BAB 2
TINJAUAN PUSTAKA

2.1 Animasi 3D

Animasi merupakan salah satu bentuk media visual yang efektif dalam menyampaikan informasi dengan cara yang menarik dan mudah dimengerti oleh audiens. Penggunaan animasi sangat luas, baik untuk keperluan seperti iklan, pengumuman, pembelajaran, *game* dan lain sebagainya. Animasi juga melibatkan teknik menyusun gambar-gambar sehingga menciptakan ilusi gerakan, sehingga memberikan pengalaman visual yang dinamis dan menarik bagi pengguna [15].

Animasi yang paling diminati saat ini adalah yang memiliki kualitas grafis yang lebih tinggi, terutama animasi 3D. Perkembangan teknologi telah memungkinkan pembuatan animasi dengan detail grafis yang sangat baik, seperti yang ditemukan dalam animasi 3D [16]. Animasi 3D dalam komputer menggambarkan gambar dengan kedalaman dengan dimensi X, Y, dan Z, meskipun pada dasarnya tetap merupakan representasi 2D di layar kaca. Berbeda dengan animasi 2D yang datar, animasi 3D memiliki volume dan bentuk yang lebih nyata dan dapat dilihat dari berbagai sudut pandang [15].

2.2 Blender 3D

Blender adalah perangkat lunak desain 3D gratis dan *open source* yang mendukung seluruh proses pembuatan 3D, termasuk pemodelan, *rigging*, animasi, simulasi, *rendering*, *compositing*, dan *motion tracking*. Selain itu, Blender juga dapat digunakan untuk mengedit video dan membuat *game*. Blender dapat dijalankan di berbagai platform seperti Linux, Windows, dan Macintosh, dan menggunakan antarmuka OpenGL untuk memberikan pengalaman yang seragam kepada pengguna. Untuk memastikan kompatibilitas dengan berbagai platform, Blender secara teratur diuji oleh tim pengembangan [17].

2.3 Unity

Unity adalah sebuah perangkat lunak pembuat permainan yang dibuat oleh Unity Technologies. Perangkat lunak ini pertama kali dirilis pada tahun 2005 dan telah menjadi salah satu dari banyak pilihan yang digunakan oleh para pengembang permainan profesional di seluruh dunia. Unity membantu dalam membuat permainan dengan fitur rendering yang sudah disediakan di dalamnya. Dengan kemampuan dan kecepatannya, Unity dapat membuat program interaktif tidak hanya dalam 2 dimensi, tetapi juga dalam 3 dimensi [17]. Unity dirancang khusus untuk mendukung penggunaan plugin dari perangkat lunak pihak ketiga. Selain itu, Unity memiliki toko aset sendiri yang menyediakan beragam plugin yang dibutuhkan oleh para pengembang permainan. Toko aset ini menyediakan plugin yang dibuat oleh berbagai pengembang, untuk keperluan pengembangan permainan, yang dapat digunakan oleh pengembang lainnya [18].

Unity adalah *game engine* yang mendukung berbagai platform seperti Android, iOS, PC, dan lainnya. Terdapat dua versi untuk Unity, yaitu versi gratis dan berbayar (Pro), tetapi keduanya memiliki fitur untuk menciptakan permainan berkualitas tinggi [17]. Adapun fitur-fitur dari Unity antara lain adalah [19]:

1. IDE atau Integrated Development Environment adalah lingkungan pengembangan yang terpadu.
2. Multiplatform.
3. Engine grafis menggunakan berbagai teknologi seperti Direct3D untuk Windows, OpenGL untuk Mac dan Windows, OpenGL ES untuk iOS, serta API propietary untuk Wii.
4. Pemrograman permainan dilakukan melalui Mono, sebuah platform open source yang dapat diakses melalui UnityScript (mirip JavaScript), C#, atau Boo (mirip Python).

2.4 Navigation (*NavMesh*)

Navigation System memungkinkan pengembang untuk membuat karakter yang cerdas dalam bergerak di permainan. Navigation System menggunakan navigasi Meshes (jaring) untuk membentuk suatu area navigasi pada lingkungan permainan. Jaring navigasi dibentuk secara otomatis dari geometri sesuai dengan lingkungan dalam game. Adanya benda penghalang dalam permainan seperti kursi, batu, pohon memungkinkan mengubah navigasi karakter ketika berjalan. Sehingga secara otomatis karakter akan melewati benda penghalang untuk menuju ke titik destinasi yang telah ditentukan. Navigation System bertujuan untuk mengontrol pergerakan karakter dan memberikan kemampuan karakter mengambil keputusan seperti melewati tangga untuk mencapai lantai dua atau melompat untuk melewati beberapa lubang. Navigation System terdiri dari beberapa bagian, yaitu :

1. NavMesh singkatan dari Navigation Mesh, adalah struktur data yang menggambarkan permukaan yang dapat dilalui karakter dari permainan dan memungkinkan untuk menemukan jalan dari suatu lokasi ke lokasi yang dituju.
2. NavMesh Agent Komponen bantuan untuk membuat karakter dapat menghindari satu sama lain sambil bergerak menuju tujuan mereka.
3. Off-Mesh Link Komponen yang digunakan untuk pergerakan seperti melompati pagar, membuka pintu sebelum melewatinya dll.
4. NavMesh Obstacles Komponen untuk mendeskripsikan sebuah hambatan sehingga karakter dapat menghindari dan menemukan jalur lain untuk sampai tujuan.

2.5 Algoritma A-Star

Algoritma A* (dibaca A-star) adalah algoritma pencarian jalur yang digunakan untuk menemukan rute terpendek antara titik awal dan tujuan. Algoritma A*, yang pertama kali diperkenalkan oleh Peter Hart, Nils Nilsson, dan Bertram Raphael pada tahun 1968, telah mendapatkan perhatian luas karena keunggulannya dalam menyelesaikan masalah pencarian jalur [3]. Algoritma ini banyak digunakan dalam eksplorasi peta untuk menemukan jalur

optimal yang harus diambil. A* memanfaatkan pencarian *Best First Search* (BFS) untuk menemukan jalur dengan Cost/bobot terkecil, dan menggunakan nilai heuristik sebagai parameternya. Heuristik mengacu pada jarak garis lurus sebenarnya dari titik awal ke tujuan. Algoritma A* menggabungkan nilai g(n), yang mewakili Cost/bobot yang terakumulasi dari titik awal ke titik berikutnya dan nilai h(n) yang mewakili nilai heuristik [20]. Pernyataan tersebut dapat diungkapkan secara matematis sebagai berikut:

$$F(n) = g(n) + h(n)$$

Nilai $F(n)$ mewakili estimasi jalur terpendek yang sudah ditemukan. Sementara $g(n)$ (Cost geografis) merupakan total jarak yang diperoleh dari titik awal ke titik saat ini (sampai sejauh ini). Nilai $h(n)$ (Cost heuristik) adalah jarak yang diestimasi dari titik saat ini ke titik tujuan. Fungsi heuristik digunakan untuk memperkirakan seberapa jauh jalur yang akan diambil ke titik tujuan [20].

Algoritma A* sering digunakan dalam permainan papan dan permainan strategi karena kemampuannya yang stabil dan cepat, yang semakin meningkat dalam beberapa tahun terakhir. Meskipun demikian, algoritma ini tidak sempurna dan mungkin membutuhkan bantuan tambahan atau penyesuaian untuk menangani tugas-tugas yang lebih kompleks. Walaupun A* biasanya lebih cepat dalam menemukan jalur daripada algoritma lainnya, tetapi tidak selalu menjamin bahwa jalur yang ditemukan adalah yang terpendek. Sebuah penelitian menunjukkan bahwa dalam beberapa situasi, seperti pada peta strategi dan labirin, A* hanya berhasil menemukan jalur terpendek sekitar 85% dari waktu. Ini menegaskan pentingnya mempertimbangkan keterbatasan dan potensi kekurangan dari algoritma A* dalam penggunaannya pada aplikasi tertentu [21].

