BAB 2
LANDASAN TEORI

2.1 Sejarah *Game*

Generasi *game* pertama muncul dari ATARI 2600 merupakan konsole *game* pertama yang sukses di mansanya atari 2600 ini dirilis pada oktober 1977, setelah itu dikenal dengan nama VCS (*Video Computer System*). Setelah sukses pada generasi pertama atari tidak menyerah sampai disitu saja untuk memajukan dunia *game* elektronik pada generasi kedua muncullah atari 7800 konsol ini dirilis pada juni 1986 pada atari 7800 ini ada sedikit kemajuan dengan menambahkan *joystik* sehingga user dapat lebih mudah untuk memainkan konsole *game* ini namun pada masanya harga dari konsole *game* ini selangit yaitu $140 pada jamannya mungkin ini menjadi salah satu permainan termahal.[3]

Setelah Atari 7800 ada NES *Nintendo entertaintment System* dan inilah konsole permainan pertama kali yang menggunakan 8 bit. *Nintendo* ini menghasilkan produk produk *game* yang lain daripada yang lain salah satu *game* yang *compatible* dengan NES ini adalah super mario bros. setelah NES laris manis dengan super marionya lalu muncul lagi konsole *game* sega mega *drive*. Se*game*ga *drive* ini merupakan generasi ke tiga dari dunia *game* sega menggunakan 16 bit dan dirilis pada tahun 1988, kalau tadi *nintendo* dikenal dengan super mario brossnya sega juga tidak mau kalah dengan mengeluarkan *gamesonic the hedhog*.[3]

Pada generasi ke empat dari sejarah *game* produksi *game* semakin menjamur dan berkembang pesat salah satu contoh pada generasi ke empat ini lahirlah *play station* merupakan salah satu *game* konsole terlaris dan terbaik di indonesia bahkan di dunia merupakan terobosan baru di dunia *game* dengan menggunakan 32 bit generasi kelima *play station* atau yang lebih akrab sebut *Play Station* (PS) semakin gencar melakukan produksi lainnya yang lebih mutakhir dengan mengeluarkan konsole *game* PS2 namun pada generasi ke lima ini sangat banyak sekali saingan saingan PS2 ,XBox, sega *Saturn*, *Dreamcast*, dari generasi

kelima ini sudah mulai banyak perusahaan perusahaan yang memproduksi konsole *game.*[3]

Generasi sekarang, pada tahun ini sudah banyak sekali konsole *game* menjamur di dunia PS juga masih terus mengembangkan sistemnya dengan mengeluarkan PS3, Xbox juga mengeluarkan XBOX 360, lalu *nintendo* juga mengeluarkan WII. Dari generasi pertama sampai generasi sekarang perkembangan *game* sangat cepat sekali berkembang, bahkan sekarang lagi dikembangkan *game* yang menggunakan kacamata khusus sehingga *user* dapat terhisap masuk kedalamnya.[3]

2.2 Pengertian *Game*

Game berasal dari kata bahasa inggris yang memiliki arti dasar Permainan. Permainan dalam hal ini merujuk pada pengertian "kelincahan intelektual" (*intellectual playability*). *Game* juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Ada target-target yang ingin dicapai pemainnya.[6]

Kelincahan intelektual, pada tingkat tertentu, merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal. Pada awalnya, *game* identik dengan permainan anak-anak. Seringkali berpikir *game* merupakan suatu kegiatan yang dilakukan oleh anak-anak yang dapat menyenangkan hati mereka. Dengan kata lain, segala bentuk kegiatan yang memerlukan pemikiran, kelincahan intelektual dan pencapaian terhadap target tertentu dapat dikatakan sebagai *game*. Tetapi yang akan dibahas pada kesempatan ini adalah *game* yang terdapat di komputer, baik *off line* maupun *online.* [6]

