**LAPORAN KEGIATAN**

**KERJA PRAKTEK**

***PENGEMBANGAN PERANGKAT LUNAK HAULING DISTANCE UNTUK PEMODELAN TAMBANG***

PT. PAMA PERSADA NUSANTARA

Jalan Rawagelam 1 No.9

Jakarta Industrial Estate Pulogadung, Jakarta

Periode : 23 Juni 2014 - 24 Juli 2014



***Oleh :***

**Muhammad Ibnu Fahmi (5111100021)**

**Aldy Syahdeini (5111100122)**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA**

**2014**

# 

# LEMBAR PENGESAHAN I

**Judul : Pengembangan Perangkat Lunak Hauling Distance untuk pemodelan tambang**

**Lokasi : PT. PAMA PERSADA NUSANTARA**

**Jalan Rawagelam 1 No.9 Jakarta Industrial Estate Pulogadung, Jakarta**

**Periode : 23 Juni 2014 – 24 Juli 2014**

**Surabaya, 24 Juli 2014**

**Mengetahui,**

**Pembimbing Lapangan**

**Rahma Mustika Ningrum**

**NIK :**

# 

# LEMBAR PENGESAHAN II

**Judul : Pengembangan Perangkat Lunak Hauling Distance untuk pemodelan tambang**

**Lokasi : PT. PAMA PERSADA NUSANTARA**

**Jalan Rawagelam 1 No.9 Jakarta Industrial Estate Pulogadung, Jakarta**

**Periode : 23 Juni 2014 – 24 Juli 2014**

**Dosen Pembimbing,**

**Dr. Tohari Ahmad, S.Kom, MIT**

**NIP 132 306 296**

Daftar Isi

[LEMBAR PENGESAHAN I ii](#_Toc404774141)

[LEMBAR PENGESAHAN II iii](#_Toc404774142)

[Daftar Gambar vi](#_Toc404774143)

[BAB I PENDAHULUAN 1](#_Toc404774144)

[1.1. Latar Belakang 1](#_Toc404774145)

[1.2. Rumusan Masalah 2](#_Toc404774146)

[1.3. Tujuan dan Manfaat Proyek 2](#_Toc404774147)

[1.5. Metodologi 4](#_Toc404774148)

[1.6. Waktu Pelaksanaan 4](#_Toc404774149)

[1.7. Sistematika Penulisan 4](#_Toc404774150)

[BAB II PROFIL PERUSAHAAN 6](#_Toc404774151)

[2.1. Sejarah Pendirian PT. PAMA PERSADA NUSANTARA 6](#_Toc404774152)

[2.2. Visi Misi PT. PAMA PERSADA NUSANTARA 6](#_Toc404774153)

[2.3. Deskripsi Perusahaan 7](#_Toc404774154)

[BAB III TINJAUAN PUSTAKA 8](#_Toc404774155)

[3.1.*DirectX* 8](#_Toc404774156)

[3.1.1 Cara Kerja dari *DirectX* 8](#_Toc404774157)

[3.1.2 Versi dari *DirectX* 9](#_Toc404774158)

[3.2 *Direct3D* 10](#_Toc404774159)

[3.3 *Visual Studio* 11](#_Toc404774160)

[3.4 Basis data *oracle* 13](#_Toc404774161)

[BAB IV ANALISIS DAN PERANCANGAN 15](#_Toc404774162)

[4.1. Analisis Kebutuhan 15](#_Toc404774163)

[4.2. Definisi Umum Aplikasi 15](#_Toc404774164)

[4.3. Perancangan Antar muka 16](#_Toc404774165)

[4.3.1. Antar muka memilih *file* dalam *folder* 16](#_Toc404774166)

[4.4.2. Antar muka membuka dan membaca *file* dari *database* 17](#_Toc404774167)

[BAB V IMPLEMENTASI 22](#_Toc404774168)

[5.2 Implementasi Kelas Control.cs 30](#_Toc404774169)

[5.3 Implementasi kelas fileReaderMap.cs 30](#_Toc404774170)

[5.4 Implementasi Kelas fileReaderSt 33](#_Toc404774171)

[5.5 Implementasi kelas pointer.cs 34](#_Toc404774172)

[BAB VI UJICOBA DAN EVALUASI 39](#_Toc404774173)

[6.1 Lingkungan Uji Coba 39](#_Toc404774174)

[6.2 Uji Coba Perangkat Lunak 39](#_Toc404774175)

[6.2.1. Antar muka memilih dan membuka *file* dalam *folder* 39](#_Toc404774176)

[6.2.2. Antar muka Antar muka program ketika *map* dan jalan baru ditampilkan 40](#_Toc404774177)

[6.2.3. Antar muka *map* yang telah berisi area yang telah di-*drop* 41](#_Toc404774178)

[6.2.4. Antar muka fungsi *tooltip* dan *note* pada program 42](#_Toc404774179)

[6.2.5. Antar muka *file dialog* merubah warna 43](#_Toc404774180)

[6.2.6. Antar muka hasil fungsi *swap*, *add* dan *delete* pada jalan 43](#_Toc404774181)

[6.2.7. Antar muka *save* *file* 44](#_Toc404774182)

[BAB VII KESIMPULAN DAN SARAN 46](#_Toc404774183)

[7.1. Kesimpulan 46](#_Toc404774184)

[7.2. Saran 46](#_Toc404774185)

[DAFTAR PUSTAKA 48](#_Toc404774186)

# Daftar Gambar

[Gambar 4. 1 Halaman memilih *file* dalam f*older* 17](#_Toc404764224)

[Gambar 4. 2 Halaman membuka dan membaca *file* dari *database* 18](#_Toc404764225)

[Gambar 4. 3 Halaman pencarian pada *database* 19](#_Toc404764226)

[Gambar 4. 4 Halaman *save file* 20](#_Toc404764227)

[Gambar 4. 5 Halaman berisi area yang telah di-*drop* 21](#_Toc404764228)

[Gambar 4. 6 Halaman *file dialog* merubah warna 21](#_Toc404764229)

[Gambar 6. 1 Antar muka memilih dan membuka *file* dalam *folder* 4](#_Toc404764215)0

[Gambar 6. 2 Antar muka program ketika *map* dan jalan baru di tampilkan 4](#_Toc404764216)1

[Gambar 6. 3 Antar muka halaman *map* yang telah berisi area yang telah di-*drop* 4](#_Toc404764217)2

[Gambar 6. 4 Antar muka fungsi *tooltip* dan *note* pada program 4](#_Toc404764218)2

[Gambar 6. 5 Antar muka *file dialog* merubah warna 4](#_Toc404764219)3

[Gambar 6. 6 Antar muka hasil fungsi *swap*, *add* dan *delete* pada jalan 4](#_Toc404764220)4

[Gambar 6. 7 Antar muka *save* *file* 4](#_Toc404764221)5

Daftar Tabel

Tabel 1. Perkembangan Versi C# 13

Tabel 2. Spesifikasi Lingkungan Uji Coba Perangkat Lunak 39

# BAB I PENDAHULUAN

Bab pendahuluan menjelaskan segala hal yang berkaitan dengan latar belakang, permasalahan, tujuan dan manfaat proyek, batasan masalah, waktu pelaksanaan, dan sistematika penulisan yang digunakan dalam kerja praktik dan sistematika penulisan.

## 1.1. Latar Belakang

Semakin berkembangnya dunia teknologi informasi, khususnya pada bidang grafika komputer. Grafika komputer adalah suatu bidang yang mempelajari bagaimana menghasilkan suatu gambar menggunakan komputer. Berbagai kerangka kerja dalam grafika komputer telah di kembangkan salah satu kerangka kerja yang paling popular ialah *DirectX,* yang di bangun oleh microsoft*.*

*DirectX* sendiri sampai saat ini telah berkembang dengan versi terakhir yaitu 11.0. Berbagai bentuk dan metode pemodelan telah diterapkan dalam sekala kecil hingga besar, seperti pemodelan dan simulasi untuk mendukung proses pemetaan dan pengambilan keputusan.

Dengan adanya inovasi dari teknologi ini membuat semua bagian perusahaan berusaha mengembangkan kemampuannya, termasuk departemen CIS (*Computer Information System*) pada perusahaan PT. PA*MAP*ERSADA NUSANTARA (PAMA).

Untuk menyokong dan mengoptimalkan pekerjaan pada bagian yang lain, divisi *HDS* (*Hauling Distance*) berusaha menciptakan sebuah program yang dapat menampilkan data 2D yang merupakan representasi dari tambang dan jalan truk yang berada pada lokasi tambang milik PAMA, karena berbagai keterbatasan dari perangkat lunak yang ada sehingga diputuskan untuk melakukan pembaharuan sistem dan meningkatkan kemampuan dari perangkat lunak *HDS* menjadi *HDS* 3D.

Dengan adanya perangkat lunak ini, diharapkan dapat menampilkan peta lokasi tambang dalam bentuk *virtual* 3D yang dapat dilihat dari segala sudut dan dapat dilakukan beberapa *editing* seperti menambahkan pesan dan menurunkan tinggi dari sebuah area. Sehingga data *virtual* 3D terlihat nyata seperti obyek sebenarnya.

*HDS* 3D sendiri dimaksudkan untuk mempermudah komunikasi antara para pekerja di dalam perusahaan PAMA dalam berbagi informasi berbentuk visualisasi mengenai kondisi tambang dan proyek yang berjalan saat ini pada lokasi tertentu, dengan adanya perangkat lunak ini diharapkan akan menginkatkan kualitas dan efisiensi pekerja PAMA dalam memenuhi target-targetnya.

## 1.2. Rumusan Masalah

1. Bagaimana cara menampilkan data *Global Positioning System (GPS*) kedalam bentuk tiga dimensi?
2. Bagaimana cara mendeteksi *event* pada tetikus dan papan ketik?
3. Bagaimana cara melakukan translasi dan rotasi pada *map*?
4. Bagaimana cara mendapatkan posisi titik yang di pilih oleh tetikus?
5. Bagaimana menyimpan semua properti dari peta dan *tooltip*?
6. Bagaimana menyeleksi dan membuat sebuah area dalam peta?

## 1.3. Tujuan dan Manfaat Proyek

Adapun Tujuan dari proyek ini antara lain:

1. Mengimplementasikan data *GPS* kedalam bentuk tiga dimensi.
2. Sebagai sarana untuk merepresentasikan kondisi tambang.
3. Sebagai digital *map* untuk mempermudah *transfer* informasi antar pusat dan lokasi tambang.
4. Sebagai media simulasi dari bentuk tambang pada runtut waktu tertentu.
5. Sebagai penentu rute dari penggalian.

Adapun manfaat dari perangkat lunak *Hauling Distance*  ini antara lain:

1. Pengambilan Keputusan.
2. Dengan merepresentasikan kondisi tambang dapat digunakan sebagai saran pengambil keputusan.
3. Sebagai media komunikasi dari kondisi tambang.
4. Dengan data hasil pengolahan dari perangkat lunak dapat digunakan sebagai media komunikasi dari kondisi tambang.
5. Sebagai media simulasi.
6. Dengan melakukan *editing* dengan perangkat lunak, hasilnya dapat digunakan sebagai media awal untuk mempermudah simulasi pertambangan.

**1.4. Batasan Masalah**

1. Data pada tambang direpresentasikan dengan titik bukan bidang.
2. Mendapatkan masukan berupa *file* CSV untuk *map* dan rute dapat menyimpan data dengan format CSV juga, sedangkan untuk menyimpan atribut tambahan berupa *tooltip* disimpan dengan format .PMX.
3. Pembuatan satu jalan atau rute pada titik-titiknya harus berbeda.
4. Tidak ada pembuatan permukaan pada titik, hanya representasi data berupa titik-titik.

## 1.5. Metodologi

Untuk menyelesaikan permasalahan yang diberikan oleh kantor kerja praktik, kami menggunakan beberapa metodologi sebagai berikut:

1. Wawancara dengan bimbingan langsung

Selama kerja praktik, dalam menentukan bentuk struktur dari sistem yang akan dibuat, penyusun melakukan bimbingan langsung dengan pembimbing lapangan yaitu Ibu Rahma. Selanjutnya, kami melakukan bimbingan mengenai jalannya proses dan kebutuhan dari aplikasi yang akan kami buat.

1. Studi Literatur

Studi literatur dengan mempelajari kerangka kerja pemrograman *DirectX* dengan bahasa pemrograman *C#.* Selain itu, studi dilakukan untuk mengetahui jalannya proses representasi data ke dalam bentuk 3D. Untuk *integrated development environment* kami menggunakan *Visual studio 2013*.

## 1.6. Waktu Pelaksanaan

Kerja Praktik ini dilaksanakan selama tanggal 23 Juni 2014 sampai dengan 23 Juli 2014 di kantor PT Pamapersada Nusantara. Dengan jam kerja dimulai pada pukul 07.30 sampai dengan 16.30 WIB.

