4. IMPLEMENTASI

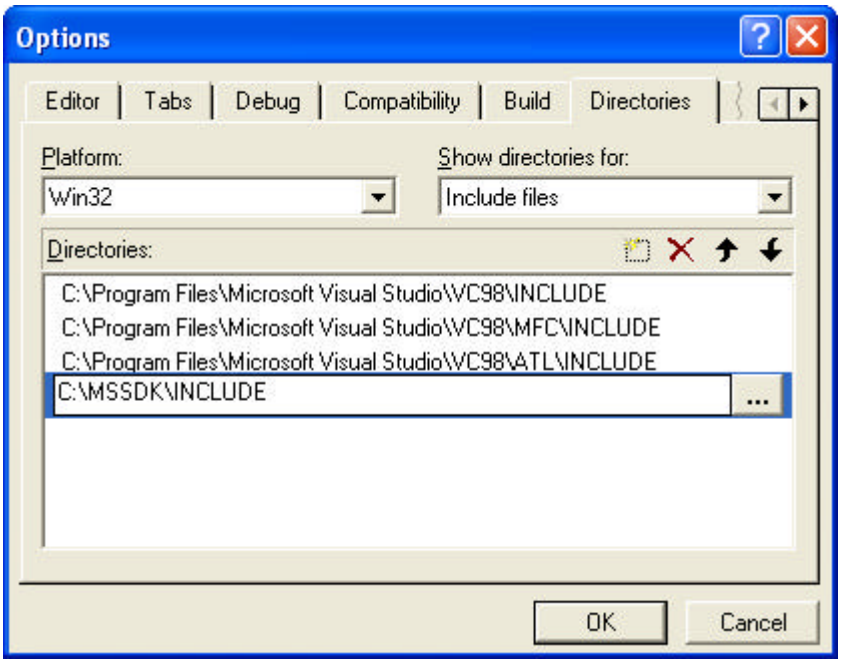
4.1. Langkah awal implementasi

Sebelum proses pemrograman dapat dilakukan, untuk dapat menggunakan DirectX terdapat beberapa hal yang harus dilakukan supaya function dari DirectX dapat dijalankan pada aplikasi yang ditulis. Hal yang harus dilakukan agar dapat melakukan pemrograman dengan DirectX ialah

a. Melakukan proses instalasi Microsoft Visual C++ ver 6.0 (MSVC 6.0).

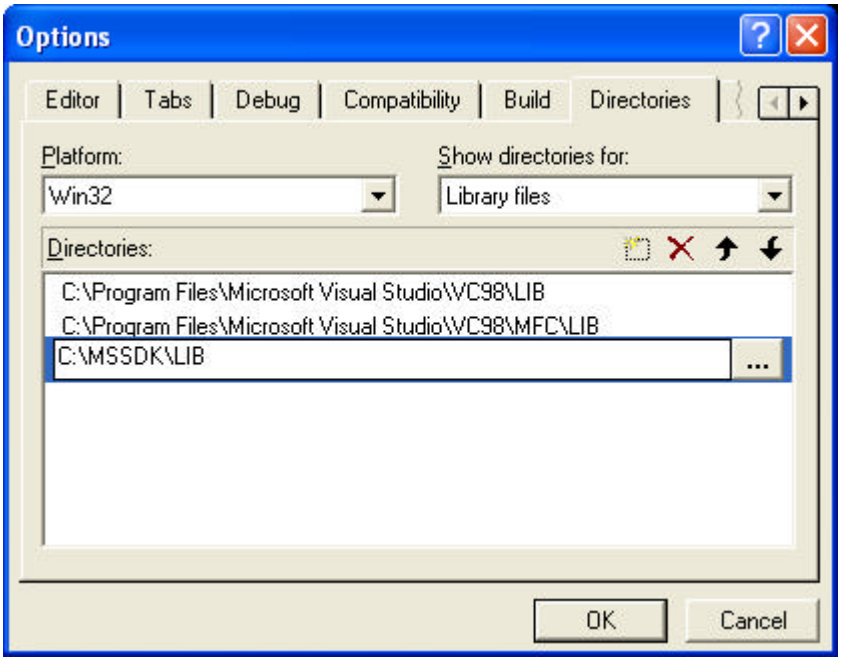
b. Melakukan proses instalasi DirectX 8.1 Software Developer's Kit.

c. Menambah *include files path* pada MSVC Untuk penambahan *path* dapat dilakukan dengan cara memilih *Tools*, *Option*, kemudian pada *Option* memilih *Directory tab*, setelah itu pada *Show directories for* diubah agar menunjukan *Include files*. Kemudian dilakukan penambahan *path* yang diperlukan, *path* yang baru dibawa keatas daftar dengan menggunakan tombol panah keatas.