Tahapan Algoritma A*

Berikut adalah tahapan-tahapan algoritma A*:

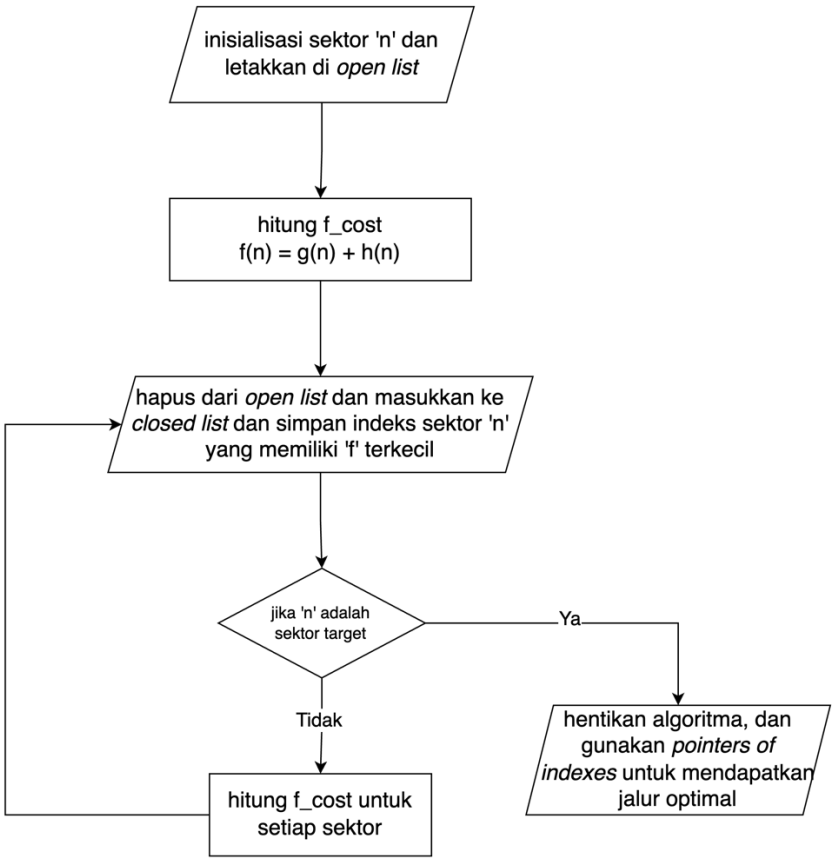
1. Inisialisasi

1. Buat daftar terbuka (*open list*) untuk menyimpan node yang akan dijelajahi.
2. Buat daftar tertutup (*closed list*) untuk menyimpan node yang telah dijelajahi.
3. Tambahkan node awal ke *open list*.
4. Atur g_cost node awal menjadi nol.
5. Hitung nilai Heuristik (h_cost) node awal dan atur nilai f_cost = g_cost + h_cost.

2. Pemeriksaan Kondisi Berhenti
   6. Jika *open list* kosong, berarti tidak ada jalur yang ditemukan.
   7. Jika node tujuan terdapat dalam *closed list*, berarti jalur terpendek telah ditemukan.
3. Pemrosesan *Current Node*
   8. Pilih node dengan nilai f_cost terkecil dari *open list*. Node ini disebut sebagai *current_node*.
   9. Pindahkan *current_node* dari *open list* ke *closed list*.
   10. Untuk setiap node tetangga (*neighbor*) dari *current_node*:
       1. Hitung Cost sementara (g_cost_tentative) untuk mencapai *neighbor* dari titik awal, yaitu g_cost dari *current_node* ditambah Cost untuk berpindah dari *current_node* ke *neighbor*.
       2. Jika *neighbor* tidak ada dalam *closed_list* atau g_score_tentative lebih kecil dari g_cost dan *neighbor* yang ada di *closed list*:
          1. Perbaharui g_cost dari *neighbor* menjadi g_cost_tentative.
          2. Hitung nilai heuristik (h_cost) untuk *neighbor*.
          3. Hitung nilai f_cost untuk *neighbor* menjadi g_cost + h_cost.
          4. Jika *neighbor* tidak ada dalam *open list*, tambahkan ke *open list*.
          5. Jika *neighbor* ada dalam *open list*, perbaharui posisinya dalam *list* jika f_cost yang baru lebih kecil.

4. Ulangi

11. Ulangi tahap 2 dan 3 hingga kondisi berhenti terpenuhi.

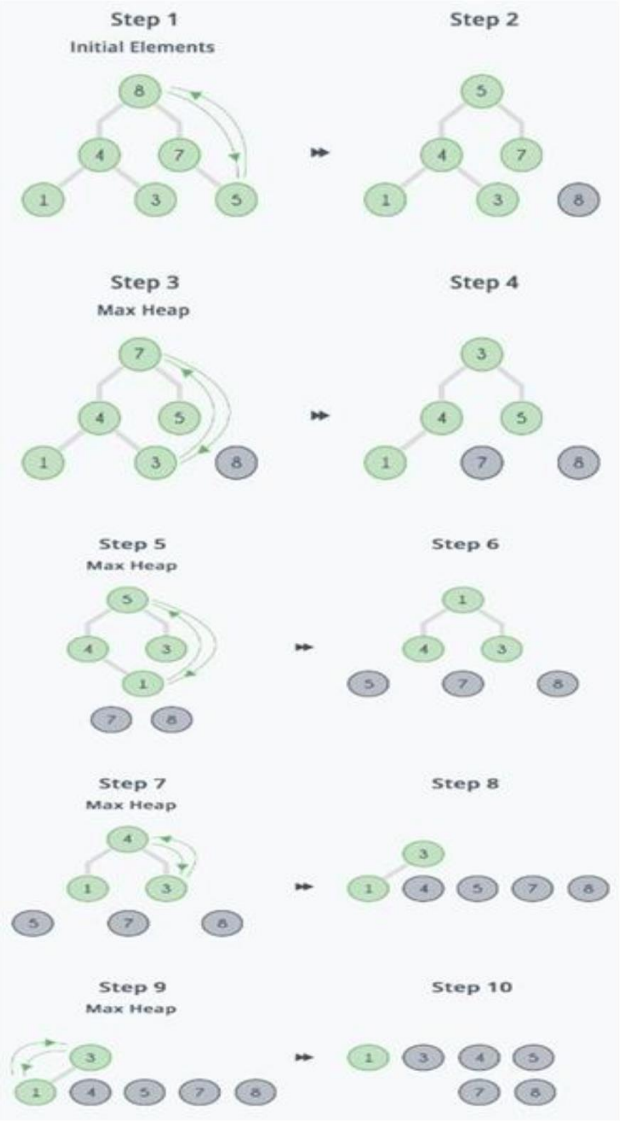


Gambar 2.1 Tahapan Algoritma A-Star

2.6 HeapSort

Heap Sort adalah algoritma pengurutan data berdasarkan perbandingan, dimana larik masukan dianggap sebagai Complete Binary Tree (CBT) dan diubah menjadi struktur data heap. Proses pengurutan dimulai dengan membangun heap, kemudian data dengan nilai tertinggi dipindahkan ke posisi terakhir dalam larik terurut. Proses ini diulang sampai seluruh data terurut. Meskipun lebih lambat daripada quick sort, heap sort memiliki kompleksitas algoritma n log n pada kasus terburuk [22]. Ada dua jenis heap, yaitu maksimum dan minimum. Pada heap maksimum, simpul induk memiliki nilai yang lebih besar dari anak-anaknya, sedangkan pada heap minimum, simpul induk

memiliki nilai yang lebih kecil dari anak-anaknya [23]. Ilustrasi proses heap sort ditunjukkan pada **Gambar 2.2**.



