BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

Kajian terdahulu diambil dari penelitian dengan judul "Penyimpanan Data Teks Ke Dalam Suara Digital Dengan Metode *Low Bit Coding*" oleh Marganda Papar Sihombing, 2004. Penelitian tersebut membahas tentang teknik *watermaking* atau hak cipta dengan metode *low bit coding* pada suara digital dalam format WAV, informasi *watermarking* yang akan disisipkan dan diekstrak ke dan dari suara digital berupa teks dengan format txt. Pembuatan program aplikasi menggunakan Borland C++ Builder.

Steganografi dengan judul "Aplikasi Video Steganography Dengan Metode Least Significant Bit" oleh Dian Dwi Hapsari, Lintang Yuniar Banowosari, 2009. Dalam penelitian ini aplikasi steganografi mampu membaca file video dengan tipe format 3gp, avi, flv, dan mpeg, sebagai media penyisipan, serta dapat menyembunyikan pesan rahasia ke dalam media video dengan berupa data gambar, teks, video, dokumen. Bahasa pemrograman yang digunakan adalah *Java 2 System Development Kit (J2SDK)* versi 1.4.2_03 dan perangkat lunak *Edit Plus* untuk mengimplementasikan koding program.

Dalam Tugas Akhir ini dilakukan pengembangan dari penelitian tersebut yaitu untuk menghasilkan aplikasi penyisipan pesan teks ke dalam file citra dengan tipe *bitmap* menggunakan algoritma *Least Significant Bit,* bahasa pemrograman menggunakan Borland Delphi 7.

2.2 Landasan Teori

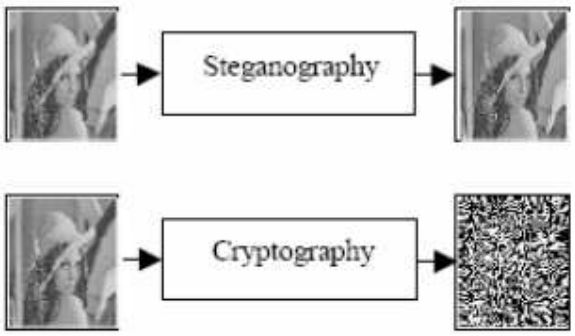
2.2.1 Pengertian Steganografi

Kata steganografi (*steganography*) berasal dari bahasa Yunani yaitu *Steganos* yang artinya tersembunyi, atau terselubung. Dan *Graphein* yang artinya menulis, sehingga kurang lebih artinya adalah menulis tulisan yang tersembunyi atau terselubung. Teknik ini meliputi banyak sekali metode komunikasi untuk menyembunyikan pesan rahasia. Metode ini termasuk tinta yang tidak tampak, *microdots*, pengaturan kata, tanda tangan digital, jalur tersembunyi dan komunikasi *spektrum* lebar. Atau dalam bahasa umum, steganografi adalah teknik menyembunyikan data atau pesan rahasia (*hiding message*) didalam media penampung berupa citra digital sehingga keberadaan (*eksistensi*) data rahasia tersebut tidak terdeteksi oleh indera penglihatan manusia. (Munir, 2004).

Steganografi membutuhkan dua properti yaitu media penampung dan data rahasia yang akan disembunyikan. Steganografi digital menggunakan media digital sebagai penampung, misalnya citra, suara (*audio*), teks, dan video. Sedangkan data rahasia yang akan disembunyikan juga dapat berupa citra, suara (*audio*), teks, atau video.

Penggunaan steganografi antara lain bertujuan untuk menyamarkan eksistensi (keberadaan) data rahasia sehingga sulit dideteksi, dan melindungi hak cipta suatu produk. Steganografi memiliki hubungan erat dengan kriptografi, tetapi metode ini sangat berbeda kriptografi. Kriptografi mengacak pesan sehingga tidak dimengerti, sedangkan steganografi menyembunyikan pesan sehingga tidak terlihat. Pesan dalam *ciphertext* mungkin akan menimbulkan

kecurigaan, sedangkan pesan yang dibuat steganografi tidak menimbulkan kecurigaan. Perbedaan antara steganografi dengan kriptografi dapat dilihat pada Gambar 2.1



Gambar 2.1 Perbedaan antara steganografi dengan kriptografi

Sebuah contoh klasik untuk menjelaskan steganografi adalah seperti persoalan tahanan penjara. Diilustrasikan Alisa dan Bobi berada sama-sama di penjara. Pada suatu saat keduanya akan menyusun rencana untuk kabur dari penjara. Alisa harus mengirimkan pesan tersebut melalui seorang pembawa pesan, yaitu Fredi. Jika Alisa mengenkripsi pesan rahasianya, Fredi tentu akan curiga. Oleh karena itu Alisa dan Bobi perlu menggunakan suatu teknik sehingga Fredi tidak dapat mendeteksi adanya pesan rahasia. Teknik tersebut dikenal dengan sebutan steganografi.

