BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Simulasi

Simulasi berasal dari kata *simulate* yang artinya berpura-pura atau berbuat seakan akan. Sebagai metode mengajar, simulasi dapat diartikan sebagai cara penyajian pengalaman belajar dengan menggunakan situasi tiruan untuk memahami tentang konsep, prinsip atau keterampilan tertentu. Simulasi dapat digunakan sebagai metode mengajar dengan asumsi tidak semua proses pembelajaran dapat dilakukan secara langsung pada objek sebenarnya.

Demikian juga untuk mengembangkan pemahaman dan penghayatan terhadap suatu peristiwa, maka penggunaan simulasi akan sangat bermanfaat. Simulasi adalah suatu proses peniruan dari sesuatu yang nyata beserta keadaan sekelilingnya (state of affairs). Aksi melakukan simulasi ini secara umum menggambarkan sifat-sifat karakteristik kunci dari kelakuan sistem fisik atau sistem yang abstrak tertentu. Simulasi didefinisikan sebagai tehnik untuk membuat konstruksi model matematika untuk suatu proses atau situasi, dalam rangka menduga secara karakteristik atau menyelesaikan masalah berkaitan dengannya dengan menggunakan model yang diajukan. Jadi simulasi mempelajari atau memprediksi sesuatu yang belum terjadi dengan cara meniru atau membuat model sistem yang dipelajari dan selanjutnya mengadakan eksperimen secara numerik dengan menggunakan komputer.*(Maisaroh Nasution ; 2014: 94)*

II.2. Pengertian Animasi

Film animasi berasal dari dua disiplin ilmu, yaitu film yang berakar pada dunia fotografi dan animasi yang berakar pada dunia gambar. Animasi dipandang sebagai suatu hasil proses dimana obyek-obyek yang digambarkan atau divisualisasikan tampak hidup. Gambar digerakkan melalui perubahan sedikit demi sedikit dan teratur sehingga memberikan kesan hidup.

Dalam dunia penyiaran ada ketentuan dalam penentuan resolusi animasi. Resolusi tersebut berpengaruh pada frame per *secondnya*. menurut NTSC (*NationaltelevitionStandardComitee*) ukuran dasar yang digunakan atar *frame* per *second* adalah 24 *frame* per *second* (24fps).(*Chabib Syafrudin* ; *2013: 389*)

1. Jenis Film Animasi
    a. Animasi 2 Dimensi (2D)
    b. Animasi 3 Dimensi (3D)
    c. Animasi Tanah Liat (*ClayAnimation*)
    d. Animasi Jepang (*Anime*)
2. Bentuk Film Animasi
    a. Film *Spot* (10 sampai 60 detik)
    b. Film *Pocket* Cartoon (50 detik sampai 2 menit)
    c. Film Pendek (2 sampai 20 menit)
    d. Film Setengah Panjang (20 sampai 50 menit)
    e. Film Panjang (minimal 50 menit)
3. Proses Pembuatan Produksi Film Animasi
    a. Pembuatan

b. Tema

c. *Logline*

d. Sinopsis

e. *CharacterDevelompment*

f. Membuat *storyboard*

g. Membuat gambar

h. Pewarnaan *digital* (*coloring*)

i. Dubbing

j. Konversi ke CD

II.3. 3 Dimensi

II.3.1. Konsep Dasar Objek 3D

Objek 3D adalah representasi dari data geometrik 3 dimensi sebagai hasil dari pemrosesan dan pemberian efek cahaya terhadap grafika komputer 2d. Hasil ini kadang kala ditampilkan secara waktu nyata (real time) untuk keperluan simulasi. Secara umum prinsip yang dipakai adalah mirip dengan objek 2d, dalam hal: penggunaan algoritma, grafika vektor, model frame kawat (wire frame model), dan grafika rasternya.

