**BAB II**

**LANDASAN TEORI**

**2.1 Pusat Massa**

Pusat massa merupakan posisi rata-rata dari sebuah atau sekumpulan objek berdasarkan massanya. Pusat massa sering disamakan dengan istilah pusat gravitasi. Pusat massa umumnya digunakan untuk menyederhanakan persamaan gerak seperti momentum sudut dan momen inersia [9]. Contoh sistem objek terdapat pada gambar 2.1.

Pusat massa dapat terletak di dalam maupun di luar objek. Posisi ini merupakan titik dimana benda terpengaruhi apabila benda tersebut diberikan gaya. Sebuah objek akan berpindah tanpa berotasi ketika diberi gaya tepat pada posisi pusat massa.

Kalkulasi letak pusat massa suatu sistem dalam suatu sumbu dapat ditentukan dengan melakukan pembagian antara jumlah hasil perkalian titik sumbu dan massa tiap benda dengan total massa [1, 2, 4] :

(2.1)

(2.2)

(2.3)

dimana :

= massa *segment*

= pusat massa *segment* pada sumbu X

= massa total benda

= pusat massa sistem pada sumbu X



**Gambar 2.1 Sistem objek dalam suatu diagram kartesian dua dimensi**

(sumber : <https://cdn.kastatic.org/ka-perseus-images/4cf588bbf9e241a4c6c8c13ab9c6eb582eca6f37.svg>)

**2.2 Pusat Massa Segment Tubuh Manusia**

R. A. Clark et al. mengatakan bahwa pusat massa *segment* umumnya dihitung dengan mencari persentase dari panjang *proximal end*, sedangkan massa *segment* umumnya dihitung dengan mencari persentase dari total massa manusia [3]. Hasil dari penelitian R. A. Clark et al. ditampilkan dalam tabel 2.1 dan tabel 2.2.

**Tabel 2.1 Persentase Massa Segment terhadap Massa Total**

|  |  |  |
| --- | --- | --- |
| Segment Mass Percents: | | |
| Segment | Males | Females |
| Head & Neck | 6.94 | 6.68 |
| Trunk | 43.46 | 42.58 |
| Upper Arm | 2.71 | 2.55 |
| Fore Arm | 1.62 | 1.38 |
| Hand | 0.61 | 0.56 |
| Thigh | 14.16 | 14.78 |
| Shank | 4.33 | 4.81 |
| Foot | 1.37 | 1.29 |

**Tabel 2.2 Persentase Panjang Segment terhadap Proximal End**

|  |  |  |  |
| --- | --- | --- | --- |
| Segment Length Percents: | | | |
| Segment | Males | Females | End Points |
| Head & Neck | 50.02 | 48.41 | Top Of Head –C7 |
| Trunk | 43.10 | 37.82 | MidS - MidH |
| Upper Arm | 57.72 | 57.54 | SJC - EJC |
| Fore Arm | 45.74 | 45.59 | EJC - WJC |
| Hand | 79.00 | 74.74 | WJC - MCPIII |
| Thigh | 40.95 | 36.12 | HJC - KJC |
| Shank | 43.95 | 43.52 | KJC - AJC |
| Foot | 44.15 | 40.14 | Heel - Toe |

**2.3 BioVision Hierarchy Data**

*File* BVH merupakan ekstensi *file* yang berisi *hierachical strcuture* dan *motion data*. Sebuah *file* BVH dibagi kedalam dua bagian, bagian pertama diawali dengan *header* “*HIERARCHY*”, sedangkan bagian kedua ditandai dengan *header* “*MOTION*” [7].

Bagian *hierarchy* mendefinisikan satu atau lebih *skeleton* secara rekursif. Setiap *skeleton* diawali dengan satu *root* yang merupakan akar dari *skeleton* diikuti dengan satu atau lebih *joint* yang merupakan tungkai. Setiap *joint* diakhiri dengan *end site* yang merupakan representasi panjang dari *joint* terakhir. Setiap bagian dibatasi dengan kurung kurawal. Contoh hierarki skeleton terdapat pada gambar 2.2.

Pendefinisian *root* dan *joint* diikuti dengan nama dan berisi data tentang *offset, channels,* dan *children*. *End Site* hanya memiliki *offset* karena tidak akan dilakukan transformasi. *Offset* merupakan jarak relatif suatu *joint* terhadap *parent*-nya. *Channels* merupakan urutan data berupa translasi kartesian dan rotasi *euler* yang akan didapatkan dari bagian *motion*.