Gambar 2.2 Ilustrasi steganografi pada tahanan penjara

Pada Gambar 2.2 diatas, satu-satunya cara untuk berkomunikasi antara Alisa dan Bobi adalah melalui seorang sipir penjara yaitu Fredi. Alisa menulis surat pada selembar kertas, lalu surat tersebut diserahkan kepada Fredi. Fredi tentu saja dapat membaca isi surat tersebut sebelum disampaikan kepada Bobi. Hal yang sama juga dapat dilakukan Bobi bila ia hendak membalas pesan dari Alisa.

Jika Alisa ingin menulis pesan rahasia kepada Bobi mengenai rencana waktu pelarian mereka dari penjara. Pesan tersebut bunyinya adalah ”Lari jam satu”, bagaimana cara Alisa mengirim pesan tersebut tanpa diketahui oleh Fredi. Ada dua cara yang dapat digunakan, solusi pertama yaitu kriptografi dimana Alisa harus mengenkripsi pesan tersebut menjadi *ciphertext*. Fredi yang menyampaikan pesan tersebut pasti curiga dan menduga *ciphertext* tersebut merupakan pesan rahasia karena tulisannya yang tidak lazim sehingga kemungkinan Fredi akan menahan surat tersebut.

Alternatif 1 : mengenkripsinya menjadi *ciphertext*

xjT#9uvmY!rc$

Fredi pasti curiga!

Solusi yang kedua yaitu Alisa menyembunyikan pesan rahasia tersebut di dalam tulisan lain dengan cara menyisipkan setiap huruf pesan rahasia pada awal setiap kata seperti pada contoh alternatif 2 di bawah, Fredi yang menyampaikan pesan tentunya tidak akan curiga dan tidak menyadari keberadaan pesan rahasia di dalamnya.

Alternatif 2 : menyembunyikannya di dalam pesan lain

Lupakan Asal Rumor Itu Jangan Ambil Manfaatnya Setelah Aku Tutup Usia.

Fredi tidak akan curiga!

2.2.2 Sejarah Steganografi

Catatan pertama tentang steganografi ditulis seorang sejarawan Yunani, Herodotus. Yaitu ketika Histaeus seorang raja kejam Yunani dipenjarakan oleh raja Darius di Susa pada abad 5 Sebelum Masehi. Histeus harus mengirim pesan rahasia kepada anak laki-lakinya, Aristagoras di Militus. Histaeus menulis pesan dengan cara mentato pesan pada kulit kepala budak dan ketika rambut budak itu mulai tumbuh Histaeus mengutus budak itu ke Militus untuk mengirim pesan di kulit kepalanya tersebut kepada Aristagoras. Cerita lain tentang steganografi juga pernah dilakukan oleh Herodotus, yaitu dengan cara menulis pesan pada papan kayu yang ditutup dengan lilin. Demeratus seorang utusan yang akan

mengabarkan berita kepada Sparta bahwa Xerxes bermaksud menyerbu Yunani. Agar tidak diketahui pihak Xerxes, Demeratus menulis pesan dengan cara mengisi tabung kayu dengan lilin dan menulis pesan dengan cara mengukirnya pada bagian bawah kayu, lalu papan kayu tersebut dimaksudkan kedalam tabung kayu, kemudian tabung kayu ditutup kembali dengan lilin.

Pada abad 20, steganografi benar-benar mengalami perkembangan. Selama berlangsung perang Boer, Lord Boden Powell (pendiri gerakan kepanduan) yang bertugas untuk membuat tanda posisi sasaran dari basis artileri tentara Boer, untuk alasan keamanan, Boden Powell menggambar peta-peta posisi musuh pada sayap kupu-kupu agar gambar-gambar peta sasaran tersebut tidak diketahui oleh musuh. Perang Dunia II adalah periode pengembangan teknik-teknik baru steganografi. Pada awal perang dunia II walaupun asih digunakan teknik tinta yang tidak terlihat, namun teknik-teknik baru mulai dikembangkan seperti menulis pesan rahasia kedalam kalimat lain yang tidak berhubungan langsung dengan isi pesan rahasia tersebut. kemudian teknik menulis pesan rahasia ke dalam pita koreksi karbon mesin ketik, dan juga teknik menggunakan pin berlubang untuk menandai kalimat terpilih yang digunakan dalam pesan. Teknik terakhir adalah *microdots* yang dikembangkan oleh tentara Jerman pada akhir Perang Dunia II.

Dari contoh-contoh steganografi konvensional tersebut dapat dilihat bahwa semua teknik steganografi konvensional berusaha merahasiakan komunikasi dengan cara menyembunyikan pesan. Maka sesungguhnya prinsip dasar dalam steganografi lebih dikonsentrasikan pada kerahasiaan komunikasinya bukan pada datanya. (Munir, 2004) .

2.2.3 Manfaat Steganografi

Seperti perangkat keamanan lainnya, steganografi dapat digunakan untuk berbagai macam alasan, beberapa diantaranya untuk alasan yang baik, namun dapat juga untuk alasan yang tidak baik. Untuk tujuan *legitimasi* dapat digunakan pengamanan seperti citra dengan tanda air dengan alasan untuk perlindungan hak cipta. Digital tanda air (yang juga dikenal dengan sidik jari, yang dikhususkan untuk hal-hal menyangkut hak cipta) sangat mirip dengan steganografi karena menggunakan metode penyembunyian dalam arsip, yang muncul sebagai bagian asli dari arsip tersebut dan tidak mudah dideteksi oleh kebanyakan orang. Selain itu, steganografi juga dapat digunakan sebagai catatan tag untuk citra langsung.

