**TUGAS BESAR**

**GRAFIKA KOMPUTER**

**“MEMBANGUN OBJEK PANTAI 3D”**

***Diajukan untuk memenuhi***

***Tugas Mata Kuliah Grafika Komputer***

***Program Strata Satu Jurusan Teknik Informatika***

***Fakultas Teknik dan Ilmu Komputer***

***Universitas Komputer Indonesia***

**OLEH :**

**KELOMPOK 6**

**Andrio Wiliam FR 10109369**

**Rahmat Siahaan 10109386**

**Adhadiat Satya N 10109403**

****

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK DAN ILMU KOMPUTER**

**UNIVERSITAS KOMPUTER INDONESIA**

**BANDUNG**

**2013**

DAFTAR ISI

[DAFTAR ISI 1](#_Toc362473617)

[BAB I PENDAHULUAN 1](#_Toc362473618)

[I.1 Latar Belakang Masalah 1](#_Toc362473619)

[I.2 Identifikasi Masalah 2](#_Toc362473620)

[I.3 Maksud dan Tujuan 2](#_Toc362473621)

[I.4 OBJEK PEMBANGUNAN 2](#_Toc362473622)

[BAB II LANDASAN TEORI 4](#_Toc362473623)

[II.1 Pengertian Grafika Komputer 4](#_Toc362473624)

[II.2 Elemen – Elemen Dasar Grafika Komputer 5](#_Toc362473625)

[II.3 Grafik Komputer 2D 5](#_Toc362473626)

[II.4 Grafik Komputer 3D 6](#_Toc362473627)

[II.5 Perbedaan Grafik 2D dan Grafik 3D 6](#_Toc362473628)

[II.6 Transformasi 6](#_Toc362473629)

[II.6.1 Jenis jenis dari transformasi dasar: 7](#_Toc362473630)

[II.7 *Color* (Warna) 9](#_Toc362473631)

[II.7.2 Pembagian Warna 9](#_Toc362473632)

[II.7.3 Lighting (pencahayaan) 10](#_Toc362473633)

[II.7.4 Bayangan 11](#_Toc362473634)

[II.7.5 Blending (Pencampuran) 11](#_Toc362473635)

[II.7.6 Texture Mapping 12](#_Toc362473636)

[II.7.7 Pengertian OpenGl 14](#_Toc362473637)

[II.7.8 Dev C++ 14](#_Toc362473638)

[BAB III LANGKAH PEMBUATAN DAN IMPLEMENTASI 16](#_Toc362473639)

[III.1 Langkah Pembuatan 16](#_Toc362473640)

[III.2 Perangkat Lunak Pendukung 16](#_Toc362473641)

[III.3 Perangkat Keras Pendukung 16](#_Toc362473642)

[III.4 Objeck List Yang terealisasi 17](#_Toc362473643)

[III.5 Source Code Program OpenGL 17](#_Toc362473644)

[III.6 Tampilan Program OpenGL 60](#_Toc362473645)

[BAB IV KESIMPULAN DAN SARAN 61](#_Toc362473646)

[IV.1 Kesimpulan 61](#_Toc362473647)

[IV.2 Saran 61](#_Toc362473648)

# PENDAHULUAN

## Latar Belakang Masalah

Pantai adalah daerah di tepi perairan yang dipengaruhi oleh air pasang tertinggi

dan air surut terendah (Bambang Triatmojo, *“Teknik Pantai”*). Garis pantai adalah garis batas pertemuan antara daratan dan air laut, dimana posisinya tidak tetap dan dapat berubah sesuai dengan pasang surut air laut dan erosi pantai yang terjadi. Perubahan garis pantai disebabkan oleh faktor alam dan/atau faktor manusia. Faktor alam diantaranya gelombang laut, arus laut, angin, sedimentasi sungai, kondisi tumbuhan pantai serta aktivitas tektonik dan vulkanik. Sedangkan faktor manusia antara lain pembangunan pelabuhan dan fasilitas-fasilitasnya (misalnya breakwater), pertambangan, pengerukan, perusakan vegetasi pantai, pertambakan, perlindungan pantai serta reklamasi pantai.

Pantai selalu menyesuaikan bentuk profilnya sedemikian sehingga mampu menghancurkan energi gelombang yang datang. Penyesuaian bentuk tersebut merupakan tanggapan dinamis alami terhadap laut. Proses dinamis pantai sangat dipengaruhi oleh *littoral transport*, yang didefinisikan sebagai gerak sedimen di daerah dekat pantai (*nearshore zone*) oleh gelombang dan arus. *Littoral transport* dapat dibedakan menjadi dua macam yaitu transpor sepanjang pantai (*longshore transport*) dan transpor tegak lurus pantai *(onshore-offshore transport)*. Material pasir yang ditranspor disebut dengan *littoral drift*. Transpor tegak lurus pantai terutama ditentukan oleh kemiringan gelombang, ukuran sedimen dan kemiringan pantai. Pada umumnya gelombang dengan kemiringan besar menggerakkan material kearah laut (abrasi), dan gelombang kecil dengan periode panjang menggerakkan material kearah darat (akresi).

Untuk itu kelompok kami akan membangun Pantai dengan matahari sunset di sore hari, dengan keadaan langit yang sedikit berawan, di pesisir pantai terdapat gedung mercusuar untuk memantau para perahu nelayan yang berlayar dimalam hari, terdapat touris yang sedang bersantai diatas pasir pantai, adapula yang sedang duduk santai di kursi santai sambil dinaungi payung sambil meminum minuman segar dari pedagang-pedagang disekitar pantai.pengembangannya menggunakan OpenGL.

Untuk pembuatan tugas besar kali ini, penulis menyertakan beberapa objek dalam pembuatannya, diantaranya :

1. Adanya objek utama yaitu Pantai
2. Pasir
3. Awan
4. Matahari
5. Kursi Santai
6. Baling-baling
7. Payung

## Identifikasi Masalah

Berdasarkan uraian latar belakang diatas dapat diidentifikasikan masalah yaitu :

1. Bagaimana cara memodelkan pantai dengan tammpilan 3D dengan menggunakan OpenGL
2. Memodelkan pantai sehingga dapat di ilustrasiakan dalam surcode program OpenGL.

## Maksud dan Tujuan

Berdasarkan permasalahan yang diteliti, maka maksud dari penulisan tugas besar ini adalah untuk membangun Objeck Pantai 3D. Sedangkan tujuan yang akan dicapai adalah :

1. Membangun aplikasi untuk memodelkan suatu tempat denganmelihat dari model paketnya.
2. Membangun suatu pemodelan untuk memodelkan dengan perintah-perintah yang sudah ada.
3. Mengaplikasikan materi openGL dari mata kuliah Grafika Komputer.

