các giá trị màu 0.....255 tương ứng với các mức điện áp 0,0....1,0 volt; với các giá trị này, các nguồn tạo màu(thí dụ nguồn điện quang) được điều hưởng.

Hình 7.5. Xác định giá trị màu nhờ bảng chiếu màu

Tuy nhiên , ý tưởng này vẫn còn nhược điểm trong các vùng điểm ảnh. Những ảnh hưởng của các chức năng thì phụ thuộc rất nhiều vào phần cứng(tức là sự phân giải cụ thể của màn hình). Ở đây, các ô hình vuông nhỏ và các cửa sổ nói

chung đươch phân giải một cách phù hợp với kích thước màn hình; cụ thể, nó không phải là cac nét vẽ liên tục đầy đủ, kà nó chỉ là những khối điểm ảnh được xác định theo mong muôn của người lập trình. Bây giờ, nếu chúng ta pha lẫn đồ họa và bài text (bao gồm các kí tự); do đó, vị trí của bài text ở tong cửa sổ đồ họa phụ thuộc vào độ phân giải của màn hình; khi đó có thể dẫn tới các hiệu quả của bảng thiết kế nói chung không được cân đối và đẹp đẽ; đồng thời, nó cũng gây trở ngại cho một giao diện của người sử dụng sau này.

Hình 7.6. Mô tả cách thực hiện một bảng đồ họa

Một quyết định quan trọng để dẫn tới các nét vẽ định hướng điểm ảnh là nét vẽ phân giải. Ở các net vẽ này, chẳng hạn một chữ cái thì không đăc trưng bởi các điểm ảnh của nó, mà nó đăc trưng bởi hình biên và màu sắc cung như cấu trúc(texture) diện tích chứa đựng các điểm ảnh. Điều này thì trước hết là bất tiện và không hiệu nghệ khi lưu trưx và trình bày; tuy nhên, khác với trường các điểm ảnh, việc mô tả đương biên được phóng to hay thu nhỏ một cách bất kì. Hình7.6 cho thấy: bên trái mô tả định hướng các điểm ảnh (a); bên phải mô tả mức độ phân bổ cho chữ A (b). Do đó, những điểm ảnh mà người ta nhìn thấy, nó vẫn còn thô kệch.

7.2.3. Hệ thống cửa sổ và quản lý trình diễn

Một trong các phương tiện trình bày quan trọng nhất được hệ thống các máy tính Xerox-START đảm nhiệm, đó là sự biểu thị các dữ liệu trong các mặt cắt hình vuông của màn hình hay trong các cửa sổ. Ngày nay, kỷ thuật này đã trở nên phổ dung ở tất cả các máy tính; nó dẫn tới khả năng phải xác định địa chỉ bộ nhớ lập ảnh của một đơn vị biểu thị (thí dụ thiết bị đầu cuối, màn hình đồ họa...); do đó, cần phải lái việc xuất cấu trúc tế vi của một tiến trình trên một phần được dịch vụ xác định của màn hình. Nếu trước đây các chuỗi ký tự được dẫn ra bởi một ký tự đặc biệt (thí dụ ESC) thì đồng thời chúng cũng đạt được một lệnh cho bộ vi xử lý trình diễn; do đó, điều này đã được thay thế tại các hệ thống cửa sổ nhờ các chức năng tổng hợp; ở đó, các số liệu cửa sổ (kích thước, vị trí, mặt cắt...) có thể được thay thế một cách tương tác nhờ các thiết bị nhập (chuột, cần lái...).

Cấu trúc tiến trình của một ứng dung đồ họa đã được biến đổi không ngừng. Nếu trước đây, chương trình ứng dung còn chứa đựng tất cả các thủ tục xuất đồ họa như là một thư viện (xem hình 7.7); do đó, ngày nay hầu như điều này đã được tách biệt.

Hình 7.7.**********

Công việc trình diễn như xác định vị trí chuột, kích chuột, dẫn một bài text...được quản lý bởi một tiến trình xác thực hay bởi một cửa sổ điều hành (Window manager), (xem hình 7.8). Cả hai tiến trình sau đây được kết nối với

nhau theo kiểu quan hệ client/ server: tiến trình người sử dung đại diện cho một client, còn tiến trình của cửa sổ điều hành đại diện cho một server với sự trình diễn các đồ họa mong muốn.

Do đó, việc quản lý được thiết kế một cách căn bản như một chương trình lặp đi lặp lại; mà tại đó, các hoạt động được thực hiện như một phản ứng khi trình diễn của người sử dung.

```
InitProcess
LOOP
  WaitForEvent (Mouseclick, KeyBoard, DisplayUser Programms,...)
ExecuteNecessaryProcedure;
END (*LOOP*)
```

Hình 7.8**********

Hệ thống các tiến trình cửa sổ gần như được điều khiển một cách biến đổi, mà ở đó, cũng giống như các biến cố không đồng bộ, không chỉ các việc trình diễn được thực hiện từ bên ngoài (nhấp chuột), mà cả các công việc đồ họa của chương trình ứng dung cũng được thực hiện từ bên ngoài. Ơ đây, hệ thống các biến cố trao đổi thông tin nhận được sự trợ giúp, cho nên nó đã đem lại sự trình diễn và các nhiệm vụ khi tạo dựng thông tin; và do đó, nó cũng móc nối vào hàng đợi trung tâm của cửa sổ quản lý.

Việc phân bổ các chức năng giữa chương trình người sử dung và cửa sổ quản lý làm yêu cầu thêm về việc tách chia các nhiệm vụ một cách rõ ràng; ở đó giữa các nhiệm vụ chuẩn của cửa sổ quản lý (các nhiệm vụ của giao diện người sử dụng) và các yếu tố đồ họa đặc biệt của chương trình người sử dung được phân biệt với nhau, tức chúng có những ưu điểm khác nhau như sau:

- Các kết quả của nhiều tiến trình độc lập có thể đúc kết trong một hệ thống cưả sổ nói chung; do đó, chúng được làm sáng tỏ từ hai ví dụ sau đây:
  - Kết quả do các bộ cảm biến (sensors) khác nhau trong điều khiển công nghiệp thì khác nhau.
  - Việc mô tả song song các kết quả tại nhiều chương trình với chức năng giống nhau thì độ chính xác lỗi phần mềm sẽ muôn màu muôn vẽ.

