PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK VISUALISASI OBYEK TIGA DIMENSI DENGAN *HIERARCHICAL B-SPLINES*

TUGAS AKHIR

Oleh :

YETTY ENDARWATI

2694 100 060

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2000

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK VISUALISASI OBYEK TIGA DIMENSI DENGAN *HIERARCHICAL B-SPLINES*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Supeno Djanali, M.Sc.	Rully Soelaiman, S.Kom.
NIP. 130 368 610	NIP. 132 085 802

SURABAYA
Februari, 2000

Karya ini kupersembahkan untuk :

Ibu, Bapak, dik Heny, dik Endri dan dik Sidik

ABSTRAK

ABSTRAK

Untuk merepresentasikan obyek yang terbentuk dari kurva dan permukaan irregular harus dipakai persamaan polinomial parametrik seperti polinom B-Splines. Polinom ini dapat dipakai untuk merepresentasikan dan memanipulasi kurva dan permukaan berbentuk bebas (free form). Beberapa sistem pemodelan dan animasi grafika komputer tiga dimensi manggunakan representasi B-splines untuk kurva dan permukaan karena properti-properti geometrik yang dimiliki seperti kehalusan (smoothness) dan pengontrolan terhadap kontinyuitas parametrik C^n antar bagian.

Fungsi-fungsi spline memainkan aturan dasar pada beberapa analisa numerik dan pemodelan geometrik. Daerah hierarchical spline didefinisikan sebagai liniar span dari tensor produk B-spline dari grid level yang berbeda. Ide dasar dari metode ini sama dengan pendekatan-pendekatan sebelumnya, khususnya untuk konstruksi dari wavelets spline. Wavelets adalah tool matematik untuk fungsi-fungsi dekomposisi secara hierarchi, yang memungkinkan sebuah fungsi untuk dideskripsikan ke dalam bagian-bagian dari keseluruhan permukaan, dan detail yang dimiliki terbentuk dari permukaan luas ke permukaan sempit.

Disini akan diberikan mekanisme seleksi untuk B-Splines, yang menjamin kebebasan linier dengan menyertakan kontrol lokal secara lengkap dari penghalusan yang dapat dilakukan dengan menambah beberapa B-splines. Penyelesaian ruang spline rata mempunyai persamaan kegunaan dengan pendekatan polinomial diskontinyu. Selain itu, hirarki dasar B-Splines adalah stabil lemah, dimana kestabilannya tumbuh secara konstan seperti O(n), dimana n adalah jumlah level-level grid. Lebih jauh lagi, disini akan didefinisikan quasi interpolant yang didasarkan pada adaptasi prinsip seleksi dan yang menyelesaikan pendekatan lokal optimal. Multilevel dan hirarki ruang B-spline ini sangat sesuai untuk approksimasi dan interpolasi data yang diacak. Approksimasi secara iterasi untuk algoritma dimana ruang spline dapat diadaptasi secara lokal ke dalam data yang diberikan.

Tugas akhir ini mengembangkan metode untuk mengotomatisasi pembuatan permukaan B-spline dari himpunan (set) titik kontrol. Hasil dari permukaan hierarachi yang dibuat secara akurat dan ekonomis akan menghasilkan kembali sebuah mesh, yang bebas dari undulasi yang berlebihan pada level-level intermediate dan menghasilkan sebuah gambaran multiresolusi yang tepat untuk animasi dan pemodelan interaktif.

KATA PENGANTAR

KATA PENGANTAR

Segala puji dan syukur saya panjatkan kehadirat Allah SWT. Yang telah memberikan rahmat, hidayah, rizki dan cinta-Nya sehingga penulis dapat menyelesaikan pembuatan Tugas Akhir ini yang berjudul :

PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK

VISUALISASI OBYEK TIGA DIMENSI DENGAN

HIERARCHICAL B-SPLINE

Tugas Akhir ini dikerjakan guna memenuhi salah satu persyaratan akademis mahasiswa Teknik Informatika untuk meraih gelar sarjana Strata satu (S1). Tugas akhir ini memiliki bobot 4 SKS (Satuan Kredit Semester) dan bertujuan untuk memberi bekal pengalaman dalam hal merancang, membuat dan mengimplementasikan perangkat lunak sebagai penerapan ilmu-ilmu yang telah dipelajari di jurusan Teknik Informatika.

Dalam tugas akhir ini penulis mengetengahkan salah satu metode yang digunakan dalam bidang Grafika Komputer untuk merepresentasikan obyek tiga dimensi. Metode *Hierarchical B-Spline* banyak digunakan dalam Perancangan Obyek yang Dibantu Komputer (*Computer Aided Design*) dan sangat efektif untuk rekonstruksi obyek.

Dengan selesainya Tugas Akhir ini, penulis mengucapkan terima kasih yang setulus-tulusnya dan penghargaan yang sebesar-besarnya kepada semua pihak yang telah dengan ikhlas memberikan dukungan dan bantuan baik dalam material maupun non-material serta doa semoga Allah SWT membalas bantuan mereka. Saya ucapkan terima kasih kepada :

1. Dr. Ir. Arief Djunaidy, M.Sc., selaku Ketua Jurusan Teknik Informatika FTI – ITS, yang telah memberikan bimbingan, petunjuk dan pengarahan selama penulis menempuh studi di Jurusan Teknik Informatika FTI – ITS.
2. Dr. Ir. Supeno Djanali, M.Sc., selaku dosen pembimbing I yang telah memberikan bimbingan, petunjuk dan pengarahan kepada penulis dalam upaya menyelesaikan Tugas Akhir ini.
3. Rully Soelaiman, S.Kom., selaku dosen pembimbing II yang telah memberikan bimbingan, petunjuk dan pengarahan serta dukungan dan semangat yang tak ternilai kepada penulis dalam upaya menyelesaikan Tugas Akhir ini.
4. Bapak dan Ibu tercinta dan tersayang yang selalu mengiringi penulis dengan doa yang tak pernah putus, yang selalu menjaga dengan kesabaran dan keikhlasan, adik-adikku, dik Heny, dik Endri, dan dik Sidik yang saya sayangi yang telah menemani dengan doa dan harapan, yang selalu memberi inspirasi dan semangat.
5. Mbah Sis "Jetis", Mbah Putri dan Mbah Kakung, Budhe Tin dan Pakdhe Mul, Pakdhe dan Budhe Wid, Om Edy dan Bulik Indah, Om Yoyok dan Bulik Ndari, Om Bedjo dan Bulik Endang, Om Dayat dan Bulik Heru, Om Kamal dan Bulik Hambar, Om Budi dan Mbak Titik, Om Bakar dan Bulik Rukhayah, Mba' Iyah, Om Roni dan Bulik atas doa, dorongan, semangat dan bantuannya yang tak ternilai. Adik-adikku Ceca "Mumuk", Mbang Ngade, Nova "Kriting", Putri, Dika, Lia, Hanif, De' Oma, Watik, Andri "ano-ano" yang selalu mengubah airmata menjadi canda tawa yang membahagiakan. Semua keluarga besar Boyolali atas dukungan moral dan bantuannya.

6. Intan "Ike" Vesti Kesuma "sponsor tunggalku" atas semua yang telah diberikan baik moril dan material, persahabatan, perhatian, kasih sayang, dorongan, semangat dan doa yang tulus, airmata, keusilan dan kejahilannya. Berbeda bukan berarti harus terpisah, "Wek" tak tunggu wisudamu. Sahabatku "gundhex-gundhex" tersayang Riza "sang moderator" dan Uppi "Keren" yang ngak keren atas persahabatan dan segalanya serta apa yang telah kita lalui bersama. Sal "dot" Man atas kursus kilat Visual C-nya. Warga Bumi Marina I'im atas "tumpangan rumahnya", Ika atas kesabaran dan nasehatnya dan Shanty, semoga kompak terus. Dina, Maya atas apa yang telah dilalui bersama.
7. Seluruh warga C0A yang selalu kompak dan dengan setia menemani, mengganggu dan memberi semangat, terutama untuk : 'presiden AJK' Tom, Bambang 'brengozz', Waswib, Itonk, Putut, pak YY, si Par "komting irigasi", Dedy, Nurul hadi, mas Jabier, Anjeb, Sutama, Agussa, Agus B, Dewa, Fajar, Arya, Yoyok, Ahmad 'server' Hadi, Oom, Liek, Nanang, Sigit, Darno, Aris, Firdaus, Anton, Seger, Wahyudi, Arthur, Yudi "kodok", Andi "kodok", mbah Noer, Juhar, Gusmang, Dudi atas bantuan *compile*-nya, Hamsy atas idenya, Ableh, Arif, Toni, pak Oedjoe, Firman, Yance atas bantuan ngetiknya, Yanuar kus, Wayan atas komputernya, Lusi, Bilqis, Rita, Luna, Myrna, Nonik, Linda, Astri, dan semua rekan-rekan C0A yang tidak bisa saya sebutkan satu persatu.
8. Semua rekan-rekan di lab. SISFO : Mas Ary atas "tamparan" *readme*-nya, Mas Bowo yang bersama sama berjuang, Pramu, Fauzi, Royyan, Anib, dan yang lainnya atas keusilan dan kejahilannya.

9. Seluruh staf dosen Teknik Informatika : Mas Yudi, Mas Sugeng, Pak Mu'in, Cak Sis, Mas Soleh, Mas Khodir, mbak Erna, Dwi AJK. Terima kasih atas segala bantuannya selama ini.

Akhirnya penulis berharap ide dasar dari Tugas Akhir ini dapat memberikan manfaat dan inspirasi kepada kita semua. Penulis menyadari bahwa masih terdapat kekurangan dalam menyusun Tugas Akhir ini. Untuk itu saran dan kritik yang membangun sangat penulis harapkan guna menambah manfaat serta mengurangi kelemahan dan kekurangan yang ada.

Surabaya, Pebruari 2000

Penulis

Yetty Endarwati

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemodelan obyek saat ini sudah banyak dikembangkan, bahkan sudah sampai pada obyek majemuk, dengan menggunakan metode yang beragam pula. Pemodelan obyek adalah metode pembentukan atau penyusunan suatu obyek dengan menggunakan aturan-aturan yang ada dalam komputer grafik. Visualisasi obyek, yang merupakan metode untuk menampilkan obyek yang sudah dimodelkan, saat inipun sudah banyak dikembangkan.

Salah satu cara mendapatkan obyek adalah dengan menggunakan persamaan parametrik. Polinomial parametrik seperti *B-Splines* dapat digunakan untuk memperoleh berbagai macam bentuk dan permukaan. Terdapat beberapa mekanisme dari *B-Splines*, yang menjamin kebebasan linier pada saat kontrol lokal dari *refinement* lengkap, yang dapat dilakukan adalah dengan membagi ke dalam beberapa *B-splines*. Proses ini menghasil daerah *spline* yang halus dan mempunyai properti approksimasi yang sama dengan approksimasi polinomial diskontinyu.

Beberapa sistem pemodelan dan animasi komputer grafik 3 dimensi menggunakan representasi B-spline untuk kurva dan permukaan karena memiliki properti geometrik seperti kehalusan dan pengontrolan parameter C^n kontinyu antar bagian (*patch*). Bagaimanapun juga, tensor produk alami dari parameter yang ada akan mempersulit untuk melakukan konstruksi secara lengkap dari permukaan yang kontinyu dimana bagian-bagian yang berbeda mempunyai perbedaan jumlah lokal detail.

Pembuatan permukaan dengan formula yang bebas merupakan pekerjaan yang menantang dalam sistem pemodelan geometrik. Permasalahan yang muncul adalah pengkonversian himpunan titik yang kabur ada menjadi model geometrik yang berguna, yang dipisah menjadi rekonstruksi permukaan.

Pada tugas akhir ini, diberikan prosedur untuk rekonstruksi sebuah tensor produk permukaan B-spline dari sebuah himpunan titik-titik 3D. Permukaan B-spline akan didefinisikan dengan menggunakan sebuah konstruksi *surface spline*, dan menunjukkan bahwa beberapa pendekatan membawa ke suatu prosedur yang efisien untuk mencocokkan permukaan.

1.2 Perumusan Masalah

Permasalahan yang dihadapi adalah bagaimana membuat pemodelan obyek dengan menggunakan *Hierarchical B-spline*, untuk kemudian memvisualisasikannya. Permasalahan yang ada termasuk bagaimana membuat multiresolusi dari obyek, yang merupakan faktor sangat penting dalam rekonstruksi obyek.

Untuk sampai pada perangkat lunak yang mampu memvisualisasikan obyek geometri tiga dimensi terdapat beberapa sub-problem :

- Desain sistem yang sesuai untuk perangkat lunak yang dibuat.
- Struktur dan rancangan antarmuka yang interaktif untuk representasi kurva dan permukaan yang akan diciptakan dan dimanipulasi.
- Proses penciptaan ruang pandang tampak atas, tampak samping, tampak depan dan tampak tiga dimensi dengan memakai fungsi-fungsi *Library* yang disediakan OpenGL.

- Pemilihan metode pewarnaan dan pencahayaan yang sesuai dengan memakai fungsi-fungsi *library* yang disediakan OpenGL.

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah pembuatan perangkat lunak yang mampu memvisualisasikan obyek-obyek berbasis permukaan dan memberikan fasilitas secukupnya untuk melakukan rekonstruksi terhadap obyek tersebut.

Manfaat dari penelitian adalah memberikan alternatif pemodelan obyek berbasis permukaan, dan memberikan visualisasi dari obyek tersebut. Selanjutnya dapat digunakan dalam rekonstruksi obyek.

1.4 Pembatasan Masalah

Dalam Tugas Akhir ini dibahas cara penciptaan obyek dengan metode *Hierarchical B-Splines* sehingga diperoleh sebuah obyek tiga dimensi yang kompleks, berbentuk permukaan. Pada prinsipnya obyek permukaan dibentuk melalui pembuatan kurva-kurva terlebih dahulu berdasarkan mesh-mesh yang terbentuk sesuai dengan hirarki yang ada.

Obyek yang dapat dibentuk oleh perangkat lunak ini adalah obyek-obyek yang *irregular*. Sedangkan untuk obyek-obyek geometri seperti bola, tabung, kubus dan yang lainnya belum dapat dibuat.

Visualisator yang dihasilkan menggunakan teknik pewarnaan dan pencahayaan dari *library* OpenGL akan tetapi tidak sampai menggunakan *shadow*, *texture mapping* dan *trimming* karena visualisator berorientasi pada cara-cara alternatif dalam penciptaan obyek tiga dimensi.

1.5 Sistematika Pembahasan

Pembahasan dalam Tugas Akhir ini dilakukan bagian per bagian. Pembagian pokok bahasan disusun dengan metode induksi yaitu pembahasan mulai dari masalah-masalah yang umum, menuju ke masalah yang lebih spesifik, sesuai dengan tema bahasan. Adapun uraian singkat mengenai bab-bab selanjutnya adalah sebagai berikut :

Bab I menjelaskan mengenai latar belakang permasalahan, batasan masalah, tujuan dan sasaran pembahasan serta sistematika pembahasan keseluruhan Tugas Akhir ini.

Bab II membahas mengenai teori-teori penunjang Tugas Akhir ini yaitu definisi dan properti fungsi basis B-Spline, definisi dan properti kurva dan definisi *Hierarchical B-Spline*, dan approksimasi multi resolusi *B-Splines*.

Bab III membahas mengenai konsep dasar OpenGL sebagai library yang berfungsi sebagai alat bantu di bidang ilmu komputer grafik untuk merancang, menspesifikasikan berbagai macam obyek dan menghasilkan aplikasi tiga dimensi yang interaktif.

Bab IV menguraikan struktur data dan algoritma yang dipilih untuk mengimplementasikan konsep-konsep yang telah diuraikan dalam bab-bab terdahulu.

Bab V Hasil uji coba dan evaluasi perangkat lunak yang dibentuk berdasar Bab II dan Bab III diatas.

Bab VI merupakan kesimpulan dan saran dari keseluruhan Tugas Akhir ini.

BAB II

PEMBENTUKAN KURVA DAN PERMUKAAN DENGAN *HIERARCHICAL B-SPLINES*

BAB II

PEMBENTUKAN KURVA DAN PERMUKAAN DENGAN *HIERARCHICAL B-SPLINES*

Untuk merepresentasikan obyek yang terbentuk dari kurva dan permukaan *irregular* harus dipakai persamaan polinomial parametrik seperti polinom B-Splines. Polinom ini dapat dipakai untuk merepresentasikan dan memanipulasi kurva dan permukaan berbentuk bebas (*free form*).

Beberapa sistem pemodelan dan animasi grafika komputer tiga dimensi manggunakan representasi B-splines untuk kurva dan permukaan karena properti-properti geometrik yang dimiliki seperti kehalusan (*smoothness*) dan pengontrolan terhadap kontinyuitas parametrik C^n antar bagian. Tetapi, produk tensor alami dari parameterisasi membuat hal ini sulit dilakukan untuk membuat permukaan yang kompleks dan kontinyu dimana untuk region yang berbeda mempunyai jumlah lokal detail yang berbeda.

Selanjutnya untuk model-model yang lebih komplek, akan sulit untuk membuat perubahan *broad-scale* ke model permukaan tanpa distorsi atau perubahan detail-detail dalam skala yang kecil permukaan. Hal ini meningkatkan waktu dan cost dari model-model free form untuk pekerjaan seperti animasi wajah.

Obyek tiga dimensi baik kurva maupun permukaan tersusun dari komponen-komponen berikut ini:

1. Titik kontrol, salah satu karakteristik dari obyek yang dibentuk dengan B-Spline adalah ditentukan oleh posisi dari sekumpulan titik yang disebut titik

kontrol dan koneksi antar titik tersebut membentuk poligon. Dengan menggerakkan salah satu dari titik maka akan dihasilkan perubahan yang bersifat lokal, dalam artian hanya mempengaruhi titik yang disekitarnya saja dan tidak mempengaruhi seluruh bentuk obyek.

2. Bobot (*weight*) dari tiap titik kontrol, jika pada umumnya bobot = 1 maka tiap titik kontrol mempunyai pengaruh yang sama terhadap bentuk dari obyek. Menaikkan nilai dari berat akan menggangkat kurva sampai mendekati titik kontrol dan sebaliknya.
3. *Knots vektor*, faktor ini menyebabkan beberapa titik kontrol akan mempunyai pengaruh yang lebih kuat dari yang lain seperti tinggi maksimum dari kurva.
4. Fungsi basis, tiap titik kontrol mempunyai fungsi basis yang menentukan seberapa kuat pengaruh titik kontrol tersebut terhadap kurva dan permukaan selama waktu tertentu.

2.1 Definisi dan Properti dari Fungsi Basis dari *B-Spline*[1]

Jika $U = \{u_0, \ldots, u_m\}$ adalah suatu urutan bilangan real yang tidak menurun dengan bentuk $u_i <= u_{i+1}$, i = 0,...,*m-1*, maka u_i disebut knot ke-i dan U adalah vektor knot. Fungsi basis B-Spline yang ke-i dengan pangkat ke-p (order p+1) dinotasikan dengan $N_{i,p}$ (u) yang didifinisikan sebagai berikut :

$N_{i,0} = 1$ jika $u_i \le u \le u_{i+1}$

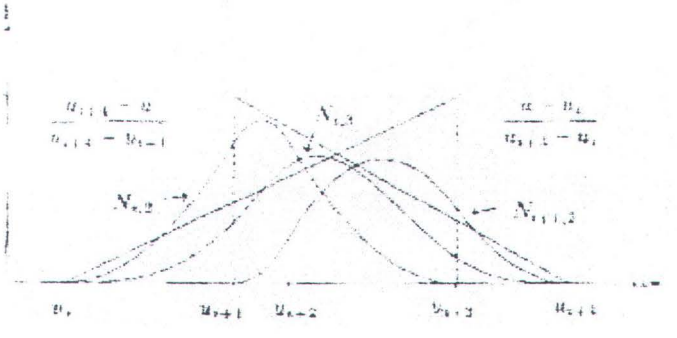
$N_{i,0} = 0$ selain kondisi diatas

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

[1] Pielg Les, Tiller W, "*The NURBS Book (Second Edition)*", Springer, 1995, halaman 50

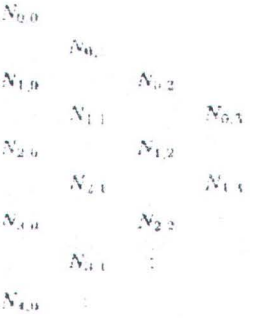
dengan catatan bahwa :

- *Ni,0(u)* dimana saja adalah nol kecuali pada interval u∈ $[u_i, u_{i+1}]$
- Untuk p > 0, $N_{i,p}(u)$ adalah kombinasi linear dari dua fungsi basis dengan pangkat ($_p$-1) seperti pada gambar 2.1.



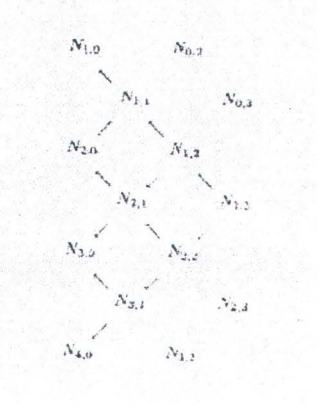
Gambar 2.1. Fungsi basis *B-Spline* yang rekursif

- Untuk menghitung satu set fungsi basis maka dibutuhkan inputan dari vektor *U* dan pangkat *p*.
- Rumus 2.1. dapat menghasilkan pembagian dengan 0 dan untuk seterusnya hasil baginya dianggap 0.
- Interval $[u_i, u_{i+1})$ dinamakan dengan knot span yang ke-*i*, dapat mempunyai panjang 0, karena masing-masing knot tidak harus berbeda, boleh sama asal urutannya tidak menurun.
- Perhitungan komputasi fungsi basis yang ke-*p* menghasilkan tabel segitiga seperti yang ditunjukkan ilustrasi sebagai berikut:



Properti yang paling berguna untuk menentukan karakteristik geometri dari kurva dan permukaan yang dibentuk dari fungsi basis *B-Spline* adalah sebaga berikut:

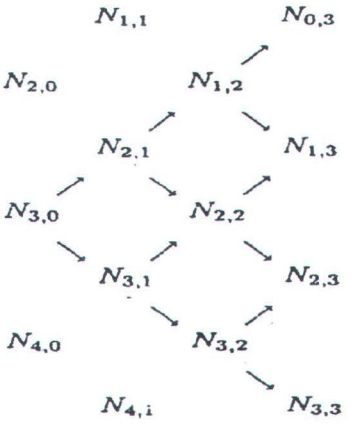
Properti 2.1. $N_{i,p}$ = 0 jika u adalah di luar interval $[u_i, u_{i+p+1})$. Ini yang sering disebut dengan *local suport* dan dapat diilustrasikan dengan tabel segitiga sebagai berikut:



$N_{1,3}$ adalah kombinasi dari $N_{1,0}$, $N_{2,0}$, $N_{3,0}$, dan $N_{4,0}$. Selanjutnya $N_{1,3}$ tidak sama dengan 0 jika $u \in [u_1, u_5)$

Properti 2.2. Pada sembarang knot span $[u_j, u_{j+1})$ maka fungsi yang tidak nol adalah $N_{j-p,p}, \ldots, N_{j,p}$. Misal pada interval $[u_3, u_4)$ fungsi pangkat 0 yang tidak sama dengan nol adalah $N_{3,0}$, sehingga fungsi cubic

yang tidak nol pada interval tersebut adalah $N_{0,3},\ldots,N_{3,3}$ seperti pada ilustrasi berikut :



Properti 2.3 $N_{i,p}(u) \geq 0$ untuk sembarang *I, p* dan *u* (tidak negatif). Ini dibuktikan dengan induksi pada *p*. Jika untuk $p = 0$ benar maka diasumsikan benar juga untuk *p*-1, $p \geq 0$ dengan sembarang *i* dan *u* sesuai rumus :

$$N_{i,p}(u) = \frac{u-u_i}{u_{i+p}-u_i} N_{i,p-1}(u) + \frac{u_{i+p+1}-u}{u_{i+p+1}-u_{i+1}} N_{i+1,p-1}(u) \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

Sesuai Properti 2.1, $N_{i,p,0} = 0$ jika $u \notin [u_i, u_{i+1})$ tetapi untuk $u \in [u_i, u_{i+1})$ maka

$$\frac{u-u_i}{u_{i+p}-u_i}$$

adalah tidak negatif. Dengan asumsi $N_{i,p-1}$ positif maka suku pertama juga positif. Demikian halnya dengan suku ke dua sehingga Ni,p tidak negatif.

