BAB 2
TINJAUAN PUSTAKA

2.1 Kanker

Kanker merupakan salah satu penyakit yang banyak menyerang manusia dewasa ini. Penyebab terjadinya penyakit kanker sangat dipengaruhi oleh pola hidup masing-masing individu. Berikut akan dibahas lebih lanjut mengenai apa itu penyakit kanker, apa saja penyebabnya, dan gejala yang timbul.

2.1.1 Pengertian Kanker

Kanker adalah penyakit yang tidak mengenal status sosial dan dapat manyerang siapa saja dan muncul akibat pertumbuhan tidak normal dari sel-sel jaringan tubuh yang berubah menjadi sel kanker dalam perkembangannya. Sel kanker berbahaya karena dapat menyebabkan kematian baik secara langsung maupun tidak langsung. Sel kanker tumbuh dengan cepat, sehingga sel kanker pada umumnya cepat menjadi besar. Di samping itu, sel kanker dapat menyebar (metastasis) ke bagian alat tubuh lainnya yang jauh dari tempat asalnya melalui pembuluh darah dan pembuluh getah bening sehingga tumbuh kanker baru di tempat lain. [2]

Penyebaran sel kanker ke jaringan sehat pada alat tubuh lainnya dapat merusak alat tubuh tersebut sehingga fungsi alat tersebut menjadi terganggu. Awalnya kanker tidak menimbulkan keluhan karena hanya melibatkan beberapa sel. Bila sel kanker bertambah, maka keadaan bergantung kepada orang yang terkena. Misalnya, pada usus berongga besar, tumor harus mencapai ukuran besar sebelum memicu keluhan. [3] Pada taraf stadium lanjut sel kanker menyebar sampai ke organ vital seperti otak atau paru lalu mengambil nutrisi yang dibutuhkan oleh organ tersebut, akibatnya organ itu rusak dan mati.

2.1.2 Penyebab Kanker

Menurut Namora dan Hasmida [2] Ada empat faktor utama penyebab kanker seperti lingkungan, makanan, biologis, dan psikologis. Berikut ini adalah penjelasan mengenai keempat faktor penyebab kanker tersebut, yaitu:

1. Lingkungan

Lingkungan yang dapat menyebabkan kanker antara lain:

a. Bahan kimia

   Zat yang terdapat pada asap rokok yang dapat menyebabkan kanker paru pada perokok aktif dan perokok pasif (orang yang bukan perokok atau tidak sengaja menghirup asap rokok orang lain) dalam jangka waktu yang lama.

b. Penyinaran yang berlebihan

   Sinar ultra violet yang berasal dari matahari dapat menimbulkan kanker kulit. Sinar radio aktif sinar X yang berlebihan atau radiasi dapat menimbulkan kanker kulit dan leukimia.

c. Merokok

   Rokok putih bertanggung jawab 90% dari semua kasus kanker paru-paru yang menjadi penyebab utama kematian baik dari wanita daripada pria.

d. Polusi udara

   Menurut Chen Zichou seorang ahli Institut Penelitian Kanker mengatakan, penyebab utama meningkatnya jumlah kanker di China disebabkan polusi udara, lingkungan, dan kondisi air yang kian hari kian memburuk.

2. Makanan

Makanan yang dapat menyebabkan kanker antara lain adalah:

a. Daging yang mengandung hormon (DES or Diethylstilbestrol).
b. Bahan pemanis buatan seperti biang gula dan saccharin.

c. Nitrosamines pada bahan-bahan pengawet buatan, dan bahan pewarna buatan, yang umumnya dipakai dalam produk daging, yang telah diproses dan juga banyak dalam produk makanan kaleng.
d. Zat pewarna yang ada dalam makanan, minuman, kosmetik, maupun obat obatan.
e. Zat radioaktif yang sekarang ini terdapat hampir di seluruh bulatan bumi sebagai akibat dari percobaan bom atom serta peledakan bom, yang masuk dalam tubuh manusia melalui makanan, khususnya susu.
f. Kebanyakan makan garam.
g. Makanan yang sudah menjadi tengik.

3. Biologi

Faktor biologi penyebab penyakit kanker diantaranya adalah:

a. Virus

Salah satu virus yang dapat menyebabkan kanker adalah virus HIV (human immunodefiency virus). Dimana virus HIV (human immunodefiency virus) ini dapat merusak sistem kekebalan tubuh. Akibatnya wanita yang terinfeksi virus HIV (human immunodefiency virus) akan rentan terhadap infeksi HPV (human papillomavirus).

b. Hormon

Hormon adalah zat yang dihasilkan kelenjar tubuh yang fungsinya adalah mengatur kegiatan alat-alat tubuh dan selaput tertentu.

c. Keturunan

Sejumlah penelitian menemukan bahwa sekitar 5% dari kasus kanker diakibatkan oleh faktor keturunan. Faktor keturunan ini memang susah untuk dihindari. Tetapi sejauh apa peranan gen yang abnormal masih belum diketahui.

4. Psikologis

Faktor psikologis penyebab penyakit kanker diantaranya adalah:

a. Kepribadian

   Orang dengan tipe kepribadian tertutup termasuk tipe yang mudah terkena stres. Akibatnya mereka akan memiliki risiko tinggi untuk terkena penyakit kanker dan jantung.

b. Stres

   Salah satu sebab menurunnya kekebalan tubuh (immunitas) adalah adanya stres dan kondisi stres ini akan melemahkan respon imunitas. Menurunnya sistem imunitas ini mempermudah masuknya sel-sel kanker menyerang tubuh, karena kemampuan sel tersebut untuk mengenal dan melawan musuh tidak dapat berfungsi secara baik.

2.1.3 Gejala-Gejala Kanker pada anak

Menurut Astana [4], kanker dapat mengenai semua organ tubuh dengan tanda-tanda:

a. Mata

   Bintik putih pada mata "Mata Kucing", mata tampak lebih besar, mata menonjol, pendarahan pada mata secara spontan, mata mendadak juling. Umumnya terjadi pada anak-anak dibawah 4 tahun.

b. Pembengkakan

   Misalnya pada hati, limpa, leher, buah zakar, kelenjar getah bening dan tulang.

c. Rasa Nyeri

   Pada tulang atau sendi.

d. Tanda-tanda Neurologis

   Sakit kepala yang berkepanjangan dan disertai mual atau muntah yang menyemprot pada saat bangun tidur, gangguan keseimbangan, penurunan kesadaran, kejang, perubahan perilaku, kelumpuhan anggota gerak dan saraf otak.

e. Tanda-tanda Umum

   Pucat, demam yang tdak diketahui sebabnya, pendarahan yang abnormal

(lebam/biru-biru di kulit, mimisan, kencing berdarah), penurunan berat badan dan tidak nafsu makan.

