BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Di dalam pengolahan citra, terdapat banyak sekali cara dan metode untuk menghasilkan suatu citra agar lebih menarik. Salah satu hasil pengolahan citra tersebut adalah *autostereogram*, dimana pengolahan citra jenis ini dapat membuat citra menjadi lebih menarik. hal ini disebabkan citra jenis ini dapat membuat impresi pada otak manusia sebagai obyek 3D yang keluar dari citra 2D biasa

Citra *Autostereogram* merupakan salah satu bentuk hasil dari *imageprocessing* untuk menyembunyikan suatu obyek 3D di dalam citra 2D. Untukdapat melihatnya, dapat dilakukan dengan 2 cara, yaitu metode *cross eyed*(melihat dengan fokus mata didepan citra) atau *wall eyed* (melihat denganfokus mata dibelakang citra). Pertama kali citra ini ditemukan dandiperkenalkan oleh Charles Wheatstone pada tahun 1838. Citra-citra tersebut kelihatan sederhana, tetapi karena citra tersebut memiliki efek 3D maka citra tersebut menjadi sangat menarik.

Proses pembuatan citra *Autostereogram* terjadi dengan menggabungkan obyek yang akan ditampilkan sebagai obyek 3D dan *pattern* yang akan menjadi *background*. Pada *texture pattern* dilakukan sedikit perubahan *pixel* yang disesuaikan dengan obyek yang akan ditampilkan.

1.2 Rumusan Masalah

Berdasarkan hal di atas maka timbul suatu permasalahan, adapun perumusan masalah dari Skripsi ini adalah :

1. Bagaimana hasil dari proses *pixel* yang akan di*generate* menjadi obyek yang sama dengan obyek yang akan dimasukkan ke dalam citra *Autostereogram*.
2. Bagaimana menentukan kedalaman obyek dalam citra *Autostereogram*
3. Bagaimana hasil dari pengamatan yang dilakukan pengamat pada jarak yang berbeda

1.3 Tujuan

Tujuan dari Skripsi ini adalah untuk membuat perangkat lunak yang dapat melakukan proses penggabungan *pattern* dan citra obyek ke dalam satu citra *stereogram*. Selain itu penulis ingin mengenalkan citra *Autostereogram* kekhalayak umum.

1.4 Batasan Masalah

Sebagai batasan masalah penulisan skripsi ini, penulis memberikan batasan antara lain :

1. *Image* yang akan diproses terdiri dari *pattern* dan *image* obyek (*depth mask)* atau Image 3 Dimensi.
2. Format *image* yang di-*input*-kan adalah bitmap (*.bmp , *.jpg).
3. Hasil dari aplikasi ini akan disimpan dalam format bitmap (*.bmp , *.jpg).
4. Desain dari aplikasi ini akan menggunakan software Borland Delphi 7.
5. Image hasil tidak dapat dipisahkan lagi bila telah dieksekusi.

1.5 Metodologi Penelitian

Langkah-langkah dalam pengerjaan skripsi:

1. Studi literatur tentang:
    1.1.*Stereogram image* dan penerapannya.
2. Perencanaan dan pembuatan aplikasi
3. Pengujian dan analisa aplikasi
    3.1 Pengujian aplikasi yang telah dibuat
    3.2 Analisa hasil *output* dari aplikasi
4. Pengambilan Kesimpulan

1.6 Manfaat

Citra *Autostereogram* bermanfaat dalam berbagai bidang, mencakup bidang hiburan, eksplorasi luar angkasa serta yang paling utama adalah bidang kedokteran. Penerapan manfaat pada bidangnya antara lain :

a. Hiburan

Gambar Stereogram bisa berupa buku, poster yang dipajang diruangan atau film animasi/ citra bergerak seperti VCD. Film animasi Stereogram tipe

Anaglyph pernah popular di Indonesia beberapa tahun yang lalu, dimana untuk menontonnya harus memakai kacamata 3D.

Citra 1.1 Kacamata 3 D

b. Teknologi Eksplorasi Ruang Angkasa

Dengan stereogram para peneliti dapat memperkirakan bentuk permukaan planet lain di angkasa luar. Ini adalah contoh Anaglyph dari permukaan planet Mars yang difoto oleh NASA dengan pesawat ulang alik Mars Exploration Rovers yang diluncurkan tahun 2003.

c. Klinik dan Kedokteran

Dokter dan mahasiswa kedokteran menjumpai alat stereoskop dan anaglyph pada pelajaran Fisiologi Kedokteran. Selain itu, kartu Stereogram dan Anaglyph sering digunakan para ahli mata dan orthoptists untuk penegakan diagnosis maupun penatalaksanaan kelainan akomodasi mata.

1.7 Sistematika Penulisan

BAB I : PENDAHULUAN

Bab ini berisi tentang latar belakang, perumusan dan batasan masalahnya, tujuan skripsi, tinjauan pustaka, dan sistematika penulisan.

BAB II : DASAR TEORI

Berisi berbagai tinjauan pustaka yang berhubungan dengan algoritma untuk pembuatan citra Autostereogram, Pengolahan Citra Digital, format file BMP, format file JPEG, pengertian, sejarah, dan prinsip kerja *Autostereogram* serta teori dan aplikasi statistika dalam citra.

BAB III : METODOLOGI

Berisi perencanaan pembuatan keseluruhan sistem dalam aplikasi yang akan dibuat beserta proses implementasi sistem dari rancangan yang telah dibuat sebelumnya dengan menggunakan *Borland Delphi* 7.

BAB IV : PERANCANGAN

Menjelaskan langkah-langkah perancangan aplikasi perancangan perangkat lunak untuk menghasilkan citra *autostereogram*

BAB V : PENGUJIAN

Menjelaskan langkah-langkah pengujian dari sistem yang telah dibuat dan membahas hasil pengujiannya.

BAB VI : KESIMPULAN DAN SARAN

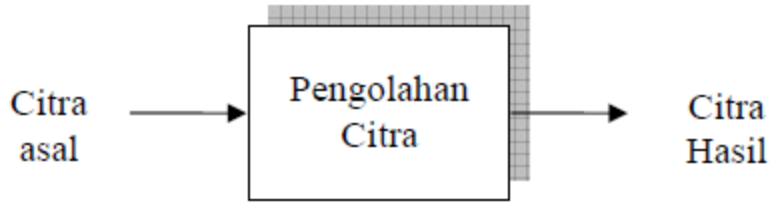
Berisi Kesimpulan dan Saran.

BAB II
TINJAUAN PUSTAKA

2.1 Pengolahan Citra

Pengolahan citra adalah pemrosesan citra, khususnya dengan menggunakan komputer, menjadi citra yang kualitasnya lebih baik. Pengolahan Citra bertujuan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin (dalam hal ini komputer).

Teknik-teknik pengolahan citra mentransformasikan citra menjadi citra lain. Jadi, masukannya adalah citra dan keluarannya juga citra, namun citra keluaran mempunyai kualitas lebih baik daripada citra masukan. Termasuk ke dalam bidang ini juga adalah pemampatan citra (*image compression*).



Citra 2.1. Pengolahan Citra

Umumnya, operasi-operasi pada pengolahan citra diterapkan pada citra bila :

1) Perbaikan atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra,
2) Elemen di dalam citra perlu dikelompokkan, dicocokkan, atau diukur,
3) Sebagian citra perlu digabung dengan bagian citra yang lain.

2.2. BMP (*Bitmap*)

BMP yang juga dikenal sebagai DIBs (*Device Independent Bitmap*) merupakan format standar sistem operasi Microsoft Windows dan IBM OS/2. Karena itu, hampir semua perangkat lunak yang bekerja di bawah Windows dan OS/2 mendukung format BMP ini.

Format ini mendukung resolusi warna mulai dari monokrom hingga *true color* (4,3 milyar warna), dan mendukung mode warna RGB, Indexed Color, Grayscale, dan Bitmap (Adobe Systems Incorporated: 2002). Kebanyakan file BMP disimpan tidak terkompresi, walaupun sebenarnya format BMP ini mendukung kompresi secara *run-length encoding* (RLE) untuk citra 4-bit dan 8-bit yang menggunakan format Windows.

Struktur sebuah file BMP terdiri dari 3 atau 4 bagian seperti pada diagram di bawah ini. Bagian pertama adalah header, diikuti dengan bagian informasi. Jika citra yang bersangkutan memiliki palet, maka bagian ketiga adalah palet warna, dan yang terakhir adalah data *pixel*.

Bitmap atau *raster* merupakan citra yang tersusun atas titik-titik elemen citra, disebut *pixel*. Masing-masing *pixel* memiliki informasi warna. Jumlah kemungkinan warna yang dapat ditampilkan oleh suatu *pixel* tergantung pada satuan *bit* yang dimiliki citra *bitmap* tersebut. Citra *bitmap* 8 *bit* berarti *pixel-pixel* yang menyusunnya dapat menampilkan kemungkinan warna sebanyak 2 pangkat 8, atau 256 warna. Citra *bitmap* dengan resolusi (jumlah *pixel* setiap satuan ukur) besar, akan terlihat lebih halus dibandingkan yang memiliki resolusi rendah. Resolusi citra *bitmap* dinyatakan dalam satuan *dot per inch* (dpi) atau *pixel per inch* (ppi).

Struktur penyimpanan pada *file* bitmap terbagi menjadi tiga bagian besar. Bagian pertama berukuran 54 *byte* terletak pada bagian awal *file* yang digunakan untuk menyimpan *header* dari *file* BMP. Bagian kedua berukuran 1024 *byte* berada setelah *header* dan digunakan untuk menyimpan informasi *pallette* yang disusun dengan susunan BGR (*blue*, *green*, dan *red*). Bagian ketiga adalah sisa *byte* dari *file* BMP yang berisi informasi citra.

Pada tabel 2.1 dapat dilihat bagaimana struktur penyimpanan file dengan format BMP. Di tabel tersebut dijelaskan data apa yang dapat diperoleh dan ukuran dari tiap bagian yang diperoleh.

Tabel 2.1. Struktur File BMP

Offset	Field	Size	Contents
0000h	Identifier	2 bytes	The characters identifying the bitmap. The following entries are possible: 'BM' - Windows 3.1x, 95, NT 'BA' - OS/2 Bitmap *Array* 'CI' - OS/2 Color Icon 'CP' - OS/2 Color Pointer

			'IC' - OS/2 Icon 'PT' - OS/2 Pointer
0002h	File Size	1 dword	Complete file size in bytes.
0006h	Reserved	1 dword	Reserved for later use
000Ah	Bitmap Data Offset	1 dword	Offset from beginning of file to the beginning of the bitmap data
000Eh	Bitmap Header Size	1 dword	Length of the Bitmap Info Header used to describe the bitmap colors, compression, The following sizes are possible: 28h - Windows 3.1x, 95, NT 0Ch - OS/2 1.x F0h - OS/2 2.x
0012h	Width	1 dword	Horizontal width of bitmap in pixels.
0016h	Height	1 dword	Vertical height of bitmap in pixels.
001Ah	Planes	1 word	Number of planes in this bitmap.
001Ch	Bits Per Pixel	1 word	Bits per pixel used to store palette entryinformation. This also identifies in an indirect way the number of possible colors. Possible values are: 1 - Monochrome bitmap 4 - 16 color bitmap 8 - 256 color bitmap 16 - 16bit (high color) bitmap 24 - 24bit (true color) bitmap 32 - 32bit (true color) bitmap
001Eh	Compression	1 dword	Compression specifications. The following values are possible: 0 - none (Also identified by BI_RGB) 1 - RLE 8-bit / pixel (Also identified by BI_RLE4) 2 - RLE 4-bit / pixel (Also identified by BI_RLE8) 3 - Bitfields (Also identified by BI_BITFIELDS)
0022h	Bitmap Data Size	1 dword	Size of the bitmap data in bytes. This number must be rounded to the next 4 byte boundary.
0026h	HResolut On	1 dword	Horizontal resolution expressed in pixel per meter.
002Ah	VResoluti On	1 dword	Vertical resolution expressed in pixels per-meter.
002Eh	Colors	1 dword	Number of colors used by this bitmap. For a 8-bit / pixel bitmap this will be 100h or 256.

0032h	Important Colors	1 dword	Number of important colors. This number Colors will be equal to the number of colors when every color is important.
0036h	Palette		N * 4 byte The palette specification. For every entry in the palette four bytes are used to describe the RGB values of the color in the following way: 1 byte for blue component 1 byte for green component 1 byte for red component 1 byte filler which is set to 0 (zero)
0436h	Bitmap Data	x bytes	Depending on the compression specifications, this field contains all the bitmap data bytes which represent indices in the color palette

2.3. JPEG (*Joint Photograpic Experts)*

Joint Photograpic Experts (JPEG) adalah suatu citra bitmap yang dikompres dan bersifat *lossy*. Tipe citra ini dirancang untuk kompresi beberapa *full-color* atau *gray-scale* dari suatu citra. JPEG mendukung 24- bit *color depth* atau sama dengan 16,7 juta warna (224 = 16.777.216 warna).

JPEG bekerja dengan merubah citra *spasial* dan merepresentasikan ke dalam pemetaan frekueunsi. Frekuensi tersebut diolah dengan menggunakan *Discrete Cosine Transform (DCT)* untuk memisahkan antara informasi frekuensi yang rendah dan tinggi dari sebuah citra. Dari proses pemisahan frekuensi tersebut, nilai yang tinggi akan diseleksi untuk dihilangkan dimana nilai tersebut tergantung dari pengaturan kualitas yang digunakan. Waktu Kompresi dan dekompresi dilaksanakan dengan simetris. JPEG sebenarnya hanyalah algoritma kompresi, bukan merupakan nama *format file*. *File* yang biasa disebut JPEG pada jaringan sebenarnya adalah JFIF (*JPEG File Interchange Format*).

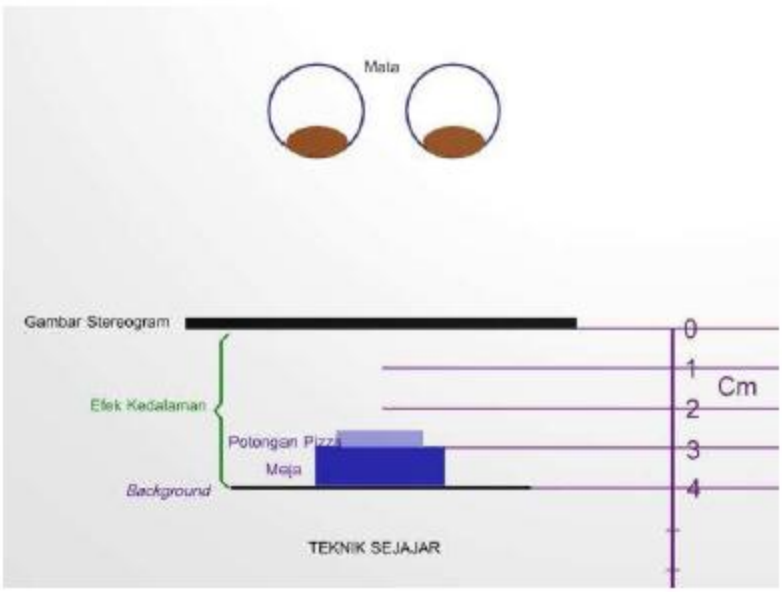
2.4. Citra *Autostereogram*

Stereogram atau *Autostereogram* adalah sebuah citra yang didesain untuk menipu otak manusia sehingga seolah-olah melihat obyek yang berupa 3D. Untuk dapat melihat obyek 3D tersebut, otak harus dapat mengkoordinasikan mata agar dapat memfokuskan pandangan pada titik focus yang sesuai dengan posisi citra yang diulang.

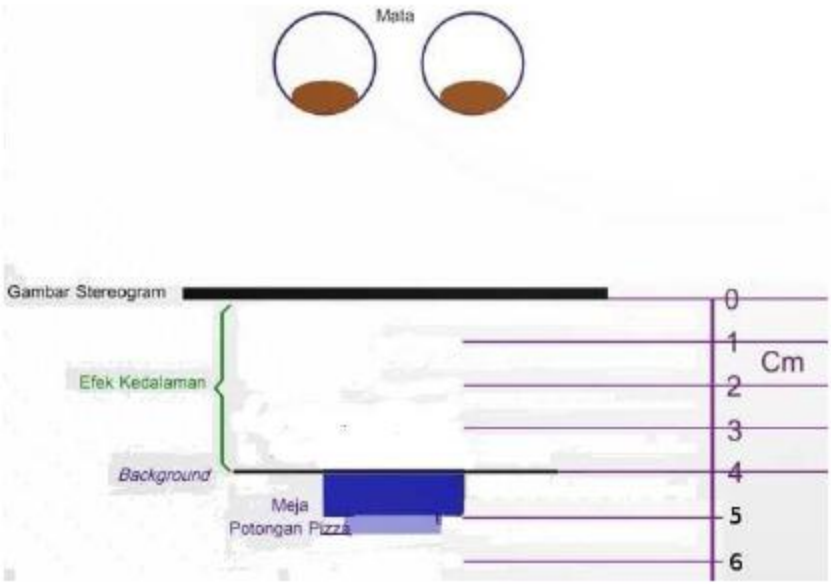
Tipe *Autostereogram* yang paling sederhana adalah *Autostereogram* yang terdiri dari *pattern* yang diulang secara horisontal dan biasa dikenal dengan nama *wallpaper stereogram*. Ketika *wallpaper* tersebut dilihat dengan titik fokus yang sesuai, pola *pattern* tersebut akan terlihat muncul ke depan citra.

Ada dua buah cara untuk melihat *Autostereogram*. Yaitu dengan cara *wall-eyed* dan *cross-eyed*. Namun kebanyakan *stereogram* dibuat agar dapat dilihat hanya dengan satu cara saja, yaitu cara *wall-eyed*. *Wall-eyed* adalah metode melihat citra *stereogram* dengan cara memandang pada obyek di belakang citra untuk mendapatkan titik fokus yang sesuai. Sehingga mata kiri hanya melihat bagian obyek memang harus dilihat dengan mata kiri saja. Begitu pula dengan mata kanan. Untuk mempermudah pemahaman, hal ini dapat dilakukan pada permukaan citra yang dapat memantulkan bayangan. Sehingga pengamat melihat pencerminan bayangan dirinya dibelakang citra.

Cara yang lain adalah dengan cara seorang pengamat meletakkan jari pada posisi diantara citra dan mata. Pengamat memfokuskan pandangan pada jari sambil memperhatikan citra *stereogram*. Bila tidak terlihat, maka pengamat perlu mengubah posisi jarinya ke epan atau ke belakang hingga fokusnya tepat. Metode ini lebih dikenal dengan nama *cross-eyed*.



Citra 2.2. Penampakan gambar *Autostereogram* dengan teknik sejajar



Citra 2.3. Penampakan gambar *Autostereogram* dengan teknik silang

2.5. Sejarah *Autostereogram*

Pada tahun 1838, seorang ilmuwan berkebangsaan Inggris bernama Charles Wheatstone mempublikasikan penjelasan mengenai *binocular vision* (penglihatan ganda yang menghasilkan persepsi bahwa obyek memiliki kedalaman) yang menuntun dia untuk membuat citra *stereoscopic* dan membuat suatu alat bernama *stereoscope* yang memungkinkan orang untuk melihat citra 3D dari citra 2D.

Pada pertengahan antara tahun 1849 dan 1850, David Brewster seorang ilmuwan dari Scotlandia mengembangkan *stereoscope* yang dibuat oleh Wheatstone dengan menggunakan lensa untuk menggantikan cermin. Brewster menyadari bahwa dengan melihat suatu pola yang berulang pada *wallpaper* dengan titik fokus di belakang *wallpaper* dapat memanipulasi kerja otak sehingga membuat otak seolah-olah melihat suatu ruang *virtual* yang terletak dibelakang *wallpaper* itu sendiri. *Stereogram* seperti ini dikenal dengan nama *single-image stereograms*.

Dr. Bela Julesz, Psikolog berkebangsaan Hungaria, tahun 1959 dengan test Stereopsis-nya, yaitu percobaan menguji kemampuan mata untuk melihat efek 3 dimensi dari citra 2 dimensi, menciptakan citra stereogram dari titik-titik acak, yang kemudian dinamakan *Random Dot Stereogram* (Stereogram Titik-Acak).

Julesz menggunakan perangkat komputer untuk membuat 2 pasang citra titik acak tersebut. Ketika dilihat dengan Stereoskop, tampaklah bentuk 3 dimensinya. Ini membuktikan bahwa persepsi kedalaman melibatkan proses sistem saraf otak yang kompleks.

Citra 2.4 Stereoskop

Pada tahun 1979, muridnya di Institut Smith-Kettlewell yaitu Dr. Christopher Tyler, mengkombinasikan antara teori *Single-Image Wallpaper Stereograms* dengan *Random Dot Stereogram*. Tyler berhasil menciptakan *Random Dot Autostereogram*, yaitu sebuah citra titik acak yang akan nampak efek 3 dimensinya tanpa bantuan peralatan apa pun. Citra ini selanjutnya juga dinamakan SIRDS (*Single Image Random Dot Stereogram*), Stereogram Titik Acak Satu Citra, masih hitam putih.

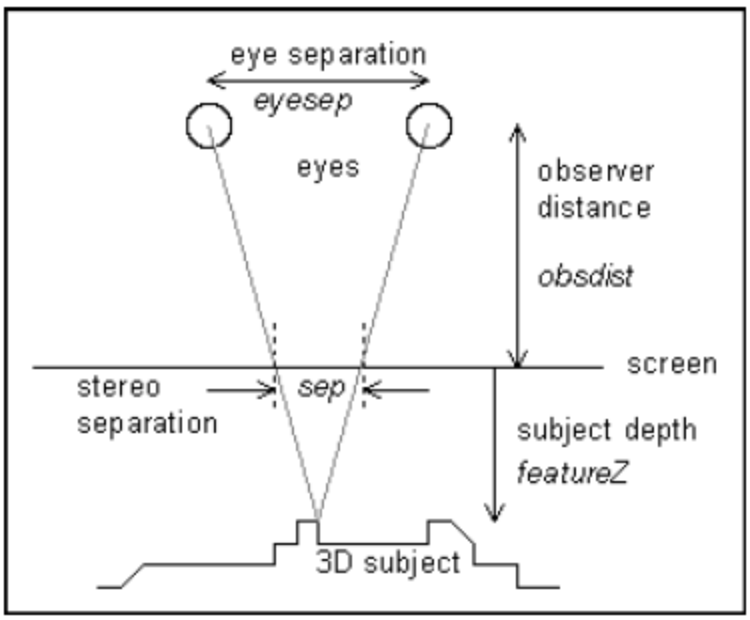
Pada tahun 1991, programmer Tom Baccei dan artis Cheri Smith berkolaborasi untuk menindak lanjuti penemuan Julesz dan Tyler. Dengan bantuan programmer Bob Salitsky, mengembangkan Setereogram dengan lebih canggih. Mereka menciptakan program stereogram berwarna (*fullcolour*).

2.6. Cara Kerja *AutoStereogram*

Stereopsis atau *stereo vision* adalah pencampuran penglihatan dari dua buah citra yang serupa namun tidak sama satu dengan yang lain yang akan menghasilkan persepsi yang memiliki kepadatan dan kedalaman (3D) di otak manusia. *Stereopsis* adalah hasil dari suatu mekanisme yang kompleks yang menghasilkan kesan 3D dengan memasangkan tiap titik (atau beberapa titik) di salah satu mata dengan titik yang sepadan di mata yang lain.

Ketika otak memperoleh *input* berupa *pattern* pada *wallpaper*, maka otak akan sedikit kesulitan untuk melihat dengan akurat pada titik focus dibelakang *pattern* tersebut. Tetapi bila hal tersebut dapat dilakukan, yaitu dengan memasangkan elemen pada *pattern* maka otak dapat memasangkan pola yang serupa seperti yang terlihat pada salah satu mata dengan elemen lain yang serupa yang terletak disebelah elemen pertama. Ini memberikan ilustrasi akan suatu obyek yang memiliki pola yang sama dengan *pattern* tersebut tetapi terletak di belakang tembok yang sebenarnya. Jarak obyek yang terbentuk tergatung pada jarak antar elemen yang identik.

