Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik 201851544
Vol. 5, No. 1, Maret 2018, hlm. 95-103 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 51/E/KPT/2017 e-ISSN: 2528-6579

ANALISA PERBANDINGAN ALGORITMA A* DAN *DYNAMIC PATHFINDING ALGORITHM* DENGAN *DYNAMIC PATHFINDING ALGORITHM* UNTUK NPC PADA *CAR RACING GAME*

Yoppy Sazaki[1], Hadipurnawan Satria[2], Anggina Primanita[3], Muhammad Syahroyni[4]

[1,2,3,4] Universitas Sriwijaya
Email: [1]yoppysazaki@gmail.com, [2]hadipurnawan.satria@gmail.com, [3]anggina.primanita@gmail.com, [4]msyahroyni@hotmail.com

(Naskah masuk: 20 November 2017, diterima untuk diterbitkan: 24 Pebruari 2018)

Abstrak

Permainan mobil balap adalah salah satu permainan simulasi yang membutuhkan *Non-Playable Character* (NPC) sebagai pilihan lawan bermain ketika pemain ingin bermain sendiri. Dalam permainan mobil balap, NPC membutuhkan *pathfinding* untuk bisa berjalan di lintasan dan menghindari hambatan untuk mencapai garis finish. Metode *pathfinding* yang digunakan oleh NPC dalam game ini adalah *Dynamic Pathfinding Algorithm* (DPA) untuk menghindari hambatan statis dan dinamis di lintasan dan Algoritma A* yang digunakan untuk mencari rute terpendek pada lintasan. Hasil percobaan menunjukkan bahwa NPC yang menggunakan gabungan DPA dan Algoritma A* mendapatkan hasil yang lebih baik dari NPC yang hanya menggunakan Algoritma DPA saja, sedangkan posisi rintangan dan bentuk lintasan memiliki pengaruh yang besar terhadap DPA.

Kata kunci: *Algoritma A*; Dynamic Pathfinding Algorithm; Pathfinding; Non-Playable Character; Race Car Game*

Abstract

Race car games are one of the simulation games that require Non-Playable Character (NPC) as an opponent's choice of play when players want to play on their own. In a race car game, the NPC needs pathfinding to be able to walk on the track and avoid obstacles to reach the finish line. The pathfinding method used by NPCs in this game is Dynamic Pathfinding Algorithm (DPA) to avoid static and dynamic obstacles in the path and A Algorithm used to find the shortest path on the path. The experimental results show that NPC using combined DPA and A* algorithms get better results from NPC using only the DPA Algorithm alone, whereas obstacles and track positions have a large effect on DPA.*

.

Keywords: *Algorithm A*; Dynamic Pathfinding Algorithm; Pathfinding; Non-Playable Character; Race Car Game*

1. PENDAHULUAN

Game adalah sebuah sistem yang didalamnya pemain terlibat dalam konflik buatan, yang didefinisikan oleh aturan, yang menghasilkan hasil yang terukur (Salen & Zimmerman, 2003). Kualitas dari suatu *game* berkaitan langsung dengan nilai hiburan dari *game* tersebut dimana nilai hiburan dari suatu *game* di dapat dari pengalaman seorang *player* dalam bermain *game* tersebut. *Simulation Game* adalah *game* yang menyesuaikan dengan situasi dunia nyata. *Car Racing Game* merupakan salah satu contoh dari *simulation game*. *Car Racing Game* mensimulasikan kompetisi balap mobil dengan situasi kompetisi yang ada di dunia nyata, di dalam *game* tersebut terdapat rute yang harus di lewati oleh beberapa pemain yang bersama-sama ingin mencapai garis *finish* di urutan pertama.

Beberapa pemain yang ada di dalam *Car Racing Game* tersebut dapat dikendalikan oleh manusia atau kecerdasan buatan yang di implementasikan pada NPC (*Non-Playable Character*). Peran NPC pada *game* bergantung pada genre *game* tersebut. NPC bisa berperan untuk membantu pemain dalam mencapai *goal* yang ada pada *game* tetapi NPC juga bisa berperan untuk menghalangi atau mendahului pemain mencapai *goal* yang ada pada *game*. Pada *Car Racing Game*, NPC berperan untuk mendahului pemain *game* mencapai garis *finish* pada jalur *racing game* yang ditentukan, untuk mencapai garis *finish*, sebuah NPC harus melakukan *pathfinding* atau pencarian rute pada lintasan lomba yang sedang dimainkan. Pencarian rute tersebut juga mencakup penghindaran hambatan-hambatan acak yang ada pada lintasan.

Untuk dapat memenangkan perlombaan sebuah NPC harus mendapatkan rute terpendek dan menghindari hambatan-hambatan pada lintasan perlombaan dengan baik, ada beberapa algoritma yang digunakan untuk melakukan pencarian rute pada lintasan perlombaan. Salah satunya adalah Algoritma

A* yang melakukan pencarian rute terpendek dengan titik-titik yang ada di setiap lintasan secara keseluruhan. Berdasarkan penelitian terdahulu oleh (Wang & Lin, 2012), hasil Algoritma A* efektif untuk keadaan hambatan yang statis, selanjutnya mereka mengembangkan sebuah algoritma *pathfinding* yaitu, *Dynamic Pathfinding Algorithm* (DPA) yang efektif untuk menghindari secara real-time hambatan-hambatan acak yang muncul secara dinamis tetapi waktu tempuh suatu lintasan sedikit lebih tinggi dari algoritma A* karena rute yang dihasilkan lebih panjang dari rute yang dihasilkan oleh algoritma A*. Berdasarkan uraian di atas, maka dalam penelitian ini akan membandingkan *pathfinding* menggunakan gabungan algoritma *Dynamic Pathfinding* (DPA) dan A* dengan *Dynamic Pathfinding Algorithm* (DPA) untuk NPC pada *Car Racing Game.*