- Hệ thống trình diễn đồ họa có thể được chuan dịch trên các máy tính khác nhau ở trong mạng máy tính, xem mục 7.2.5. ở dưới đây. Điều này tạo ra những chức năng sau đây:

  - Kiểm tra mạng: trên mỗi máy tính tồn tại một tiến trình đại diện (agent); trong sự độc lập với một chương trình trung tâm, tiến trình đại diện này biểu thị trạng thái của máy tính trong một cửa sổ riêng lẻ.

- Quản lý mạng: việc phân bổ và quản lý các phần mềm có thể được tổ chức bởi một không gian làm việc tập trung; trong đó, cụ thể trên mỗi máy tính có một tiến trình làm cân bằng thiết bị đầu cuối tồn tại như một tiến trình đại diện; mà tiến trình này phô bày việc xuất dữ liệu trên màn hình trình diễn trung tâm.

➢ Việc quản lý cửa sổ được điều chỉnh tập trung; còn đối với một áp dung, việc nhìn thấy và cảm nhận (look and feel) thì giống nhau.

➢ Việc sử lý các biến cố có thể được chuan giao cho một hệ thống con (chẳng hạn cho một server); và do đó, nó làm giảm phụ tải cho bộ vi xử lý cuả các ứng dung một cách thực thụ

Ý tưởng này có nghĩa rằng, trạng thái hiện hành của một cửa sổ thì không chỉ quen thuộc đối với chương trình người sử dung; mà còn cho thấy, các giá trị của các biến trạng thái (như vị trí cửa sổ, kích thước cửa sổ...) phải đượctạo lập một cách rõ ràng bởi một server.

Các cửa sổ có thể xếp chồng lên nhau hoặc có thể bị che phủ từng phần. Đối với các phần không được biểu thị của các cửa sổ , có hai ý kiến:

- Một là, cuả sổ được server lưu trữ, do đó, kể từ khi cửa sổ được mở ra cho tới khi một phần cuả sổ bị che phủ (thí dụ khi mở rộng một cửa sổ, hay khi di chuyển cửa sổ chồng lên trên) thì nội dung cửa sổ được điền vào một cách tự động. Điều này đòi hỏi nhiều dung lượng bộ nhớ đối với tiên trình server; nhưng, nó tránh được các thủ tục bổ sung để điền vào nội dung cửa sổ còn thiếu ở trong chương trình người sử dung.
- Hai là, trong trường hợp có nhu cầu, chương trình ứng dung phải biểu thị các phần cửa sổ còn thiếu một cách mới mẻ. Điều này đòi hỏi dung lượng bộ nhớ ít hơn, vì vậy, người ta có thể sử dung các thủ tục chuyên dung ở chương trình người sử dung. Cho nên, nó dẫn tới một sức chịu tải động học cao hơn cho bộ vi sử lý người sử dung.

Việc thực thi cụ thể thì phụ thuộc vào hệ thông cửa sổ được sử dung. Nếu người ta phó mặc quyết định này cho cấu hình tồn tại trong thời gian làm việc của server(tạo lập việc lưu trữ) hay cho người lập trình người ứng dung; do đó, có thể xuất hiện trường hợp, khi lập trình ứng dung cũng như khi lưu trữ trên server, nó sẽ được giảm thiểu và do đó, việc ứng dung sẽ xảy ra lỗi ở vùng cửa sổ được thiết lập; tuy rằng việc lập trình chanứg có lỗi gì cả. Do đó, mục đích là phải quy định các quy ước chặt chẽ cho sự phô diễn.

7.2.4. Hiện thực ảo

Mới đây, để mô phỏng hoàn toàn các cảnh chuyển động thì cần phải dẫn xen vào những thiết bị xuất nhập đặc biệt; những thiết bị này tạo ra không gian 3 chiều (*3 demension:* 3D) trong giao diện người sử dụng. Cụ thể, bao gồm các thứ tự như chuột, các bộ chỉ thị…; những thứ này cho phép điều chỉnh vị trí theo 3 chiều; ngoài ra còn có các thiết bị nhập, mà vị trí đầu dò của nó (tracker) được xác định. Những hệ thống tân tiến hơn còn xác định được trạng thái biểu lộ của con người nhờ hệ thống các camera và các sensor cảm biến khoảng cách. Nhờ đó, các kiểu trình diễn như động tác khoa chân múa tay (*gestic*) hay động tác bắt chước (*mimic*) là có thể thực hiện được một cách dễ dàng. Ngay cả kỷ thuật trình diễn đồ hoạ cũng được bổ sung thêm nhờ các hệ thống kính đặc biệt hay các thiết bị trình diễn khác; vì vậy người ta có thể điều hưởng việc trình diễn bất cứ như thế nào theo ý muốn của con mắt, để có một không gian đầy ấn tượng xuất hiện trước mặt khán giả.

Ngoài các khả năng cho đến này chưa được làm đầy và ngoài chi phí kinh tế và kỷ thuật còn quá cao; tuy nhiên, kỷ thuật hiện thực ảo không gian mang lại những suy nghĩ mới cho giao diện người sử dụng, mà chúng chỉ làm hoàn hảo thêm và tạo điều kiện cho những áp dụng mới, mà với điều kiện này, việc cần thiết phải dẫn tới những vấn đề bổ sung thêm cho trường hợp điều khiển từ xa. Tất cả các khả năng và các vấn đề được mô tả trong các mục trước được chuyển tải một cách trực tiếp trong ngữ cảnh ba chiều (3D); do đó, đối với giao diện 3D, việc phân loại hệ thống nêu ở trên được coi là hoàn mỹ trong một cấu trúc client/ server.

7.2.5. Quản lý giao diện người sử dụng

Vai trò đặc biệt của giao diện đồ hoạ là chiếm lĩnh việc quản lý các đối tượng, mà người gọi là hệ thống quản lý giao diện người sử dụng (*user interface management system:*UIMS). Việc quản lý này cũng chính là quản lý các gói thông tin phần mềm. Với sự trợ giúp của việc xuất nhập bằng các thiết bị khác nhau (như chuột, bảng đồ hoạ, loa âm thanh…), người ta có thể phân đoạn các thủ tục đồ họa thuần khiết để nhằm dễ dàng quản lý những biến cố khi nhập và những mong muốn khi xuất. Hình ảnh thông thường của giao diện người sử dụng chính là sự trình bày hình thái các đối tượng đồ hoạ (thí dụ thu hẹp một cửa sổ hay sử dụng các biểu tượng…), điều này đạt được do có bước quản lý này. Giao diện này thì độc lập với giao diện đồ họa của người sử dụng (*graphical user interface:* GUI). Giao diện đồ họa của người sử dụng được mô tả nhờ một danh sách gồm các thủ tục, các đối tượng và các giao thức.