Gambar 2.2 Proses HeapSort [23]

Pada gambar 2.2, dijelaskan bahwa langkah-langkah algoritma heapsort untuk mengurutkan array yang terdiri dari 8 elemen. Elemen array diubah menjadi struktur data heap maksimum. Heap maksimum adalah struktur data pohon di mana setiap elemen memiliki nilai yang lebih besar atau sama dengan nilai anak-anaknya. Langkah ini ditunjukkan di bagian atas, di mana array awal (4, 3, 7, 1, 8, 5, 3, 1) diubah menjadi heap maksimum. Lalu Elemen maksimum dari heap dihapus dan

ditempatkan pada posisi terakhir array yang diurutkan. Pada gambar, elemen maksimum (8) dihapus dari heap dan ditempatkan di posisi ke-7 array yang diurutkan. Heap kemudian diperbaiki kembali untuk mempertahankan sifat heap maksimum. Selanjutnya langkah-langkah diatas diulangi sampai semua elemen heap telah dihapus dan ditempatkan di array yang diurutkan. Pada gambar, langkah ini diulangi dengan elemen 5, 7, 3, 1, 4, dan 3 dihapus dari heap dan ditempatkan di array yang diurutkan. Setelah semua elemen heap telah dihapus, array yang diurutkan akan terisi dengan elemen-elemen dalam urutan menaik. Hasil akhir pada gambar 3 adalah array dengan elemen yang sudah terurut yaitu (1, 1, 3, 3, 4, 5, 7, 8).

2.7 Literatur Review

Adapun review literatur yang menjadi referensi dalam penelitian ini dan memiliki hubungan terkait dengan masalah penelitian dapat dilihat pada tabel 2.1 berikut:

Tabel 2.1 Literatur Review

Review Literatur Pertama [24]	
Sumber Jurnal	Digital Object Identifier
Journal Judul	Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment
Penulis	Gang Tang, Conqiang Tang, Christophe Claramunt, Xiong Hu, Peipei Zhou
Tahun Publikasi	2021
Latar belakang masalah	Latar belakang masalah dalam penelitian ini mencakup evolusi otomatisasi logistik dan peran Automated Guided Vehicle (AGV) dalam transportasi barang di berbagai lingkungan industri. Meskipun AGV telah digunakan secara luas, tantangan muncul dalam perencanaan jalur yang efisien dan bebas tabrakan di lingkungan yang dinamis. Metode pemetaan lingkungan seperti diagram Voronoi dan grafik visibilitas telah diterapkan, namun, algoritma pencarian jalur, terutama A-star, masih memiliki kelemahan. Oleh karena itu, penelitian ini mengusulkan

	pendekatan baru yang mengintegrasikan keunggulan A-star dengan interpolasi B-spline untuk mengatasi kendala jalur yang tidak beraturan dan meningkatkan kelancaran pergerakan AGV. Dengan menanggapi masalah ini, penelitian ini memiliki dampak penting dalam memajukan efisiensi perencanaan jalur AGV di lingkungan industri.
Algoritma yang dipakai	Algoritma A* (A-Star), B-spline
Metodologi Penelitian	Penelitian ini memaparkan suatu metode perencanaan jalur berbasis algoritma geometri A-star, diterapkan pada Automated Guided Vehicle (AGV) untuk mengatasi permasalahan seperti jumlah node yang banyak, jarak jauh, sudut belok besar, serta masalah sawtooth turning path dan jalur silang yang umumnya muncul dalam algoritma A-star tradisional. Langkah pertama melibatkan pemodelan lingkungan port menggunakan metode grid, kemudian dilanjutkan dengan penyaringan node dalam daftar dekat berdasarkan fungsi P (x, y) dan W (x, y). Node yang tidak memenuhi persyaratan dihapus untuk menghindari pembentukan jalur yang tidak teratur. Sambil meningkatkan stabilitas AGV pada jalur belokan, polyline pada jalur belokan digantikan dengan kurva B-spline kubik.
Hasil dan Kesimpulan	Hasil penelitian menunjukkan bahwa, dibandingkan dengan tujuh algoritma lainnya, algoritma geometri A-star menghasilkan pengurangan jumlah node sebesar 10% ~ 40%, sedangkan jumlah belokan berkurang 25%, sudut belok berkurang 33,3%, dan total jarak berkurang 25,5%. Simulasi perencanaan jalur secara keseluruhan mengonfirmasi efektivitas algoritma geometri A-star.
Saran dan Lanjutan	Terdapat beberapa saran untuk penelitian lanjutan dari jurnal ini. Pertama, hingga saat ini, dimensi AGV belum dianggap sebagai fitur waktu nyata dan mungkin akan diakomodasi dalam perhitungan penghindaran obstacle. Kedua, algoritma ini utamanya diterapkan pada lanskap statis, dan pengaruh pergerakan kendaraan dalam pemandangan dinamis belum diperhitungkan. Studi lebih lanjut harus lebih mendalam dalam mengeksplorasi dynamic scene yang kompleks, yang memiliki relevansi signifikan dalam konteks dunia nyata. Ketiga, karena

	gerakan aktual AGV cenderung berupa garis lurus utama, kemampuan terpantul sepenuhnya tidak relevan. Terakhir, penelitian ini memiliki rencana untuk menyelaraskan keseluruhan pendekatan dan algoritma dalam konfigurasi eksperimental AGV di lingkungan pelabuhan yang sesungguhnya.
Review Literatur Kedua [6]	
Sumber Jurnal	Jurnal Teknologi Komputer Sean Institute
Journal Judul	Simulasi Pencarian Rute Terpendek dengan Metode Algoritma A* (A-Star)
Penulis	Oslan Juliana Simbolon
Tahun Publikasi	2022
Latar belakang masalah	Latar belakang masalah penelitian ini berpusat pada pencarian jalur terpendek dalam konteks jaringan pengarahan perjalanan, di mana pengarah jalan berupaya menentukan jalur efisien antara titik awal dan tujuan pada kotak-kotak yang dianggap sebagai graf terhubung. Pemilihan representasi graf bertujuan untuk mempermudah pemodelan masalah, di mana jalur terpendek diukur berdasarkan jarak antara kotak-kotak yang saling terhubung. Simulasi didefinisikan sebagai representasi model sistem nyata yang melibatkan pengolahan data dan evaluasi hasil simulasi. Algoritma A* dipilih sebagai metode pencarian jalur terpendek, terutama dalam penggunaannya dengan kotak-kotak berwarna sebagai elemen-elemen graf. Penelitian ini memungkinkan pengguna untuk membuat jalur sendiri dengan mengidentifikasi posisi awal dan tujuan, dan kemudian aplikasi menampilkan jalur terpendek antara kedua titik tersebut. Fokusnya adalah pada interaktifitas simulasi yang melibatkan pemilihan dan navigasi menggunakan kotak-kotak berwarna.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Penelitian ini memaparkan suatu metode perencanaan jalur berbasis algoritma A-star. Algoritma A* menggunakan estimasi jarak terdekat untuk mencapai tujuan (goal) dan

	memiliki nilai heuristik yang digunakan sebagai dasar pertimbangan. Heuristik adalah kriteria, metoda, atau prinsip-prinsip untuk menentukan pilihan sejumlah alternatif untuk mencapai sasaran dengan efektif.
Hasil dan Kesimpulan	Hasil dari penelitian ini berupa implementasi program simulasi untuk mencari rute terpendek dari posisi awal ke posisi tujuan menggunakan bahasa pemrograman Visual 2008. Representasi visual dari graf dibuat dengan menyatakan objek sebagai simpul, dan hubungan antar simpul dijelaskan melalui titik-titik. Program simulasi ini menyediakan kemudahan dalam menentukan jalur terpendek yang akan ditempuh dari posisi awal menuju tujuan.
Saran dan Lanjutan	Terdapat beberapa saran untuk penelitian lanjutan dari jurnal ini. Pertama, hingga saat ini, dimensi AGV belum dianggap sebagai fitur waktu nyata dan mungkin akan diakomodasi dalam perhitungan penghindaran obstacle. Kedua, algoritma ini utamanya diterapkan pada lanskap statis, dan pengaruh pergerakan kendaraan dalam pemandangan dinamis belum diperhitungkan. Studi lebih lanjut harus lebih mendalam dalam mengeksplorasi dynamic scene yang kompleks, yang memiliki relevansi signifikan dalam konteks dunia nyata. Ketiga, karena gerakan aktual AGV cenderung berupa garis lurus utama, kemampuan terpantul sepenuhnya tidak relevan. Terakhir, penelitian ini memiliki rencana untuk menyelaraskan keseluruhan pendekatan dan algoritma dalam konfigurasi eksperimental AGV di lingkungan pelabuhan yang sesungguhnya.
Review Literatur Ketiga 25]	
Sumber Jurnal	Penelitian Ilmu Komputer, Sistem Embedded and Logic
Journal Judul	Android-Based Shortest Path Finding Using A-Star (A*) Algorithm in Bekasi City
Penulis	Herlawati, Prima Dina Atika, Ajif Yunizar Pratama Yusuf, Fata Nidaul Khasanah, Endang Retnoningsih, Beno Aditya Sanusi, Gedhe Hilman Wakhid