Dalam pencarian rute terpendek Algoritma A* memberikan hasil yang lebih baik dibanding *Dynamic Pathfinding Algorithm*, tetapi dalam menghindari obstacle dinamis pada lintasan, *Dynamic Pathfinding Algorithm* memberikan hasil yang lebih baik daripada Algoritma A*. Berdasarkan kemampuan dari dua algoritma tersebut, maka pada penelitian ini bertujuan

1. Apakah gabungan *Dynamic Pathfinding Algorithm* (DPA) dan Algoritma A* dapat diimplementasikan pada *Car Racing Game* ?.
2. Bagaimana hasil perbandingkan *pathfinding* menggunakan gabungan *Dynamic Pathfinding Algorithm* (DPA) dan Algoritma A* dengan *Dynamic Pathfinding Algorithm* (DPA) yang digunakan untuk NPC pada *Car Racing Game* ?
3. Faktor apa saja yang berpengaruh akibat dari perbandingan pada tujuan no 2 di atas ?.

Penelitian ini terdapat beberapa batasan masalah, yaitu sebagai berikut :

1. Penelitian ini hanya terfokus kepada *pathfinding* untuk NPC pada *Car Racing Game* dengan grafis 3 dimensi.
2. NPC yang digunakan untuk melakukan pathfinding adalah NPC dengan algoritma mobil balap dasar dan dengan kecepatan mobil yang konstan.
3. *Car Racing Game* ini dibuat pada *platform desktop*.

Metode penelitian yang dilakukan dalam penelitian ini adalah:

1. Menganalisa algoritma A* dan *Dynamic Pathfinding Algorithm* (DPA) yang akan diimplementasikan pada Car Racing Game.
2. Mengumpulan data-data ukuran grid A*, jarak titik *collision* DPA, dan model lintasan racing game.
3. Mengembangkan *Car Racing Game* menggunakan metode pengembangan *Rational Unified Process* (RUP).
4. Melakukan pengujian algoritma pada *Car Racing Game*.
5. Analisis hasil penelitian.

2. PENELITIAN TERKAIT

Permasalahan yang diangkat pada penelitian ini adalah pencarian rute terpendek yang ada pada *Simulation Game* khususnya *Car Racing Game*. Terdapat beberapa penelitian sebelumnya yang menjadi dasar dari penelitian ini. Salah satunya adalah penelitian yang dilakukan oleh (Tan, Chen, Tai, & Yen, 2008) mengenai kecerdasan buatan untuk membuat lintasan *racing game*. Pada penelitian tersebut pencarian lintasan racing game dilakukan dengan menggunakan algoritma A* untuk membuat lintasan, algoritma A* pada penelitian ini juga digunakan untuk menghindari *obstacle* yang ada pada lintasan. Algoritma A* di sini dapat menghindari *obstacle* yang dinamis dan banyak.

Penelitian lainnya tentang *Simulation Game* (Khantanapoka & Chinnasarn, 2009), algoritma A* yang digunakan dalam penelitian ini terfokus pada pencarian lintasan yang digunakan dalam permainan 2D dan 3D pada *Real Time Strategy*. Algoritma A* mencari jarak terpendek dalam lintasan untuk mencapai garis *finish*. Dalam penelitian ini algoritma tersebut digunakan untuk lintasan *multi-layer*. Hasil penelitian mereka menunjukkan bahwa Algoritma A* menghasilkan jalan yang tidak halus.

Selanjutnya (J. Wang & Lin, 2012) melanjutkan penelitian mengenai pengaplikasian *pathfinding* pada *Simulating Car Racing Game* dengan membandingkan empat metode *pathfinding* yaitu Algoritma A*, *Modified A* Algorithm : reducing waypoints by a line-of-sight algorithm*, *Modified A* Algorithm : only searching the forward direction* dan algoritma yang mereka (J. Y. Wang & Lin, 2010) kembangkan pada tahun 2010 yaitu *Dynamic Pathfinding Algorithm* (DPA). Algoritma A* digunakan untuk mencari jarak terpendek dalam lintasan dengan mencari semua *waypoints* di sepanjang lintasan. Dua modifikasi dari Algoritma A* dikembangkan untuk mengurangi *waypoints* pada lintasan sehingga komputasi pencarian lintasannya lebih ringan. Sedangkan DPA adalah algoritma yang dikembangkan mereka pada tahun 2010 untuk memecahkan masalah penghindaran *obstacle* yang dinamis.

2.1. Artificial Intelligence (AI).

Artificial Intelligence (AI) adalah kecerdasan buatan pada mesin yang memiliki kemampuan untuk memahami lingkungan mereka dan mengambil tindakan yang memaksimalkan peluang keberhasilan (Murphy & Redfern, 2013). Dalam hal ini biasanya *Artificial Intelligence* dikembangkan untuk menghalangi upaya pemain mencapai tujuan pemain atau misi pemain dengan menjadi musuh atau *Non-Playable Character* (NPC).

Permainan AI sebagai upaya melampaui interaksi kode untuk memasuki dunia yang benar-benar sistem interaktif yang responsif, adaptif, dan cerdas (Ram, Ontañón, Mehta, Ontanón, & Mehta, 2007). Sistem ini seperti yang dijelaskan berusaha untuk belajar tentang beberapa pemain yang dapat menyesuaikan perilaku mereka sendiri dan pengalaman pemain atau kolektif mereka selama bermain *game* juga menggunakan informasi untuk mengatasi kemampuan pemain yang

disediakan oleh penulis permainan. Hasil ini adalah tingkat kecerdasan yang secara dinamis akan mengembangkan dan memberikan pengalaman yang lebih kaya kepada pemain.

