BAB 2

LANDASAN TEORI

2.1 Teori-teori Dasar / Umum

Pada bab ini menjelaskan beberapa teori secara umum yang secara mendasar digunakan dalam pembuatan pemodelan 3D jembatan *cable stayed* dan *suspension* di dalam web.

2.1.1 Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak adalah penetapan dan penggunaan prinsip-prinsip rekayasa untuk mendapatkan perangkat lunak yang ekonomis, yaitu perangkat lunak yang dapat diandalkan dan bekerja efisien pada mesin yang riil (Roger, 2005, p10) .

Dalam penelitian ini kita menggunakan paradigma Waterfall Model atau Classic Life Cycle sebagai metode perancangan.

2.1.1.1 Pengertian Perangkat Lunak

Dalam buku Roger (2005,p10), perangkat lunak adalah :

1. Perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang digunakan.
2. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional.
3. Dokumen yang menggambarkan operasi dan kegunaan program.

2.1.1.2 Karakteristik Perangkat Lunak

Menurut Roger (2005,p10-14), perangkat lunak memiliki ciri khas yang berbeda dari perangkat keras, yaitu :

1. Perangkat lunak dibangun dan dikembangkan, tidak dibuat dalam bentuk yang klasik.

   Meskipun banyak kesamaan di antara pabrik perangkat keras dan perangkat lunak, aktivitas keduanya secara mendasar sangat berbeda. Dalam kedua aktivitas tersebut, kualitas yang tinggi dicapai melalui perancangan yang baik, tetapi di dalam fase pembuatan perangkat keras, selalu ditemukan masalah kualitas yang tidak mudah untuk disesuaikan dengan perangkat lunak.

2. Perangkat lunak tidak pernah usung.

   Perangkat lunak tidak rentan terhadap pengaruh lingkungan yang merusak yang menyebabkan perangkat keras menjadi usung.

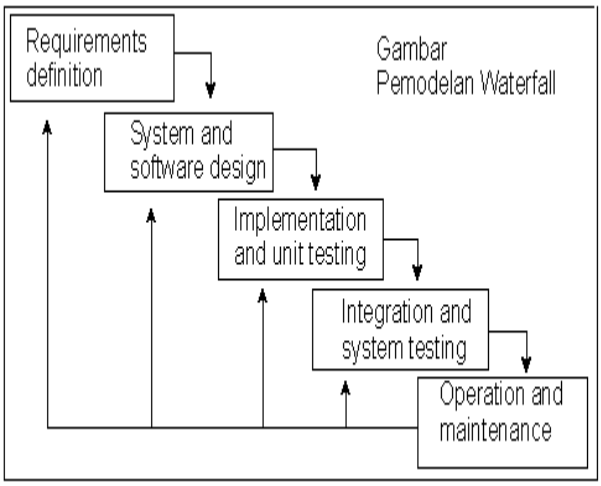
3. Sebagian besar perangkat lunak dibuat secara *custom-built,* serta tidak dapat dirakit dari komponen yang sudah ada.

2.1.1.3 Paradigma Perangkat Lunak Waterfall Model

Pada paradigma sekuensial linear atau yang disebut dengan Waterfall Model, mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial

yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan.

Fase-fase dalam Waterfall Model menurut referensi Sommerville :



Gambar 2.1 Pemodelan Waterfall

1. *Requirements definition*

   Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada perangkat lunak. Untuk memahami sifat program yang dibangun, harus memahami domain informasi, tingkah laku, unjuk kerja, dan antarmuka *(interface*) yang diperlukan.

2. *System and software design*

   Karena perangkat lunak selalu merupakan bagian dari sebuah sistem yang lebih besar, kerja dimulai dengan membangun syarat dari semua elemen sistem dan mengalokasikan beberapa subset dari kebutuhan ke perangkat lunak tersebut.

3. *Implementation and unit testing*

   Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara unit.

4. *Integration and system testing*

   Penyatuan unit-unit program kemudian diuji secara keseluruhan (*system testing*).

5. *Operation and maintenance*

   Mengoperasikan program dilingkungannya dan melakukan pemeliharaan, seperti penyesuaian atau perubahan karena adaptasi dengan situasi sebenarnya.

2.1.2 *Unified Modelling Language (UML)*

UML adalah sebuah bahasa yang telah menjadi standar dalam industri untuk menvisualisasikan, menspesifikasi, merancang, dan mendokumentasi sistem peranti lunak (Booch et al, 1999, p14).

UML menyediakan kumpulan alat yang sudah terstandarisasi, yang digunakan untuk mendokumentasikan analisis dan perancangan sebuah sistem perangkat lunak (Kendall & Kendall, 2005, p663).

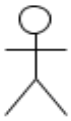
2.1.2.1 *Use Case* Diagram

Use case diagram menggambarkan sekumpulan *use case* dan *actor* serta hubungannya (Booch et al, 1999, p234). Sebuah *use case* menggambarkan interaksi antara *actor* dengan sistem.

Simbol-simbol yang digunakan pada *use case* diagram yaitu :

1. *Actor*

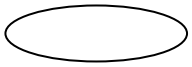
   Menspesifikasikan seperangkat peranan yang user dapat perankan ketika berinteraksi dengan sistem.



