VISUALISASI DAN NAVIGASI MEDIA PENYIMPANAN DISK DENGAN MENGGUNAKAN METODE SPLIT SCREEN SYSTEM

Yulia
Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Kristen Petra
e-mail: yulia@peter.petra.ac.id

Edy Kendengis
Alumnus Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Surabaya
e-mail: paleleh@yahoo.com

ABSTRAK: Disk merupakan alat yang sangat bermanfaat bagi sebuah sistem komputer untuk menyimpan data berupa file. Untuk itu sangat dibutuhkan program yang dapat merepresentasikan, menavigasi dan mengoperasikan file-file tersebut. Banyak program komputer yang digunakan untuk merepresentasikan, menavigasi dan mengoperasikan file-file pada media penyimpanan disk dalam bentuk teks dengan berbagai macam fasilitas, tetapi sangat sedikit program yang merepresentasikan file-file dalam bentuk grafik atau secara visual serta mampu memberikan kelebihan-kelebihan tersendiri, seperti kemampuan untuk merepresentasikan file-file yang paling banyak menggunakan ruang media penyimpanan disk. Kemampuan seperti ini sangat bermanfaat pada saat melakukan manajemen file-file pada sebuah media penyimpanan disk.

Tujuan dari perancangan ini adalah membuat suatu program komputer berbasis *windows* yang dapat memvisualisasikan media penyimpanan disk yang dilengkapi dengan kemampuan navigasi dan fasilitas manajemen file baik pada personal komputer maupun komputer yang berada pada jaringan lokal.

Pada perancangan ini menggunakan metode *split screen system* dan algoritma *tree map*. *Split screen system* digunakan untuk dapat melakukan navigasi, dimana pada layar monitor dibagi menjadi dua bagian untuk memvisualisasikan file-file pada media penyimpanan disk, bagian yang kecil (*map navigator*) memvisualisasikan secara keseluruhan isi dari suatu *drive* atau *directory* dan bagian yang besar (*map view*) memvisualisasikan detil dari bagian tertentu berdasarkan kotak fokus yang berada pada *map navigator*. Kotak fokus yang berada pada *map navigator* dapat digerakkan dengan *mouse* dan besarnya kotak tersebut merupakan perbandingan dari ukuran *map navigator* dengan ukuran *map view*. Algoritma *tree map* digunakan untuk melakukan kalkulasi ukuran gambar kotak dari suatu file yang akan direpresentasikan. Algoritma ini membutuhkan suatu struktur pohon yang lengkap dari suatu *drive* atau *directory* yang akan digambar.

Pembuatan program komputer berbasis *windows* ini, diimplementasikan dengan bahasa pemrograman *Borland Delphi 5.0*.

Kata kunci: visual, navigasi, *tree map, split screen system.*

***ABSTRACT**: Disk is a very useful device to a computer to save data file. For that, is much needed to have a program that has a capability to represent, navigate, and operate that files. There are many computer programs that used to present, navigate, and operate disk files in text form with a lot of facilities, but only a few programs that can represent files in a graphic form and have special feature, like capability to find the biggest files that use disk space. This capability is very useful in disk file management.*

This software use split screen system method and tree map algorithm. Split screen system is used to navigate, where the monitor screen divided into two parts to visualize files in a disk, the small part (map navigator) visualize the whole content of a drive or directory and the big part visualize the detail of a part according to focus box in the map navigator. A mouse can move focus box in the map navigator and the box size is the comparison between map navigator sizes with map view size. Tree map algorithm is used to calculate the size of box picture of the represented file. This algorithm needs a complete tree structure from a drive or directory that will be painted.

The software to make this windows based program is Borland Delphi 5.0.

***Keywords**: visual, navigation, tree map, split screen system.*

1. PENDAHULUAN

Pada perancangan ini dibuat sebuah program visualisasi dan navigasi media penyimpanan disk dalam bentuk peta dari suatu *disk drive*, dimana di dalam peta ini juga bisa dilihat visualisasi dari suatu *directory* atau *sub directory* dan bagian tertentu dari peta ini dapat diperbesar (dengan metode *split screen*) untuk memperjelas bagian tersebut.