Bagian *motion* berisi *Frames* (jumlah *frame* total), *Frame Time* (selisih waktu setiap *motion* data dalam satuan detik), diikuti dengan baris-baris *motion* *data*. Satu baris *motion data* merupakan isi dari *channels*. Setiap *motion data* dari kiri ke kanan di *mapping* kedalam *channels* dari atas ke bawah [6].



**Gambar 2.2 Contoh Hierarki Skeleton**

(sumber : http://mocappys.com/wp-content/uploads/2014/03/BaseJointNames.jpg)

**2.4 Transformasi Matrix**

Penentuan transformasi pada tiga dimensi memerlukan *matrix 4x4 homogenous coordinate*. Matriks ini mampu merepresentasikan translasi, rotasi, dilatasi dan proyeksi. Matriks T merupakan matriks transformasi translasi. Matriks R dibagi menjadi 3 (Rx, Ry, dan Rz) yang masing-masing merupakan matriks rotasi pada sumbu tersendiri. Matriks S merupakan matriks *scaling* / dilatasi [5].

Penelitian ini menggunakan *Column major matrices* dan *right handed coordinate system. Column major matrices* menganggap kolom pertama adalah transformasi pada sumbu X, kolom kedua adalah transformasi pada sumbu Y, kolom ketiga adalah transformasi pada sumbu Z, dan kolom keempat merupakan vektor translasi. Perkalian matriks pada *column major matrices* dilakukan dengan cara menempatkan matriks pertama pada posisi paling kanan sampai, kemudian diikuti dengan matriks-matriks selanjutkan ke arah kiri.

*Right handed coordinate system* merupakan cara menafsirkan kemana arah sumbu Z. Penentuan arah dapat menggunakan jempol kanan sebagai sumbu X, jari telunjuk kanan sebagai sumbu Y, maka jari tengah akan menunjukan arah positif dari sumbu Z. Perbedaan *left handed coordinate system* dan *right handed coordinate system* terletak pada arah positif sumbu Z yang masing-masing saling membelakangi seperti yang digambarkan pada gambar 2.3.



**Gambar 2.3 Left Handed Coordinates dan Right Handed Coordinates**

*(sumber :* [*https://upload.wikimedia*](https://upload.wikimedia)*.org/wikipedia/commons/thumb/b/b2/3D\_Cartesian\_ Coodinate\_ Handedness.jpg/220px-3D\_Cartesian\_Coodinate\_Handedness.jpg)*

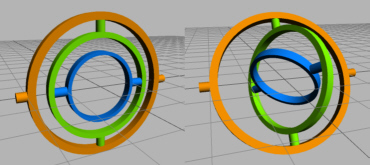
Perkalian pada matriks tidak bersifat komutatif (TRS ≠ SRT) urutan transformasi dilakukan dari dilatasi, kemudian diikuti oleh rotasi, dan diakhiri dengan translasi.

**2.5 Euler Rotate Order**

Transformasi tiga dimensi menggunakan matriks mementingkan order rotasi yang menjadi konvensi transformasi dalam suatu sistem. Konvensi ini disebut *euler rotate order*. *File* BVH yang digunakan dalam penulisan ini, *rotate order* yang digunakan adalah *ZXY* yang berarti transformasi rotasi dilakukan tiga kali, diawali dengan rotasi di sumbu Z, kemudian rotasi di sumbu X, dan diakhiri rotasi di sumbu Y.

*Rotate order* umumnya akan selalu konsisten untuk mencegah *gimbal lock*. *Gimbal lock* merupakan keadaan dimana salah satu sumbu putar sama dengan sumbu putar lainnya. Keadaan ini mengakibatkan dua sumbu putar menghasilan hasil perputaran yang sama sehingga transformasi rotasi tiga dimensi tidak dimungkinkan.

*Quaternions* dapat mencegah terjadinya *gimbal lock* dalam melakukan transformasi rotasi. Semua rotate order dalam penelitian ini telah didefinisikan dengan Rz, Rx, Ry, maka tidak perlu menggunakan *quaternions* untuk mendapatkan rotasi yang konsisten. Objek hanya dapat berputar pada satu sumbu apabila terjadi *gimbal lock* digambarkan pada gambar 2.4.