Terakhir, steganografi dapat digunakan untuk melakukan perawatan atas kerahasiaan informasi yang berharga, untuk menjaga data tersebut dari kemungkinan *sabotase*, pencuri, atau dari pihak yang tidak berwenang. Sayangnya, steganografi juga dapat digunakan untuk alasan yang ilegal. Sebagai contoh, jika seseorang telah mencuri data mereka dapat menyembunyikan arsip curian tersebut ke dalam arsip lain dan mengirimkannya keluar tanpa menimbulkan kecurigaan siapapun karena tampak seperti *email* atau arsip normal. Selain itu, seseorang dengan hobi menyimpan pornografi, atau lebih parah lagi menyimpannya dalam *hard disk*. Mereka dapat menyembunyikan hobi buruknya melalui steganografi. Begitu pula dengan masalah terorisme, steganografi dapat digunakan oleh para teroris untuk menyamarkan komunikasi mereka dari pihak luar.

2.2.4 Steganografi Pada Media Digital File Gambar

Pada komputer, gambar adalah suatu *array* dari bilangan yang merepresentasikan intensitas terang pada point yang bervariasi (pixel). Pixel ini menghasilkan raster data gambar. Suatu ukuran gambar yang umum adalah 640 x 480 pixel dan 256 warna (8 bit per pixel), suatu gambar akan berisi kira-kira 300 kilobit data. Gambar digital disimpan juga secara khusus di dalam file 24-bit atau 8-bit. Gambar 24-bit menyediakan lebih banyak ruang untuk menyembunyikan informasi, bagaimanapun itu sudah sangat besar. Semua variasi warna untuk pixel yang diperoleh dari tiga warna dasar yaitu warna merah, hijau dan biru.

Setiap warna dasar direpresentasikan dengan 1 byte, gambar 24-bit menggunakan 3 byte per pixel untuk merepresentasikan suatu nilai warna. 3 byte ini dapat direpresentasikan sebagai nilai hexadecimal, decimal dan biner. Dalam banyak halaman web, warna latar belakang direpresentasikan dengan bilangan 6 digit hexadecimal, yang aktualnya tiga ikatan merepresentasikan merah,hijau dan biru. Latar belakang putih akan mempunyai nilai FFFFFF: 100% merah (FF), 100% hijau (FF) dan 100% biru (FF). Nilai decimalnya 255, 255, 255 dan binernya adalah 11111111, 11111111, 11111111 yang adalah tiga byte yang menghasilkan putih.

Definisi latar belakang putih adalah analog dengan definisi warna dari pixel tunggal dalam suatu gambar. Pixel merepresentasikan kontribusi pada ukuran file. Untuk contoh, misalkan kita mempunyai suatu gambar 24-bit luasnya 1,024 pixel dengan ketinggian 768 pixel, yang merupakan resolusi umum untuk grafik beresolusi tinggi. Suatu gambar mempunyai lebih dari dua juta pixel,

masing-masing mempunyai definisi yang akan menghasilkan suatu kelebihan file 2 Mbyte. Karena gambar 24-bit masih relatif tidak umum pada internet, ukuran seperti ini akan menarik perhatian selama *transmisi*. Kompresi file akan menguntungkan, jika tidak perlu transmisi file seperti itu. Steganografi pada media digital file gambar digunakan untuk mengeksploitasi keterbatasan kekuatan sistem penglihatan manusia dengan cara menurunkan kualitas warna pada file gambar yang belum disisipi pesan rahasia. Sehingga dengan keterbatasan tersebut manusia sulit menemukan *gradasi* penurunan kualitas warna pada file gambar yang telah disisipi pesan rahasia.

2.2.5 Kriteria Steganografi yang Baik

Penyembunyian data atau pesan rahasia ke dalam citra digital akan mengubah kualitas citra tersebut. Kriteria yang harus diperhatikan dalam penyembunyian data adalah :

1. *Fidelity*

Mutu citra penampung tidak jauh berubah. Setelah penambahan data rahasia, citra hasil steganografi masih terlihat dengan baik. Secara kasat mata seseorang tidak mengetahui kalau didalam citra tersebut terdapat data rahasia.

2. *Robustness*

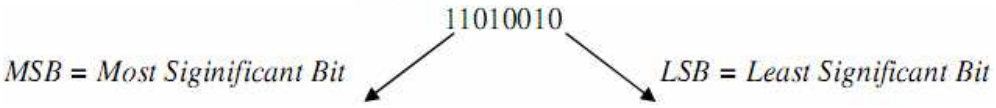
Data yang disembunyikan harus tahan terhadap manipulasi yang dilakukan pada citra penampung (seperti pengubahan kontras, penajaman, pemampatan, rotasi, perbesaran gambar, pemotongan, dan sebagainya). Bila pada citra dilakukan operasi pengolahan citra, maka data yang disembunyikan tidak rusak.

