PENGEMBANGAN GAME STRATEGI "OUR KINGDOM"

Gunawan[1)], Fandi Halim[2)], Jetson Surya[3)]

Sistem Informasi STMIK MIKROSKIL Medan
Jl. Thamrin No. 140, Sei Rengas II, Medan Area, Medan 20212
Email: gunawan@mikroskil.ac.id[1)], fandi@mikroskil.ac.id[2)], jet.sylvaster@yahoo.co.id[3)]

Abstrak

Game sering dinilai memiliki sisi negatif, mulai dari memiliki konten kekerasan, membuat pemain kecanduan, dan dampak negatif lainnya. Terlepas dari penilaian tersebut, game tentunya merupakan salah satu hiburan yang digemari. Pengembangan ini dilakukan untuk mengembangkan game yang tetap memberikan hiburan yang diharapkan, di samping itu memberikan edukasi sederhana, namun bermanfaat bagi para pemain.

Pengembangan game menggunakan metodologi prototyping dan dirancang dengan bantuan beberapa diagram UML. Game kemudian dibuat menggunakan bahasa pemrograman Action Script 3 dengan teknologi Flash, penyimpanan data menggunakan local shared object. Penentuan jangkauan pergerakan unit menggunakan algoritma Dijkstra, sedangkan pencarian rute (path finding) menggunakan algoritma A (A-Star).*

Hasil dari pengembangan ini adalah sebuah game offline yang memiliki nilai edukasi dengan fokus utama mengalahkan musuh lewat pengaturan strategi. Di sela permainan akan disisipkan nilai edukasi. Pemberian edukasi ini akan disesuaikan dengan alur cerita agar tidak mengganggu fokus utama pemain dalam bermain game.

Kata kunci: *Edukasi, game strategi, prototyping, Dijkstra, A*.*

1. Pendahuluan

Game merupakan hiburan yang diminati berbagai kalangan, dari sebuah *game puzzle* sederhana sampai pada *game* yang sangat kompleks. *Game* dalam bentuk elektronik telah mengalami perkembangan yang sangat pesat. Perkembangan *game* yang pesat kurang didukung oleh nilai edukasi yang ditawarkan dalam *game* [1]. Beberapa *game* bahkan mengangkat kekerasan sebagai daya tariknya, seperti pada seri *game Grand Theft Auto*. Banyaknya pemain yang sebagian juga terdiri dari anak-anak membuat hal ini dapat memberi pengaruh buruk bagi mental pemain [2].

Pemberian nilai edukasi untuk pemain tidak harus melalui *game* dengan tema edukasi secara penuh. Untuk *game* bertema fantasi, materi pengetahuan dapat diberikan dengan cara menyisipkannya di sela-sela permainan. Penyisipan materi ini akan lebih baik jika dilakukan dengan cara yang wajar dan dapat bersinergi dengan alur cerita dalam *game*. Hal ini diharapkan dapat mengurangi unsur paksaan yang dirasakan jika dibandingkan dengan menawarkan materi pengetahuan secara mentah-mentah.

Permasalahan yang dapat dirumuskan adalah kurangnya nilai edukasi yang ditawarkan dalam *game*, terutama bagi anak-anak, sehingga akan dikembangkan sebuah *game* ber-*genre* strategi yang memiliki nilai edukasi. Manfaat yang diharapkan dari pengembangan ini adalah pemain dapat menambah wawasan melalui materi pengetahuan serta bisa melatih otak melalui pengelolaan unit dan pengaturan strategi untuk mengalahkan musuh dalam setiap level pertarungan *game*.

