2. LANDASAN TEORI

Pada bab ini dijelaskan mengenai semua teori yang digunakan di dalam pembuatan *Game*. Dengan menerapkan salah satu Metode *Decision Tree* yaitu UCB1 untuk menentukan pemilihan strategi dalam *AI* berbasis PC. Berikut adalah hal-hal yang dijelaskan pada bab ini, antara lain pengertian mengenai teori UCB1, *Game*, *Artifical Intelegence* (*AI*), C#, dan Unity 3D. Berikut adalah penjelasan mengenai teori-teori tersebut.

2.1 UCB1

UCB1 merupakan salah satu metode yang mempunyai potensi tinggi dalam penerapannya di berbagai *genre game* (Sturtevant R.N., 2015). UCB1 sendiri merupakan suatu *Bandit Algorithm*, dikarenakan pengambilan keputusannya yang dimodelkan dengan *n-arm bandits* (*Slot Machine*), dimana cara kerjanya yaitu mencari hasil perhitungan yang terbaik dari setiap *arm*. Pada awalnya UCB1 tidak cocok untuk *Role Play System*, tetapi setelah mendapatkan sedikit modifikasi, UCB1 berhasil menghasilkan pengikatan yang baik. Setelah Mengalami Modifikasi, UCB1 pertama kali diterakan pada permainan Gunting-Batu-Kertas yang menggunakan *three-armed bandit*. Dimana untuk setiap *arm*, UCB1 memantain rata-rata perhitungan *payoff* yang didapat ketika memainkan *arm*. Perhitungan di lakukan dengan rumus sebagai berikut:

$$v(i) = x(i) + \sqrt{\frac{k \ln(t)}{c(i)}} \qquad (2.1)$$

Keterangan :

v(i) [float]: *payoff* yang didapat dari hasil perhitungan tiap *arm*

x(i) [real]: hasil dari penggunaan *arm*

c(i) [real]: berapa kali *arm* tersebut dijalankan

k [real]: bilangan konstanta

t [real]: waktu ketika menjalankan UCB1

Pada Rumus (2.1), bilangan konstanta diisi dengan menggunakan nilai *default* 2. Besar atau kecilnya nilai konstanta *(k)* akan berpengaruh terhadap hasil *payoff (v)*. Jika semakin besar nilai konstanta, maka perpindahan antar *arm* akan lebih cepat. Sebaliknya jika nilai konstanta semakin kecil, maka perpindahan antar *arm* akan lebih lama. Nilai *t* dalam perhitungan UCB1 akan selalu ditambah setiap kali perhitungan UCB1 dijalankan (Sturtevant R.N., 2015).

Berikut adalah salah satu contoh hasil perhitungan dari permainan Gunting-Batu-Kertas yang perhitungannya menggunakan UCB1. Diasumsikan bahwa musuh akan selalu mengeluarkan Batu.

	Rock			Paper			Scissors		
Time	c(I)	x(I)	v(I)	c(I)	x(I)	v(I)	c(I)	x(I)	v(I)
0	**0**	**0**	∞	0	0	∞	0	0	∞
1	1	0	0.00	**0**	**0**	∞	0	0	∞
2	1	0	1.18	1	1	2.18	**0**	**0**	∞
3	1	0	1.48	**1**	**1**	**2.48**	1	−1	0.48
4	1	0	1.67	**2**	**1**	**2.18**	1	−1	0.67
5	1	0	1.79	**3**	**1**	**2.04**	1	−1	0.79
6	1	0	1.89	**4**	**1**	**1.95**	1	−1	0.89
7	**1**	**0**	**1.97**	5	1	1.88	1	−1	0.97

Table 2.1 Tabel Contoh Perhitungan UCB1 (Sturtevant R.N., 2015)

Pada Table 2.1 di simulasikan bahwa Player akan mendapatkan asumsi terbaik yaitu untuk Kertas dari Time ke-2 sampai ke-6. Untuk Time 1,2,3, UCB1 akan mencoba semua kemungkinan terlebih dahulu untuk mendapatkan hasil *payoff* pertama.