## 1.7. Sistematika Penulisan

Laporan kerja praktik ini terbagi menjadi 7 bab dengan rincian sebagai berikut:

**BAB I: PENDAHULUAN**

Berisi pendahuluan yang terdiri dari latar belakang, permasalahan, tujuan dan manfaat proyek, batasan masalah, waktu pelaksanaan, dan sistematika penulisan yang digunakan dalam kerja praktik.

**BAB II: PROFIL PERUSAHAAN**

Bab ini berisi penjelasan mengenai sejarah, deksripsi dan visi mauapun misi dari perusahaan.

**BAB III: TINJAUAN PUSTAKA**

Bab ini berisi dasar teori dari metode/teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

**BAB IV: ANALISIS DAN PERANCANGAN**

Pada bab ini dijelaskan mengenai desain antar muka aplikasi.

**BAB V: IMPLEMENTASI**

Bab ini menjelaskan tentang kode implementasi yang telah dibuat.

**BAB VI: UJI COBA DAN EVALUASI**

Pada bab ini menjelaskan tentang proses ujicoba fungsi-fungsi perangkat lunak dan hasil evaluasinya.

**BAB VII: KESIMPULAN DAN SARAN**

Bab ini berisi kesimpulan dan saran dari proses pelaksanaan tugas praktik.

# BAB II PROFIL PERUSAHAAN

Bab ini berisi tentang gambaran umum dari tempat kerja praktik yaitu PT. PAMA PERSADA NUSANTARA.

## 2.1. Sejarah Pendirian PT. PAMA PERSADA NUSANTARA

Pamapersada Nusantara, merupakan perusahaan perseroan yang lebih dikenal dengan nama PAMA. PAMA adalah anak perusahaan dari PT United Tractors Tbk dan masih termasuk dalam Astra *Group*. PAMA adalah sebuah perusahaan yang bergerak dibidang tambang. Kini PAMA telah memiliki lebih dari 10 anak perusahaan yang masing-masing bergerak dibidang tambang. Beberapa anak perusahaan PAMA yang berada di Kalimantan antara lain PT Prima Multi Mineral dan PT Kalimantan Prima Persada. PAMA berawal dari divisi pada PT. United Tractors yang mengerjakan proyek dibidang tambang pada tahun 1974. Kemudian pada tahun 1993, PAMA mulai berdiri sendiri dan terbentuklah PT. Pamapersada Nusantara. Sebagai anak perusahaan dari PT. United Tractors kini PAMA berkembang menjadi perusahaan sebesar PT. United Tractors. Sebagai kontraktor, banyak pekerjaan yang PAMA kerjakan, mulai dari mengelola sejumlah tambang batubara, mengelola proyek jalan, penggalian bumi, serta mengerjakan proyek bendungan yang mana seluruh lokasi pekerjaan itu berada di seluruh tanah air.

## 2.2. Visi Misi PT. PAMA PERSADA NUSANTARA

* Visi PT Pa*map*ersada Nusantara

Menjadi kontraktor pertambangan yang terkemuka di Indonesia dan menjadi pemimpin di bidang tersebut serta menjadi kontraktor pertambangan yang tetap memperhatikan alam sekitar.

* Misi PT Pa*map*ersada Nusantara

Memberikan pelayanan yang bermutu tinggi dan berkualitas baik kepada para pelanggan.

## 2.3. Deskripsi Perusahaan

PT PAMA PERSADA NUSANTARA terletak di Jl. Rawagelam 1 No.9, Jakarta Industrial Estate Pulo Gadung, Jakarta 13930 - Indonesia.

# BAB III TINJAUAN PUSTAKA

Bab ini berisi dasar teori dari metode/teknologi yang digunakan dalam menyelesaikan kerja praktik.

## 3.1.*DirectX*

*Microsoft* *DirectX* ialah sebuah koleksi dari *Application Programming Interfaces* (*API*) untuk menangani pekerjaan yang berkaitan dengan multimedia khususnya pemrograman permainan dan *video*, pada *platforms microsoft*. Nama dari semua *APIs* dimulai dengan *Direct*, seperti *Direct3D, DirectDraw, DirectMusic, DirectPlay, DirectSound,* dan seterusnya. Nama *DirectX* sendiri diciptakan sebagai singkatan dari semua *APIs* tersebut dan selanjutnya menjadi nama dari koleksi *API*. Saat microsoft memutuskan untuk mengembangkan sebuah *console game*. Kata X digunakan sebagai basis nama dari *Xbox* untuk mengidentifikasikan bahwa *console* tersebut berdasarkan teknologi dari *microsoft* *DirectX* [1].

### 3.1.1 Cara Kerja dari *DirectX*

Komponen utama *DirectX* terdiri dari tiga matrik yaitu

1. Matrik *world*

Merupakan matrik yang menunjukkan letak vektor asli sesuai pada bidang X, Y, Z dengan pusat 0, 0, 0.

1. Matrik *view*

Matrik ini berfungsi sebagai pengubah sudut pandang dari layar yang bersifat dua dimensi terhadap vektor obyek.

1. Matrik *projection*

Yang bertugas seperti sebuah kamera, yaitu untuk memproyeksi semua vektor dari *world*, seperti translasi dan rotasi dari kamera [2].

Semua *vertex* yang akan di gambar pada matrik *world* akan di proyeksi oleh matrik proyeksi dan akan di ubah berdasarkan matrik *view*, sehingga dapat disimpulkan persamaan dasarnya yaitu:

Vektor pada Matrikworld x Matrik Projection x MatrikView = representasi Vektor di layar

### 3.1.2 Versi dari *DirectX*

Berikut merupakan versi-versi dari *DirectX* dan penjelasan pada tiap versinya:

1. *DirectX* versi 10

*DirectX* 10 hanya tersedia untuk windows vista dan setelahnya, versi *windows* yang lebih dulu seperti *windows XP* tidak bisa menjalankan *DirectX* 10, *DirectX* sepuluh sangat ekstensif. Banyak dari bekas *DirectX API* yang sudah tidak di gunakan lagi/kuno, dan di gantikan dengan *API* yang lebih baru, seperti *DirectInput* yang mulai di gantikan *XInput, DirectSound* yang di gantikan oleh *Cros-Plarform Audio Creation Tool System* (*XACT*) dan beberapa tambahan *API* lain yang digunakan untuk memaksimalkan akselerasi dari perangkat keras saat itu.

1. *DirectX* versi 11

Microsoft mengeluarkan *DirectX* 11 pada acara *Game festival* 8 di Seattle, dengan fitur yang telah diperbaharui termasuk dukungan pada *GPGPU/DirectCompute*, dan *Direct3D* 11 dengan kemampuan melakukan *multi-threading* untuk mendukung para pengembang permainan komputer untuk mengembangkan permainan komputer yang lebih memberdayakan kemampuan *multi-core processor*. *Direct3D* 11 berjalan pada *windows vista, windows 7* dan *windows 8*. Bagian dari *API* yang baru seperti penanganan sumber daya *multi-threaded,* serta *shader* dapat di *support* dengan versi ini.

Beberapa penambahan yang dilakukan pada *DirectX* 11 yaitu:

* *DirectX* 11.1

*DirectX* 11.1 mendukung sistem operasi *windows* *8*. *DirectX* 11.1 juga mendukung *WDDM* 1.2 untuk meningkatkan performa, dan meningkatkan integrasi dengan *Direct2D, direct3D* dan *DirectCompute*. Kepustakaan *Xinput* dari *XNA* *framewok*. *DirectX* 11.1 juga menambahkan fitur *stereoscopic* 3D untuk meningkatkan kualitas *video*.

* *DirectX* 11.2

*DirectX* 11.2 mendukung *windows 8.1*, dengan penambahan pada fitur *Direct2D* seperti realisasi geometri.

* *DirectX* 11.X

Untuk *DirectX* 11.1 khusus di jalankan pada *Xbox One*. Versi ini mempunyai penambahan fitur, seperti *draw bundle*, yang dianggap sebagai bagian dari *DirectX* 12.

1. *DirectX* versi 12

Versi terbaru dari *DirectX* ini diumumkan oleh *microsoft* GDC pada Maret 2014 dan ditargetkan untuk permainan komputer pada tahun 2015. Tujuan utama dari *DirectX* 12 adalah mengurangi *overread* pada perangkat keras [1]*.*

## 3.2 *Direct3D*

*Direct3D* ialah bagian dari *microsoft* *DirectX* *application programming interface (API)*. *Direct3D* tersedia untuk sistem operasi besutan microsoft, dan beberapa sistem operasi gratis yang mendukung perangkat lunak *wine. Direct3d* ialah basis dari grafika vektor *API* pada *Xbox* dan *Xbox* 360 sistem *console. Direct3D* biasa digunakan untuk melakukan pemrosesan pada grafik tiga dimensi pada aplikasi dimana performa sangat penting, seperti permainan komputer. *Direct3D* juga membuat aplikasi dapat berjalan secara *full screen* atau *window*, tergantung dari *programmer* yang mengatur fiturnya. *Direct3d* menggunakan akselerasi perangkat keras jika memang tersedia pada kartu grafis yang tersedia, hal ini membuat perangkat keras melakukan percepatan pada semua *3D rendering pipeline* atau bahkan akselerasi sebagian. Integrasi dengan teknologi *DirectX* yang lain membuat *Direct3d* dapat mengikutsertakan fitur seperti *video* *mapping*, *hardware* 3D *rending* pada bidang 2D dan menyediakan grafika 2D dan 3D pada hubungan media interaktif.

*Direct3D* ialah sebuah 3D *API*, yang mengandung banyak perintah untuk melakukan 3D *rendering*, namun, sejak versi 8, *Direct3d* digantikan oleh kerangka kerja *DirectDraw* yang mengambil alih kerja untuk melakukan *render* pada grafik 2D [3].

## 3.3 *Visual Studio*

*Visual Studio* ialah sebuah *integrated development environment* (*IDE*) dari microsoft yang digunakan untuk melakukan pengembangan perangkat lunak untuk sistem operasi *windows*, dan juga *website*, aplikasi *website* atau *web service.* *Visual studio* menggunakan *microsoft development platform* seperti *windows API,* *windows forms, windows presentation foundation* (WPF)*, windows store* dan *microsoft silverlight*. *Visual studio* dapat memproduksi *native code*.

*Visual studio* mempunyai sebuah kode *editor* yang mendukung *intelisSense* dan juga *code refactoring*, *Integrated debugger* pada *visual studio* bekerja baik menjadi *source-level debugger* dan *machine level debugger*. Perangkat yang sudah tersedia (*built-in*) termasuk desainer *form* untuk membangun aplikasi *GUI*, desainer *web*, desainer *class* dan desainer skema basis data. *Visual studio* memberikan keleluasaan bagi pengembang untuk meningkatkan fungsionalitas berupa *plug-ins* pada semua tingkatan.

*Visual studio* juga mendukung berbagai bahasa pemrograman, menyediakan *language-specific service*. Beberapa bahasa pemrograman yang telah terpasang pada visual studio ialah *C, C++, C++/CLI, VB.NET, C#* dan *F#* dan mendukung bahasa pemrograman lain seperti *Python* dan *Ruby* [4].

* 1. ***C#***

*C#* ialah sebuah paradigma bahasa pemrograman yang meliputi *strong typing, imperative, declarative, functional, generic, object-oriented (class-based)* dan *component-oriented programming disciplines.* Bahasa pemrograman ini di kembangkan oleh microsoft didalam prakarsa .*NET* dan kemudian di *Ecma* dan *ISO*. *C#* adalah satu dari bahasa pemrograman yang didesain sebagai c*ommon language infrastructure.*

*C#* dimaksudkan sebagai bahasa pemrograman yang simple, terbaru, *general-purpose,* dan bahasa pemrograman berorientasi obyek. Versi *C#* 5.0 ialah versi yang terbaru dan dirilis pada Agustus 2012.

Nama *C#* sendiri terinspirasi oleh notasi musik dimana # mengidentifikasikan bahwa nada yang dituliskan harus dibuat lebih tinggi. Simbol # juga menandakan empat + yang berarti peningkatan pada bahasa pemrograman pendahulunya yaitu *C++* [5].

Berikut merupakan daftar pengembangan dari versi *C#* dapat dilihat pada tabel 1.

Tabel 1. Perkembangan Versi *C#*

|  |  |  |  |
| --- | --- | --- | --- |
| Versi | Tanggal rilis | Rangka kerja *.NET* | *Visual studio* |
| *C# 1.0* | Januari 2002 | *.NET Framework 1.0* | *Visual Studio .NET 202* |
| *C# 1.2* | April 2003 | *.NET Framework 1.0* | *Visual Studio .NET 2003* |
| *C# 2.0* | November 2006 | *.NET Framework 2.0* | *Visual Studio 2005* |
| *C# 3.0* | November 2007 | *Net Framework 2.0, Net Framework 3.0, Net Framework 3.5* | *Visual studio 2008, Visual studio 2010* |
| *C# 4.0* | April 2010 | *.NET Framework 4* | *Visual Studio 2010* |
| *C# 5.0* | Agustus 2012 | *.NET Framework 4.5* | *Visual Studio 2012, Visual Studio 2013* |

## 3.4 Basis data *oracle*

Basis data *oracle* atau biasa disebut *oracle* *RDBMS*, ialah sebuah *object-relational database management system* yang di produksi oleh oracle.

