BAB 2
LANDASAN TEORI

2.1 *Game*

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan 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*. Suatu cara belajar yang digunakan dalam menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukkan strategi strategi yang rasional. Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh John von Neumann and Oskar Morgenstern yang berisi :

Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi." (*Theory of Games and Economic Behavior* (3d ed. 1953)) [8].

2.1.1 Pengertian *Game*

Pengertian *game* menurut beberapa ahli[7] :

1. Menurut Agustinus Nilwan dalam bukunya Pemrograman Animasi dan *Game* Profesional terbitan Elex Media Komputindo, *game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan *game*. Atau jika ingin membuat *game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

2. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).
3. Menurut Chris Crawford, seorang computer *game* designer mengemukakan bahwa *game*, pada intinya adalah sebuah interaktif, aktivitas yang berpusat pada sebuah pencapaian, ada pelaku aktif *(player)*, ada pelaku pasif *(NPC*).
4. Menurut David Parlett, *Game* adalah sesuatu yang memiliki "akhir dan cara mencapainya" : artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.
5. Menurut Roger Caillois, seorang sosiolog Perancis, dalam bukunya yang berjudul Les jeux et les hommes menyatakan *game* adalah aktivitas yang mencakup karakteristik berikut: *fun* (bebas bermain adalah pilihan bukan kewajiban), *separate* (terpisah), *uncertain, non-productive, governed by rules* (ada aturan), *fictitious* (pura-pura).
6. Menurut Greg Costikyan, *Game* adalah "sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam *game* demi mencapai tujuan".

2.1.2 Jenis-Jenis *Game*

Berikut klasifikasi *game* berdasarkan jenis platform yang digunakan [7]:

1. *Arcade games*,

   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 *game*s 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).

2. *PC Games*.

   PC Games yaitu video *game* yang dimainkan menggunakan Personal Computers.

3. *Console games*.

   Console games yaitu video *game*s yang dimainkan menggunakan console tertentu, seperti Playstation 2, Playstation 3, XBOX 360, dan Nintendo Wii.

4. *Handheld games.*

   Handheld games yaitu yang dimainkan di console khusus video *game* yang dapat dibawa kemana-mana, contoh Nintendo DS dan Sony PSP.

5. *Mobile games.*

   Mobile games yaitu yang dapat dimainkan atau khusus untuk mobilephone atau PDA.

2.1.3 Genre *Game*

Berikut klasifikasi *game* berdasarkan genre permainannya [7] :

1. Aksi *Shooting*

   Video game jenis ini sangat memerlukan kecepatan refleks, koordinasi mata dan tangan, juga timing, inti dari *game* jenis ini adalah tembak, tembak dan tembak. Termasuk didalamnya tedapat :

   a. *First person shooting (FPS)* seperti *Counter Strike* dan *Call of Duty.*
      Drive n shoot, menggunakan unsur simulasi kendaraan tetapi tetap dengan tujuan utama menembak dan menghancurkan lawan, contoh : *Spy Hunter, Rock and Roll Racing, Road Rash.*
   b. *Shoot em up,* seperti Raiden, 1942, dan gradius.
   c. Beat em up (tonjok hajar) seperti *Double Dragon* dan *Final Fight*, lalu *hack and slash* (tusuk tebas) seperti *Shinobi* dan *Legend of Kage.*
   d. *Light gun shooting*, yang menggunakan alat yang umumnya berbentuk seperti senjata, seperti *Virtua Cop* dan *Time Crisis.*
   e. *Fighting* (pertarungan) Ada yang mengelompokan *video game fighting* di bagian Aksi, jenis ini memang memerlukan kecepatan refleks dan

koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus (hafal caranya dan lancar mengeksekusinya).

2. Petualangan.

   Jenis *video game* aksi-petualangan memerlukan kelihaian pemain dalam bergerak, *refleks*, 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.

