STUDI ANALISIS PERBANDINGAN ALGORITME *PATHFINDING* PADA SIMULASI UNITY 3D

Aninditya Anggari Nuryono[1)], Igi Ardiyanto[2)], Sunu Wibirama[3)]
[1), 2), 3)] Departemen Teknik Elektro dan Teknologi Informasi,
Fakultas Teknik, Universitas Gadjah Mada
anindityanuryono.sie14@mail.ugm.ac.id, igi@ugm.ac.id, sunu@ugm.ac.id

Abstrak

*Pathfinding digunakan suatu objek untuk mencari jalur dari satu tempat ke tempat lain berdasarkan keadaan peta dan objek lainnya. Dalam pathfinding dibutuhkan algoritme yang dapat dengan cepat memproses dan menghasilkan arah yang terpendek untuk mencapai suatu lokasi tujuan. Algoritme pathfinding yang diulas adalah algoritme A*dan A* smooth Algoritme A* memiliki fungsi heuristik. Algoritme A* smooth merupakan modifikasi dari algoritme A*. Algoritme A* smooth ini bekerja dengan melakukan modifikasi raycast A*. Algoritme A* memanfaatkan node dengan petak-petak kecil. Setiap algoritme ini diimplementasikan ke dalam game object Unity 3D. Setiap game object akan bergerak secara bersamaan untuk menuju titik tujuan dengan posisi awal dan tujuan yang berbeda-beda dengan menghindari banyak halangan. Hasil uji yang didapat adalah algoritme A* smooth lebih unggul dibandingkan dengan algoritme A* dan NavMesh. Waktu tempuh yang dibutuhkan game object dengan algoritme A* smooth lebih cepat 1,6 detik dan 9,6 detik dibandingkan dengan algoritme A* dan NavMesh.*

Kata kunci – *Algoritme A*, Unity 3D, Pathfinding, NavMesh*

1. PENDAHULUAN

Unity 3D adalah perangkat pengembangan *game* yang komprehensif. *Game* merupakan sebuah bentuk hiburan yang dapat dimainkan menggunakan media elektronik. Dalam *game* dibutuhkan AI supaya *game* tersebut dapat menjadi lebih hidup dan menarik (Norvig & Russell, 2010). Sebagian besar di dalam sebuah *game* memilik permasalahan seperti pencarian jalur dari satu titik ke titik yang lain. *Game* yang lebih kompleks juga perlu mempertimbangkan struktur peta *game* dan lokasi target yang dapat dicapai. *Pathfinding* merupakan upaya untuk melakukan pencarian dalam hal ini memanfaatkan *artificial intelligent* (AI).

Penelitian terkait mengenai studi teknik *pathfinding* untuk pada robotika dan video *game*s pada 10 tahun terakhir (Abd Algfoor, Sunar, & Kolivand, 2015). Penelitian ini mengategorikan algoritme *pathfinding* berdasarkan pencarian lingkungan 2D dan 3D. Algoritme yang dipaparkan pada penelitian ini, seperti teknik algoritme A* dan D*. Industri video *game*s dapat menggunakan algoritme *pathfinding* ini pada generasi mendatang, yang akan berbasis pada interaktif Augmented Reality sebagai ekspektasi tren masa depan.

Pada penelitian selanjutnya menyajikan teknik *path planning adaptive grid*, sebuah pendekatan berdasar citra untuk membangkitkan *navigation mesh* (NavMesh) (Akaydın & Güdükbay, 2013). NavMesh direkonstruksi berdasar citra yang diambil dari atas pada model urban 3D. Simulasi navigasi di

keramaian dilakukan pada lingkungan virtual. Perbandingan antara metode *adaptive grid* dengan algoritme *path planning* yang lain, yaitu Dijkstra dan A* dilakukan untuk memperoleh akurasi dan memori yang lebih baik. Pada *static path planning*, metode *adaptive grid* menunjukkan kinerja yang lebih baik. *Adaptive grid* dapat diterapkan pada *static* maupun *dynamic planning.*

