BAB II

LANDASAN TEORI

2.1 Simulasi Permainan

Menurut Huang, Yao, Li, dan Gu (2012), simulasi permainan adalah sistem perangkat lunak komputer yang dikembangkan untuk mempelajari bagaimana mendapatkan kondisi yang menguntungkan dalam permainan. Cara mencari dan menemukan strategi yang sesuai dalam konteks situasi permainan pada suatu kondisi adalah salah satu masalah penting selama keseluruhan proses permainan. Simulasi permainan dikembangkan dan didukung oleh kombinasi teori permainan, kecerdasan buatan, dan teknologi perangkat lunak komputer, dan bagaimana mencari dan memilih strategi yang efektif berdasarkan situasi permainan yang dilakukan adalah bagian penting dari simulasi permainan.

Dalam hal antarmuka, tujuan dari desain antarmuka adalah untuk membuat antarmuka dimana semua orang, baik yang berpengalaman maupun tidak, dapat menggunakannya (Chandra et al, 2006). Penelitian Chandra menggunakan antarmuka yang sederhana untuk simulasi permainan, dimana antarmuka hanya memberikan informasi fungsionalitas dari permainan tersebut.

Berdasarkan penelitian oleh Sacerdotianu, Ilie, dan Badica (2011), simulasi permainan memiliki manfaat berupa abstraksi dan modularitas yang diberikan kepada pengembang perangkat lunak. Dengan menggunakan abstraksi, pengembang perangkat lunak dapat mendefinisikan peran dan fungsi abstrak kerangka kerja, sementara modularitas berguna untuk dengan mudah mendefinisikan aplikasi game tertentu yang terdiri dari modul yang digabungkan secara longgar dan sangat kohesif yang dapat digabungkan baik pada *compile-*

time maupun saat *run-time*. Abstraksi dan modularitas menawarkan generalitas dan fleksibilitas yang dibutuhkan untuk pengembangan kerangka permainan yang generik. Keduanya didemonstrasikan dengan berbagai permainan sederhana yang disimulasikan di atas kerangka permainan generik.

2.2 Permainan Kartu Digital

Computer Card Games (CCGs), atau permainan kartu digital, adalah bentuk modern dari permainan kartu yang merupakan genre permainan yang telah ada selama berabad-abad. Secara spesifik, sejarawan melacak permainan kartu telah ada sejak abad ke-10 (Crawford, 1982). Permainan kartu juga digunakan untuk tujuan pendidikan sejak zaman dulu. Contohnya, permainan kartu telah digunakan untuk membantu beberapa murid dalam memahami dan menerapkan aspek dasar psikologi perkembangan. Meskipun apa yang dipelajari siswa melalui permainan kartu terlepas dari preferensi belajar siswa, menggabungkan alat pembelajaran inovatif seperti permainan kartu meningkatkan pengalaman belajar mereka (Barclay dkk., 2011).

Permainan kartu abad pertengahan sangat berbeda dengan yang muncul di masa *Age of Enlightenment*. Pada awalnya, tujuan dari permainan adalah perolehan kombinasi kartu, yang dengan kombinasi yang baik, memiliki nilai yang ditentukan hanya jika dikelompokkan dengan orang lain dalam berbagai cara. Prinsip semacam itu bisa dibilang mencerminkan masyarakat hierarkis abad pertengahan di mana sifat-sifat orang ditentukan oleh posisinya dalam kelompok sosial. Maraknya individualisme diterjemahkan ke dalam permainan kartu melalui pengenalan kartu truf, sebuah kartu yang mengubah urutan normal kartu. Pertama, kartu dipilih secara kebetulan dan kemudian melalui penawaran, kartu

truf berkorelasi dengan masyarakat yang lebih *mobile*. Nilai kartu tidak lagi berasal dari posisinya, tapi dari jumlah dan rangkingnya (Pisac, 2013).

Permainan kartu juga dilihat sebagai permainan sederhana, dimana pemain menggunakan kartu dengan karakteristik tertentu dan berkonsentrasi pada pembentukan kombinasi kartu yang tepat (Crawford, 1982), sambil mempertimbangkan kemungkinan, klasifikasi, pengelompokan, perbandingan dan pencocokan masalah. Faktanya, kegiatan klasifikasi sangat penting bagi permainan kartu karena para pemain memahami konsep pembelajaran melalui pembentukan kelompok kartu yang sesuai, sehingga membentuk pemikiran yang kritis, refleksi, dan pembelajaran berbasis masalah (Kordaki, 2015).