Autostereogram menggunakan prinsip ini untuk menciptakan kedalaman dari suatu obyek. Jika pada suatu area dari pola pada *pattern* diulang-ulang pada jarak yang relatif dekat maka kedalaman obyek itu akan terlihat semakin jauh dengan *background*. Begitu pula sebaliknya, bila pola itu itu diulang pada jarak yang lebih jauh maka akan terlihat lebih dekat dari *background*-nya.



Citra 2.5. Struktur *AutoStereogram*

2.7. *Depth mask*

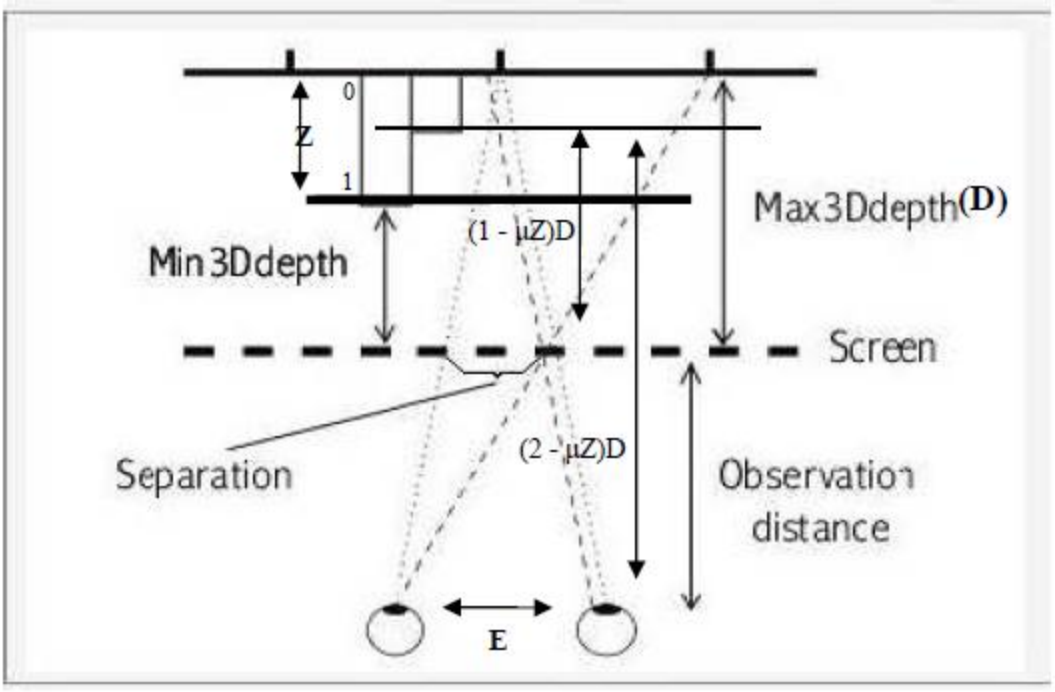
Depth mask adalah sebuah citra yang berupa *grayscale* yang mempresentasikan jarak antar *pixel* dengan menggunakan nilai *grayscale* antara hitam dan putih. Dengan menggunakan nilai pada *grayscale* ini, sebuah *grayscale depth-mask* dapat dibuat dengan warna hitam, abu-abu, dan putih yang

dipresentasikan dengan jarak 20, 10 dan 0 *pixel*. *Depth-mask* adalah sebuah kunci untuk menghasilkan suatu *Autostereogram*.

Untuk membuat *depth mask* sederhana, dapat digunakan suatu *software* pengolah citra. Pertama kali dibuat suatu obyek yang nantinya akan ditampilkan sebagai obyek 3D pada citra *Autostereogram*. Kemudian dilakukan proses pewarnaan pada obyek tersebut dimana warna dari obyek tersebut berupa *grayvalue*. Nilai dari obyek tersebut yang memiliki warna putih lebih besar akan berada lebih menonjol pada *output* citra *Autostereogram*. Hal ini berlaku juga dengan *background* dari citra *depth mask* tersebut

2.8. Algoritma Dasar Pembuatan Citra *AutoStereogram*

Algoritma dasar pembuatan gambar *Autostereogram* adalah mengambil nilai *pixel* pada tiap titik di seluruh bagian citra. Nilai *pixel* ini akan bernilai mulai dari 0 (hitam) sampai 1 (putih). Sedangkan posisi kedalaman dari obyek 3D yang akan ditampilkan diberi nilai 1/3 (μ). Kedalaman yang dimaksudkan ini adalah jarak antara *background* dan obyek 3D pada citra *stereogram*. Semakin besar nilai pada μ akan menyebabkan obyek 3D akan semakin dekat dengan *background* pada citra *stereogram*. Gambar 2.4 akan memberikan citra mengenai variabel-variabel yang akan dipakai.



Citra 2.6. Citra variabel yang dipakai posisinya

Variabel *S* pada gambar diatas menunjukkan besarnya titik separasi antar titik dari koordinat *depth mask* yang akan diolah (*Z*[x][y]). Nilai dari variabel *Z* sendiri akan berkisar antara 0 sampai 1. Dengan demikian, jarak 1- μD di depan *background Autostereogram* akan didapat dengan rumus :

$$S = \frac{1 - \mu Z}{2 - \mu Z} E$$

Keterangan Rumus :

S : jarak pergeseran *pixel*

μ : posisi kedalaman obyek yang terdekat dan yang terjauh dari mata

Z : nilai dari pixel (0 – 255)

E : jarak antara mata kiri dan kanan (dalam satuan *inchi*)

Berdasarkan rumus diatas, dapat dilihat bahwa relasi antara S dan Z adalah relasi yang merupakan dasar dari pembuatan citra *stereogram*.

Sebuah obyek yang akan ditampilkan pada suatu citra *Autostereogram* harus berada pada posisi diantara *near plane* (permukaan obyek 3D yang berada pada posisi paling dekat dengan mata) dan *far plane* (permukaan obyek 3D yang berada pada posisi paling jauh dengan mata). Obyek yang berada pada jarak paling jauh disamakan dengan posisi penglihatan mata pada posisi terjauh yang besarnya sama dengan jarak mata ke permukaan citra (D). Nilai D ini akan menjadi acuan yang cukup mudah untuk melihat citra *stereogram* karena ketika melihat citra *Autostereogram*, mata harus melihat pada titik fokus dibelakang citra.

Meskipun demikian, tidak dapat dipungkiri bahwa pembuatan dua buah titik yang nantinya akan menjadi obyek 3D justru akan semakin kacau karena tiap titik tersebuat akan menjadi acuan untuk bagian obyek 3D yang lainnya. Karena itu, untuk menghasilkan obyek 3D yang serupa dengan bentuk pada *depth mask* perlu dilakukan suatu proses yang disebut *Hidden Surface Removal.*

2.9. *Hidden Surface Removal*

Pada saat melakukan transisi di tiap titik untuk menghasilkan obyek 3D, tidak jarang permukaan pada *foreground* mengganggu penglihatan mata akan

kedalaman suatu obyek 3D. Padahal pada suatu obyek 3D, kedalaman dari obyek itu sendiri sangat berpengaruh untuk mengetahui obyek tersebut adalah obyek 3D atau bukan.

Hidden surface Removal adalah suatu masalah teknis yang menyangkut masalah detail dari obyek tersebut. Guna dari proses ini adalah untuk menampilkan bagian titik mana yang perlu ditampilkan sebagai obyek 3D dan bagian mana yang seharusnya tidak ditampilkan karena tertutup oleh obyek bagian depan.

Nilai dari suatu titik tidak perlu ditampilkan apabila *z1* > *zt*, dimana *z1* adalah koordinat titik dari bagian yang akan dikaburkan. Sedangkan *zt* adalah koordinat penglihatan dari mata sampai obyek 3D. Nilai koordinat *x* tergantung pada jarak *t*. Apabila ada suatu nilai yang mengganggu nilai dari titik yang ada di depannya ($t > 0$ dan $zt = 1$), maka titik tersebut tidak lagi perlu dilihat oleh mata. Hal itu bisa dihitung dengan persamaan :

$$\frac{1}{(Zt - Z_0)\mu D} = \frac{E/2}{(2 - \mu Z_0)D}$$

dengan demikian, persamaan untuk *zt* bisa didapat dengan cara :

$$Zt = Z_0 + \frac{2(2 - \mu . Z_0)t}{\mu E}$$

Keterangan Rumus :

Zt : titik yang dilihat oleh mata pada saat melihat obyek *stereogram*

Zo : titik dari obyek stereogram

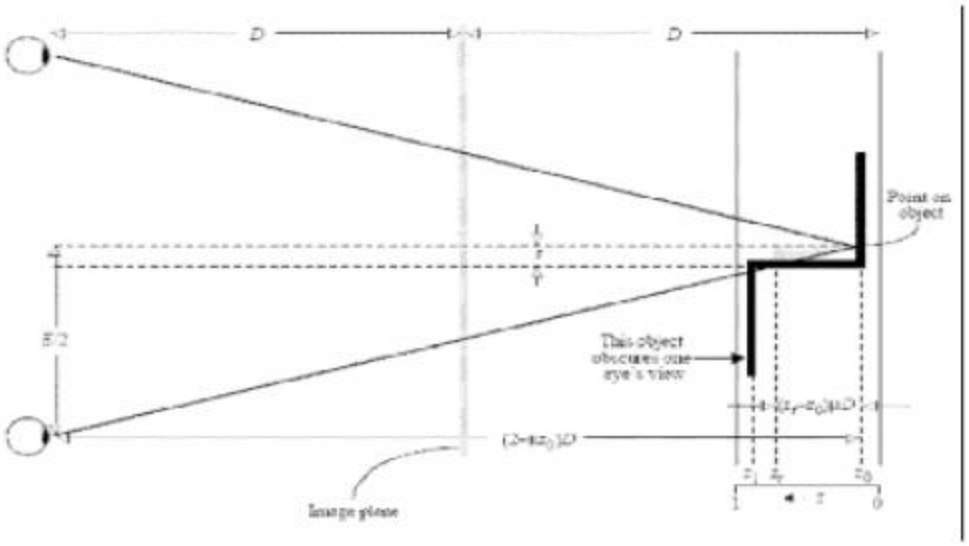
μ : posisi kedalaman obyek yang terdekat dan yang terjauh dari mata

E : jarak antara mata kiri dan kanan (dalam satuan *inchi*)

t : jarak antara titik di obyek dan garis yang dilihat mata

D : jarak antara citra dengan mata

Agar lebih jelas, gambar berikut akan memberikan visualisasi terhadap apa yang dimaksud dari rumus *hidden surface removal*.



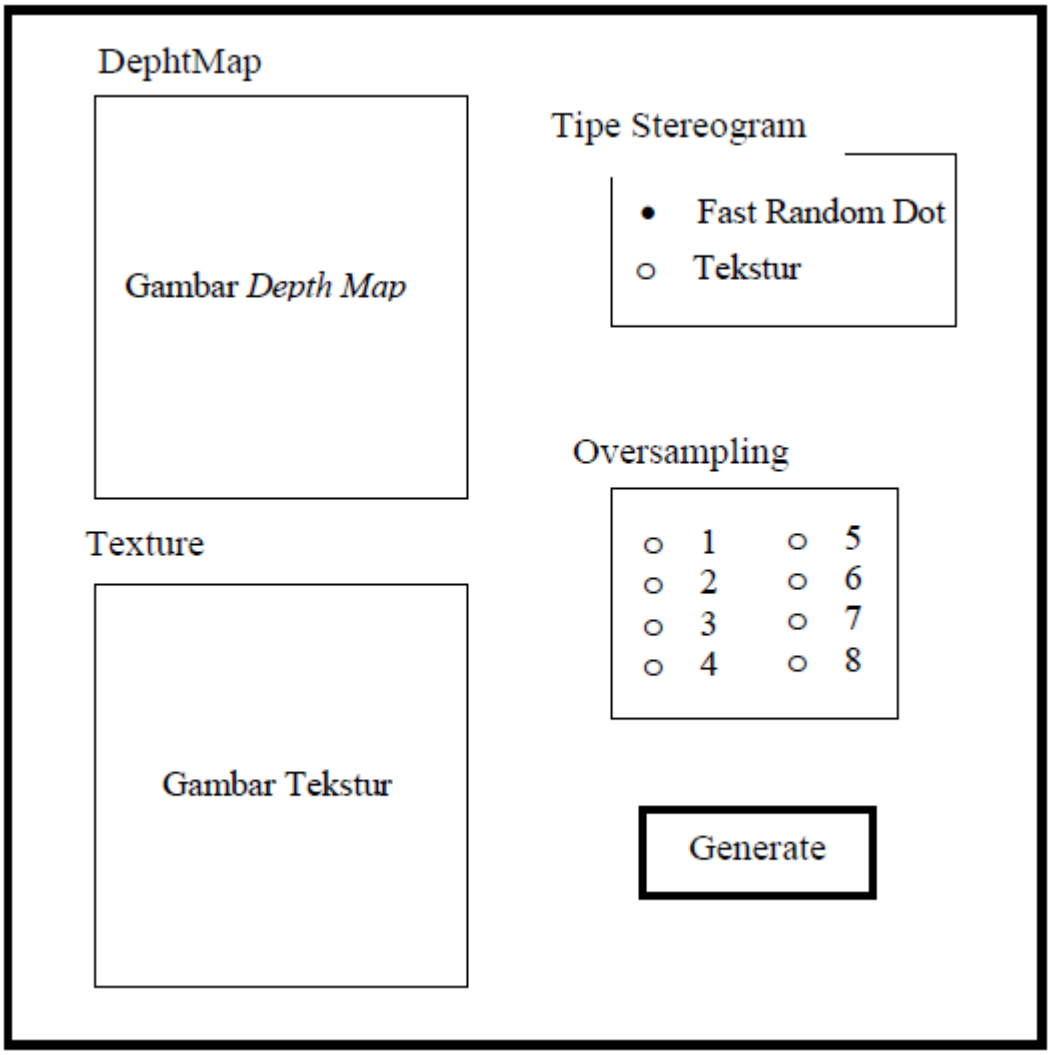
Gambar 2.7. *Hidden Surface Removal*

BAB III
METODE PENELITIAN

Dalam merancang aplikasi pada skripsi ini terlebih dahulu dilakukan pembuatan desain antar muka aplikasi dan desain proses. Desain proses berguna untuk mengintegrasikan semua proses yang terjadi dalam aplikasi yang akan dibuat. Desain data berguna untuk mengetahui data apa saja yang dibutuhkan dalam proses yang akan dikerjakan. Sedangkan perancangan antarmuka berfungsi sebagai antar muka interaksi antara pengguna dengan sistem aplikasi yang dibuat, sehingga pengguna dapat mengoperasikan aplikasi yang dibuat.

3.1. Desain Aplikasi

Pada aplikasi ini terdapat dua *form*, yaitu *form* utama yang akan digunakan untuk meng-*input*-kan citra *pattern* yang akan digunakan untuk sebagai *background* pada citra *output* dan citra *depth mask* yang akan dijadikan obyek 3D pada citra *output*. *Form* kedua adalah *form* yang hanya akan berisi citra *output* dari program yang telah diproses.



Citra 3.1 Desain Interface Aplikasi

Desain diatas merupakan interface yang muncul saat program di-*Run*. Desain tersut terdiri dari beberapa property, antara lain :

a. Image Box tempat citra depthmap & tekstur

Property ini digunakan untuk me-*load* citra DepthMap & Tekstur, dengan hanya meng-klik satu kali pada Image Box maka akan muncul jendela open untuk mengambil citra. Hanya citra dengan format BMP dan JPEG saja yang bisa di-*load* selanjutnya untuk diproses menjadi citra *Autostereogram*.

b. Tipe *Stereogram*

Tipe citra *Autostereogram* yang akan diproses dalam aplikasi ini adalah proses *Fast Random Dot* dan *Textured Autostereogram*, terdapat dua opsi dalam property ini, bila menginginkan citra hasil dengan tipe *Fast Random Dot* maka tinggal mengklik *Fast Random Dot*, sedangkan jika menginginkan citra dengan tipe *textured* tinggal mengklik opsi tekstur.

c. Oversampling

Property ini berfungsi untuk menentukan tingkat kejelasan dari Citra yang akan dihasilkan. Fungsi *oversampling* tidak berfungsi bila kita memilih tipe *Fast Random Dot Autostereogram*, hal ini dikarenakan hanya memproses dengan dua buah image yang digabungkan.

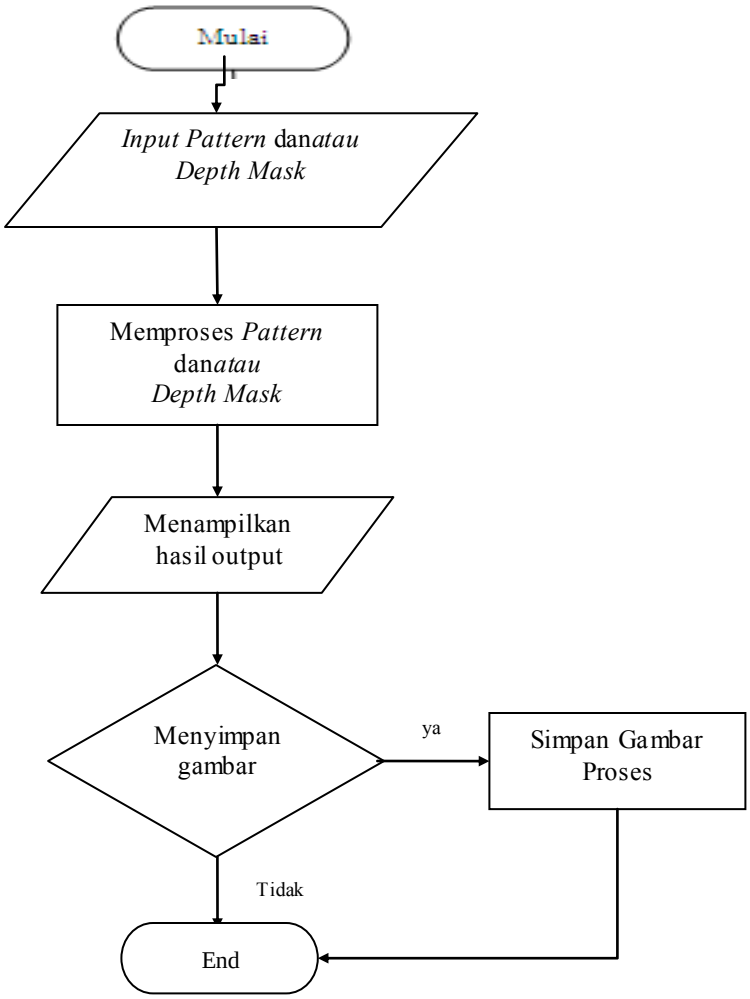
d. Generate

Proses selanjutnya adalah menekan tombol *generat,* citra pun akan keluar pada form *output*. Kitapun bisa menyimpannya dengan format BMP atau JPEG atau kembali ke form utama lagi

3.2. Deskripsi Sistem

Program *Autostereogram* ini menggunakan proses *Surface Hidden Removal,* proses ini adalah inti dari pengolahan citra *Depth Mask* & *Pattern*. Dimana 2 citra yang bertumpuk akan diseleksi bagian titik mana yang perlu ditampilkan sebagai obyek 3D dan bagian mana yang seharusnya tidak ditampilkan karena tertutup oleh obyek bagian depan.

3.2.1. Garis Besar Sistem Kerja Perangkat Lunak

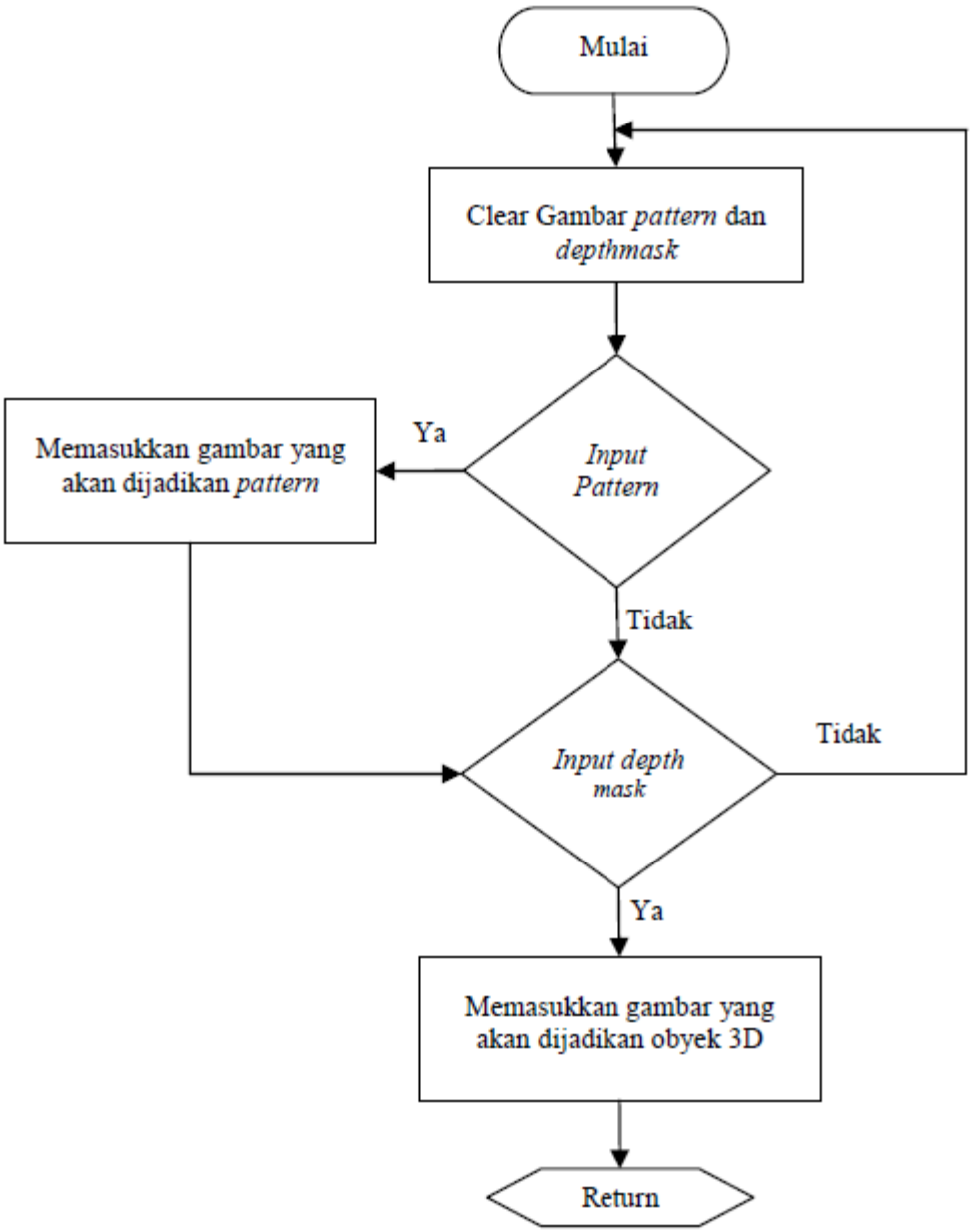


Gambar 3.2. Garis Besar Sistem Kerja Perangkat Lunak

Gambar 3.2 diatas menunjukkan garis besar sistem yang dibuat juga garis besar dari semua proses yang terjadi pada perangkat lunak yang dibuat. Di awal sistem, *user* harus meng-*input*-kan minimal hanya *depth mask*-nya saja. Bila tidak, maka program tidak akan dapat berjalan. Sistem untuk menghasilkan citra *stereogram* dengan titik *random* sebagai *backgorund* hanya akan berjalan, jika *user* telah memberi *depth mask* sebagai *input*. Sistem untuk menghasilkan citra *textured autostereogram* dengan *pattern* sebagai *backgorund* hanya berjalan jika *user* telah memberi 2 citra yang terdiri dari *pattern* dan *depth mask*. Setelah kedua *input* dimasukkan, maka program akan memproses dengan otomatis hingga *output* dihasilkan. Setelah citra *output* ditampilkan, *user* dapat menyimpan citra dalam

format BMP atau *format* JPEG. Bila *user* tidak ingin menyimpan citra, *user* dapat menutup program atau memulai proses dari awal lagi.

