# BAB II

# LANDASAN TEORI

# Pengertian *Game*

*Game* atau dalam bahasa Indonesia disebut dengan *game* adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan refreshing. Andang Ismail (2009:26) menuturkan bahwa *game* ada dua pengertian. Pertama, *game* adalah sebuah aktifitas bermain yang murni mencari kesenangan tanpa mencari menang atau kalah. Kedua, *game* diartikan sebagai aktifitas bermain yang dilakukan dalam rangka mencari kesenangan dan kepuasan, namun ditandai pencarian menang-kalah.

Dalam Jurnal Ilmiah Komputer dan Informatika (KOMPUTA) oleh Abdi Tanjung pada tahun 2013 mengatakan bahwa pola umum tantangan dari suatu *game* menunjukan *genre* dari *game* itu sendiri (Tanjung, 2013). Dengan perkembangan informasi seperti sekarang, *genre* dari *game* masih terus berkembang, sebagai contoh simulasi menari (*dance simulation*) yang diperkenalkan oleh desainer *game* dari Jepang. Namun secara umum *game* dapat dibagi menjadi beberapa jenis berdasarkan *genre* yang diterapkannya.

*Game* RPG atau *Role Playing Games* melibatkan masalah taktik, logika, dan eksplorasi atau penjelajahan. Dan juga kadang meliputi teka-teki dan masalah ekonomi karena pada *game* ini biasanya melibatkan pengumpulan barang-barang rampasan dan menjualnya untuk mendapatkan senjata yang lebih baik. (Andrew Rollings, 2003).

Pong dan Pac-Man termasuk dalam kategori *game* kasual. Pengisi waktu ini mencakup berbagai *genre* sperti teka-teki, permainan kartu dan papan, *game* strategi, arcade. *Game* kasual miliki aturan sederhana, tidak memerlukan keterampilan khusus dan ditargetkan untuk audiens massal. Ketika mereka pertama kali muncul, perangkat keras komputer tidak cukup kuat untuk menangani grafis dan algoritma yang lebih kompleks. (*Telecommunication Standardization Sector of the International Telecommunications Union*, 2011).

*Sports game* sesuai dengan namanya yaitu *game* yang mensimulasikan kegiatan olahraga. Banyak pembuat *game* telah memasukan aspek-aspek diluar olahraga itu sendiri. Misalnya, sebagian *game* sepakbola memungkinkan pemain bisa membuat dan menyesuaikan tim dan memainkanya di seluruh musim. Selain itu, banyak *game* olahraga yang memasukan elemen manajemennya kedalam *game*. Sehingga ada cukup banyak variasi dalam *genre* ini. (Stahl, 2005).

*Game* simulasi atau sering disebut *Sims* membuat para pemain *game* mengalami situasi dunia nyata dari tempat yang lebih aman. Sejak 1950-an, Departemen Pertahanan telah melatih militer dengan permainan simulator seperti simulator penerbangan, tank dan perang. Banyak sekali hal yang disimulasikan dari dunia nyata bersifat mahal dan berbahaya. Dengan memproduksi *game* simulasi ini akan lebih praktis, murah dan lebih mudah dengan hasil yang kurang lebih sama. (Pedersen, 2003).

*Game* petualangan bukan merupakan sebuahkompetisi atau simulasi seperti *game* yang lainnya. *Game* yang menawarkan eksplorasi dan pemecahan teka-teki sebagai daya tarik utama. *Game* jenis ini tidak menawarkan proses untuk dikelola atau mengalahkan musuh melalui strategi dan taktik. *Game* petualangan merupakan cerita interaktif mengenai karakter yang dikontrol oleh pemain. Dalam *game* petualangan pemain dituntut kemampuan berfikirnya untuk menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter, menggunakan benda-benda yang tepat dan diletakan di tempat yang tepat. Penalaran, kreativitas, dan keingintahuan adalah keterampilan paling umum yang diperlukan dari seorang pemain *game* petualangan yang baik.

*Game* puzzle merupakan *game* yang bertujuanuntuk memecahkan sebuah puzzle / teka - teki, terkadang tanpa menyatukan dengan jalan cerita atau tujuan yang lebih besar. *Game* jenis ini biasanya bervariasi pada satu tema saja.

*Genre* aksi mencakup permainan apa pun di mana pemain mengatasi tantangan dengan seperti sasaran yang tepat dan waktu respons yang cepat. *Game* aksi kadang-kadang bisa menggabungkan tantangan lain seperti balapan, teka-teki, atau mengumpulkan objek, tetapi mereka tidak menjadi pusat *genre*.