Tahun Publikasi	2021
Latar belakang masalah	Latar belakang masalah penelitian adalah terdapat masalah dalam mendapatkan informasi mengenai rute terdekat menuju tempat tertentu bagi pengunjung kota Bekasi. Berdasarkan permasalahan yang ada, penelitian ini mengusulkan sebuah aplikasi untuk mencari informasi tempat-tempat yang ingin dikunjungi pengunjung berdasarkan rute terdekat. Algoritma A-Star (A*) diterapkan yang menggunakan estimasi jarak dengan mencari jalur terdekat ke tujuan menggunakan fungsi heuristik sebagai dasar untuk memilih dari beberapa alternatif secara efektif.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Metode penelitian dalam menentukan bagaimana meningkatkan akurasi pada rute perjalanan dan menerapkan algoritma A-Star (A*) dalam mencari beberapa lokasi terdekat. Dilakukan juga pengumpulan data dengan melakukan pengamatan langsung terhadap objek yang bersangkutan dan mencatat informasi penting seperti koordinat, alamat, dan nama tempat. Pengolahan data, pada tahap ini akan dilakukan perhitungan Algoritma A-Star (A*) dengan menggunakan rumus. Pengujian aplikasi dengan Algoritma A-Star (A*) dilakukan dengan metode Black Box Testing untuk mendapatkan hasil yang diharapkan. Setelah dilakukan pengujian menghasilkan aplikasi android untuk mencari rute terdekat menggunakan algoritma A-Star (A*).
Hasil dan Kesimpulan	Dapat disimpulkan bahwa aplikasi pencarian beberapa tempat terdekat dapat digunakan untuk mendapatkan informasi lokasi seperti restoran, kafe atau mall terdekat dengan mudah dan efisien. Dengan menampilkan daftar tempat terpilih sesuai radius yang dimasukkan, aplikasi juga dapat menampilkan jarak dan rute dari lokasi pengguna (titik awal) ke tempat yang dipilih (titik tujuan). Algoritma A-Star (A*) dapat diterapkan sebagai algoritma komputer yang dapat mencari rute terdekat beserta jarak yang

	ditempuh dalam pencarian lokasi dan rute pada aplikasi berbasis android.
Saran dan Lanjutan	Penelitian selanjutnya akan dikembangkan lebih lanjut untuk mendapatkan tampilan yang lebih menarik, menambahkan menu petunjuk arah agar pengguna tidak hanya melihat tampilan rute, menambahkan menu history agar pengguna dapat menggunakan aplikasi ini dalam mode offline.
Review Literatur Keempat [26]	
Sumber Jurnal	Advances in Social Science, Education and Humanities Research
Journal Judul	The Application of a Star (A*) Algorithm on the Android-Based Pacman Adaptation Educational Game as a Learning Media for SMK
Penulis	Ahfaz Bactiar Febliama, Nimas Dian Fitria, Anik Nur Handayani
Tahun Publikasi	2018
Latar belakang masalah	Latar belakang masalah penelitian ini adalah permainan game yang dapat menstimulasi perkembangan dapat diterapkan sebagai media pembelajaran. Implementasi media pembelajaran game ini untuk siswa SMK memerlukan modifikasi yang membutuhkan Artificial Intelligence (AI) menggunakan algoritma A* supaya game dapat menjadi lebih sulit.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Metode penelitian yang dilakukan yaitu perencanaan jalur berbasis algoritma A-star. Peneliti menentukan berbagai objek dalam game, aturan dan cara main, serta target pemain, dan implementasi algoritma A*.
Hasil dan Kesimpulan	Hasil dari penelitian ini berupa implementasi program game android Pacman yang dimodifikasi dengan nama Go Straight Game. Pada game ini juga terdapat 3 tingkat kesulitan dari musuh. Kesimpulan dari game yang dibuat adalah algoritma A* dapat diterapkan pada permainan pacman. Permainan pacman ini dapat diperbaharui dengan

	adanya soal-soal pembelajaran, sehingga selain memberikan keseruan dapat juga mengasah kecerdasan.
Saran dan Lanjutan	Saran dari penelitian yang telah dilakukan adalah perbaikan pada desain game.
Review Literatur Kelima [3]	
Sumber Jurnal	Journal of Physics: Conference Series
Journal Judul	Application of A-Star Algorithm on Pathfinding Game
Penulis	Ade Candra, Mohammad Andri Budiman, Rahmat Irfan Pohan
Tahun Publikasi	2021
Latar belakang masalah	Latar belakang masalah penelitian ini berpusat pada pencarian jalur terpendek menggunakan Artificial Intelligence (AI) dengan algoritma A* pada permainan penanaman pohon. Algoritma A-Star digunakan untuk menentukan jalur terdekat yang bisa ditempuh musuh ke pohon dan waktu yang dibutuhkan untuk menempuh jalur tersebut. Selanjutnya pemain harus berusaha melindungi pohon tersebut dari musuh dengan cara menyentuh musuh atau menggunakan penghalang agar pohon tersebut tetap hidup.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Metode penelitian yang dilakukan yaitu perencanaan jalur berbasis algoritma A-star dengan Heuristik A-Star yang memungkinkan pergerakan dalam empat arah berbeda.. Algoritma ini akan menunjukkan jalur yang akan dilalui musuh untuk sampai ke pohon tersebut dan waktu yang dibutuhkan untuk menempuh jalur tersebut.
Hasil dan Kesimpulan	Hasil dari penelitian ini yaitu game 2D berbasis android menunjukkan bahwa jumlah node yang dikunjungi akan bertambah jika grid mempunyai banyak node hambatan, dan rata-rata waktu proses pathfinding A-Star adalah 0,0732 detik. Kesimpulan dari game yang dibuat adalah semakin banyak objek yang menjadi penghalang (unwalkable node),

	maka semakin lama waktu proses yang diperlukan untuk mencari rute dari node objek musuh ke node objek pohon.
Saran dan Lanjutan	-
Review Literatur Keenam [27]	
Sumber Jurnal	International Journal Of Information System & Technology
Journal Judul	Comparative Analysis of Pathfinding Algorithms A *, Dijkstra, and BFS on Maze Runner Game
Penulis	Silvester Dian Handy Permana, Ketut Bayu Yogha Bintoro, Budi Arifitama, Ade Syahputra
Tahun Publikasi	2018
Latar belakang masalah	Latar belakang masalah penelitian ini adalah pada Game Maze Runner yang memerlukan algoritma pathfinding untuk sampai ke tujuan dengan jalur terpendek. Algoritma ini digunakan pada NPC yang akan berpindah dari node awal ke node tujuan. Namun penggunaan algoritma yang salah dapat mempengaruhi lamanya proses komputasi untuk mencari jalur terpendek. Semakin lama proses komputasi, maka semakin lama pula pemain harus menunggu. Penelitian ini membandingkan algoritma pathfinding A*, Dijkstra, dan Breadth First Search (BFS) pada game Maze Runner.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Metodologi penelitian yang digunakan dalam penelitian ini adalah uji komparatif dengan menggunakan 3 level pada permainan Maze Runner dengan posisi rintangan yang sama untuk setiap algoritma yang diuji. Hal yang diukur dalam uji perbandingan ini adalah waktu proses, panjang jalur, dan berapa banyak balok yang dimainkan pada proses komputasi yang ada. Dalam penelitian ini peneliti akan menggunakan 3 level yang berbeda dengan blok Tetris yang berbeda pula pada setiap levelnya. Setelah menyelesaikan pengujian dengan 3 level berbeda, maka dapat diputuskan algoritma mana yang terbaik untuk diterapkan pada game Maze Runner ini.