Penelitian berikutnya menggabungkan algoritme A* dan *occupancy grids* untuk *pathfinding* diluangkan yang belum dijelajahi (Stamford, Khuman, Carter, & Ahmadi, 2014). Navigasi *pathfinding* ini disimulasikan menggunakan Unity 3D. Penggabungan algoritme tersebut diterapkan pada *Non-Playable Character* (NPC) untuk membuat representasi lingkungan dengan sendirinya dan merencanakan jalur berdasarkan informasi ini.

Dari beberapa penelitian yang telah dikaji sebelumnya, penelitian ini melakukan uji performa perbandingan algoritme A*, A* *smooth*, dan NavMesh dalam sisi waktu tempuh pergerakan objek dengan beberapa skenario pada simulasi Unity 3D.

2. METODE PENELITIAN

Pathfinding merupakan metode yang sangat dibutuhkan pada berbagai *game*, terutama *game* 3D. *Pathfinding* digunakan untuk menentukan arah pergerakan suatu objek dari satu tempat ke tempat lain berdasarkan keadaan peta dan objek lainnya. Dalam pemecahan *pathfinding* akan dibutuhkan algoritme yang dapat dengan cepat memproses dan menghasilkan arah yang terpendek untuk mencapai suatu lokasi tujuan. Salah satu algoritme yang digunakan untuk *pathfinding* adalah algoritme A*, A* *smooth* dan NavMesh.

Algoritme A* merupakan perbaikan dari metode Dijkstra dengan memodifikasi fungsi heuristik. A* akan meminimalkan total biaya lintasan yang terdapat di metode Dijkstra. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal. Pada pencarian jalur kasus sederhana, ketika tidak terdapat halangan pada peta, A* bekerja secepat dan seefisien Dijkstra. Pada kasus peta dengan halangan, A* dapat menemukan solusi rute tanpa terjebak oleh halangan yang ada (Kallmann & Kapadia, 2016). Pseudocode dan diagram alir algoritme A* dapat dilihat pada Algortime 1.

Algoritme 1. Pseudocode Algoritme A*

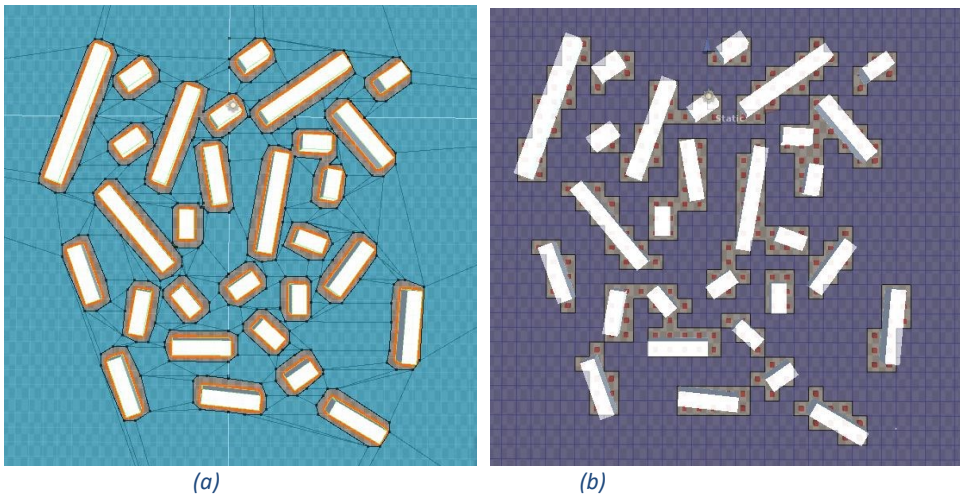
```
OPEN
CLOSED
add the start node to OPEN
loop
current = node in OPEN with the lowest f_cost
remove current from OPEN
add current to CLOSED
        if current is the target node //path has been found
                return
foreach neighbour of the current node
```