Selain itu, karena kesederhanaan dalam peralatan dan peraturan, permainan kartu berbasis pendidikan dapat digunakan sebagai titik masuk untuk memotivasi dan melibatkan pemain pemula dalam pengalaman belajar berbasis permainan (Gosper dan McNeil, 2012). Bahkan, permainan kartu yang dirancang dengan tepat memiliki potensi untuk meningkatkan kemampuan komunikatif pemain melalui interaksi dengan pemain lain (Bochennek dkk., 2007). Pada percobaan yang dilakukan oleh Eric Zhi Feng Liu dan Po-Kuang Chen (2013) tentang pengaruh pembelajaran melalui permainan kartu kepada murid, telah disimpulkan bahwa peserta menunjukkan sikap positif terhadap penggunaan permainan kartu dan merasa bahwa pendekatan tersebut berkontribusi terhadap pembelajaran. Mayoritas siswa menerima metode pembelajaran ini dan berharap untuk terus menggunakan pendekatan ini di masa depan. Para siswa juga menyatakan bahwa belajar dengan permainan kartu dapat membantu para siswa mendapatkan pengetahuan ilmiah dan bahwa metode pembelajaran berbasis *game* meningkatkan minat mereka terhadap pelajaran.

Permainan kartu tidak dapat dipelajari secara terpisah. Permainan kartu mencerminkan, membentuk dan bertemu dengan konteks sosialnya dengan beragam cara. Jika permainan dipahami dengan cara normatif sebagai kegiatan yang terpisah dari kehidupan biasa dan tidak terbebani dengan pertimbangan material, maka kualitas permainan akan berkurang menjadi sekedar representasi belaka mengenai konteks dari mana mereka muncul. Antropologi telah menghasilkan berbagai penelitian dengan tujuan untuk mengkonseptualisasikan batas-batas kategoris ini. Selain itu, merasakan kegembiraan dan ketegangan sebagai satu-satunya emosi dalam bermain membuat orang bermain dengan menentukan analisis dan berpikir melalui kombinasi kartu terbaik, jauh dari sifat permainan yang biasa (Pisac, 2013).

Salah satu contoh permainan kartu yang saat ini sedang populer adalah Cardfight!! Vanguard. Cardfight!! Vanguard adalah sebuah permainan menggunakan media kartu yang diciptakan oleh Bushiroad Trading Card Company (Theddy, 2015). Permainan ini berasal dari Jepang dan diciptakan oleh berbagai pengarang komik bergenre permainan kartu. Permainan kartu ini menjadi terkenal dan banyak peminat sejak Cardfight!! Vanguard memperoleh serial *anime* (kartun animasi yang berasal dari Jepang) yang diproduksi oleh TMS Entertainment, dan disiarkan oleh TV Tokyo semenjak 8 Januari 2011. Berbagai video promosi juga telah diunggah melalui internet dan televisi untuk mempromosikan produk Cardfight!! Vanguard.

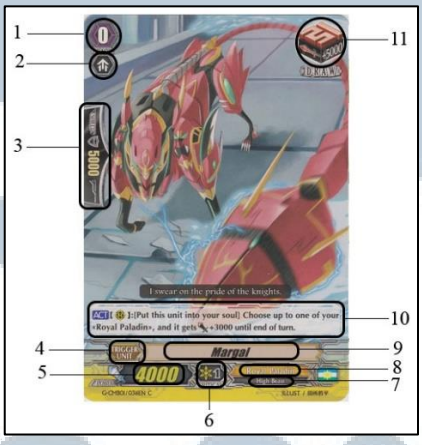
2.3 Permainan Cardfight!! Vanguard

Cardfight!! Vanguard adalah sebuah permainan dimana para pemain memakai "Vanguard" sebagai diri mereka dalam permainan. Desain kartu, definisi lambang pada kartu, arena permainan, beserta aturan cara bermain

Cardfight!! Vanguard tercatat di dalam Cardfight!! Vanguard Playbook (2015) yang dipublikasikan oleh Bushiroad.