Properti 2.4 Tiap fungsi basis *B-Splines*, $N_{i,p(u)}$ adalah tidak negatif untuk tiap u dan jika semua dijumlah hasilnya sama dengan 1 yang dikenal dengan istilah *partition of unity*

$$\sum_{i=0}^{n} N_{i,p}(u) = 1$$

untuk tiap $u \in [u_0, u_{p+n+1}]$.

Ada beberapa jenis knot vektor yang dalam penggunaannya masing-masing sangat mempengaruhi bentuk dari fungsi basis *B-Spline* dan kurva *B-Spline*. Dan satu-satunya syarat yang harus dipenuhi adalah urutannya tidak menurun dan memenuhi syarat $u_i \leq u_{i+1}$. Jenisnya adalah sebagai berikut:

1) Uniform knot vektor[2]

Tiap-tiap knot mempunyai selisih atau jarak yang sama. Contoh:

(0, 1, 2, 3, 4)

(0, 0.25, 0.5, 0.75, 1)

2) Open Uniform knot vektor[3]

Knot vektor yang mempunyai nilai knot yang kembar pada awal dan akhir knot vektor sesuai dengan jumlah pangkat ditambah satu (p+1) dari fungsi basis B-Spline. Sedangkan untuk knot bagian dalam mempunyai jarak atau selisih yang sama. Contoh :

p = 2 (0, 0, 0, 1, 2, 3, 3, 3)

p = 3 (0, 0, 0, 0, 1, 2, 2, 2, 2)

dengan rumus umum:

$$\left.\begin{array}{ll} ui = 0 & 0 \leq i \leq p \\ ui = i\text{-}p & p+1 \leq i \leq n \\ ui = n\text{-}p+1 & n+1 \leq i \leq n+p+1 \end{array}\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.3)$$

[2] Roger, David F,"Mathematical Element For Computer Graphics (Second Edition)", McGraw Hill, New York, 1985, halaman 307

[3] Ibid, halaman 310

3) **Non Uniform knot vektor**[4]

Knot vektor jenis ini sama dengan jenis open uniform hanya saja mempunyai jarak atau selisih yang tidak sama pada knot bagian dalam. Maka untuk mendapatkan jarak yang tidak sama, knot bagian dalam dikalikan dengan jarak antara masing-masing titik polygon (titik kontrol).

$$\left.\begin{array}{ll} u_i = 0 & 0 \le i \le p \\ u_{i+p} = \left[\dfrac{\left[\dfrac{i}{n-p+1}\right]c_i + \sum_{j=0}^{i-1} c_j}{\sum_{i=0}^{n-1} c_i}\right](n-p+1) & 1 \le i \le n-p \\ u_i = n-p+1 & n+1 \le i \le n+p+1 \end{array}\right\} \quad \text{.....(2.4)}$$

Dimana $c_i = |\mathbf{P}_{i+1} - \mathbf{P}_i|$

Maka bentuk umum dari knot vektor untuk no. 2 dan no. 3 adalah sebagai berikut:

$$U = \{\underset{p+1}{a,\ldots,a}, u_{p+1}, \ldots . u_{m-p-1}, \underset{p+1}{b,\ldots,b}\}$$

dimana jumlah knot pertama dan yang terakhir adalah sebanyak p+1. Sedangkan untuk knot vektor yang tidak periodik ada dua properti yang penting berkaitan dengan fungsi basis:

Properti 2.5. Jika m+1 adalah jumlah dari knot maka ada fungsi basis sebanyak n+1 dimana n=m-p-1; $N_{0,p}(a) = 1$ dan $N_{n,p}(b) = 1$.

Properti 2.6. Knot vektor dengan bentuk

[4] Ibid, halaman 328

$$U = \{0,\ldots,0,1,\ldots,1\}$$
$$\quad p+1 \quad p+1$$

akan menghasilkan polinomial Bernstein dengan pangkat *p*.

2.2. Definisi dan Properti dari Kurva *B-Spline*[5]

Kurva B-Spline dengan pangkat ke-p didefinisikan sebagai berikut:

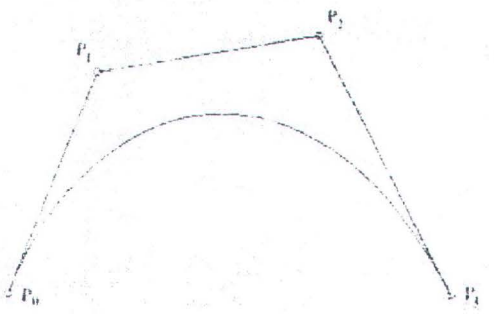
$$C(u) = \sum_{i=0}^{n} N_{i,p}(u) P_i \qquad a \le u \le b \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2.5)$$

dimana $\{\mathbf{P}_i\}$ adalah titik kontrol dan $\{N_{i,p}(u)\}$adalah fungsi basis *B-Spline* dengan pangkat ke-*p*, yang dibatasi dengan knot vektor yang tidak periodik *(non uniform)*

$$U = \{a,\ldots,a,u_{p+1},\ldots.u_{m-p-1},b,\ldots,b\}$$
$$\quad p+1 \qquad\qquad\qquad p+1$$

Yang berjumlah sebanyak *m*+1 knots. Polygon yang terbentuk dari $\{\mathbf{P}_i\}$ Disebut kontrol polygon. Ada beberapa properti yang penting dan mengikuti properti dari fungsi basis *B-Spline* diatas:

Properti 2.7. Jika *n*=*p* dan *U*={0,...,0,1,...,1} maka **C**(*u*) adalah kurva Bezier, seperti Gambar 2.2.



Gambar 2.2. Cubic *B-Spline* dengan *U* ={0,0,0,0,1,1,1,1}

Properti 2.8. **C**(*u*) adalah kurva yang berbentuk *piecewise polynomials* (fungsi yang disusun dari sekumpulan polinomial yang ditetapkan oleh

[5] Pielg Les, Tiller W, "The NURBS Book (Second Edition)", Springer, 1995, halaman 81

interval-interval yang bersebelahan dan tergabung untuk membentuk kurva yang kontinyu); pangkat *p*, jumlah titik kontrol *n*+1 dan jumlah knot ditentukan dengan hubungan *m*=*n*+*p*+1. Pada Gambar 2.3(a) dan (b) menunjukkan fungsi basis dan kurva *B-Spline* dimana garis putus-putus dan garis utuh yang saling bergantian menandakan knot-knot yang membentuk kurva.

Properti 2.9. Interpolasi padda titik akhir yaitu $\mathbf{C}(0) = \mathbf{P}_0$ dan $\mathbf{C}(1) = \mathbf{P}_n$.

Properti 2.10 Kurva yang dibentuk dari *B-Spline* adalah *affine invariant*. Untuk mentransformasikan kurva *B-Spline* maka yang ditransformasikan hanyalah titik kontrol dan kurva yang baru dihasilkan dari posisi titik kontrol yang baru. Jika titik $\{\mathbf{C}(u)\}$ pada kurva *B-Spline* akan ditransformasikan ke titik $\{\mathbf{C}'(u)\}$ memakai matriks transformasi **M** dan offset vektor **tr** maka

$$\mathbf{C}'(u) = \mathbf{C}(u)\mathbf{M} + \mathbf{tr} = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{P}_i\mathbf{M} + \mathbf{tr} \qquad \text{.....(2.6)}$$

Yang ditransformasikan hanya titik kontrol dari $\mathbf{C}(u)$ dan kemudian dihitung lagi dengan rumus *B-Spline* untuk menghasilkan kurva *B-Spline* yang ditransformasi. Dan bukti bahwa kurva *B-Spline affine invariant* secara analitik adalah jika tiap titik dikalikan dengan matriks transformasi

$$\sum_{i=0}^{n} (\mathbf{P}_i\mathbf{M} + \mathbf{tr}) N_{i,p}(u) \qquad \text{.....(2.7)}$$

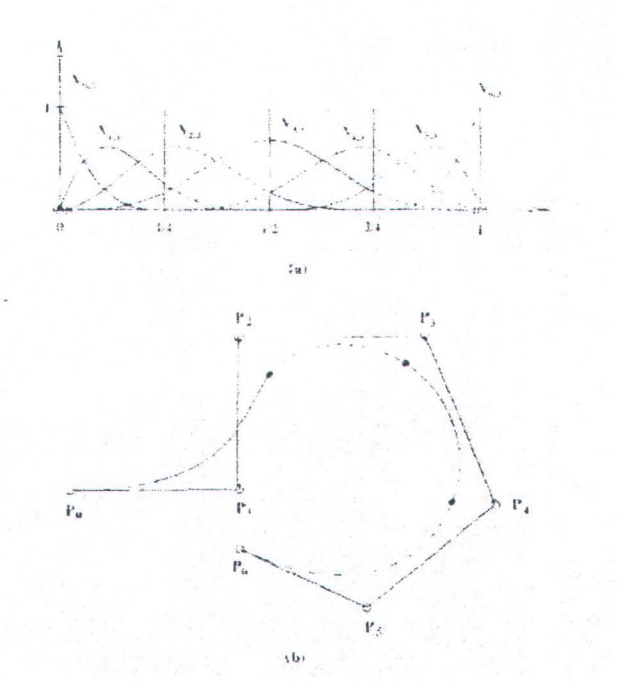
dan kurva baru yang terbentuk sebagai hasil dari transformasi tiap $\mathbf{P}_i$ untuk menunjukkan bahwa rumus (2.7) identik dengan (2.6) maka $N_{i,p}(u)$ dikalikan dengan $(\mathbf{P}_i\mathbf{M} + \mathbf{tr})$

$$\mathbf{C}'(u) = \sum_{i=0}^{n} \mathbf{P}_i \mathbf{M} N_{i,p}(u) + \sum_{i=0}^{n} \mathbf{tr} N_{i,p}(u)$$

Karena properti (Properti 2.4) *partition of unity*

$$\sum_{i=0}^{n} N_{i,p}(u) = 1$$

maka penjumlah yang kedua hanya tinggal **tr** sehingga terbukti.

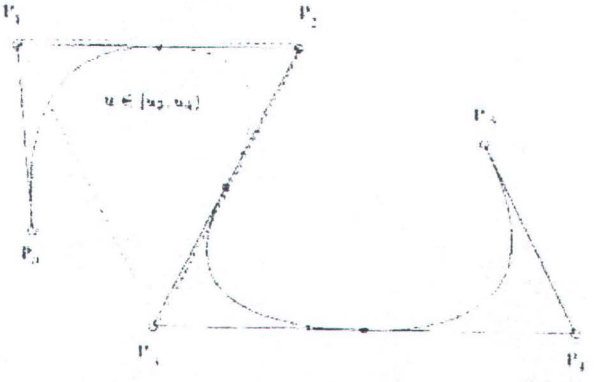


Gambar 2.3(a).Fungsi Basis Cubic dengan *U*={0,0,0,0,1/4,1/2,3/4,1,1,1,1};

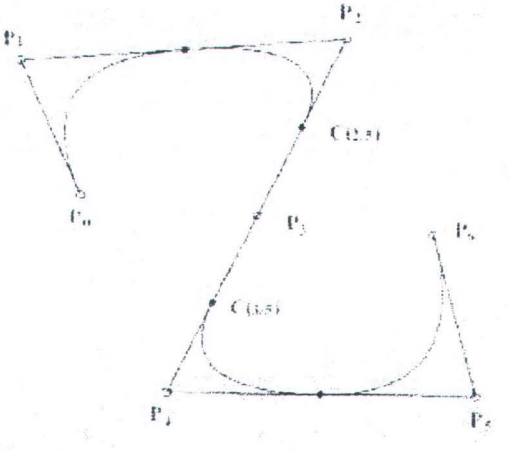
(b) kurva Cubic

Properti 2.11. Properti *convex hull* yang sangat kuat. Kurva berada di dalam *convex hull* yang dibentuk dari kontrol polygonnya sendiri. Pada kenyataannya jika $u \in [u_i, u_{i+1}), p \le i \le m - p - 1$ maka **C**(*u*) berada di dalam *convex hull* dari titik kontrol $\mathbf{P}_{i-p},\ldots,\mathbf{P}_i$ seperti pada Gambar 2.4. Hal ini sesuai dengan properti tidak negatif dan *partition of unity* (Properti 2.3 dan 2.4) dan properti yang menyatakan bahwa $N_{j,p}(u)$ = 0 untuk $j < i-p$ dan $j > i$ jika

$u \in [u_i, u_{i+1})$ (Property 2.2). Sedangkan pada Gambar 2.5 ditunjukkan bagaimana cara membuat kurva quadratic yang mengandung segmen garis lurus, karena $\mathbf{P}_2$, $\mathbf{P}_3$, $\mathbf{P}_4$ adalah *colinear* maka *convex hull* yang kuat memaksa kurva menjadi garis lurus dari **C**(2/5) hingga **C**(3/5).

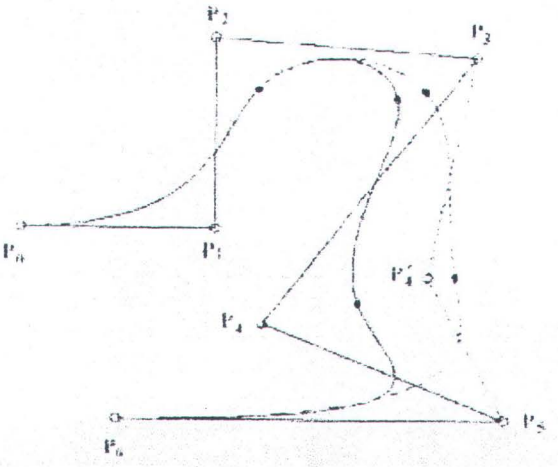


Gambar 2.4 Convex Hull yang kuat dari kurva *B-Spline* Quadratic untuk $u \in [u_i, u_{i+1})$ maka **C**(u) adalah segitiga $\mathbf{P}_{i-2},\ldots,\mathbf{P}_i$.



Gambar 2.5 Kurva quadratik *B-Spline* dengan U = {0,0,0,1/5,2/5,3/5, 4/5,1,1,1}maka kurva adalah garis lurus antara **C**(2/5) dan **C**(3/5).

Properti 2.12. Perubahan setempat yaitu dengan menggerakkan {$\mathbf{P}_i$} maka akan merubah $\mathbf{C}(u)$ hanya pada interval $\lfloor u_i, u_{i+p+1} \rfloor$. Seperti pada Gambar 2.6 hal ini sesuai dengan Properti 2.1 dimana $N_{i,p}(u)$ = 0 untuk $u \notin \lfloor u_i, u_{i+p+1} \rfloor$



Gambar 2.6 Kurva Cubic dengan U = {0,0,0,0,1/4,2/4,3/4,1,1,1,1}; Pergerakan titik $\mathbf{P}_4$ hanya antara interval [1/4,1)

Properti 2.13 Semakin kecil pangkat maka semakin dekat kurva dengan polygonnya seperti Gambar 2.7. Pada gambar tersebut kurva dibentuk dari enam titik kontrol dan knot vektor yang sama.

p = 1 : U = {0, 0, 1/5, 2/5, 3/5, 4/5, 1, 1}

p = 2 : U = {0, 0, 0, 1/4, 1/2, 3/4, 1, 1, 1}

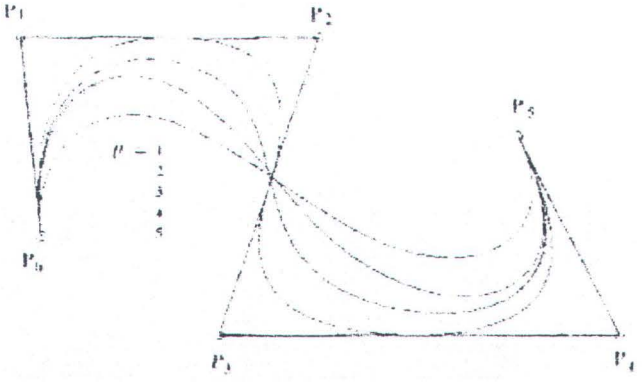
p = 3 : U = {0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1}

p = 4 : U = {0, 0, 0, 0, 0, 1/2, 1, 1, 1, 1, 1}

p = 5 : U = {0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1}

Semakin kecil pangkat maka semakin sedikit titik kontrol yang ikut berperan dalam perhitungan kurva *B-Spline*. Bahkan pada p=1

tiap titik pada **C**(u) adalah interpolasi linier antara dua titik kontrol yaitu polygon itu sendiri.



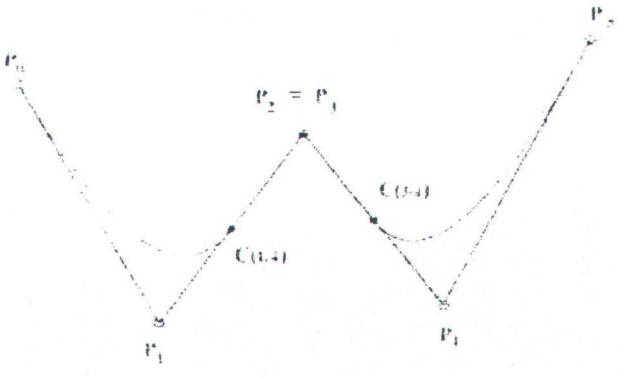
Gambar 2.7. Kurva *B-Spline* dengan pangkat berbeda

Properti 2.14 Titik yang bergerak di kurva sepanjang u = 0 dan u = 1 maka fungsi $N_{i,p}(u)$ berfungsi seperti saklar. Pada Gambar 2.3 ketika u bergerak dari knot satu ke knot yang lain bentuk $N_{i,p}$ mempengaruhi bentuk $\mathbf{P}_i$ (ditandai dengan garis utuh dan garis putus-putus).

Properti 2.15 Properti *variation diminishing*; Tidak ada bidang yang memiliki jumlah perpotongan dengan kurva lebih banyak daripada perpotongannya dengan kontrol polygon (ganti kata bidang dengan garis untuk kurva dua dimensi).

Properti 2.16 Sangat mungkin untuk menggunakan titik kontrol yang bertepatan koordinatnya (sama posisi) seperti Gambar 2.8. yang menunjukkan kurva quadratic dengan titik kontrol ganda $\mathbf{P}_2 = \mathbf{P}_3$. Bagian yang menarik terletak pada **C**(1/4) dan **C**(3/4). Pada **C**(1/2) = $\mathbf{P}_2 = \mathbf{P}_3$ maka segmen kurva yang terbentuk antara **C**(1/4) dan **C**(1/2) dengan **C**(3/4) dan **C**(3/4) berupa garis lurus. Hal ini sesuai

dengan properti 2.5 dimana kurva terletak pada *convex hull* P_1 P_2 P_3 yang berupa garis jika $u \in [1/4, 1/2)$.



Gambar 2.8 Kurva Quadratic dimana titik $P_2 = P_3$

2.3. Definisi dan Properti dari Pemukaan *B-Spline*[6]

Permukaan *B-Spline* dapat dirumuskan sebagai berikut:

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j} \qquad \text{.....(2.8)}$$

Dengan dua macam knot vektor

$$U = \{\underbrace{a,\ldots,a}_{p+1}, u_{p+1},\ldots u_{r-p-1}, \underbrace{b,\ldots,b}_{p+1}\}$$

$$V = \{\underbrace{a,\ldots,a}_{q+1}, u_{q+1},\ldots u_{s-q-1}, \underbrace{b,\ldots,b}_{q+1}\}$$

Knot vektor U dengan jumlah $r + 1$ dan V dengan $s + 1$ dimana $r = n + p + 1$ dan $s = m + q + 1$

Properti dari *tensor product* fungsi basis permukaan *B-Spline* mengikuti properti dari fungsi basis *B-Spline* yaitu:

---

[6] Ibid, halaman 100

Properti 2.17 Tidak negatif: $N_{i,p}(u)N_{j,q}(v) \geq 0$ untuk semua *i, j, k, p, q, u, v*

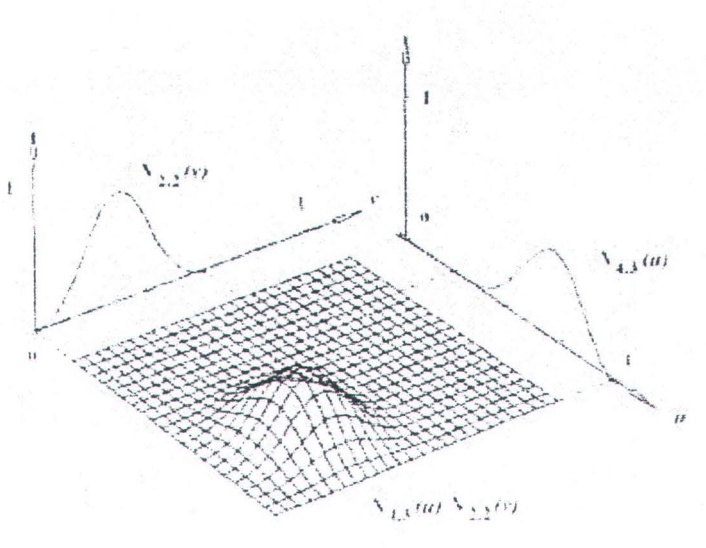
Properti 2.18 *Partition of unity:*

$$\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v) = 1$$

Untuk semua $(u,v) \in [a,b)x[a,b)$

Properti 2.19 Jika *n=p, m=q, U*={0,...,0,1,...1} dan *V*={0,...,0,1,...1} maka $N_{i,p}(u)N_{j,q}(v) = B_{i,n}(u)B_{j,m}(v)$ untuk semua *i,j*. Perkalian fungsi *B-Spline* menjadi perkalian polinomial Bernstein.

