RANCANG BANGUN GAME 3D "ENA BURENA" DENGAN ALGORITMA A* DAN COLLISION DETECTION MENGGUNAKAN UNITY 3D BERBASIS DESKTOP DAN ANDROID

Patah Herwanto (pherwanto@yahoo.com),
Trisna Sonjaya (trisnasonjaya@gmail.com)

ABSTRAK

Game Ena Burena adalah game ber-*genre shooter* yaitu FPS *(First Person Shoter*) dan TPS *(Third Person Shooter) yang berjalan* pada *platform Desktop* dan *Android.* Pergerakan karakter di suatu *game* sangat erat kaitannya dengan kecerdasan buatan *(Artificial Intelligence)* yang membuat karakter seolah-olah hidup. Hal ini sangat berpengaruh dalam pembuatan *game* untuk membuat karakter NPC *(Non Playable Character)* dalam melakukan pencarian rute terdekat untuk sampai pada titik akhir. Apabila pada saat melakukan sebuah pencarian dan menemukan sebuah penghalang, maka diperlukan sebuah metode untuk mendeteksi adanya tubrukan antar objek agar tidak menembus objek yang ada.Dalam pencarian rute pada *game* ini digunakan algoritma A* dan untuk mendeteksi tumbukan yang mungkin akan terjadi pada saat pencarian rute tersebut maka digunakan teknik *Collision Detection* supaya NPC maupun *player* melakukan pergerakan sesuai yang diinginkan dan *game* 3D menjadi lebih realistis dan sempurna dan ketika karakter berinteraksi satu sama lain tidak akan saling bertabrakan

Kata Kunci : *game, TPS,* 3D, A*, *collision detection, Desktop, Android.*

1. PENDAHULUAN

Game telah menjadi satu hal yang ada di dalam keseharian kita. Dahulu, *game* hanya dijadikan sarana hiburan semata namun sekarang game telah menjadi luas fungsinya, misalnya game dapat dijadikan sarana pembelajaran, lahan bisnis, dan dipertandingkan sebagai salah satu dari cabang olahraga oleh para profesional. Perkembangan *game platform* juga dapat dilihat secara langsung oleh masyarakat, pada mulanya *game* hanya dimainkan di komputer *PC* (*Personal Computer)* dan *console (Nintendo, SEGA, Playstation, XBOX* dll*),* tetapi sekarang sudah memasuki era *mobile game. Mobile game* adalah sebuah *game* yang didesain dan dimainkan oleh *mobile devices*, seperti *PDA*, *smartphone, tablet PCs*, dan *portable media player*. Dan sekarang ini, *mobile game* telah dibuat di berbagai macam *platform* seperti Symbian, Apple IOS, *Android* serta Windows Phone. Keuntungan tersendiri memainkan *mobile game* adalah portabilitas, yaitu *player* dapat bermain *game* dimana saja mereka mau selama mereka mempunyai *mobile devices* yang mampu menjalankan *mobile games*.

Pada game ini digunakan Algoritma A* untuk mencari rute terpendek dalam pencarian suatu jalaur dan *collision detection* untuk mendeteksi tumbukan antar objek

1.1 Algoritma A * (A *Star*)

Algoritma ini merupakan algoritma *Best First Search* (BFS) 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* (A Star) adalah *complete* dan optimal. Pada pencarian *rute* kasus sederhana, dimana tidak terdapat halangan pada peta, A* bekerja secepat dan seefisien BFS. Pada kasus peta dengan halangan, A* dapat menemukan solusi rute tanpa terjebak oleh halangan yang ada.

Pencarian menggunakan algoritma A* mempunyai prinsip yang sama dengan algoritma BFS, hanya saja dengan dua faktor tambahan.

1. Setiap sisi mempunyai "*cost*" yang berbeda-beda, seberapa besar *cost* untuk pergi dari satu simpul ke simpul yang lain.
2. *Cost* dari setiap simpul ke simpul tujuan bisa diperkirakan. Ini membantu pencarian, sehingga lebih kecil kemungkinan kita mencari ke arah yang salah. *Cost* untuk setiap simpul tidak harus berupa jarak. *Cost* bisa saja berupa waktu bila kita ingin mencari jalan dengan waktu tercepat untuk dilalui. Sebagai contoh, bila kita berkendaraan melewati jalan biasa bisa saja merupakan jarak terdekat, tetapi melewati jalan tol biasanya memakan waktu lebih sedikit.

Algoritma A* bekerja dengan prinsip yang hampir sama dengan BFS, kecuali dengan dua perbedaan, yaitu :

1. Simpul-simpul di *list* "terbuka" diurutkan oleh *cost* keseluruhan dari simpul awal ke simpul tujuan, dari *cost* terkecil sampai *cost* terbesar. Dengan kata lain, menggunakan *priority queue* (antrian prioritas). *Cost* keseluruhan dihitung dari *cost* dari simpul awal ke simpul sekarang (*current node*) ditambah *cost* perkiraan menuju simpul tujuan.
2. Simpul di *list* "tertutup" bisa dimasukkan ke list "terbuka" bila jalan terpendek (*cost* lebih kecil) menuju simpul tersebut ditemukan. Karena *list* "terbuka" diurutkan berdasarkan perkiraan *cost* keseluruhan, algoritma mengecek simpul-simpul yang

mempunyai perkiraan *cost* yang paling kecil terlebih dahulu, jadi algoritmanya mencari simpul-simpul yang kemungkinan mengarah ke simpul tujuan. Karena itu, lebih baik perkiraan *cost*-nya, lebih cepat pencariannya. *Cost* dan perkiraannya ditentukan oleh kita sendiri. Bila *cost*-nya adalah jarak, akan menjadi mudah.

Cost antara simpul adalah jaraknya, dan perkiraan *cost* dari suatu simpul ke simpul tujuan adalah penjumlahan jarak dari simpul tersebut ke simpul tujuan. Atau agar lebih mudahnya bisa ditunjukkan seperti berikut ini.

$$f(n) = g(n) + h(n)$$

dengan :
$f(n)$ = fungsi evaluasi
$g(n)$ = biaya *(cost)* yang sudah dikeluarkan dari keadaan sampai keadaan n
$h(n)$ = estimasi biaya untuk sampai pada suatu tujuan mulai dari n

Perhatikan bahwa algoritma ini hanya bekerja bila *cost* perkiraan tidak lebih besar dari *cost* yang sebenarnya. Bila *cost* perkiraan lebih besar, bisa jadi jalan yang ditemukan bukanlah yang terpendek. *Node* dengan nilai terendah merupakan solusi terbaik untuk diperiksa pertama kali pada $g(n) + h(n)$. Dengan fungsi *heuristic* yang memenuhi kondisi tersebut, maka pencarian dengan algoritma A* dapat optimal.

1.2 Implementasi Algoritma A*

Setiap permainan memiliki aturan main. Hal ini mempermudah upaya menghasilkan ruang pencarian dan memberikan kebebasan pada para peneliti dari bermacam-macam ambisi dan kompleksitas sifat serta kurangnya struktur permasalahan. Papan konfigurasi yang digunakan untuk memainkan permainan ini mudah direpresentasikan pada komputer dan tidak memerlukan bentuk yang kompleks. Permainan dapat menghasilkan sejumlah besar pencarian ruang. Hal ini cukup besar dan kompleks sehingga membutuhkan suatu teknik yang tangguh untuk menentukan alternatif pengeksplorasian ruang permasalahan. Teknik ini dikenal dengan nama *heuristic* dan merupakan area utama dari penelitian tentang AI. Banyak hal yang

biasanya dikenal sebagai kecerdasan tampaknya berada dalam *heuristic* yang digunakan oleh manusia untuk menyelesaikan permasalahannya.

Fungsi *heuristic* H(n), algoritma A* dapat memfokuskan pencarian pada *node-node* yang berada pada arah yang mendekati *node* tujuan. Kemudian pencarian diterminasikan pada waktu *node* tujuan diperiksa. Hal ini dapat meminimalisasikan jumlah *node* yang harus diperiksa dan arena waktu yang diperlukan untuk mendapatkan jalur berbanding lurus dengan jumlah *node* yang diperiksa, maka waktu pencarian dapat diminimalisasikan. Walaupun jumlah *node* yang diperiksa dapat diminimalisasikan, algoritma A* mempunyai kasus terburuk.