2.3.1 Desain Kartu

Kartu yang digunakan dalam permainan Cardfight!! Vanguard hanyalah satu jenis kartu, yaitu kartu unit, tidak seperti permainan kartu lainnya seperti Magic The Gathering yang memiliki jenis kartu *Land* dan *Creature* (Cowling, Ward dan Powley, 2012). Berdasarkan Cardfight!! Vanguard Playbook (2015), desain kartu Cardfight!! Vanguard seperti yang diilustrasikan pada Gambar 2.1 adalah sebagai berikut.



Gambar 2.1 Desain kartu Cardfight!! Vanguard (Cf-vanguard.com, 2017)

1. *Grade*, menunjukkan tingkat kekuatan kartu, dan membatasi kartu apa yang bisa dimainkan saat melakukan *ride* dan *call* secara normal.
2. *Skill Icon*, menunjukkan kemampuan unit, dan skill icon berbeda untuk setiap *grade*.
3. *Shield*, angka yang menunjukkan kemampuan bertahan sebagai guardian. Unit dengan *shield* yang besar lebih berguna untuk bertahan.
4. *Card Type*, selain unit normal, ada juga unit *trigger* dengan simbol *trigger*.
5. *Power*, angka yang menunjukkan kekuatan unit dalam pertarungan, *power* yang besar lebih kuat baik dalam menyerang maupun bertahan.

6. *Critical*, angka yang menunjukkan jumlah *damage* yang bisa diberikan pada Vanguard lawan.
7. *Race*, atau ras dari kartu. Terkadang mengarah untuk kemampuan atau efek kartu.
8. *Clan*, atau klan dari kartu. Sama seperti *race*, terkadang mengarah untuk kemampuan atau efek kartu.
9. *Card Name*, atau nama dari kartu.
10. *Ability Text*, yaitu kemampuan spesial kartu beserta kondisi, biaya, dan efeknya.
11. *Trigger Icon*, lambang yang menunjukkan tipe dari efek yang didapat ketika kartu ini terbuka saat *drive check* atau *damage check*.

2.3.2 Definisi Lambang pada Kartu

Lambang pada kartu menyimpan informasi penting dalam permainan. Menurut Cardfight!! Vanguard Playbook (2015), salah satu jenis lambang yang ada dalam permainan adalah *Skill Icons* seperti pada Gambar 2.2.

Gambar 2.2 Jenis *Skill Icons* (Cardfight!! Vanguard Playbook, 2015)

1. *Boost*, ketika unit di baris depan menyerang, unit dengan lambang ini di baris belakang dapat menambah *power* unit di depannya sesuai dengan *power* miliknya.
2. *Intercept*, ketika fase bertahan, unit dengan lambang ini di baris depan dapat dipindahkan ke *guardian circle* untuk membantu pertahanan.
3. *Twin Drive*, ketika Vanguard dengan lambang ini menyerang, pemain melakukan *drive check* sebanyak dua kali, dan *drive check* dilakukan satu per satu.

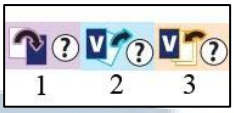
4. *Triple Drive*, ketika Vanguard dengan lambang ini menyerang, pemain melakukan *drive check* sebanyak tiga kali, dan *drive check* dilakukan satu per satu.

Selain *skill icons*, berdasarkan Cardfight!! Vanguard Playbook (2015), ada juga lambang yang mengindikasi jenis dari kemampuan atau *ability* kartu dalam *Ability Text* seperti pada Gambar 2.3.

Gambar 2.3 Jenis *Ability Icons* (Cf-vanguard.com, 2017)

1. *Activated Ability*, yaitu kemampuan yang dapat dijalankan secara bebas dengan membayar *cost*. Kemampuan ini dapat diaktifkan berkali-kali selama *cost* terpenuhi.
2. *Automatic Ability*, yaitu kemampuan yang dijalankan ketika suatu kondisi terpenuhi. Jika kondisi terpenuhi beberapa kali selama permainan, kemampuan akan aktif selama beberapa kali pula. *Automatic ability* memiliki *cost* dan opsional (pemain dapat memilih untuk mengaktifkannya atau tidak).
3. *Continuous Ability*, yaitu kemampuan dimana jika tidak ada kondisi yang disebutkan di dalam teks, akan terus dijalankan selama kartu tersebut ada dalam arena permainan. Jika ada kondisi, kemampuan akan aktif selama kondisinya terpenuhi.