Gambar 2.2 Simbol *actor* dalam *use case* diagram

2. *Use Case*

   Sebuah deskripsi dari seperangkat aksi-aksi yang berurutan yang ditampilkan sebuah sistem.



Gambar 2.3 Simbol *use case* dalam *use case* diagram

3. *Association*

   Sebuah relasi *structural* yang menghubungkan antara *actor* dengan *use case*.

Gambar 2.4 Simbol *association* dalam *use case* diagram

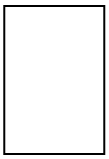
4. *Extend*

   Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

Gambar 2.5 Simbol *extend* dalam *use case* diagram

5. *System Boundary*

   Menggambarkan batasan dari sistem yang akan dibuat yang mengelilingi sejumlah use case.



Gambar 2.6 Simbol *system boundary* dalam *use case* diagram

2.1.2.2 Sequence Diagram

Sequence diagram (Booch et al, 1999, p245) menggambarkan sekumpulan objek dan interaksinya, termasuk *message* yang dikirim terhadap urutan waktu. *Sequence* diagram digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai tanggapan dari sebuah *event* untuk menghasilkan keluaran tertentu.

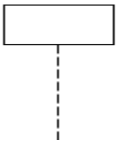
Diawali dari apa saja yang memicu aktifitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan keluaran yang dihasilkan. Masing-masing objek memiliki *lifeline* vertical sedangkan *message* digambarkan secara horizontal.

Simbol-simbol yang digunakan dalam sequence diagram, yaitu:

1. *Actor*, melambangkan user yang berhubungan dengan sistem dan melaksanakan sequence yang terkait.

Gambar 2.7 Simbol *actor* dalam *sequence* diagram

2. *Object Lifeline*, merupakan sebuah garis yang merepresentasikan adanya sebuah objek dalam jangka waktu tertentu.

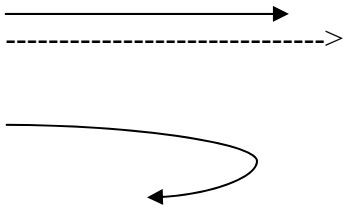


Gambar 2.8 Simbol *object lifeline* dalam *sequence* diagram

3. *Activition*, menggambarkan periode waktu ketika pemrosesan terjadi dalam objek tersebut.

Gambar 2.9 Simbol *activation* dalam *sequence* diagram

4. *Message, Return, Callback Message*, penyampaian pesan dari satu objek ke objek lain atau ke diri sendiri.



Gambar 2.10 Simbol *message, return, callback message* dalam *sequence* diagram

2.1.2.3 Activity Diagram

Activity diagram dirancang untuk menyederhanakan apa saja yang terjadi selama berlangsungnya sebuah operasi atau proses. *Activity* diagram sebenarnya merupakan flowchart yang menunjukkan aliran control dari suatu aktivitas ke aktivitas lain (Booch et al, 1999, p257). Jenis diagram ini biasanya digunakan

untuk merepresentasikan aliran kerja dan operasi obyek dalam sistem. *Activity* Diagram memberikan visualisasi, menspesifikasikan, mengkonstruksi serta mendokumentasikan kelompok obyek yang dinamis.

Activity Diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses berjalan, sementara *use case* menggambarkan bagaimana *actor* menggunakan sistem untuk melakukan aktivitas.

Simbol-simbol yang digunakan dalam *activity* diagram, yaitu :

1. *Initial state*, yaitu titik awal yang menandai di mulainya sebuah aktivitas.

Gambar 2.11 Simbol *initial state* dalam *activity* diagram

2. *Final state*, titik akhir yang menandai akhir dari sebuah aktivitas.

Gambar 2.12 Simbol *final state* dalam *activity* diagram

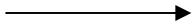
3. *Decision*, titik pengambilan keputusan dari beberapa kondisi.

Gambar 2.13 Simbol *decision* dalam *activity* diagram

4. *Action state*, *state* aksi yang mengeksekusi sebuah aksi dan kemudian mentransisikannya ke state lain.

Gambar 2.14 Simbol *action state* dalam *activity* diagram

5. *Control flow*, melambangkan transisi dari satu state ke state lain.



Gambar 2.15 Simbol *control flow* dalam *activity* diagram

2.1.3 *Worl Wide Web*

Menurut Turban (2005, p50), *World Wide Web* adalah aplikasi yang digunakan dalam internet yang berfungsi sebagai transportasi data yang diterima sebagai standar untk menyimpan, menerima dan *formatting*, dan menampilkan informasi melalui *client server architectur.*

2.1.4 *Web Application*

Menurut Roger (2005, p41), *web Application* adalah satuan aplikasi yang cukup luas, pada bentuk paling sederhana, *web-aplication* dapat berupa serangkaian *hypertext files* yang terhubung yang memberikan informasi berupa *text* dengan sedikit gambar/grafik. Seiring dengan perkembangannya, ia berkembang sehingga memiliki banyak fungsi, fitur, dan content, juga terhubung dengan database korporasi dan aplikasi bisnis yang rumit.

2.2 Teori Khusus

Adapun beberapa teori khusus yang digunakan dalam pembuatan pemodelan 3D jembatan *cable stayed dan suspension* di dalam web.