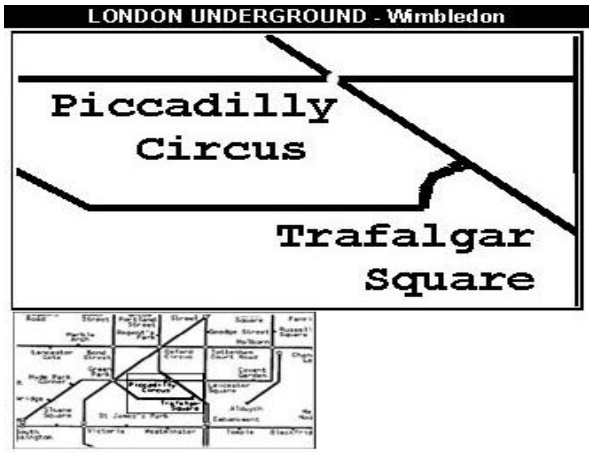
Program aplikasi ini akan dilengkapi dengan berbagai fasilitas untuk mengatur, merawat dan membuka file, seperti melakukan *copy–paste file(s), cut-paste file(s), search file(s), sorting file(s), delete file(s), drag and drop file(s), rename file, undo, running/open file* secara langsung, cetak secara langsung, dimana semua fasilitas tersebut dapat diaplikasikan pada komputer yang sedang terhubung pada jaringan komputer seperti fasilitas *Neighborhood* pada *Windows Explorer*.

Untuk mendukung semua fasilitas di atas, program aplikasi ini juga akan dapat menampilkan nama-nama *directory* dalam bentuk pohon (*treeview*) dan nama-nama file dalam bentuk daftar file (*listview*), untuk itu tampilannya dapat diubah dalam bentuk pohon dan bentuk daftar file (seperti *windows explorer*), yang terintegrasi dengan petanya.

2. SPLIT SCREEN SYSTEM

Split Screen System merupakan sistem tampilan layar, dimana layar monitor dibagi menjadi dua bagian. Bagian pertama digunakan untuk menampilkan detil dari bagian tertentu dari hasil pemetaan dan bagian kedua yang lain digunakan untuk menampilkan keseluruhan hasil pemetaan yang ukuran skalanya sudah disesuaikan dengan ukuran layar monitor. Pada bagian yang pertama terdapat suatu gambar kotak yang dapat digerakkan, dimana bagian yang dilalui oleh kotak tersebut akan diperbesar gambarnya pada bagian yang kedua untuk mempermudah operator memahami konteks dari hasil pemetaan tersebut.

Berikut ini gambar dari implementasi teknik *split screen system*.



Gambar 1. Contoh layar pada metode *Split Screen System*

3. TREE MAP

Tree Map adalah suatu bentuk representasi yang dirancang sebagai suatu visualisasi dari manusia tentang struktur pohon yang kompleks. Suatu struktur pohon yang berubah-ubah dapat digambarkan dengan representasi *2-D space filling*. Ide dasar utama dari *tree map* ini adalah untuk mendapatkan representasi yang lebih baik dari penggunaan tempat penyimpanan pada *harddisk* yang mempunyai *multilevel directory* dari *subdirectory* dan file-file, seperti dalam Unix, Macintosh Finder atau MS-DOS. Pada aplikasi ini, file-file adalah titik (*node*) daun dan *subdirectory* adalah titik yang ada di dalamnya. Kebanyakan sistem operasi menampilkan isi dari suatu *node* pada suatu waktu dengan menggunakan nama dari file-file dan *subdirectory* atau dengan menggunakan *icon* untuk menampilkannya. Pemakai dapat mengubah *tree* dengan menekan *mouse* pada *directory folder icon* atau dengan menggunakan suatu perintah (*command*, seperti CD untuk *change directory*). Sebagian kecil sistem yang lain berusaha untuk menampilkan lebih dari satu *node* pada suatu waktu.

