BAB 2

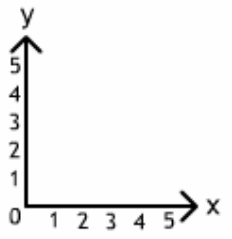
DASAR TEORI

2.1 Teori Dasar Grafika Komputer

Pada bagian ini akan dibahas tentang teori dasar dari grafika komputer, yang akan digunakan dalam pembahasan teknik *environment mapping*.

2.1.1 Sistem Koordinat Dua Dimensi

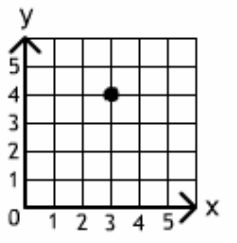
Sistem koordinat digunakan untuk membedakan posisi atau lokasi suatu titik. Sistem koordinat yang sering digunakan adalah sistem koordinat *cartesian* dua dimensi seperti yang digambarkan pada gambar 2.1. Dalam sistem koordinat *cartesian* dua dimensi digunakan 2 besaran (*x*,*y*). Sumbu x digunakan untuk memproyeksikan panjang dan sumbu y untuk tinggi.



Gambar 2. 1 Sistem Koordinat *Cartesian* dua dimensi, dengan sumbu x dan y

2.1.2 Titik / Pixel

Titik atau *pixel* (singkatan dari *picture element*) merupakan elemen penyusun terkecil dari suatu objek dalam grafika komputer. Titik didefinisikan pada suatu posisi (x,y) dalam koordinat *cartesian*. Gambar 2.2 menunjukkan sebuah titik pada posisi (3,4) dalam koordinat *cartesian*.

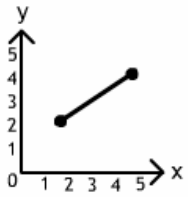


Gambar 2. 2 Titik dengan koordinat *cartesian* (3,4)

Titik memiliki atribut yaitu warna dari titik tersebut. Selain itu, titik juga memiliki *property* yang terdiri dari posisi atau koordinat (x,y) titik tersebut.

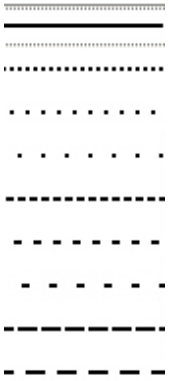
2.1.3 Garis

Garis merupakan persamaan matematika yang disusun oleh sekumpulan titik. Suatu titik merupakan penyusun dari garis jika koordinat titik tersebut memenuhi persamaan garis $y = mx + c$. Gambar 2.3 menunjukkan sebuah garis dengan koordinat titik – titik ujung (2,2) dan (5,4).



Gambar 2. 3 Garis dengan koordinat titik – titik ujung (2,2) dan (5,4)

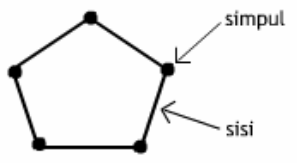
Garis memiliki atribut yang terdiri dari warna, tebal dan jenis dari garis tersebut (gambar 2.4). Garis juga memiliki *property* yang terdiri dari koordinat dari titik – titik ujung garis tersebut (x_1,y_1) dan (x_2,y_2).



Gambar 2. 4 Jenis garis

2.1.4 Poligon

Poligon merupakan bidang tertutup yang dibentuk oleh minimal 3 simpul (titik sudut atau *vertices*) yang terhubung oleh garis (sisi atau *edge*). Garis – garis tersebut tidak saling memotong. Contoh : segitiga, segi lima, segi delapan, dsb. Gambar 2.5 menunjukkan sebuah poligon yang berupa segi lima.



Gambar 2. 5 Poligon segi lima memiliki lima simpul dan lima sisi

Poligon juga memiliki atribut seperti : tebal tepi, jenis tepi, warna tepi dan jenis isi poligon (gambar 2.6). *Property* dari poligon terdiri dari koordinat titik – titik sudut penyusunnya.



Gambar 2. 6 Jenis isi poligon (*hollow, solid, patterned*)

2.2 Teori Grafika Komputer Tiga Dimensi

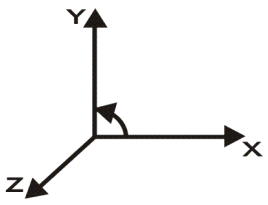
Teori grafika komputer tiga dimensi yang akan digunakan dalam pembahasan teknik *environment mapping* akan dibahas pada bagian ini.

2.2.1 Sistem Koordinat Tiga Dimensi

Koordinat dua dimensi dirasakan kurang memadai dalam merepresentasikan suatu objek tiga dimensi. Pada grafik tiga dimensi terdapat 2 buah sistem koordinat, yaitu sistem koordinat *cartesian* tiga dimensi dan sistem koordinat *spheris*.