Pada awalnya Larry Ellison dan dua temanya yang merupakan bekas teman kerjanya, Bob Miner dan Ed Oates, memulai untuk membuat sebuah konsultan yang di panggil *Software Development Laboratories* (*SDL*) pada 1977. *SDL* ialah asal mula dari perangkat lunak oracle. Nama oracle sendiri berasal dari sebuah nama sandi dari sebuah proyek dimana Ellison bekerja sebelumnya.

Oracle dan IBM menekankan pada *platform* menengah seperti *UNIX* dan *Linux*, sementara microsoft cenderung meraih pasar untuk kelas rendah.

Basis data *oracle* juga bersaing dengan basis data sumber-terbuka, seperti *PostgreSQL, Firebird,* dan *MySQL*. Perangkat lunak *EnterpriseDB* yang berbasis *PostgreSQL*, belum lama ini mengumumkan fitur yang kompatibel dengan *oracle* dengan harga yang sangat wajar dan murah [6].

# BAB IV ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan mengenai analisis kebutuhan dari aplikasi yang dibangun serta detail dari perancangan fungsional sistem, data, maupun antar muka.

## 4.1. Analisis Kebutuhan

Secara umum spesifikasi kebutuhan pengguna dari aplikasi yang dibangun meliputi:

1. **Melihat Peta Tambang**

Menampilkan data 3D dari suatu tambang sehingga dapat terlihat panjang lebar dan ketinggian suatu tambang.

1. **Melihat Jalan *Full Truck***

Menampilkan jalan secara 3D, sehingga dapat terlihat dimana daerah yang terlalu curam dan jalan yang baik.

1. **Melakukan *Editing* pada Jalan**

Melakukan perubahan pada struktur jalan, baik mengubah letak titik–titik dari jalan tersebut ataupun melakukan penambahan dan pengurangan titik–titik pada jalan tersebut.

1. **Menentukan Suatu Area**

Membuat suatu area menjadi area tertentu sehingga berbeda dengan area/lingkungan yang lainnya.

1. **Melakukan Perubahan Ketinggian pada Area**

Melakukan pengurangan atau penambahan pada suatu area, sehingga sesuai dengan keadaan nyata tambang tersebut.

## 4.2. Definisi Umum Aplikasi

Aplikasi ini merupakan program yang dapat menampilkan data 2D yang merupakan representasi dari tambang dan jalan truk yang berada pada lokasi tambang milik PAMA, karena berbagai keterbatasan dari perangkat lunak yang ada sehingga diputuskan untuk melakukan pembaharuan sistem dan meningkatkan kemampuan dari perangkat lunak *HDS* menjadi *HDS* 3D.

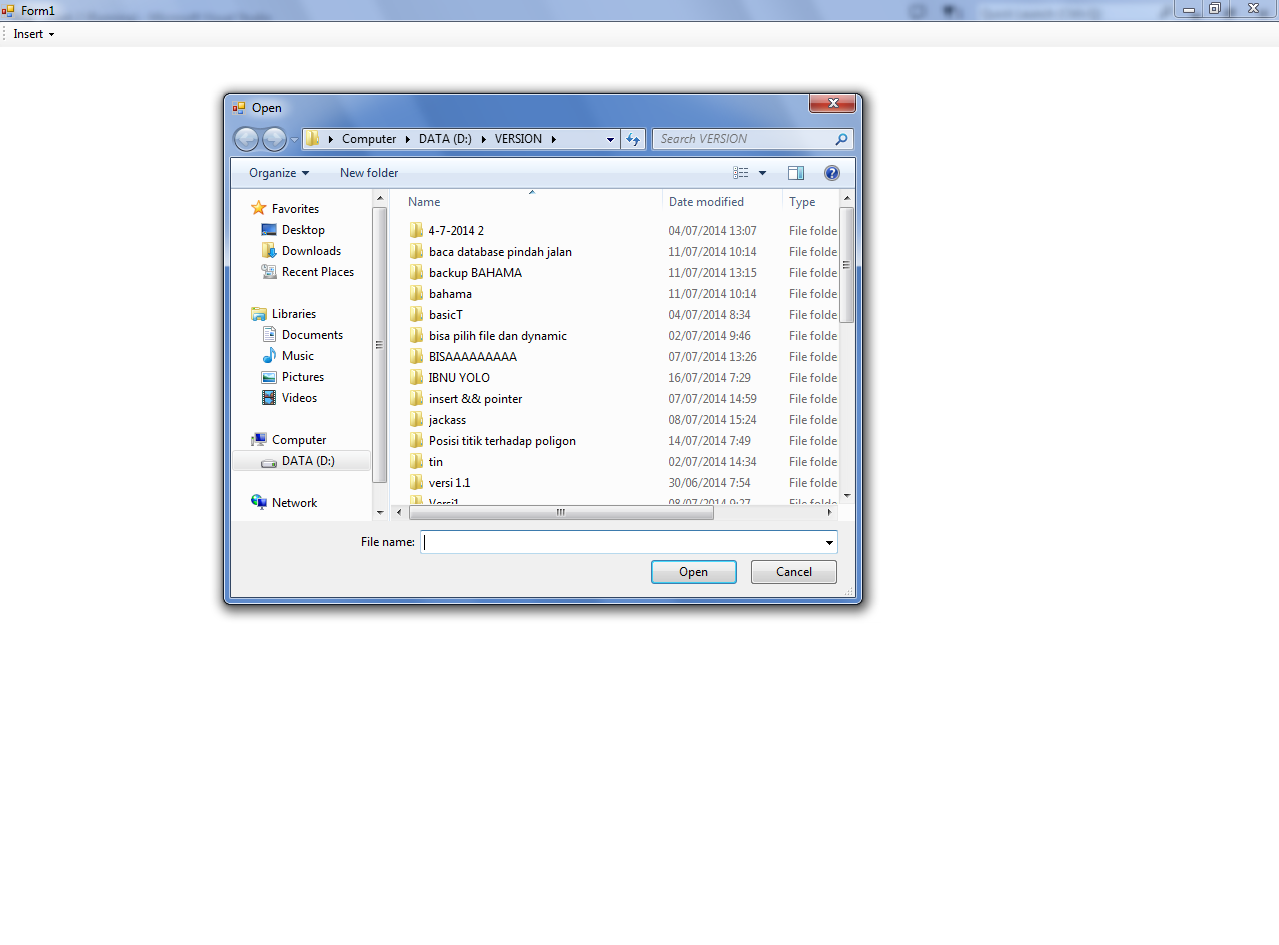
Dengan adanya perangkat lunak ini, diharapkan dapat menampilkan peta lokasi tambang dalam bentuk *virtual* 3D yang dapat dilihat dari segala sudut, dan dapat dilakukan beberapa *editing* seperti menambahkan pesan dan menurunkan tinggi dari sebuah area. Sehingga data *virtual* 3D terlihat nyata seperti obyek sebenarnya, perangkat lunak *HDS* 3D.

*HDS* 3D sendiri dimaksudkan untuk mempermudah komunikasi antara para pekerja di dalam perusahaan PAMA dalam berbagi informasi berbentuk visualisasi mengenai kondisi tambang dan proyek yang berjalan saat ini pada lokasi tertentu, dengan adanya perangkat lunak ini diharapkan akan menginkatkan kualitas dan keefisienan pekerja PAMA dalam memenuhi target-targetnya.

## 4.3. Perancangan Antar muka

### 4.3.1. Antar muka memilih *file* dalam *folder*

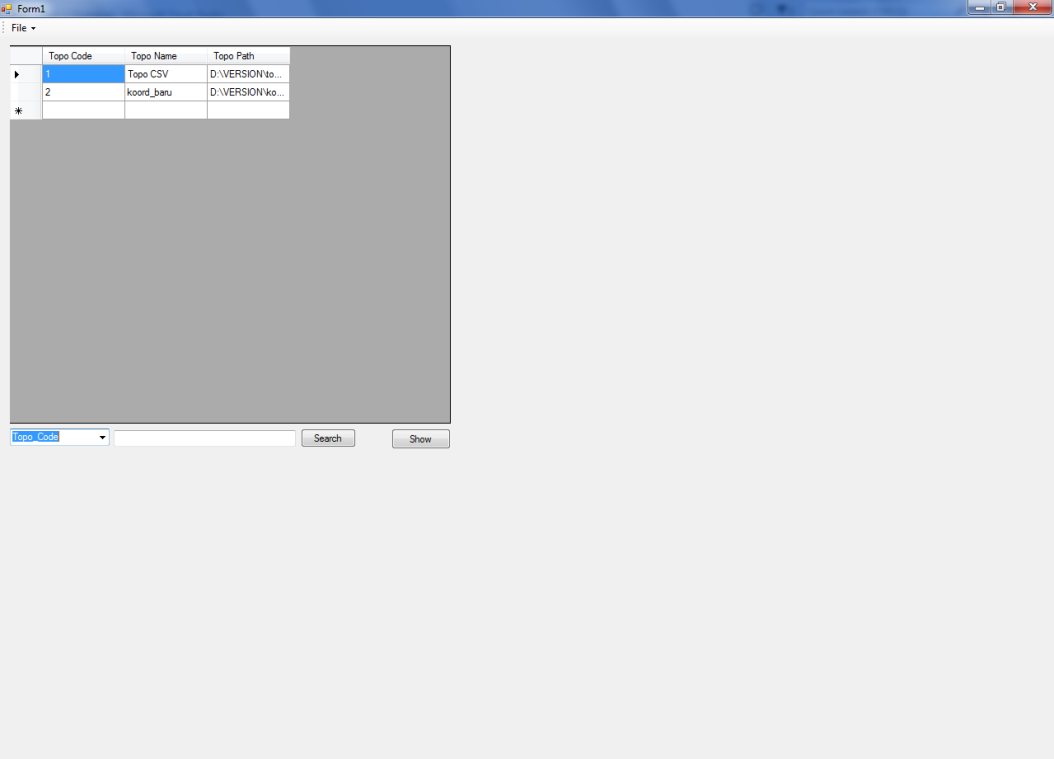
Antar muka Beranda digunakan untuk menampilkan keterangan materi yang dibahas dalam aplikasi ini. Gambar 4.1 menunjukkan halaman memilih *file* dalam *folder* yang akan ditampilkan ke layar. Kita dapat memilih *file map*, jalan atau jalan, serta *file* *tooltip*.



Gambar 4. 1 Halaman memilih f*ile* dalam f*older*

### 4.4.2. Antar muka membuka dan membaca *file* dari *database*

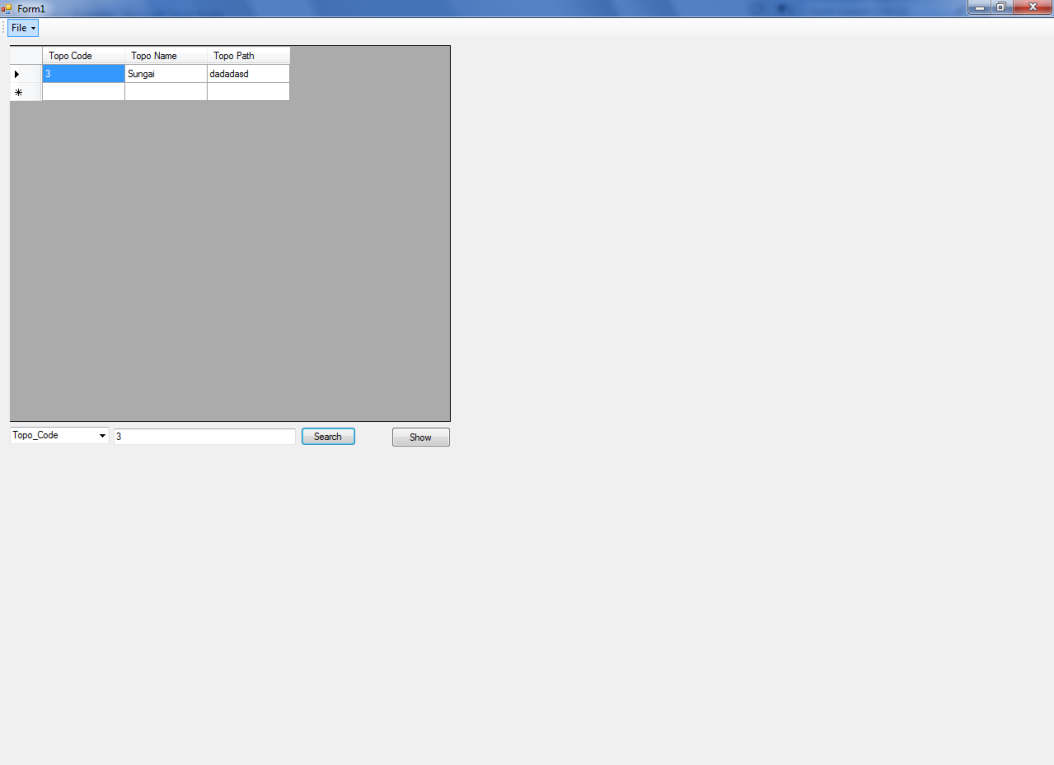
Antar muka membuka dan membaca *file* dari *database* digunakan untuk menampilkan keterangan kompetensi dari materi yang dibahas dalam aplikasi ini. Gambar 4.2 menunjukkan halaman membuka dan membaca *file* dari *database* baik *map* maupun jalan, yang nantinya akan ditampilkan pada program.