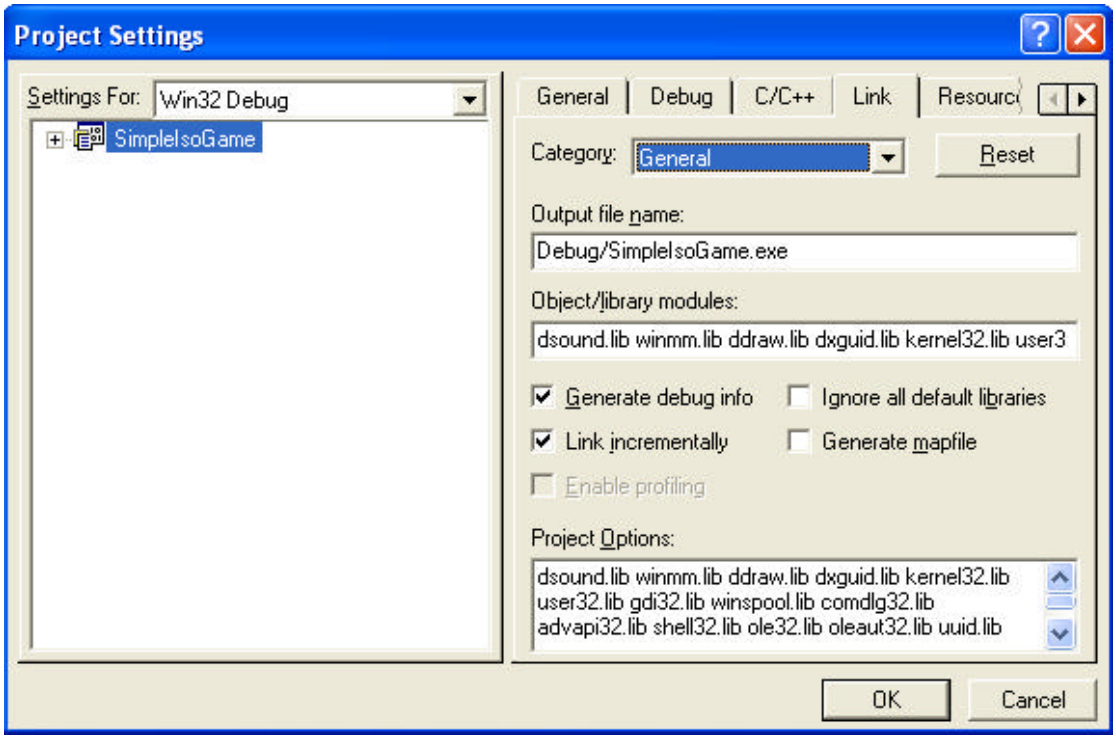
Gambar 4.1 Penambahan *include files path* pada MSVC

d. Menambah *library files path* pada MSVC. Untuk penambahan *path* dapat dilakukan dengan cara memilih *Tools*, *Option*, kemudian pada *Option* memilih *Directory tab*, setelah itu pada *Show directories for* diubah agar menunjukan *Library files*. Kemudian dilakukan penambahan *path* yang diperlukan, *path* yang baru dibawa keatas daftar dengan menggunakan tombol panah keatas.



Gambar 4.2 Penambahan *library files path* pada MSVC

e. Menambahkan *library* pada *library module* di *project setting*. Dilakukan dengan cara memilih *Project*, *Settings* atau dengan menggunakan *shortcut alt*+F7, pada *Project Setting* memilih *link tab*, kemudian pada *Object* / *libary modules text box* ditambahkan *library* yang hendak di *link*. *Libary* yang ditambahkan antara lain ddraw.lib, dxguid.lib, dsound.lib, dan winmm.lib.



Gambar 4.3 Penambahan *Library* pada *library modul*

4.2. Hasil implementasi

Dalam pemrograman yang dilakukan untuk melakukan implementasi hasil rancangan yang telah dilakukan dan membantu pembuatan aplikasi maka dibuat fungsi – fungsi untuk digunakan dalam aplikasi. Berikut ini daftar fungsi yang dibuat disertai dengan penjelasan singkat kegunaan fungsi tersebut.

a. Fungsi dibawah ini digunakan untuk membuka gambar dan melakukan penggambaran.

```
struct TILEINFO
    {
    RECT rcSrc;//source rectangle
    POINT ptAnchor;//anchoring point
    RECT rcDstExt;//destination extent
    };

    class CTileSet
    {
    private:
    DWORD dwTileCount;
```

```
    TILEINFO*  ptiTileList;
    LPSTR lpszReload;
    LPDIRECTDRAWSURFACE7 lpddsTileSet;
    public:
    CTileSet();
    ~CTileSet();
    void Load(LPDIRECTDRAW7 lpdd,LPSTR lpszLoad);
    void Unload();
    TILEINFO* GetTileList();
    void ClipTile(LPDIRECTDRAWSURFACE7 lpddsDst,RECT* prcClip,int xDst,int
    yDst,int iTileNum);
};
```