Để thiết lập một sự quản lý như thế có nhiều phương pháp khác nhau. Đó là việc trình bày các quy tắc cho các biến cố và thiết lập việc quản lý khi lập trình trong hệ thống quản lý giao diện người sử dụng.

Thí dụ về cách biểu lộ:

Nếu một biểu tượng (icon) được dùng cho một tài liệu, nó sẽ được chọn (bằng cách nhấp nút chuột), được lôi đi và sau đó được buông ra tại một vị trí mong muốn. Việc lôi và thả này (*drap and drop*) tác dụng lên hệ thống quản lý giao diện người sử dụng; khi đó, tài liệu chi tiết được di trượt và được biểu thị tới vị trí muốn thiết đặt của người sử dụng.

Kết quả của gọi hệ thống cũng như kết quả của các hình vẽ đồ hoạ của biểu tượng (thí dụ biểu tượng về các tài liệu văn bản thì dấu mình trong cửa sổ về chủ đề *printing,* biểu tượng về máy in printer thì dẫn tới trạng thái in, khi máy in làm việc một tờ giấy in tài liệu được dẫn ra chậm chạp…) được đảm nhiệm bởi hệ thống quản lý giao diện người sử dụng và chúng thì giống nhau ở tại các chương trình người sử dụng.

Cách làm như vừa nói có thể được viết bằng các ngôn ngữ lập trình hay ngôn ngữ logic tả thực. Ưu điểm của hệ thống quản lý giao diện người sử dụng là ở chỗ, nó có thể thay đổi một cách dễ dàng và tập trung. Cho nên, việc làm thích hợp các chức năng cần thiết cũng như các việc sửa lỗi… có tác dụng cố định giống nhau ở tất cả các chương trình người sử dụng.

Khác với các thành phần của giao diện người sử dụng chuẩn, đối với các chương trình ứng dụng, các giao diện chuẩn này đều giống nhau và hầu như chúng cũng được phân loại bởi các nhà thiết kế hệ điều hành; do đó, tình trạng này có vẻ khác nhau ở các thành phần đồ họa chuyên dụng. Ở đây, sự nỗ lực hết sức của người lập trình là rất cần thiết, nó được tận dụng ở ngay mỗi chương trình. Đồng thời, để đơn giản hóa công việc lập trình này và để cho phép việc lập trình tương tác mà không cần biểu thị về các chức năng của giao diện người sử dụng, người ta nhận được nhiều phương pháp khác nhau:

✓ **Tạo lập các tệp tin đặc biệt** (*recource files*):

Trong trình soạn thảo đồ họa hay trong hộp dụng cụ kết cấu tài nguyên của hệ thống quản lý giao diện người sử dụng, người ta có thể kết hợp với nhau một cách phù hợp các yếu tố như menu, biểu tượng (icon), các nút chọn (*select button*), các biến cố chuột, các chuỗi âm thanh (*tone sequence*)… Các biến cố sẽ được dẫn tới như các cấu trúc tệp tin ở trong các tệp tin nguồn và được nạp hay được sử dụng trong thời gian vận hành hệ thống quản lý giao diện người sử dụng. Các tên (được quy định ở các trình soạn thảo)của các đối tượng hình thành nên các chương trình người sử dụng, nhằm áp dụng các đối tượng thích hợp có sự trợ giúp của giao diện đồ họa người sử dụng.

✓ **Tạo lập mã chương trình** *(programm code)*:

Song song với việc tạo lập về các tác động đồ họa (tương tác trực quan) và các phản ứng với sự trợ giúp của một trình soạn thảo đồ hoạ, điều cần thiết là phải mô tả một cách đầy đủ các gọi chương trình (*program call*) cho giao diện đồ họa người sử dụng bằng một ngôn ngữ lập trình nào đó thành một tệp tin. Nếu tệp tin được biên dịch, chúng ta sẽ nhận được một bức ảnh đồ họa trong hệ thống quản lý

giao diện người sử dụng. Một thí dụ đặc trưng cho hệ thống lập trình, lập trình kiểu này là ngôn ngữ lập trình DELPHI của hãng Borland; hệ thống này được tạo lập bởi mã chương trình trong lập trình hướng đối tượng PASCAL.

7.3. Hệ thống cửa sổ và hoạ tiết ở Unix

Giao diện người sử dụng trong Unix đã được tiêu chuẩn hoá trong môi trường soạn thảo cơ sở (*Common Desktop Envionment:* CDE) như UNIX-98 và UNIX-99. Việc thực thi có ý nghĩa nhất đối với các hệ thống Unix này liên quan đến một hệ thống các cửa sổ đặc biệt, gọi là hệ thống X-Windows. Sau đây chúng ta sẽ nghiên cứu hệ thống cửa sổ này tác dụng như thế nào (?)

Một trong các dự án quan trọng của những năm 80 là dự án ANTHENA của viện công nghệ Bossten; tại đây, người ta bắt đầu thử nghiệm nhằm thiết đặc một phạm vi công việc cho các máy tính nối mạng với nhau. Một phần quan trọng của dự án đã nhận được sự tài trợ của hảng Intel; đó là việc thiết lập một giao diện người sử dụng; giao diện này được tồn tại ở trong mạng một cách phân bổ và độc lập với máy tính đang dùng. Người ta gọi phần này là sự kế tục của một hệ thống cửa sổ trước đó (Window System) được phát triển một cách đơn giản để trở thành một hệ thống mới có tên *X-Window-System*.

Trong các mục sau đây, chúng ta muốn xem xét hệ thống này một cách kỹ càng hơn. Khi đó, chúng ta sẽ đạt được việc làm sáng tỏ các mô hình về một bản phác thảo đồ họa cơ bản.

7.3.1. Phác thảo đồ họa kiểu client/ server với hệ thống X-Window

Hệ thống X-Window thì bao gồm một thư viện đồ hoạ Xlip; thư viện này được trình tiện dụng các tiến trình server sử dụng; tiến trình server chứa đựng cửa sổ quản lý và các công cụ thực thi xuất nhập trên giao diện người sử dụng.

