P-ISSN: 2548-9704
E-ISSN: 2686-0880

ROLE PLAYING GAME (RPG) QUIZ MENGGUNAKAN ALGORITMA A* DAN PERLIN NOISE

Yusfrizal

Universitas Potensi Utama
Jl. K. L. Yos Sudarso Km.6,5 No.3-A Medan (20241)
E-mail : yusfrizal80@gmail.com

ABSTRAK

Salah satu jenis permainan yang terkenal yaitu RPG. RPG merupakan permainan yang memakai dasar cerita untuk kemudian diubah menjadi sebuah permainan. Dimana pemain berada di sebuah dunia khayalan dan memainkan tokoh dalam cerita tersebut. RPG terdiri dari beberapa elemen kunci di antaranya yaitu penjelajahan, dan pertarungan. Algoritma A* digunakan untuk mencari jalur terpendek antara dua buah titik. Dalam penelitian ini algoritma A* digunakan untuk memecahkan masalah pencarian jalur terpendek dari Non Player Character (NPC) ke karakter pemain. Sedangkan NPC bergerak mengikuti jalur yang dibuat menggunakan algoritma A* untuk mengejar dan menyerang karakter pemain. Sedangkan algoritma Perlin Noise merupakan algoritma untuk menghasilkan coherent noise yang memenuhi ruang. Algoritma Perlin Noise menghasilkan texture 2 dimensi yang tidak berubah yang kemudian diproses menjadi terrain yang dapat dijelajahi. Permainan quiz yang dihasilkan dari penelitian ini dapat dimainkan di Personal Computer (PC) secara singleplayer dan offline. Berdasarkan hasil pengujian yang telah dilakukan, permainan ini merupakan gabungan antara quiz serta memuat semua elemen kunci dari game RPG seperti cerita dan pengaturan, penjelajahan dan pencarian, item dan inventory, perkembangan karakter, experience dan level, pertarungan, UI dan grafis.

Kata Kunci: permainan, jalur terpendek, algoritma A*, algoritma Perlin Noise

ABSTRACT

One of the most popular types of games is RPG. RPG is a game that uses a basic story which is then turned into a game. Where players are in an imaginary world and play the characters in the story. RPG consists of several key elements including exploration and combat. The A algorithm is used to find the shortest path between two points. In this study the A* algorithm is used to solve the problem of finding the shortest path from a Non Player Character (NPC) to a player character. Meanwhile, the NPC moves along a path made using the A* algorithm to chase and attack the player character. Meanwhile, the Perlin Noise algorithm is an algorithm for producing coherent noise that fills space. The Perlin Noise algorithm produces an unchanging 2-dimensional texture which is then processed into an exploreable terrain. The quiz game resulting from this research can be played on a Personal Computer (PC) in single player and offline mode. Based on the results of the tests that have been done, this game is a combination of quizzes and contains all the key elements of RPG games such as story and settings, exploration and quests, items and inventory, character development, experience and levels, combat, UI and graphics.*

Keywords: game, shortest path, A algorithm, Perlin Noise algorithm*

P-ISSN: 2548-9704
E-ISSN: 2686-0880

I. PENDAHULUAN

Perkembangan teknologi dari waktu ke waktu mengalami kemajuan yang sangat cepat telah menyebabkan berbagai perubahan di kehidupan manusia mulai dari kedokteran, keuangan, transportasi, komunikasi, industri, hingga hiburan. Salah satu hiburan yang tercipta karena perkembangan teknologi yaitu game. Game sendiri disukai dari mulai anak – anak, remaja hingga orang dewasa. Game terbagi menjadi 2 ada yang dua dimensi dan yang tiga dimensi (3D). Game 3D memiliki tampilan yang lebih nyata dibandingkan game 2D [1].

Game terdiri dari beberapa jenis antara lain FPS, RTS, Action, RPG, dll. Salah satu genre game terpopuler yaitu game RPG. RPG merupakan singkatan dari Role Playing Game atau dalam bahasa indonesia berarti permainan peran, yaitu sebuah jenis game yang memakai dasar cerita untuk kemudian diubah menjadi sebuah permainan. Dimana pemain berada di sebuah dunia khayalan dan memainkan tokoh dalam cerita tersebut. Game sendiri lebih dikenal sebagai hiburan padahal game dapat dijadikan sebagai salah satu sarana untuk menambah pengetahuan [2]

Pertarungan merupakan salah satu elemen yang tidak dapat dipisahkan dari game di mana pemain akan melawan musuh yang dikendalikan oleh komputer menggunakan kecerdasan buatan untuk mengatur tindakannya. Kecerdasan buatan merupakan salah satu cabang dari ilmu komputer untuk memberikan suatu pengetahuan pada komputer agar dapat mampu menyelesaikan tugas – tugas atau berpikir seperti manusia. Saat ini banyak sekali bidang – bidang yang memanfaatkan kecerdasan buatan sebagai alat bantu dalam melakukan pekerjaan antara lain bidang kesehatan, industri, penerbangan, militer, dan tidak terkecuali game [3].

Permasalahan yang sering terjadi yaitu kebanyakan RPG hanya mementingkan hiburan saja tanpa ada unsur pengetahuannya. Oleh karena itu penulis ingin membuat game RPG yang tidak hanya memberikan hiburannya tapi juga memberikan pengetahuan berupa kuis yang harus diselesaikan oleh pemain jika ingin menyelesaikan game. Kuis yang harus diselesaikan pemain telah dibagi ke dalam beberapa tema antara lain matematika dan kimia. Kuis yang dipilih yaitu jenis pilihan ganda karena pilihan ganda lebih mudah untuk dikerjakan oleh pemain daripada kuis dengan jenis lain [4].

