BAB 2
LANDASAN TEORI

2.1 Pengertian Game

Game saat ini merupakan suatu bentuk hiburan yang umum dan dapat ditemui dalam kehidupan sehari-hari. Berbicara tentang pengertian game, J. Huizinga dalam karya klasiknya *Homo Ludens* (1938) memberikan definisi berikut untuk permainan : permainan adalah aktivitas yang berlangsung dalam beberapa batasan ruang dan waktu, dalam urutan yang terlihat, berdasarkan kepada aturan yang secara bebas diterima dan diluar lingkup kebutuhan.

Sebuah kamus mendefinisikan game sebagai bentuk universal dari hiburan yang secara umum mengandung aktivitas apa saja yang menarik. Sebuah game kelihatannya melibatkan tiga komponen yaitu pemain yang ingin berpartisipasi, aturan yang mendefinisikan batasan dari game, dan tujuan yang menimbulkan konflik dan persaingan antara pemain.

Walaupun kebanyakan definisi yang ditemui dalam karya literatur mengangkat tiga aspek tersebut, kita dapat membuat perbedaan yang lebih halus dari ketiga aspek tersebut. Bentuk hubungan dari tiga aspek tersebut adalah tantangan, konflik dan permainan.

Dalam game, peraturan mendefiniskan game itu sendiri serta tujuan dari game tersebut. Ketika pemain memilih untuk berpartisipasi dalam sebuah game, mereka setuju untuk mengikuti peraturan yang ada. Tujuan atau goal dari game memotivasi pemain untuk menyelesaikan game tersebut [4].

2.2 Role Playing Game (RPG)

Role Playing Game atau biasa disingkat RPG adalah salah satu genre *video game* yang terkenal. Dalam game RPG, pemain mengendalikan seorang karakter dalam lingkungan virtual. RPG dibedakan dari genre lain dikarenakan

selalu berkembangnya jumlah interaksi atau aksi yang dapat dilakukan dalam dunia virtual game RPG.

Tujuan dari pemain dalam sebuah game RPG adalah untuk menyelesaikan minimal satu misi (*quest*) utama dan atau dengan beberapa misi sampingan. Yang menjadi tolak ukur apakah pemain berhasil atau tidak dalam misinya, ditentukan oleh respon dari interaksi dengan lingkungan game dan penduduknya.

Para penduduk game ini disebut sebagai *non-player character* (NPC) dikarenakan mereka dikontrol oleh sistem dan bukan oleh pemain. Kualitas interaksi antara pemain dengan NPC dalam game merupakan tolak ukur keberhasilan suatu game RPG [5].

2.3 Sejarah Pangeran Diponegoro

Kisah atau latar yang diangkat dalam game RPG yang dibangun adalah kisah mengenai Pangeran Diponegoro yakni dimulai dari usaha perlawanannnya terhadap penjajahan Belanda hingga akhirnya beliau ditangkap oleh pihak Belanda dan diasingkan ke Makassar.

Berikut secara lengkap kisah sejarah mengenai Pangeran Diponegoro dari mulai usahanya melawan penjajah, hingga diasingkan ke Makassar :

Diponegoro terlahir bernama Raden Mas Ontowiryo, adalah putra sulung dari Sultan Hamengku Buwono III (HB III) atau disebut juga sultan Raja, lahir pada hari Jumat Wage, 8 Muharram tahun Be 1712 Wuku Wayang, atau masehi 11 November 1785 di Yogyakarta. Diponegoro sebagai seorang anak raja, maka lelaki berdarah biru ini mendapat gelar kebangsawanan tertinggi di Jawa yaitu Pangeran didepan namanya.