Gambar 4. 2 Halaman membuka dan membaca *file* dari *database*

4.3.3. Antar muka pencarian pada *database*

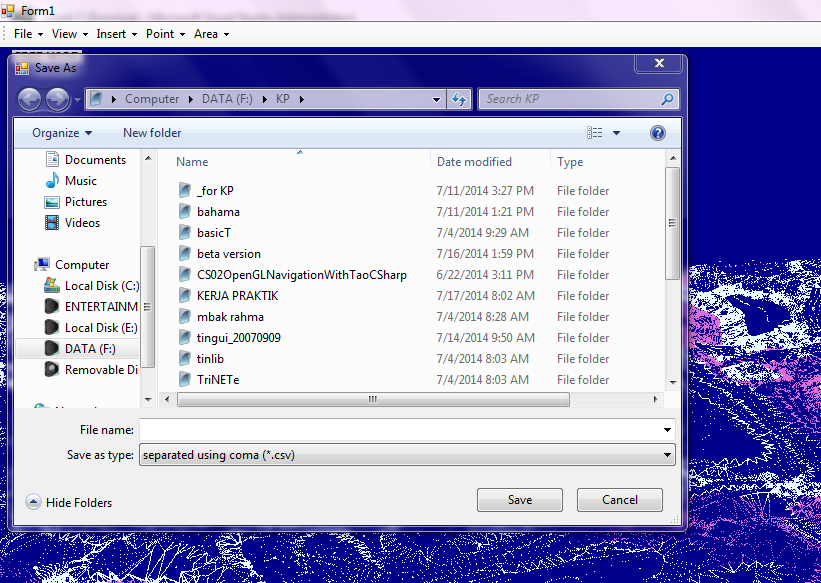
Antar muka pencarian pada *database* digunakan untuk melakukan pencarian pada *file* yang ada di *database*. Gambar 4.3 menunjukkan halaman pencarian pada *database* berdasarkan pilihan kriteria tertentu.



Gambar 4. 3 Halaman pencarian pada *database*

4.3.4. Antar muka *save* *file*

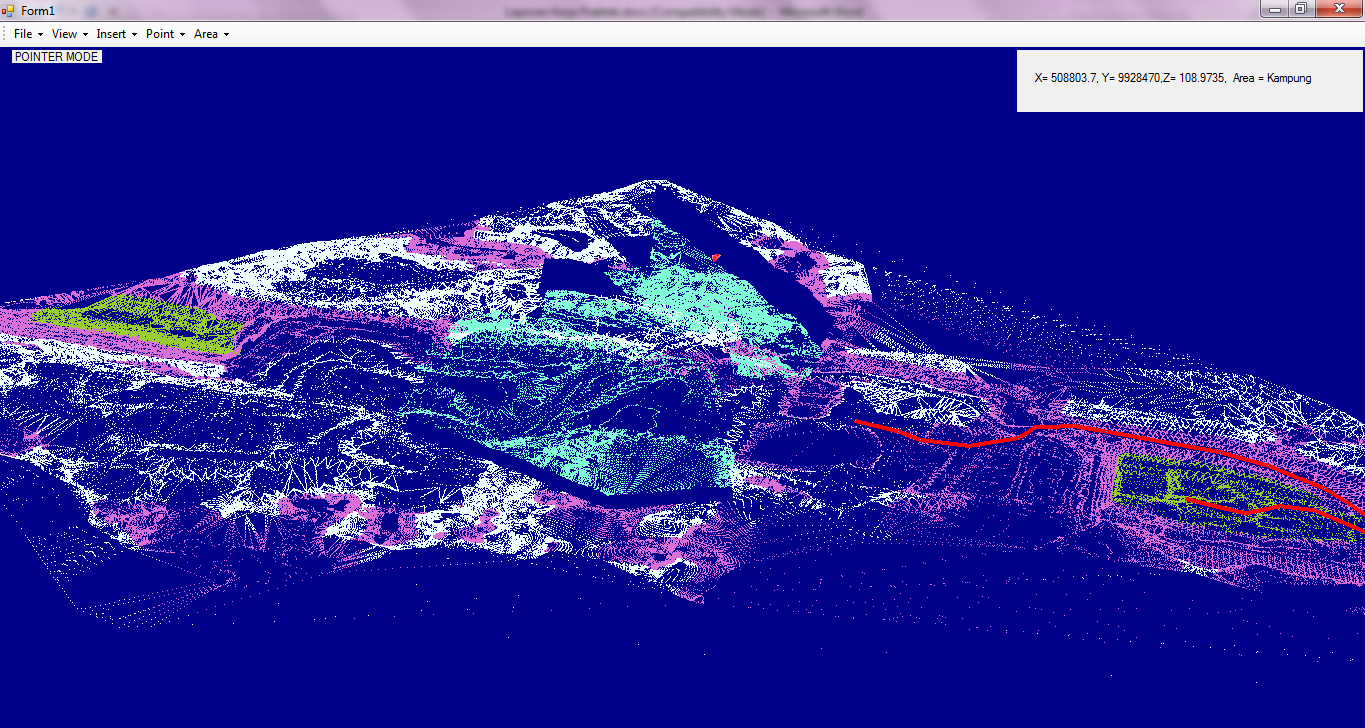
Antar muka *save file* digunakan untuk menyimpan *file*, baik itu *tooltip*, *map* dan jalan. Gambar 4.4 menunjukkan halaman *save* *file*.



Gambar 4. 4 Halaman *save file*

4.3.5. Antar muka *map* yang berisi area yang telah di-*drop*

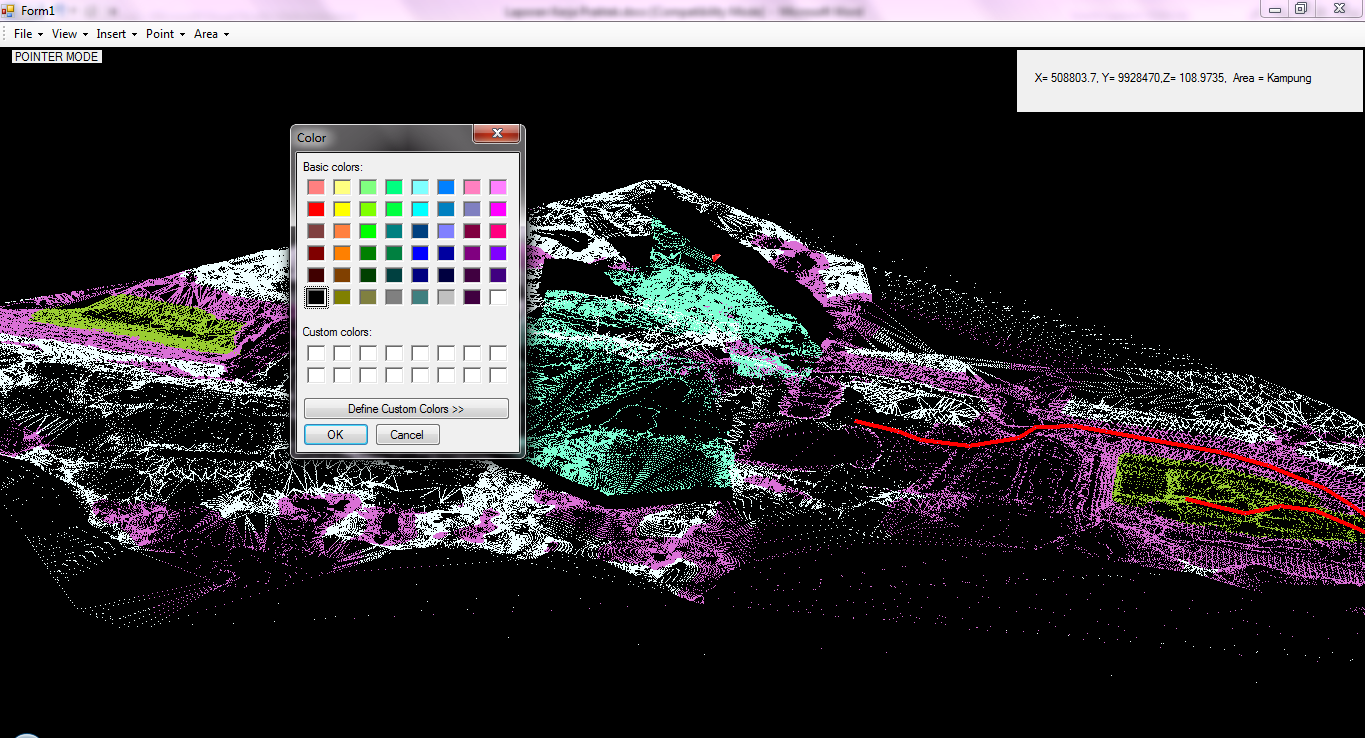
Antar muka *map* yang berisi area yang telah di-*drop* menunjukkan beberapa area yang telah di kurangi ketinggiannya. Gambar 4.5 menunjukkan halaman berisi *area* yang telah di-*drop*.



Gambar 4. 5 Halaman berisi area yang telah di-*drop*

4.3.6. Antar muka *file dialog* merubah warna

Antar muka *file dialog* merubah warna menunjukkan fitur tambahan dari program ini yang dapat menampilkan pilihan perubahan warna pada *program*. Gambar 4.6 merupakan halaman *file dialog* merubah warna.



Gambar 4. 6 Halaman *file dialog* merubah warna

# BAB V IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi berupa kode program yang digunakan untuk membangun aplikasi *Hauling Distance* 3D. Kode program telah terbagi menjadi beberapa fungsi dan setiap fungsi akan diberikan keterangan tentang maksud dari kode tersebut.

* 1. **Implementasi kelas form.cs**

Kelas form merupakan kelas utama dari aplikasi *HDS* 3D.  
Kelas ini berfungsi sebagai *form* utama, pengatur jalan utamanya aplikasi, mendeteksi *event* dan melakukan proses terhadap data, seperti terlihat pada kode sumber 5.1.

Terdapat beberapa fungsi utama pada kelas form yaitu:

1. Fungsi Inisialisasi

Tujuan dari fungsi inisialisasi ini adalah untuk melakukan pengaturan awal pada aplikasi seperti *mode* dan melakukan pembuatan obyek-obyek dasar, seperti di dalam metode *initializeAll*.