Hình 7.9**********

Bản phác thảo đồ họa kiểu client/server được thực hiện một cách đơn giản trong thư viện Xlib. Để tạo ra một sự kết nối với server cũng như để mở đầu cuộc trao đổi, client phải thực hiện một gọi hệ thống XOpenDiskplay(); gọi hệ thống này chứa đựng tên máy tính và số liệu màn hình như là một đối số; tất cả các nhiệm vụ tiếp theo được tiến hành một cách tự động ở địa chỉ này. Nếu nhiều tiến trình cùng được thực hiện gọi hệ thống này trên các máy tính khác nhau, do đó, nhiệm vụ của các tiến trình này đều xảy ra trên một và chỉ một server mà thôi. Những thông tin có lợi ở trong mạng có thể được chỉ rõ ở trên màn hình nói chung, mà trước đó không cần phải tu chỉnh bởi chương trình người sử dụng.

Các thông tin về vị trí và kích cở của cửa sổ được người sử dụng sắp xếp một cách tương tác trên màn hình, các thông tin này thì chỉ quen thuộc đối với server.

Một tiện dụng muốn cho thấy các dữ liệu và cửa sổ của nó, trước heet, nó phải bắt đầu dò tìm với lệnh XGetWindow().

7.3.2. Phác thảo cửa sổ với hệ thống X-Window

Thư viện Xlip chỉ chứa đựng những chức năng đồ họa đơn giản với các hình thức trình diễn khác nhau. Đối với một cửa sổ tổng hợp với thanh trượt, xem hình 7.10, người sử dụng một số lượng lớn các gọi hệ thống của các chức năng sơ cấp này.

Bản phác thảo logic về việc tạo lập và đặt tên cửa sổ đối với hệ thống X-Windows là rõ ràng và đơn giản. Xuất phát từ một cửa sổ cơ sở, tất cả các cửa sổ (được định nghĩa theo đó) được biểu thị bằng một hình chữ nhật trong cửa sổ cơ sở này; và như đã nói, cái đó trở nên như những mẫu bài báo tách biệt (*clipping*).

Dưới khái niệm *cửa sổ*, người ta hiểu đó là một hình chữ nhật sơ cấp. Hình chữ nhật này chiếm một đường biên và mẫu nền, nó được định nghĩa trên bình diện của thư viện Xlip. Hình 7.11 chỉ ra một hệ thống các cửa sổ khác nhau; ở đây, mỗi cửa sổ được biểu thị bởi một chữ cái.

Mỗi cửa sổ có thể chứa đựng một hay nhiều cửa sổ con (*subwindow*), kiểu cấu trúc này được biểu thị qua một cây, mà điểm gốc là cửa sổ cơ sở (*root window*). Cửa sổ cơ sở là toàn bộ cửa sổ của màn hình.

7.3.3. Mở rộng kiểu dáng đồ họa

Trên cơ sở các chức năng đơn giản của thư viện Xlib, các lớp mới cao hơn phải được thiết lập với các chức năng cao hơn nữa. Lớp trung gian giữa các gọi hệ thống của chương trình người sử dụng và các chức năng được chi tiết hoá của thư viện Xlib gọi là Xtoolkit (hộp công cụ) và nó có thể được thực hiện rất khác nhau. Các đối tượng đồ họa của lớp này được gọi là Dialogobjects (các đối tượng hội thoại), chúng được tạo lập rất khác nhau và tuỳ thuộc vào người lập trình. Để giữ vững các ưu điểm của một giao diện người sử dụng thống nhất trên bình diện cao hơn và để tiêu chuẩn hóa các họa tiết (*motif*) và các tính chất (*look and feel*) của các đối tượng hội thoại, một giao diện người sử dụng thống nhất cho Unix được để cập. Đó là cơ sở phần mềm đồ họa mở rộng (*open software foundation*), chúng được dẫn giải theo một hiệp định của các nhà sản xuất. Bên cạnh bản hướng dẫn tạo kiểu dáng (*motif style guide*), còn có một bản mô tả về việc quản lý cửa sổ cần thiết (*motif window manager*), chúng bao gồm những đối tượng hội thoại nhằm mở rộng kiểu dáng (*motif widget*) khi lập trình hay thiết kế đồ họa.

Tuy nhiên, các mô hình tạo màu và tạo ảnh mành đồ họa có khuynh hướng loại bỏ một cách trực tiếp các chức năng của thư viện Xlib. Mành đồ họa này làm việc với các bộ fonts điểm ánh (*pixelfonts*) chưa được xử lý. Từ lý do này, hãng máy tính SUN đã thiết đặt một hệ thống cửa sổ mạng máy tính (*network-window-system*: NEWS). Hệ thống này là cơ sở để mở rộng ngôn ngữ mô tả trang (*post-script*) cũng như để sử dụng các fonts và các đối tượng đồ họa đã được xử lý.

Ở trong hệ thống X-Window, với các đối tượng hội thoại, người ta nhận được các đặc điểm cơ bản sau đây:

+ Các nút bấm được mô phỏng có thể ấn lên xuống được (XmPushButton).
+ Các bài texts (cố định) có thể trình diễn được (XmText).
+ Các thanh trượt có thể làm dịch chuyển nội dùng trên cửa sổ (XmScrollBar).
+ Các trường nhập về đồ họa có thể di chuyển được (XmDrawingArea).
+ Các tồn tại một cửa sổ chọn tệp tin (XmFileSelectionBox).

Độc lập với các tác động họa tiết đồ họa, hảng máy tính SUN còn tạo thêm hộp công cụ Xview, còn hảng máy tính AT&T tạo thêm hộp công cụ Xt^+, cả hai đều được biên soạn trong thư viện Xlib.

Giao diện dùng lập trình ứng dụng được mô tả bởi các họa tiết nhờ một ngôn ngữ đã được chuẩn hoá, gọi là ngôn ngữ giao diện người sử dụng (*user interface language*: UIL). Trong ngôn ngữ này, các yếu tố của giao diện người sử dụng cần thiết có thể được tiếp nhận để mở rộng hàm số và hình học. Hình 7.13 chỉ ra sự phân lớp của phần mềm đồ họa mở rộng

Hình 7.13**********

Đối với những nhiệm vụ khác nhau, hộp công cụ có những công dụng sau đây:

+ Việc tạo ra hay xóa đi các họa tiết của bản thiết kế đồ họa mở rộng có thể thực hiện một cách năng động.
+ Việc thay đổi các họa tiết mở rộng có thể thực hiện bất cứ khi nào trong thời gian quản lý cửa sổ.
+ Việc xuất- nhập hay việc tạo lập một sự trình diễn quay hồi được quản lý tập trung.
+ Các cơ chế trao đổi thông tin giữa các ứng dụng cũng như giữa các cửa sổ của chúng luôn luôn được thiết đặc sẵn sàng. Điều này tạo ra một thiết bị của bộ đệm tập trung, còn gọi là cơ cấu bảng kẹp để cắt và dán (*clipboard- mechanismus ò cut-and-paste*).