2.2.1 Jembatan

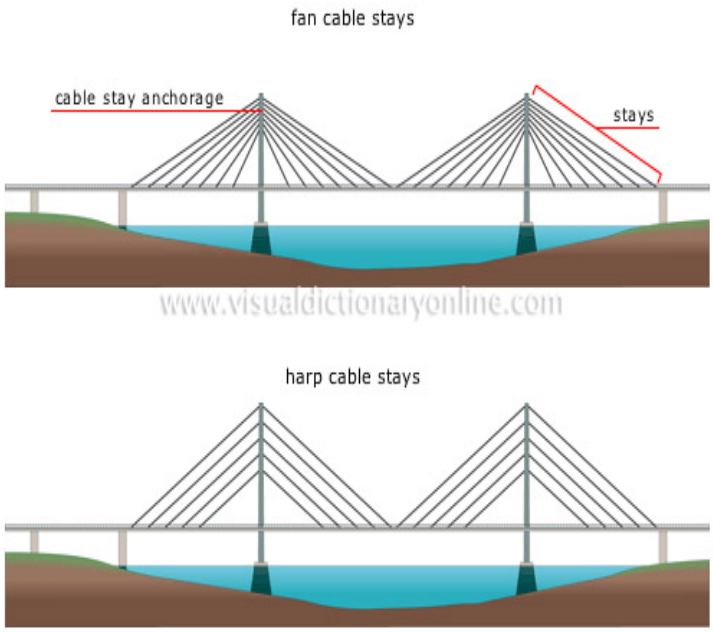
Jembatan merupakan satu struktur/bangunan yang dibuat untuk menyeberang di atas rintangan seperti sungai, jurang, rel kereta api ataupun jalan raya. Saat ini beberapa jembatan besar juga telah dibangun untuk menyeberangi laut menghubungkan antar pulau-pulau.

2.2.1.1 Jembatan *Cable Stayed*

Jembatan *cable stayed* (Kabel Tetap) sudah dikenal sejak lebih dari 200 tahun yang lalu (Walther, 1988) yang pada awal era tersebut umumnya dibangun dengan menggunakan kabel vertical dan miring seperti Dryburgh Abbey Footbridge di Skotlandia yang dibangun pada tahun 1817. Sejak saat itu jembatan *cable stayed* mengalami banyak perkembangan dan mempunyai bentuk yang bervariasi dari segi material yang digunakan maupun segi estetika. Pada umumnya jembatan *cable stayed* menggunakan gelagar baja, rangka, beton atau beton pratekan sebagai gelagar utama (Zarkasi dan Rosliansjah, 1995).

(www.visualdictionaryonline.com)

Dua jenis desain utama jembatan *cable-stayed* :



Gambar 2.16 Dua desain utama jembatan *cable-stayed*

Kelebihan jembatan cable stayed :

1. Kabel lurus memberikan kekakuan yang lebih besar dari kabel melengkung. Disamping itu, analisis non linier tidak perlu dilakukan untuk geometri kabel lurus.
2. Kabel diangker pada lantai jembatan dan menimbulkan gaya aksial tekan yang menguntungkan secara ekonomis dan teknis.
3. Tiap – tiap kabel penggantung lebih pendek dari panjang jembatan secara keseluruhan dan dapat diganti satu persatu.

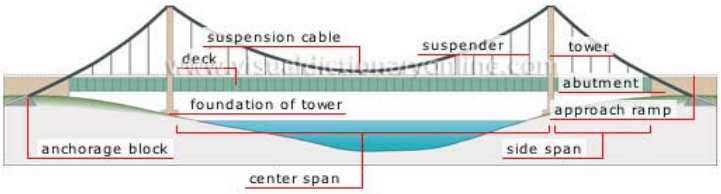
Kelemahan Jembatan *Cable Stayed*

1. Diperlukan metode pelaksanaan yang cukup teliti jika jembatan *Cable Stayed* dibangun dengan bentang yang lebih panjang, bagian yang terkantilever sangat rentan terhadap getaran akibat angin selama masa konstruksinya.
2. Sama halnya dengan jembatan penggantung, kabel penggantungnya memerlukan perawatan yang intensif untuk melindungi dari karat.

2.2.1.2 Jembatan *Suspension*

Tipe jembatan *suspension* atau jembatan gantung ini sering digunakan untuk jembatan bentang panjang. Pertimbangan pemakaian tipe jembatan gantung adalah dapat dibuat untuk bentang panjang tanpa pilar ditengahnya. Jembatan gantung terdiri atas pelengkung penggantung dan batang penggantung (hanger) dari kabel baja, dan bagian yang lurus berfungsi mendukung lalulintas (dek jembatan). Selain bentang utama, biasanya jembatan gantung mempunyai bentang luar (*side span*) yang berfungsi untuk mengikat atau mengangkerkan kabel utama pada balok angker.

(www.visualdictionaryonline.com)



Gambar 2.17 Desain struktur jembatan *suspension*

Kelebihan Jembatan Gantung

1. Seluruh struktur jembatan dapat dibangun tanpa perancah dari tanah.
2. Struktur utamanya nampak gagah dan mengekspresikan fungsinya dengan baik.
3. Merupakan pilihan yang ekonomis untuk jembatan dengan panjang bentang lebih dari 600 meter.