Salah satu algoritma kecerdasan buatan yang dapat diterapkan di game RPG yaitu algoritma A*. Algoritma A* adalah algoritma pencarian graph yang menemukan jalur dari node awal ke node akhir. Algoritma ini menggunakan fungsi heuristic untuk menentukan urutan di mana pencarian dilakukan dengan mengunjungi node dalam graf. Fungsi heuristic yang digunakan algoritma A* untuk memecahkan kasus bervariasi tergantung dari kasus yang akan dihadapi, disini digunakan untuk memecahkan kasus pencarian jalur terpendek dari NPC ke pemain [5].

Salah satu algoritma yang sering digunakan di dalam PCG yaitu algoritma Perlin Noise yang akan menghasilkan texture 2D yang akan tetap sama hasilnya selama masukkannya tidak berubah. Kemudian texture 2D ini dapat dijadikan sebagai heightmap untuk kemudian diwarnai berdasarkan pengaturan yang telah diatur kemudian dibuat mesh dan dirubah ketinggiannya berdasarkan heightmap yang telah dibuat maka jadilah 3D terrain yang dihasilkan hanya dari baris kode tanpa membuka program 3D [6].

2. METODOLOGI

Metode yang digunakan dalam penelitian ini adalah metode pengumpulan data dan metode pengembangan sistem yang akan dirancang.

2.1 Metode Pengumpulan Data

P-ISSN: 2548-9704
E-ISSN: 2686-0880

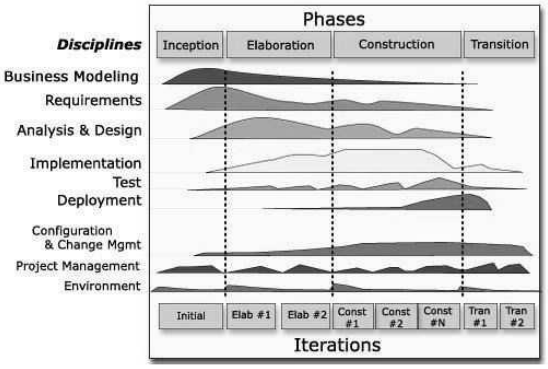
Studi literatur merupakan pilihan dalam mencari dan mengumpulkan literatur-literatur ilmiah yang diambil selain dari buku-buku yang ada. Serta mencari dari internet yang terkait dengan permasalahan yang dihadapi dalam penyusunan penelitian.

2.2 Metode Pengembangan Sistem

Metode penyelesaian masalah yang digunakan dalam penelitian ini adalah menggunakan metode RUP (Rational Unified Process). RUP (Rational Unified Process) adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang (iterative), fokus pada arsitektur (architecture-centric), lebih diarahkan berdasarkan penggunaan kasus (use case driven) [7].

RUP merupakan proses rekayasa perangkat lunak dengan pendefinisian yang baik (well defined) dan penstrukturan yang baik (well structured). RUP menyediakan pendefinisian struktur yang baik untuk alur hidup proyek perangkat lunak. RUP adalah sebuah produk proses perangkat lunak yang dikembangkan oleh Rational Software yang diakuisisi oleh IBM di bulan Februari 2003 [8].

RUP memiliki empat buah tahap atau fase yang dapat dilakukan pula secara iteratif. Berikut ini adalah gambar tahapan RUP:



Gambar 2.1. Tahap Rational Unified Process

Pada Gambar 1 di atas merupakan tahapan RUP yang diawali dengan Fase Inception, Elaboration, Construction, dan diakhiri dengan fase Transition.

Berikut ini penjelasan untuk setiap fase RUP :

1. Inception (permulaan), tahap ini lebih pada memodelkan proses bisnis yang dibutuhkan (business modeling) dan mendefinisikan kebutuhan akan sistem yang akan dibuat (requirements). Tahapan yang dibutuhkan pada tahap ini adalah memahami ruang lingkup dari proyek (termasuk pada biaya, waktu, kebutuhan, resiko dan lain sebagainya) dan membangun kasus bisnis yang dibutuhkan. Pada tahapan ini dibuat proposal penelitian dan membuat penjadwalan [9].
2. Elaboration (perluasan/perencanaan), tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah arsitektur sistem yang diinginkan dapat dibuat atau tidak. Mendeteksi resiko yang mungkin terjadi dari arsitektur yang dibuat. Tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (prototype). Pada tahap ini dibuat analisa masalah yang ada dan membuat konsep pemecahan masalah [10].
3. Construction (konstruksi), tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih pada implementasi dan pengujian sistem yang focus pada implementasi perangkat lunak pada kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari Initial Operation Capability Milestone atau batas tonggak kemampuan operasional awal. Pada tahap ini dilakukan sederatan iterasi kemudian membuat perancangan aplikasi, coding program, dan testing beta performance.
4. Transition (transisi), tahap ini lebih pada deployment atau instalasi sistem agar dapat dimengerti oleh user. Tahap

P-ISSN: 2548-9704
E-ISSN: 2686-0880

ini menghasilkan produk perangkat lunak dimana menjadi syarat dari Initial Operation Capability Milestone atau batas/tonggak kemampuan operasional awal. Aktifitas pada tahap ini termasuk pada pelatihan user, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan user. Pada tahap ini dibuat apa yang sudah dimodelkan menjadi suatu produk jadi [11].