Objek 3D sering disebut sebagai model 3D. Namun, model 3D ini lebih menekankan pada representasi matematis untuk objek 3 dimensi. Data matematis ini belum bisa dikatakan sebagai gambar grafis hingga saat ditampilkan secara visual pada layar komputer atau printer. Proses penampilan suatu model matematis ke bentuk citra 2d biasanya dikenal dengan proses 3D rendering (*Angga Prasetio Romadhon ; 2013 : 222*)

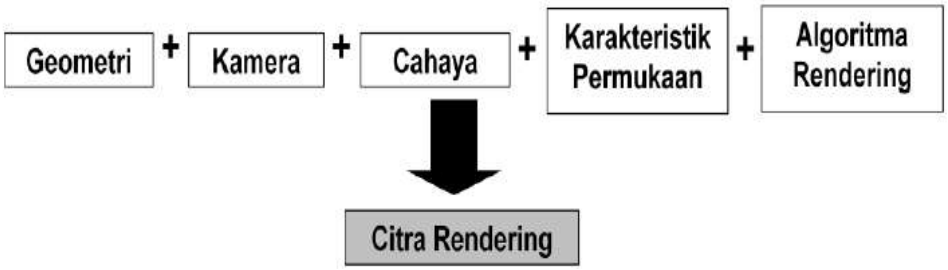
II.3.2. Konsep Dasar Rendering Objek 3D

Rendering merupakan sebuah proses untuk menghasilkan sebuah citra 2D dari data 3D. Proses ini bertujuan untuk untuk memberikan visualisasi pada user mengenai data 3D tersebut melalui monitor atau pencetak yang hanya dapat menampilkan data 2D.

Gambar yang dibuat melalui proses rendering digital adalah gambar digital atau raster image, jenis gambar yang sama dengan yang biasa kita lihat sehari-hari pada desktop komputer atau wallpaper. Gambar digital tersebut dibuat melalui proses rendering digital sebagai langkah besar terakhir sebelum disusun menjadi animasi. Animasi sebagai tujuan akhir biasa digunakan dalam film, video game, permainan komputer, simulator, dan untuk efek khusus di televisi. Masing-masing menggunakan proses rendering digital yang menggunakan fitur dan teknik berbeda untuk mencapai hasil yang diinginkan. (*Angga Prasetio Romadhon ; 2013 : 223*)

Secara umum, proses untuk menghasilkan rendering dua dimensi dari objek-objek 3D melibatkan 5 komponen utama :

a. Geometri
b. Kamera
c. Cahaya
d. Karakteristik Permukaan
e. Algoritma Rendering



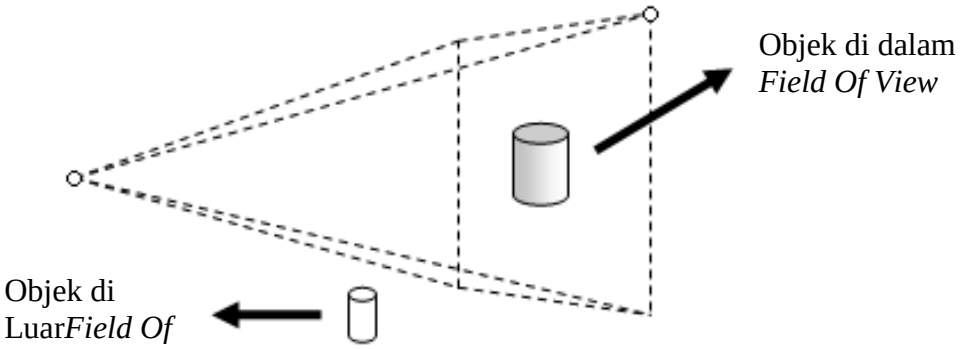
Gambar II.1. Skema Proses Rendering

Sumber :(*Angga Prasetio Romadhon ; 2013 : 223*)

II.3.3. Kamera

Dalam grafika 3D, sudut pandang (point of view) adalah bagian dari kamera. Kamera dalam grafika 3D biasanya tidak didefinisikan secara fisik, namun hanya untuk menentukan sudut pandang kita pada sebuah world, sehingga sering disebut virtual camera.

Pada kamera, dikenal field of view yaitu daerah yang terlihat oleh sebuah kamera. Field of view pada grafika 3D berbentuk piramid, karena layar monitor sebuah komputer berbentuk segiempat. Objek-objek yang berada dalam field of view ini akan terlihat dari layar monitor, sedang objek-objek yang berada di luar field of view ini tidak terlihat pada layar monitor. Field of view ini sangat penting dalam pemilihan objek yang akan diproses dalam rendering. Objek-objek diluar field of view biasanya tidak akan diperhitungkan, sehingga perhitungan dalam proses rendering, tidak perlu dilakukan pada seluruh objek. (*Angga Prasetio Romadhon ; 2013 : 223*)



Gambar II.2. Sudut *Field of view*

Sumber :(*Angga Prasetio Romadhon ; 2013 : 223*)

II.3.4. Cahaya

Sumber cahaya pada grafika 3D merupakan sebuah objek yang penting, karena dengan cahaya ini sebuah world dapat terlihat dan dapat dilakukan proses rendering. Sumber cahaya ini juga membuat sebuah world menjadi lebih realistis dengan adanya bayangan dari objek-objek 3D yang ada.