Properti 2.20 $N_{i,p}(u)N_{j,q}(v) = 0$ jika (*u,v*) berada diluar kotak $[u_i, u_{i+p+1})$ x $[v_i, v_{i+q+1})$ seperti Gambar 2.9.



Gambar 2.9 Fungsi Basis cubic x quadratik untuk $N_{4,3}(u)$ dan $N_{2,2}(v)$ dengan *U*={0,0,0,0,1/4,1/2,3/4,1,1,1,1} dan *V*={0,0,0,1/5,2/5,3/5,3/5,4/5, 1,1,1}

Properti 2.21 Pada knot span antara $[u_{i0}, u_{i0+1})\mathrm{x}[v_{i0}, v_{i0+1})$maka fungsi basis $N_{i,p}(u)N_{j,q}(v)$ yang tidak nol jika $i_0 - p \le i \le i_0$ dan $j_0 - p \le j \le j_0$

Properti 2.22 Jika *n=p, m=q, U* = {0,...,0,1,...,1} dan *V* = {0,...,0,1,...,1} maka **S**(*u,v*) adalah permukaan Bezier sesuai Properti 2.19

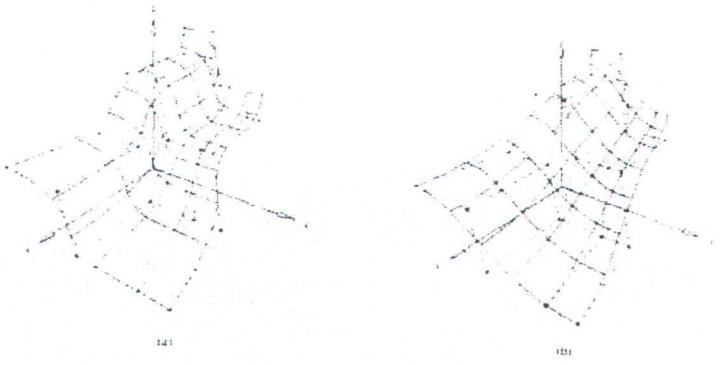
Properti 2.23 Permukaan *B-Spline* melewati empat titik kontrol sudut: **S**(0,0) = $\mathbf{P}_{0,0}$, **S**(1,0) = $\mathbf{P}_{n,0}$,**S**(0,1) = $\mathbf{P}_{0,m}$ dan **S**(1,1) = $\mathbf{P}_{n,m}$. Hal ini sesuai dengan Properti 2.18 dimana

$$N_{i,p}(0)N_{0,q}(0) = N_{n,p}(1)N_{0,q}(0) = N_{0,p}(0)N_{m,q}(1) = N_{n,p}(1)N_{m,q}(1) = 1$$

Properti 2.24 *Affine invariance*; transformasi affine diterapkan pada permukaan dengan menerapkan pada titik kontrol sesuai dengan Properti 2.18.

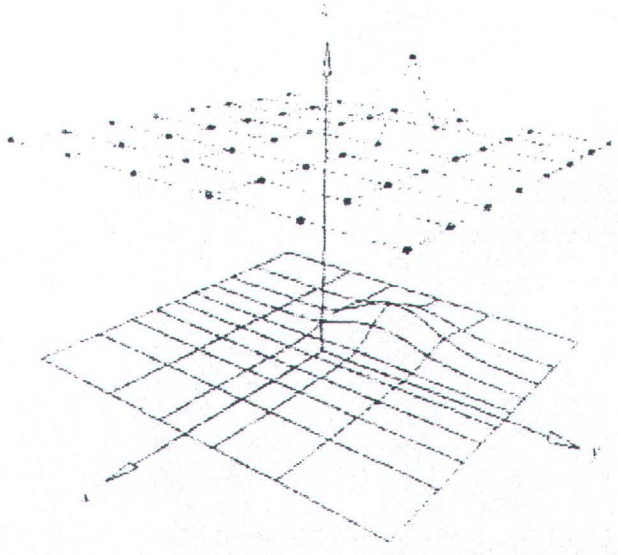
Properti 2.25 *Convex hull* yang kuat: Jika $(u,v) \in [u_{i0}, u_{i0+1})\mathrm{x}[v_{i0}, v_{i0+1})$maka **S**(*u,v*) berada di dalam convex hull dari titik kontrol $\mathbf{P}_{i,j}, i_0 - p \le i \le i_0$dan $j_0 - q \le j \le j_0$ sesuai dengan Properti 2.17, 2.18, 2.21.

Properti 2.26 Jika pangkat semakin kecil maka semakin baik pendekatannya. Pada Gambar 2.10(a) adalah permukaan dengan *p=q*=2 yang sering disebut dengan biquadratic sedangkat pada Gambar 2.10(b) permukaan dengan *p=q*=4 yang disebut dengan biquartic. Masing-masing menggunakan titik kontrol yang sama.



Gambar 2.10(a) Permukaan biquadratik; (b)Permukaan biquartic dengan menggunakan titik kontrol yang sama

Properti 2.27 Perubahan setempat: jika $\mathbf{P}_{i,j}$ digerakkan maka akan mempengaruhi permukaan hanya pada kotak $[u_i, u_{i+p+1}) \times [v_i, v_{i+q+1})$ sesuai dengan Properti 2.14. Pada Gambar 2.11 permukaan pada permulaan berbentuk datar karena semua titik kontrol terletak pada bidang yang sama (Properti 2.19), dan jika $\mathbf{P}_{3,5}$ digerakkan maka bentuk permukaan yang terpengaruh hanya pada kotak [1/4, 1)x[2/5, 1).



Gambar 2.11 Permukaan mula-mula planar dengan U = {0,0,0,1/4,1/2,3/4,1,1,1} dan V = {0,0,0,0,1/5,2/5,3/5,4/5,1,1,1,1}; selanjutnya titik $\mathbf{P}_{3,5}$ digerakkan yang berpengaruh hanya pada [1/4,1)x[2/5,1).

2.4 Definisi *Hierarchical B-Spline*[7]

Hierarchical B-splines adalah representasi permukaan multi resolusi yang mempunyai karakteristik berikut ini[8] :

1. *Refinement* lokal dari produk tensor permukaan *spline* menempatkan detail pada daerah yang membutuhkan saja.
2. Perubahan permukaan multi resolusi yang menerima detail permukaan selama manipulasi permukaan dilakukan secara *broad-scale*.
3. Representasi permukaan yang ekonomis. Karena kontrol point yang diperlukan lebih sedikit untuk merepresentasikan permukaan.

---

[7] Forsey, David and Wong, David, "*Multiresolution Surface Reconstruction For Hierarchical B-Splines*", The Department of Computer Science, The University of British, Columbia, 1995, halaman 2

[8] Ibid, halaman 2

4. Kemampuan animasi multi resolusi yang ada memudahkan untuk membuat animasi permukaan secara lengkap.

Permukaan direpresentasikan sebagai urutan dari level-level, masing-masing terdiri dari kumpulan permukaan *B-spline* $H^{[k]}$ dengan resolusi dua kali dari *parent surface* pada level k-1.

$$H^{[k]}(t,u) = \sum_{i=0}^{m}\sum_{j=0}^{n} V_{i,j}^{[k]} B_{i,\kappa}^{[k]}(t) B_{j,\tau}^{[k]}(u) \quad \text{.....(2.9)}$$

$H^{[k]}(t,u)$ didefinisikan oleh mesh n x m dari kontrol *vertice* $V_{i,j}^{[k]}$, dan fungsi-fungsi basis polinomial $B_{i,\kappa}(t)$ dan $B_{(j,\tau)}(u)$, dari masing-masing derajat κ dan τ. Parameter t dan u berkisar antara nilai minimum t_{min} , u_{min} dan nilai maksimum t_{max} , u_{max} . Nilai tetap dari t dan u, disebut *knots*, yang bersesuaian dengan join antar *patch* polinomial dan memformulasi sebuah urutan knot pada masing-masing parameter $\{t_o, \ldots, t_{m+\kappa}\}$ dan $\{u_o, \ldots, u_{n+\tau}\}$. Fungsi-fungsi dasar diasumsikan memiliki support lokal seperti masing-masing fungsi dasar hanya berpengaruh pada area tertentu dari *spline*.

Apabila digunakan pada representasi permukaan hirarki, basis dibutuhkan untuk proses perbaikan, dengan kata lain masing-masing basis dapat diekspresikan ulang sebagai kombinasi linier dari satu atau lebih fungsi-fungsi basis yang lebih kecil dengan penambahan *knots*.

Apabila $M^{[0]}$ merupakan *initial mesh* level 0, didefinisikan oleh kontrol vertices $V^{[k]}$, point-point $R^{[k+1]}$ yang ada dalam mesh yang diperbaiki atau disempurnakan $M^{[1]}$ dikomputasikan dengan kombinasi linier dari $V^{[k]}$. Dituliskan dalam format matrik sebagai berikut :

$$R^{[k+1]} = SV^{[k]} \quad \text{..... (2.10)}$$

Komponen-komponen dari subdevisi matrik S tergantung pada order dan tipe dari fungsi-fungsi dasar, jumlah dan lokasi *knots* yang disisipkan dan topologi permukaan. Pada produk tensor permukaan seperti *B-spline*, perbaikan (*refinement*) dilakukan secara non-lokal pada knots yang ditambahkan pada domain parametrik yang menghasilkan penambahan kolom atau baris baru dari *patches* yang ada pada permukaan yang aktif.

Pada representasi permukaan *hierarchical B-spline*, masing-masing vertek kontrol $V^{[k+1]}$ pada $M^{[k+1]}$ direpresentasikan pada *offset form* berikut ini :

$$V^{[k+1]} = R^{[k+1]} \oplus \vec{O}^{[k+1]} \quad \text{.....(2.11)}$$

dimana $R^{[k+1]}$ diturunkan oleh perbaikan poin tengah (*mid-point refinement*) dari $V^{[k]}$ seperti pada persamaan 2.10. $\vec{O}^{[k+1]}$ adalah vektor offset dan $\oplus$ adalah operator offset.

Posisi $V_{i,j}^{[k]}$ yang berada tepat setelah sebuah level baru dibuat sama dengan $R_{i,j}^{[k]}$ dan $\vec{O}_{i,j}^{[k+1]} = 0$. Beberapa perubahan posisi dari titik kontrol $V_{i,j}^{[k]}$ direpresentasikan pada vektor offset $\vec{O}_{i,j}^{[k+1]}$ sebagai suatu perubahan posisi relatif terhadap poin referensi $R_{i,j}^{[k+1]}$. Perubahan-perubahan ke $V_{i,j}^{[k+1]}$ (dan ke $\vec{O}_{i,j}^{[k+1]}$) tidak mengubah posisi dari poin referensi. Dengan kata lain, perubahan yang terjadi pada *parent surface* mengubah $R_{i,j}^{[k+1]}$, dan menghasilkan perpindahan $V_{i,j}^{[k+1]}$.

Form referensi dan offset memperbolehkan sebuah permukaan *hierarchical B-spline* diperbaiki secara lokal sehingga patch-patch dapat

ditambahkan ke region tertentu dari permukaan, dan meningkatkan performa dari representasi karena offset-offset yang tidak nol (*non-zero offset*) mendefinisikan model permukaan secara total.

Offset-offset berkoresponden ke lainnya antara model dari permukaan pada dua level dari representasi. Operator offset menentukan bagaimana perbedaan tersebut diinterpretasikan. Pada kasus sederhana operator yang digunakan adalah penambahan vektor yang sederhana, dan operasinya hampir sama dengan dekomposisi wavelet. Tetapi operator offset dapat menjadi komplek dan digunakan untuk menambah performa dari permukaan, sehingga feature-feature detail mengikuti deformasi *broad scale* dari permukaan atau kerangka (*underlying skeleton*).

Pendekatan hirarki tidak terbatas pada *B-spline* tetapi dapat diaplikasikan pada beberapa representasi dimana fungsi-fungsi basis mempunyai support lokal dan dapat disempurnakan (*refinable*).

2.5 Approksimasi multi resolusi *B-Spline*[9]

Terdapat inisialiasi kontrol mesh $M^{[k+1]}$ dengan point-point $V^{[k+1]}$ mendefinisikan permukaan *B-spline*, ide dasar yang digunakan adalah men-*generate* suatu mesh $M^{[k]}$ yang halus untuk $H^{[k]}$, dimana, pada saat proses *refine*, meminimalkan magnitude dari offset pada $H^{[k+1]}$. Perhitungan dilakukan pada mesh kontol, bukan pada beberapa data poin yang mungkin digunakan untuk membuat $M^{[k+1]}$.

---

[9] Ibid, halaman 5

2.5.1 Inisialisasi Approksimasi[10]

Meminimalkan offset dilakukan dengan melakukan dengan rumusan

$$\left\| SV^{[k]} - V^{[k+1]} \right\|^2 \quad \text{.....(2.12)}$$

yang digunakan untuk penyelesaian sistem linier

$$Ax = b \quad \text{..... (2.13)}$$

dimana x adalah vektor kolom dari vertice-vertice kontrol yang tidak diketahui $V^{[k]}$, A ditentukan oleh sub devisi matrik *S*, dan *b* adalah vektor kolom dari $V^{[k+1]}$.

2.5.2 Konstrain Kehalusan[11]

Untuk mengontrol hasil dari persamaan 2.13, digunakan konstrain kehalusan (*smoothness constrain*) selama approksimasi. Terdapat berbagai macam aturan dan yang digunakan adalah model template sederhana yang meminimalkannya

$$\iint \lambda \left(\left\| V_{uu}^{[k]} \right\|^2 + 2 \left\| V_{uv}^{[k]} \right\|^2 + \left\| V_{vv}^{[k]} \right\|^2 \right) dudv \quad \text{.....(2.14)}$$

dimana λ adalah parameter regulasi yang mengontrol kekerasan dari piringan (*plate*). Kontrol mesh diperlakukan sebagai grid dari poin-poin sample pada permukaan kontrol mesh. Secara diskrit dapat dituliskan sebagai berikut :

$$\sum_{i=0}^{m} \sum_{j=0}^{n} \lambda \left(\left(V_{i,j}^{[k]} \right)_{uu}^2 + 2 \left(V_{i,j}^{[k]} \right)_{uv}^2 + \left(V_{i,j}^{[k]} \right)_{vv}^2 \right) \quad \text{.....(2.15)}$$

dimana

λ adalah parameter regulasi.

---

[10] Ibid, halaman 5
[11] Ibid, halaman 5

$\left(V_{i,j}^{[k]}\right)_{uu}^2$ adalah derivasi uu kedua yang dikomputasi dengan pembeda tengah (*central differencing*) [1 -2 1].

$\left(V_{i,j}^{[k]}\right)_{uv}^2$ adalah derivasi uv kedua yang dikomputasikan dengan pembeda kedepan (*forward differencing*) $\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$.

$\left(V_{i,j}^{[k]}\right)_{vv}^2$ adalah derivasi vv kedua yang dikomputasikan dengan pembeda tengah (*central differencing*) $[1\ -2\ 1]^T$.

Komponen-komponen tersebut dikumpulkan kedalam matrik energi kehalusan (*smoothness energy matrix*) *P*, yang dipasangkan kedalam sistem linier untuk tiap-tiap koordinat x, y dan z.

Perumusan berikut ini digunakan untuk meminimalkan offset-offset dan energi yang didapat

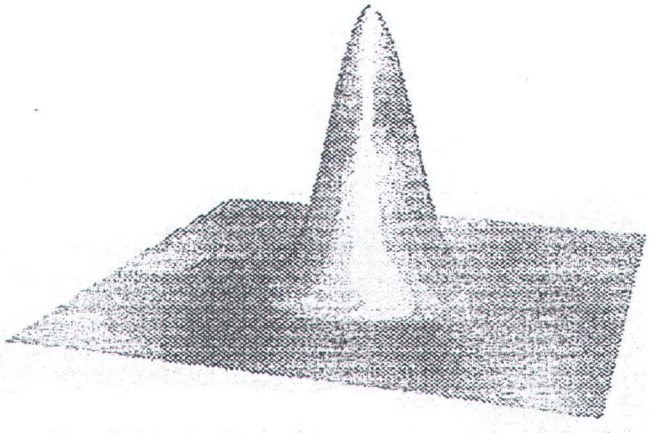
$$\left\|SV^{[k]} - V^{[k+1]}\right\|^2 + \lambda\left\|PV^{[k]}\right\|^2 \quad \text{.....(2.16)}$$

dimana dapat dituliskan sebagai penjelasan lebih lanjut permasalahan *least square*

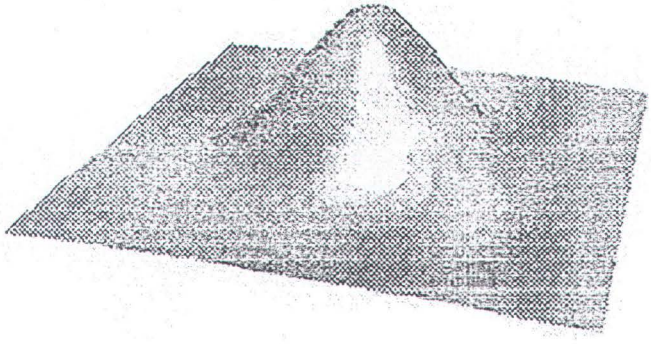
$$\begin{pmatrix} S \\ \sqrt{\lambda P} \end{pmatrix} V^{[k]} = \begin{pmatrix} V^{[k+1]} \\ 0 \end{pmatrix} \quad \text{.....(2.17)}$$

2.5.3 Approksimasi Bobot[12]

Diaplikasikan pada mesh / mata jala yang datar kecuali untuk ujung extrim tunggal (gambar 2.12), metode di atas menghasilkan permukaan pada gambar 2.13 (dengan $\lambda = 0.1$). Ini adalah penyelesaian yang bagus, karena hasilnya rata dan tidak menunjukkan adanya bentuk gelombang dari penyelesaian *least-square* yang tidak berhias, tapi tidak menangkap pengertian dari permukaan sederhana dengan bentuk-bentuk tambahan.



Gambar 2.12 Permukaan awal dengan titik kontrol 19X19



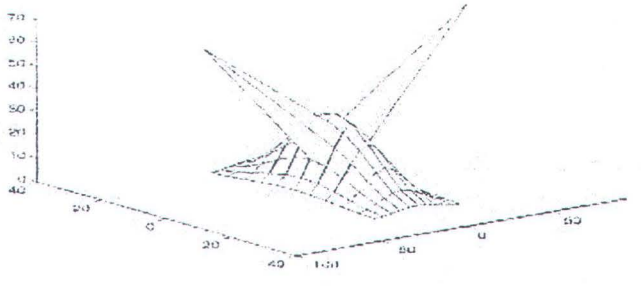
Gambar 2.13. Approksimasi dengan $\lambda = 0.1$

[12] Ibid, halaman 6

Untuk mengidentifikasi bentuk-bentuk ini, derivatif parsial kedua uu dan vv (dengan pendekatan kurva) dari setiap vertex dalam $V^{[k+1]}$ dihitung (menggunakan central differencing). Hasil-hasilnya dinormalisasi dan digunakan sebagai berat w_{ii} untuk setiap kontrol vertex sebagai indikator dari pentingnya vertex tersebut dalam pendekatan ini. Semakin tinggi kurva, semakin sedikit kegunaan vertex tersebut dalam menentukan bentuk pendekatan. Perubahan dari persamaan (2.16) adalah sebagai berikut :

$$\begin{bmatrix} ws \\ \sqrt{\lambda p} \end{bmatrix}^{V^{[k]}} = \begin{bmatrix} wv[k+1] \\ 0 \end{bmatrix} \quad \text{.....(2.18)}$$

sebagai contoh, gambar 2.14 menunjukkan kontrol mesh untuk permukaan yang ada, dengan bobot yang sesuai dengan gambar 2.15. Setelah $V^{[k+1]}$ ditentukan, permukaan hirarki dua tingkat diciptakan dengan menggunakan nilai referensi $R^{[k+1]}$ untuk menentukan *offset* yang diperlukan untuk menghitung untuk semua B-spline B_k^p lakukan $V^{[k]}$ dengan persamaan (2.12). Untuk mereduksi total angka dari *offset* yang diperlukan untuk merepresentasikan permukaan, *offset* yang lebih kecil dari toleransi yang diberikan diset menjadi nol. Toleransi ini adalah toleransi yang menentukan seberapa dekat pendekatan ini sesuai dengan *initial mesh*.



Gambar 2.14 Kontrol mesh awal (11x11)

- Untuk mendapatkan tensor produk B-splines pada level-level grid yang berbeda.
- Mekanisme seleksi untuk *B-splines* dilakukan dengan sederhana.
- Menggunakan kontrol lokal dari *refinement*.
- Untuk mendapatkan kebebasan linier dari *B-splines* pada setiap level.
- Dengan derajad TP B-splines (α, α) akan dihasilkan permukaan *B-Spline* $C^{\alpha-1}$.
- Kestabilan lemah.
- Mendapatkan jenis pendekatan lokal optimal.
- Korelasi geometrik antara titik-titik kontrol multilevel *B-spline* dan permukaan *spline*.
- Sangat sesuai untuk penyelesaian dan interpolasi data acak sebaik untuk penyelesaian pendekatan dengan *quasi interpolasi*.