Produk perangkat lunak juga disesuaikan dengan kebutuhan yang didefinisikan pada tahap inception. Jika semua kriteria objektif terpenuhi maka dianggap sudah memenuhi Product Release Milestone (batas/tonggak peluncuran produk) dan pengembangan perangkat lunak selesai dilakukan. Akhir dari keempat fase ini adalah produk perangkat lunak yang sudah lengkap. Keempat fase pada RUP akan dijalankan secara berurutan dan iteratif dimana setiap iterasi dapat digunakan untuk memperbaiki iterasi berikutnya.

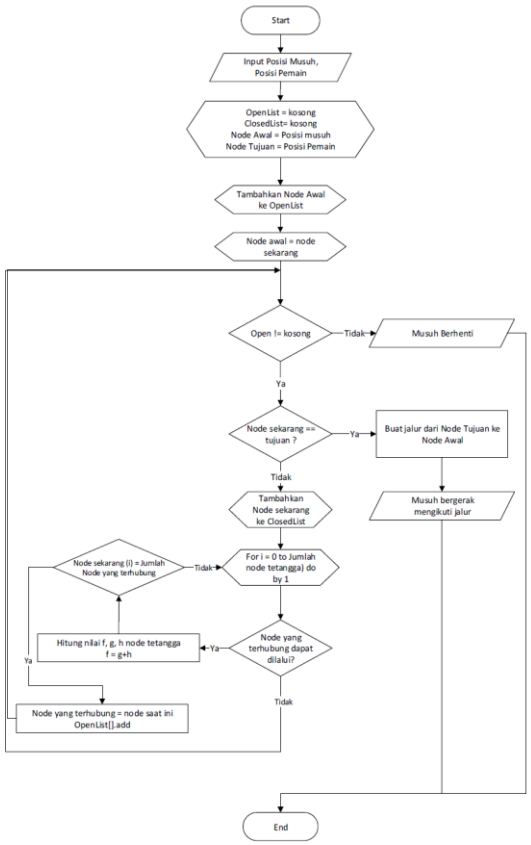
2.3 Analisis Sistem

Kebutuhan fungsional berisi proses-proses apa saja yang nantinya dilakukan oleh sistem. Kebutuhan fungsional juga berisi informasi-informasi apa saja yang harus ada dan dihasilkan sistem. Game yang dikembangkan harus mempunyai functional requirements sebagai berikut:

1. Karakter pemain dapat mengikuti jalan cerita dan quest yang telah ditentukan secara berurutan.
2. Karakter pemain dapat mengelola inventory.
3. Karakter pemain dapat berkembang menjadi lebih kuat dengan mengganti equipment dengan yang lebih baik dan menaikkan level.
4. Karakter pemain dapat bertarung melawan mush dengan tipe real-time combat.
5. Di dalam game terdapat 2 kota serta karakter pemain dapat berpindah antar kota dengan mudah,
6. Terdapat 2 buah kuis bertipe pilihan ganda dengan tema matematika dan kimia.

2.4 Analisis Algoritma

Algoritma A* yang akan diterapkan pada game RPG berjudul Prosedur merupakan algoritma pencarian jalur terpendek. Penerapan algoritma diterapkan pada musuh, dimana musuh akan mengikuti jalur yang telah dibuat menggunakan algoritma A* dengan tujuan karakter pemain.



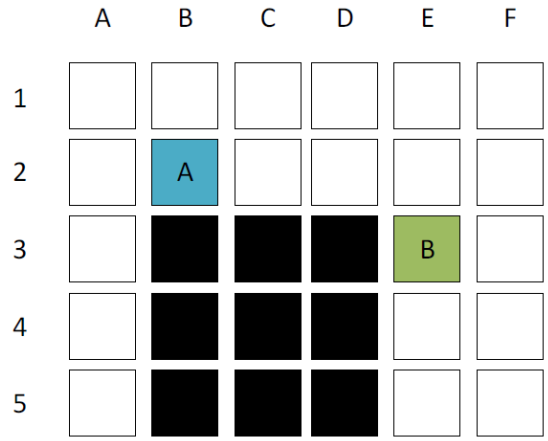
Gambar 2.2. Flowchart Proses Algoritma A*

Untuk mengetahui penerapan algoritma A* di dalam game RPG Prosedur maka di bawah ini diberikan langkah – langkah proses pembangkitan jalur :

1. Langkah pertama, Input posisi musuh dan posisi pemain. OpenList dan ClosedList masih kosong. Masukan node awal, dimana node awal ini adalah posisi musuh. Masukan node

P-ISSN: 2548-9704
E-ISSN: 2686-0880

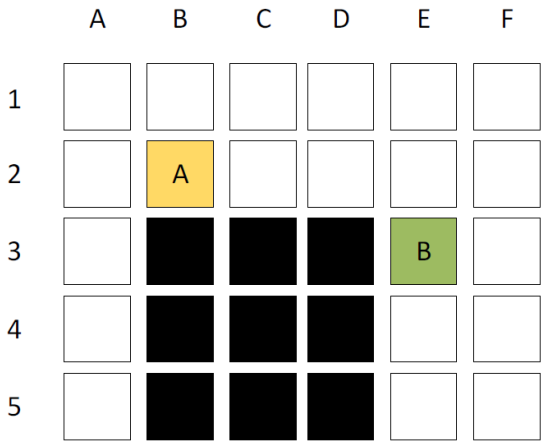
tujuan sebagai posisi pemain.



Gambar 2.3. Grid Pencarian Jalur Terpendek

Keterangan :
A = Nama node posisi musuh (B2)
B = Nama node posisi pemain (E3)

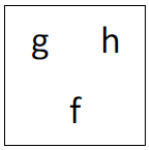
2. Langkah kedua, masukan node awal ke OpenList, kemudian jadikan node awal sebagai node sekarang.