Special Actions Icon untuk mengaktifkan kemampuan berbeda, dan cara membayar *cost*-nya pun berbeda. Menurut Cardfight!! Vanguard Playbook (2015), ada tiga jenis *Special Actions Icon* seperti yang diilustrasikan pada Gambar 2.4.



Gambar 2.4 Jenis *Special Actions Icons* (Cf-vanguard.com, 2017)

1. *Counter Blast*, dibayar dengan menutup 1 kartu atau lebih di *damage zone*.
2. *Soul Blast*, dibayar dengan mengeluarkan 1 kartu atau lebih dari *soul* ke *drop zone*.
3. *Soul Charge*, dibayar dengan memasukkan 1 kartu atau lebih dari kartu teratas dalam *deck* ke dalam *soul*.

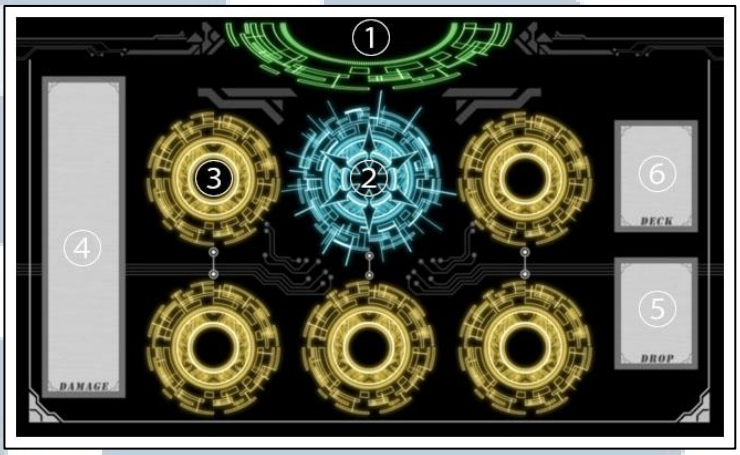
Ada beberapa kartu, yaitu unit *trigger*, yang memiliki *trigger icon* dalam desain kartunya. Jika pemain mendapatkan *trigger*, maka pemain tersebut dapat memilih satu unit untuk mendapat tambahan *power* sebanyak 5000 poin, dan mendapatkan kemampuan tambahan sesuai dengan jenis *trigger* yang didapat. *Trigger* yang ada, seperti yang diilustrasikan pada Gambar 2.5, dibagi menjadi 4 jenis.

Gambar 2.5 Jenis *Cost Icons* (Cf-vanguard.com, 2017)

1. *Critical* untuk menambah jumlah *critical* yang dimiliki oleh satu unit.
2. *Stand* untuk mengubah posisi satu *rear-guard* menjadi *stand* agar dapat menyerang lagi.
3. *Draw* untuk menarik satu kartu tambahan dari *deck*.
4. *Heal* untuk menyembuhkan 1 *damage* pemain. Pemain hanya dapat memasukkan maksimal 4 *Heal trigger* dalam *deck*-nya, sedangkan untuk *trigger* lainnya bebas selama memiliki tepat 16 *trigger* dalam *deck*.

2.3.3 Arena Permainan

Desain arena permainan pada Cardfight!! Vanguard seperti pada Gambar 2.6 memiliki penjelasan sebagai berikut.



Gambar 2.6 Arena permainan Cardfight!! Vanguard (Cf-vanguard.com, 2017)

1. *Guardian Circle*, tempat untuk menaruh guardian saat bertahan dari serangan lawan. Pemain dapat memanggil lebih dari 1 guardian di saat yang bersamaan.
2. *Vanguard Circle*, dimana Vanguard berada. Kartu-kartu yang berada di bawah Vanguard dinamakan *soul*, dan digunakan untuk membayar *cost* seperti *soul blast*.
3. *Rear-guard Circle*, tempat untuk memanggil *rear-guard* yang akan bertarung demi pemain. Ada 5 *rear-guard circle*, dan hanya 1 *rear-guard* yang dapat diletakkan di tiap *rear-guard circle*.
4. *Damage Zone*, ketika terkena serangan dan menerima *damage*, maka kartu *damage* diletakkan disini. Jika ada 6 kartu di *damage zone*, maka pemain kalah.
5. *Drop Zone*, tempat untuk menaruh kartu yang telah dihilangkan dari arena.