Sebuah sumber cahaya memiliki jenis. Pada grafika 3D dikenal beberapa macam sumber cahaya, yaitu :

1. ***Pointlight***

Memancar ke segala arah, namun intensitas cahaya yang diterima objek bergantung dari posisi sumber cahaya. Tipe ini mirip seperti lampu pijar dalam dunia nyata.

2. *Spotlight*

Memancarkan cahaya ke daerah tertentu dalam bentuk kerucut. Sumber cahaya terletak pada puncak kerucut. Hanya objek-objek yang terletak pada daerah kerucut tersebut yang akan nampak.

3. *Ambientlight*

Cahaya latar/alam. Cahaya ini diterima dengan intensitas yang sama oleh setiap permukaan pada benda. Cahaya latar tersebut dimodelkan mengikuti apa yang terjadi di alam, dalam keadaan tanpa sumber cahaya sekalipun, benda masih dapat dilihat.

4. *Directionallight*

Memancarkan cahaya dengan intensitas sama ke suatu arah tertentu. Letak tidak mempengaruhi intensitas cahayanya. Tipe ini dapat menimbulkan efek seolah-olah sumber cahaya berada sangat jauh dari objek.

5. *Parallelpoint*

Sama halnya dengan directional light, hanya pencahayaan ini memiliki arah dan posisi.

II.4. *3D Max*

3ds Max adalah sebuah software yang dikhususkan dalam pemodelan 3 dimensi ataupun untuk pembuatan animasi 3 dimensi. Selain terbukti handal untuk digunakan dalam pembuatan objek 3 dimensi, 3ds Max juga banyak digunakan dalam pembuatan desain furnitur, konstruksi, maupun desain interior.

Selain itu, 3ds Max juga sering digunakan dalam pembuatan animasi atau film kartun.

3ds Max yang dilengkapi dengan bahasa *scripting* (*MaxScripting*) terbukti ampuh untuk membuat game 3 dimensi, mulai dari yang sederhana hingga yang rumit sekalipun. Dengan kemampuan tersebut, banyak orang maupun instansi memanfaatkan software 3ds Max untuk membuat suatu desain atau iklan yang berguna sebagai media publikasi produk atau karya mereka kepada public. 3ds Max memungkinkan pengguna untuk membuat tampilan 3 dimensi yang sangan menarik.(*Galih Purnowo ; 2010 : 1*)

II.5. Kecelakaan

Berdasarkan hasil investigasi KNKT, sumber kecelakaan kereta api disebabkan oleh bermacam macam faktor, diantaranya sarana jalan / rel, maintenance kereta api, dan kesalahan yang dilakukan oleh masinis. Faktor manusia (SDM) memiliki pengaruh cukup besar, yaitu 23 % dari faktor keseluruhan penyebab kecelakaan kereta api, dengan total kecelakaan yang diakibatkan faktor manusia atau SDM sebesar 135 dari 588 total kecelakaan. Hal ini dipengaruhi oleh faktor kewaspadaan dari seorang masinislah yang menjadi sorotan utama, Tingkat dari kewaspadaan seorang pekerja menjadi sangat begitu penting ketika pekerja tersebut melakukan pekerjaannya, terlebih lagi apabila pekerjaan tersebut membawa banyak nyawa atau suatu pekerjaan yang mengendalikan fasilitas transportasi seperti seorang masinis kereta api.