Hasil dan Kesimpulan	Hasil dari penelitian menunjukkan bahwa A*, Dijkstra, dan Breadth First Search dapat digunakan untuk mencari yang rute terpendek dalam Game Maze Runner. A* merupakan algoritma terbaik dalam pathfinding khususnya pada permainan Maze/grids. Hal ini didukung dengan proses komputasi yang dibutuhkan minimal dan waktu pencarian yang relatif singkat. Penggunaan algoritma yang tepat dapat membuat game menjadi lebih baik dari segi proses komputasi, penggunaan memori, dan waktu komputasi.
Saran dan Lanjutan	Saran dari penelitian ini adalah melanjutkan penelitian dengan algoritma A* yang dimodifikasi untuk mendapatkan hasil komputasi yang lebih baik. Kemudian diakukan juga tes pada A* untuk menemukan jalur terpendek dalam kasus lain seperti permainan balap, strategi, olah raga, dll.
Review Literatur Ketujuh [2]	
Sumber Jurnal	Highlights in Science, Engineering and Technology
Journal Judul	Research on the A Star Algorithm for Finding Shortest Path
Penulis	Yumeng Yan
Tahun Publikasi	2023
Latar belakang masalah	Latar belakang masalah penelitian ini adalah pencarian jalur secara bertahap menggunakan intelligence menjadi penting, karena dapat mengoptimalkan rute robot dan masalah pergerakan NPC dalam game dapat ditingkatkan. Di antara banyak algoritma jalur terpendek, algoritma A-star untuk mencari jalur merupakan salah satu cara yang sangat klasik. Dalam tesis ini, penulis terutama mempelajari prinsip algoritma A star dan membandingkannya dengan algoritma Dijkstra.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Metodologi penelitian yang digunakan dalam penelitian ini adalah pencarian jalur berbasis algoritma A-star Heuristic dan dibandingkan dengan algoritme Dijkstra. Dengan menggunakan fungsi heuristic, maka dapat diterapkan konsep Manhattan distance yang memungkinkan pergerakan ke empat arah, yaitu atas, bawah, kanan, dan

	kiri. Implementasi lingkungan dan setting rintangan serta titik awal dilakukan melalui metode Raster. Simulasi A* yang diimplementasikan diuji menggunakan Matlab.
Hasil dan Kesimpulan	Hasil simulasi algoritma A star menggunakan Matlab menunjukkan hasil yang lebih baik untuk pencarian jalur terpendek dibanding algorimte Dijkstra.
Saran dan Lanjutan	-
Review Literatur Kedelapan [9]	
Sumber Jurnal	IEESE Internasional Journal of Science and Technology (IJSTE)
Journal Judul	Implementation Method A * (A-Star) For NPC Enemies In 3D Game Vocabulary Learning Arabic
Penulis	Badzrotul Mufida
Tahun Publikasi	2016
Latar belakang masalah	Latar belakang masalah penelitian ini adalah bahasa Arab khususnya bagi umat islam merupakan bahasa yang sangat istimewa, karena kitab untuk seluruh umat Islam menggunakan bahasa Arab. Untuk itu dibuatlah sebuah game 3D belajar kosakata bahasa Arab yang ditujukan untuk pemula. Dengan metode pembelajaran seperti ini diharapkan dapat meningkatkan keinginan belajar masyarakat yang ingin memulai belajar bahasa Arab. Dalam game ini pemain akan dibawa ke sebuah pulau fantasi di dalam pulau yang terdapat karakter-karakter yang membentuk populasi pulau dan jumlah buah-buahan. Dan pemain harus mengantarkan karakter tersebut ke dalam karakter rumah masing-masing.
Algoritma yang dipakai	Algoritma A* (A-Star)
Metodologi Penelitian	Metodologi penelitian yang digunakan dalam pembuatan game ini adalah algoritma A* dan Unity 3D untuk membangun video game. Game ini ditujukan untuk penggunaan pada perangkat android.
Hasil dan Kesimpulan	Hasil penelitian menunjukkan penggunaan algoritma A* berhasil diterapkan untuk menghasilkan perilaku pencarian

	NPC. Dalam proses pencariannya, player mampu melewati segala rintangan dan berhasil menemukan keberadaan lokasi populasi NPC tersebut. Game "Browse kampoeng arab" telah diuji ke sembilan sistem operasi berbeda pada platform android, di berbagai perangkat periferal game ini menunjukkan tingkat keberhasilan yang berbeda-beda dalam pengujian sistem dan tampilan.
Saran dan Lanjutan	-
Review Literatur Kesembilan [28]	
Sumber Jurnal	Jurnal Mantik
Journal Judul	Implementation of A Star Algorithm in Android Based Alien Shooter Games
Penulis	Eka Febriyanto Riski, Agung Triayudi, Eri Mardiani
Tahun Publikasi	2020
Latar belakang masalah	Latar belakang masalah membahas signifikansi dan evolusi Game First Person Shooter (FPS) dalam mengatasi stres di era modern. Dalam masyarakat yang terbebani oleh berbagai aspek kehidupan, permainan seperti FPS menjadi salah satu alternatif efisien untuk meredakan tekanan pikiran. Berkembang dari permainan tradisional menjadi bentuk digital dan 3 Dimensi (3D), game telah melampaui sekadar hiburan dan menjadi industri besar dengan peran yang meluas, termasuk sebagai media pembelajaran dan lahan usaha. Kemajuan teknologi telah memperkaya jenis dan variasi permainan, membawa dampak positif terutama dalam mengatasi stres. Oleh karena itu, penelitian ini menciptakan game 3D berjudul "Implementasi Algoritma A Star pada Game Alien Shooter Berbasis Android" untuk mengeksplorasi penerapan algoritma A Star dalam konteks permainan tembak-menembak alien di platform Android. Kesimpulannya, permainan modern tidak hanya sekadar hiburan tetapi juga menjadi bagian integral dari kehidupan sehari-hari, menyediakan pengalaman bermanfaat dan diversifikasi dalam menghadapi tekanan kehidupan.
Algoritma yang dipakai	Algoritma A* (A-Star), Best Fit Serch (BFS)