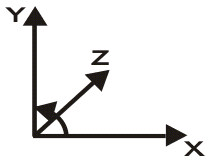
2.2.1.1 Sistem Koordinat Cartesian Tiga Dimensi

Dalam sistem koordinat *cartesian* dua dimensi hanya terdapat 2 besaran saja (sumbu x dan y). Sedangkan untuk memodelkan objek tiga dimensi yang memiliki kedalaman, diperlukan 1 sumbu tambahan yaitu sumbu z, maka digunakanlah sistem koordinat *cartesian* tiga dimensi. Sistem koordinat *cartesian* dibedakan menjadi 2 yaitu, sistem koordinat tangan kanan dan sistem koordinat tangan kiri. Sistem koordinat tangan kanan diilustrasikan dengan jari telunjuk tangan kanan sebagai sumbu x diputar sebesar 90 derajat ke arah jari tengah tangan kanan sebagai sumbu y, sehingga arah ibu jari tangan kanan sebagai sumbu z mendekati pengamat (arah z positif). Gambar 2.7 menunjukkan orientasi sumbu koordinat menggunakan sistem koordinat tangan kanan.



Gambar 2. 7 Sistem Koordinat Tangan Kanan

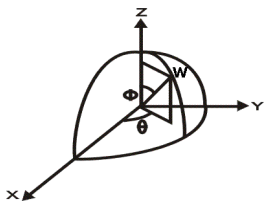
Sedangkan pada sistem koordinat tangan kiri dapat diilustrasikan dengan jari telunjuk tangan kiri sebagai sumbu x diputar sebesar 90 derajat ke arah jari tengah tangan kiri sebagai sumbu y, sehingga arah ibu jari tangan kiri sebagai sumbu z menjauhi pengamat (arah z positif). Gambar 2.8 menunjukkan orientasi sumbu koordinat menggunakan sistem koordinat tangan kiri. Dalam komputer grafik lebih banyak digunakan sistem koordinat tangan kanan.



Gambar 2. 8 Sistem Koordinat Tangan Kiri

2.2.1.2 Sistem Koordinat Spheris

Pada sistem koordinat *spheris*, sebuah titik dianggap terletak pada permukaan kulit bola yang memiliki jari – jari R, dan titik pusat bola berimpit dengan titik pusat sistem koordinat. Dapat diambil sembarang titik, misalnya titik W sehingga dikenal besaran *colatitud* dan *azimuth*. *Colatitud* (ϕ) adalah besarnya sudut yang dibentuk antara sumbu z dan garis yang ditarik dari titik pusat ke titik W. *Azimuth* (θ) adalah besarnya sudut antara bidang xz dengan bidang yang melewati titik W dan sumbu z. Gambar 2.9 menunjukkan sistem koordinat *spheris*.



Gambar 2. 9 Sistem Koordinat Spheris

Dengan trigonometri, komponen koordinat *spheris* dapat dinyatakan dalam bentuk R, ϕ, dan θ yaitu :

Wx = R sin (ϕ) cos (θ)

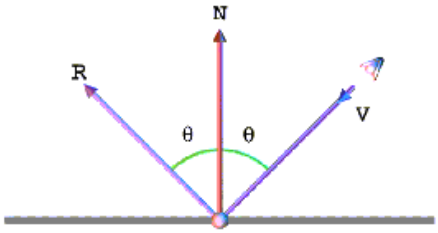
Wy = R sin (ϕ) sin (θ)

Wz = R cos (ϕ)

2.2.2 Refleksi

Refleksi merupakan sifat material yang dapat mencerminkan bentuk objek – objek di sekitarnya jika terkena sinar. Besarnya sudut pantul bergantung pada besarnya sudut datang dari sinar dan bentuk permukaan dari objek yang terkena sinar tersebut.

Secara umum refleksi dapat diilustrasikan seperti gambar 2.10 berikut :



Gambar 2. 10 Vektor refleksi $\vec{R}$, berasal dari vektor $\vec{V}$, dengan garis normal $\vec{N}$, dan dengan sudut θ

Rumus untuk menghitung vector $\vec{R}$ dengan besar sudut θ, vektor normal $\vec{N}$ dan vektor asal $\vec{V}$:

$$\vec{R} = \vec{V} - 2(\vec{V} \cdot \vec{N})\vec{N}$$

2.2.3 Proyeksi

Proyeksi merupakan proses mentransformasikan titik – titik di sistem koordinat berdimensi n ke titik – titik di sistem koordinat dengan dimensi yang lebih kecil dari n. Dalam grafik tiga dimensi, proyeksi didefinisikan dengan pancaran sinar proyeksi yang berasal dari sebuah pusat proyeksi yang dipancarkan sampai mengenai masing – masing titik dari objek tiga dimensi dan memotong sebuah bidang proyeksi sehingga terbentuk hasil proyeksi.

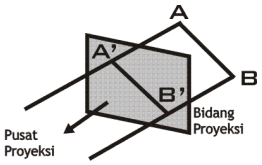
Dua teknik proyeksi yang sering digunakan dalam grafik tiga dimensi yaitu proyeksi perspektif dan proyeksi paralel. Perbedaan diantara kedua teknik itu terletak pada letak pusat proyeksi terhadap bidang proyeksi. Pada proyeksi perspektif, jarak pusat proyeksi ke bidang proyeksi terdefinisi. Pada proyeksi paralel jarak pusat proyeksi ke bidang proyeksi tidak terdefinisi (sangat jauh).