Bidang pengembangan *games* yang populer untuk penelitian adalah *Artificial Intelligence* (AI). Pengembang *game* mengembangkan teknik AI untuk *game* komputer karena tidak hanya akan memberikan dampak yang signifikan dalam industri *game*, tetapi juga untuk menciptakan *game* yang memiliki potensi yang memiliki dampak kuat dalam beberapa *domain* lainnya, termasuk pelatihan, pendidikan dan hiburan.

Meskipun *game* biasanya berhubungan dengan hiburan, *game* juga dapat merujuk kepada non-hiburan atau aplikasi yang merujuk pada pendidikan. Ada banyak aplikasi pendidika pada *game*. Tujuan utama dari aplikasi ini adalah untuk menerapkan teknologi simulasi dalam rangka untuk berlatih dan mengevaluasi pengembangan pelatihan dan keterampilan individu atau kelompok kerja dalam bidang tertentu. Aplikasi ini dapat mencakup aplikasi pelatihan medis atau militer. Ada juga minat yang besar dalam menerapkan pelajaran dari desain *game* untuk penggunaan militer dan aplikasi perusahaan (Ram et al., 2007).

Banyak komunitas pengembang *game* baru terus-menerus mencari pendekatan, metode, teknik dan alat-alat yang cocok untuk menerapkan AI dalam permainan mereka. Tujuan utama adalah mereka ingin mengembangkan lebih kompleks AI untuk *game* komputer untuk membuat *game* yang lebih adaptif dan menarik bagi pemain. *Game* telah menjadi *domain* yang menarik untuk *Artificial Intelligence* (AI) penelitian, terutama permainan papan seperti catur, dam, atau *backgammon*. Namun antara tahun 1999 dan 2000, jumlah penelitian tentang AI mengidentifikasi video *game* interaktif sebagai domain penelitian yang menarik. Contoh AI dalam permainan utama termasuk *Half Life* dan *FEAR* (*First Encounter Assault Recon*).

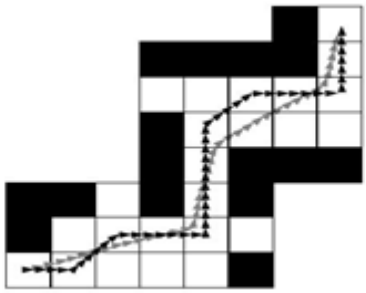
2.2. Representasi *Map.*

Sebuah permainan memiliki *map* yang digunakan sebagai tempat berlangsungnya permainan. Map tersebut tidak bisa secara langsung digunakan dalam

bentuk fisik. Dibutuhkan representasi *map* untuk dapat menggunakan *map* tersebut dalam permainan. Khususnya pada proses pencarian rute didalam *map*. Representasi *map* dibutuhkan oleh algoritma pencarian rute untuk menentukan *node* yang akan dihitung dalam proses algoritma pencarian rute tersebut. Terdapat beberapa meteode yang digunakan untuk merepresentasikan *map* pada permainan diantaranya adalah *grids*, *waypoint graph*, dan *navigation mesh.*

2.2.1. Grids

Representasi *grids* sering disebut juga sebagai *tile graphs*. *Grid* tersebut digunakan untuk membagi *map* menjadi *cell* yang teratur berbentuk kotak, segitiga, atau heksagonal (Millington & Funge, 2009). Map direpresentasikan melalui array berdasarkan *cell grid* yang terbuka dan tertutup. *Grids* biasanya digunakan untuk merepresentasikan *map* 2 dimensi, tetapi dapat digunakan juga untuk merepresentasikan *map* 3 dimensi (Sturtevant, 2011).



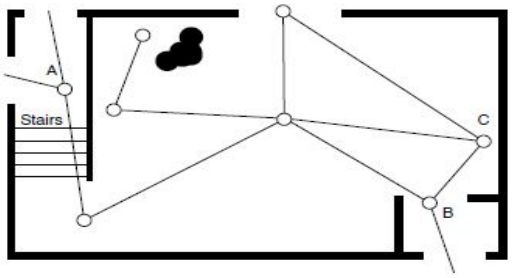
Gambar 1. Representasi *Grid* (Millington and Funge, 2009)

Grids merupakan representasi *map* yang paling sederhana dan mudah diimplementasikan, *grids* dapat dengan mudah diubah secara eksternal menggunakan *text editor* (Rabin, 2014). Pada proses pencarian rute dengan representasi *grids*, *node* diletakkan di setiap koordinat *cell grids*. Sehingga *grids* yang dibuat harus mencakup seluruh *map* yang ada pada *game*. Keadaan ini membuat *grids* menggunakan memori yang cukup besar jika *map* yang direpresentasikan memiliki *size* yang cukup besar juga.

2.2.2.Waypoint Graph

Waypoint graph merepresentasikan *map* sebagai grafik abstrak. *Waypoint graph* tidak memiliki pemetaan yang jelas antara *node* dalam grafik dan area yang dapat dijalani. *Waypoint graph* banyak digunakan sebelum *navigation mesh* popular (Rabin, 2013).

Waypoint graph relative mudah untuk diimplementasikan. *Waypoint* diletakkan pada area yang dapat dilewati oleh karakter *game* dan dihubungkan oleh garis. Setiap garis menghubungkan dua *waypoint.* Karakter *game* dapat memilih garis yang akan dilewati tanpa khawatir dengan *obstacle* yang ada. Selain itu *waypoint* lebih mudah untuk diubah jika terjadi



Gambar 2. Representasi *Waypoint* (Millington and Funge, 2009)

perubahan pada *map* yang diketahui sebelumnya. Tetapi jika perubahan tersebut tidak diketahui sebelumnya maka perubahan pada *waypoint* akan menjadi sulit. Untuk mendapatkan rute yang bagus, *waypoint* mungkin memerlukan peletakan secara manual.