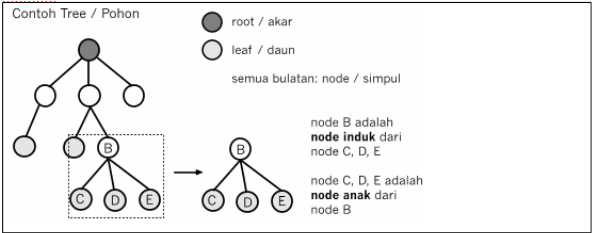
**Gambar 2.4 Gimbal Lock Sumber**

(sumber : https://i.stack.imgur.com/SRwlZ.jpg)

**2.6 Tree**

*Tree* adalah sebuah struktur data yang secara bentuk menyerupai sebuah pohon yang terdiri dari serangkaian *node* yang saling berhubungan. *Node-node* tersebut dihubungkan oleh sebuah vektor. Setiap *node* dapat memiliki nol atau lebih *node* anak (*child*). Sebuah *node* yang memiliki *node* anak disebut *node* induk (*parent*). Konvensi ilmu komputer menyatakan *tree* bertumbuh ke bawah dan tidak seperti pohon di dunia nyata yang bertumbuh ke atas, dengan demikian *node* anak akan digambarkan berada di bawah *node* induknya [10].

*Node* yang berada di pangkal tree disebut *node* *root* (akar), sedangkan *node* yang berada paling ujung pada piramida tree disebut *node* *leaf* (daun). Contoh ilustrasi *tree* dapat dilihat pada gambar 2.31.



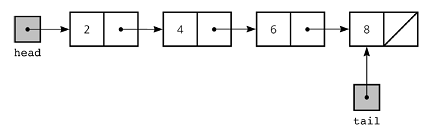
**Gambar 2.5 Struktur Data Tree**

(sumber : https://www.cs.cmu.edu/~sleator/papers/dynamic-trees.pdf)

**2.7 Linked List**

*Linked list* merupakan sebuah struktur data *linear* dimana setiap elemen juga merupakan objek pemisah. Sebuah *linked list* umumnya memiliki satu *head* yang merupakan awal dan satu atau lebih *tail* yang merupakan akhir. Setiap objek dalam *linked list* berisi data dan sebuah *pointer* ke objek lainnya. Sebuah *linked* *list* berakhir ketika mencapai *tail* dimana *pointer* pada objek bernilai *null*.

Relasi *parent-child* dalam sebuah *hierachial skeleton* umumnya dibuat dalam bentuk *linked list*. *Root*  dapat merepresentasikan *head* dan *end site* dapat merepresentasikan *tail*. *Linked list* merupakan struktur *data* yang digunakan dalam menyimpan, memproses, dan menggambarkan *skeleton* [6].



**Gambar 2.6 Contoh Linked List**

(sumber : [https://s3-us-west-2.amazonaws.com/ib-assessment tests/problem\_images/singly-ll.png](https://s3-us-west-2.amazonaws.com/ib-assessment%20tests/problem_images/singly-ll.png))

**2.8 Motion Capture**

*Motion capture* adalah proses perekaman informasi gerak dari suatu objek dari *interval* waktu tertentu dan diubah menjadi data numerik. Tujuan digunakannya *motion capture* adalah untuk mendapatkan data gerakan dari makluk hidup sehingga dapat disimulasikan ulang untuk keperluan animasi.