3.2.2. Proses Memasukkan *Pattern* dan atau *Depth Mask*



Gambar 3.3. Proses memasukkan *Depthmask* dan *pattern*

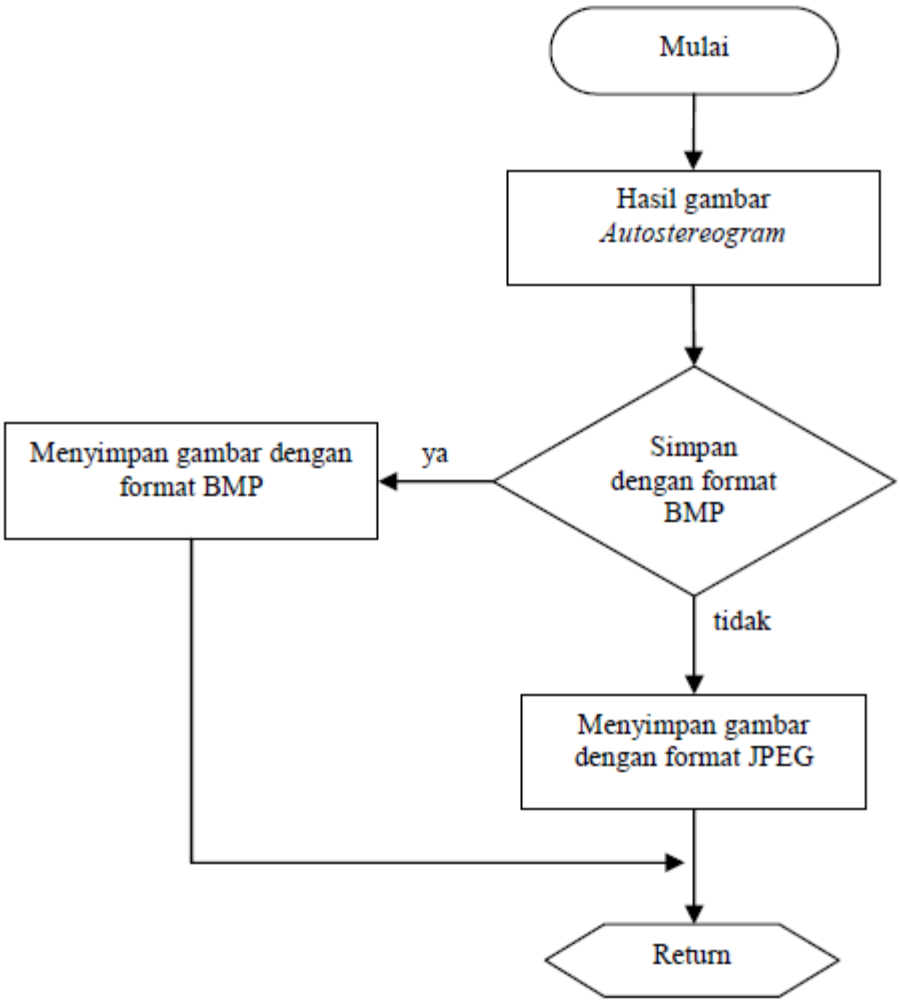
Flowchart diatas menunjukkan proses yang terjadi pada saat memasukkan *depth mask* dan atau *pattern*. *User* dapat memilih untuk memasukkan citra berupa *pattern* yang nantinya akan digunakan sebagai *background* dan pola dari obyek yang akan ditampilkan pada citra *output* atau tidak memasukkan *pattern*. Meskipun *user* tidak memasukkan *pattern*, *user* tetap harus memasukkan *depth*

mask karena *depth mask* tersebut akan digunakan untuk menampilkan obyek 3D pada citra *output*.

Di dalam keseluruhan proses tersebut, terdapat beberapa *sub*-proses untuk menghasilkan citra *Autostereogram*. *Sub*-proses tersebut adalah proses *hidden surface removal* dan pengecekan nilai *array*. Setelah proses selesai dilakukan maka dilakukan hasilnya akan terlihat di citra *output*.

Proses *Hidden Surface Removal* adalah proses yang cukup penting di sistem perangkat lunak ini. Proses ini akan mem-*filter pixel* yang seharusnya tidak ditampilkan. Di dalam proses ini terdapat perhitungan secara matematis sesuai yang telah dijelaskan di bab 2.

3.2.3 Proses Menyimpan Citra



Gambar 3.14. Proses menyimpan citra

Pada gambar proses diatas dapat dijelaskan alur penyimpanan citra *output*. Pada proses ini *user* dapat menyimpan citra yang telah diproses ke dalam format BMP atau JPEG.

3.3 Kesimpulan dan Saran

Pada tahap ini, diambil kesimpulan dari hasil pengujian dan analisa terhadap Aplikasi Perangkat Lunak Untuk Menghasilkan Citra Autostrereogram . Tahap selanjutnya adalah membuat saran untuk perbaikan terhadap penelitian selanjutnya sehingga dapat menyempurnakan kekurangan-kekurangan yang ada.

BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

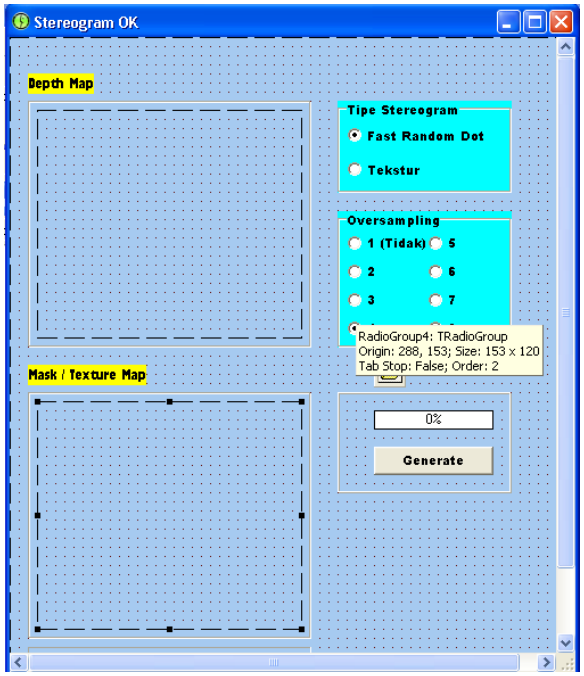
Di bab ini akan dibahas implementasi dari dasar teori dan desain sistem yang sudah dijelaskan di bab sebelumnya hingga menjadi suatu aplikasi yang siap digunakan. Implementasi yang akan dijelaskan mencakup *Delphi 7* sebagai perangkat lunak untuk membuat aplikasi ini, dan pengujian program beserta hasil *image processingnya.*.

4.1. Implementasi Sistem yang Digunakan

Pembuatan aplikasi Skripsi ini menggunakan aplikasi bahasa pemrograman *pascal* dan *Delphi 7* sebagai perangkat lunaknya. Selain itu, komponen yang digunakan untuk membuat aplikasi ini komponen *standart* dari *Delphi 7* itu sendiri.

4.1.1. Menu Utama dan Form Output

Aplikasi ini terdiri dari menu utama dan *form output*, Dalam menu utama terdapat beberapa komponen antara lain : tipe *stereogram* (*FastRandomDot* atau Tekstur), *Oversampling* ($1\ ^{s}/_{d}\ 8$) serta tombol *generate*



Gambar 4.1 Form Utama Aplikasi

Fungsi komponen-komponen aplikasi :

a. Komponen Tipe *Stereogram*

Komponen ini terdiri dari dua pilihan tipe, yatiu *FastRandomDot* dan Tekstur. Bila ingin menghasilkan citra *Autostereogram* dengan titik acak hitam putih, maka opsi *FastRandomDot* yang dipilih. *FastRandomDot* akan menghasilkan citra *Autostereogram* meskipun tanpa kita masukkan tekstur. Sedangakan bila menginginkan citra *Autostereogram* dengan latar pola atau tekstur, kita harus meng-*inputkan DepthMap* dan Tekstur. Hal ini dikarenakan tekstur dibutuhkan sebagai *background* atau latar dari *DepthMap*.

b. Komponen *Oversampling*

Fungsi komponen ini adalah sebagai parameter tingkat pergeseran *pixel* atau tingkat kejelasan citra hasil setelah diproses. Semakin tinggi nilai *oversampling*-nya, maka makin keluar citra *DepthMap* dari latar atau *background*-nya.

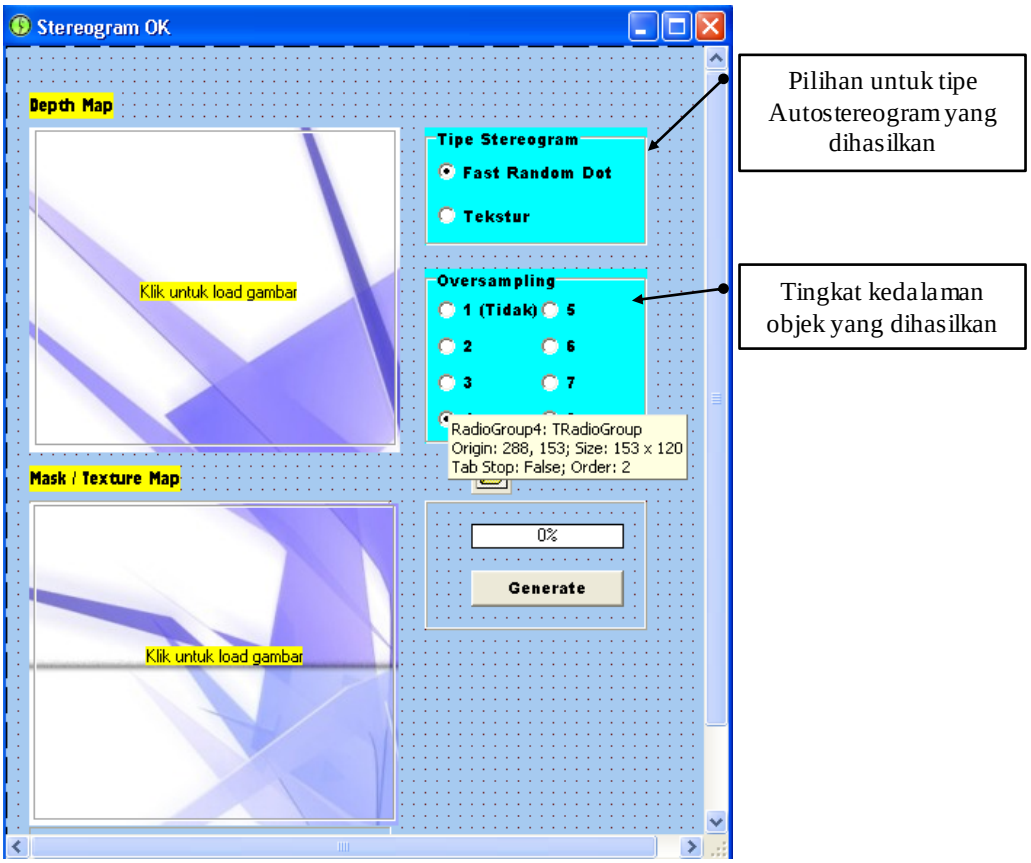
c. Komponen tombol *Generate*

Setelah parameter-parameter diatas telang di-*setting*, langkah selanjutnya adalah menekan *button Generate* dimana tombol ini akan menghasilkan citra *Autostereogram* dengan tipe yang diinginkan.

Pada *form output*, hasil dari panggabungan citra akan ditampilkan. Kita bisa menyimpan dalam bentuk *file* JPEG ataupun BMP.

Gambar 4.2 *Form Output*

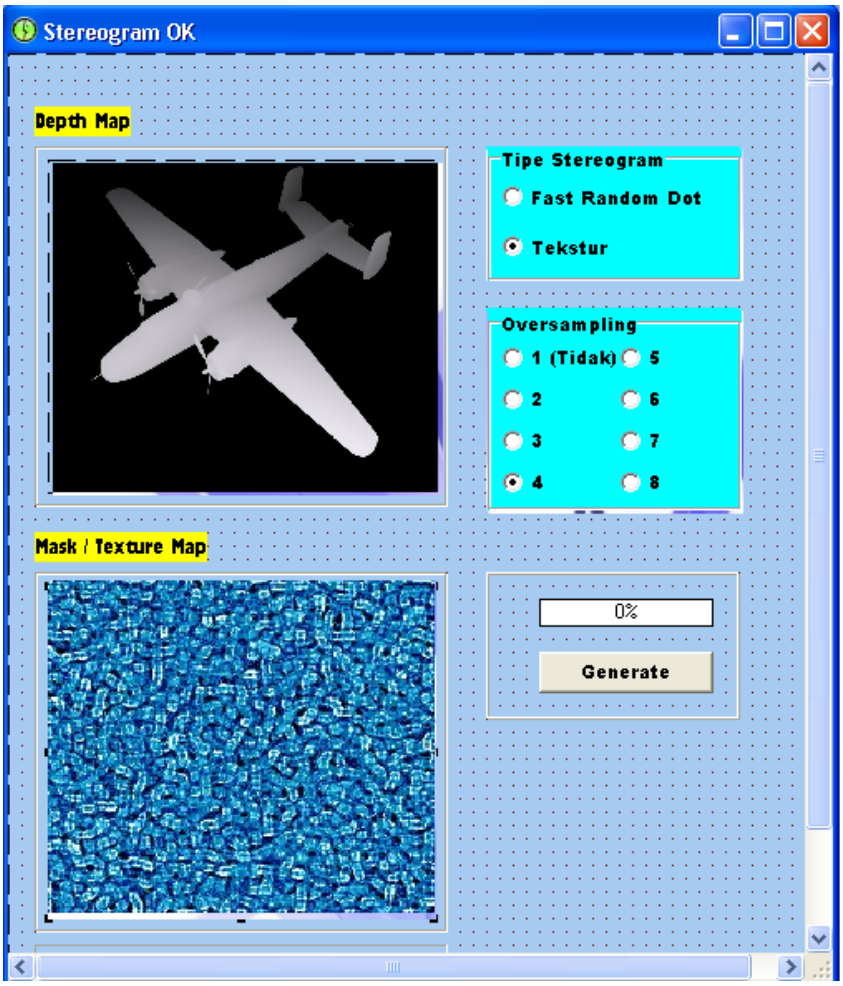
4.1.2. Cara Menjalankan Aplikasi



Gambar 4.3 Menu Utama

Langkah I :

Klik 1 kali pada panel *load* citra, baik itu depthmap saja untuk menghasilkan *Autostereogram* dengan *FastRandomDot* atau *depthmap* dan tekstur sama-sama diisi dengan *depthmap* dan tekstur sebagai *input* untuk citra *Autostereogram* dengan tekstur sebagai *background*



Gambar 4.4*Load* citra *DepthMap* dan Tekstur

Langkah II :

Tentukan pilihan tipe *Autostereogram* yang ingin dihasilkan. Klik tipe *Random Dot* untuk menhasilkan citra *Autostereogram* dengan titik acak atau klik opasi tekstur untuk menghasilkan citra *Autostereogram* dengan tekstur sebagai *background.*

Gambar 4.5 Tipe *Stereogram*

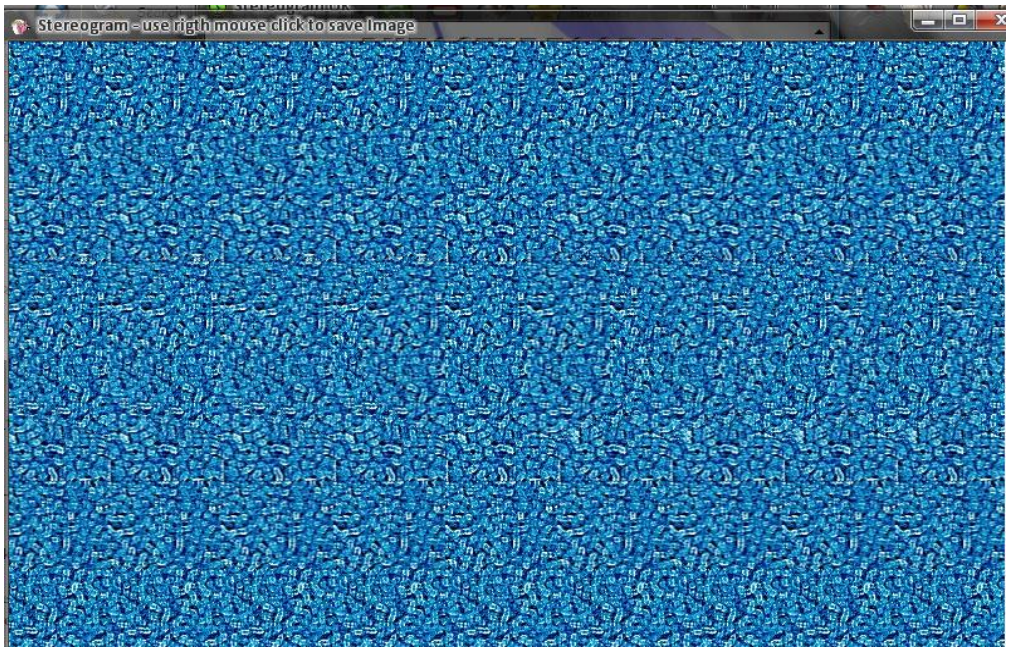
Langkah III :

Bila tipe *Stereogram* yang dipilih adalah tekstur, langkah selanjutnya adalah menentukan tingkat kedalaman citra *depthmap* yang akan dihasilkan dengan menentukan nilai oversamplingnya. Nilai standart dari *oversampling* ini adalah 4, semakin besar nilai *oversampling*-nya, maka tingkat kedalamannya akan bertambah, sehingga memudahkan pengamat untuk melihat hasil akhir tanpa harus melakukan akomodasi mata yang telalu memaksa. Nilai oversamplingtidak dapat diubah bila pengamat memilih tipe *Fast Random Dot*, nilai standartnya adalah bernilai 1 saja.

Gambar 4.6 *Oversampling*

Langkah IV :

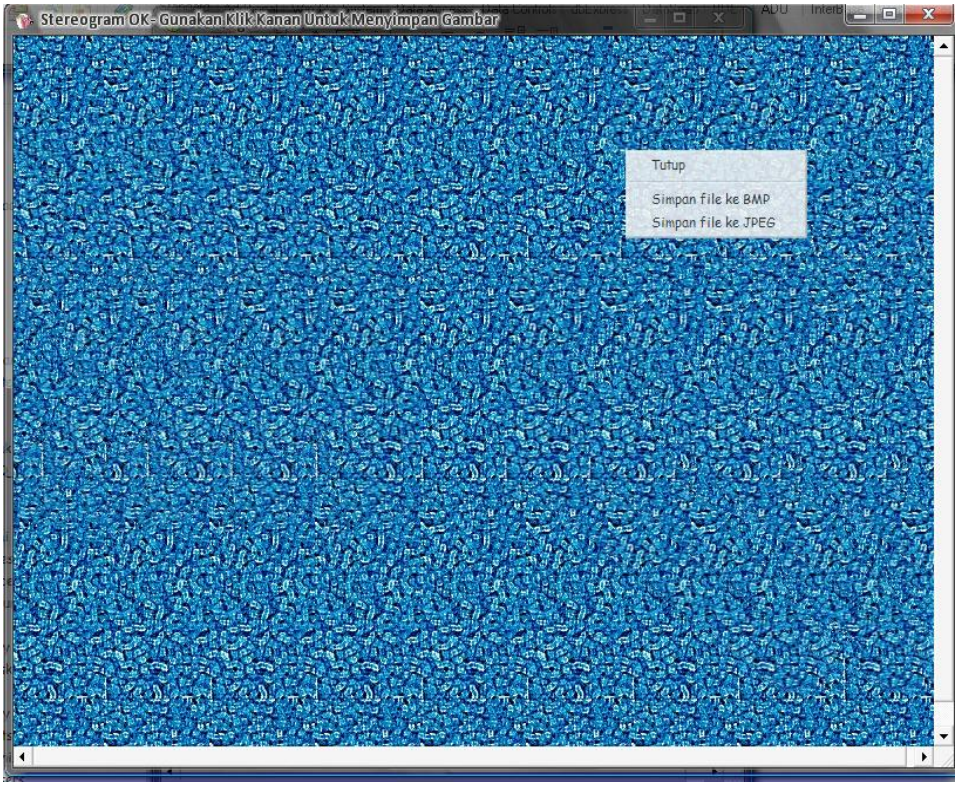
Setelah semua langkah diatas dilakukan, maka langkah terakhir adalah mengeksekusinya dengan menekan button generate, akan muncul form *output* hasil dari *Autostereogram* dengan resolusi 800 *x* 500 *pixel.*



Gambar 4.7 *Form Output Autostereogram*

Langkah V :

Bila ingin menyimpan dalam bentuk file BMP, klik kanan pada *form* dan pilih opsi “Simpan ke File BMP”. Jika mengiinginkan menyimpan dalam bentuk file JPEG, klik kanan pada form dan pilih opsi “Simpan ke File JPEG”. Atau pilih opsi “Tutup” bila tidak ingin menyimpan atau kembali ke menu utama.



Gambar 4.8 *Form output* untuk menyimpan citra

4.2. Pengujian dan Analisa Sistem

Pengujian dilakukan pada komputer dengan Processor Intel Dual Core 1,8 Ghz, VGA 128 Mb dan Memory 1 GB. Pengujian ini menggunakan aplikasi *Autostereogram* pada operasi sistem Microsoft Windows XP Profesional Edition 2002 Service Pack 2.

Pada aplikasi perangkat lunak yang telah diimplementasikan ini akan dilakukan beberapa pengujian. Pengujian-pengujian tersebut ada beberapa kondisi, antara lain :

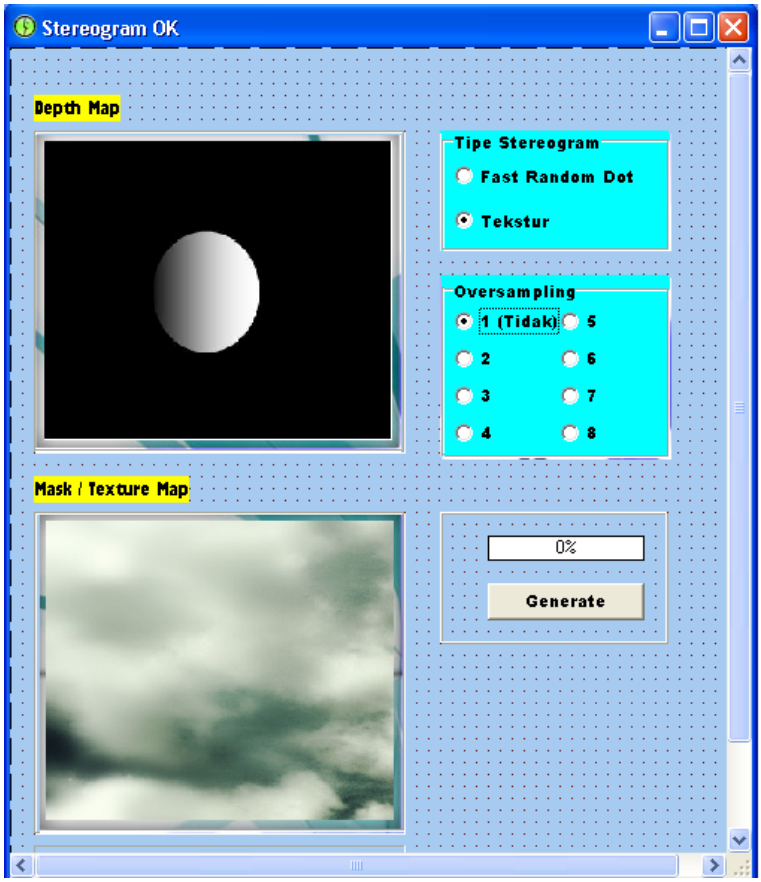
a. Pengujian untuk perbedaan bentuk dan ukuran *DepthMap*
b. Pengujian untuk perbedaan posisi objek.
c. Pengujian untuk nilai *oversampling* yang diubah-ubah.
d. Melakukan *survey* terhadap hasil *image processing*

Pada 4 poin pengujian diatas ditujukan kepada aplikasi ini dalam mengolah citra serta seberapa bagus kedalaman *image* yang dihasilkan, apakah

sudah memenuhi syarat jarak pandang pengamat dalam mengamati citra atau kah belum memenuhinya.