Pada kasus ini, sebagian besar ataupun keseluruhan *node* pada jalan diperiksa, sehingga algoritma A* bekerja seperti algoritma dijkstra atau BFS *(Best-First-Search).* Ada dua hal yang dapat menyebabkan keadaan terburuk ini, yaitu keadaan sepadan dan jika jalur yang dicari tidak ditemukan.

1.3 Unity 3D

Unity 3D adalah sebuah software development yang terintegrasi untuk menciptakan video game atau konten lainnya seperti visualisasi arsitektur atau real-time animasi 3D. Unity 3D dapat digunakan pada microsoft Windows dan MAC OS X, dan permainan yang dihasilkan dapat dijalankan pada Windows, MAC, Xbox 360, PlayStation 3, Wii, iPad, iPhone, Android dan Linux. Unity 3D juga dapat menghasilkan permainan untuk browser dengan menggunakan plugin Unity Web Player. Unity 3D juga memiliki kemampuan untuk mengekspor permainan yang dibangun untuk fungsionalitas Adobe Flash 3D.

2. PERANCANGAN

Game Ena Burena : Melarikan Diri Dari Negeri Asing yang bergenre TPS (Third Person Shooter). Game ini menceritakan kisah seorang prajurit Indonesia yang bernama Kapten Ena Burena yang mendapat tugas dari hasil konferensi para Presiden, Dokter dan Profesor di seluruh dunia untuk menyerang negeri antah berantah yang dihuni para robot jahat dengan menggunakan mesin waktu yang dikirim ke tahun 2050.

Game ini bergrafis 3D yang cara permainannya sendiri masih menggunakan sistem single Player. AI (Artificial Intelligence) yang digunakan untuk musuh dalam

Game ini adalah algoritma A* dan collision detection sebagai pendeteksi tubrukan apabila mengenai object.

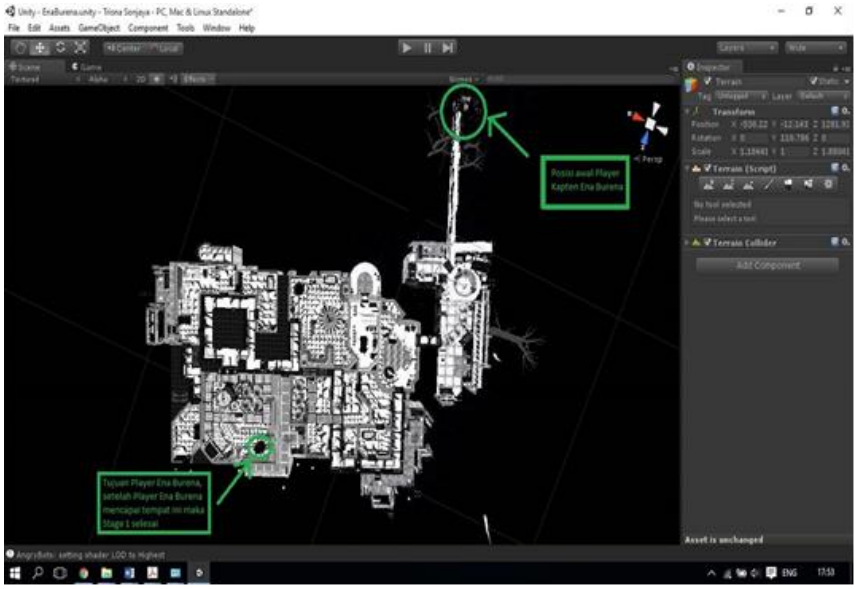
2.1 Storyline

Game ini mengambil latar belakang di dunia antah berantah yang hanya bisa ditempuh oleh mesin waktu pada tahun 2050. Pada suatu waktu Bumi mendapatkan suatu ancaman yang datang melalui masa depan. Ancaman tersebut datang melalui pesan yang dilontarkan oleh robot-robot dari masa depan kepada seluruh umat manusia di dunia melalui komputer, laptop, smartphone dan alat komunikasi lainnya yang digunakan manusia. Pesan tersebut berisikan "Halo kalian para manusia-manusia kami para robot akan menyerang Bumi pada tahun 2050 maka dari itu bersiaplah". Semenjak datangnya pesan tersebut seluruh manusia dimuka bumi menjadi panik namun mereka tidak begitu mempercayainya. Setelah adanya pesan tersebut setelah beberapa jam kemudian, alat elektronik komputer, laptop dan smartphone berubah menjadi benda bergerak menyerupai robot dan membuat semua umat manusia ketakutan namun benda tersebut tidak menyerang. Dan dari situlah mereka percaya bahwa pesan tersebut adalah benar-benar suatu ancaman.

Para Presiden, Dokter dan para Profesor di dunia mengadakan konferensi di Amerika dan mereka berhasil membuat mesin waktu. Mesin waktu tersebut menjadi harapan seluruh umat manusia. Kemudian Presiden di seluruh dunia menyiapkan pasukan terbaiknya untuk menyerang Negeri Antah Berantah yang ada di masa depan. Setelah mesin waktu sampai di kota Antah Berantah pada tahun 2050 maka perang manusia dengan robot pun dimulai. Perwakilan pasukan dari seluruh dunia ini mempunyai misi yaitu menghancurkan kota Antah Berantah yang berisi robot-robot yang jahat. Misi tersebut hampir mendekati kegagalan dikarenakan banyak pasukan yang gugur ditengah peperangan. Hanya ada satu prajurit yang masih hidup pada saat peperangan berlangsung. Prajurit tersebut berasal dari kota Nambo Indonesia yaitu kapten Ena Burena. Kapten Ena Burena hanya mempunyai dua pilihan yaitu mati atau berusaha menghancurkan robot-robot jahat tersebut. Kapten Ena Burena berusaha sekuat tenaga untuk menghancurkan robot-robot tersebut dengan mengandalkan dua senjata yang dimilikinya dan kabur dari Negeri Antah Berantah tersebut.

2.2 Storyboard

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah,. Berikut merupakan *Storyboard* dari *Game* Ena Burena *:* Melarikan Diri Dari Negeri Asing.

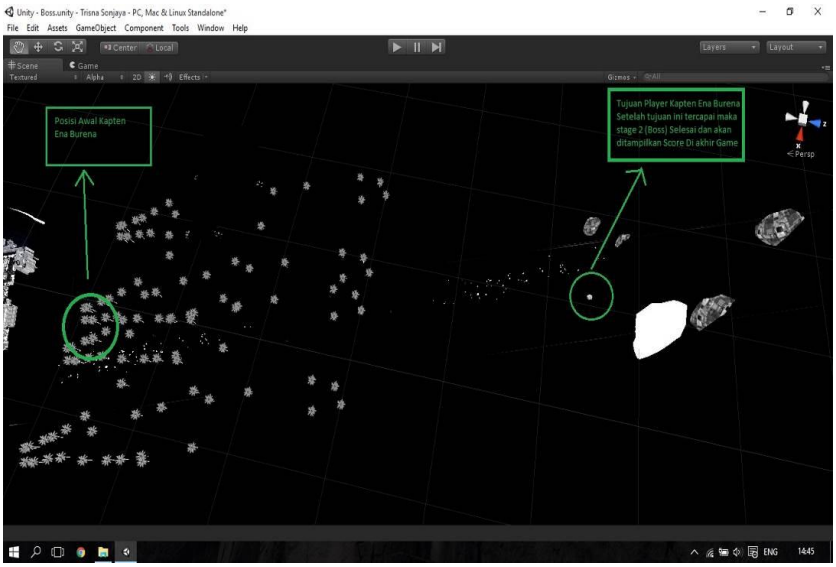


Gambar 1 : *Storyboard Stage 1* pada *Game.*

Stage 1 adalah awal mula *Player* Kapten Ena Burena menjalankan misinya untuk menghancurkan para robot. Kapten Ena Burena dibekali dua senjata *Vektor Sub Machine Guns.* Kapten Ena Burena pertama muncul melalui mesin waktu ditempat yang telah ditujkan diatas, kemudian harus melewati *lift* yang ada untu turun ke bawah. Setelah berhasil turun kebawah kemudian Kapten Ena Burena harus melewati area selanjutnya dengan membunuh para robot jahat. Robot pada *Stage* 1 bervariasi ada yang besar, kecil dengan kecepatan yang berbeda-beda maka dari itu Kapten Ena Burena harus waspada dan bergegas untuk membunuh para robot jahat tersebut. Jika Kapten Ena Burena Mati maka akan berlanjut ke *checkpoint* yang telah disediakan sehingga tidak tidak perlu memainkannya kembali dari awal.