3. *Recovery*

Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah data *hidding*, maka sewaktu-waktu data rahasia di dalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut.

2.2.6 Metode LSB *(Least Significant Bit)*

Metode ini adalah bagian dari teknik *spatial domain*. Teknik ini memodifikasi langsung nilai byte dari *cover object* (nilai byte dapat merepresentasikan intesitas / warna pixel atau *amplitude*). Metode LSB (*Least Significant Bit*) merupakan salah satu bentuk yang paling sederhana dalam steganografi. Penyembunyian data dilakukan dengan mengganti bit-bit data yang tidak terlalu berpengaruh didalam segmen citra dengan bit-bit data rahasia. Pada susunan bit didalam sebuah byte (1 *byte* = 8 bit) ada bit yang paling berarti yang disebut MSB (*Most Significant Bit*), dan ada bit yang paling kurang berarti yang disebut LSB (*Least Significant Bit*).



Jadi LSB adalah bit paling kanan. Pengubahan terhadap bit ini tidak akan berpengaruh dalam persepsi *auditori* atau *visual*. Perubahan LSB pada gambar sangat sulit untuk diketahui secara kasat mata, sehingga cocok untuk menjadi media dalam steganografi. Dengan mengubah nilai-nilai LSB gambar sesuai

dengan pesan rahasia yang diinginkan, mata manusia tidak akan dapat menemukan perbedaan antara gambar yang asli dengan yang sudah dimasukkan pesan. Pada gambar bitmap 24-bit, tiap pixel-nya mengandung 24-bit kandungan warna atau 8-bit untuk masing-masing warna dasar (R, G, dan B), dengan kisaran nilai kandungan antara 0 (00000000) sampai 255 (11111111) untuk tiap warna. Perubahan LSB ini pada gambar jenis ini hanya akan merubah 1 nilai dari 256 nilai, sehingga gambar hasil steganografi akan sulit dibedakan dengan gambar yang asli.

Sebagai contoh pada penyisipan pesan pada file gambar 24-bit. Gambar 24-bit berarti pada tiap pixelnya mengandung 24-bit informasi warna. Pixel mengandung 3 X 3byte terdiri dari 1 byte warna merah, 1 byte warna biru dan 1 byte sisanya warna hijau. Berikut ini adalah contoh penyisipan pada sebuah pixel berwarna merah :

1. Data yang akan disisipkan misalnya adalah bit 010.
2. Akan disisipkan pada sebuah pixel berwarna merah yang bernilai 00110011 – 10100010 -11100010
3. Ganti tiap LSB dengan bit pada penyisipan pesan.
4. Hasilnya adalah pixel yang bernilai 0011001<u>0</u> – 1010001<u>1</u> - 11100010.

Bit yang cocok untuk diganti adalah bit LSB, hanya mengubah nilai byte satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan byte tersebut menyatakan warna merah, maka perubahan satu bit LSB tidak mengubah warna merah tersebut secara berarti. Lagi pula, indera penglihatan manusia tidak dapat membedakan perubahan yang kecil. Keuntungan

menggunakan algoritma LSB adalah mudah diimplementasikan dan proses *encoding* cepat. Sedangkan kelemahan dari algoritma LSB adalah tidak tahan terhadap pengubahan (*modifikasi*) terhadap *cover object*.

Ukuran maksimal dari pesan rahasia yang dapat ditampung pada citra dgital yang akan dimanipulasi tergantung dari resolusi citra digital yang dinyatakan dalam bentuk pixel, dan juga panjang string. Dimana setiap 3 pixel dapat mewakilkan 1 karakter untuk disimpan dalam citra digital. Untuk lebih jelasnya lihat perbandingan antara resolusi citra digital dengan karakter yang dapat ditampung, ditunjukkan pada Tabel 2.1.

Tabel 2.1 Maksimal jumlah karakter

Resolusi (pixel)	Jumlah karakter
512 x 512	98.112
600 x 400	89.775
800 x 600	179. 700
1024 x 768	294.528

Dapat disimpulkan, semakin besar resolusi citra maka semakin banyak jumlah karakter yang dapat ditampung untuk disembunyikan. Hal ini dikarenakan semakin besar resolusi media citra digital maka semakin banyak pixel yang dapat dimanipulasi.

2.3 Pengolahan Citra

Citra digital adalah gambar dua dimensi yang dapat ditampilkan pada layar monitor komputer sebagai himpunan berhingga (*diskrit*) nilai digital yang disebut pixel (*picture elements*). Pixel adalah elemen citra yang memiliki nilai yang menunjukkan intensitas warna.