6. *Deck*, tempat untuk menaruh *deck* atau tumpukan kartu pemain selama permainan.

2.3.4 Aturan Permainan

Permainan selesai ketika salah satu kondisi terpenuhi, dan kondisi yang diperlukan adalah sebagai berikut.

1. Saat Vanguard terkena serangan, maka pemain mendapatkan *damage* sesuai jumlah *critical* serangan tersebut. Pemain yang mendapatkan 6 *damage* terlebih dahulu dinyatakan kalah dalam permainan.
2. Selain dari jumlah *damage*, pemain yang tidak memiliki kartu tersisa di dalam *deck*-nya juga dinyatakan kalah.

Permainan dimulai dengan kedua pemain memilih satu kartu dari *deck* miliknya untuk menjadi Vanguard pertama, atau *first Vanguard*, dan menarik 5 kartu dari *deck*. Setelah itu, permainan akan berlanjut ke putaran salah satu pemain, dan tiap putaran dibagi menjadi berbagai fase, sebagai berikut.

1. Setiap putaran, tiap pemain mengubah *state* semua kartunya menjadi *stand*, lalu menarik satu kartu dari *deck*-nya. Fase ini disebut dengan *stand and draw phase*.
2. Kemudian, pemain dapat menaikkan tingkat atau *grade* dari Vanguard miliknya untuk meningkatkan *power* Vanguard. Fase ini disebut dengan *ride phase*. *Ride* dilakukan dengan memilih satu kartu dari tangan dan kemudian menurunkannya di *Vanguard Circle*, menjadikannya sebagai Vanguard yang baru sementara Vanguard yang ada sebelumnya masuk ke dalam *soul*.
3. Setelah *ride phase*, permainan berlanjut ke fase *main phase*, dimana pemain dapat memanggil kartu lain sebagai *rear-guard* untuk persiapan

menyerang. Pemain hanya dapat memanggil *rear-guard* yang memiliki *grade* di bawah *grade* Vanguard-nya. Selama bermain, pemain tidak hanya memakai Vanguard saja sebagai bagian dari permainan, melainkan menggunakan *rear-guard* sebagai "teman" dan juga menambah jumlah serangan.

4. Jika pemain sudah siap untuk melakukan serangan, pemain dapat masuk ke dalam *battle phase*. Dalam *battle phase*, pemain memilih satu unit di bagian depan, atau *front row*, untuk menyerang satu kartu lawan. Adapun, saat diserang, Vanguard bisa dilindungi dengan cara menaruh kartu di zona pertahanan, atau *guardian circle*. Menaruh kartu di *guardian circle* bisa dari tangan ataupun dari kemampuan *rear-guard* dengan *grade* 2 yang disebut *intercept*. Jumlah kartu yang perlu ditaruh di *guardian circle* tergantung dari *power* serangan dan *shield* dari *guardian*. Unit yang digunakan untuk menyerang diubah posisinya dari semula tegak atau *stand* menjadi miring atau kondisi *rest*. Unit yang bisa menyerang adalah unit yang masih dalam posisi *stand*. Jika yang digunakan untuk menyerang adalah *rear-guard*, maka serangan berlanjut seperti biasa.

   Namun, apabila yang menyerang adalah Vanguard, maka pemain melakukan *drive check* untuk menambah kekuatan serangannya di putaran itu. *Drive check* dilakukan dengan membalikkan satu kartu teratas pada *deck* milik pemain, dan jika mendapatkan kartu dengan simbol *trigger*, maka kemampuan *trigger* dapat dimanfaatkan untuk meningkatkan kekuatan serangan. Jika pemain menerima *damage* dari serangan lawan, maka pemain tersebut melakukan *damage check*, dengan sistem yang sama seperti *drive check*, namun kartunya tidak ke tangan, tetapi ke *damage zone*.

Dari sinilah jumlah *damage* pemain dihitung hingga ada yang menyentuh 6 *damage*.