Aplikasi untuk representasi *directory* dari komputer bertujuan untuk menampilkan keseluruhan file-file pada suatu *space filling visualization* yang mengijinkan pemakai untuk mengatur file-file yang besar dengan cepat dan menghapusnya ketika *hard disk* telah penuh. File yang besar ini mungkin ada pada beberapa level pada struktur *tree*. Pendekatan untuk masalah ini adalah dengan

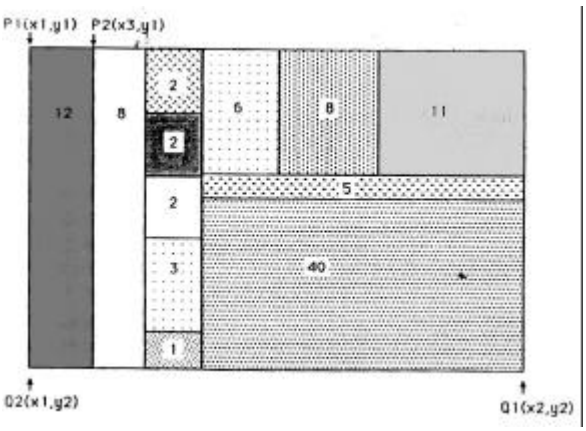
menggunakan representasi 1-D (satu dimensi) dengan panjang dari tiap file dikodekan dengan panjang suatu garis yang berwarna-warni. Hal ini tidak dapat dipraktekkan karena garis tersebut akan terlalu panjang untuk digambarkan. Suatu pendekatan 3-D atau dimensi yang lebih tinggi dan mungkin sulit untuk diaplikasi-kan. Pendekatan 2-D *space filling* merupakan pendekatan yang paling potensial untuk diaplikasikan. Jika tiap file digambarkan sebagai kotak kecil, metode ini dapat dilakukan.

Pendekatan visualisasi struktur pohon yang dikenal dengan *tree map* mampu memecahkan masalah di atas dan juga menyediakan kemungkinan untuk aplikasi yang lain.

3.1 Algoritma Tree Map

Pendekatan sederhana untuk menggam-barkan struktur pohon (*tree*) adalah suatu akar (*root*), grafik dengan titik *root* pada bagian atas dan titik anak-anak di bawah titik orang tua (*parent node*) dengan sebuah garis yang menghubungkannya. Algoritma ini membutuhkan suatu *tree root* dan area kotak yang didefinisikan dengan koordinat kiri atas dan kanan bawah $P_1(x_1,y_1)$, $Q_1(x_2,y_2)$ (dapat dilihat pada gambar 2). Jumlah sisi yang keluar dari akar menunjukkan jumlah partisi dari daerah [x_1, y_2]. Karena *subtree* paling kiri terdiri dari suatu *fraction* (Size(child[1])/Size(root)) dari jumlah total *bytes* pada akar (*root)*, maka garis partisi vertikal pertama digambar pada :

$x_3 = x_1+(\text{Size(child[1])/Size(root)}) * (x_2–x_1)$.



Gambar 2. Implementasi Tree map

Algoritma ini kemudian diulang kembali untuk *tree* yang paling kiri saat ini dengan menggunakan perputaran 90^0 dari kotak $P2(x_3,y_1)$, $Q1(x_2,y_2)$ dan membagi pada arah y-axis. Proses ini terus dilakukan selama - *subtrees* yang lain membuat partisi pada kotak $P2(x_3,y_1)$, $Q1(x_2,y_2)$. Partisi pada level genap adalah secara vertikal sedang partisi pada level ganjil pada *tree* adalah secara horisontal (dapat dilihat pada gambar 2).