b. Fungsi dibawah ini digunakan untuk membantu CTimer

```
LPDIRECTDRAWSURFACE7 LoadFromFile (LPDIRECTDRAW7 lpdd, LPCTSTR
lpszFileName);
DWORD ConvertColorRef(COLORREF crColor, DDPIXELFORMAT* pddpf);
```

c. Fungsi didalam CTimer digunakan untuk melakukan perhitungan waktu.

```
class CTimer
{
    private:
    DWORD dwStartClock;
    public:
    CTimer();
    DWORD StartClock(void);
    DWORD WaitClock(DWORD count);
};
```

d. Fungsi berikut ini digunakan dalam inisialisasi awal aplikasi.

```
void DS_Init();
void DD_Init();
void Prog_Init();
```

e. Fungsi berikut ini digunakan sebagai fungsi utama dalam pemrosesan data dan penggambaran pada aplikasi.

```
void Prog_Loop();
void Draw();
```

f. Fungsi berikut ini digunakan untuk membantu pemrosesan data dalam fungsi utama.

```
int HitCalculation(UNIT *TempAttacker,bool Special,UNIT TempDefender);
void LevelUp(UNIT *TempUnit,int ExpValue,int Level);
void MapChanger(int iTarget, int iWant,POINT pMiddle,int iRange);
void UnitMove(UNIT *TempUnit,int MaxMove);
bool FindPath(int TempMap[][MAPWIDTH],POINT ptStart,POINT ptEnd);
```

g. Fungsi berikut ini digunakan untuk membantu proses menampilkan data bergambar dalam fungsi utama.

```
void AttackFrame(UNIT *TempUnit,POINT ptTarget);
void Show(UNIT *TempUnit,POINT ptTemp,bool Test);
void TextMoveUp(POINT ptPos,int Pos,int Text,int Color,LPDIRECTDRAWSURFACE7
lpdds);
void Message(int Face);
```

h. Fungsi berikut ini digunakan untuk membantu membuka dan menjalankan suara yang dipakai.

```
LPDIRECTSOUNDBUFFER LoadSound ( LPDIRECTSOUND8 lpds, LPCTSTR
lpszFileName );
void SoundPlay(int sound);
```

i. Fungsi berikut ini digunakan dalam inisialisasi nilai awal permainan.

```
void Game_Init();
void Level1();
void Level2();
```

j. Fungsi berikut ini digunakan untuk menampilkan data statistik.

```
int DrawTextGDI(char *text, int x,int y,COLORREF color, LPDIRECTDRAWSURFACE7
lpdds);
int DrawRectangle(RECT rect, int color,LPDIRECTDRAWSURFACE7 lpdds,bool Fill);
```

k. Fungsi berikut ini digunakan untuk menutup aplikasi.

```
void Prog_Done();
void SoundRelease(LPDIRECTSOUNDBUFFER* lpdsb);
```

Dalam sub bab ini akan dilakukan penjelasan terhadap fungsi-fungsi yang digunakan dalam menjalankan aplikasi. Fungsi yang dijelaskan merupakan fungsi penting untuk menjalankan aplikasi.

4.2.1. WinMain

WinMain merupakan fungsi yang dipanggil oleh sistem sebagai *entry point* awal dalam aplikasi berbasis Win32. Isi dari prosedur WinMain dalam aplikasi ini dapat dibagi menjadi dua bagian penting, pembuatan Windows yang

digunakan dan perulangan *infinite* yang menjadi tempat proses aplikasi berjalan. Berikut ini fungsi dari WinMain.

```
int WINAPI WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR
lpCmdLine,int nShowCmd)
{

	...

	WNDCLASSEX wcx;
	wcx.cbSize=sizeof(WNDCLASSEX);
	wcx.style=CS_OWNDC | CS_HREDRAW | CS_VREDRAW | CS_DBLCLKS;
	wcx.lpfnWndProc=TheWindowProc;
	wcx.cbClsExtra=0;
	wcx.cbWndExtra=0;
	wcx.hInstance=hInstMain;
	wcx.hIcon=LoadIcon(NULL,IDI_APPLICATION);
	wcx.hCursor=LoadCursor(NULL,MAKEINTRESOURCE(BLUEARROW));
	wcx.hbrBackground=(HBRUSH)GetStockObject(BLACK_BRUSH);
	wcx.lpszMenuName=NULL;
	wcx.lpszClassName=WINDOWCLASS;
	wcx.hIconSm=NULL;

	if(!RegisterClassEx(&wcx)) return(0);

	hWndMain=CreateWindowEx(0, WINDOWCLASS, WINDOWTITLE, WS_POPUP |
WS_VISIBLE, 0,0,320,240,NULL,NULL,hInstMain,NULL);
```