Berdasarkan cara penyimpanan atau pembentukannya, citra digital dapat dibagi menjadi dua jenis. Jenis pertama adalah citra digital yang dibentuk oleh kumpulan pixel dalam *array* dua dimensi. Citra jenis ini disebut citra *bitmap* (*bitmap image*) atau citra raster (*raster image*). Jenis citra yang kedua adalah citra yang dibentuk oleh fungsi-fungsi geometri dan matematika. Jenis citra ini disebut grafik vektor (*vector graphics*). Dalam pembahasan ini, yang dimaksud citra digital adalah citra *bitmap*. Citra digital (*diskrit*) dihasilkan dari citra analog (*kontinu*) melalui digitalisasi. Digitalisasi citra analog terdiri atas penerokan (*sampling*) dan kuantisasi (*quantization*). Penerokan adalah pembagian citra ke dalam elemen-elemen *diskrit* (pixel), sedangkan kuantisasi adalah pemberian nilai intensitas warna pada setiap pixel dengan nilai yang berupa bilangan. Banyaknya nilai yang dapat digunakan dalam kuantisasi citra bergantung kepada kedalaman pixel, yaitu banyaknya bit yang digunakan untuk merepresentasikan intensitas warna pixel. Kedalaman pixel sering disebut juga kedalaman warna. Citra digital yang memiliki kedalaman pixel *n* bit disebut juga citra *n*-bit. Berdasarkan warna-warna penyusunnya, citra digital dapat dibagi menjadi tiga macam. Yaitu citra biner (*monokrom*), citra skala keabuan (*grayscale*) dan citra berwarna (*true color*).

2.3.1 Citra Biner (*Monokrom*)

Citra biner memiliki nilai pada setiap titik bernilai 0 atau 1, masing-masing merepresentasikan warna tertentu. Contoh yang paling lazim yaitu warna hitam bernilai 0 dan warna putih bernilai 1. Setiap titik pada citra hanya membutuhkan 1 bit, sehingga setiap byte dapat menampung informasi 8 titik. Standar citra yang digunakan untuk tampilan pada layar komputer yaitu nilai biner berhubungan dengan ada tidaknya cahaya yang terdapat di dalam monitor komputer. Angka 0 menunjukkan tidak ada cahaya, sehingga warna yang direpresentasikan adalah hitam sedangkan angka 1 terdapat cahaya, sehingga warna yang direpresentasikan adalah putih.

2.3.2 Citra Skala Keabuan (*Gray Scale*)

Citra skala keabuan memberi kemungkinan warna yang lebih banyak daripada citra biner, karena ada nilai-nilai lain di antara nilai minimum dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Contoh untuk skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$, dan nilai maksimumnya adalah $2^4-1 = 15$, sedangkan untuk skala keabuan 8 bit maka jumlah kemungkinan nilainya adalah $2^8 = 256$ dan nilai maksimumnya adalah $2^8-1 = 255$. Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara hitam sebagai warna minimal dan warna putih sebagai warna maksimal.

2.3.3 Citra Warna (*True Color*)

Pada citra warna, setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari 3 warna dasar, yaitu: merah, hijau dan biru. Format citra ini sering disebut sebagai citra RGB (*red*, *green*, *blue*). Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit), misalnya warna kuning merupakan kombinasi warna merah dan hijau sehinga nilai RGB adalah 255 2550, sedangkan warna ungu muda nilai RGB-nya adalah 150 0 150, dengan demikian setiap titik pada citra warna membutuhkan data 3 byte.

Jumlah kombinasi warna yang mungkin untuk format citra ini adalah 2^{24} atau lebih dari 16 juta titik warna, oleh sebab itu format ini dinamakan *true color*. Cara penyimpanan elemen RGB untuk setiap piksel dalam memori *bitmap* adalah dengan format *little-endian*, sehingga byte paling rendah berisi nilai elemen B, diikuti nilai elemen G dan byte paling tinggi berisi nilai elemen R.

2.4 Format Citra Digital

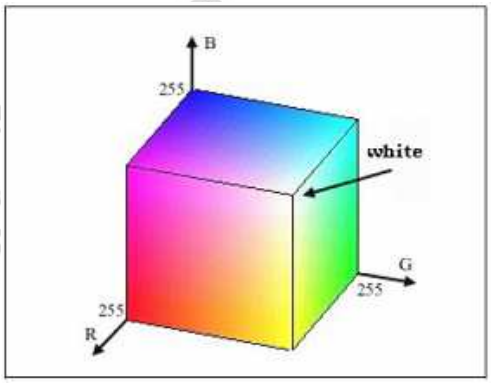
Citra digital dapat disimpan dalam berbagai macam format. Beberapa format citra digital dapat memanfaatkan metode kompresi dalam penyimpanan data citra. Kompresi yang dilakukan dapat bersifat *lossy* maupun *lossless*, bergantung kepada jenis format yang digunakan. Kompresi yang bersifat *lossy* menyebabkan penurunan kualitas citra, meskipun dalam beberapa kasus penurunan kualitas tersebut tidak dapat dikenali oleh mata manusia. Beberapa format citra digital yang banyak ditemui adalah BMP, JPEG, GIF, PNG, dan lain-lain. Pada penelitian ini format yang digunakan adalah BMP atau *bitmap*.

2.4.1 *Bitmap*

Kriteria yang paling penting dari citra ini adalah kedalaman warna yaitu berapa banyak bit per pixel yang didefinisikan dari sebuah warna. *Bitmap* dengan mengikuti kriteria tadi maka dapat dilihat :

a. 8 bit = 256 warna (*gray scale*)

b. 24 bit = 16.777.216 warna (*true color*)



Gambar 2.3 Warna *bitmap*.

