BAB 2

LANDASAN TEORI

2.1 Pengertian Game

Game berasal dari kata bahasa inggris yang berarti permainan. Permainan melibatkan kecerdasan intelektual pemainnya dengan berbagai rintangan dan ada terget yang harus dicapai. Permainan dapat dilakukan dengan cara bermain di lapangan atau dengan bermain *video games*.

Secara umum, *video game* merupakan sarana hiburan interaktif yang menyenangkan dan dapat merangsang daya pikir maupun syaraf motorik pemain. Berbagai jenis teknik bermain berdasarkan jenis *game* nya diantaranya ialah dengan menggunakan *Joystick*, *keyboard*, dan yang paling mutakhir saat ini adalah dengan *touchscreen*.

Menurut Agustinus Nilwan dalam bukunya "Pemrograman Animasi dan Game Profesional" terbitan Elex Media Komputindo, mengungkapkan bahwa *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan *game* [5]. Berdasarkan penjelasan diatas, dapat disimpulkan bahwa video game memiliki unsur animasi dan teknik pembuatannya serta logika untuk dapat membuatnya.

2.1.1 Jenis - Jenis Game

Berdasarkan *platform* yang digunakan, jenis *game* dapat dikategorikan sebagai berikut [6]:

1. *Arcade games*

   Arcade games sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki *box* atau mesin yang memang khusus

di design untuk jenis video *game*s tertentu seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan setir mobil.

2. *PC Games* , yaitu video *game* yang dimainkan menggunakan komputer.
3. C*onsole games*, yaitu *video game*s yang dimainkan menggunakan *console* tertentu, seperti *Playstation 2*, *Playstation 3, XBOX 360,* dan *Nintendo Wii*.
4. *Handheld games*, yaitu yang dimainkan di *console* khusus yang dapat dibawa kemana-mana, contoh *Nintendo DS* dan *Sony PSP*.
5. *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA.

2.1.2 Genre Game

Selain jenis – jenis, *game* juga terbagi oleh berbagai genre. Berdasarkan karakteristiknya, terdapat banyak genre yang tersedia. Beberapa diantaranya [6] :

1. **Action**

Karakteristik dari *game* jenis ini adalah melawan atau menghidari musuh, dapat dengan tembak menembak maupun dengan berkelahi. Contoh *game* jenis ini adalah *game Contra*.

Gambar 2.1 Tampilan *Game Contra*

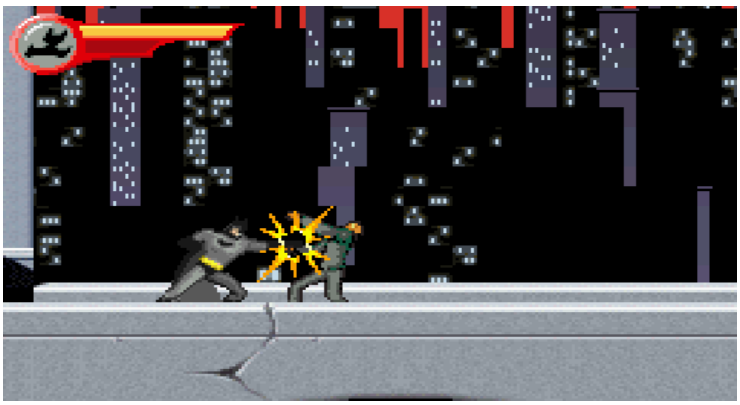
2. **Fighting (Pertarungan)**

Karakteristik dari *game* jenis ini adalah pertarungan atau perkelahian antar dua pemain didalam *game*. Ciri khas dari *game* jenis ini adalah satu lawan satu. Contohnya adalah *Street Fighter*.

Gambar 2.2 Tampilan *Game Street Fighter*

3. **Aksi – Petualangan**

Karakteristik *game* jenis ini hampir sama dengan *action*, namun bedanya disini adalah berlanjutnya aksi pemain dalam menjalankan misi. Biasanya ada sistem *check point* untuk menentukan tujuan terakhir sebelum melangkah lebih jauh lagi. Contoh *game* nya adalah *Batman rise of Sin Tzu, Crash Bandicoot adventure*.