Pengembangan dilakukan menggunakan metodologi *prototyping* dengan tahapan sebagai berikut:

1. *Planning*: merencanakan dan mengidentifikasi kebutuhan *game* yang akan dikembangkan.
2. *Analysis*: menganalisis kebutuhan yang sudah diidentifikasi.
3. *Design*: merancang dan menunjukkan bagaimana kebutuhan yang ada akan diimplementasikan dalam *game*.
4. *System Prototype*: dari fase analisis dan desain akan dihasilkan sebuah *prototype* untuk diimplementasikan. Jika masih diperlukan perubahan, maka akan kembali dilakukan analisis dan desain untuk menghasilkan *prototype* berikutnya. Jika tidak, maka *game* siap untuk digunakan.
5. *Implementation*: melakukan pengujian terhadap *prototype* yang dihasilkan untuk mengetahui apakah kebutuhan yang ada sudah diimplementasikan dengan benar atau belum.

Pengembangan dilakukan dengan menggunakan teknologi Flash (bahasa Action Script 3.0), *local shared object* untuk penyimpanan data, *use case diagram* untuk menunjukkan kebutuhan (*requirement*), *game layout chart* untuk membantu memahami jalur permainan, serta *storyboard* untuk menunjukkan sketsa antarmuka pengguna dan navigasi permainan.

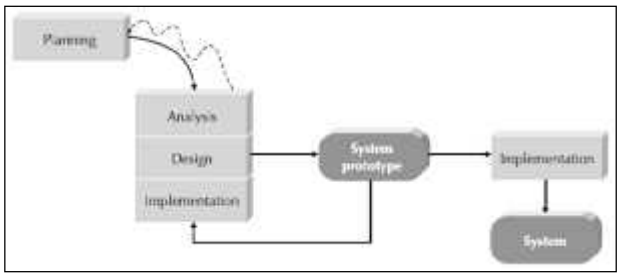
1.1 Pengembangan *Game*

Membangun sebuah *game* adalah proyek pengembangan perangkat lunak. Teknik yang digunakan untuk mengelola proses-proses yang ada seharusnya diambil dari lingkup rekayasa perangkat lunak. Beberapa model siklus hidup pengembangan sistem yang dikenal karena keberhasilannya antara lain *classic waterfall, modified waterfall*, dan *iterative prototyping.*

Iterative prototyping merupakan model pengembangan yang terbaik untuk sebagian besar *game* baru, dimana konsep utamanya adalah mendapatkan prototipe kasar secepat mungkin, diuji, mempertahankan yang sudah cocok dan menghilangkan yang tidak cocok, kemudian kembali lagi dan membuat prototipe yang baru [3].

Dalam bukunya mengenai produksi dan pengembangan *game***,** Bethke Ivan menyarankan menggunakan disiplin ilmu dari *Unified Software Development Process* yang dibuat oleh Ivar Jacobson, Grady Booch, dan James Rumbaugh. *Unified Modeling Language* (UML) adalah penulisan standar dan bahasa visual untuk *Unified Process* [4].

Dalam *prototyping* (Gambar 1), fase analisis, desain, dan implementasi dilakukan secara berdampingan, dan semua fase ini dilakukan dengan berulang-ulang dalam sebuah siklus hingga sistem lengkap dikembangkan. Prototipe pertama biasanya adalah bagian pertama dari sistem yang digunakan. Prototipe ini digunakan untuk menunjukkan pada pengguna maupun sponsor proyek untuk dimintai tanggapan. Tanggapan-tanggapan ini akan digunakan untuk menganalisis, mendesain, serta mengimplementasikan kembali prototipe berikutnya yang memberikan lebih banyak fitur. Proses ini berlanjut dalam sebuah siklus sampai analis, pengguna, dan sponsor setuju bahwa prototipe ini telah memberikan fungsi yang cukup untuk digunakan dalam organisasi [5].



Gambar 1.*Fase-Fase Dalam Metodologi Prototyping*

1.2 Pengembangan Konsep

Pengembangan konsep merupakan pengerjaan yang masih kabur atau kasar dari sebuah tahap desain *game.* Tujuannya adalah untuk menentukan mengenai apa *game* tersebut dan menuliskan dengan jelas supaya setiap orang dapat mengerti dengan cepat [3].