## OBJEK PEMBANGUNAN

1. Pantai, terdiri dari garis-garis dan lekukan lekuan
2. Pasir, terdiri dari garis-garis dan lekukan lekukan
3. Awan, terdiri dari gelombang
4. Matahari sunset, terdiri dari lingkaran
5. Mercusuar, terdiri dari tabung, dan kubus
6. Payung, terdiri dari tabung dan kerucut atau lingkaran
7. Kursi santai, terdiri dari persegi

## Kontirbusi Anggota Kelompok

1. Andrio :

* mendesain gambar untuk terain pasir dan pantai
* membuat net voli
* membuat kincir angin
* membuat awan
* Membuat meja

1. Rahmat Siahaan :

* Membuat pohon
* Membuat batu
* Membuat kursi santai
* Membuat Laporan

1. Adhadiat Satya Nugraha :

* Membuat bola voli
* Membuat matahari
* Membuat payung
* Membuat kursi santai

# LANDASAN TEORI

## Pengertian Grafika Komputer

Grafika komputer (*Computer graphics*) adalah bagian dari [ilmu komputer](http://id.wikipedia.org/wiki/Ilmu_komputer) yang berkaitan dengan pembuatan dan manipulasi gambar secara digital. Bentuk sederhana dari grafika komputer adalah [grafika komputer 2D](http://id.wikipedia.org/w/index.php?title=Grafika_komputer_2D&action=edit&redlink=1) yang kemudian berkembang menjadi [grafika komputer 3D](http://id.wikipedia.org/wiki/Grafika_komputer_3D), [pemrosesan citra](http://id.wikipedia.org/w/index.php?title=Pemrosesan_citra&action=edit&redlink=1) *(image processing),* dan [pengenalan pola](http://id.wikipedia.org/wiki/Pengenalan_pola) *(pattern recognition).* Grafika komputer sering dikenal juga dengan istilah [visualisasi](http://id.wikipedia.org/wiki/Visualisasi) data.

Bagian dari grafika komputer meliputi:

1. [Geometri](http://id.wikipedia.org/wiki/Geometri): mempelajari cara menggambarkan permukaan bidang
2. [Animasi](http://id.wikipedia.org/wiki/Animasi): mempelajari cara menggambarkan dan memanipulasi gerakan
3. [Rendering](http://id.wikipedia.org/w/index.php?title=Rendering&action=edit&redlink=1): mempelajari [algoritma](http://id.wikipedia.org/wiki/Algoritma) untuk menampilkan efek cahaya
4. [Citra](http://id.wikipedia.org/wiki/Citra) (Imaging): mempelajari cara pengambilan dan penyuntingan gambar.

Teknik-teknik yang dipelajari dalam grafika komputer adalah teknik-teknik bagaimana membuat atau menciptakan gambar menggunakan komputer. Ada perbedaan yang sangat mendasar antara foto dan gambar, yaitu pada foto semua detail obyek terlihat sedangkan pada gambar (baik itu gambar manusia atau gambar komputer) tidak dapat memperlihatkan semua detail yang ada tetapi hanya detail-detail yang dianggap penting dalam menunjukkan pola suatu gambar.

## Elemen – Elemen Dasar Grafika Komputer

Ada beberapa elemen dasar dari grafika komputer antara lain:

1. Point

*Point* adalah sebuah titik yang digunakan untuk membangun obyek. Setiap titik dalam obyek 3 dimensi memiliki nilai dalam x, y dan z.

1. Polyline

*Polyline* adalah sebuah fungsi yang dibentuk dari beberapa garis yang saling berhubungan dan membentuk sebuh kurva yang terbuka.

1. Polygon

*Polygon* adalah suatu fungsi yang mirip dengan polyline hanya saja hasilnya adalah kurva tertutup, sedangkan *polyline* hasilnya kurva terbuka.

1. Filled Polygon (Face)

*Filled Polygon* adalah sebuah *polygon* yang bagian dalamnya diwarnai atau dipenuhi dengan sebuah warna tertentu. *Filled polygon* biasanya digunakan sebagai *face* dari pembentukan obyek–obyek 3 Dimensi.

1. Gradate Polygon

*Gradate polygon* adalah sebuah *polygon* yang bagian dalamnya memiliki warna – warna yang bergradasi dari satu warna ke warna yang lainnya.

## Grafik Komputer 2D

Grafik komputer 2D adalah pembuatan objek gambar dengan menggunakan 2 titik sebagai acuannya yaitu sumbu x dan y. Grafik 2D ini dapat digunakan dalam berbagai aplikasi yang pada awalnya dikembangkan pada teknologi cetak tradisional dan gambar, seperti tipografi, kartografi, gambar teknik, iklan, dan lain-lain.

Grafik komputer 2D ini merupakan langkah paling awal dalam membentuk model objek yang akan dibangun dalam grafik komputer 3D. Dalam aplikasi, gambar dua dimensi adalah bukan hanya representasi dari objek dunia nyata, tetapi sebuah artefak independen dengan nilai tambah semantik. Keseluruhan obyek 2D dapat dimasukkan dengan jumlah lebih dari satu, model yang akan dibentuk sesuai dengan kebutuhan. Tahap rekayasa hasil obyek 2D dapat dilakukan dengan aplikasi program grafis seperti Adobe Photoshop, Corel Draw, dan lain sebagainya.

## Grafik Komputer 3D

Grafik komputer 3D merupakan representasi dari data geometrik 3 dimensi sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafik komputer 2D. hasilnya dapat ditampilkan secara real time untuk keperluan simulasi. Prinsip yang dipakai mirip dengan grafik komputer 2D dalam penggunaan algoritma, grafika vektor, model frame kawat (wire frame model), dan grafik rasternya.

Grafik komputer 3D sering disebut sebagai model 3D. Namun, model 3D ini lebih menekankan pada representasi matematis untuk objek 3 dimensi. Obyek pada grafik 3D adalah sekumpulan titik-titik 3D (x,y,z) yang membentuk suatu face (bidang) yang digabungkan menjadi satu kesatuan. Face sendiri adalah gabungan titik-titik yang membentuk bidang tertentu. Data matematis ini belum bisa dikatakan sebagai gambar grafis hingga saat ditampilkan secara visual pada layar komputer atau printer. Proses penampilan suatu model matematis ke bentuk citra 2 D biasanya dikenal dengan proses 3D rendering.

## Perbedaan Grafik 2D dan Grafik 3D

Perbedaan yang paling mendasar dan terlihat dengan sangat jelas adalah tampilan gambarnya. Gambar 2D tampil flat adn frame tampilannya cenderung terbatas karena objek gambarnya disajikan hanya dengan sumbu x dan y. Sedangkan pada grafik 3D, gambar yang ditampilkan lebih hidup, membentuk ruang, tidak flat, serta framenya lebih luas yang dikarenakan gambar 3D disajikan dengan 3 sumbu, yaitu x, y, dan z.

## Transformasi

Transformasi dasar pada objek dua dimensi yang pertama adalah translasi (*translation*). Translasi berarti memindahkan suatu objek sepanjang garis lurus dari suatu lokasi koordinat tertentu ke lokasi yang lain. Transformasi skala (*scaling)* digunakan untuk mengubah ukuran suatu objek, sedangkan rotasi (*rotation)* adalah pemindahan objek menurut garis melingkar.

### Jenis jenis dari transformasi dasar:

1. Translasi

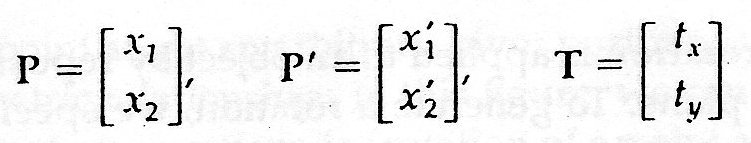
Translasi dilakukan dengan penambahan translasi pada suatu titik koordinat dengan translasi vektor atau shift vektor, yaitu (tx,ty), dimana tx adalah translation vektor menurut sumbu x, sedangkan ty adalah translation vektor menurut sumbu y. koordinat baru titik yang ditranslasi dapat diperoleh dengan

X’= x + tx

Y’= y + ty

Dimana (x,y) adalah koordinat asal suatu objek dan (x’ , y’) adalah koordinat baru objek tersebut setelah ditranslasi.