Saat ini perkembangan *game*s di komputer sangat cepat. Para pengelola industri *game* berlomba-lomba untuk menciptakan *game* yang lebih nyata dan menarik untuk para pemainnya. Hal inilah yang membuat perkembangan*games* di komputer sangat cepat. Sehingga *game*s bukan hanya sekedar permainan untuk mengisi waktu luang atau sekedar hobi. Melainkan sebuah cara untuk meningkatkan kreatifitas dan tingkat intelektual para penggunanya.Jadi, bermain *game* adalah suatu proses "*fine tuning"* (atau penyamaan frekuensi) dari logika berpikir anak-anak kita dengan logika berpikir aplikasi komputer yang canggih tadi. Pada saat bersamaan, *game* juga secara nyata mempertajam daya analisis

para penggunanya untuk mengolah informasi dan mengambil keputusan cepat yang jitu. Namun, tentu saja kenyataan juga harus kita masukkan kedalam perhitungan. Kenyataan itu diantaranya adalah kecanduan para pemain penggunanya yang takut terhadap permainan komputer semacam ini. Mereka bisa lupa segala-galanya akan tugas mereka yang lain termasuk tugas menuntut ilmu.

2.3 *Game* 3 Dimensi

Game 3 dimensi adalah lingkungan yang menggunakan parameter atau pengukuran yang dibutuhkan untuk mendefinisikan sifat-sifat suatu objek yaitu panjang, lebar, dan tinggi atau bisa juga ukuran dan bentuk. dalam mematika dan fisika, 3 dimensi adalah parameter yang dibutuhkan untuk mnggambarkan posisi dan sifat-sifat obyek dalam suatu ruang. dalam konteks khusus, satuan ukuran dapat pula disebut'dimensi' atau 'matra', meter atau inchi dalam model geografis atau biaya dan harga dalam model ekonomi.

2.4 Jenis-jenis *game*

Berdasarkan jenisnya *game* dibagi menjadi beberapa kategori diantaranya :

a. Berdasarkan platfrom atau alat yang digunakan : [8]
    1. *Arcadegames*, yaitu video *game* dengan mesin yang memang khusus di desain untuk jenis video*games* tertentu dan memiliki fitur seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil dan lain-lain.
    2. PC *Games* , yaitu video *game* yang dimainkan menggunakan Personal Computers.
    3. *Console games*, yaitu video *game*s yang dimainkan menggunakan *console* tertentu, seperti *Playstation* 2, *Playstation* 3, XBOX 360, dan *Nintendo* Wii.
    4. *Handheld games*, yaitu yang dimainkan di console khusus video *game* yang dapat dibawa kemana-mana, contoh *Nintendo* DS dan *Sony* PSP.
    5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA.

b. Berdasarkan genre permainanya : [8]

1. Shooting

Shooting merupakan salah satu *genre* dalam *game* biasanya mensimulasikan tembak-tembakan, atau pukul-pukulan bisa juga tusuk-tusukan, tergantung cerita dan tokoh di dalamnya, video *game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga *timing*, inti dari *game* jenis ini adalah tembak.

2. Fighting

Fighting (pertarungan) jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan skill karakter yang di mainkan, pengenalan karakter dan timing sangatlah penting, dan berbeda seperti *game* Aksi pada umumnya yang umumnya hanya melawan *Artificial Intellegence* atau melawan komputer saja, pemain jenis fighting *game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya.

3. Petualangan

Refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. Video *Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat. termasuk didalamnya petualangan dengan teks atau sistem tunjuk dan klik.

4. Simulasi, Konstruksi dan manajemen

Video *Game* jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun *laboratorium cloning*. Video *Game* jenis ini membuat

pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas.

5. Role Playing

Video *game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dan sebagai) dalam berbagai parameter yang biasanya ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter dan lain-lain.

6. Strategi

Kebalikan dari video *game* jenis action yang berjalan cepat dan perlu refleks secepat kilat, video *game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. Video *game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusal pelatihan tempur, tergantung dari tema ceritanya. Pemain *game* strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan *game action*. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, peletakan pasukan, mencari dan memanfaatkan sumbe rdaya (uang, besi, kayu,minyak,dan lain-lain), hingga ke pembelian dan peng-*upgrade*-an pasukan atau teknologi.

7. Olahraga

Biasanya *game genre* ini permainannya diusahakan serealistik mungkin di buat semirip mungkin dengan aslinya walau kadang ada yang menambah unsur fiksi dalam *game* olahraga tersebut.