Dokumen yang dihasilkan berupa [3]:
1. *High Concept*: satu atau dua kalimat yang mendeskripsikan mengenai sebuah *game*.
2. *Game Proposal* (*Pitch Doc*): dokumen dalam satu atau dua halaman yang merumuskan sebuah *game*, merupakan versi singkat dari *Concept Document.*
3. *Concept Document*: versi detail dari *Pitch Doc*, digunakan untuk menjelaskan lebih rinci mengenai bagian-bagian yang sudah dituliskan dalam *Game Proposal*. Terdiri dari *high concept, genre, gameplay, features, setting, story, target audience, hardware platform, estimated schedule and budget, competitive analysis, team,* serta *risk analysis.*

1.3 Metode Pencarian A*

Algoritma A* merupakan perbaikan dari metode *best-first search* dengan memodifikasi fungsi heuristiknya. A* meminimumkan total biaya lintasan. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal. Biaya yang dihitung didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai [6]:

$$f(n) = g(n) + h(n) \quad \text{.....(1)}$$

Dimana: f(n)=fungsi evaluasi, g(n)=biaya yang sudah dikeluarkan dari keadaan awal sampai keadaan n, serta h(n)=estimasi biaya untuk sampai pada suatu tujuan mulai dari n.

Pada algoritma A* dibutuhkan dua antrian, yaitu [6]:
1. OPEN, yang berisi simpul-simpul (*node*) yang sudah dibangkitkan, sudah memiliki fungsi heuristik namun belum diuji.
2. CLOSED, berisi simpul-simpul yang sudah diuji.

Berikut ini merupakan algoritma A* [6]:
1. Set: OPEN={S}, dan CLOSED={}, dengan S adalah simpul yang dipilih sebagai keadaan awal.
2. Kerjakan jika OPEN belum kosong:
   a. Cari simpul n dari OPEN dimana nilai f(n) minimal. Kemudian tempatkan n pada CLOSED.
   b. Jika n adalah simpul tujuan, maka keluar. SUKSES.
   c. Ekspansi simpul n ke anak-anaknya.
   d. Kerjakan untuk setiap anak n, yaitu n':
      1) Jika n' belum ada di OPEN atau CLOSED, maka:
         a) Masukkan n' ke OPEN. Kemudian *setbackpointer* dari n' ke n.
         b) Hitung:
            1. h(n')
            2. g(n')=g(n) + c(n, n'); dengan c(n, n') adalah biaya dari n ke n'
            3. f(n')=g(n') + h(n')
      2) Jika n' telah ada di OPEN atau CLOSED dan jika g(n') lebih kecil (untuk versi n' baru), maka:
         a) Buang versi lama n'.
         b) Ambil n' di OPEN, dan *setbackpointer* dari n' ke n.

1.4 Algoritma Dijkstra

Algoritma Dijkstra dikemukakan oleh E. W. Dijkstra (1959) yang dapat digunakan untuk menyelesaikan salah satu permasalahan dalam teori *graph.* Misalkan ada sebanyak n buah simpul, beberapa atau semua pasang dari simpul tersebut dihubungkan dengan sebuah sisi; panjang setiap sisi diketahui. Batasan yang dibuat adalah dimana paling sedikit ada satu jalur di antara dua simpul manapun. Permasalahan yang dimiliki adalah mencari simpul-simpul untuk membuat sebuah *tree* yang memiliki panjang total minimum antara n buah simpul. Sebuah *tree* adalah *graph* dengan satu dan hanya satu jalur antara setiap dua simpul [7].

Dalam metode ini, sisi dibagi menjadi tiga kelompok:

I. Sisi-sisi yang akan digunakan dalam *tree* yang akan dibentuk.
II. Sisi-sisi yang akan dipilih untuk dimasukkan ke dalam sisi kelompok I.
III. Sisi-sisi lainnya (yang ditolak atau yang belum diperiksa).