3. Simulasi, Konstruksi dan manajemen.

   Video Game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. *Video Game* jenis ini membuat pemain harus berpikir untuk mendirikan, membangun dan mengatasi masalah dengan menggunakan dana yang terbatas. Contoh: *Sim City*, *The Sims, Tamagotchi.*

4. *Role Playing.*

   Video game jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh/peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya, dimana siring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dll) dalam berbagai parameter yang biasanya ditentukan dengan naiknya *level*, baik dari status kepintaran, kecepatan dan kekuatan karakter. seperti *Final Fantasy, Dragon Quest* dan *Xenogears. Never Winter Nights, baldurs gate, Elder Scroll,* dan *Fallout.*

5. *Puzzle.*

   Video game jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai 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. *Tetris, Minesweeper, Bejeweled, Sokoban* dan *Bomberman.*

6. Simulasi kendaraan.

   Video *Game* jenis ini memberikan pengalaman atau interaktifitas sedekat mungkin dengan kendaraan yang aslinya, muskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada detil dan pengalaman realistik menggunakan kendaraan tersebut.

7. Olahraga.

   Video *Game* jenis ini bersifat singkat padat jelas, bermain sport di PC atau konsol anda. Biasanya permainannya diusahakan serealistik mungkin walau kadang ada yang menambah unsur fiksi seperti *NBA JAM*. Contohnya pun jelas, Seri *Winning Eleven*, seri *NBA*, seri *FIFA, John Madden NFL, Lakers vs Celtics, Tony hawk pro skater*, dll.

2.2 *Game* Edukasi

Game edukasi adalah *game* digital yang dirancang untuk pengayaan pendidikan (mendukung pengajaran dan pembelajaran), menggunakan teknologi multimedia interatif [8].

2.2.1 Kriteria *Game* Edukasi

Berikut kriteria *game* edukasi yang ideal [8] :

1. Rasa ingin tahu, fantasi dan kontrol pengguna
   a. Motivasi intrinsik, menyenangkan untuk digunakan. Pengguna memiliki kontrol terhadap permainan.
   b. Integrasi antara materi edukasi dan aspek fantasi permainan
   c. Pengguna memegang kendali permainan dan dapat mengatur kecepatan bermain.
   d. Mendorong keingintahuan, pengguna dapat melakukan eksplorasi bebas. Permainan mengandung rahasia tersembunyi.

e. Pengguna dapat menyimpan kemajuan permainan.
f. Menyediakan lebih dari jalur. Ada beberapa cara untuk memenangkan permainan
g. Pengguna dapat melakukan trial-error untuk menyelesaikan permainan.
h. Keberhasilan ditentukan oleh pengetahuan, bukan kebetulan.
i. Task disajikan secara incremental dan bertahap.
j. Simulasi realistik dunia.
k. Materi edukasi disesuaikan dengan materi dunia nyata.
l. Personalisasi: pengguna dapat mengubah karakter dan objek-objek.
m. Menyediakan banyak pilihan dan tema.
n. Hindari pengulangan, hindari *drill-practice.*
o. Sediakan elemen tidak terduga dan kejutan.
p. Point tidak hilang saat menjawab pertanyaan.

2. Tantangan
   a. Kinerja pemain terukur dan didefinisikan dengan baik.
   b. Tantangan diberikan secara terus menerus tapi disesuaikan dengan tingkat pemahaman pemain.
   c. Kompleksitas semakin berkembang sejalan dengan kemampuan pengguna. Tersedia banyak *level*.
   d. Memonitor kinerja pengguna, berdasarkan hasil monitoring ini kesulitan kemudian disesuaikan.
   e. Pengguna dapat melihat progressnya setiap saat.
   f. Menyediakan hint dan instruksi untuk membantu pengguna.
   g. Bantuk pengguna menemukan *'flow state* yaitu keseimbangan antara tantangan dengan kemampuan untuk menghadapi tantangan.
3. Sosialisasi
   a. Pemain dapat berkolaborasi bersama.
   b. Pemain dapat saling berkompetisi.
   c. Pemenang bisa lebih dari satu (*multiple winner*)
4. Pedagogi
   a. Objektif pembelajaran jelas.

b. Aktivitas pembelajaran dilaksanaan bersamaan dengan permainan.
c. Menyebutkan secara eksplisit target umur dan menyesuaikan rancangan permainan sesuai umur tersebut.
d. Memberi kesempatan kepada guru dan orangtua untuk ikut berperan.
e. Memberikan petunjuk yang jelas sehingga pemain dapat berkonsentrasi kepada isi permainan, bukan bagaimana cara menggunakan permainan.
f. Menyediakan tutorial sehingga pemain tidak perlu membaca manual.