8. Edukasi

Video *Game* jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, baik itu untuk belajr mengenal warna, mengenal huruf dan angka, matematika, belajar bahasa asing dan lain-lain. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan *design visual* ataupun animasinya.

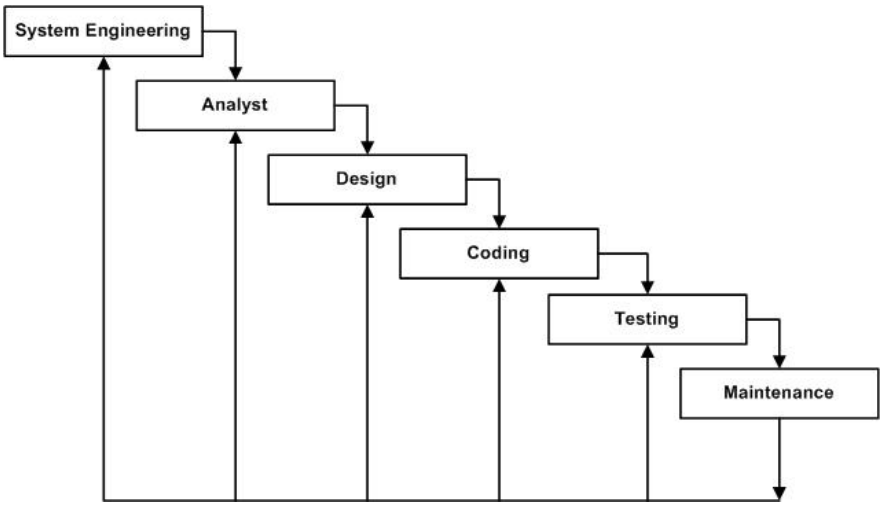
c. Berdasarkan jenis tampilan permainannya : [8]

1. *First Person Shooter (FPS),* sesuai judulnya *game* ini mengambil pandangan orang pertama pda *game*nya sehingga seolah-olah kita sendiri yang berada dalam *game* tersebut, kebanyakan *game* ini mengambil setting peperangan *game* dengan senjata-senjata militer (di indonesia, *game* jenis ini sering di sebut *game* tembak-tembakan). Contoh *game* genre ini antara lain Duke Nukem 3D, Quake, Blood, Unreal Tournament, Half-Life, Counter-Strike, Halo, Perfect Dark, TimeSpliters, Call of Duty, System Shock, dan GoldenEye 007.
2. *Real-Time Strategi (RTS),* merupakan *game* yang pemainnya menekankan kepada kehebatan strategi pemainya, biasanya pemain memainkan tidak hanya 1 karakter saja akan tetapi banyak karakter. Contoh Age of Empires, Rise of Nations, Stronghold, Warcraft.
3. *Cross-Platform Online,* merupakan *game* yang dapat dimainkan secara online dengan hardware yang berbeda misalnya saja Need For speed Undercover dapat dimainkan secara online dari PC maupun Xbox 360 (Xbox 360 merupakan hardware atau console *game* yang memiliki konektivitas ke internet sehingga dapat bermain secara online).
4. *Browser Games,* Merupakan *game* yang dimainkan pada browser seperti Firefox, Opera, IE. Syarat dimana sebuah browser dapat memainkan *game* ini adalah browser sudah mendukung javascript, php, maupun flash.

5. *Massive Multiplayer Online games Role Playing Game (MMORPG),* adalah *game* dimana pemain bermain dalam duina yang skalanya besar (>100 pemain), setiap pemain dapat berinteraksi langsung seperti halnya di dunia nyata. Contoh dari genre peermainan ini World of Warcraft, The Lord of the Rings Online: Shadows of Angmar, Final Fantasy, Ragnarok, Dotta (Lindsay Grace, 2005).

2.5 Metode Waterfall

Metode yang akan digunakan pada kasus ini adalah model *Waterfall*. Model ini mengusulkan sebuah pendekatan kepada perkembangan *software* yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan.[4] Dimodelkan setelah siklus rekayasa konvensional, model sekuensial linier melingkupi aktivitas – aktivitas yang dapat dilihat pada gambar dibawah ini :



Gambar 2-1Siklus *Metode Waterfall* [4]

1. Rekayasa dan Pemodelan (*SystemEngginering)*

   Rekayasa perangkat lunak merupakan tahapan yang pertama kali dilakukan untuk merumuskan sistem yang akan dibangun. Hal ini bertujuan untuk memahami aplikasi *game* yang akan dibangun.