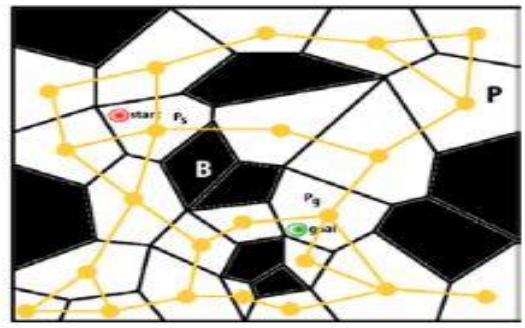
2.2.3. Navigation Mesh

Navigation mesh merupakan kombinasi dari *grids* dan *waypoint graph*. *Navigation mesh* menentukan area mana yang dapat dilewati oleh karakter *game* dengan merepresentasikan *map* menggunakan *convex polygons* (Tonzour, 2004). *Polygon* dapat merepresentasikan *map* lebih akurat dari *grid* karena *polygon* dapat merepresentasikan *map* dengan blok *non grid*. Selain itu perencanaan rute pada *navigation mesh* biasanya cepat karena *navigation mesh* merepresentasikan *map* secara kasar. Tetapi tidak mengurangi kualitas dari rute karena karakter *game* bebas berjalan disetiap sudut.

Navigation mesh sering kali membutuhkan algoritma geometri yang menandakan bahwa *navigation mesh* lebih sulit untuk diimplementasikan (Chen, 2009). Selain itu perubahan pada *navigation mesh* lebih sulit atau mahal untuk diimplementasikan, terutama apabila dibandingkan dengan perubahan pada representasi *grids* (Rabin, 2013).

2.3 Algoritma A*

Algoritma A* adalah metode yang sudah banyak diketahui dan digunakan secara luas pada banyak *game*. Metode ini memberikan keseimbangan yang baik antara kecepatan dan ketepatan.



Gambar 3. Representasi *Navigation Mesh* (Cui and Shi, 2012)

Pada umumnya, algoritma ini mencoba mencari lintasan terbaik dan paling efisien untuk mencapai garis *finish*. Algoritma ini berdasarkan fungsi heuristic. *Pathfinding Analysis* digunakan untuk menyediakan jalur yang tepat untuk gerakan dari posisi awal ke posisi tujuan dalam berbagai permainan, baik bagian NPC dan bagian kecerdasan buatan. Hal tersebut dapat menyelesaikan masalah pergerakan untuk menghindari *obstacle* secara fleksibel. Setiap algoritma akan memanggil tumpukan dan antrian yang berbeda dari setiap *genre*. *Genre* yang populer adalah *Computer Game* dan Permainan strategi *real time*. *Game* tersebut telah lama ada dan sangat berkembang dengan teknologi serta bekerja dalam bidang kecerdasan buatan. Algoritma A* dapat dinyatakan sebagai berikut (Wang dan Lin, 2012).

$$f(n) = g(n) + h(n) \quad (1)$$

dimana :
g(n) = nilai yang tepat dari titik awal untuk simpul n.
h(n) = heuristic nilai dari simpul n ke tujuan.

2.4. *Dynamic Pathfinding Algorithm* (DPA)

Dynamic Pathfinding Algorithm (DPA) merupakan algoritma pathfinding yang digunakan untuk mencari jalan terpendek dan menghindari *obstacle* yang dinamis. Algoritma ini mencari *obstacle* yang berada di dekat mobil selama ia berpindah pada lintasan. DPA baik digunakan untuk *random obstacle* seperti musuh yang ada dalam permainan.

Cara kerja *Dynamic Pathfinding Algorithm* (DPA) sangat mudah. Dalam permainan, kita membuat beberapa deteksi warna di depan atau di sekitar mobil yang bergerak. Kadang kala mobil menemukan *obstacle* dan kadang mobil berjalan lancar (tidak ada *obstacle*). Jika posisi warna mobil yang terdeteksi sama dengan warna lintasan (hitam), maka tidak ada musuh atau *obstacle* di depannya. Sebaliknya, jika posisi warna mobil yang terdeteksi adalah putih (tidak sama dengan warna lintasan) maka ada *obstacle* di sekitarnya. Jika hal tersebut terjadi, maka mobil harus berpindah atau bergerak untuk menghindari *obstacle* tersebut. Untuk menghindari *obstacle* tersebut, dibuat pengaturan tetap untuk NPC. Jika *obstacle* berada di sisi kiri mobil, maka mobil akan berpindah searah jarum jam, dan jika *obstacle* berada di sisi kanan mobil, maka mobil akan bergerak sebaliknya. Pergerakan tersebut terjadi berdasarkan radian yang ditentukan. Kecepatan tersebut ditentukan agar pergerakan mobil terlihat alami dan halus.

```
start
        while finish is false
                createCollision()
                if Collision active is two
                        set Collision to 45 degree
offset
                        if Collision is in right
                                if Collision is in left
                                        set Rem active
                                else
                                        going to left
                        else
                                going to right
                        endif
                        set position to the start
                else
                        if Collision is in right
                                going to left
                        else
                                if Collision is in left
                                        going to right
                                else
                                        move foward
                                endif
                        endif
                endif
        endwhile
end
```

Gambar 4. *Pseudocode Dynamic Pathfinding Algorithm*

3. HASIL DAN ANALISA

3.1. Implementasi Antarmuka

Impelementasi antarmuka pada gambar 5 merupakan *interface* dari *game* yang dikembangkan. Implementasi dalam penelitian ini dibuat berdasarkan perancangan sebelumnya.

