GOAL-ORIENTED BEHAVIOR TREE UNTUK KOORDINASI *MULTI-AGENT SYSTEMS* DALAM GAME RPG

Laurentius Dika Andreano Vega, Latius Hermawan
Teknik Informatika, Universitas Katolik Musi Charitas
Jl. Bangau No.60, 9 Ilir, Kec. Ilir Timur III, Palembang, Sumatera Selatan, Indonesia
Laurentiusdika28@gmail.com

ABSTRAK

Penggunaan *Artificial Intelligence* pada game RPG membutuhkan *agent* yang otonom, namun kerangka kerja *Goal-Oriented Behavior Tree* (GOBT) saat ini masih memiliki kelemahan karena hanya berfokus pada *agent* individu dan mengabaikan koordinasi sumber daya. Penelitian ini bertujuan merancang dan mengevaluasi mekanisme koordinasi *Multi-Agent Systems* (MAS) berbasis GOBT dalam lingkungan Game RPG. Untuk mengatasi masalah persaingan tugas antar *agent*, dikembangkan mekanisme koordinasi menggunakan sistem lelang (*auction*) dan *reservation* yang dikelola oleh komponen pusat yaitu *CropManager*. Sistem ini diimplementasikan dalam simulasi pertanian berbasis Unity dan diuji secara kuantitatif dengan membandingkan skenario MAS dengan sistem terpusat melawan skenario Non-MAS. Hasil eksperimen menunjukkan bahwa mekanisme koordinasi yang diusulkan berhasil meningkatkan efisiensi distribusi tugas secara signifikan, yang dibuktikan dengan pencapaian nol kejadian konflik dan nol waktu eksekusi yang terbuang (wasted time). Integrasi GOBT dan *CropManager* terbukti efektif sebagai pendekatan AI untuk menciptakan agent yang kooperatif dan responsif dalam lingkungan permainan yang kompleks dan dinamis.

Kata kunci : *Goal-Oriented Behavior Tree, Multi-Agent Systems, Game AI, NPC,* Koordinasi *Agent*.

1. PENDAHULUAN

Industri game mengalami pertumbuhan yang signifikan setiap tahun, terutama dalam genre *Role-Playing Game* (RPG), di mana pemain berperan sebagai karakter fiksi dan menjalankan peran dalam dunia tersebut [1].

Salah satu elemen kunci dalam RPG yang sangat memengaruhi pengalaman pemain adalah kehadiran *Non-Playable Characters* (NPC) [2], [3], [4].

NPC yang mampu bereaksi terhadap pemain dan lingkungannya telah terbukti meningkatkan realisme dunia permainan dan memperkaya pengalaman bermain [5], [6], [7].

Untuk mengatasi kompleksitas lingkungan RPG yang dinamis, diperlukan *Artificial Intelligence* (AI) yang fleksibel, modular, dan efisien guna mengembangkan *intelligent agent* yang mampu bertindak secara otonom [8].

Salah satu kerangka kerja AI yang menjanjikan adalah *Goal-Oriented Behavior Tree* (GOBT), yang mengintegrasikan keunggulan *Goal-Oriented Action Planning* (GOAP), *Behavior Tree* (BT), dan pemilihan tindakan berdasarkan *Utility Systems* [9].

Pendekatan ini mampu mengurangi biaya komputasi GOAP sekaligus mengatasi keterbatasan BT yang cenderung kaku dalam merespons perubahan lingkungan [10], [11].

Meskipun memiliki keunggulan signifikan, implementasi GOBT hingga saat ini sebagian besar berfokus pada konteks *agent* individu yang bertindak tanpa mekanisme koordinasi dengan *agent* lain. Penerapan GOBT dalam konteks interaksi dan koordinasi di dalam *Multi-Agent Systems* (MAS) masih terbatas. Selain itu, sebagian besar penelitian terkini mengenai implementasi MAS masih sangat berfokus pada genre permainan strategi dan simulasi, dengan sedikit perhatian terhadap genre RPG. Ketiadaan mekanisme koordinasi antar *agent* ini akan memicu persaingan target dan pemborosan sumber daya komputasi ketika beberapa *agent* beroperasi di area yang sama [12].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan merancang dan mengimplementasikan mekanisme koordinasi MAS berbasis GOBT di dalam lingkungan RPG. Pendekatan yang diusulkan dilakukan melalui integrasi sistem pertukaran informasi tugas dan lelang (*auction*) status area target yang dikelola secara terpusat untuk mencegah konflik antar *agent* [13], [14].