2.2 Game

Game merupakan kata dalam bahasa Inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan refreshing. Dalam penggunaannya kaa game sering digunakan untuk menyebutkan video game. Untuk selanjutnya akan dibahas mengenai pengertian game, genre game, casual game, game edukasi, *puzzle* game, dan game balancing.

2.2.1 Pengertian *Game*

Menurut Erick Zimmerman, dan Katie Salen [5] *Game* merupakan suatu sistem yang memiliki aturan-aturan tertentu dimana pemain akan terlibat di dalam suatu permasalahan sehingga dapat menghasilkan suatu hasil yang dapat diukur yaitu menang atau kalah. *Game* merupakan sesuatu hal yang dimainkan dengan suatu aturan tertentu yang biasa digunakan untuk tujuan kesenangan dan dapat juga digunakan untuk tujuan pendidikan. *Game* umumnya melibatkan stimulasi mental, fisik, atau keduanya. Banyak *game* yang dapat membantu mengembangkan keterampilan praktis yang berfungsi sebagai latihan, atau melakukan peran pendidikan, simulational, atau psikologis.

Sedangkan Jesse [6] menyimpulkan bahwa apa yang disebut *game* memiliki kualitas – kualitas yang membedakan *game* dengan aktifitas lainnya. *Game* harus diikuti secara sukarela, memiliki tujuan, konflik, dan aturan, ada pihak yang menang dan kalah, interaktif, memiliki tantangan, dapat menciptakan suatu nilai internal sendiri, dapat membuat pemain terlibat, dan merupakan sebuah sistem formal yang tertutup. Dia menambahkan bahwa sebuah *game* memiliki masalah untuk diselesaikan, namun tidak semua masalah adalah sebuah *game*. Berdasarkan hal ini serta seluruh kualitas dari sebuah *game*, Jesse [6] menyimpulkan bahwa

game adalah sebuah aktifitas penyelesaian masalah yang menggunakan sikap bermain sebagai pendekatannya.

Menurut Andrew Rollings dan Dave Morris [7], *game* memiliki 5 komponen terpenting, yaitu:

a. Fitur

   Fitur merupakan salah satu hal yang membedakan setiap *game* yang ada. Fitur disini dapat merupakan fitur yang dapat membantu menggambarkan alan cerita *game* ke dalam bentuk-bentuk yang dapat dilihat maupun dirasakan.

b. *Game*play

   *Game*play dapat membantu para developer untuk mengetahui cara kerja suatu *game*, dimana fitur yang telah dibuat akan membentuk suatu *game*play.

c. Interface

   Interface merupakan setiap tampilan yang ada di dalam *game*. Interface yang baik adalah dengan icon yang sangat minimum dan tidak membosankan para pemain *game*.

d. Aturan

   Aturan merupakan kumpulan peraturan-peraturan dalam *game*.

e. Desain Level

   Desain level disini mencakup *style*, *background*, dan jalan cerita dari *game*.

2.2.2 Genre *Game*

Menurut Wolf [8], *game* komputer dikelompokkan dalam berbagai genre atau jenis permainannya antara lain :

1. Side Scrolling *Game* : Dalam *game* ini, pemain dapat menggerakkan karakter kekanan, kiri, atas maupun bawah. Contoh *game* ini adalah Mario bros.
2. Shooting *Game* : Pemain harus mencari dan menembak musuh untuk mencapai tujuan tertentu. *Game* jenis ini dapat dibedakan menjadi 2 kategori, yaitu First Person Shooting, dan Third Person Shooting. Contoh *game* yang masuk ke dalam jenis *game* ini diantaranya adalah counter strike.
3. Role Playing *Game* (RPG) : Pada *game* ini pemain diminta untuk memerankan suatu karakter dan mengemban misi khusus. *Game* ini terhitung memiliki kompleksitas frame yang cukup tinggi. Sebagai contoh, ketika bertemu dengan karakter lain, maka pemain akan dibawa ke tampilan frame baru, dan pada saat itu kita akan mendapatkan sesuatu atau malah melakukan pertarungan. Contoh *game* yang tergolong RPG diantaranya adalah: Final Fantasy, Ragnarok, dan lain sebagainya.
4. Real Time Strategy (RTS) : Sebagian besar *game* yang masuk kategori ini merupakan *game* peperangan. Pada *game* ini pemain dibebani misi tertentu dan dibekali dengan pasukan seadanya (sebagai modal). Selanjutnya pemain dapat menggerakkan, memperbanyak, dan melengkapi persenjataan pasukan pasukan tersebut sambil merancang strategi untuk mempertahankan dan menguasai.
5. Simulation : Merupakan *game* yang mensimulasikan suatu keadaan riil.Contohnya simulasi pengendalian pesawat terbang pada *game* Microsoft Flight Simulator, kemudian simulasi kehidupan sehari-hari pada *game* The SIMS, dan simulasi seorang manajer tim sepakbola pada *game* Champhionship Manager.

6. Racing : *Game* racing merupakan suatu permainan balap otomotif. Contoh *game* racing yang paling terkenal adalah: Need for Speed, Grand Tourismo, Nascar Rumble, Top Gear, dan lain sebagainya.
7. Fighting : Yang masuk kategori *game* jenis ini adalah *game-game* fighting atau tarung. Contoh yang sangat terkenal dari *game* fighting ini diantaranya adalah: Street Fighter, Mortal Combat, Teken, dan lain sebagainya.
8. Educational *Game* : Merupakan suatu permainan yang secara khusus didesain untuk mengajarkan tentang subjek-subjek tertentu, pendalaman terhadap suatu konsep, pengenalan terhadap kejadian-kejadian bersejarah, atau membantu mereka dalam mempelajari suatu keterampilan yang mereka miliki.

Tipe *Game* bila diasosiasikan dengan kemampuan dan keahlian

Tipe *game* dapat dibedakan dengan mengasosiasikan kemampuan dan keahlian yang berhubungan dengan pembelajaran dan pendidikan (Aguilera & Mendiz) [9], antara lain :

1. Arcade dan Platform *Game* dapat menjadi instrumen dalam perkembangan psikomotor dan spatial orientation.
2. Sports dan dynamic *game* dapat memudahkan koordinasi psikomotor dan meringankan stres,
3. Strategy dan role *game* dapat membantu menstimulasi motivasi dari dalam dan refleksi akan nilai dari *game* tersebut.
4. *Puzzle* dan question *game* dapat membantu untuk mengembangkan kemampuan untuk mengungkapkan alasan dan berfikir logis
5. *Game* simulasi dapat membantu dalam perkembangan semua kemampuan intelektual.