Sebagai seorang Pangeran atau putra Sultan dengan kepedihan hati melihat tingkah laku pemerintah kolonial Belanda saat itu dalam keinginan menguasai wilayah Jawa secara penuh dan melemahkan pemerintahan kesultanan, Pangeran Diponegoro kemudian mengadakan perlawanan terhadap Belanda. Karena mengetahui ketidak-setujuan Diponegoro pada cara-cara Belanda menghancurkan kekuasaan pemerintahan kesultanan Yogyakarta, maka Belanda melakukan siasat jahat yaitu dengan menuduh Diponegoro menyiapkan pemberontakan dan

mengangkat senjata melawan Belanda dikarenakan kecewa tidak diangkat menjadi raja yang sungguh merupakan sebuah fitnah. Setelah Diponegoro mengundurkan diri dari keanggotaan Dewan Mangkubumi, Ia pun menetap di Tegalrejo. Belanda mengetahuinya dan menyerang Tegalrejo pada tanggal 20 Juli 1825. Pasukan Diponegoro lalu berpindah ke Selarong, sebuah daerah berbukit yang kemudian dijadikan markas besarnya.

Menyadari bahwa tidak akan pernah bisa menang tanpa penderitaan rakyat biasa, setelah berperang hampir 5 tahun, Pangeran Diponegoro akhirnya menyetujui penyelesaian secara perundingan. Maret 1830 Diponegoro bersedia melakukan perundingan di Magelang, tetapi ternyata hal tersebut hanya taktik Belanda saja. Diponegoro ditangkap oleh Belanda melalui tipu muslihat. Hari itu adalah hari suci umat Muslim (Idul Fitri 1245H). Diponegoro dengan hati suci pun menerima undangan dari Belanda, yang katanya tidak akan merusak kesucian hari besar ini. Pangeran Diponegoro memang sangat sulit dan tidak pernah mau berkompromi serta percaya kepada apa yang dijanjikan Belanda. Namun pada hari yang suci itu beliau bersedia. Suasana masih diliputi kesucian dan kehidmatan. Diponegoro datang untuk bermufakat. Namun bukan mufakat yang didapat melainkan Diponegoro ditangkap. Diponegoro minta diasingkan saja ke Mekkah. Belanda menyanggupi, namun sebelum ke Mekkah kata Belanda singgah dulu ke Manado. Janji untuk dibuang ke Mekkah pun tidak kunjung datang, hingga kemudian Pangeran Diponegoro wafat di Makassar pada tahun 1855.

2.4 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek atau *Object Oriented Programming* (*OOP*) merupakan suatu pendekatan kepada pengembangan perangkat lunak dimana struktur perangkat lunaknya berdasarkan interaksi objek satu sama lain untuk menyelesaikan suatu pekerjaan. Interaksi ini digambarkan dengan pengiriman dan penerimaan pesan antara dua objek. Dalam merespon sebuah pesan, sebuah objek dapat melakukan suatu aksi.

Konsep *OOP* mulai muncul pada pertengahan tahun 1960 dengan bahasa pemrograman yang disebut Simula dan lebih lanjut berkembang pada tahun 1970

dengan kedatangan Smalltalk. Walaupun pengembang perangkat lunak tidak sangat berpegang pada kemajuan bahasa *OOP* yang muncul awal ini, metodologi berorientasi objek terus berkembang. Pada pertengahan tahun 1980-an, terjadi kebangkitan ketertarikan terhadap metodologi berorientasi objek. Secara khusus, bahasa berorientasi objek seperti C++ dan Eiffel menjadi populer dikalangan programmer. OOP kemudian terus berkembang pada tahun 1990-an dengan kedatangan java dan bahasa pemrograman berbasis objek yang lain.

Pemrograman Berorientasi Objek atau OOP memiliki karakteristik sebagai berikut :

1. Objek

   Dalam istilah OOP, objek adalah suatu struktur untuk menggabungkan data dan prosedur untuk bekerja dengan data.

2. Abstraksi

   Ketika berinteraksi dengan suatu objek, kita biasanya hanya terpaku pada bagian dari propertinya. Tanpa kemampuan abstraksi atau menyaring properti dari suatu objek, kita pasti kesulitan dalam memproses informasi atau mengerjakan suatu tugas.