2. Analisis Perangkat Lunak (*Software Analysis*)

   Analisis dilakukan terhadap permasalahan yang dihadapi serta untuk menetapkan kebutuhan perangkat lunak dari aplikasi yang dibangun.

3. Perancangan perangkat Lunak (*Software Design*)

   Tahap desain merupakan tahap penerjemahan dari data yang telah dianalisis ke dalam bentuk yang mudah dimengerti oleh pengguna.

4. Implementasi perangkat lunak (*Coding*)

   Coding merupakan tahap penerjemahan data yang telah dirancang ke dalam bahasa pemrograman tertentu.

5. Pengujian perangkat lunak (*Testing*)

   Tahap pengujian dilakukan terhadap perangkat lunak yang telah dibangun. Proses pengujian berfokus pada logika internal perangkat lunak serta memastikan apakah hasil yang diinginkan tercapai atau tidak.

6. Pemeliharaan (*Maintenance*)

   Maintenance merupakan penanganan dari suatu perangkat lunak yang telah selesai dibangun sehingga dapat dilakukan perubahan-perubahan atau penambahan sesuai dengan permintaan pengguna.

2.6 Metode *potential field*

Konsep dasar metode *Potential Field* digambarkan seperti partikel besi yang begerak menuju objek melalui medan magnet yang dibuat oleh objek yang dituju. Pergerakan initergantung dari medan magnet yang ada, partikel akan ditarik kearah tujuan atau justru sebaliknya partikel besi tersebut akan ditolak oleh medan magnet pada saat bertemu halangan.[2]

2.6.1 Cara Kerja *Potential Fields*

Sistem kerja *Potential Field* adalah dengan memberi nilai yang tinggi pada daerah dekat target dan nilai akan semakin berkurang jika jaraknya semakin jauh dari target.

Setiap *Potential Field* mempunyai radius aksi. Diluar dari radius maka tidak ada kekuatan yang akan menarik agen untuk menuju target, hal ini mirip dengan medan magnet. Pada saat agen berada pada jarak pengaruh yang dibuat oleh target, maka agen akan mendapat vektor ruang yang disebut *action* vektor yang merepresentasikan kecepatan dan tujuan dari agen tersebut.

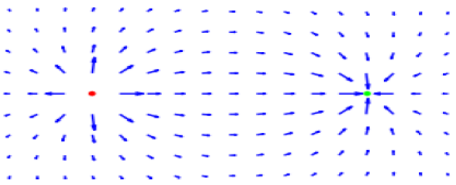
2.6.2 Metode *Potential field*

Konsep dasar metode *potential field* digambarkan seperti partikel besi yang bergerak menuju objek melalui medan magnet yang dibuat oleh objek yang dituju. Pergerakan ini tergantung dari medan magnet yang ada, yaitu partikel akan ditarik ke arah tujuan atau justru sebaliknya partikel tersebut akan ditolak oleh medan magnet pada saat bertemu halangan.

Setiap *potential field* dapat dirancang tersendiri untuk digunakan dalam menghadirkan suatu perilaku khusus.Dengan mengkombinasikan beberapa *potential field* yang ada maka pergerakannya dapat mencapai tingkat perilaku yang beragam.

Potential field total (Uart) = *Potential fieldAtraktif* (U_x) + *Potential fieldRepulsif*(Uo)

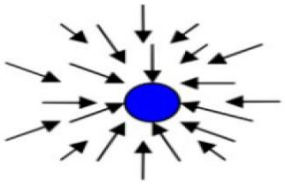
Akibat dari penggabungan ini adalah nilai *potential field* yang didapat pada daerah tujuan merupakan nilai paling tinggi dari sekitarnya sehingga partikel akan bergerak menuju tujuan. Sedangkan pada halangan nilai-nilainya adalah lebih rendah, dengan tujuan agar partikel dapat menghindari halangan (Gambar dibawah).