Các nhiệm vụ này được giải quyết bởi hộp công cụ các họa tiết với các phương tiện lập trình hướng đối tượng. Hộp công cụ chứa đựng các cơ cấu di truyền lại và các kiểu sắp xếp tổng hợp, mà người lập trình có thể sử dụng một cách thuận lợi. Người ta nhận thấy rằng, với các nhiệm vụ lập trình đồ họa thông thường, người ta đạt được chức năng Xlib cho một ứng dụng một cách tiện lợi, không đòi hỏi nhiều công việc phức tạp.

Mỗi một sự mở rộng về họa tiết đồ họa thì biểu thị một số thuộc tính nào đó; tức là, nó đặc trưng cho một tính năng nào đó; mà với tính năng này, một không gian bộ nhớ được phân bổ. Người ta có thể đọc thấy sự che phủ bộ nhớ này khi khởi xướng đối tượng từ một tệp tin nguồn.

Có hai kiểu mở rộng họa tiết đồ họa. Đó là kiểu đơn giản và kiểu kết hợp. Sau đây, chúng ta sẽ lần lượt nói về hai kiểu này:

✓ *Kiểu mở rộng đồ họa đơn giản*: Đó là kiểu đồ họa nguyên sơ; chúng được dẫn xuất từ cấp hạng XmPrimitives (nguyên sơ) và được biểu thị sát cạnh nhau trên màn hình.

Người ta lưu ý rằng, mỗi lớp được dẫn ra có một ý nghĩa đặc biệt đối với lớp phía trên nhờ các thuộc tính (với các biến) và các phương pháp (với các thủ tục).

✓ *Kiểu mở rộng đồ họa kết hợp*: Kiểu mở rộng này có tác dụng như một thùng đựng đa năng (container), chúng chứa đựng nhiều kiểu đồ họa mở rộng khác nhau, gọi là các đồ họa khởi thuỷ (*primitives*).

Các thùng đựng đa năng xác định cách bố trí hình học của các kiểu đồ họa mở rộng đang chứa đựng. Hình 7.15 là một thí dụ về dáng vẻ bên ngoài của một cửa sổ; cửa sổ này làm nhiệm vụ xuất các thông tin tới người sử dụng. Kiểu mở rộng đồ họa kết hợp với thùng đựng đa năng này còn được gọi là kiểu mở rộng hỗn hợp hay kiểu mở rộng sắp xếp.

Hình 7.15**********

Bên phải hình 7.15 là cấu trúc các đối tượng của kiểu mở rộng kết hợp; cấu trúc này có liên quan tới quan hệ đã chứa đựng. Kiểu đồ họa mở rộng hỗn hợp được biểu thị là kiểu mở rộng cha (*parent widget*). Kiểu sắp xếp được kết nối một cách trực tiếp với việc sắp xếp các cửa sổ. Bình thường, mỗi kiểu mở rộng có một cửa sổ hình chữ nhật khép kín. Nhờ đó, một sự sắp xếp cửa sổ đều xuất hiện bởi một quan hệ chứa đựng sự sắp xếp mở rộng. Cửa sổ cha thuộc kiểu mở rộng cha, nó chứa đựng cửa sổ con; trong cửa sổ con cũng chứa đựng kiểu mở rộng con tiếp theo…

Cửa sổ cao nhất (*top- level-window*) khi sắp xếp các kiểu mở rộng được định nghĩa bởi công dụng được gọi là kiểu mở rộng vỏ (*shell-widget*). Nói một cách chính xác, nó chứa đựng một kiểu mở rộng hỗn hợp, trong đó, tất cả các kiểu mở rộng khác được chứa đựng, và nó dịch vụ cho việc trao đổi thông tin giữa cửa sổ quản lý và sự sắp xếp các kiểu mở rộng hỗn hợp. Nhưng, nó cũng có thể chứa đựng kiểu mở rộng vỏ tiếp theo; đó chính là nút nẫy (pop-up). Nút nẫy sẽ mở ra các cửa sổ (để hướng dẫn, quản lý hay thông báo lỗi…), các cửa sổ này có thể thình lình xuất hiện và rồi lại biến mất…

7.3.4. Phương pháp diễn tả các sự kiện.

Tiến trình server của hệ thống X-Window bao gồm những vấn đề tương tự như đã đề cập ở trong thư mục 7.2.3; tại mỗi vòng lặp với gọi hệ thống XtMainLoop(), chúng ta kích chuột trên một đối tượng hội thoại, thì cái gì sẽ xảy ra ?

Cửa sổ điều hành của hệ thống X-Window sẽ phân bổ tất cả sự trình diễn cho đối tượng hội thoại; khi đó con trỏ chuột đứng ngay trên đối tượng này và nó được kích hoạt (ở vị trí phía trên của danh sách trình diễn). Nếu cửa sổ không quan tâm

tới biến cố này, thì do đó, nó sẽ đạt được sự sắp xếp từ dưới lên trên, tới cửa sổ cha và tiếp tục cho đến khi: hoặc là nó đạt tới cửa sổ gốc, hoặc là nó làm ngơ như không hay biết gì cả.

Bổ sung thêm khả năng này, người ta có thể điều chỉnh nhờ việc gắn chặt các yếu tố muốn quan tâm (như việc thảo một mặt nạ chứa đựng các sự kiện – *eventsmask*-của một cửa sổ). Điều cần thiết đối với một cửa sổ là phải dẫn tới các yếu tố của các cửa sổ kế cạnh (sắp xếp theo phương ngang) nhờ sự che phủ (*grabbing*) như đã nói. Điều này thì có lợi, nếu người ta muốn lôi một đường thẳng vào trong phần che phủ của một cửa sổ; khi đó, chương trình để lôi các đường thẳng cần dùng điểm thứ hai ở trong cửa sổ che phủ của nó để kết thúc việc trình diễn.