Secara umum dapat dikatakan semakin banyaknya warna, maka akan diperlukan keamanan yang ketat atau tinggi dikarenakan *bitmap* memiliki area yang sangat luas dalam sebuah warna yang seharusnya dihindarkan. Dilihat dari kedalaman atau kejelasan dari sebuah warna, *bitmap* dapat mengambil sejumlah data tersembunyi dengan perbandingan sebagai berikut (ukuran *ratio* dari *bitmap* dalam byte = ukuran dari data yang disembunyikan) :

a. 8 bit = 256 warna : 8 : 1

b. 24 bit = 16.777.216 warna : 8 : 1

Perbandingan tersebut diperoleh dari penentuan LSB dalam suatu byte, untuk citra 8 bit letak LSB adalah pada bit terakhir sedangkan untuk citra 24 bit

letak LSB adalah pada bit ke-8, bit ke-16 dan bit ke-24 dimana masing-masing byte mewakili warna merah (*red*), warna hijau (*green*) dan warna biru (*blue*).

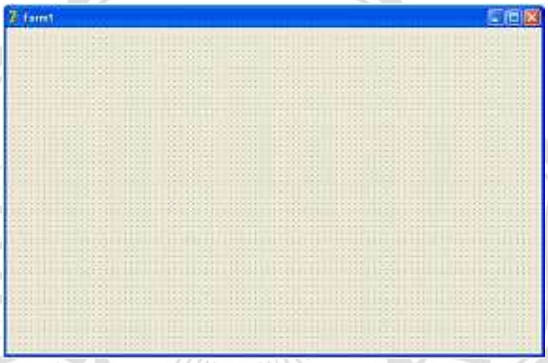
2.5 Borland Delphi 7

Borland Delphi 7 adalah sebuah aplikasi yang bekerja pada sistem operasi Windows yang berguna untuk membuat aplikasi Windows dan merupakan *compiler* dari bahasa pemrograman *Pascal*. Borland Delphi memberikan fasilitas untuk membangun aplikasi *visual*, terutama dalam hal perancangan antarmuka grafis GUI (*Graphical User Interface*).

Kelebihan Borland Delphi yaitu kecepatan pada saat kompilasi program juga menjadi faktor yang mempengaruhi pemilihan bahasa pemrograman yang digunakan. Karena program dikembangkan berdasarkan bahasa *Pascal* yang telah diakui dan telah dikenal secara luas, maka untuk pengembangan akan lebih mudah. Delphi dapat digunakan untuk membuat aplikasi dengan berbagai macam keperluan, antara lain untuk melakukan perhitungan yang sederhana, aplikasi multimedia bahkan untuk aplikasi yang terkoneksi ke internet, tetapi keunggulan dari Delphi adalah dalam pengolahan basis data.

Borland Delphi memiliki tampilan bidang kerja yang disebut IDE (*Integrated Development Environment*) yang berkualitas, yang secara garis besar terdiri atas tiga bagian utama yaitu *Window* utama, *Object Inspector* dan *Editor*. *Window* utama terdiri dari *Menu Bar, Tool Bar* dan *Component Pallete*. *Object Inspector* menyediakan dua kelompok pengaturan komponen yaitu *properties* dan *event handler*. *Editor* disediakan dua buah yaitu *form editor* dan c*ode editor*.

Berikut ini merupakan beberapa penjelasan mengenai lingkungan kerja Delphi. *Form* adalah tempat untuk melakukan perancangan dengan meletakkan komponen-komponen ke dalam *form* tersebut. Pada *form* terdapat tiga tombol yaitu *minimize, maximize* dan *close.* Terdapat juga *Title Bar* untuk menempatkan judul *form* yang nantinya akan menjadi judul dari *form* tersebut, selain itu juga terdapat *icon* untuk menjalankan aplikasi pada saat program telah dikompilasi, untuk lebih jelas tentang *Form Designer* dapat dilihat pada Gambar 2.4.

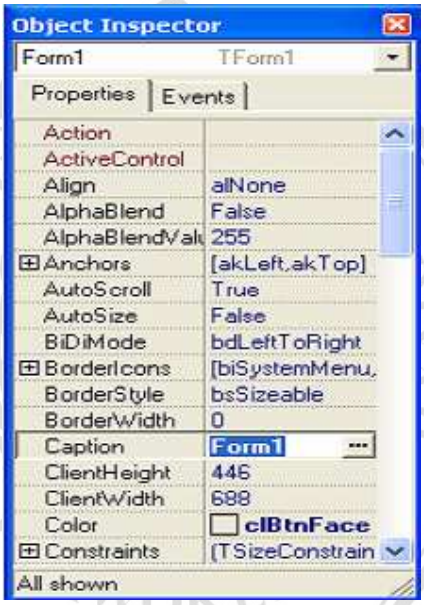


Gambar 2.4. *Form Designer*.

Component Pallete merupakan komponen-komponen yang digunakan untuk merancang program pada *form* sesuai dengan keperluan kita. Dalam komponen ini bisa ditambahkan dengan komponen yang baru misalnya dengan menambahkan komponen *Skin*, untuk lebih jelas tentang *Component Pallete* dapat dilihat pada Gambar 2.5.