4.2.1. Pengujian Untuk Perbedaan Bentuk Dan Ukuran *Depthmap*

4.2.1.1. Pengujian Dengan Bulatan Putih Pada *Depthmap*



Gambar 4.9 Pengujian bulatan putih di tengah

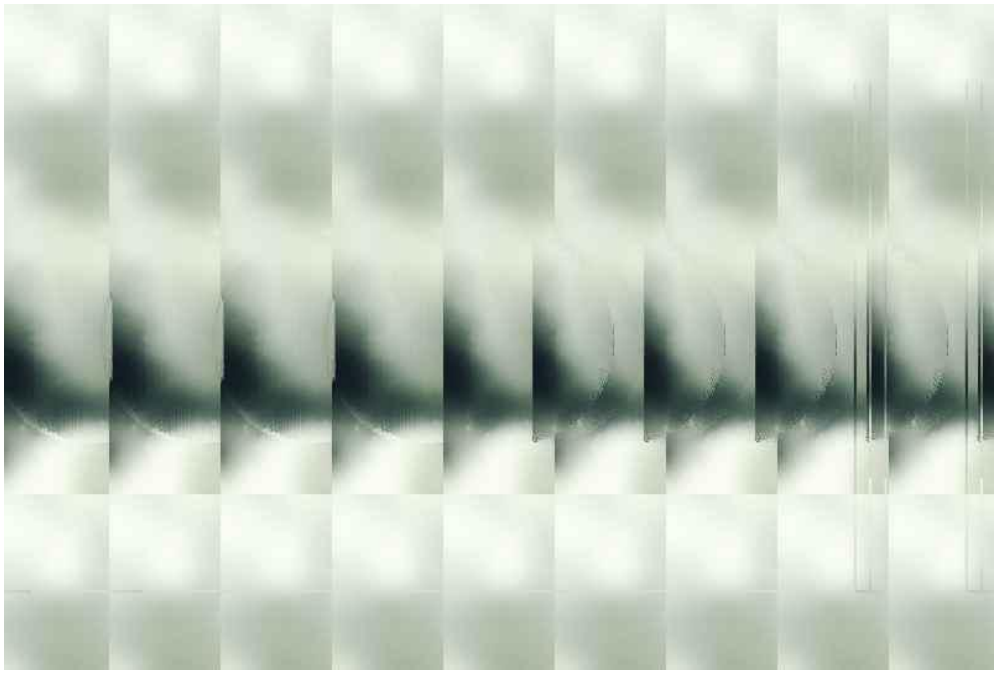
Dari Gambar 4.6 diatas kita mendapati data sebagai berikut :

Tabel 4.1 Data inputan *depthmap* bulatan putih dan hasil

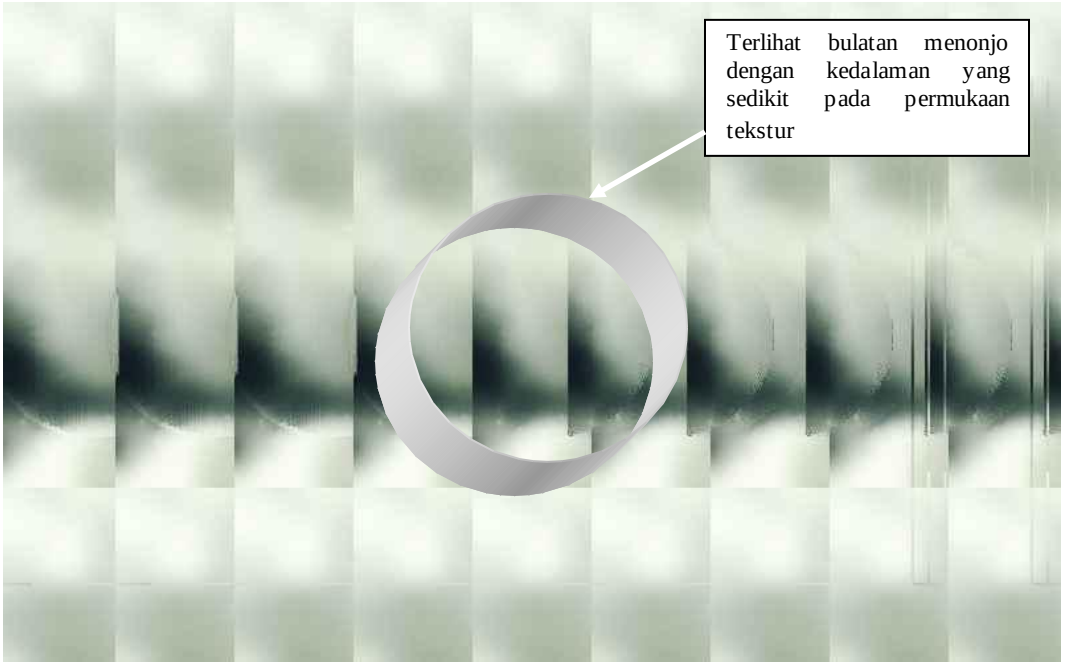
Ukuran DepthMap (Kb)	Ukuran Texture (Kb)	Tipe Stereogram	Oversampling	Ukuran Citra Output (Kb)
7,16	16,2	Tekstur	1 (Tidak)	16,2

Analisa hasil gambar :

Hasil *image processing* dari program dengan akomodasi mata maksimal adalah nampak bulatan yang memnonjol sedikit di tengah pada lapisan tekstur. Dengan jarak pandang ±20cm dari mata pengamat.

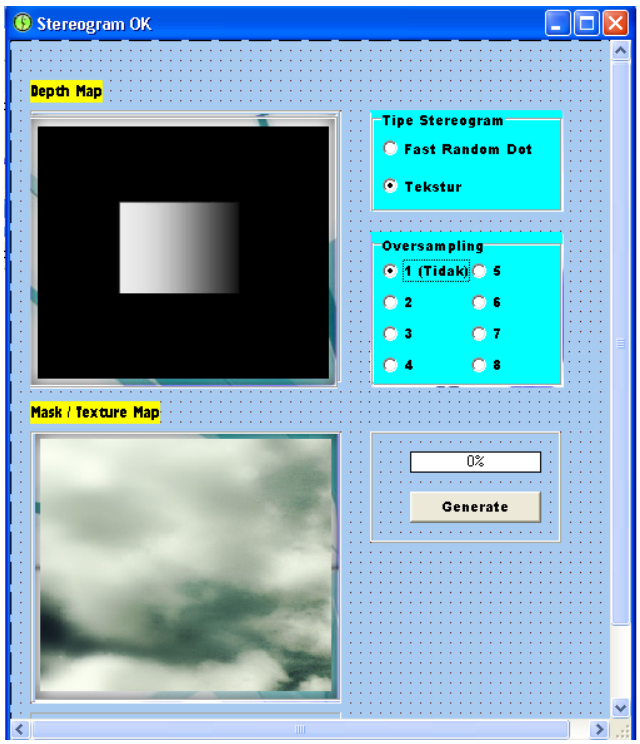


Gambar 4.10 Hasil pengujian dengan *depthmap* timbul bundaran pada permukaan tekstur



Gambar 4.11 Ilustrasi hasil pengujian dengan *depthmap* timbul bundaran pada permukaan tekstur

4.2.1.2. Pengujian Dengan Objek *Depthmap* Kotak Putih



Gambar 4.12 Pengujian kotak putih

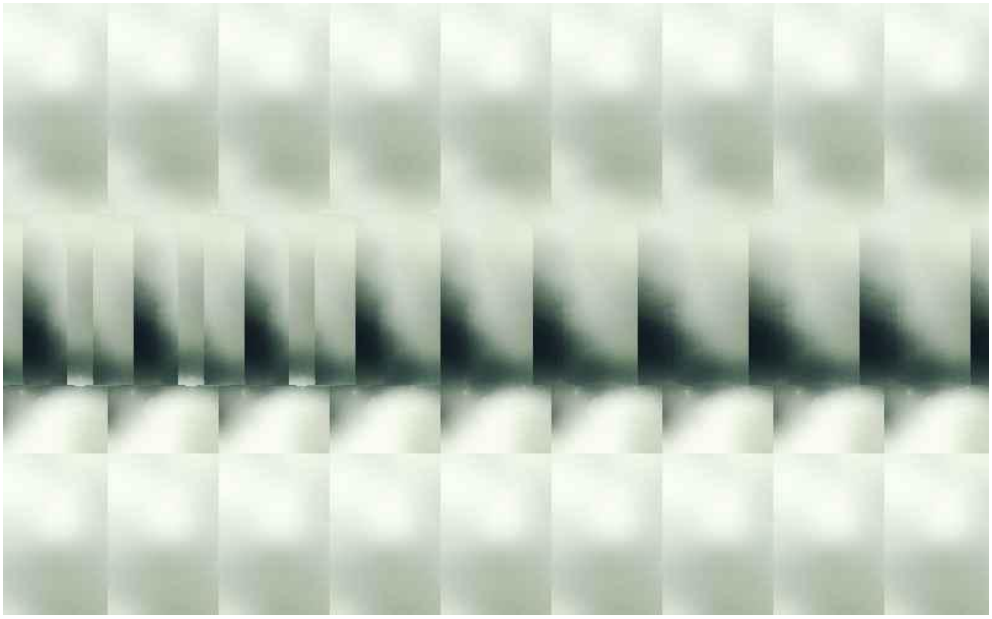
Dari gambar diatas kita mendapati data sebagai berikut :

Tabel 4.2 Data inputan depthmask kotak putih dan hasil

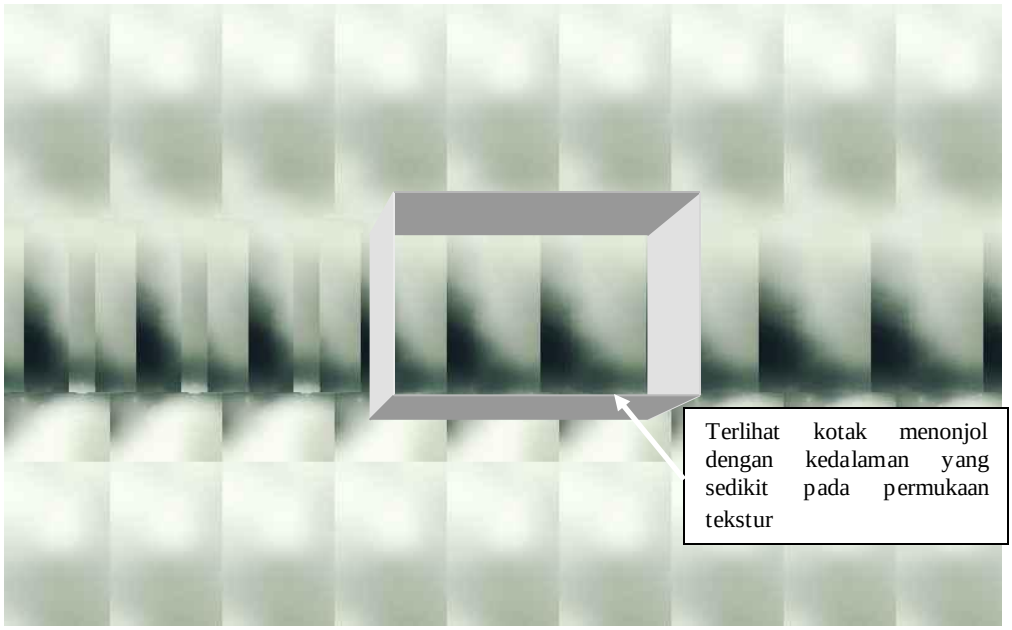
Ukuran DepthMap (Kb)	Ukuran Texture (Kb)	Tipe Stereogram	Oversampling	Ukuran Citra Output (Kb)
5,04	16,2	Tekstur	1 (Tidak)	15,4

Analisa hasil gambar :

Hasil *image processing* dari program dengan akomodasi mata maksimal adalah nampak bentuk kotak yang memnonjol sedikit di tengah pada lapisan tekstur. Dengan jarak pandang ±20cm dari mata pengamat.



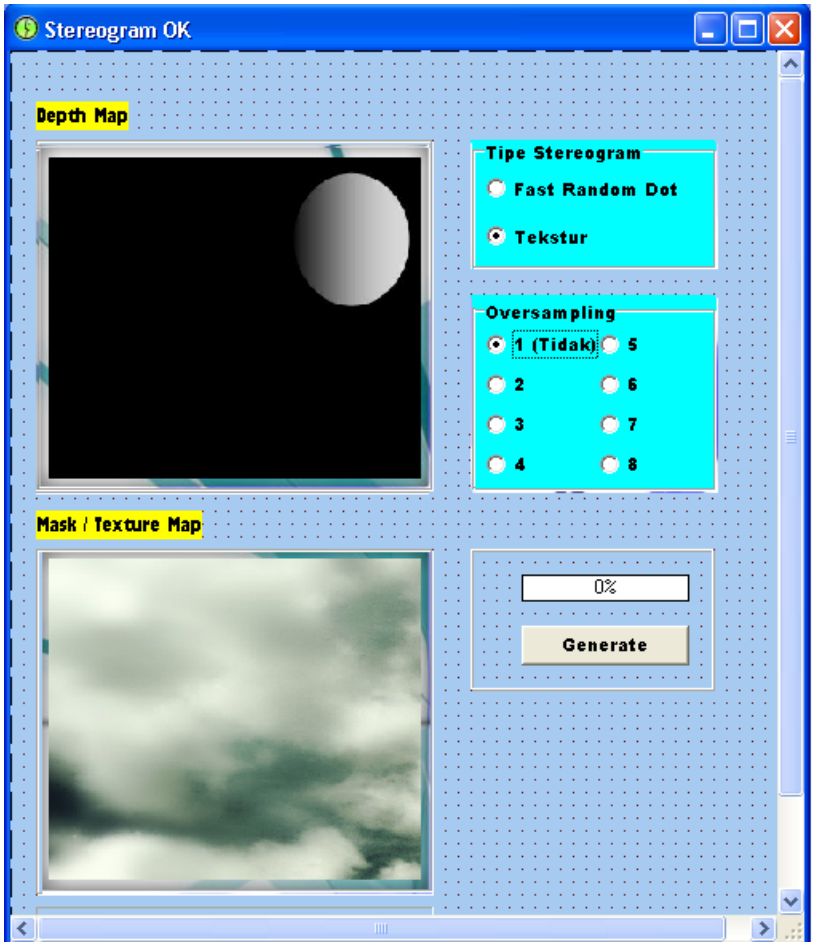
Gambar 4.13 Hasil pengujian dengan *depthmap* timbul kotak pada permukaan tekstur



Gambar 4.14 Ilustrasi hasil pengujian dengan *depthmap* timbul kotak pada permukaan tekstur

4.2.2. Pengujian Untuk Perbedaan Posisi Objek.

4.2.2.1. Posisi *Depthmap* Bulatan Putih Pada Pojok Kanan Atas



Gambar 4.15 Pengujian bulatan putih pojok kanan atas

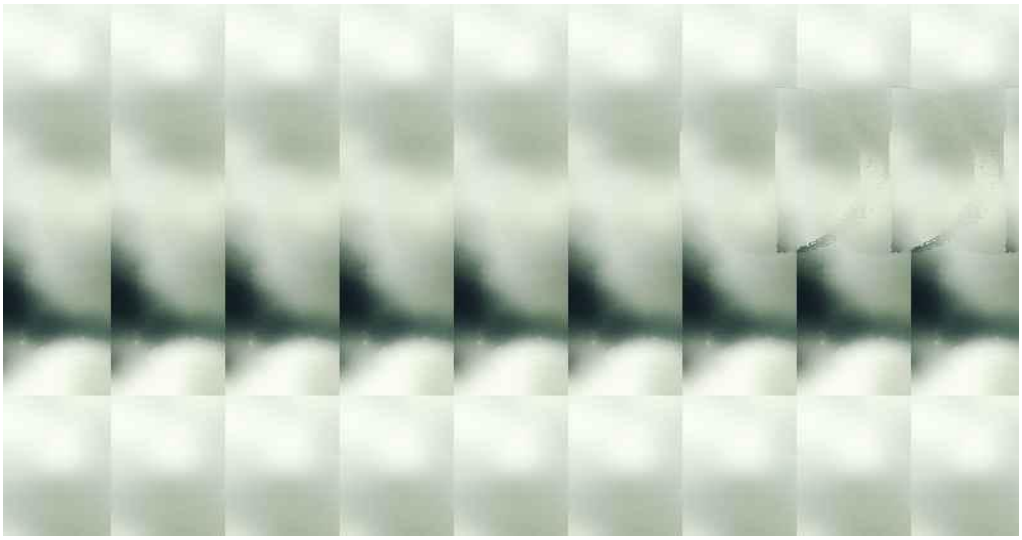
Dari gambar diatas kita mendapati data sebagai berikut :

Tabel 4.3 Data inputan *depthmap* bulatan putih pojok kanan atas dan hasil

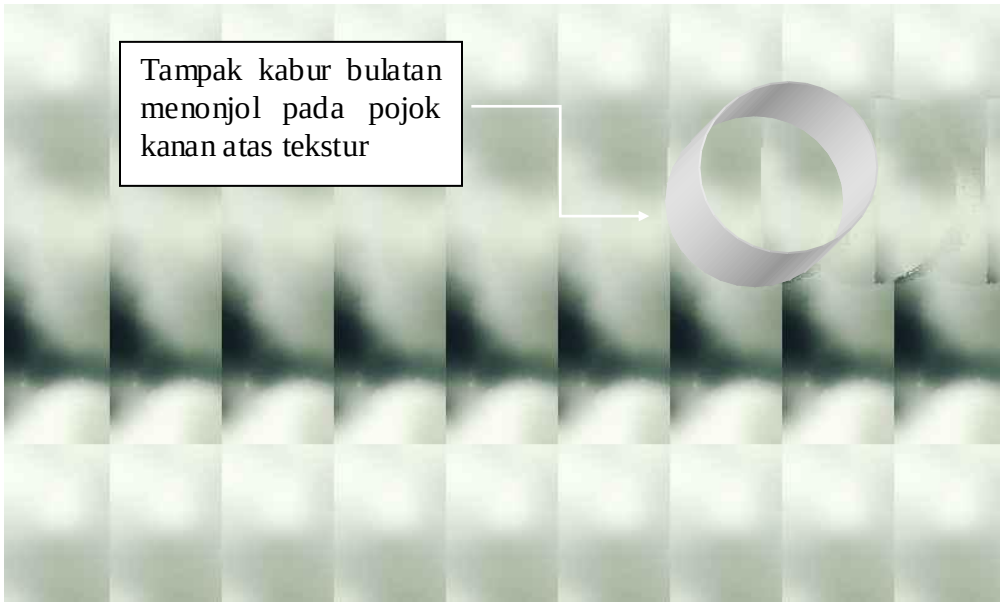
Ukuran DepthMap (Kb)	Ukuran Texture (Kb)	Tipe Stereogram	Oversampling	Ukuran Citra Output (Kb)
5,36	16,2	Tekstur	1 (Tidak)	15,1

Analisa hasil gambar :

Hasil *image processing* dari program dengan akomodasi mata maksimal adalah nampak bulatan yang memnonjol kabur di pojok kanan atas pada lapisan tekstur. Dengan jarak pandang ±20cm dari mata pengamat, objek sulit kita lihat karena posisinya yang berada dipojok dari tekstur. Hal ini dikarenakan focus mata yang umumnya terfokus pada bagian bagian tengah dari tekstur.

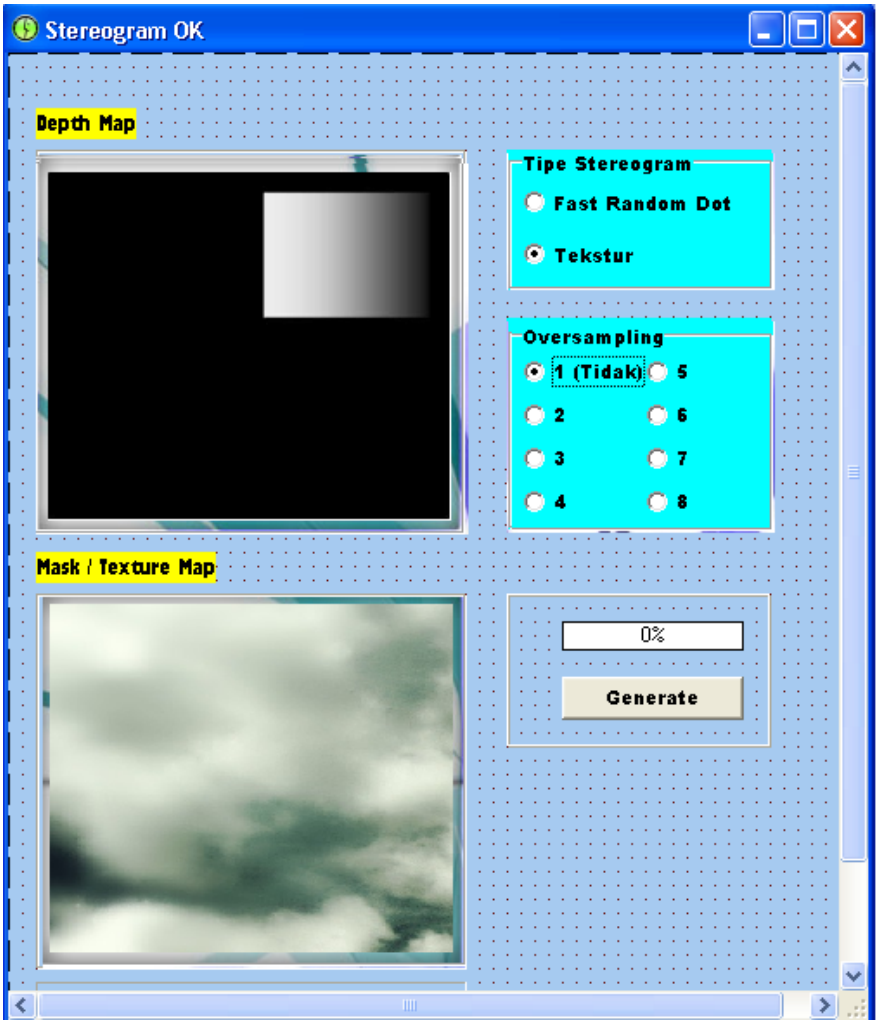


Gambar 4.16 Hasil pengujian dengan *depthmap* timbul bundaran pojok kanan atas pada permukaan tekstur



Gambar 4.17 Ilustrasi hasil pengujian dengan *depthmap* timbul bundaran pojok kanan atas pada permukaan tekstur

4.2.2.2. Posisi *Depthmap* Kotak Putih Pada Pojok Kanan Atas



Gambar 4.18 Pengujian kotak putih pojok kanan atas

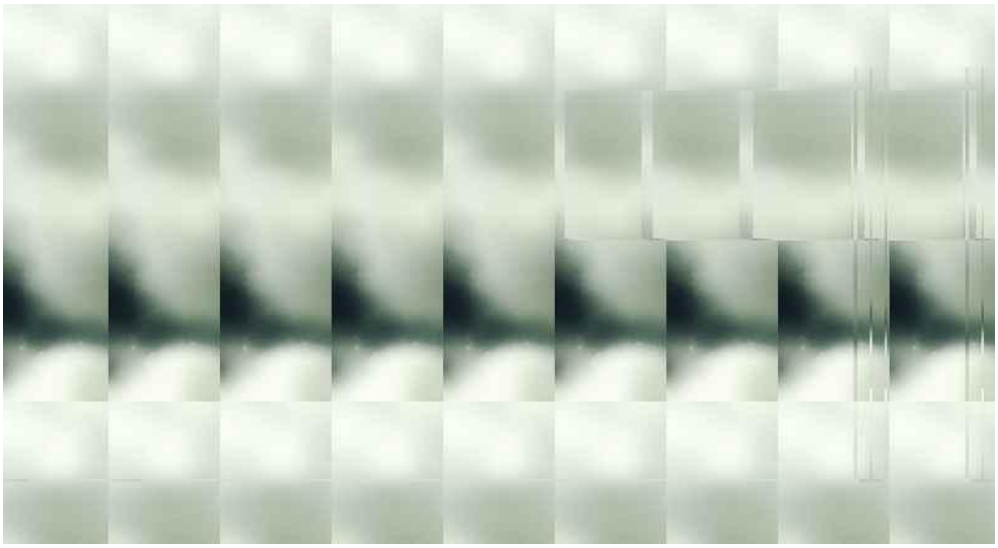
Dari gambar diatas kita mendapati data sebagai berikut :

Tabel 4.4 Data inputan *depthmp* kotak putih pojok kanan atas dan hasil

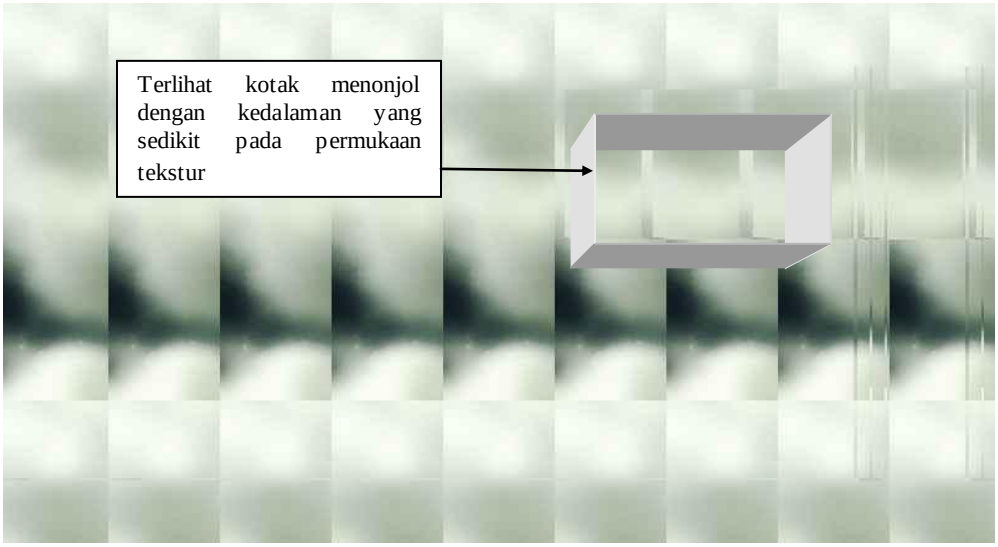
Ukuran DepthMap (Kb)	Ukuran Texture (Kb)	Tipe Stereogram	Oversampling	Ukuran Citra Output (Kb)
7,16	16,2	Tekstur	1 (Tidak)	15,6

Analisa hasil gambar :

Hasil *image processing* dari program dengan akomodasi mata maksimal adalah nampak kotak yang memnonjol kabur di pojok kanan atas pada lapisan tekstur. Dengan jarak pandang ±20cm dari mata pengamat, objek sulit kita lihat karena posisinya yang berada dipojok dari tekstur. Hal ini dikarenakan fokus mata yang umumnya terfokus pada bagian bagian tengah dari tekstur.



