BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Perancangan atau desain didefinisikan sebagai proses aplikasi berbagai teknik dan prinsip bagi tujuan pendefinisian suatu perangkat, suatu proses atau sistem dalam detail yang memadai untuk memungkinkan realisasi fisiknya (Nataniel Dengen, Heliza Rahmania Hatta ;48:2009).

Menurut Robert J. Verzello/John Reuter III (sumber : Analisa dan Desain Sistem Informasi, Jogiyanto HM, hal 196), perancangan sistem adalah tahap analisis dari siklus pengembangan sistem, pendefenisian dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi, menggambarkan bagaimana suatu sistem dibentuk.

II.2. Multimedia

Multimedia dapat dikatakan suatu bentuk baru dalam pembuatan program-program komputer dengan penggambungan lebih dari suatu media. Meskipun hanya mengandung sedikitnya dua elemen, sudah dikatakan sebagai multimedia. Pengertian multimedia menurut Rosch: “Multimedia adalah kombinasi dari komputer dan video”; Adapaun pengertian menurut Mc Cornick: “Multimedia secara umum merupakan kombinasi tiga elemen, yaitu suara, gambar dan teks”; Menurut Turban dkk: “Multimedia adalah kombinasi dari paling sedikit dua media *input* atau *output* dari data, media ini dapat audio (suara, musik), animasi, video, teks, grafik dan gambar”Menurut Robin dan Linda: “Multimedia

merupakan alat yang dapat menciptakan presentasi yang dinamis dan interaktif yang mengkombinasikan teks, grafik, animasi, audio, dan gambar video" .

Dengan demikian multimedia dapat diartikan sebagai pemanfaatan komputer untuk membuat dan menggabungkan teks, grafik, audio, gambar bergerak (video dan animasi) dengan menggabungkan *link* dan *tool* yang memungkinkan pemakai melakukan navigasi, berinteraksi, berkreasi dan berkomunikasi. Dalam definisi ini terkandung empat komponen penting multimedia yaitu: 1) Harus ada komputer yang mengkordinasikan apa yang dilihat dan didengar, yang berinteraksi dengan kita; 2) Harus ada *link* yang menghubungkan kita dengan informasi; 3) Harus ada alat navigasi yang memandu kita; 4) Multimedia menyediakan tempat kepada kita untuk mengumpulkan, memproses, dan mengomunikasikan informasi dan ide kita sendiri (Septiana Firdaus,dkk ; 2 : 2012, Jurnal pendidikan vokasi , vol 3, Nomor 1, Februari 2013).

II.3. Permainan *Wormgame*

Permainan *Wormgame* pada Ponsel mempunyai 9 level yang akan mempengaruhi perubahan kecepatan pergerakan pada tubuh cacing. Makanan yang diperoleh merupakan penambahan perolehan nilai, semakin banyak makanan yang didapat maka semakin banyak nilai yang diperoleh oleh pemain.

Pemain yang memperoleh nilai tertinggi dinyatakan sebagai pemenang dalam permainan. Permainan dinyatakan berakhir apabila cacing menabrak dinding pembatas arena pencarian makanan atau menabrak dirinya sendiri. Pesan yang akan ditampilkan setelah permainan berakhir.

Perangkat lunak *Game* adalah sebuah permainan semu pada layar yang sangat membutuhkan ketelitian didalam membuat maupun memainkannya. *Game* dapat dimainkan pada berbagai media yang mempunyai layar monitor (tampilan). Media yang dapat dipergunakan seperti komputer, ponsel, *playstation*, maupun TV. *Game* ada beberapa jenis seperti pesawat tempur, *puzzle*, balapan, dan yang lainnya. Permainan tersebut dapat dimainkan oleh 2 macam pengguna yaitu *Single-user* (seorang pengguna) dan *Multi-user* (banyak pengguna).

Teknik permainan pada *wormgame* yaitu terdiri dari 3 kriteria, diantaranya adalah sebagai berikut :

1. Pemain melawan media
2. Pemain melawan pemain lainnya
3. Media melawan media

II.4. Elemen dalam *Game*

Untuk mewujudkan suatu *Game* yang bermutu dan disukai, maka perlu diperhatikan beberapa elemen dalam *Game* yang sangat mempengaruhi dalam hal jenis permainannya yaitu :

1. Desain *Game*
2. Pemrograman *Game*
3. Grafis *Game*

II.5. Desain *Game*

Desain *Game* merupakan ide cerita dari suatu *Game*. Desain *Game* tidak sama dengan pemrograman *Game*, karena desain *Game* cakupannya lebih luas

dari pemrograman. Dimana isinya merupakan ide cerita, rancangan level-level dan nilai yang diberikan.