   Sebagai hasil dari abstraksi, ketika dua orang berinteraksi dengan objek yang sama, mereka biasanya berurusan dengan sebagian atribut yang berbeda. Sebagai contoh, ketika kita mengendarai mobil, kita hanya perlu mengetahui kecepatan dan arah yang dituju oleh mobil. Dikarenakan mobil yang kita pakai menggunakan transmisi otomatis, kita tidak perlu mengetahui RPM (*revolution per minute*) dari mesin, jadi informasi ini dibuang. Di lain sisi, informasi ini boleh jadi merupakan informasi penting bagi seorang pembalap, jadi tidak akan dibuang informasi tersebut.

   Ketika membangun objek dalam aplikasi OOP, penting untuk menghubungkan konsep abstraksi ini. Objek hanya mengandung informasi yang berhubungan dengan konteks aplikasi.

3. Enkapsulasi

   Fitur panting lain dari OOP adalah enkapsulasi. Enkapsulasi adalah proses dimana akses langsung terhadap data tidak diperbolehkan.

Dengan enkapsulasi, data yang ada di sitem menjadi lebih aman dan dapat diandalkan.

4. Polimorfisme

   Polimorfisme adalaha kemampuan dari dua objek berbeda untuk merespon pesan yang sama dalam cara unik mereka sendiri. Sebagai contoh, kita dapat melatih anjing untuk merespon perintah menggonggong dan burung untuk merespon perintah berkicau. Di lain sisi, kita dapat melatih keduanya untuk merespon perintah berbicara. Melalui polimorfisme, anjing akan merespon dengan menggonggong dan burung akan merespon dengan berkicau.

   Dalam OOP, polimorfisme dapat diimplementasikan melalui sebuah proses yang disebut *overloading.* Dengan *overloading*, metode atau prosedur yang berbeda dapat diimplementasikan untuk objek yang sama.

5. Pewarisan

   Pewarisan atau *inheritance* dalam OOP digunakan untuk mengelompokkan objek dalam program sesuai karakteristik umum dan fungsinya. Hal ini membuat pekerjaan dengan objek menjadi lebih mudah dan lebih intuitif. Pewarisan juga membuat pemrograman lebih mudah karena karakteristik umum dapat disimpan ke dalam objek induk dan dapat diwariskan ke semua objek anak.

6. Agregasi

   Agregasi adalah ketika sebuah objek terdiri dari gabungan objek lain yang bekerja bersama. Sebagai contoh, objek mesin pemotong rumput merupakan gabungan dari objek roda, objek mesin, objek besi, dan seterusnya. Agregasi dalam OOP merupakan fitur yang memungkinkan kita secara akurat memodelkan dan mengimplementasikan proses bisnis dalam suatu program [7].

2.5 Unified Modeling Languange (UML)

Unified Modeling Language (UML) merupakan keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan

desain sistem perangkat lunak, khususnya sistem yang dibangun dengan menggunakan pemrograman berorientasi objek (OOP). Definisi ini merupakan definisi yang sederhana. Pada kenyataannya, pendapat orang-orang mengenai UML berbeda satu sama lain. Hal ini dikarenakan oleh sejarahnya sendiri dan oleh perbedaan persepsi tentang apa yang membuat sebuah proses rancang-bangun perangkat lunak efektif.

Bahasa pemodelan grafis telah ada di industri perangkat lunak sejak lama. Pemicu utama di balik semuanya adalah bahwa bahasa pemrograman berada pada tingkat abstraksi yang tidak terlalu tinggi untuk memfasilitasi diskusi tentang desain.

Meskipun faktanya adalah bahasa pemodelan grafis telah ada sejak lama, masih terdapat banyak pertentangan di dalam industri perangkat lunak tentang perannya. Pertentangan ini berpengaruh langsung pada bagaimna orang-orang memandang peran UML itu sendiri.

UML merupakan standar relatif yang terbuka yang dikontrol oleh *Object Management Group* (OMG), sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khusunya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar COBRA (*Common Object Request Broker Architecture*) [8].