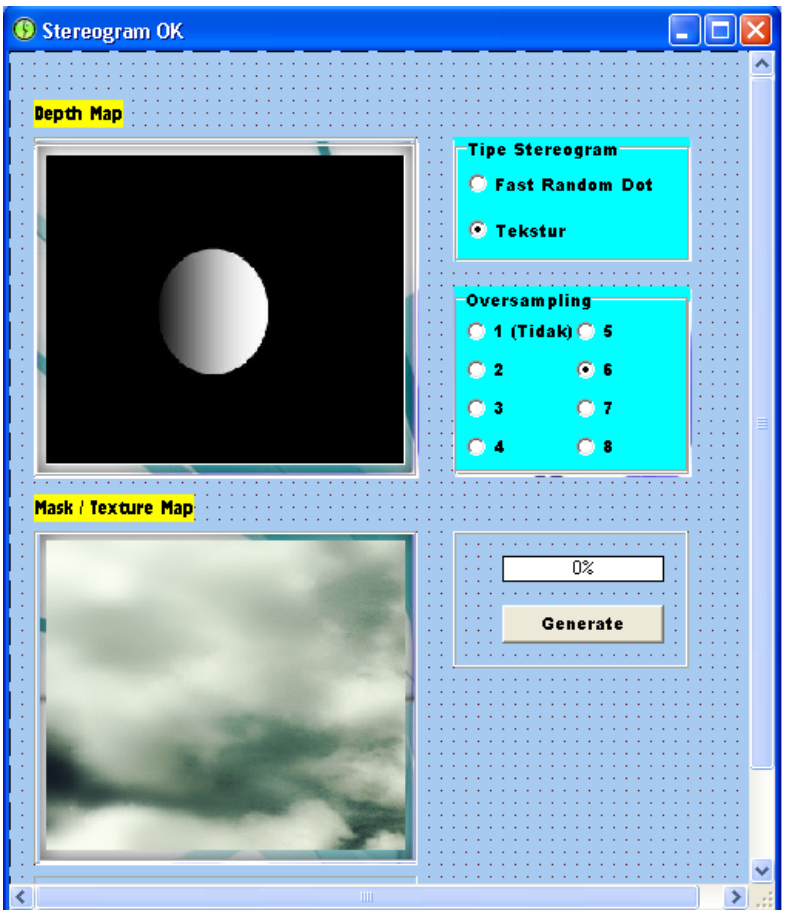
Gambar 4.19 Hasil pengujian dengan *depthmap* timbul kotak pojok kanan atas pada permukaan tekstur



Gambar 4.20 Ilustrasi hasil pengujian dengan *depthmap* timbul kotak pojok kanan atas pada permukaan tekstur

4.3. Pengujian Nilai *Oversampling* Yang Diubah - Ubah

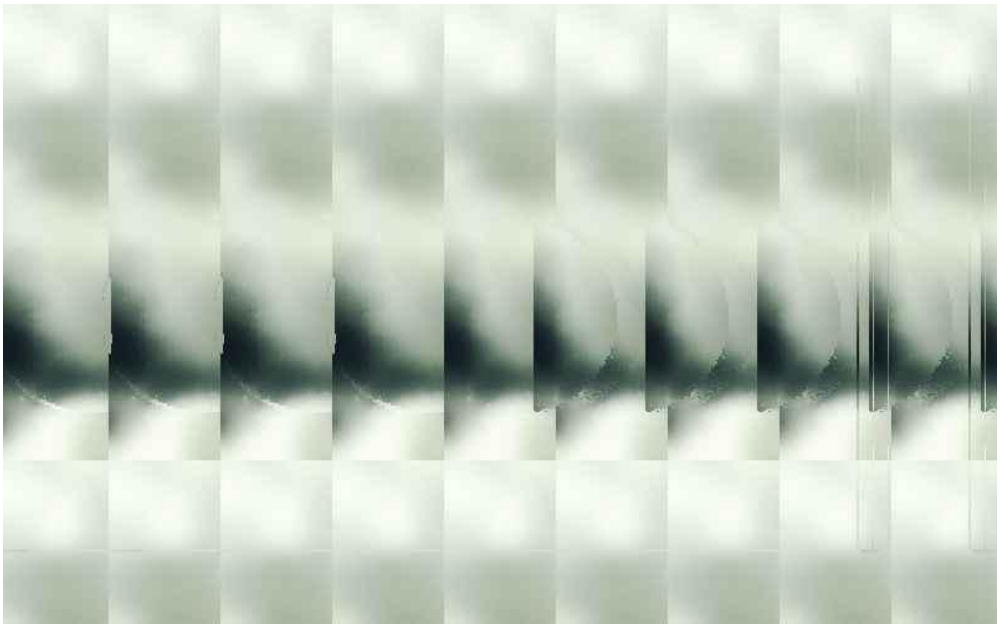
Di pengujian ini untuk membuktikan bahwa nilai oversampling yang diubah-ubah akan menghasilkan kesan citra *depthmask* yang makin timbul pada permukaan tekstur. Dengan inputan citra bulatan putih dan *background* yang sama, akan diuji dengan nilai *oversampling* diubah mulai 1 sampai 8. Pengamat mengamati secara manual, yaitu tanpa bantuan alat atau dengan mata fokus sejajar.



Gambar 4.21 Pengujian *depth map* dengan nilai *oversampling* yang diubah-ubah

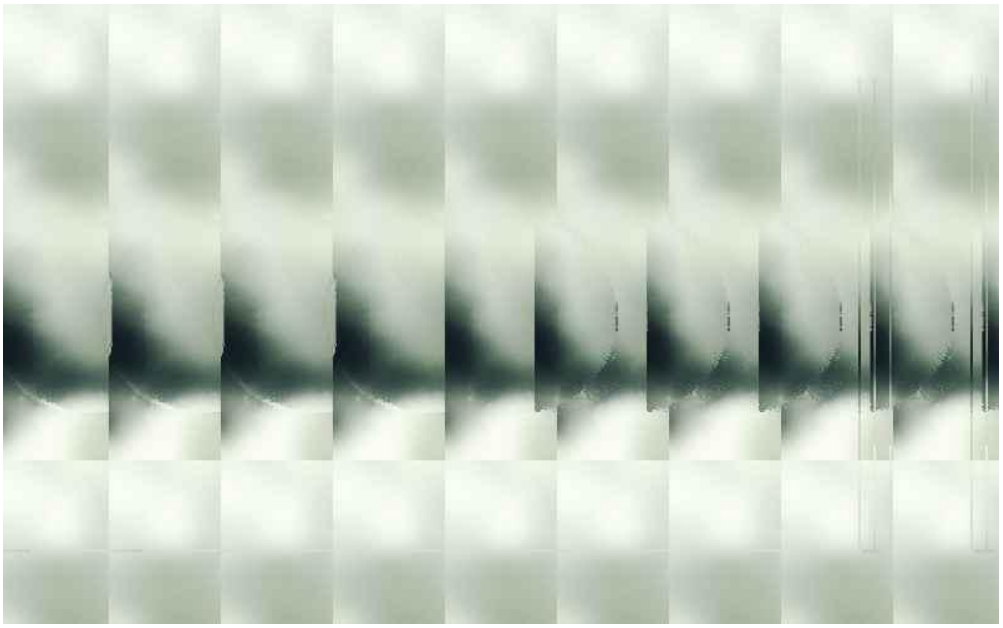
Dari penggabungan gambar diatas, citra disimpan dengan format JPEG sehingga diperoleh hasil dengan nilai *oversampling* diubah mulai 1 sampai 8 sebagai berikut :

- *Oversampling* dengan nilai 1(tidak)



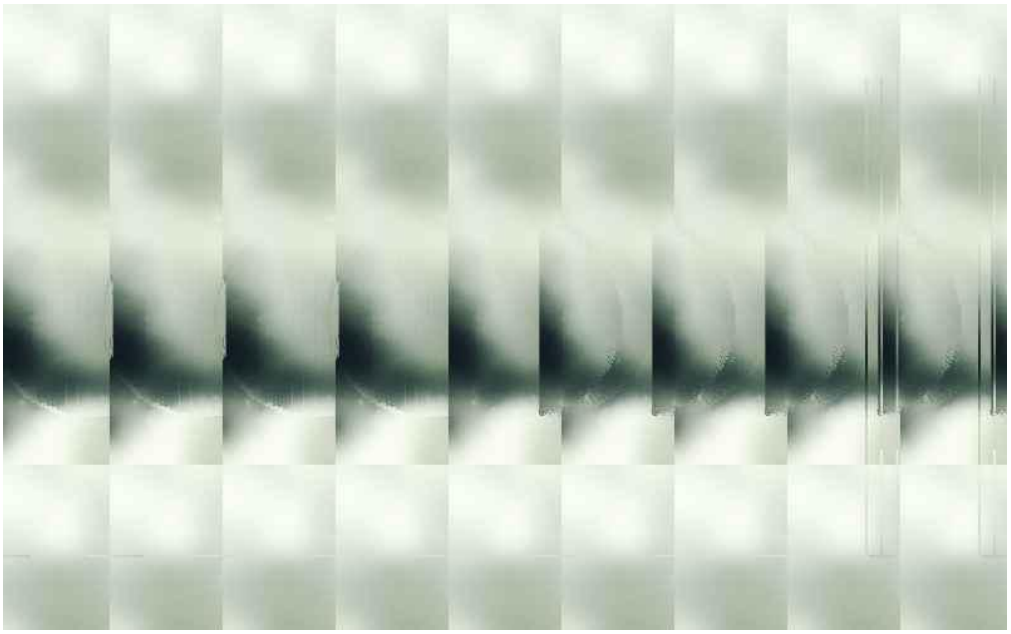
Gambar 4.22 Hasil pengujian dengan nilai *oversampling* 1

- *Oversampling* dengan nilai 2



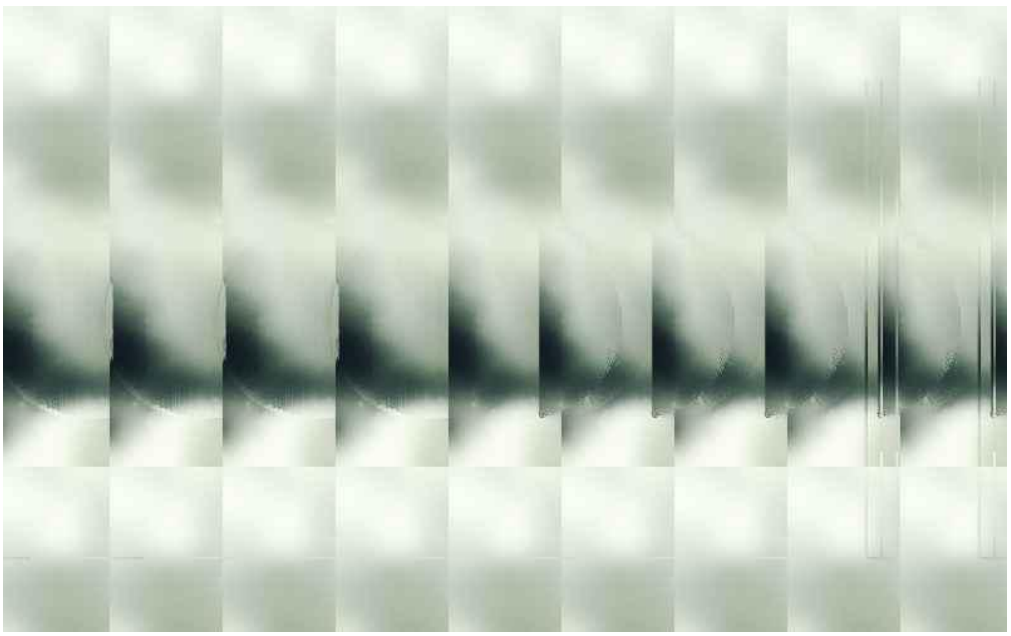
Gambar 4.23 Hasil pengujian dengan nilai *oversampling* 2

- *Oversampling* dengan nilai *3*



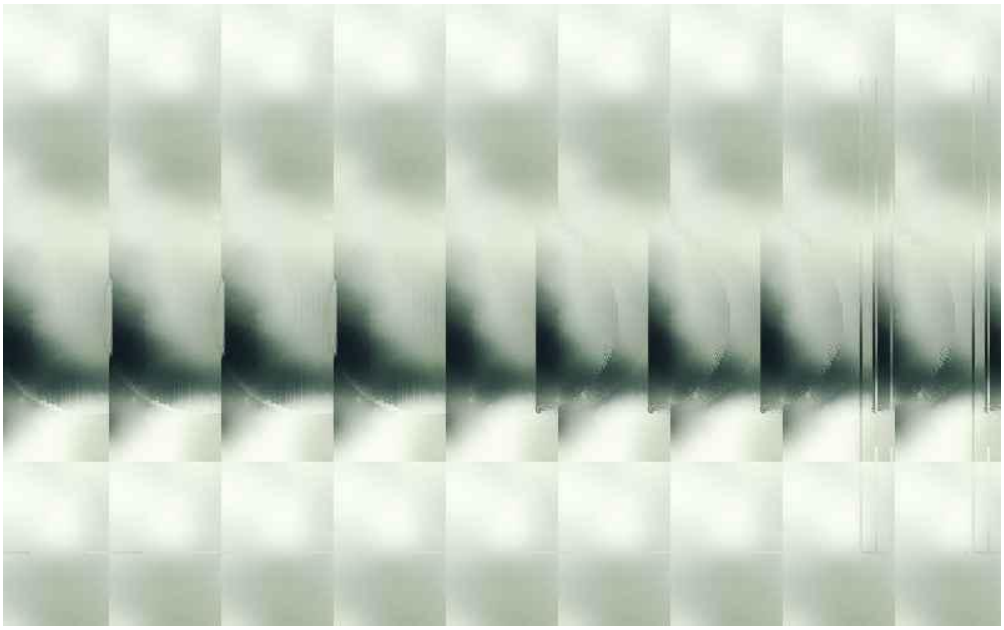
Gambar 4.24 Hasil pengujian dengan nilai *oversampling* 3

- *Oversampling* dengan nilai 4



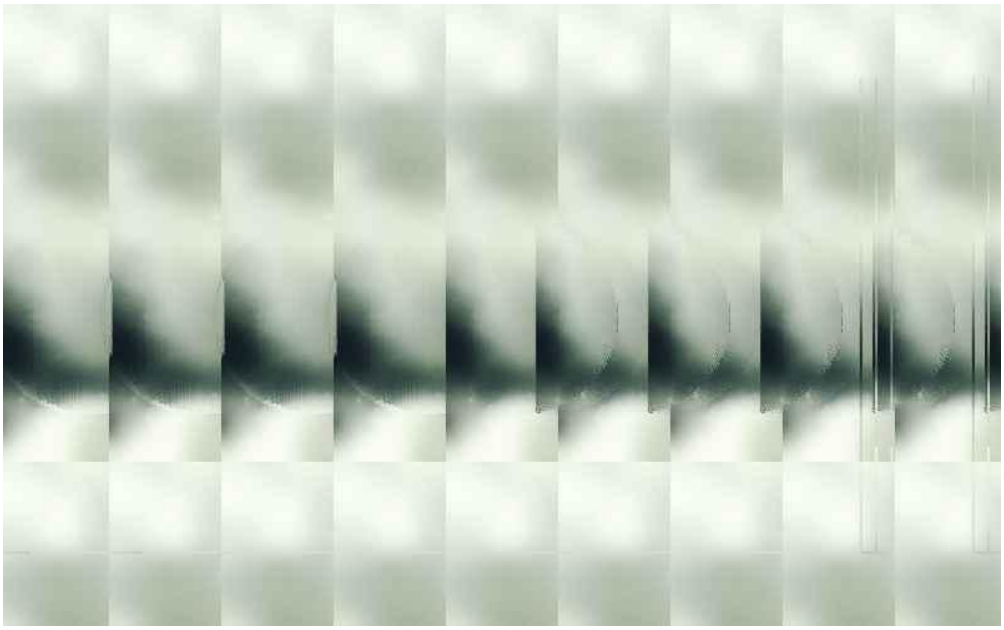
Gambar 4.25 Hasil pengujian dengan nilai *oversampling* 4

- *Oversampling* dengan nilai 5



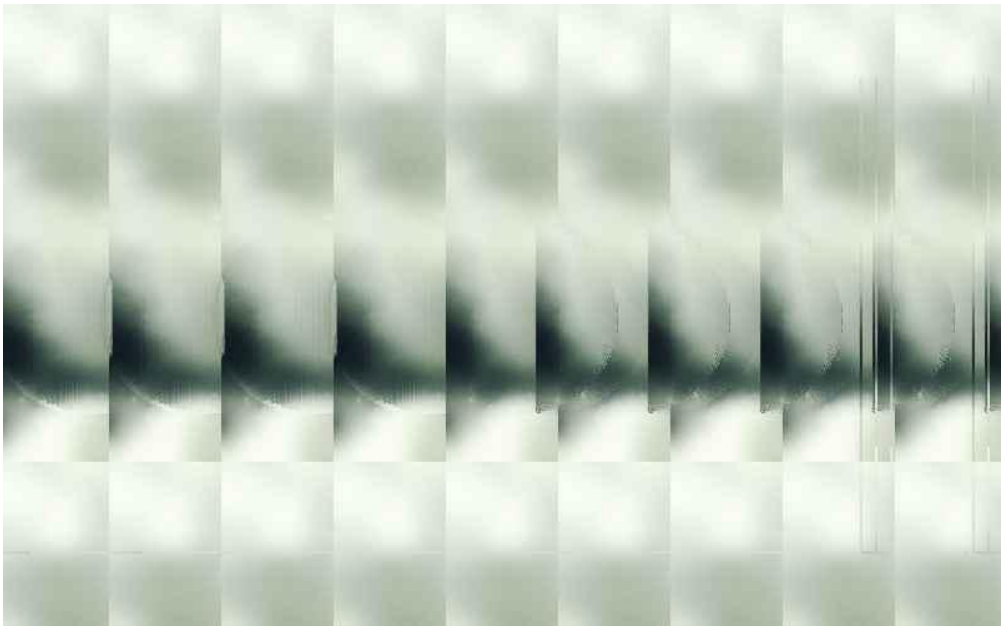
Gambar 4.26 Hasil pengujian dengan nilai *oversampling* 5

- *Oversampling* dengan nilai 6



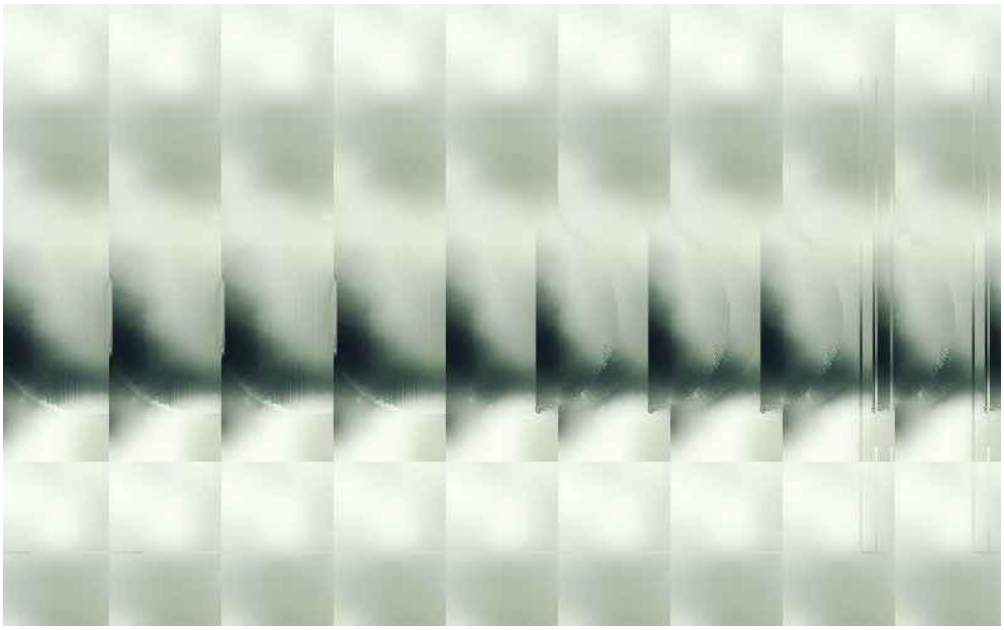
Gambar 4.27 Hasil pengujian dengan nilai *oversampling* 6

- *Oversampling* dengan nilai 7



Gambar 4.28 Hasil pengujian dengan nilai *oversampling* 7

- *Oversampling* dengan nilai 8



Gambar 4.29 Hasil pengujian dengan nilai *oversampling* 8

Dari pengamatan yang dilakukan, nilai *oversampling* dengan nilai 1 sampai 4 tidak begitu terlihat perubahan yang berarti sekali dikarenakan

pergeseran *pixel* citra yang hanya sedikit, citra cenderung mendekat kea rah pengamat. Berbeda dengan nilai *oversampling* antara 5 sampai 8, citra serasa timbul lebih keluar atau menjauhi pengamat dikarenakan pergeseran *pixel* dari citra *depthmask* yang bernilai banyak.