Bây giờ chúng ta có thể định nghĩa như thế nào về phương pháp diễn tả biến cố cho một ứng dụng cụ thể? Về điều này, một phản ứng mong muốn phải được lập trình như một thủ tục tác động hay một lập thức gọi trở lại (*callback-routine*) và được móc nối với một thủ tục đặc biệt XtCallBack() trong thời gian chạy (*runtime*). Nếu một biến cố xuất hiện trong XtMainLoop(); do đó, biến cố được kéo dài tiếp tục cho tới khi đạt được một biến cố đại diện của các cửa sổ. Một đại diện biến cố như vậy có thể hoặc là nó tự biểu thị biến cố, hoặc là nó được nhìn lui trong bảng chuyển đổi; khi đó, một thủ tục nào đó được gọi. Bấy giờ, ở vị trí này, đầu tiên, lập thức gọi callback được gọi tới.

Vấn đề biểu thị sơ đồ là ở chỗ, tại một lỗi ở trong lập thức callback, một sự sửa chữa lỗi bằng tay bởi người sử dụng là không thể được. Mỗi sự trình diễn có thể được chỉ được xảy ra ở trong XtMainLoop(), nghĩa là có một cái gì đó hạn chế sự tiếp diễn ở trong lập thức callback. Một khả năng nữa là phải ghi chép lỗi và phải kết thúc lập thức callback. Ở vị trí thích hợp trong chương trình, người ta phải đọc chọn trên các lỗi xuất hiện lần cuối.

7.4. Hệ thống cửa sổ trong Windows NT

Giao diện người sử dụng của Windows NT được phát triển từ giao diện cũ của Windows 3.1. Điều đặc biệt là, giao diện 16 Bit cũ để lập trình về xuất/ nhập đồ họa (*application programming interface*: API) được mở rộng thành 32 Bit, gọi là Win 32 API và được trang bị thêm các chức năng của hệ điều hành. Thí dụ, người lập trình có thể truy cập trên một hệ điều hành đa nhiệm, đa tiến trình, có ưu tiên và an toàn. Hệ điều hành này sử dụng một hệ thống bộ nhớ 32 Bit tuyến tính.

Tuy nhiên, tất cả các tên (tên các tệp tin, tên các chức năng…) được giữ nguyên theo hệ thống cũ, chỉ có các bộ chỉ thị và các thủ tục thì không có quan hệ gì cả với mô hình bộ nhớ được dựa trên dãy tuần tự, mà chỉ có quan hệ với mô hình bộ nhớ ảo tuyến tính. Các chức năng bổ sung cho việc đồng bộ, cho các cơ chế an toàn đối tượng và cho việc quản lý bộ nhớ thì cho truy cập trên phần lớn nhân hệ điều hành *Windows-NT-Executive*.

Hệ thống giao diện mở rộng Win32-API đã được tạo ra như một trong các hệ thống con (xem hình 1.7). Do đó, tính chất look-and-feel của giao diện đã được làm thích hợp một cách mạnh mẽ ở hình dáng bên ngoài của Windows 3.1 quen thuộc, nhằm đơn giản hóa sự quá độ từ Windows NT tới người sử dụng. Ở *version 4 of Windows NT*, phương pháp hợp lý không chỉ làm thay đổi giao diện của hệ thống Win32-API trong các nhân của hệ điều hành, mà còn làm thích hợp các đặc trưng look-and-feel của Windows 95, nó là phần kế tục của Windows 3.1.

Tuy nhiên, Windows NT là một hệ thống đa nhiệm, chứ không phải là hệ thống đa người sử dụng. Điều đó có nghĩa rằng, nhiều người sử dụng không có thể làm việc đồng thời trên một máy tính. Với điều này, phương pháp hợp lý là, sự tồn tại các tiến trình đại diện phải là của các người sử dụng xác định. Khác với hệ thống X-Window, hiện nay trong Windows NT, không có giao diện người sử dụng phân bổ được trợ giúp. Mỗi ứng dụng được dự kiến thực hiện một nhiệm vụ đồ họa, nó phải thực hiện nhiệm vụ này trên một máy tính cục bộ, chứ không thể thực hiện trên một máy tính khác, gọi là displaycomputer. Nhưng, nếu người ta muốn đạt được nhiệm vụ thiết kế đồ họa với các tiến trình khác nhau trên các máy tính khác nhau, do đó, trên máy tính này, một tiến trình phô diễn được người sử dụng lập trình phải được khởi động; tiến trình này đón nhận các dữ liệu (dùng cho ứng dụng) qua mạng máy tính (xem chương 6 về các cấu trúc trao đổi thông tin khác nhau). Khi đó, nhiệm vụ sẽ được chuyển giao trên displaycomputer hệ thống đồ họa.

7.4.1. Các cấu trúc cơ sở

Hệ thống con Win32 được phân thành 5 bộ phận, các bộ phận này thì phù hợp với hệ thống giao diện của Win32-API.

Giao diện Win32-API bao gồm các thành phần riêng lẻ sau đây:

+ Cửa sổ quản lý (*windows manager*);
+ Giao diện thiết bị đồ họa (graphic device interface: GDI);
+ Bộ kích tạo thiết bị đồ họa (graphic device driver: GDD);
+ Các chức năng hệ điều hành (với tệp tin kernel 132.dll);
+ Các chức năng khuyên giải (*console function*) đối với việc xuất/ nhập text.

Các bộ phận này được chứa đựng trong các thư viện, gọi là thư viện kết nối năng động (*dynamic link library*: DLL); khi khởi động, chúng được nạp như một hệ thống.

Giao diện thiết bị đồ họa GDI cho phép vẽ phác thảo và thao tác các đối tượng đồ họa tổng thể đơn giản, như điểm, đường (thẳng và cong), vòng tròn, cửa sổ, các thanh trượt…Các chức năng đồ họa được soạn thảo trên các chức năng của bộ kích tạo đồ họa; các bộ kích tạo này lại gọi tới bộ kích tạo của nhân hệ điều hành và do đó, các phần cứng cũng được điều khiển thích ứng. Vì mỗi lần gọi các gọi các chức năng đồ họa của chương trình người sử dụng sẽ thiết lập một chuỗi các nhân hệ điều hành bao gồm giao diện thiết bị đồ họa GDI, bộ kích tạo thiết bị đồ

họa GDD,…và do đó vô cùng mất thời gian! Vì thế, để gia tăng hiệu suất, một công nghệ có tên arttribute caching (sự trữ dấu thuộc tính) được điền thêm: Người ta thấy có nhiều gọi hệ thống của chương trình người sử dụng gặp nhau với đối tượng tương tự; cho nên, chúng sẽ được thu gom ở chương trình người sử dụng và được dẫn tiếp như là một thông tin riêng lẻ tới giao diện thiết bị đồ họa GDI.

