Versi Online:
https://journal.ubm.ac.id/index.php/alu
DOI: http://dx.doi.org/10.30813/j-alu.v2i1.1571

Jurnal Algoritma, Logika dan Komputasi
Vol.II (No. 1) : 111 - 120. Th. 2019
p-ISSN: 2620-620X
e-ISSN: 2621-9840

IMPLEMENTASI ALGORITMA A-STAR PADA PERMAINAN KOMPUTER ROGUELIKE BERBASIS UNITY

A-Star Algorithm Implementation On Roguelike Computer Game Based On Unity

Hendi Hermawan, hendi.hermawan@upj.ac.id[1)], Hari Setiyani, hari.setiyani@gmail.com[2)]

[1)]Informatika, Universitas Pembangunan Jaya, Pembangunan Jaya Center for Urban Studies (PJ-CUS)
[2)]Teknik Informatika, Sekolat Tinggi Teknologi Informasi NIIT I-Tech

ABSTRACT

There are many genres found in video games, one of which is role-playing games (RPGs). Roguelike was originally a type of RPG subgenre whose game was based on a risk-aware pattern. The Roguelike game is synonymous with games that require players to play efficiently in order to minimize the risk of losing game progress. This research is in the form of making a Roguelike game by implementing the A-star algorithm to search for the shortest path in the process of chasing enemies against player characters. Overall, this final assignment research is carried out by applying a prototype system development model. The results of this study are able to prove that using the A-star algorithm is one of the right methods for the shortest search in the process of chasing enemies against player characters.
***Keywords**: A-star, Roguelike, unity*

ABSTRAK

Ada banyak genre yang terdapat di *video game* yang salah satunya adalah *role-playing game* (RPG). Roguelike awalnya merupakan jenis subgenre RPG yang permainannya didasari pola mewaspadai risiko. Permainan Roguelike identik dengan permainan yang mengharuskan pemainnya dapat bermain secara efisien demi meminimlisir resiko hilangnya progres permainan. Penelitian ini berupa pembuatan permainan roguelike dengan mengimplementasikan algoritma A-star untuk pencarian jalur terpendek pada proses pengejaran musuh terhadap karakter pemain. Secara keseluruhan penelitian ini dilakukan dengan menerapkan model pengembangan sistem prototype. Hasil pada penelitian ini adalah dapat membuktikan bahwa dengan menggunakan algoritma A-star merupakan salah satu metode yang tepat untuk pencarian terpendek dalam proses pengejaran musuh terhadap karakter pemain.
Kata Kunci: *A-star, roguelike, unity*

PENDAHULUAN

Ada banyak *genre* yang terdapat di video game yang salah satunya adalah *role-playing game* (RPG). Roguelike awalnya merupakan jenis *subgenre* RPG yang permainannya didasari pola mewaspadai risiko. Permainan Roguelike identik dengan permainan yang mengharuskan pemainnya dapat bermain secara efisien demi meminimlisir resiko hilangnya progres permainan. Yang dimaksud dengan kata efisien disini dapat diartikan seperti pemahaman sistematika permainan, pengambilan langkah yang tepat agar permainan dapat terus berlanjut dengan kondisi yang telah ditentukan, dan sebagainya.

Tingkat kesulitan pada permainan ini terletak pada bagaimana pemain mengatur permainan dengan langkah yang tepat sehingga dapat meminimalisir resiko hilangnya progres permainan dan permainan dapat terus berlanjut. Hat tersebut bukanlah hal yang mudah mengingat permainan ini biasanya

Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840

menerapkan kondisi yang menjadi pedoman permainan. Kondisi sangat berkaitan dengan kelangsungan permainan. Hal tersebut menjadi tantangan tersendiri bagi pemain yang ingin memainkan permainan jenis ini.

Pada penelitian ini akan dibuat sebuah permainan dengan jenis roguelike dengan menerapkan konsep yang sama seperti game jenis roguelike pada umumnya. Pada prosesnya, dalam permainan yang akan dibangun ini terdapat musuh yang akan mengejar karakter pemain agar tidak dapat menuju goal. Pemain harus menggerakkan karakter menjauh dari musuh. Musuh akan bergerak ke arah karakter seiring dengan jalannya karakter.

Pada game caveman bergenre roguelike membutuhkan algoritma yang dapat menunjang terbentuknya aplikasi sesuai dengan kebutuhan permainan. Salah satu kebutuhan dari permainan ini adalah proses pencarian jalur terpendek untuk musuh dalam melakukan pengejaran terhadap karakter pemain. Dalam memenuhi kebutuhan tersebut, dapat memanfaatkan ilmu kecerdasan buatan (AI) dengan menggunakan salah satu metode Best First Search (BFS) yaitu pencarian dengan algoritma A-star (A*).