2.6.1 Kondisi-kondisi dalam Batasan Grid dan *B-Splines*[15]

Misalkan sebuah *lattice* dinamis dalam R^2 dengan grid-point ($k_1/2^p$, $k_2/2^p$) terdapat dalam level hirarki p maka *B-splines* B_k^p dengan derajad (α, β) dapat direpresentasikan sebagai berikut[16] :

$$B_k^p(x) = B_{k_1 k_2}^p(x_1, x_2) \quad \text{.....(2.19a)}$$

$$= B_\alpha(2^p x_1 - k_1) B_\beta(2^p x_2 - k_2) \quad \text{.....(2.19b)}$$

dan open support

$$\text{supp} B_k^p = \left(k_1/2^p, (k_1+\alpha+1)/2^p\right) \times \left(k_2/2^p, (k_2+\beta+1)/2^p\right) \quad \text{.....(2.20)}$$

jika $\Omega^0 \supseteq \Omega^1 \supseteq$ ... adalah deret bersarang dari domain dengan properti :

---

[15] Ibid, halaman 3
[16] Ibid, halaman 3

$$\Omega^{p+1} = \bigcup_{j \in J} \text{supp } B_j^p \qquad \text{untuk } J \subseteq Z^2 \quad \text{...(2.21)}$$

$$\partial\Omega^p \cap \partial\Omega^{p+1} = 0$$

dari persamaan (2.21) dapat diperoleh :

$$\Omega^p - \bigcup_{k \in D^p} \text{supp } B_k^p \quad \text{..(2.22)}$$

2.6.2 Kondisi dalam Set Indeks[17]

Dari *B-Splines* B_k^p kita memilih ini tanpa dukungan yang terdiri dari Ω^{p+1} sebagai fungsi basis untuk ruang multiple *B-Splines.*

DEFINISI

Dengan $I^p := [k \in Z^2 \mid \text{supp } B_k^p \subseteq \Omega^p \text{ dan supp } B_k^p \not\subset \Omega^{p+1}$ ....(2.23)

kita membatasi ruang multilevel *B-spline*

$$Sn := S_n^{\alpha,\beta} := \text{span}\left\{B_k^p \mid p \geq 0 \text{ dan } k \in I^p\right\} \quad \text{....................................(2.24)}$$

dengan derajad (α, β) sebagai linier span dari *B-spline.*

B_k^p, $p \geq 0$, $k \in I^p$

PARAMETRISASI

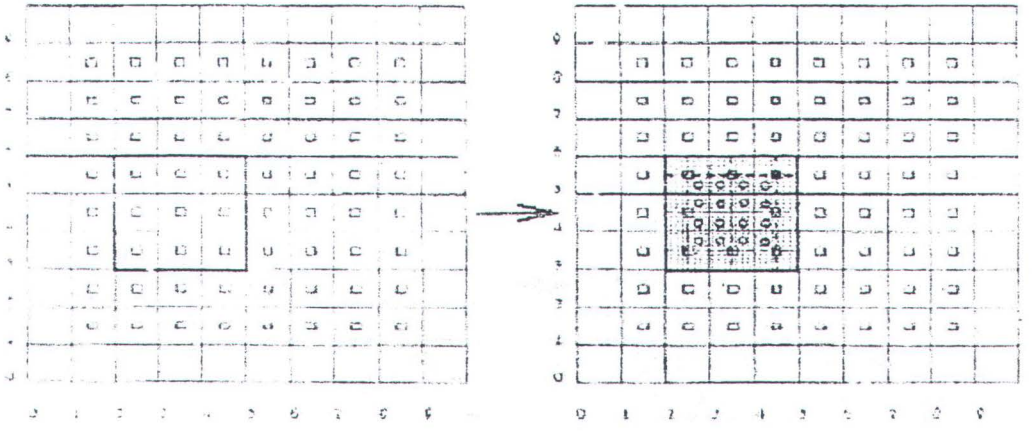
Fungsi *spline* S keluar dari ruang multilevel *B-spline* S_Ω diwakili oleh $S : R^2 \to R.$

$$x \to S(x) = \sum_{p \geq u} \sum_{k \in I^p} b_k^p B_k^p(x) \quad \text{dengan } b_k^p \in R \quad \text{..............................(2.25)}$$

---

[17] Ibid, halaman 4

2.6.3 Domain Ω^1 terkecil[18]



Gambar 2.16 Penggambaran domain terkecil

Batasan domain adalah sebagai berikut :

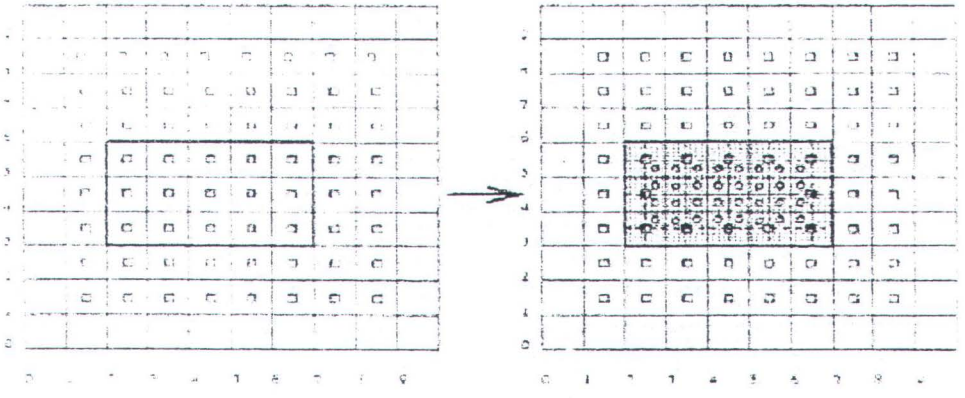
- $\Omega^1 = (2,5) \times (3,6)$
- $B_{2,3} \notin S_\Omega$

Domain yang terbentuk adalah sebagai berikut :

- Terdapat 16 *B-spline* baru pada level 1.
- Empat segmen di tengah Ω^1 menyediakan flesibilitas penuh dari h/2-mesh.
- Sehingga, karena semua level 1 *B-spline* yang mendukung sudah penuh, maka segmen-segmen ini menjadi anggota dari I^1.

[18] Ibid, halaman 5

Ω^1 yang dibatasi sebagai gabungan support tiga *B-Spline*[19].



Gambar 2.17 Penggabungan domain dengan support 3 *B-Spline*.

Batasan domain adalah sebagai berikut :

- $\Omega^1 = (2,7) \times (3,6)$
- $B_{2,3} \notin S_n$, $B_{3,3} \notin S_n$, $B_{4,3} \notin S_n$

Dari batasan domain tersebut diatas akan menghasilkan 32 *B-spline* baru pada level 1.

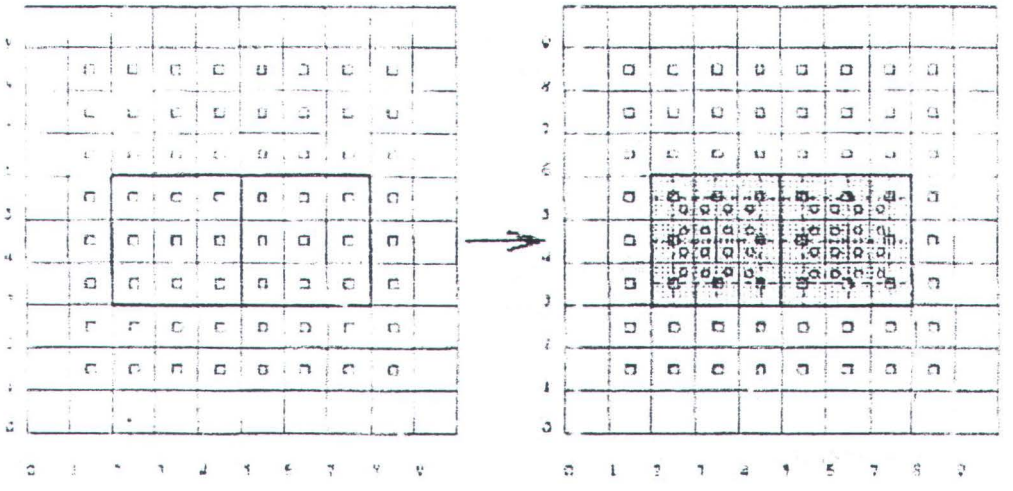
Definisi kasus garis batas dari Ω^1 dengan dua disjoint domain[20].

Batasan domain adalah sebagai berikut :

- $\Omega^1 = (2, 5) \times (3, 6) \cup (5, 8) \times (3, 6)$.
- Garis antara koordinat (5, 3) dan (5, 6) tidak berada dalam Ω^1.
- $B_{2,3} \notin S_n$,dan $B_{6,3} \notin S_n$,.

Dari batasan domain tersebut diatas akan menghasilakn 32 *B-Spline* baru pada level 1.

[19] Ibid, halaman 6
[20] Ibid, halaman 7



Gambar 2.18 Domain dengan 2 disjoin

2.6.4 Properti Multilevel *B-Spline*[21]

Theorema 1 :

Fungsi dasar support adalah terhingga.

Theorema 2 :

Jika ruang multilevel *B-spline* S_n mempunyai derajad (α, α), maka permukaan *spline* $S \in S_n$ adalah $C^{\alpha-1}$.

Theorema 3 :

$$\sum_{p\geq 0} \sum_{k\in I^p} B_k^p(x) \neq 1 \quad \text{.....(2.26)}$$

Theorema 4 :

B-spline yang dinyatakan dalam $\left\{B_k^p \mid p \geq 0, k \in I^p\right\}$ .....(2.27) adalah linear bebas.

Dalam teori kebebasan linier multilevel *B-spline*, kita hanya tahu estimasi yang lebih lemah.

---

[21] Ibid, halaman 8

Theorema 5 :

Jika S_n, Ω^p dan I^p dengan p≥0, diketahui,. Maka

$$C_0^p \max_{k \in I^p} | b_k^p | \leq \| \sum_{k \in I^p} b_k^p B_k^p \| \infty \leq \max_{k \in I^p} | b_k^p |, \qquad \forall p \geq 0 \ldots\ldots\ldots\ldots\ldots (2.28)$$

$$\text{dan } C_0 \max_{\substack{p \geq 0 \\ k \in I^p}} | b_k^p | \leq \| \sum_{p=p\min}^{p\max} \sum_{k \in I^p} b_k^p B_k^p \| \infty \leq C_1 \max_{\substack{p \geq 0 \\ k \in I^p}} | b_k^p | \ldots\ldots\ldots\ldots\ldots\ldots (2.29)$$

Konstanta C_0 dan C_1 hanya tergantung pada α, β, P_{min} dan P_{max}. Dimana, konstanta C_1 adalah $C_1 = (1 + P_{max} - P_{min})$.

2.6.5 Approksimasi dengan *Hierarchical B-Splines*[22]

Theorema 6 :

Jika Q adalah quasi interpolasi yang sesuai untuk ruang multilevel B-spline. Untuk x ∈ Ω^0, jika *m* adalah integral dengan x ∈ $\omega^m \setminus \omega^{m+1}$. Maka untuk setiap fungsi $\int \in C^{(\alpha^1 + I_1 \beta^1 + 1)}$ dengan $(\alpha^1, \beta^1) \leq (\alpha, \beta)$

$$\left| \int (x_1 x_2) - (Q \int)(x_1, x_2) \right| \leq const \left[\left(\frac{\left(h_{x_1}^0 (x_1) \right)}{2^m} \right)^{\alpha^1 + 1} \left\| f^{(\alpha^1 + 1, 0)} \right\| \infty + \left(\frac{\left(h_{x_2}^0 (x_2) \right)}{2^m} \right)^{\beta^1 + 1} \left\| f^{(0, \beta^1 + 1)} \right\| \infty \right] \ldots\ldots\ldots (2.30)$$

dimana konstanata *const* tidak tergantung pada *t*, deret-deret knot $h_{x_1}^0$ dan $h_{x_2}^0$.

Keterangan :

- $\omega^p := \{x \in \Omega \forall \omega^p := \{x \in \Omega^p \mid \forall_{k \in Z^2} x \in \text{supp } B_k^p \Rightarrow B_k^p \in S_n\}$
- Pada *x* order penyelesaian yang sama seperti dengan TP *B-spline* dibatasi dalam grid pada level *m*.
- Kesalahan penyelesaian lokal dapat dikurangi dengan pembersian lokal.

---

[22] Ibid, halaman 10

2.6.5.1 Algoritma Approksimasi Umum[23]

- Hitung ruang multilevel *B-spline* dengan 1 level dan lebar grid $S(x) = \sum_k b_k^0 B_k^0(x) \equiv 0$.
- Lakukan :
  - ➢ Hitung koefisien *B-spline* untuk menyelesaikan data yang diberikan
  - ➢ Untuk semua *B-spline* B_k^p lakukan :
    - ❑ Jika error (data, *S(x)*, B_k^p) > c maka
      - ❖ Hapus B_k^p.
      - ❖ Tambahkan *B-spline* yang sesuai pada level p+1.
      - ❖ Ujilah apakah ruang multilevel *B-spline* memenuhi kondisi-kondisi pada Ω^p dan index-set I^p dan jika diperlukan, tambahkan *B-spline* yang lain pada level *p*.
    - ❑ endif.
  - ➢ endfor.
- while (max error > c).

2.6.5.2 Approksimasi Data Acak[24]

Langkah-langkah penyelaesaian approksimasi untuk data acak adalah sebagai berikut :

- Bangkitkan koordinat data acak (x_1,z_1), $1 \le i \le n$.
- Selesaikan sistem linier dari persamaan $A.b = z$ dengan :

---

[23] Ibid, halaman 11
[24] Ibid, halaman 12

$$n \times m_{matrix} A = (a_{ij})_{\substack{1 \le i \le n \\ 1 \le j \le m}} \quad \text{.....(2.31)}$$

$$a_i = \left(B^0_{k_1}(x_1), ..., B^0_{k_p}(x_1), B^1_{k_{11}}(x_1), ...\right) \quad \text{.....(2.32)}$$

dan vector Z = (z_1, ..., z_n).....(2.33)

dimana :

◊ Matrix A adalah sparse matrik (*Bounded support*).

◊ Matrix A merupakan *full rank* matrik (bebas linier)

- Selesaikan $|| Ab - z ||_2 \rightarrow$ min.
- Jika penyelesaiannya tidak unique, minimalkan *thin pate energy* secara *additional*.
- Gunakan penyelesaian iterasi. Misalnya dengan MinRes.
- Tentukan *start-vector* untuk *refined-mesh* dengan *subdivision*.

Algoritma untuk approksimasi data acak adalah sebagai berikut [25]:

- Tentukan Ω^0 untuk semua titik data acak $x_1 \subset \Omega^0$, $1 \le i \le n$.
- Bangkitkan ruang multilevel B-spline dengan satu level dan lebar *grid-width*

  $S(x) = \sum_k b^0_k B^0_k(x) \equiv 0$

- Hitung $I = I_0 \delta_\varepsilon = \frac{\varepsilon_0 \varepsilon}{l_0}$
- Lakukan :
  - Bangkitkan matrix $A = \left(a_{ij}\right)_{i,j,a_{ij}} = B^0_k(x_1)$ dan vektor $z = (z_i)_i$ dengan $1 \le i \le n$, $p \ge 0$, $k \in l^p$.
  - Hitung $\hat{A} = A^t.A.\hat{z} = A^t z$.

[25] Ibid, halaman 13

- Selesaikan persamaan linier Âb = $\hat{Z}$ dengan penyelesai iteratif MinRes. Gunakan sebagai *start-vector*$(b_k^p)_p \geq 0, k \in I^p$
- Untuk semua *B-Spline* B_k^p lakukan :
  - jika error (z_1, S(x_i), B_k^p) > ϵ+l$\not\in\delta_\epsilon$ maka
    - Bagilah B_k^p.
  - endif.
- endfor.
- Hitung *l* = *max* {*0, l-1*}.

- while (max error >c).

BAB III

KONSEP DASAR OPENGL

BAB III

KONSEP DASAR OPENGL

OpenGL adalah suatu library untuk pemodelan dan grafik tiga dimensi yang mempunyai kelebihan dalam kecepatan dan dapat digunakan dimanapun. OpenGL bukanlah bahasa pemrogamaman tetapi merupakan API *(Application Programming Interface)*. Apabila suatu program disebut *OpenGL-based* atau OpenGL *application* hal ini berarti bahwa program tersebut ditulis dalam suatu bahasa pemrograman (seperti C atau pascal) yang memanggil satu atau lebih fungsi dari *library* OpenGL.

Dengan menggunakan OpenGL pembuatan suatu model harus dimulai dari *low level* yaitu berasal dari sekumpulan fungsi geometri primitif untuk seperti fungsi untuk menggambarkan titik, garis dan poligon. Pada umumnya urutan langkah dalam menggunakan fungsi OpenGl adalah sebagai berikut :

1. Konstruksi obyek dari bentuk geometri yang primitif dengan menggunakan rumus matematika yang sesuai.
2. Mengatur posisi obyek diruang tiga dimensi dan memilih sudut pandang yang diinginkan untuk menampilkan obyek di layar.
3. Menghitung warna dan cahaya untuk obyek termasuk texturing.
4. Mengkonversikan deskripsi matematika dari obyek dan warna ke pixel pada layar.

Dalam bab ini dijelaskan mengenai konsep dasar dan fungsi-fungsi OpenGl diambil dari buku *"OpenGL Super Bible"* oleh Richard S. Wright Jr., Michael Sweet dan buku *"OpenGL Programming Guide"* oleh Jackie Neider, Tom Davis dan Mason Woo.

3.1 Sintaks Perintah OpenGL

Semua peritah OpenGL mengikuti aturan penulisan yang memberi keterangan mengenai dari *library* mana fungsi tersebut berasal, berapa banyak argumen yang diperlukan tipe datanya, serta keterangan mengenai apa yang dilakukannya. Format perintah OpenGL adalah :

<awalan library><perintah><optional jumlah argumen><optional type argumen>

Berikut adalah penjelasan mengenai perintah dasar OpenGL. Semua perintah dasar OpenGL menggunakan awalan gl diikuti dengan huruf kapital pada setiap kata membentuk nama perintah (seperti contoh `glClearColor`). Untuk mendefinisikan konstanta diawali dengan GL_, dengan menggunakan huruf kapital dan garis bawah untuk memisahkan kata (seperti `GL_COLOR-_BUFFER_BIT`). Terkadang beberapa huruf dan angka ditambahkan pada akhir perintah (seperti *3f* pada `glColor3f`). Dalam hal ini angka 3 menunjukkan berapa argument yang harus ada pada perintah tersebut dan akhiran huruf f menunjukkan jenis datanya yaitu floating.

Beberapa perintah OpenGL menerima sebanyak 8 macam tipe data yang berbeda, dengan membedakan akhiran pada perintahnya. Huruf yang digunakan sebagai akhiran yang menspesifikasikan jenis datanya dapat dilihat pada tabel 3.1.

Tabel 3.1. Tipe data pada OpenGL

Akhiran	Tipe yg bersesuaian Dalam C	Definisi Tipe Dalam OpenGL
B 8-bit Integer	Signed char	`Glbyte`
s 16-bit Integer	Short	`Glshort`
i 32-bit Integer	Long	`Glint, GLsizei`
f 32-bit Floating-point	Float	`GLfloat, GLclampf`
d 64-bit Floating point	Double	`GLdouble, GLclampd`
Ub 8-bit unsigned Integer	Unsigned chaar	`Glubyte,GLboolean`
Us 16-bit unsigned Integer	Unsigned	`Glushort`
Ui 32-bit unsigned Integer	Unsigned	`Gluint,GLenum, GLbitfield`

Sebagai contoh pada dua perintah berikut ini :

```
glVertex2i( 1,3 );
glVertex2f( 1.0,3.0 )
```

adalah sama yaitu meletakkan titik di layar pada koordinat x=1 dan y=2, perbedaannya perintah yang pertama menspesifikasikan titik dengan tipe data integer 32-bit dan yang kedua adalah dengan tipe data *single precision floating point.*

Beberapa perintah OpenGL menambahkan huruf akhir v, yg menunjukkan bahwa perintah tersebut menggunakan pointer ke array/vektor. Di bawah ini contoh perbedaannya.

```
float color_array[]={1.0 , 0.0 ,0.0}
glColor3f (1.0 , 0.0 ,0.0);
```

```
glColor3fv(color_array);
```

3.2 Library yang Berhubungan dengan OpenGL

OpenGL menyediakan sejumlah set perintah untuk menggambar dan semua penggambaran yang lebih tinggi tingkatnya harus dilakukan dengan mengambil fungsi dasar dari perintah ini. Maka dari itu dapat dibuat *library* sendiri di atas program OpenGL yang mempermudah pemrograman lebih lanjut. Fungsi asli dari OpenGL sendiri selalu dimulai dari awalan *gl* terdapat pada *library* opengl32.dll dan file gl.h. Sedangkan beberapa *library* yang telah ditulis untuk menyediakan fungsi – fungsi tambahan pada OpenGL adalah:

- *OpenGL Utility Library (GLU)* yang di dalamnya terdapat sejumlah rutin yang menggunakan level bawah dari perintah OpenGL. Contohnya perintah untuk menginisialisasi matrik yang digunakan untuk menspesifikasikan orientasi penglihatan dan proyeksi atau juga perintah untuk melakukan perintah render pada permukaan. Rutin – rutin ini mempunyai awalah *glu*. *library* ini disediakan sebagai bagian dari implementasi OpenGL.
- *OpenGL Extension* untuk X Window yang menyediakan fungsi untuk menciptakan *OpenGL context* dan mengasosiasikannya dengan mesin yang menggunakan sistem *X window*. Rutin – rutin mempunyai awalan *glx*.
- *Auxilary* atau *Aux library* terdapat pada *library* glaux.lib dan *file* galux.h. Perintah atau fungsi yang akan digunakan selalu menggunakan awalan aux. Auxilary ini diciptakan sebagai sarana untuk mempelajari dan membuat program OpenGL tanpa harus memperhatikan envronment yang sedang dipakai. Fungsi ini yang membuat, mengatur, memanipulasi windows dan masukan dari pemakai. Akan tetapi tidak semua kode yang akan menyusun

aplikasi bisa diandalkanpada fungsi aux ini, karena fungsi ini hanya berfungsi sebagai sarana mempermudah dalam mempelajari OpenGL dengan contoh.

3.3 Menggambar Obyek Geometri

Pada OpenGL ada dua dasar operasi gambar yaitu membersihkan *windows* dan menggambar obyek geometri termasuk titik, garis dan poligon.