Pada *Stage* 1 ini ada teka-teki yaitu Sang Kapten harus mencari jalur pintu yang terhubung ke computer. Setelah menemukan computer yang terhubung ke jalur pintu sang Kapten hanya perlu mendekatinya dan menunggu beberapa saat sehingga pintu tersebut dapat terbuka. Jalur teka-teki pintu ini terdapat 3 yang pertama dan kedua hanya satu jalur sedangkan yang ketiga terdapat 2 jalur sehingga sang Kapten dapat

dipastikan memakan waktu yang cukup lama dibanding dengan jalur teka-teki pintu pertama dan kedua. Disamping harus memecahkan teka-teki pintu ini sang Kapten harus membunuh para robot jahat yang berkeliaran kalau tidak sang Kapten akan mati dan memulai kembali pada *checkpoint* yang ada. Setelah sang Kapten berhasil melewati pintu dan membunuh para robot sang Kapten harus mencari tempat tujuan untuk kabur dari tempat antah berantah yang ditandai dengan *particle system* yang sama pada awal memulai game. Setelah sang Kapten mencapai tujuannya maka *Stage* 1 pun selesai dan berlanjut ke *stage* 2.



Gambar 2 : *Storyboard Stage* 2 *(Boss)* pada *Game.*

Pada *stage* 2 ini Kapten Ena Burena harus menemukan *computer hub* yaitu tujuan akhirnya untuk bisa kabur dari negeri antah berantah. Sang Kapten akan dihadapkan dengan robot-robot yang sama pada *stage 1* namun dengan ukuran yang bervariasi ada yang kecil dan ada juga yang besar *(Boss)* sehingga *stage* 2 ini akan menjadi lebih sulit dibandingkan dengan stage 1 sebelumnya. Sang Kapten pada *stage* ini hanya menggunakan satu senjata *Vektor Sub Machine Guns* dikarenakan senjata yang satunya rusak karena dipakai melawan para robot pada stage 1. *Stage* 2 ini didesain pada gurun antah berantah yang ada di masa depan tahun 2050 yang merupakan lanjutan dari *stage* 1. *Stage* 2 ini sangat berbeda dengan *stage* 1 dikarenakan *stage* ini lebih mengarah kepada peperangan dilapangan tidak seperti *stage* 1 yang peperangannya didalam ruangan yang dipenuhi *environtment* dan objek-objek lainnya.

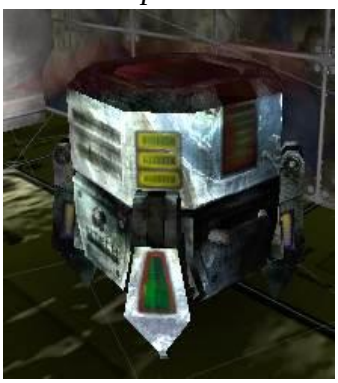
Pada akhir *game* setelah sang Kapten menuju tujuannya maka akan ditampilkan skor selama bermain, musuh yang dimusnahkan dan lain-lainnya.

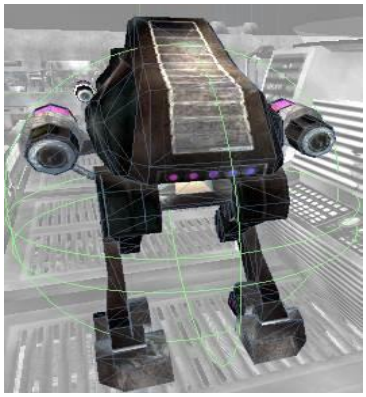
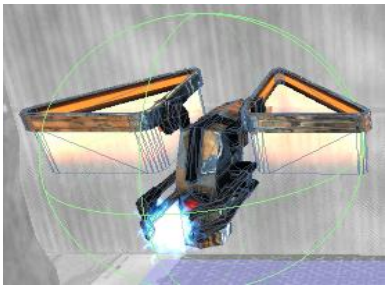
2.3 Gameplay

Dalam *Game* Ena Burena *:* Melarikan Diri Dari Negeri Asing, pemain harus menghancurkan robot yang ada dihadapan Kapten Ena Burena, tidak harus semua robot dimusnahkan karena dalam *Game* ini yang terpenting adalah *player* mencapai tujuannya untuk menyelesaikan *stage* dan berpindah ke *stage* selanjutnya. Namun jika ingin menambah skor disarankan untuk menghancurkan semua robot karena pada akhir *game* akan ditampilkan skor tentang berapa banyak robot yang telah dimusnahkan. Disetiap *stage* disediakan beberapa fasilitas *checkpoint* yang berguna ketika *player* mati, *player* bisa hidup kembali di tempat dia melewati *checkpoint.*

Setiap lawan yang dihadapi memiliki cara yang berbeda dalam memusnahkan *player,* berjalan mengejar *player,* daya tahan serangan dari *player,* kecepatan meregenerasi *health.* Di setiap *stage,* musuh memiliki kelebihan dan ukuran yang berbeda dengan *stage* sebelumnya.

Tabel 1 Penjelasan kemampuan *Enemy* dan *Player.*

Model Karakter	Kemampuan
Spider 	Pada musuh ini, *spider* mempunyai kemampuan untuk menghancurkan diri ketika mendekati *player*. Karakter *Spider* ada yang diam saja ketika *player* mendekati *spider,* ada juga yang langsung mendekati *player* untuk meledakan diri ketika *spider* terusik atau tertembak oleh *player*. Apabila *player* menjauhi *spider,* maka *spider* akan kembali ke pos pertama dia diam. *Spider* ada yang mempunyai kemampuan untuk meregenerasi *health* dan tidak..

Mech 	Pada musuh ini, Mech bertugas untuk membunuh *player* dengan misil yang keluar dari senjatanya yang berada di pinggir kepalanya. Mech mempunyai kemampuan patroli, mengejar *player* hingga mati dan menembakan misil bertubi-tubi secara diagonal. Namun ketika *player* bersembunyi dibalik dinding, Mech akan berhenti mengejar dan akan kembali berpatroli ke *waypoint* dimana dia berpatroli. Mech juga mempunyai kemampuan untuk meregenerasi *health* da nada yang tidak.
Kamikaze Buzzer 	Pada musuh ini, *kamikaze buzzer* bertugas membunuh *player* dengan sengatan listrik yang keluar dari mulutnya. *Kamikaze Buzzer* mempunyai kemampuan untuk membunuh *player* dengan arah random secara diagonal. *Kamikaze Buzzer* juga mempunyai kemampuan untuk meregenerai *health* da nada yang tidak.

Lanjutan. Tabel 1 Penjelasan kemampuan *Enemy* dan *Player*.

Ena Burena 	Ini adalah karakter Ena Burena yang user mainkan untuk menjalani misinya. Ena Burena mempunyai kemampuan menembak musuh bertubi-tubi hingga musuh hancur. Ena Burena memiliki *health* yang berada dipunggungnya , sehingga ketika *health* akan habis maka indicator *health* itu akan berkedip-kedip itu menandakan bahwa Ena Burena *healthnya* akan habis dan jiga terkena serangan robot maka Ena Burena akan mati. Untuk

	mengantisipasi agar tidak mati Ena Burena harus mengindari serangan musuh dan meregenerasi *health* dengan cara bersembunyi dari para robot sehingga *health* bisa terisi kembali namun memakan waktu yang lumayan lama. Ketika *Ena Burena* mati, dia akan *respawn* di-*checkpoint* yang telah dilewatinya. Ena Burena juga mempunyai kemampuan untuk meregenerasi *health* namun lebih cepat dibanding dengan para robot.