Gambar 2.3 Tampilan *Game Batman Rise of Sin Tzu*

4. ***Puzzle***

Karakteristik *game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula *game* jenis ini adalah juga unsur permainan dalam petualangan. Contohnya adalah *Tetris, Adventure of lolo*.

Gambar 2.4 Tampilan *Game Adventure of lolo*

2.2 Game Berbasis Web

Game berbasis *web* adalah salah satu media permainan dimana dapat dimainkan melalui *web browser*. Terdapat sangat banyak *game online* yang tersedia di jaringan internet, mulai dari *game* 2 Dimensi hingga *game* 3 Dimensi yang rumit dan bergrafis tinggi.

Game berbasis *web* dapat diakses dari mana pun pemain mau. Asal ada internet, *game* tersebut dapat berjalan dengan baik dan tidak perlu adanya instalasi. *Harddisk* tempat penyimpanan pemain *game* pun tidak akan menjadi penuh. Semua itu menjadi mungkin dengan adanya internet [7].

2.3 Pemograman Berorientasi Objek (PBO)

Dalam membangun sebuah *game*, ada dua cara atau pola bahasa pemrograman, yaitu terstuktur dan berorientasi objek. Namun yang diterapkan di skripsi ini adalah pemrograman berorientasi objek.

PBO atau dikenal dengan pemograman berorientasi objek merupakan bahasa pemograman yang semua data dan fugsi didalam paradigma ini dibungkus ke dalam kelas-kelas atau objek-objek.

Model data berorientasi objek dikatakan dapat memberi feksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar. Lebih jauh lagi, pendukung PBO mengklaim bahwa PBO lebih mudah dipelajari dibandingkan dengan pendekatan sebelumnya, dan pendekatan PBO lebih mudah dikembangkan dan dirawat [8].

Dengan menggunakan PBO maka dalam melakukan pemecahan suatu masalah, tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut. Pemograman orientasi objek menekankan konsep berikut [8] :

1. Kelas (*Class*) adalah kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemograman berorientasi objek.
2. Objek (*Object*) membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah progam komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. Abstraksi (*Abstract*) adalah kemampuan sebuah untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "Pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan.
4. Enkapsulasi (*Encapsulation*) adalah Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak, hanya metode dalam objek tersebut yang diberi ijin untuk mengakses keadaannya.

5. Polimorfisme (*Polimorfism*) melalui pengiriman pesan. Tidak tergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan, metode tertentu yang dapat berhubungan dengan sebuah pengiriman pesan tergantung kepada onjek tertentu dimana pesan tersebut dikirim.
6. Inheritas (*Inheritance*) adalah mengatur polimorfisme dan enkapsulasi dengan mengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada. Objek-objek ini dapat membagi (dan memperluas) perilaku mereka tanpa harus mengimplementasi ulang perilaku tersebut.

2.4 Algoritma A*

Algoritma A* merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search* dan *Greedy Best First Search*. Algoritma ini sering digunakan didalam pembuatan *game* untuk pencarian jalur. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai berikut :

$$f(n) = g(n) + h(n) \qquad ...(2.1)$$

Dengan keterangan sebagai berikut :

$f(n)$ = hasil dari perhitungan $g(n) + h(n)$

$g(n)$ = biaya pergerakan yang diberikan dari kondisi awal hingga kondisi n

$h(n)$ = perkiraan biaya dari posisi awal hingga tujuan

Heuristic adalah penilai yang memberi harga pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan *Heuristic* yang benar, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya) yang dicari. *Heuristic* yang digunakan adalah *Euclidean distance*, dengan rumus perhitungan yang dituliskan sebagai berikut :

$$h(n) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} \quad ...(2.2)$$

Beberapa terminologi dasar yang terdapat pada algoritma A* yaitu [9]:

1. Simpul (*node*) adalah petak-petak kecil sebagai representasi dari area pencarian (*pathfinding*). Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.
2. Simpul asal / mulai (source *node*) adalah sebuah terminologi untuk posisi awal sebuah benda.
3. Simpul akhir / tujuan (destination *node*) adalah tempat tujuan yang ingin dicapai pada pencarian.
4. Simpul sekarang (current *node*) adalah simpul terbaik sebelumnya yang dipilih dan menjadi titik acuan untuk membangkitkan simpul tetangganya.
5. Open list adalah tempat menyimpan data simpul yang diakses dari simpul asal / mulai (source *node*) atau dari tetangga simpul sekarang (current *node*) yang belum pernah berada di open list maupun closed list.
6. Closed list adalah tempat menyimpan simpul yang pernah menjadi simpul sekarang (current *node*).
7. Halangan/penghalang adalah simpul yang tidak dapat dilalui.

Adapun langkah langkah yang dilakukan untuk mengimplementasikan algoritma ini seperti berikut [10] :

1. Identifikasi arena dengan menentukan lokasi implementasi, kemudian identifikasi posisi objek awal, posisi objek tujuan, dan posisi penghalang.
2. Tentukan objek yang menjadi *source node* atau simpul asal dan objek yang menjadi *destination node* atau simpul tujuan yang nantinya akan menentukan pergerakan dari algoritma.
3. Kemudian simpan *source node* ke dalam *open list*, lalu semua *neighbour node* atau simpul tetangga disekitarnya dibangkitkan sebagai suksesor atau calon penerus *node* sebelumnya. Jika semua suksesornya tidak ada di *open list* maupun di *closed list*, maka masukkan suksesornya ke *open list*.

4. Hitung nilai *f(n)* dari setiap suksesor kemudian pilih suksesor dengan nilai *f(n)* terkecil yang kemudian *current node* yang telah ditentukan menjadi *parent* yang baru. Karena *parent* baru telah ditentukan, maka *current node* sebelumnya ditutup atau dimasukkan kedalam *closed list*.
5. Ulangi langkah ke tiga dan empat hingga mencapai *destination node* dengan mengabaikan penghalang dan *node* yang sudah berada di *closed list*.

2.5 Tools Yang Digunakan

Terdapat beberapa tools yang digunakan untuk merancang serta membangun *game*, diantaranya adalah *Unified Modeling Language* (UML) dengan *Star* UML, *Quintus, Brackets*, serta *Tiled*.

2.5.1 Unified Modeling Language (UML)

Adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu [11] :

1. *Use Case Diagram*, untuk memodelkan proses bisnis.
2. *Conceptual Diagram*, untuk memodelkan konsep – konsep yang ada didalam aplikasi.
3. *Sequence Diagram* , untuk memodelkan pengiriman pesan (*message*) antar objek.
4. *Collaboration Diagram*, untuk memodelkan interaksi antar objek.
5. *State Diagram*, untuk memodelkan perilaku objek didalam sistem.
6. *Activity Diagram*, untuk memodelkan perilaku *use cases* dan objek didalam sistem.
7. *Class Diagram*, untuk memodelkan struktur kelas.
8. *Object Diagram*, untuk memodelkan struktur objek.
9. *Component Diagram*, untuk memodelkan komponen objek.
10. *Deployment Diagram*, untuk memodelkan distribusi aplikasi.

2.5.2 *Star UML*

Star UML adalah software permodelan yang mendukung UML *(Unified Modeling Language).* Berdasarkan pada UML version 1.4 dan dilengkapi 11 macam diagram yang berbeda, mendukung notasi UML 2.0 dan juga mendukung pendekatan MDA *(Model Driven Architecture)* dengan dukungan konsep UML. *Star UML* dapat memaksimalkan pruduktivitas dan kualitas dari suatu *software project.*

Konsep Dasar pada Star UML :

1. *Model, View and Diagram*

   Star UML membuat perbedaan konseptual yang lebih jelas antara *models, views and diagrams.* Model adalah elemen yang memuat informasi untuk model software. *View* adalah suatu ekspresi visual dari informasi di dalam model dan Diagram adalah suatu koleksi dari elemen yang memberikan pemikiran *user* di dalam mendesain secara spesifik.