3.3.1 Membersihkan Window

Menggambar pada layar komputer berbeda dengan menggambar pada kertas putih yang dari pabriknya sudah berwarna putih. Pada komputer memori untuk menampilkan gambar biasanya diisi dengan gambar yang berasal dari perintah gambar paling akhir jadi perlu dibersihkan dengan warna latar belakang sebelum digambar lagi. Warna latar belakang yang dipilih tergangtung aplikasi misalnya jika ingin membuat animasi rotasi bumi, bulan dan matahari maka warna latar belakang yang dipilih pasti hitam seperti gelapnya angkasa luar. Sintaks perintah `glClearColor(Glclampf red, Glclampf green, Glclampf blue, Glclampf alpha)` digunakan untuk memilih warna, yang akan digunakan untuk membersihkan latar belakang dalam *mode* RGBA. Dan selanjutnya digunakan perintah `glClear (Glbitfield mask)` untuk membersihkan buffer yang dispesifikasikan dengan warna yang telah ditentukan. Contoh berikut ini menunjukkan perintah yang digunakan untuk membersihkan latar belakang dengan warna hitam dan buffer apa yang akan dibersihkan.Dalam hal ini buffer warna yang akan dibersihkan karena buffer warna merupakan tempat gambar disimpan.

```
glClearColor(0.0 , 0.0 , 0.0);
```

```
glClear(GL_COLOR_BUFFER_BIT);
```

3.3.2 Spesifikasi Warna

Pada OpenGL mendeskripsikan obyek dan warna obyek adalah proses yang berjalan sendiri–sendiri. Karena pada umumnya seorang *programmer* pada umumnya akan mengatur warna (*coloring scheme*) terlebih dahulu baru menggambar obyek. Sebelum warna diubah maka semua obyek yang digambar sesudah perintaah tersebut akan menggunakan warna terakhir yang terdaftar pada *coloring scheme.*

Untuk warna digunakan perintah `glColor3f()` jika lebih dari tiga maka argumen keempat adalah alpha yang akan dijelaskan pada bagian *blending* sebagai salah satu efek yang dipunyai OpenGL. Contoh berikut menunjukkan urutan langkah dalam proses spesifikasi warna sebelum obyek digambar.

```
glColor3f(0.0 , 0.0 , 1.0); //setting warna
draw_Object(A); //gambar obyek A
```

3.3.3 Memaksa Proses Menggambar Sampai Selesai

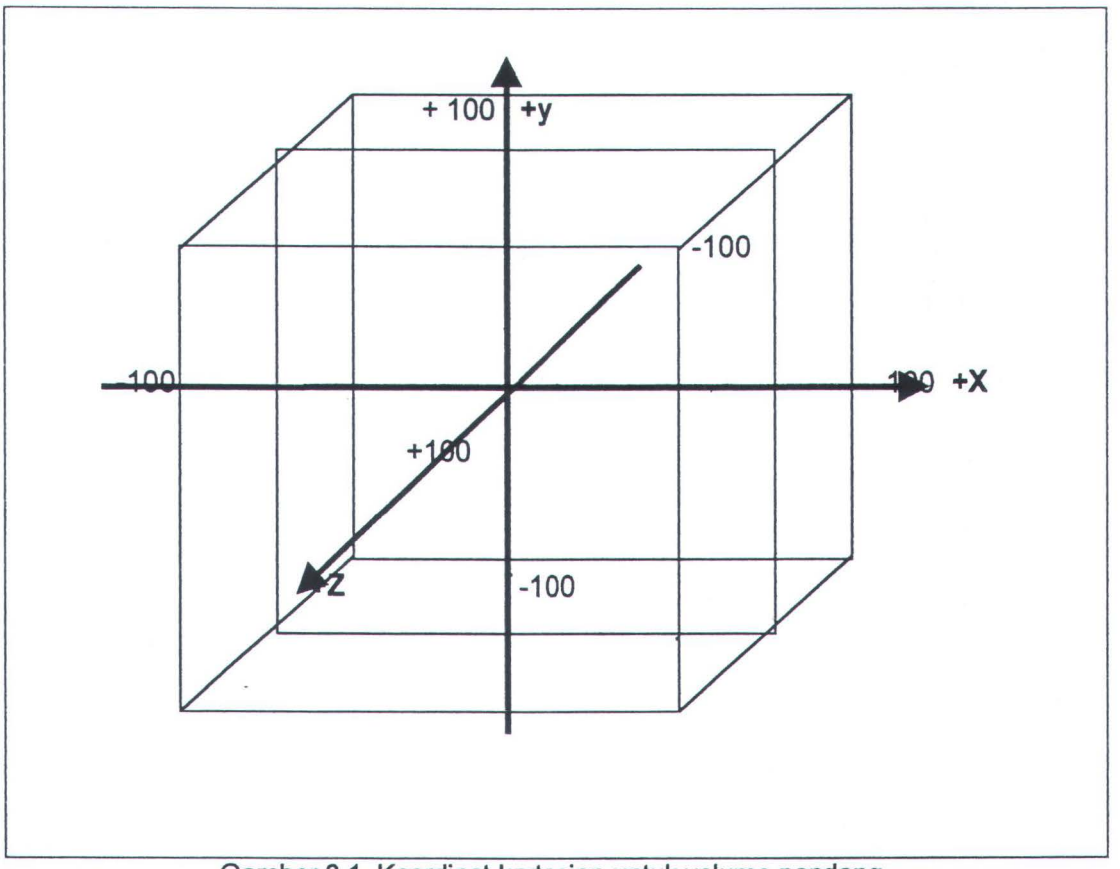
Kebanyakan sistem grfik yang sudah modernsudah menggunakan sistem *graphics pipeline.* Dimana CPU utama memberikan *issue* perintah menggambar dan hardware lain yang melakukan transformasi, *clipping, shading atau texturing.* Pada arsitekstur yang demikian proses tidak dilakukan pada satukomputer karena setiap komputer mempunyai tugas sendiri. CPU utama tidak harus menunggu proses pada masing – masing komputer tadi selesai tapi bisa dengan segera memberikan *issue* perintah gambar yang berikutnya. Untuk inilah OpenGL menyediakan perintah `glFlush()` yang memaksa *client* untuk segera

mengirim paket network walaupun belum penuh. Program sebaiknya ditulis dengan memakai perintah ini karena `glFlush()` tidak memaksa proses gambar untuk selesai tetapi mamaksa proses gambar untuk segera dieksekusi, sehingga dijamin semua perintah gambar yang sebelumnya akan segera dieksekusi dalam suatu waktu tertentu.

3.3.4 Menggambar di Bidang Tiga Dimensi

Untuk menggambar jenis grafik apapun pada komputer biasanya dimulai dengan *pixel*. *Pixel* adalah elemen terkecil dari layar monitor yang mempunyai atribut warna dan posisi. Sedangkan untuk membentuk garis, poligon, lingkaran dan lain – lain dapat dilakukan melalui urutan – urutan *pixel* yang berbeda. Menggambar dengan menggunakan OpenGL mempunyai perbedaan dengan bahasa lai, yaitu: tidak perlu memikirkan koordinat layar secara fisik tetapi hanya perlu menspesifikasikan posisi koordinat dengan volume penglihatan. OpenGL memikirkan sendiri cara bagaimana menggambar titik, garis dan lainnya yang berada dalam ruang 3D ke gambar 2D pada layar komputer.

Pada gambar 3.1 dibawah ini memperlihatkan suatu ruang pandang yang sederhana yang akan digunakan pada bagian ini. Area gambar yang dibatasi ini adalah ruang koordinat kartesian yang mempunyai *range* dari –100 hingga 100 untuk sumbu x,y dan z. Secara sederhana bidang ini dapat dianggap sebagaibidang gambar untuk perintah – perintah OpenGL.



Gambar 3.1. Koordinat kartesian untuk volume pandang

100x100x100

Untuk menggambar titik digunakan suatu perintah OpenGL yaitu: `glVertex`. Fungsi dapat mempunyai 2 sampai 4 parameter dari berbagai macam tipe data. Sebagai contoh perintah `glVertex` di bawah ini akan menspesifikasikan sebuah titik pada posisi 5 sumbu x, 5 sumbu y, dan 0 untuk sumbu z.

```
glVertex3f (5.0f , 5.0f , 0.0f).
```

Sekarang setelah diketahui cara untuk menspesifikasikan sebuah titik di ruang pada OpenGL. Selanjutnya yang harus ditambahkan adalah informasi tambahan mengenai titik tersebut, apakah titik tersebut merupakan akhir dari sebuah garis, atau merupakan sebuah titik sudut dari sebuahpoligon atau lainnya, karena

definisi geometrikdari sebuah vertex sebenarnya bukanlah hanya sebuah titik pada layar tetapi lebih merupakan suatu titik dimana terjadi interseksi antara dua buah garis atau kurva.

Primitif adalah interpretasi sejumlah set atau deretan titik pada sebuah bentuk yang digambar pada layar. Pada OpenGL terdapat spuluh macam primitif dari mulai menggambar sebuah titik hingga poligon. Untuk itu digunakan perintah glBegin sebagai cara memberitahu OpenGL agar memulai mengiterpretasikan sederetan titik sebagai salah satu bentuk primitif. Dan untuk menghakhiri deretan titik ini digunaka perintah glEnd. Sebagai contoh adalah dua perintah berikut ini :

peintah 1:

```
glBegin(GL_PIONTS); //menspesifikasikan titik sebagai primitif
    glVertex3f(0.0f,0.0f,0.0f);//menspesifikasikan posisisi suatu
                                titik
    glVertex3f(5.0f,5.0f,5.0f);//menspesifikasikan posisi szuatu
                                titik yang lain
glEnd();                    //Mengakhiri perintah menggambar titik
```

perintah 2 :

```
glBegin(GL_LINES); //menspesifikasikan garis sebagai primitif
    glVertex3f(0.0f,0.0f,0.0f);//menspesifikasikan posisi titik awal
                                garis
    glVertex3f(15.0f,15.0f,15.0f);//menspesifikasikan posisi titik
                                akhir
glEnd();                    //Mengakhiri perintah menggambar titik
```

Perintah satu hasilnya berupa dua buah titik di layar pada posisi (0.0f, 0.0f, 0.0f) dan (5.0f, 5.0f, 5.0f) sedangkan pada perintah kedua akan menghasilkan garis yang melalui titik (0.0f, 0.0f, 0.0f), (5.0f, 5.0f, 5.0f) dan (15.0f,15.0f, 15.0f). perlu diketahui bahwa jumlah perintah `glVertex` yang berada diantara `glBegin` dan `glEnd` tidak dibatasi.Tabel 3.2 dibawah ini adalah mengenai jenis primitif dan kegunaannya yang digunakan sebagai parameter dalam glBegin.

Tabel 3.2 Primitif dari OpenGL yang digunakan sebagai argumen dalam glBegin

Mode	Tipe Primitif
GL_POINT	Titik yang dispesifikasikan adalah titik – titik tunggal
GL_LINES	Titik yang dispesifikasikan digunakan untuk menghasilkan garis. Setiap dua titik merupakan satu garis yang berdiri sendiri. Bila jumlahnya ganjil maka titik yang terakhir akan diabaikan
GL_LINE_STRIP	Titik yang dispesifikasikan digunakan untuk menghasilkan garis yang bersambung. Artinya setelah titik yang pertama maka titik selanjutnya merupakan tujuan perpanjangan garis (sama seperti perintah lineto pada bahasa C)
GL_LINE_LOOP	Sama seperti GL_LINE_STRIP, hanya terdapat tambahan garis yang menghubung-kan antara titik petama dan yang terakhir. Ini digunakan untuk menggambar daerah tertutup yang tidak mengikuti aturan GL_POLYGON
GL_TRIANGLES	Titik yang dispesifikasikan digunakan untuk menghasilkan segitiga. Setiap tiga titik merupakan satu segitiga yang berdiri sendiri.

GL_TRIANGLES_STRIP	Titik yang dispesifikasikan digunakan untuk sederetan segitiga. Setelah tiga titik pertama terbentuk setiap titik pada segitiga tersebut akan digunakan dengan dua titik berikutnya untuk membentuk segitiga yang lain
GL_TRIANGLES_FAN	Titik yang dispesifikasikan digunakan untuk menghasilkan segitiga yang berputar. Titik pertama berfungsi sebagai titk pusat dan setiap titik setelah titik yang ketiga akan digabung dengan yang berikutnya dan titik pusat
GL_QUADS	Setiap empat titik digunakan untuk membentuk segiempat. Jika jumlah titiknya bukan merupakan kelipatan empat makan sisanya diabaikan.
GL_QUAD_STRIP	Titik yang dispesifikasikan digunakan untuk membentuk segiempat dengan aturan sebuah segiempat didefinisikan untuk setiap pasang titik setelah sepasang titik yang pertama.
GL_POLYGON	Titik yang dispesifikasikan digunakan untuk membentuk suatu poligon yang konveks. Sudut dari poligon tidak boleh berinterseksi. Titik yang terakhir secara otomatis akan dihubungkan dengan titik yang pertama

Ketika menggambar titk tunggal, secara default size titik adalah satu. Untuk mengubah size ini digunakan dengan fungsi glPointsize() dengan parameter ukurannya. Selain mendefinisikan ukuran dari titik dapat jugadilakukan pengubahab letak dari garis ketika melakukan penggambaran. Hal ini dapat

dilakukan dengan perintah glLineWidth() yang mengambil parameter besarnya lebar garis.

3.3.5 Metode Hidden_Surface Removal

Jika ada dua obyek digambar, gambar A kemudian gambar B maka pada suatu sudut pandang tertentu akan tampak obyek B menutupi obyek A tapi jika melihat dari sudut pandang yang berlawanan maka obyek A harus berada di depan obyek B. Relasi saling menutupi ini harus tetap dipertahankan dalam menggambar layar yang realistik. Oleh karena itu digunakan metode *hidden_surface removal* yaitu penglihatan bagian dari obyek solidyang terhalang oleh obyek lain. Dalam OpenGL dikenal *buffer* yang bertugas menangani masalah ini yaitu *depth buffer* yang bekerja dengan cara menyimpan kedalaman dan jarak dari sudut pandang tiap pixel pada window.

Perhitungan grafika dalam hardware ataupun software mengkoversikan semua permukaan obyek yang digambar menjadi sekelompok pixel pada window dimana permukaan akan tampak jika tidak terhalangi oleh sesuatu. Disini juga dilakukan perhitungan jarak dari mata ke obyek. Dengan menggunakan *depth buffer*, sebelum setiap pixel digambar maka sebuah perbandingan dilakukan dengan nilai yang sebelumnya pernah dimasukkan ke dalam *depth buffer*. Jika pixel baru lebih dekat ke mata darip[ada nilai pixel yang sebelumnya di dalam *depth buffer* maka nilai kedalaman dan warna pixel yang baru akan menggantikan pixel sebelumnya dan begitu juga sebaliknya. Sehingga metode *hidden_serface removal* ini akan semakin meningkatkan kinerja penggambaran karena informasi yang terbaru sudah dibandingkan dengan informasi lama dan hanya menggambar salah satu saja. Untuk menggunakan depth buffer maka

tinggal mengaktifkan perintah OpenGl yaitu `glEnable(GL_DEPTH-TEST)` dan ini hanya dilakukan sekali.

3.4 Teknik Viewing pada OpenGL

Garis besar dalam bidang ilmu grafika komputer adalah bagaimana menghasilkan gambar dua dimensi dari obyek tiga dimensi karena obyek tersebut digambar di layaar komputer yang merupakan bidang dua dimensi. Beberapa operasi pada komputer yang mengkonversikan kordinat obyek tiga dimensi ke posisi pixel pada layar komputer :

- Transformasi, yang direpresentasikan dengan paerkalian matriks, termasuk pemodelan obyek (modeling), penglihatan (viewing), dan operasi proyeksi (projection). Termasuk juga di dalamnya rotasi, translasi, skala,reflesi, proyeksi ortographic dan perspective.
- Karena layar adalah windows segiempat, maka obyek yang terletak diluar windows harus dibuang (clipped) dari clipping plane.
- Langkah akhir menyelesaikan mapping dari koordinat yang ditransformasikan menjadi pixel di layar yang dinamakan transfoemasi *viewport*.

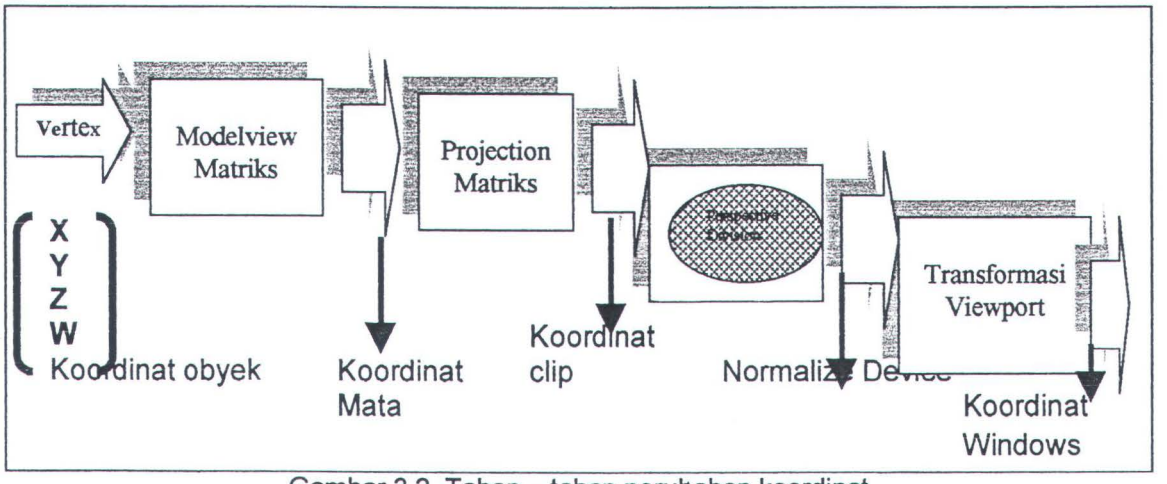
Proses transformasi untuk menghasilkan ruang pandang yang diinginkan hampir sama dengan cara orang untuk mengambil foto dengan menggunakan kamera. Langkah – langkahnya adalan sebagai berikut :

1. Menyiapkan tripot sebagai penyanggah kamera dan mengarahkan kamera menuju sebuah ruang pandang sesuai dengan yang diinginkan (pada OpenGL diistilahkan *Viewing Transformation)*
2. Mengatur ruang pandang (obyek) yang akan difoto dilihat dari kamera sesuai dengan keinginan (*Modeling Transformation)*

3. Memilih lensa kamera atau ingin memperbesar atau sebaliknya (*Projection Transformation)*
4. Mengatur seberapa besar ukuran foto yang nanti akan dihasilkan (*Viewport Transformation)*

Jika semua tahap di atas telah dilaksanakan dengan benar maka gambar foto akan dihasilkan dengan baik atau dalam istilah OpenGL dianggap ruang pandang sudah siap dipakai atau harus dipersiapkan terlebih dahulu sebelum obyek digambarkan.

Untuk menghasilkan koordinat yang dipakai di windows maka harus dispesifikasikan sebuah matriks 4x4 yang kemudian dikalikan dengan koordinat dari tiap vertex yang mendefinisikan obyek, baik itu untuk transformasi *viewing, modeling,* ataupun *projection.* Gambar 3.2 menunjukkan urutan proses transformasi untuk menghasilkan koordinat windows.



Gambar 3.2. Tahap – tahap perubahan koordinat

Vertex dalam OpenGL selalu terdiri dsari 4 koordinat x,y,z,w walaupun pada kebanyakan kasus w selalu 1 sedangkan pada dua dimensi w =0.

Transformasi viewing dan modeling telah dispesifikasikan akan dikombinasikan sehingga membentuk matriks *Modelview*, yang akan dikalikan dengan koordinat obyek untuk menghasulkan koordinat mata (*eye coordinates*).

Selanjutnya OpenGL menggunakan matriks *projection* untuk menghasilkan koordinat *clip*. Pada tahap ini didefinisikan volume ruang pandang atau dengan kata lain obyek di luar volume tersebut tidak akan ditampilkan pada layar. Setelah itu baru dilanjutkan dengan *perspective division* yaitu membagi koordinat dengan w untuk menghasilkan *Normalize Device Coordinate*. Dan pada akhirnya koordinat hasil transformasi tersebut diubah menjadi koordinat pada windows dengan menggunakan transformasi *viewport*. Koordinat akhir inilah yang dibutuhkan guna memanipulasi obyek pada layar.

Berbagai macam transformasi yang sudah dijelaskan di atas masing – masing memiliki matriks sendir yaitu untuk transformasi *viewing* dan *modeling* di gunaka matriks *modelview*, sedangkan untuk transformasi proyesi digunakan matrik proyeksi. Untuk menspesifikasikan matriks modelview, projection atau texture yang akan dimodifikasi digunakan perintah glMatrixMode(Glenum mode) dengan argumen untuk mode yaitu : `GL_MODELVIEW`, `GL_PROJECTION` atau `GL_TEXTURE`. Perintah transformasi selanjutnya akan mempengaruhi matriks yang telah dispesifikasikan oleh `glMatrixMode()`. Sedangkan untuk membersihkan masing – masing matriks tersebut diatas bisa digunakan `glLoadIdentity()`. Contoh berikut menunjukkan bahwa sebelum melakukan transformasi viewing atau modeling maka stack matrik yang terakhir terlebih dahulu harus dibersihkan dengan matrik identitas.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

Langkah ini sangat perlu karena setiap perintah transformasi selalu mengalikan stack matrks yang paling akhir dengan matrik baru dispesifikasikan dan hasilnya disimpan di stack matrik yang paling akhir tersebut. Jika tidak

mengalikan stack matrik yang terakhir tadi dengan matrik identitas maka dianggap transformasi dilanjutkan dari matrk transformasi yang terakhir digunakan.

3.4.1 Transformasi Modeling

Transformasi modeling adalah transformasi untuk menentukan posisi dan orientasi dari sebuah model. Dalam hal ini kita dapat melakukan rotasi, transisi, penskalaan atau gabungan dari ketiganya.

Tiga rutin OpenGL untuk transformasi modeling adalah `glTranslatef()`, `glRotatef()` dan `glscalef()`. Rutin – rutin tersebut mentransformasikan sebuah obyek dengan jalan menggeser, memutar, membesarkan atau mengecilkan obyek tersebut. Perintah – perintah tersebut sama dengan jika kita membuat sebuah matrik translasi, rotasi maupun *matrik scaling* dan kemudian memanggil perintah `glMutMatrix()` dengan matrik tersebut sebagai argumen.

Sintaks perintah `glTranslate{f,d}(TYPEx,TYPEy,TYPEz)` digunakan untuk mengalikan matrik yang sedang aktif dengan sebuah matrik yang mentranslasikan obyek berdasarkan nilai argumen x,y,z yang diberikan. Sedangkan sintaks perintah `glRotate{f,d}(TYPEx, TYPEy,TYPEz)` digunakan untuk mengalikan matrik yang sedang aktif dengan matrik yang memutar obyek dengan arah yang berlawanan dengan arah jarum jam, sebesar sudut yang diberikan argumen angle dan berdasarkan sumbu yang diberikan argumen x, y atau z. Dan perintah yang terakhir yaitu `glScale{f,d}(TYPEx, TYPEy,TYPEz)` digunakan untuk mengalikan matrik yang sedang aktif dengan matrik yang memperbesar, memperkecil atau merefleksikan obyek . Masing –

masing koordinat x, y, z dari setiap titik pada obyek dikalikan dengan argumen x, y, z. Perintah glScale merupakan satu – satunya perintah transformasi modeleng yang merubah ukuran obyek. Jika nilai yang dimasukkan lebih dari 1.0 maka obyek akan diperbesar, jika nilai yang dimasukkan kurang dari 1.0 maka obyek akan diperkecil dan jika nilai yang dimasukkan negatif maka obyek akan direfleksikan (dicerminkan).