5. Setelah selesai menyerang, berlanjut pada *end phase*, dimana giliran berpindah kepada lawannya.

2.4 Kecerdasan Buatan

Kecerdasan buatan adalah mesin yang mampu berpikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia (Sutojo dkk., 2011). Kecerdasan Buatan memiliki potensi yang luar biasa untuk mempercepat penemuan ilmiah dalam bidang biologi dan kedokteran, dan untuk mengubah perawatan kesehatan (Ethics of Artificial Intelligence, 2015). Kecerdasan buatan juga dapat diimplementasikan dalam permainan komputer, dan tidak sedikit pula permainan komputer yang diciptakan dengan kecerdasan buatan di dalam permainan. Pada bulan Juni 2015, misalnya, Microsoft memulai Proyek Malmo, sebuah platform pengembangan kecerdasan buatan yang didasarkan pada permainan berbasis "*world-building*" populer yang disebut "Minecraft" yang baru saja dibeli. Pada bulan November 2016, Activision Blizzard, pemilik "Starcraft II", sebuah permainan strategi fiksi ilmiah di mana pemain membangun dan memimpin tentara manusia dan asing, mengumumkan sesuatu yang serupa dengan DeepMind, sebuah perusahaan yang mengembangkan kecerdasan buatan milik Alphabet, perusahaan induk Google.

Kecerdasan buatan, atau Artificial Intelligence (AI), memiliki peran yang penting di masa modern seperti sekarang ini (LinLin dkk., 2017). Dalam permainan, kecerdasan buatan memiliki peran untuk bermain bersama manusia, menggantikan manusia sebagai pemain, dan mengajari pemain baru dengan mendemonstrasikan berbagai strategi dan langkah-langkah dalam suatu

permainan (Kim dkk., 2017). Permainan komputer, yang merupakan salah satu cabang penting dalam mengembangkan AI, mendapat perhatian yang cukup banyak seiring berkembangnya AI. Sistem dalam permainan komputer meliputi representasi papan catur, fungsi evaluasi, dan algoritma pencarian. Saat ini, sistem papan catur umumnya bergaya datar sederhana. Unity Engine adalah salah satu platform untuk pengembangan *game* tipe *synthesizing* dan berbasis pada Unity Technologies. Unity dapat digunakan untuk membangun *game* 3D dan 2D berkualitas tinggi. Sebagai contoh, menurut LinLin dkk. (2017), teknologi Unity 3D dapat digunakan untuk membangun papan catur 3D, dan sebagai hasilnya, efek interaksi antara manusia dan komputer meningkat. Sebaliknya, algoritma pencarian juga diteliti dengan menggabungkan algoritma Monte Carlo Tree Search (MCTS) dengan Algoritma Minimax.

2.5 Monte-Carlo Tree Search

Menurut Imagawa dan Kaneko (2015), Monte-Carlo Tree Search Algorithm, atau MCTS, berhasil mencapai kesuksesan yang luar biasa, terutama dalam *game* Go. Algoritma MCTS juga telah banyak diterapkan di dalam jenis *game* lain; misalnya *game* yang *real-time*; namun, ada beberapa jenis *game* di mana algoritma pencarian minimax klasik yang menggunakan fungsi evaluasi justru mengungguli metode MCTS, misalnya dalam permainan catur.

MCTS sendiri merupakan jenis algoritma *best-first* yang secara iteratif memperluas dan mengevaluasi *game tree* dengan menggunakan simulasi acak yang sering disebut sebagai *playout*. Dalam penelitian Ward dan Cowling (2009) tentang implementasi Monte-Carlo Tree Search dalam permainan kartu, *deck* yang tak terlihat dan juga kartu yang ada dalam *deck* dan tangan lawan merupakan informasi yang tidak dimiliki oleh pemain. *Tree* yang dibangun merupakan

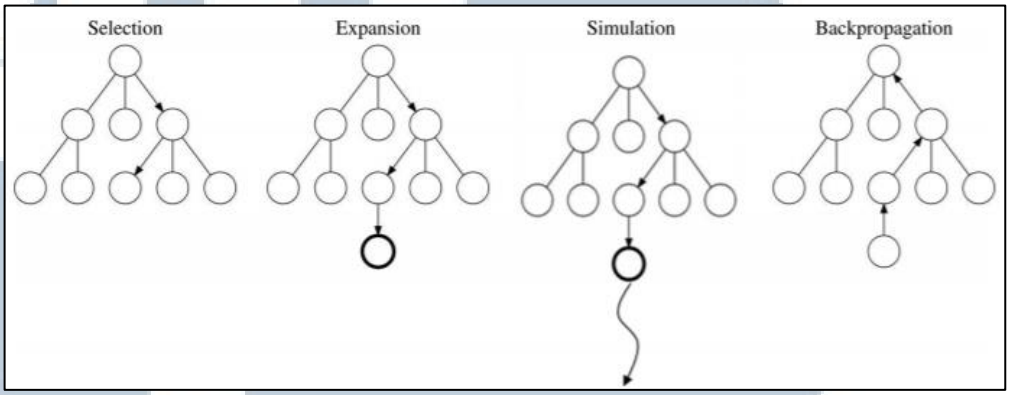
percabangan dari permainan yang mungkin bisa dimainkan dari kartu di tangan pemain, dan bergantung pada *stage* dalam permainan yang telah dicapai.