5. Teknologi
   a. Menggunakan teknologi yang tersedia di sekolah dan masyarakat umum (tidak membutuhkan hardware yang terlalu tinggi)
   b. Memberikan lisensi yang terjangkau sekolah.
   c. Menggunakan antarmuka yang intuitif.
6. Pengguna anak-anak dan yang berkebutuhan khusus
   a. Memberikan petunjuk yang dibacakan selain petunjuk tertulis.
   b. Gambar, objek, layar tertata rapi.
   c. Permainannya sendiri merupakan aktivitas yang penting (*play for the sake of play*)
   d. Transformasi terlihat. Saat anak melakukan interaksi, ada instant *feedback.*
   e. Input dan output yang mudah terlihat.
   f. Tantangan diberikan secara gradual.
   g. Permainan menyenangkan untuk diulang, beberapa karakter, musik mudah teringat oleh anak.
   h. Objek dan suara diambil dari kehidupan sehari-hari yang mudah dikenali.

2.3 Algoritma A*

Algoritma ini merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Search dan Greedy Best-First Search*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai $f(n) = g(n) + h(n)$. Dengan perhitungan biaya seperti ini, Algoritma A* adalah compelete dan optimal. [6]

Algorima A*, merupakan salah satu contoh algoritma pencarian yang cukup popular di dunia. Jika mengetikkan Algoritma A* pada sebuah mesin pencari, seperti google.com, maka akan ditemukan lebih dari sepuluh ribu literatur mengenai algoritma A*.

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah *starting point*, simpul (*nodes*), A, *open list*, *closed list*, harga (*cost*), halangan (*unwalkable*). *Starting point* adalah sebuah terminologi untuk posisi awal sebuah benda. A adalah simpul yang sedang dijalankan dalam algortima pencarian jalan terpendek. Simpul adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. *open list* adalah tempat menyimpan data simpul yang mungkin diakses dari *starting point* maupun simpul yang sedang dijalankan. *Closed list* adalah tempat menyimpan data simpul sebelum A yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan. Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke A, dan H, jumlah nilai perkiraan dari sebuah simpul ke simpul tujuan. Simpul tujuan yaitu simpul yang dituju. Rintangan adalah sebuah atribut yang menyatakan bahwa sebuah simpul tidak dapat dilalui oleh A.

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (*starting point)* menuju simpul tujuan dengan memperhatikan harga (F) terkecil. Diawali dengan menempatkan A pada *starting point*, kemudian memasukkan seluruh simpul yang bertetangga dan tidak memilik atribut rintangan dengan A ke dalam *open list*. Kemudian mencari nilai H terkecil dari simpul-simpul dalam *open list* tersebut. Kemudian memindahkan A ke simpul yang memiliki nilai H terkecil. Simpul sebelum A disimpan sebagai *parent* dari A dan dimasukkan ke dalam *closed list*. Jika terdapat simpul lain yang bertetangga dengan A (yang sudah berpindah) namun belum termasuk kedalam anggota *open list*, maka masukkan simpul-simpul tersebut ke dalam *open list.* Setelah itu, bandingkan nilai G yang ada dengan nilai G sebelumnya (pada langkah awal, tidak perlu dilakukan perbandingan nilai G). Jika nilai G sebelumnya lebih kecil maka A kembali ke posisi awal. Simpul yang pernah dicoba dimasukkan ke dalam *closed list.* Hal

terebut dilakukan berulangulang hingga terdapat solusi atau tidaka ada lagi simpul lain yang berada pada *open list* [6].

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu dalam kasus *game pathfinding* dengan penerapan algoritma A* *(A Star).* Adapun istilah-istilah yang akan dibahas yaitu *path, open list, closed list,* nilai f, g dan n.