Proyeksi perspektif menghasilkan efek visual yang menyerupai sistem fotografi dan sistem visual pada manusia. Hal ini membuat proyeksi perspektif menampilkan objek yang lebih nyata daripada objek yang dihasilkan oleh proyeksi paralel. Proyeksi paralel biasanya digunakan untuk pengukuran yang pasti. Karena perbandingan panjang objek di ruang tiga dimensi dengan panjang objek di bidang proyeksi selalu sama.

2.2.3.1 Proyeksi Paralel

Proyesi paralel dapat diilustrasikan dengan sebuah titik pada bidang tiga dimensi $W(w_x,w_y,w_z)$ diproyeksikan ke bidang dua dimensi dengan cara menghapus salah satu komponen titiknya, misalnya komponen z (membuat nilai w_z sama dengan nol) sama dengan memproyeksikan titik W ke bidang xy. Dalam

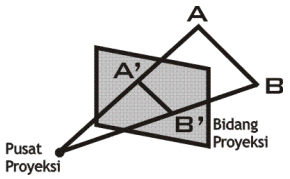
hal ini bidang xy disebut dengan bidang pandang (*viewplane*). Proyeksi dari setiap titik diperoleh dengan menjatuhkan sinar (disebut *proyektor*), dengan arah jatuh sinar tegak lurus terhadap bidang proyeksi. Proyeksi ini disebut dengan *proyeksi orthogonal*. Gambar 2.11 menggambarkan proyeksi paralel.



Gambar 2. 11 Proyeksi Paralel

2.2.3.2 Proyeksi Perspektif

Proyeksi perspektif sering juga disebut *frustrum projection*, merupakan sekumpulan sinar paralel yang diproyeksikan terhadap titik hilang (*vanishing point*). Jika sekumpulan sinar sejajar dengan salah satu sumbu koordinat, maka titik hilang disebut dengan titik hilang sumbu. Sedangkan jika ada titik hilang sumbu yang berpotongan dengan bidang proyeksi maka disebut dengan titik hilang utama. Objek yang berada di dekat *view plane*[1] akan terlihat lebih besar bila dibandingkan dengan objek yang letaknya jauh dari *view plane*, hal ini membuat proyeksi perspektif lebih terlihat nyata. Gambar 2.12 menunjukkan proyeksi perspektif.



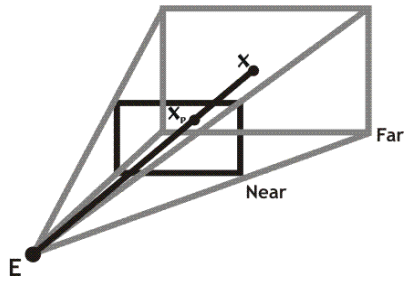
Gambar 2. 12 Proyeksi Perspektif

[1] *View Plane* adalah bidang pandang berbentuk piramid yang dihasilkan dari suatu proyeksi

2.2.4 Kamera Model

Dalam grafika komputer tidak semua objek dapat ditampilkan pada layar, sehingga diperlukan suatu batasan objek – objek mana saja yang akan ditampilkan. Objek – objek tersebut dibatasi dengan suatu daerah yang disebut *view volume*.

Untuk menampilkan ke layar maka harus dilakukan proyeksi terhadap *view volume* pada *view plane*, hanya sebagian dari *view plane* yang akan ditampilkan pada layar. Bagian dari *view plane* ini berbentuk bujur sangkar, disebut dengan *viewport* seperti terlihat pada gambar 2.13 berikut ini :



Gambar 2. 13 Kamera Model dengan *View Frustrum*

View plane dibatasi oleh dua *plane*[2] (*Near* dan *Far*), yang paralel terhadap *view plane* ini disebut dengan *view frustrum*. Paralel *plane* yang dekat dengan mata (*E*) disebut dengan *near plane*, sedangkan *plane* yang letaknya jauh dari mata disebut dengan *far plane*.

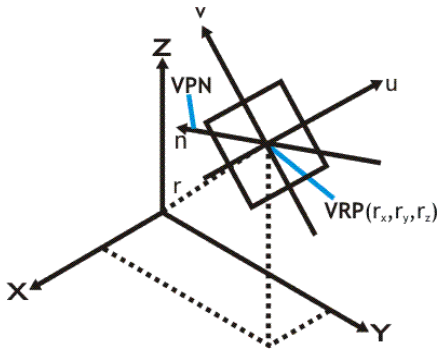
Kombinasi dari lokasi kamera (*eye point*), *view plane*, *viewport*, dan *view frustrum* disebut dengan kamera model. Kamera model merupakan suatu cara untuk menempatkan kamera dan mengorientasikannya pada ruang tiga dimensi.

[2] *Plane* adalah bidang yang berbentuk persegi panjang.

Salah satu cara yang digunakan dalam kamera model yaitu sistem koordinat arah pandang.