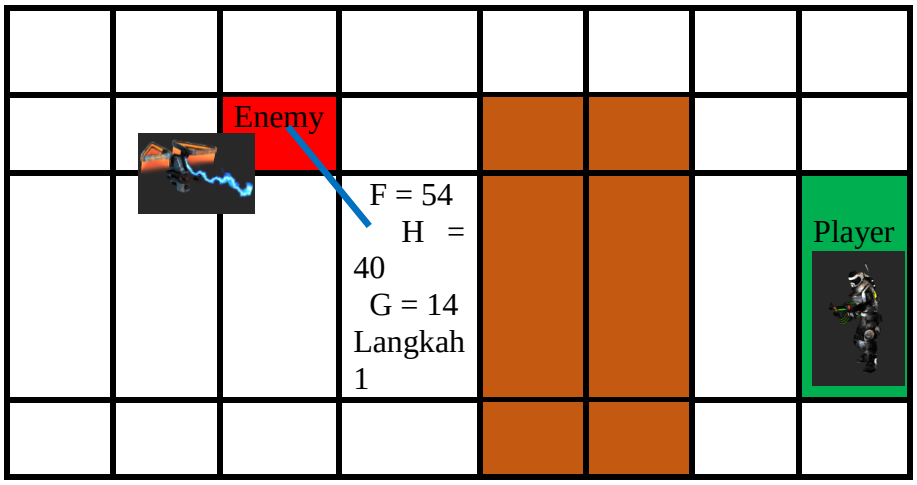
2.4 Representasi Artifisial Intelegent Menggunakan Algoritma A*

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu dalam kasus *Game* ini 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 terpilih sudah tertutup).

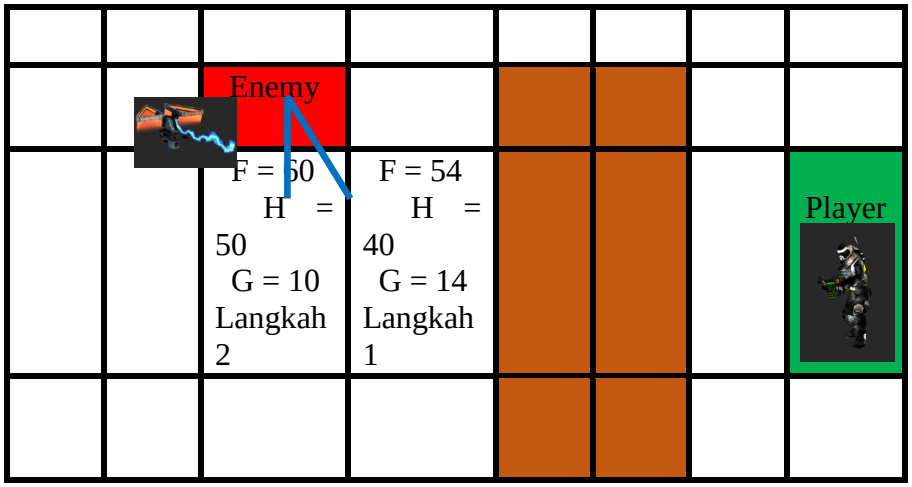
Berikut ini adalah penjelasan AI yang digunakan musuh dalam pengejaran *player,* dalam pengujian ini asumsi bahwa *player* terdeteksi oleh musuh dan *player* dalam keadaan diam ditempat. Dalam pengejaran ini, musuh menggunakan algoritma A* untuk pencarian rute terdekat agar *player* bisa ditembak.

Apabila *player* telah keluar dari batas pendeteksian, maka musuh akan kembali ke tempat semula dia berdiri melalui jejak pencarian yang telah musuh buat dan apabila *player* terhalang oleh objek yang besar maka pencarian akan diakhiri dan musuh kembali ke tempat semula dia berada.



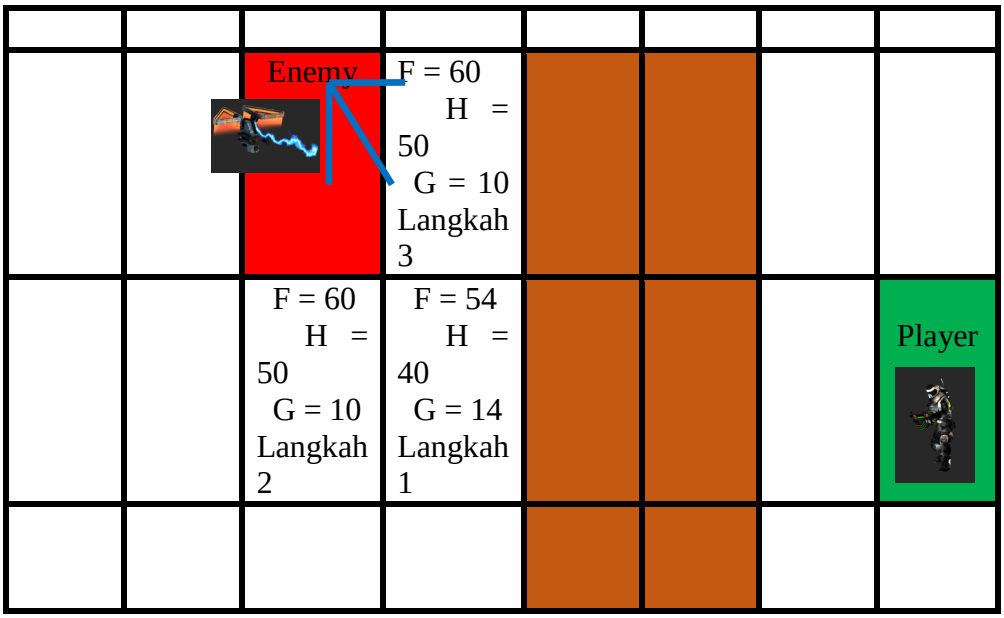
Gambar 3 Pencarian menggunakan algoritma A* langkah 1.

Dari *enemy* mencari secara diagonal dengan nilai perkiraan biaya yaitu G = 14 dan estimasi biaya yaitu H = 40 maka dengan rumus fungsi evaluasi yaitu F = G + H dihasilkan nilai F = 54.



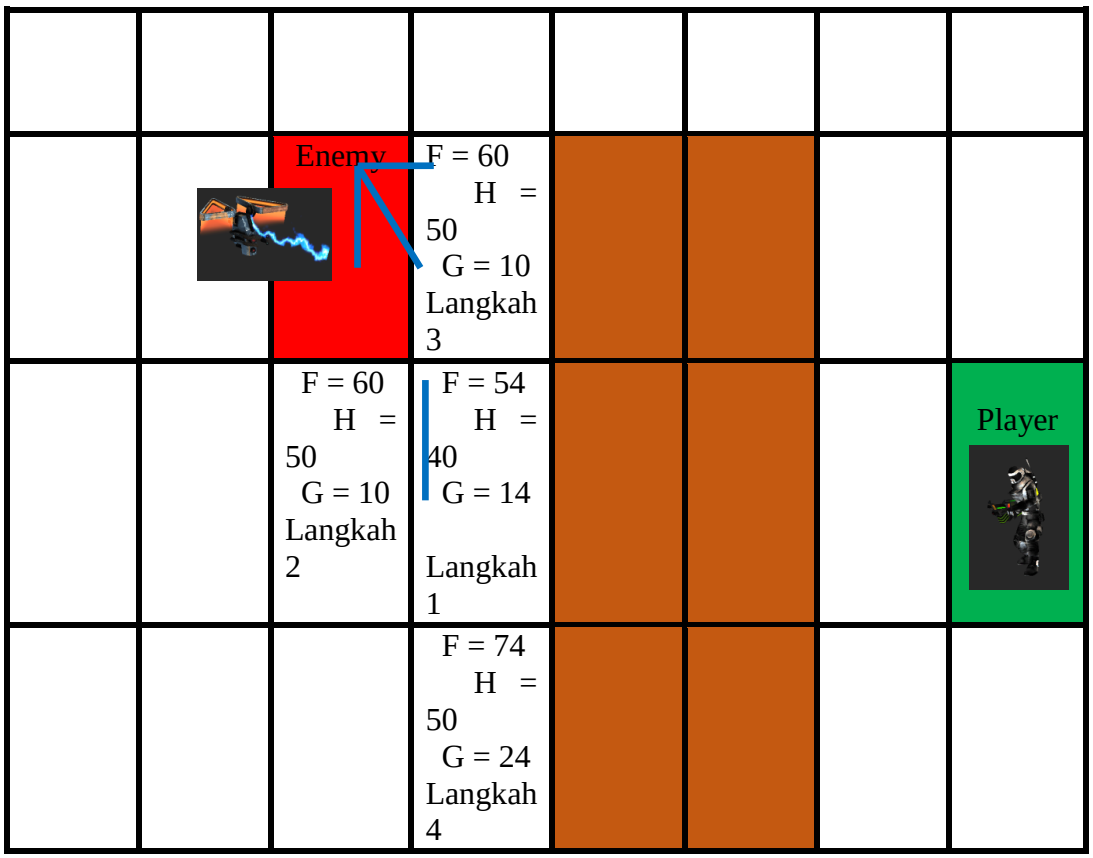
Gambar 4 : Pencarian menggunakan algoritma A* langkah 2.