Algoritma A* menggunakan dua senarai yaitu *OPEN* dan *CLOSED. OPEN* adalah senarai (*list*) yang digunakan untuk menyimpan simpul-simpul yang pernah dibangkitkan dan nilai heuristiknya telah dihitung tetapi belum terpilih sebagai simpul terbaik (*best node*) dengan kata lain, *OPEN* berisi simpul-simpul masih memiliki peluang untuk terpilih sebagai simpul terbaik, sedangkan *CLOSED* adalah senarai untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

1. *OPEN LIST* adalah *list* yang menyimpan kemungkinan *path* yang akan diperiksa. *OPEN LIST* dibuat terurut berdasarkan nilai f. *OPEN LIST* digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada *path* tujuan. *OPEN* berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik (*best node*).
2. *CLOSED* adalah senarai (*list*) untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik (*best node*) atau senarai yang menyimpan jalan yang sudah diperiksa dari *open list*. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup). Kedua list (*OPEN LIST* dan *CLOSED LIST*) ini bertujuan juga untuk menghindari penelusuran berkali-kali jalan (*rute*) yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam *OPEN LIST*.
3. Nilai F adalah *cost* perkiraan suatu *path* yang teridentifikasi. Nilai F merupakan hasil dari f(n).
4. Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan

untuk menuju ke *path* sekarang.

5. Setiap simpul *(node)* harus memiliki informasi nilai h(n), yaitu estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H.

Fungsi f sebagai estimasi fungsi evaluasi terhadap *node* n, dapat dituliskan :

$$f(n) = g(n) + h(n) \qquad (2.1)$$

dengan :

$f(n)$ = fungsi evaluasi (jumlah $g(n)$ dengan $h(n)$)

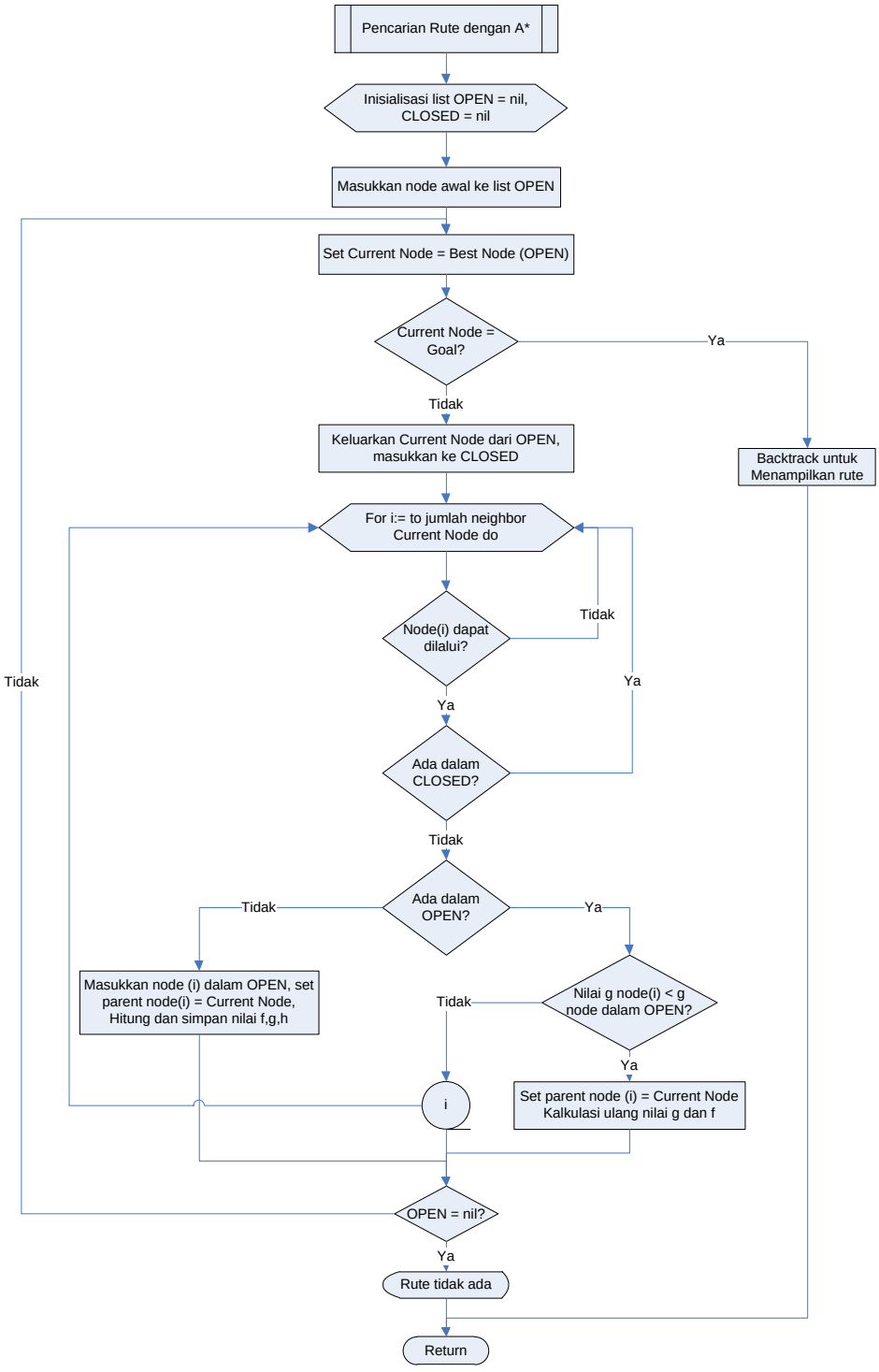
$g(n)$ = biaya *(cost)* yang dikeluarkan dari keadaan awal sampai keadaan n