Dengan kata lain desain *Game* adalah langkah awal untuk membuat semua elemen *Game*, dan perlu diketahui pula bahwa desain *Game* yang kurang lengkap akan membuat pemain cepat bosan dan dampaknya dapat membuat *Game* tersebut cepat ditinggalkan.

II.6. Pemrograman *Game*

Pemograman *Game* merupakan implementasi dari desain yang menggabungkan semua elemen-elemen lain untuk dibentuk menjadi sebuah permainan. Perlu diketahui juga bahwa pemrograman *Game* sebagian besar digunakan untuk mengontrol gerakan pada layar, pemrograman suara, deteksi tubrukan, dan sebagainya. Dimana pertama kali program dijalankan akan melakukan proses *looping* dengan proses penting antara lain kecerdasan objek, gerakan objek, *Input divice* dan pemberian kondisi pilihan. Pada proses kecerdasan objek dirancang animasi yang dilakukan oleh komputer dengan cara *random* (acak), sedangkan pada gerakan objek akan dirancang aturan dari gerakan dan pada *Input divice* akan dideteksi tombol-tombol yang akan digunakan.

II.7 Grafis *Game*

Grafis *Game* ponsel merupakan elemen *Game* yang cukup penting karena grafis *Game* merupakan tampilan luar dari *Game* tersebut. Grafis *Game* haruslah dibuat semenarik mungkin sehingga dengan melihatnya saja orang jadi tertarik untuk mencoba memainkannya. Proses pembuatan grafis ini biasanya ditangani

oleh seorang desainer grafis yang mengkhususkan diri dalam pembuatan grafis komputer, bukan oleh seorang pemogram karena sangatlah jarang pemogram mencakup juga sebagai desainer grafis, dimana dalam merancang grafis dibutuhkan bakat dan pengalaman.

Untuk pembuatan grafis *Game*, desainer dapat menggunakan program-program pengelola grafis seperti *corel draw* dan yang lainnya. Penggunaan program *Delux* Paint buatan *Electronics Arts* yang merupakan perangkat lunak khusus untuk pembuatan grafis *Game*, akan tetapi keputusan menggunakan perangkat lunak tersebut kembali kepada yang akan menggunakannya.

II.8. Langkah-langkah Membuat Permainan *Game* Secara Umum

Langkah-langkah yang harus dipertimbangkan dalam pembuatan sebuah permainan adalah mencakup teori dalam merencanakan, merancang dan mengembangkan permainan yang akan dibuat.

Secara umum langkah-langkah dalam membuat permainan dapat di simpulkan sebagai berikut :

1. Tentukan Permainan Apa Yang Akan Dibuat

   Tentukan tipe permainan macam apa yang akan dibuat, penentuan ini merupakan dasar sebelum mulai bekerja. Dapat meniru *Game-Game* yang telah ada atau mengkhayal sampai menemukan ide yang kira-kira cocok dan bagus untuk dibuat suatu program permainan.

2. Defenisikan Model Permainan dan Tujuannya

   Ide yang telah di dapat dituangkan dalam bentuk model permainan yang ingin dibuat.

3. Sesuaikan Dengan Perangkat Keras dan Piranti Lunak Komputer

   Pada saat sekarang penyesuaian ini sangatlah diperlukan sekali karena perkembangan perangkat keras dan perangkat lunak sedemikian pesatnya.

4. Defenisikan Secara Jelas *Game Worlds*-nya

   Yang dimaksud dengan *Game* worlds-nya adalah elemen-elemen utama yang terdapat dalam suatu program permainan yang dibuat sesungguhnya.

II.9. Pemrograman GUI (*Graphical User Interface*) Pada *Midlet.*

Berbeda dengan komputer, perangkat *handheld* semacam ponsel, *Palm* dan *organizer* memiliki keterbatasan misalnya ukuran layar yang jauh lebih kecil dan masukan *Input* yang tidak berupa *mouse*. Keterbatasan-keterbatasan ini menyebabkan perlunya ada teknik pemrograman GUI yang berbeda dengan teknik pemrograman yang umumnya digunakan pada aplikasi-aplikasi yang dijalankan pada komputer. CLDC (*Connected Limited Divice Connection*) tidak menyediakan fungsi-fungsi untuk GUI, namun fungsi-fungsi ini akan ditangani oleh MIDP (*Mobile Information Divice Profile*).