2. *Project and Unit*

   Project adalah *unit* manajemen dasar di dalam *Star UML*. Suatu *projec*t dapat mengatur satu atau lebih model *software*. *Project* merupakan *top-level package* yang selalu ada di dalam beberapa model *software*. Secara umum, satu *project* disimpan dalam satu *file*. *File project* disimpan ke dalam format XML dengan *extension* ".UML". Semua model, *views* dan *diagrams* yang dibuat dengan *Star UML* disimpan dalam satu *file project*. *File project* berisikan informasi sebagai berikut :

a. *UML profile* yang digunakan dalam projek.
b. *Unit file* yang direferensi oleh projek.
c. Informasi untuk semua model yang ada di dalam projek.
d. Informasi untuk semua *diagrams* dan *views* yang ada di dalam *project*. Ada beberapa kasus dimana satu *project* perlu disimpan di dalam beberapa *file – file* kecil sehingga para pengembang dapat bekerja di dalam satu *project* secara bersamaan. Di dalam kasus ini suatu *project* dapat mengatur

bermacam – macam *unit*. Suatu *unit* mempunyai struktur hirarki dan berisikan beberapa *sub-unit*. *Unit* disimpan sebagai ".UML " *file* dan beberapa mengacu pada *file project* (.UML) atau *unit file* lainnya (.UNT). Komposisi *unit* hanya package, *subsystem* dan elemen model yang dapat membentuk satu unit. Semua elemen di bawah jenis elemen *package* ini disimpan di dalam masing – masing *file unit* (.UNT).

3. *Module*

   Modul adalah suatu *package* yang menyediakan fungsi – fungsi baru dan fitur sebagai perluasan dari *Star UML*. Modul dapat dibuat sebagai kombinasi dari beberapa elemen – elemen *extension* dan juga membuat beberapa jenis elemen – elemen di dalam suatu modul [12].

2.5.3 Quintus

Quintus adalah mesin pembangun *game* berbasis *HTML 5* yang didesain modular dan ringan, dengan *sintax JavaScript* yang ringan dan *friendly*. Sebagai pengganti dari struktur mesin *game* OOP standar ke mesin *HTML 5 JavaScript*, *Quintus* mengambil beberapa perintah *jQuery* dan menyediakan *plugin*, *event* dan *sintax* pemilih. Sebagai kebalikan dari sekedar model *single-inheritance*, *Quintus* menyediakan model komponen yang fleksibel disamping *inheritance* yang tradisional untuk membuatnya lebih mudah untuk digunakan kembali secara fungsional dan membagikannya keseluruh permainan dan objek [13].

2.5.4 Brackets

Brackets adalah aplikasi *web code editor* yang bersifat *open source* berbasis HTML 5 dan *JavaScript* yang digunakan untuk mendesain *web*, termasuk juga untuk *game* berbasis *web* [14].

2.5.5 Tiled

Tiled adalah aplikasi *map editor* yang dibuat oleh *Thorbjørn Lindeijer* pada tahun 2008. Fungsi dari aplikasi ini adalah dari membuat *background* hingga membuat peta. Terdapat tools yang dapat digunakan seperti *stamp brush, terrain brush, bucket fill tool, rectangular select* dan lain sebagainya yang

memungkinkan penggunanya dapat membuat *background* atau peta menjadi lebih mudah [15].

2.6 Metode pengujian Perangkat Lunak

Pengujian aplikasi bertujuan untuk mencari kesalahan (*bugs*) dalam sebuah aplikasi. Pengujian yang baik adalah pengujian yang memiliki kemungkinan besar dalam menemukan kesalahan. Oleh karena itu dalam merancang dan mengimplementasikan sistem, baik berbasis komputer atau produk harus ada pengujiannya. Menurut James Bach mengenai karakteristik kemampuan suatu aplikasi dapat diuji (*testability*) sebagai berikut : "kemampuan perangkat lunak untuk dapat diuji adalah seberapa mudahkah sebuah program computer untuk bisa diuji"[16]. Agar lebih mempermudah pemahaman menggenai kemampuan aplikasi dapat diuji, akan dijelaskan dengan mengenalkan tentang kemampuan sebuah perangkat lunak untuk bisa diuji seperti berikut :