Dari *enemy* mencari secara vertical dengan nilai G = 10 dan H = 50 maka dengan rumus F = G + H dihasilkan F = 60.



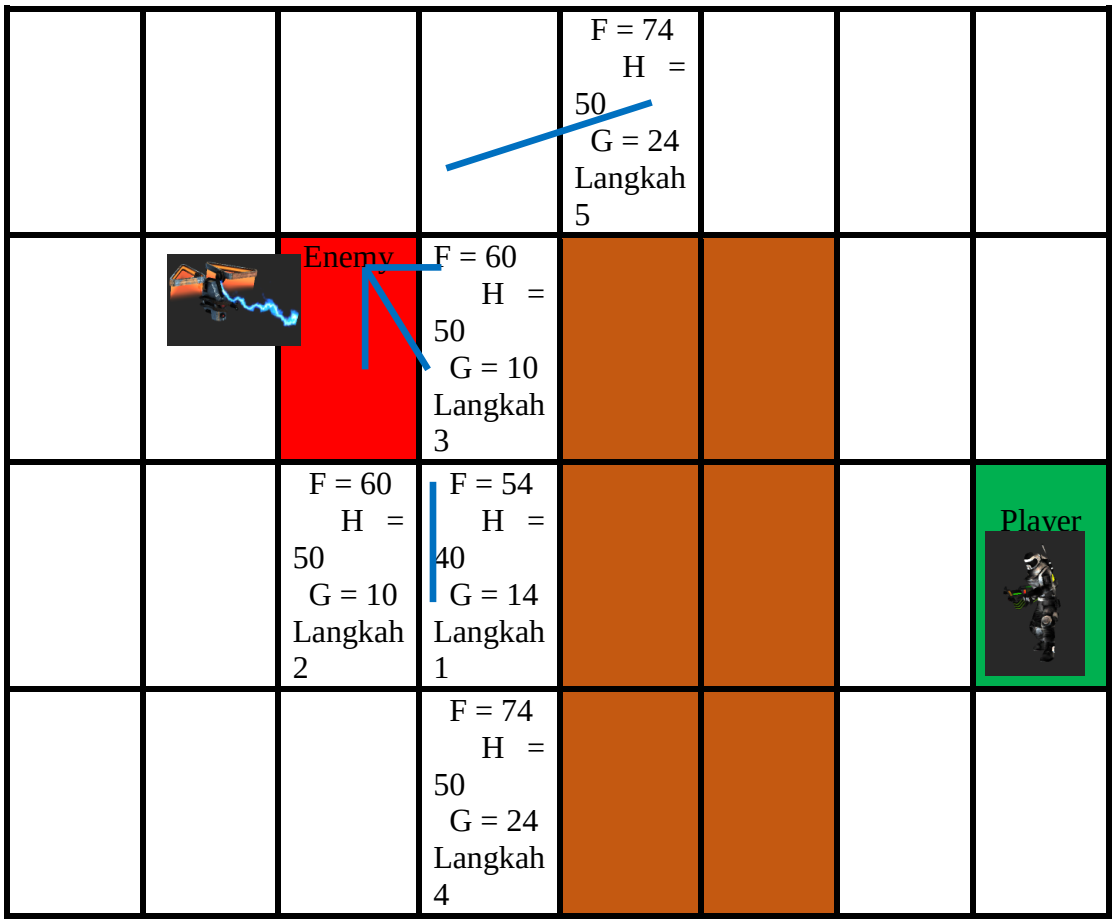
Gambar 4. Pencarian menggunakan algoritma A* langkah 3.

Kemudian dari *enemy* mencari secara horizontal dengan nilai G = 10 dan H = 50 maka dengan rumus F = G + H dihasilkan F = 60.



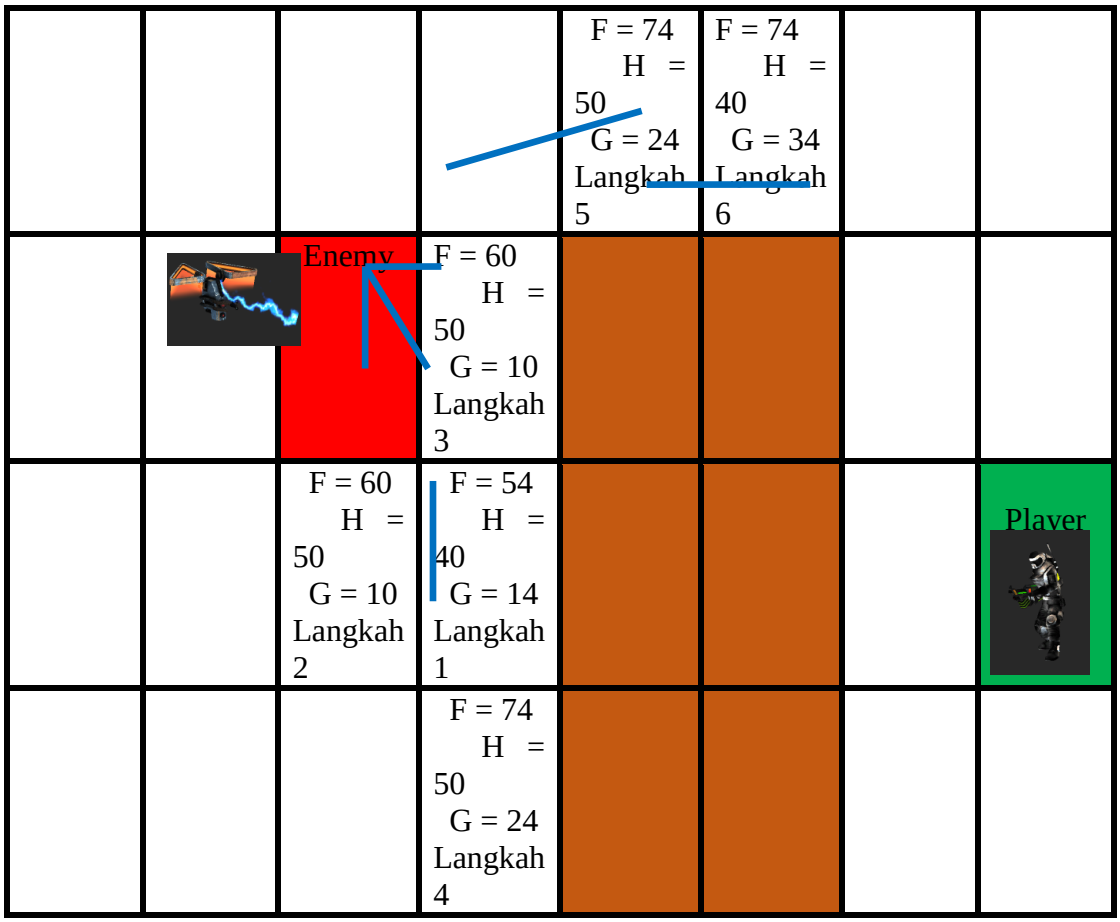
Gambar 5. Pencarian menggunakan algoritma A* langkah 4.