Đối với thư viện GDI cũng được quan tâm bằng một công nghệ tương tự. Đó là công nghệ batching (sự phân đợt) cho các phản ứng nhanh. Tất cả việc gọi hàm chức năng được nạp vào hàng đợi, cho tới khi hàng đợi đầy hay cho tới khi nhập các dữ liệu. Tiếp đó, bộ đệm tổng thể sẽ chuyển giao cho hệ thống con Win32 bởi một thông tin duy nhất. Cơ chế này xảy ra rất nhanh, không có hiệu ứng lùi xuất hiện khi trình diễn đồ họa trên một máy tính chuẩn.

Trình điều hành cửa sổ tạo ra đặc tính look-and-feel, trong đó, nó dịch vụ các cơ cấu cửa sổ ví như các nút ấn để mở rộng hay thu nhỏ cửa sổ, tạo ra các thanh trượt, kể cả việc quản lý danh sách trình diễn (thí dụ cửa sổ nào che phủ tên cửa sổ nào); … ngoài ra, nó cũng thiết đặt việc nhập dữ liệu cho một cửa sổ của một chương trình tương ứng. Vì vậy, nó còn sử dụng các thủ tục xuất/ nhập, để thực hiện chương trình ứng dụng độc lập với các thiết bị phần cứng đang tồn tại (thí dụ kiểu chuột).

Vì MS-Windows được coi là hệ thống chiếm ít bộ nhớ, do đó, cửa sổ quản lý không lưu trữ những phần cửa sổ không nhìn thấy, mà nó chỉ đảm nhiệm vụ để báo hiệu cho ứng dụng, nếu cửa sổ phải được phác họa một cách mới mẻ; vì trước đó các phần cửa sổ không thể nhìn thấy thì nay đã được nhìn thấy. Ngay cả, việc quản lý một bộ nhớ trung gian (ở hệ thống mở rộng) đối với các đối tượng (thí fuj clipboard) cũng được nó thực hiện.

Các chức năng khuyên giải là các chức năng tập trung để xuất/nhập các ký tự và được các hệ thống con sử dụng (POSIX, OS/2, MS-Windows…). Việc xuất-nhập các bài text của các chương trình bao gồm các giao diện người sử dụng hướng ký tự không có sự trợ giúp đồ họa, chúng được thực hiện một cách có mục đích ở trong các cửa sổ đặc biệt, gọi là cửa sổ khuyên giải (consolewindow); cửa sổ này được hệ thống mở một cách tự động.

7.4.2. Vài nét về thiết kế giao diện người sử dụng

Ngược lại với hệ thống mở rộng X-Window, hệ thống cửa sổ của Windows NT chỉ cần người ta nắm được vài loại cửa sổ tổng hợp. Thực ra, ở đây chỉ phân biệt có hai loại cửa sổ: cửa sổ chính và cửa sổ phụ.

Sau đây, chúng ta muốn khảo sát vài kỷ thuật tương tác cơ bản, chúng được trợ giúp bởi hệ thống giao diện Win32- API với các đối tượng và các phương pháp phù hợp. Mỗi cửa sổ chính ở trong Windows NT (cũng giống như cửa sổ họa tiết ở trong hệ thống X-Windows) có những phần chính sau đây:

+ Khung cửa sổ (*frame*): nó có thể được sử dụng khi kích chuột để thu nhỏ hay phóng to;

+ Thanh ghi tiêu đề (*title bar*): trong đó người ta có thể nhìn thấy tên của ứng dụng;

+ Biểu tượng (*icon*): nó là biểu trưng của ứng dụng;

+ Thanh cuộn (*scroll bar*): người ta dùng khi muốn xem các phần nội dùng chưa được thể hiện cửa sổ;

+ Thanh menu (*menu bar*): chỉ ra các trường khác nhau mà người ta cần tới;

+ Các nút ấn điều khiển (*control pop-up*): dùng để thu nhỏ, phóng to hay đóng cửa sổ lại;

+ Thanh trạng thái (*status bar*): trên đó, các thông tin trạng thái được dẫn ra.

Các cửa sổ có thể được xếp chồng lên nhau hay được che phủ; ở đây, cửa sổ hoạt động là cửa sổ không bị cửa sổ khác che phủ, nó được biểu thị là cửa sổ nổi lên phía trước nhất trong các cửa sổ thể hiện trên màn hình.

Thanh cuộn được dùng làm dịch chuyển thích ứng để thể hiện từng khoảng tài liệu lên cửa sổ soạn thảo. Khoảng tài liệu được nhìn thấy trên màn hình này là một phần dữ liệu của toàn bộ tài liệu chứa đựng trong cửa sổ này.

Cửa sổ chính này còn có nhiều tính chất cần được lưu ý. Thật vậy, người ta có thể định nghĩa sử dụng các cửa sổ ở phía dưới với nhiều cách khác nhau. Khác với hệ thống mở X-Window, ở loại này chỉ có vài khả năng.

Vì ở Windows NT cách thể hiện các dữ liệu tập trung được dùng phổ biến, do đó đối với mỗi tệp tin, người ta có thể mở một cửa sổ riêng lẻ; ở đó, trên thanh tiêu đề, không còn thấy biểu tượng của chương trình tiện dụng, mà chỉ xuất hiện các dữ liệu. Bản chất của cửa sổ dữ liệu ở trong cửa sổ chính của chương trình ứng dụng được điều chỉnh nhờ giao diện đa tài liệu. Giao diện này cho thấy rằng, cửa sổ dữ liệu còn gọi là cửa sổ con sẽ được mở rộng phù hợp với mức độ bên ngoài và bên trong của cửa sổ tiện dụng. Ngoài ra, thanh tiêu đề của cửa sổ dữ liệu được bỏ đi; tên thì được làm lan rộng cùng với tên của ứng dụng ở trong dạng vừa có tên tệp tin vừa có tên của ứng dụng; còn biểu tượng và nút nẫy điều khiển của thanh tiêu đề dữ liệu xuất hiện trong thanh menu của ứng dụng. Hình 1.18 cho thấy: ở bên trái thì cửa sổ dữ liệu là một phần thu nhỏ ở trên cửa sổ ứng dụng; còn ở bên phải thì cửa sổ dữ liệu được nới rộng ra trùng khít với cửa sổ ứng dụng. Cửa sổ dữ liệu được kết nối trực tiếp với cửa sổ ứng dụng. Khi cửa sổ ứng dụng đóng lại, do đó, cửa sổ dữ liệu cũng được đóng lại một cách tự động.