Kelemahan Jembatan Gantung :

Apabila lantai kerja tidak cukup kaku, maka jembatan penggantung akan bergoyang dan menjadi tidak stabil jika terkena angin dan getaran akibat resonansi, seperti pada jembatan Tacoma Narrows, Seattle, Amerika dan jembatan Millenium, River Thames, London.

2.2.2 Grafika Komputer

2.2.2.1 Pengertian Grafika Komputer

Menurut Syifaun Nafisah dan Nazrul Effendy (2010, p20), grafika komputer adalah seperangkat alat yang terdiri dari hardware dan software untuk membuat gambar, grafik atau citra realistic untuk seni, game computer, foto dan film animasi.

Sedangkan menurut J.D Foley dkk. (1996, p2), grafika computer merupakan bagian internal dari semua user interface computer, dan sangat diperlukan untuk mengvisualisasikan dua-dimensi (2D), tiga-dimensi (3D), dan dimensi obyek lainnya yang lebih tinggi.

2.2.2.2 Peranan dan Penggunaan Grafika Komputer

Grafika komputer digunakan di berbagai bidang seni, *sains*, bisnis pendidikan dan hiburan, antara lain (Nafisah, 2010, p21-22) :

1. Antarmuka Pengguna

   Antarmuka pengguna ini dipakai pada aplikasi komputer, yaitu dengan *Graphical User Interface* (GUI). Komponen utama GUI adalah *Window manager* dimana pengguna dapat mengatur tampilan dari *Windows.*

2. Perpetaan (*Cartography*)

   Aplikasi yang sering dimanfaatkan untuk keperluan perpetaan, baik untuk menyimpan, memanipulasi dan melihat peta melalui komputer, salah satunya adalah *Autocad Map*.

3. *Computer Aided Design*

   Paket aplikasi untuk CAD pada umumnya dilengkapi dengan beberapa *window* yang memperlihatkan beberapa gambar dari sudut pandang berbeda. Contoh aplikasi untuk CAD : *Autocad* dan 3D Studio.

4. Presentasi Grafik

   Grafika komputer dapat membuat suatu aplikasi yang dapat memberikan informasi yang kompleks dalam bentuk yang mudah dipahami dan dimengerti.

5. Pendidikan

   Beberapa aplikasi yang dapat digunakan dalam bidang pendidikan antara lain :

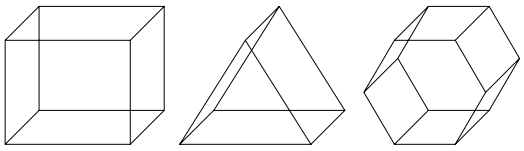
   - *Computer Assisted Testing* (CAT) – Ujian berbantuan computer.
   - *Computer Assisted Guidence* (CAG) – Pengarahan Berbantuan Komputer.
   - *Computer Managed Instruction* (CMI) – Komputer untuk merencanakan pembelajaran, evaluas belajar, serta secara langsung memantau prestasi siswa.

2.2.2.3 Pemodelan Objek 3D

Selain objek titik, garis, lingkaran dan *elipse* yang merupakan elemen dasar objek pada grafika, terdapat bermacam-macam bentuk objek, yaitu objek 2D ataupun objek 3D. Objek 2D adalah objek yang direpresentasikan di dalam sebuah bidang yang terdiri dari sumbu horizontal x dan sumbu vertical y. Sedangkan objek 3D adalah representasi objek dunia nyata (berbentuk tiga dimensi) ke dalam layar monitor yang merupakan bidang 2D, dengan tambahan sumbu ketiga yakni sumbu z. Sumbu z pada bidang 3D berfungsi sebagai atribut kedalaman sebuah objek yang digambarkan menembus layar monitor dari depan ke belakang.

Sebuah titik dalam bidang 2D terbentuk dari dua koordinat (x,y), dimana dua buah titik atau lebih dapat saling dihubungkan oleh garis. Kumpulan dari titik yang terhubung dengan garis dapat membentuk objek segi banyak tertutup (polygon) maupun objek segi banyak terbuka.

Objek 3D memiliki tiga buah koordinat (x,y,z) memungkinkan penempatan titik, objek dan polygon semakin serupa hingga sebuah objek 3D dapat memiliki volume.



Gambar 2.18 Bentuk-bentuk objek 3D

2.2.2.4 3D Objek dan Komponen

Computer modeling memiliki beberapa *building block* ruang biasa digunakan untuk menciptakan suatu objek. Semua komponen ini dibangun di atas satu sama lain untuk menciptakan kesatuan kompleks dengan kemampuan tambahan daripada komponen individu. Kesatuan ini merupakan titik, garis dan tepi, *polygon*, kurva dan permukaan (Capizzi, 2002, p43-48).