Kadang-kadang transformasi dinyatakan dalam bentuk matriks, sehingga matriks tranformasi untuk translasi dapat dinyatakan sebagai berikut :



Dengan demikian translasi dua dimensi dapat ditulis dalam bentuk matriks :

|  |
| --- |
| P’ = P + T |

Disamping dinyatakan dalam vektor kolom, matriks transformasi dapat dituliskan dalam bentuk vektor baris, sehingga menjadi P = [ x y ] dan T = [ tx ty ]. Bentuk vektor kolom adalah standar dari symbol matematik, yang juga berlaku bagi notasi grafik seperti GKS dan PHIGS.

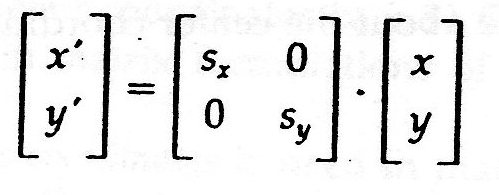
1. Skala

Transformasi skala adalah perubahan ukuran suatu objek. Koordinat baru dapat diperoleh dengan melakukan perkalian nilai koordinat dengan scaling factor, yaitu (sx , sy) ,dimana sx adalah scaling factor menurut sumbu x, sedangkan sy adalah scaling factor menurut sumbu y. koordinat baru titik yang diskala dapat diperoleh dengan

X’= x + sx

Y’ = y + sy

Dimana (x , y) adalah koordinat asal suatu objek dan (x’,y’) adalah koordinat setelah diskala. Matriks transformasi untuk skala dapat dinyatakan sebagai berikut:



Dengan demikian skala dapat juga dituliskan

P’ = S . P

Scaling factor sx dan sy dapat diberikan sembarang nilai positif. Nilai lebih dari 1 menyebabkan objek diperbesar, sebaliknya bila nilai lebih kecil dari 1, maka objek akan diperkecil. Bila sx dan sy mempunyai nilai yang sama, maka skala disebut uniform scaling. Nilai yang tidak sama dari sx dan sy menghasilkan differential scaling, yang biasa digunakan pada program aplikasi.

1. Rotasi

Rotasi dua dimensi pada suatu objek kan memindahkan objek tersebut menurut garis melingkar. Pada bidang xy. Untuk melakukan rotasi diperlukan sudut rotasi θ dan pivot point (xp’ yp ) atau rotasi point dimana objek di rotasi, seperti pada gambar 5-3 nilai positif dari sudut rotasi menentukan arah rotasi berlawanan dengan jarum jam, dan sebaliknya nilai negative akan memutar objek searah jarum jam.

Rotasi dapat dilakukan dengan pivot point yaitu titik pusat koordinat, seperti pada gambar 5-4. Pada betuk ini, r adalah jarak konstan dari titik pusat, sudut φ adalah sudut posisi suatu titik dengan sumbu horizontal, sedangkan θ adalah sudut rotasi. Menggunakan trigonometri, transformasi dapat dinyatakan dengan sudut θ dan φ sebagai berikut:

X’= r cos (φ + θ) = r cos φ cos θ – r sin φ sin θ

y’ = r sin (φ + θ) = r cos φ sin θ + r sin φ cos θ

sedangkan dengan koordinat polar diketahui bahwa

x = r cos φ, y = r sin φ

dengan melakukan substitusi, diperoleh rumus transformasi untuk rotasi suatu titik (x, y) dengan sudut rotasi θ sebagai berikut:

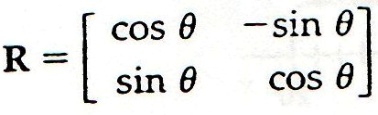
x’ = x cos θ – y sin θ

y’ = x sin θ – y cos θ

matriks transformasi untuk rotasi dapat dinyatakan sebagai berikut:

P’ = R . P

Rotasi dapat dinyatakan dalam bentuk lain, yaitu matriks. Matriks rotasi dapat dituliskan dengan



Rotasi suatu titik terhadap pivot point (xp’ yp ) seperti pada gambar 5-5, menggunakan bentuk trigonometri, secara umum dapat dituliskan sebagai berikut:

X’ = xp +(x - xp) cos θ – (y - yp) sin θ

Y’ = yp + (x – xp) sin θ + (y – yp) cos θ

## *Color* (Warna)

Bentuk gelombang elektromagnetik yang terkandung dalam cahaya yang berasal dari sumber cahaya. Spectrum warna memiliki panjang gelombang elektomagnetik antara 350-750 nanometer .

### Pembagian Warna

RGB (Red-Green-Blue) : warna dasar yang dijadikan patokan warna secara universal (primary colors)

CMYK (Cyan-Magenta-Yellow-Black) : Sistem representasi pada warna tinta yang diterapkan dalam dunia fotografi dan produksi grafika

Sistem Warna Lingkaran (the color wheel) : Digunakan untuk mengkombinasikan dan mengharmonikan warna pada karya seni dan design

Warna dapat didefinisikan secara obyektif/fisik sebagai sifat cahaya yang dipancarkan, atau secara subyektif/psikologis sebagai bagian dari pengalaman indera pengelihatan. Secara obyektif atau fisik, warna dapat diberikan oleh panajang gelombang. Dilihat dari panjang gelombang, cahaya yang tampak oleh mata merupakan salah satu bentuk pancaran energi yang merupakan bagian yang sempit dari gelombang elektromagnetik.

Dari sekian banyak warna, dapat dibagi dalam beberapa bagian yang sering dinamakan dengan sistem warna Prang System yang ditemukan oleh Louis Prang pada 1876 meliputi :

1. Hue, adalah istilah yang digunakan untuk menunjukkan nama dari suatu warna, seperti merah, biru, hijau dsb.
2. Value, adalah dimensi kedua atau mengenai terang gelapnya warna. Contohnya adalah tingkatan warna dari putih hingga hitam.
3. Intensity, seringkali disebut dengan chroma, adalah dimensi yang berhubungan dengan cerah atau suramnya warna.

### Lighting (pencahayaan)

Lighting merupakan proses menghitung intensitas cahaya terutama pada 3-Dimensi point, biasanya diatas suatu permukaan.

Beberapa cara mengatasi masalah pencahayaan, antara lain :

1. Mengerti persepsi dari cahaya (warna)
2. Membuat sebuah solusi untuk merepresentasikan dan menghasilkan warna menggunakan komputer.
3. Mengerti akan pengaruh cahaya dan objek

### Bayangan

Bayangan akan muncul saat cahaya jatuh menyinari suatu objek.

Pada dunia maya, layaknya cahaya, terdapat beberapa jenis bayangan yang dapat dihasilkan oleh komputer.

Bayangan bekerja sama dengan cahaya untuk memberi kesan natural atau realistic pada scene yang ada. Bayangan dapat membantu mendefinisikan posisi objek-objek, apakah berada di lantai atau melayang di udara.Bayangan yang dihasilkan bisa tajam dan solid namun bisa juga lembut dan buram (blurry).Keberadaan bayangan atau ketiadaannya dapat digunakan untuk memberi keseimbangan dan kontras pada objek-objek di dalam scene.

### Blending (Pencampuran)

Pencampuran merupakan fungsi yang menggabungkan nilai warna dari sumber dan tujuan. Operasi campuran yaitu cara yang paling alami untuk mengetahui bahwa komponen RGB adalah suatu fragmen yang mewakili warna dan komponen alfa adalah suatu fragmen yang mewakili sifat tidak tembus cahaya.