Gambar 2.4. Grid Pencarian Jalur Terpendek

3. Langkah ketiga, periksa apakah node sekarang merupakan node tujuan ? Jawabannya bukan.
4. Langkah keempat, lakukan pemeriksaan terhadap semua node tetangga. Jika node tetangga tidak dapat dilalui maka lewati.
5. Langkah kelima, untuk semua node tetangga yang dapat dilalui lakukan perhitunggan. Kemudian masukkan ke OpenList . Pilih node dengan f terkecil. Jika sudah maka ulangi langkah ke 3 – 5 hingga menemukan node tujuan atau OpenList = kosong.

Posisi musuh = B2.
Posisi pemain = E3.
Nilai untuk langkah vertikal dan horizontal = 10.
Nilai untuk langkah diagonal = 14, nilai 14 didapat dari rumus pythagoras
Dimana $\sqrt{10^2 + 10^2} = 14{,}141$ dibulatkan kebawah menjadi 14.
Nilai posisi awal = 0.
Untuk penempatan f, g, dan h pada node dapat dilihat pada gambar berikut:



Gambar 2.5. Penempatan f, g, dan h pada Node

Masukan posisi awal musuh (B2) ke OpenList dan jadikkan sebagai node terbaik dan bangkitkan node tetangga, kemudian periksa apakah node tetangga dapat dilalui atau tidak. Masukkan semua node tetangga yang belum berada di OpenList ke OpenList. Hitung nilai f, g dan h pada node yang dibangkitkan (perhitungan ini dimulai dari nilai g, h dan terakhir f), lalu cari nilai f yang paling rendah pada OpenList kemudian masukan ke dalam ClosedList. Fungsi f sebagai estimasi fungsi evaluasi terhadap node n, dapat dituliskan sebagai berikut :

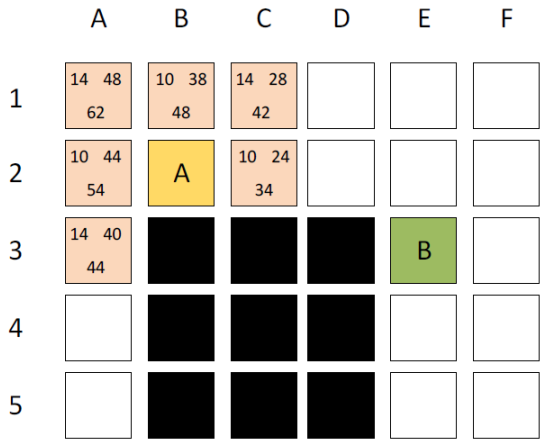
f(n) = g(n) + h(n)
Dimana :
f = g(n) + h(n)
g = Jarak dari node awal
h = Jarak dari node tujuan

P-ISSN: 2548-9704
E-ISSN: 2686-0880

Tabel 2.1. Langkah Pertama Penyelesaian Algoritma A*

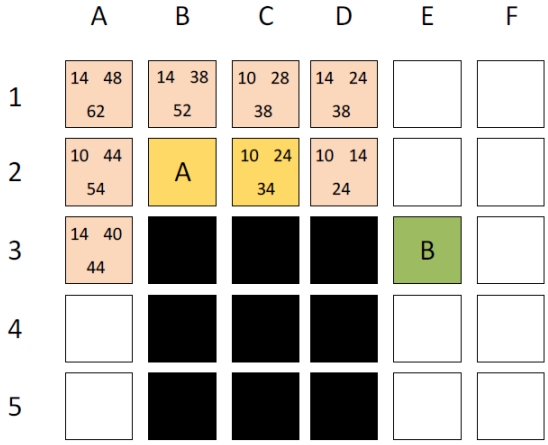
Node	Penyelesaian
n(A1)	f(n) = g(n) + h(n) f(A1) = g(A1)+h(A1) = 14+48 = 62
n(B1)	f(n) = g(n) + h(n) f(B1) = g(B1)+h(B1) = 10+38 = 48
n(C1)	f(n) = g(n) + h(n) f(C1) = g(C1)+h(C1) = 14+28 = 42
n(A2)	f(n) = g(n) + h(n) f(A2) = g(A2)+h(A2) = 10+44 = 54
n(C2)	f(n) = g(n) + h(n) f(C2) = g(C2)+h(C2) = 10+24 = 34
n(A3)	f(n) = g(n) + h(n) f(A3) = g(A3)+h(A3) = 14+40 = 44

Berdasarkan perhitungan langkah pertama diatas maka hasil perhitungan yang memiliki nilai f terkecil yaitu C2 dengan nilai 34.



Gambar 2.6. Penempatan Nilai Node Tetangga Dari Node Awal

Node C2 terpilih sebagai node terbaik karena mempunyai nilai f terendah. Kemudian periksa apakah node C2 merupakan node tujuan. Ternyata node C2 bukan merupakan node tujuan. Masukkan bestnode ke dalam ClosedList karena sudah menjadi bestnode dan bangkitkan node tetangga dari node C2 yang dapat dilalui dan bukan termasuk ClosedList. Maka didapatkan node tetangganya yaitu node (B1,C1,D1,D2).