Gambar 2.5 *Component Pallete*.

Object Inspector adalah sarana pengaturan objek yang telah diletakkan pada *form* atau form itu sendiri, *Object Inspector* memiliki dua halaman *Properties* dan halaman *Event*. Properties berkaitan dengan sifat komponen seperti ukuran, warna dan sebagainya. Sedangkan *event* adalah kejadian atau peristiwa yang kita inginkan, untuk lebih jelas mengenai *Object Inspector* dapat dilihat pada Gambar 2.6.

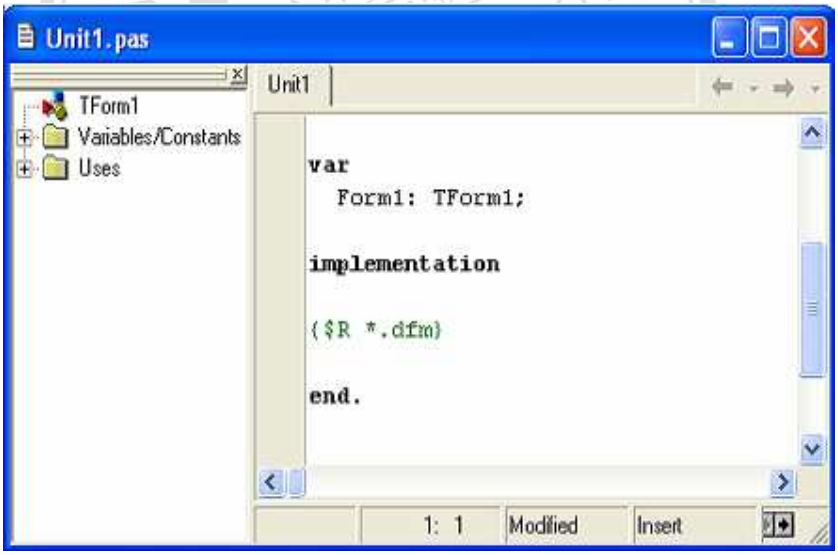


Gambar 2.6 *Object Inspector*.

Object tree view merupakan sarana untuk mengubah hubungan logis antar komponen didalam *object*, dari sini bisa dilihat bagaimana *hirarki* yang ada didalam *form* aplikasi, lebih jelas mengenai *Object Treeview* dapat dilihat pada Gambar 2.7.

Gambar 2.7 *Object Treeview*.

Fungsi *Code Editor* untuk menulis dan menyunting kode program yang akan dibuat. Salah satu kemudahan menggunakan Delphi adalah tersedianya kerangka dasar kode *prosedur*, sehingga perancang tinggal melengkapinya, untuk lebih jelas mengenai *Code Editor* dapat dilihat pada Gambar 2.8.



Gambar 2.8 *Code Editor*.

2.5.1 Komponen Borland Delphi 7

Object merupakan fasilitas yang digunakan dalam perancangan antarmuka sebuah aplikasi. Delphi menampung objek-objek tersebut ke dalam suatu paket yang diberi nama VCL (*Visual Component Library*) dan diimplementasikan dengan suatu pallete component.

Objek-objek yang disediakan oleh Delphi secara keseluruhan memiliki properties, berikut ini merupakan beberapa contoh properties dari objek tersebut.

a. *Caption* dan *text*

Caption dan text digunakan untuk memuat tulisan atau teks yang tampil dalam suatu komponen.

b. *Enable* dan *visible*

Suatu kontrol dapat disembunyikan dengan mengubah *properties visible* dengan nilai *false*, maka komponen yang telah diubah nilainya tersebut tidak akan terlihat dalam hasil aplikasi namun masih bisa diakses lewat perancangan program.

Semua aplikasi *windows* memakai *event* untuk mengelola interaksi antar program dan pemakainya, semua *Event* yang muncul ditimbulkan oleh pemakai atau oleh suatu operasi dalam sistem *windows.* Dalam Delphi, untuk merespon *event* tersebut mengaktifkan suatu kerangka *prosedur* yang disebut *prosedur* penanganan *event.* Jika tidak ada *prosedur* yang disiapkan untuk merespon *event* maka akan diabaikan, berikut ini contoh beberapa *event* yang sering digunakan dalam Delphi.

a. *Onclick*

Fungsi dari *Onclick* yaitu untuk mengeksekusi apabila tombol kiri *mouse* di tekan.

b. *OnDblClick*

Fungsi dari *OnDblClick* yaitu ketika pemakai melakukan klik tombol mouse dua kali maka suatu perintah yang ada didalamnya akan dieksekusi.

c. *OnKeyPress*

Fungsi dari *OnKeyPress* yaitu akan mengeksekusi apabila ada suatu masukan perintah dari keyboard yaitu tombol enter.