Source kode diatas digunakan untuk membuat Windows. *Source* kode tersebut terdiri dari beberapa tahap antara lain membuat Window *class*, mendaftarkan Windows *class* pada Windows agar dapat dikenali, dan membuat Windows dengan menggunakan CreateWindowEx.

```
	if(!Prog_Init()) return(0);
	MSG msg;
	for(;;)
	{
		Time.Start_Clock();
		if(PeekMessage(&msg,NULL,0,0,PM_REMOVE))
		{
			if(msg.message==WM_QUIT) break;
			TranslateMessage(&msg);
			DispatchMessage(&msg);
		}
		Prog_Loop();
		Time.Wait_Clock(20);
	}
	Prog_Done();
	return(msg.wParam);
	...
```

Source kode diatas merupakan bagian dari WinMain yang berhubungan dengan pembuatan program aplikasi. Dari kode yang ada dapat terlihat cara kerja dari aplikasi dan hubungan yang ada antara WinMain dengan WindowProc. Pemanggilan WindowProc dan Prog_Loop terletak didalam suatu perulangan yang berjalan terus menerus sampai ada message WM_QUIT untuk menghentikan program aplikasi yang sedang berjalan.

4.2.1.1 Pemeriksaan waktu

Didalam prosedur WinMain terdapat fungsi yang digunakan untuk membatasi kecepatan dari jalannya aplikasi. Fungsi ini ditujukan untuk membatasi kecepatan aplikasi agar tidak melebihi batas kecepatan tertentu. Berikut ini deklarasi dari pembatas waktu yang digunakan.

```
class CTimer
{
private:
        DWORD dwStartClock;
public:
        CTimer();
        DWORD StartClock(void);
        DWORD WaitClock(DWORD count);
};
```

Terdapat dua buah fungsi yang digunakan untuk pemeriksaan waktu yaitu StartClock dan WaitClock. StartClock berguna untuk memulai perhitungan waktu dengan cara mendapatkan waktu dalam hitungan *milisecond* semenjak sistem dimulai menggunakan fungsi dari Win32 API GetTickCount(). WaitClock digunakan untuk menghitung perbedaan waktu antara waktu yang didapat dari StartClock dan waktu saat itu. Bila perbedaan waktu yang ada telah mencapai nilai tertentu baru perulangan dilanjutkan.

4.2.2. WindowProc

WindowProc dari aplikasi yang dibuat digunakan untuk memproses semua masukan yang masuk yang digunakan dalam aplikasi. Hasil dari pemrosesan data masukan tergantung dari jenis masukan itu sendiri dan tujuan masukan tersebut. Secara umum dalam aplikasi ini masukan yang berhubungan dengan mouse digunakan untuk memberikan maupun membatalkan perintah dan masukan yang menggunakan tombol *esc* digunakan untuk keluar dari program aplikasi. Berikut ini fungsi dari WindowProc.

```
LRESULT CALLBACK WindowProc(HWND hwnd,UINT uMsg,WPARAM wParam,LPARAM
lParam)
{switch(uMsg)
{
	case WM_KEYDOWN:
		{
			switch(wParam)
			{
			case VK_ESCAPE:
				{
					GAME_STATE=GAME_STATE_EXIT;
					return(0);
				}break;
			}
		}break;
...
	case WM_DESTROY:
		{
			PostQuitMessage(0);
			return(0);
		}break;
	}
	return(DefWindowProc(hwnd,uMsg,wParam,lParam));
}
```

Source kode diatas merupakan cara kerja WindowProc untuk penekanan tombol *esc* yang akan mengakibatkan program aplikasi. Pada saat penekanan tombol *esc* dilakukan nilai dari kondisi aplikasi diubah dari Game_State_Running ke Game_State_Exit. Pada saat perulangan memasuki fungsi Prog_Loop nilai dari kondisi akan menyebabkan perintah pada kondisi Game_State=Game_State_exit dijalankan. Perintah yang dijalankan pada

Game_State_Exit ditujukan untuk menutup aplikasi yang sedang berjalan. Hal ini dilakukan dengan menjalankan fungsi DestroyWindow, fungsi ini digunakan untuk menutup Windows yang dimaksud dengan mengirim message WM_DESTROY dan WM_NCDESTROY. Message yang dikirim digunakan untuk mengaktifkan fungsi PostQuitMessage pada WindowProc yang akan mengirim perintah WM_QUIT untuk keluar dari perulangan.

Ada dua macam nilai *return* yang digunakan pada Winproc ini, yaitu nilai 0 dan DefWindowProc. Nilai 0 digunakan untuk memberitahu Windows bahwa *message* ini telah diproses. Sedangkan DefWindowProc digunakan untuk memanggil default WindowProc untuk melakukan pemrosesan default pada *message* yang tidak diproses oleh aplikasi, hal ini dilakukan untuk memastikan setiap *message* akan diproses.