Algoritme 1. Pseudocode Algoritme A* (Lanjutan)

```
if neighbour is not traversable or neighbour is in CLOSED
        skip to the next neighbour
if new path to neighbour is shorter OR neighbour is not in
OPEN
    set f_cost of neighbour
    set parent of neighbour to current
    if neighbour is not in OPEN
        add neighbour to OPEN
```

Modifikasi Algoritme A* dilakukan untuk mendapatkan hasil pencarian jalur yang lebih baik. Modifikasi dari algoritme A* ini disebut A* *smooth.* Modifikasi merupakan proses perhitungan jalur baru setelah jalur dihitung. Modifikasi dari algoritme A* ini terdapat pada *ray cast*. *Ray cast* di Unit 3D berfungsi untuk membentuk garis lurus pada jalur yang dilalui.

Navigation Mesh (NavMesh) adalah struktur data yang menggambarkan permukaan walkable dunia *game* dan memungkinkan untuk menemukan jalur dari satu lokasi walkable ke lokasi lain di dunia *game*. Struktur data dibangun, atau di-*baked*, secara otomatis dari geometri yang telah ada. NavMesh terdiri dari poligon cembung yang menutupi ruang kosong, sehingga jalur dapat ditemukan tanpa terjadinya tabrakan dengan halangan. *Mesh* ini adalah algoritme *pathfinding* yang terintegrasi pada Unity 3D. Pada Gambar 1, merupakan NavMesh poligon cembung dan A* berbentuk kotak persegi atau *node* petak-petak kecil. Area berwarna putih dan abu-abu mewakili area yang tidak dapat dilalui dan digunakan. Area berwarna biru merepresentasikan area yang dapat dilalui dan digunakan.

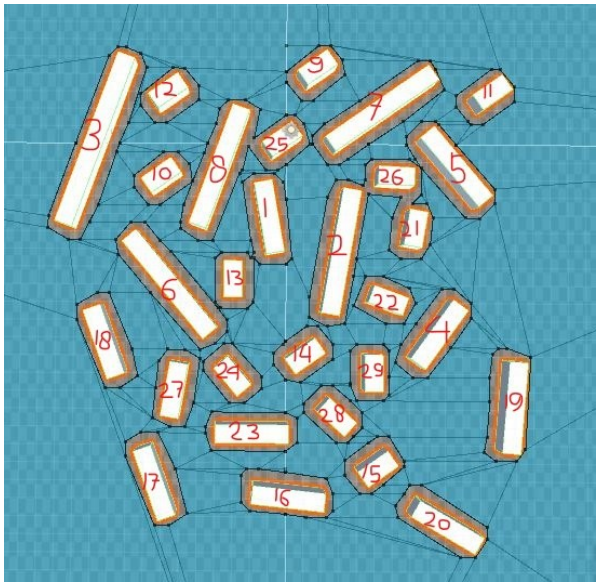


(a) (b)

Gambar 1 Area Algoritme NavMesh (a), dan A (b) di-baked*

3. HASIL PENELITIAN DAN PEMBAHASAN

Metode *pathfinding* diimplementasikan pada objek dan dijalankan pada Unity 3D. Data pencarian jalur pada *pathfinding* yang digunakan merupakan posisi koordinat objek yang terdiri dari koordinat x dan z pada simulasi Unity 3D. Data yang digunakan mempunyai variasi berupa 29 posisi koordinat halangan, tiga agen dengan posisi awal yang sama dan satu posisi titik tujuan. Masing-masing posisi halangan dengan uji simulasi di Unity 3D dapat dilihat pada Gambar 2.