Pada metode *InitializeGraphics* dilakukan inisialisasi dari obyek *device* yang berguna sebagai penghubung antara grafik dan aplikasi, selain itu dilakukan *tagging* untuk *event handler* yaitu saat *device* pertama kali dibuat dan pada *event* tetikus. Pada metode *device\_DeviceReset* terdapat inisialisasi dari matrik utama *DirectX*.

|  |
| --- |
| public void initializeAll()  {  mode = Control.MODE.FREE;  labelMode.Text = "FREE MODE";  *pointer*Main = new *Pointer*(); // set *pointer* merah  data = new DataGrid();  *tooltip*s = new List<*Pointer*>(); // inisialisasi semua *tooltip*  \_vertexTempArea = new List<Vector3>();  VertexDeclaration(); // pindah semua vertex ke dalam vertices*Map*  InitializeGraphics();  /\*initialize optimization, untuk deteksi perubahan \*/  old\_projection = device.Transform.Projection;  old\_view = device.Transform.View; ;  old\_world = device.Transform.World;  }  /\* Initilize graphic,material and handler  \*/  public void InitializeGraphics()  {  presentParams.Windowed = true;  presentParams.*Swap*Effect = *Swap*Effect.Discard;  device = new *Microsoft*.DirectX.Direct3D.Device(0, *Microsoft*.DirectX.Direct3D.Device*Tipe*.Hardware, this, CreateFlags.SoftwareVertexProcessing, presentParams);  device.DeviceReset += new EventHandler(device\_DeviceReset);  device\_DeviceReset(null, null);  // tag event handler  this.MouseMove += new System.Windows.Forms.MouseEventHandler(ply\_Move);  this.MouseDown += new System.Windows.Forms.MouseEventHandler(ply\_MouseDown);  this.MouseUp += new System.Windows.Forms.MouseEventHandler(ply\_MouseUp);  this.MouseWheel += new System.Windows.Forms.MouseEventHandler(ply\_MouseWheel);  }  /\*Initialize all the object and material  \*/  private void device\_DeviceReset(object sender, EventArgs e)  {  #region Material initialization  device.*Render*State.CullMode = Cull.None;  device.*Render*State.ZBufferEnable = true;  device.*Render*State.Lighting = false;  device.*Render*State.FillMode = FillMode.Solid;  device.*Render*State.AntiAliasedLineEnable = true;  device.*Render*State.DitherEnable = true;  device.*Render*State.EnableAdaptiveTessellation = true;  device.*Render*State.IndexedVertexBlendEnable = true;  device.*Render*State.UseWBuffer = true;  Vector3 maxPos = fr*Map*.pointMax,  minPos = fr*Map*.pointMin,  delta;  #endregion  #region basic initialization parameter  // Centering ,it's basically translating the world to the center  delta = maxPos - minPos;  center = fr*Map*.getMean();  device.Transform.World = Matrix.Translation(center \* -1f); //Initial transformation  device.Transform.World \*= Matrix.RotationX(-1.75f); // Initial rotation  // Calculate for initial placing  panjang = Math.Max(delta.X, delta.Y);  distance = panjang \* 0.5f / (float)Math.Tan(Math.PI / 8) + delta.Z \* 0.5f;  // Calculate *save* rotate minimum projection  panjang = Math.Max(panjang, delta.Z);  device.Transform.Projection = Matrix.PerspectiveFovLH(  (float)Math.PI / scale, 1.0f, distance - panjang, distance + panjang + 1000);  #endregion  } |

Kode Sumber 5. 1 Fungsi inisialisasi pada form.cs

1. Fungsi Deklarasi

Fungsi ini untuk memindahkan data titik untuk *map* jalan dan area dari *file* yang telah di-*load* ke dalam variabel penampung yang nantinya digunakan untuk memproses atau dimasukan kedalam vektor untuk di *rendet,* terdapat tiga metode yaitu VertexDeclaration untuk mendeklarasikan *vertex* pada peta dan men­translasi­kan vektor tersebut, begitu juga pada VertexDeclarationSt untuk mendeklarasikan rute dan VertexDeclarationArea untuk mendeklarasikan area, seperti terlihat pada kode sumber 5.2.

|  |
| --- |
| \*/  public void VertexDeclaration()  {  /\*deklarasi semua titik pada *MAP*\*/  vertices*Map* = new CustomVertex.PositionColored[\_vertexTemp*Map*.Count]; // untuk menyimpan titik yang akan di draw  float distance = (fr*Map*.pointMax.Z - fr*Map*.pointMin.Z) / 3; // distance digunakan untuk membedakan vertex DEPTH low,medium,high  for (int i = 0; i < \_vertexTemp*Map*.Count; i++) // lakukan pemindahan posisi vertex dari \_vertexTemp ke vertices *Map*  {  vertices*Map*[i].Position = new Vector3(\_vertexTemp*Map*[i].point.X, \_vertexTemp*Map*[i].point.Y, \_vertexTemp*Map*[i].point.Z);  // Settting warna pada vertex, depth low,medium dan hihg  if (\_vertexTemp*Map*[i].point.Z < (fr*Map*.pointMin.Z + distance))  vertices*Map*[i].Color = VtColor[0].ToArgb();  else if (\_vertexTemp*Map*[i].point.Z < (fr*Map*.pointMin.Z + (distance \* 2)))  vertices*Map*[i].Color = VtColor[1].ToArgb();  else  vertices*Map*[i].Color = VtColor[2].ToArgb();  if (\_vertexTemp*Map*[i].nameArea != "area0") //kalo vertex masuk area  vertices*Map*[i].Color = Color.Aquamarine.ToArgb();  }  *pointer*Main.setVertice*Pointer*(new Vector3(\_vertexTemp*Map*[0].point.X, \_vertexTemp*Map*[0].point.Y, \_vertexTemp*Map*[0].point.Z), xyzLabel); // set titik awal dari *pointer* merah  // munculkan semua menubar  toolStrip*Drop*DownButton2.Visible = true;  toolStrip*Drop*DownButton4.Visible = true;  toolStrip*Drop*DownButton5.Visible = true;  toolStrip*Drop*DownButton3.Visible = true;  vertexAfterProject*Map* = new Vertex[vertices*Map*.Length]; //inisialisasi semua vertexAfterproject  }  /\* Deklarasi vertex Jalan  \*/  // Cuma di panggil sekali saja  private void vertexDeclarationSt()  {  // pindah semua point dari \_vertexTempSt ke verticesSt (untuk gambar line) dan verticesSt\_point (untuk gambar point)  for (int j = 0; j < \_vertexTempSt.Count; j++) // ambil list of jalan  {  if (verticesSt.Count <= j) // jika belum di deklarasikan (vertices.Count < j)  {  verticesSt.*Add*(new CustomVertex.PositionColored[\_vertexTempSt[j].Count]);  verticesSt\_point.*Add*(new CustomVertex.PositionColored[\_vertexTempSt[j].Count]);  for (int i = 0; i < \_vertexTempSt[j].Count; i++) // pindah semua isi dari vertexTemp ke verticesSt untuk di draw  {  verticesSt[j][i].Position = new Vector3(\_vertexTempSt[j][i].point.X, \_vertexTempSt[j][i].point.Y, \_vertexTempSt[j][i].point.Z);  verticesSt[j][i].Color = Color.Red.ToArgb();  verticesSt\_point[j][i].Position = new Vector3(\_vertexTempSt[j][j].point.X, \_vertexTempSt[j][i].point.Y, \_vertexTempSt[j][i].point.Z);  verticesSt\_point[j][i].Color = Color.Orange.ToArgb();  if (\_vertexTempSt[j][i].nameArea != "area0") // jika di dalam *file* yang di baca terdapat area  {  verticesSt[j][i].Color = Color.Peru.ToArgb(); // ubah nama jalan yang di dalam area  }  }  vertexAfterProjectJalan.*Add*(new Vertex[verticesSt[j].Length]); // isi nilai vertexAfterproject jalan  }  }  }  // digunakan untuk mengganti semua variable vertexJalan jika terjadi perbuahan pada vertexJalan  private void vertexDeclarationArea()  {  verticesArea = new CustomVertex.PositionColored[\_vertexTempArea.Count + 1];  // pindah semua \_vertexTempArea ke verticesArea untuk di draw  for (int i = 0; i <= \_vertexTempArea.Count; i++)  {  try  {  if (i == \_vertexTempArea.Count && \_vertexTempArea.Count != 1) // jika areanya di akhir maka gabungkan ujung vertex dengan vertex awal  {  verticesArea[i].Position = new Vector3(\_vertexTempArea[0].X, \_vertexTempArea[0].Y, \_vertexTempArea[0].Z);  verticesArea[i].Color = Color.Green.ToArgb();  }  else // jika tidak, pindahkan saja areanya  {  verticesArea[i].Position = new Vector3(\_vertexTempArea[i].X, \_vertexTempArea[i].Y, \_vertexTempArea[i].Z);  verticesArea[i].Color = Color.Green.ToArgb();  }  }  catch (Exception eads)  {  }  }  } |

Kode Sumber 5.2 Fungsi deklarasi pada form.cs

1. Fungsi untuk Me-*render­*

Fungsi ini digunakan untuk melakukan *render* terhadap semua titik sesuai dengan mode yang sedang aktif, seperti terlihat pada kode sumber 5.3.

|  |
| --- |
| public void *Render*()  {  device.Clear(ClearFlags.Target, BGcolor, 1.0f, 0);  if (afterRead*File* == true) // Check if *file* is already been readed  {  device.BeginScene();  // Inisialisasi device  device.Transform.View = Matrix.LookAtLH(  new Vector3(XAxis, YAxis, -distance),  new Vector3(XAxis, YAxis, 0.0f),  new Vector3(0.0f, 1.0f, 0.0f)  );  device.VertexFormat = CustomVertex.PositionColored.Format;  device.DrawUserPrimitives(Primitive*Tipe*.PointList, \_vertexTemp*Map*.Count, vertices*Map*); // mendraw semua verteces *Map*  device.Transform.Projection = Matrix.PerspectiveFovLH(  (float)Math.PI / scale, 1.0f, distance - panjang, distance + panjang  );  if (mode == Control.MODE.*POINTER*) //Draw semua *pointer* jika mode *pointer*  {  device.DrawUserPrimitives(Primitive*Tipe*.TriangleFan, 4, *pointer*Main.posPoint); // gambar semua *pointer* dan *tooltip*  foreach (*Pointer* pTemp in *tooltip*s)  device.DrawUserPrimitives(Primitive*Tipe*.TriangleFan, 4, pTemp.posPoint);  }  if (showSt) // Drawing jalan ,jika jalan sudah di load  {  for (int k = 0; k < verticesSt.Count; k++) //iterasi tiap jalan  {  device.DrawUserPrimitives(Primitive*Tipe*.LineStrip, verticesSt[k].Length - 1, verticesSt[k]);  device.DrawUserPrimitives(Primitive*Tipe*.PointList, verticesSt[k].Length, verticesSt\_point[k]);  }  }  if (makeArea) //digunakan drawing line terluar saat pembuatan area  {  vertexDeclarationArea();  if (\_vertexTempArea.Count > 0)  {  device.DrawUserPrimitives(Primitive*Tipe*.LineStrip, \_vertexTempArea.Count - 1, verticesArea); //gambar tiap titik yang di dapat pada \_vertexTempArea  }  }  device.EndScene();  device.Present();  }  }  /\* Initialize vertexes of Mine  \*/ |

Kode Sumber 5.3 Fungsi *render* pada form.cs

1. Fungsi *handler*

Fungsi *handler* berguna sebagai pendeteksi masukan dari pengguna, salah satu *handler* yang utama ialah mendeteksi klik pada tetikus, metode ini akan mendeteksi apakah klik tetikus menyentuh titik pada peta dan rute, melakukan seleksi area dan mendeteksi klik tetikus pada *tooltip*, pada kode program dibawah ini saya berikan ialah fungsi pendeteksi klik untuk kumpulan titik pada *map*, seperti terlihat pada kode sumber 5.4.

|  |
| --- |
| private void ply\_MouseDown(object sender, MouseEventArgs e)  {  #region check click *Map*  for (int i = 0; i < vertexAfterProject*Map*.Length; i++)  {  // Cek jika vertex masuk ke titik  if (X\_Mkoord > vertexAfterProject*Map*[i].point.X - epsilon && X\_Mkoord < vertexAfterProject*Map*[i].point.X + epsilon && Y\_Mkoord > vertexAfterProject*Map*[i].point.Y - epsilon && Y\_Mkoord < vertexAfterProject*Map*[i].point.Y + epsilon)  {  *save*Point = i;  if (makeArea) // Jika dalam mode makearea masukkan titik ke dalam point area  {  Vector3 tempArea = \_vertexTemp*Map*[i].point;  \_vertexTempArea.*Add*(tempArea);  }  if (!showPointAsGreen) //if the point is outside the jalan, set kembali *pointer* menjadi merah  *pointer*Main.set*Tipe*(*Pointer*.*TIPE*.normal\_red);  // making the copy of vertex and unproject it  Vertex temp = new Vertex(vertexAfterProject*Map*[i].point);  temp.*note* = vertexAfterProject*Map*[i].*note*;  temp.nameArea = \_vertexTemp*Map*[i].nameArea;  temp.point.Unproject(device.Viewport, device.Transform.Projection, device.Transform.View, device.Transform.World); // unproject vertex ke 3D world coordinate  if (!MultiDepth)  *pointer*Main.setVertice*Pointer*(temp, xyzLabel); // setting the vertice *pointer*  else //multidepth jika Z banyak  {  *pointer*Main.setVertice*Pointer*MultiDepth(temp, xyzLabel); // setting vertice *pointer* dengan methode multidepth  }  if (vertexAfterProject*Map*[i].*note* != "") //check if there is a *note* in the projectionVertex  {  *note*Container.Visible = true;  *note*TextBox.Text = vertexAfterProject*Map*[i].*note*;  }  else  {  *note*TextBox.Text = "";  *note*Container.Visible = false;  }  if (!MultiDepth)  break;  }  }  }  *Render*();  #endregion  } |

Kode Sumber 5.4 Fungsi *handler* pada form.cs

## Implementasi Kelas Control.cs

Kelas control merupakan kelas yang digunakan sebagai pertanda (*flags*) jalanya aplikasi, terdapat dua modus pada aplikasi yang disimpan dalam sebuah *struct* dalam kelas ini, yaitu modus FREE dan modus POINTER*,* seperti terlihat pada kode sumber 5.5.

|  |
| --- |
| /\*  \* This class will be control the nature of the HDS  \*/  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Week\_2  {  staticclassControl  {  publicenumMODE  {  FREE,  *POINTER*  };  }  } |

Kode Sumber 5.5 Kelas control pada control.cs

## Implementasi kelas fileReaderMap.cs

Kelas fileReaderMap seperti terlihat pada kode sumber 5.6 digunakan untuk menyimpan properti dari titik yang akan di-*load* yang berupa titik dari m*ap* tambang dan membaca dari *file*, kelas ini mempunyai dua metode yaitu:

1. readFile

Metode ini akan membaca *file* yang *path*-nya dikirim sebagai parameter dan mengembalikan sebuah List<vertex> yang merupakan data-data dari *file* yang telah diubah ke obyek *vertex* terlebih dahulu.