Sistem koordinat arah pandang sering disebut dengan sistem koordinat *uvn*, sesuai dengan nama – nama sumbu koordinat yang menjadi komponennya.

Sistem koordinat arah pandang dapat diilustrasikan dengan bidang pandang berimpit dengan bidang *uv*, sedangkan mata terletak pada sumbu *n*. Hal ini mirip dengan sistem koordinat *cartesian*, dimana mata terletak pada sumbu *z*, dan bidang pandang pada bidang *xy*.



Gambar 2. 14 Sistem Koordinat Arah Pandang

Hubungan antara sistem koordinat arah pandang dengan sistem koordinat *cartesian*, diilustrasikan pada gambar 2.14. Dari gambar tersebut, bidang *uv* merupakan bidang pandang. Pada bidang pandang ini ditentukan sebuah titik yang disebut dengan titik acuan pandang (*view reference point*, *VRP*), dan dianggap sebagai titik pusat dari sistem koordinat arah pandang. Dalam sistem koordinat dunia, komponen vektor *n* dinyatakan sebagai (n_x,n_y,n_z). Dengan adanya sumbu ke arah *n* inilah sistem koordinat *uvn* terbentuk.

Sumbu *u* dan sumbu *v* saling tegak lurus. Sedangkan vektor *n* mempunyai arah tegak lurus terhadap bidang *uv*, dan karena sumbu *u* dan sumbu *v* terletak pada

bidang *uv*, maka jelaslah bahwa arah vektor *n* tegak lurus terhadap sumbu *u* dan sumbu *v*.

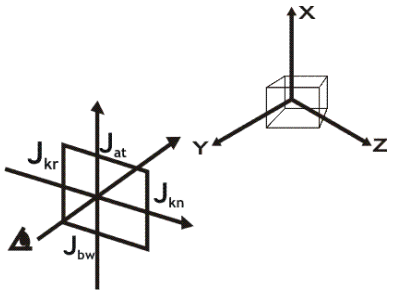
Jika vektor *n* dan *v* diketahui, dengan mudah bisa dihitung komponen vektor *u*. Karena arah vektor *u* tegak lurus pada vektor *v* dan *n*, maka vektor *u* bisa dihitung dengan menggunakan *cross product* dari vektor *n* dan *v*, yaitu

$u = n \times v$.

Setelah sistem koordinat arah pandang lengkap dengan berbagai komponennya, maka perlu ditentukan batas jendela dan posisi matanya. Jendela terletak pada sistem koordinat bidang pandang, dan batas – batasnya dinyatakan sebagai (j_{kr}, j_{at}, j_{kn}, j_{bw}). Dari batas – batas tersebut terlihat bahwa pada arah mendatar, jendela mempunyai batas – batas dari $u = j_{kr}$ sampai $v = j_{kn}$, pada arah tegak, jendela mempunyai batas dari $u = j_{at}$ sampai $v = j_{bw}$.

Posisi mata, yaitu $m(m_u, m_v, m_n)$ dapat diletakkan pada sembarang tempat pada sistem koordinat arah pandang. Posisi mata yang paling sering dipakai adalah dengan menempatkannya pada sumbu *n* sejauh M dari *VRP*, yaitu pada posisi $m = (0, 0, M)$.

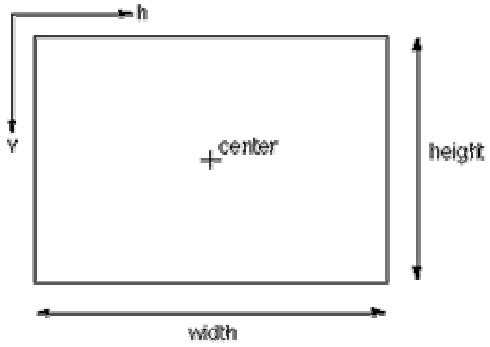
Gambar 2.15 berikut mengilustrasikan posisi jendela dan mata pada sistem koordinat arah pandang.



Gambar 2.15 Posisi jendela dan mata

2.2.5 Model Layar

Pada grafika komputer, layar penampil memiliki titik pusat koordinat (0,0) terletak pada posisi paling atas dan paling kiri. Sistem *windowing* pada komputer grafik berbentuk kotak yang didefinisikan berupa koordinat (kiri,atas) dan (kanan,bawah) seperti yang terlihat pada gambar 2.16 berikut :



Gambar 2. 16 Titik pusat koordinat pada layar penampil

Dari gambar tersebut, untuk posisi tengah (*center*), lebar layar dan tinggi layar dapat dihitung sbb :

