Penerapan Algoritma A* Untuk Pergerakan Dinamis NPC Musuh Pada Game Metroidvania

Implementation A Algorithm For Dynamic Enemy Movement in Metroidvania Game*

Adrian Cavallino Pramana Putra
Teknik Informatika, Universitas Dian Nuswantoro
E-mail: 111201912286@mhs.dinus.ac.id

Abstrak

Metroidvania adalah genre pada video game yang merupakan pengembangan dari genre platformer, Genre ini berasal dari penggabungan nama 2 game platformer terkenal yaitu Super Metroid dan Castlevania. Game dengan genre ini mengutamakan eksplorasi dan aksi dalam progress level, dimana pemain diharuskan untuk menjelajahi map berukuran besar dan mencari item atau benda benda yang memberikan upgrade kepada karakter pemain. Selain sistem progresi, NPC musuh pada game Metroidvania cenderung memiliki kompleksitas dalam perilakunya sehingga membuat game semakin menantang, meski begitu banyak juga game Metroidvania yang mengabaikan perilaku musuh tersebut. Maka dari itu pada penelitian ini akan dibuat sebuah algoritma pencarian rute terpendek guna membuat gerakan musuh menjadi semakin dinamis dengan menggunakan algoritma A*.

Kata kunci: AI , Algoritma A*, Metroidvania, Pathfinding

Abstract

Metroidvania is a genre in video games which is the development of the platformer genre, this genre comes from the merging of the names of 2 well-known platformer games namely Super Metroid and Castlevania. Games with this genre prioritize exploration and action in level progress, where players are required to explore large maps and look for items or objects that provide upgrades to the player`s character. In addition to the progression system, enemy NPCs in Metroidvania games tend to have complexity in their behavior, which makes the game more challenging, even though there are also many Metroidvania games that ignore enemy behavior. Therefore, in this study, an algorithm for finding the shortest route will be made to make enemy movements more dynamic using the A algorithm.*

Keywords: A Algorithm, AI, Metroidvania, Pathfinding*

1. PENDAHULUAN

Metroidvania merupakan salah satu diantara banyaknya genre dari video game. Kata Metroidvania berasal dari gabungan 2 nama video game yang cukup terkenal pada masanya di platform NES (*Nintendo Entertaintment System*), yaitu Super Metroid dan Castlevania : Symphony of The Night. Awalnya kedua game ini adalah game platformer, dimana hal yang membedakan antara kedua game ini dengan game platformer biasa adalah sistem progresinya dimana pada kedua game tersebut menawarkan peta yang lebih luas dan melebar secara vertikal dan horizontal [1]. Selain progresinya, yang membedakan game metroidvania dengan game platformer biasa adalah unsur action yang lebih menonjol dari platformer, dimana jika pada platformer pemain hanya diharuskan melewati atau menginjak musuh yang bergerak mondar mandir secara vertikal / horizontal, sedangkan pada metroidvania, karena pemain diberikan variasi serangan yang lebih luas, maka musuh harus dibuat lebih menantang dan bergerak dengan dinamis agar lebih sulit diprediksi.

NPC (*Non-Playable Character*) musuh merupakan salah satu komponen kunci pada game