1. Faktor sumber dan tujuan

Pada proses pencampuran, nilai cairan warna yang masuk fragmen (sumber) digabungkan dengan warna yang sesuai dengan nilai saat ini yang disimpan pada piksel (tujuan) dalam dua tahap proses. Yang pertama menghitung faktor sumber dan tujuan, factor-faktor tersebut adalah RGBA quadruplets yang masing-masing dikalian dengan komponen-komponen R, G, B dan nilai-nilai dari sumber dan tujuan. Kemudian komponen yang sesuai dalam dua set RGBA quadruplets. Secara sistematis, faktor sumber dan tujuan pencampuran (SR, Sg, Sb, Sa) dan (Dr, Dg, dB, Da) dan nilai RGBA ditandai dengan s atau d dan terakhir nilai RGBA dicampurkan yang diperoleh dengan (RsSr + RdDr, GsSg + GdDg, BsSb + BdDb, AsSa + Adda) dimana setiap komponen adalah quadruplets is eventually clamped to [0,1].

Dengan menggunakan glBlendFunc () untuk persediaan pada dua hal utama, yang pertama menentukan bagaimana faktor sumber dan tujuan harus dihitung dan yang kedua menunjukan bagaimana faktor sumber dan tujuan dihitung. Dan untuk proses pencampurannya harus ada faktor pengaktifannya menggunakan : glEnable (GL\_BLEND). Menggunakan glDisable () dengan GL\_BLEND untuk menonaktifkan Pencampuran dan menggunakan konstan GL\_ONE (sumber) dan GL\_ZERO (tujuan) memberikan hasil yang sama seperti ketika Pencampuran dinonaktifkan. Nilai-nilai ini bersifat default dengan void glBlendFunc (GLenum *sfactor,* GLenum *dfactor)*.

Mengontrol bagaimana nilai warna dalam fragmen yang diproses digabungkan dengan yang sudah disimpan dalam framebuffer (tujuan).Pendapat *sfactor* menunjukkan bagaimana untuk menghitung faktor sumber Pencampuran dan *dfactor* menunjukkan bagaimana untuk menghitung faktor tujuan Pencampuran. Campuran faktor yang diasumsikan terletak pada rentang [0,1]; setelah nilai warna dalam sumber dan tujuan digabungkan, setelah dihitung kisaran [0,1].

### Texture Mapping

Texture mapping merupakan teknik pemetaan sebuah tekstur pada pola gambar wireframe, dimana wireframe yang telah dibuat akan ditampilkan memiliki kulit luar seperti tekstur yang diinginkan. Dalam pemberian tekstur, perlu diperhatikan dasarnya seperti:

1. Menentukan tekstur
2. Membaca atau membangkitkan tekstur
3. Menandai tekstur
4. Mengenablekan tekstur
5. Menandai koordinat tekstur pada vertek
6. Menentukan parameter tekstur
7. Wrapping , filtering, dsb.

Langkah-langkah dalam memulai mapping sebuah tekstur yakni dengan spesifikasi dibawah ini :

1. Menentukan Tekstur Image
2. Mendefinisikan tekstur image dari sebuah array teksel (element tekstur ) ke dalam memory cpu : Glubyte my\_texels[512][512];
3. Mendefinisikan seperti semua peta piksel yang lain
4. Gambar yang didefinisikan (baik secara manual maupun dengn suatu fungsi matematik tertentu).
5. Membangkitkan dengan kode aplikasi
6. Mengenablekan tekstur mapping

glEnable(GL\_TEXTURE\_2D)

OpenGL mendukung 1 sampai 4 dimensional tekstur mapping

Mendefinisikan gambar sebagai sebuah tekstur

glTexImage2D (target, level, components, w, h, border, format, type, texels );

Keterangan :

target: tipe dari teksture, e.g. GL\_TEXTURE\_2D

level: digunakan untuk *mipmapping*

components: element per texel

w, h: lebar dan tinggi dari texels pada pixels

border: digunakan untuk smoothing

format and type: menjelaskan texels

texels: pointer ke array texel

glTexImage2D(GL\_TEXTURE\_2D, 0, 3, 512, 512, 0,GL\_RGB, GL\_UNSIGNED\_BYTE, my\_texels);

Mengubah gambar tekstur

OpenGL meminta dimensi tekstur untuk menjadi dasar dari 2

Jika dimensi dari image bukan power ke 2, gluScaleImage (format,w\_in,h\_in,type\_in,\*data\_in,w\_out,h\_out,type\_out,\*data\_out);

data\_in adalah gambar inputan.

data\_out adalah gambar hasil

Mapping Tekstur :

Didasarkan pada koordinat tekstur parametric

glTexCoord\*() ditetapkan pada masing – masing vertex

### Pengertian OpenGl

OpenGL adalah API (Application Programing Interface) yang dikenalkan oleh SGI (Silicon Graphics Inc) kali pertama pada tahun 1980-an. Awalnya OpenGL diciptakan untuk mendukung proses rendering, pada saat itu hanya digunakan untuk mendukung proses redering yang dilakukan oleh komputer graphic produksi SGI. Namun akhirnya, OpenGL dijadikan standar oleh berbagai perusahaan software dan hardware.

open GL (program java for openGL). adalah bahasa yang dapat dijalankan dimanapun dan di sembarang platform apapun, diberagam lingkungan : internet, intranets, consumer electronic products, dan computer applications. Bahasa pemrograman berorientasi objek telah menjadi aliran utama (mainstream), java benar-benar berorientasi objek sejati, melebihi C++.Segala sesuatu dijava kecuali sedikit type dasar (int, float, double, char) adalah objek.

### Dev C++

Dev-C + + adalah sebuah lingkungan pengembangan terpadu gratis (IDE) didistribusikan di bawah GNU General Public License untuk pemrograman di C dan C + +. MinGW, sebuah kompiler gratis, dibundel dengan itu. IDE ditulis dalam Delphi.

Proyek ini diselenggarakan oleh SourceForge. Dev-C + + pada awalnya dikembangkan oleh programmer Colin Laplace. Dev-C + + berjalan secara eksklusif pada Microsoft Windows.

Pertumpahan darah Dev-C + + adalah fitur lengkap Integrated Development Environment (IDE) untuk C dan C + + bahasa pemrograman. Ia menggunakan MinGW atau TDM-GCC pelabuhan 64bit dari GCC sebagai compiler nya. Dev-C + + juga dapat digunakan dalam kombinasi dengan Cygwin atau kompiler berbasis GCC lainnya.

Salah satu aspek tambahan Dev-C + + adalah penggunaan DevPaks, dikemas ekstensi pada lingkungan pemrograman dengan library tambahan, template, dan utilitas. DevPaks sering mengandung, namun tidak terbatas pada, GUI utilitas, termasuk toolkit populer seperti GTK +, wxWidgets, dan FLTK. DevPaks lainnya termasuk perpustakaan untuk lebih penggunaan fungsi lanjutan.

Dev-C + + umumnya dianggap program Windows-only, tetapi ada upaya untuk membuat versi Linux: file header dan pembatas jalan yang switchable antara platform.

# LANGKAH PEMBUATAN DAN IMPLEMENTASI

## Langkah Pembuatan

Pembuatan objek yang kami lakukan dengan pemanggilan fungsi-fungsi umum yang sudah tersedia di library OpenGL itu sendiri. Berikut objek-objek yang kami panggil :

**Rincian Aplikasi**

* + - 1. Terain yang dibuat terdiri dari terrain air dan terrain pasirpantai
      2. Matahari, awan, bola,danbatudibuat dengan mengabungakan Bola/Lingkaran (*Sphere*) dengan lingkaran yang *solid*di transformasikan*.*
      3. Net, Kursisantai, mejadibuat dengan menggabungkan kubus (*cube*) dengan balok yang *solid*yang di transformasikan dan di rotasi sedemikian rupa
      4. Pohon-pohonan menggabungkan kerucut (*cone*) sebagai batang pohon dan lingkaran (*sphere*) sebagai dedaunananya yang di transformasikan dan di rotasi sedemikian rupa sehingga terlihat seperti pohon
      5. Batu dibuat dengan lingkaran (*sphere*)

## Perangkat Lunak Pendukung

Perangkat lunak yang dipasang pada sistem komputer yang digunakan untuk membangun aplikasi ini adalah sebagai berikut :

Sistem Operasi *Windows* 7.