Disamping itu, *game* juga dapat menekankan pemain untuk melakukan kegiatan:

1. Reading (membaca) : *Game* juga dapat digunakan untuk mempromosikan membaca buku sesuai dengan *game* yang sedang dimainkan, misalnya lord of the rings.
2. Logical Thinking : *Game* membantu dalam berfikir bagaimana memcahkan masalah dengan mengusulkan strategi, mengorganisir elemen dalam mengantisipasi tujuan.
3. Observation : Pemain menggunakan kemampuan observasi dalam bermain dengan mengamati perbedaan visual dan ruang serta jumlah elemen dalam layar.
4. Spatially, Geography : Mengembangkan kemampuan pemain dalam membaca peta dan mengenali bentuk ruang.

1. Casual Game

Menurut Fullerton [10], casual *game* adalah sebuah *game* yang dimaksudkan untuk dinikmati oleh setiap orang, baik pria dan wanita, maupun tua dan muda. Dengan ini, casual *game* dapat diartikan sebagai sebuah *game* yang jauh dengan permainan hentakan, kekerasan, dan kompleksitas permainan dengan tujuan untuk menarik pemain seluas mungkin dari berbagai kalangan.

2. Game Edukasi

Game edukasi adalah game yang khusus dirancang untuk mengajarkan user suatu pembelajaran tertentu, pengembangan konsep dan pemahaman dan membimbing mereka dalam melatih kemampuan mereka, serta memotivasi mereka untuk memainkannya. [16]

Kriteria Game Edukasi

Menurut Hurd dan Jenuings, perancangang yang baik haruslah memenuhi kriteria dari education game itu sendiri. Berikut ini adalah beberapa kriteria dari sebuah education game, yaitu:

1. Nilai Keseluruhan (*Overall Value*)

   Nilai keseluruhan dari suatu game terpusat pada desain dan panjang durasi game. Aplikasi ini dibangun dengan desain yang menarik dan

interaktif. Untuk penentuan panjang durasi, aplikasi ini menggunakan fitur timer.

2. Dapat Digunakan (*Usability*)

   Mudah digunakan dan diakses adalah poin penting bagi pembuat game. Aplikasi ini merancang sistem dengan interface yang user friendly sehingga user dengan mudah dapat mengakses aplikasi.

3. Keakuratan (*Accuracy*)

   Keakuratan diartikan sebagai bagaimana kesuksesan model/gambaran sebuah game dapat dituangkan ke dalam percobaan atau perancangannya. Perancangan aplikasi ini harus sesuai dengan model game pada tahap perencanaan.

4. Kesesuaian (*Appropriateness*)

   Kesesuaian dapat diartikan bagaimana isi dan desain game dapat diadaptasikan terhadap keperluan user dengan baik. Aplikasi ini menyediakan menu dan fitur yang diperlukan user untuk membantu pemahaman user dalam menggunakan aplikasi.

5. Relevan (*Relevance*)

   Relevan artinya dapat mengaplikasikan isi game ke target user. Agar dapat relevan terhadap user, sistem harus membimbing mereka dalam pencapaian tujuan pembelajaran. Karena aplikasi ini ditujukan untuk anak-anak maka desain antarmuka harus sesuai dengan nuansa anak-anak yaitu menampilkan warna-warna yang ceria.

6. Objektifitas (*Objectives*)

   Objektifitas menentukan tujuan user dan kriteria dari kesuksesan atau kegagalan. Dalam aplikasi ini objektivitas adalah usaha untuk mempelajari hasil dari permainan.

7. Umpan Balik (*Feedback*)

   Untuk membantu pemahaman user bahwa permainan (*performance*) mereka sesuai dengan objek game atau tidak, *feedback* harus disediakan. Aplikasi ini menyajikan animasi dan efek suara yang mengindikasikan kesuksesan atau kegagalan permainan.

3. *Puzzle Game*

Menurut Rollings dan Adams [11], *puzzle game* adalah permainan yang meitikberatkan tentang teka-teki. Agar menarik, *puzzle game* harus menantang (tai tidak terlalu keras), visual menarik, dan menyenangkan untuk dimainkan. Delapan langkah utama dari pembuatan *puzzle game* antara lain:

1. Cari inspirasi: hal ini dapat berasal dari berabagai sumber, termasuk perrmainan lainnya. Tetris terinspirasi oleh permainan nonkomputer yang disebut Pentominoes.
2. Sederhara: menyederhanakan tugas-tugas yang terlihat sulit agar dapat dengan mudah dimengerti dan dipahami oleh orang yang memainkannya.
3. Buat ketetapan konstruksi: satu-satunya cara untuk memastikan bahwa konsep bekerja dengan baik adalah dengan membangun sebuah prototipe kertas atau versi yang sederhana dalam sesuatu seperti Macromedia Flash untuk melihat apakah ia bekerja.
4. Tentukan Aturan: ini adalah bagian kunci dari desain teka-teki. Kebanyakan diitandai dalam empat hal: papan, atribut, bergerak, dan tujuan atau kondisi kemenangan.
5. Membangun teka-teki: sebuah teka-teki menantang pemain untuk mendapatkan solusi dari masalah yang ada, tentu saja jalan tidak sederhana. Setiap teka-teki mengharuskan pemain membuat pilihan, beberapa diantaranya menyebabkan buntu. Teka-teki yang baik memerlukan wawasan dari pemain.
6. Test: test digunakan untuk mengetahui apakan *puzzle* yang dibuat terlalu mudah atau sulit. Berusaha mencari kelemahan dari *game*play atau user interface agar terlihat baik bagi pemain.
7. Urutan rancangan: merancang *puzzle* secara berurutan secara linear, memikirkan perpindahan *game* dari satu tempat ke tempat lainnya secara baik.

8. Presentasi: mempresentasikan *puzzle game* yang telah dibuat, mulai dari story line, *game*play, aturan main, dan ketemtuan-ketentuan yang berlaku dalam *game*.

2.3 Unified Modeling Language (UML)

Menurut Dharwiyanti [12], *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, 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.

2.3.1 Diagram UML

UML adalah bahasa untuk mendokumentasikan, memodelkan, dan membangun sistem perangkat lunak dalam bentuk diagram visual. UML dicetuskan oleh *Three Amigos* (tiga sekawan), Grady Booch, James Rumbaugh, dan Ivar Jacobson [12].