Berikut adalah data lengkap mengenai citra *autostereogram* yang telah diuji:

Tabel 4.5 *Autostereogram* dengan objek bulatan putih

No	Ukuran *Depthmap*	Ukuran Textur	Ukuran Hasil	*Oversampling*	Tipe	Format
1	7,16 kb	16,2 kb	16,4 kb	1	Tekstur	JPEG
2			16,2 kb	2	Tekstur	JPEG
3			16,2 kb	3	Tekstur	JPEG
4			16,2 kb	4	Tekstur	JPEG
5			16,2 kb	5	Tekstur	JPEG
6			192 kb	1	F R D	JPEG
7			1,52 Mb	1	F R D	Bmp
8			1,52 Mb	1	Tekstur	Bmp
9			1,52 Mb	2	Tekstur	Bmp
10			1,52 Mb	3	Tekstur	Bmp

Keterangan :

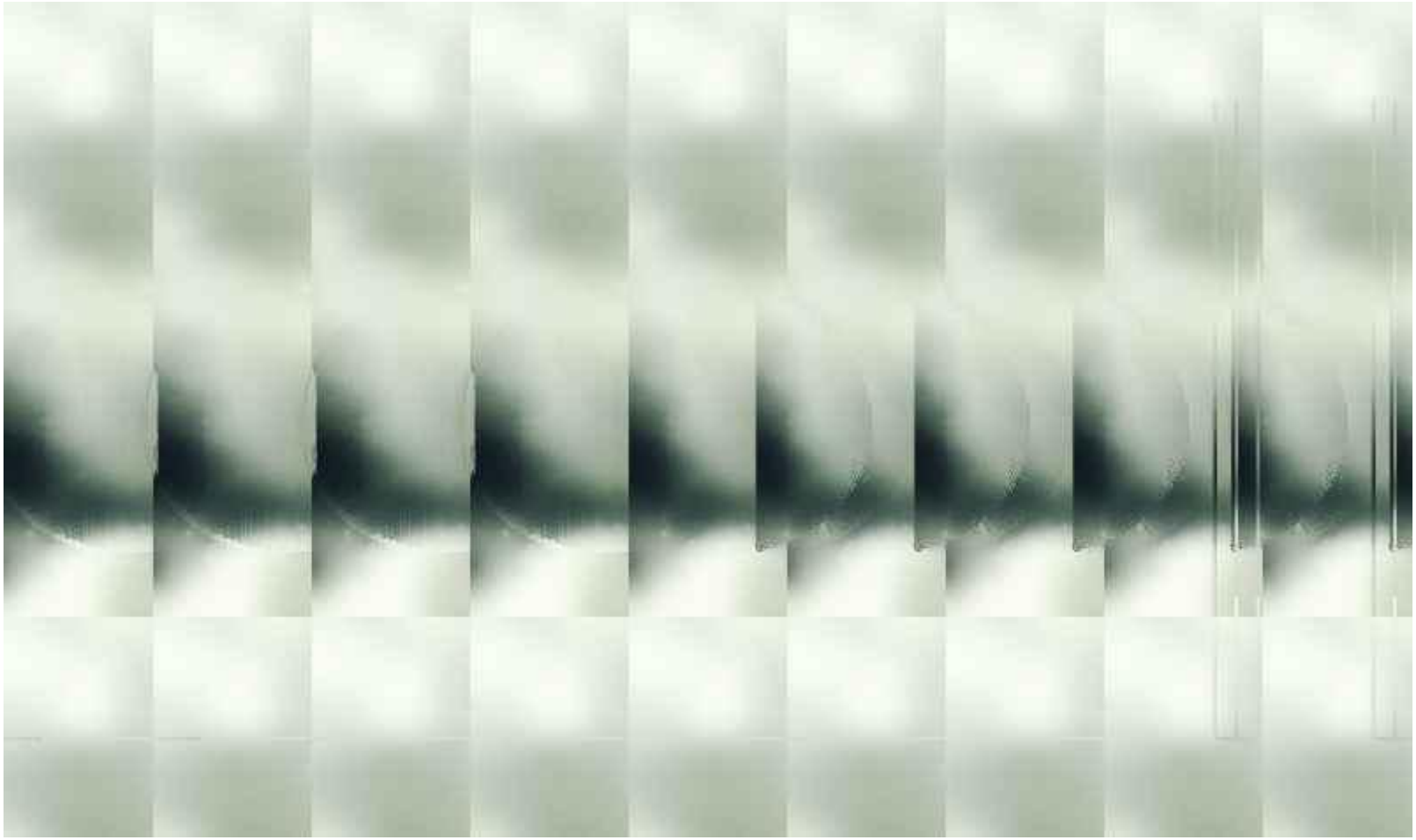
F R D = *Fast Random Dot*

4.4. Pengujian Dengan *Survey* Terhadap Hasil *Image Processing*

Pengujian ini bertujuan untuk memastikan atau mmembuktikan bahwa aplikasi ini telah berhasil menggabungkan *depthmask* dan tekstur serta berhasil menampakkan objek yang berupa *depthmask* yang terletak dibelakang tekstur. *Survey* dilakukan pada 10 orang dengan penglihatan yang normal (tidak berkaca mata) atau tanpa alat bantu apapun.

Dengan nilai *oversampling* yang diubah, objek *depthmask* dengan bentuk dan posisi yang berbeda seperti pada poin 4.1 dan 4.2, nilai oversampling standart

yaitu 4, serta jarak pengamat dari objek yang diubah yaitu mulai dari 10,20 dan 30 cm dengan waktu pengamatan maksimal 5 menit, dapat diperoleh data sebagai berikut :



Gambar 4.30 *Autostereogram* bulat putih pada motif

Tabel 4.6 Hasil survey citra *Autostereogram* bulatan putih

No	Pengamat	Jarak Pengamat dengan objek	Penampakan *DepthMap*			Keterangan
			Ya	Samar-samar	Tidak	
1	A	10 cm	X			Nampak timbul bulatan penuh
		20 cm	X			Nampak timbul bulatan penuh
		30 cm		X		Tampak setengah lingkaran tak timbul
2	B	10 cm			X	Hanya tampak motif
		20 cm		X		Tampak setengah lingkaran tak timbul
		30 cm	X			Nampak timbul bulatan penuh
3	C	10 cm	X			Nampak timbul bulatan penuh
		20 cm	X			Nampak timbul bulatan penuh
		30 cm		X		Tampak setengah lingkaran tak timbul
4	D	10 cm	X			Nampak timbul bulatan penuh
		20 cm	X			Nampak timbul bulatan penuh
		30 cm	X			Nampak timbul bulatan penuh
5	E	10 cm	X			Nampak timbul bulatan penuh
		20 cm	X			Nampak timbul bulatan penuh
		30 cm		X		Tampak setengah lingkaran tak timbul
6	F	10 cm		X		Tampak setengah lingkaran tak timbul
		20 cm			X	Hanya tampak motif
		30 cm		X		Tampak setengah lingkaran tak timbul
7	G	10 cm	X			Nampak timbul bulatan penuh
		20 cm	X			Nampak timbul bulatan penuh
		30 cm	X			Nampak timbul bulatan penuh
8	H	10 cm		X		Tampak setengah lingkaran tak timbul
		20 cm			X	Hanya tampak motif
		30 cm		X		Tampak setengah lingkaran tak timbul
9	I	10 cm			X	Hanya tampak motif
		20 cm		X		Tampak setengah lingkaran tak timbul
		30 cm	X			Nampak timbul bulatan penuh
10	J	10 cm		X		Tampak setengah lingkaran tak timbul
		20 cm			X	Hanya tampak motif
		30 cm		X		Tampak setengah lingkaran tak timbul

Berdasarkan data diatas, penulis dapat menganalisa citra hasil dari aplikasi *Autostereogram* sebagai berikut :

- Citra *Austostereogram* dengan objek bulatan putih intinya adalah 50% pengamat dapat melihat penampakan *DepthMap* yang nampak kearah

pengamat pada bidang gambar dengan posisi 10 cm dari objek, 50% pengamat melihat objek pada jarak 20 cm dan 40% pengamat melihat objek pada jarak 30 cm.

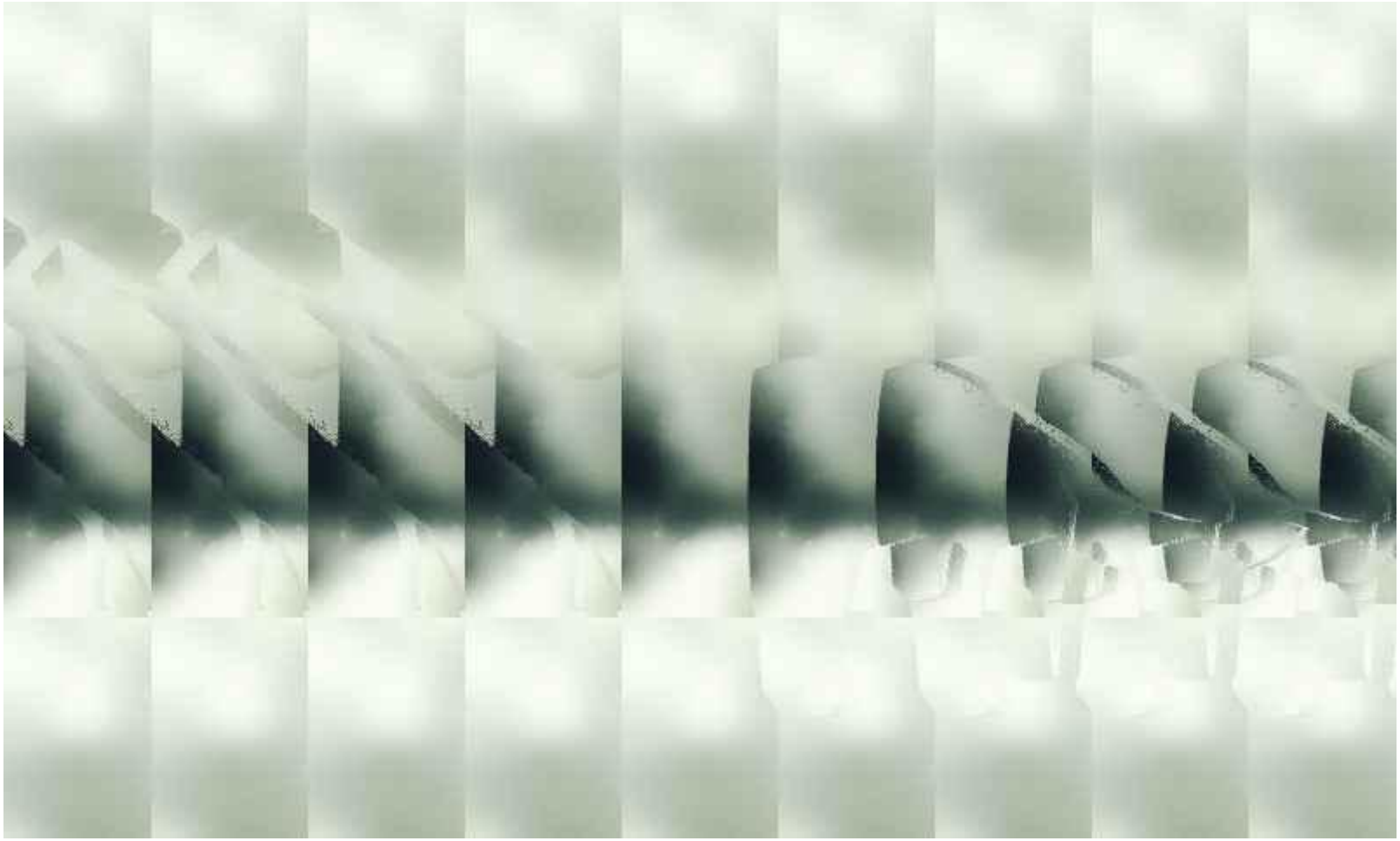
- Sedangkan 30% pengamat hanya melihat samar atau bahkan tidak melihat objek yang dimaksud pada bidang.

Lebih rinci pengujian ini, penulis mengambil kesimpulan dari survey dari data sebagai berikut :

Tabel 4.7 Jarak Penampakan Objek Bulatan

No	Pengamat	Jarak Pandang			Keterangan
		10 cm	20 cm	30 cm	
1	A	X	X	-	Melihat Objek
2	B	-	-	X	Melihat Objek
3	C	X	X	-	Melihat Objek
4	D	X	X	X	Melihat Objek
5	E	X	X	-	Melihat Objek
6	F	-	-	-	Tidak melihat Objek
7	G	X	X	X	Melihat Objek
8	H	-	-	-	Tidak melihat Objek
9	I	-	-	X	Melihat Objek
10	J	-	-	-	Tidak melihat Objek

Bisa dilihat pada tabel, 7 pengamat bisa melihat objek pada jarak antara 10, 20 dan 30 cm atau 70% pengamat menyetujui telah melihat objek, sedangkan 3 orang atau 30% tidak melihat objek yang dimaksud pada bidang gambar.



Gambar 4.31 *Autostereogram* dinosaurus pada motif

Tabel 4.8 Hasil survey citra *Autostereogram* dinosaurus

No	Pengamat	Jarak Pengamat dengan objek	Penampakan *DepthMap*			Keterangan
			Ya	Samar-samar	Tidak	
1	A	10 cm	X			Nampak timbul dinosaurus
		20 cm	X			Nampak timbul dinosaurus
		30 cm		X		Tampak lengkungan tak timbul
2	B	10 cm			X	Hanya tampak motif
		20 cm		X		Tampak lengkungan tak timbul
		30 cm		X		Nampak timbul dinosaurus
3	C	10 cm	X			Nampak timbul dinosaurus
		20 cm	X			Nampak timbul dinosaurus
		30 cm		X		Tampak lengkungan tak timbul
4	D	10 cm	X			Nampak timbul dinosaurus
		20 cm	X			Nampak timbul dinosaurus
		30 cm	X			Nampak timbul dinosaurus
5	E	10 cm			X	Hanya tampak motif
		20 cm		X		Tampak lengkungan tak timbul
		30 cm		X		Tampak lengkungan tak timbul
6	F	10 cm			X	Hanya tampak motif
		20 cm		X		Tampak lengkungan tak timbul
		30 cm		X		Tampak lengkungan tak timbul
7	G	10 cm	X			Nampak timbul dinosaurus
		20 cm	X			Nampak timbul dinosaurus
		30 cm	X			Nampak timbul dinosaurus
8	H	10 cm			X	Hanya tampak motif
		20 cm			X	Hanya tampak motif
		30 cm		X		Tampak lengkungan tak timbul
9	I	10 cm			X	Hanya tampak motif
		20 cm			X	Hanya tampak motif
		30 cm		X		Tampak lengkungan tak timbul
10	J	10 cm			X	Hanya tampak motif
		20 cm			X	Hanya tampak motif
		30 cm		X		Tampak lengkungan tak timbul

Berdasarkan data diatas, penulis dapat menganalisa citra hasil dari aplikasi *Autostereogram* sebagai berikut :

- Citra *Austostereogram* dengan objek dinosaurus intinya adalah 40% pengamat dapat melihat penampakan *DepthMap* yang nampak kearah

pengamat pada bidang gembar dengan posisi 10 cm dari objek, 40% pengamat melihat objek pada jarak 20 cm dan 20% pengamat melihat objek pada jarak 30 cm.

- Sedangkan 60% pengamat hanya melihat samar atau bahkan tidak melihat objek yang dimaksud pada bidang.

Lebih rinci pengujian ini, penulis mengambil kesimpulan dari survey dari data sebagai berikut :

Tabel 4.9 Jarak Penampakan Objek Dinosaurus

No	Pengamat	Jarak Pandang			Keterangan
		10 cm	20 cm	30 cm	
1	A	X	X	-	Melihat Objek
2	B	-	-	-	Tidak melihat Objek
3	C	X	X	X	Melihat Objek
4	D	X	X	X	Melihat Objek
5	E	-	-	-	Tidak melihat Objek
6	F	-	-	-	Tidak melihat Objek
7	G	X	X	X	Melihat Objek
8	H	-	-	-	Tidak melihat Objek
9	I	-	-	-	Tidak melihat Objek
10	J	-	-	-	Tidak melihat Objek

Bisa dilihat pada tabel, 4 pengamat bisa melihat objek pada jarak antara 10, 20 dan 30 cm atau 40% pengamat menyetujui telah melihat objek, sedangkan 6 orang atau 60% tidak melihat objek yang dimaksud pada bidang gambar.

Pada pengujian letak objek ditepi kanan bidang, para pengamat tidak dapat melihat objek timbul secara utuh atau hanya terlihat samar-samar garis lekukan dari objenya saja.

BAB V

PENUTUP

5.1. Kesimpulan

Dari pengujian yang telah dilakukan, dapat disimpulkan menjadi beberapa hal, antara lain:

1. Hasil dari proses *generate* pada *depth mask* yang terletak di bagian tepi kiri maupun tepi kanan akan sulit untuk dilihat karena mata manusia tidak dapat melihat pasangan titik (titik koresponden) yang terletak di luar bidang citra.
2. Semakin putih warna obyek pada *depth mask*, maka hasil *autostereogram* akan membuat obyek tersebut berada lebih dekat dengan pengamat. Sebaliknya bila warna pada obyek semakin gelap, maka hasil *stereogram* akan membuat obyek tersebut berada lebih jauh dari pengamat. Pengamatan dilakukan dengan meggunakan teknik *wall eyed*
3. Pada pen-*survey*-an, bila jarak pengamat berada pada posisi yang tepat, jarak terbaik adalah 20 cm dari citra objek *Autostereogram* akan terlihat sempurna.

5.2. Saran

Pada aplikasi ini masih terdapat banyak kekurangan dalam pemrosesan citranya, Untuk pengembangan selanjutnya, dapat dikembangkan suatu *pre-processing* atau aplikasi untuk membuat *depth mask* dari suatu *image* yang akan di-*input*-kan serta beberapa tambahan untuk *processingnya*..

Daftar Pustaka

[1]. Robi'in Bambang. 2004. *Pemrograman Grafis Multimedia Menggunakan Delphi*. Yogyakarta: ANDI

[2]. *Wikipedia*, 2007.*The Free Encyclopedia Autostereogram*. (Online), (http://en.wikipedia.org/wiki/Autostereogram, diakses tanggal 1 Mei 2012)

[3]. Kay, David C, & Levine John R. 1992. *Graphics file formats*. United State of America:Windcrest Books.

[4]. Thimbleby, Harold W.,Inglish, Stuart., and Witten, Ian H. (n.d) *Displaying 3D images: algorithms for single image random dot stereograms*. 26 Januari 2012.

[5]. Naf'an Akhun. 2006**.** *The Making of Stereogram-Cara Pembuatan Stereogram*, (Online), download dari : (http://genghiskhun.com/Download FLIGHT of IDEAS.html, diakses tanggal 1 Mei 2012)

[6]. http://www.blender.org/

[7]. W.A. Steer PhD. 2002. *All About Stereogram*, (Online), (http://www.techmind.org/stereo.html, diakses 23 April 2012)

[8]. Kamilasari. 2006. *Grafik Komputer dan Pengolahan Citra : Konsep Dasar*, (Online), download dari : (http://www.google/Pengolahan Citra.html, diakses 3 Mei 2012)

[9]. Adobe Systems Incorporated. 2002 . "*About Color Modes and Models*." Adobe Photoshop 7 Help Files.