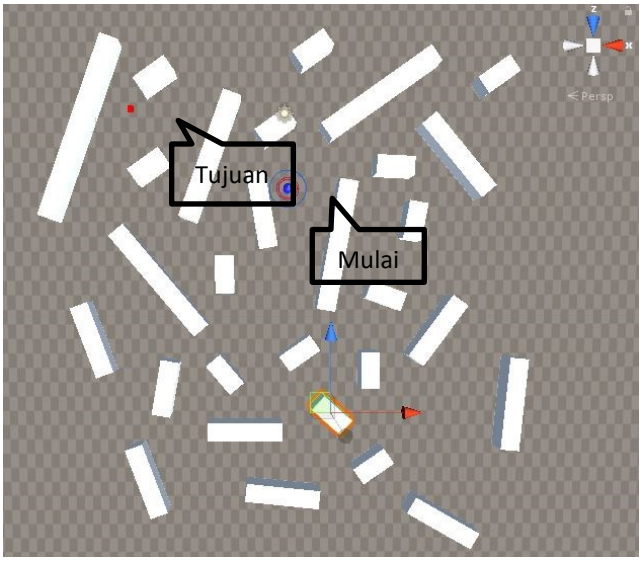


Gambar 2 Posisi halangan di Unity 3D

Pada simulasi di Unity 3D, data yang akan dianalisis dan dibahas adalah pergerakan jalur *pathfinding* yang diimplementasikan ke *game* object. Simulasi *pathfinding* di Unity 3D memiliki tiga skenario yaitu dua skenario dengan titik koordinat yang berbeda dan satu skenario dengan titik awal ke tiga objek memiliki koordinat yang sama. Percobaan ini dilakukan sebanyak lima kali. Hal ini dilakukan untuk menguji konsistensi jalur yang dilalui oleh objek.

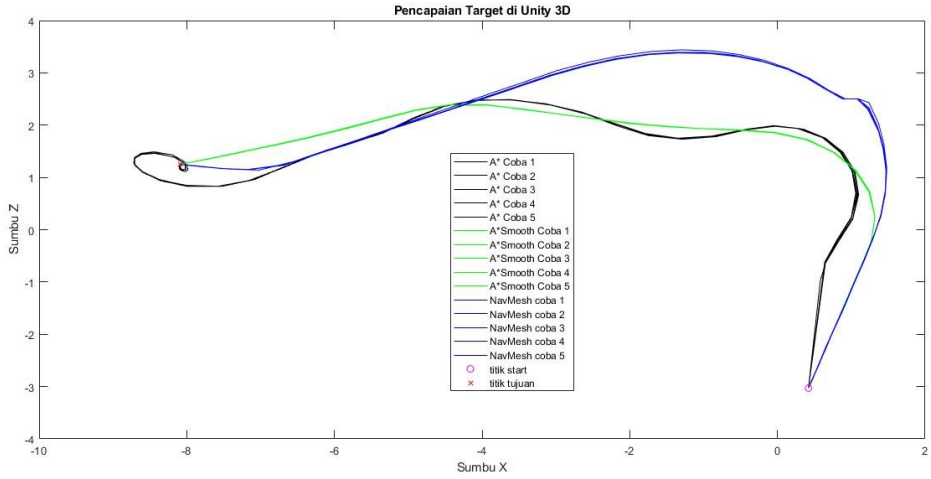
a. Skenario 1

Pada skenario 1 ini dengan koordinat titik awal berada pada sumbu x dengan koordinat 0,42, pada sumbu y dengan koordinat 0, dan pada sumbu z dengan koordinat -3,03. Letak tujuan berada pada sumbu x dengan koordinat -8,09, pada sumbu y dengan koordinat 0, dan pada sumbu z dengan koordinat 1,25. Letak objek pada titik awal dan tujuan dapat dilihat pada Gambar 3.



Gambar 3 Posisi awal dan tujuan objek di skenario 1

Pada Gambar 3, objek berwarna biru merupakan letak titik awal, sedangkan objek berwarna merah merupakan titik tujuan. Hasil pergerakan objek untuk mencapai target dapat dilihat pada Gambar 4 Jalur warna hitam merupakan jalur yang dihasilkan oleh pergerakan algoritme A*. Jalur berwarna hijau merupakan jalur yang dihasilkan oleh algoritme A* *smooth* dan jalur berwarna biru merupakan jalur yang dihasilkan oleh algoritme NavMesh.