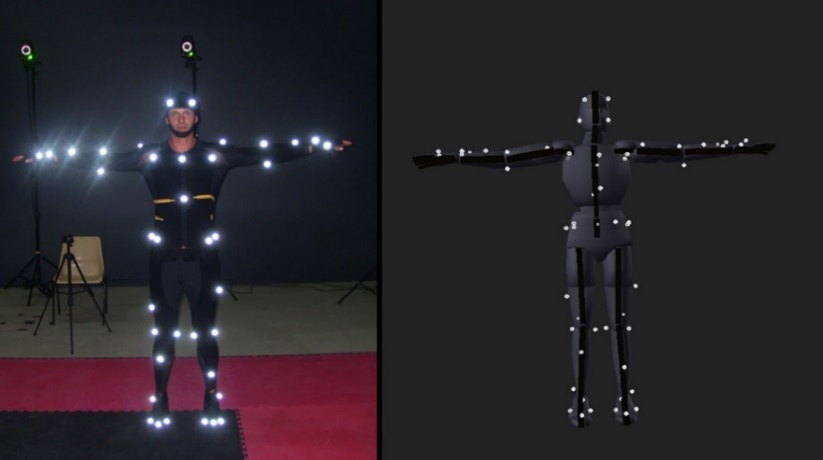
Aplikasi *motion capture* sangat bervariasi dari waktu ke waktu sesuai dengan perkembangan komputer. Alat *motion capture* dapat dikelompokkan menjadi dua bagian. Pengelompokkan ini dapat didasarkan kepada alat perekam dan jenis objek yang direkam. *Motion capture* terbagi menjadi tiga jenis yaitu optik, magnetik dan mekanik, sementara jenis objek yang direkam dikategorikan menjadi dua bagian yaitu *marker* dan *markerless* [11].

**2.8.1 Motion Capture Optik**

*Optical Motion Capture* adalah bidang penting dalam komputer *vision*, dan banyak digunakan dalam komputer grafis serta diteliti dalam banyak penelitian. Pentingnya *Optical Motion Capture* sebagian besar disebabkan oleh masalah-masalah yang relevan yang terlibat dalam proses dan untuk berbagai aplikasi untuk data gerak nyata. Gerakan realistis (nyata) diperlukan untuk melakukan sintesis dan analisa gerakan manusia [11].

*Optical Motion Capture* menggunakan kamera untuk merekonstruksi postur tubuh. Salah satu pendekatan memperkerjakan satu set kamera untuk menangkap beberapa *marker* yang disinkronisasi pada tubuh manusia. Sebuah sistem penangkap gerakan memiliki aplikasi dalam komputer grafis untuk animasi karakter, dalam *virtual reality* untuk manusia kontrol antar muka, dalam *video game* untuk simulasi realistis dari gerak manusia. Jenis *optical motion capture* terbagi menjadi dua yaitu *passive maerker* dan *active marker*.

Sistem *motion capture* dengan *passive marker* menggunakan *marker-marker* yang dilingkupi dengan bahan reflektif untuk memantulkan cahaya yang dihasilkan oleh lampu pada samping kamera. Ambang *threshold* dari kamera dapat disesuaikan sehingga hanya cahaya dari pantulan marker saja yang diambil oleh kamera dengan mengabaikan benda lain dengan menggunakan bantuan perangkat lunak yang digunakan untuk mengidentifikasi posisi-posisi relatifnya.



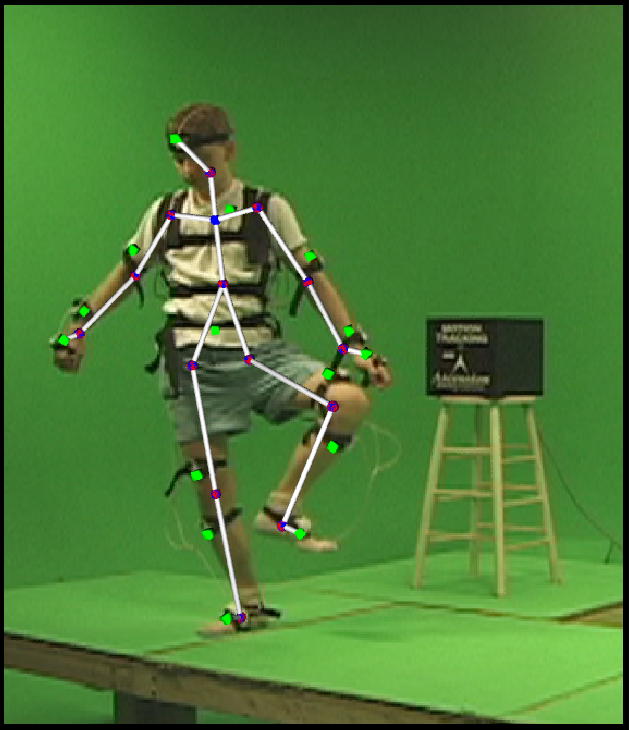
**Gambar 2.7 Contoh dari penggunaan marker aktif**

(sumber : https://andrewbedforddotcom1.files.wordpress.com/2014/

10/marker-placement1.jpg)

**2.8.2 Motion Capture Magnetik**

Sistem magnetik menggunakan sensor elektromagnetik terhubung ke komputer yang dapat menghasilkan data tiga dimensi secara *realtime* dengan biaya pengolahan yang rendah. Sistem *magnetic motion capture* digunakan untuk mendapatkan informasi tentang *joint parameters* dari *articulated hierachy*. Teknik ini memungkinkan untuk menentukan panjang tungkai atau lengan, lokasi-lokasi penghubung (*joint locations*) dan penempatan sejumlah sensor untuk manusia tanpa pengukuran eksternal.