Kemudian dari *node* bernilai F=54, mencari secara vertikal dengan nilai G=24 yang didapatkan dari langkah 1 ditambah dengan niai G vertikal yaitu 10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=74.



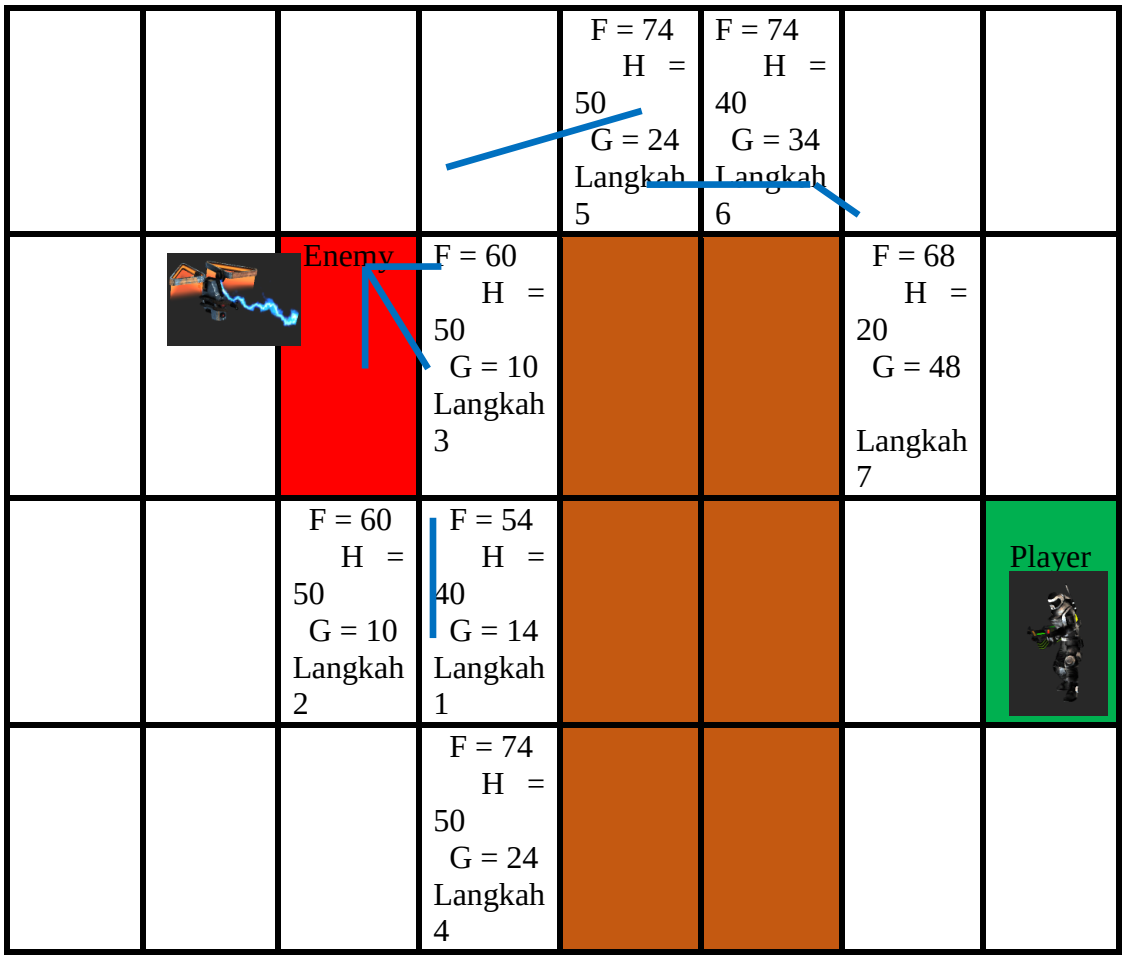
Gambar 6. Pencarian menggunakan algoritma A* langkah 5.

Kemudian dari *node* yang bernilai F=60, mencari secara diagonal dengan nilai G=24 yang didapatkan dari langakah 1 ditambah dengan nilai G vertikal yaitu 10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=74.



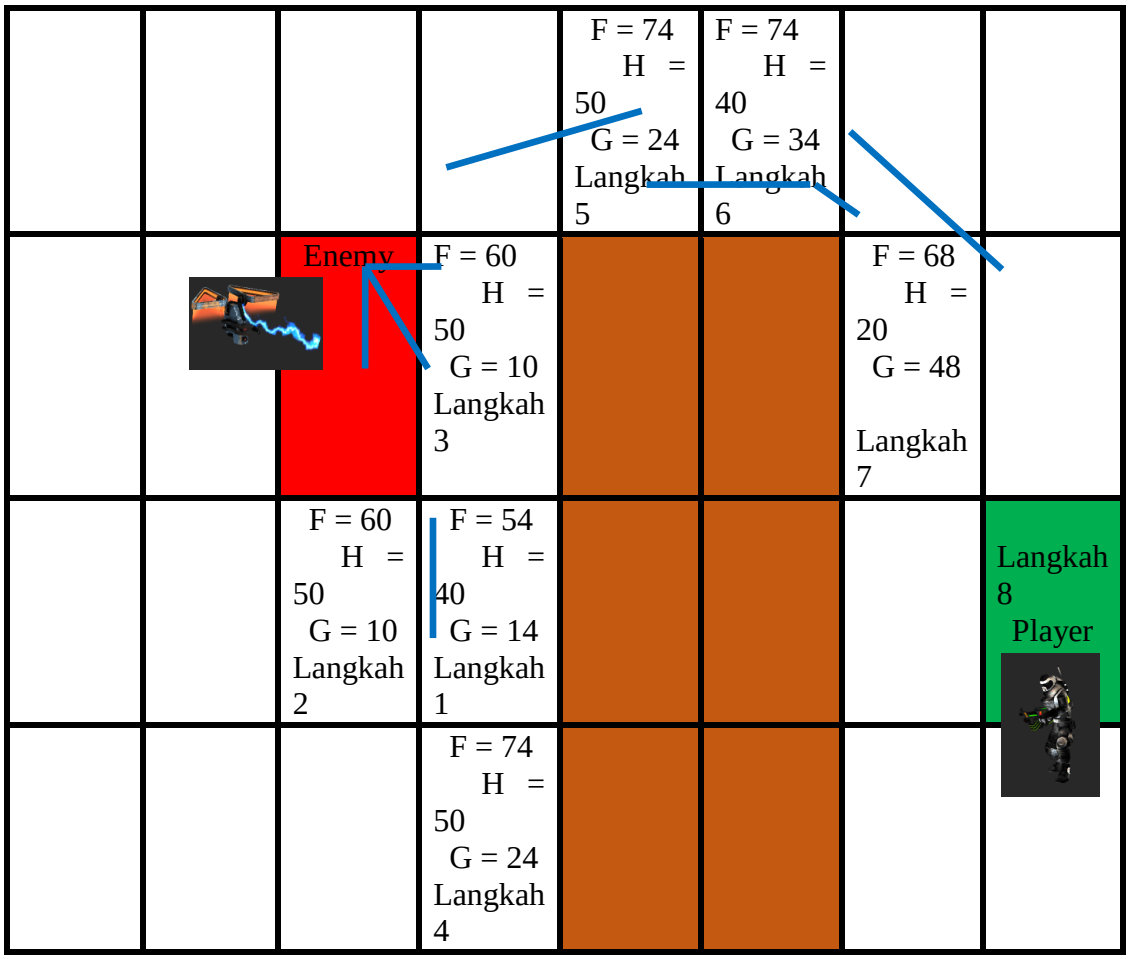
Gambar 7 Pencarian menggunakan algoritma A* langkah 6.

Kemudian dari *node* yang bernilai F=74, mencari secara horizontal dengan nilai G=34 yang didapatkan dari langkah 5 ditambah dengan nilai G vertikal yaitu 10, dan H=40 maka dengan rumus F=G+H , dihasilkanlah nilai F=74.



Gambar 8. Pencarian menggunakan algoritma A* langkah 7.

Kemudian dari *node* yang bernilai F=74, mencari secara diagonal dengan nilai G=48 yang didapatkan dari langkah 6 ditambah dengan nilai G vertikal yaitu 10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=68.