Dev C++ v.4.9.9.2

Glut32 dll

## Perangkat Keras Pendukung

Kebutuhan minimal perangkat keras (*hardware*) yang diperluka untukmengimplementasikan program aplikasi ini adalah sebagai berikut :

* + - 1. Prosesor 1.60 Ghz
      2. Ram 2 Gb
      3. VGA 128MB
      4. Keyboard dan Mouse

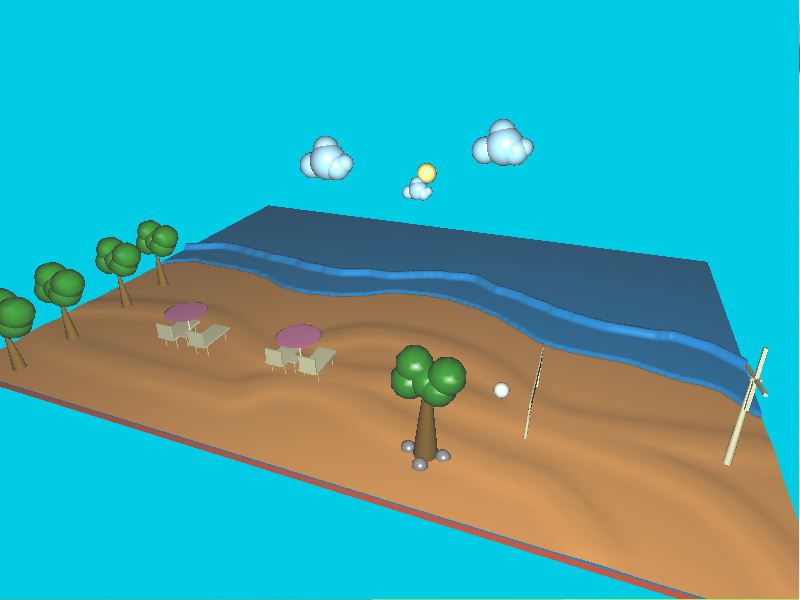
## Objeck List Yang terealisasi

1. Pantai
2. Pasir
3. Awan
4. Net voli
5. Bola voli
6. Matahari
7. Payung
8. Kursi santai
9. Meja
10. Pohon

## Source Code Program OpenGL

|  |
| --- |
| //--OPEN GL--  #include <windows.h>  #include <GL/glut.h>  #include <stdio.h>  #include <gl\gl.h>  #include <gl\glu.h>  #include <stdlib.h>  #include "imageloader.h"  #include "vec3f.h"  GLuint texture[20];  typedef struct Image Image;  #define checkImageWidth 100  #define checkImageHeight 100  GLubyte checkImage[checkImageWidth][checkImageHeight][3];  using namespace std;  float sudutnya = 30;  static GLfloat spin, spin2 = 0.0;  float angle = 0;  float lastx, lasty;  GLint stencilBits;  //--- Scrip Awal Untuk View Objeck--///  static int viewx = 140;  static int viewy = 200;  static int viewz = 360;  float rot = 0;  ///--- Clas Untuk Membuat Terain---///  class Terrain { //class untuk terain  private:  int w; //--Lebar--\\  int l; //--Panjang--\\  float\*\* hs; //--Heights--\\  Vec3f\*\* normals; //--Inisialisasi Objeck apabila normal--\\  bool computedNormals;    public:  Terrain(int w2, int l2) {  w = w2;  l = l2;  hs = new float\*[l];  for (int i = 0; i < l; i++) {  hs[i] = new float[w];  }  normals = new Vec3f\*[l];  for (int i = 0; i < l; i++) {  normals[i] = new Vec3f[w];  }  computedNormals = false;  }  ~Terrain() {  for (int i = 0; i < l; i++) {  delete[] hs[i];  }  delete[] hs;  for (int i = 0; i < l; i++) {  delete[] normals[i];  }  delete[] normals;  }  int width() {  return w;  }  int length() {  return l;  }  //--Mengatur ketinggian Terain di (x, z) ke y--\\  void setHeight(int x, int z, float y) {  hs[z][x] = y;  computedNormals = false;  }  //--Mengembalikan ketinggian Pola di (x, z)--\\  float getHeight(int x, int z) {  return hs[z][x];  }  //--Menghitung meletakkan semua memory yang dikehendaki untuk kondisi normal-\\  void computeNormals() {  if (computedNormals) {  return;  }  //Compute the rough version of the normals  Vec3f\*\* normals2 = new Vec3f\*[l];  for (int i = 0; i < l; i++) {  normals2[i] = new Vec3f[w];  }  for (int z = 0; z < l; z++) {  for (int x = 0; x < w; x++) {  Vec3f sum(0.0f, 0.0f, 0.0f);  Vec3f out;  if (z > 0) {  out = Vec3f(0.0f, hs[z - 1][x] - hs[z][x], -1.0f);  }  Vec3f in;  if (z < l - 1) {  in = Vec3f(0.0f, hs[z + 1][x] - hs[z][x], 1.0f);  }  Vec3f left;  if (x > 0) {  left = Vec3f(-1.0f, hs[z][x - 1] - hs[z][x], 0.0f);  }  Vec3f right;  if (x < w - 1) {  right = Vec3f(1.0f, hs[z][x + 1] - hs[z][x], 0.0f);  }  if (x > 0 && z > 0) {  sum += out.cross(left).normalize();  }  if (x > 0 && z < l - 1) {  sum += left.cross(in).normalize();  }  if (x < w - 1 && z < l - 1) {  sum += in.cross(right).normalize();  }  if (x < w - 1 && z > 0) {  sum += right.cross(out).normalize();  }  normals2[z][x] = sum;  }  }  //--Objeck Keadaan Normal--//  const float FALLOUT\_RATIO = 0.5f;  for (int z = 0; z < l; z++) {  for (int x = 0; x < w; x++) {  Vec3f sum = normals2[z][x];  if (x > 0) {  sum += normals2[z][x - 1] \* FALLOUT\_RATIO;  }  if (x < w - 1) {  sum += normals2[z][x + 1] \* FALLOUT\_RATIO;  }  if (z > 0) {  sum += normals2[z - 1][x] \* FALLOUT\_RATIO;  }  if (z < l - 1) {  sum += normals2[z + 1][x] \* FALLOUT\_RATIO;  }  if (sum.magnitude() == 0) {  sum = Vec3f(0.0f, 1.0f, 0.0f);  }  normals[z][x] = sum;  }  }  for (int i = 0; i < l; i++) {  delete[] normals2[i];  }  delete[] normals2;  computedNormals = true;  }  //-Scrip Kembali Ke Posisi Normal Apabila Objeck Sudah di Posisi (x, z)-/  Vec3f getNormal(int x, int z) {  if (!computedNormals) {  computeNormals();  }  return normals[z][x];  }  };  //---end class---//  //--Fungsi InitReading Di Terain--//  void initRendering() {  glEnable(GL\_DEPTH\_TEST);  glEnable(GL\_COLOR\_MATERIAL);  glEnable(GL\_LIGHTING);  glEnable(GL\_LIGHT0);  glEnable(GL\_NORMALIZE);  glShadeModel(GL\_SMOOTH);  }  //--Fungsi Proses Loader Image--//  int LoadBitmap(char \*filename)  {  FILE \* file;  char temp;  long i;  //Gambar BMP  struct {  int biWidth;  int biHeight;  short int biPlanes;  unsigned short int biBitCount;  unsigned char \*data;  } infoheader;  GLuint num\_texture;  // Open the file for reading  if( (file = fopen(filename, "rb"))==NULL) return (-1);  /\* start reading width & height \*/  fseek(file, 18, SEEK\_CUR);  fread(&infoheader.biWidth, sizeof(int), 1, file);  fread(&infoheader.biHeight, sizeof(int), 1, file);  fread(&infoheader.biPlanes, sizeof(short int), 1, file);  if (infoheader.biPlanes != 1) {  printf("Planes from %s is not 1: %u\n", filename, infoheader.biPlanes);  return 0;  }  // read the bpp  fread(&infoheader.biBitCount, sizeof(unsigned short int), 1, file);  if (infoheader.biBitCount != 24) {  printf("Bpp from %s is not 24: %d\n", filename, infoheader.biBitCount);  return 0;  }  fseek(file, 24, SEEK\_CUR);  // read the data  if(infoheader.biWidth<0){  infoheader.biWidth = -infoheader.biWidth;  }  if(infoheader.biHeight<0){  infoheader.biHeight = -infoheader.biHeight;  }  infoheader.data = (unsigned char \*) malloc(infoheader.biWidth \* infoheader.biHeight \* 3);  if (infoheader.data == NULL) {  printf("Error allocating memory for color-corrected image data\n");  return 0;  }  if ((i = fread(infoheader.data, infoheader.biWidth \* infoheader.biHeight \* 3, 1, file)) != 1) {  printf("Error reading image data from %s.