3.4.2 Transformasi Viewing

Memanggil transformasi viewing dapat dianalogikan dengan mendefinisikan dan meletakkan kamera pada posisinya. Sebelum mendefinisikan transformasi viewing perlu untuk menggeser matrik yang sedang aktif dengan perintah `glLoadIdentity()`.

Penyelesaian transformasi viewing dapat dilakukan dengan beberapa jalan, seperti yang dijelaskan di bawah ini. Kita dapat juga memilih untuk menggunakan letak dan orientasidefault dari titik pandang, yang terletak di pusat koordinat dan memandang ke arah sumbu z negatif. Adapun cara yang digunakan untuk menyelesaikan transformasi viewing adalah :

- Menggunakan salah satu perintah dari transformasi modeling yaitu `glRotate*()` atau `glTranslate*()`.
- Menggunakan rutin *Utility Library* `gluLookAt()` untuk mendefinisikan garis pandang. Rutin ini mengandung perintah untuk rotasi dan translasi.
- Membuat rutin sendiri yang mengandung perintah rotasi dan translasi. Beberapa aplikasi mungkin membutuhkan rutin sendiri yang memungkinkan aplikasi tersebut menentukan viewing yang sesuai.

Seringkali seorang programmer mengatur layar pada sekitar pusat sistem koordinat atau di lokasi lain yang sesuai, kemudian ingin melihatnya dari posisi pandang yang berubah – ubah untuk mendapatkan hasil yang sesuai. Rutin `gluLookAt()` pada OpenGL dirancang untuk tujuan tersebut. Rutin tersebut memerlukan 3 set argumen untuk menentukan letak titik pandang, mendefinisikan titik referensi terhadap arah dari kamera dan menunjukkan dimana arah atas. Perintah yang digunakan adalah :

```
gluLookAt(glDouble eyeX, glDouble eyeY, glDouble eyeZ,
      glDouble centerX, glDouble centerY, glDouble centerZ,
      glDouble upX, glDouble upY, glDouble upZ)
```

Peirntah tersebut mendeffinisikan matrik viewing dan mengalikan dengan matrik yang sedang aktif. Titik pandang yang diperlukan ditentukan oleh *eyeX, eyeY, eyeZ.* Argumen *centerX, centerY, centerZ* menyatakan sembarang titik sepanjang garis pandang tetapi biasanya ada beberapa titik di tengah garis pandang tersebut. Argumen *upX, upY, upZ* menunjukkan dimana arah atas.

3.4.3 Transformasi Proyeksi

Transformasi proyeksi menentukan bagaimana suatu obyek ditransformasikan ke layar. Ada dua macam proyeksi yang ada pada OpenGL yaitu perspektif dan ortografik. Proyeksi perspektif adalah proyeksi yang melihat obyek seperti segala sesuatu yang ada di kehidupan sehari–hari. Maksudnya adalah bagian obyek yang letaknya jauh dari pemandangan akan tampak lebih kecil. Jika ingin menampilkan obyek sesuai dengan kenyataan maka dipilihlah proyeksi perspektif ini. Hal ini terjadi karena proyeksi perspektif menggunakan *viewing* volume seperti piramida terpotong bagian atasnya (frustrum). Sintaks

perintah `gluPerspective(Gldouble fovy, Gldouble aspect, Gldouble zNear, Gldouble zFar)` digunakan untuk menciptakan matrik pandangan perspektif dan mengalikan dengan matrik terakhir pada stack. Argumen fovi adalah sudut dari bidang pandang pada bidang x-z yang nilainya berkisar antara (0.0, 180.0). Argumen *Aspect* adalah rasio sebagai hasil dari pembagian *weight* dan *height* pada *fustrum*. Argumen zNear dan Zfar adalah jarak antara titik pandang dan *clipping plane* sepanjang sumbu z negatif dan nilainya selalu positif.

Proyeksi *orthographic* adalah proyeksi yang menampilkan obyek ke layar tanpa mengubah ukuran dari obyek yang sesungguhnya. Proyeksi ini sering dipakai dalam dalam desain arsitek dan CAD. Sintaks perintah `glOrtho(Gldouble left, Gldouble right, Gldouble bottom, Gldouble top, Gldouble near, Gldouble far)` digunakan untuk menghasilkan matriks *orthographic viewing volume* dan mengalikannya dengan matriks terakhir. *Near clipping plane* adalah segi empat dengan sudut kiri bawah yang ditentukan oleh *(left bottom, -near)* dan sudut kanan atas yang ditentukan oleh *(right, top, -near)*.

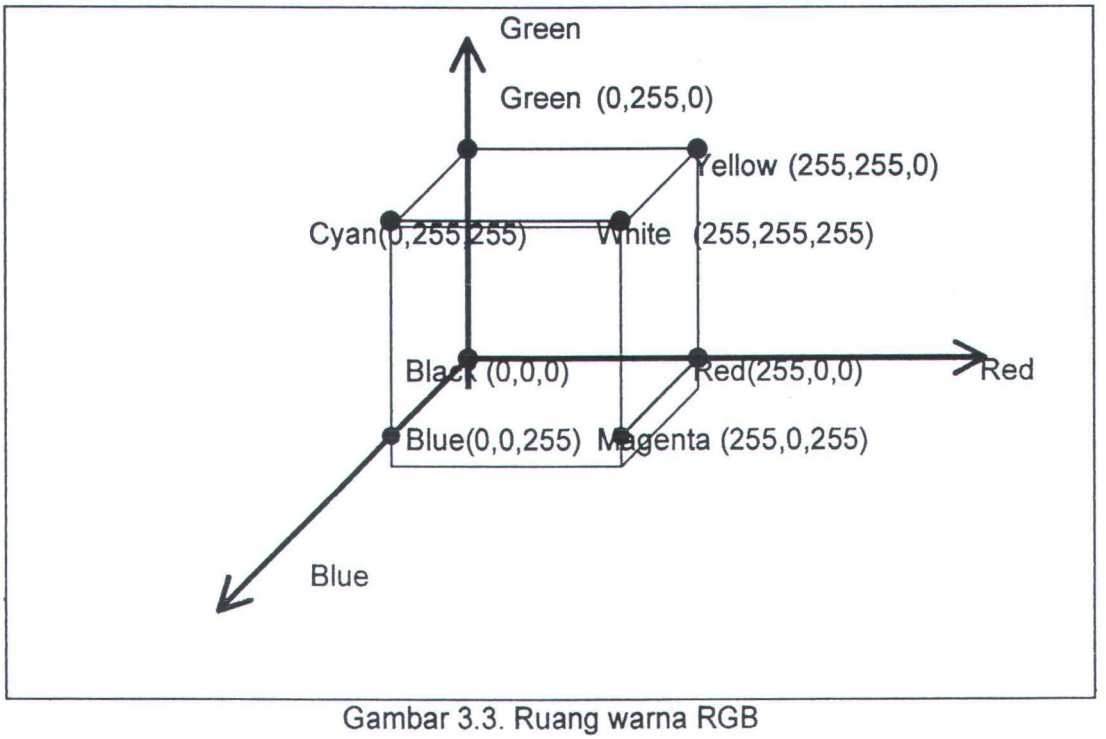
3.4.4 Transformasi Viewport

Transformasi *viewport* menspesifikasikan daerah untuk gambar yang akan menempati layar komputer. *Viewport* sendiri artinya bidang segiempat pada window dimana *image* digambar. Transformasi perspektif dan vieport mengatur bagaimana ruang pandang terhadap obyek di-mapping di layar komputer. Jadi cara kerja transformasi ini sama dengan proses akhir fotografi yaitu mendefinisikan ukuran dan lokasi pada gambar. Sintaks perintah

`glViewPort(Glint x, Glint y, Glsizei width, Glsizei height)` digunakan untuk mendefinisikan segiempat pixel dari window dimana gambar terakhir telah di-mapping. Parameter (x,y) menspesifikasikan sudut kiri bawah dari *viewport* yang telah di-inisialisasi (0,0), sedangkan *weidth* dan *heigth* adalah ukuran dari segi empat *viewport,* biasanya adalah lebar dan tinggi window.

3.5 Pewarnaan pada OpenGL

OpenGL menspesifikasikan warna sebagai gabungan intensitas komponen merah, hijau dan biru. Berdasarkan ini maka dapat dibentuk suatu ruang warna RGB yang menunjukkan kombinasi warna yang dapat digunakan. Gambar 3. Dibawah ini menunjukkan warna sebagai ruang dengan merah, hijau dan biru sebagai sumbunya.



Gambar 3.3. Ruang warna RGB

3.6 Pencahayaan

Metode pencahayaan pada OpenGL terdiri dari tiga komponen cahaya yaitu *ambient, Diffuse* dan *specular.* Cahaya *Ambient* adalah cahaya yang datang kepermukaan obyek dari sembarang arah sebagai akibat dari pantulan ruangan di sekeliling obyek sehingga tidak mempunyai arah datang yang khusus. Cahaya *Diffuse* adalah cahaya yang dari arah tertentu tetapi dipantulkan ke segala arah oleh permukaan obyek. Jika cahaya diarahkan langsung tanpa melalui sudut tertentu maka obyek pada sisi dimana sinar menyorot akan tampak lebih terang dari sis yang lain. Cahaya *Specular* adalah cahaya yang datang dari arah tertentu dan terpantul ke arah tertentu pula.

BAB IV

PERANCANGAN DAN IMPLEMENTASI PERANGKAT LUNAK

BAB IV

PERANCANGAN DAN

IMPLEMENTASI PERANGKAT LUNAK

Dalam bab ini dibahas mengenai perancangan dan implementasi dari perangkat lunak yang dihasilkan yang merupakan penerapan dari teori-teori yang telah diuraikan pada Bab II dan Bab III sebelumnya.

Perangkat lunak dibuat dengan memakai bahasa pemrograman Visual C++ 6.0 dan menggunakan sistem operasi *Windows 98*. Perangkat lunak ini juga ditunjang oleh OpenGL sebagai API yang dipakai untuk merancang, menspesifikasikan berbagai macam obyek dan menghasilkan aplikasi tiga dimensi yang interaktif.

4.1 Perancangan Data

Setiap perungkat lunak pasti terdiri dari dua komponen utama yaitu data dan proses. Perangkat lunak ini secara garis besar mengelompokkan data-data yang berkaitan dengan proses pembuatan obyek tiga dimensi menjadi tiga bagian yaitu data masukkan, data saat pemrosessan dan data keluaran.

4.1.1 Data Masukan

Data masukkan yang akan digunakan perangkat lunak ini berasal dari berkas masukkan yang bertipe teks. Data masukan tersebut berupa data lengkap dari obyek yaitu level, jumlah patch sepanjang u dan v, indeks u dan v, knot pertama u dan v, titik referensi (*reference point*) dan titik offset (*offset point*).

Tata bahasa yang digunakan penulis untuk menyusun deskripsi data obyek yang tersimpan dalam berkas masukan yang bertipe teks adalah sebagai berikut :

[Level] = *level dari mesh*
[UPatch] = *jumlah patch sepanjang u*
[VPatch] = *jumlah patch sepanjang v*
[UIndex] = *indeks awal sepanjang u pada mesh level diatasnya*
[VIndex] = *indeks awal sepanjang v pada mesh level diatasnya*
[UKnots] = *nilai knots vektor pertama untuk u*
[VKnots] = *nilai knots vektor pertama untuk v*
[dU] = *delta u*
[dV] = *delta v*
[Reff] = *titik referensi*
x: *koordinat x*, **y:** *koordinat y*, **z:** *koordinat z*, **w:** *koordinat w*
[Off] = *titik offset*
x: *koordinat x*, **y:** *koordinat y*, **z:** *koordinat z*, **w:** *koordinat w*
[Status] = *status dari mesh*

Keterangan untuk masing-masing data adalah sebagai berikut :

1. *Level* menunjukkan nomor level dari mesh yang direpresentasikan.
2. *UPatch* menunjukkan jumlah patch yang ada sepanjang u.
3. *VPatch* menunjukkan jumlah patch yang ada sepanjang u.
4. *Uindex* menyatakan indeks awal yang digunakan sepanjang u yang ada pada mesh level diatasnya.
5. *Vindex* menyatakan indeks awal yang digunakan sepanjang v yang ada pada mesh level diatasnya.
6. *UKnots* menjukkan nilai knots vektor yang pertama untuk u yang digunakan oleh mesh tersebut.

7. *VKnots* menjukkan nilai knots vektor yang pertama untuk v yang digunakan oleh mesh tersebut.
8. *dU* menyatakan selisih u yang digunakan.
9. *dV* menyatakan selisih v yang digunakan.
10. *Reff* menunjukkan koordinat dari masing-masing titik referensi yang ada pada mesh tersebut, yang tediri dari nilai x, y, z dan w.
11. *Off* menunjukkan koordinat dari masing-masing titik offset yang ada pada mesh tersebut, yang tediri dari nilai x, y, z dan w.
12. *Status* menunjukkan status posisi dari mesh tersebut.

4.1.2 Data Saat Pemrosesan

Pada saat perangkat lunak ini bekerja dalam membuat pemodelan dan mem-visualisasikan obyek ke layar ada beberapa data utama yang berperan penting yaitu :

1. Knot Vektor, data ini dihasilkan untuk melakukan penghitungan mesh dan permukaan obyek.
2. Data mesh, data yang berisi semua parameter yang diperlukan untuk menggambarkan mesh termasuk koordinat titik referensi, koordinat titik offset dan fungsi basis untuk masing-masing titik kontrol.
3. Data permukaan, data yang berisi semua parameter yang diperlukan untuk menggambarkan permukaan. Data ini dibagi menjadi dua bagian, untuk arah u dan untuk arah v yang masing-masing terdiri dari koordinat titik referensi dan koordinat titik offset serta fungsi basis untuk masing-masing titik kontrol.
4. Koordinat windows data mesh, data yang berisi koordinat pada windows. Semua data koordinat vertex diubah oleh proses transformasi menjadi

koordinat window yang diperlukan oleh OpenGL untuk melakukan proses penggambaran.

5. Koordinat windows data permukaan, data yang berisikan koordinat windows.
6. Data cahaya dan permukaan, data yang berisi data permukaan berikut dengan data properti dari cahaya untuk permukaan tersebut. Data ini dihasilkan dari proses pencahayaan yang menggunakan *library* OpenGL.
7. Data cahaya, material dan permukaan, data yang berisi data permukaan berikut dengan data properti dari cahaya dan material untuk permukaan tersebut. Data ini dihasilkan dari proses pencahayaan dan inisialisasi material yang menggunakan *library* OpenGL.
8. Koordinat mata, koordinat yang sesuai dengan pandangan mata manusia.
9. Koordinat *Clipped*, adalah koordinat hasil dari proses transformasi proyeksi dimana semua data yang ada diluar *viewing volume* yang digunakan tidak akan digambarkan.
10. Koordinat *Normalized Device*, adalah koordinat *clipped* dibagi dengan koordinat w sebagai hasil dari proses *Perspective Division*.
11. Koordinat *Window*, koordinat akhir dari proses transformasi yan digunakan OpenGL untuk menggambarkan obyek pada layar.

4.1.3 Data Keluaran

Hasil pengolahan dari perangkat lunak ini adalah sebuah gambar yang menampilkan obyek yang dimodelkan. Keluaran lain dari perangkat lunak ini adalah dari dari obyek itu sendiri yang disimpan dalam bentuk berkas teks yang selanjutnya dapat dibuka kembali oleh perangkat lunak ini untuk ditampilkan kembali jika ingin melakukan perubahan.

4.2 Perancangan Proses

Pada prinsipnya perangkat lunak ini terdiri dari proses utama yaitu pemodelan dan visualisasi. Secara rinci kedua proses utama tersebut akan dibahas berikut ini.

4.2.1 Perancangan Proses Pemodelan

Proses pemodelan yang dilakukan adalah melakukan penghitungan data masukkan secara lengkap sehingga mendapatkan data lengkap dari kurva dan permukaan yang kemudian akan divisualisasikan. Masukkan yang digunakan adalah adalah data masukan seperti yang telah dijelaskan pada subbab 4.1.1. diatas. Selanjutnya data masukan tersebut diproses sehingga mendapatkan data komplek yang akan digunakan untuk menghasilkan gambar obyek.

Proses pertama yang dilakukan adalah melakukan pembangkitan data mesh. Pada proses ini didapatkan data masing-masing mesh yang terbentuk berdasarkan level dari mesh yang ada. Hasil dari proses pembangkitan data mesh ini digunakan untuk mendapatkan data permukaan yang nantinya akan digunakan dalam proses penggambaran.

Proses penghitungan yang berikutnya adalah melakukan proses penghitungan knot vektor apabila terdapat perubahan titik kontrol. Pada awal perhitungan knot vektor ini dianggap konstan dan begitu terjadi perubahan letak titik pada layar baru melakukan proses penghitungan knot vektor. Hasil proses penghitungan ini digunakan untuk mendapatkan fungsi basis yang baru dari masing-masing titik kontrol yang ada. Untuk selanjutnya fungsi basis ini digunakan untuk mendapatkan data permukaan yang baru dari obyek yang akan

digambarkan. Untuk lebih jelasnya mengenai proses yang ada pada perangkat lunak ini dapat dilihat pada subbab 4.3 Data Alir Diagram.

4.2.2 Perancangan Proses Visualisasi

Proses penggambaran atau visualisasi terdiri dari empat bagian subproses yaitu transformasi, pencahayaan, inisialisasi material dan proses penggambaran pada layar. Keempat proses tersebut menggunakan library dari OpenGL.

Proses transformasi bertujuan merubah koordinat vertex menjadi koordinat windows karena OpenGL melakukan proses penggambaran dengan menggunakan koordinat layar bukan koordinat sesungguhnya yang diinputkan pemakai. Proses ini sendiri dibagi lagi menjadi empat proses dan sudah dijelaskan dalam **Bab III** pada bagian **Analogi Kamera.**

Proses penggambaran obyek dibedakan menjadi dua bagian yaitu penggambaran kurva dan penggambaran permukaan. Untuk penggambaran kurva data yang diperlukan adalah koordinat windows data kurva.

Sedangkan untuk proses penggambaran permukaan, koordinat windows data permukaan tidak langsung digunakan tapi terlebih dahulu diolah dengan proses pencahayaan dan proses inisialisasi material untuk permukaan obyek sehingga gambar yang dihasilkan lebih menunjukkan efek tiga dimensi yang sesungguhnya.

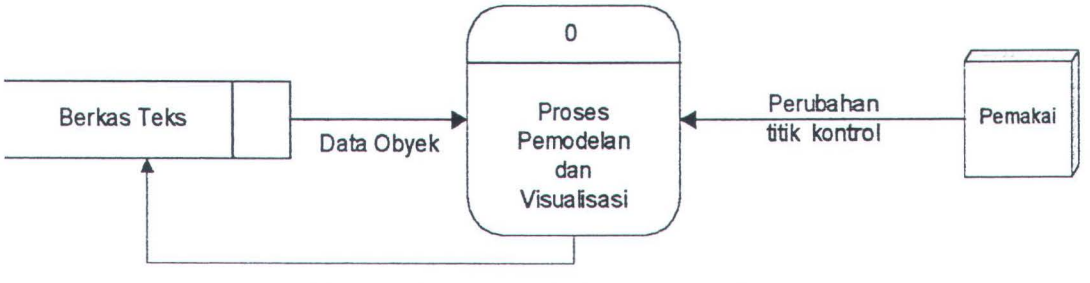
Proses pencahayaan adalah proses pengaktifan perhitungan cahaya untuk permukaan dan elemen sumber cahaya apa saja yang akan digunakan. Data permukaan tadi dimasukkan ke dalam proses pencahayaan menghasilkan

data permukaan yang sudah mengandung elemen cahaya untuk selanjutnya digunakan dalam proses inisialisasi material.

Pada proses inisialisasi material ini permukaan diberi empat macam elemen material. Untuk keterangan lebih lanjut baca Bab III bagian Pencahayaan dan Properti Material. Setelah keluar dari proses ini maka data obyek tiga dimensi untuk permukaan sudah lengkap yaitu data permukaan yang mengandung properti cahaya dan meterial. Data ini selanjutnya akan digunakan dalam proses penggambaran untuk menghasilkan data gambar.

4.3 Diagram Alir Data (Data Flow Diagram)

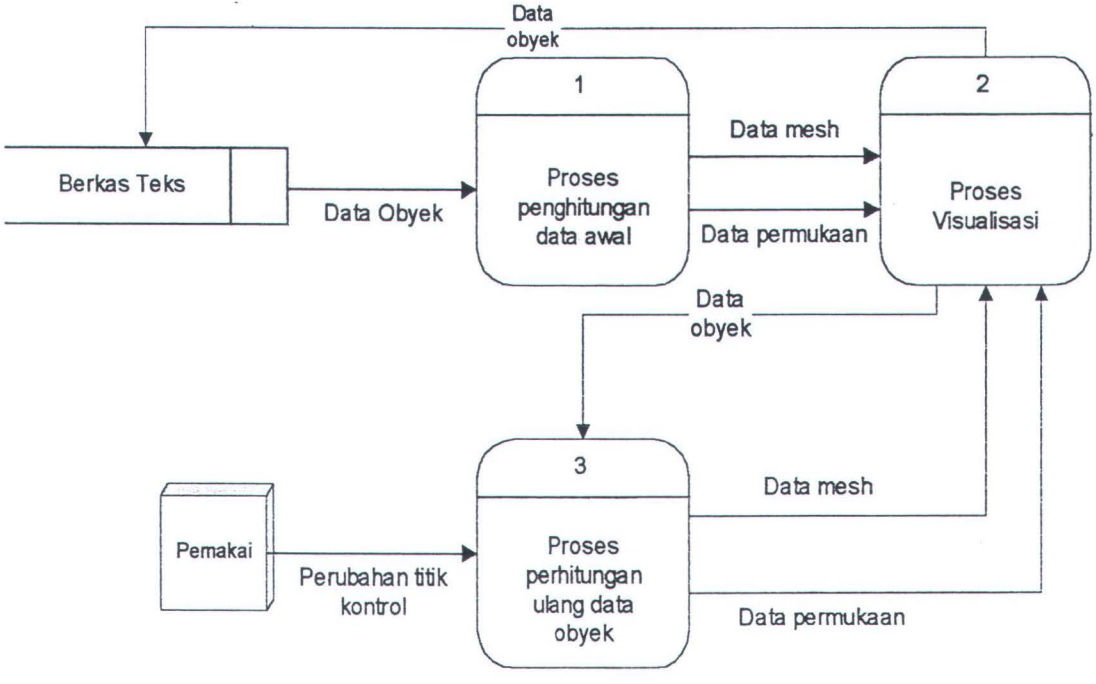
Data alir diagram untuk perangkat lunak ini ditunjukkan dalam gambar-gambar berikut ini. Pada gambar 4.1. ditampilkan data alir diagram level 0.