Nếu các cửa sổ dữ liệu thuộc một tệp tin riêng lẻ, do đó, khả năng này phải được sử dụng sớm hơn để chia xẻ cửa sổ chính thành các cửa sổ thành phần. Do đó, người ta có thể dự định một nút nẩy đồ họa cho thanh cuộn, công việc này làm xuất hiện một thanh tách. Việc mô tả các dữ liệu ở trong cửa sổ là nhiệm vụ của một ứng dụng cụ thể nào đó.

Khác với cửa sổ chính, các cửa sổ phụ có kích cở cố định và được sử dụng để thông báo hay để mô tả các thông số bên trong khi đó nó được gọi là cửa sổ đặc tính tờ biểu. Các cửa sổ phụ được lưu ý tới vị trí tương đối của chúng ở trong cửa

sổ chính và được kết nối chặt chẽ tại cửa sổ chính. Chẳng hạn, nếu một cửa sổ chính sẽ là cửa sổ hoạt động và có thể nhìn thấy được một cách đầy đủ, do đó, cửa sổ phụ sẽ được lộ ra một cách tự động để được nhìn thấy. Thí dụ, đối với cửa sổ phụ có thể vừa gọi xong thì trở thành một hộp thoại để tìm và thay thế, hộp thoại in ấn, hộp thoại chọn font, hộp thoại chọn màu, các cửa sổ trộn màu và cửa sổ thông báo.

Qua cửa sổ chính và cửa sổ phụ, hệ thống giao diện Win32-API còn tạo ra một số lượng lớn các đối tượng hội thoại khác nhau như nút nẩy điều khiển, các nút tuỳ chọn, các nút kiểm tra, hộp các danh sách chọn tĩnh động, các danh sách này làm việc như một menu đầy hoặc vơi hay các tệp tin được bày ra như một trình quản lý các tệp tin với các biểu tượng và các bài text. Ngoài ra còn có các ô với các bài text đã có sẵn hay có thể soạn thảo thêm vào và các thanh dụng cụ.

7.5. Các bài tập của chương 7

Bài tập 7.1. Về giao diện người sử dụng và lập trình trực quan

Nếu cơ chế giải pháp tập trung của một vấn đề kiểu tương tự ở trong thực tế thì đã rõ, người ta có thể mô hình hoá nó bằng hình vẽ ở trong giao diện người sử dụng.

a). Bên cạnh các quan hệ cố định đơn giản giữa mô hình và giải thuật được tạo lập. Sự mô phỏng này được thiết lập một cách rất đơn giản, nghĩa là trong đó, tránh được mọi phiền phức, thí dụ nhờ việc sử dụng các biểu tượng.

b). Nếu người sử dụng phải nhận thức sự logic bên trong của chương trình, do đó, điều này được thiết kế cho một việc học tập sáng tạo; ở đó, đầu tiên tính tổng hợp được cô đúc và rất thuận tiện trong quá trình sử dụng.

Bạn hãy thực hiện các nguyên tắc này cho một ví dụ đơn giản. Đối với việc quản lý các dữ liệu, người ta có thể sử dụng các công việc trong một nhà kho lưu trữ như các hình ảnh của người lập trình. Ở đây, việc cung cấp, việc sắp xếp các bao gói, việc chất kho và phân loại thì đã rõ, chúng tồn tại như một sự tham chiếu của người sử dụng.

Cho cái đó bạn hãy thiết kế một sơ đồ, mà ở đó, các đối tượng tệp tin (các thuộc tính và các phương pháp) được thu xếp cho các đối tượng trực quan.

Bài tập 7.2. Bảng phân ảnh màu

Bạn hãy trình bày những vấn đề sau đây một cách chặt chẽ dưới quan điểm của cá nhân bạn:

a).Giả sử, chúng ta mô tả một cách trực tiếp màu sắc của một điểm ảnh cho một đồ họa đúng theo yêu cầu mà không cần CLUT với độ chính xác 24 Bit, gọi là độ sâu của màu. Bộ nhớ nhắc lại ảnh cho một ảnh có kích cở 1024x768 (điểm ảnh) tối thiểu phải bằng bao nhiêu? Và, bằng bao nhiêu, nếu nó phải lưu trữ ảnh này trên không gian 3 chiều ?

b). Bộ nhớ nhắc lại ảnh tối thiểu phải bằng bao nhiêu, nếu 16 Bit màu có thể đồng thời nhìn thấy và một sự chuyển chỗ từng bước với một CLUT là có thể?

c). Bạn hãy khái quát và thiết lập một công thức cho nhu cầu bộ nhớ s không có CLUT ở độ sâu của màu khoảng f Bit/một điểm ảnh và khi số lượng điểm ảnh là N cũng như nhu cầu bộ nhớ là s_{CLUT}/s và bạn hãy chỉ ra rằng, giả sử quan hệ đó thì nhỏ hơn 1 với điều kiện N>n.

Bài tập 7.3.Giao diện người sử dụng ở trong hệ điều hành phân bổ

Trong mục 7.2.3, bản phác thảo về một server trình diễn được dẫn. Quyết định cho cái đó có những chương trình, mà chúng có thể nạp hay biểu thị những bài text và những bức ảnh ở trên Internet; đó là các trình đọc lướt siêu text như Netscape và Internet Explorer. Các chương trình này bổ sung trên máy tính Dataserver của Internet đảm nhiệm công việc đó; các việc dò tìm cũng như các nhiệm vụ tương tự có thể được thực hiện một cách cấp tốc bởi trình Browser. Vì vậy, người ta có thể hoàn thành được nhiều công việc với một hệ thống Application client và Display server.

a). Bạn hãy so sánh các quan hệ client/ server trong cả hai hệ thống.

b). Những kiểu trao đổi thông tin nào thống lĩnh giữa client và server?

Bài tập 7.4. Cấu trúc client/ server

Bản phác thảo cục bộ về hệ điều hành Windows NT có những ưu điểm và nhược điểm nào trong sự so sánh với hệ thống phân bổ X-Window? Bạn hãy suy nghĩ và tiến hành với các ứng dụng như:

+ Điều khiển các tiến trình của một nhà máy thép bằng một mạng máy tính.

+ Đánh giá các số liệu của các phòng ban khác nhau qua mạng cục bộ các máy tính của nhà máy.

+ Thiết lập một phần mềm quản lý cho một nhóm máy tính kết nối mạng.