$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Perhitungan nilai H digunakan fungsi *heuristic*, metode yang digunakan di dalam contoh ini adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya serta mengabaikan penghalang atau *collision,* yang kemudian nilainya dikalikan dengan 10. Atau dirumuskan dengan:

$$H = 10*(\text{abs}(current\text{X}-target\text{X}) + \text{abs}(current\text{Y}-target\text{Y})) \qquad (2.2)$$

Secara umum dapat digambarkan dengan *flowchart* sebagai berikut:



Gambar 2.1 *Flowchart* Algoritma A* [6]

2.4 Palagan Ambarawa

Palagan Ambarawa adalah sebuah peristiwa perlawanan rakyat terhadap Sekutu yang terjadi di Ambarawa, sebelah selatan Semarang, Jawa Tengah. Pada tanggal 20 Oktober 1945, tentara Sekutu di bawah pimpinan Brigadir Bethell mendarat di Semarang dengan maksud mengurus tawanan perang dan tentara Jepang yang berada di Jawa Tengah. Kedatangan sekutu ini diboncengi oleh NICA. Kedatangan Sekutu ini mulanya disambut baik, bahkan Gubernur Jawa Tengah Mr Wongsonegoro menyepakati akan menyediakan bahan makanan dan keperluan lain bagi kelancaran tugas Sekutu, sedang Sekutu berjanji tidak akan mengganggu kedaulatan Republik Indonesia. [2]

Pasukan Sekutu dan NICA yang telah sampai di Ambarawa dan Magelang untuk membebaskan para tawanan tentara Belanda, para tawanan tersebut malah dipersenjatai sehingga menimbulkan kemarahan pihak Indonesia. Insiden bersenjata timbul di kota Magelang, hingga terjadi pertempuran. Di Magelang, tentara Sekutu bertindak sebagai penguasa yang mencoba melucuti Tentara Keamanan Rakyat dan membuat kekacauan. TKR Resimen Magelang pimpinan Letkol. M. Sarbini membalas tindakan tersebut dengan mengepung tentara Sekutu dari segala penjuru. Namun mereka selamat dari kehancuran berkat campur tangan Presiden Soekarno yang berhasil menenangkan suasana. Kemudian pasukan Sekutu secara diam-diam meninggalkan Kota Magelang menuju ke benteng Ambarawa. Akibat peristiwa tersebut, Resimen Kedu Tengah di bawah pimpinan Letkol. M. Sarbini segera mengadakan pengejaran terhadap mereka. Gerakan mundur tentara Sekutu tertahan di Desa Jambu karena dihadang oleh pasukan Angkatan Muda di bawah pimpinan Oni Sastrodihardjo yang diperkuat oleh pasukan gabungan dari Ambarawa, Suruh dan Surakarta.

Tentara Sekutu kembali dihadang oleh Batalyon I Soerjosoempeno di Ngipik. Pada saat pengunduran, tentara Sekutu mencoba menduduki dua desa di

sekitar Ambarawa. Pasukan Indonesia di bawah pimpinan Letkol. Isdiman berusaha membebaskan kedua desa tersebut, namun ia terlebih dahulu gugur terlebih dahulu. Sejak gugurnya Letkol. Isdiman, Komandan Divisi V Banyumas, Kol. Soedirman merasa kehilangan seorang perwira terbaiknya dan ia langsung turun ke lapangan untuk memimpin pertempuran. Kehadiran Kol. Soedirman memberikan napas baru kepada pasukan-pasukan RI. Koordinasi diadakan di antara komando-komando sektor dan pengepungan terhadap musuh semakin ketat. Siasat yang diterapkan adalah serangan pendadakan serentak di semua sektor. Bala bantuan terus mengalir dari Yogyakarta, Solo, Salatiga, Purwokerto, Magelang, Semarang, dan lain-lain.