Kinerja MAS dievaluasi melalui perbandingan langsung dengan skenario Non-MAS menggunakan pengujian kuantitatif. Metrik utama yang dinilai meliputi jumlah kejadian konflik target, total waktu eksekusi yang terbuang (*wasted time*), tingkat tumpang tindih *agent*, serta ketidakstabilan dalam pemilihan target [15].

Integrasi mekanisme komunikasi ini diharapkan dapat meningkatkan skalabilitas dan responsivitas *agent* dalam industri game.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Beberapa penelitian sebelumnya telah mengeksplorasi penggunaan GOAP dan BT dalam mengendalikan NPC. Penelitian oleh Stephane dan Eric [16] menunjukkan bahwa GOAP memerlukan biaya komputasi yang tinggi pada skala lingkungan yang besar, sementara pada penelitian Liu et all [17] BT memiliki batasan dalam adaptabilitas terhadap lingkungan yang terus berubah secara dinamis. Untuk

mengatasi kelemahan tersebut, kerangka kerja GOBT mulai dikembangkan, seperti yang ditunjukkan oleh penelitian Hong et all [9] yang membuktikan efisiensi GOBT dalam menciptakan *agent* yang responsif. Meskipun memiliki keunggulan signifikan, implementasi GOBT pada penelitian-penelitian tersebut sebagian besar masih berfokus pada konteks *agent* individu yang bertindak tanpa mekanisme koordinasi dengan *agent* lain.

2.2. *Artificial Intelligence*

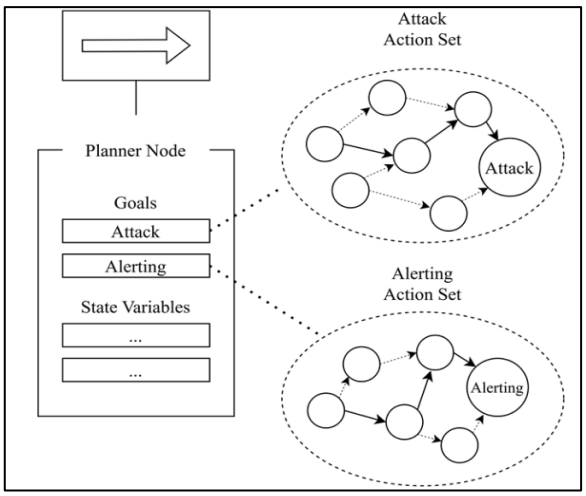
Artificial Intelligence (AI) didefinisikan sebagai bidang ilmu yang berfokus pada pengembangan sistem komputasi agar mampu melakukan tugas-tugas kognitif layaknya manusia. Secara teoretis, bidang ilmu ini mempelajari *intelligent agent* yang memproses persepsi dari lingkungannya dan meresponnya melalui serangkaian tindakan. Tujuan utama ilmu ini adalah membangun entitas yang mampu mengambil keputusan secara efektif, aman, dan rasional saat menghadapi berbagai situasi yang dinamis. Dalam konteks pengembangan *game*, AI diimplementasikan agar *virtual agent* bisa berperilaku cerdas, sehingga dunia permainan terasa lebih hidup dan responsif bagi pemain. Secara umum, pengembangan AI berfokus pada perancangan sistem yang mampu mencapai tujuan secara optimal dan memaksimalkan utilitas, baik melalui penalaran logis maupun pembelajaran data [18], [19].

2.3. Game

Game merupakan aktivitas yang melibatkan pemain untuk mencapai tujuan tertentu berdasarkan aturan yang berlaku. Aktivitas ini dibentuk oleh empat elemen utama, yaitu bermain (*play*), bersandiwara (*pretense*), tujuan (*goal*), dan peraturan (*rules*). Secara teknis, game dirancang sebagai ujian fisik atau mental yang bertujuan untuk menghibur atau memberikan imbalan kepada pesertanya. Penerapan aturan dan tujuan tersebut berfungsi mengubah proses pencapaian target menjadi sebuah tantangan, seperti mengalahkan lawan atau mengatasi rintangan. Pada akhirnya, game beroperasi sebagai sebuah sistem di mana pemain menghadapi konflik untuk menghasilkan pencapaian akhir yang dapat diukur [20].