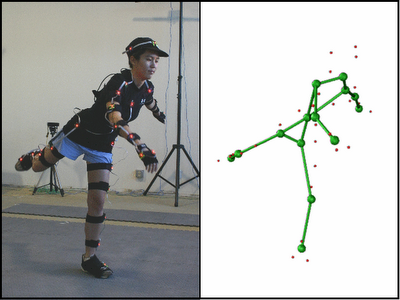


**Gambar 2.8 Contoh Motion Capture Magnetik**

(sumber : https://www.semanticscholar.org/paper/Automatic-Joint-Parameter-Estimation-from-Magnetic-O%27Brien-Bodenheimer/139b753e65a208c84b1ef0ce55a1f6871b6ad597/figure/0)

**2.8.3 Motion Capture Mekanik**

Sistem mekanis menggunakan pakaian khusus dengan mekanik terintegrasi dengan sensor yang mendaftarkan gerak artikulasi secara realtime dan tidak diperlukan pengolahan. Sistem tersebut terdapat *potensiometer* untuk mengukur rotasi sendi dan diketahui panjang dan penghubung (*links*) yang kaku (*rigid*). Gerakan manusia akan terekam dalam sistem mekanik yang dirancang menyelimuti seluruh tubuh.



**Gambar 2.9 Contoh Motion Capture Mekanik**

(sumber : https://kalyankrishna4886.files.wordpress.com/2010/11/optical\_motion\_capture.png)

**2.9 OpenGL**

OpenGL adalah *cross-language*, *multi-paltform* antarmuka pemrograman aplikasi untuk rendering dua dimensi dan tiga dimensi. API biasanya digunakan untuk berinteraksi dengan *graphics pocessing unit* (GPU), untuk mencapai render *hardware-accelerated*. OpenGL dikembangkan oleh Silicon Graphics Inc (SGI) dari tahun 1991 dan dirilis pada bulan januari tahun 1992 dan secara luas digunakan dalam CAD, *virtual reality*, visualisasi ilmiah, visualisasi informasi, simulasi penerbangan, dan *video game*. OpenGL dikelola oleh teknologi non-profit Khronos Group.

OpenGL adalah murni berkaitan dengan rendering yang berarti OpenGL tidak menyediakan API yang berkaitan dengan *input*, *audio*, atau *windowing*. GPU dapat menyediakan fungsionalitas tambahan dalam bentuk ekstensi. Ekstensi dapat memperkenalkan fungsi baru dan konstanta baru. *Vendor* dapat menggunakan ekstensi untuk mengekspos API kustom tanpa perlu dukungan dari vendor lain atau Khornos Grup secara keseluruhan yang dapat meningkatkan fleksibilitas OpenGL.

**2.10 C++**

Bahasa pemrograman C++ adalah bahasa pemrograman komputer yang dibuat oleh Bjarne Stroustrup dimana bahasa pemrograman ini merupakan pengembangan dari bahasa C yang dikembangkan di Bell Labs oleh Dennis Ritchie pada awal tahun 1970-an sebagai pendahulunya. Bahasa tersebut dirancang sebagai bahasa pemrograman yang dijalankan pada sistem Unix.

Bahasa pemrograman C++ menambahkan konsep-konsep objek seperti *class* dengan sifat-sifatnya seperti *inheritance* dan *overloading*. Salah satu perbedaan yang paling mendasar dengan bahasa C adalah dukungan terhadap konsep pemrograman beroritasi objek (*Object Oriented Programing*) seperti halnya bahasa pemrograman java. Semua *library* yang ada pada bahasa C telah tersedia dalam bahasa C++.

Bahasa ini termasuk dalam bahasa pemrogaman level tengah (middle level) yang berarti merupakan campuran elemen bahasa pemrograman level tinggi (*highest level*) merupakan bahasa pemrograman yang besifat *case sensitive* yang artinya penulisan dengan menggunakan huruf besar dan huruf kecil diartikan berbeda oleh compiler.