2.2 Game

Game adalah sebuah sistem dimana didalamnya *player* akan berhadapan dengan konflik buatan yang diatur berdasarkan rule tertentu dan akan mengeluarkan sebuah hasil yang dapat diukur (Salen & Zimmerman, 2003). Sebuah Game harus memiliki *goal* yang harus dicapai *player* dan dalam mencapai *goal* tersebut *player* harus di libatkan dalam proses menghasilkan *goal* tersebut dimana *player* harus merasa memiliki kontrol terhadap apa yang akan terjadi di

dalam game tersebut (Schwab, B., 2014). Sehingga dapat dikatakan sebuah game adalah sebuah sistem yang di dalamnya player akan membuat keputusan untuk mencapai goal dari game tersebut dengan dibatasi oleh *rule-rule* yang ada di game. Sebuah *game* juga tidak lepas dari *NPC* (*Non-player Character*), dimana *NPC* akan sangat berperan dalam semua situasi dan kondisi yang berada didalam *game*. *NPC* merupakan object atau sebuah karakter yang digerakkan oleh *AI* di dalam sebuah game. *NPC* juga harus dapat bertindak layaknya sifat dari karakter tersebut. Sehingga *NPC* juga harus dapat bertindak dan berperilaku lebih *real* terhadap *player* dan situasi (Welsh Rich, 2015).

Dalam kamus bahasa Indonesia "*Game*" diartikan sebagai permainan. Permainan merupakan bagian dari bermain dan bermain juga bagian dari permainan keduanya saling berhubungan. Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, *play* dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, di sini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan (Avedon, E. M. and Smith, B. S., 1971). *Game* bertujuan untuk menghibur, biasanya *game* banyak disukai oleh anak–anak hingga orang dewasa. *Games* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam game terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Tetapi *game* juga bisa merugikan karena apabila kita sudah kecanduan game kita akan lupa waktu dan akan mengganggu kegiatan atau aktifitas yang sedang kita lakukan.

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*Intellectual Playability Game*) yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya (Dill Kevin, 2014). Dalam *game*, ada target-target yang ingin dicapai pemainnya. Berdasarkan jenis "*Platform*" atau alat yang digunakan, *game* dibagi menjadi beberapa kategori, yaitu:

- *Arcade games*, yaitu yang 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 *games* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa "masuk" dan "menikmati", seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya).
- *PC Games* , yaitu video *game* yang dimainkan menggunakan *Personal Computers.*
- *Console games*, yaitu video *games* yang dimainkan menggunakan *console* tertentu, seperti *Playstation 2, Playstation 3, XBOX 360* dan *Nintendo Wii*.
- *Handheld games*, yaitu yang dimainkan di *console* khusus *video game* yang dapat dibawa kemana-mana, contoh *Nintendo DS* dan *Sony PSP*.
- *Mobile games*, yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau PDA.

Berdasarkan genre permainannya**,** game dapat dibagi menjadi beberapa kategori (GameRanking-Video Game Reviews from around the Internet, 2016), yaitu:

- *Action – Shooting*: video *game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga *timing*.
- *Fighting* (pertarungan): Jenis ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan permainan, pengenalan karakter dan *timing*. Dan berbeda seperti game aksi pada umumnya yang hanya melawan *Artificial Intellegence*. Pemain jenis *fighting game* juga dapat bermain dengan melawan pemain lainnya. Sebagai contoh: *Street Fighter, Tekken, Mortal Kombat, Soul Calibur* dan *King of Fighter.*
- *Action – Adventure*: *Game* jenis ini sudah berkembang jauh hingga menjadi genre campuran *action beat-em up*, dan sekarang jenis ini cenderung untuk memiliki visual 3D dan sudut pandang orang ke tiga. Sebagai contoh: *Tomb Rider, Grand Theft Auto* dan *Prince of Persia.*