Ada 4 elemen sebagai penyusun dasar sebuah *game* (Chris Crawford, 1984). Elemen pertama yaitu mekanika. Berbentuk prosedur dan aturan dalam permainan agar pemain dapat mencapai sebuah tujuan. Setiap *genre* dari *game* memiliki mekanika sendiri yang menjadi pembeda antara *genre* satu dengan yang lain. Elemen mekanika ini juga sebagai pembeda antara media hiburan seperti buku dan film karena kedua media tersebut tidak memiliki mekanika.

 Elemen kedua adalah cerita. Dalam penyusunan sebuah permainan cerita menggambarkan urutan peristiwa yang menjelaskan alur permainan. Setiap permainan memiliki cerita masing-masing. Permainan paling sederhanapun pasti memiliki cerita dibaliknya. Cerita akan tergambar jelas jika dibantu elemen-elemen penyusun lain sehingga nilai akan tersampaikan dengan maksimal.

Elemen selanjutnya adalah estetika. Estetika adalah bagaimana suatu permainan itu terlihat, terdengar, tercium dan terasa. Estetika merupakan aspek yang sangat penting dari perancangan permainan karena memilki hubungan langsung dengan pengalaman pemain. Estetika menjadi hal yang paling di utamakan bagi pembuat *game* karena menjadi penarik minat utama bagi para pemain *game* untuk memainkan ciptaan mereka.

Element terakhir adalah teknologi. Teknologi menjadi elemen terpenting karena menggabungkan elemen-elemen yang lain untuk saling bekerna secara bersama-sama. Teknologi menjadi tempat mekanika berkerja, menampilkan estetika dari sebuah *game* sehingga cerita dapat disampaikan dengan sempurna. Elemen ini akan memberi pengaruh baik jika dimanfaatkan dengan baik apalagi elemen ini selalu berkembang dari masa ke masa untuk meningkatkan ruang dari elemen-elemen lain untuk berkembang.

*Game* strategi adalah permainan yang membutuhkan pemikiran dan perencanaan. Pemenang permainan ditentukan melalui pertempuran pikiran. (Pedersen, 2003). *Game* strategi berbeda dari *genre* lain karena perancang menciptakan aturan dan tujuan, tetapi pemainlah yang memutuskan strategi apa yang digunakan untuk mencapai tujuan tersebut dan mengecoh lawan. *Game* perang adalah *game* strategi meskipun mereka adalah simulasi kejadian nyata atau fiktif. *Game* strategi dapat dimainkan sebagai *game* *real-time* atau *turn-based* yang adalah sebagai berikut.

## Real Time Strategy

Menururt Matt Barton and Bill Loguidice *Real time Strategy* yang biasa disingkat RTS adalah *game* berbasis waktu yang dimana *game* tidak mengenal giliran. Setiap pemain dapat mengatur/memerintah unit dalam waktu apapun. *Game* *Real-time Strategy* juga dapat disebut sebagai simulasi waktu nyata atau *game* perang real-time. *Game* *Real-time Strategy* sering dibandingkan dengan *game* Turn-base Strategy, di mana setiap pemain memiliki waktu untuk mempertimbangkan langkah selanjutnya dengan hati-hati tanpa harus khawatir tentang tindakan lawannya sedangkan tidak ada giliran dan pemain bisa kapan saja menyerang dengan unit yang ada.

 Dalam *game* *real-time*, semua pemain termasuk pemain yang dikendalikan komputer bersaing tanpa henti, secara bersamaan. Contoh gamenya adalah Age of Empires.

## Turn Base Strategy

Jenis game ini adalah game dimana pemain melakukan gerakan secara bergantian satu sama lain. saat kita mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya.*game* *turn-based*, setiap pemain menyelesaikan gilirannya sebelum pemain berikutnya dimulai. *Point Of view* atau perspektif permainan strategi yang khas adalah tampilan seperti sudut pandang seekor burung sehingga memungkinkan semua pemain melihat seluruh area bermain. Contoh game *turn-base strategy* adalah Catur.

# Games Battleships

Berdasarkan *Battleships* dikenal di seluruh dunia sebagai *game* pensil dan kertas yang berasal dari Perang Dunia I. Diterbitkan oleh berbagai perusahaan sebagai *game* pad-dan-pensil pada 1930-an, dan dirilis sebagai *game* papan plastik oleh Milton Bradley pada tahun 1967. *game* telah melahirkan versi elektronik, video *game*, aplikasi perangkat pintar dan film.