Penyebab terjadinya suatu kecelakaan adalah akibat rasa kantuk dan kurangnya konsentrasi saat mennyetir . Rasa kantuk itu tersebut dipengaruhi juga

oleh tingkat kewaspadaan dan hypnosis. Dan untuk tingkat kewaspadaan dan hypnosis dipengaruhi oleh beberapa faktor yakni kelelahan, rasa ngantuk, lingkungan, kondisi psikis, dan distraksi saat mengemudi. Untuk tingkat kelelahan dipengaruhi lagi oleh beberapa aspek yaitu beban kerja fisik, massa kerja, dan bagaimana pembagian shift kerjanya. Untuk rasa ngantuk juga dipengaruhi oleh beberapa aspek yaitu kualitas tidur, beban mental, shift kerja, beban fisik kerja, dan usia dari masinis itu sendiri. Untuk faktor lingkungan dipengaruhi juga oleh beberapa aspek seperti pencahayaan, kebisingan, dan suhu. Kecelakaan kereta api disebabkan juga oleh tingkat kelelahan atau *fatigue* level yang terjadi pada seorang masinis saat mengendalikan kereta api. Tingkat kelelahan ini disebabkan oleh lama bekerja dan waktu istirahat terakhir dan akan menyebabkan hilangnya tingkat konsentrasi dan tingkat kefokusan seorang masinis dalam membaca simbol-simbol dan menangkap informasi-informasi yang diberikan sehingga memicu terjadinya kecelakaan kereta api.*(Wiwik Budiawan ; 2014 : 168-169)*

II.6. Trasportasi

II.6.1. Pengertian Transportasi Kereta Api (KA)

Transportasi Kereta Api (KA) merupakan alat angkut yang efisien, cepat, relatif aman dan adaptif terhadap perkembangan jaman. Keunggulan-keunggulan tersebut membuat Kereta Api dimanfaatkan masyarakat untuk mobilitas. Kereta Api merupakan sarana pendukung dalam usaha menurunkan polusi yang diakibatkan oleh banyaknya kendaraan bermotor.

Terdapat berbagai macam masalah dalam operasional Kereta Api. Berdasarkan data jenis kecelakaan Kereta Api yang diambil dari Komite Nasional

Keselamatan Transportasi (KNKT) pada tahun 2007 hingga 2011, kecelakaan tumburan antar KA yaitu 29%, anjlokan 64% dan lain-lain sebesar 7%. Faktor penyebab kecelakaan Kereta Api diakibatkan oleh faktor prasarana 32%, sarana 34%, manusia 17%, eksternal 7% dan operasional 10% (*Database KNKT, 2011*). Berdasarkan data KNKT, faktor yang dominan menjadi penyebab kecelakaan Kereta Api, adalah sarana dan prasarana. *(Anggo, Yayan, Wahyu; 2013: 1)*

II.6.2. Pengertian Mobil dan Sepeda Motor

Mobil adalah kendaraan beroda empat atau lebih yang membawa komponen permesinan sendiri. Jenis-jenis mobil terdiri atas sedan, bis, van, dan truk. Sepeda motor adalah kendaraan beroda dua yang ditenagai oleh sebuah mesin. Ditinjau daro aspek tektologi, sepeda motor merupakan alat/saranan transportasi bermesin yang paling sederhana. Dalam hal ini, sepeda motor merupakan pengembangan dari sepeda konvensional. Konsep teknologi sepeda motor berada diantara sepeda konvensional dan mobil.*(Herni Kusantati ; 2006 : 108)*

II.7. Multimedia

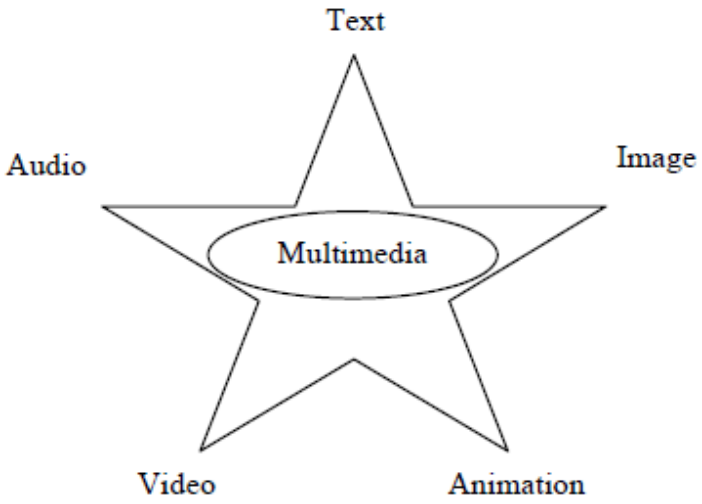
II.7.1. Pengertian multimedia

Multi-banyak, Media-sarana berkomunikasi untuk melewatkan informasi.Suatu sistem yang terdiri dari perangkat keras, perangkat lunak dan alat – alat lain seperti televisi, monitor video dan sistem piringan optik atau sistem stereo yang dimaksudkan untuk menghasilkanpenyajian audio visual yang utuh.Beberapa pakar mengartikan multimedia sebagai berikut :