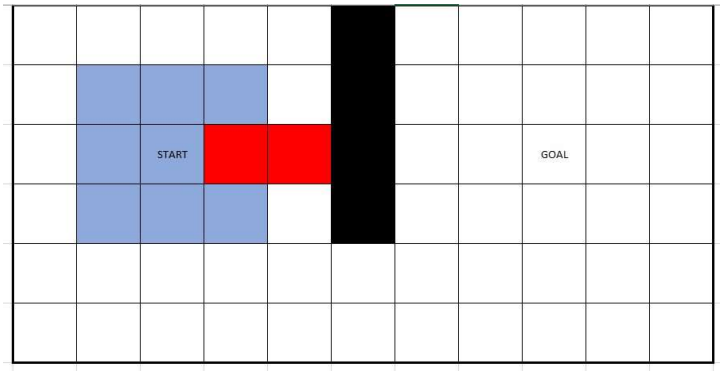
metroidvania, karena perilaku dari NPC dapat menjadi sesuatu yang penting dari dibuatnya sebuah game, contohnya pergerakannya [2]. Pergerakan NPC merupakan bagaimana NPC tersebut melakukan gerakan tertentu pada saat saat tertentu, seperti ketika musuh sedang tidak melakukan apapun atau biasa disebut *Idle* maka musuh hanya akan bergerak kekanan dan kekiri atau keatas dan kebawah, namun ketika pada radius tertentu terdapat pemain yang datang maka musuh akan langsung bergerak mengejar pemain. Karena pada dasarnya game metroidvania adalah game platformer, maka diperlukan mekanik utama pada platformer seperti melompat dan bergerak secara horizontal untuk menghindari rintangan [3]. Hal yang sama dapat dilakukan pada NPC musuh, akan tetapi jika hal tersebut dilakukan pemain akan dengan mudah merasa bosan karena pergerakan musuh yang terkesan monotan dan terlalu mudah. Untuk itu perlu dibuat NPC mudah yang dapat berperilaku selayaknya pemain. Untuk membuat game yang realistis diperlukan sebuah kecerdasan buatan atau *Artificial Intelegence* pada NPC [4]. Dengan dibuatnya sebuah NPC yang realistis dan bertindak selayaknya dikendalikan oleh pemain lain, pemain akan merasakan adanya interaksi dengan objek dalam game sehingga pemain akan lebih menikmati game tersebut [5]. Untuk membuat NPC realistis salah satu cara yang dapat dilakukan adalah membuat NPC yang dapat melakukan pencarian rute, contohnya seperti musuh yang akan pergi ke titik dimana pemain membuat kegaduhan dengan mencari rute yang paling pendek. Untuk kasus tersebut, agar NPC dapat bergerak melalui titik terpendek diperlukan sebuah algoritma pencarian rute terpendek [6]. Dimana algorima ini akan diuji dengan cara mengukur kemampuannya dalam mencari dan membentuk jalur untuk pergerakan musuh [7].

Algoritma A* (A-star) merupakan salah satu dari beberapa algoritma pencarian rute terpendek yang ada. A* merupakan pengembangan dari algoritma BFS (*Best First Search*) dengan pembeda pada A* mempertimbangkan perkiraan jarak dari titik sekarang dengan titik tujuan atau biasa disebut nilai heuristik [8]. Dengan adanya nilai heuristik, pemilihan jalur dapat dilakukan dengan lebih optimal karena jika jalur memiliki nilai heuristik yang semaikin kecil maka besar kemungkinan bahwa jalur tersebut merupakan jalur yang paling pendek untuk mencapai posisi akhir / tujuan. Menurut hasil penelitian yang dilakukan oleh [9], dalam eksekusi perhitungan algoritma A* diperlukan *Open* dan *Closed list*, dimana pada *Open List* akan berisikan node / rute yang mengelilingi node sekarang, sedangkan pada *Closed List* akan berisikan node yang awalnya pada *Open List* dan memiliki jarak paling pendek menuju tujuan diantara node lainnya . Selain A*, algoritma yang sering digunakan untuk pencarian rute terpendek adalah algoritma Dijkstra, namun yang membuat A* lebih baik dari Dijkstra adalah waktu komputasinya [10]. Hal ini dikarenakan dasar dari algoritma Dijkstra adalah algoritma *Bruteforce* dimana algoritma ini akan melakukan mencari semua rute yang memungkinkan untuk mencapai titik tujuan, kemudian menentukan rute mana yang memiliki cost paling rendah, Sedangkan A* menentukan rute terpendek berdasarkan hitungan cost dari setiap node, dimana cost dari tiap node dipengaruhi oleh jarak dari titik awal ke node sekarang, dan node sekarang ke titik tujuan.

2. METODE PENELITIAN

Algoritma yang akan digunkan pada penelitian ini adalah algoritma A* sebagai salah satu algoritma pencarian rute terpendek. Dimana perancangan algoritma A* akan dilakukan secara bertahap dengan tahapan sebagai berikut :

1. Membuat map pada game menjadi grid yang akan digunakan sebagai node yang akan menyimpan atribut yang diperlukan dalam pencarian rute terpendek. Untuk itu akan digunakan fitur Tilemap yang telah disediakan oleh Unity.
2. Mempersiapkan 2 buah list yang akan menjadi open dan closed list, open list akan berfungsi menyimpan node tetangga dari node sekarang sehingga tiap node tetangga tersebut dapat di cek apakah node tersebut merupakan node terbaik atau bukan, jika node tersebut merupakan node tersebut merupakan node terbaik maka node tersebut akan masuk kedalam closed list dan menjadi node sekarang (*current node*).