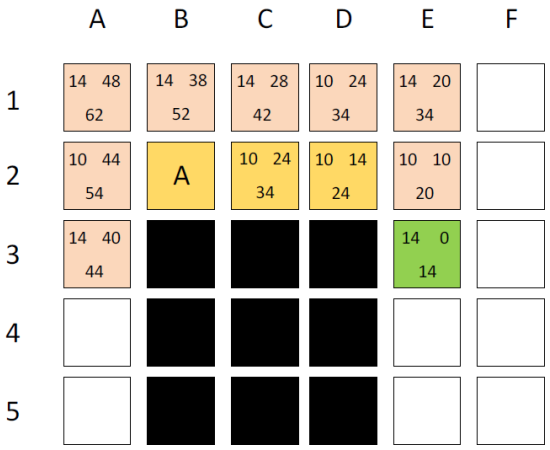
Gambar 2.7. Perhitungan Langkah Kedua

Tabel 2.2 Langkah Kedua Penyelesaian Algoritma A*

Node	Penyelesaian
n(B1)	f(n) = g(n) + h(n) f(B1) = g(B1)+h(B1) = 14+38 = 52
n(C1)	f(n) = g(n) + h(n) f(C1) = g(C1)+h(C1) = 10+28 = 38
n(D1)	f(n) = g(n) + h(n) f(F6) = g(F6)+h(F6) = 14+24 = 38
n(D2)	f(n) = g(n) + H(n) f(F6) = g(F6)+H(F6) = 10+14 = 24

P-ISSN: 2548-9704
E-ISSN: 2686-0880

Berdasarkan perhitungan langkah kedua diatas, maka hasil perhitungan memiliki nilai f terkecil yaitu 24 pada node D2. Node D2 terpilih sebagai BestNode karena mempunyai nilai F terendah. Kemudian dicek apakah node D2 merupakan node tujuan. Ternyata node D2 bukan merupakan node tujuan. Node D2 di masukan ke dalam ClosedList karena sudah menjadi bestnode dan bangkitkan node tetangga dari node D2 yang dapat dilalui dan bukan termasuk ClosedList. Maka didapatkan node tetangganya yaitu node (C1,D1,E1,E2,E3).

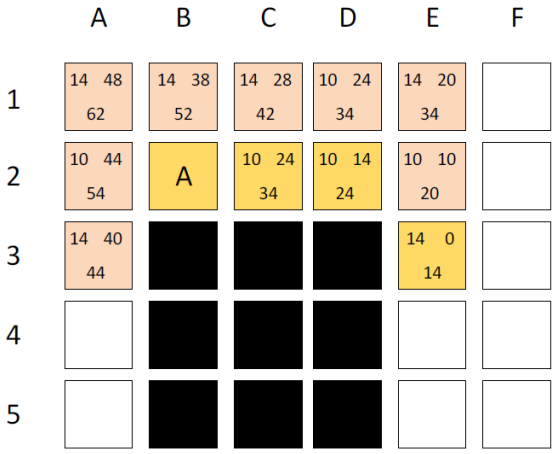


Gambar 2.8. Perhitungan Langkah Ketiga

Tabel 2.3. Langkah Ketiga Penyelesaian Algoritma A*

Node	Penyelesaian
n(C1)	f(n) = g(n) + h(n) f(C1) = g(C1)+h(C1) = 14+28 = 42
n(D1)	f(n) = g(n) + h(n) f(D1) = g(D1)+h(D1) = 10+14 = 24
n(E1)	f(n) = g(n) + h(n) f(E1) = g(E1)+h(E1) = 14+20 = 34
n(E2)	f(n) = g(n) + h(n) f(E2) = g(E2)+h(E2) = 10+10 = 20
n(E3)	f(n) = g(n) + h(n) f(E3) = g(E3)+h(E3) = 14+0 = 14

Berdasarkan perhitungan langkah ke tiga diatas, maka hasil dari perhitungan yang memiliki nilai F terkecil yaitu F=34 pada node E4.



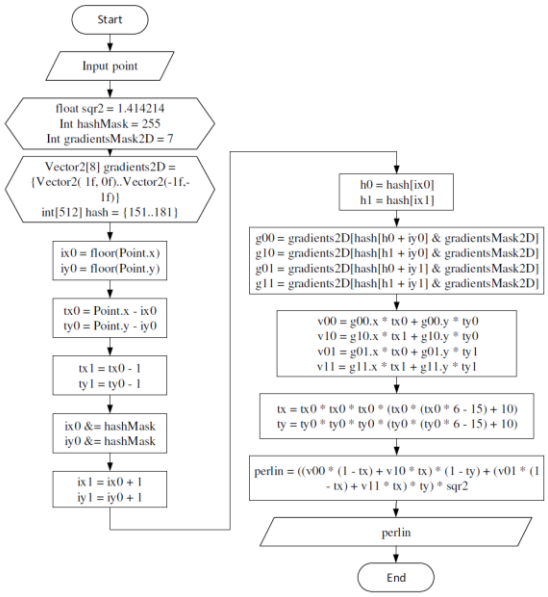
Gambar 2.9. Jalur Terpendek

Karena E3 merupakan node tujuan maka algoritma dihentikan dan dibuat jalur terpendek mulai dari node tujuan hingga node awal (B2, C2, D2, E3).

Perlin Noise adalah algoritma untuk menghasilkan coherent noise yang memenuhi ruang. Coherent noise berarti bahwa untuk setiap dua titik dalam ruang, nilai fungsi noise berubah dengan halus saat bergerak dari satu titik ke titik yang lain - yaitu tidak ada diskontinuitas. Perlin Noise yang termasuk gradient noise yang berarti menggunakan pseudo-random gradien pada titik dalam kisi lalu interpolasi dan haluskan titik [12].