1. Multimedia secara umum merupakan kombinasi 3 elemen yaitu suara, gambar dan teks.
2. Multimedia adalah kombinasi dari paling sedikit 2 media input atau output dari data, media inidapat audio (suara, musik), animasi, video, teks, grafik, dan gambar.
3. Multimedia merupakan alat yang dapat menciptakan prestasi yang dinamis dan intraktif yang mengkombinasikan teks grafik, animasi, audio dan gambar video.
4. Multimedia adalah pemanfaatan komputer untuk membuat dan menggabungkan teks, grafik,audio, gambar bergerak (video dan animasi) dengan menggabungkan link dan tool yang memungkinkan pemakai melakukan navigasi, berintraksi, berkreasi dan berkomunikasi.

II.7.2. Kelebihan Multimedia

Dari berbagai media informasi, multimedia memiliki suatu kelebihan tersendiri yang tidak dapat digantikan oleh penyajian media informasi lainya.Kelebihan dari multimedia adalah menarik indra dan menarik minat, karena merupakan gabungan antara pandangan, suara dan gerakan. Multimedia terbagi dalam beberapa element-element multimedia,seperti yang rerlihat dalam gambar dibawah ini (*Chrisna Atmadji ; 2010 : 59*)



Gambar II.3. Komponen Multimedia

Sumber :(*Chrisna Atmadji ; 2010 : 60*)

1. Teks

Bentuk data multimedia yang paling mudah disimpan dan dikendalikan adalah teks. Teks dapat membentuk kata, surat atau narasi dalam multimedia yang menyajikan bahasa.

2. Image (*grafik*)

Alasan untuk menggunakan gambar dalam presentasi atau publikasi multimedia adalah karenalebih menarik perhatian dan dapat mengurangi kebosanan dibandingkan dengan teks. Gambar dapat meringkas menyajikan data yang kompleks dengan cara yang baru dan lebih berguna.

3. Bunyi (audio)

PC multimedia tanpa bunyi hanya disebut unimedia, bukan multimedia. Bunyi dapat ditambahkan dalam multimedia melalui suara, musik dan efek-efek suara.

4. Video

Video menyediakan sumberdaya yang kaya dan hidup bagi aplikasi multimedia.

5. Animasi

Dalam multimedia, animasi merupakan penggunaan komputer untuk menciptakan gerak padalayer.

6. Virtual Reality

Virtual reality merupakan penggunaan multimedia untuk penerapan secara langsung.

II. 8 UML *(Unified Modelling Language)*

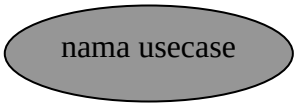
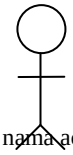
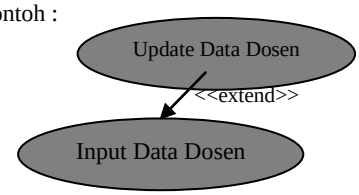
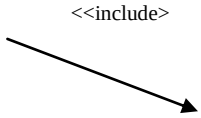
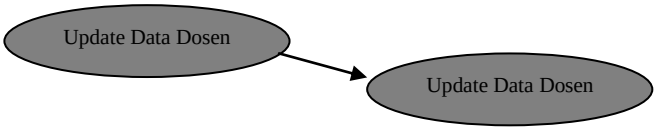
Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap

bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh Object Management Group (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.*(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram Use Case yang terdapat di dalam UML. Adapun maksud dari Use Case Diagram diterangkan dibawah ini.

II. 8.1 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain. *(Yuni Sugiarti ; 2013 : 41)*

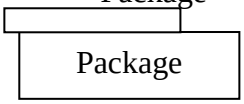
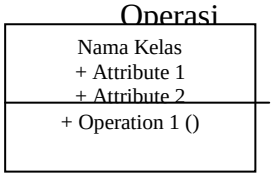
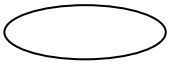
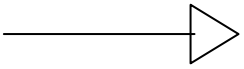
Simbol	Deskripsi
Usecase 	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase namause case.
Aktor  nama actor	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan dengan menggunakan kata benda diawal frase nama aktor.
Assosiasi / Association ————————	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
Exetnd 	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjuk pada use case yang dituju. Contoh : 
Include 	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsi atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