Algoritma A-star biasa digunakan dalam beberapa penelitian untuk mencari jalur terpendek pada satu area tertentu. Dengan dilakukannya observasi, kemungkinan algoritma A-star dapat diterapkan sebagai proses pencarian jalur terpendek pada permainan roguelike. Penerapan algoritma A-star dapat digunakan untuk memenuhi kebutuhan proses pengejaran musuh terhadap karakter pemain. Hal tersebut menarik untuk dilakukan penelitian lebih lanjut yaitu menerapkan algoritma A-star untuk pencarian jalur terpendek musuh dalam mengejar karakter pemain tersebut.

METODE PENELITIAN

Algoritma A-*star*

Algoritma A-star (A*) adalah algoritma yang dikemukakan oleh Hart, Nilson, dan Raphael pada tahun 1968. Algoritma A* merupakan salah satu algoritma Branch & Bound atau disebut juga sebagai sebuah algoritma untuk melakukan pencarian solusi dengan menggunakan informasi tambahan (heuristik) dalam menghasilkan solusi yang optimal. [7]

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah starting point, current node, simpul, neighbor node, open set, closed set, came from, harga (cost), walkability, target point. Starting point adalah sebuah terminologi untuk posisi awal sebuah benda. Current node 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. Neighbor node adalah simpul-simpul yang bertetangga dengan current node. Open set adalah tempat menyimpan data simpul yang mungkin diakses dari starting point maupun simpul yang sedang dijalankan. Closed set adalah tempat menyimpan data simpul sebelum current node yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan. Came from adalah tempat menyimpan data ketetanggaan dari suatu simpul, misalnya y came from x artinya neighbor node y dari current node x. Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari starting point ke current node, dan H, jumlah nilai perkiraan dari sebuah simpul ke target point. Target point yaitu simpul yang dituju. Walkability adalah sebuah atribut yang menyatakan apakah sebuah simpul dapat atau tidak dapat dilalui oleh current node.

Algoritma A* menerapkan teknik heuristik dalam membantu penyelesaian persoalan. Heuristik adalah penilai yang memberi harga pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik yang benar, maka A* pasti akan mendapatkan solusi (jika ada) yang dicari. Dengan kata lain, heuristik adalah fungsi optimasi yang

Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840

menjadikan algoritma A* lebih baik dari pada algoritma lainnya. Namun heuristik masih merupakan estimasi / perkiraan biasa saja Sama sekali tidak ada rumus khususnya. Artinya, setiap kasus memiliki fungsi heuristik yang berbedabeda. Algoritma A* ini bisa dikatakan mirip dengan algoritma Dijkstra, namun pada algoritma Dijkstra, nilai fungsi heuristiknya selalu 0 (nol) sehingga tidak ada fungsi yang mempermudah pencarian solusinya.

Nilai ongkos pada setiap simpul n menyatakan taksiran ongkos termurah lintasan dari simpul n ke simpul target (target node), yaitu:

F (n) = nilai taksiran lintasan termurah dari simpul status n ke status tujuan

Dengan kata lain, F (n) menyatakan batas bawah (lower bound) dari ongkos pencarian solusi dari status n. Fungsi heuristik yang terdapat pada algoritma A* untuk menghitung taksiran nilai dari suatu simpul dengan simpul yang telah dilalui seperti pada rumus (1)

$$F(n) = G(n) + H(n) \quad \text{.....(1)}$$

Dimana :

F (n) = ongkos untuk simpul n

G (n) = ongkos mencapai simpul n dari akar

H (n) = ongkos mencapai simpul tujuan dari simpul n

Algortima A* secara ringkas langkah demi langkahnya adalah sebagai berikut.

1. Tambahkan starting point ke dalam openset.
2. Ulangi langkah berikut:
   a. Carilah biaya F terendah pada setiap simpul dalam open set. Node dengan biaya F terendah kemudian disebut current node.
   b. Masukkan ke dalam closed set.
      i. Untuk setiap 8 simpul (neighbor node) yang berdekatan dengan current node:
      ii. Jika tidak walkable atau jika termasuk closed set, maka abaikan.
      iii. Jika tidak ada pada open set, tambahkan ke open set.
   c. Jika sudah ada pada open set, periksa apakah ini jalan dari simpul ini ke current node yang lebih baik dengan menggunakan biaya G sebagai ukurannya. Simpul dengan biaya G yang lebih rendah berarti bahwa ini adalah jalan yang lebihbaik. Jika demikian, buatlah simpul ini (neighbor node) sebagai came from dari current node, dan menghitung ulang nilai G dan F dari simpul ini
   d. Stop ketika Anda:
      i. menambahkan target point ke dalam closed set, dalam hal ini jalan telah ditemukan, atau
      ii. Gagal untuk menemukan target point, dan open set kosong. Dalam kasus ini, tidak ada jalan.
3. Simpan jalan. Bekerja mundur dari target point, pergi dari masing-masing simpul ke simpul came from sampai mencapai starting point. Itu adalah jalan Anda.