Gambar 4.1. Data alir diagram level 0

Data yang digunakan pada DAD level 0 ini berasal dari berkas masukan yang bertipe teks. Data masukan dari berkas ini berisi data lengkap dari obyek seperti yang telah dijelaskan sebelumnya pada bagian data masukan. Selain masukan dari berkas pada DAD level 0 ini juga terdapat masukan dari pemakai. Masukan ini memberikan data baru pada obyek yang secara langsung diberikan oleh pemakai yaitu berupa data titik kontrol dari obyek yang telah

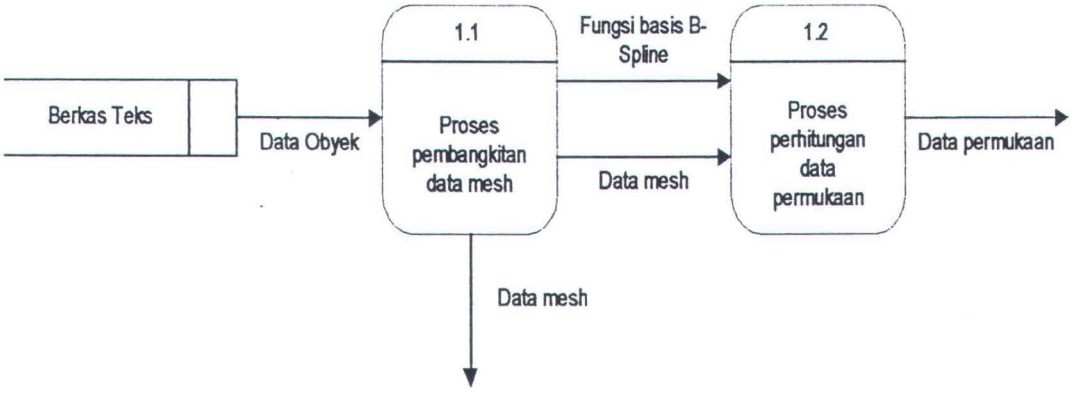
divisualisasikan. Hal ini dapat dilakukan dengan cara pemakai langsung memilih titik yang akan dipindahkan dengan menggunakan mouse kemudian menariknya pada suatu titik tertentu. Sehingga masukan langsung dari pemakai baru dapat dilakukan setelah obyek divisualisasikan terlebih dahulu. Data obyek merupakan hasil keluaran dari perangkat lunak ini dan disimpan dalam berkas teks yang nantinya dapat dibuka kembali jika ingin ditampilkan atau melakukan perubahan pada obyek tersebut. Data alir diagram level 0 diatas memiliki DAD level 1 seperti yang ditunjukkan pada Gambar 4.2 yang masing-masing prosesnya akan diuraikan lebih lanjut pada DAD level 2.



Gambar 4.2. Data alir diagram level 1, Proses Pemodelan dan Visualisasi

Data yang digunakan pada DAD level 1 ini adalah data obyek lengkap yang diambil dari berkas teks yang digunakan untuk proses perhitungan data untuk menghasilkan semua data mesh dari obyek dan data permukaan, yang

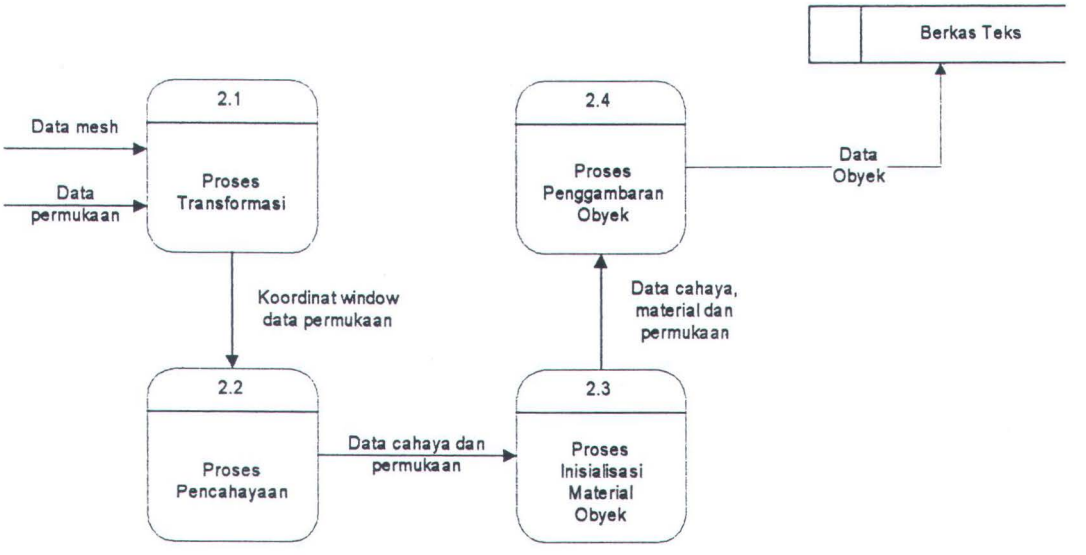
selanjutnya digunakan untuk proses visualisasi (penggambaran) pada layar. Data obyek yang sudah divisualisasi ini akan digunakan sebagai masukan bersama sama dengan masukan dari pemakai untuk proses perhitungan ulang dari obyek. Proses ini akan menghasilkan data mesh dan data permukaan yang baru yang selanjutnya juga akan divisualisasikan atau digambarkan pada layar melalui proses visualisasi.



Gambar 4.3. Data alir diagram level 2, Proses Penghitungan Awal

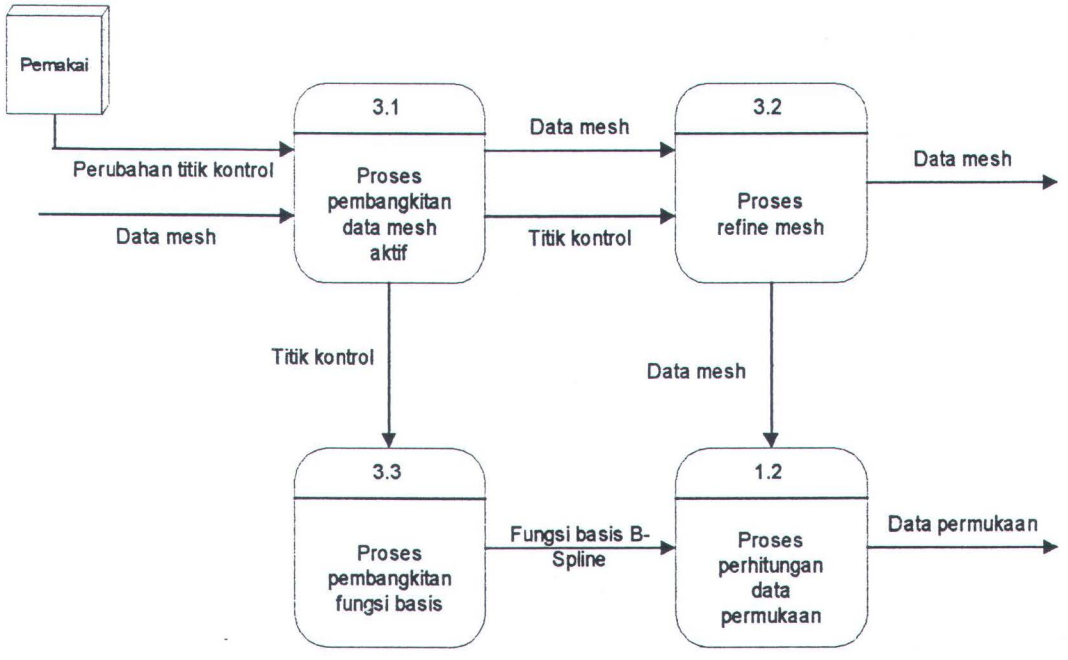
DAD level 2 proses penghitungan data awal ini merupakan proses yang mengolah data obyek lengkap untuk pertama kali yaitu pada saat berkas teks yang berisi data obyek dibuka dan dibaca untuk kemudian divisualisasikan.

Pada DAD level 2 proses penghitungan data awal ini, data obyek digunakan untuk proses pembangkitan data mesh. Data mesh dan fungsi basis yang dihasilkan digunakan untuk proses perhitungan data permukaan. Pada proses perhitungan data permukaan ini akan dihasilkan data permukaan yang masih origin, dan bersama-sama dengan data mesh yang dihasilkan oleh proses sebelumnya akan digunakan untuk proses penggambaran.



Gambar 4.4. Data alir diagram level 2, Proses Visualisasi

Pada DAD level 2 untuk proses visualisasi (penggambaran), data mesh dan data permukaan sebelum digambardi layar harus terlebih dahulu diubah menjadi koordinat window dengan menggunakan proses transformasi. Khusus untuk penggambaran obyek permukaan, data hasil keluaran proses transformasi diproses lebih lanjut dalam proses pencahayaan dengan mengaktifkan elemen cahaya untuk obyek permukaan. Selain proses pencahayaan, data permukaan juga diproses dalam inisialisasi material untuk memilih elemen apa saja yang akan digunakan oleh obyek permukaan sehinggga dihasilkan obyek tiga dimensi yang realistik. Dan tahap paling akhir adalah proses penggambaran obyek permukaan yang menghasilkan gambar obyek tiga dimensi dan data gambar tersebut data disimpan dalam berkas keluaran yang bertipe teks dengan format sesuai dengan aturan penulisan yang telah dijelaskan sebelumnya. Tahapan proses transformasi akan lebih lanjut dijelaskan pada gambar 4.6.

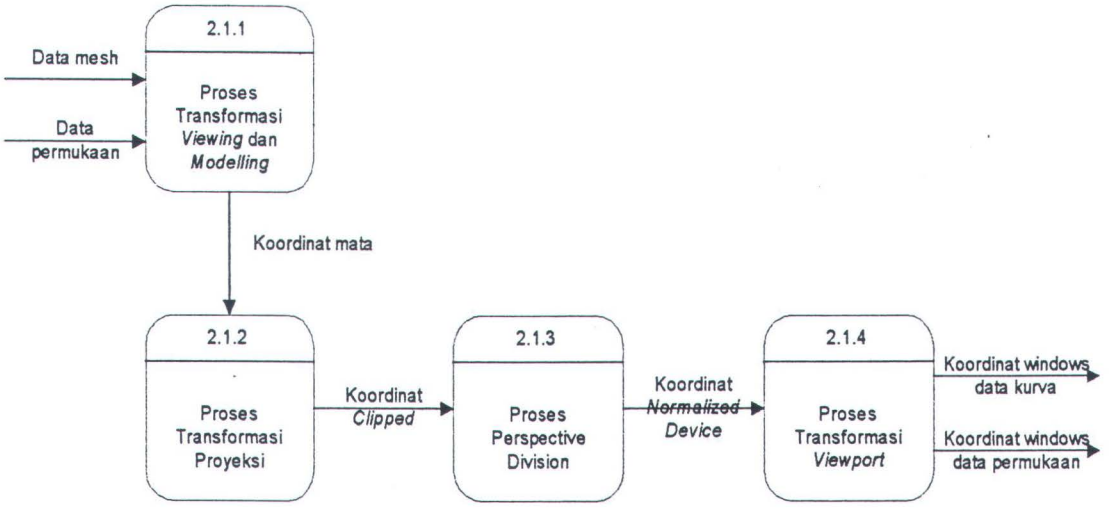


Gambar 4.5. Data alir diagram level 2, Proses Penghitungan Ulang Data

DAD tersebut menggambarkan proses penghitungan data obyek setelah pemakai mengubah data pada obyek yang telah divisualisasikan sebelumya. Masukan dari user bersama sama dengan data mesh telah yang ada sebelumnya diproses kembali untuk mendapatkan data mesh yang aktif dalam artian mendapatkan data mesh yang mengalami perubahan pada proses pembangkitan data mesh aktif. Data mesh dan titik kontrol yang digunakan pada proses *refine mesh* yang akan menghasilkan data mesh yang baru, sementara titik kontrol juga digunakan pada proses pembangkitan fungsi basis dimana fungsi basis *B-Spline* yang dihasilkan akan digunakan pada proses penghitungan data permukaan bersama san dengan data mesh yang baru yang dihasilkan oleh proses *refine mesh*. Proses ini menghasilkan data permukaan yang baru yang

bersama dengan data mesh yang baru akan digunakan pada proses visualisasi (penggambaran) pada layar.

Gambar 4.6. menunjukkan data alir diagram yang terjadi dalam proses transformasi yang mengubah koordinat vertex menjadi koordinat *windows*.



Gambar 4.6. Data alir diagram level 3, Proses Transformasi.

4.4 Implementasi Struktur Data

Untuk menunjang perancangan perangkat lunak diciptakan beberapa tipe data utama meliputi point, Mesh, World dan Window. Berikut ini akan dibahas garis besar dari struktur data yang ada.

Untuk menyimpan data titik yang digunakan yang memiliki komponen-komponen x, y, z dan w direpresentasikan sebagai vektor yang bertipe float sebagai berikut.

```
typedef float Vector[4]; // representasi vvektor
typedef Vector point;  // representasi titik (x,y,z,w)
typedef float Matrix[16]; // representasi matrik
```

Untuk menyimpan semua data dari sebuah mesh yang memiliki kompenen-komponen seperti level, patch, titik referensi dan titik offset direpresentasikan dalam struct berikut ini.

```
struct _Mesh {
  int level; // nomor level mesh
  int m;     // jumlah patch arah u
  int n;     // jumlah patch arah v
  int m0;    // initial mesh arah u
  int n0;    // initial mesh arah v
  point *ref;  // reference point
  point *off;  // offset point
  Matrix *poly[3]; //koef. polynomial

  UIPoint **uipts; // list point
  float u0;    //initial u
  float v0;    //initial v
  float du;
  float dv;
  Mesh **mask;
  Mesh *upper;
  Mesh *lower;
  Mesh *prev;
  Mesh *next;
};
```

Untuk menyimpan semua data permukaan yang digunakan diciptakan tipe data struct World. Semua data pada saat proses berjalan disimpan dalam tipe data ini, untuk selanjutnya dipakai dalam proses visualisasi obyek. Kecuali untuk data koordinat mata, koordinat clipped, koordinat normalized device dan koordinat windows semuanya transparan karena data diolah sekaligus pada saat pemanggilan fungsi-fungsi transformasi OpenGL.

```
typedef struct _World {
  Orient Orients[4]; // orientation

  Matrix TopProj;  // tampak atas
  Matrix FrontProj;// tampak depan
  Matrix SideProj; // tampak samping
```

```
  Matrix PerProj; // tampak perspektif

  SolidType Solid;// type obyek

  Mesh *surf;  // data mesh
  uipList uipl; // list point
  UIPoint **uipts; // list UI point
  int numuipts; // jumlah ui point
  int selected; // flag untuk seleksi obyek
  GLuint AxesList; // list axes

  int level; // level
  BOOL DrawAxes; // flag penggabaran axes
  BOOL MultiColor; // flag color
  BOOL ConPoints; // flag  control point
  BOOL SelectPoint; // flag select point
  BControl BCMode;
  int CurrButton; // handle mouse
  int StartX; // posisi mouse  X awal
  int StartY; // posisi mouse  Y awal
  int LastX; // posisi mouse  X akhir
  int LastY; // posisi mouse  Y akhir
  float Aspect; // aspect
  int Subdiv; // pembagi

  ProjectionType Projection;  // select projection

} World;
```

4.5 Implementasi Proses

Proses utama yang ada pada perangkat lunak ini adalah bagaimana menghitung vektor knot baru apabila ada perubahan pada titik kontrol yang dilakukan oleh pemakai. Apabila pemakai memindahkan point di salah satu mesh yang ada maka mesh akan memindahkan titik point. Masukan untuk prosedur ini adalah indeks (m,n), level dari titik yang dipindahkan dan vektor yang merepresentasikan titik yang dipindahkan.

```
void meshMove(Mesh *mesh, int m, int n, int level, Vector v)
{
  if(level <0)
    Error("In meshMove, invalid level \n");
  if(m<0 || m> mesh->m)
    Error("In meshMove, invalid m \n");
```

```
  if(n<0 || n> mesh->n)
    Error("In meshMove, invalid n \n");
  if( EOA((mesh->mask),m,n,mesh->m) && EOA((mesh->mask),m-1,
      n,mesh->m) && EOA((mesh->mask),m,n-1,mesh->m) &&
      EOA((mesh->mask),m-1,n-1,mesh->m))
    Warning("In meshMove, picked point is not at the top
             level. \n");
  if( mesh->level < level)
    Warning("In meshMove, this point is not allowed
             to be picked at the current level.\n");*/
  else while(mesh->level != level){
    m = mesh->m0 + m/2;
    n = mesh->n0 + n/2;
    mesh= mesh->upper;
  }

  vScale(v,36.0/16);
  vAdd(EOA((mesh->off),m+1,n+1,mesh->m+3),v,
       EOA((mesh->off),m+1,n+1,mesh->m+3));
  updateSurf(mesh, MAX(m-2,-1), MAX(n-2,-1),
             MIN(m+2, mesh->m+1),MIN(n+2,mesh->n+1));

}
```

Apabila prosedur ini dijalankan akan merubah semua data permukaan dimana terdapat mesh yang berubah. Mesh akan memindahkan data titik kontrol berdasarkan perubahan titik yang ada. Sebagai catatan perubahan permukaan yang terjadi hanya pada permukaan yang berhubungan langsung dengan mesh yang memiliki titik yang berubah posisi tersebut.

Apabila terjadi perubahan titik kontrol maka akan terjadi perubahan-perubahan pada mesh dimana titik tersebut ada. Perubahan yang ada tersebut akan mengakibatkan perubahan pada level *refinement* di sekitar titik yang dipindahkan. Untuk menginisialisasi perubahan level dilakukan dengan tahapan berikut.

```
  tm=allocMesh(4,4);
  tm->upper=mesh;
  tm->level= mesh->level + 1;
  tm->m0 = m-1;
```

```
tm->n0 = n-1;
tm->du = mesh->du/2;
tm->dv = mesh->dv/2;
tm->u0 = mesh->u0 + tm->m0 * mesh->du;
tm->v0 = mesh->v0 + tm->n0 * mesh->dv;
    p0=-1;
    q0=-1;
    p1=5;
    q1=5;
    for(i=p0;i<=p1;i++)
      for(j=q0;j<=q1;j++){
        getRef(tm,i,j);
        vSet(EOA((tm->off),i+1,j+1,tm->m+3), 0, 0, 0, 0);
  }
uip = EOA((tm->uipts),2,2,tm->m+1)= EOA((mesh->uipts), m,
          n, mesh->m+1);
EOA((mesh->uipts),m,n,mesh->m+1)=(UIPoint *)-1;
uip -> mesh = tm;
uip -> m = 2;
uip -> n = 2;
```

Setelah melakukan inisialisasi proses selanjutnya adalah melakukan cek apabila perubahan titik yang terjadi mempengaruhi beberapa mesh disekitarnya. Apabila hal ini terjadi akan dilakukan proses penggabungan mesh.

```
flag=0;
  p0 = MAX(tm->m0,0);
  q0 = MAX(tm->n0,0);
  p1 = MIN(tm->m0+tm->m/2-1, mesh->m-1);
  q1 = MIN(tm->n0+tm->n/2-1, mesh->n-1);
  for(i=p0;i<=p1;i++)
    for(j=q0;j<=q1;j++)
      if(!flag && (om=EOA((mesh->mask),i,j,mesh->m))){
        flag=1;
      }
while(flag){
  flag=0;

    mm=meshMerge(om,tm);

    p0 = MAX(mm->m0,0);
    q0 = MAX(mm->n0,0);
    p1 = MIN(mm->m0+mm->m/2-1, mesh->m-1);
    q1 = MIN(mm->n0+mm->n/2-1, mesh->n-1);
    for(i=p0;i<=p1;i++)
```

```
        for(j=q0;j<=q1;j++)
          if(EOA((mesh->mask),i,j,mesh->m)==om)
            EOA((mesh->mask),i,j,mesh->m) = NULL;
          else if(!flag)
            if((nm=EOA((mesh->mask),i,j,mesh->m))){
              flag=1; }

      deleteMesh(mesh,om);
      freeMesh(om);
      freeMesh(tm);
      tm=mm;
      om=nm;
  }
```

Setelah proses penggabungan mesh dilakukan maka proses selanjutnya adalah mendapatkan representasi polinomial dan titik-titik baru. Titik-titik baru yang mungkin ada akan disimpan ditambahkan dalam list yang yang berisi data titik yang telah ada.

```
getPolyRep(tm);

  ul=uiplNewList();
  for(i=2;i<=tm->m-2;i++)
    for(j=2;j<=tm->n-2;j++)
      if(!(EOA((tm->uipts),i,j,tm->m+1)))
        if(i%2 || j%2){
            EOA((tm->uipts),i,j,tm->m+1)= getUIPoint
                                          (tm,i,j,0.0,0.0);
            uiplAppend(ul,EOA((tm->uipts),i,j,tm->m+1));
        }
        else{
         EOA((tm->uipts),i,j,tm->m+1)=
         EOA((mesh->uipts),tm->m0+i/2,tm->n0+j/2,mesh->m+1);
         uip=EOA((tm->uipts),i,j,tm->m+1);
         EOA((mesh->uipts),tm->m0+i/2,tm->n0+j/2,mesh->m+1)
              = (UIPoint *)-1;
          if(uip==NULL)
             Warning("In meshRefine, uip==NULL\n");
          if(uip==(UIPoint *)-1)
             Warning("In meshRefine, uip== -1\n");
          uip -> mesh = tm;
          uip -> m = i;
          uip -> n = j;
        }


  tm -> upper = mesh;
```

```
  insertMesh(mesh,tm);

  p0 = MAX(tm->m0,0);
  q0 = MAX(tm->n0,0);
  p1 = MIN(tm->m0+tm->m/2, mesh->m);
  q1 = MIN(tm->n0+tm->n/2, mesh->n);
  for(i=p0; i<p1; i++)
    for(j=q0; j<q1; j++)
        EOA((mesh->mask),i,j,mesh->m) = tm;
  return ul;
}
```

Apabila pemindahan titik yang dilakukan oleh pemakai tidak menyebabkan penambahan penambahan titik maka proses ini akan mengembalikan nilai NULL. Proses ini akan menyebabkan terbentuknya mesh yang baru sehingga menyebabkan perubahan pada data permukaan.