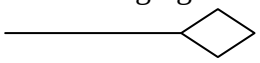
Gambar II.3. ***Use Case*** **Diagram**

Gambar II.4. *Sumber : (Yuni Sugiarti ; 2013 ; 42)*

II. 8.2 *Class Diagram*

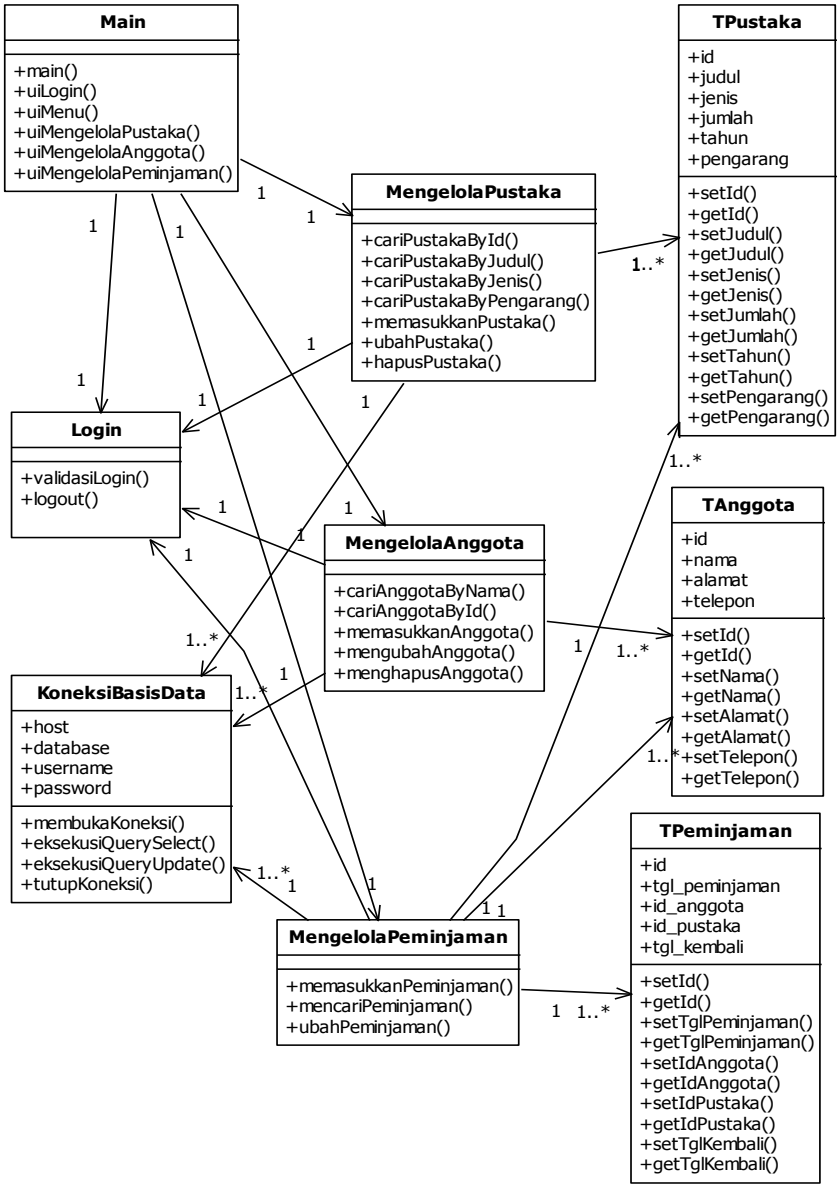
Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka/Interface  interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity.
Asosiasi Berarah/directed assosiasi	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity.
Generasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum – khusus)
Kebergantungan/Defedency	Relasi antar kelas dengan makna kebergantungan antar kelas

Agregasi 	Relasi antar kelas dengan makna semua bagian (whole-part)

Gambar II.5. *Class* Diagram

Sumber : (Yuni Sugiarti ; 2013:59)