Kompleksitas waktu dari A * tergantung pada heuristiknya. Dalam kasus terburuk, jumlah simpul yang diperluas adalah eksponensial dari panjang solusi (jalur terpendek), tetapi polinomial ketika ruang pencarian adalah pohon, ada sebuah simpul tujuan tunggal, dan fungsi heuristik h memenuhi kondisi berikut:

$$| H (x) - h * (x) | = O (\log h * (x))$$

dimana h* adalah heuristik optimal, biaya yang tepat yang didapat dari x ke tujuan. Dengan kata lain, kesalahan dari h tidak akan tumbuh lebih cepat dari logaritma dari "heuristik sempurna" h * yang mengembalikan jarak yang sebenarnya dari x ke tujuan.

Penerapan algoritma A-star (A*) pada permainan ini terletak pada proses penggerakan musuh dari karakter yaitu monster, dimana musuh akan bergerak mendekati karakter pemain yang posisinya akan berubah-ubah secara terus- menerus selama permainan. Pengejaran karakter akan berhenti jika musuh berhasil sampai ke tempat karakter pemain dan ditandai dengan berakhirnya permainan.

Prinsip algoritma A-star adalah mencari jalur terpendek dari sebuah simpul awal (starting point) menuju simpul akhir

Versi Online:
http://journal.ubm.ac.id/jalu
DOI: http://dx.doi.org/10.30813/j-alu.v2i1.1571

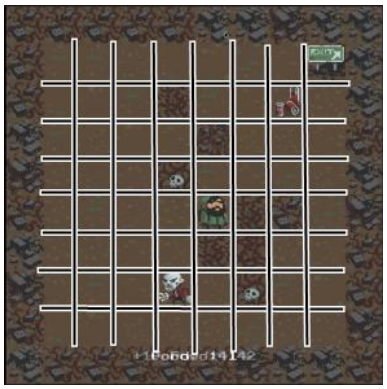
Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840

(target point) dengan memperhatikan nilai f(n) terkecil dimana f(n) adalah nilai (biaya) lintasan terkecil antara simpul awal dan simpul tujuan. Dimana berlaku fungsi seperti rumus (2)

$$F(n) = G(n) + H(n) \quad \text{.....} (2)$$

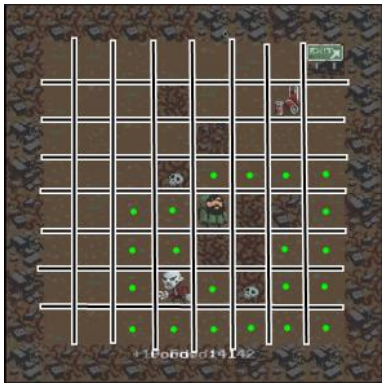
Dimana:
F (n) = biaya untuk simpul n
G (n) = biaya mencapai simpul n dari akar
H (n) = biaya mencapai simpul tujuan dari simpul n

Untuk mencapai titik tujuan(karakter), musuh yang berada di posisi awal akan membandingkan jarak dari masing-masing jalur yang tersedia. Jalur tersebut mengandung nilai yang jika dijumlahkan akan menghasilkan nilai akhir (biaya total) untuk sampai ke titik tujuan.

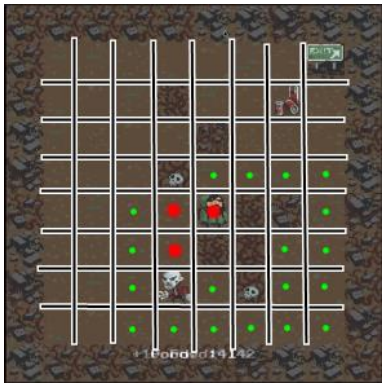


Gambar 1. Pemetaan area permainan

Gambar 1 merupakan contoh pemetaan pada permainan yang akan dibuat. Dapat diasumsikan bahwa musuh berada di simpul awal dan karakter merupakan simpul tujuan. Setiap kotak mengandung nilai G(n) dan H(n). Untuk sampai ketujuan, langkah awal adalah membandingkan nilai yang ada pada setiap kotak dengan menjumlahkan nilai tersebut.
Dari gambar yang telah dipetakan diawal maka dapat diketahui bahwa setiap kotak memiliki nilai G(n) dan H(n). Setelah dilakukan perhitungan maka dilakukan perbandingan antara simpul simpul yang memiliki kemungkinan sebagai jalur terpendek untuk sampai ke simpul akhir.