Simpul-simpul yang ada dibagi menjadi dua kelompok:

A. Simpul-simpul yang dihubungkan oleh sisi-sisi kelompok I.
B. Simpul-simpul lainnya (satu dan hanya satu sisi dari kelompok II yang akan menuntun pada setiap simpul di kelompok B).

Pencarian dimulai dengan memilih sembarang simpul sebagai anggota kelompok A dan menambahkan semua sisi yang berakhir pada simpul ini ke dalam sisi kelompok II. Pada kondisi awal, sisi kelompok I kosong. Kemudian lakukan:

1. Sisi yang terpendek dari kelompok II dikeluarkan lalu dimasukkan ke kelompok I. Dari pemindahan ini, maka satu simpul akan terpindahkan juga dari kelompok B ke kelompok A.
2. Ambil sisi yang memiliki awal dari simpul yang baru saja dipindahkan ke kelompok A dan yang memiliki akhir pada simpul yang masih berada di kelompok II. Jika sisi yang diambil tersebut lebih panjang dari sisi yang memiliki simpul awal dan akhir yang sama, maka sisi yang diambil ditolak; jika lebih pendek, maka sisi yang diambil tersebut menggantikannya, sisi yang terganti itu kemudian ditolak.

Proses perulangan dilakukan hingga sisi kelompok II dan simpul kelompok B sudah kosong. Sisi-sisi yang berada dalam kelompok I membentuk *tree* yang dicari [7].

2. Pembahasan

2.1 Analisis Proses

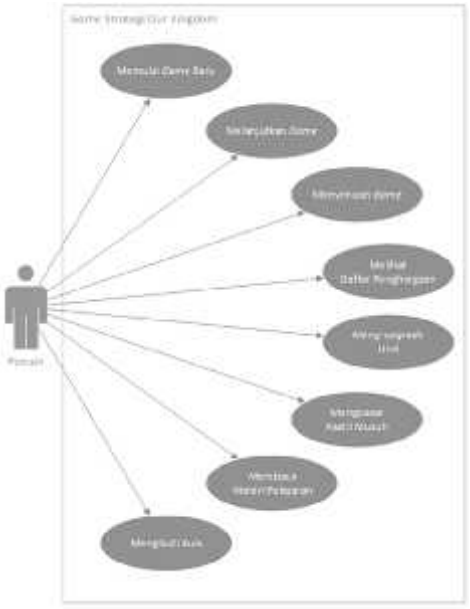
Analisis proses dilakukan dengan mengambil contoh dua buah *game* strategi, yaitu Brigandine: The Legend of Forsena dan Hoshigami: Ruining Blue Earth. Dari kedua *game* ini, dapat dirangkum beberapa aksi yang dilakukan pemain secara umum, yaitu memulai sesi permainan baru, berfokus pada penyelesaian misi-misi dan personalisasi elemen permainan, serta *game* dinyatakan tamat jika misi utama telah dicapai. Penyelesaian misi dan personalisasi elemen permainan merupakan aksi utama dari proses bermain yang dinikmati oleh pemain hingga *game* diakhiri. G*ame* yang dikembangkan diharapkan dapat memberikan nilai edukasi dengan menyisipkan aksi yang dapat menyerap materi pengetahuan ke dalam aktivitas bermain *game*. Aksi tersebut sebisa mungkin diberikan dengan tetap bersinergi dengan alur *game* yang sedang berlangsung.

2.2 Identifikasi Kebutuhan

Dari analisis proses yang dilakukan, diusulkan *game* yang diringkas dalam konsep sebagai berikut:

1. *High Concept*: Putra raja yang ingin menjayakan kembali kerajaannya.
2. *Genre*: Strategi (*Turn-Based Tactic*).
3. *Game play*: Pemain melakukan penyerangan terhadap kastil musuh untuk menguasainya. Setiap melakukan penyerangan, pemain akan memasuki tahap pertempuran (*battle*) terhadap musuh yang dilakukan dengan sistem *turn-based*. Berbagai *upgrade* dapat dilakukan seiring permainan berlangsung. Pemain juga dapat membaca beberapa materi pelajaran yang didapat dengan menguasai kastil-kastil musuh.
4. Fitur:
   a. *Upgrade* senjata dan kelas, dapatkan serangan-serangan baru.
   b. *Achievement* yang dapat di-*unlock*.
   c. Materi pelajaran yang dapat dibaca.
5. *Setting*: Zaman peperangan klasik.
6. Cerita: Kerajaan Grassia yang jaya dan memiliki daerah kekuasaan yang luas tiba-tiba mendapatkan kabar buruk, seorang tangan kanan kepercayaan raja berkhianat. Raja terbunuh akibat pengkhianatan tersebut dan terjadi kekacauan di beberapa daerah karena putra penerus raja itu masih sangat kecil. Kerajaan Marilith yang sudah lama menunggu kesempatan untuk menyerang Kerajaan Grassia pun memanfaatkan kesempatan ini untuk menguasai daerah-daerah yang dimiliki Kerajaan Grassia. Syukurlah putra raja berhasil diselamatkan oleh ratu ke sebuah desa terpencil yaitu Desa Lua.
   Beberapa tahun kemudian, putra raja pun tumbuh besar, dan dia bernama Varith. Di Desa Lua, dia belajar berbagai hal agar bisa melawan Kerajaan Marilith. Dia pun mengenal dekat para penduduk desa dan menjalin hubungan baik dengan beberapa pimpinan wilayah yang juga sedang melawan Kerajaan Marilith.
   Putra raja yang sudah matang ini akhirnya memutuskan untuk melangsungkan pemberontakan terhadap Kerajaan Marilith bersama warga desa tempat dia tinggal hingga ke daerah-daerah lainnya. Pemberontakan ini dilakukan untuk melahirkan dan menjayakan kembali Kerajaan Grassia.
7. Karakter Utama:
   a. Varith: Putra raja Kerajaan Grassia, dibawa oleh ibunya bersembunyi di Desa Lua sejak ia kecil. Ia bertekad melahirkan dan menjayakan kembali Kerajaan Grassia.
   b. Yurin: Penduduk Desa Lua, selalu menjaga Varith sejak Varith dibawa ke Desa Lua oleh ibunya untuk bersembunyi.
   c. Selia: Kakak Yurin, salah satu pejuang dari Desa Lua. Berwatak keras, menghormati putra mahkota, dan sangat menyayangi adiknya.
   d. Frick: Pemimpin sekelompok pemberontak kecil yang membenci pemerintahan Kerajaan Marilith.
8. *Target Audience*: Pemain berusia 14 tahun ke atas.
9. *Hardware Platforms*: PC.

Gambar 2 berikut ini adalah *use case diagram* yang menunjukkan kebutuhan tersebut.



Gambar 2.*Use Case Diagram Kebutuhan Sistem*

2.3 Rancangan Proses

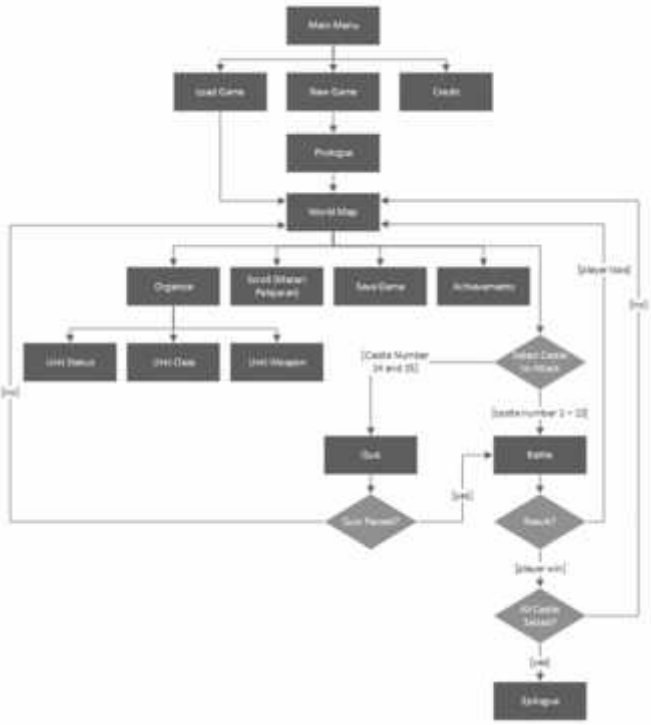
Game yang dirancang diharapkan bisa menyajikan materi pengetahuan kepada pemain di sela permainan, kemudian menjalani evaluasi lewat kuis yang berhubungan dengan materi yang didapat. Proses ini dapat dilihat pada *activity diagram* berikut (Gambar 3).