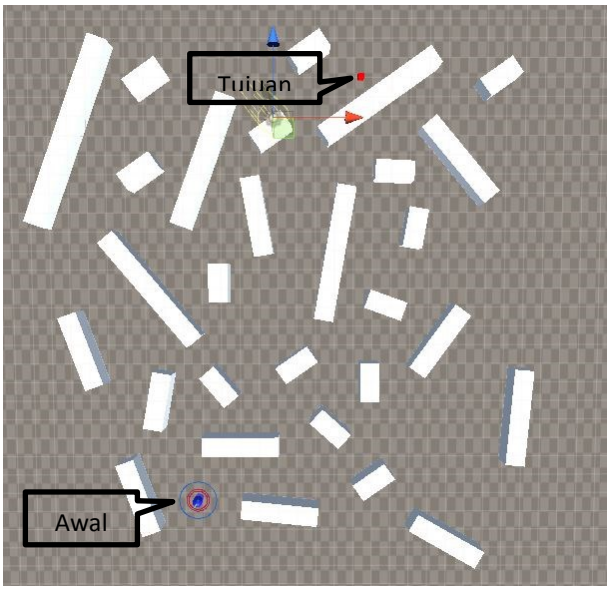


Gambar 4 Jalur pergerakan objek di skenario 1

b. Skenario 2

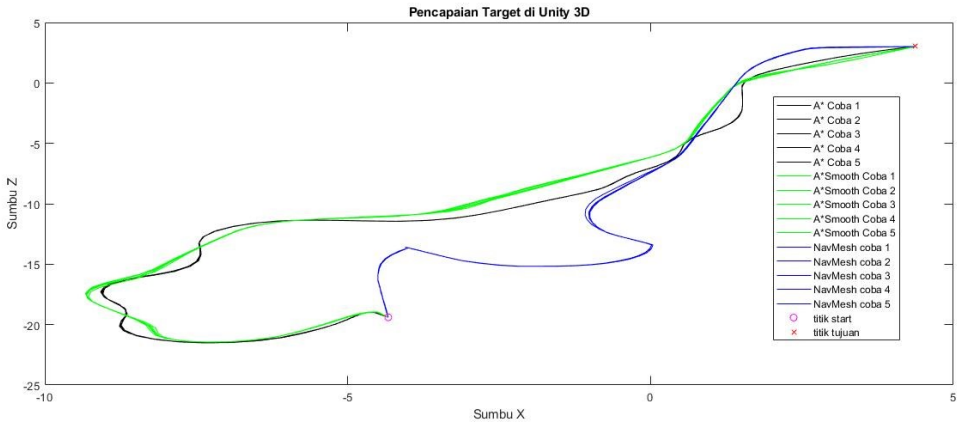
Pada skenario 2 ini dengan koordinat titik awal berada pada sumbu x dengan koordinat -4,33, pada sumbu y dengan koordinat 0, dan pada sumbu z dengan koordinat -19,4. Letak tujuan berada pada sumbu x dengan koordinat 4,37, pada sumbu y dengan koordinat 0, dan pada sumbu z

dengan koordinat 3,01. Letak objek titik pada koordinat titik awal dan tujuan dapat dilihat pada Gambar 5.