Gambar 5. Implementasi Antar Muka

Pada tahap perancangan antarmuka sebelumnya telah dijelaskan bahwa terdapat beberapa pilihan menu pada antarmuka menu utama yaitu *start, help, credit,* dan *quit*. Setiap pilihan menu memiliki fungsi masing-masing. Salah satu pilihan menu tersebut adalah menu *help* yang bertujuan untuk memberikan informasi mengenai cara memainkan *game* tersebut. Antarmuka menu *help* tersebut dapat dilihat pada Gambar 6.

Gambar 6. Antarmuka Menu *Help*

3.2. Analisis Data

Data yang digunakan pada perangkat lunak yang akan dibangun adalah contoh lintasan mobil balap dari sebuah *racing game* berjudul Need for Speed : Most Wanted. Contoh lintasan mobil balap ini di modelling ke dalam bentuk 3D untuk digunakan pada proses pengembangan *game* ini.

3.3. Analisis Unit-Unit *Game*

Game yang dirancang adalah *Car Racing Game* yang menggunakan karakter mobil sebagai *avatar* yang mewakili permainan. *Car Racing Game* memiliki lintasan yang harus dilewati *avatar* untuk mencapai garis *finish* agar objektif game terpenuhi. Untuk dapat mencapai garis *finish*, pemain dan NPC dapat bergerak maju, mundur, belok kanan, dan belok kiri mengikuti lintasan pada *game*.

Karakter mobil pada *game* ini memiliki ukuran 0.6x1.5 *unit in unity* dan bergerak dengan kecepatan 2 *unit in unity / second* secara konstan. Ordo *grid* yang digunakan untuk merepresentasikan lintasan pada game adalah 120x120 dengan ukuran *cell* sebesar 2 *unit in unity*. Jarak antara mobil dengan titik *collission* yang akan digunakan oleh metode DPA sebesar 8 *unit in unity* dihitung dari titik tengah mobil. Obstacle pada *game* berbentuk kotak dengan ukuran 2x2 *unit in unity* dan bergerak dengan kecepatan 0.5 *unit in unity / second.*

Mekanik mobil pada *game* menggunakan *engine script* dari *Unity Car Tutorial*. Walaupun mobil bergerak secara konstan, pada saat berbelok mobil mendapatkan sedikit percepatan agar proses berbeloknya mobil membentuk kurva sehingga terlihat realistis.

3.3. Analisis Algoritma A*

Algoritma A* adalah algoritma yang akan digunakan untuk mencari rute terpendek pada lintasan balapan yang dipakai. Pencarian rute tersebut membutuhkan sebuah *grid* x,y yang mencakup seluruh lintasan. Setiap titik tengah *cell* pada *grid* tersebut akan menjadi sebuah node yang akan menentukan apakah *node* tersebut masuk kedalam *walkabale area* atau *unwalkable area*. Kemudian Algoritma A* akan mencari rute terpendek dari *node start* menuju ke *node finish* melaui *node-node* yang berada di dalam *walkable area*.

Proses Algoritma A* bergantung pada jumlah baris dan kolom yang dibutuhkan untuk mencakup keseluruhan lintasan. Semakin banyak baris dan kolom, maka akan semakin banyak *node* yang terbentuk dan semakin banyak pula perhitungan yang akan dilakukan. Namun, rute yang di hasilkan akan semakin detail. Sedangkan jika baris dan kolom jumlahnya sedikit, maka proses pencarian rute akan berjalan lebih cepat karena perhitungan yang dilakukan sedikit. Namun, rute yang dihasilkan tidak terlalu detail karena ukuran *cell* nya akan lebih besar yang berarti bahwa sebuah *node* akan mewakili area yang cukup besar.

3.4. Analisis Dynamic Pathfinding Algorithm

Dynamic Pathfinding Algorithm (DPA) adalah metode yang digunakan untuk menghindari hambatan-hambatan yang ada pada lintasan. DPA menggunakan 2 titik *collission* yang berada di depan objek mobil. Titik *collission* tersebut digunakan untuk mendeteksi hambatan yang ada di kanan dan di kiri area depan mobil. Jika kedua titik tersebut mendeteksi adanya hambatan secara bersamaan, berarti hambatan tersebut tepat berada di depan objek mobil. Pada kondisi tersebut, titik-titik *collission* akan melakukan perpindahan sebesar 45° ke kanan dan ke kiri mobil dengan tujuan untuk melakukan pendeteksian hambatan di area sisi kanan dan kiri objek mobil.

3.4. Analisis Gabungan Algoritma A* dan *Dynamic Pathfinding Algorithm*.

Gabungan Algoritma A* dan *Dynamic Pathfinding Algorithm* didasari oleh kelebihan dari algoritma masing-masing. Kemampuan Algoritma A* dalam

mencari rute terpendek dan kemampuan *Dynamic Pathfinding Algorithm* dalam menghindari hambatan - hambatan dinamis akan digabungkan dan dijalankan bersama pada sebuah NPC pada *Car Racing Game*.

Sebelum permainan mobil balap dimulai, Algoritma A* akan mencari rute terpendek pada lintasan tersebut dari garis *start* hingga garis *finish* tanpa dipengaruhi oleh hambatan yang ada pada lintasan. Kemudian rute terpendek tersebut akan digunakan oleh NPC sebagai panduan NPC untuk mencapai garis finish ketika permainan dimulai.