\n", filename);  return 0;  }  for (i=0; i<(infoheader.biWidth \* infoheader.biHeight \* 3); i+=3) { // reverse all of the colors. (bgr -> rgb)  temp = infoheader.data[i];  infoheader.data[i] = infoheader.data[i+2];  infoheader.data[i+2] = temp;  }  fclose(file); // Closes the file stream  glGenTextures(1, &num\_texture);  glBindTexture(GL\_TEXTURE\_2D, num\_texture); // Bind the ID texture specified by the 2nd parameter  // The next commands sets the texture parameters  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT); // If the u,v coordinates overflow the range 0,1 the image is repeated  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR); // The magnification function ("linear" produces better results)  glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR\_MIPMAP\_NEAREST); //The minifying function  glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);  // Finally we define the 2d texture  glTexImage2D(GL\_TEXTURE\_2D, 0, 3, infoheader.biWidth, infoheader.biHeight, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, infoheader.data);  // And create 2d mipmaps for the minifying function  gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, infoheader.biWidth, infoheader.biHeight, GL\_RGB, GL\_UNSIGNED\_BYTE, infoheader.data);    free(infoheader.data); // Free the memory we used to load the texture  return (num\_texture); // Returns the current texture OpenGL ID  }  //--Membuat efek gambar dengan menggunakan skala, rotasi, translasi , fog,terrain--//  //--Fungsi Proses Pembuatan Terrrain Dan proses inisialisai Terrrain--//  Terrain\* loadTerrain(const char\* filename, float height) {  Image\* image = loadBMP(filename);  Terrain\* t = new Terrain(image->width, image->height);  for (int y = 0; y < image->height; y++) {  for (int x = 0; x < image->width; x++) {  unsigned char color = (unsigned char) image->pixels[3\*(y  \* image->width + x)];  float h = height \* ((color / 255.0f) - 0.5f);  t->setHeight(x, y, h);  }  }  delete image;  t->computeNormals();  return t;  }  float \_angle = 100.0f;  //buat tipe data terain  Terrain\* \_terrain;  Terrain\* \_terrainTanah;  Terrain\* \_terrainAir;  //-- Untuk untuk di display DAN Fungsi mengaktifkan pencahayaan--\\  const GLfloat light\_ambient[] = { 0.3f, 0.3f, 0.3f, 1.0f };  const GLfloat light\_diffuse[] = { 0.7f, 0.7f, 0.7f, 1.0f };  const GLfloat light\_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };  const GLfloat light\_position[] = { 1.0f, 1.0f, 1.0f, 1.0f };  const GLfloat light\_ambient2[] = { 0.3f, 0.3f, 0.3f, 0.0f };  const GLfloat light\_diffuse2[] = { 0.3f, 0.3f, 0.3f, 0.0f };  const GLfloat mat\_ambient[] = { 0.8f, 0.8f, 0.8f, 1.0f };  const GLfloat mat\_diffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };  const GLfloat mat\_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };  const GLfloat high\_shininess[] = { 100.0f };  void cleanup() {  delete \_terrain;  delete \_terrainTanah;  }  //untuk di display  void drawSceneTanah(Terrain \*terrain, GLfloat r, GLfloat g, GLfloat b) {    float scale = 500.0f / max(terrain->width() - 1, terrain->length() - 1);  glScalef(scale, scale, scale);  glTranslatef(-(float) (terrain->width() - 1) / 2, 0.0f,  -(float) (terrain->length() - 1) / 2);  glColor3f(r, g, b);  for (int z = 0; z < terrain->length() - 1; z++) {    //Makes OpenGL draw a triangle at every three consecutive vertices  glBegin(GL\_TRIANGLE\_STRIP);  for (int x = 0; x < terrain->width(); x++) {  Vec3f normal = terrain->getNormal(x, z);  glNormal3f(normal[0], normal[1], normal[2]);  glVertex3f(x, terrain->getHeight(x, z), z);  normal = terrain->getNormal(x, z + 1);  glNormal3f(normal[0], normal[1], normal[2]);  glVertex3f(x, terrain->getHeight(x, z + 1), z + 1);  }  glEnd();  }  }  void awan()  {  glPushMatrix();  glColor3ub(153, 223, 255);  glutSolidSphere(10, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(10,0,1);  glutSolidSphere(5, 20, 20);  glPopMatrix();  glPushMatrix();  glTranslatef(-2,6,-2);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glEndList(); //untuk menutup glnewlist  }  void awan2()  {  glPushMatrix();  glColor3ub(153, 223, 255);  glutSolidSphere(10, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(10,0,1);  glutSolidSphere(5, 20, 20);  glPopMatrix();  glPushMatrix();  glTranslatef(-2,6,-2);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glEndList(); //untuk menutup glnewlist  }  void awan3()  {  glPushMatrix();  glColor3ub(153, 223, 255);  glutSolidSphere(10, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(10,0,1);  glutSolidSphere(5, 20, 20);  glPopMatrix();  glPushMatrix();  glTranslatef(-2,6,-2);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(7, 50, 50);  glPopMatrix();  glEndList();  //untuk menutup glnewlist  }    void matahari()  {  glPushMatrix();  glColor3ub(255, 253, 116);  glutSolidSphere(10, 60, 60);  glPopMatrix();    glEndList();  }  void pohon(){  //batang  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(104,70,14);  glRotatef(270,1,0,0);  gluCylinder(pObj, 4, 0.7, 30, 25, 25);  glPopMatrix();  }  void daun()  {  glPushMatrix();  glColor3ub(18,118,13);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(10,5,1);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-8,6,-2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glEndList(); //untuk menutup glnewlist9  }  void pohon2(){  //batang  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(104,70,14);  glRotatef(270,1,0,0);  gluCylinder(pObj, 4, 0.7, 30, 25, 25);  glPopMatrix();  }  void daun2()  {  glPushMatrix();  glColor3ub(18,118,13);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(10,5,1);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-8,6,-2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glEndList(); //untuk menutup glnewlist9  }  void pohon3(){  //batang  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(104,70,14);  glRotatef(270,1,0,0);  gluCylinder(pObj, 4, 0.7, 30, 25, 25);  glPopMatrix();  }  void daun3()  {  glPushMatrix();  glColor3ub(18,118,13);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(10,5,1);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-8,6,-2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glEndList(); //untuk menutup glnewlist9  }  void pohon4(){  //batang  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(104,70,14);  glRotatef(270,1,0,0);  gluCylinder(pObj, 4, 0.7, 30, 25, 25);  glPopMatrix();  }  void daun4()  {  glPushMatrix();  