BAB III

OBJEK DAN METODE PENELITIAN

3.1 Objek Penelitian

Objek penelitian adalah hal awal yang harus ditentukan dalam kegiatan penelitian sehingga penelitian dapat dilakukan secara efektif dan efisien sesuai dengan tujuan penelitian. Pemilihan dan penentuan objek penelitian yang tepat diharapkan dapat menunjang kegiatan selama penelitian, sehingga hal-hal yang diperlukan dalam penelitian akan mudah dicapai. Yang menjadi objek dalam penelitian ini adalah proses pembuatan aplikasi Game. Penyusun melakukan penelitian dari berbagai sumber.

3.2 Sejarah Game

Game berbasis multimedia awal mulanya berasal dari game komputer .Permainan komputer merupakan salah satu jenis media baru yang mulai ada sejak pertengahan kedua abad ke-20. Dibandingkan jenis-jenis media sebelumnya, permainan komputer menawarkan sebuah segi yang tidak diperoleh media lain, yakni interaksi antara pemakai dengan program permainan komputer yang menghasilkan keragaman stimulus berdasarkan input yang diberikan pemainnya. Bagaimana seorang pemain bermain, bertindak dan membuat pilihan-pilihan mempengaruhi keseluruhan pengalaman bermainnya tersebut. Permainan-permainan tersebut juga mempunyai kemampuan untuk mempresentasikan pemainnya dunia-dunia mikro dalam arti memberikan konteks-konteks untuk berpikir menghadapi permasalahan, mengambil tindakan sendiri sebagai bagian

solusinya dan membangun arti intuitif dari peran mereka dalam dunia permainan tersebut (Holland, Jenkins dan Squire, 2003).

3.3 Unsur – Unsur *Game*

3.3.1 *Sprites*

Sprites adalah kumpulan gambar-gambar yang digunakan untuk icon, karakter ataupun gambar bergerak dalam game. *Sprites* dapat dibuat dari gambar-gambar yang sudah dibuat dalam sebuah *art packages*, *download* dari internet maupun dibuat secara manual dengan menggunakan software seperti *Adobe Photoshop*.

3.3.2 *Objects*

Sprites tidak dapat melakukan sesuatu dengan sendirinya. *Sprites* hanya menyimpan gambar-gambar dari elemen-elemen yang berbeda dalam sebuah fame. Objects adalah bagian dari *game* yang mengontrol bagaimana elemen-elemen tersebut bergerak dan berinteraksi satu sama lain. Contohnya adalah karakter dalam game.

3.3.3 *Events dan Actions*

Events adalah hal-hal penting yang terjadi didalam sebuah *game*, seperti ketika objects saling bertumbukan maupun ketika user menekan tombol pada keyboard. *Actions* adalah hal-hal yang terjadi sebagai respon dari sebuah event, seperti perubahan arah pergerakan objects, perubahan *score*, atau memainkan sebuah lagu.

3.3.4 *Backgrounds*

Backgrounds adalah gambar –gambar yang digunakan sebagai gambar latar belakang game dan merupakan jenis lain dari resource, seperti sprites dan objects. Pada umumnya ukuran backgrounds sama dengan resolusi game.

3.3.5 *Sounds*

Sounds juga termasuk *resource game*, jenis *sounds* dalam game ada dua, yaitu musics dan *sound effect*. *Sounds* dangat mempengaruhi atmosfer dalam game. Masters berupa lagu yang terus mengalun selama game dimainkan, namun lagu yang dimainkan pada umumnya bergantung pada tempat atau kondisi game. *Sounds effect* adalah suara yang hanya terdengar ketika terjadi suatu hal tertentu, misal penekanan tombol, karakter terkena pukulan dan sebagainya. Kegunaan dari *sound effects* adalah untuk membantu user mengetahui *action* yang dilakukan maupun hal yang sedang terjadi.

3.3.6 *Turn Based*

Turn Based adalah sebuah sistem dalam game untuk menentukan giliran dimana hanya satu user yang dapat bergerak dalam suatu waktu tertentu. Contoh game antara lain Tic Tac Toe, Scrabble, Catur, Permainan Kartu dan lain-lain.