Versi-versi dari UML:

a. UML 1.0 diresmikan pada Januari 1997

b. UML 1.1 diresmikan pada Juli 1997

c. UML 1.2 diresmikan pada Juni 1998

d. UML 1.3 diresmikan pada tahun 1999

e. UML 1.5

f. UML 2.0

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu :

a. *Use case Diagram* untuk memodelkan proses bisnis
b. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi
c. *Sequence Diagram* untuk memodelkan pengiriman pesan (message) antar *object*s
d. *Collaboration Diagram* untuk memodelkan interaksi antar *object*s
e. *State Diagram* untuk memodelkan perilaku *object*s di dalam sistem
f. *Activity Diagram* untuk memodelkan perilaku *Use cases* dan *Objects* di dalam sistem
g. *Class Diagram* untuk memodelkan struktur kelas
h. *Object Diagram* untuk memodelkan struktur *object*
i. *Component Diagram* untuk memodelkan komponen *object*
j. Deployment Diagram untuk memodelkan distribusi aplikasi

UML menyediakan 8 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use case* Diagram untuk menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.
2. *Sequence* Diagram untuk menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu.
3. *Collaboration* Diagram menggambarkan interaksi antar objek seperti *sequence* diagram, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message.
4. *Statechart* Diagram untuk menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima.

5. *Activity* Diagram *Activity* diagrams menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.
6. *Class* Diagram untuk menggambarkan struktur dan deskripsi *class*, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.
7. Component Diagram untuk menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan(dependency) di antaranya.
8. Deployment Diagram untuk menggambarkan detail bagaimana komponen dideploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.

Untuk pengembangan aplikasi permainan ini akan digunakan 4 macam UML diagram, yaitu :

1. *Use case* Diagram
2. *Class* Diagram
3. *Sequence* Diagram
4. *Activity* Diagram

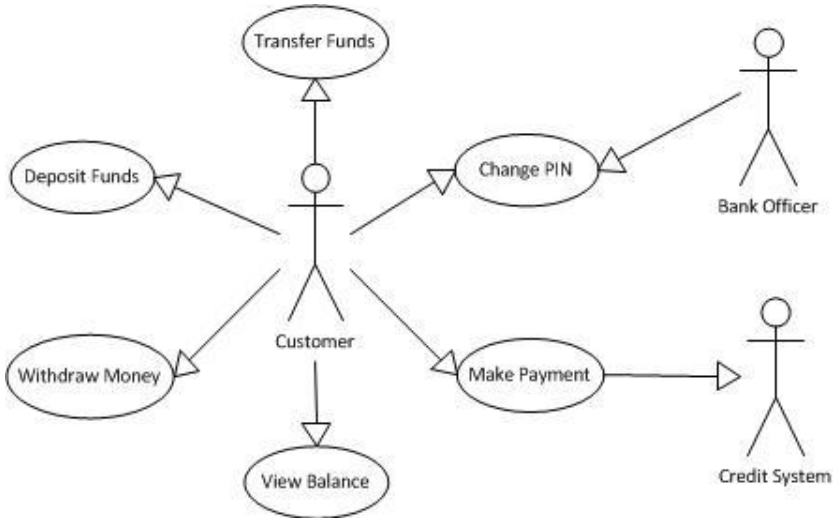
2.3.1.1 Use case Diagram

Use case mengidentifikasi dan menjelaskan fungsi sistem dari perspektif pengguna luar dan dengan cara dan terminologi yang mereka mengerti. Secara akurat dan menyelesaikan permintaan ini, pengguna tingkat tinggi terlibat yang merupakan ahli subjek materi yang memiliki pengetahuan tentang proses *business* atau peristiwa.[12]

Gambaran *use case* diwakili oleh sebuah elips horizontal dengan nama muncul kasus penggunaan di atas, bawah, atau di dalam elips. sebuah *use case* mewakili tujuan tunggal dari sistem yang menggambarkan urutan kegiatan dan interaksi pengguna dalam usaha mencapai tujuan. Pembuatan *use case* telah terbukti menjadi teknik yang sangat baik untuk lebih memahami dan dokumen

persyaratan sistem. *Use case* sendiri adalah tidak dianggap sebagai persyaratan fungsional, tetapi cerita (skenario).

Use case diagram terdiri atas diagram untuk proses dan aktor. Aktor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. Proses merepresentasikan operasi-operasi yang dilakukan oleh aktor. Proses digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. Aktor yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.



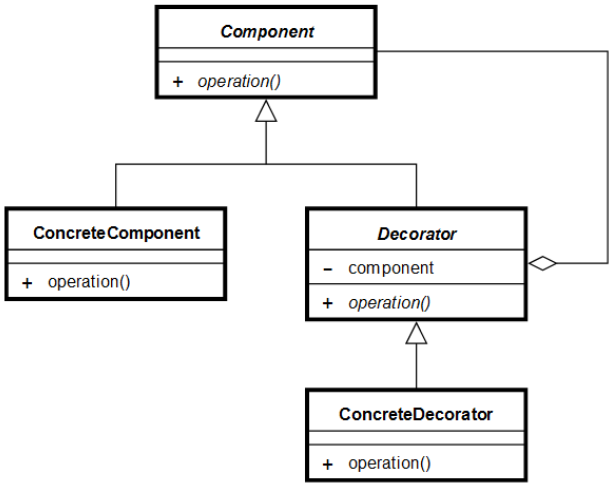
Gambar 1.2 Contoh Use Case Diagram [12]

2.3.1.2 Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. [12]

Menurut Roger S. Pressman [1] *Class* diagram digunakan untuk memodelkan kelas-kelas, termasuk atribut dan operasinya serta hubungan dan asosiasinya dengan kelas lainnya. *Class* diagram menyediakan sebuah gambaran

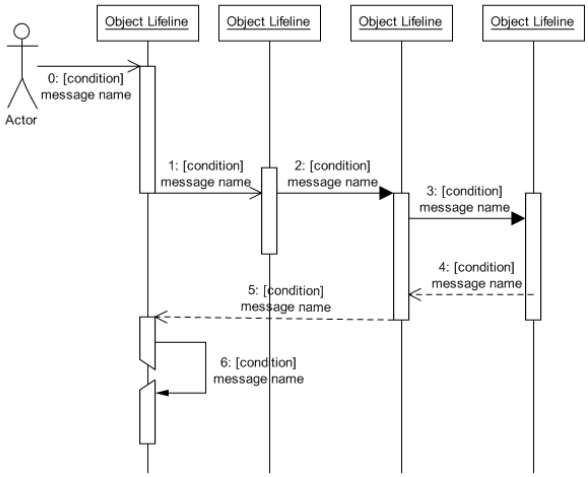
statis atau struktural dari sebuah sistem. *Class* diagram tidakmemperlihatkan komunikasi dinamis antar objek dari kelas yang ada dalam diagram.