Supaya dapat dilihat dengan jelas, harus digunakan warna yang berbeda untuk tiap daerah. Efek dari melihat ribuan kotak kecil adalah seperti papan catur dengan ukuran kotak yang berbeda-beda. Pengkodean yang berwarna dapat menggambarkan tipe file yang berbeda (seperti text, program, biner, grafik, *spreadsheets*), pemilik dari program (tiap pemilik mempunyai warna yang berbeda), frekuensi penggunaan (warna yang lebih terang untuk yang lebih sering digunakan), atau umur dari file (file yang lebih tua mungkin lebih kuning atau lebih abu-abu). Jika area yang kemudian mem-punyai warna yang sama, maka sangat diperlukan garis pembatas. Karena kebutuh-an pemakai sangat banyak, maka tidak ada solusi spesifik yang dapat memenuhi semua situasi itu, dan pemakai harus mempunyai suatu *control panel* untuk beberapa para-meter dan juga untuk mengindikasi warna yang mana yang digunakan untuk menandai nilai-nilai atribut (*attribute values*).

Karena file-file yang membutuhkan ruang pada *harddisk* adalah titik daun (*leaf nodes*), tiap titik interior harus mempunyai ukuran total dari tiap *subtree* (jika perlu, maka keluarkan jumlah tempat penyimpanan yang dipakai oleh tiap file dan *subdirectory* mulai dari level itu naik sampai ke *root*). Jika data ini tidak dipelihara oleh sistem, maka harus ada jalur awal yang melewati *tree* untuk mengumpulkan data ini dan menempatkan-nya pada tiap titik interior. Jika *user* ingin melihat persentase total penggunaan ruang disk, maka titik *root* harus anak (*child*) tambahan yang merupakan *record* buatan (*dummy record)* yang ukurannya adalah keseluruhan bagian disk yang tidak digunakan.

Algoritma *tree map* mengasumsikan ada sebuah struktur *tree* yang mana tiap titik

(*node*) itu terdiri dari *record* dengan direktorinya atau nama file (*nama*), jumlah anak (*jum_anak*), dan suatu *array of pointers* ke level berikutnya (*anak[1..jum_anak]*). Argumen atau parameter dari algoritma *tree map* ini adalah:

Akar : suatu *pointer* ke akar (*root*) dari *tree* atau *subtree*

P,Q : array dengan panjang 2 dengan pasangan koordinat (x,y) dari sudut yang berseberangan dari suatu kotak tertentu (asumsikan bahwa Q terdiri dari koordinat atas dan P koordinat bawah, tetapi hal ini tidak berpengaruh pada kebenaran algoritma, hanya urutan kotak yang akan digambar)

Axis : bervariasi antara 0 dan 1 untuk menandakan potongan yang akan dilakukan (vertikal atau horisontal)

Warna : menunjukkan warna yang akan digunakan untuk suatu kotak tertentu

Sedangkan prosedur yang dibutuhkan untuk algoritma *tree map* ini adalah:

Gambar_Kotak: Suatu prosedur untuk menggambar kotak dengan menggunakan warna yang diberikan dan me-*reset* variabel warna

Ukuran : Suatu fungsi yang mengembalikan nilai berupa jumlah byte pada titik yang ditunjuk oleh parameter. Sebagai alternatif, ukuran dapat dihasilkan terlebih dahulu dan disimpan di tiap titik.

Pemanggilan awal adalah :

```
Treemap(akar, P, Q, 0, warna)
```

Dimana P dan Q adalah sudut kanan paling atas dan sudut kiri paling bawah dari tampilan. Dengan memberi nilai awal nol pada parameter *axis*, maka partisi mula-mula adalah secara vertikal. Dalam hal ini diasumsikan bahwa parameter P dan Q adalah *passed by value* :

```
Treemap(akar,P[0..1], Q[0..1], axis, warna)
```

```
1. Gambar_Kotak(P, Q, warna)// menggambar area kotak
2. Lebar:= Q[axis] – P[axis]// mengeluarkan nilai lokasi dari potongan kotak berikutnya
3. For I := 1 to jum_anak do
4. Q[axis]:= P[axis] + Size(anak[I])/size(akar))* lebar
5. Treemap(anak[I],P,Q,1-axis, warna)
6. P[axis] := Q[axis]
7. endfor
```

4. PERANCANGAN PROGRAM

Pada bagian ini akan membahas mengenai desain dari pembuatan program visualisasi dan navigasi media penyimpanan.