Sebagai contoh, dalam awal permainan, jumlah langkah yang dapat diambil sangat sedikit dimana biasanya menurut Ward dan Cowling (2009) terbatas hanya untuk memainkan kartu yang sederhana, dan sekitar putaran keempat, jumlah langkah yang dapat diambil meningkat drastis dikarenakan pemain dapat memainkan kartu yang lebih banyak dari sebelumnya. Adapun iterasi MCTS menurut Imagawa dan Kaneko (2015) terdiri dari empat tahap, yaitu tahap seleksi, ekspansi, simulasi, dan terakhir *backpropagation*, seperti yang telah diilustrasikan pada Gambar 2.7.

1. Pertama, pada tahap seleksi, algoritma turun dari *root* ke *leaf* dengan secara rekursif memilih *child* yang paling utama dalam suatu *node*. Berdasarkan penelitian Imagawa dan Kaneko (2015), rumus pada algoritma UCB telah digunakan untuk menentukan *child* yang utama dalam langkah ini.
2. Jika sebuah *leaf* dikunjungi beberapa kali dan mencapai ambang batas, dimana menurut Imagawa dan Kaneko (2015) normalnya dua kali, *leaf* akan melebar atau ekspansi, dan salah satu *child*-nya dipilih secara acak. Ambang batas inilah yang nantinya akan disebut sebagai *arbitrary limit*, dan dapat ditentukan secara bebas oleh peneliti.
3. Pada tahap simulasi, permainan diputar secara acak mulai dari posisi yang sesuai dengan *leaf* tersebut. Permainan biasanya terdiri dari gerakan acak yang seragam. Namun, beberapa penelitian termasuk penelitian Imagawa dan Kaneko (2015), menunjukkan bahwa penggabungan pengetahuan pada *domain* yang bersangkutan akan meningkatkan kinerjanya. Fungsi simulasi

menurut Ward dan Cowling (2009) sendiri adalah untuk mencatat apakah pemain menang atau kalah dalam permainan itu.

4. Akhirnya, hasil dari permainan; antara menang, seri, atau kalah; disebarkan dari *leaf* ke *root* pada tahap *backpropagation*.



Gambar 2.7 Empat tahap iterasi MCTS (Cowling, Ward dan Powley, 2012)

Pendekatan yang menggunakan metode pencarian Monte Carlo memiliki premis dasar yaitu tidak memeriksa semua kemungkinan pergerakan yang tersedia di setiap langkah, tetapi akan dipertimbangkan semua gerakan yang tersedia pada langkah pertama dan kemudian memainkan permainan sampai akhir beberapa kali menggunakan posisi yang dihasilkan dari masing-masing gerakan pertama dan menggunakan *random move selector* atau *pattern generator* untuk memilih gerakan selanjutnya. Idenya adalah bahwa sementara metode tidak dapat memeriksa setiap kemungkinan pergerakan, hasil simulasi permainan akan secara statistik mewakili kekuatan setiap gerakan (Ward dan Cowling, 2009).

Menurut Perez, Mostaghim, Samothrakis, dan Lucas (2013), MCTS adalah algoritma yang membangun sebuah *tree* dalam *memory*. Setiap *node* dalam *tree* menyimpan data statistik yang menunjukkan seberapa sering sebuah gerakan dimainkan dari keadaan tertentu, berapa kali setiap gerakan dimainkan, dan hasil rata-rata yang diperoleh setelah menerapkan suatu gerakan dalam state tertentu.

Tree dibangun secara iteratif dengan mensimulasikan tindakan dalam permainan, lalu membuat pilihan gerakan berdasarkan statistik yang tersimpan di dalam *node*.