Gambar 3.*Activity Diagram Game Usulan*

Navigasi jalannya permainan *game* yang diusulkan dapat dilihat pada *layout chart* layar permainan pada Gambar 4.

2.4 Rancangan Elemen Permainan

Game dibangun dari beberapa elemen yang saling mendukung, yaitu komponen maupun fitur dalam *game* yang memegang peranannya masing-masing, namun tetap saling berhubungan dan membangun *game* tersebut. Beberapa elemen tersebut yaitu *Orb*, Kastil, *Battle*, Tingkat Kesulitan *Battle*, Tipe Serangan, Perhitungan *Damage* Serangan, Perhitungan *Experience* (EXP), serta Atribut Karakter, Senjata, Kelas Unit, Serangan Spesial, dan *Achievement*.



Gambar 4.*Layout Chart Game Usulan*

2.5 Rancangan *Artificial Intelligence* (AI)

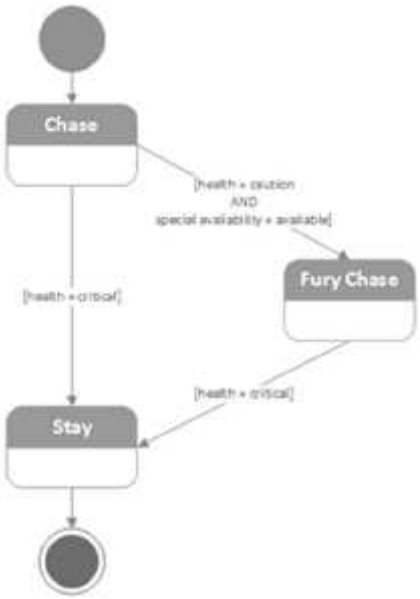
Untuk menentukan aksi yang akan dilakukan unit musuh, terdapat dua jenis kondisi yang dapat digunakan sebagai penentu aksi, yaitu:

1. *Health*: Keadaan kerusakan yang dihitung berdasarkan persentase nilai *Health Point* (HP) terhadap nilai MaxHP. Terdapat tiga nilai kondisi, yaitu:
   a. *Good*: nilai persentase lebih besar dari 75%.
   b. *Caution*: nilai persentase dari 25% sampai 75%.
   c. *Critical*: nilai persentase lebih kecil dari 25%.
2. *Special Availability*: Merupakan kemampuan untuk menggunakan serangan spesial yang dihitung berdasarkan nilai *Special Point* (SP). Terdapat dua nilai kondisi yang memungkinkan, yaitu:
   a. *Available*: dengan syarat nilai SP yang dimiliki lebih besar atau sama dengan nilai pemakaian SP serangan spesial yang akan digunakan.
   b. *Unavailable*: dengan syarat nilai SP yang dimiliki lebih kecil dari nilai pemakaian SP serangan spesial yang akan digunakan.

Berdasarkan kondisi-kondisi yang ada, maka unit musuh dapat memiliki tiga *state* seperti pada gambar *state machine diagram* di Gambar 5.

Agar unit musuh dapat mengejar unit pemain dan bergerak pada jalur yang benar, maka digunakan algoritma sebagai berikut:

1. Jika ada unit pemain tepat satu kotak di atas, bawah, kanan, atau kiri, maka serang salah satu unit pemain tersebut dan akhiri aksi.