4.1 Desain struktur data *node* pada struktur pohon untuk *tree map*.

Struktur data ini menyimpan informasi suatu file atau *directory* yang digunakan pada struktur pohon, dimana struktur pohon tersebut akan digunakan pada saat melakukan proses *tree map*. Pada setiap *node* pada struktur pohon mewakili suatu file atau *directory* pada media penyimpanan *disk.*

Tabel 1. Tabel Struktur Data *Node*

Nama	Tipe Data	Keterangan
Name	String	Menyimpan nama file/*directory*
Path	String	Menyimpan *path*
FileAttr	Integer	Menyimpan atribut file (*read-only, hidden, archive* dan *system*)
Size	LongWord	
Level	Byte	Menyimpan level dari *node*
P, Q	UserPoint	Menyimpan posisi kiri-atas dan kanan-bawah dari kotak yang mewakili file yang bersangkutan
NumFile, NumFolder	Integer	Untuk menyimpan banyaknya file anak dan banyaknya *directory*
FSelected	Boolean	Bernilai *true* jika kotak dari file pada hasil pemetaan terpilih.
FColor	TColor	Menyimpan kode warna yang diacak
Next, Child, Parent	Pointer	*Pointer* ini menunjuk ke-*node* ayah, anak dan anak yang berikutnya,

Ketiga *field* yang bertipe *pointer*, merupakan *pointer* yang akan membentuk suatu *linklist* sehingga membentuk suatu struktur pohon.

4.2 Desain struktur data objek file

Struktur data ini digunakan pada *listview* dan *treeview*, setiap *item* pada *listview* atau *treeview* merupakan suatu objek file yang distrukturkan sebagai berikut.

Tabel 2. Tabel Struktur Data Objek File

Nama	Tipe Data	Keterangan
ID	PItemIDList	Menyimpan ID file, PItemIDList adalah tipe data pointer file yang diimplementasikan oleh Microsoft.
Empty	Boolean	
DisplayName, Path, Type Name	String	
Size	LongWord	
ImageIndex, Attributes	Integer	ImageIndex menyimpan nomor indeks *icon* dari suatu file pada *windows imagelist sistem*.
ModDate	TdateTime	Menyimpan tanggal file, formatnya sesuai dengan sistem operasi *windows*.

Field empty akan bernilai *true* jika semua *field* pada struktur data ini telah terisi, hal ini dibutuhkan karena pada awal dibentuknya struktur data ini oleh suatu objek file, tidak semua field diisi, field yang belum terisi, akan diisi setiap kali terjadi *scroll* pada *listview* atau *expanding item* pada treeview.

4.3 User Define Type

```
Tcoloring     = (exten, ran, method)
TSortType     = (byNameA, byNameD,
                 byTypeA, byTypeD,
                 bySizeA, bySizeD,
                 byDateA, byDateD)
TFileOp       = (foCopy, foDelete, foMove,
                 foRename)
TFileOpFlag   = (fofAllowUndo,
                 fofConfirmMouse,
                 fofFilesOnly,
                 fofMultiDestFiles,
                 fofNoConfirmation,
                 fofNoConfirmMkDir,
                 ofRenameOnCollision,
                 fofSilent,
                 fofSimpleProgress,
                 fofWANTMAPPINGHAN
                 DLE)
TfileOpFlags  = set of TFileOpFlag;
```

4.4 Desain Tampilan Layar

Layar utama terdiri dari dua bagian utama dimana bagian pertama ditampilkan dalam bentuk *treeview* dan bagian kedua ditampilkan dalam bentuk *listview* dan dapat pula dalam bentuk peta (*mapview*). Pada bagian pertama layar, tampilan berbentuk *tree. Tree* tersebut dapat menampilkan isi dari tiap *drive* yang ada pada komputer dimana program visualisasi dan navigasi ini dijalankan. Suatu *tree* dapat terdiri dari nama-nama *drive*, *directory*, serta *subdirectory* yang ada.