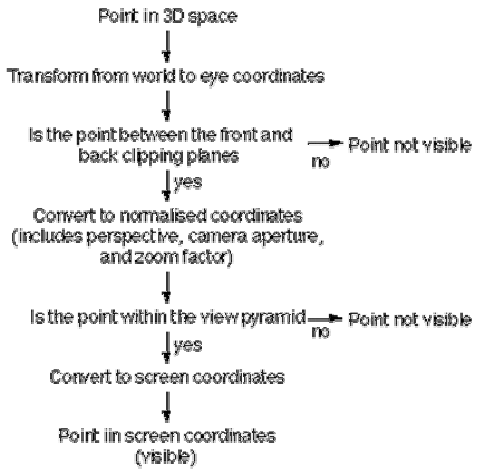
horizontal center = (kiri + kanan) / 2

vertical center = (atas + bawah) / 2

lebar = kanan – kiri

tinggi = bawah – atas

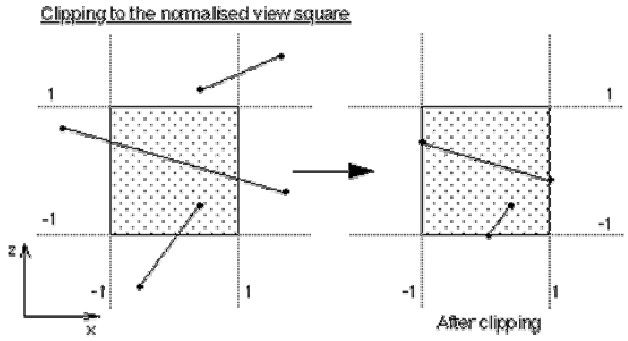
Dalam menampilkan gambar pada layar penampil, dilakukan perbandingan ukuran *window* dengan kamera model. Sehingga jika dilakukan perubahan ukuran pada kamera model, *window* dapat dengan mudah menyesuaikan ukurannya. Gambar 2.17 menunjukkan proses untuk menampilkan sebuah titik ke layar.



Gambar 2. 17 Proses untuk menampilkan sebuah titik ke layar

2.2.6 Pemotongan (*Clipping*)

Kadang kala objek – objek yang hendak ditampilkan ada bagian yang berpotongan dengan *view volume* sehingga perlu dilakukan proses pemotongan (*clipping*). Gambar 2.18 menunjukkan proses *clipping* dan *culling*.

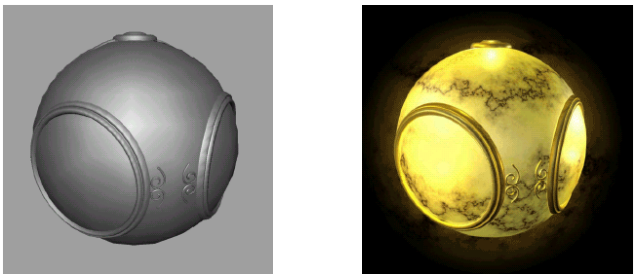


Gambar 2. 18 Proses pemotongan (*clipping*) dilakukan pada garis di tengah dan bawah, proses penghilangan (*culled*) garis di atas

2.3 Pemetaan Tekstur

Suatu objek akan terlihat lebih nyata dengan adanya tekstur yang sesuai dengan objek tersebut dibandingkan dengan objek yang tidak memakai tekstur /

polos. Gambar 2.19 (a) menunjukkan objek tanpa tekstur, sedangkan gambar 2.19 (b) menunjukkan objek dengan tekstur.



(a) **(b)**

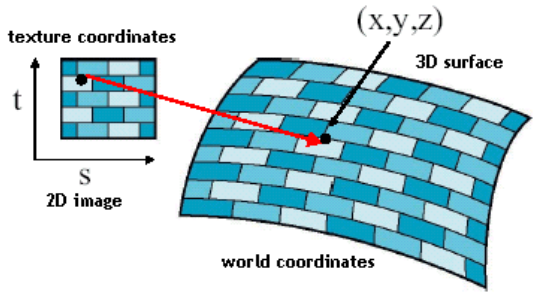
Gambar 2. 19 Perbedaan penggunaan tekstur pada objek (a) Objek tanpa tekstur, (b) Objek dengan tekstur

Dalam komputer grafik, pemakaian tekstur dilakukan dengan melakukan pemetaan terhadap tekstur pada permukaan objek. Tekstur yang dipetakan dapat berupa gambar atau pola. Jika tekstur itu berupa pola maka dapat dilakukan perulangan untuk memetakannya, sehingga ukuran filenya akan lebih kecil dibandingkan jika harus dipetakan seluruhnya. Secara sederhana dalam pemetaan tekstur dilakukan perhitungan koordinat dunia (*world coordinates*) yang berasal dari koordinat teksturnya. Gambar 2.20 mengilustrasikan pemetaan tekstur.