Dalam *Battleships* board pemain disediakan beberapa buah tempat yang disusun oleh kotak-kotak sebagai medan perang. Unit-unit perang dengan ukuran yang bervariasi disusun dalam medan perang masing-masing. Letak dari unit-unit perang ini tidak terlihat dalam di pemain lawan. Pemain berusaha untuk menghancurkan unit-unit perang pemain lawan hingga unit milik musuh hancur semua.

 Berdasarkan pekembangannya beberapa *game* *Battleships* sudah adaptasi modern seperti yang ada pada *google play* dan console dengan *game*play dan variasi yang berbeda namun masih memiliki karakteristik aslinya yaitu unit yang dipakai bertarung adalah *Battleships*, tidak ada unit lain seperti tank atau pesawatdan *Battleships* juga berfungsi sama dengan nyawa pemain sehingga jika semua *Battleships* milik pemain hancur maka pemain tersebut kalah.

# Unity

Unity *Game* Engine adalah software yang digunakan untuk membuat video *game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis, selain untuk membuat *game*, Unity 3D juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan real-time 3D animasi, selain sebagai *game* engine Unity 3D juga dapat digunakan sebagai sebuah editor bagi *game* yang sudah ada.

Unity 3D dibuat dengan menggunakan bahasa pemrograman 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, *Gamestudio*, 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, kita memerlukan sebuah plugin, yaitu Unity Web Pemain, sama halnya dengan Flash Pemain pada Browser. *Game* Unity 3D juga mendukung dalam pembuatan *Game* untuk console *Game* Xbox 360 dan PlayStation 3.

# Android

 Android merupakan sistem operasi berbasis Linux yang dikembangkan untuk perangkat mobile. Android menawarkan sebuah lingkungan yang berbeda untuk pemgembang. Android tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga, ini berbeda dengan sistem operasi perangkat mobile sebelumnya seperti Windows Mobile, i-Phone OS, Symbian dan masih banyak lagi (Susanto, 2011).

# *PathFinding*

 Pathfinding adalah Perencanaan, dengan aplikasi komputer, antara dua titik. Pathfinding terkait dengan masalah jalur terpendek, dalam teori grafik, yang meneliti bagaimana mengidentifikasi jalur yang paling memenuhi beberapa kriteria (terpendek, termurah, tercepat, dll) antara dua titik dalam jaringan besar.

 Pathfinding dalam *Game* adalah proses pemindahan posisi karakter game dari lokasi awal ke lokasi tujuan yang diinginkan (David M. Broug dan Glenn Seemann, 2004).

# *Object Detection*

 Object Detection adalah salah satu aspek yang paling mendalam dari visi komputer karena memungkinkan untuk mencari, mengidentifikasi, menghitung, dan melacak objek apa pun yang ada dalam ([Moses Olafenwa](https://medium.com/%40guymodscientist?source=post_page-----6d087063f6ff----------------------),2016).

# *Extreme Programming*

 Extreme Programming adalah suatu model yang termasuk dalam pendekatan agile yang diperkenalkan oleh Kent Back. Menurut penjelasannya, definisi XP adalah sebagai berikut: “Extreme Programming (XP) adalah metode pengembangan software yang cepat, efisien, beresiko rendah, fleksibel, terprediksi, scientific, dan menyenangkan.“.

 Model ini cenderung menggunakan pendekatan Object-Oriented. Tahapan-tahapan yang harus dilalui antara lain: Planning, Design, Coding, dan Testing. Sasaran Extreme Programming adalah tim yang dibentuk berukuran antara kecil sampai medium saja, tidak perlu menggunakan sebuah tim yang besar. Hal ini dimaksudkan untuk menghadapi perubahan-perubahan requirements yang sangat cepat. Extreme Programming merupakan agile methods yang paling banyak digunakan dan menjadi sebuah pendekatan yang sangat terkenal. Kelebihan dan Kekurangan Extreme Programming adalah:

### Kelebihan

Kelebihan extreme programming adalah:

1. User dapat terlibat langsung dalam memantau berjalannya pembuatan system.
2. Pembangunan system dibuat lebih cepat
3. feedback yang sangat cepat, setiap kesalahan ditemukan maka akan langsung diperbaiki.
4. ide baru gampang di implementasikan tanpa perlu mengatur ulang perencanaan.

### Kekurangan

Kekurangan extreme programming adalah:

1. story kemungkinan besar tidak lengkap sehingga Developer harus selalu siap dengan perubahan karena perubahan akan selalu diterima.
2. Tidak bisa membuat kode yang detail di awal (prinsip simplicity dan juga anjuran untuk melakukan apa yang diperlukan hari itu juga).
3. XP tidak memiliki dokumentasi formal yang dibuat selama pengembangan. Satu-satunya dokumentasi adalah dokumentasi awal yang dilakukan oleh user.