Berdasarkan penelitian yang dilakukan oleh Ward dan Cowling (2009), selain menggunakan strategi acak dan aturan main, strategi ketiga yang digunakan untuk membuat keputusan tentang kartu mana yang akan dimainkan yaitu Monte Carlo dengan *bandit based* untuk menemukan langkah terbaik dan diperkuat menggunakan algoritma UCB (Upper Confidence Bounds). Algoritma ini ditujukan untuk memaksimalkan hasil dari beberapa permainan, dan telah ditingkatkan untuk mengatasi *multi-armed bandit problem*. Algoritma dasarnya pada Rumus 2.1 menurut Ward dan Cowling (2009) adalah sebagai berikut.

Inisialisasi: Mainkan tiap mesin sekali.

Loop: Mainkan mesin j yang memaksimalkan *reward*

$$Reward = \bar{x}_j + c\sqrt{\frac{\ln n}{n_j}} \quad \ldots(2.1)$$

dimana $\bar{x}_j$ adalah rata-rata hasil dari mesin j, dan C adalah angka konstan antara 0 sampai 1 yang mengontrol seberapa beragam pencariannya. Nilai yang lebih besar memberikan pencarian yang lebih seragam sementara nilai yang lebih kecil memberikan pencarian yang lebih selektif. n_j menunjukkan berapa kali mesin j telah dimainkan dan $n = \sum_i n_i$ menunjukkan keseluruhan jumlah permainan yang dilakukan. Adapun hasil dihitung berdasarkan aturan dalam Cardfight!! Vanguard Rulebook (2015) yang menyatakan bahwa nilai *power* dari suatu unit penyerang yang lebih besar dari *power* unit yang bertahan akan membuat serangan tersebut *hit*, atau mengenai unit yang bertahan tersebut, sehingga hasil yang didapat bervariasi sesuai dengan perbandingan *power* kedua unit.

MCTS dianggap sebagai algoritma yang dapat digunakan kapan saja, karena mampu memberikan langkah selanjutnya yang valid untuk dipilih kapan

saja. Hal ini merupakan bentuk independen dari sekian banyak iterasi yang bisa dilakukan oleh algoritma, walaupun pada umumnya, iterasi yang lain biasanya menghasilkan hasil yang lebih baik. Hal ini juga berbeda dengan algoritma lain seperti A* dalam *game* yang hanya memerlukan satu pemain, dan Min-Max yang standar untuk permainan dengan dua pemain yang biasanya menyediakan langkah berikutnya hanya setelah selesai melakukan langkah awal. Hal ini membuat MCTS menjadi metode yang sesuai untuk domain yang *real-time*, dimana waktu untuk mengambil keputusan terbatas, dan mempengaruhi jumlah iterasi yang dapat dilakukan (Perez, 2013).

2.6 Black-box Testing

Black-box Testing, atau yang biasanya dikenal sebagai *dynamic testing*, adalah semacam pengujian dimana perilaku dan cara kerja *software* diuji dengan memeriksa *input* dan *output software* tersebut. Dalam *black-box testing*, struktur internal ataupun implementasi *software* tidak diperhatikan. *Input software* diterapkan dan kemudian *output* yang sesuai diperoleh, dan berdasarkan dengan *input* dan *output* tersebut, *bug* dapat diidentifikasi (Nouman, Pervez dan Hasan, 2016).

Test case yang digunakan sebagai *input* mengacu pada aturan yang terdapat dalam Cardfight!! Vanguard Rulebook (2015). Aturan yang membatasi kartu yang dapat dimainkan selama permainan antara lain *grade* yang tertera pada Vanguard menunjukkan batas *grade* unit yang dapat dipanggil ke arena baik dalam *rear-guard circle* maupun *guardian circle*, dimana *grade* unit yang dipanggil harus sama atau lebih rendah dari *grade* Vanguard. Selain itu, *grade* Vanguard juga membatasi kartu yang dapat digunakan untuk melakukan *ride*

pada saat *ride phase*, dimana pemain hanya dapat menggunakan kartu dengan *grade* yang sama atau lebih besar 1 tingkat dari *grade* Vanguard sebelumnya.

Output berupa *action* yang dilakukan oleh kecerdasan buatan. Dari *output* dapat diketahui apakah kecerdasan buatan mengikuti aturan permainan sesuai dengan *rulebook* atau tidak.