$$x = x(s,t)$$

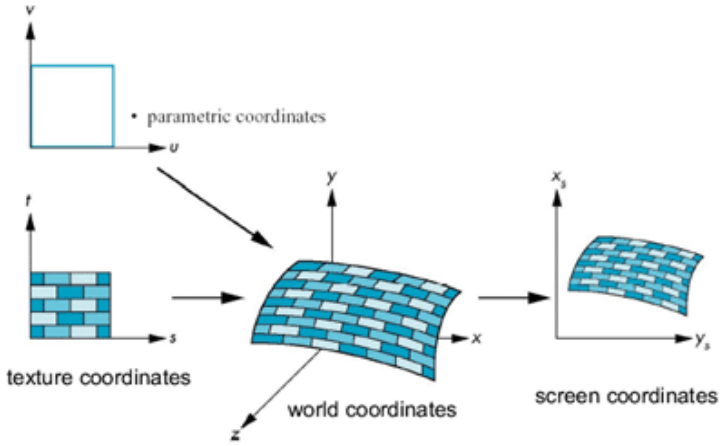
$$y = y(s,t)$$

$$z = z(s,t)$$



Gambar 2. 20 Ilustrasi dari pemetaan suatu tekstur

Dalam memetakan suatu tekstur pada suatu permukaan dikenal 4 komponen yang terlibat. *Parametric coordinates*, digunakan untuk memodelkan permukaan objek. *Texture coordinates*, digunakan untuk mengenali posisi dari tekstur yang akan dipetakan. *World coordinates*, dimana pemetaan dilakukan. *Screen coordinates*, dimana pemetaan dari tekstur ditampilkan. Gambar 2.21 mengilustrasikan 4 komponen dalam pemetaan tekstur.



Gambar 2. 21 Empat komponen dalam pemetaan tekstur

2.4 Pencahayaan

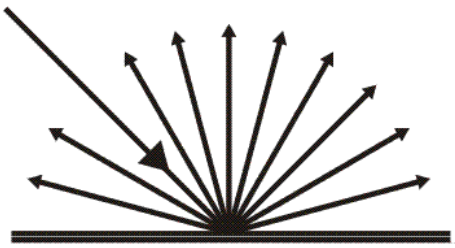
Dalam komputer grafik, faktor pencahayaan cukup penting. Suatu objek akan terlihat lebih nyata dan menarik jika diberi pencahayaan yang sesuai. Beberapa tipe pencahayaan, yaitu :

- *Directional Light Source*, tipe pencahayaan yang sumber cahayanya dianggap terletak sangat jauh dari objek, sehingga seluruh berkas cahaya yang mengenai objek datang secara paralel. Contoh *directional light* di dunia nyata adalah cahaya matahari yang menyinari bumi.
- *Positional Light Source*, tipe pencahayaan yang sumber cahaya ini serupa dengan lampu yang terletak dekat dengan objek yang disinari, sehingga jarak/posisi antara cahaya dengan objek akan mempengaruhi terang/gelapnya objek.
- *Spotlight*, tipe pencahayaan yang mirip dengan *positional light source*, hanya saja dengan batasan daerah pencahayaan.

OpenGL juga telah menyediakan model pencahayaan yang terdiri dari *diffuse*, *ambient*, dan *specular*.

2.4.1 Diffuse

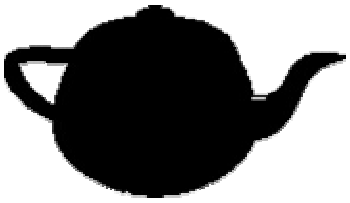
Diffuse merupakan sifat material yang menyebarkan cahaya ke semua arah. Jika sinar mengenai suatu objek yang memiliki material *diffuse* maka permukaan objek tersebut terlihat lebih jelas. Intensitas cahaya *diffuse* lebih kecil dikarenakan sifatnya penyebarannya yang ke semua arah. Gambar 2.22 menggambarkan arah penyebaran cahaya pada model *diffuse*.



Gambar 2. 22 Arah penyebaran cahaya pada model *diffuse*

2.4.2 Ambient

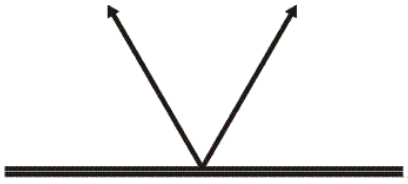
Ambient merupakan cahaya latar belakang yang dipantulkan oleh objek secara seragam ke segala arah. Intensitas cahayanya kecil karena penyebarannya ke segala arah. Gambar 2.23 menunjukkan model pencahayaan dengan *ambient*.



Gambar 2.23 Model pencahayaan dengan *ambient*

2.4.3 Specular

Model pencahayaan dimana penyebaran cahayanya hanya ke arah tertentu saja, sehingga hukum yang berlaku disini adalah sudut pantul sama dengan sudut datang cahaya. Intensitas cahayanya tinggi karena hanya dipantulkan ke satu arah saja. Objek yang memiliki material ini memiliki kesan berkilau. Gambar 2.24 menunjukkan penyebaran cahaya *specular*.



Gambar 2. 24 Arah penyebaran cahaya pada model *specular*

2.5 Komputer Grafik menggunakan OpenGL

OpenGL merupakan sebuah *open-source graphics library* yang dirancang untuk berjalan di berbagai platform berbeda (*multiplatform*). Library yang dikembangkan oleh perusahaan Silicon Graphics ini, dapat digunakan pada

hampir semua *tool* pemrograman, seperti Microsoft Visual C++, Visual Basic, Borland C++, Borland Delphi, Java, dll.