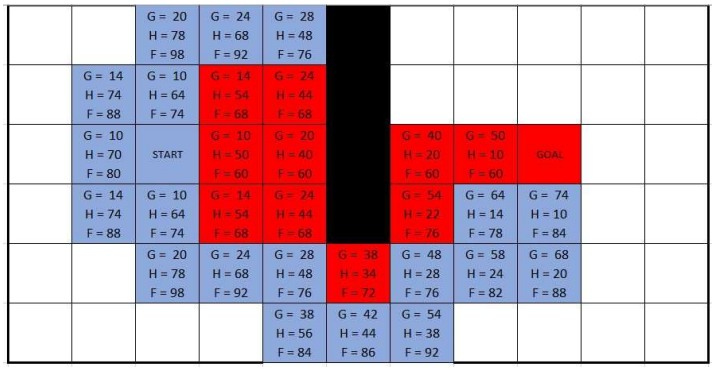
Gambar 1 Map 1

3. Melakukan perhitungan untuk tiap node tetangga dari node sekarang, dimana dengan menggunakan grid, tiap node tersebut akan memiliki 3 atribut, *g cost* sebagai jarak antara titik sekarang dengan titik awal, *h cost* sebagai jarak perkiraan antara titik sekarang dengan titik tujuan, dan *f cost*, dimana *f cost* akan diperoleh dengan menggunakan rumus.

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

Keterangan :
f(x) = cost total pada node ke- x
g(x)= jarak antara titik ke-x ke titik start
h(x)= estimasi jarak antara titik ke- x ke titik tujuan

4. Memastikan bahwa node dengan f cost terkecil merupakan node yang bisa dilewati, dan merupakan sebuah obstacle atau penghalang. Sehingga jika node terbaik merupakan letak obstacle, maka node tersebut tidak masuk ke closed list.



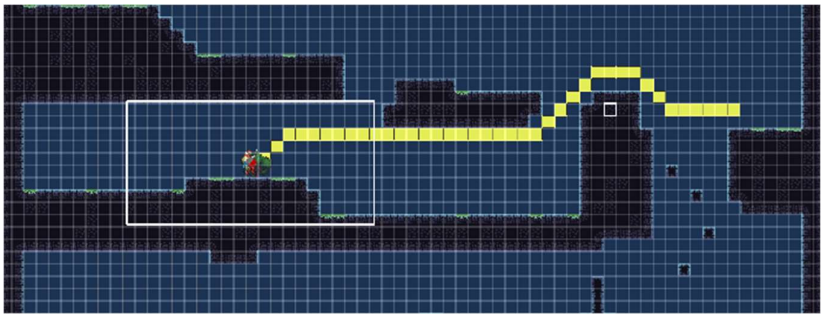
Gambar 2 Cost Jalur

5. Jika node sekarang sudah merupakan node yang dituju, maka akan diambil node parent dari tiap node sebelum node tujuan. Dimana parent node adalah node yang sebelumnya memilih node sekarang sebagai node terbaik. Sehingga ketika semua node parent yang mengarah pada tujuan sudah ditemukan, maka akan terbentuk path / jalur yang dapat ditempuh oleh musuh.

Setelah membuat perhitungan untuk mencari rute terpendek yang akan ditempuh oleh NPC. Selanjutnya adalah membuat musuh untuk bergerak sesuai dengan path yang sudah dibentuk. Untuk itu akan digunakan fungsi yang disediakan oleh Unity yaitu MoveTowards dan juga menggunakan Collider agar musuh tidak dapat menembus obstacle.

