BAB II

TINJAUAN PUSTAKA

II.1. *Game*

Game atau permainan merupakan sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah biasanya tidak dalam konteks serius atau dengan tujuan untuk refreshing. Permainan juga adalah sarana yang dibuat untuk memenuhi kebutuhan manusia akan hiburan, sehingga sampai saat ini, permainan terus berkembang sesuai dengan kebutuhan dan tidak dapat dipisahkan dari manusia. Berevolusi mengikuti perkembangan intelegensi seiring dengan perkembangan jaman. Konsep bermain, bentuk dan cara memainkan adalah hal yang terevolusi dan membuat permainan yang ada saat ini tidak lagi seperti permainan-permainan terdahulu.

Teori permainan adalah suatu cara belajar yang digunakan dalam menganalisa sejumlah pemain maupun perorangan yang menunjukkan strategi-strategi rasional. Teori permainan pertama kali ditemukan oleh sekelompok ahli matematika pada tahun 1944. Teori itu di kemukakan oleh John Von Neumann and Oscar Morgenstern, menurutnya permainan terdiri atas sekumpulan peraturan yang membangun situasi 5 bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri ataupun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah

keterangan diterima setiap pemain sebagai kemajuan pemain dan sejumlah kemenangan ataupun kekalahan dalam berbagai situasi (Rickman Roedavan, 2014:1).

II.1.1. *Game Engine*

Game Engine adalah sebuah perangkat lunak yang dirancang untuk membuat *game*. Sebuah *game engine* biasanya dibangun dengan mengenkapsulasi beberapa fungsi standar yang umum digunakan dalam pembuatan sebuah *game*. Misalnya, fungsi *rendering*, pemanggilan suara, *network*, atau pembuatan partikel untuk special *effect*. Sebagian besar *game engine* umumnya berupa *library* atau sekumpulan fungsi- fungsi yang penggunannya dipadukan dengan bahasa pemrograman.

Sebuah *game engine* biasanya dilengkapin dengan fungsi-fungsi grafis atau komputasi dasar yang jarang diketahui oleh *developer*. *Game Engine* menjadikan proses pembuatan menjadi lebih cepat dan mudah, (Rickman Roedavan, 2014:2).

II.1.2. Perkembangan *Game Engine*

Di awal tahun 2000-an, *game engine* mengalami perkembangan yang cukup singnifikan. Beberapa *game engine* mulai dilengkapi dengan *World Editor*. Sehingga alih-alih menggunakan perangkat lunak 3D seperti 3D Max, Blender, pembuatan level atau ‘dunia’ *game* dapat dibuat melalui sebuah perangkat lunak tersendiri yang telah dirancang khusus untuk *game engine* tersebut.

Perkembangan ini bukannya tanpa masalah. Di satu sisi, *software* ini memudahkan para *developer* pemula yang membutuhkan kecepatan untuk melihat produk dari rancangan mereka. Tapi di sisi lain, perangkat lunak ini bisa 'membatasi' kemampuan *developer* untuk berkreasi lebih jauh. Sehingga bagi sebagian *developer*, membangun 'dunia' *game* lewat *script* tetap puluhan kali lebih *powerfull*. Dengan kata lain, *game engine* tersebut tidak lagi harus tergantung dengan *software development* seperti Delphi atau Ms. Visual C++, karena *game engine* tersebut telah dilengkapin *code editor* dan *compiler* sendiri.

Berdasarkan permainan *game genre* bisa dikategorikan menjadi beberapa kategori, antara lain:

1. *Action Game*

Tipe *game* ini biasanya meminta pemain untuk melakukan tembak menembak dengan musuh. Pemain dituntut untuk memiliki reaksi yang cepat dan tepat untuk bisa menghindari serangan musuh dan kemudian membalas serangan tersebut. Pengembangnya diharapkan bisa melakukan optimasi terbaik untuk game jenis ini, karena game jenis ini harus memiliki tingkat akurasi tinggi dalam memainkannya dan tidak terdapat delay atau jeda dalam permainannya.