4.2.3. Inisialisasi awal

Proses inisialisasi awal terbagi menjadi beberapa tahap diantaranya inisialisasi komponen DirectX agar dapat digunakan, mempersiapkan tampilan gambar yang akan digunakan, dan mempersiapkan font yang digunakan. Fungsi dibawah ini merupakan fungsi utama yang dipanggil untuk proses inisialisasi.

```
void Prog_Init()

    ...
    AddFontResource("Comicbd.ttf");
    hfntMain=CreateFont(10,10,0,0,700,0,0,0,0,0,0,0,0,"Comicbd");
    ...
```

Source kode diatas merupakan bagian dari fungsi Prog_Init yang mengatur pengaturan *font* yang digunakan. Ada dua fungsi yang digunakan dalam pengaturan *font* yang digunakan, AddFontResource yang digunakan untuk

menambahkan *font* resource ke sistem *font* tabel dan CreateFont yang digunakan untuk membuat *font* dengan karakteristik tertentu.

4.2.3.1. DirectDraw

Inisialisasi penggunaan DirectDraw dalam program aplikasi dilakukan oleh fungsi DD_Init. Inisialisasi DirectDraw yang dilakukan dapat dikelompokkan menjadi dua bagian, berikut ini fungsi inisialisasi DirectDraw

```
void DD_Init()
{
        DirectDrawCreateEx(NULL,(void**)&lpdd,IID_IDirectDraw7,NULL);
        lpdd->SetCooperativeLevel(hWndMain,DDSCL_EXCLUSIVE |
        DDSCL_FULLSCREEN | DDSCL_ALLOWREBOOT);
        lpdd->SetDisplayMode(800,600,16,0,0);
```

Source kode diatas digunakan untuk membuat DirectDraw objek dengan menggunakan DirectDrawCreateEx. Selain itu pada *source* kode diatas juga mengatur koperatif level dari DirectDraw serta mode tampilan yang digunakan, dalam hal ini pengaturan ditujukan untuk membuat *full screen* Windows yang mengijinkan penggunaan tombol *ctr+alt+del* dengan tampilan layar 800*600 dengan kualitas warna 16bit.

```
        DDSURFACEDESC2 ddsd;
        memset(&ddsd,0,sizeof(DDSURFACEDESC2));
        ddsd.dwSize=sizeof(DDSURFACEDESC2);
        ddsd.dwFlags=DDSD_CAPS | DDSD_BACKBUFFERCOUNT;
        ddsd.ddsCaps.dwCaps=DDSCAPS_PRIMARYSURFACE | DDSCAPS_COMPLEX |
        DDSCAPS_FLIP;
        ddsd.dwBackBufferCount=1;
        LPDIRECTDRAWSURFACE7 lpdds;
        lpdd->CreateSurface(&ddsd,&lpddsMain,NULL);
        DDSCAPS2 ddscaps;
        memset(&ddscaps,0,sizeof(DDSCAPS2));
        ddscaps.dwCaps=DDSCAPS_BACKBUFFER;
        lpddsMain->GetAttachedSurface(&ddscaps,&lpddsBack);
}
```

Source kode diatas mengatur pembuatan *primary surface* yang mempunyai satu *back buffer*. Deskripsi dari *primary surface* yang akan dibuat harus diatur terlebih dahulu melalui pengaturan DDSURFACEDESC2 *class*. *Primary surface*

dibuat berdasarkan data yang telah dipersiapkan dengan menggunakan fungsi CreateSurface. Setelah pembuatan *primary surface* dilakukan selanjutnya pada *source* kode diatas dilakukan pembuatan *backbuffer* dan mengatur pemasangan *backbuffer* pada *primary surface*.

4.2.3.2. DirectSound

Inisialisasi penggunaan DirectSound dilakukan dengan memanggil fungsi DS_Init. Didalam fungsi tersebut selain dilakukan penginisialisasian penggunaan DirectSound juga dilakukan proses membuka suara yang digunakan.

```
LPDIRECTSOUNDBUFFER LoadSound(LPDIRECTSOUND lpds, LPCTSTR lpszFileName );
void DS_Init();

    ...
    DirectSoundCreate8(NULL,&lpds,NULL);
    lpds->SetCooperativeLevel(hWndMain,DSSCL_NORMAL);
    ...
```

Source kode diatas merupakan bagian dari fungsi DS_Init yang digunakan untuk menginisialisasi penggunaan DirectSound pada program aplikasi. Suara yang akan digunakan dibuka dengan fungsi LoadSound. Fungsi ini akan menggunakan *library* WAVLoader untuk membaca format *file wav* kemudian memindahkannya ke DirectSound Buffer.