- *Points* (titik) atau disebut juga dengan *vertex*, tidak memiliki *volume* tetapi masih dapat ditemukan dengan menggunakan dimensi X, Y, dan Z.
- *Lines and Edges* (garis dan tepi). Garis didefinisikan dengan lokasi X, Y, dan Z dengan dua titik akhir. Tepi adalah suatu tipe dari garis yang didefinisikan dengan sisi perbatasan dari *polygon* atau suatu permukaan.
- *Polygon* adalah suatu permukaan yang ada antara tiga atau lebih tepi-tepi.
- *Curve* (kurva), yaitu dalah satu tipe garis yang biasanya digambarkan dengan beberapa titik dan berselisih dari garis lurus.
- *Surface* (permukaan), yaitu suatu bahan teoritis yang terbentang pada dua atau lebih kurva.

2.2.2.5 Tipe 3D Modeling

Menurut Kelly L.Murdock (2005, p306), ada beberapa tipe-tipe modeling antara lain:

- *Primtives* : Bentuk parametric dasar seperti kubus dan piramida.
- *Shapes* dan *splines* : Bentuk *vector* yang sederhana seperti: lingkaran, bintang, busur lingkaran dan tulisan dan *splines* seperti Helix. Objek-objek ini dapat sepenuhnya direnderer.
- *Meshes* : Model kompleks yang dibuat dengan banyak bentuk polygon yang dihaluskan bersama-sama saat objek di *render*.
- *Polys* : Objek yang terdiri dari berbagai macam *polygon*, mirip dengan objek mesh tetapi memiliki fitur yang unik.
- *Patches* : Berdasarkan dengan kurva *spilne, patches* dapat dimodifikasi.
- NURBS : Singkatan dari Non-Uniform Rational B-Splines. NURBS hampir sama dengan loft objek yang mana memiliki control points yang dapat mengkontrol bagaimana permukaan menyebar menyeluruh pada kurva.
- *Compound objects* : Beberapa kumpulan dari tipe modeling termasuk objek seperti *Booleans, loft*, dan *scatter object.*

2.2.2.6 Transformasi pada Objek 3D

Transformasi merupakan suatu proses yang dilakukan untuk memodifikasi bentuk objek tanpa merusak bentuk dasar

dari objek. Terdapat beberapa jenis transformasi, yaitu translasi, rotasi dan skala.

1. Translasi

Translasi adalah transformasi terhadap suatu objek dengan menggesernya dari suatu posisi ke posisi lain. Proses translasi dilakukan dengan melakukan operasi pertambahan suatu titik dengan jarak translasi. Misalkan suatu titik A(x,y) ditranslasikan sejauh $T(t_x,t_y)$, maka posisi titik setelah ditranslasikan adalah A'(x',y'). Pada translasi berlaku rumus :

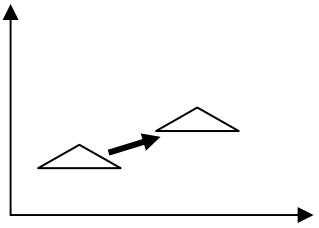
$$x' = x + t_x$$
$$y' = y + t_y$$

Persamaan di atas dapat juga direpresentasikan dengan menggunakan matrix.

$$A = \begin{bmatrix} x \\ y \end{bmatrix}, A' = \begin{bmatrix} x' \\ y' \end{bmatrix}, T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Sehingga, persamaan di atas dapat pula dituliskan dengan cara berikut :

$$A' = A + T$$



Gambar 2.19 Translasi pada objek 2D

Pada bidang 3 dimensi, metode yang digunakan merupakan pengembangan dari metode 2 dimensi dengan menyertakan koordinat z. Untuk melakukan translasi suatu titik A(x,y,z) sejauh $T(t_x,t_y,t_z)$ digunakan rumus :

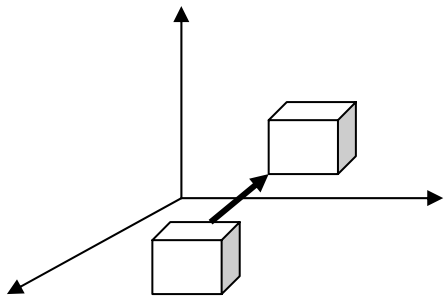
$$x' = x + t_x$$
$$y' = y + t_y$$
$$z' = z + t_z$$

Persamaan di atas, dapat pula ditampilkan dalam bentuk matriks translasi di bawah ini :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

atau dapat pula dituliskan dengan persamaan berikut :

$$A' = A + T$$



Gambar 2.20 Translasi pada Objek 3D

2. Skala

Skala dapat diartikan sebagai suatu perubahan terhadap objek tertentu yang mengakibatkan berubahnya ukuran objek secara keseluruhan. Operasi ini dilakukan dengan melakukan

perkalian terhadap koordinat titik A(x,y) dengan faktor skala s_x dan s_y untuk menghasilkan koordinat baru A'(x',y').