Gambar 5 Posisi awal dan tujuan objek di skenario 2

Hasil pergerakan objek untuk mencapai target koordinat tersebut dapat dilihat pada Gambar 6. Jalur warna hitam merupakan jalur yang dihasilkan oleh pergerakan algoritme A*. Jalur berwarna hijau merupakan jalur yang dihasilkan oleh algoritme A* *smooth* dan jalur berwarna biru merupakan jalur yang dihasilkan oleh algoritme NavMesh

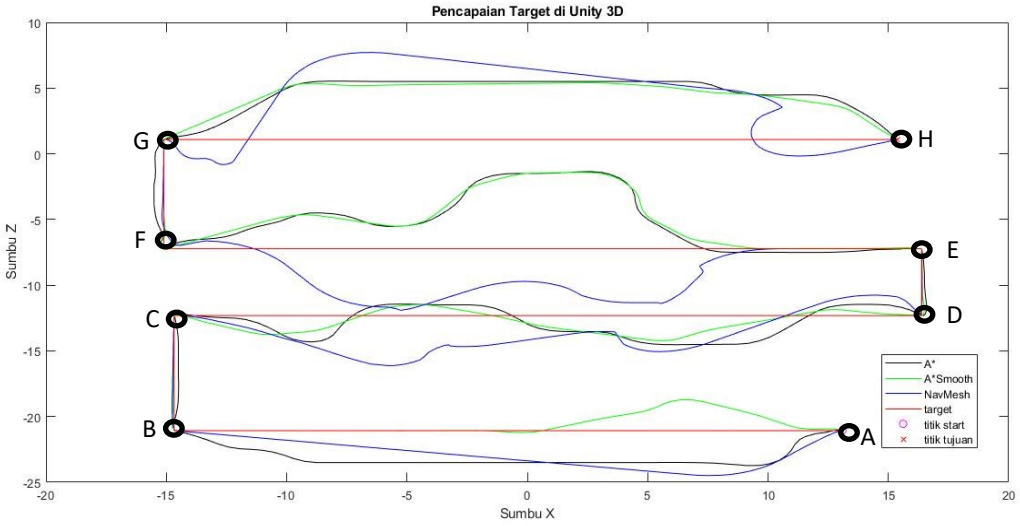


Gambar 6 Jalur pergerakan objek di skenario 2

c. Skenario 3

Pada skenario 3 Target tujuan objek yang ingin dicapai pada skenario ini memiliki koordinat yang berubah karena tujuan akan bergerak. Titik awal objek dan tujuan memiliki koordinat yang sama. ini dengan koordinat titik awal objek dan tujuan terletak pada sumbu x dengan koordinat 13,09, pada sumbu y dengan koordinat 0, dan sumbu z dengan koordinat 21,07. Objek akan mengikuti target. Ketika target bergerak dari

titik A ke titik B, maka objek akan mengikutinya hingga target sampai ke titik H. Visualisasi pergerakan pencapaian target yang bergerak dapat dilihat pada Gambar 7.



Gambar 7 Jalur pergerakan objek di skenario 3

d. Waktu Tempuh

Pengujian sistem dilakukan pada kemampuan algoritme dalam mencari jalur. Data waktu tempuh akan dibandingkan untuk mengetahui seberapa cepat algoritme A*, A* *smooth*, dan NavMesh dalam mencari jalur. Semakin kecil hasil waktu tempuh yang diperoleh, makan semakin cepat objek sampai ke tujuan. Waktu tempuh objek tergantung pada posisi koordinat titik awal dan tujuan, yang berpengaruh pada panjang jalur yang dilalui. Semakin jauh koordinat titik awal dan tujuan, maka waktu tempuh yang didapat semakin besar. Semakin dekat koordinat titik awal dan tujuan, maka waktu tempuh yang didapat semakin kecil. Jauh dan dekatnya titik awal dan tujuan berpengaruh pada panjang jalur. Data waktu tempuh yang didapat berdasarkan letak titik awal dan tujuan dengan koordinat yang telah diulas pada skenario sebelumnya.

Berdasarkan percobaan yang dilakukan sebanyak 5 kali pada skenario 1, waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme A* adalah 4,54 detik. Waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme A* *smooth* adalah **3 detik**. Waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme NavMesh adalah 4,4 detik. Hal ini menunjukkan bahwa algoritme A* *smooth* memiliki waktu tempuh lebih unggul dibandingkan dengan algoritme A* dan NavMesh.