Bagian kedua layar dapat berbentuk *listview* dan dapat berbentuk *mapview* atau peta. *Listview* dibagi menjadi empat macam bentuk tampilan yaitu *Small Icons Style, Large Icons Style, List Style,* dan *Details Style*.

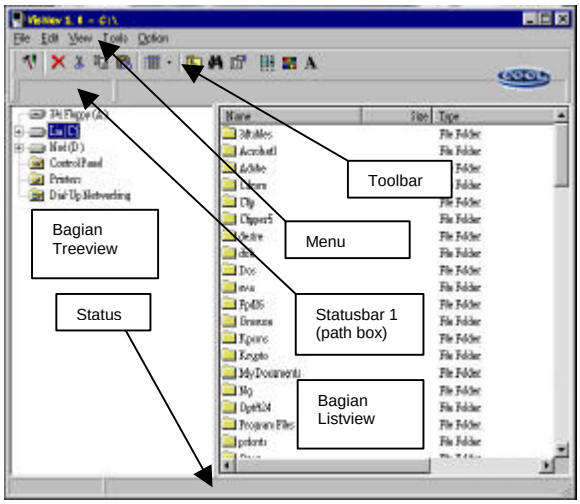
Seperti yang disebutkan di atas bahwa bagian kedua dari layar dapat berbentuk *listview* dan dapat pula berbentuk peta atau *mapview.* Peta ini dibagi menjadi dua bagian, yaitu bagian untuk menampilkan keseluruhan isi media penyimpanan dan bagian untuk menampilkan hasil pembesaran bagian tertentu yang pada pada bagian pertama. Teknik pembagian layar tersebut menggunakan cara *split screen system.*

5. EVALUASI HASIL

Berikut ini dua gambar dari tampilan layar utama yang telah didesain, gambar pertama layar dibagi atas *treeview* dan *listview* sedang bagian kedua layar dibagi atas *treeview* dan *mapview*.

Tiap file dan *folder* pada *listview* mempunyai menu *popup.* Jenis menu *popup* untuk tiap file berbeda tergantung dari tipe file (*extension* dari file tersebut), misalnya saja menu *popup* untuk tipe file *document* (.DOC) akan berbeda dengan menu *popup* dari tipe file .EXE. Untuk menampilkan menu *popup* dilakukan memilih suatu file

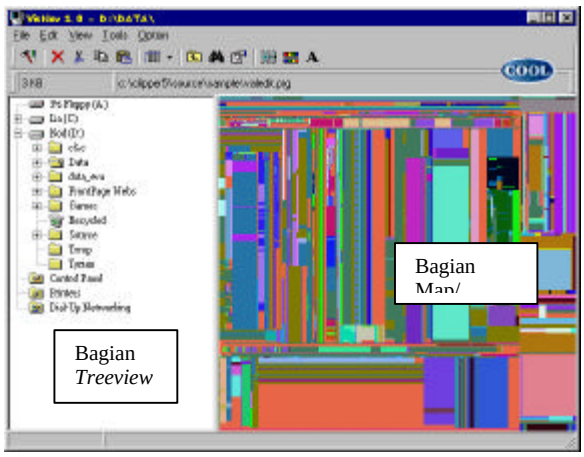
atau *folder* tertentu, kemudian klik bagian *mouse* sebelah kanan.



Gambar 3. Tampilan Utama 1 (Treeview dan Listview)

Fasiltas-fasilitas yang dapat dilakukan pada listview tersebut antara lain fasilitas untuk Open, Print, Find, Cut, Copy, Paste, Delete, Rename, Create Shortcut, Sorting, Undo, View with ACDSee, Print with ACDSee (jika ACDSee sudah di-install), Go To, Up One Level, Find Top Size, Change Font, menampilkan *property* suatu *folder*, dan Hide Listview.