**2.11 Flowchart Diagram**

*Flowchart Diagram* adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam program atau prosedur sistem secara logika. Bagan alir (*flowchart*) digunakam terutama untuk alat bantu komunikasi dan untuk dokumentasi.

*Flowchart* terbagi menjadi beberapa jenis diantaranya: Bagan alir sistem (*system flowchart*), Bagan alir dokumen (*document flowchart*), Bagan alir skematik (*schematic flowchart*), Bagan alir program (*program flowchart*), dan Bagan alir proses (*process flowchart*). Simbol-simbol pada *Flowchart* *Diagram* dapat dilihat pada tabel 2.3.

**Tabel 2.3 Simbol-simbol Flowchart Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Gambar | Nama | Keterangan |
| 1. |  | Flow Direction | Simbol untuk menghubungkan antara simbol yang satu dengan simbol yang lain. |
| 2. |  | Terminator | Simbol untuk permulaan (*start*) atau akhir (*stop*) dari suatu kegiatan. |
| 3. |  | Connector | Simbol untuk keluar-masuk atau penyambungan proses dalam lembar/halaman yang sama. |
| 4. |  | Manual Operation | Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh komputer. |
| 5. |  | Processing | Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer. |
| 6. |  | Decision | Simbol pemilihan proses berdasarkan kondisi yang ada. |
| 7. |  | Input-Output | Simbol yang menyatakan proses *input* dan *output* tanpa tergantung dengan jenis peralatannya. |
| 8. |  | Manual Input | Simbol untuk pemasukan data secara *manual*. |
| 9. |  | Preparation | Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam *storage*. |
| 10. |  | Online Storage | Simbol yang menyatakan inputberasal dari diskatau disimpan ke *disk*. |

**2.12 Unified Modeling Language**

*Unified Modeling Language* (UML) merupakan bahasa pemodelan standar yang memiliki sintaks dan semantik. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. UML diaplikasikan dengan beberapa maksud seperti percangan perangkat lunak, sarana komunikasi antara perangkat lunak dengan proses bisnis, penjabaran sistem secara rinci dan pendokumentasian sistem yang ada.

Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para pengembang aplikasi untuk bekerja sama dengan bahasa model yang sama dalam mengaplikasikan beragam sistem. UML merupakan Alat komunikasi yang konsisten dalam mendukung para pengembang saat ini. UML menyediakan sembilan jenis diagram yaitu *: Class Diagram, Packet Diagram, Use Case Diagram, Sequence Diagram, Communication Diagram, Statechart Diagram, Activity Diagram, Component Diagram* dan *Deployment Diagram.*

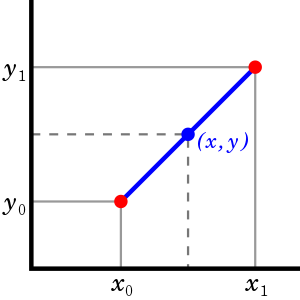
*Activity Diagram* merupakan penggabungan dari berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir*. Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

**Tabel 2.4 Simbol-simbol Activity Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Gambar | Nama | Keterangan |
| 1. |  | Activity | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain. |
| 2. |  | Action | *State* dari sitem yang mencerminkan eksekusi dari suatu aksi. |
| 3. |  | Initial | Bagaimana objek dibentuk atau diawali. |
| 4. |  | Final | Bagaimana objek dibentuk dan dihancurkan |
| 5. |  | Fork | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran. |

**2.12 Interpolasi Linier**

Interpolasi linier merupakan metode numerik peramalan dimana nilai suatu angka diapit oleh dua angka numerik lainnya. Interpolasi linier menghasilkan angka sisipan berdasarkan suatu derajat *alpha*. Dimisalkan x adalah angka pertama dan y adalah angka kedua dan nilai *alpha* (a) adalah 0.6, maka hasil dari interpolasi linier akan menghasilkan angka baru yang berada 60% dari x ke y [12]. Contoh ilustrasi interpolasi linier ada pada gambar 2.10.



**Gambar 2.10 Ilustrasi Interpolasi Linier**

(sumber : https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/LinearInterpolation.svg/300px-LinearInterpolation.svg.png)