2. *Adventure Game*

Berbeda dengan jenis action *game*, permainan jenis ini umumnya meminta pemain untuk mengeksplorasi tempat (di dalam *game*) yang sudah dikondisikan sebelumnya. Tempat-tempat itu bisa berupa kastil, hutan, bawah laut dan lain lain. Tujuan dari eksplorasi tersebut biasanya adalah mencari pesan rahasia, mendapatkan benda-benda tertentu, bertempur dengan musuh dan lain-lain.

3. *Sport Game*

Game jenis ini biasanya bersifat kompetisi dan bisa dimainkan oleh beberapa pemain, baik itu secara individual maupun tim. Sesuai namanya, kebanyakan *game* dengan *genre* ini adalah *game* bertema olahraga.

4. *Role Playing Game*

Game jenis ini kebanyakan adalah *online game*, walaupun tidak sedikit juga yang bisa dimainkan secara offline. Inti dari permainan jenis ini adalah perkembangan/pertumbuhan karakter utama yang dimainkan oleh pemain. Ketika sedang bermain secara *online*, pemain bisa berkompetisi, bertukar informasi maupun bertanding dengan pemain lainnya yang juga sedang bermain secara *online.*

5. *Simulation Game*

Game simulasi bertujuan untuk mensimulasikan pengalaman seperti terbang didalam pesawat, serealistis dan sepraktis mungkin, dan juga terdapat hokum fisik seperti di dunia nyata. Beberapa *game* membutuhkan untuk banyak membaca sebelum *game* dimainkan, sementara *game* yang lain sudah termasuk petunjuk singkat. *Game* simulasi seringkali dikaitkan dengan simulator sungguhan, dan sebagian dari *game* tersebut memang berguna dalam pelatihan militer sungguhan atau untuk kepentingan umum.

6. *City Building*

Adalah permainan pembangunan kota, yang mana sub *genre* spesialis dari permainan simulasi ekonomi, pemain bertindak sebagai perencana keseluruhan atau pemimpin yang harus memenuhikebutuhan dan keinginan

karakter dalam *game* dengan cara membangun bangunan untuk makanan, tempat tinggal, kesehatan, keagamaan, pertumbuhan ekonomi dan lain-lain. Kesuksesan diraih saat anggaran kota membuat keuntungan terus-menerus dan *war*ga kota mengalami peningkatan kesejahteraan, kesehatan, dan lainnya. Disamping itu semua, perkembangan militer juga sering dimasukkan, fokusnya dalam kekuatan ekonomi, (Rickman Roedavan, 2014:4).

II.2. *Sniper*

Sniper (Penembak Runduk) adalah seorang prajurit infanteri yang secara khusus terlatih untuk mempunyai kemampuan membunuh musuh secara tersembunyi dari jarak jauh dengan menggunakan senapan. Sedangkan petembak jitu (*marksman*, *sharpshooter*) adalah istilah yang dipakai pada bidang militer. Seorang penembak jitu terlatih untuk menembak secara tepat dan akurat dengan menggunakan senapan tipe tertentu. Beberapa doktrin militer memakai penembak jitu yang tergabung dalam infanteri tingkat regu. Penembak jitu modern sering disamakan dengan penembak runduk (*sniper*), padahal, keduanya sebenarnya berbeda.

Sniper pada umumnya menggunakan senapan runduk *bolt-action* khusus, sedangkan penembak jitu menggunakan senapan semi-otomatis, yang biasanya berupa senapan tempur atau senapan serbu yang dimodifikasi dan ditambah teleskop. *Sniper* telah mendapatkan pelatihan khusus untuk menguasai teknik bersembunyi, pemakaian kamuflase, keahlian pengintaian dan pengamatan, serta kemampuan infiltrasi garis depan. Ini membuat *sniper* memiliki peran strategis

yang tidak dimiliki penembak jitu. Penembak jitu dipasang pada tingkat regu, sedangkan sniper pada tingkat batalion dan tingkat kompi.