Gambar 2. Proses Pencarian Path



Gambar 3. Path Solusi

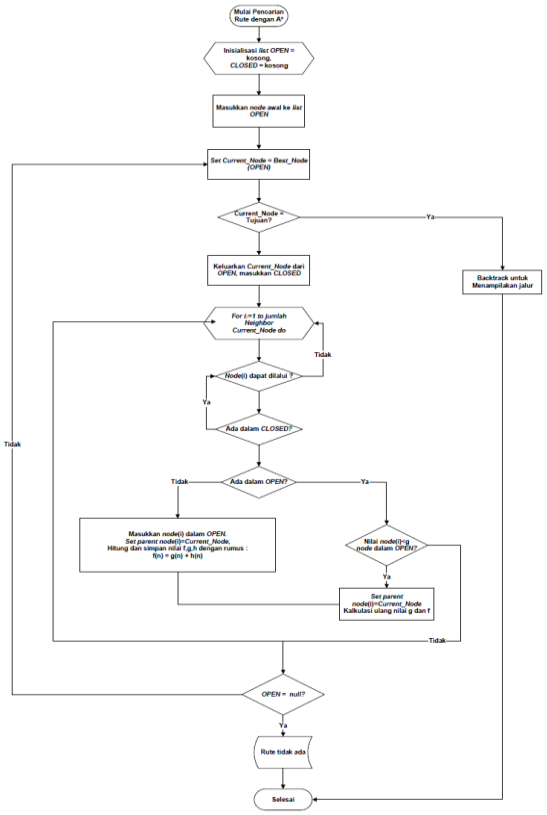
Dari perhitungan yang telah dilakukan pada setiap path maka dapat diketahui nilai dari masing-masing jalur yang memungkinkan tersebut. jalur dengan nilai F(n) terkecil merupakan solusi untuk pencarian jalur terpendek dari simpul awal (starting point) ke simpul akhir (Target point) seperti yang digambarkan dengan titik merah pada gambar 3.

Flowchart

Diagram ini bisa memberi solusi selangkah demi selangkah untuk penyelesaian masalah yang ada di dalam proses atau algoritma. Berikut adalah flowchart untuk proses perhitungan algoritma A-star.

Versi Online:
http://journal.ubm.ac.id/jalu
DOI: http://dx.doi.org/10.30813/j-alu.v2i1.1571

Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840



Gambar 4. flowchart proses perhitungan algoritma A-star

Analisis Sistem Berjalan

Sistem game caveman bergenre roguelike pada dasarnya menerapkan metode pengacakan biasa. Pada permainan asalnya, musuh bekerja dengan cara menghalang karakter pemain untuk sampai ke lokasi finish. Musuh akan bergerak satu langkah seiring dengan pergerakan karakter pemain. Dan jika musuh terjebak pada kondisi terhalang oleh batu maka musuh tidak dapat bergerak menuju karakter pemain

Analisis Sistem Direncanakan

Untuk permainan yang akan dibuat, pada dasarnya akan menggunakan sistem yang sama dari segi pengacakan. Hanya saja untuk cara kerja musuh akan diubah. Jika pada sistem yang sebelumnya cara kerja musuh yaitu dengan menghalangi langkah karakter pemain, maka pada permainan yang akan dibuat kali ini musuh akan bekerja dengan cara melakukan pengejaran terhadap karakter pemain. Proses pengejaran tersebut yang nantinya akan digunakan penerapan algoritma A-star sebagai metode pencarian jalur terpendek dari posisi musuh terhadap karakter pemain

Deskripsi Sistem

Permainan ini menerapkan sistem endless game. Pemain dituntut untuk terus memainkan permainan yang telah di mulai sampai menemukan suatu faktor yang membuat kondisi selesai.Salah satu faktor yang dapat membuat permainan selesai adalah jika karakter pemain terkena musuh. Musuh pada permainan ini akan terus mengejar ke arah karakter pemain. Proses pengejaran musuh terhadap karakter akan menggunakan penerapan algoritma A-star yaitu mencari jalur terpendek untuk sampai pada lokasi karakter.

Permainan akan selalu diawali pada level pertama dengan layout sederhana tanpa musuh. Pada level tersebut pemain dapat mengarahkan karakter ke atas dengan tombol up, kebawah dengan tombol down, ke kiri dengan tombol left, dan ke kanan dengan tombol right. Pemain dapat mengumpulkan poin yang tersedia pada areal permainan untuk mendapatkan skor tertinggi. Selain itu, Pemain harus mengarahkan karakter menuju pintu exit agar dapat naik ke level selanjutnya yang lebih tinggi.