glColor3ub(18,118,13);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(10,5,1);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-8,6,-2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glEndList(); //untuk menutup glnewlist9  }  void pohon5(){  //batang  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(104,70,14);  glRotatef(270,1,0,0);  gluCylinder(pObj, 4, 0.7, 30, 25, 25);  glPopMatrix();  }  void daun5()  {  glPushMatrix();  glColor3ub(18,118,13);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(10,5,1);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-8,6,-2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(-10,-3,0);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glPushMatrix();  glTranslatef(6,-2,2);  glutSolidSphere(10, 10, 10);  glPopMatrix();  glEndList(); //untuk menutup glnewlist9  }  void bola()  {  glPushMatrix();  glColor3ub(230,255,230);  glutSolidSphere(10, 60, 60);  glPopMatrix();    glEndList();  }  void meja()  {  glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 8, 0);  glScalef(2, 0.2, 2);  glutSolidCube(0.8);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0,7,0);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();      }  void meja2()  {  glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 8, 0);  glScalef(2, 0.2, 2);  glutSolidCube(0.8);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0,7,0);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();      }  void kursi()  {    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 8, 2.7);  glScalef(2, 2, 0.2);  glutSolidCube(1.2);  glRotatef(0,0,0,0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 7, 0);  glScalef(2, 0.2, 4);  glutSolidCube(1.4);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();  }  void kursi2()  {    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 8, 2.7);  glScalef(2, 2, 0.2);  glutSolidCube(1.2);  glRotatef(0,0,0,0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 7, 0);  glScalef(2, 0.2, 4);  glutSolidCube(1.4);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();  }  void kursi3()  {    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 8, 2.7);  glScalef(2, 2, 0.2);  glutSolidCube(1.2);  glRotatef(0,0,0,0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 7, 0);  glScalef(2, 0.2, 4);  glutSolidCube(1.4);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();  }  void kursi4()  {    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 8, 2.7);  glScalef(2, 2, 0.2);  glutSolidCube(1.2);  glRotatef(0,0,0,0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 7, 0);  glScalef(2, 0.2, 4);  glutSolidCube(1.4);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(1,5.9,-2);  glScalef(0.1,2.5,0.1);  glutSolidCube(1.0);  glPopMatrix();  }  void kincir()  {  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(-2,7,-10);  glRotatef(260,1,0,0);  gluCylinder(pObj, 1, 0.5, 25, 25, 25);  glPopMatrix();    glPushMatrix();  glTranslatef(-3, 33,-10);  glRotatef(sudutnya,1,0,0);  glColor3ub(204,204,153);  glScalef(1, 1, 20);  glutSolidCube(0.8);  glPopMatrix();    glPushMatrix();  glTranslatef(-3, 33,-10);  glRotatef(sudutnya,1,0,0);  glColor3ub(204,204,153);  glScalef(1, 20, 1);  glutSolidCube(0.8);  glPopMatrix();    glPushMatrix();  glTranslatef(-3, 33,-10);  glColor3ub(204,204,153);  glScalef(3, 1, 1);  glutSolidCube(0.8);  glPopMatrix();    }  void putar(int value)  {  sudutnya += 5;  if (sudutnya > 360){  sudutnya -= 360;  }  glutPostRedisplay();  glutTimerFunc(25, putar, 0);  }  void lapanganvoli()  {  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0, 27, -38);  glScalef(0.2, 3, 14);  glutSolidCube(4);  glRotatef(120,0,0,100);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0,7,-10);  glRotatef(260,1,0,0);  gluCylinder(pObj, 1, 0.5, 25, 25, 25);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0,7,-60);  glRotatef(260,1,0,0);  gluCylinder(pObj, 1, 0.5, 25, 25, 25);  glPopMatrix();  }  void payung()  {  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(204,100,153);  glTranslatef(0, 32, 0);  glScalef(15, 0.2, 15);  glutSolidSphere(1, 20, 6);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0,7,0);  glRotatef(270,1,0,0);  gluCylinder(pObj, 1, 0.5, 25, 25, 25);    glPopMatrix();    }  void payung2()  {  GLUquadricObj \*pObj;  pObj =gluNewQuadric();  gluQuadricNormals(pObj, GLU\_SMOOTH);  glPushMatrix();  glColor3ub(204,100,153);  glTranslatef(0, 32, 0);  glScalef(15, 0.2, 15);  glutSolidSphere(1, 20, 6);  glPopMatrix();    glPushMatrix();  glColor3ub(204,204,153);  glTranslatef(0,7,0);  glRotatef(270,1,0,0);  gluCylinder(pObj, 1, 0.5, 25, 25, 25);    glPopMatrix();    }  void batu()  {  glPushMatrix();  glColor3ub(112,112,112);  glutSolidSphere(40, 100, 50);  glPopMatrix();  glEndList();  }  void batu2()  {  glPushMatrix();  glColor3ub(112,112,112);  glutSolidSphere(40, 100, 50);  glPopMatrix();  glEndList();  }  void batu3()  {  glPushMatrix();  glColor3ub(112,112,112);  glutSolidSphere(40, 100, 50);  glPopMatrix();    glEndList();  }  unsigned int LoadTextureFromBmpFile(char \*filename);  void display(void) {  glClearStencil(0); //clear the stencil buffer  glClearDepth(1.0);//-Scrip untuk menghindari polygon yang bertumpuk--\  glClearColor(0.0, 0.8, 0.9, 1);  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT | GL\_STENCIL\_BUFFER\_BIT); //clear the buffers  glLoadIdentity();//-- inisialisasi matrik modelview Objeck--\\  gluLookAt(viewx, viewy, viewz, 0.0, 0.0, 5.0, 0.0, 1.0, 0.0);//Scirp Pengecekan sudut pandang pada objek yang terlihat--\  glPushMatrix();  drawSceneTanah(\_terrain, 0.8, 0.5, 0.2);  glPopMatrix();  glPushMatrix();  drawSceneTanah(\_terrainTanah, 0.7f, 0.2f, 0.1f);  glPopMatrix();  glPushMatrix();  drawSceneTanah(\_terrainAir, 0.0f, 0.4f, 0.7f);  glPopMatrix();    glPushMatrix();  glTranslatef(-30,50,-130);  awan();  glPopMatrix();    glPushMatrix();  glTranslatef(-140,65,-150);  glScalef(2.0,2.0,2.0);  awan2();  glPopMatrix();    glPushMatrix();  glTranslatef(40,85,-200);  glScalef(2.0,2.0,2.0);  awan3();  glPopMatrix();    glPushMatrix();  glTranslatef(70,-20,150);  glScalef(2.0, 2.0, 2.0);  pohon();  glPopMatrix();    