1. Kemampuan untuk bisa dioperasikan *(operability).*
   Semakin baik kinerjanya, semakin efisien perangkat lunak untuk bisa diuji.
2. Kemampuan untuk bisa diobservasi (*observability*).
   Apa yang dilihat, maka itulah yang diuji. Masukan (input) tersedia sebagai bagian dari pengujian yang menghasilkan keluaran (output) yang berbeda. Bagian dari variable sistem terlihat atau dapat di pertanyakan selama eksekusi, sehinggan keluaran yang salah dapat dengan mudah diidentifikasi serta kesalahan internal dapat secara otomatis dideteksi dan dilaporkan.
3. Kemampuan untuk dapat dikontrol (*controllability*).
   Semakin baik sebuah aplikasi dapat dikontrol, maka pengujian akan semakin dapat diotomatisasi dan di optimalkan.
4. Kemampuan untuk dapat disusun (*decomposability*).
   Dengan mengontrol ruang ingkup pengujian, maka akan lebih cepat untuk mengisolasi masalah dan melakukan pengujian ulang dengan lebih baik.
5. Kesederhanaan (*simplicity*).

Semakin sedikit yang diuji maka akan semakin cepat pengujiannya. Program harus menunjukan kesederhanaan fungsional misalnya fitur –fitur dan kesederhanaan kode program.

6. Stabilitas *(stability*).
   Semakin sedikit perubahan, maka semakin sedikit gangguan untuk pengujian.
7. Kemampuan untuk dapat dipahami (*understandability)*.

Setelah mengetahui tentang kemampuan aplikasi dapat diuji berdasarkan dari pendapat James Bach, selanjutnya yang perlu diketahui adalah karakteristik mengenai pengujian. Karakteristik pengujian menurut Kaner, Falk, dan Nguyen mengambarkan atribut – atribut pengujian yang baik sebagai berikut [16] :

1. Pengujian yang baik memiliki probabilitas tinggi untuk menemukan kesalahan.
   Agar mencapai tujuan ini, penguji harus memahami perangkat lunak dan mencoba untuk mengembangkan sebuah gambaran mental bagaimana suatu perangkat lunak bisa gagal. Idealnya kelas kegagalan dapat diselidiki.
2. Pengujian yang baikt tidak berulang – ulang.
   Waktu dan sumber daya pngujian terbatas. Tidak ada gunanya melakukan pengujian yang memiliki tujuan yang sama dengan pengujian yang lain. Setip pengujian harus memiliki tujuan yang berbeda (bahkan jika itu hanya sedikit berbeda).
3. Pengujian yang baik harus menjadi bibit terbaik.
   Dalam sebuah kelompok pengujian yang memiliki tujuan serupa, keterbatasan waktu dan sumber daya dapat mengurangi pelaksanaan bahkan hanya sebagian kecil dari pengujian ini. Dalam hal kasus seperti itu pengujian yang memiliki kemungkinan tertinggi dalam mengungkap seluruh kelas kesalahan harus digunakan.
4. Pengujian yang baik tidak harus selalu sederhana ataupun rumit.
   Meskipun terkadang keadaan sangat memungkinkan untuk menggabungkan serangkaian pengujian mejadi satu kasus pengujian saja, dengan efek

samping yang biasanya terjadi adalah banyaknya kesalahan yang harus ditutupi. Maka secara umum , setiap pengujian harus dilaksanakan secara terpisah.

Setiap produk rekayasa dapat diuji dalam satu dari dua cara berikut :

1. Mengetahui fungsi yang telah ditentukan.
   Dengan mengetahui fungsi – fungsi suatu produk yang telah dirancang untuk bekerja, maka pengujian pun dapat dilakukan untuk menunjukan bahwa masing – masing fungsi sepenuhnya dapat beroperasi dengan baik, sementara pada saat yang sama juga dapat mencari kesalahan pada setiap fungsi.
2. Dengan mengetahui cara kerja internal sebuah produk.
   Pengujian dapat dilakukan untuk memastikan bahwa semua perseneling telah terhubung, dengan mengetahui operasi – operasi internal telah dilakukan sesuai dengan spesifikasi dan semua komponen internal telah memadai untuk dieksekusi. Pendekatan pengujian prtama kali membutuhkan pandangan eksternal dan disebut dengan pengujian kotak – hitam *(black - box testing,)* dan yang kedua membutuhkan pandangan internal yang disebut dengan kotak – putih *(white – box testing).*