Jika level pertama sudah selesai maka permainan akan secara otomatis menampilkan areal baru untuk level kedua. Pada level kedua ini menerapkan aturan permainan yang sama pada level sebelumnya, hanya saja pada level kedua ini mulai ditampilkan musuh dari karakter sebagai tantangan untuk pemain. Selain harus mengumpulkan poin dan mencapai pintu exit, pemain juga harus menyusun strategi agar dapat menghindarkan karakter dari musuh. Musuh akan bergerak 1 langkah ke arah karakter seiring dengan gerakan karakter. Jika pemain telah selesai pada level tersebut maka permainan akan langsung menampilkan permainan level selanjutnya yang lebih tinggi.

Setiap level permainan akan menampilkan layout yang berbeda

Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840

mengikuti konsep pengacakan random dan akan menambahkan jumlah musuhnya. Musuh dari karakter akan terus bergerak mengarah ke karakter. Dalam kasus ini musuh akan bergerak dengan menerapkan sistem algoritma A-star untuk mencapai goal yaitu mengejar karakter. Semakin tinggi level permainan maka tingkat kesulitannya pun akan semakin tinggi. Karena permainan ini menerapkan sistem endless maka permainan akan terus berlangsung selama pemain dapat menghindarkan karakter dari musuh dan menuju pintu exit.

Parameter Keberhasilan

Pemain dapat mengarahkan karakter menggunakan tombol up, down, left, dan right pada keyboard. Untuk mencapai goal pemain harus mengarahkan karakter ke pintu exit pada area yang tersedia. Selain mencapai goal di pintu exit, pemain dikatakan berhasil jika dapat mengumpulkan skor setinggi-tingginya dengan cara mengambil poin yang telah tersedia. Pemain dapat memenangkan permainan dengan mencapai level setinggi-tingginya dengan skor setinggi-tingginya.

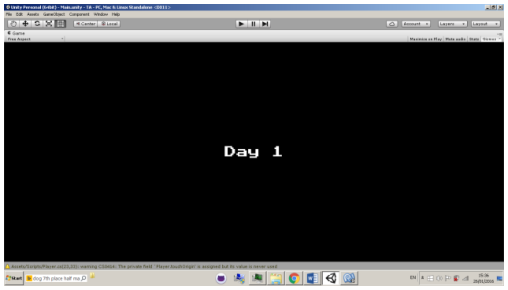
Dalam mencapai goal, pemain juga harus dapat menghindarkan karakter dari musuh yang akan terus mengejarnya. Pengejaran yang dilakukan oleh musuh tersebut menerapkan algoritma A-star dimana target point nya adalah karakter pemain. Penerapan algoritma A-star dapat dikatakan berhasil jika musuh dapat terus mengejar ke arah lawan sebelum lawan sampai ke pintu exit dengan melalui jalur terpendek untuk sampai ke karakter pemain.

HASIL DAN PEMBAHASAN

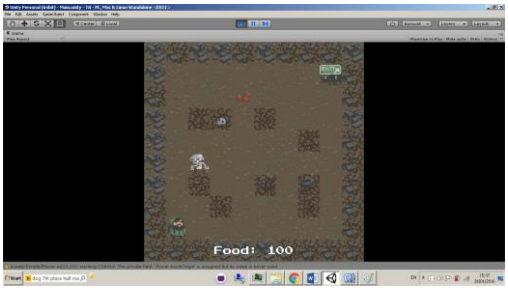
Implementasi

Dari hasil perancangan dan pembuatan permainan ini, terdapat beberapa form tampilan dari permainan tersebut. Diantaranya adalah sebagai berikut.

Permainan akan diawali dengan tampilan sesuai dengan gambar 5 sebagai tampilan utama pada saat permainan dibuka.

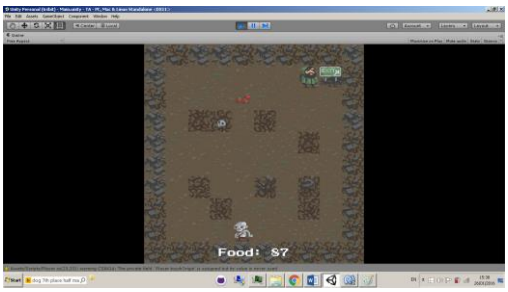


Gambar 5. Form Awal

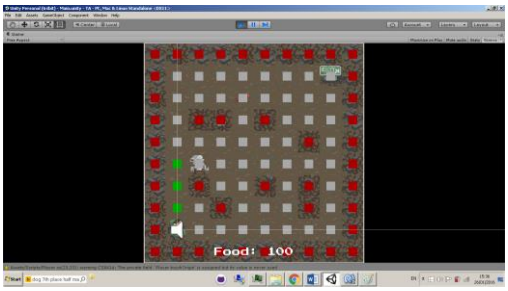


Gambar 5. Form Awal Permainan

Kondisi awal permainan terlihat seperti pada gambar 5 dimana pemain berada di ujung arena permainan.