2.5.1 Tujuan UML

Tujuan UML diantaranya untuk :

1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

2.5.2 Notasi dalam UML

UML menyediakan beberapa notasi dan artifak standar yang bisa digunakan sebagai alat para pelaku dalam proses analisis dan desain. Artifak di dalam UML didefinisikan sebagai informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak.

a. Actor

   Actor adalah segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi actor ini bisa berupa orang, perangkat keras, atau mungkin objek lain dalam sistem yang sama biasanya yang dilakukan actor adalah memberikan informasi pada sistem dan atau memerintahkan sistem untuk melakukan sesuatu.

b. Class

   Class merupakan pembentuk utama dari sistam berorientasi objek karena class menunjukkan kumpulan objek yang memiliki atribut dan operasi yang sama. Class digunakan untuk mengabstaksikan elemen – elemen dari sistem yang sedang dibangun.

c. Interface

   Interface merupakan kumpulan operasi tanpa implementasi dari suatu class.

d. Use Case

   Use case menjelaskan urutan kegiatan yang dilakukan actor dan sistem untuk mencapai suatu tujuan tertentu walaupun menjelaskan kegiatan. Namun use case hanya menjelaskan apa yang dilakukan oleh actor dan sistem, bukan bagaimana actor dan sistem melakukan kegiatan tersebut.

e. Interaction

   Interaction digunakan untuk menunjukkan baik aliran pesan atau informasi antar objek maupun hubungan antar objek.

f. Package

   Package adalah kontainer atau wadah konspetual yang digunakan untukmengelompokkan elemen – elemen dari sistem yang sedang di bangun, sehingga bisa dibuat model yang lebih sederhana. Tujuannya

adalah untuk mempermudah penglihatan (visibility) dari model yang sedang dibangun.

g. Note

Note digunakan untuk memberikan keterangan dan komentar tambahan dari satu elemen sehingga bisa langsung terlampir dalam model. Note ini bisa ditampilkan ke semua elemen notasi yang lain.

h. Dependency

Depedency merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain.

i. Boundary Class

Boundary class adalah class yang memodelkan interaksi satu atau banyak aktor dengan sistem.

j. Control Class

Control class adalah class yang mengkoordinasi aktivitas dalam sistem class ini menghubungkan boundary class dengan entity class.

k. Entity Class

Entity class adalah class yang menghubungkan dengan data atau informasi yang digunakan oleh sistem. Entity class ini adalah class yang menyimpan dan mengelola data dapat dikombinasikan.

2.5.3 Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek. berikut akan dijelaskan 4 macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi object, yaitu use case diagram, sequence diagram, collaboration diagram, dan class diagram.

1. Use Case Diagram

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. Use case diagram terdiri atas diagram untuk use case dan actor. Actor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. Selain itu, Use case merepresentasikan operasi-operasi yang dilakukan

oleh actor. Use case digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. Actor yang melakukan operasi dihubungkan dengan garis lurus ke use case.

2. Sequence Diagram

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

3. Activity Diagram

   Activity diagram digunakan untuk menggambarkan aliran aktivitas, langkah-langkah penyelesaian dan interaksi yang terjadi antara pengguna dengan sistem dalam suatu proses.

4. 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.

2.6 XNA Game Studio

XNA Game Studio merupakan tool pengembangan perangkat lunak atau biasa disebut IDE (*Integrated Development Environment*) yang didesain untuk mempermudah pengembangan game untuk platform Microsoft Windows, Xbox 360, dan Windows Phone. XNA Game Studio memperluas Microsoft Visual Studio dengan dukungan untuk XNA Framework dan tools pendukungnya.

XNA Framework adalah sebuah *class library* yang berisi fitur-fitur yang ditujukan secara spesifik untuk pengembangan game. Sebagai tambahan, XNA Game Studio menyediakan tools untuk menambahkan konten grafik dan suara ke dalam game yang dikembangan.

XNA Framework didesain untuk mengikuti pola desain .NET Framework. Dengan XNA Game Studio, Pengembang (*Developer*) Game dapat menggunakan

kemampuan (*capability*) dari XNA Framework dan lebih umum .NET Framework untuk pengembangan game [9].