4.2.3.3. Membuka Gambar

Agar dapat menggunakan gambar yang telah dipersiapkan untuk aplikasi ini, maka gambar tersebut terlebih dahulu harus dibaca dan diproses dari bentuk aslinya terlebih dahulu. Fungsi dibawah ini digunakan untuk melakukan pemrosesan terhadap gambar yang akan digunakan.

```
struct TILEINFO
{
	RECT rcSrc;
	POINT ptAnchor;
	RECT rcDstExt;
};

class CTileSet
{
private:
	DWORD dwTileCount;
	TILEINFO*  ptiTileList;
	LPSTR lpszReload;
	LPDIRECTDRAWSURFACE7 lpddsTileSet;

public:
	CTileSet();
	~CTileSet();
	void Load(LPDIRECTDRAW7 lpdd,LPSTR lpszLoad);
	void Unload();
	TILEINFO* GetTileList();
	void ClipTile(LPDIRECTDRAWSURFACE7 lpddsDst,RECT* prcClip,int xDst,int
	yDst,int iTileNum);
};

LPDIRECTDRAWSURFACE7 LoadFromFile(LPDIRECTDRAW7 lpdd,LPCTSTR
lpszFileName);
DWORD ConvertColorRef(COLORREF crColor, DDPIXELFORMAT* pddpf);
```

Fungsi Load pada CtileSet digunakan untuk membuka gambar dan melakukan perhitungan agar memudahkan pemakaian gambar. Dalam pengerjaannya fungsi Load akan memanggil fungsi LoadFromFile untuk membuka gambar yang hendak digunakan dan menyimpannya dalam bentuk DirectDraw *offscreen surface*.

```
	...
	hbmNew=(HBITMAP)LoadImage(NULL,lpszFileName,IMAGE_BITMAP,0,0,
	LR_LOADFROMFILE);
	hbmOld=(HBITMAP)SelectObject(hdcMem,hbmNew);
	GetObject(hbmNew,sizeof(BITMAP),(LPVOID)&bmp);

	DDSURFACEDESC2 ddsd;
	memset(&ddsd,0,sizeof(DDSURFACEDESC2));
	ddsd.dwSize=sizeof(DDSURFACEDESC2);
	ddsd.dwFlags=DDSD_CAPS | DDSD_WIDTH | DDSD_HEIGHT;
	ddsd.ddsCaps.dwCaps=DDSCAPS_OFFSCREENPLAIN;
	ddsd.dwWidth=bmp.bmWidth;
	ddsd.dwHeight=bmp.bmHeight;
	LPDIRECTDRAWSURFACE7 lpdds;
	lpdd->CreateSurface(&ddsd,&lpdds,NULL);
	HDC hdcSurf;
	lpdds->GetDC(&hdcSurf);
```

```
BitBlt(hdcSurf,0,0,bmp.bmWidth,bmp.bmHeight,hdcMem,0,0,SRCCOPY);
...
```

Source kode diatas menunjukan cara kerja fungsi LoadFromFile. Fungsi tersebut akan membaca deskripsi gambar *bitmap* yang hendak digunakan dan menyimpannya dalam bentuk GDI. Dengan menggunakan deskripsi *bitmap* yang telah didapat, fungsi LoadFromFile akan membuat DirectDraw *surface* dengan ukuran yang sama. Setelah DirectDraw *surface* selesai dibuat dengan menggunakan fungsi BitBlt gambar akan ditransfer dari GDI surface ke DirectDraw *surface*.

Setelah gambar selesai dibuka maka fungsi Load akan melakukan perhitungan untuk mendapatkan banyaknya *tile* yang tersimpan dalam gambar tersebut. Selain itu juga mendapatkan nilai lainnya yang berguna untuk melakukan fungsi penggambaran.

```
...
DWORD dwColor = ConvertColorRef(crTransparent,&ddsd.ddpfPixelFormat);
DDCOLORKEY ddck;
ddck.dwColorSpaceHighValue=dwColor;
ddck.dwColorSpaceLowValue=dwColor;
lpddsTileSet->SetColorKey(DDCKEY_SRCBLT,&ddck);
...
```

Source kode diatas merupakan bagian terakhir dari fungsi Load. Pada bagian tersebut fungsi Load akan mengatur warna yang digunakan pada *bitmap* yang akan diatur sebagai warna transparan. Bila suatu warna ditetapkan sebagai warna transparan maka warna tersebut tidak akan diikutsertakan untuk digambar pada fungsi DirectDraw yang digunakan untuk menggambar Blt dan BltFast.

Selain fungsi Load, fungsi lainnya yang termasuk dalam CtileSet yang akan digunakan secara langsung ialah fungsi ClipTile. Fungsi ini menggunakan fungsi DirectDraw BltFast untuk menggambar pada *backbuffer*.

Fungsi destruktor pada CtileSet secara otomatis memanggil fungsi Unload utnuk melakukan pembersihan pada program yang berhubungan dengan gambar yang digunakan.