# *Object Oriented Class*

 *object oriented class* atau *class* adalah salah satu bagian dari *Object Oriented Programming* yang merupakan paradigma pemrogaman yang berorientasikan kepada object.

 *Class* dapat didefinisikan sebagai struktur data atau cetak biru dari suatu object. Lebih jelasnya adalah sebuah bentuk dasar atau blueprint yang mendefinisikan variable, method umum pada semua object. Object sendiri adalah kumpulan variable dan fungsi yang dihasilkan dari template khusus atau disebut class. Object adalah elemen pada saat run-time yang akan diciptakan, dimanipulasi, dan dibuang / dihancurkan ketika eksekusi. Adapun *class* merupakan definisi statik dari himpunan object yang mungkin diciptakan sebagai instantiasi dari *class.*

# Perbandingan Game Sejenis

 perbandingan game sejenis ini menjadi ajuan penulis untuk merubah dan mendesain game battleships dari gameplay turn-base strategy menjadi real time strategy. penulis mengambil 4 game untuk menjadi perbandingan dan mendapatkan kesimpulan sebagai berikut.

## Perbandingan Dengan gameplay Turn-Base Strategy

dalam perbandingan ini penulis mengambil 2 game battleship sebagai bahan perbandingan dengan gameplay *turn-base strategy* dan mendapatkan bahwa pada gameplay *turn-base strategy* pemain diberikan 3-5 kapal dan diberikan kesempatan untuk memposisikan kapal pada tempat bermain seperti pada gambar dan setelah itu pemain akan bergantian menyerang kapal musuh dengan menebak dan menembak tiap kolom yang ada. pemain yang menghancurkan semua kapal milih musuhnya akan memenangkan permainan. contoh game seperti gambar.



gambar 2.1 turn base strategy battleship

## Perbandingan Dengan gameplay *Real-Time Strategy*

dalam perbandingan ini penulis mengambil 2 game dengan gameplay *real-time strategy* yaitu Games [Command & Conquer](https://id.wikipedia.org/wiki/Command_%26_Conquer) dan Warcraft. dari 2 game dengan gameplay *real time strategy* tersebut penulis mendapatkan mendapatkan bahwa pemain telah diberikan tentara(unit) dengan kelebihan dan kekurangan dan harus menemukan dan mengalahkan boss musuh dengan unit yang tersedia tersebut. dalam game pemain dapat mengerakan kemanapun unit yang dikendalikan pada arena bermain dan saat musuh ada dalam jarak deteksi unit maka unit akan secara otomatis menyerang dan masuk kedalam sistem combat sehingga pemain dapat mengontrol unit. pemain juga dapat menentukan unit untuk terus bertarung atau tidak secara langsung karena tidak ada giliran dan game berjalan secara nyata sehingga pemain bebas dalam menentukan cara bermainnya. contoh gameplay *real-time strategy* seperti pada gambar.



 gambar 2.2 [Command & Conquer](https://id.wikipedia.org/wiki/Command_%26_Conquer) rts gameplay

 setelah melakukan penelitian pada 4 game di atas penulis dapat melakukan perbandingan untuk membuat game battleships dengan gameplay *turn-base strategy* dan merubah gameplay-nya menjadi *real-time strategy.*

# Unified Modelling Language (UML)

 UML adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak. UML merupakan bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem. UML dapat digunakan untuk menerangkan sistem yang berorientasi pada objek secara lebih jelas dan detail disajikan dalam bentuk diagram atau gambar yang meliputi *class* beserta atribut dan operasinya, serta hubungan antar *class* yang meliputi *inherintance*, *association*, dan komposisi (Fowler, 2005).

 Pada pembuatan *game* ini, tipe UML yang digunakan yaitu :

## Use Case Diagram (UCD)

 Menurut Grady Booch dalam bukunya, suatu *use case diagram* menampilkan sekumpulan *use case* dan aktor (pelaku) dan hubungan diantara *use case* dan aktor tersebut. *Use case diagram* digunakan untuk penggambaran *use case* statik dari suatu sistem. *Use case diagram* penting dalam mengatur dan memodelkan kelakuan dari suatu sistem.