Gambar 2-2 *Potential field* gabungan

1. *Potential field Aktraktif*

*Potential field*atraktif adalah *potential field* yang mengatur bagaimana setiap agen yang ada bergerak mengarah ke tujuan, seperti yang diperlihatkan dalam Gambar dibawah.



Gambar 2-3 *potential field* aktraktif

Perhitungan nilai *potential field* tujuan, didapatkan dari konsep *potential field* dari elektrostatika, yaitu dengan menggunakan Persamaan (1).

Persamaan 1

$$U_{x_d}(x) = \frac{1}{2} k_p (x - x_d)^2 \quad \text{.....(1)}$$

Dimana :

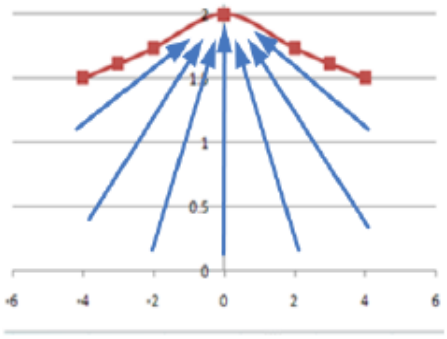
Uxd = *Potential field* atraktif yang dibuat oleh nilai tujuan

X = Posisi sebuah *agent*

Xd = goal position

Kp = Konstanta pada *potential field* atraktif

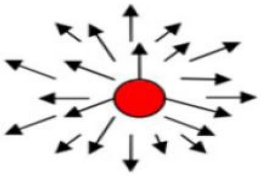
Persamaan (1) dapat digunakan untuk menentukan nilai *potential field* aktraktif dengan memanfaatkan puncak gundukan. Dalam artian, semua partikel yang berada di bawah nilai puncak gundukan tersebut akan tertarik ke puncak gundukan (yang mempunyai nilai tertinggi). Jadi dapat diibaratkan puncak gundukan adalah target yang akan dituju. Ilustrasi ini seperti yang diperlihatkan dalam Gambar dibawah .



Gambar 2-4 Implementasi grafik Eksponensial untuk *Potential field* Aktraktif.

2. *Potential fieldRepulsif*

Potential fieldrepulsif adalah *potential field* yang mengatur bagaimana setiap agen dapat menghindari halangan (*obstacle*) yang ada, seperti yang diperlihatkan dalam Gambar dibawah.



Gambar 2-5 *potential field* repulsif

Perhitungan nilai *potential field refulsif*, didapatkan dari konsep *potential field* dari elektrostatika, dengan menggunakan Persamaan (2).

$$U_o(\boldsymbol{x})=\begin{cases}\frac{1}{2}\eta\left(\frac{1}{\rho}-\frac{1}{\rho_0}\right)^2 & if\ \rho\le\rho_0,\\ 0 & if\ \rho>\rho_0\end{cases} \quad \text{.....(2)}$$

Dimana:

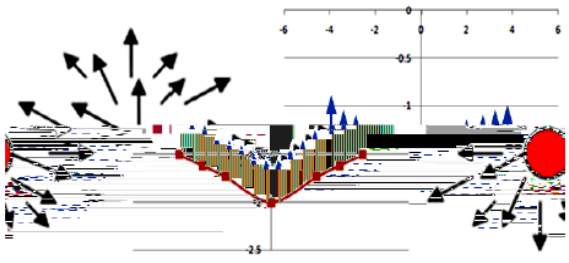
$U_o(x)$ = *Repulsive potential field created by the obstacle*

eta(η) = Konstanta pada *potential field* repulsif /*constant gain*

ρ = *shortest distance to the obstacle*

ρ_o = *limit distance of the potential field influence*

Persamaan (2) jika dimanfaatkan untuk menghitung nilai *potential repulsif*, maka partikel tidak akan tertarik ke puncak gundukan. Namun sebaliknya, partikel akan menolak dan berlari menuju nilai yang lebih tinggi. Jadi dapat diibaratkan garis yang membentuk fungsi eksponensial itu, harus dijauhi atau dalam artian halangan harus dihindari seperti yang diperlihatkan dalam Gambar dibawah.



Gambar 2-6 Implementasi Fungsi Eksponensialuntuk *Potential field* Aktraktif.