Gambar 6. Form Jalannya Permainan

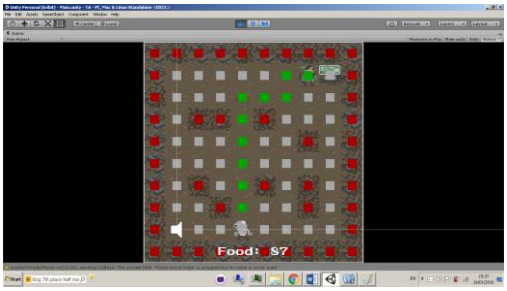


Gambar 6. Path Menuju target point

Versi Online:
http://journal.ubm.ac.id/jalu
DOI: http://dx.doi.org/10.30813/j-alu.v2i1.1571

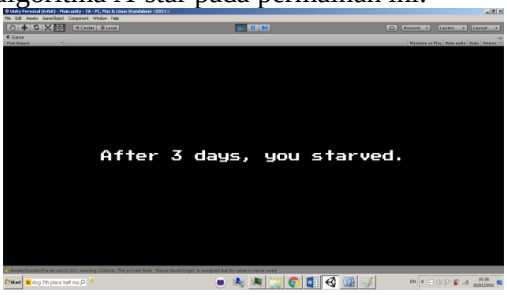
Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840

Dalam proses permainan karakter harus menuju pintu exit. Dimana dalam proses mencapai pintu exit pemain harus mengkalkulasikan langkah yang efisien agar tidak kehabisan poin, dan pemain harus menghindarkan karakter dari musuh yang mengejarnya.



Gambar 7. Path Menuju Target Point (2)

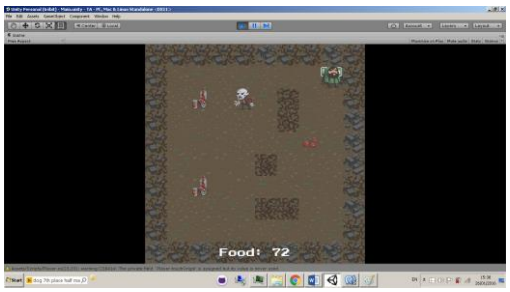
Gambar tersebut merupakan path (jalur) yang dapat ditempuh oleh musuh dalam mengejar target pont (karakter). Jalir tersebut dihasilkan dari penerapan algoritma A-star pada permainan ini.



Gambar 8. Form Kondisi Kalah

Jika karakter terkena oleh musuh, atau karakter kehabisan poin sebelum sampai pintu exit maka akan tampil sebuah form seperti pada gambar 6 dimana permainan akan berhenti dan pemain harus mengulang dari level awal jika ingin kembali bermain.

Gambar 9 menunjukan posisi karakter jika sudah mencapai pintu exit. Jika karakter telah selesai pada level tersebut maka akan muncul level selanjutnya dengan proses permainan dengan konsep yang sama.



Gambar 9. Form Finish

Pengujian Blackbox

Pengujian black box adalah pengujian yang memastikan input dan output menampilkan hasil yang sesuai dengan persyaratan. Juga menguji performa program atau fungsi – fungsi yang tidak bekerja dengan benar. Terdapat dua pengujian yang dilakukan dengan pengujian black box yakni pengujian fungsional dan pengujian sistem.

Tabel 1. Pengujian Tombol Up

Data Input	Harapan	Pengamatan	Hasil
Fungsi tombol up	Karakter bergeser ke atas	Karakter dapat bergeser ke atas	Berhasil

Tabel 2. Pengujian Tombol Down

Data Input	Harapan	Pengamatan	Hasil
Fungsi tombol down	Karakter bergeser ke bawah	Karakter dapat bergeser ke bawah	Berhasil

Tabel 3. Pengujian Tombol Left

Data Input	Harapan	Pengamatan	Hasil
Fungsi tombol left	Karakter bergeser ke kiri	Karakter dapat bergeser ke kanan	Berhasil

Versi Online:
http://journal.ubm.ac.id/jalu
DOI: http://dx.doi.org/10.30813/j-alu.v2i1.1571

Jurnal Algoritma, Logika dan Komputasi
Vol.I (No. 2) : 7 - 16. Th. 2018
p-ISSN: 2620-620X
e-ISSN: 2621-9840

Tabel 4. Pengujian Tombol Right

Data Input	Harapan	Pengamatan	Hasil
Fungsi tombol right	Karakter bergeser ke kanan	Karakter dapat bergeser ke kanan	Berhasil

Tabel 5. Pengujian terkena musuh

Data Input	**Harapan**	**Pengamatan**	**Hasil**
Kondisi terkena musuh	Permainan berakhir dan kembali ke level awal	Permainan berakhir dan kembali ke level awal	Berhasil