Saat permainan dimulai NPC akan mengikuti rute terpendek hasil dari Algoritma A* dan menjalankan *Dynamic Pathfinding Algorithm* untuk mendeteksi hambatan pada lintasan. Ketika NPC mendeteksi adanya hambatan maka NPC akan mengikuti *Dynamic Pathfinding Algorithm* untuk menghindari hambatan tersebut. Ketika hambatan tidak terdeteksi maka NPC akan mengikuti rute dari Algoritma A* kembali. Proses tersebut akan di ulang hingga NPC mencapai garis finish.

```
start
        SearchRuteA()
        StartGame()

        while finish is false
                if Obstacle is true
                        DynamicPathfinding()
                endif

                Move()
        endwhile
end
```

Gambar 7. *Pseudocode* Pemrosesan Gabungan Algoritma A* dan *Dynamic Pathfinding*

3.5. Hasil Pengujian Metode

Dalam melakukan pengujian metode, teknik yang digunakan dalam penelitian ini adalah teknik pengujian kualitatif. Pengujian kualitatif dilakukan dengan cara membandingkan kemampuan NPC yang menggunakan gabungan Algoritma A* dengan *Dynamic Pathfinding Algorithm* dan NPC yang hanya menggunakan *Dynamic Pathfinding Algorithm* pada tikungan dengan derajat tikungan yang berbeda-beda dan pencapaian garis *finish* pada lintasan perlombaan kosong dan lintasan perlombaan yang memiliki *obstacle*.

Tabel 1. Tikungan 30 derajat

Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
Kanan				
1.	Berhasil	34.0098	Berhasil	66.0557
2.	Berhasil	33.6497	Berhasil	65.6663
3.	Berhasil	33.7134	Berhasil	65.5805
Kiri				
1.	Berhasil	34.0134	Berhasil	66.0432
2.	Berhasil	33.6662	Berhasil	65.7621
3.	Berhasil	33.8125	Berhasil	65.5523

Tabel 2. Tikungan 60 derajat

Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
Kanan				
1.	Berhasil	38.352	Berhasil	65.882
2.	Berhasil	38.0653	Berhasil	66.257
3.	Berhasil	37.32941	Berhasil	68.706
Kiri				
1.	Berhasil	38.333	Berhasil	65.999
2.	Berhasil	37.9898	Berhasil	66.2542
3.	Berhasil	37.2944	Berhasil	67.9745

Tabel 3. Tikungan 90 derajat

Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
Kanan				
1.	Berhasil	48.654	Berhasil	84.5897
2.	Berhasil	44.902	Berhasil	81.2925
3.	Berhasil	42.796	Gagal	-
Kiri				
1.	Berhasil	48.458	Berhasil	83.8769
2.	Berhasil	44.834	Berhasil	82.4432
3.	Berhasil	42.788	Gagal	-

Tabel 4. Spiral

Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
1.	Berhasil	41.329	Berhasil	86.1920
2.	Berhasil	40.546	Berhasil	86.1991
3.	Berhasil	41.428	Berhasil	86.6228

Tabel 5. Letter S

Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
1.	Berhasil	82.967	Berhasil	154.204
2.	Berhasil	80.881	Berhasil	154.035
3.	Berhasil	78.928	Berhasil	154.169

Tabel 6. Lintasan Kosong

Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
1	Berhasil	151.849	Gagal	-
2	Berhasil	150.997	Gagal	-
3	Berhasil	151.643	Gagal	-

Tabel 7. Lintasan Obstacle Statis

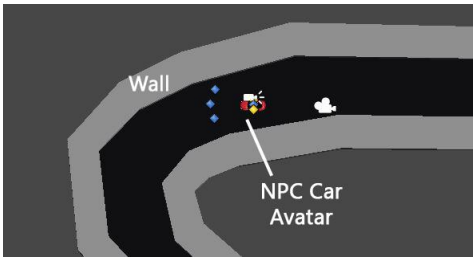
Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
1	Berhasil	151.185	Berhasil	261.5538
2	Berhasil	150.4079	Gagal	-
3	Berhasil	153.0912	Berhasil	262.1604
4	Gagal	-	Gagal	-
5	Gagal	-	Gagal	-

Tabel 8. Lintasan Obstacle Dinamis

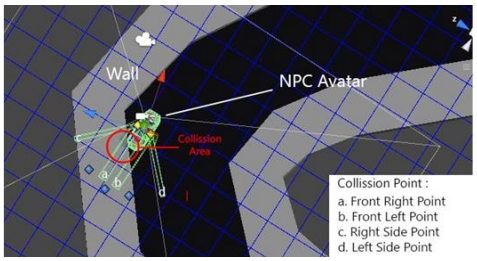
Percobaan ke-	Gabungan Algoritma A* dan DPA		DPA	
	Status	Waktu (detik)	Status	Waktu (detik)
1	Berhasil	149.074	Berhasil	260.7668
2	Berhasil	149.7056	Gagal	-
3	Berhasil	151.2679	Gagal	-
4	Gagal	-	Gagal	-
5	Gagal	-	Gagal	-

3.6. Analisa

Berdasarkan hasil pengujian metode pada Tabel 1 hingga Tabel 6 dapat dilihat bahwa NPC yang menggunakan gabungan Algoritma A* dan DPA berhasil mencapai garis *finish* pada seluruh percobaan yang dilakukan pada tikungan dengan derajat yang berbeda-beda dan pada lintasan perlombaan yang tidak memiliki *obstacle*. Percobaan pada lintasan perlombaan



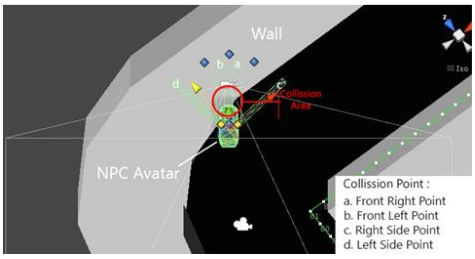
Gambar 8. Posisi dan Arah Mobil Sebelum Tabrakan



Gambar 9. Posisi Kegagalan Metode DPA pada Lintasan Perlombaan Kosong

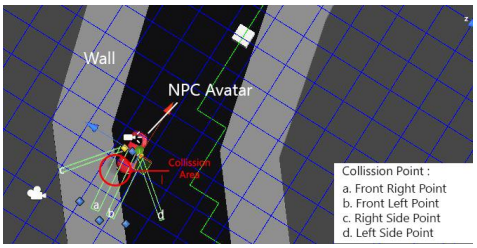
yang memiliki *obstacle* yang dilakukan dengan metode gabungan Algoritma A* dan DPA berhasil mencapai garis finish sebanyak 3 kali dari 5 kali percobaan. Sedangkan NPC yang hanya menggunakan DPA berhasil mencapai garis *finish* pada seluruh percobaan dengan lintasan tikungan 30 derajat, 60 derajat, spiral dan letter S. Namun, pada lintasan lainnya NPC yang hanya menggunakan DPA mengalamai kegagalan pada beberapa percobaan.