Metodologi Penelitian	Metode penelitian dalam jurnal ini memperkenalkan algoritma A* (A Star), yang menggabungkan prinsip Best First Search (BFS), Uniform Cost Search, dan Greedy Best First Search. Dalam notasi matematika, Cost total (f(n)) dihitung dengan menambahkan Cost sebenarnya (g(n)) dan perkiraan Cost (h(n)), ditulis sebagai f(n) = g(n) + h(n). Algoritma A* menjadi lengkap dan optimal melalui perhitungan Cost ini. Dalam situasi pencarian rute tanpa hambatan, A* bekerja secepat dan seefisien BFS. Ketika terdapat rintangan, A* mampu menemukan solusi rute tanpa terjebak oleh hambatan tersebut. Metode ini menggunakan grafik untuk memperluas ruang status dan memiliki prinsip pencarian yang mirip dengan BFS, dengan penambahan dua faktor utama: Setiap node memiliki Cost unik antar node, yaitu besarnya Cost yang dikeluarkan dari satu node ke node lainnya. Cost dari setiap node ke node tujuan dapat diperkirakan, memberikan bantuan dalam pencarian untuk mengurangi kemungkinan kesalahan arah. Cost tidak terbatas pada jarak dan dapat mencakup waktu tempuh, seperti saat mencari jalur tercepat dengan kendaraan. Algoritma A* mengurutkan node dalam daftar 'terbuka' berdasarkan Cost keseluruhan, menggunakan antrian prioritas. Node dalam daftar 'tertutup' dapat ditambahkan ke daftar 'terbuka' jika jalur terpendek ke node tersebut ditemukan. Dengan urutan prioritas, algoritma memeriksa node-node dengan perkiraan pengeluaran terkecil terlebih dahulu, meningkatkan keefisienan pencarian. Keseluruhan, semakin baik perkiraan Costnya, semakin cepat algoritma

	menemukan solusi, dan Cost serta perkiraan Cost dapat disesuaikan sesuai kebutuhan.
Hasil dan Kesimpulan	Hasil dari penelitian ini sesuai denga napa yang diharapkan. Dimana NPC dapat menemukan pemain yang memasuki wilayah NPC.
Saran dan Lanjutan	Saran untuk pengembangan penelitian ini yaitu membuat jarak radius dengan Player. Sehingga Ketika Player berada di radius NPC, makan script A* akan aktif, jika Player di luar radius NPC, maka script A* tidak akan aktif dan NPC akan bergerak secara random tanpa adanya titik akhir.
Review Literatur Kesepuluh [29]	
Sumber Jurnal	IEEE International Conference on Computer and Information Technology
Journal Judul	Optimized Application and Practice of A* Algorithm in Game Map Path-Finding
Penulis	Huilai Zou, Lili Zong, Hua Liu, Chaonan Wang, Zening Qu, Youtian Qu
Tahun Publikasi	2010
Sumber Jurnal	Journal Of Information Technology And Its Utilization
Journal Judul	Implementation Of Collision Avoidance System Algorithm In Npc Game 3D
Penulis	Gugah Alwan Hamanako, Adi Sucipto
Tahun Publikasi	2023
Latar belakang masalah	Latar belakang masalah dalam jurnal ini membahas mengenai Sistem Operasi (OS) smartphone kebanyakan adalah Android dan IOS, dan saat ini yang paling populer adalah Android dengan berbagai game yang dapat diakses dengan mudah. Oleh karena itu dibutuhkan Artificial Intelligence dari Non-Player Character (NPC) yang dapat menghindari setiap rintangan dengan menggunakan algoritma Collision Prevention System.

Algoritma yang dipakai	Collision Prevention System
Metodologi Penelitian	Metode penelitian dalam jurnal ini menggunakan Collision Prevention System untuk game Balap Karung 3D.
Hasil dan Kesimpulan	Hasil dari penelitian ini adalah sebuah aplikasi game Android yang menerapkan Sistem Penghindaran Tabrakan yang dapat membuat AI NPC menghindari rintangan. Penerapan algoritma pada NPC dilakukan dalam 3 tahap yaitu perancangan dengan membuat flowchart algoritma, kemudian penulisan program dari Collision Prevention System, dan terakhir pengujian AI NPC. Pengujian dilakukan dengan membandingkan reaksi NPC 1 dan NPC 2 dalam melewati rintangan ketika berada pada jalur yang berbeda dengan dilakukan secara bersama-sama antar NPC. Berdasarkan pengujian pada AI NPC, NPC berhasil menghindari rintangan di depannya dan pemain, dan hampir 93% berhasil.
Review Literatur Kesebelas [30]	
Sumber Jurnal	Jurnal Widya
Journal Judul	Implementasi Algoritma Heapsort dalam Game Pembelajaran Algoritma Sorting
Penulis	Fenisa Lourence Tobing, Fenina Adline Twince Tobing, Jimmy Peranginangin
Tahun Publikasi	2022
Latar belakang masalah	Banyak pelajar Teknik Informatika yang baru pertama kali belajar pemrograman mengalami kesusahan untuk mempelajari algoritma dan mengalami kesulitan belajar pemrograman karena kekurangan bahan pembelajaran dan instruksi personal, yang mengakibatkan tingkat *drop-out* yang tinggi .Dari permasalahan tersebut, ada salah satu cara untuk menyelesaikan permasalahan ini adalah dengan mengganti metode pembelajarannya. Metode yang dapat dilakukan adalah membuat sebuah proses pembelajaran yang fun untuk pelajar, misalnya dengan membuat proses

	pembelajaran tersebut menjadi suatu permainan, atau disebut juga *learning through playing*. Implementasi pembelajaran algoritma sorting, terutama HeapSort, dapat dibuat menarik dengan dijadikan sebuah permainan kartu. Dengan ini diharapkan agar pembelajaran pemrograman dapat menjadi lebih menarik dan lebih mudah bagi pelajar yang akan mendalami materi bersangkutan
Algoritma yang dipakai	Heapsort
Metodologi Penelitian	Program ini dirancang untuk pembelajaran melalui permainan kartu menggunakan Heapsort. Proses perancangan program bantu pembelajaran dilakukan dalam beberapa tahap melalui algoritma heap sort menurut McMillan. Tahap pertama, hilangkan node pada root (posisi teratas). Lalu, tahap kedua yaitu pindahkan node pada posisi terakhir ke root.
Hasil dan Kesimpulan	Dari hasil simulasi yang sudah dilaksanakan, algoritma Heapsort ini telah berfungsi dan dapat diimplementasikan dalam bentuk permainan kartu sebagai bahan pembelajaran materi sorting dalam pemrograman. Dengan diimplementasikannya rancangan algoritma ini, pembelajaran pemrograman pada materi sorting dapat menjadi lebih mudah dan menarik bagi pelajar yang akan mendalaminya.
Saran dan Lanjutan	-

2.8 Unified Modelling Language (UML)

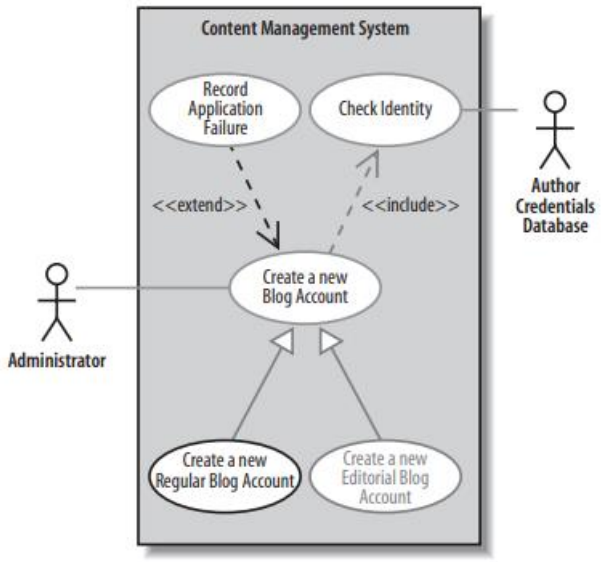
Unified Modelling Language (UML) merupakan istilah dari bahasa penulisan spesifikasi standar cetak biru perangkat lunak UML digunakan untuk menspesifikasikan, mendokumentasikan, dan membangun sistem perangkat lunak [31].

UML merupakan metodologi yang dikembangkan oleh sistem Pemrograman Berorientasi Objek (OOP) dan sekelompok perangkat tool

sebagai pendukung dari sistem tersebut. Bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, mendokumentasikan suatu sistem informasi. UML menyediakan beberapa diagram untuk memodelkan suatu objek aplikasi di antaranya :

2.8.1 Use case Diagram

Use case Diagram mendeskripsikan fungsional sistem yang memodelkan dari perspektif dunia luar sistem. *Use case* Diagram merupakan gambaran atau representasi dari interaksi antara sistem dengan lingkungannya. *Use case* menjelaskan apa yang sistem kita harus kerjakan (fungsional). Satu use case dapat mewakili satu kebutuhan user. *Use case* Diagram terdiri dari beberapa notasi yaitu Aktor, Use case, Association, dan System Boundary [31].