Tabel 6. Pengujian finish

Gambar	**Fungsi**	**Keterangan**	**Gambar**
Kondisi finish satu level	Masuk ke level selanjutnya	Sistem menampilkan area level yang baru.	Berhasil

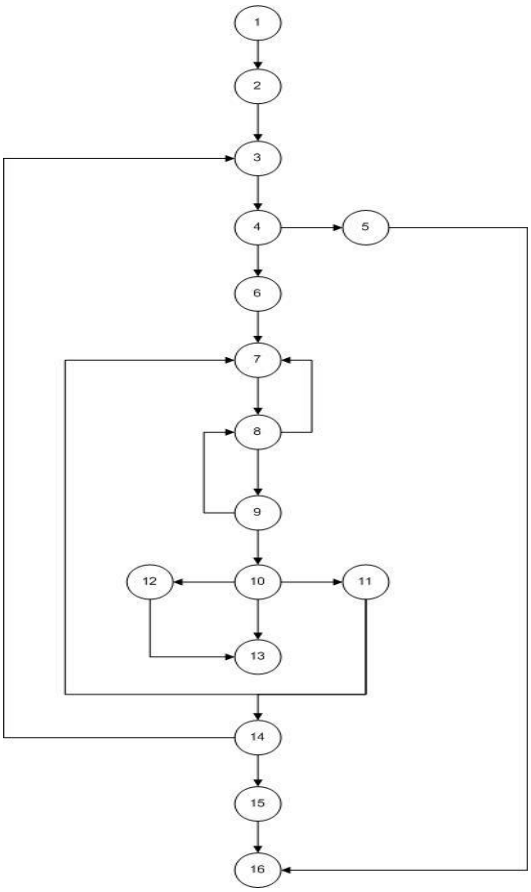
Pengujian Whitebox

Pengujian dengan menggunakan metode *White Box* adalah pengujian yang dilakukan berdasarkan fungsi – fungsi yang ada pada form. Berikut adalah hasil pengujian *white box* pada aplikasi ini.

Algoritma:

1. Inisiasi open & close
2. Masukan node awal ke list open
3. Set Current_node = Best_node (open)
4. Current_node = tujuan?
5. Backtrack untuk menampilkan jalur
6. Keluarkan current_node dari open, masukan closed
7. For i=1 to jumlah neighbor current_node do
8. Node (i) dapat dilalui?
9. Ada dalam closed?
10. Ada dalam open?
11. Nilai node(i) <g node dalam open?
12. Masukan node (i) dalam open. Set parent node(i)= current_node, hitung dan simpan nilai f.g.h dengan rumus: f(n)=g(n)+h(n)
13. Set parent node(i)=Current_node kalkulasi ulang nilai g dan h
14. Open = null?
15. Rute tidak ada
16. Selesai

Berikut adalah tabel fungsi pada pengujian whitebox.



Gambar 10. Flowgraph Algoritma A-star

Tabel 7. Pengujian White Box

Fungsi	**Keterangan**
using UnityEngine; using System.Collections; using System.Collections.Generic; public class Pathfinding : MonoBehaviour { private Transform seeker, target;	Fungsi tersebut digunakan untuk proses pengejaran musuh terhadap karakter menggunakan algoritma A-*star*

```
public List<int> addw;
public GameManager manager;
public Grid grid;
void Awake() {
grid = GetComponent<Grid>();
}
void Start(){
manager =
GameObject.FindObjectOfType
<GameManager>();
}
void Update() {
GameObject Enemy =
GameObject.FindGameObjectW
ithTag("Enemy");
if(Enemy != null &&
manager.doingSetup == false){
FindPath(GameObject.FindGam
eObjectWithTag("Enemy").tran
sform.position + new
Vector3(1,1,0),GameObject.Fin
dGameObjectWithTag("Player"
).transform.position + new
Vector3(1,1,0));
}
}
void FindPath(Vector3 startPos,
Vector3 targetPos) {
Node startNode =
grid.NodeFromWorldPoint(start
Pos);
Node targetNode =
grid.NodeFromWorldPoint(targ
etPos);
List<Node> openSet = new
List<Node>();
HashSet<Node> closedSet =
new HashSet<Node>();
openSet.Add(startNode);
while (openSet.Count > 0) {
Node currentNode = openSet[0];
// Debug.Log(openSet.Count);
for (int i = 1; i < openSet.Count;
i ++) {
if (openSet[i].fCost <
currentNode.fCost ||
openSet[i].fCost ==
currentNode.fCost &&
openSet[i].hCost <
currentNode.hCost) {
currentNode = openSet[i];
}}
// Debug.Log("HAAAA");
openSet.Remove(currentNode);
closedSet.Add(currentNode);
if (currentNode == targetNode)
{
RetracePath(startNode,targetNo
de);
return;
}
// Debug.Log("HAAAA2");
foreach (Node neighbour in
grid.GetNeighbours(currentNod
e)) {
if (!neighbour.walkable ||
closedSet.Contains(neighbour))
{
continue;
}
// Debug.Log("HAAAA3");
int
newMovementCostToNeighbou
r = currentNode.gCost +
GetDistance(currentNode,
neighbour);
if
(newMovementCostToNeighbo
ur < neighbour.gCost ||
!openSet.Contains(neighbour)) {
neighbour.gCost =
newMovementCostToNeighbou
r;
neighbour.hCost =
GetDistance(neighbour,
targetNode);
neighbour.parent = currentNode;
if
(!openSet.Contains(neighbour))
openSet.Add(neighbour);}}}}
void RetracePath(Node
startNode, Node endNode) {
List<Node> path = new
List<Node>();
Node currentNode = endNode;
while (currentNode !=
startNode) {
path.Add(currentNode);
currentNode =
currentNode.parent;}
path.Reverse();
grid.path = path;
int isX = grid.path[0].gridX + 1;
int isY = grid.path[0].gridY + 1;
//Debug.Log(isX + " " + isY);}
int GetDistance(Node nodeA,
Node nodeB) {
int dstX =
Mathf.Abs(nodeA.gridX -
nodeB.gridX);
int dstY =
Mathf.Abs(nodeA.gridY -
nodeB.gridY);
if (dstX > dstY)
// Debug.Log("X" + 14*dstY +
10* (dstX-dstY));
//   Debug.Log("Y" + 14*dstX +
10* (dstY-dstX));
return 14*dstY + 10* (dstX-
dstY);
return 14*dstX + 10 * (dstY-
dstX);}}
```