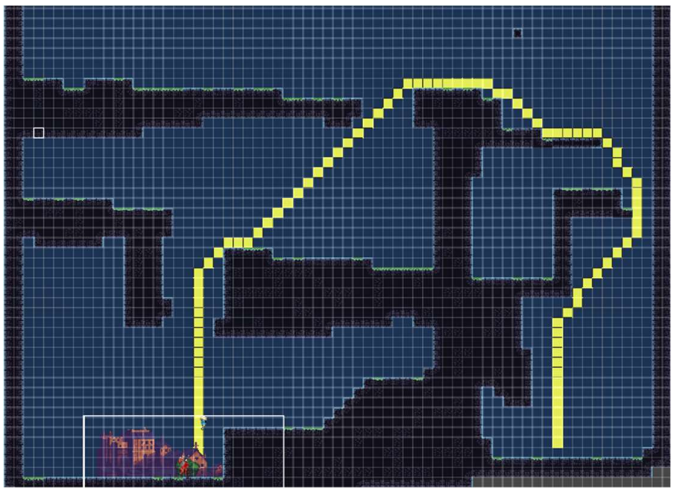
3. HASIL DAN PEMBAHASAN

Pada penelitian ini pengujian dilakukan dengan menguji algoritma A* pada 10 bentuk level pada game dimana tiap level akan memiliki bentuk yang berbeda sehingga musuh diharuskan untuk menemukan rute paling optimal untuk menuju ke pemain. Kondisi untuk memulai algoritma adalah ketika pemain melewati semak semak, maka musuh akan menuju titik semak semak berada, dan akan langsung mengejar pemain jika melihatnya.

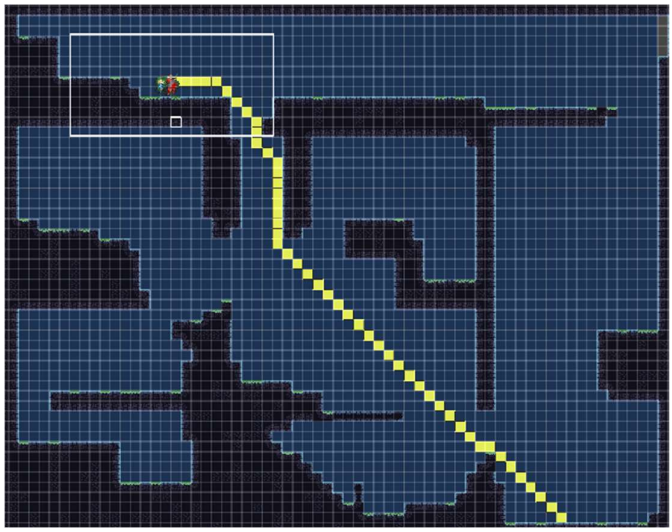


Gambar 3 Pergerakan pada Map 1

Pada map pertama, musuh sudah bisa menghindari obstacle berupa platform dengan melewati rute terpendek yang dapat dilalui menuju lokasi semak semak, rute yang terbentuk dari hasil perhitungan algoritma adalah tile berwarna kuning.