Gambar 2.3 Contoh Use Case Diagram

Sumber : Russ Miles Kim Hamilton, Buku Learning UML 2.0, O'Reilly Media, 2006

2.8.2 Use case Scenario

Use case scenario merupakan hasil instansi dan penjelasan dari setiap use case. Ada beberapa komponen yang perlu dipahami dalam use case scenario, yang diantaranya adalah [31]:

5. Name, memberikan penjelasan singkat tentang nama use case.
6. Actors, daftar aktor yang dapat mengakses use case.
7. Goals, menjelaskan apa yang aktor coba untuk dapatkan dari use case.
8. Precondition, kondisi sistem sebelum use case dijalankan.
9. Summary, memberikan penjelasan singkat tentang deskripsi informal sesuai use case.
10. Related use case, daftar use case yang berhubungan dengan use case tersebut.
11. Steps, menjelaskan setiap langkah yang dijalankan pada use case tersebut.
12. Post condition, kondisi sistem setelah use case dijalankan.

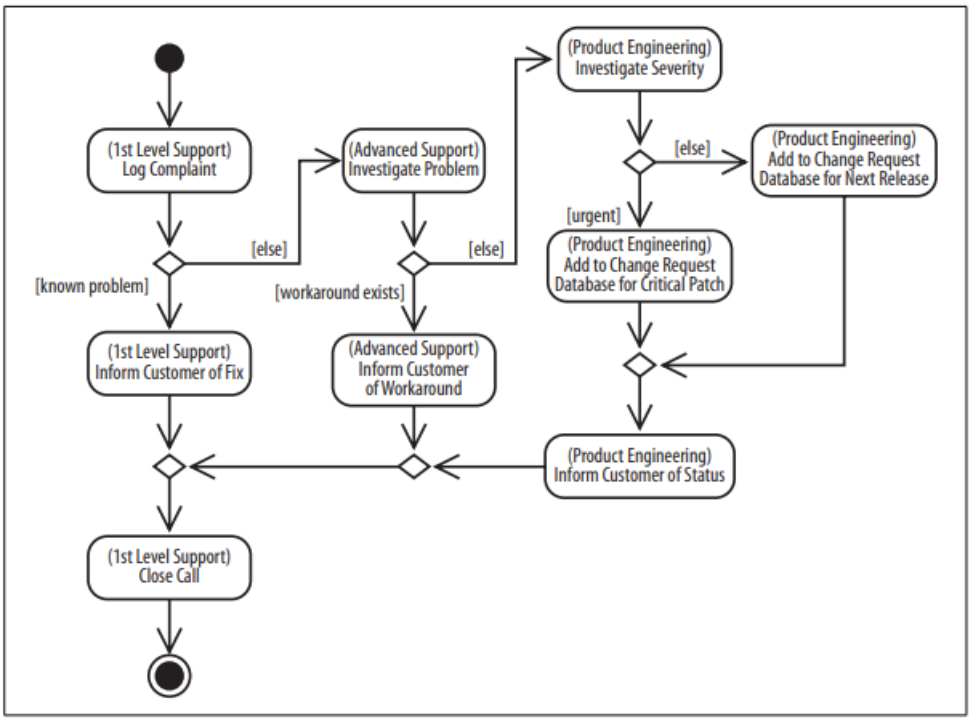
Use case name	Membuat account baru blog editorial	
Related Requirements	Requirement A.1.	
Goal In Context	Penulis baru atau permnitaan account baru blog editorial oleh administrator	
Preconditions	Penulis memiliki kesesuaian bukti identitas	
Succesfull End Condition	Account baru editorial dibuat untuk penulis	
Failed End Condition	Aplikasi gagal membuar Account baru editorial	
Primary Actors	Administrator	
Secondary Actors	None	
Trigger	Administrator meminta CMS untuk membuar akun editorial baru yang akan memungkinkan untuk mengedit entri dalam satu set blog	
Base Use Cases	Membuat account baru blog editorial	
Main Flow	Step	Action
	1	Meminta administrator sistem membuat account baru blog
	2	Administrator memilih jenis editorial account
	3	Administrator memasukan data detail penulis
	4	Administrator memilih blog yang memiliki hak akses editorial account
	5 Include :: Periksa identitas	Memeriksa detail data penulis
	6	Editorial account baru dibuat
	7	Ringkasan rincian rekening editorial baru diemail ke penulis
Extensions	Step	Branching Action
	5.1	Penulis tidek diperbolehkan untuk mengedit blog yang ditunjukan
	5.2	Aplikasi menolah account editorial

Gambar 2.4 Contoh Use Case Scenario

Sumber : Russ Miles Kim Hamilton, Buku Learning UML 2.0, O'Reilly Media, 2006

2.8.3 Activity Diagram

Activity Diagram merupakan bentuk visual dari diagram alur yang berisi aktivitas dan tindakan juga dapat berisi pilihan, pengulangan, dan concurrency. *Activity* Diagram memberikan gambaran yang lebih baik tentang langkah – langkah dari Use case. Komponen utama dari *Activity* Diagram yaitu aksi. Diantara aksi – aksi suatu sistem ini mirip dengan diagram alir kecuali diagram aktivitas dapat menunjukan aliran – aliran konklusi. Notasi diagram aktivitas (aksi) terdiri dari beberapa simbol untuk membuat aksi – aksi tersebut. Diantaranya, simbol awal dan akhir, transisi, dan aksi [31].

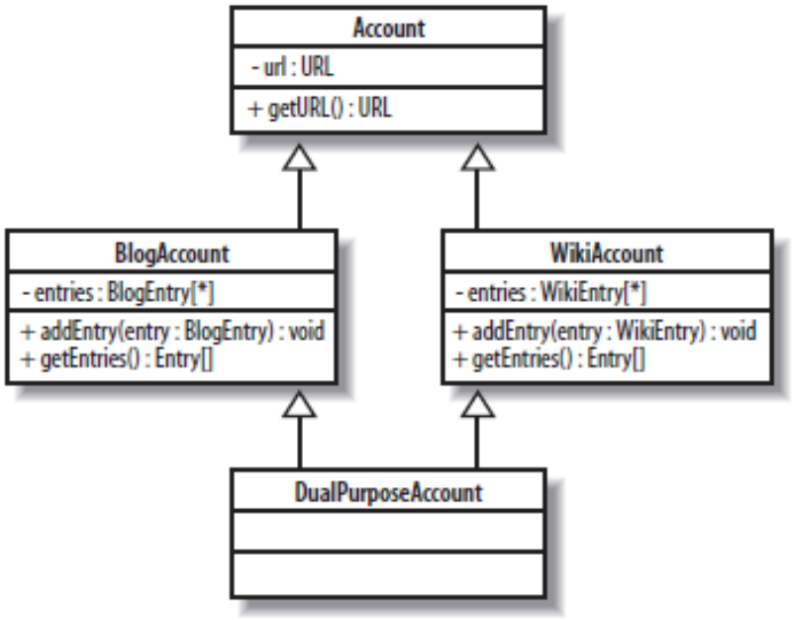


Gambar 2.5 Contoh Activity Diagram

Sumber : Russ Miles Kim Hamilton, Buku Learning UML 2.0, O'Reilly Media, 2006

2.8.4 Class Diagram

Class Diagram merupakan salah satu diagram UML yang menggambarkan kelas – kelas dalam sebuah sistem dan hubungannya antara satu dengan yang lainnya. Serta dimasukan sebagai atribute dan operasi. *Class* diagram memperlihatkan hubungan antar kelas baik private atau publik dan penjelasan detail tiap – tiap kelas dalam model desain dari suatu sistem. Proses analisis dalam *class* diagram memperlihatkan aturan – aturan dan tanggung jawab terhadap entitas yang dibuat [31].