$$x' = x.s_x \ , \ y' = y.s_y$$

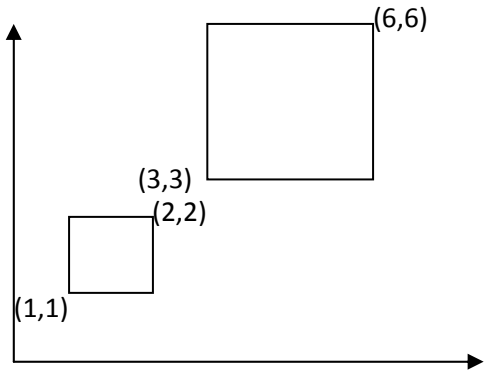
Faktor skala s_x merupakan skala object pada sumbu x, sementara s_y skala pada sumbu y. Persamaan di atas dapat juga dituliskan dalam bentuk matriks.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} . \begin{bmatrix} x \\ y \end{bmatrix}$$

atau

$$A' = S.A$$

Nilai faktor skala > 1 akan menyebabkan bidang menjadi lebih besar dan posisi titik menjadi s kali lebih jauh, sementara untuk nilai faktor skala <1 akan mengurangi ukuran objek. Nilai faktor skala = 0 akan menyebabkan objek hilang, meskipun sebenarnya objek tersebut berubah menjadi titik, yaitu (0,0).



Gambar 2.21 Skala Objek 2D dengan nilai skala 3 untuk x dan y

Pada objek 3D, maka persamaan skala yang digunakan adalah sebagai berikut :

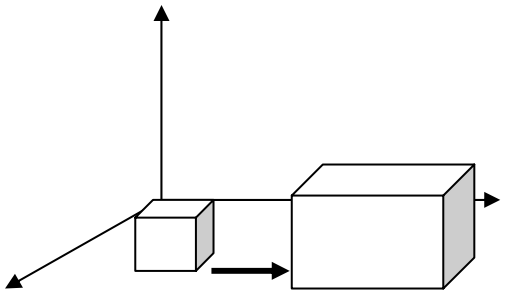
$$x' = x.s_x,\ y' = y.s_y,\ z' = z.s_z$$

Operasi matriks untuk skala pada bidang 3D di titik A(x,y,z) bisa ditulis sebagai berikut :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

atau

$$A' = S.A$$



Gambar 2.22 Skala pada Objek

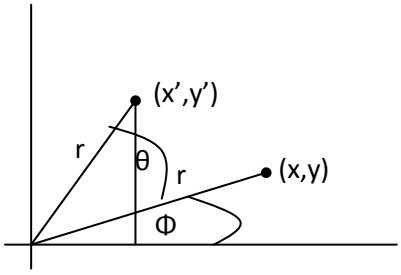
3. Rotasi

Rotasi adalah suatu operasi yang menyebabkan objek bergerak berputar pada titik pusat atau sumbu putar yang dipilih berdasarkan sudut putaran tertentu. Rotasi dilakukan dengan menambahkan besaran pada absis X dan koordinat Y. Rotasi 2 dimensi pada suatu objek akan memindahkan objek

tersebut menurut garis melingkar pada bidang xy. Untuk melakukan rotasi diperlukan suatu rotasi θ dan *pivot point* (x_r,y_r) atau rotasi *point* dimana objek ini dirotasi.

Nilai positif untuk sudut rotasi akan menentukan perputaran berlawanan dengan arah jarum jam terhadap *pivot point*, sedangkan nilai negatif akan menentukan perputaran sesuai dengan arah jarum jam.

Pertama kali yang harus dilakukan adalah dengan menentukan persamaan transformasi untuk rotasi dari titik A sementara *pivot point* adalah titik pusat (0,0). Hubungan antara sudut dan koordinat dari titik asli dan titik yang telah ditransformasi dapat dilihat pada gambar berikut.



Gambar 2.23 Hubungan antara sudut dan koordinat pada rotasi

Pada gambar di atas, r merupakan konstanta untuk jarak titik dari titik pusat, sudut Φ merupakan besarnya sudut titik awal dari sumbu mendatar, dan sudut θ merupakan sudut perputaran. Menggunakan trigonometri standar, kita dapat menentukan koordinat transformasi sebagai berikut :

$$x' = r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$x' = r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

Koordinat titik awal pada koordinat polar adalah

$$x = r\cos\phi, y = r\sin\phi$$

Dengan melakukan substitusi persamaan di atas, akan didapatkan persamaan transformasi untuk perputaran titik pada posisi (x,y) sebesar sudut θ.

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

Dengan representasi vektor, kita dapat menuliskan persamaan rotasi dalam bentuk matriks :

$$A' = R.A$$

dimana matriks rotasi adalah

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Rotasi pada objek 3D berbeda dengan objek 2D. Tidak seperti aplikasi pada 2D, dimana seluruh transformasi hanya berputar pada bidang xy seperti halnya kita melihat perputaran jam, pada 3D kita dapat menentukan perputaran pada setiap sumbu.