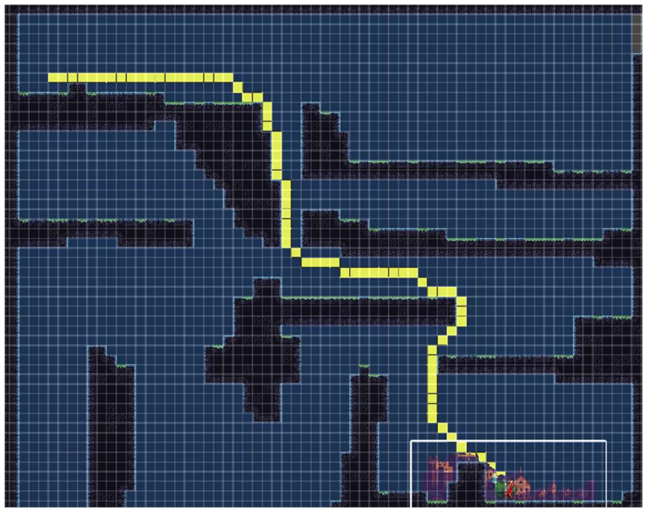


Gambar 4 Pergerakan pada Map 2



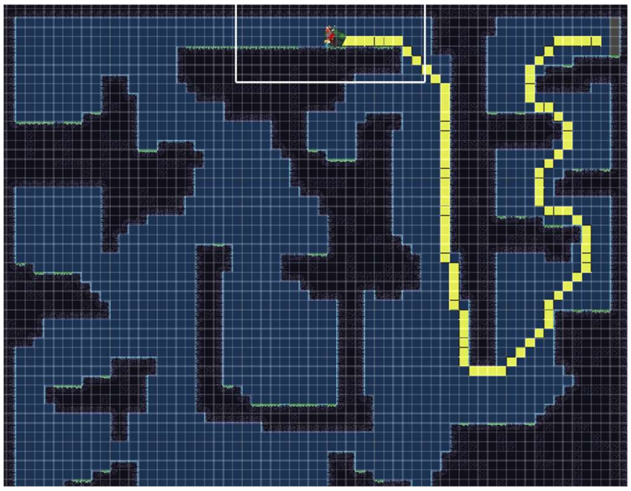
Gambar 5 Pergerakan pada Map 3

Pada map kedua dan ketiga dibuat persimpangan dengan jalan buntu, dapat diketahui algoritma berjalan dengan baik dikarenakan meskipun pada jalan buntu tersebut memiliki f cost yang semakin kecil tetapi semua node tersebut tidak dapat menjadi path karena node tersebut terhalang obstacle untuk mencapai tujuan.



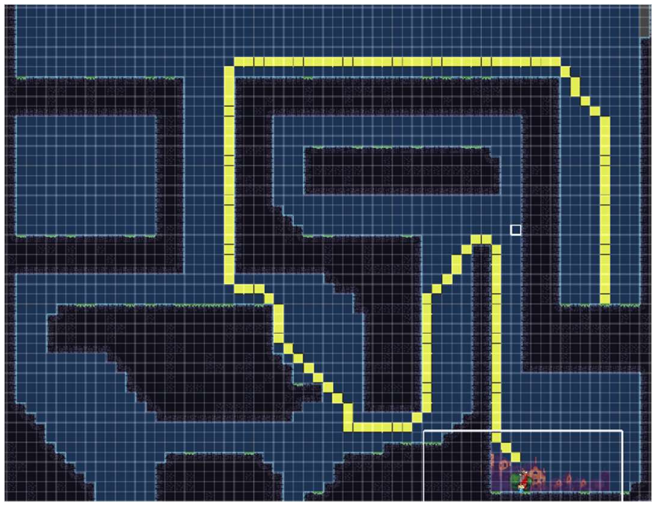
Gambar 6 Pergerakan pada Map 4

Pada map keempat diberikan persimpangan yang mana persimpangan tersebut juga dapat menuju ke titik goal, tetapi path yang terbentuk tetap path dengan jalur terpendek sehingga dapat diketahui algoritma berjalan sesuai dengan yang diinginkan.