Garis besar algoritma untuk membuat noise sederhana. Masukkan titik inputan, kemudian lihat setiap titik grid disekitar titik inputan. Dalam 1 dimensi akan ada 2 titik grid disekitarnya. Dalam dua dimensi akan ada empat titik grid sekitarnya; dalam tiga dimensi akan ada

delapan. Dalam n dimensi, akan ada 2n titik grid sekitarnya. Lakukan proses smoothing kemudian interpolasikan dan hasilnya akan berupa bilangan riil. Karena untuk 3D terrain generation yang dibutuhkan adalah perlin noise 2D maka yang dibahas hanya perlin noise 2D [12].



Gambar 2.10. Flowchart Algoritma Perlin Noise 2D

Proses pembentukan 3D terrain dimulai dengan menentukan koordinat mana saja yang akan dibuat terrain. Misalnya koordinat (0,0) akan dibuat terrain maka dicari berapa ketinggiannya dengan menggunakan algoritma perlin noise. Berikut langkah – langkahnya :

1. Langkah 1 :
   Inputan berupa vector2 atau bisa disebut point dengan nilai (0,0) dimana axis x bernilai 0 dan axis y bernilai 0.
2. Langkah 2:
   Atur terlebih dahulu nilai di dalam algoritma sebagai berikut:
   float sqr2 = 1.414214
   int hashMask = 255
   int gradientsMask2D = 7
   Inisialisai array gradients2D dengan ukuran 8 dengan elemen {Vector2(1f, 0f),.., Vector2(-1f,-1f)}
   Inisialisai array hash dengan ukuran 512 dengan elemen {151,.., 180}
3. Langkah 3:
   Lalu bulatkan point ke bawah lalu masukkan ke variabel:
   ix0 = floor(point.x) iy0 = floor(point.y)
   Proses:
   ix0 = floor(0) iy0 = floor(0)
   Hasil:
   ix0 = 0 iy0 = 0
4. Langkah 4:
   Hitung:
   tx0 = point.x - ix0 ty0 = point.y - iy0;
   Proses:
   tx0 = 0 – 0 ty0 = 0 – 0
   Hasil:
   tx0 = 0 ty0 = 0
5. Langkah 5:
   Hitung:
   tx1 = tx0 – 1 ty1 = ty0 – 1
   Proses:
   tx1 = 0 – 1 ty1 = 0 – 1
   Hasil:
   tx1 = -1 ty1 = -1
6. Langkah 6:
   Hitung:
   ix0 &= hashMask iy0 &= hashMask
   Proses:
   ix0 &= 255 atau bisa ditulis ix0 = ix0 & 255
   iy0 &= 255 atau bisa ditulis iy0 = iy0 & 255
   ix0 = 0 & 255
   0000 0000 0000 0000 0000 0000 0000 0000
   0000 0000 0000 0000 0000 0000 1111 1111 &
   0000 0000 0000 0000 0000 0000 0000 0000
   iy0 = 0 & 255
   0000 0000 0000 0000 0000 0000 0000 0000
   0000 0000 0000 0000 0000 0000 1111 1111 &
   0000 0000 0000 0000 0000 0000 0000 0000
   Hasil:
   0000 0000 0000 0000 0000 0000 0000 0000 = 0
   ix0 = 0 iy0 = 0

P-ISSN: 2548-9704
E-ISSN: 2686-0880

7. Langkah 7:
   Hitung:
   ix1 = ix0 + 1 iy1 = iy0 + 1
   Proses:
   ix1 = 0 + 1 iy1 = 0 + 1
   Hasil:
   ix1 = 1 iy1 = 1
8. Langkah 8:
   Hitung:
   h0 = hash[ix0] h1 = hash[ix1]
   Proses:
   h0 = hash[0] h1 = hash[1]
   Hasil:
   h0 = 151 h1 = 160
9. Langkah 9:
   Hitung
   g00 = gradients2D[hash[h0 + iy0] & gradientsMask2D];
   g10 = gradients2D[hash[h1 + iy0] & gradientsMask2D];
   g01 = gradients2D[hash[h0 + iy1] & gradientsMask2D];
   g11 = gradients2D[hash[h1 + iy1] & gradientsMask2D];
   Hasil:
   g00 = gradients2D[hash[151+ 0] & 7];
   g10 = gradients2D[hash[160+ 0] & 7];
   g01 = gradients2D[hash[151+ 1] & 7];
   g11 = gradients2D[hash[160+ 1] & 7];
   g00 = gradients2D[hash[151+ 0] & 7]
   g00 = gradients2D[hash[151] & 7]
   g00 = gradients2D[17 & 7]
   17 = 0001 0001
   7 = 0000 0111 &
   0000 0001
   g00 = gradients2D[1]
   g00 = Vector2(-1, 0)
   g10 = gradients2D[hash[160+ 0] & 7];
   g10 = gradients2D[hash[160] & 7]
   g10 = gradients2D[119& 7]
   119 = 0111 0111
   7 = 0000 0111 &
   0000 0111
   g10 = gradients2D[7]
   g10 = Vector2(-1, -1)
   g01 = gradients2D[hash[152] & 7];
   g01 = gradients2D[182 & 7];
   182 = 1011 0110
   7 = 0000 0111 &
   0000 0110
   g01 = gradients2D[6];
   g01 = Vector2(1, -1)
   g11 = gradients2D[hash[161] & 7];
   g11 = gradients2D[248 & 7];
   248 = 11111000
   7 = 0000 0111 &
   0000 0000
   g11 = gradients2D[0];
   g11 = Vector2(1, 0)
   Hasil:
   g00 = Vector2(-1, 0)
   g10 = Vector2(-1, -1)
   g01 = Vector2(1, -1)
   g11 = Vector2(1, 0)
10. Langkah 11:
    Hitung:
    v00 = g00.x * tx0 + g00.y * ty0
    v10 = g10.x * tx1 + g10.y * ty0
    v01 = g01.x * tx0 + g01.y * ty1
    v11 = g11.x * tx1 + g11.y * ty1
    Proses:
    v00 = -1 * 0 + 0 * 0
    v00 = 0 + 0
    v00 = 0
    v10 = -1 * -1+ -1 * 0
    v10 = 1 + 0
    v10 = 1
    v01 = 1 * 0 + -1 * -1
    v01 = 1 + 1
    v01 = 2
    v11 = 1 * -1 + 0 * -1
    v11 = -1 + 0
    v11 = -1
    Hasil:
    v00 = 0 v10 = 1
    v01 = 2 v11 = -1
11. Langkah 12:
    Hitung:
    tx = tx0 * tx0 * tx0 * (tx0 * (tx0 * 6 - 15) + 10)
    ty = ty0 * ty0 * ty0 * (ty0 * (ty0 * 6 - 15) + 10)
    Proses:
    tx = 0 * 0 * 0 * (0 * (0 * 6 - 15) + 10)
    tx = 0
    ty = 0 * 0 * 0 * (0 * (0 * 6 - 15) + 10)
    ty = 0
    Hasil:

P-ISSN: 2548-9704
E-ISSN: 2686-0880

tx = 0 ty = 0

12. Langkah 13:
    Hitung:
    perlin = ((v00 * (1 - tx) + v10 * tx) * (1 - ty) + (v01 * (1 - tx) + v11 * tx) * ty) * sqr2
    Proses:
    perlin = ((0 * (1 - 0) + 1 * 0) * (1 - 0) + (2 * (1 - 0) + -1 * 0) * 0) * 1.4142135623730950
    perlin = ((0 * (1) + 1 * 0) * (1) + (2 * (1) + -1 * 0) * 0) * 1.4142135623730950
    perlin = ((0 + 0) * (1) + (2 + 0) * 0) * 1.4142135623730950
    perlin = ((0) * (1) + (2) * 0) * 1.4142135623730950
    perlin = (0 + 0) * 1.4142135623730950
    perlin = 0 * 1.4142135623730950
    perlin = 0

Nilai perlin noise di koordinat (0,0) adalah 0, maka ketinggian terrain di koordinat (0,0) adalah 0.

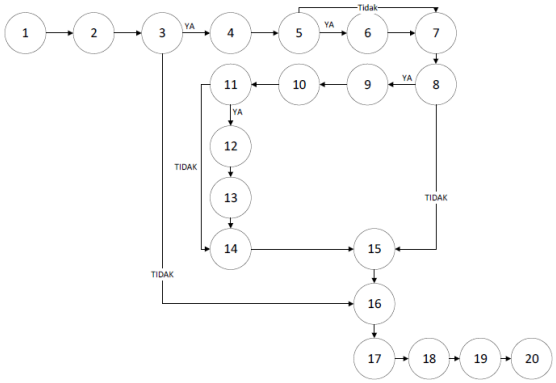
3. HASIL DAN PEMBAHASAN

Setelah melakukan analisis dan perancangan sistem, tahap selanjutnya adalah tahap implementasi. Implementasi merupakan tahap untuk penerapan hasil dari analisis dan perancangan sistem yang telah di dilakukan sebelumnya yang bertujuan agar hasil dari analisis dan perancangan dapat digunakan sesuai dengan kebutuhan.

Pengujian merupakan suatu proses uji coba dan evaluasi game ini, apakah setiap proses pada game tersebut sudah memenuhi syarat atau belum. Jika belum, maka dilakukan analisis sistem kembali. Terdapat dua pengujian untuk menguji game ini, yaitu pengujian kotak hitam (Black Box) dan pengujian kotak putih (White Box).

Adapun metode yang digunakan dalam pengujian white box ini adalah metode basis path. Metode basis path mengijinkan pendesain kasus uji untuk mendapatkan perkiraan logic yang kompleks dari desain prosedural dan menggunakan perkiraan ini untuk mendefenisikan aliran eksekusi adalah sebagai berikut:

1. V(G) = E – N + 2 hasilnya sama dengan V(G) = P + 1
2. Flowgraph mempunyai region yang sama dengan jumlah V(G) maka sistem dikatakan sudah terbukti efektif dan efisien.



Gambar 3.1. Flow Graph Alogritma A*

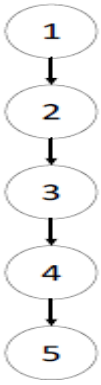
Keterangan Flow Graph :
Node (N) = 20
Edge (E) = 23
Predicate Node (P) = 4

1. Perhitungan
   V(G) = E – N + 2
   V(G) = 23 – 20 + 2 = 5
   V(G) = P + 1
   V(G) = 4 + 1 = 5

Berdasarkan hasil perhitungan cyclomatic complexity terdapat lima independent path (jalur) yaitu:

Path 1 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9- 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20
Path 2 : 1 – 2 – 3 – 16 – 17 – 18 – 19 – 20
Path 3 : 1 – 2 – 3 – 4 – 5 – 7 – 8 – 9- 10 – 11 – 12 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 20
Path 4 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 15 – 16 – 17 – 18 – 19 – 20
Path 5 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9- 10 – 11 – 14 – 15 – 16 – 17 – 18 – 19 – 20

P-ISSN: 2548-9704
E-ISSN: 2686-0880



Gambar 3.2. Flow Graph Perlin Noise

Keterangan Flow Graph :
Node (N) = 5
Edge (E) = 4
Predicate Node (P) = 0

Perhitungan :
V(G) = E – N + 2
V(G) = 4 – 5 + 2 = 1
V(G) = P + 1
V(G) = 0 + 1 = 1