Untuk melakukan perputaran pada sumbu z, maka dihasilkan persamaan sebagai berikut :

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

Persamaan di atas dapat diubah ke dalam bentuk matriks sebagai berikut :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Atau dapat juga ditulis

$$A' = R_z(\theta).A$$

Hal yang sama juga dapat dilakukan untuk rotasi pada sumbu x dan sumbu y. Untuk perputaran pada sumbu x, persamaan yang didapat adalah :

$$\begin{aligned} x' &= x \\ y' &= y \cos\theta - z \sin\theta \\ z' &= y \sin\theta + z \cos\theta \end{aligned}$$

Apabila diubah ke dalam bentuk matriks akan menghasilkan :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Atau dapat juga ditulis

$$A' = R_x(\theta).A$$

Sedangkan perputaran pada sumbu y, persamaan yang didapat adalah :

$$\begin{aligned} x' &= z\sin\theta + x\cos\theta \\ y' &= y \\ z' &= z\cos\theta - x\sin\theta \end{aligned}$$

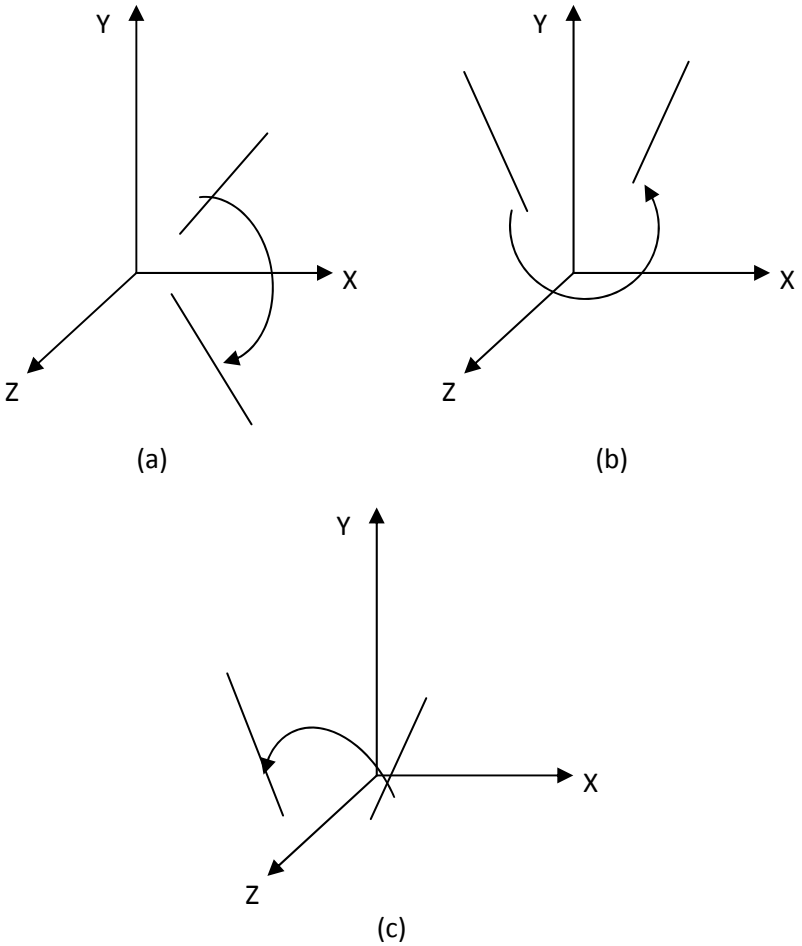
Bentuk matriks untuk persamaan di atas adalah :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Atau dalam bentuk sederhananya adalah :

$$A' = R_y(\theta).A$$

Gambar-gambar berikut ini menggambarkan mengenai perputaran pada masing-masing sumbu, yaitu x, y dan z.



Gambar 2.24 Rotasi Objek 3D (a). Rotasi pada sumbu x (b).Rotasi pada sumbu y (c). Rotasi pada sumbu z

2.2.3 HTML (*Hypertext Markup Language*)

HTML (*Hypertext Markup Language*) adalah dokumen yang mengatur bahasa-bahasa yang digunakan untuk mendesain kebanyakan halaman *web*. HTML adalah suatu sistem untuk *marking-up, tagging*, sehingga dokumen tersebut dapat dipublikasikan ke *web* (Conolly, 2005, p1001).

2.2.3.1 HTML 5

HTML 5 merupakan versi yang terakhir daripada HTML, dengan ditambahnya beberapa fitur baru seperti *canvas* yang digunakan untuk menampilkan gambar dan animasi, mendukung video dan animasi; dan juga adanya struktur elemen baru seperti *article*, *header, section* dan *footer* yang memberikan kemudahan dalam menempatkan format CSS (Meyer, 2010, p2).

2.2.3.2 HTML 5 *Canvas*

HTML 5 *Canvas* adalah area layar *bitmapped* yang dapat dimanipulasi secara langsung dengan menggunakan *Javascript*. HTML 5 *Canvas* memungkinkan untuk membentuk berbagai bentuk, teks, menampilkan gambar, menggunakan warna, melakukan rotasi, manipulasi piksel, dan berbagai jenis garis, kotak, kurva. Sehingga HTML5 *Canvas* dapat menciptakan animasi, aplikasi maupun *games* yang ditampilkan secara langsung dari jendela *browser* (Fulton, 2011, p1).

2.2.4 JavaScript

Javascript adalah bahasa yang digunakan untuk membuat program yang digunakan agar dokumen HTML yang ditampilkan dalam *browser* menjadi lebih interaktif, tidak sekedar indah saja (Betha Sidik, 2011, p1). Java Script memberikan beberapa fungsionalitas ke dalam halaman *web*, sehingga dapat menjadi sebuah program yang disajikan dengan menggunakan antarmuka *web*.