Sniper dalam peperangan dalam posisinya pada unit militer, lokasi menembak, dan taktik berbeda pada setiap negara. Secara umum, tujuan sniper dalam peperangan adalah mengurangi kemampuan tempur musuh dengan cara membunuh sasaran yang bernilai tinggi, seperti perwira. Dalam doktrin Amerika Serikat, Inggris, dan banyak negara lainnya , sniper dipakai dalam tim sniper, yang berisi hanya dua orang. Dua orang ini mempunyai fungsi yang berbeda, satu sebagai penembak, dan satu orang lagi sebagai *spotter* yaitu penunjuk sasaran. Dalam prakteknya, spotter dan penembak biasa bergiliran menembak, agar mengurangi kelelahan pada mata (Henry, Ahmad Kurniawan 2010:11).

II.3. *Unity*

Pengertian *Unity* adalah salah satu *game engine* yang mudah digunakan, hanya membuat objek dan diberikan fungsi untuk menjalankan objek tersebut. *Unity* 3D dibuat dengan menggunakan bahasa perogram C++, tapi pengguna tidak perlu menggunakan bahasa C++ yang sulit, karena *Unity* 3D mendukung bahasa program lain seperti JavaScript, C#, dan Boo. *Unity* memiliki kemiripan dengan *game engine* lainnya seperti, Blender *game engine*, Virtools, *game studio*, adapaun kelebihan dari *Unity* 3D, *Unity* dapat dioperasikan pada platform Windows dan Mac Os dan dapat menghasilkan *game* untuk Windows, Mac, Linux, Wii, iPad, iPhone, google Android dan juga browser. Untuk browser, memerlukan sebuah plugin, yaitu *Unity* Web Player, sama halnya dengan Flash

Player pada Browser. *Game Unity* 3D juga mendukung dalam pembuatan *game* untuk *console game* Xbox 360 dan PlayStation 3, dalam setiap objek mempunyai variabel, variabel inilah yang harus dimengerti supaya dapat membuat game yang berkualitas. Berikut ini adalah bagian-bagian dalam *Unity* adalah :

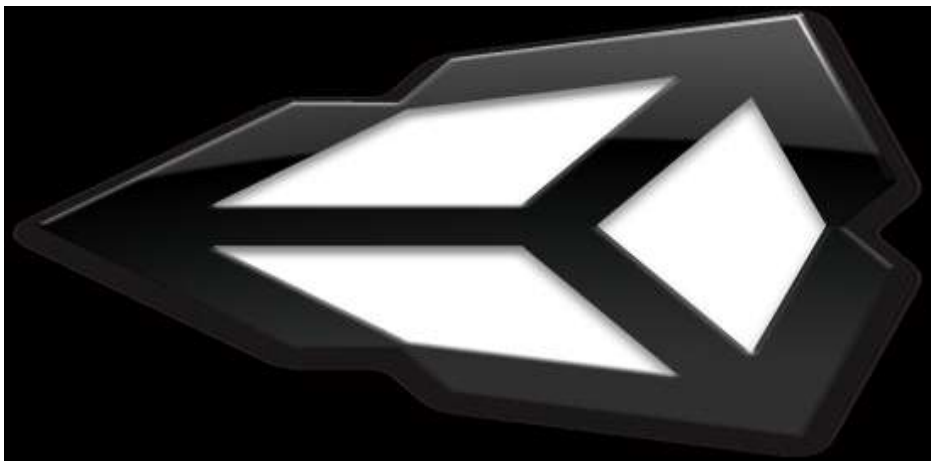
1. ***Asset*** adalah tempat penyimpanan dalam *Unity* yang menyimpan suara, gambar, video, dan tekstur.
2. ***Scenes*** adalah area yang berisikan konten-konten dalam *game*, seperti membuat sebuah *level*, membuat menu, tampilan tunggu, dan sebagainya.
3. ***Game Objects*** adalah barang yang ada di dalam *assets* yang dipindah ke dalam *scenes*, yang dapat digerakkan, diatur ukurannya dan diatur rotasinya.
4. ***Components*** adalah reaksi baru, bagi objek seperti *collision*, memunculkan partikel, dan sebagainya. *Script*, yang dapat digunakan dalam *Unity* yaitu *Javascript*, *C#* dll .
5. ***Prefabs*** adalah tempat untuk menyimpan satu jenis *game objects*, sehingga mudah untuk diperbanyak.