2.5.1 Graphics Library Utility Toolkit (GLUT)

GLUT merupakan pustaka tambahan yang berjalan pada OpenGL, dibuat untuk mempermudah penanganan window dan interaksi dengan user. Tanpa GLUT, memprogram OpenGL agak sedikit sulit karena perlu menggunakan fungsi – fungsi penanganan *window* dan *event* dari *windows API*.

2.5.2 Sintaks Dasar dalam OpenGL

Penamaan fungsi – fungsi dasar dalam OpenGL dimulai dengan tulisan “gl” yang diikuti dengan kata berikutnya yang diawali dengan huruf kapital.

Contohnya : glBegin (), glClear (), glPixels (), glPolygonMode (), dsb.

Beberapa fungsi membutuhkan argumen yang nilainya konstan dan tertentu. Untuk penulisan konstanta, dimulai dengan tulisan “GL” yang diikuti kata yang ditulis dalam huruf kapital semua, dengan *underscore* (_) sebagai penghubungnya. Contohnya: GL_AMBIENT_AND_DIFUSSE, GL_2D, GL_RGB, GL_CCW, dsb.

Tipe data dalam OpenGL ditulis dengan diawali dengan tulisan “GL” yang diikuti dengan kata yang ditulis dalam huruf kecil semua.

Contohnya : GLbyte, GLshort, GLint, GLfloat, GLdouble, GLboolean, dsb.

2.5.3 Perintah – perintah dalam OpenGL

Beberapa perintah dalam OpenGL diberi nomor satu, dua dan tiga pada akhir katanya yang diikuti pula dengan tipenya. Hal ini untuk menunjukkan jumlah parameter yang digunakan beserta dengan tipe dari parameter tersebut.

Contoh :

glColor3f (1.0, 0.0, 0.0);

‘3’ digunakan untuk menunjukkan 3 parameter

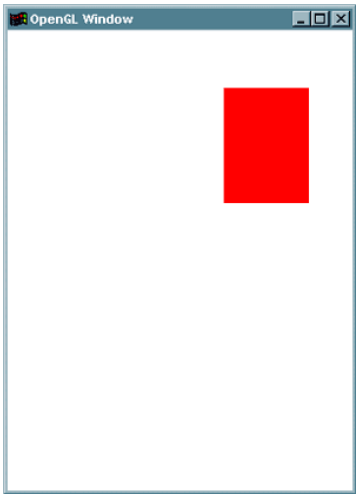
‘f’ digunakan untuk menunjukkan tipe yang digunakan adalah floating point. Kadang ada juga perintah – perintah yang diakhiri dengan huruf ‘v’, hal ini menunjukkan perintah – perintah tersebut dalam format vektor. Jika perintah tidak diakhiri dengan ‘v’, maka perintah tersebut dalam format skalar.

2.5.4 Header Files

Dalam semua program grafik yang menggunakan OpenGL, perlu disertakan beberapa *file header*. File header yang perlu disertakan adalah GLUT yang menyediakan rutin untuk pengaturan dalam menampilkan, memproyeksikan suatu objek.

```
#include <GL/glut.h>
```

Gambar 2.25 berikut ini menunjukkan sebuah *window* OpenGL.



Gambar 2. 25 Tampilan Jendela OpenGL

2.6 Environment Mapping

Dalam *texture mapping* terdapat suatu teknik yang disebut teknik *environment mapping.* Teknik ini memantulkan tekstur dari lingkungan di sekelilingnya pada suatu permukaan tiga dimensi yang melingkupi suatu objek. Teknik *environment* mapping ini pertama kali diperkenalkan oleh Blinn dan Newell pada tahun 1976.

Suatu objek yang menggunakan teknik *environment mapping*, jika dilihat dari berbagai sudut maka dapat memantulkan tekstur lingkungan pada bagian dimana objek tersebut dilihat sehingga tekstur yang terlihat akan berbeda jika dilihat pada sudut pandang yang berbeda. Hal ini sedikit berbeda dibandingkan dengan *texture mapping* biasa. Pada *texture mapping* biasa, tekstur yang terlihat akan tetap sama walaupun dilihat dari sudut pandang manapun.

Contoh dari teknik *environment mapping* pada dunia nyata dapat diilustrasikan dengan sebuah mobil yang diparkir di depan halaman sebuah rumah. Jika dilihat dari atas maka atap permukaan mobil memantulkan awan dan langit dimana mobil itu ada, jika dilihat bagian sisi mobil yang menghadap pintu rumah, maka akan dipantulkan pintu dan benda – benda lain yang ada di hadapannya. Jika dilihat pada sisi lainnya maka dapat terlihat pepohonan yang ada dihadapannya.

2.7 Analisis dan Perancangan Berorientasi Objek

Pada awalnya terdapat dua pendekatan yang digunakan dalam melakukan analisis perangkat lunak, yaitu analisis berorientasi data dan berorientasi aksi. Analisis berorientasi data lebih mengutamakan pada data daripada aksi, begitu juga sebaliknya, analisis berorientasi aksi lebih mengutamakan pada aksi daripada data.