Kegagalan NPC yang hanya menggunakan DPA pada lintasan perlombaan kosong dapat dilihat pada Gambar 8 dan 9 berikut.



Gambar 10. Posisi Kegagalan DPA pada Lintasan Tikungan 90

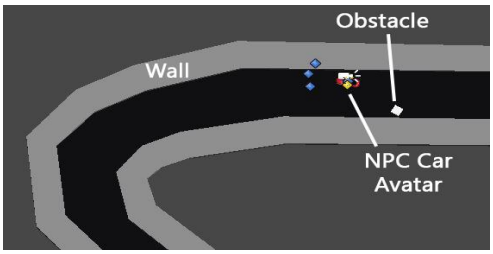
Pengambilan keputusan untuk berbelok ataupun jalan lurus pada DPA didasarkan oleh titik *collider* yang ada di depan dan samping avatar mobil. Ketika NPC bertemu dengan tikungan tajam dalam posisi hampir



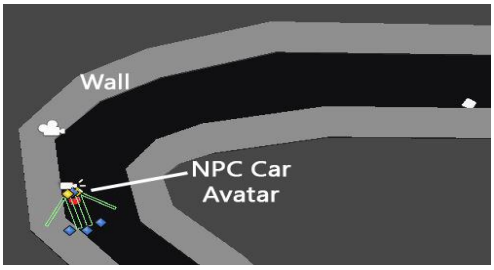
Gambar 11. Posisi Kegagalan NPC pada Lintasan dengan Obstacle Dinamis

tegak lurus, mobil terlambat untuk berbelok dan menyentuh dinding lintasan. Keadaan ini juga terjadi pada saat mobil di uji pada tikungan 90 derajat. Dapat dilihat pada gambar 10.

Pada lintasan perlombaan yang memiliki *obstacle*, NPC yang hanya menggunakan DPA mengalami kegagalan yang sama dengan kegagalan pada lintasan kosong. Kegagalan itu terjadi ketika ada obstacle yang berada di dekat tikungan dan berada di depan avatar mobil. Keputusan NPC untuk menghindar dari obstacle membuat NPC mendekati sisi luar tikungan dengan tajam. Saat NPC mengambil keputusan untuk menghindari dinding lintasan, ruang berbelok NPC sudah terlalu kecil karena NPC sudah terlalu dekat dengan sisi luar tikungan pada lintasan. Kegagalan NPC tersebut dapat dilihat pada gambar 11.



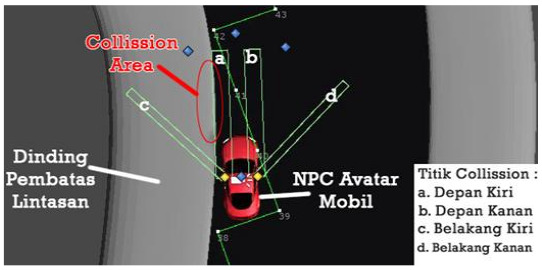
Gambar 12. Posisi NPC Setelah Menghindari Obstacle dan Sebelum Melewati Tikungan



Gambar 13. Posisi NPC Berhasil Melewati Tikungan

Namun pada kondisi tertentu, NPC dengan DPA berhasil melewati lintasan yang memiliki obstacle. Keberadaan obstacle mempengaruhi arah dan posisi NPC di lintasan karena NPC dengan DPA tidak memiliki rute fix yang akan diikuti oleh NPC melainkan pergerakannya berdasarkan trigger dari titik collider yang ada di depan NPC. Sehingga pada posisi obstacle tertentu, kegagalan NPC yang selalu terjadi pada lintasan kosong tidak terjadi. Kondisi tersebut dapat dilihat pada gambar 12 dan 13 berikut.

Gambar 12 menunjukkan posisi NPC setelah menghindari obstacle sehingga NPC sudah mendekati sisi luar tikungan sebelum melalui area tikungan. Hal ini menyebabkan NPC dapat mengikuti alur tikungan dari awal dimana sisi tikungan masih landai dan secara bertahap menuju sisi tikungan yang tajam sehingga NPC berhasil melewati tikungan tersebut yang ditunjukkan

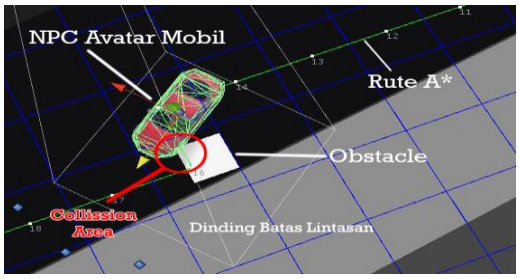


Gambar 14. Keberhasilan NPC yang Menggunakan Gabungan Algoritma A* dan DPA Melewati Area yang Sama

oleh gambar 13. Bandingkan dengan gambar 8 dimana NPC langsung menuju sisi tikungan yang tajam, sehingga ruang untuk berbelok menjadi sangat sempit dan NPC menyentuh dinding lintasan yang ditunjukkan oleh gambar 9.