Pada skenario 2, waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme A* adalah 8,38 detik. Waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme A* *smooth* adalah **8,14 detik**. Waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme NavMesh adalah 8,8 detik. Hal ini menunjukkan

bahwa algoritme A* *smooth* memiliki waktu tempuh lebih unggul dibandingkan dengan algoritme A* dan NavMesh.

Pada skenario 3, pergerakan objek yang mengikuti pergerakan target dari titik A hingga H, didapat waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme A* adalah 34,1 detik. Waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme A* *smooth* adalah **32,5 detik**. Waktu tempuh yang dihasilkan oleh objek dengan menggunakan algoritme NavMesh adalah 42,1 detik. Hal ini menunjukkan bahwa algoritme A* *smooth* memiliki waktu tempuh lebih unggul dibandingkan dengan algoritme A* dan NavMesh.

Berdasarkan dari ketiga skenario diatas, algoritme A* *smooth* lebih unggul dibandingkan dengan algoritme A* dan NavMesh karena memiliki waktu tempuh yang lebih sedikit. Hal ini karena algoritme A* *smooth* merupakan modifikasi dari algoritme A* dari sisi *ray cast*. Fungsi *ray cast* di Unity 3D adalah memberi informasi ke objek yang menggunakan fungsi *ray cast* untuk mengetahui objek di lingkungan sekitar dan dapat mengembalikan informasi tambahan seperti adanya titik persimpangan atau bukan maupun halangan. *Ray cast* pada algoritme A* juga digunakan untuk uji *line-of-sight* atau mengetahui sesuatu yang ada depan objek.

4. SIMPULAN

Pathfinding digunakan suatu objek untuk mencari jalur dari satu tempat ke tempat lain berdasarkan keadaan peta dan objek lainnya. Dalam *pathfinding* dibutuhkan algoritme yang dapat dengan cepat memproses dan menghasilkan arah yang terpendek untuk mencapai suatu lokasi tujuan. Hasil uji yang didapat pada simulasi 3D adalah algoritme A* *smooth* lebih unggul dibandingkan dengan algoritme A* dan NavMesh. Waktu tempuh yang dibutuhkan *game* object pada skenario 1, dengan algoritme A* *smooth* lebih cepat 1,54 detik, dan 1,4 detik dibandingkan dengan A* dan NavMesh. Pada skenario 1, dengan algoritme A* *smooth* lebih cepat 0,24 detik, dan 0.66 detik dibandingkan dengan A* dan NavMesh. Pada skenario 3, dengan algoritme A* *smooth* lebih cepat 1,6 detik, dan 9,66 detik dibandingkan dengan A* dan NavMesh.

5. DAFTAR PUSTAKA

Abd Algfoor, Z., Sunar, M. S., & Kolivand, H. (2015). A comprehensive study on pathfinding techniques for robotics and video games. *International Journal of Computer Games Technology*, *2015*. https://doi.org/10.1155/2015/736138

Akaydın, A., & Güdükbay, U. (2013). Adaptive grids: an image-based approach to generate navigation meshes. *Optical Engineering*, *52*(2), 27002. https://doi.org/10.1117/1.OE.52.2.027002

Kallmann, M., & Kapadia, M. (2016). Geometric and discrete path planning for interactive virtual worlds. *ACM SIGGRAPH 2016 Courses on - SIGGRAPH '16*, 1–29. https://doi.org/10.1145/2897826.2927310

Norvig, P., & Russell, S. J. (2010). *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall.

Stamford, J., Khuman, A. S., Carter, J., & Ahmadi, S. (2014). Pathfinding in partially explored games environments: The application of the A*

Algorithm with occupancy grids in Unity3D. In *2014 14th UK Workshop on Computational Intelligence (UKCI)* (pp. 1–6). https://doi.org/10.1109/UKCI.2014.6930151