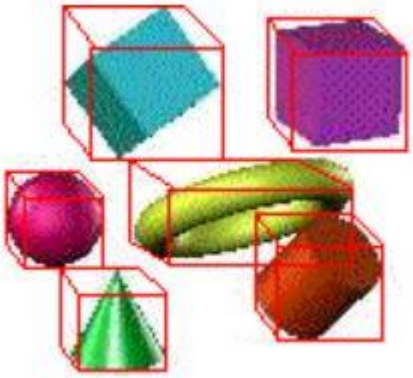


Gambar 9 Pencarian menggunakan algoritma A* langkah 8.

Maka dihasilkan jalur optimum untuk pencarian *enemy* menuju *player* dengan menggunakan algoritma A*. Proses mencari *node* dengan nilai terkecil ini dilakukan berulang-ulang sampai menemukan *node* tujuan. Apabila *node* yang menjadi titik tujuan telah ditemukan, program akan melakukan *backtrack* ke *parent* dari tiap *node* untuk mendapatkan rangkaian *node* yang membentuk rute yang akan paling optimum yang diinginkan. Setelah musuh memilih tujuan, maka pencarian pun dipilih yaitu dengan menggunakan algoritma A*.

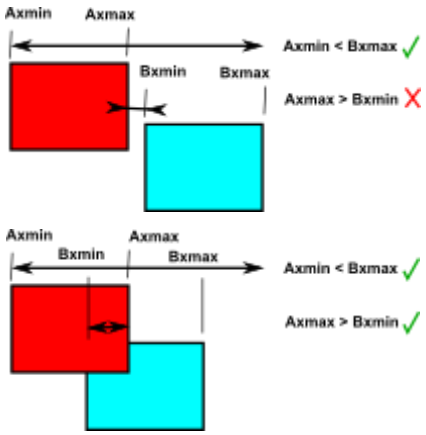
2.5 Representasi Collision Detection

Untuk memastikan tidak ada objek yang saling menembus dibutuhkan *collision detection* berdasarkan susunan geometri dari objek itu sendiri

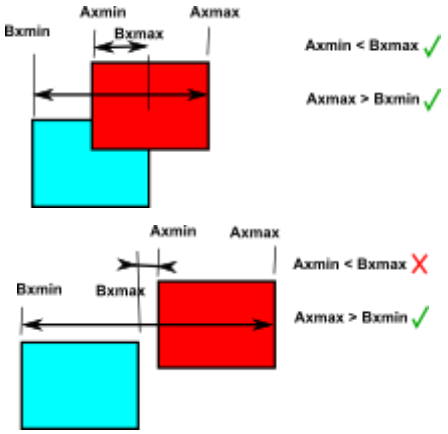


Gambar 10. *Bounding Boxes.*

Dalam gambar di atas tap bentuk dikelilingi oleh garis batas merah. Jika batas merah tersebut saling menumpuk bisa dikatakan objek di dalamnya sudah bertubrukan dengan objek di dalamnya batas merah yang lain atau belum menumbuk sama sekali. Pengujian lebih dalam diperlukan untuk mendeteksi tumbukan didalam objeknya. Tetapi jika batasnya tidak saling bertubrukan maka objeknya pun juga belum bertabrakan. Dengan cara ini bisa menghemat sumber daya CPU untuk pengujian bentuk yang lebih kompleks. Kekurangannya cara ini terpaku pada sumbu yang sejajar jika sebuah objek berotasi proses akan menjadi semakin panjang karena setiap pendeteksian harus didahului dengan pemeriksaan koordinat yang selalu berubah ketika objek melakukan rotasi.

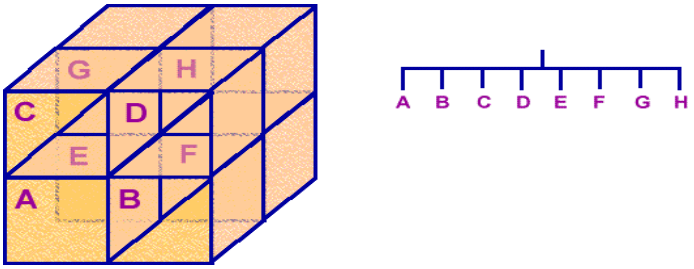


Gambar 11 Perhitungan kotak Axmin, Axmax, Bxmin dan Bxmax.



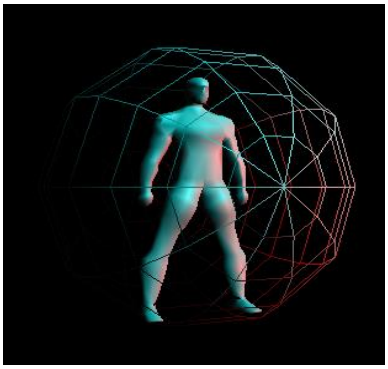
Gambar 12 Perhitungan kotak Axmin, Axmax, Bxmin dan Bxmax.

Sebagai contoh jika kotak 'A' didefinisikan oleh AxMin, AxMax, Aymin, AyMax, Azmin, dan AZMAX. dan kotak 'B' didefinisikan oleh BxMin, BxMax, ByMin, Bymax, BzMin, dan BzMax. Kemudian kotak tumpang tindih jika semua kondisi berikut ini benar: AxMin <BxMax dan AxMax> BxMin Aymin <Bymax dan AyMax> ByMin Azmin <BzMax dan AZMAX> BzMin Diagram di atas menunjukkan kondisi di x dimensi. Untuk kotak akan tumpang tindih dalam 3 dimensi maka mereka juga harus tumpang tindih dalam y dan z dimensi. Namun ini hanya berlaku jika kotak bounding yang sumbu sejajar. Jika kotak bounding didefinisikan dalam koordinat lokal dan salah satu kotak berada di bawah mengubah kelompok dengan rotasi maka kita harus baik: menggunakan algoritma untuk mendeteksi persimpangan kotak berorientasi sewenang-wenang, di koordinat absolut, yang akan jauh lebih kompleks. atau menghitung ulang kotak batas pada setiap frame di sumbu sejajar koordinat mutlak dan melakukan hal-hal di setiap frame adalah overhead besar. Jika satu kotak di sekitar objek tidak memberikan deteksi tabrakan cukup akurat untuk bentuk maka dimungkinkan untuk menggunakan beberapa kotak seperti octrees.



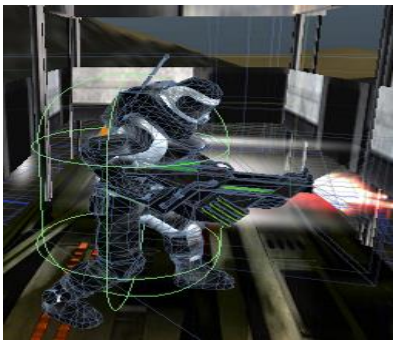
Gambar 13 anak kotak Octrees.

Octrees dapat digunakan untuk mewakili bentuk untuk rendering misalnya atau kita mungkin menggunakan beberapa metode lain, seperti poligon, untuk rendering tetapi menggunakan octrees untuk deteksi tabrakan atau penghapusan objek tersembunyi. Untuk mengeliminasi kekurangan yang ada pada metode bounding boxes dan Octtrees maka digunakan metode *bounding spheres. Bounding Spheres* memiliki orientasi yang lebih bebas, jadi pendeteksian tidak terpaku pada sumbu yang sejajar seperti metode *bounding boxes.* Tetapi *bounding spheres* tidak terlalu cocok dengan objek yang panjang dan tipis dan akan sering terjadi deteksi tabrakan palsu



Gambar 14. *Bounding Spheres.*

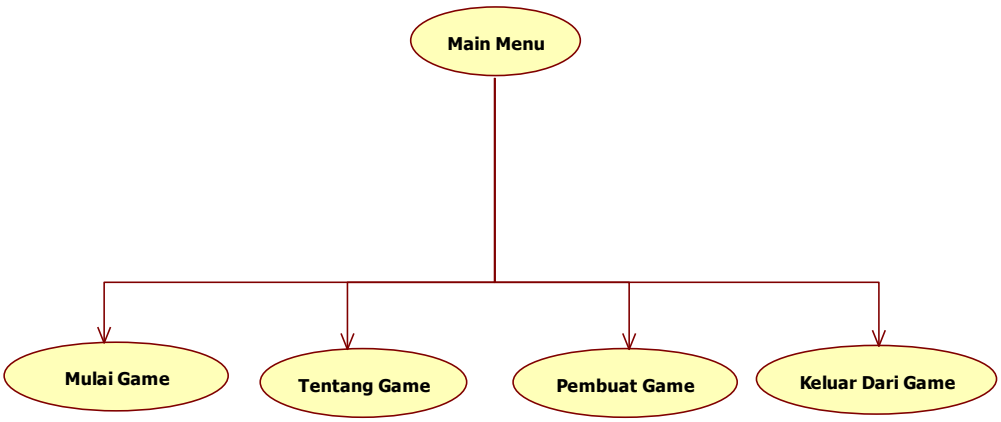
Untuk memaksimalkan *collision detection* maka pada *game* ini digunakan *bounding capsules* sehingga *player* ketika bertubrukan dengan objek lain dapat meminimalisir deteksi tubrukan palsu.