Gambar 2.2 Contoh *Class* Diagram [12]

2.3.1.3 Sequence Diagram

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.[12]



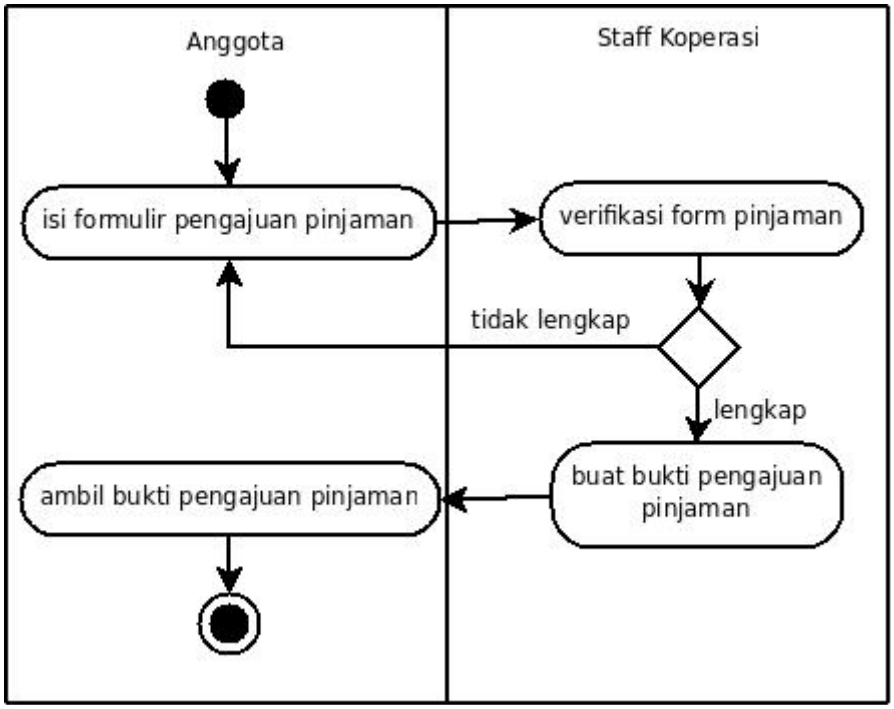
Gambar 2.3 Contoh *Sequence* Diagram [12]

Berikut ini merupakan macam-macam notasi dari *sequence* diagram:

1. *Actor*: dilambangkan dengan gambar actor dari *use case*.
2. *System*: dilambangkan dengan sebuah kotak.
3. *Lifelines*: garis putus-putus vertikal, yang merupakan perpanjangan dari actor menuju dan simbol sistem, yang mengindikasikan rentetan.
4. *Activation bars*: sebuah bar yang mengindikasikan periode waktu kapan partisipan aktif di dalam interaksi.
5. *Input messages*: panah horizontal dari actor menuju sistem yang mengindikasikan pesan masuk.
6. *Output messages*: panah horizontal dari sistem menuju actor, ditunjukkan dengan panah bergaris putus-putus.

2.3.1.4 *Activity Diagram*

Activity diagram, secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis atau *use case*. Diagram ini juga dapat digunakan untuk memodelkan aksi yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari aksi tersebut.



Gambar 2.4 Contoh *Activity Diagram*[12]

Berikut ini merupakan notasi dari *Activity Diagram*:

1. *Initial node*: menunjukan permulaan dari progres, dilambangkan dengan sebuah lingkaran solid.
2. *Actions*: dilambangkan dengan sebuah kotak dengan ujung membulat merepresentasikan setiap langkah individual. Urutan dari aksi membentuk keseluruhan dari aktivitas yang ditunjukkan oleh diagram.
3. *Flow*: dilambangkan dengan panah yang mengindikasikan progres melalui *actions*. Setiap panah tidak diperlukan kata untuk mengidentifikasikannya kecuali untuk menentukan pilhan.
4. *Decision*: dilambangkan dengan lambang wajik. Terdapat satu panah yang masuk dan terdapat 2 atau lebih panah yang keluar dari pilihan wajik. Setiap panah terdapat keterangan yang menerangkan kondisi.
5. *Merge*: dilambangkan dengan lambang wajik, terdapat 2 atau lebih panah yang masuk namun hanya satu panah yang keluar.
6. *Fork*: dilambangkan dengan kotak garis warna hitam, dimana hanya satu panah yang masuk, namun terdapat dua atau lebih panah yang keluar.
7. *Join*: dilambangkan dengan kotak garis warna hitam seperti fork, namun terdapat dua atau lebih panah yang masuk dan satu panah yang keluar.

Activity final: menunjukkan akhiran dari suat progress, dilambangkan dengan suatu lingkaran solid dilingkari oleh lingkaran kosong.

2.5 Android

Android adalah Sistem operasi pada ponsel yang dikembangkan oleh Google maka dibentuklah Open Handset Alliance. Android adalah OS open source pertama untuk ponsel .Android juga merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan key-*application* yang di release oleh Google. Saat ini disediakan Android SDK (*Software* Development Kit) sebagai alat bantu dan API (*Application* Programming Interface) diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman *Java*. [18]

2.5.1 Fitur-fitur Android

Berikut ini adalah fitu-fitur yang ada pada Android :

a. Framework Aplikasi yang mendukung penggantian komponen dan reusable
b. Mesin virtual Dalvik dioptimalkan untuk perangkat *mobile*
c. Integrated browser berdasarkan engine open source WebKit

d. Grafis yang dioptimalkan dan didukung oleh *library* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional akselerasi *hardware*)

e. SQLite untuk penyimpanan data

f. Media Support yang mendukung audio, *video*, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

g. GSM Telephony (tergantung *hardware*)

h. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*)

i. Kamera, GPS, kompas, dan accelerometer (tergantung *hardware*)

j. Lingkungan Development yang lengkap dan kaya termasuk perangkat emulator, tools untuk debugging, profil dan kinerja memori, dan plugin untuk Eclipse IDE.