Pada tampilan utama terlihat ada dua buah *statusbar* yang masing-masing berfungsi sebagai berikut. *Statusbar 1* digunakan untuk menampilkan konteks suatu file yang dilalui oleh *cursor mouse* pada *mapview* dan *Statusbar 2* untuk menampilkan status-status khusus dari suatu *folder*, misalnya menampilkan banyaknya file, *subfolder* dan *size* dari suatu *folder*.

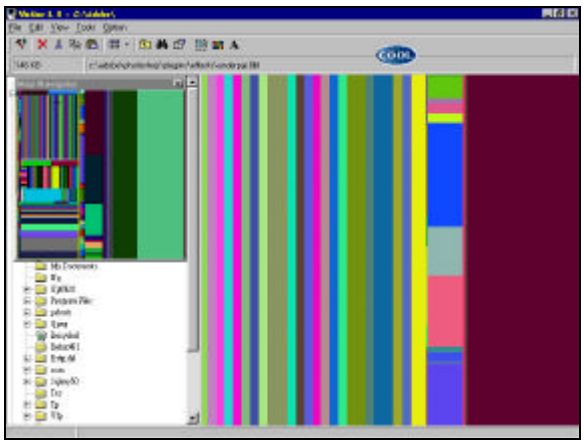


Gambar 4. Tampilan Utama 2 (Treeview dan Map)

Selain tampilan utama di atas, terdapat tampilan *control* yang digunakan untuk memilih kontrol-kontrol yang diinginkan, kontrol-kontrol ini meliputi kontrol warna dari suatu tipe file, kontrol *browser* (tampilan utama, *treeview* dan *listview*) dan kontrol *map*.

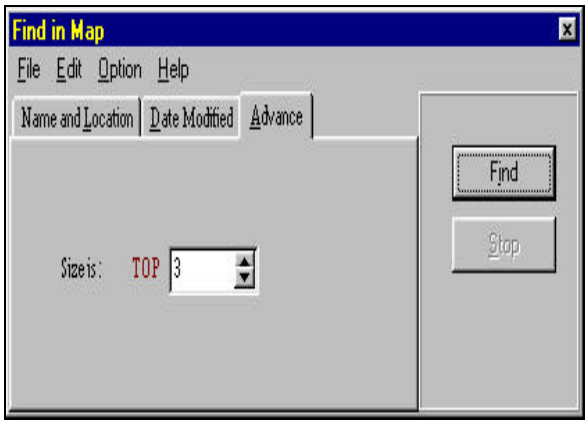
Berikut ini adalah tampilan peta untuk menampilkan isi dari suatu *drive* atau *folder* yang dipilih atau aktif pada *treeview* dalam bentuk gambar peta.

Pada tampilan ini dibagi menjadi dua bagian yaitu bagian pertama adalah Map Navigator dan bagian kedua adalah hasil pembesaran salah satu bagian yang terfokus pada Map Navigator. Berikut ini gambar dari kedua bagian dari peta.



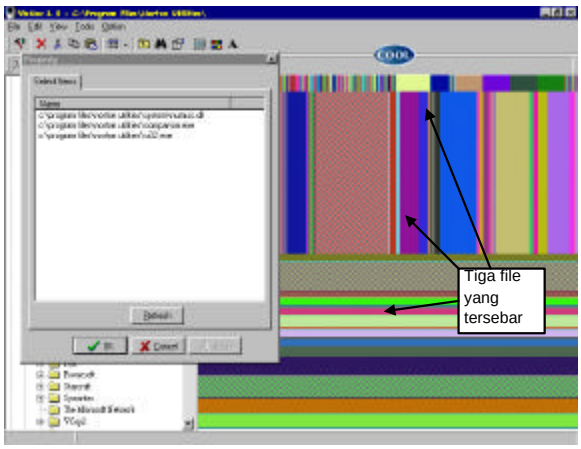
Gambar 5. Peta Visualisasi

Pada perancangan ini juga disediakan fasilitas untuk melihat file-file apa saja yang menggunakan tempat terbesar pada suatu *folder* atau *subfolder*. Pada *window* di bawah ini user dapat mengisikan jumlah file dengan ukuran terbesar yang ingin diketahui