Kelemahan dari kedua jenis pendekatan ini adalah data dan aksi tidak dapat dipisahkan pada kenyataannya. Suatu *item* data tidak dapat berubah kecuali dilakukan aksi pada *item* data tersebut. Demikian juga dengan aksi tanpa adanya data yang berhubungan tidak berarti apa – apa.

Karena kelemahan diatas, maka munculah pendekatan berorientasi objek yang memberi penekanan yang sama antara data dan aksi. Dalam analisis berorientasi objek, data disebut sebagai atribut dan aksi sebagai *methods*.

Analisis berorientasi objek saat ini lebih sering dipakai dalam mengembangkan perangkat lunak. Selain karena faktor tersebut diatas, faktor pendorong lainnya dalam penggunaan analisis berorientasi objek, diantaranya karena analisis ini memberi kemudahan dalam pengembangan perangkat lunak karena analisis ini menggunakan pendekatan yang "membungkus" kerumitan yang ada. Selain itu dengan menggunakan analisis ini, pengembang diberi kemudahan dalam melakukan pemeliharaan karena kode program dibagi ke dalam modul – modul sesuai dengan objeknya, sehingga jika dilakukan perubahan pada suatu objek tidak banyak mempengaruhi objek - objek lainnya.

Notasi UML[3] digunakan dalam analisis berorientasi objek yang merupakan representasi permasalahan dengan bentuk Diagram *Use Case*, skenario, Diagram Kelas dan Diagram Transisi.

Diagram *use case* memberi gambaran mengenai fungsi – fungsi yang ada pada perangkat lunak secara umum, selain itu juga dapat memberi gambaran interaksi antara kelas – kelas yang ada.

[3] Unified Modeling Language

Skenario memberikan gambaran umum interaksi yang terjadi antara pengguna dan objek yang ada pada perangkat lunak, seperti yang telah dijabarkan dalam diagram *use case*.

Dalam membangun perangkat lunak menggunakan pendekatan berorientasi objek, kelas – kelas yang dibutuhkan direpresentasikan dalam diagram kelas. Tujuannya adalah untuk menentukan kelas – kelas yang dibutuhkan, atribut-atribut dari setiap kelas – kelas tersebut, dan hubungan antara kelas – kelas tersebut.

Diagram transisi digunakan untuk menjelaskan perubahan *state* (keadaan). Perubahan apa yang terjadi, penyebab perubahan tersebut, dan kapan perubahan tersebut terjadi.

Dalam membangun perangkat lunak dengan menggunakan analisis berorientasi objek, selalu diikuti dengan perancangan. Seperti halnya dengan analisis, perancangan berorientasi objek juga menggunakan notasi UML.

Perancangan berorientasi objek dilakukan dalam beberapa tahapan, yaitu : pembuatan diagram interaksi yang terdiri dari diagram sekuen dan diagram kolaborasi, pembuatan diagram kelas rinci, pembuatan diagram klien – objek, dan pembuatan rancangan rinci perangkat lunak.

Diagram interaksi terdiri atas diagram sekuen dan diagram kolaborasi. Pada dasarnya diagram sekuen dan diagram kolaborasi memiliki isi yang tidak jauh berbeda. Perbedaannya hanya terletak pada tujuan pembuatannya. Diagram sekuen lebih menekankan pada kronologi pengiriman pesan – pesan. Sedangkan diagram kolaborasi lebih menekankan pada relasi antar objek.

Diagram sekuen dan diagram kolaborasi dibuat berdasarkan pada skenario yang telah disusun pada tahapan analisis beroreintasi objek, sehingga untuk setiap skenario akan dibuatkan diagram sekuen dan diagram kolaborasinya.

Pada tahap analisis, diagram kelas hanya menggambarkan kelas – kelas dan atribut – atributnya saja, belum menggambarkan aksi dan metodenya. Pada tahap perancangan diagram kelas akan dijelaskan secara lebih lengkap, mulai dari kelas – kelas yang ada, atribut dari setiap kelasnya hingga metodenya.

Suatu objek yang mengirimkan suatu pesan ke objek lainnya, akan menjadi *client*. Sedangkan suatu objek yang menerima pesan dari objek lainnya akan menjadi *object*. Dalam diagram klien-objek, pesan yang dikirimkan oleh *client* direpresentasikan dengan anak panah dari *client* menuju ke *object*.

2.8 Frames Per Seconds (FPS)

Frames Per Seconds adalah frame rate atau banyaknya frame gambar yang dapat di tampilkan setiap detiknya.

Frames Per Seconds dalam dunia komputer grafik berbeda dengan Frames Per Seconds dalam dunia animasi atau film. Dalam animasi atau film, FPS sudah ditentukan oleh pembuat animasi / film, berapa Frame yang akan di tampilkan dalam satu detiknya. Sedangkan dalam komputer grafik banyaknya Frames yang akan ditampilkan dalam setiap detiknya sangat tergantung dari kinerja Hardware yang digunakan.

Jadi dalam komputer grafik, Frames Per Seconds yang ditampilkan tidak ditentukan oleh si pembuat software seperti layaknya dalam animasi atau film, melainkan tergantung kemampuan kinerja hardware untuk menampilkannya.