Untuk proses visualisasi perangkat lunak ini menggunakan *library* OpenGL, proses inisialisasi dari OpenGL yang digunakan adalah menset pencahayaan dan material.

```
  GLfloat light_position[] = { 0.3, 0.5, 0.8, 0.0 };
  GLfloat lm_ambient[] = { 0.4, 0.4, 0.4, 1.0 };

  glLightfv(GL_LIGHT0, GL_POSITION, light_position);
  glLightModelfv(GL_LIGHT_MODEL_AMBIENT, lm_ambient);
  glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);

  glEnable(GL_LIGHT0);
  glDepthFunc(GL_LEQUAL);
  glEnable(GL_DEPTH_TEST);
  glShadeModel(GL_SMOOTH);
  glPointSize(6.0);
  glEnable(GL_COLOR_MATERIAL);
  glColorMaterial(GL_FRONT_AND_BACK, L_AMBIENT_AND_DIFFUSE);

  glFlush();
```

Sementara untuk menginisialisasi penggunaan ruang pandang tiga dimensi digunakan setviewport. Proses ini dilakukan untuk memberi batasan dari ruang pandang yang digunakan dalam menampilkan obyek. Apabila terdapat

sebagian obyek yang akan ditampilkan berada diluar ruang pandang yang telah disiapkan maka bagian dari obyek tersebut tidak akan ditampilkan.

```
void worldReshapeViewPort(World *w, int sizex, int sizey,
WinQuad domwq)
{
  int iWidth, iHeight;

  iWidth = sizex;
  iHeight = sizey;

  worldSetViewports(w,sizex,sizey,domwq);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

  if (iWidth < iHeight)
    glOrtho(-1.2, 1.2, -1.2/w->Aspect, 1.2/w->Aspect,
    DIST_TO_CLIP, 100);
  else
    glOrtho(-1.2*w->Aspect, 1.2*w->Aspect, -1.2, 1.2,
    DIST_TO_CLIP, 100);
  glGetFloatv(GL_PROJECTION_MATRIX,w->TopProj);
  glGetFloatv(GL_PROJECTION_MATRIX,w->FrontProj);
  glGetFloatv(GL_PROJECTION_MATRIX,w->SideProj);

  glLoadIdentity();
  gluPerspective(VIEW_ANGLE, w->Aspect, DIST_TO_CLIP, 100);
  glGetFloatv(GL_PROJECTION_MATRIX,w->PerProj);

  glMatrixMode(GL_MODELVIEW);
  glFlush();
}
```

Karena terdapat empat buah ruang pandang dalam satu layar maka harus dicek terlebih dahulu posisi dari masing-masing ruang pandang tersebut. Ruang pandang tersebut adalah ruang pandang yang digunakan untuk menampilkan obyek tampak depan, tampak samping, tampak atas dan menampilkan obyek secara perspektif.

```
void worldSetViewports(World *w, int sizex, int sizey,
WinQuad domwq)
{
  w->Orients[wqFront].vpllx = 0;
  w->Orients[wqFront].vplly = sizey*PORT_TOP_LOW;
  w->Orients[wqFront].vpwid = sizex/2;
```

```
  w->Orients[wqFront].vphei = sizey*(PORT_TOP_HI -
      PORT_TOP_LOW);

  w->Orients[wqSide].vpllx = sizex/2;
  w->Orients[wqSide].vplly = sizey*PORT_TOP_LOW;
  w->Orients[wqSide].vpwid = sizex/2;
  w->Orients[wqSide].vphei = sizey*(PORT_TOP_HI -
      PORT_TOP_LOW);

  w->Orients[wqTop].vpllx = 0;
  w->Orients[wqTop].vplly = sizey*PORT_BOT_LOW;
  w->Orients[wqTop].vpwid = sizex/2;
  w->Orients[wqTop].vphei = sizey*(PORT_BOT_HI -
      PORT_BOT_LOW);

  w->Orients[wqPer].vpllx = sizex/2;
  w->Orients[wqPer].vplly = sizey*PORT_BOT_LOW;
  w->Orients[wqPer].vpwid = sizex/2;
  w->Orients[wqPer].vphei = sizey*(PORT_BOT_HI -
      PORT_BOT_LOW);

  w->Aspect = ((GLdouble)sizex)/(sizey*
                      ((PORT_TOP_HI - PORT_TOP_LOW)+
                     (PORT_TOP_HI - PORT_TOP_LOW)));

  if (domwq != wqNone) {
    w->Orients[domwq].vpllx = 0;
    w->Orients[domwq].vplly = 0;
    w->Orients[domwq].vpwid = sizex;
    w->Orients[domwq].vphei = sizey;
    w->Aspect = ((GLdouble)sizex)/((GLdouble)sizey);
  }
}
```

BAB V

UJI COBA DAN PEMBAHASANNYA

BAB V

UJI COBA DAN PEMBAHASANNYA

Dalam bab ini dijabarkan hasil uji coba terhadap perangkat lunak yang diimplementasikan dalam Tugas Akhir ini, serta pembahasannya terhadap hasil uji coba tersebut. Untuk memperjelas gambaran mengenai perangkat lunak visualisator ini sebelumnya akan dibahas sepintas mengenai sistem pendukung yang digunakan berupa spefisikasi perangkat keras dan perangkat lunak.

5.1 Spesifikasi Sistem Pendukung

Uji coba penciptaan obyek tiga dimensi oleh perangkat lunak visualisator ini dilakukan dengan menggunakan spesifikasi perangkat keras sebagai berikut :

- Prosesor Intel Pentium 166.
- Memory Fisik 32 Megabyte EDO.
- VGA Memory 1 Megabyte.
- Hard Disk 4,5 Gigabyte.
- Display 16 juta warna.

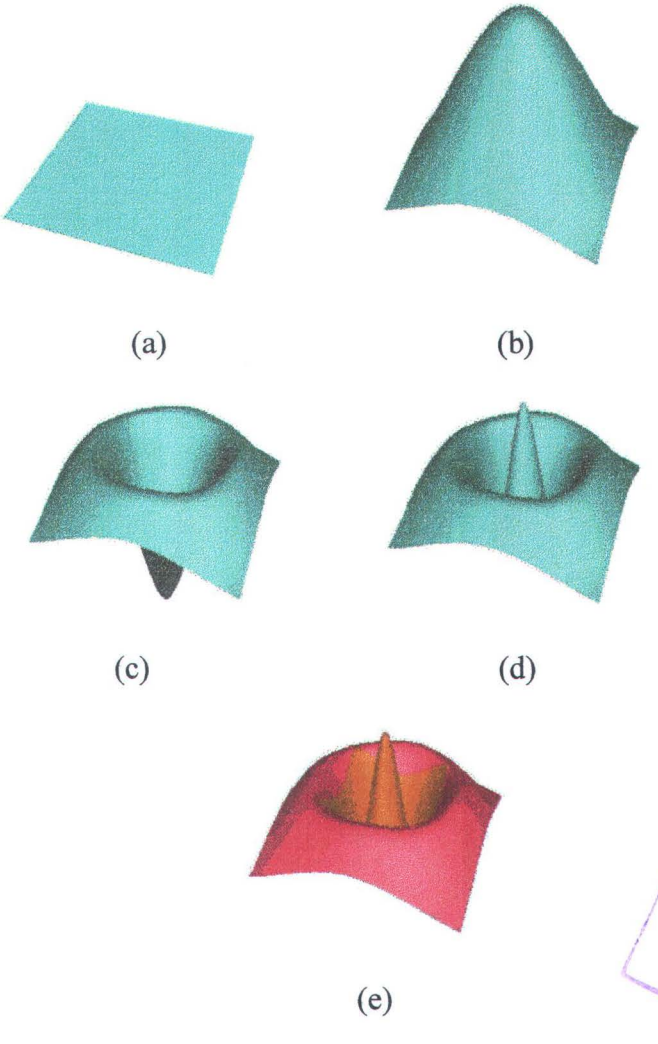
Kebutuhan perangkat lunak sebagai sistem pendukung yang membangun perangkat lunak visualisator ini adalah sebagai berikut :

- Sistem operasi Windows 95.
- Bahasa Penrograman Visual C++ Versi 6.
- Application Programming Interface (API) OpenGL.

5.2 Hasil Uji Coba

Gambar-gambar berikut ini adalah gambar yang didapatkan dengan menggambil obyek pada ruang pandang yang telah disediakan. Gambar-gambar tersebut diambil pada ruang pandang perspektif dengan kondisi seperlunya seperti melakukan rotasi dan translasi sehingga obyek yang tergambar dapat dilihat seperti dibawah ini.

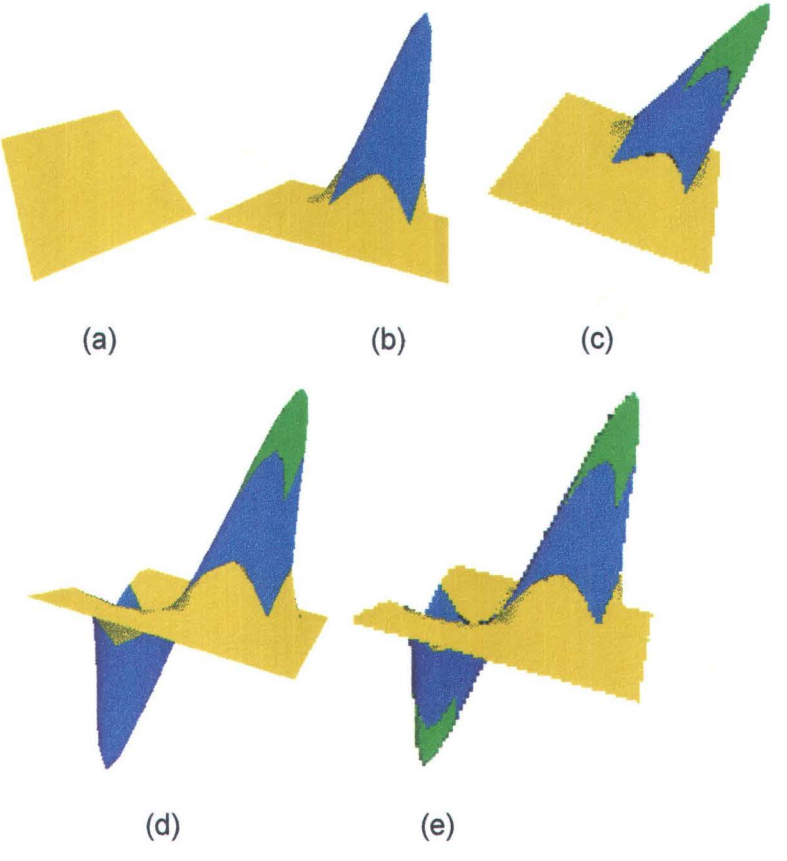
Berikut ini akan diperlihatkan urutan proses hingga dapat membentuk obyek dengan tingkatan hirarki yang berbeda.



Gambar 5.1. Gambar Hasil Uji Coba 1 (Plop)

Keterangan gambar 5.1 :

a. Obyek dasar dengan patch 5X5 dan digambarkan dengan kondisi solid (bukan wireframe) merupakan obyek dengan level 0.
b. Menarik sebuah titik pada obyek yang telah tergambar, sehingga merubah level dari obyek tersebut menjadi 1.
c. Menarik kembali titik yang tadi ditarik pada arah yang berlawanan, obyek yang terbentuk merupakan obyek dengan level 2.
d. Melakukan hal sama dengan seperti pada langkah ketiga (c), sehingga menghasilkan obyek dengan tingkat level 3.
e. Obyek lengkap, pembedaan level dari obyek diperlihatkan dengan perbedaan warna.

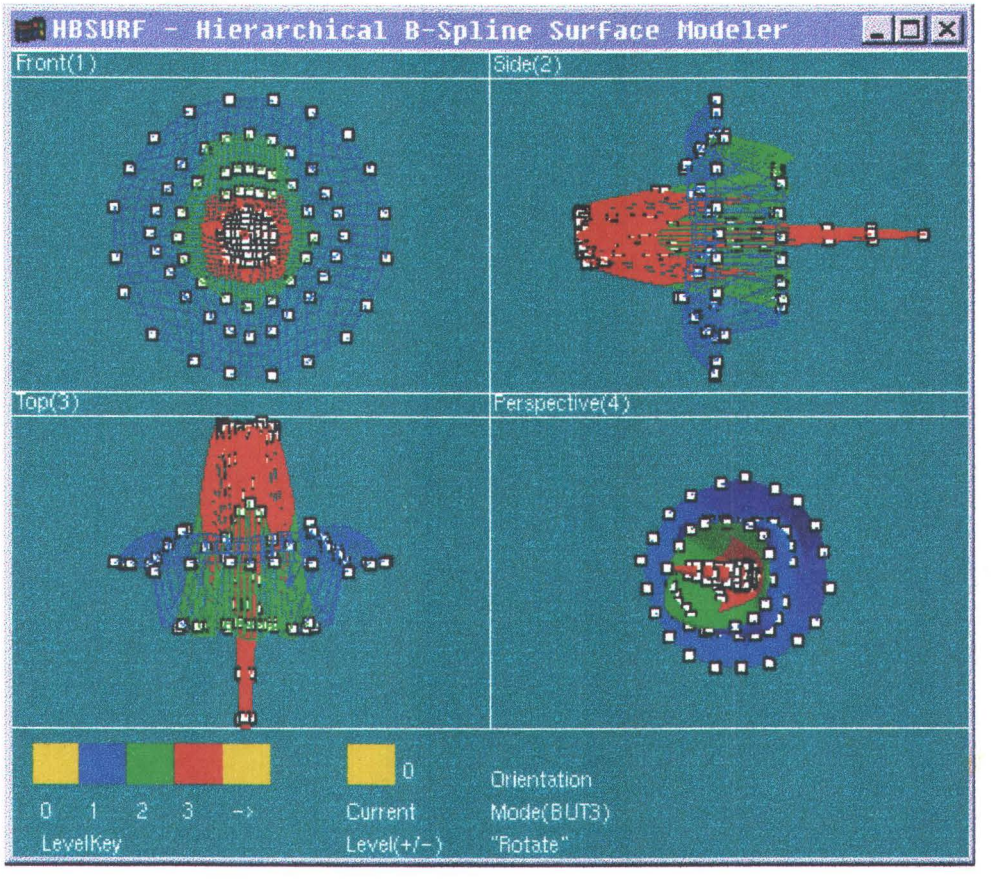


Gambar 5.2. Gambar Hasil Uji Coba 2

Gambar 5.2. memperlihatkan hasil uji coba pada bidang dasar dengan patch 5x5 dimana pada bidang tersebut dilakukan pemindahan dua titik yang berbeda. Pada gambar tersebut diperlihatkan hirarki yang terjadi pada setiap perubahan. Tingkatan hirarki atau level ditunjukkan dengan perbedaan warna.

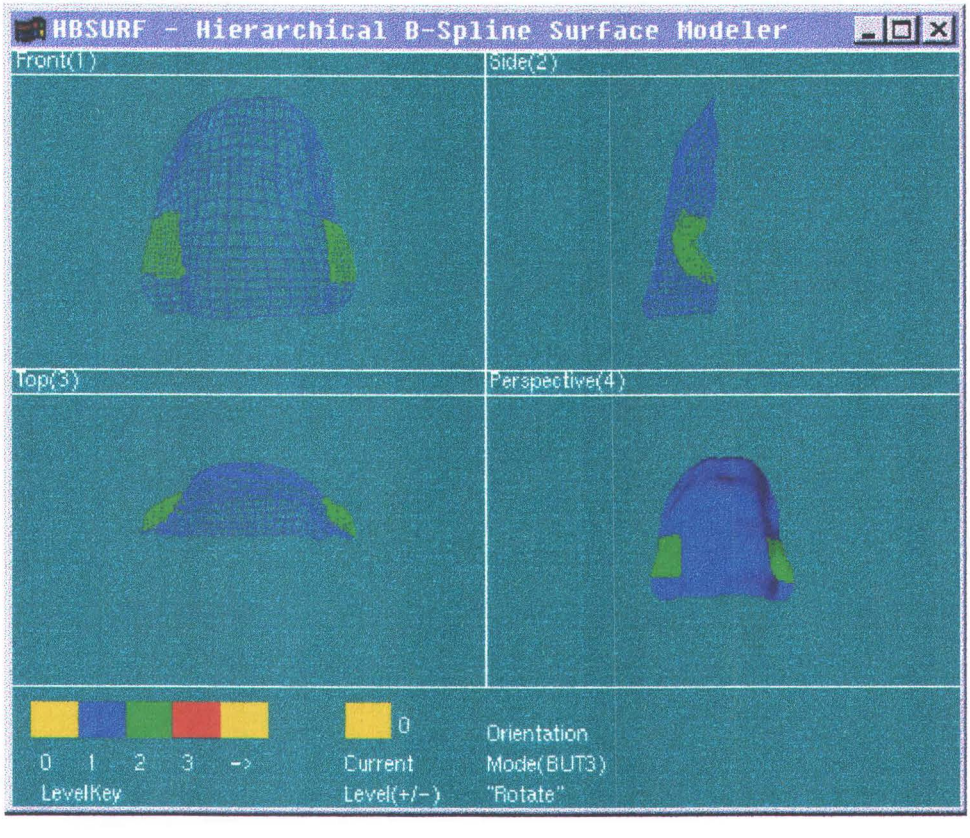
Dari uji coba yang telah dilakukan seperti yang terlihat pada gambar 5.1 dan gambar 5.2 diatas dapat dilihat tingkatan-tingkatan hirarki dari masing-masing obyek.

Berikut ini akan diperlihatkan beberapa obyek yang lebih komplek yang memiliki tingkat hirarki level yang berbeda.



Gambar 5.3. Obyek Concent yang digambarkan dengan Hierarchical B-Spline

Pada gambar diatas terlihat bahwa obyek tersebut merupakan obyek yang memiliki tingkatan level sampai level 4. Dari gambar tersebut diatas juga terlihat bahwa semakin tinggi tingkat level dari suatu bagian (domain) permukaan, maka domain tersebut memiliki jumlah detail yang semakin rapat. Titik-titik yang terlihat berwarna putih adalah merupakan kontrol point, dimana kontrol point tersebut dapat diubah baik letak maupun level dari domain yang ditempatinya.



Gambar 5.4. Obyek Creature

Pada gambar diatas diperlihatkan obyek dengan mengabaikan kontrol point yang ada. Obyek diatas dibuat dengan hirarki tertinggi sebesar 2, juga diperlihatkan detail yang ada pada obyek tersebut.

BAB VI

PENUTUP

BAB IV

PENUTUP

Dalam bab ini diuraikan beberapa hal yang menjadi kesimpulan dari hasil uji coba terhadap perangkat lunak yang diimplementasikan dalam Tugas Akhir ini, serta beberapa saran bagi pembaca yang diharapkan pada waktu mendatang dapat menyempurnakan perangkat lunak ini sehingga semakin berguna di bidang ilmu grafika komputer.

6.1 Kesimpulan

Beberapa kesimpulan yang dapat ditarik dari hasil uji coba perangkat lunak ini adalah sebagai berikut :

1. Dengan tersedianya perangkat lunak ini pemakai dapat melakukan modifikasi terhadap obyek tiga dimensi yang telah tervisualisasikan dengan mudah, hanya dengan memindah titik-titik yang diperlukan saja. Perangkat lunak yang dibuat mampu membuat dan memvisualisasikan obyek-obyek tiga dimensi *free form* berdasarkan hirarki pembentuk obyek. Karena memiliki independent multilevel *B-Spline* linier maka penghalusan (*refinement*) dapat dilakukan dengan menggunakan kontrol secara lokal, dan semua level memiliki kebebasan linier dari b-splines sehingga menyederhanakan seleksi untuk *B-Splines*.
2. Fasilitas transformasi yang digunakan untuk merubah posisi obyek dilakukan dengan menggunakan fasilitas transformasi yang disediakan oleh *library* OpenGL seperti glTranslate, glRotate dan glScale. Dari aspek pemrograman,

fasilitas yang disediakan oleh OpenGL ini cocok untuk perangkat lunak yang bertipe *pure viewer*. Karena fasilitas transformasi yang disediakan oleh *library* OpenGL mengubah posisi kamera, dan tidak mengubah posisi obyek terhadap kamera sehingga posisi titik kontrol obyek selalu tetap dan pemakai tidak dapat mengubah kembali posisi titik kontrol. Sedangkan perangkat lunak ini dibuat dengan tujuan utama merubah posisi titik kontrol sehingga dapat mengubah bentuk obyek. Agar dapat mengubah titik kontrol maka pada saat obyek diubah posisinya kamera harus tetap supaya posisi koordinat titik kontrol yang baru dapat diubah kembali.

6.2 Kemungkinan Pengembangan

Perangkat lunak ini dilengkapi dengan fasilitas standar seperti menggerakkan obyek (translation) sepanjang sumbu x, y dan z, memutar obyek (rotation) berdasarkan sumbu x, y dan z, memperbesar dan mengecilkan obyek (zooming) serta melihat obyek dari empat sisi yaitu tampak depan, tampak samping, tampak atas dan perspektif.

Selain itu perangkat lunak ini juga menyediakan fasilitas khusus yaitu dapat menggerakkan dan mengubah titik kontrol obyek sehingga dapat mengubah bentuk dari obyek secara keseluruhan.

Beberapa kemungkinan pengembangan lebih lanjut yang dapat dilakukan pada perangkat lunak ini adalah sebagai berikut:

1. Menambahkan fasilitas untuk menampilkan obyek-obyek yang berbentuk geometri regular yang memiliki persamaan geometri umum seperti *conic section*, tabung (*cylinder*) dan bola (*sphere*).

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. Anton, Howard, "*Elementary Linear Algebra (Fifth Edition)*", Anton Textbook, Inc, 1987,
2. Eck, Matthras and Hoppe, Huques, " *Automatic Reconstruction of B-Splines Surfaces of Arbitrary Topological Type*", University of Dhamstadt.
3. Forsey, David and Wong, David, "*Multiresolution Surface Reconstruction For Hierarchical B-Splines*", The Department of Computer Science, The University of British, Columbia, 1995.
4. Hill, Francis S, "*Computer Graphics*", Maxwell Macmilan, New York, 1996.
5. Kraft, Rainer, "*Adaptive and Linearly Independent Multilevel B-Splines*", Proceedings of Chamonix, 1996.
6. Mortenson, Michael M, "*Geometric Modelling (Second Edition)*", John Willey And Sons, Inc, New York, 1997.
7. Roger, David F, "*Mathematical Elements For Computer Graphics (Second Edition)*", McGraw Hill, New York, 1985.
8. Stollnitz, Eric J, DeRose, Tony D and Salesin, David H, "*Wavelets for Computer Graphics : A Primer Part 2*", IEEE Computer Graphics And Applications, 15(4) 75, Juli 1995.
9. Watt, Allan, "*Advanced Animation And Rendering Techniques*", Adison Wesley Publisher Ltd, New York, 1992.