Untuk meningkatkan kualitas pemetaan atau tokoh dalam *game*, *Unity* 3D mendukung penggunaan software pengolahan gambar lain, seperti 3D Max, Maya, Softimage, Blander, Modo, ZBrush, Cinema4D, Cheetah3D, Adobe Photoshop, Adhobe Fireworks, dll, (Rickman Roedavan, 2014:6).

II.3.1. Sekilas Tentang *Unity*

Unity Technologies dibangun di tahun 2004 oleh David Helgason, Nicholas Francis dan Joachim Ante. *Game engine* ini dibangu atas dasar

kepedulian mereka terhadap indie *developer* yang tidak bisa membeli *game engine* karena terlalu mahal. Fokus perusahaan ini adalah membuat sebuah perangkat lunak yang biasa digunakan oleh semua orang, khususnya untuk membangun sebuah *game*. Di tahun 2009, *Unity* diluncurkan secara gratis dan di april 2012, *Unity* mencapai popularitas tertinggi dengan lebih dari 1 juta *developer* terdaftar di seluruh dunia.



Gambar II.1. Logo *Unity*
(Sumber : Rickman Roedavan, 2012 :7)

II.4. *Unified Mondelling Language* (UML)

Unified Modelling Langguage (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berpradigma berorientasi objek". Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

Dapat disimpulkan bahwa *Unified Modelling Language* (UML) adalah sebuah bahasa pemodelan yang berorientasi objek dan menjadi standar dalam

visualisasi, merancang, dan mendokumentasi sistem perangkat lunak untuk penyederhanaan permasalahan-permasalahan yang kompleks (Rosa A.S & M. Shalahuddin, 2011 : 118).

II.4.1. *Use Case Diagram*

Menurut (Rosa A.S & M. Shalahuddin, 2011 : 210), *use case* digunakan selama masa pengumpulan kebutuhan dan analisis untuk mempresentasikan fungsionalitas dari system. *Use case* berpesat pada perilaku system pada sudut pandang luar. Syarat penamaan pada *use case* adalah nama didefenisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefenisian apa yang disebut aktor dan *use case*.

1. **Aktor** merupakan orang, proses, atau system lain yang berinteraksi dengan system informasi yang akan dibuat di luar system informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. ***Use Case*** merupakan fungsionalitas yang disediakan system sebagai unit – unit yang saling bertukar pesan antar unit atau aktor.

Berikut tabel II.1 menerapkan symbol – symbol pada diagram *use case*.

Table II.1. *Diagram Use Case*

No.	Gambar	Nama	Keterangan
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

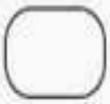
(Sumber : *Simonn Bennet, Steve Marcob dan Ray Farmer, 2011 : 146*)

II.4.3. *Activity Diagram*

Diagram activity secara *grafis* digunakan untik menggambarkan rangkaian aliran aktivitas baik proses bisnis atau *use case*. *Diagram* ini juga dapat digunakan untuk memodelkan *action* yang akan dilakukan saat sebuah operasi dieksekusi dan memodelkan hasil dari *action* tersebut. *Node* pada sebuah *activity diagram* disebut sebagai *action*, sehingga diagram tersebut menampilkan sebuah *activity* yang tersusun dari *action* (Rosa A.S & M. Shalahuddin, 2011 : 225).

Berikut tabel II.2 menerapkan symbol – symbol pada *activity diagram*.

Tabel II.2. ***Diagram Activity***

NO	GAMBAR	NAMA	KETERANGAN
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

(Sumber : ***Simonn Bennet, Steve Marcob dan Ray Farmer, 2011 : 162*****)**