Tanggal 23 November 1945 ketika matahari mulai terbit, mulailah tembak-menembak dengan pasukan Sekutu yang bertahan di kompleks gereja di Jl. Margo Agoeng. Pasukan Indonesia terdiri dari Yon. Imam Adrongi, Yon. Soeharto dan Yon. Soegeng. Tentara Sekutu mengerahkan tawanan-tawanan Jepang dengan diperkuat tanknya, menyusup ke tempat kedudukan Indonesia dari arah belakang, karena itu pasukan Indonesia pindah ke Bedono.

Sekutu memang tidak pernah berhenti berulah. Kedatangan awal di Semarang untuk semata mengurus tahanan perang Jepang justru berbuntut menjadi kekacauan. Rakyat marah ketika melihat para tahanan yang sebagian besar merupakan mantan tentara Belanda tersebut justru dipersenjatai. Serangan dilancarkan oleh Tentara Keamanan Rakyat yang berhasil memukul mundur pasukan Sekutu hingga mereka terpaksa bertahan di kompleks gereja. Tanggal 12 Desember 1945, kesatuan-kesatuan TKR datang untuk menyerang dan memulai perang selama 1,5 jam. Indonesia berhasil merebut Ambarawa dan memukul mundur Sekutu. [2]

2.5 Unity

Unity Game Engine adalah *software* atau *game engine* yang digunakan untuk membuat *video game* berbasis dua atau tiga dimensi dan dapat digunakan

secara gratis. Selain untuk membuat *game*, *unity* juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan *real-time* 3D animasi. [13]

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. *Unity* bisa digunakan untuk *game*s PC dan *games online*. Untuk *games online* diperlukan sebuah plugin, yaitu *Unity Web Player*, yang sama halnya dengan *flash player* pada *browser*. Bahasa pemrograman yang digunakan bermacam-macam, mulai dari *javascript*, C#, dan *boo*.

Unity tidak bisa melakukan desain atau *modelling*, dikarenakan *unity* bukan merupakan *tools* untuk mendesain. Banyak hal yang bisa di lakukan di unity, ada fitur *audio reverb zone , particle effect , sky box* untuk menambahkan langit, dan masih banyak lagi, dan juga bisa langsung *edit texture* dari *editor* seperti *photoshop* dll.[13]

Features (Scripting) di dalam *unity* adalah sebagai berikut:

1. Mendukung 3 bahasa pemrograman, *JavaScript*, C#, dan *Boo*.
2. *Flexible and EasyMoving*, *rotating*, dan *scaling objects* hanya perlu sebaris kode. Begitu juga dengan *duplicating*, *removing*, dan *changing properties.*
3. *Multi Platform Game* bisa di deploy di PC, Mac, Wii, iPhone, iPad dan *browser*, *android*.
4. *Visual Properties Variables* yang di definisikan dengan *scripts* ditampilkan pada editor. Bisa digeser, di *drag* and *drop*, bisa memilih warna dengan *color picker.*
5. Berbasis .NET, penjalanan program dilakukan dengan *Open Source* .NET *platform, Mono.*

2.6 3D Studio Max

3D Studio Max adalah salah *satu software* atau perangkat lunak yang sering digunakan oleh perancang produk untuk membuat animasi atau pemodelan dalam bentuk 3 dimensi. Aplikasi canggih ini dirilis oleh salah satu perusahaan

Autodesk Media & Entertainment yang pada mulanya dikenal sebagai *Discreet and Kinetix*. *3D Max* merupakan salah satu dari sekian banyak aplikasi *modeling* untuk membuat model 3D dan paling banyak digunakan oleh perancang yang tersebar di seluruh dunia [14].

Sejalan dengan berkembangnya teknologi termasuk juga dalam bidang komputerisasi. 3D *Max* pun mengalami perubahan-perubahan untuk menyesuaikan dengan kemampuan komputer yang semakin tinggi dalam hal grafis. *3D Max* dikembangkan dari aplikasi sebelumnya yang bernama *3D Studio for Dos*, tetapi aplikasi ini hanya diperuntukan untuk *platform Win32*.