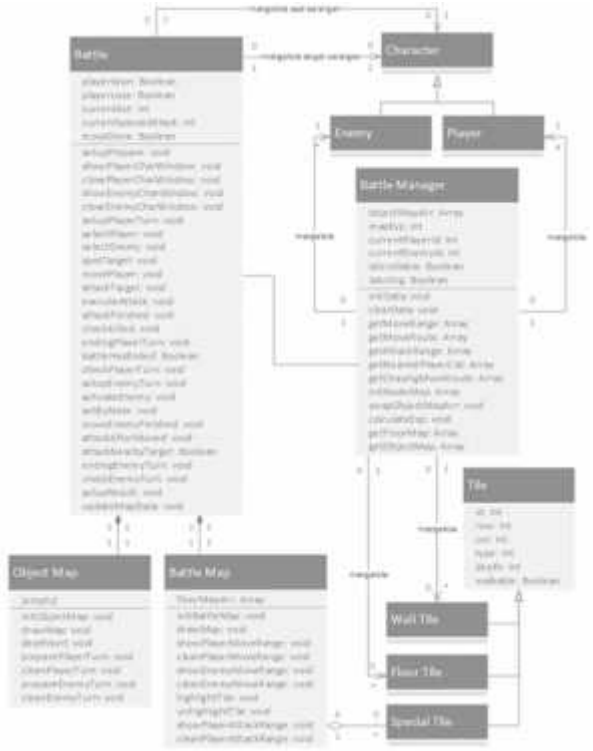


Gambar 5.*State Machine Diagram Aksi Unit Musuh*

2. Jika tidak ada, maka:
   a. Buat daftar unit pemain yang disusun berdasarkan jarak terdekat. Jarak ini dihitung dari jumlah selisih baris dan selisih kolom antara posisi musuh dan posisi pemain.
   b. Untuk setiap unit pemain dalam daftar, lakukan:
      1) Untuk setiap satu kotak di atas, bawah, kanan, atau kiri unit pemain sebagai simpul akhir, cari rute pergerakan penuh dari posisi unit musuh sebagai simpul awal sampai simpul akhir menggunakan algoritma A*.
      2) Jika rute pergerakan tidak ditemukan, maka akhiri aksi.
      3) Jika rute pergerakan ditemukan, maka:
         1. Jika biaya (*cost*) rute pergerakan lebih besar dari jangkauan gerak (nilai MOVE) unit musuh, maka kurangi (*pop*) rute pergerakan tersebut hingga memiliki *cost* yang sama dengan nilai MOVE.
         2. Jika simpul akhir dari rute pergerakan tersebut merupakan simpul yang tidak valid (sudah terisi objek lain), maka kurangi (*pop*) rute pergerakan hingga simpul akhir merupakan simpul yang valid.
   c. Ubah posisi unit musuh untuk bergerak menelusuri setiap simpul yang ada dalam rute pergerakan yang sudah valid.
   d. Untuk posisi unit musuh yang baru, jika ada unit pemain tepat satu kotak di atas, bawah, kanan, atau kiri, maka serang salah satu unit pemain tersebut dan akhiri aksi. Jika tidak ada, maka akhiri aksi.

2.6 Rancangan Kelas

Untuk membantu menggambarkan struktur statis dari *game* yang dirancang, kelas-kelas yang penting ditemukan kemudian digambarkan dalam *class diagram* seperti Gambar 6 berikut ini.



Gambar 6.*Class Diagram Battle*

2.7 Hasil

Hasil akhir dari *game* yang dikembangkan memiliki tampilan menu utama seperti Gambar 7 berikut ini.

Gambar 7.*Tampilan Menu Utama*

World Map (Gambar 8) akan muncul setelah prolog selesai atau setelah pemain memilih salah satu *slot* pada bagian *Load Game.*

Gambar 8.*Tampilan World Map*

Gambar 9 berikut ini adalah tampilan yang muncul saat pemain mengklik salah satu materi yang ingin dibaca.