Gambar 6. Window Find Top Size

Setelah diproses akan muncul hasil seperti :



Gambar 7. Hasil Find Top Size

6. KESIMPULAN DAN SARAN

6.1 Kesimpulan

Pembuatan program visualisasi dan navigasi telah menghasilkan suatu aplikasi berbasis *windows* yang mempermudah pemakai komputer untuk mengetahui informasi dari *file-file* atau *folder* pada media penyimpanan *disk*, baik lokal maupun pada komputer yang berada pada jaringan. Program aplikasi ini memberikan beberapa fasilitas yang merupakan implementasi dari metode-metode atau algoritma yang dipakai seperti metode *split screen system* dan *algoritma tree map.* Fasilitas-fasilitas yang dimaksud adalah kemampuan program ini dalam memvisualisasikan media penyimpanan *disk* sehingga pemakai dapat mengetahui file-file yang menggunakan tempat yang besar pertama, kedua, ketiga dan seterusnya.

Perancangan ini telah memperdalam pengetahuan tentang penggunaan objek-objek yang handal dari Delphi yang memungkinkan program ini memiliki kemampuan menyerupai Windows Explorer dan memiliki kemampuan berhubungan dengan komputer lain, pada jaringan lokal.

6.2 Saran

Pengembangan suatu program komputer merupakan hal yang sering dilakukan, untuk itu dibutuhkan saran-saran yang dapat dijadikan masukan pada saat melakukan pengembangan pada program ini.

Implementasi dari algoritma *tree map* telah menghasilkan visualisasi yang baik tetapi algoritma ini tidak dapat memvisualisasikan file-file yang berukuran sangat kecil jika dibandingkan dengan ukuran *directory*-nya. Untuk itu algoritma *tree map* pada program ini digunakan dengan tujuan untuk mengetahui file-file yang menggunakan tempat yang terbesar pertama, kedua, ketiga dan seterusnya dan bukan digunakan untuk mengetahui file-file yang berukuran kecil. Hal ini tidak memberikan kerugian yang berarti pada program ini karena file-file yang tidak tervisualisasi tidak memberikan pengaruh yang besar pada penggunaan media penyimpanan *disk*.

Kelemahan algoritma *tree map* di atas telah menyebabkan kesalahan kecil pada saat visualisasi. Sebagai contoh misalnya pada suatu *directory* ada 500 file dan ada 50 file yang tidak tervisualisasikan, maka *tree map* akan menggambar sebuah kotak yang tidak *valid* yang merupakan kotak dari 50 file tersebut. Untuk itu perlu sedikit modifikasi pada algoritma *tree map*.

Ada banyak cara yang dapat dilakukan untuk pewarnaan pada visualisasi media penyimpanan *disk*, tetapi tidak semua cara dapat diterapkan karena masing-masing cara mempunyai tujuan tertentu. Karena itu proses pewarnaan harus memperhatikan tujuan dari visualisasi sehingga dapat memilih cara yang sesuai. Sebagai contoh pada visualisasi program ini bertujuan menampilkan file terbesar dan untuk navigasi, maka teknik pewarnaan yang dibutuhkan adalah pewarnaan yang dapat membedakan antara kotak yang satu dengan kotak yang lain.

DAFTAR PUSTAKA

1. Furnas, G.W., *Generalized Fisheye Views,* Proceedings CHI'86 Human Factors in Computer Systems, Boston, 1986, hlm. 16-23.

2. Shneiderman, B., *Designing The User Interface: Strategies for Effective Human-Computer Interaction,* Addison Wesley, 1987.

3. Spence, R. dan Apperley, M., *Data Base Navigation: An Office Environment for Professional,* Behavioural Information Technology, Vol. 1, 1982, hlm. 43-54.