2.6.1 Pengujian Kotak Hitam (Black Box)

Pengujian kotak hitam, juga disebut pengujian perilaku, yang berfokus pada persyaratan fungsional perangkat lunak. Artinya, teknik pengujian kotak hitam memungkinkan untuk membuat beberapa kumpulan kondisi masukan yang sepenuhnya akan melakukan semua kebutuhan fungsional untuk program. Pengujian kotak hitam bukan teknik alternative untuk pengujian kotak putih. Sebaliknya, ini merupakan pendekatan pelengkap yang mungkin dilakukan untuk mengungkap kelas yang kesalahan yang berbeda dari yang diungkapkan oleh metode pengujian kotak putih.

Pengujian kotak hitam berupaya untuk menemukan kesalahan dalam kateori berikut :

1. Fungsi yang salah atau hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses basis yang salah atau hilang
4. Kesalahan perilaku dan kinerja
5. Kesalahan inisialisasi dan penghentian [17].

2.6.2 Pengujian Kotak Putih (White Box)

Pengujian kotak putih atau yang terkadang disbut sebagai pengujian kotak kaca, merupakan filosofi perancangan test case yang menggunakan struktur control yang dijelaskan sebagai bagian dari perancangan peringkat komponen untuk menghasilkan test case. Dengan menggunakan metode pengujian kotak putih , dapat diperoleh hasil :

1. Test case yang menjamin bahwa semua jalur independen di dalam modul telah di eksekusi setidaknya satu kali
2. Test case yang melaksanakan semua keputusan logis pada sisi benar dan yang salah
3. Test case yang melaksanakan semua loop pada batas mereka dan dalam batas – batas operasional mereka.

Test case yang melakukan struktur data internal untuk memastikan kesahihannya [16].

2.6.3 Pengujian Alpha

Pengujian *alpha* dilakukan disisi pengembang oleh sekelompok perwakilan dari pengguna akhir. Perangkat lunak ini diguanakan dalam posisi natural, dimana pengembang “melihat dengan kacamata” pengguna dan mencatat kesalahan – kesalahan dan masalah - masalah yang timbul.

2.6.4 Pengujian Beta

Pengujian *beta* merupakan pengujian yang dilakukan secara objektif, dimana dilakukan pengujian secara langsung terhadap pengguna dengan menggunakan kuesioner mengenai kepuasan pengguna atas aplikasi yang telah

dibangun. Adapun metode penilaian pengujian yang digunakan adalah metode kuantitatif berdasarkan data dari pengguna.

Dalam penelitian kuantitatif peneliti menggunakan istrumen untuk mengumpulkan data. *Creswell* (2012) menyatakan bahwa " peneliti kuantitatif dalam mengumpulkan data menggunakan instrument merupakan alat untuk mengukur, mengobservasi yang dapat mnghasilkan data kuantitatif.

Instrument penelitian digunakan untuk mengukur nilai variable yang diteliti, dengan demikian jumlah instrument yang akan digunakan untuk penelitian akan tergantung pada jumlah variable yang diteliti. Bila variabelnya lima, maka jumlah instrument nya juga lima. Instrument – instrument penelitian sudah ada yang dibakukan, tetapi masih ada yang harus dibuat oleh peneliti sendiri. Instrument penelitian digunakan untuk melakukan pengukuran dengan tujuan menghasilkan data kuantitatif yang akurat, maka setiap instumen harus mempunyai skala. Skala pengukuran terdiri dari banyak macam yaitu [16] :

1. *Skala Likert*
2. *Skala Guttman*
3. *Rating scale*
4. *Semantic Deferential*

Dalam *game* kapten Indonesia yang dibangun, skala pengukuran untuk data kuantitatif menggunakan skala *Guttman.*