1. getMean

Metode ini digunakan untuk mendapatkan rata tengah X, Y, Z dari kumpulan data yang telah dibaca.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.IO;  using *Microsoft*.DirectX;  using *Microsoft*.DirectX.Direct3D;  using System.Windows.Forms;  namespace Week\_2  {  class*file*Reader*Map*  {  List<Vertex> vertexes;  publicList<Vector3> tempArea; // untuk menyimpan kumpulan vertex  string strline = "";  string[] \_value;  publicVector3 pointMin; // untuk menyimpan point Minimum  publicVector3 pointMax; // untuk menyimpan point Maximum  float sumX = 0, sumY = 0, sumZ = 0;  publicList<Vertex> read*File*(string *file*Name)  {  // set default point Min / Max  pointMin = newVector3(float.MaxValue, float.MaxValue, float.MaxValue);  pointMax = newVector3(float.MinValue, float.MinValue, float.MinValue);  try  {  StreamReader sr = newStreamReader(*file*Name);  vertexes = newList<Vertex>();  tempArea = newList<Vector3>();  // read *file* perline  while (!sr.EndOfStream)  {  strline = sr.ReadLine();  \_value = strline.Split(','); // split tiap ada ,  //*save* posisi dan nama area dari titik  Vertex temp = newVertex(float.Parse(\_value[0]), float.Parse(\_value[1]), float.Parse(\_value[2]));  temp.nameArea = \_value[3];  if (temp.nameArea != "area0") // jika area tidak default(area0) maka masukkan ke dalam list Area  tempArea.*Add*(newVector3(temp.point.X, temp.point.Y, temp.point.Z));  // Cari nilai minimum dan maximum dari titik  if (temp.nameArea == "area0")  {  if (pointMin.X > temp.point.X)  {  pointMin.X = temp.point.X;  }  if (pointMin.Y > temp.point.X)  {  pointMin.Y = temp.point.Y;  }  if (pointMin.Z > temp.point.Z)  {  pointMin.Z = temp.point.Z;  }  // Max  if (pointMax.X < temp.point.X)  {  pointMax.X = temp.point.X;  }  if (pointMax.Y < temp.point.X)  {  pointMax.Y = temp.point.Y;  }  if (pointMax.Z < temp.point.Z)  {  pointMax.Z = temp.point.Z;  }  }  sumX += temp.point.X;  sumY += temp.point.Y;  sumZ += temp.point.Z;  vertexes.*Add*(temp);  }  sr.Close();  return vertexes;  }catch(Exception ex)  {  MessageBox.Show(ex.Message);  returnnull;  }  }  // dapatkan rata-rata dari semua titik  publicVector3 getMean()  {  float \_x, \_y, \_z;  \_x = (pointMax.X - pointMin.X) / 2 + pointMin.X;  \_y = (pointMax.Y - pointMin.Y) / 2 + pointMin.Y;  \_z = (pointMax.Z - pointMin.Z) / 2 + pointMin.Z;  returnnewVector3(\_x, \_y, \_z);  }  }  } |

Kode Sumber 5.6 Kelas fileReaderMap.cs

## Implementasi Kelas fileReaderSt

Kelas fileReaderSt digunakan untuk menyimpan properti dari titik yang digunakan untuk menyimpan rute dan membaca dari *file*, seperti terlihat pada kode sumber 5.7. Kelas ini mempunyai dua metode yaitu:

1. read*File*

Metode ini akan membaca *file* yang *path*-nya dikirim sebagai parameter, dan mengembalikan sebuah List*<*vertex*>* yang merupakan data-data dari *file* yang telah diubah ke obyek *vertex* terlebih dahulu.

1. getMean

Metode ini digunakan untuk mendapatkan rata tengah X, Y, Z dari kumpulan data yang telah dibaca.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using *Microsoft*.DirectX;  using *Microsoft*.DirectX.Direct3D;  using *Microsoft*.DirectX.DirectInput;  using System.Drawing;  using System.Windows.Forms;  namespace Week\_2  {  [Serializable]  classVertex  {  publicVector3 point; //untuk menyimpan posisi  publicString *note*; //untuk menyimpan *note*  publicString nameArea = ""; //untuk menyimpan nama area  public Vertex(float x, float y, float z)  {  this.point = newVector3(x, y, z);  *note* = "";  nameArea = "";  }  public Vertex(Vector3 v3)  {  this.point = v3;  *note* = "";  }  publicvoid setPoint(Vector3 \_arg)  {  this.point = \_arg;  }  }  }  } |

Kode Sumber 5.7 Kelas vertex pada vertex.cs

## Implementasi kelas pointer.cs

Kelas pointer digunakan untuk menyimpan letak penunjuk dimana titik yang dipilih saat ini, kelas ini juga sebagai obyek dari *tooltip* yang menyimpan kode area dan pesan pada titik tertentu, obyek *pointer* berfungsi sebagai penanda dari titik yang diklik saat ini, pada kelas ini terdapat beberapa metode, seperti terlihat pada kode sumber 5.8.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using *Microsoft*.DirectX;  using *Microsoft*.DirectX.Direct3D;  using *Microsoft*.DirectX.DirectInput;  using System.Drawing;  using System.Windows.Forms;  namespace Week\_2  {  [Serializable]  class*Pointer*  {  publicCustomVertex.PositionColored[] posPoint = newCustomVertex.PositionColored[6];  public*TIPE* *tipe*;  publicString *note*;  publicVertex pin; // untuk menyimpan vertex yang sedang di tunjuk oleh *pointer*  publicVector3 *save*Point= newVector3(0,0,0);  publicenum*TIPE*// *tipe* dari *pointer*  {  normal\_red, static\_blue, static\_green, static*Note*\_yellow  }  public *Pointer*()  {  this.set*Tipe*(*Pointer*.*TIPE*.normal\_red);  *note* = "";  } |

Kode Sumber 5.8 Kelas Pointer

Pada kelas ini terdapat metode untuk mengatur properti dari obyek. Metode setNote berguna untuk mengeset pesan yang tersimpan pada titik dari kelas pointer saat ini, sedangkan metode setVerticePointer digunakan untuk mengeset titik dari *pointer* saat ini untuk modus Multidepth digunakan setVerticePointerMultidepth*,* seperti terlihat pada kode sumber 5.9.

|  |
| --- |
| publicvoid set*Note*(String \_arg) // untuk mengeset *note*  {  *note* = \_arg;  }  publicvoid setVertice*Pointer*(Vector3 temp, Label label) // set posisi dan update label  {  posPoint[0].Position = newVector3(temp.X, temp.Y, temp.Z + 25);  posPoint[1].Position = newVector3(temp.X - 10, temp.Y, temp.Z + 75);  posPoint[2].Position = newVector3(temp.X + 10, temp.Y, temp.Z + 75);  posPoint[3].Position = newVector3(temp.X + 10, temp.Y + 10, temp.Z + 75);  posPoint[4].Position = newVector3(temp.X - 10, temp.Y + 10, temp.Z + 75);  posPoint[5].Position = newVector3(temp.X - 10, temp.Y, temp.Z + 75);  if(pin!=null)  label.Text = "X= " + temp.X + ", Y= " + temp.Y + ",Z= " + temp.Z +", Area = " + this.pin.nameArea;  }  publicvoid setVertice*Pointer*MultiDepth(Vector3 temp, Label label) // untuk Multidepth  {  posPoint[0].Position = newVector3(temp.X, temp.Y, temp.Z + 25);  posPoint[1].Position = newVector3(temp.X - 10, temp.Y, temp.Z + 75);  posPoint[2].Position = newVector3(temp.X + 10, temp.Y, temp.Z + 75);  posPoint[3].Position = newVector3(temp.X + 10, temp.Y + 10, temp.Z + 75);  posPoint[4].Position = newVector3(temp.X - 10, temp.Y + 10, temp.Z + 75);  posPoint[5].Position = newVector3(temp.X - 10, temp.Y, temp.Z + 75);  if(pin!=null) // jika yang di tunjuk masi null (saat pertama di load, *pointer* belum menunjuk apa-apa)  {  if (*save*Point.X == temp.X && *save*Point.Y == temp.Y ) // jika X,Ynya sama maka Znya akan di tambah  label.Text += ",Z= " + temp.Z;  else  label.Text = "X= " + temp.X + ", Y= " + temp.Y + ",Z= " + temp.Z + ", Area = " + this.pin.nameArea;  }  if (*save*Point != temp) // Jika point yang disimpan tidka sama dengan point yang baru  *save*Point = temp;  }  publicvoid setVertice*Pointer*(Vertex \_v, Label label) //mengeset *pointer* dan label  {  if (\_v != null)  {  pin = \_v;  this.setVertice*Pointer*(\_v.point, label);  }  else  MessageBox.Show("please move the *pointer* first");  }  publicvoid setVertice*Pointer*MultiDepth(Vertex \_v, Label label)  {  if (\_v != null)  {  pin = \_v;  this.setVertice*Pointer*MultiDepth(\_v.point, label);  }  else  MessageBox.Show("please move the *pointer* first");  }  publicvoid setVertice*Pointer*(Vertex \_v,Vector3 temp, Label label)  {  pin = \_v;  this.setVertice*Pointer*(temp, label);  } |

Kode Sumber 5.9 Metode setter pada kelas pointer

metode selanjutnya ialah setTipe yang berguna untuk mengeset tipe dari *pointer*, terdapat tiga tipe dari *pointer* yaitu normal\_red, static\_blue, static\_yellow dan static\_green*.* Normal\_red digunakan sebagai penunjuk titik yang sedang aktif saat ini, sedangkan static\_blue digunakan untuk menunjukkan *tooltip*, static\_yellow digunakan untuk menunjukkan *tooltip* yang mempunyai pesan di dalamnya dan static\_green apabila *pointer* berada pada titik dalam rute atau jalan, seperti terlihat pada kode sumber 5.10.

|  |
| --- |
| publicvoid set*Tipe*(*TIPE* t)  {  //set warna dan *tipe* da ri *pointer*  *tipe* = t;  if (t == *TIPE*.normal\_red)  {  posPoint[0].Color = Color.Red.ToArgb();  posPoint[1].Color = Color.LightPink.ToArgb();  posPoint[2].Color = Color.Red.ToArgb();  posPoint[3].Color = Color.LightPink.ToArgb();  posPoint[4].Color = Color.Red.ToArgb();  posPoint[5].Color = Color.LightYellow.ToArgb();  }  elseif (t == *TIPE*.static\_blue)  {  posPoint[0].Color = Color.Blue.ToArgb();  posPoint[1].Color = Color.LightBlue.ToArgb();  posPoint[2].Color = Color.Blue.ToArgb();  posPoint[3].Color = Color.LightBlue.ToArgb();  posPoint[4].Color = Color.Blue.ToArgb();  posPoint[5].Color = Color.LightBlue.ToArgb();  }  elseif (t == *TIPE*.static\_green)  {  posPoint[0].Color = Color.Green.ToArgb();  posPoint[1].Color = Color.LightGreen.ToArgb();  posPoint[2].Color = Color.Green.ToArgb();  posPoint[3].Color = Color.LightGreen.ToArgb();  posPoint[4].Color = Color.Green.ToArgb();  posPoint[5].Color = Color.LightGreen.ToArgb();  }  elseif (t == *TIPE*.static*Note*\_yellow)  {  posPoint[0].Color = Color.Yellow.ToArgb();  posPoint[1].Color = Color.LightYellow.ToArgb();  posPoint[2].Color = Color.Yellow.ToArgb();  posPoint[3].Color = Color.LightYellow.ToArgb();  posPoint[4].Color = Color.Yellow.ToArgb();  posPoint[5].Color = Color.LightYellow.ToArgb();  }  }  publicVector3 getPointPos() //getting position dari *pointer*  {  returnnewVector3(posPoint[0].X, posPoint[0].Y, posPoint[0].Z - 25);  }  } |

Kode Sumber 5.10 Method setTipe dan getPointPos pada kelas pointer

# BAB VI UJICOBA DAN EVALUASI

Pada bab ini akan dibahas proses uji coba yang dilakukan untuk setiap fungsionalitas dari aplikasi *HDS* 3D. Selain itu, akan diuraikan pula hasil evaluasi dari proses uji coba tersebut.