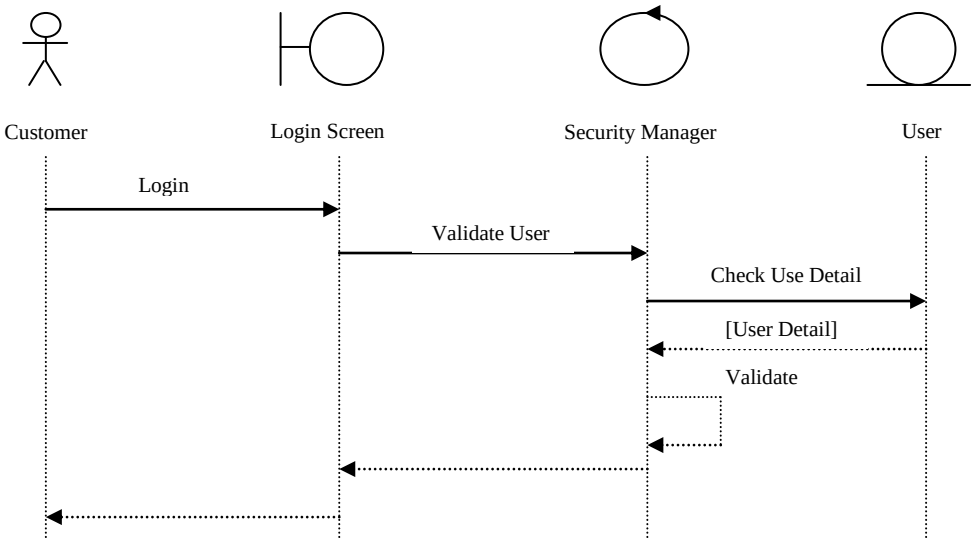
Gambar II.6. Contoh *Class* Diagram

Sumber : (Yuni Sugiarti ; 2013 : 63)

II. 8.3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.7. Contoh *Sequence* Diagram

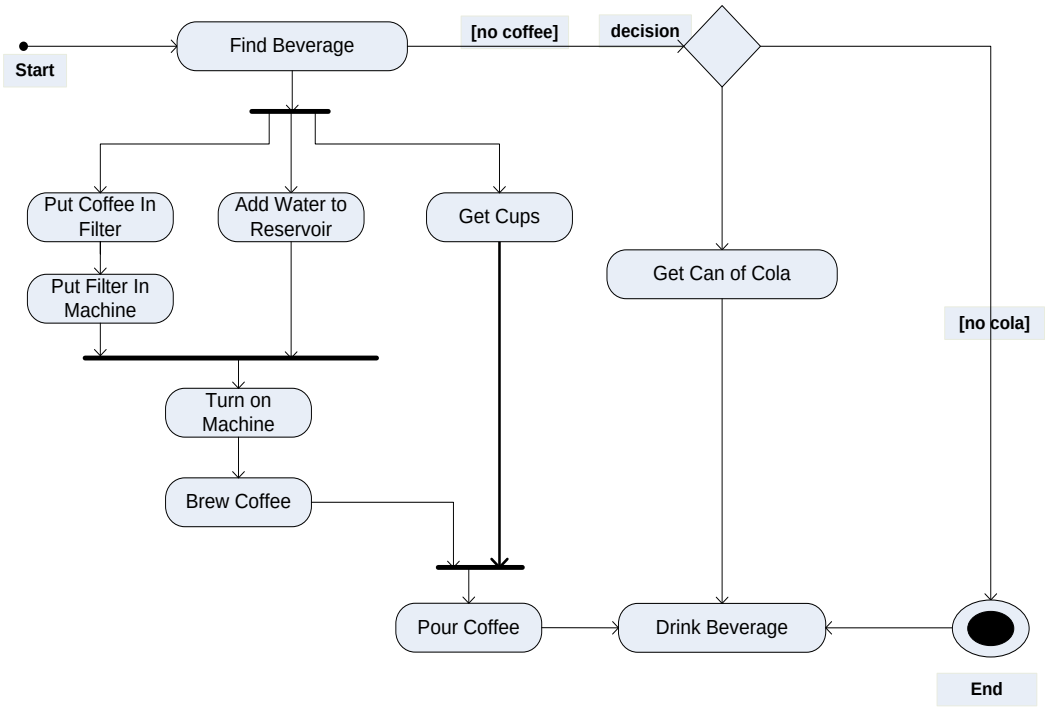
Sumber : (Yuni Sugiarti ; 2013 : 63)

II. 8.4. *Activity Diagram*

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

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas.



Gambar II.8. *Activity* Diagram

Sumber : (Yuni Sugiarti ; 2013 : 76)