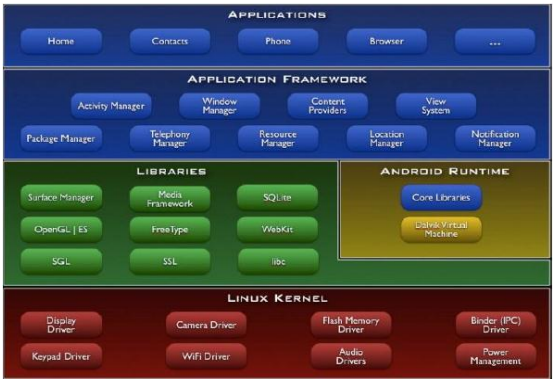
k. Dukungna Perangkat Tambahan : Android dapat memanfaatkan kamera, layar sentuh, accelerometers, magnetometers, GPS, akselerasi 2D (dengan perangkat orientasi, scalling,konversi format piksel), dan akselerasi grafis 3D.

l. Multi-Touch : kemampuan layaknya handset modern yang dapat menggunakan dua jari atau lebih untuk berinteraksi dengan perangkat.

m. Market : handphone memiliki tempat penjualan aplikasi, market pada android merupakan katalog aplikasi yang dapat di download dan di-install melalui internet.

2.5.2 Arsitektur Pada Android

Sistem operasi Android memiliki 4 lapisan (layer) yang merupakan komponen *system* Android. Gambar berikut merupakan lapisan arsitektur *system* operasi android.



Gambar 2.5 Lapisan Arsitektur *System* Operasi Android[18]

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut:

a. *Application* dan *Widgets*

Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di layer terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman java.

b. *Application*s *Framework*s

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Arsitektur aplikasi dirancang supaya pengembang dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*).

Komponen-komponen yang termasuk di dalam *Application Framework*s adalah sebagai berikut:

1. *View* yang kaya dan *extensible* yang dapat digunakan untuk membangun aplikasi, termasuk *list*, *grid*, kotak teks, tombol, dan bahkan sebuah *embeddable web*.
2. *Content provider* yang memungkinkan aplikasi untuk mengakses data (seperti dari daftar kontak *help*) atau membagi data tersebut.
3. *Resource Manager* yang menyediakan akses ke kode non sumber lokal seperti string, gambar, dan tata letak file.
4. *Notification Manager* yang memungkinkan semua aplikasi menampilkan alert yang bisa dokostumisasi didalam status bar.
5. *Activity Manager* yang mengelola siklus hidup aplikasi dan menyediakan navigasi umum backstack.

c. *Libraries*

Libraries ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di

atas kernel. Layer ini meliputi berbagai *Library* C/C++ inti seperti Libe dan SSL, serta:

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries* untuk manajemen tampilan.
3. *Libraries* grafik mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
4. *Libraries* SQLite untuk dukungan *database*.
5. *Libraries* SSL dan *Webkit* terintegrasi dengan *web browser* dan keamanan.
6. *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embedded web View.*
7. *Libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API‟s.

d. Android Run Time

Layer yang membuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine (DVM)* merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam *Android Run Time* dibagi menjadi dua bagian yaitu:

1. *Core Libraries*: Aplikasi Android dibangun dalama bahasa java, sementara Dalvik sebagai *virtual* mesinnya buka *Virtual Machine Java,* sehingga diperlukan sebuah libraries yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh *Core Libraries.*

2. *Dalvik Virtual Machine: Virtual* mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mapu membuat *linux kernel* untuk melakukan *threading* dan manajemen tingkat rendah.

e. Linux Kernel

Linux Kernel adalah *layer* dimana inti dari *operating system* Android itu berada. Berisi file-file sistem yang mengatur sistem *processing, memory, resource, drivers,* dan sistem-sistem operasi Android lainnya. *Linux Kernel* yang digunakan Android adalah *linux kernel* versi 2.6.

2.6 *Tools* Pendukung Pembuatan Aplikasi Android

2.6.1 Java Development Kit (JDK)

JDK adalah *software* development kit merupakan seperangkat kode, *library*, form dan paket yang berguna untuk membuat aplikasi atau applet *java*. Di dalam JDK terdapat *Java* Runtime Environment (JRE) yang akan memproses byte code yang dihasilkan dari interpenter *java*. Tugas dari interpenter *java* ini adalah untuk mengcompile source.*java* menjadi source.*class* yang selanjutnya source.*class* inilah yang akan dijalankan oleh JRE. [20]

2.6.2 Software Development Kit (SDK)

SDK adalah suatu kumpulan dari *library* dan toolkit seperti emulator untuk mengembangkan atau menciptakan aplikasi untuk suatu perangkat lunak, perangkat computer, *system* operasi atau platform. Di dalam SDK terdapat tools yang dibutuhkan dalam pengembangan android, diantaranya yaitu adb shell yang merupakan singkatan dari android development bridge yang dapat menjalankan terminal android seperti terminal pada *system* operasi linux, dan command yang terdapat dalam adb shell sendiri seperti command linux pada umumnya.

2.6.3 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environtment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform independent)*. Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan *web*, dan lain sebagainya.

Eclipse awalnya diciptakan oleh IBM pada bulan November 2001 dan didukung oleh konsorsium vendor perangkat lunak. Eclipse bersifat *open source* di bawah lisensi dari Eclipse Public Licensi (EPL). Berikut ini adalah versi resmi dari Eclipse [19]:

1. Eclipse Europa Packages (2007 - v 3.3)

2. Eclipse Ganymede Packages (2008 - v 3.4)

3. Eclipse Ganymede SR1 Packages (v 3.4.1)

4. Eclipse Ganymede SR2 Packages (v 3.4.2)

5. Eclipse Galileo Packages (2009 - v 3.5)

6. Eclipse Galileo SR1 Packages (v 3.5.1)

7. Eclipse Galileo SR2 Packages (v 3.5.2)

8. Eclipse Helios Packages (2010 - v 3.6)

9. Eclipse Helios SR1 Packages (v 3.6.1)

10. Eclipse Helios SR2 Packages (v 3.6.2)

11. Eclipse Indigo Packages (2011 - v 3.7)

12. Eclipse Juno Sr1 Packages (v 4.2.1)

12. Eclipse Juno Sr2 Packages (v 4.2.2)

2.7 Java

Menurut definisi yang diberikan oleh Sun MicroSystem, JAVA adalah bahasa pemrograman yang *simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded,* dan *dinamyc,*.

Java bahasa yang *simple:* Rancangan bahasa java dibuat sedemikian rupa sehingga dengan cepat dapat dikenali dan dipahami oleh pemrogram pemula sekalipun karena mirip dengan bahasa C/C++ dan yang kedua lebih sederhana dari C/C++ dalam beberapa hal yang sulit pada C/C++ sudah tidak ada pada Java, misalnya pemakaian pointer.