- *Adventure*: Bedanya dengan jenis video *game* aksi-petualangan, refleks dan kelihaian pemain dalam bergerak, berlari, melompat hingga memecut atau menembak tidak diperlukan di sini. *Video Game* murni petualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat.
- Simulasi Konstruksi dan manajemen: Video *Game* jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Dari mencari jodoh dan pekerjaan, membangun rumah, gedung hingga kota, mengatur pajak dan dana kota hingga keputusan memecat atau menambah karyawan. Dunia kehidupan rumah tangga sampai bisnis membangun konglomerasi, dari jualan limun pinggir jalan hingga membangun laboratorium *cloning*. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Sebagai contoh: *Sim City, The Sims* dan *Tamagotchi.*
- *Role Playing*: Video *game* cenderung bermain peran, memiliki penekanan pada tokoh atau peran perwakilan pemain di dalam permainan. Karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain dalam berbagai parameter yang ditentukan dengan naiknya level, baik dari status kepintaran, kecepatan dan kekuatan karakter.
- Strategi: Video *game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana. Video game strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal sekelompok orang dengan berbagai jenis tipe kemampuan, sampai kendaraan, bahkan hingga pembangunan berbagai bangunan, pabrik dan pusal pelatihan tempur, tergantung dari tema ceritanya. Pemain *game* strategi melihat dari sudut pandang lebih meluas dan lebih kedepan dengan waktu permainan yang biasanya lebih lama dan santai dibandingkan game

action. Unsur-unsur permainannya biasanya berkisar sekitar, prioritas pembangunan, mencari dan memanfaatkan sumberdaya (uang, besi, kayu dan sebagainya), hingga ke pembelian dan *upgrade* pasukan atau teknologi. Game jenis ini terbagi atas:

- *Real time Strategy, game* berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang di ambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateginya. Contoh: *Starcraft, Warcraft, dan Command* and *Conquer.*
- *Turn based Strategy, game* yang berjalan secara bergiliran, saat kita mengambil keputusan dan menggerakan pasukan, saat itu pihak lawan menunggu, begitu pula sebaliknya. Sebagai contoh: *Front Mission, Super robot wars, Final Fantasy tactics, Heroes of might* and *magic, Master of Orion.*

- Puzzle: Video *game* jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, hingga mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini. Sering pula permainan jenis ini adalah juga unsur permainan dalam video *game* petualangan maupun *game* edukasi. Sebagai Contoh: *Tetris, Minesweeper, Bejeweled, Sokoban* dan *Bomberman.*
- Simulasi kendaraan: Video *Game* jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, meskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut. Video *game* simulasi kendaraan yang sempat tenar di tahun 90-an ini mengajak pemain untuk menaiki kendaraan dan berperang melawan kendaraan lainnya. Dan kebanyakan diantaranya memiliki judul sama dengan nama kendaraannya. Contoh: *Apache 64, Comanche, Abrams, YF-23* dan *F-16 fighting eagle.*

- Olahraga*:* Singkat padat jelas, bermain sport di PC atau konsol. Permainan ini dibuat dengan tingkat realistis yang tinggi. Contohnya adalah: *Winning Eleven, NBA, FIFA, John Madden NFL, Lakers vs Celtics* dan *Tony hawk pro skater*.

2.3 Artificial Intelegence (AI)

Artificial Intelligence adalah sebuah komputasi yang membuat komputer dapat mengetahui sesuatu, berpikir, dan bertindak (Dill Kevin, 2014). Dalam sebuah game, Artificial Intelligence ditekankan pada proses berpikir mesin agar serupa dengan manusia. Tujuan *Artificial Intelligence* didalam sebuah game yaitu untuk meningkatkan tingkat kesulitan dari *game* tersebut tetapi kemampuan *artificial intelligince* perlu di batasi agar tidak melebihi kemampuan dari manusia agar tercipta keseimbangan dalam sebuah *game* sehingga membuat game lebih menarik. *AI* juga harus mempunyai 3 macam *decision problem*, yaitu *action, outcomes, dan preferences,* dimana ketiganya tersebut akan mempengaruhi bagaimana *AI* akan bertindak terhadap *player* (Tadelis Steven, 2013). Kecerdasan Buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas (H. A. Simon, 1987).

Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia (Rich and Knight, 1991). Kecerdasan Buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode *heuristic* atau dengan berdasarkan sejumlah aturan (Encyclopedia Britannica). Berbagai litelatur mengenai kecerdasan buatan menyebutkan bahwa ide mengenai kecerdasan buatan diawali pada awal abad 17 ketika Rene Descartes mengemukakan bahwa tubuh hewan bukanlah apa-apa melainkan hanya mesin-mesin yang rumit. Kemudian Blaise Pascal yang menciptakan mesin penghitung digital mekanis pertama pada 1642. Selanjutnya pada abad 19, Charles Babbage dan Ada Lovelace bekerja pada mesin penghitung mekanis yang dapat diprogram.

Tahun 1950-an adalah periode usaha aktif dalam AI. Program AI pertama yang bekerja ditulis pada 1951 untuk menjalankan mesin Ferranti Mark I di University of Manchester (UK): sebuah program permainan naskah yang ditulis oleh Christopher Strachey dan program permainan catur yang ditulis oleh Dietrich Prinz. John McCarthy membuat istilah "Kecerdasan Buatan" pada konferensi pertama pada tahun 1956, selain itu dia juga menemukan bahasa pemrograman Lisp. Alan Turing memperkenalkan "*Turing test*" sebagai sebuah cara untuk mengoperasionalkan test perilaku cerdas. Joseph Weizenbaum membangun ELIZA, sebuah chatterbot yang menerapkan psikoterapi (Negnevitsky, M., 2011).

Selama tahun 1960-an dan 1970-an, Joel Moses mendemonstrasikan kekuatan pertimbangan simbolis untuk mengintegrasikan masalah di dalam program Macsyma, program berbasis pengetahuan yang sukses pertama kali dalam bidang matematika. Marvin Minsky dan Seymour Papert menerbitkan Perceptrons, yang mendemostrasikan batas jaringan syaraf sederhana dan Alain Colmerauer mengembangkan bahasa komputer Prolog. Ted Shortliffe mendemonstrasikan kekuatan sistem berbasis aturan untuk representasi pengetahuan dan inferensi dalam diagnosa dan terapi medis yang diyakini sebagai sistem pakar pertama. Hans Moravec mengembangkan kendaraan terkendali komputer pertama untuk mengatasi jalan yang mempunyai rintangan secara mandiri.

Soft computing merupakan sebuah inovasi dalam membangun sistem cerdas yaitu sistem yang memiliki keahlian seperti manusia pada domain tertentu, mampu beradaptasi dan belajar agar dapat bekerja lebih baik jika terjadi perubahan lingkungan. *Soft computing* mengeksploitasi adanya toleransi terhadap ketidaktepatan, ketidakpastian, dan kebenaran parsial untuk dapat diselesaikan dan dikendalikan dengan mudah agar sesuai dengan realita (Lofti A Zadeh, 1994). Metodologi-metodologi yang digunakan dalam Soft computing adalah :

- Logika Fuzzy/Fuzzy Logic (mengakomodasi ketidaktepatan).
- Jaringan Syaraf Tiruan/Neurall Network (menggunakan pembelajaran).
- Probabilistic Reasoning (mengakomodasi ketidakpastian).
- Algoritma Genetika/Evolutionary Computing (optimasi).

2.4 C# Code

C# adalah bahasa pemrograman baru yang diciptakan oleh Microsoft yang dikembangkan dibawah kepemimpinan Anders Hejlsberg yang telah menciptakan berbagai macam bahasa pemrograman termasuk Borland Turbo C++ dan Borland Delphi (Anders Hejlsberg, 2004). Bahasa C# juga telah di standarisasi secara internasional oleh ECMA. Seperti halnya bahasa pemrograman yang lain, C# bisa digunakan untuk membangun berbagai macam jenis aplikasi, seperti aplikasi berbasis *windows* (*desktop*) dan aplikasi berbasis web serta aplikasi berbasis *web services*.