Gambar 15 : *Bounding Capsules.*

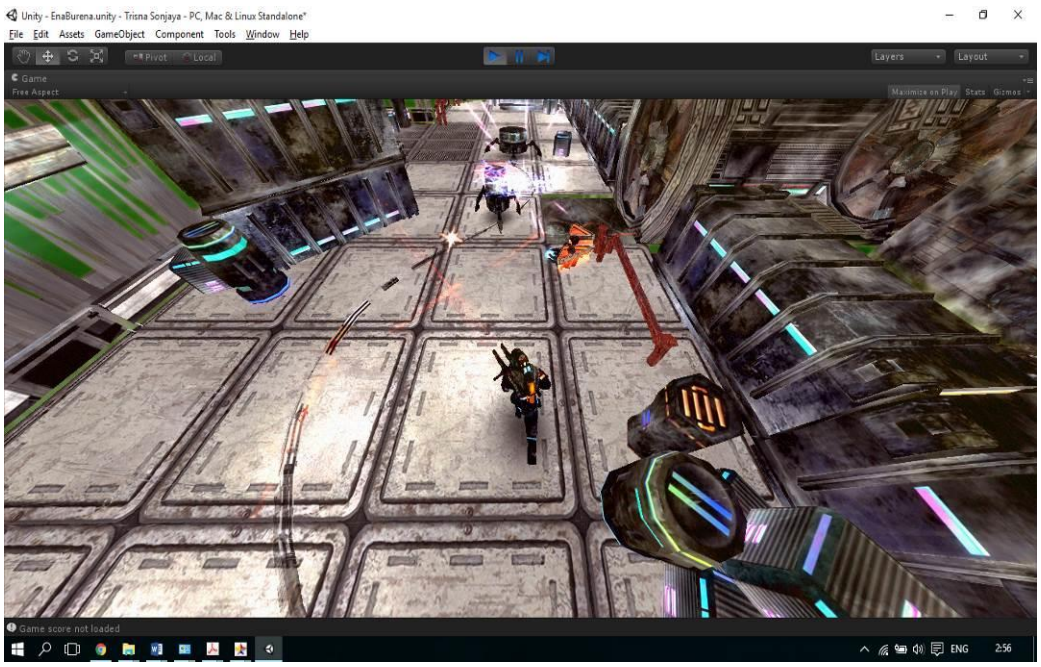
3. IMPLEMENTASI

Untuk memudahkan dalam pembangunan game Ena Burena ini di implementasikan dengan menggunakan *Unity* 3D, *Autodesk* 3ds *Max* 2015 dan bahasa pemograman C#. Berikut menu pada game ini :

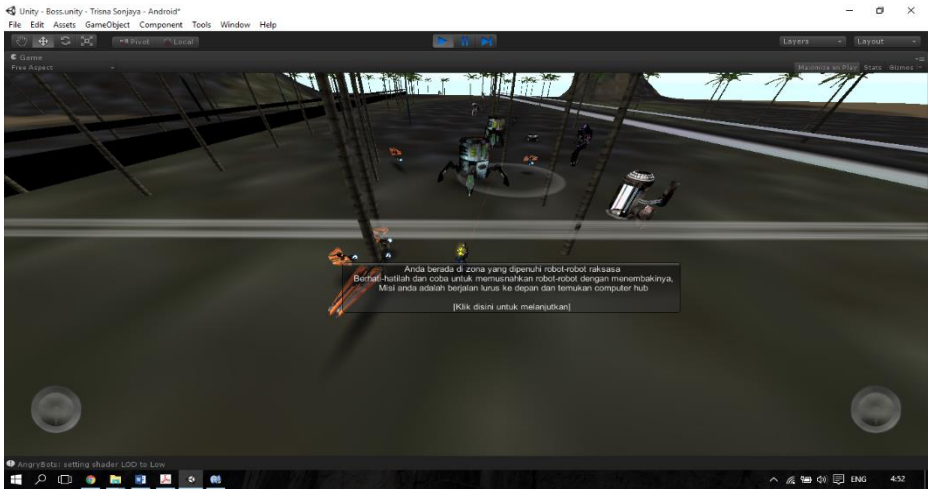


Gambar 16. Struktur menu pada game Ena Burena

.



Gambar 17. Tampilan *Stage* 1 *player* dengan *enemy* pada *Platform Desktop*.



Gambar 18. Tampilan *stage* 2 pada *Platform Android.*

4. KESIMPULAN

Collision detection sangat penting dalam sebuah *game* karena tanpa adanya *collision detection* (deteksi tubrukan) maka satu objek dengan objek lainnya akan saling berpapasan dan bertubrukan dengan adanya *Collision detection* maka game trampak lebih realistis sedangkan Algoritma A* (A Star) berfungsi sebagai metode *pathfinding* yaitu digunakan pada musuh *(enemy)* mencari jalur terbaik untuk berinteraksi dengan *player.*

Hasil implementasikan game ini masih jauh dari sempurna masih harus terus dikembangkan dan di sempurnakan seperti penambahan penambahan *stage* karena jika hanya terdapat 2 *stage* saja pemain pasti akan merasa cepat bosan dan untuk pengembangan selanjutnya diharapkan dapat memainkan *game* ini secara *multiplayer* baik itu secara LAN *(Lokal Area Network)* maupun menggunakan *server* Internet.

5. DAFTAR PUSTAKA

Rachmatullah, A., 2002, Mempelajari C#: Bahasa Pemrograman Modern, E-Book.

Henry, Samuel., 2005, Panduan Praktis Membuat Game 3D, GRAHA ILMU, Yogyakarta.

Nugroho, Adi., 2009, Rekayasa Perangkat Lunak Menggunakan UML dan Java.

Karamian , Vahé., Introduction to Game Programming Using C# and Unity 3D

Widiyanto, Rahmat., 2010, Teknik Profesional Photosop CS 3

Alhadi, E. J. Article About 4 OS Android, Windows, Linux, and Mac, terhubung

Untoro, F. X. W. Y, 2010, Algoritma dan Pemrograman dengan Bahasa Java, Graha Ilmu, Yogyakarta

Blackman, S., 2011, Beginning 3D Game Development with Unity, Apress, New York.

Nugraha, R.M. 2011, Penggunaan Struktur Data Quad-Tree dalam Algoritma Collision Detection pada Vertical Shooter Game, Makalah IF3051 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Bandung.

Unity, 2016, unity docs docs.unity3d.com/ScriptReference/Animation.html, diakses pada Januari 2016.

Unity, 2016, unity tutorial unity3d.com/learn/tutorials/modules, diakses pada januari 2016.

Autodesk, Inc. Autodesk 3ds Max Products. Autodesk. [Online] 2012. [Dikutip: Februari 2016.] usa.autodesk.com/3ds-max/.

Technologies, Unity. Create Games With Unity. Unity. [Online] 2012. [Dikutip: Februar 2016.] unity3d.com/.

Technologies, Unity. Runtime Classes. Unity. [Online] 2012. [Dikutip: Februari 2016.] docs.unity3d.com/Documentation/ScriptReference/20_class_hierarchy.html