Berdasarkan hasil perhitungan cyclomatic complexity terdapat satu independent path (jalur) yaitu:
Path 1: 1 – 2 – 3 – 4 – 5

4. KESIMPULAN

Kesimpulan yang dapat diambil dari hasil pengujian adalah sebagai berikut :

1. Implementasi algoritma A* berhasil membuat musuh menemukan jalur terpendek menuju karakter pemain untuk kemudian mengejar dan menyerang.
2. 3D terrain generator berjalan sesuai dengan harapan menggunakan algoritma perlin noise, dimana terrain dihasilkan dari baris-baris script tanpa proses pembuatan secara manual.
3. Penggabungan antara game RPG dengan game kuis berjalan dengan baik dimana game RPG tidak dapat diselesaikan tanpa menyelesaikan game kuis terlebih dahulu.

5. SARAN

Adapun saran dari penelitian ini adalah sebagai berikut :

1. Game akan lebih menyenangkan jika dibuat online dengan genre MMORPG (Massively Multiplayer Online Role Playing Games) sehingga dapat dimainkan bersama-sama dengan teman.
2. Pemain dapat melakukan kustomisasi karakter sehingga dapat membuat karakter sendiri yang unik. Sedangkan untuk model karakter akan lebih baik jika menggunakan model berbentuk manusia sehingga lebih banyak variasi animasinya.
3. Dunia di game akan lebih menarik jika dibuat open-world yang luas tanpa dibatasi penghalang. Serta supaya pemain tidak cepat bosan ditambahkannya variasi musuh yang lebih pintar dan tipe pemainan yang lebih beragam.
4. Supaya game tidak terkesan cepat selesai quest dapat diperbanyak dan cerita dibuat nonlinear atau mempunyai beberapa akhir.

DAFTAR PUSTAKA

[1] A. G. I. Hutabarat and A. C. Padmasari, “Rancang Bangun Game Tradisional ‘Tambah Satu’ berbasis Platform Android,” *J. Pendidik. Multimed. Edsence*, vol. 2, no. 1, pp. 29–44, 2020.

[2] P. C. Joni, “Analisis Kualitas Software Pada Pembangunan Mobile Game RPG Berdasarkan Kebutuhan Kualitas Untuk Mobile Game,” *IT J. Res. Dev.*, vol. 3, no. 1, pp. 62–71, 2018.

[3] N. Ratama, M. Kom, and M. Munawaroh, *Konsep kecerdasan buatan dengan pemahaman logika fuzzy dan penerapan aplikasi*. Uwais Inspirasi Indonesia, 2019.

[4] F. H. Selian and R. N. Rambe, “Pengaruh Penggunaan Aplikasi Quizizz dalam Meningkatkan Hasil Belajar Siswa Madrasah Ibtidaiyah

di Masa Pandemi," *Edukatif J. Ilmu Pendidik.*, vol. 4, no. 6, pp. 7370–7377, 2022.

[5] R. N. B. Sitepua and I. G. N. A. C. Putraa, "Penentuan Rute Terpendek Menggunakan Algoritma A Star (Studi Kasus: Distributor Barang)."

[6] A. Damayanti, "Distribusi tekstur kepadatan kabut pada simulasi 2D kabut heterogen menggunakan perlin noise." Universitas Islam Negeri Maulana Malik Ibrahim, 2019.

[7] R. Apriansyah and N. Hadinata, "Penerapan Customer Relationship Management (CRM) Berbasis Web pada Sistem Informasi Penjualan Arofa Moslem Wear Menggunakan Metode Rational Unified Process (RUP)," in *Bina Darma Conference on Computer Science (BDCCS)*, 2019, vol. 1, no. 3, pp. 806–811.

[8] D. A. Sukma, "Pengembangan sistem informasi monitoring tugas akhir (MONITA) jurusan ilmu komputer fakultas matematika dan ilmu pengetahuan alam universitas lampung," 2019.

[9] A. Widyanto, "Penerapan Metode RUP pada Sistem Informasi Unit Kegiatan Mahasiswa STMIK PalComTech," *J. SISFOKOM (Sistem Inf. dan Komputer)*, vol. 9, no. 3, pp. 323–331, 2020.

[10] R. Firmansyah, "Aplikasi Reservasi Arung Jeram Dengan Metode Rational Unified Process (Studi Kasus Magelang Explore)." Universitas Muhammadiyah Magelang, 2021.

[11] N. Purwati, H. Halimah, and A. Rahardi, "Perancangan Website Program Studi Sistem Informasi Institut Informatika Dan Bisnis Darmajaya Bandar Lampung," *J. SIMADA (Sistem Inf. dan Manaj. Basis Data)*, vol. 1, no. 1, pp. 71–80, 2018.

[12] A. Jain, A. Sharma, and Rajan, "Adaptive & Multi-Resolution Procedural Infinite Terrain Generation with Diffusion Models and Perlin Noise," in *Proceedings of the Thirteenth Indian Conference on Computer Vision, Graphics and Image Processing*, 2022, pp. 1–9.

[13] A. Rohmanu, "Rancang Bangun Sistem Monitoring Kerusakan Mesin Produksi Berbasis Mikrokontroler Arduino Uno Di Pt. Nakakin Indonesia," *J. Inform. SIMANTIK*, vol. 7, no. 1, pp. 6–11, 2022.

[14] S. J. Sinaga, "Desain Pengendali Sensor Jarak Pada Robot Mobil Dengan Penghalang Tidak Diketahui," 2021.