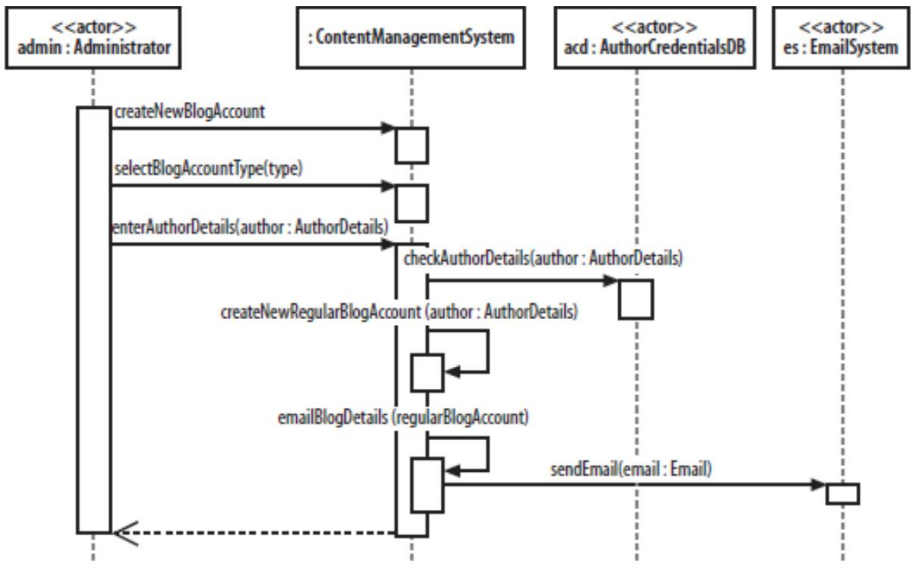
Gambar 2.6 Contoh Class Diagram

Sumber : Russ Miles Kim Hamilton, Buku Learning UML 2.0, O'Reilly Media, 2006

2.8.5 Sequence Diagram

Sequence Diagram menggambarkan interaksi objek didalam dan di sekitar sistem berupa pesan yang digambarkan terhadap waktu. Sequence diagram mendefinisikan urutan kejadian yang dapat menghasilkan output yang diinginkan. Sequence diagram mirip dengan activity diagram menggambarkan

alur kejadian dan aliran data namun kurang mampu dalam menjelaskan detail dari sebuah algoritma [31].



Gambar 2.7 Contoh Sequence Diagram

Sumber : Russ Miles Kim Hamilton, Buku Learning UML 2.0, O'Reilly Media, 2006

2.9 Scenario Pengukuran Parameter

Penelitian ini bertujuan untuk mengukur efektivitas dan efisiensi algoritma A* yang dioptimasi dengan heapsort dalam meningkatkan kinerja pencarian jalur (pathfinding) dalam game 3D. Metode pengukuran yang digunakan meliputi:

2.9.1 Jumlah Node

Jumlah node merupakan jumlah titik atau elemen dalam suatu struktur data atau sistem, seperti graf, pohon, atau jaringan.

Fungsi evaluasi total dalam A* biasanya dinyatakan sebagai:

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

Keterangan:

1. $f(n)$ adalah total Cost yang diperkirakan untuk mencapai tujuan melalui simpul n.
2. $g(n)$ adalah Cost sebenarnya dari simpul awal ke simpul n.
3. $h(n)$ adalah nilai heuristik, atau estimasi Cost dari simpul n ke tujuan.

Pada penelitian ini, heuristik yang digunakan adalah Heuristik Manhattan. Heuristik Manhattan adalah metode untuk mengukur jarak antara dua titik dalam lingkungan grid berbentuk persegi panjang, di mana pergerakan hanya diperbolehkan dalam empat arah utama: atas, bawah, kiri, dan kanan [21].

Heuristik Manhattan, yang umumnya digunakan untuk grid 2D, dapat diperluas ke ruang 3D dengan mempertimbangkan dimensi ketiga, yaitu ketinggian atau sumbu Z. Dalam konteks 3D, heuristik Manhattan menghitung jarak antara dua titik dengan menjumlahkan perbedaan absolut di sepanjang sumbu X, Y, dan Z [9]. Berikut adalah formula untuk heuristik Manhattan dalam ruang 3D:

$$h(n) = |x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1|$$

Keterangan:

1. x_1, y_1, z_1 adalah koordinat dari simpul saat ini.
2. x_2, y_2, z_2 adalah koordinat dari simpul tujuan.
3. $|x_2 - x_1|, |y_2 - y_1|$ dan $|z_2 - z_1|$ masing-masing adalah perbedaan absolut dalam posisi sepanjang sumbu X, Y, dan Z.

2.9.2 Waktu Pencarian Jalur

Algoritma A* yang dioptimasi dengan heapsort diukur efisiensinya dalam menemukan jalur tercepat di peta game. Metrik ini menggunakan stopwatch atau timer untuk mencatat waktu yang dibutuhkan algoritma untuk menemukan jalur dari titik awal ke titik akhir. Perbandingan waktu pencarian jalur antara algoritma A* standar dan yang dioptimasi dengan heapsort dilakukan untuk menentukan efektivitas optimasi. Satuan yang digunakan untuk pengujian waktu pencarian jalur yaitu *millisecond* (ms).

2.9.3 Optimalisasi Jalur

Selain kecepatan, algoritma A* yang dioptimasi dengan heapsort juga diukur kemampuannya dalam menemukan jalur terpendek. Metrik ini menghitung jarak total antara setiap titik yang berdekatan pada jalur yang ditemukan. Perbandingan jarak jalur antara algoritma A* standar dan yang dioptimasi dengan heapsort dilakukan untuk menentukan efektivitas optimasi dalam hal menemukan jalur yang lebih pendek. Jarak jalur memiliki satuan unit jarak (unit), yang bisa diinterpretasikan sebagai panjang dari jalur yang ditemukan. Satuan ini bergantung pada skala dan pengukuran di dalam lingkungan game 3D.

2.9.4 Nilai Heuristik dan Jalur Robustness (Ketahanan)

Kelancaran jalur juga menjadi aspek penting dalam algoritma pencarian jalur. Metrik ini menghitung jumlah tikungan pada jalur yang ditemukan, mengindikasikan kelancaran pergerakan agen dalam game. Perbandingan kelancaran jalur antara algoritma A* standar dan yang dioptimasi dengan heapsort dilakukan untuk menentukan efektivitas optimasi dalam hal menemukan jalur yang lebih halus dan alami. Penilaian ini menggunakan skala 1-10 dengan rincian:

1. 1-3: Banyak perubahan arah tajam, jalur sangat berbelit-belit.
2. 4-6: Beberapa perubahan arah tajam, jalur cukup mulus tetapi masih ada bagian yang berbelit-belit.
3. 7-9: Sedikit perubahan arah tajam, jalur sangat mulus dan efisien.
4. 10: Tidak ada perubahan arah tajam, jalur sepenuhnya mulus dan optimal

Nilai heuristik dalam algoritma pencarian jalur, seperti A*, adalah estimasi atau pendekatan yang digunakan untuk memperkirakan Cost tersisa dari simpul saat ini ke tujuan akhir. Ini adalah bagian dari fungsi evaluasi total dalam Algoritma A* yang digunakan untuk menentukan urutan simpul mana yang akan dievaluasi [2]. Penilaian ini menggunakan skala 1-10, dengan rincian sebagai berikut:

1. 1-3: Jalur panjang dan kurang efisien, banyak zigzag, jauh dari rute optimal.
2. 4-6: Jalur relatif panjang dengan beberapa bagian zigzag, sebagian besar rute mengikuti jalur optimal.
3. 7-9: Jalur pendek dan efisien, sedikit zigzag, hampir mengikuti rute optimal.
4. 10: Jalur sangat pendek, sangat efisien, dan sepenuhnya mengikuti rute optimal.