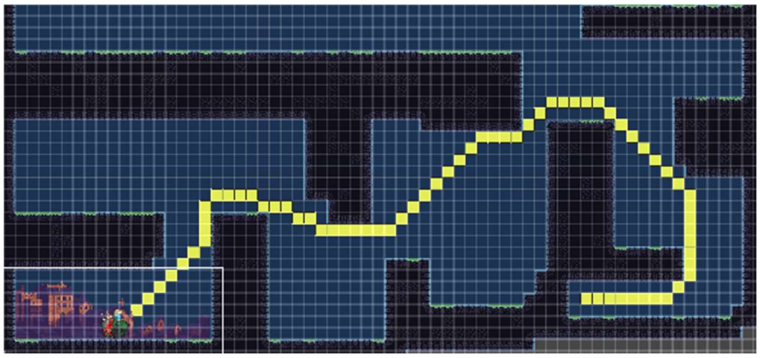


Gambar 6 Pergerakan pada Map 5

Gambar 7 Pergerakan pada Map 6

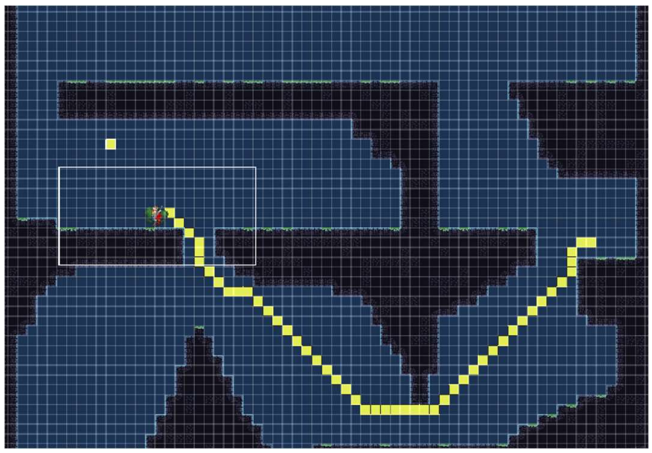


Gambar 8 Pergerakan pada Map 7



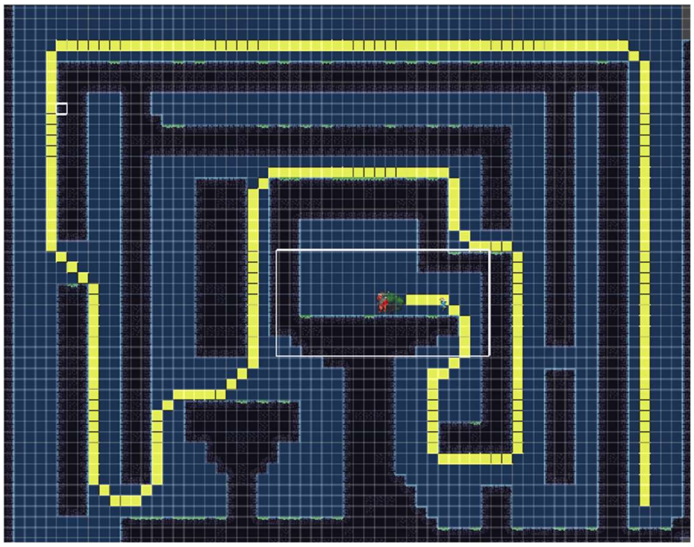
Gambar 9 Pergerakan pada Map 8

Pada map 5 , 6, 7, 8, pengujian dilakukan untuk mengukur kemampuan musuh untuk bergerak memutari jalan yang tertutup oleh obstacle meskipun posisi musuh dan titik tujuannya berdekatan.



Gambar 10 Pergerakan pada Map 9

Map 9 menguji ketepatan jalur terpendek dimana letak musuh dan tujuannya diletakan diketinggian yang sedikit berbeda dan diberikan persimpangan ke atas dan kebawah, dengan meletakkan titik tujuan lebih tinggi daripada titik awal musuh, terdapat kemungkinan jalur keatas akan diambil dikarenakan ketinggan tujuannya. Tetapi sesuai dengan path yang terbentuk musuh memilih untuk melewati jalur bawah dikarenakan meskipun posisi tujuannya lebih tinggi daripada titik awal musuh, mengambil jalur keatas akan memerlukan cost tambahan karena perlunya memutari sedikit obstacle tambahan. Sehingga musuh mengambil jalur kebawah sebagai jalur terpendek.



Gambar 11 Pergerakan pada Map 10

Map terakhir berbentuk seperti labirin dimana musuh harus memutari dan memilih persimpangan mana yang akan diambil untuk mencapai lokasi tujuan. Berdasarkan gambar, meskipun objek tujuan sudah diletakkan pada tengah labirin, musuh tetap dapat menemukan path terpendek untuk mencapai tujuan.

4. KESIMPULAN DAN SARAN

1. Kesimpulan
   Kesimpulan dari penelitian ini adalah sebagai berikut :
   a. Penggunaan Algoritma A* pada game metroidvania menghasilkan potensi pergerakan musuh yang lebih luas, karena meskipun dengan platform yang cukup kompleks musuh tetap bisa menemukan dan berjalan menuju titik tujuan tanpa terhalang apapun.
   b. Target dapat berupa objek apapun dalam game, sehingga tujuan dapat disesuaikan dengan kebutuhan dan tujuan dari musuh tersebut.

2. Saran
   Pada penelitian ini masih terdapat banyak kekurangan yang dapat diperbaiki pada penelitian selanjutnya. Saran untuk penilitian selanjutnya adalah sebagai berikut :
   a. Penentuan node yang akan ditempuh dapat lebih optimal jika digunakan binary tree dibandingkan melakukan looping semua node tetangga.
   b. Musuh dapat di spawn secara random sehingga pemain tidak tahu dari mana musuh akan datang
   c. Target dapat diubah menjadi objek bergerak.