Java adalah bahasa yang *object oriented*: Konsep objek diterapkan pada pembuatan program Java, bahkan java menggunakan konsep objek secara murni. Java adalah bahasa yang *distributed*: Bahasa Java dirancang sedemikian rupa sehingga mendukung pembuatan aplikasi dalam jaringan computer. Java adalah bahasa yang *robust*: rancangan java dibuat sedemikian rupa hingga mengurangi kemungkinan menjadi beku *(freeze)* ketika dijalankan. Java adalah bahasa yang s*ecure*: secara otomatis java menerapkan pengamanan terhadap aplikasi sehingga mengurangi kemungkinan terjadinya serangan dari pengguna jaringan. Java adalah bahasa yang *architecture neutral*: program java yang telah dikompilasi

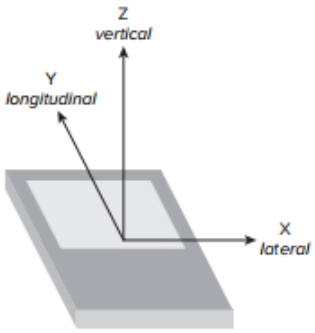
dapat dijalankan. Java adalah bahasa yang *portable*: program java sangat mudah dipindahkan dari satu mesin ke mesin lain tanpa harus dikompilasi ulang.

Java adalah bahasa yang h*igh-performance*: pada hakikatnya program C yang telah dikompilasi berjalan jauh lebih cepat dari Java, namun hasil kompilasi ini belum tentu bisa jalan pada computer yang berbeda arsitektur. Program java dikompilasi menjadi *bytecode* yang bisa jalan diberbagai mesin, namun ketika dijalankan *bytecode* ini harus diterjemahkan oleh JVM ke dalam bahasa mesin yang sesuai, sehingga lebih lambat dari C, namun karena java dapat digunakan pada jaringan computer yang kenyataannya tidak terlalu cepat karena hambatan saluran komunikasi maka program Java masih terasa cukup cepat.

Java adalah bahasa yang m*ultithreaded*: Java dirancang untuk menangani berbagai aplikasi yang berjalan secara bersamaan, misalnya memainkan lagu, sambil melakukan *download*. Kejadian ini disebut dengan *multithreaded*. Java adalah bahasa yang d*ynamic*: bahasa java dirancang untuk beradaptasi dengan lingkungan yang dapat berubah dengan cepat [13].

2.8 *Accelerometer*

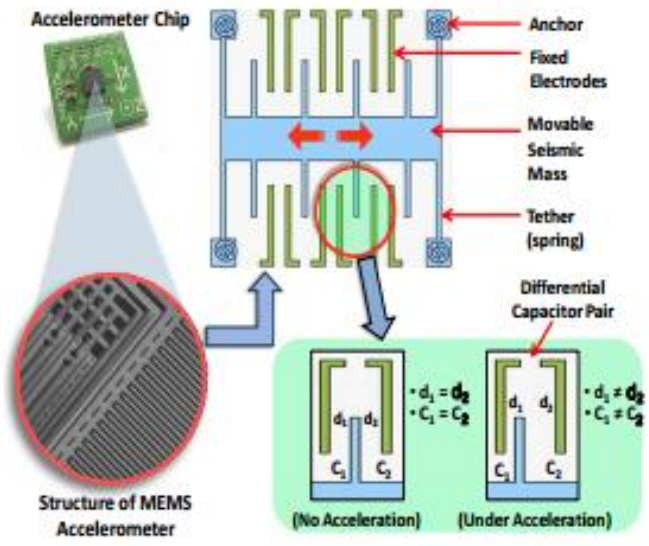
Accelerometer, seperti namanya, digunakan untuk mengukur percepatan. Percepatan didefinisikan sebagai laju perubahan kecepatan, sehingga dapat mengukur seberapa cepat akselerasi pada perangkat berubah dalam arah tertentu. Dengan menggunakan *accelerometer*, anda dapat mendeteksi gerakan, laju perubahan dari kecepatan gerakan yang terjadi pada perangkat. Sangat penting untuk dicatat bahwa *accelerometer* tidak mengukur kecepatan, sehingga anda tidak bisa mengukur kecepatan secara langsung berdasarkan pembacaan *accelerometer* tunggal. Sebaliknya, anda perlu untuk mengukur perubahan dalam percepatan dari waktu ke waktu [14]. Percepatan dapat diukur oleh tiga sumbu arah : *forward-backward* (*longitudinal*), *left-right* (literal), dan *up-down* (*vertical*). Sensor manager melaporkan perubahan sensor di semua tiga sumbu arah. Seperti yang di ilustrasikan pada gambar di bawah ini :



Gambar 2.6 Tiga Sumbu Arah *Accelerometer*

1. *Vertical* (*up-down*), dimana arah positif berarti ke atas gerakan perangkat yang sedang di tinggikan dan sebaliknya adalah nilai negatif
2. *Longitudinal* (*forward-backward*), dimana gerakan kedepan yang menandakan percepatan positif dan gerakan kebelakang yang menandakan percepatan negatif.
3. *Sideways lateral* (*left-right*), dimana gerakan kearah kanan dari perangkat yang mewakili nilai positif dan gerakan perangkat ke arah kiri yang mewakili nilai negatif. Sensor *manager* mempertimbangkan perangkat "diam" ketika menghadap ke atas pada permukaan yang datar dalam orientasi *potret*. Seperti dijelaskan sebelumnya, kita dapat memantau perubahan percepatan dengan menggunakan sensor *listener*.

Berikut adalah gambar arsitektur chip sensor accelerometer yang digunakan pada smartphone



Gambar 2.7 Arsitektur Chip Sensor Accelerometer Pada Smartphone

2.9 *Collision detection*

Fungsionalitas *collision detection* merupakan fitur penting bagi aplikasi *game*. Tabrakan atau collision yang dimaksud adalah antara dua atau lebih Sprite, TiledLayer atau Image dalam satu tampilan. Terkait dengan ini, *Game* API mendukung dua teknik untuk mendeteksi terjadinya collision/tabrakan yaitu [15]:

1. Implementasi dapat membandingkan area-area persegi empat yang merepresentasi-kan sebuah Sprite dengan Sprite, TiledLayer atau Image lainnya. Collision terjadi jika area-area persegi empat tersebut beririsan. Teknik ini merupakan cara yang cepat untuk men*collision detection*, tetapi memungkinkan terjadinya ketidaktepatan di dalam men*collision detection* untuk bentuk-bentuk yang tidak persegi empat.
2. Implementasi dapat membandingkan tiap piksel sebuah Sprite dengan Sprite lainnya. Jika sebuah piksel yang tidak transparan (opaque) dari sebuah Sprite beririsan dengan sebuah piksel yang tidak transparan (opaque) dari Sprite yang lain, collision terjadi. Teknik ini memerlukan komputasi yang lebih banyak, tetapi akan memberikan hasil yang lebih

akurat. Sprite memiliki collision rectangle, yang secara default terletak pada titik (0, 0) dengan lebar dan tinggi yang sama dengan obyek Sprite.