2.6.3 Action Vector

Cara untuk menggambarkan *Potential Field* adalah berfikir bahwa Potential Field adalah pemetaan dari sebuah vektor (agen) menuju vektor yang lain (*goal*) vektor yang dihasilkan tersebut disebut action vektor. *Action* vektor ini adalah komponen yang paling penting dalam pendekatan *Potential Field*. Karena hal ini akan menentukan pergerakan agen.

Action vektor terdiri dari sebuah panjang vektor dan arah.

1. Kekuatan adalah panjang vektor
2. Arah adalah arah vector dalam area.

Kekuatan dari vektor tergantung dari banyaknya gaya yang digunakan *Potential Field* untuk mendorong agen. Jumlah tersebut berubah tergantung dari posisi agen, apakah berada di dalam atau di luar. Jika di dalam radius *Potential Field* maka akan ada dua nilai yang menentukan kekuatannya. Pertama jarak antara agen dengan tujuan, kedua seberapa penting manfaat target bagi agen jika terdapat multi target. Arah akan tergantung dari tipe *Potential Field* yang dihadapi oleh agen.

Vektor-vektor ini merepresentasikan gaya dan hal ini tergantung apakah target merupakan tujuan agen yang akan mengarahkan agen menuju tujuan atau atraktif atau merupakan objek yang harus dihindari oleh agen yang membuat agen menjauh dari hambatan repulsif.

2.7 Perangkat Lunak Pendukung

Pada bagian ini akan dijelaskan tentang beberapa perangkat lunak dan bahasapemrograman yang digunakan untuk mendukung dalam pengembangan aplikasi yangdibuat. Bahasa pemrograman yang digunakan diantaranya adalah *Java* dan C#. Selainitu terdapat perangkat lunak yang digunakan yaitu *Unity game* 3D.

2.7.1 OOP (Object Oriented Programming)

OOP (*Object Oriented Programming*) atau yang dikenal dengan Pemrograman Berorientasi Objek merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah kita tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Pemrograman orientasi-objek menekankan konsep-konsep berikut: [9]

1. Kelas (*Class*)

Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh 'class of dog' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

2. Objek (*Object*)

Membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

3. Abstraksi (*Abstract*)

Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

4. Enkapsulasi (*Encapsulation*)

Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. Polimorfisme (*Polimorfism*)

Melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan, metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim

6. Inheritas (*Inheritance*)

Mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada -

objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).

2.7.2 UML *(Unified Mo*deling *Lang**uage)***

UML singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar yang memiliki sintaks dan semantik. Ketika membuat model mengunakan konsep *UML* ada aturan-aturan yang harus diikuti. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya [10].

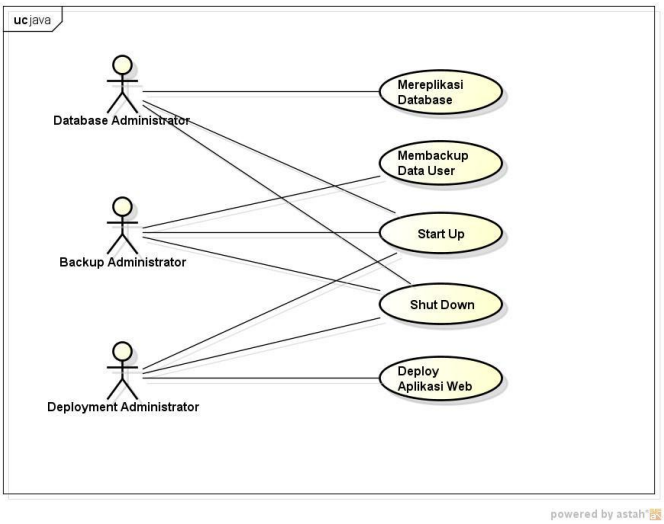
UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Beberapa litelatur menyebutkan bahwa *UML* menyediakan 9 jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misalnya diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian, model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain: [20]