C# juga merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework (Andrew Stellman, 2007). Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya (Seperti Java, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan. Menurut standar *ECMA-334 C# Language Specification*, nama C# terdiri atas sebuah huruf Latin **C** yang diikuti oleh tanda pagar yang menandakan angka **#**. Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik dan tanda pagar # tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam *keyboard* standar. C# juga memiliki kelebihan dan kekurangan (Anders Hejlsberg, 2004), yaitu:

- *Flexible*: C# program dapat di eksekusi di mesin computer sendiri atau di transmiskan melalu web dan di eksekusi di computer lainnya.
- *Powerful*: C# memiliki sekumpulan perintah yang sama dengan C++ yang kaya akan fitur yang lengkap tetapi dengan gaya bahasa yang lebih diperhalus sehingga memudahkan penggunanya.
- *Easier to use*: C# memodifikasi perintah yang sepenuhnya sama dengan C++ dan memberitahu dimana letak kesalahan kita bila ada kesalahan dalam aplikasi, hal ini dapat mengurangi waktu kita dalam mencari *error* Visually oriented: The .NET *library code* yang digunakan oleh C# menyediakan bantuan yang dibutuhkan untuk membuat tampilan yang

complicated dengan *frames, dropdown, tabbed windows, group button, scroll bar, background image* dan lainnya.

- *Secure*: Semua bahasa pemprograman yang digunakan untuk kebutuhan internet mesti memiliki *security* yang benar-benar aman untuk menghindari aksi kejahatan dari pihak lain seperti *hacker*, C# memiliki segudang fitur untuk menanganinya.

  Memory management lebih mudah karena adanya *garbage collector*, yang membebaskan *memory* secara otomatis sehingga dapat mencegah *memory leak.*

  Type safe, konversi implisit dari tipe data hanya men-*support* turunan dan operasi dari tipe data yang melebar dan ini dideteksi ketika *compile*. Banyak fungsi yang tersedia di *Base Class Library .NET Framework*. *.NET Framework* berkembang cepat dan semakin banyak fitur yang membuat produktivitas kita bertambah. Untuk pengembangan aplikasi bisnis, umum atau enterprise, penggunaan C# akan lebih produktif daripada bila menggunakan C++. Bahasa C# masih merupakan turunan dari bahasa C, tetapi lebih mudah dan produktif seperti *Visual Basic* dengan tetap mempertahankan fleksibilitas dan "*power*" dari bahasa C.
- Banyaknya operator dan fleksibilitas penulisan program terkadang membuat user tidak dapat menguasai sepenuhnya. Bagi pemula pada umumnya akan kesulitan menggunakan pointer.

2.5 Unity 3D

Unity 3D adalah sebuah *game engine* yang berbasis *cross-platform*. Unity dapat digunakan untuk membuat sebuah *game* yang bisa digunakan pada perangkat komputer, *mobile phone, PS3* dan *X-BOX* (Suvak, J., 2014). Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat game, arsitektur bangunan dan simulasi. Unity bisa untuk *games PC* dan *games online*. Untuk *games online* diperlukan sebuah *plugin*, yaitu *Unity Web Player*, sama halnya dengan *Flash Player* pada *Browser.*

Unity tidak dirancang untuk proses desain atau modelling, dikarenakan unity bukan *tool* untuk mendesain. Unity juga mendukung bergai macam fitur, yaitu: *audio reverb zone, particle effect*, dan *Sky Box* untuk menambahkan langit.

Fitur *scripting* yang disediakan Unity 3D, mendukung tiga bahasa pemrograman, yaitu: JavaScript, C# dan Boo. Unity juga merupakan *tool* yang cukup *Flexible* and *EasyMoving*. Untuk melakukan *rotating* dan *scaling objects* juga dapat digunakan dengan mudah. Begitu juga dengan *Duplicating*, *removing*, dan *changing properties*. *Visual Properties Variables* yang didefinisikan dengan *scripts* ditampilkan pada *editor*.