3.3.7 Sudut Pandang User

Sudut pandang user dalam game ada 2, yaitu :

1) *First Person View* adalah sudut pandang orang pertama, dimana user dijadikan seperti karakter yang dimainkan, sehingga user melihat dari sudut pandang karakter yang dimainkan. Contoh game : Counter Strike, HALO
2) *Third Person View* adalah sudut pandang orang ketiga, dimana user dapat melihat keseluruhan tubuh karakter dalam game. Contoh game : Ragnarok Online, FATE

3.4 Animasi Game

Animasi dapat dibuat dengan tiga teknik berbeda, yaitu *Image, Xoring* serta *make*. Dan dalam pergerakannya dapat bertipe *object sprite* atau *object frame*. Juga bisa dibedakan atas metode animasi yang digunakan antara animasi frame, *bibliting* dan *realtime*. Walaupun terbagi atas berbagai definisi berbeda, tapi dalam prakteknya teori-teori tersebut dapat digabungkan atau saling berhubungan sehingga tidak murni dipakai sendiri. Macam-macam animasi yang digunakan dalam membuat sebuah game akan diterangkan sebagai berikut :

3.4.1 Animasi Dengan Teknik *Image*

Animasi dengan teknik ini adalah menyimpan image sebagai sebuah *sprite* dalam memori yang kemudian akan ditampilkan di backgroundnya. Dalam teknik ini animasi yang disimpan harus berlatar belakang sesuai backgroundnya. Animasi dengan teknik ini biasanya sulit dalam pembuatan gambarnya, sebab harus banyak dan melakukan penyamaan dan posisi. Akan tetapi teknik ini mudah dalam hal memainkan animasinya.

3.4.2 Animasi Dengan Teknik *Xoring*

Teknik ini adalah teknik animasi yang mudah dan sederhana, sebab selain gambarnya satu *sprite*, cara menampilkannya juga jauh lebih mudah dibanding dengan teknik sebelumnya. Pembuatan gambarnya sangat mudah, sebab yang dibuat adalah spritenya saja dan tidak perlu menyamakan dengan backgroundnya. Kelemahan dari teknik ini adalah memiliki efek buruk yaitu tembus pandang dan mengganti warna *sprite*, maka hal ini tidak baik digunakan dalam animasi yang backgroundnya bergambar.

3.4.3 Animasi Dengan Teknik *Make*

Animasi dengan teknik ini biasanya digunakan untuk animasi umum, tapi biasanya digunakan untuk proses pembuatan animasi. Animasi dengan teknik ini memiliki sprite yang terus menerus digenerate oleh program, kemudian ditampilkan dengan perhitungan tertentu.

Animasi ini biasanya dilakukan oleh 3D *modelling* dan *shading software* seperti AutoCad, 3D Studio, Presidio 3D Workshop dan lain-lain. Operasi diatas haruslah *digenerate* secara langsung dengan perhitungan sehingga saat pembuatan hampir bersamaan dengan saat menampilkannya

3.4.4 Animasi Dengan Tipe Object Frame

Animasi ini menitik beratkan animasi yang dimainkan hanya pada sprite objectnya saja, akan tetapi seluruh backgroundnya juga seolah-olah ikut

digerakkan.

3.4.5 Metode Animasi frame

Metode ini adalah metode animasi yang mendukung tipe object frame. Karena metode animasi dengan metode full-screen, maka frame yang tampil haruslah disiapkan terlebih dahulu dalam bebera page sebelumnya. Karena hal tersebut maka pengambilan gambarnya haruslah sangat cepat, sehingga tidak menjadikan animasi lamban dan tersendat. Animasi frame ini haruslah menampilkan gambar fullscreen yang bergerak, agar efek tersendat dari pergantian frame tidak menyolok.

3.4.6 Metode Animasi *BitBlt*

Metode animasi ini biasanya disebut sprite animation, array animation, blocked animation, partial screen animation, snapshot animation atau arcade animation. Prinsip dari metode ini adalah menyimpan image dan memainkan animasinya dalam bentuk satu atau bebera sprite kecil. BitBlt ini digunakan untuk membuat program permainan dan program berbasis grafis terutama dalam membuat animasi sederhana.