## 6.1 Lingkungan Uji Coba

Uji coba dilakukan dengan menggunakan *notebook* yang telah ter-*install* *DirectX* 10. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dapat dilihat pada tabel 2.

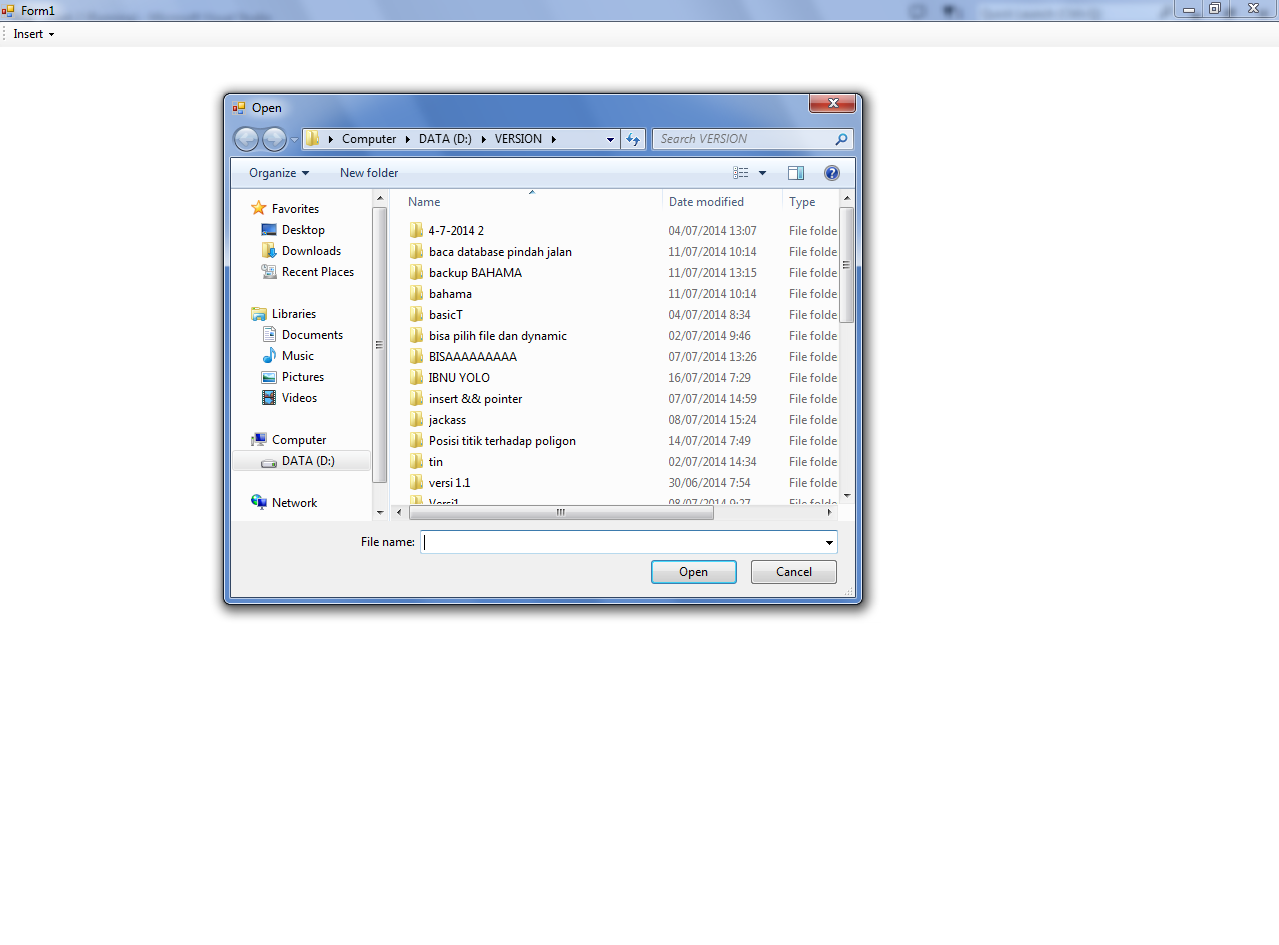
**Tabel 2. Spesifikasi Lingkungan Uji Coba Perangkat Lunak**

|  |  |
| --- | --- |
| Perangkat Keras | *Acer Aspire 4738Z*  *Processor* : 2,3 GHz  *Ram* : 2 Gb  *Memory* : 320 Gb |
| Perangkat Lunak | *Operating System* : *Windows 7*  *Microsoft* *Visual Studio 2012*  *DirectX 10.0* |

## 6.2 Uji Coba Perangkat Lunak

### 6.2.1. Antar muka memilih dan membuka *file* dalam *folder*

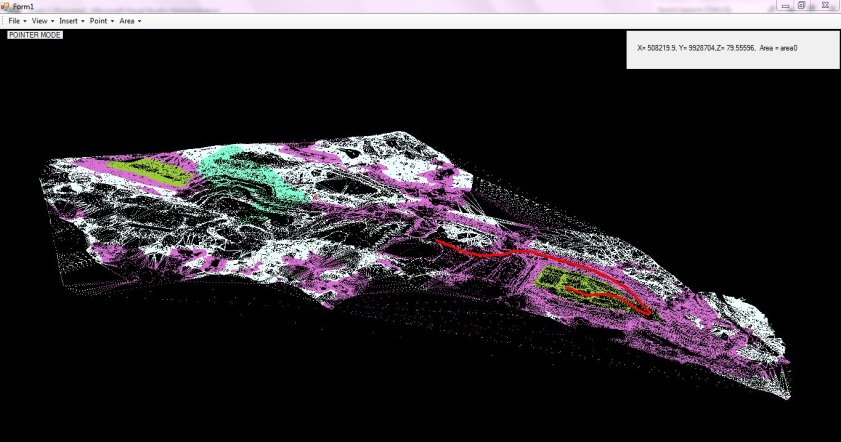
Gambar 6.1 merupakan memilih dan membuka *file* dalam *folder*. Pengguna dapat memilih *file map*, jalan maupun *tooltip* untuk ditampilkan pada program.



Gambar 6. 1 Antar muka memilih dan membuka *file* dalam *folder*

### Antar muka Antar muka program ketika *map* dan jalan baru ditampilkan

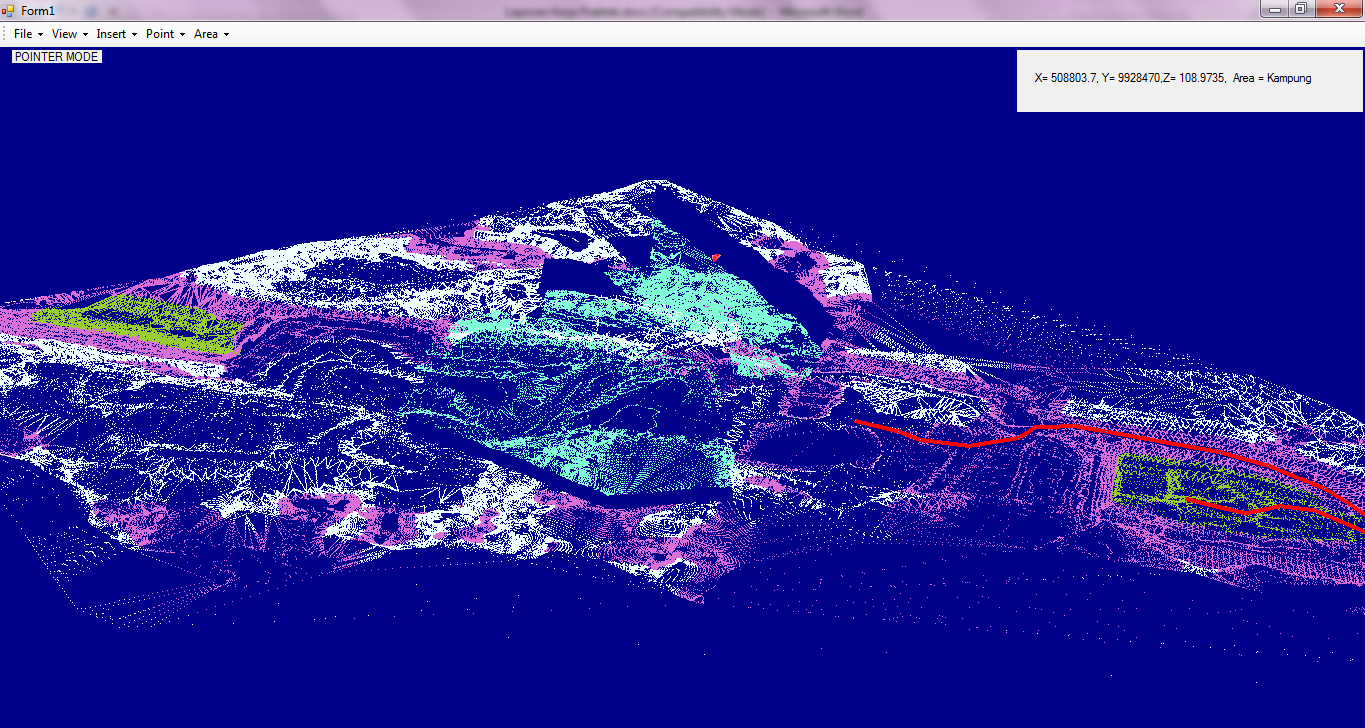
Gambar 6.2 merupakan hasil tampilan program ketika *map* dan jalan baru di tampilkan. Perbedaan warna pada *map* menunjukkan ketinggian yang berbeda-beda antar titiknya.



Gambar 6. 2 Antar muka program ketika *map* dan jalan baru ditampilkan

### Antar muka *map* yang telah berisi area yang telah di-*drop*

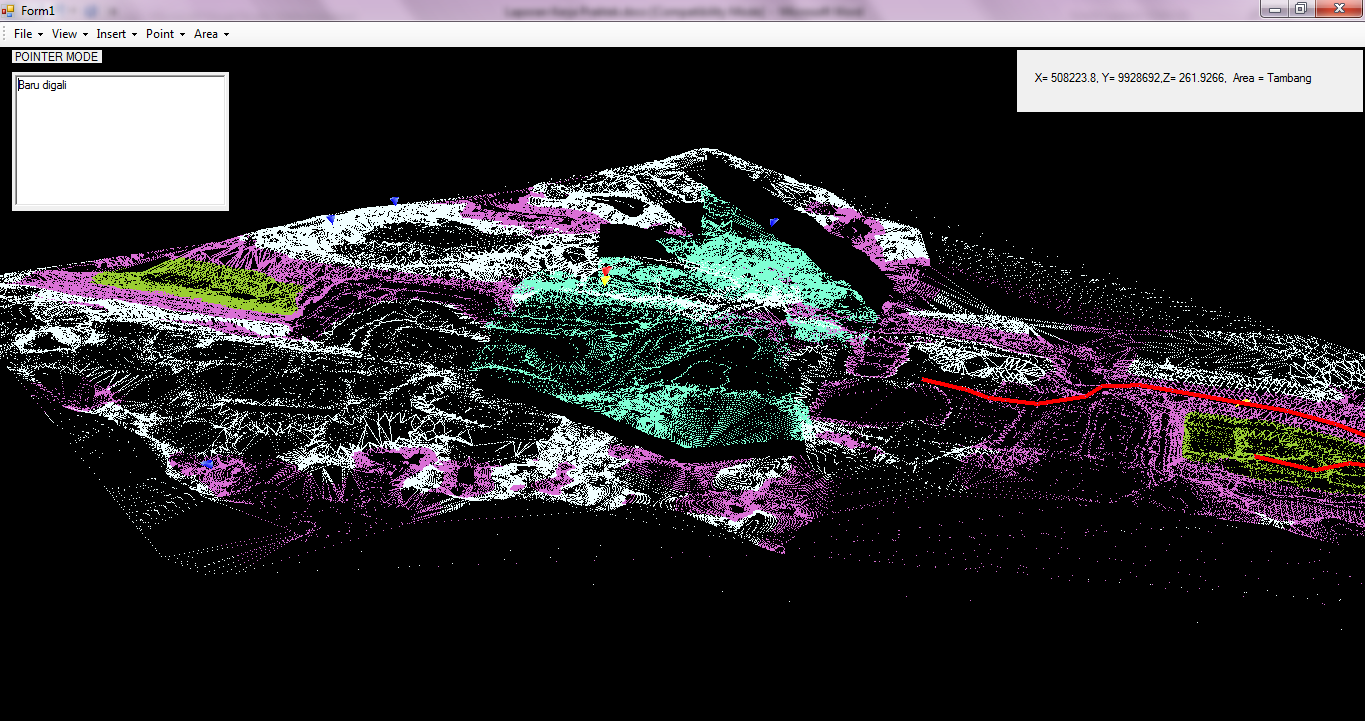
Gambar 6.3 merupakan tampilan pada halaman peta yang telah berisi area yang telah di-*drop*. Pemberian area pada peta akan otomatis mengubah warna pada area tersebut.