Lampiran

```
unit Stereogram;

interface

uses WinTypes, WinProcs, Messages, SysUtils, Classes, Controls,
  Forms, Graphics, FastDIBOlej, ExtCtrls;


const

  MaxWidth = 1024; {lebar maksimum citra}
  MaxHeight = 800; {tinggi maksimum citra}
  MaxOversamp = 8;

  {Lebar maksimum * max oversampling}
  maxX = MaxWidth * MaxOversamp;

  DepthLevels = 256; {Dari 128 perubahan tidak tampak}

type


  TStereoGramType = (sgtFastRandomDot, sgtColoredDot,
sgtRandomDotTextured, sgtTextured);
  TBlurType = (blurNone, blurLight, blurMedium, blurHeavy);
  TRandomDotColorType = (dctBlackWhite, dctRGB);

  TStereogram = class(TComponent)
  private

    patHeight,
      xdpi,
      ydpi,
      yShift: Integer;

    lastlinked,
      i,
      vwidth,
      eyeSep,
      veyeSep,
      maxsep,
      vmaxsep,
      s, k, kk,
      poffset,
      featureZ,
      sep,
      left, right: Integer;

    vis: Boolean;
    red, green, blue: Integer;
    obsDist, mindepth, maxdepth: double;
    hCol: Integer;
```

```
    {Array untuk menyimpan nilai pixel}
    lookL,
      lookR: array[0..maxx] of word;

   {hasil color untuk garis}
    colour: array[0..maxx] of TCOLOR;

    SeparTable: array[0..DepthLevels] of Integer;

      { Tempat khusus dari TStereogram }
        { Penyimpanan untuk property DepthMap }
    FDepthMap: TPicture;
        { Untuk menyimpan property Height }
    FHeight: Integer;
        { Untuk menyimpan property MaskMap }
    FMaskMap: TPicture;
        { Untuk menyimpan property ResultMap }
    FResultMap: TPicture;
        { Untuk menyimpan property Separation }
    FSeparation: Integer;
        { Untuk menyimpan property Type }
    FType: TStereoGramType;
        { Untuk menyimpan property Width }
    FWidth: Integer;
        { Penunjuk dari aplikasi yang diatur oleh OnGenerater,
jika ada }
    FOnGenerate: TNotifyEvent;

    FDotIntensity: integer;
    FRandomDotColor: TRandomDotColorType;

    FobsDist: Double;

    Fmaxdepth: Double;

    Fmindepth: Double;

    FOversam: Integer;

    FBlur: TBlurType;
    FBlurParam: Integer;
    FDPI: Integer;

    {FastDIBs untuk akses cepat ke pixel}
    FFastDepthMap,
      FFastMaskMap,
      FFastResultMap: TFastDIBOlej;

    procedure AutoInitialize;
        { Metode yang digunakan dari beberapa objek yang di buat
secara bebas }
    procedure AutoDestroy;
        { Penulisan metode untuk property DepthMap }
    procedure SetDepthMap(Value: TPicture);
        { Pembacaan metode untuk property Height }
    function GetHeight: Integer;
        { Penulisan metode untuk property Height }
    procedure SetHeight(Value: Integer);
        { Penulisan metode untuk property MaskMap }
```

```
    procedure SetMaskMap(Value: TPicture);
        { Pembacaan metode untuk property Separation }
    function GetSeparation: Integer;
        { Penulisan metode untuk property Separation }

    procedure SetSeparation(Value: Integer);

    procedure SetDPI(Value: Integer);

    procedure Setoversam(Value: Integer);
        { Pembacaan metode untuk property Type }
    function GetType: TStereoGramType;
        { Penulisan metode untuk property Type }
    procedure SetType(Value: TStereoGramType);
        { Pembacaan metode untuk property Width }
    function GetWidth: Integer;
        { Penulisan metode untuk property Width }
    procedure SetWidth(Value: Integer);

    procedure SetDotIntensity(Value: Integer);
    procedure SetObsDist(Value: Double);
    procedure SetMaxdepth(Value: Double);
    procedure SetMindepth(Value: Double);
    procedure PrepareParams;
    procedure PrepareImages;
    procedure ReturnResulTPicture;
    function GetColor(FBMP: TFastDIBOlej; Col, Row: Integer):
TColor;
    procedure SetColor(FBMP: TFastDIBOlej; Col, Row: Integer; R,
G, B: Byte);

    procedure PrepareSeparTable;
    procedure PrepareRandomTexture;
    procedure GenerateStereogram;
    function ReadResultMap: TPicture;

    procedure MakeLineRandomDot(Row: Integer);
    procedure MakeLineTextured(Row: Integer);

  protected

      { metode yang terlindungi dari TStereoGram }
    procedure Loaded; override;

  public
    Progress: Integer; {Proses dalam persen}

    property ResultMap: TPicture read ReadResultMap; {Hasil yang
disimpan setelah Generate}
    property FastResultMap: TFastDIBOlej read FFastResultMap;
{Hasil yang disimpan setelah Generate}

      { Metode secara umum dari TStereoGram }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;


    {Generate dimulai - hasil dari bitmap telah disimpan ResultMap
dan FastResultMap}
```

```
    procedure Generate;

  published
      { Properti-properti yang dihasilkan dari TStereoGram }
    property OnGenerate: TNotifyEvent read FOnGenerate write
FOnGenerate;
    property DepthMap: TPicture read FDepthMap write SetDepthMap;
    property Height: Integer read GetHeight write SetHeight
default 600;
    property MaskMap: TPicture read FMaskMap write SetMaskMap;
    property Separation: Integer read GetSeparation write
SetSeparation default 90;
    property StereoType: TStereoGramType read GetType write
SetType;
    property Width: Integer read GetWidth write SetWidth default
800;
    property DotIntensity: Integer read FDotIntensity write
SetDotIntensity default 50;
    property DPI: Integer read FDPI write SetDPI default 72;
    property Blur: TBlurType read FBlur write FBlur;
    property RandomDotColor: TRandomDotColorType read
FRandomDotColor write FRandomDotColor;
    property Oversampling: Integer read Foversam write Setoversam;
    property ObservDistance: Double read FobsDist write
SetobsDist;
    property Max3Ddepth: Double read Fmaxdepth write Setmaxdepth;
    property Min3Ddepth: Double read Fmindepth write Setmindepth;

  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TStereogram]);
end;

{ Pengumpulan metode yang dibuat dari variabel dan nilai properti
pembuatan objek2}

procedure TStereoGram.AutoInitialize;
begin
  Progress := 0;
  FDepthMap := TPicture.Create;
  FMaskMap := TPicture.Create;
  FResultMap := TPicture.Create;
  FFastResultMap := TFastDIBOlej.Create;
  FHeight := 600;
  FWidth := 800;
  FDotIntensity := 50;
  FRandomDotColor := dctBlackWhite;
  FDPI := 72;
  FBlur := blurNone;
  Foversam := 1;
  FobsDist := 14;
  Fmaxdepth := 14;
  Fmindepth := 7;
```

```
  PrepareParams;

end; { of AutoInitialize }

{ Method to free any objects created by AutoInitialize }

procedure TStereoGram.AutoDestroy;
begin
  FDepthMap.Free;
  FMaskMap.Free;
  FResultMap.Free;
  FFastResultMap.Free;
end; { of AutoDestroy }


procedure TStereoGram.SetDepthMap(Value: TPicture);
begin
     { Use Assign method because TPicture is an object type }
  FDepthMap.Assign(Value);
  if (Width < DepthMap.Width) then Width := DepthMap.Width;
  if (Height < DepthMap.height) then Height := DepthMap.Height;
end;

{ Read method for property Height }

function TStereoGram.GetHeight: Integer;
begin
  Result := FHeight;
end;

{ Write method for property Height }

procedure TStereoGram.SetHeight(Value: Integer);
begin
  if value < 1 then FHeight := 600
  else
    if Value > MaxHeight then FHeight := MaxHeight
    else FHeight := Value;
  if FType = sgtColoredDot then
  begin
    FHeight := DepthMap.Height;
  end;

end;

{ Write method for property MaskMap }

procedure TStereoGram.SetMaskMap(Value: TPicture);
begin
  FMaskMap.Assign(Value);
end;

{ Read method for property Separation }
function TStereoGram.GetSeparation: Integer;
begin
  Result := FSeparation;
end;
```

```
{ Write method for property Separation }

procedure TStereoGram.SetSeparation(Value: Integer);
begin
  FMaxDepth := ((-1 * Value * FDPI * FobsDist) / (Value -
(Trunc(FDPI * 2.5))) + 1) / FDPI;
  Fmaxdepth := (Trunc(Fmaxdepth * 1000)) / 1000;
  if Fmindepth > FMaxDepth then Fmindepth := Fmaxdepth - 5;
  PrepareParams;
end;


{Prepare separation values to table used for faster resuls}
procedure TStereoGram.PrepareSeparTable;
var
  x: Integer;
  featureZ: Integer;
begin
  for x := 0 to DepthLevels - 1 do
  begin
    featureZ := Trunc(maxdepth - x * (maxdepth - mindepth) /
DepthLevels);
    SeparTable[x] := Round((veyeSep * featureZ) / (featureZ +
obsDist));
  end;
end;


procedure TStereoGram.Setoversam(Value: Integer);
begin
  if Value < 1 then FOversam := 1
  else
    if Value > MaxOversamp then FOversam := MaxOversamp
    else
      FOversam := Value;
  if (StereoType = sgtFastRandomDot) or (StereoType =
sgtColoredDot) then
    FOversam := 1;
end;


procedure TStereoGram.SetObsDist(Value: Double);
begin
  FobsDist := Value;
  PrepareParams;
end;

procedure TStereoGram.SetMaxdepth(Value: Double);
begin
  FmaxDepth := Value;
  PrepareParams;
end;

procedure TStereoGram.SetMindepth(Value: Double);
begin
  FminDepth := Value;
  PrepareParams;
end;
```

```
{ Membaca metode untuk property Type }

function TStereoGram.GetType: TStereoGramType;
begin
  Result := FType;
end;

{ Penulisan metode untuk property Type }

procedure TStereoGram.SetType(Value: TStereoGramType);
begin
  if Value = sgtFastRandomDot then FOversam := 1;
  if Value = sgtColoredDot then
  begin
    Width := DepthMap.Width;
    Height := DepthMap.Height;
  end;
  FType := Value;
end;



{ Membaca metode property Width }

function TStereoGram.GetWidth: Integer;
begin
  Result := FWidth;
end;

{ Penulisan metode untuk property Width }

procedure TStereoGram.SetWidth(Value: Integer);
begin
  if Value < 1 then FWidth := 600
  else if Value > MaxWidth then FWidth := MaxWidth else
    FWidth := Value;
  if FType = sgtColoredDot then
  begin
    FWidth := DepthMap.Width;
  end;
end;

procedure TStereoGram.SetDPI(Value: Integer);
begin
  if Value < 10 then FDPI := 10
  else FDPI := Value;

  if (FDPI > Width) then
  begin
    FDPI := Width div 2;
  end;
  if (FDPI > Height) then
  begin
    FDPI := Height div 2;
  end;
  PrepareParams;
end;
```

```
procedure TStereoGram.SetDotIntensity(Value: Integer);
begin
  if Value < 1 then FDotIntensity := 1
  else
    if Value > 99 then FDotIntensity := 99
    else FDotIntensity := Value;
end;

procedure TStereoGram.PrepareParams;
begin

  patHeight := FMaskMap.Height;
  xdpi := FDPI;
  ydpi := FDPI;
  yShift := ydpi div 16;
  vwidth := width * Foversam;
  obsDist := FDPI * FobsDist;
  eyeSep := Round(FDPI * 2.5);
  veyeSep := eyeSep * Foversam;
  maxdepth := FDPI * FMaxDepth;
  maxsep := Trunc(((eyeSep * maxdepth) / (maxdepth + obsDist)));



  if (maxSep > FMaskMap.Width) and
    (FMaskMap.Width > 0) and ((FType = sgtRandomDotTextured) or
(FType = sgtTextured))
    then maxSep := FMaskMap.Width;

  FSeparation := MaxSep;

  if FType = sgtRandomDotTextured then
    patHeight := MaxSep;

  vmaxsep := Foversam * maxsep;
  s := vwidth div 2 - vmaxsep div 2;//jarak pergeseran pixel//
  poffset := vmaxsep - (s mod vmaxsep);
//  mindepth := Trunc((sepfactor * maxdepth * obsdist) / ((1 -
sepfactor) * maxdepth + obsdist));
  mindepth := FDPI * FMinDepth;

  K := (Width - DepthMap.Width) div 2;
  KK := (Height - DepthMap.Height) div 2;


  {BLUR CONST}
  if FBlur = blurLight then FBlurParam := 15;
  if FBlur = blurMedium then FBlurParam := 10;
  if FBlur = blurHeavy then FBlurParam := 7;

end;


procedure TStereoGram.PrepareImages;
begin
  FFastResultMap.Free;
  FFastResultMap := TFastDIBOlej.Create;
  FFastDepthMap := TFastDIBOlej.Create;
```

```
  FFastMaskMap := TFastDIBOlej.Create;
  FFastResultMap.LoadFromHandle(HGDIOBJ(FResultMap.Bitmap.Handle),
24, 0);
  FFastDepthMap.LoadFromHandle(HGDIOBJ(FDepthMap.Bitmap.Handle),
24, 0);
  FFastMaskMap.LoadFromHandle(HGDIOBJ(FMaskMap.Bitmap.Handle), 24,
0);
end;

procedure TStereoGram.ReturnResulTPicture;
begin
  FFastDepthMap.Free;
  FFastMaskMap.Free;
end;

function TStereoGram.GetColor(FBMP: TFastDIBOlej; Col, Row:
Integer): TColor;
begin

  Result := (FBMP.Pixels[Row, Col].B shl 16) +
    (FBMP.Pixels[Row, Col].G shl 8) +
    (FBMP.Pixels[Row, Col].R);
end;

procedure TStereoGram.SetColor(FBMP: TFastDIBOlej; Col, Row:
Integer; R, G, B: Byte);
var
  FAF: TFColor;
begin
  Faf.r := R;
  Faf.g := G;
  Faf.b := B;
  FBMP.Pixels24[Row, Col] := Faf;
end;


procedure TStereoGram.Generate;
begin
  if (Progress > 0) and (Progress < 100) then Exit;
 {Test bitmaps}
  if FDepthMap.Bitmap.Empty then
    raise Exception.Create('DepthMap kosong, Isikan DephtMap
terlebih dahulu!');

  if (FType = sgtColoredDot) or (FType = sgtTextured) then
  begin
    if FMaskMap.Bitmap.Empty then
      raise Exception.Create('MaskMap Kosong');
  end;
  if (FType = sgtColoredDot) then
  begin
    if (FDepthMap.Width <> FMaskMap.Width) or
      (FDepthMap.Height <> FMaskMap.Height) then
      raise Exception.Create('Jika menggunakan ColourDot, MaskMap
dan DepthMap harus memiliki ukuran yang sama');
  end;
  if (FType = sgtColoredDot) then
  begin
    if (FDepthMap.Width <> Width) or
```

```
      (FDepthMap.Height <> Height) then
      raise Exception.Create('Jika menggunakan ColourDot, DepthMap
& citra hasil harus memiliki ukuran yang sama');
  end;
  ResultMap.Bitmap.Width := Width;
  ResultMap.Bitmap.Height := Height;
  ResultMap.Bitmap.Monochrome := False;
  {Langkah utama untuk menghasilkan Citra}
  PrepareParams;
  PrepareImages;
  PrepareSeparTable;
  GenerateStereogram;
  ReturnResulTPicture;
end;


function TStereoGram.ReadResultMap: TPicture;
begin
  FFastResultMap.Draw(FResultMap.Bitmap.Canvas.Handle, 0, 0);
  Result := FResultMap;
end;

procedure TStereoGram.MakeLineRandomDot(Row: Integer);
var
  x, y: Integer;
  kkk: Integer;
begin

  y := Row;
  for x := 0 to Width do LookL[x] := x;

  for x := 0 to Width - 1 do
  begin

    if (Y <= KK) or (Y >= (FDepthMap.Height + KK)) or
      ((X div FOversam) <= K) or ((X div FOversam) >=
(FDepthMap.Width + K))
      then
    begin
      hCol := 0;
    end
    else
    begin
      hCol := Trunc(((GetColor(FFastDepthMap, x div Foversam - k,
y - kk) / $00FFFFFF) * (DepthLevels - 1)));
    end;

    if FBlur = blurNone then
    begin
      Sep := SeparTable[hCol];
    end
    else
    begin
      featureZ := Trunc(maxdepth - hCol * (maxdepth - mindepth) /
DepthLevels);
      sep := Round((veyeSep * featureZ) / (featureZ + obsDist) +
((random(10) - 5) / FBlurParam));
    end;
```

```
    left := x - (sep div 2);
    right := left + sep;
    if (0 <= left) and (right < Width) then
    begin
      kkk := LookL[left];
      while (kkk <> left) and (kkk <> right) do
      begin
        if (kkk < right) then
        begin
          left := kkk;
        end
        else
        begin
          left := right;
          right := kkk;
        end;
        kkk := LookL[Left];
      end;
      LookL[left] := right;
    end;
  end;

  for x := Width - 1 downto 0 do
  begin
    if (LookL[x] = x) then
    begin
      if FRandomDotColor = dctBlackWhite then
      begin
        if random(100) > FDotIntensity then
        begin
          SetColor(FFastResultMap, x, y,
            255, 255, 255);
        end
        else
        begin
          SetColor(FFastResultMap, x, y,
            0, 0, 0);
        end;
      end
      else
      begin
        SetColor(FFastResultMap, x, y,
          Random(4) * 64 + 64, Random(4) * 64 + 64, Random(4) * 64
+ 64);
      end;

    end
    else
    begin
      FFastResultMap.Pixels[y, x] := FFastResultMap.Pixels[y,
LookL[x]];
    end;
  end;
end;


procedure TStereoGram.MakeLineTextured(Row: Integer);
var
  x, xx, Y: integer;
```

```
  col: TColor;
begin

  y := Row;

  for x := 0 to vwidth - 1 do
  begin
    lookL[x] := x;
    lookR[x] := x;
  end;

  for x := 0 to vwidth - 1 do
  begin
    if ((x mod Foversam) = 0) then // SPEEDUP for oversampled
pictures
    begin
      if (Y <= KK) or (Y >= (FDepthMap.Height + KK)) or
        ((X div Foversam) <= K) or ((X div Foversam) >=
(FDepthMap.Width + K))
        then
      begin
        hCol := 0;
      end
      else
      begin
        hCol := Trunc(((GetColor(FFastDepthMap, x div Foversam -
k, y - kk) / $00FFFFFF) * (DepthLevels - 1)));
      end;
      if FBlur = blurNone then
      begin
        Sep := SeparTable[hCol];
      end
      else
      begin
        featureZ := Trunc(maxdepth - hCol * (maxdepth - mindepth)
/ DepthLevels);
        sep := Round((veyeSep * featureZ) / (featureZ + obsDist) +
((random(10) - 5) / FBlurParam));
      end;
    end;

    left := x - sep div 2;
    right := left + sep;
    vis := TRUE;
    if ((left >= 0) and (right < vwidth)) then
    begin
      if (lookL[right] <> right) then // right pt yang telah
dihubungkan
      begin
        if (lookL[right] < left) then // kedalaman yang sekarang
        begin
          lookR[lookL[right]] := lookL[right]; // pemutusan
hubungan yang lama
          lookL[right] := right;
        end else vis := FALSE;
      end;
      if (lookR[left] <> left) then // left pt yang telah
dihubungkan
      begin
```

```
        if (lookR[left] > right) then // kedalaman yang sekarang
        begin
          lookL[lookR[left]] := lookR[left]; // pemutusan hubungan
yang lama
          lookR[left] := left;
        end else vis := FALSE;
      end;
      if (vis = TRUE) then
      begin
        lookL[right] := left;
        lookR[left] := right;
      end;
         // membuat hubungan
    end;
  end;

  lastlinked := -10; // nilai awal yang kosong
  for x := s to vwidth - 1 do
  begin
    if ((lookL[x] = x) or (lookL[x] < s)) then
    begin
      if (lastlinked = (x - 1)) then colour[x] := colour[x - 1]
      else
      begin
        colour[x] := GetColor(FFastMaskMap, ((x + poffset) mod
vmaxsep) div Foversam,
          (y + ((x - s) div vmaxsep) * yShift) mod patHeight);
      end
    end
    else
    begin
      colour[x] := colour[lookL[x]];
      lastlinked := x; // keep track of the last pixel to be
constrained
    end;
  end;

  lastlinked := -10; // dummy initial value
  for x := s - 1 downto 0 do
  begin
    if (lookR[x] = x) then
    begin
      if (lastlinked = (x + 1)) then colour[x] := colour[x + 1]
      else
      begin
        colour[x] := GetColor(FFastMaskMap, ((x + poffset) mod
vmaxsep) div Foversam,
          (y + ((s - x) div vmaxsep + 1) * yShift) mod patHeight);
      end
    end
    else
    begin
      colour[x] := colour[lookR[x]];
      lastlinked := x; // keep track of the last pixel to be
constrained
    end;
  end;
```

```
  x := 0;
  for xx := 0 to (vWidth div Foversam) - 1 do
  begin
    red := 0; green := 0; blue := 0;
   // use average colour of virtual pixels for screen pixel
    i := x;
    while (i < (x + Foversam)) do
    begin
      col := colour[i];
      red := red + col and $000000FF;
      green := green + (col and $0000FF00) shr 8;
      blue := blue + (col and $00FF0000) shr (16);
      i := i + 1;
    end;
    SetColor(FFastResultMap, x div Foversam, y,
      red div Foversam, green div Foversam, blue div Foversam);
    x := x + Foversam;
  end;
end;



procedure TStereoGram.PrepareRandomTexture;
var
  x, xx: Integer;
begin

  FFastMaskMap.Free;
  FFastMaskMap := TFastDIBOlej.Create;
  FFastMaskMap.SetSize(maxsep, FHeight, 24, 0);
  for x := 0 to FFastMaskMap.Width - 1 do
  begin
    for xx := 0 to FFastMaskMap.Height - 1 do
    begin
      if FRandomDotColor = dctBlackWhite then
      begin
        if random(100) > FDotIntensity then
        begin
          SetColor(FFastMaskMap, x, xx,
            255, 255, 255);
        end
        else
        begin
          SetColor(FFastMaskMap, x, xx,
            0, 0, 0);
        end;
      end
      else
      begin
        SetColor(FFastMaskMap, x, xx,
          Random(4) * 64 + 64, Random(4) * 64 + 64, Random(5) * 64
+ 64);
      end;
    end;
  end;
end;

procedure TStereoGram.GenerateStereogram;
var
```

```
  Y: Integer;

{Permulaan dari prosedur utama}
begin
//
  Progress := 0;
  Randomize;


  if (FType = sgtRandomDotTextured) then PrepareRandomTexture;
  FFastResultMap.Width := width * Foversam;
  FFastResultMap.Height := Height;

  for Y := 0 to Height - 1 do
  begin
    {Panghitungan garis untuk masing2 tipe }

    if (FType = sgtFastRandomDot) then MakeLineRandomDot(Y);
    if (FType = sgtTextured) or (FType = sgtRandomDotTextured)
then MakeLineTextured(Y);

    Application.ProcessMessages;
    {Progress}
    Progress := Trunc(Y / (Height / 100));
    if Assigned(FOnGenerate) then
      FOnGenerate(Self);
  end;
  Progress := 100;
  if Assigned(FOnGenerate) then
    FOnGenerate(Self);
end;

constructor TStereoGram.Create(AOwner: TComponent);
begin
  { Memanggil pembuatan metode dari parent class }
  inherited Create(AOwner);
  AutoInitialize;
end;

destructor TStereoGram.Destroy;
begin
  AutoDestroy;
  inherited Destroy;
end;


procedure TStereoGram.Loaded;
begin
  inherited Loaded;
end;


end.
```

```
unit Test1;

interface

uses
  Windows, Messages, SysUtils, Jpeg, Classes, Graphics, Controls,
Forms, Dialogs,
  StdCtrls, ExtCtrls, Gauges, ComCtrls, Stereogram;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Gauge1: TGauge;
    Bevel1: TBevel;
    RadioGroup1: TRadioGroup;
    RadioGroup4: TRadioGroup;
    Stereogram1: TStereogram;
    Bevel5: TBevel;
    Bevel6: TBevel;
    Label11: TLabel;
    Label12: TLabel;
    Image2: TImage;
    Image3: TImage;
    OpenDialog1: TOpenDialog;
    Bevel2: TBevel;
    Label1: TLabel;
    procedure LoadImage(Sender: TImage; FileName: string);
    procedure Button1Click(Sender: TObject);
    procedure Stereogram1Generate(Sender: TObject);
    procedure Image2Click(Sender: TObject);
    procedure Image3Click(Sender: TObject);
    procedure Zmena(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure SetSepar(Sender: TObject);
  private
    { Private declarations }
    LastType: Integer;
  public
    { Public declarations }
    procedure SetParams;
    procedure GetParams;
    procedure SetJPEGOptions(Sender: TImage);
  end;

Const
   OD:integer=14;
   Max3D:integer=14;
   Min3D:integer=8;
   Sep:integer=90;
   DPI:integer=72;
   W:integer=800;
   H:integer=500;
var
  Form1: TForm1;

implementation

uses Unit2t;
```

```
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  Gauge1.Progress:=0;
  Application.ProcessMessages;
  StereoGram1.generate;
  Form2.Width := StereoGram1.Width + 10;
  Form2.Height := StereoGram1.height + 30;
  Form2.Image1.Width := StereoGram1.Width;
  Form2.Image1.Height := StereoGram1.Height;
  Form2.Image1.Picture.Bitmap.Assign(StereoGram1.ResultMap);
  Form2.Show;
end;

procedure TForm1.Stereogram1Generate(Sender: TObject);
begin
  if Stereogram1.Progress - Gauge1.Progress > 3  then
    Gauge1.Progress := Stereogram1.Progress;
end;

procedure TForm1.Image2Click(Sender: TObject);
begin
  OpenDialog1.Filter := 'Image files (*.BMP,*.JPG)|*.BMP;*.JPG';
  OpenDialog1.Filterindex := 0;
  if OpenDialog1.Execute then
  begin
    LoadImage(Image2, OpenDialog1.FileName);
  end;
  IntToStr(Image2.Picture.Height);
  Stereogram1.DepthMap := Image2.Picture;
  GetParams;
end;



procedure TForm1.SetJPEGOptions(Sender: TImage);
var
  Temp: Boolean;
begin
  Temp := Sender.Picture.Graphic is TJPEGImage;
  if Temp then
    with TJPEGImage(Sender.Picture.Graphic) do
    begin
      PixelFormat := jf24Bit;
      Scale := jsFullSize;
      Grayscale := False;