3.4.7 Metode Animasi *Real-Time*

Dalam metode ini biasanya semua animasi yang sedang tampil atau yang akan dibuat dilakukan bersama sehingga tidak perlu disiapkan terlebih dahulu. Karena animasi ini lambat dan tersendat maka animasi dengan metode ini akan bagus jika pergerakan yang akan dilakukan adalah tidak diketahui sebelumnya dan tiba-tiba muncul.Metode ini tidak disarankan untuk animasi biasa-biasa saja,

akan tetapi sebaiknya digunakan untuk keperluan khusus seperti rotating dan tweening

3.5 Metode Penelitian

3.5.1 Metode Pendekatan

Metode pendekatan sistem yang di gunakan adalah pendekatan dengan *Object Oriented* yang menggunakan AOO (Analisis *Object Oriented*) dan DOO (*Design Object Oriented*) yang di visualisasikan dengan UML dan di antara nya adalah sebagai berikut : Use Case, Sequence Diagram, Class Diagram, Collaboration Diagram, Component Diagram dan Deployment Diagram.

3.5.2 Metode Pengembangan Sistem

Metode pengembangan sistem yang akan digunakan adalah model prototype, karena model tersebut lebih memperhatikan kebutuhan sistem pemakai, prototype memberikan ide bagi pembuat maupun potensial tentang cara sistem berfungsi dalam bentuk lengkapnya, proses menghasilkan sebuah prototype disebut prototyping.

1. Pengumpulan Kebutuhan

   Pelanggan dan pengembang bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.
2. Membangun Prototyping

Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan format output).

3. Evaluasi Prototyping

   Evaluasi ini dilakukan oleh pelanggan apakah prototyping yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak prototyping direvisi dengan mengulangu langkah 1, 2 , dan 3.

4. Mengkodekan Sistem

   Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5. Menguji Sistem

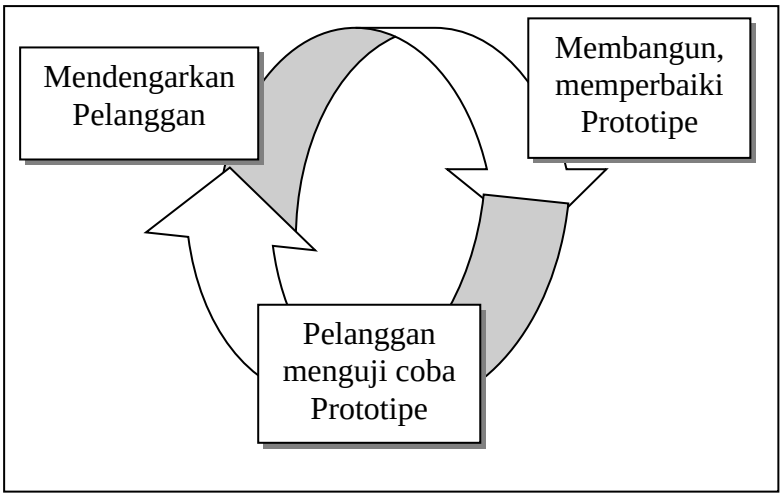
   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan. Pengujian ini dilakukan dengan White Box dan Black Box.

6. Evaluasi Sistem

   Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika ya, langkah 7 dilakukan jika tidak, ulangi langkah 4dan5.

7. Menggunakan Sistem

   Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.



Gambar 3.1. Metode Pendekatan Prototype Paradigma

(**Sumber**: Pressman, Roger S., 2002, *Rekayasa Perangkat Lunak*: Pendekatan Praktisi Jilid Dua, Penerbit: Andi Offset, Yogyakarta).

3.5.3 Alat Bantu Analisis Dan Perancangan

Dalam pembuatan aplikasi, peneliti menggunakan alat bantu analisis dan perancangan yaitu sebagai berikut :

1. Usecase Diagram.

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah usecase 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 usecase dapat meng-include fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang diinclude 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.

2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

a. Nama (dan stereotype)

b. Atribut

c. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. Private, tidak dapat dipanggil dari luar class yang bersangkutan

b. Protected, hanya dapat dipanggil oleh class yang bersangkutan dananak-anak yang mewarisinya

c. Public, dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah interface, yaitu class abstrak yang hanya memiliki metoda. Interface tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah class. Dengan demikian interface mendukung resolusi metoda pada saat run-time. Sesuai dengan perkembangan class model, class dapat dikelompokkan menjadi package. Kita juga dapat membuat diagram yang terdiri atas package.

Hubungan Antar Class

a. Asosiasi, yaitu hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah navigability menunjukkan arah query antar class.

b. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).

c. Pewarisan, yaitu hubungan hirarkis antar class. Class dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

d. Hubungan dinamis, yaitu rangkaian pesan (message) yang di-passing dari satu class kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.

3. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri antara dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki lifeline vertikal. Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari class. Activation bar menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek boundary, controller dan persistent entity.

4. Activity Diagram

Diagram ini adalah jenis spesial dari *statechart* diagram yang menunjukkan alur dari aktifitas ke aktifitas lain dalam suatu sistem. *Activity* diagram mengalamatkan secara dinamis suatu sistem. Diagram ini sangat penting dalam membuat model fungsi dari suatu sistem dan menekankan pada alur control diantara objek - objek yang bersangkutan.

3.6 Pengujian Software

3.6.1 Black box Testing

Pengujian adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan-perbedaan antara hasil yang diharapkan dengan yang terjadi. Pengujian ditujukan untuk menghasilkan perangkat lunak *(software)* yang bebas kesalahan, paling tidak secara teknik.

Pengujian black box adalah pengujian aspek *fundamental* sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pada pengujian *black-box*, kasus-kasus pengujian berdasarkan pada spesifikasi sistem. Rencana pengujian dapat dimulai sedini mungkin di proses pengembangan perangkat lunak.

Pada pengujian *black box*, mencoba beragam masukan dan memeriksa keluaran yang dihasilkan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain. Adapun faktor-faktor pengujian *black-box* adalah :

1. *File integrity*

Menekankan pada data yang dimasukkan melalui aplikasi akan tidak biasa diubah. Prosedur yang akan memastikan bahwa file yang digunakan benar dan data dalam file tersebut akan disimpan sekuensial dan benar.

2. *Service levels*

Menekankan bahwa hasil yang diinginkan didapat dalam waktu yang diinginkan oleh *user*. Untuk mencapai keinginan tersebut, harus dilakukan penyesuaian antara keinginan *user* dengan sumber daya yang ada.

3. *Ease of use*

Menekankan perluasan usaha yang diminta untuk belajar, mengoprasikan dan menyiapakan inputan, dan menginterpretasikan output dari sistem. Faktor ini tersangkut dengan *usability* system terhadap interaksi antara manusia dan system.

4. *Authorization*

Menjamin data diproses sesuai dengan ketentuan manajemen. *Authorization* menyangkut proses transaksi secara umum dan khusus. Fokus Pengujian *Black box testing* yaitu sebagai berikut :

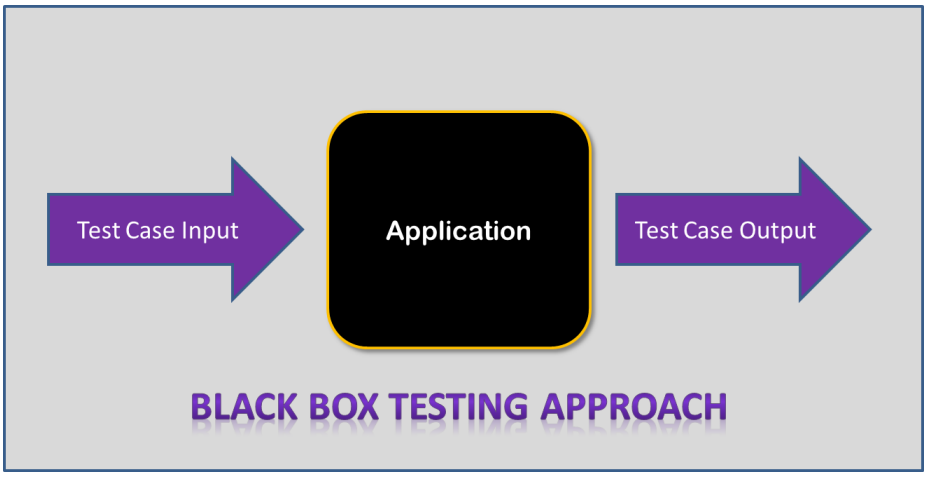
1. Menguji fungsi-fungsi khusus dari aplikasi.
2. *Test input* dan *output* untuk fungsi yang ada tanpa memperhatikan prosesnya.

Beberapa jenis kesalahan yang dapat di identifikasi :

1. Fungsi tidak benar atau hilang,
2. Kesalahan antar muka,
3. Kesalahan pada struktur data (pengaksesan basis data),

4. Kesalahan inisialisasi dan akhir program, dan

5. Kesalahan performasi.



Gambar 3.2. Pengujian Black Box Testing

(**Sumber**: http://blog.testing-whiz.com/12-2-2013).