 *Use case* menjelaskan apa yang dilakukan sistem (atau subsistem) tetapi tidak tidak menspesifikasikan cara kerjanya. *Flow of event* digunakan untuk menspesifikasikan kelakuan dari *use case*. *Flow of event* menjelaskan *use case* dalam bentuk tulisan dengan sejelas-jelasnya, diantaranya bagaimana, kapan *use case* dimulai dan berakhir, ketika *use case* berinteraksi dengan aktor, obyek apa yang digunakan, alur dasar dan alur alternatif.



 **Gambar 2.3** Notasi *Use Case Diagram*

## Sequence Diagram

Menurut Grady Booch, *sequence diagram* adalah suatu diagram interaksi yang menekankan pada pengaturan waktu dari pesan-pesan. Diagram ini menampilkan sekumpulan peran dan pesan-pesan yang dikirim dan diterima oleh instansi yang memegang peranan tersebut (Grady Booch, 2005). *Sequence diagram* menangkap objek dan *class* yang terlibat dalam skenario dan urut-urutan pesan yang ditukar antara objek diperlukan untuk melaksanakan fungsionalitas skenario. *Sequence diagram* berasosiasi dengan *use* *case* selama proses pengembangan. Dalam *Unified Model Language* (UML), objek dalam *sequence* *diagram* digambar dengan segiempat yang berisi nama objek yang diberi garis bawah. Objek dapat diberi nama dengan tiga cara : (nama objek), (nama objek dan *class*) atau (hanya nama *class* (*anonymous* object)). Berikut notasi *sequence diagram* seperti terlihat pada gambar dibawah ini :

 

 **Gambar 2.4** Notasi *Sequence Diagram*

## Activity Diagram

 Activity Diagram adalah diagram yang menggambarkan worlflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

# Visual Studio

 **Microsoft Visual Studio** merupakan sebuah [perangkat lunak](http://id.wikipedia.org/wiki/Perangkat_lunak) lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplikasi Web. Visual Studio mencakup [kompiler](http://id.wikipedia.org/wiki/Kompiler), [SDK](http://id.wikipedia.org/w/index.php?title=Software_Development_Kit&action=edit&redlink=1), [Integrated Development Environment (IDE)](http://id.wikipedia.org/wiki/Integrated_Development_Environment), dan dokumentasi (umumnya berupa [MSDN Library](http://id.wikipedia.org/w/index.php?title=MSDN_Library&action=edit&redlink=1)). Kompiler yang dimasukkan ke dalam paket Visual Studio antara lain [Visual C++](http://id.wikipedia.org/wiki/Visual_C%2B%2B), [Visual C#](http://id.wikipedia.org/w/index.php?title=Visual_C&action=edit&redlink=1), [Visual Basic](http://id.wikipedia.org/wiki/Visual_Basic), [Visual Basic .NET](http://id.wikipedia.org/wiki/Visual_Basic_.NET), [Visual InterDev](http://id.wikipedia.org/w/index.php?title=Visual_InterDev&action=edit&redlink=1),[Visual J++](http://id.wikipedia.org/w/index.php?title=Visual_J%2B%2B&action=edit&redlink=1), [Visual J#](http://id.wikipedia.org/w/index.php?title=Visual_J&action=edit&redlink=1), [Visual FoxPro](http://id.wikipedia.org/wiki/Visual_FoxPro), dan [Visual SourceSafe](http://id.wikipedia.org/w/index.php?title=Visual_SourceSafe&action=edit&redlink=1).

Microsoft Visual Studio dapat digunakan untuk mengembangkan aplikasi dalam native code (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun managed code (dalam bentuk[Microsoft Intermediate Language](http://id.wikipedia.org/wiki/Microsoft_Intermediate_Language) di atas .NET Framework). Selain itu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi [Silverlight](http://id.wikipedia.org/wiki/Silverlight), aplikasi Windows Mobile (yang berjalan di atas .NET Compact Framework).

Visual Studio kini telah menginjak versi Visual Studio 9.0.21022.08, atau dikenal dengan sebutan Microsoft Visual Studio 2008 yang diluncurkan pada [19 November](http://id.wikipedia.org/wiki/19_November) [2007](http://id.wikipedia.org/wiki/2007), yang ditujukan untuk platform Microsoft .NET *Script*3.5. Versi sebelumnya, Visual Studio 2005 ditujukan untuk platform .NET *Script*2.0 dan 3.0. Visual Studio 2003 ditujukan untuk .NET *Script*1.1, dan Visual Studio 2002 ditujukan untuk .NET *Script*1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan Visual Studio .NET, karena memang membutuhkan Microsoft .NET Framework. Sementara itu, sebelum muncul Visual Stu