1. *Use Case Diagram*

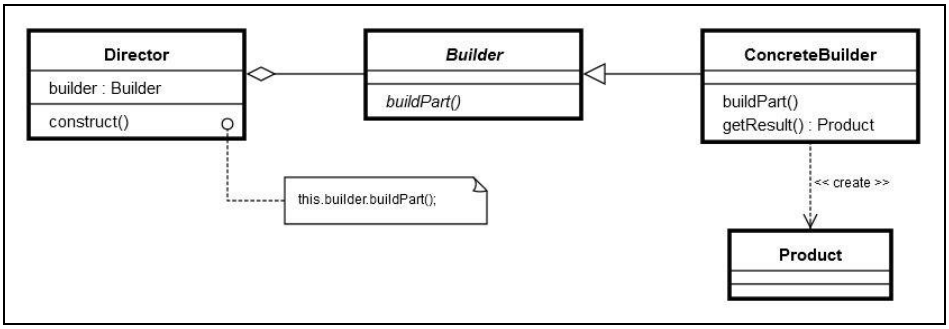
Bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor.



Gambar 2-7 *Use Case Diagram* [10]

2. *Class Diagram*

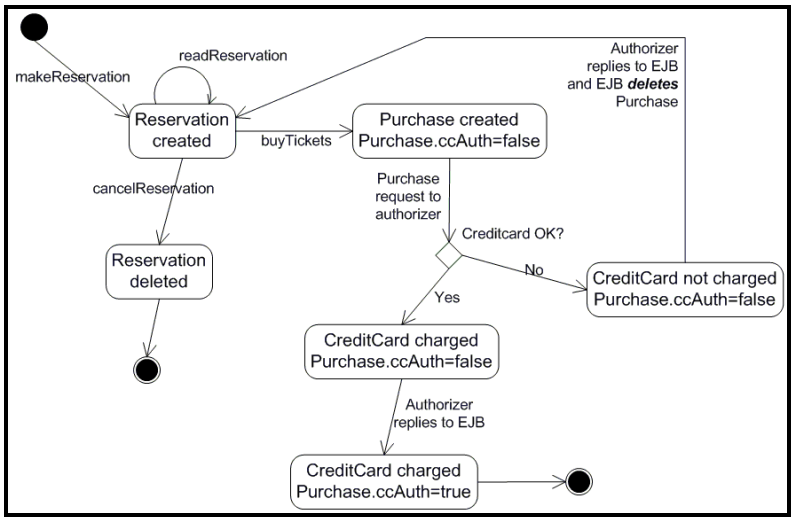
Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.



Gambar 2-8 *Class Diagram* [10]

3. *Statechart Diagram*

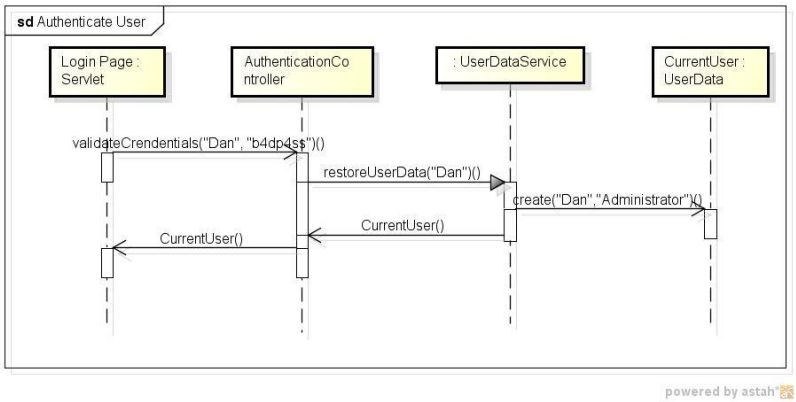
Bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status *(state)*, transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.



Gambar 2-9 *Statechart* **Diagram [10]**

4. *Sequance Diagram*

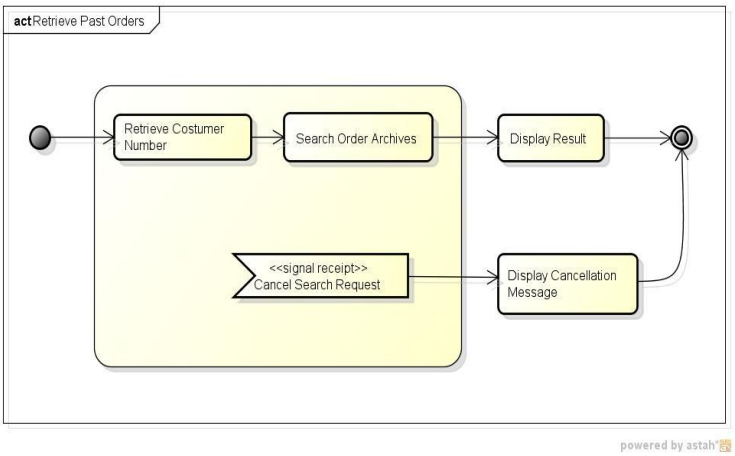
Bersifat dinamis. *Sequence Diagram* merupakan diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.