(A*).

Evaluasi Hasil Pengujian

Bedasarkan pengujian yang telah dilakukan pada aplikasi permainan tersebut, dapat diperoleh hasil sebagai berikut:

1. Secara keseluruhan permainan ini dapat bekerja dengan baik dan hasil yang diperoleh dari pengujian yang telah dilakukan sesuai dengan yang diharapkan.
2. Fungsi yang digunakan untuk membuat musuh mengejar karakter dengan algoritma A--star dapat berjalan dengan baik.

KESIMPULAN

Dari hasil pelaksanaan penelitian ini, dapat disimpulkan mengenai beberapa hal sebagai berikut:

1. Permainan roguelike dapat menerapkan algoritma A-star untuk proses pengejaran musuh terhadap karakter.
2. Dengan digunakannya algoritma A-star, proses pencarian jalur terpendek pada permainan ini dapat menjadi lebih efektif.
3. Permainan ini dapat dimainkan oleh satu pemain dimana untuk mencapai goal tertentu pemain harus dapat menyusun strategi untuk sampai ke finish dan mengumpulkan poin setinggi-tingginya.
4. Permainan ini dapat membantu pemain dalam memahami logika dari sebuah kasus sederhana. Misalnya, bagai mana cara menghindari halangan, bagaimana cara untuk sampai ke goal tertentu, dan sebagainya.

DAFTAR PUSTAKA

[1] Pamungkas, A, et al (2014), Penerapan Algoritma A* (A star) Pada Game Edukasi The Maze Island Berbasis Android. Palembang: STMIK GI MDP.

[2] Hernawan, L, et al (2013) , Penerapan Algoritma A* Pada Aplikasi Puzzle. Palembang: Sekolah Tinggi Teknik Musi.

[3] Musri, et al (2014), Pencarian Jalur Terpendek Pada Snake Game Menggunakan Algoritma A*. Pontianak: Sekolah Tinggi Manajemen Informatika dan Komputer Pontianak.

[4] Tekinbas, K, S, et al (2003) Rules of Play: Game Design Fundamentals, Massachusetts:The MIT Press.

[5] Aldrich, C (2005), learning by Doing: A Comprehensive Guide to Simulations, Computer Games, and Pedagogy in e-Learning and Other Educational Experiences, San Francisco:John Wiley & Sons.

[6] Boehm, B (1986), A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes, ACM, 11(4):14-24.

[7] Tilawah, H (2011), Penerapan Algoritma A-star (A*) Untuk Menyelesaikan Masalah Maze. Bandung: Sekolah Teknik Elektro dan Informatika.

[8] Kristanto, A (2004), Rekayasa Perangkat Lunak (Konsep Dasar), Yogyakarta:Gramedia.

[9] Wolf, M, J, P, et al (2003) , The Video Game Theory. New York*:* Routledge.

[10] Nilwan, A (1998), Pemograman Animasi dan Game Profesional 4, Jakarta: Elex Media Komputindo.

[11] Rollings, et el (2003), Andrew Rollings And Ernest Adams On Game Design, California: New Riders, 2003