3D Max menjadi program animasi komputer 3D dengan penjualan terbesar di dunia. *Software* ini memiliki kemampuan *modeling* yang kuat dan merupakan *plugin architecture* yang fleksibel dan bekerja dengan *platform Microsoft Windows*. *3D Studio Max* banyak digunakan oleh para pembuat dan perancang *video game*, *visual architecture*, *design product* dan juga studio TV untuk pembuatan animasi.[14]

2.7 Pemrograman C#

C# (dibaca : C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. 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, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# Language Specification, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam keyboard standar. Gambar 2.7 merupakan gambar logo C#.[10]



Gambar 2. 2 Logo C# [10]

2.8 Analisis Sistem Berorientasi Objek

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunkan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dab perilaku dalam suatu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, meyiapkan dokumentasi serta perancangan program dan basis data. Pertama-tama suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya.

2.8.1 OOP (Object Oriented Programing)

Pemrograman berorientasi objek (*object-oriented programming* disingkat OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya, Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar.[11]

Konsep dasar dari Pemrograman Berorientasi Objek menekankan konsep berikut:

1. *Class* - kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebuah class adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi object. Sebuah class secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP.
2. Objek - membungkus data dan fungsi bersama menjadi suatu unit dalam sebuah program komputer; objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.
3. Abstrak - Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari "pelaku" abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.
4. Enkapsulasi - Memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam dari sebuah objek dengan cara yang tidak layak hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.
5. Polimorfisme - melalui pengiriman pesan. Tidak bergantung kepada pemanggilan subrutin, bahasa orientasi objek dapat mengirim pesan; metode tertentu yang berhubungan dengan sebuah pengiriman pesan tergantung kepada objek tertentu di mana pesa tersebut dikirim.

6. Inheritas (*Inheritance*) - Mengatur polimorfisme dan enkapsulasi denganmengijinkan objek didefinisikan dan diciptakan dengan jenis khusus dariobjek yang sudah ada objek-objek ini dapat membagi dan memperluasperilaku mereka tanpa haru mengimplementasi ulang perilaku tersebut (bahasa berbasis-objek tidak selalu memiliki inheritas).[11]

2.8.2 UML (Unified Modeling Language)

UML (*Unified Modeling Language*) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem (Flowler, 2006). *Unified Modeling Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.

UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML mulai diperkenalkan oleh *Object Management Group*, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an. Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP.



Gambar 2. 3 Logo UML [5]

UML suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap system informasi. Penggunaan UML dalam industri terus meningkat, ini merupakan

standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

Bagian-bagian utama dari UML adalah *view*, *diagram*, *model element*, dan *general mechanism* [5]. Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya

2.8.2.1 Use Case Diagram

Use case adalah abstraksi dari interaksi antara system dan actor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*. [5]

2.8.2.2 Class Diagram

Class adalah dekripsi kelompok objek-objek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari kelas-kelas yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem. [5]

2.8.2.3 Component Diagram

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. *Component* merupakan implementasi software dari sebuah atau lebih *class*. *Component* dapat berupa *source code*, komponent biner, atau *executable component*. Sebuah komponent berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga

membuat pemetaan dari logical *view* ke *component view*. Sehingga *component diagram* merepresentasikan dunia riil yaitu *component software* yang mengandung *component, interface* dan *relationship*. [5]

2.8.2.4 Deployment Diagram

Deployment Diagram menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh node tertentu dan ketergantungan komponen. [5]

2.8.2.5 State Diagram

State Diagram menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu object dari suatu *class* dan keadaan yang menyebabkan state berubah. Kejadian dapat berupa object lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi class berubah oleh *state* yang berbeda. [5]

2.8.2.6 Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem. [5]

2.8.2.7 Collaboration Diagram

Menggambarkan kolaborasi dinamis seperti *sequence diagram*. Dalam menunjukkan pertukaran pesan, *collaboration diagram* menggambarkan object dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence diagram*, tapi jika penekanannya pada konteks gunakan *collaboration diagram*. [5]

2.8.2.8 Activity Diagram

Activity Diagram menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi. [5]