Gambar 2-10 ***Sequance Diagram*** **[10]**

5. *Activity Diagram*

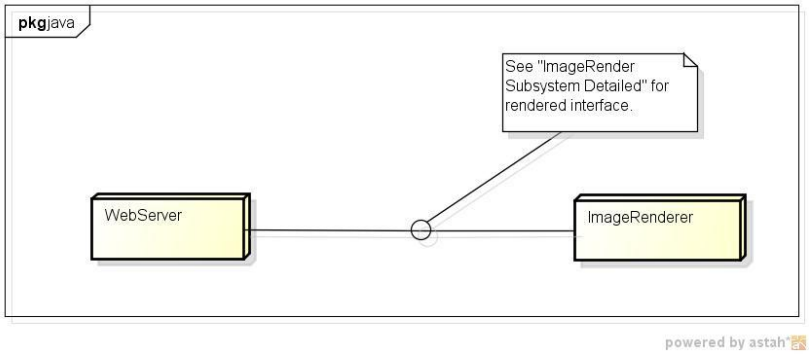
Activity Diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas laiinya seperti *Use Case* atau interaksi



Gambar 2-11 *Activity Diagram* [10]

6. *Deployment Diagram*

Bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan *(run-time)*. Memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya.



Gambar 2-12 ***Deployment Diagram*** **[10]**

2.7.3 Pengertian *Unity Game* 3D

UNITY 3D adalah sebuah *game* developing software. Dengan *software* ini, kita bisa membuat *game* 3D yang seru. Jika anda belum memakai *unity*3D, maka bersiaplah untuk mencobanya, karena *game* developer ini sangat mudah menggunakannya, dengan GUI yang memudahkan kita untuk membuat mengedit dan membuat script untuk menciptakan sebuah *game* 3D.selain bisa untuk build *game* PC, *UNITY* juga dapat digunakan untuk membangun *game console* seperti *Nintendo* Wii, PS3, Xbox 360, juga *Ipad*, *Iphone*, &*android*. Namun masing-masing membutuhkan biaya lisensinya sendiri. Selain itu juga bisa juga buat web, cuma butuh *install Unity web player*. *Unity* lebih kurang sama dengan Blender *game* engine cuma unity lebih ringan dan diintegrasi lebih didalam suasana grafikbahasa pemrograman yang dapat diterima *Unity* adalah *java script, cs script* (C#) dan *Boo script*. Kelebihan *unity* yaitu *multiple platform*, maksudnya banyak platform yang diisupport oleh unity, misal *Windows, Mac, iPhone, iPad, Android, Nintendo* Wii, dan juga *browser*. Untuk *browser*, kita memerlukan sebuah *plugin*, yaitu *Unity Web Player*, sama halnya dengan *Flash Player* pada *Browser*.[1]

Bahasa pemrograman yang digunakan bermacam-macam, mulai dari *Javascript*, C#, dan *Boo*.Pada *unity*, kita tidak bisa melakukan desain / *modelling*, dikarenakan *unity* bukan *tool* untuk mendesain Jadi jika kita ingin mendesain, kita memerlukan 3D editor lain seperti 3dsmax atau *Blender*, kemudian kita *export* menjadi format .fbx.

2.7.4 Pengertian Bahasa Pemrograman Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam.Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari *Oracle* dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam *p-code* (*bytecode*) dan dapat dijalankan pada berbagai Mesin *Virtual Java* (JVM).Java merupakan bahasa

pemrograman yang bersifat umum/non-spesifik (general purpose), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin.Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa *platform* sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun".Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

2.7.5 Bahasa C# (C sharp)

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# Language Specification, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam keyboard standar.