Collision rectangle memiliki dua peran sebagai berikut:

a. Jika *collision detection* berdasarkan piksel (pixel-level *collision detection*) tidak digunakan, collision rectangle digunakan sebagai area untuk mendeteksi collision.

b. Jika *collision detection* berdasarkan piksel (pixel-level *collision detection*) digunakan, maka hanya piksel di dalam collision rectangle yang diproses.

Collision detection adalah proses pendeteksian tabrakan antara dua objek. Sebenarnya dalam *game* sendiri tabrakan tidak hanya terjadi antara dua objek, tetapi dapat terjadi juga antara satu objek dengan banyak atau banyak objek dengan banyak objek. Dalam *game* membutuhkan *collision detection* yg akurat karena setelah menentukan terjadinya collision atau tidak kita harus menentukan apa yang terjadi pada objek yang ditabrak ataupun yang menabrak.

Collision detection ini juga berguna untuk menentukan posisi dari satu objek dengan objek yang lain sehingga tidak ada objek yang saling menembus. Sehingga *game* yang akan dibuat memiliki kesamaan dengan realita yang ada.

Collision detection memiliki teknik yg berbeda untuk diterapkan pada pembuatan *game* 2D dan *game* 3D.

Teknik *Collision detection* pada 2D ada 2 cara yaitu :

a. Rectangles *collision detection*
b. *Circle* rectangle *collision detection*

Teknik *Collision detection* pada 3D ada 2 cara yaitu :

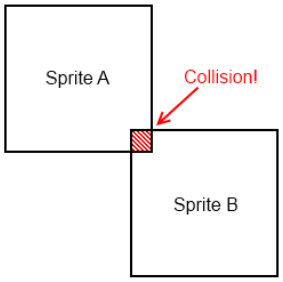
a. Bounding box *collision detection*
b. Sphere *collision detection*

2D *Collision detection*

Untuk *game* 2D setiap objek yang di perhatikan adalah vector 2Dnya, yaitu panjang dan lebar. Namun tidak setiap objek berbentuk persegi, lingkaran, segilima atau bentuk lain yang beraturan. Seperti bentuk pohon atau kuda dan lainnya, sehingga *collision detection* untuk objek2 seperti ini memerlukan pandekatan lain yang bisa mempermudah terutama dalam hal pemrograman. Karena untuk bentuk-bentuk yang tidak beraturan sangat sulit untuk dideteksi.

1. Rectangles *collision detection*

   Teknik ini merupakan pendeteksian tabrakan menggukan bound berbentuk persegi, dimana setiap sprite atau objek yang dibuat dalam *game* memiliki bound masing-masing. Cara ini cukup mudah karena hanya perlu membandingkan apakah ada intersection(irisan) pada kedua objek pada kordinat tertentu. Namun cara ini kurang akurat karena untuk sprite atau objek yang memiliki ruang kosong yang besar dengan boundnya. Hal ini akan muncul saat kondisi kedua batas objek tersebut beririsan walaupun sebenarnya tidak terjadi tabrakan maka akan dinggap terjadi tabrakan. Cara ini cocok untuk *game* yang tidak memperhatikan akurasi deteksi tabrakan atau untuk *game* yang menggunakan sprite yang bentuknya mendekati persegi.



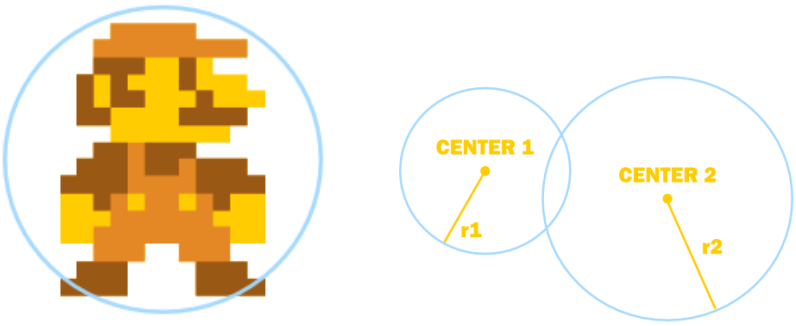
Gambar 2.8 Contoh Rectangles *collision detection*

Adapun cara lain dari hasil modifikasi teknik ini yaitu reduced size bounding box. Dimana kotak batasnya diperkecil dengan mengurangi beberapa pixel dari ukuran sprite atau objek yang sebenarnya. Sehingga dengan cara ini pada saat melakukan *collision detection* akan lebih akurat karena ruang kosong antara sprite dan bound berkurang.

Kemudian cara lain dari modifikasi teknik ini adalah dengan cara menggunakan beberapa bounding box secara bersamaan untuk satu sprite atau objek. Contohnya untuk sprite berbentuk manusia, pertama bounding box digunakan untuk bagian kepala kemudian bounding box yang lainnya digunakan untuk badan dan kaki. Cara ini lebih akurat karena banyaknya bounding box yang digunakan dapat disesuaikan dengan bentuk gambar pada sprite. Namun semakin banyak bounding box yang digunakan akan semakin banyak pula waktu yang digunakan untuk *collision detection*.

2. *Circle rectangle collision detection*

Untuk teknik ini memiliki cara yang hampir sama dengan yang sebelumnya hanya saja disini bound atau batas yang digunakan berbentuk lingkaran. Cara pengecekannya lebih mudah karena lingkaran memiliki jarak yg sama ke batas luarnya. Caranya yaitu dengan menghitung jarak kedua objek dan membandingkannya dengan jumlah radius masing-masing bound dari kedua objek, jika jaraknya lebih kecil dari jumlah radius kedua objek maka terjadi tabrakan (collision) dan jika lebih besar maka tidak terjadi tabrakan. Cara sebelumnya yaitu rectangle *collision detection* lebih akurat daripada cara ini.



Gambar 2.9 *Circle Rectangle Collision detection*