Gambar 9.*Tampilan Membaca Materi*

Gambar 10 berikut ini adalah tampilan dari lingkungan saat melakukan *battle*. Pemain dapat menggerakkan *pointer* ke salah satu unit untuk melihat informasi singkat unit tersebut. Jika pemain mengklik salah satu unit, maka akan muncul menu yang dapat dipilih pemain.

Gambar 10.*Tampilan Lingkungan Battle*

Setelah *battle* berakhir, maka akan ditampilkan hasil dari *battle* tersebut (Gambar 11), yaitu berupa informasi EXP dan *scroll* baru yang didapat.

Gambar 11.*Tampilan Hasil Battle*

3. Kesimpulan

Dari pengembangan yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Pemain masih dapat berfokus pada permainan inti dan tidak terganggu dengan materi pelajaran yang diberikan jika cara penyajian materi tidak secara langsung ditonjolkan, namun dilakukan dengan menghubungkannya dengan alur cerita.
2. Pemberian materi pengetahuan di sela permainan dapat memberi tambahan pengetahuan kepada pemain.

Saran yang dapat diberikan adalah pemanfaatan AI pada aksi unit musuh perlu ditingkatkan, misalnya dengan memperbanyak kondisi penentu aksi untuk menciptakan unit musuh yang lebih "pintar". Selain itu, pembuatan AI pada unit bos musuh dibedakan dengan AI pada unit musuh biasa, misalnya unit bos musuh maju mendekati unit pemain setelah unit musuh biasa sudah dikalahkan sebagian.

Daftar Pustaka

[1] S. Song, "Why is There A Lack of Innovation in Educational Technology for Children?", [Online]. Tersedia: http://www.forbes.com/sites/quora/2012/03/23/why-is-there-a-lack-of-innovation-in-educational-technology-for-children [Diakses 08 Agustus 2013].
[2] S. S. Brady and K. A. Matthews, "Effects of Media Violence on Health-Related Outcomes Among Young Men", *Arch Pediatr Adolesc Med*, vol. 160, no. 4, pp. 341-347, 2006.
[3] B. Bates, *Game Design*, Second Edition, Boston: Thomson Course Technology PTR, 2004.
[4] E. Bethke, *Game Development and Production*, Texas: Wordware Publishing, 2003.
[5] A. Dennis, B. H. Wixom, and D. Tegarden, *System Analysis Design UML Version 2.0*, Third Edition, Hoboken: John Wiley & Sons, Inc., 2007.
[6] S. Kusumadewi, *Artificial Intelligence: Teknik dan Aplikasinya*, Yogyakarta: Graha Ilmu, 2003.
[7] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, vol. 1, no. 1, pp. 269-271, 1959.

Biodata Penulis

Gunawan, memperoleh gelar Sarjana Komputer (S.Kom.), Program Studi Teknik Informatika STMIK MIKROSKIL Medan, lulus tahun 2003. Memperoleh gelar Magister Teknologi Informasi (M.T.I.) Program Pasca Sarjana Teknologi Informasi Universitas Indonesia Jakarta, lulus tahun 2008. Saat ini menjadi dosen tetap di STMIK MIKROSKIL Medan.

Fandi Halim, memperoleh gelar Sarjana Komputer (S.Kom.), Program Studi Sistem Informasi STMIK MIKROSKIL Medan, lulus tahun 2007. Memperoleh gelar Magister Sains (M.Sc.) Program Pasca Sarjana Information Technology Technopreneurship Universiti Sains Malaysia Penang, lulus tahun 2010. Saat ini menjadi dosen tetap di STMIK MIKROSKIL Medan.

Jetson Surya, memperoleh gelar Sarjana Komputer (S.Kom.), Program Studi Sistem Informasi STMIK MIKROSKIL Medan, lulus tahun 2013.