      Performance := jpBestQuality;
      ProgressiveDisplay := False;
    end;
end;


procedure TForm1.LoadImage(Sender: TImage; FileName: string);
var
  II: TImage;
begin
  II := TImage.Create(nil);
  try
```

```
    try
      II.Picture.LoadFromFile(Filename);
    except
      on EInvalidGraphic do
        II.Picture.Graphic := nil;
    end;
    SetJPEGOptions(II);
    if II.Picture.Graphic <> nil then
    begin
      if not (II.Picture.Graphic is TJPEGImage) then
      begin
        Sender.Picture.Bitmap := II.Picture.Bitmap;
      end
      else
      begin
        Sender.Picture.Bitmap.PixelFormat := pf24bit;
        Sender.Picture.Bitmap.Width := II.Picture.Width;
        Sender.Picture.Bitmap.Height := II.Picture.Height;
        Sender.Canvas.Draw(0, 0, II.Picture.Graphic);
      end;
    end;
  finally
    II.Free;
  end;
end;

procedure TForm1.Image3Click(Sender: TObject);
begin
  OpenDialog1.Filter := 'Image files (*.BMP,*.JPG)|*.BMP;*.JPG';
  OpenDialog1.Filterindex := 0;
  if OpenDialog1.Execute then
  begin
    LoadImage(Image3, OpenDialog1.FileName);
  end;
    IntToStr(Image3.Picture.Height);
  Stereogram1.MaskMap := Image3.Picture;
  GetParams;
end;



procedure TForm1.SetParams;
begin
  try
    if RadioGroup1.ItemIndex = 0 then Stereogram1.StereoType :=
sgtFastRandomDot;
    if RadioGroup1.ItemIndex = 1 then Stereogram1.StereoType :=
sgtTextured;

    Stereogram1.DotIntensity :=50;
    Stereogram1.Oversampling := RadioGroup4.ItemIndex + 1;
    Stereogram1.ObservDistance :=OD;
    Stereogram1.Max3Ddepth :=Max3D;
    Stereogram1.Min3Ddepth :=Min3D;
    Stereogram1.DPI :=DPI;
    Stereogram1.Height :=H;
    Stereogram1.Width :=W;
  except
  end;
```

```
end;

procedure TForm1.GetParams;
begin

  if Stereogram1.StereoType = sgtFastRandomDot then
RadioGroup1.ItemIndex := 0;
  if Stereogram1.StereoType = sgtTextured then
RadioGroup1.ItemIndex := 1;

  RadioGroup4.ItemIndex := Stereogram1.Oversampling - 1;


  image2.Picture := Stereogram1.DepthMap;
  image3.Picture := Stereogram1.MaskMap;

    IntToStr(Image2.Picture.Height);
    IntToStr(Image3.Picture.Height);

end;


procedure TForm1.Zmena(Sender: TObject);
begin
  LastType := RadioGroup1.ItemIndex;
  SetParams;
  GetParams;
end;

procedure TForm1.FormShow(Sender: TObject);
begin
  LastType := -1;
  getParams;
end;

procedure TForm1.SetSepar(Sender: TObject);
begin
  if Stereogram1.Separation <> OD then
  begin
    Stereogram1.Separation := OD;
    GetParams;
  end;
end;

end.
```

```
unit Stereogram;

interface

uses WinTypes, WinProcs, Messages, SysUtils, Classes, Controls,
  Forms, Graphics, FastDIBOlej, ExtCtrls;


const

  MaxWidth = 1024; {lebar maksimum citra}
  MaxHeight = 800; {tinggi maksimum citra}
  MaxOversamp = 8;

  {Lebar maksimum * max oversampling}
  maxX = MaxWidth * MaxOversamp;

  DepthLevels = 256; {Dari 128 perubahan tidak tampak}

type


  TStereoGramType = (sgtFastRandomDot, sgtColoredDot,
sgtRandomDotTextured, sgtTextured);
  TBlurType = (blurNone, blurLight, blurMedium, blurHeavy);
  TRandomDotColorType = (dctBlackWhite, dctRGB);

  TStereogram = class(TComponent)
  private

    patHeight,
      xdpi,
      ydpi,
      yShift: Integer;

    lastlinked,
      i,
      vwidth,
      eyeSep,
      veyeSep,
      maxsep,
      vmaxsep,
      s, k, kk,
      poffset,
      featureZ,
      sep,
      left, right: Integer;

    vis: Boolean;
    red, green, blue: Integer;
    obsDist, mindepth, maxdepth: double;
    hCol: Integer;


    {Array untuk menyimpan nilai pixel}
    lookL,
      lookR: array[0..maxx] of word;

   {hasil color untuk garis}
```

```
    colour: array[0..maxx] of TCOLOR;

    SeparTable: array[0..DepthLevels] of Integer;

      { Tempat khusus dari TStereogram }
        { Penyimpanan untuk property DepthMap }
    FDepthMap: TPicture;
        { Untuk menyimpan property Height }
    FHeight: Integer;
        { Untuk menyimpan property MaskMap }
    FMaskMap: TPicture;
        { Untuk menyimpan property ResultMap }
    FResultMap: TPicture;
        { Untuk menyimpan property Separation }
    FSeparation: Integer;
        { Untuk menyimpan property Type }
    FType: TStereoGramType;
        { Untuk menyimpan property Width }
    FWidth: Integer;
        { Penunjuk dari aplikasi yang diatur oleh OnGenerater,
jika ada }
    FOnGenerate: TNotifyEvent;

    FDotIntensity: integer;
    FRandomDotColor: TRandomDotColorType;

    FobsDist: Double;

    Fmaxdepth: Double;

    Fmindepth: Double;

    FOversam: Integer;

    FBlur: TBlurType;
    FBlurParam: Integer;
    FDPI: Integer;

    {FastDIBs untuk akses cepat ke pixel}
    FFastDepthMap,
      FFastMaskMap,
      FFastResultMap: TFastDIBOlej;

    procedure AutoInitialize;
        { Metode yang digunakan dari beberapa objek yang di buat
secara bebas }
    procedure AutoDestroy;
        { Penulisan metode untuk property DepthMap }
    procedure SetDepthMap(Value: TPicture);
        { Pembacaan metode untuk property Height }
    function GetHeight: Integer;
        { Penulisan metode untuk property Height }
    procedure SetHeight(Value: Integer);
        { Penulisan metode untuk property MaskMap }
    procedure SetMaskMap(Value: TPicture);
        { Pembacaan metode untuk property Separation }
    function GetSeparation: Integer;
        { Penulisan metode untuk property Separation }
```

```
    procedure SetSeparation(Value: Integer);

    procedure SetDPI(Value: Integer);

    procedure Setoversam(Value: Integer);
        { Pembacaan metode untuk property Type }
    function GetType: TStereoGramType;
        { Penulisan metode untuk property Type }
    procedure SetType(Value: TStereoGramType);
        { Pembacaan metode untuk property Width }
    function GetWidth: Integer;
        { Penulisan metode untuk property Width }
    procedure SetWidth(Value: Integer);

    procedure SetDotIntensity(Value: Integer);
    procedure SetObsDist(Value: Double);
    procedure SetMaxdepth(Value: Double);
    procedure SetMindepth(Value: Double);
    procedure PrepareParams;
    procedure PrepareImages;
    procedure ReturnResulTPicture;
    function GetColor(FBMP: TFastDIBOlej; Col, Row: Integer):
TColor;
    procedure SetColor(FBMP: TFastDIBOlej; Col, Row: Integer; R,
G, B: Byte);

    procedure PrepareSeparTable;
    procedure PrepareRandomTexture;
    procedure GenerateStereogram;
    function ReadResultMap: TPicture;

    procedure MakeLineRandomDot(Row: Integer);
    procedure MakeLineTextured(Row: Integer);

  protected

      { metode yang terlindungi dari TStereoGram }
    procedure Loaded; override;

  public
    Progress: Integer; {Proses dalam persen}

    property ResultMap: TPicture read ReadResultMap; {Hasil yang
disimpan setelah Generate}
    property FastResultMap: TFastDIBOlej read FFastResultMap;
{Hasil yang disimpan setelah Generate}

      { Metode secara umum dari TStereoGram }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;


    {Generate dimulai - hasil dari bitmap telah disimpan ResultMap
dan FastResultMap}
    procedure Generate;

  published
      { Properti-properti yang dihasilkan dari TStereoGram }
```

```
    property OnGenerate: TNotifyEvent read FOnGenerate write
FOnGenerate;
    property DepthMap: TPicture read FDepthMap write SetDepthMap;
    property Height: Integer read GetHeight write SetHeight
default 600;
    property MaskMap: TPicture read FMaskMap write SetMaskMap;
    property Separation: Integer read GetSeparation write
SetSeparation default 90;
    property StereoType: TStereoGramType read GetType write
SetType;
    property Width: Integer read GetWidth write SetWidth default
800;
    property DotIntensity: Integer read FDotIntensity write
SetDotIntensity default 50;
    property DPI: Integer read FDPI write SetDPI default 72;
    property Blur: TBlurType read FBlur write FBlur;
    property RandomDotColor: TRandomDotColorType read
FRandomDotColor write FRandomDotColor;
    property Oversampling: Integer read Foversam write Setoversam;
    property ObservDistance: Double read FobsDist write
SetobsDist;
    property Max3Ddepth: Double read Fmaxdepth write Setmaxdepth;
    property Min3Ddepth: Double read Fmindepth write Setmindepth;

  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TStereogram]);
end;

{ Pengumpulan metode yang dibuat dari variabel dan nilai properti
pembuatan objek2}

procedure TStereoGram.AutoInitialize;
begin
  Progress := 0;
  FDepthMap := TPicture.Create;
  FMaskMap := TPicture.Create;
  FResultMap := TPicture.Create;
  FFastResultMap := TFastDIBOlej.Create;
  FHeight := 600;
  FWidth := 800;
  FDotIntensity := 50;
  FRandomDotColor := dctBlackWhite;
  FDPI := 72;
  FBlur := blurNone;
  Foversam := 1;
  FobsDist := 14;
  Fmaxdepth := 14;
  Fmindepth := 7;

  PrepareParams;

end; { of AutoInitialize }
```

```
{ Method to free any objects created by AutoInitialize }

procedure TStereoGram.AutoDestroy;
begin
  FDepthMap.Free;
  FMaskMap.Free;
  FResultMap.Free;
  FFastResultMap.Free;
end; { of AutoDestroy }


procedure TStereoGram.SetDepthMap(Value: TPicture);
begin
     { Use Assign method because TPicture is an object type }
  FDepthMap.Assign(Value);
  if (Width < DepthMap.Width) then Width := DepthMap.Width;
  if (Height < DepthMap.height) then Height := DepthMap.Height;
end;

{ Read method for property Height }

function TStereoGram.GetHeight: Integer;
begin
  Result := FHeight;
end;

{ Write method for property Height }

procedure TStereoGram.SetHeight(Value: Integer);
begin
  if value < 1 then FHeight := 600
  else
    if Value > MaxHeight then FHeight := MaxHeight
    else FHeight := Value;
  if FType = sgtColoredDot then
  begin
    FHeight := DepthMap.Height;
  end;

end;

{ Write method for property MaskMap }

procedure TStereoGram.SetMaskMap(Value: TPicture);
begin
  FMaskMap.Assign(Value);
end;

{ Read method for property Separation }
function TStereoGram.GetSeparation: Integer;
begin
  Result := FSeparation;
end;

{ Write method for property Separation }

procedure TStereoGram.SetSeparation(Value: Integer);
begin
```

```
  FMaxDepth := ((-1 * Value * FDPI * FobsDist) / (Value -
(Trunc(FDPI * 2.5))) + 1) / FDPI;
  Fmaxdepth := (Trunc(Fmaxdepth * 1000)) / 1000;
  if Fmindepth > FMaxDepth then Fmindepth := Fmaxdepth - 5;
  PrepareParams;
end;


{Prepare separation values to table used for faster resuls}
procedure TStereoGram.PrepareSeparTable;
var
  x: Integer;
  featureZ: Integer;
begin
  for x := 0 to DepthLevels - 1 do
  begin
    featureZ := Trunc(maxdepth - x * (maxdepth - mindepth) /
DepthLevels);
    SeparTable[x] := Round((veyeSep * featureZ) / (featureZ +
obsDist));
  end;
end;


procedure TStereoGram.Setoversam(Value: Integer);
begin
  if Value < 1 then FOversam := 1
  else
    if Value > MaxOversamp then FOversam := MaxOversamp
    else
      FOversam := Value;
  if (StereoType = sgtFastRandomDot) or (StereoType =
sgtColoredDot) then
    FOversam := 1;
end;


procedure TStereoGram.SetObsDist(Value: Double);
begin
  FobsDist := Value;
  PrepareParams;
end;

procedure TStereoGram.SetMaxdepth(Value: Double);
begin
  FmaxDepth := Value;
  PrepareParams;
end;

procedure TStereoGram.SetMindepth(Value: Double);
begin
  FminDepth := Value;
  PrepareParams;
end;



{ Membaca metode untuk property Type }
```

```
function TStereoGram.GetType: TStereoGramType;
begin
  Result := FType;
end;

{ Penulisan metode untuk property Type }

procedure TStereoGram.SetType(Value: TStereoGramType);
begin
  if Value = sgtFastRandomDot then FOversam := 1;
  if Value = sgtColoredDot then
  begin
    Width := DepthMap.Width;
    Height := DepthMap.Height;
  end;
  FType := Value;
end;



{ Membaca metode property Width }

function TStereoGram.GetWidth: Integer;
begin
  Result := FWidth;
end;

{ Penulisan metode untuk property Width }

procedure TStereoGram.SetWidth(Value: Integer);
begin
  if Value < 1 then FWidth := 600
  else if Value > MaxWidth then FWidth := MaxWidth else
    FWidth := Value;
  if FType = sgtColoredDot then
  begin
    FWidth := DepthMap.Width;
  end;
end;

procedure TStereoGram.SetDPI(Value: Integer);
begin
  if Value < 10 then FDPI := 10
  else FDPI := Value;

  if (FDPI > Width) then
  begin
    FDPI := Width div 2;
  end;
  if (FDPI > Height) then
  begin
    FDPI := Height div 2;
  end;
  PrepareParams;
end;

procedure TStereoGram.SetDotIntensity(Value: Integer);
begin
  if Value < 1 then FDotIntensity := 1
```

```
  else
    if Value > 99 then FDotIntensity := 99
    else FDotIntensity := Value;
end;

procedure TStereoGram.PrepareParams;
begin

  patHeight := FMaskMap.Height;
  xdpi := FDPI;
  ydpi := FDPI;
  yShift := ydpi div 16;
  vwidth := width * Foversam;
  obsDist := FDPI * FobsDist;
  eyeSep := Round(FDPI * 2.5);
  veyeSep := eyeSep * Foversam;
  maxdepth := FDPI * FMaxDepth;
  maxsep := Trunc(((eyeSep * maxdepth) / (maxdepth + obsDist)));



  if (maxSep > FMaskMap.Width) and
    (FMaskMap.Width > 0) and ((FType = sgtRandomDotTextured) or
(FType = sgtTextured))
    then maxSep := FMaskMap.Width;

  FSeparation := MaxSep;

  if FType = sgtRandomDotTextured then
    patHeight := MaxSep;

  vmaxsep := Foversam * maxsep;
  s := vwidth div 2 - vmaxsep div 2;//jarak pergeseran pixel//
  poffset := vmaxsep - (s mod vmaxsep);
//  mindepth := Trunc((sepfactor * maxdepth * obsdist) / ((1 -
sepfactor) * maxdepth + obsdist));
  mindepth := FDPI * FMinDepth;

  K := (Width - DepthMap.Width) div 2;
  KK := (Height - DepthMap.Height) div 2;


  {BLUR CONST}
  if FBlur = blurLight then FBlurParam := 15;
  if FBlur = blurMedium then FBlurParam := 10;
  if FBlur = blurHeavy then FBlurParam := 7;

end;


procedure TStereoGram.PrepareImages;
begin
  FFastResultMap.Free;
  FFastResultMap := TFastDIBOlej.Create;
  FFastDepthMap := TFastDIBOlej.Create;
  FFastMaskMap := TFastDIBOlej.Create;
  FFastResultMap.LoadFromHandle(HGDIOBJ(FResultMap.Bitmap.Handle),
24, 0);
```

```
  FFastDepthMap.LoadFromHandle(HGDIOBJ(FDepthMap.Bitmap.Handle),
24, 0);
  FFastMaskMap.LoadFromHandle(HGDIOBJ(FMaskMap.Bitmap.Handle), 24,
0);
end;

procedure TStereoGram.ReturnResulTPicture;
begin
  FFastDepthMap.Free;
  FFastMaskMap.Free;
end;

function TStereoGram.GetColor(FBMP: TFastDIBOlej; Col, Row:
Integer): TColor;
begin

  Result := (FBMP.Pixels[Row, Col].B shl 16) +
    (FBMP.Pixels[Row, Col].G shl 8) +
    (FBMP.Pixels[Row, Col].R);
end;

procedure TStereoGram.SetColor(FBMP: TFastDIBOlej; Col, Row:
Integer; R, G, B: Byte);
var
  FAF: TFColor;
begin
  Faf.r := R;
  Faf.g := G;
  Faf.b := B;
  FBMP.Pixels24[Row, Col] := Faf;
end;


procedure TStereoGram.Generate;
begin
  if (Progress > 0) and (Progress < 100) then Exit;
 {Test bitmaps}
  if FDepthMap.Bitmap.Empty then
    raise Exception.Create('DepthMap kosong, Isikan DephtMap
terlebih dahulu!');

  if (FType = sgtColoredDot) or (FType = sgtTextured) then
  begin
    if FMaskMap.Bitmap.Empty then
      raise Exception.Create('MaskMap Kosong');
  end;
  if (FType = sgtColoredDot) then
  begin
    if (FDepthMap.Width <> FMaskMap.Width) or
      (FDepthMap.Height <> FMaskMap.Height) then
      raise Exception.Create('Jika menggunakan ColourDot, MaskMap
dan DepthMap harus memiliki ukuran yang sama');
  end;
  if (FType = sgtColoredDot) then
  begin
    if (FDepthMap.Width <> Width) or
      (FDepthMap.Height <> Height) then
      raise Exception.Create('Jika menggunakan ColourDot, DepthMap
& citra hasil harus memiliki ukuran yang sama');
```

```
  end;
  ResultMap.Bitmap.Width := Width;
  ResultMap.Bitmap.Height := Height;
  ResultMap.Bitmap.Monochrome := False;
  {Langkah utama untuk menghasilkan Citra}
  PrepareParams;
  PrepareImages;
  PrepareSeparTable;
  GenerateStereogram;
  ReturnResulTPicture;
end;


function TStereoGram.ReadResultMap: TPicture;
begin
  FFastResultMap.Draw(FResultMap.Bitmap.Canvas.Handle, 0, 0);
  Result := FResultMap;
end;

procedure TStereoGram.MakeLineRandomDot(Row: Integer);
var
  x, y: Integer;
  kkk: Integer;
begin

  y := Row;
  for x := 0 to Width do LookL[x] := x;

  for x := 0 to Width - 1 do
  begin

    if (Y <= KK) or (Y >= (FDepthMap.Height + KK)) or
      ((X div FOversam) <= K) or ((X div FOversam) >=
(FDepthMap.Width + K))
      then
    begin
      hCol := 0;
    end
    else
    begin
      hCol := Trunc(((GetColor(FFastDepthMap, x div Foversam - k,
y - kk) / $00FFFFFF) * (DepthLevels - 1)));
    end;

    if FBlur = blurNone then
    begin
      Sep := SeparTable[hCol];
    end
    else
    begin
      featureZ := Trunc(maxdepth - hCol * (maxdepth - mindepth) /
DepthLevels);
      sep := Round((veyeSep * featureZ) / (featureZ + obsDist) +
((random(10) - 5) / FBlurParam));
    end;

    left := x - (sep div 2);
    right := left + sep;
    if (0 <= left) and (right < Width) then
```

```
    begin
      kkk := LookL[left];
      while (kkk <> left) and (kkk <> right) do
      begin
        if (kkk < right) then
        begin
          left := kkk;
        end
        else
        begin
          left := right;
          right := kkk;
        end;
        kkk := LookL[Left];
      end;
      LookL[left] := right;
    end;
  end;

  for x := Width - 1 downto 0 do
  begin
    if (LookL[x] = x) then
    begin
      if FRandomDotColor = dctBlackWhite then
      begin
        if random(100) > FDotIntensity then
        begin
          SetColor(FFastResultMap, x, y,
            255, 255, 255);
        end
        else
        begin
          SetColor(FFastResultMap, x, y,
            0, 0, 0);
        end;
      end
      else
      begin
        SetColor(FFastResultMap, x, y,
          Random(4) * 64 + 64, Random(4) * 64 + 64, Random(4) * 64
+ 64);
      end;

    end
    else
    begin
      FFastResultMap.Pixels[y, x] := FFastResultMap.Pixels[y,
LookL[x]];
    end;
  end;
end;


procedure TStereoGram.MakeLineTextured(Row: Integer);
var
  x, xx, Y: integer;
  col: TColor;
begin
```

```
  y := Row;

  for x := 0 to vwidth - 1 do
  begin
    lookL[x] := x;
    lookR[x] := x;
  end;

  for x := 0 to vwidth - 1 do
  begin
    if ((x mod Foversam) = 0) then // SPEEDUP for oversampled
pictures
    begin
      if (Y <= KK) or (Y >= (FDepthMap.Height + KK)) or
        ((X div Foversam) <= K) or ((X div Foversam) >=
(FDepthMap.Width + K))
        then
      begin
        hCol := 0;
      end
      else
      begin
        hCol := Trunc(((GetColor(FFastDepthMap, x div Foversam -
k, y - kk) / $00FFFFFF) * (DepthLevels - 1)));
      end;
      if FBlur = blurNone then
      begin
        Sep := SeparTable[hCol];
      end
      else
      begin
        featureZ := Trunc(maxdepth - hCol * (maxdepth - mindepth)
/ DepthLevels);
        sep := Round((veyeSep * featureZ) / (featureZ + obsDist) +
((random(10) - 5) / FBlurParam));
      end;
    end;

    left := x - sep div 2;
    right := left + sep;
    vis := TRUE;
    if ((left >= 0) and (right < vwidth)) then
    begin
      if (lookL[right] <> right) then // right pt yang telah
dihubungkan
      begin
        if (lookL[right] < left) then // kedalaman yang sekarang
        begin
          lookR[lookL[right]] := lookL[right]; // pemutusan
hubungan yang lama
          lookL[right] := right;
        end else vis := FALSE;
      end;
      if (lookR[left] <> left) then // left pt yang telah
dihubungkan
      begin
        if (lookR[left] > right) then // kedalaman yang sekarang
        begin
```

```
          lookL[lookR[left]] := lookR[left]; // pemutusan hubungan
yang lama
          lookR[left] := left;
        end else vis := FALSE;
      end;
      if (vis = TRUE) then
      begin
        lookL[right] := left;
        lookR[left] := right;
      end;
         // membuat hubungan
    end;
  end;

  lastlinked := -10; // nilai awal yang kosong
  for x := s to vwidth - 1 do
  begin
    if ((lookL[x] = x) or (lookL[x] < s)) then
    begin
      if (lastlinked = (x - 1)) then colour[x] := colour[x - 1]
      else
      begin
        colour[x] := GetColor(FFastMaskMap, ((x + poffset) mod
vmaxsep) div Foversam,
          (y + ((x - s) div vmaxsep) * yShift) mod patHeight);
      end
    end
    else
    begin
      colour[x] := colour[lookL[x]];
      lastlinked := x; // keep track of the last pixel to be
constrained
    end;
  end;

  lastlinked := -10; // dummy initial value
  for x := s - 1 downto 0 do
  begin
    if (lookR[x] = x) then
    begin
      if (lastlinked = (x + 1)) then colour[x] := colour[x + 1]
      else
      begin
        colour[x] := GetColor(FFastMaskMap, ((x + poffset) mod
vmaxsep) div Foversam,
          (y + ((s - x) div vmaxsep + 1) * yShift) mod patHeight);
      end
    end
    else
    begin
      colour[x] := colour[lookR[x]];
      lastlinked := x; // keep track of the last pixel to be
constrained
    end;
  end;


  x := 0;
  for xx := 0 to (vWidth div Foversam) - 1 do
```

```
  begin
    red := 0; green := 0; blue := 0;
   // use average colour of virtual pixels for screen pixel
    i := x;
    while (i < (x + Foversam)) do
    begin
      col := colour[i];
      red := red + col and $000000FF;
      green := green + (col and $0000FF00) shr 8;
      blue := blue + (col and $00FF0000) shr (16);
      i := i + 1;
    end;
    SetColor(FFastResultMap, x div Foversam, y,
      red div Foversam, green div Foversam, blue div Foversam);
    x := x + Foversam;
  end;
end;



procedure TStereoGram.PrepareRandomTexture;
var
  x, xx: Integer;
begin

  FFastMaskMap.Free;
  FFastMaskMap := TFastDIBOlej.Create;
  FFastMaskMap.SetSize(maxsep, FHeight, 24, 0);
  for x := 0 to FFastMaskMap.Width - 1 do
  begin
    for xx := 0 to FFastMaskMap.Height - 1 do
    begin
      if FRandomDotColor = dctBlackWhite then
      begin
        if random(100) > FDotIntensity then
        begin
          SetColor(FFastMaskMap, x, xx,
            255, 255, 255);
        end
        else
        begin
          SetColor(FFastMaskMap, x, xx,
            0, 0, 0);
        end;
      end
      else
      begin
        SetColor(FFastMaskMap, x, xx,
          Random(4) * 64 + 64, Random(4) * 64 + 64, Random(5) * 64
+ 64);
      end;
    end;
  end;
end;

procedure TStereoGram.GenerateStereogram;
var
  Y: Integer;
```

```
{Permulaan dari prosedur utama}
begin
//
  Progress := 0;
  Randomize;


  if (FType = sgtRandomDotTextured) then PrepareRandomTexture;
  FFastResultMap.Width := width * Foversam;
  FFastResultMap.Height := Height;

  for Y := 0 to Height - 1 do
  begin
    {Panghitungan garis untuk masing2 tipe }

    if (FType = sgtFastRandomDot) then MakeLineRandomDot(Y);
    if (FType = sgtTextured) or (FType = sgtRandomDotTextured)
then MakeLineTextured(Y);

    Application.ProcessMessages;
    {Progress}
    Progress := Trunc(Y / (Height / 100));
    if Assigned(FOnGenerate) then
      FOnGenerate(Self);
  end;
  Progress := 100;
  if Assigned(FOnGenerate) then
    FOnGenerate(Self);
end;

constructor TStereoGram.Create(AOwner: TComponent);
begin
  { Memanggil pembuatan metode dari parent class }
  inherited Create(AOwner);
  AutoInitialize;
end;

destructor TStereoGram.Destroy;
begin
  AutoDestroy;
  inherited Destroy;
end;


procedure TStereoGram.Loaded;
begin
  inherited Loaded;
end;


end.
```