DAFTAR PUSTAKA

[1] T. Gangopadhyay and A. A.-I. J. O. ENGLISH, “Scaffolding in Gamification:„Metroidvania“ and Cognitive Behaviorism,” *ijells.com*, Accessed: Nov. 02, 2022. [Online]. Available: https://www.ijells.com/wp-content/uploads/2021/11/October-2021-.pdf#page=66

[2] B. Tegar, D. Irianto, S. Andryana, and A. Gunaryati, “Penerapan Algoritma A-Star Dalam Mencari Jalur Tercepat dan Pergerakan NonPlayer Character Pada Game Petualangan Labirin Tech-Edu,”

JURNAL MEDIA INFORMATIKA BUDIDARMA, vol. 5, no. 3, pp. 953–962, Jul. 2021, doi: 10.30865/MIB.V5I3.3094.

[3] M. Muiz Adziem Arrahman, J. Teknik Informatika, S. Pontianak Jl Merdeka No, K. Pontianak Kota, K. Pontianak, and K. Barat, "Penerapan Collision Detection Pada Game Platformer 'Culture Seeker,'" *E-JURNAL JUSITI : Jurnal Sistem Informasi dan Teknologi Informasi*, vol. 11, no. 1, pp. 101–111, Jun. 2022, doi: 10.36774/JUSITI.V11I1.915.

[4] G. Mutaqin, … J. F.-W. J. of, and undefined 2021, "Implementasi Metode Path Finding dengan Penerapan Algoritma A-Star untuk Mencari Jalur Terpendek pada Game 'Jumrah Launch Story,'" *journal.walisongo.ac.id*, vol. 3, no. 1, p. 43, 2021, doi: 10.21580/wjit.2021.3.1.7042.

[5] R. Danil Fajri, M. A. Syahputra, T. Raja, M. Zaki, A. Saifudin, and I. Kusyadi, "Perancangan Kecerdasan Buatan pada NPC Menggunakan UNITY 2D dan Perilakunya terhadap Player," *Jurnal Informatika Universitas Pamulang*, vol. 6, no. 3, pp. 575–578, 2021, doi: 10.32493/INFORMATIKA.V6I3.11843.

[6] L. Safira, P. Harsadi, S. Harjanto, P. Studi Informatika, and S. Sinar Nusantara, "Penerapan Navmesh Dengan Algoritma A Star Pathfinding Pada Game Edukasi 3d Go Green," *Jurnal Teknologi Informasi dan Komunikasi (TIKomSiN)*, vol. 9, no. 1, pp. 17–26, Apr. 2021, doi: 10.30646/tikomsin.v9i1.540.

[7] D. Y. Fallo and V. R. Bulu, "PENERAPAN ALGORITMA A STAR (A*) PADA GAME LABIRIN," *Jurnal Pendidikan Teknologi Informasi (JUKANTI)*, vol. 5, no. 1, pp. 118–124, Apr. 2022, doi: 10.37792/JUKANTI.V5I1.459.

[8] A. Hermawan and A. S. Tiwa, "Penerapan Algoritma A-Star untuk Pencarian Tempat Kuliner di Kota Tangerang," *Jurnal Sistem dan Informatika (JSI)*, vol. 15, no. 2, pp. 104–114, May 2021, doi: 10.30864/JSI.V15I2.335.

[9] A. S.-P. S. N. R. dan Teknologi and undefined 2021, "IMPLEMENTASI KECERDASAN BUATAN MENGGUNAKAN ALGORITMA A-STAR DAN REPULSIVE FIELD PADA SIMULASI GAME 3D:---," *journal.unpar.ac.id*, 2021, Accessed: Nov. 02, 2022. [Online]. Available: https://journal.unpar.ac.id/index.php/ritektra/article/view/4835

[10] M. Wicaksono, M. J. Wicaksono, I. Istiadi, I. D. Wijaya, F. Marisa, and S. W. Irianandа, "Museum Angkut Virtual Tour Dengan Optimasi Penelusuran Menggunakan Algoritma A-Star," *JIMP (Jurnal Informatika Merdeka Pasuruan)*, vol. 6, no. 2, Mar. 2022, doi: 10.37438/jimp.v6i2.286.