Dengan JavaScript, sebuah halaman web akan menjadi dinamis dan interaktif terhadap user karena halaman web mampu berfungsi sebagai sebuah program aplikasi yang dapat memproses masukan yang diberikan user dan memberikan hasil sesuai dengan yang telah diprogramkan.

2.2.5 *Document Object Model* (DOM) dan *Canvas*

Document Object Model (DOM) adalah sebuah antarmuka pemrograman aplikasi (API) untuk HTML serta XML. DOM memetakan seluruh halaman sebagai dokumen yang terdiri dari hierarki *node*. DOM menjelaskan metode dan *interfaces* (antarmuka) yang bekerja dengan isi konten dari halaman web (Zakas, 2005, p6).

Canvas merupakan media gambar bitmap berbentuk segi empat yang bisa digunakan untuk me-*render* grafis *game*, animasi, dan gambar visual lainnya di halaman *website* secara *client-side*. Canvas berbentuk segi empat tanpa garis tepi dan untuk memanfaatkannya bisa menggunakan JavaScript.

Elemen dalam *canvas* itu sendiri dapat diakses melalui DOM di *web browser* melalui *canvas* kontek 2D. Kita akan menggunakan DOM untuk menemukan tag <canvas> pada halaman HTML 5 dengan menggunakan manipulasi JavaScript. Ada dua spesifikasi objek DOM yang perlu kita pahami ketika menggunakan <canvas> yaitu : *window* dan *document. Window* diperlukan untuk menguji dan memastikan bahwa semua aset dan kode telah dimuat atau dijalankan sebelum aplikasi canvas dimulai. Sedangkan *document* itu berisi semua tag HTML yang ada di halaman HTML, yang diperlukan untuk melihat dan menemukan tag <canvas> itu sendiri yang telah dimanipulasikan dengan JavaScript.

2.2.6 Sistem Pembanding

Sistem pembanding yang dijadikan acuan dalam aplikasi yang kami buat antara lain: Java3D, OpenGL dan WebGL.

1. Java3D

   Java3D atau yang lebih sering disebut Java API 3D adalah sebuah antarmuka pemrograman aplikasi yang digunakan untuk membuat sebuah aplikasi tiga-dimensi menggunakan bahasa Java. Bahasa ini memberikan para programmer kontruksi tingkat tinggi untuk menciptakan dan memanipulasi geometri atau bidangtiga-dimensi. (http://graphcomp.com/info/specs/java3d/j3dguide/Intro.doc.html)

   - Kelebihan

     - Proses pembuatan bahasanya sudah berbasis OOP

     - Grafis yang ditampilkan sudah sangat bagus

- Mampu di *embed* dengan menggunakan video dan music

- Kekurangan
  - *File* yang dibuat hasilnya memiliki *file* yang besar
  - Membutuhkan spesifikasi komputer yang cukup tinggi
  - Hanya dapat dilihat di komputer pembuat object 3D / membutuhkan *file* yang tersimpan sehingga untuk pengiriman data untuk orang lain dapat menjadi penghambat.
  - Lambat, karena proses *render* yang melalui *virtual machine* java.

2. OpenGL

   OpenGL (*Open Graphics Library*) adalah bahasa yang paling banyak digunakan para pengembang aplikasi 2D dan 3D. Bahasa yang digunakan adalah bahasa C/C++ dimana OpenGL merupakan salah satu favorit dari bahasa yang digunakan dalam pembuatan game-game 3D. (http://www.khronos.org/opengl)

   - Kelebihan
     - Fleksibel. OpenGL dapat digunakan pada semua operating system.
     - Mampu menciptakan suatu grafis dengan *high-performance*.
     - Mampu mengekspos seluruh fitur hardware komputer grafis yang terbaru.
   - Kekurangan

- Hanya terbatas dalam untuk pembuatan aplikasi *game console* ataupun PC.
- Membutuhkan suatu spesifikasi *hardware* tinggi dalam memvisualisasikan di komputer.
- Masih terbatas oleh ruang lingkup *offline.*

3. WebGL

WebGL adalah *cross-platform*, bebas royalti *web* standar untuk grafis tingkat rendah API 3D berdasarkan OpenGL ES 2.0, yang terpapar melalui elemen HTML5 *canvas* sebagai antarmuka *Document Object Model*. Pengembang akrab dengan OpenGL ES 2.0 akan mengenali WebGL sebagai API Shader berbasis menggunakan GLSL, dengan konstruksi yang semantik mirip dengan OpenGL ES 2.0 API yang mendasari. Ini tetap sangat dekat dengan spesifikasi OpenGL ES 2.0, yang dikelola bahasa seperti JavaScript. WebGL membawa *plugin* 3D gratis ke *web*, diimplementasikan tepat ke *browser*. Mayor *vendor* yang mampu menampilkan aplikasi berbasis webGL, antara lain: *browser* Apple (Safari), Google (Chrome), Mozilla (Firefox), dan Opera (Opera). (http://www.khronos.org/webgl/)

- Kelebihan
  - Mampu ditampilkan di *website.*
  - Design menarik sama seperti openGL.
  - Mampu me*load file* dari 3D max.
- Kekurangan

- *Browser* banyak yang tidak *compatible* (*Cross Browser* tidak berjalan mulus).
- Sangat berat (butuh internet dengan koneksi tinggi).