Gambar 6. 3 Antar muka halaman *map* yang telah berisi area yang telah di-*drop*

### Antar muka fungsi *tooltip* dan *note* pada program

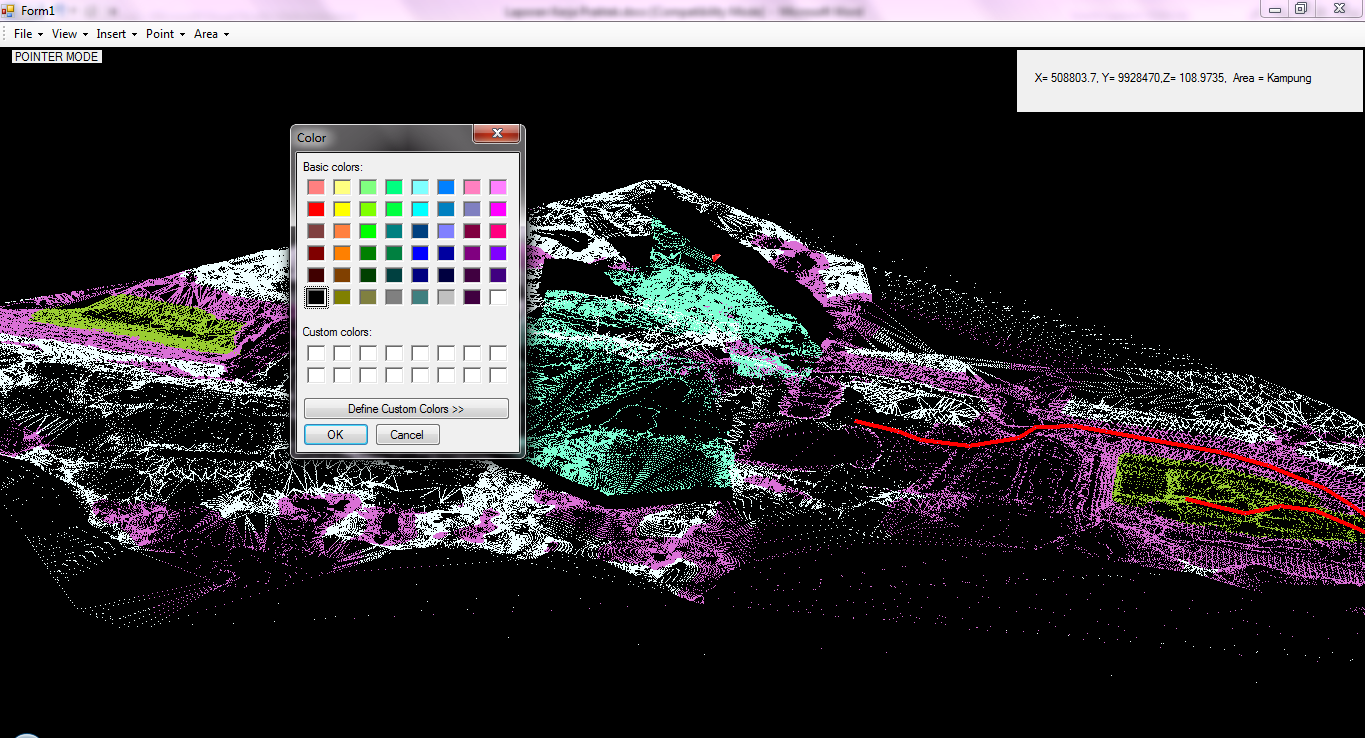
Gambar 6.4 merupakan tampilan fungsi *tooltip* dan *note* pada program. *Note* dan *tooltip* ini dapat digunakan oleh pengguna sebagai penanda suatu lokasi pada tambang.



Gambar 6. 4 Antar muka fungsi *tooltip* dan *note* pada program

### Antar muka *file dialog* merubah warna

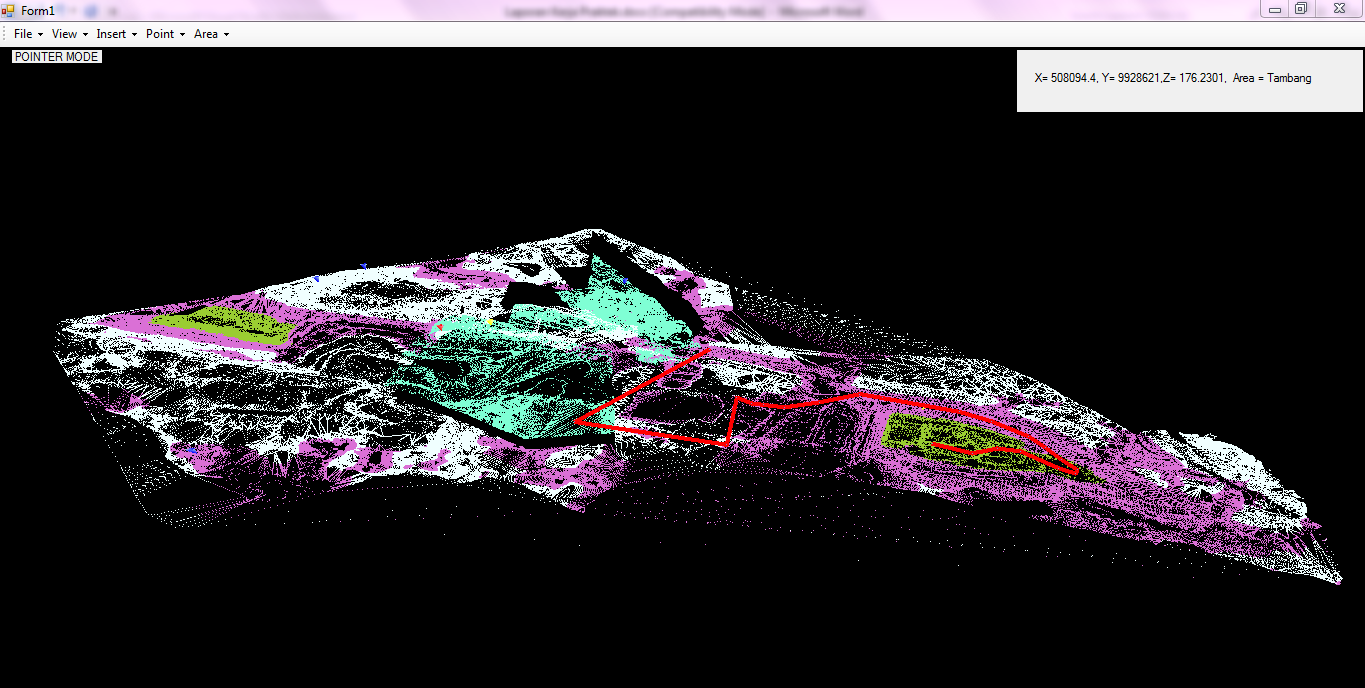
Gambar 6.5 merupakan tampilan *file dialog* merubah warna. Warna yang dapat diubah pada *map* antara lain *background* *map* dan ketinggian *map*.



Gambar 6. 5 Antar muka *file dialog* merubah warna

### Antar muka hasil fungsi *swap*, *add* dan *delete* pada jalan

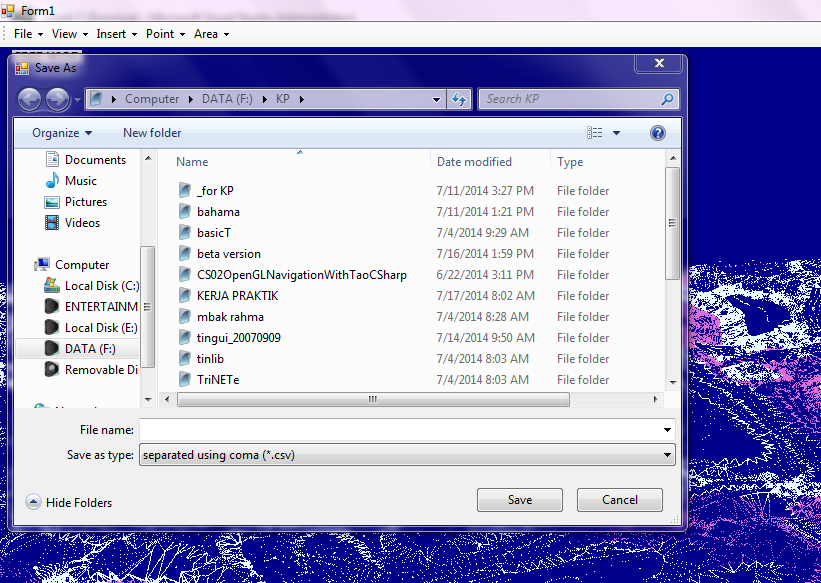
Gambar 6.6 merupakan tampilan hasil fungsi *swap, add*, dan *delete* pada jalan. Dengan fungsi tersebut pengguna dapat memindahkan, menukar, menghapus, maupun menambahkan titik-titik baru pada jalan.



Gambar 6. 6 Antar muka hasil fungsi *swap*, *add* dan *delete* pada jalan

### Antar muka *save* *file*

Hasil uji coba halaman *save* *file* yang ditunjukkan pada gambar 6.7. *Save file* digunakan agar setelah melakuakan perubahan pada *map* sewaktu-waktu pengguna dapat membuka perubahan terakhir pada *map*.



Gambar 6. 7 Antar muka *save* *file*

# BAB VII KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari proses pelaksanaan Kerja Praktik.

## 7.1. Kesimpulan

*Hauling Distance* (*HDS*) ialah sebuah perangkat lunak yang digunakan untuk memodelkan tambang dengan memanfaatkan teknologi grafika komputer *DirectX*, perangkat lunak ini membangkitkan titik-titik dalam bidang tiga dimensi yang merepresentasikan data tambang dari *GPS*.

*HDS* juga mempunyai fitur yang dapat digunakan untuk mempermudah manipulasi bentuk tambang dan rute pada tambang, serta mempartisi area yang digunakan dan menambahkan catatan dan menandai titik yang dianggap penting.

Seperti hal nya rangka kerja grafika komputer lain, *DirectX*  memanfaatkan matrik dan vektor untuk merepresentasikan data dan memanipulasinya, dengan memanfaatkan vektor ini posisi dan perbesaran benda serta interaksi antara pengguna dan model dapat dilakukan.

## 7.2. Saran

Perangkat lunak *HDS* perlu di kembangkan lagi untuk kedepan seperti penggunaan algoritma *triangulated irregular network* (*TIN*) sebagai pembangkit bentuk dari *terrain* atau *surface*, dan penggunaan algoritma *shortest path* seperti *Djikstra* juga diharapkan dapat di gunakan untuk membuat rute lebih realistis, tidak hanya dalam segi komputasi tapi basis dari perangkat lunak juga menurut kami perlu di-*update,* teknologi yang kami sarankan ialah *webgl* karena lebih sederhana dan dapat digunakan secara *online* dengan hanyamembutuhkan *web browser*.

# DAFTAR PUSTAKA

|  |  |
| --- | --- |
| [1] | Wikipedia, "Wikpedia," Wikipedia, 7 April 2013. [*Online*]. Tersedia: http://en.wikipedia.org/wiki/DirectX. [Diakses 23 Juli 2014]. |
| [2] | Riemers, "Riemers," Riemers, 2011. [*Online*]. Tersedia: http://www.riemers.net/eng/Tutorials/dxcsharp.php. [Diakses 24 Juli 2014]. |
| [3] | *Microsoft*, "*Microsoft*," *Microsoft*, 2011. [*Online*]. Tersedia: http://msdn.*microsoft*.com/en-us/library/windows/desktop/bb153264(v=vs.85).aspx. [Diakses 25 Juli 2014]. |
| [4] | Wikipedia, "Wikipedia," Wikipedia, 17 Juni 2014. [*Online*]. Tersedia: http://en.wikipedia.org/wiki/*Microsoft*\_Visual\_Studio. [Diakses 25 Juli 2014]. |
| [5] | Wikipedia, "Wikipedia," Wikipedia, 13 Juli 2014. [*Online*]. Tersedia: http://en.wikipedia.org/wiki/C\_Sharp\_(programming\_language). [Diakses 26 Juli 2014]. |
| [6] | Wikipedia, "Wikipedia," Wikipedia, 13 Juli 2014. [*Online*]. Tersedia: http://en.wikipedia.org/wiki/Oracle\_*Database*. [Diakses 29 Juli 2014]. |