Pada NPC yang menggunakan gabungan Algoritma A* dan DPA melalui kondisi lintasan perlombaan kosong maupun pada lintasan perlombaan yang memiliki *obstacle*. dapat diatasi dengan baik. Keberhasilan NPC melewati area yang sama dimana NPC yang hanya menggunakan DPA tidak berhasil pada area tersebut dapat dilihat pada gambar 14.

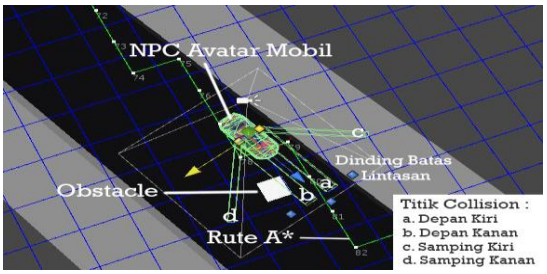
Tetapi gabungan kedua algoritma tersebut menemui kegagalan pada kondisi dimana *node* hasil pencarian rute berada diantara dinding pembatas



Gambar 15. NPC Menabrak Obstacle yang Sudah Berpindah dari Posisi Awal

lintasan dan *obstacle*. Kondisi tersebut dapat dilihat pada Gambar 15 dan 16.

Hal ini disebabkan *Artificial Intelligence* (AI) yang diterapkan pada NPC bergerak berdasarkan hasil pendeteksian *obstacle* pada lintasan. Pada kondisi diatas titik *collision* sebelah kanan mendeteksi sebuah *obstacle* pada area kanan depan mobil sehingga membuat NPC berbelok ke kiri. Setelah NPC berbelok ke kiri titik *collision* sebelah kiri mendeteksi dinding pembatas lintasan tetapi titik *collision* kanan masih mendeteksi *obstacle* yang ada di area kanan depan mobil. Karena dua titik *collision* yang ada di bagian kanan depan dan kiri depan mobil mendeteksi adanya *obstacle* yang menandakan bahwa area di depan mobil tidak bisa dilewati maka NPC mengaktifkan titik *collision* yang berjarak 45 derajat dari titik *collision* awal untuk mendeteksi *obstacle* di area samping mobil. Titik *collision* sebelah kiri mendeteksi dinding pembatas lintasan pada area samping kiri mobil sedangkan titik *collision* sebelah kanan tidak mendeteksi adanya *obstacle* apapun sehingga membuat NPC berbelok ke kanan. Tetapi karena NPC sudah sangat dekat dengan *obstacle* yang disebelah kanan maka tabrakan antara NPC dan *obstacle* disebelah kanan tetap terjadi.



Gambar 16. NPC Tidak Berhasil Mengikuti *Node* A* yang Melewati Ruang Antara Obstacle dan Dinding Pembatas Lintasan

4. KESIMPULAN DAN SARAN

Adapun kesimpulan yang didapat dari penelitian ini adalah :

1. Gabungan *Dynamic Pathfinding Algorithm* dan Algoritma A* dapat diimplementasikan pada *Car Racing Game*.
2. NPC yang menggunakan gabungan *Dynamic Pathfinding Algorithm* dan Algoritma A*

mendapatkan hasil yang lebih baik dari NPC yang hanya menggunakan Algoritma DPA pada lintasan perlombaan kosong dan lintasan perlombaan yang memiliki *obstacle*.

3. Metode representasi *grid* yang digunakan berpengaruh pada hasil rute yang didapatkan oleh Algoritma A* serta posisi obstacle dan bentuk lintasan juga berpengaruh besar pada DPA.

Saran yang didapat untuk pengembangan lebih lanjut pada penelitian ini adalah :

1. Penelitian selanjutnya dapat menggunakan representasi map yang lain.
2. Diperlukan pengembangan algoritma kecerdasan buatan yang lebih baik lagi agar NPC bisa melalui lintasan perlombaan dengan kecepatan yang berubah-ubah.

5. DAFTAR PUSTAKA

CUI, X & SHI, H (2011). A*-based Pathfinding in Modern Computer Games. International Journal of Computer Science and Network Security (IJCSNS) 11 (1). (pp. 125-130).

CUI, X & SHI, H. (2012). An Overview of Pathfinding in Navigation Mesh. International Journal of Computer Science and Network Security (IJCSNS) 12 (12). (pp. 48-51).

KHANTANAPOKA, K. & CHINNASARN, K. (2009). Pathfinding of 2D & 3D Game Real-Time Strategy with Depth Direction A* Algorithm for Multi-Layer. Eight International Symposium on Natural Language Processing. (pp. 184-188).

MILINGTON, I. & FUNGE, J. (2009). Artificial Intelligence for Games. Morgan Kauffman Publishers, Burlington, USA.

RAM, A., ONTANON, S. & MEHTA, M. (2007). Artificial Intelligence For Adaptive Computer Games. Twentieth International Flairs Conference On Artificial Intelligence. American Association For Artificial Intelligence (Aaai).

SALEN, K. & ZIMMERMAN, E. (2004). Rules of Play: Game Design Fundamentals. The MIT Press Cambridge, London, England.

TAN, C. I., CHEN, C. M., TAI, W. K., & YEN, S. J. (2008). An AI Tool: Generating Paths for Racing Game. International Conference on Machine Learning and Cybernetics 6. (pp. 3132-3137).

WANG, J. Y. & LIN, Y. B. (2010). An Effective Method of Pathfinding in a Car Racing Game. The 2nd International Conference on Computer and Automation Engineering (ICCAE) 3. (pp. 544-547).

WANG, J. Y. & LIN, Y. B. (2012). Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms. International Journal of Machine Learning and Computing 2 (1). (pp. 13-18).