Method adalah sebuah *prosedur* atau fungsi yang menerangkan tingkah laku yang dimilikinya, berikut ini merupakan beberapa contoh *method* dalam Delphi :

a. *Refresh*

Fungsi dari *refresh* yaitu untuk memerintahkan agar suatu perintah pada objek dilakukan kembali secara otomatis.

b. *Setfocus*

Fungsi dari *Setfocus* yaitu untuk memindahkan *fokus* masukan ke kontrol tertentu. Suatu masalah yang sering terjadi adalah bahwa *method* ini akan menyebabkan *error* ketika diterapkan pada kontrol yang sedang di-*disable* atau dalam keadaan *invisible* untuk menghindari hal tersebut maka *method Setfocus* tidak digunakan pada bagian *event load* dari *form* dan perlu dilakukan pengecekan terlebih dahulu terhadap *properties enable* dan *visible* sebuah objek yang akan diberi *method* ini.

c. *Read*

Fungsi dari *Read* adalah untuk membaca data dari media penyimpanan (*storage*) tertentu seperti file.

d. *Write*

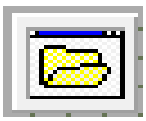
Write yaitu sebuah *method* yang digunakan untuk menulis pada media penyimpanan (*storage*).

e. *Create*

Create berfungsi untuk menyusun sebuah objek atau menginisial data sebelum objek untuk pertama kali digunakan.

2.5.2 Komponen *Open Picture Dialogs*

Fasilitas ini digunakan untuk mencari sebuah gambar pada direktori komputer. Artinya *browsing* gambar akan dilakukan ketika akan memasukan sebuah gambar didalamnya. Pengaturan ekstensi gambar yang bisa dimasukan tergantung dari property dalam komponen ini. Umumnya Delphi hanya bisa memasukan gambar dengan ekstensi BMP, tetapi bisa di masukan dengan ekstensi JPG / JPEG dan lain sebagainya.



Gambar 2.9 Komponen *open picture dialogs*.

2.5.3 Komponen *Save Picture Dialogs*

Komponen ini digunakan untuk menyimpan hasil olahan gambar. Proses ini akan terjadi ketika gambar sudah di input terlebih dahulu. Untuk itu, harus sudah ada sebuah gambar di dalam aplikasinya. Penyimpanan sebuah gambar akan berlangsung ketika dilakukan *browsing* direktori, ini dimaksudkan untuk bisa menyimpan hasil olahan gambar ke direktori yang di inginkan.

Gambar 2.10 Komponen *save picture dialogs.*

2.5.4 Kelebihan Borland Delphi 7

Berikut ini merupakan kelebihan dari Delphi 7 sebagai pertimbangan dipilihnya *Compiler* tersebut:

a. Menyediakan fasilitas yang luas mulai dari fungsi untuk membuat *form* hingga menggunakan beberapa format file basis data yang populer seperti *My SQL*, *dBase, Paradox.*

b. Dalam Delphi telah didefinisikan *template aplikasi* dan *template form* yang dapat dipakai untuk membuat semua aplikasi dengan cepat.

c. Terdapat beberapa fasilitas yang dapat diatur sesuai kebutuhan, yaitu *pallete component, editor program*, dan *template form.*

d. Delphi menghasilkan program yang terkompilasi tanpa *interpreter* dan *code* sehingga program dapat berjalan lebih cepat, karena hanya perlu menggunakan file .EXE saja tanpa harus menyertakan file .DLL.

e. Terdapat *Borland Database Engine* (BDE) yang digunakan untuk mengakses format file data dalam berbagai macam format.

2.5.5 Dasar Pemrograman Delphi

Berikut ini beberapa contoh operasi logika dalam Delphi 7 yang sering digunakan.

a. Operasi Kondisional

Perintah *if-then-else* digunakan untuk mengeksekusi sebuah blok jika memenuhi kondisi tertentu. Bila kondisi yang diseleksi terpenuhi, maka blok 35 yang mengikuti then akan diproses, sebaliknya bila kondisi tidak terpenuhi, maka akan diproses blok berikutnya, untuk penerapan *if-then-else* adalah sebagai berikut :

```
If Kondisi Then [blok perintah1]
Else [Blok perintah 2]
```

Perintah *Case Of* jika pilihan kondisi hanya sedikit, kita bisa menggunakan if tetapi jika pilihannya banyak maka kita harus menggunakan *Case-Of*. Perintah *Case-Of* digunakan untuk percabangan yang banyak, untuk contoh penerapannya adalah sebagai berikut :

```
Case variabel Kondisi Of
Case - label 1: [blok perintah1];
Case -  label 2: [blok perintah2];
…….
Case – label n: [blok perintah];
```

b. Operasi perulangan

Perintah *repeat* akan melaksanakan perulangan proses terhadap blok perintah sampai suatu keadaan dinyatakan bernilai benar (*true*), untuk contoh penerapannya adalah sebagai berikut :

```
Repeat
[blok perintah]
Until kondisi
```

Perintah *while* digunakan untuk mengeksekusi sebuah blok secara berulang selama memenuhi kondisi tertentu, untuk contoh penerapannya adalah sebagai berikut :

```
While kondisi do
begin
[blok perintah];
End;
```

Perintah *for* digunakan untuk mengeksekusi sebuah blok secara berulang dalam sebuah *range* tertentu, untuk contoh penerapannya sebagai berikut :

```
For variabel awal to akhir do
[blok perintah]
```