4.2.4. Perulangan aplikasi

Fungsi Prog_Loop merupakan fungsi yang digunakan didalam perulangan yang menjadi tempat pemrosesan data aplikasi yang digunakan. Melalui fungsi ini pemanggilan fungsi lainnya yang berhubungan dengan aplikasi dilakukan.

```
void Game_Init();
void Prog_Loop();
```

Fungsi Game_Init merupakan salah satu fungsi yang dipanggil oleh Prog_Loop pada awal aplikasi untuk menginisialisasi nilai - nilai yang dipakai dalam aplikasi. Pemanggilan fungsi ini dilakukan setiap kali suatu level aplikasi dijalankan.

```
switch(GAME_STATE)
{
	case GAME_STATE_INIT:
	{
	...
	GAME_STATE=GAME_STATE_MENU;
	}break;

	case GAME_STATE_MENU:
	{
	GAME_STATE=GAME_STATE_RESTART;
	}break;

	case GAME_STATE_RESTART:
	{
	GAME_STATE=GAME_STATE_RUNNING;
	}break;

	case GAME_STATE_RUNNING:
	{
		switch (GS)
		{
		...
			case(GS_AI)
			{
				switch(AI)
```

```
                                        {
                                        ...
                                        }
                                }break;
                        }
                }break;

                case GAME_STATE_EXIT:
                {
                ...
                }break;
}
Draw();
```

Source kode diatas merupakan bagian dari Prog_Loop dalam aplikasi ini. *Source* kode tersebut menunjukan cara kerja pembagian kondisi yang digunakan pada program aplikasi. Ada tiga kondisi utama yang digunakan dalam aplikasi ini yang terbagi dalam switch yang bersarang, yaitu

a. GAME_STATE, digunakan sebagai kondisi utama yang menjadi pemisah bagi kondisi jalannya aplikasi secara keseluruhan.
b. GS, digunakan sebagai kondisi bagi tindakan user. Berada didalam kondisi GAME_STATE. Pemrosesan semua tindakan yang diambil oleh user untuk menjalankan aplikasi berada didalam kondisi ini.
c. AI, digunakan sebagai kondisi bagi tindakan unit komputer. Berada di dalam kondisi GS_AI. Didalam kondisi ini semua tindakan unit komputer dilakukan.

Pembagian kondisi menjadi beberapa switch yang bersarang ditujukan untuk memudahkan membedakan kondisi keadaan yang sedang terjadi. Sebab apabila ada suatu tindakan yang diambil maka Prog_Loop hanya menjalankan bagian tertentu dari aplikasi sesuai dengan kondisi saat itu. Kondisi dari aplikasi saat itu dapat berupa hasil pemrosesan kondisi sebelumnya yang menunjuk ke kondisi berikutnya maupun sebagai akibat dari masukan user yang mengubah kondisi melalui pemrosesan WindowProc.

4.2.4.1. *Pathfinding* dan Pergerakan Unit

Untuk pergerakan unit pada aplikasi ini digunakan dua buah fungsi yang saling berkaitan yaitu fungsi FindPath dan fungsi UnitMove. Kedua fungsi ini digunakan untuk pergerakan semua unit yang ada yang bisa bergerak pada aplikasi ini. Fungsi dibawah ini merupakan fungsi Findpath dan fungsi UnitMove.

```
void UnitMove(UNIT *TempUnit,int MaxMove);
bool FindPath(int TempMap[][MAPWIDTH],POINT ptStart,POINT ptEnd);
```

Dalam aplikasi ini digunakan *array* 2*2 yang memiliki tipe data integer untuk menyimpan posisi dari objek yang digunakan. *Array* ini berfungsi sebagai peta yang digunakan untuk menyimpan lokasi dari objek yang tidak dapat dilewati, lokasi yang dapat dilewati oleh unit, dan jenis unit yang ada pada suatu posisi. Apabila pada *array* tersebut terdapat nilai lebih besar dari kosong maka pada saat pemeriksaan peta pada fungsi FindPath lokasi tersebut akan ditandai sebagai lokasi yang tidak dapat dilewati. Peta ini hanya menyimpan jenis dari unit yang ada pada posisi tertentu, bila suatu fungsi hendak menggunakan data suatu unit yang ada di peta maka fungsi tersebut harus mencari unit mana yang memiliki koordinat yang sama dengan koordinat unit pada peta tersebut dari daftar unit yang ada.

Fungsi FindPath digunakan untuk menemukan jalur menuju posisi yang dihendaki menggunakan peta yang ada, posisi awal dan posisi akhir. Pada saat user yang menggunakan, tujuan akhir dari fungsi FindPath diberikan oleh user melalui perintah move sedangkan posisi awal didapat dari data posisi unit tersebut. Tetapi jika unit komputer yang menggunakan fungsi FindPath tujuan akhir yang hendak dicapai merupakan posisi unit user yang hendak dicapai sedangakan posisi awal tetap merupakan posisi unit komputer tersebut.

Fungsi dari UnitMove digunakan untuk menampilkan pergerakan melalui jalur yang telah ditemukan oleh fungsi FindPath. Bila unit telah sampai ke tujuan atau kemampuan jalan dari unit telah habis sebelum unit sampai ke tujuan maka fungsi UnitMove akan dihentikan dan kondisi aplikasi diubah ke kondisi berikutnya.