glPushMatrix();  glTranslatef(-200,-1,180);  glScalef(1.0, 1.0, 1.0);  pohon2();  glPopMatrix();    glPushMatrix();  glTranslatef(-200,-1,140);  glScalef(1.0, 1.0, 1.0);  pohon3();  glPopMatrix();    glPushMatrix();  glTranslatef(-200,-1,90);  glScalef(1.0, 1.0, 1.0);  pohon4();  glPopMatrix();    glPushMatrix();  glTranslatef(-200,-1,50);  glScalef(1.0, 1.0, 1.0);  pohon5();  glPopMatrix();    glPushMatrix();  glTranslatef(-32,-35.2,115);  glScalef(5, 5, 5);  meja();  glPopMatrix();    glPushMatrix();  glTranslatef(-120,-35.2,115);  glScalef(5, 5, 5);  meja2();  glPopMatrix();  glPushMatrix();  glTranslatef(-20,-35.2,105);  glScalef(5, 5, 5);  kursi();  glPopMatrix();      glPushMatrix();  glTranslatef(-45,-35.2,105);  glScalef(5, 5, 5);  kursi2();  glPopMatrix();    glPushMatrix();  glTranslatef(-108,-35.2,105);  glScalef(5, 5, 5);  kursi3();  glPopMatrix();    glPushMatrix();  glTranslatef(-135,-35.2,105);  glScalef(5, 5, 5);  kursi3();  glPopMatrix();    glPushMatrix();  glTranslatef(120,-12,135);  glScalef(1, 1, 1);  lapanganvoli();  glPopMatrix();    glPushMatrix();  glTranslatef(223,-12,135);  glScalef(2, 2, 2);  kincir();  glPopMatrix();      glPushMatrix();  glTranslatef(-30,-12,110);  glScalef(1, 1, 1);  payung();  glPopMatrix();    glPushMatrix();  glTranslatef(-120,-12,110);  glScalef(1, 1, 1);  payung();  glPopMatrix();    glPushMatrix();  glTranslatef(-190,40,180);  glScalef(1,1,1);  daun();  glPopMatrix();      glPushMatrix();  glTranslatef(70,40,150);  glScalef(1,1,1);  daun2();  glPopMatrix();    glPushMatrix();  glTranslatef(-195,40,140);  glScalef(1,1,1);  daun3();  glPopMatrix();    glPushMatrix();  glTranslatef(-195,40,90);  glScalef(1,1,1);  daun4();  glPopMatrix();    glPushMatrix();  glTranslatef(-195,40,50);  glScalef(1,1,1);  daun5();  glPopMatrix();    glPushMatrix();  glTranslatef(100,-3.,90);  glScalef(0.5,0.5,0.5);  bola();  glPopMatrix();    glPushMatrix();  glTranslatef(70,-10.,160);  glScalef(0.11,0.11,0.1);  batu();  glPopMatrix();    glPushMatrix();  glTranslatef(80,-10.,150);  glScalef(0.11,0.11,0.1);  batu2();  glPopMatrix();    glPushMatrix();  glTranslatef(60,-10.,150);  glScalef(0.11,0.11,0.1);  batu3();  glPopMatrix();    glPushMatrix();  glTranslatef(-25,65,-140);  matahari();  glPopMatrix();            glutSwapBuffers();  glFlush();  rot++;  angle++;  }  void init(void) {  glEnable(GL\_DEPTH\_TEST);  glEnable(GL\_LIGHTING);  glEnable(GL\_LIGHT0);  glDepthFunc(GL\_LESS);  glEnable(GL\_NORMALIZE);  glEnable(GL\_COLOR\_MATERIAL);  glDepthFunc(GL\_LEQUAL);  glShadeModel(GL\_SMOOTH);  glHint(GL\_PERSPECTIVE\_CORRECTION\_HINT, GL\_NICEST);  glEnable(GL\_CULL\_FACE);        \_terrain = loadTerrain("heightmap.bmp", 20);  \_terrainTanah = loadTerrain("heightmapTanah.bmp", 20);  \_terrainAir = loadTerrain("heightmapAir.bmp", 20);  //binding texture  }  static void kibor(int key, int x, int y) {  switch (key) {  case GLUT\_KEY\_HOME:  viewy++;  break;  case GLUT\_KEY\_END:  viewy--;  break;  case GLUT\_KEY\_UP:  viewz--;  break;  case GLUT\_KEY\_DOWN:  viewz++;  break;  case GLUT\_KEY\_RIGHT:  viewx++;  break;  case GLUT\_KEY\_LEFT:  viewx--;  break;  //---mengatur warna cahaya ambient, vektor berisi nilai R,G,B--\\  case GLUT\_KEY\_F1: {  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);  glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  }  ;  break;  case GLUT\_KEY\_F2: {  glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient2);  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse2);  glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient);  glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  }  ;  break;  default:  break;  }  }  void keyboard(unsigned char key, int x, int y) {  if (key == 'd') {  spin = spin - 1;  if (spin > 360.0)  spin = spin - 360.0;  }  if (key == 'a') {  spin = spin + 1;  if (spin > 360.0)  spin = spin - 360.0;  }  if (key == 'q') {  viewz++;  }  if (key == 'e') {  viewz--;  }  if (key == 's') {  viewy--;  }  if (key == 'w') {  viewy++;  }  }  //-Mengatur dan membuat gambar tetap pada posisi yang tepat--//  void reshape(int w, int h) {  glViewport(0, 0, (GLsizei) w, (GLsizei) h);  glMatrixMode(GL\_PROJECTION);  glLoadIdentity();  gluPerspective(60, (GLfloat) w / (GLfloat) h, 0.1, 1000.0);  glMatrixMode(GL\_MODELVIEW);  }  /\* Deklarasi window size, position, dan display mode  \* Pemanggilan routin inisialisasi.  \* Memanggil fungsi untuk manampilkan objek di layar  \*/  int main(int argc, char \*\*argv) {  glutInit(&argc, argv);  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGBA | GLUT\_STENCIL | GLUT\_DEPTH); //add a stencil buffer to the window  glutInitWindowSize(800, 600);  glutInitWindowPosition(100, 100);  glutCreateWindow("Sample Terain");  init();  glutDisplayFunc(display);  glutIdleFunc(display);  glutReshapeFunc(reshape);  glutSpecialFunc(kibor);  glutKeyboardFunc(keyboard);    // mengkonfigurasi sumber cahaya ...  glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);  glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  glMaterialfv(GL\_FRONT, GL\_SHININESS, high\_shininess);  glColorMaterial(GL\_FRONT, GL\_DIFFUSE);  glutTimerFunc(25, putar, 0);    glutMainLoop();  return 0;  } |

## Tampilan Program OpenGL



Gambar Tampilan Program

# KESIMPULAN DAN SARAN

## Kesimpulan

**Berdasarkan sistem yang telah dikembangkan dan hasil pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut :**

1. Dengan menggunakan Dev C++ dan library OpenGL dapat membantu membuat Model Pantai 3D dengan baik
2. Mengetahui fungsi-fungsi pencampuran lighting, color, transformasi dan mapping.
3. Dapat melihat secara langsung perubahan yang terjadi pada objek.
4. Format gambar yang dapat digunakan untuk texturing adalah BMP

## Saran

**Ada beberapa hal yang dapat dijadikan saran dalam proses pembuatan Model Pantai 3D, antara lain :**

1. Pemanfaatan Lightning dengan lebih baik, agar object dapat terlihat dengan lebih baik
2. Penggunaan gambar selain bitmap, agar file yang digunakan bisa lebih ringan
3. Pengefisienan coding agar lebih mudah dalam mengembangkan objek