Jika melakukan pemograman GUI dengan Java, maka pemograman GUI yang memiliki kelas-kelas *window, dialogbox, messagebox,* dan komponen lainnya diturunkan dari kelas AWT (*abstract windowing toolkit)* atau *swing.* GUI pada MIDP tidak didasarkan pada AWT atau kelas-kelas *swing* tersebut dikarenakan beberapa faktor :

1. AWT dan *swing* (dirancang untuk pemrograman GUI pada dekstop computer).
2. AWT (mengasumsikan adanya beberapa interaksi antara pengguna dengan

komputer dengan beberapa perangkat *Input* semacam *mouse* padahal umumnya perangkat *handheld* semacam ponsel hanya memiliki *keypad).*

3. Implementasi AWT dan *swing* (membutuhkan pemrosesan CPU dan memori yang cukup besar , dan hal ini tidak cocok pada perangkat *handheld* yang umumnya memiliki memori yang sedikit).

II.10 Java

Java adalah suatu teknologi di dunia *software* komputer, yang merupakan suatu bahasa pemrograman, dan sekaligus suatu *platform*. Sebagai bahasa pemrograman, Java dikenal sebagai bahasa pemrograman tingkat tinggi. Java mudah dipelajari, terutama bagi programmer yang telah mengenal C/C++. Java merupakan bahasa pemrograman berorientasi objek yang merupakan paradigma pemrograman masa depan. Sebagai bahasa pemrograman Java dirancang menjadi handal dan aman. Java juga dirancang agar dapat dijalankan di semua *platform*, dan juga dirancang untuk menghasilkan aplikasi-aplikasi dengan performansi yang terbaik, seperti aplikasi *database* Oracle 8i/9i yang *core*-nya dibangun menggunakan bahasa pemrograman Java. Sedangkan Java bersifat *neutral architecture*, karena Java Compiler yang digunakan untuk mengkompilasi kode program Java dirancang untuk menghasilkan kode yang netral terhadap semua arsitektur perangkat keras.

II.11. *Unified Modeling Laguange* (UML)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

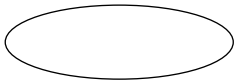
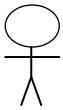
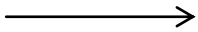
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.11.1 *Usecase* Diagram

*Usecase*diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *usecase* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *usecase* diagram, yaitu :

Tabel II.1. Simbol Use Case Diagram

Gambar	Keterangan
	*Usecase* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *usecase*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *usecase*, tetapi tidak memiliki control terhadap *usecase*.
	Asosiasi antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *usecase* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor

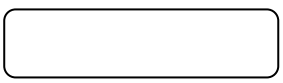
	berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *usecase* lain jika kondisi atau syarat terpenuhi.

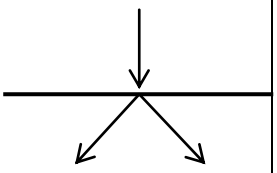
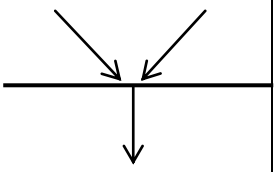
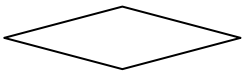
(Sumber : WinduGata ; 2013 : 4)

II.11.2 Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu :

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*Endpoint*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.

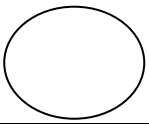
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

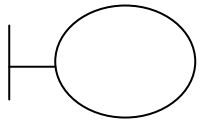
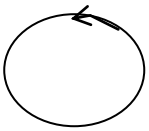
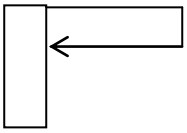
(Sumber : Windu Gata ; 2013 : 6)

II.11.3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *usecase* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang

	membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*BoundaryClass,* berisi kumpulan kelas yang menjadi *interface*atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form*cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation*mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline*terdapat *activation*.

(Sumber : Windu Gata ; 2013 : 7)

II.11.4. *Class Diagram* (Diagram Kelas)

Class diagram Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint*yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *kardinaliti*.

Tabel II.4. Simbol *Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata ; 2013 : 9)