4.2.4.2. Menampilkan data

Untuk menampilkan tulisan dan data dalam aplikasi ini digunakan fungsi dari GDI. Ada dua jenis tampilan data yang digunakan, tampilan huruf dan tampilan data berbentuk persegi. Untuk menampilkan kedua jenis data ini digunakan dua fungsi, fungsi DrawTextGDI dan DrawRectangle. Fungsi dibawah ini merupakan kedua fungsi yang digunakan untuk menampilkan data.

```
int DrawTextGDI(char *text, int x,int y,COLORREF color, LPDIRECTDRAWSURFACE7
lpdds);
int Draw_Rectangle(RECT rect, int color,LPDIRECTDRAWSURFACE7 lpdds,bool Fill);
```

```
...
SetTextColor(xdc,color);
SetBkMode(xdc, TRANSPARENT);
TextOut(xdc,x,y,text,strlen(text));
...
```

Source kode diatas merupakan bagian dari fungsi DrawTextGDI. Pada fungsi tersebut sebelum *text* hendak ditulis sebelumnya huruf yang akan digunakan diatur warnanya dan tipe *background* yang digunakan terlebih dahulu. Setelah pengaturan selesai dilakukan maka tulisan ditampilkan dengan menggunakan fungsi TextOut.

```
...
HBRUSH hbrush = CreateSolidBrush(color);
if (Fill)
      FillRect(xdc,&rect,hbrush);
else
FrameRect(xdc,&rect,hbrush);
...
```

Source kode diatas merupakan bagian dari fungsi DrawRectangle yang digunakan untuk mengatur warna dan jenis persegi yang digambar. Fingsi FillRect digunakan untuk membuat kotak yang bagian tengahnya juga terisi dengan warna.

4.2.4.3. Menampilkan gambar

Untuk menampilkan semua proses penggambaran dan perhitungan yang dilakukan pada pemanggilan satu fungsi Draw. Fungsi draw akan melakukan pemeriksaan dan perhitungan yang perlu untuk menentukan gambar mana yang harus ditampilkan sesuai dengan kondisi dan tampilan yang ada saat itu. Fungsi dibawah ini merupakan fungsi yang dipanggil untuk melakukan proses penggambaran.

```
void Draw();

...
for(;;)
        {

        ptCurrent=RowStart2;
        for(;;)
        {
        if(ptCurrent.x>=0 && ptCurrent.y>=0 && ptCurrent.x<MAPWIDTH && ptCurrent.y<
        MAPHEIGHT)
                {
                ptScreen=TilePlotter.PlotTile(ptCurrent);
                ptScreen=Scroller.WorldToScreen(ptScreen);
                tsBack.ClipTile(lpddsBack, Scroller.GetScreenSpace(),ptScreen.x,ptScreen.y,0);
                }
        if(ptCurrent.x==RowEnd2.x && ptCurrent.y==RowEnd2.y) break;
        ptCurrent=TileWalker.TileWalk(ptCurrent,ISO_EAST);
        }
        if(RowStart2.x==ptCornerLowerLeft.x && RowStart2.y==ptCornerLowerLeft.y) break;
        if(dwRowCount&1)
        {
                RowStart2=TileWalker.TileWalk(RowStart2,ISO_SOUTHWEST);
                RowEnd2=TileWalker.TileWalk(RowEnd2,ISO_SOUTHEAST);
        }
        else
        {
                RowStart2=TileWalker.TileWalk(RowStart2,ISO_SOUTHEAST);
                RowEnd2=TileWalker.TileWalk(RowEnd2,ISO_SOUTHWEST);
```

```
        }
        dwRowCount++;
}
...
```

Source kode diatas merupakan bagian dari fungsi Draw. Pada *source* kode diatas penggambaran dilakukan dalam dua buah perulangan yang baru akan dihentikan setelah penggambaran selesai dilakukan. Fungsi dari *library* digunakan untuk membantu dalam penempatan posisi unit atau objek yang hendak digambar.

4.2.5. Penutupan aplikasi

Proses penutupan aplikasi dilakukan melalui fungsi Prog_Done. Melalui fungsi ini aplikasi akan melepas semua DirectX interface dan font resource yang digunakan. Pelepasan terhadap interface DirectX yang digunakan dilakukan dengan memanggil fungsi Release. Fungsi dibawah ini merupakan fungsi yang dipanggil untuk melakukan proses penutupan aplikasi.

```
void Prog_Done();

        ...
        HDC hdc=GetDC(hWndMain);
        SelectObject(hdc,hfntOld);
        DeleteObject(hfntMain);
        ReleaseDC(hWndMain,hdc);
        RemoveFontResource("COMICBD.ttf");
        ...
```

Source kode diatas merupakan bagian dari fungsi Prog_Done yang digunakan untuk melepas *font resource* yang digunakan. Fungsi RemoveFontResource digunakan untuk menghilangkan *font* tertentu dari sistem *font* table