2.4. *Goal-Oriented Behavior Tree*

GOBT adalah kerangka kerja kecerdasan buatan yang mengintegrasikan BT, teori utilitas, dan GOAP untuk mengatasi kelemahan BT konvensional yang statis dan rentan terhadap perluasan struktur tak terkendali. Seperti yang diilustrasikan pada Gambar 1, arsitektur hibrida ini menyematkan sebuah *Planner Node* ke dalam hierarki BT untuk mengoptimalkan efisiensi pengambilan keputusan, sehingga *agent* mampu beradaptasi secara dinamis terhadap berbagai situasi tak terduga di lingkungan permainan. Konsep dasar GOBT ialah memanfaatkan teori utilitas untuk menentukan tujuan terbaik secara *real-time* berdasarkan evaluasi terhadap status internal *agent* beserta kondisi lingkungannya. Setelah tujuan tersebut ditetapkan, mekanisme perencanaan GOAP akan merancang urutan tindakan operasional melalui tahap pemrosesan kondisi prasyarat (*preconditions*) dan efek yang dihasilkan (*effects*). Untuk memastikan *agent* tetap responsif, kerangka kerja ini dirancang agar terus melakukan perhitungan ulang terhadap nilai utilitas secara dinamis selama tindakan berlangsung [9].



Gambar 1. Goal action planning menggunakan planner node

2.5. *Multi-Agent Systems*

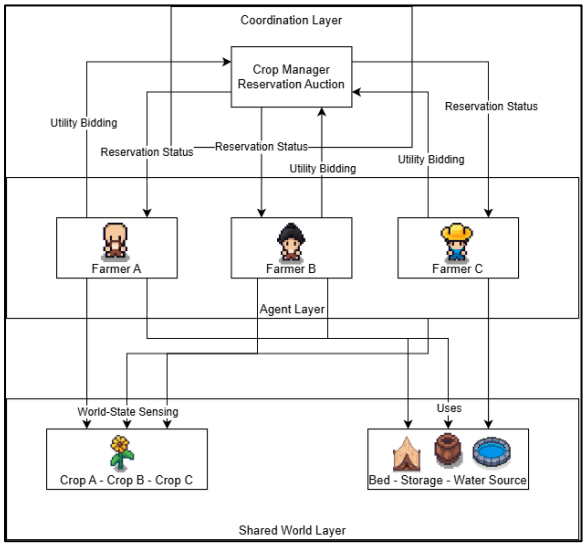
MAS merupakan sistem terdistribusi yang terdiri dari sekumpulan *agent* otonom yang saling berinteraksi melalui koordinasi dan negosiasi guna menyelesaikan tugas-tugas kompleks secara kolektif. Dalam bidang AI, MAS dipelajari sebagai kelompok *agent* yang saling memengaruhi kinerja satu sama lain melalui tindakan mereka di dalam lingkungan yang sama. Arsitektur ini menonjolkan prinsip modularitas, di mana setiap *agent* mengelola status internal secara mandiri dan berkomunikasi melalui perantara lingkungan untuk mencapai tujuan bersama. Kerangka ini telah berhasil diimplementasikan dalam berbagai domain, mulai dari industri dan logistik hingga pengembangan permainan, untuk menciptakan interaksi *agent* yang lebih realistis dan responsif [12], [19].

3. METODE PENELITIAN

Penelitian ini dilakukan pada simulasi pertanian dengan 3 *agent* dalam sebuah game RPG berbasis Unity dengan menggunakan konsep MAS. Setiap *agent* dimodelkan sebagai entitas yang memiliki kondisi internal seperti energi, tingkat kelaparan, inventaris, dan parameter kepribadian yang diatur oleh bobot utilitas. Proses pengambilan keputusan dibagi menjadi tiga tahap yang saling terkait. Pertama, pemilihan konteks dilakukan menggunakan Pohon Perilaku. Kedua, pemilihan tujuan didasarkan pada utilitas. Ketiga, perencanaan tindakan ditangani oleh GOAP.

3.1. Desain *Environment* MAS

Lingkungan simulasi ini mencakup *agent farmer*, tanaman dengan empat tahap pertumbuhan, serta objek pendukung seperti sumber air dan penyimpanan. Menggunakan prinsip MAS, setiap *agent* mengambil keputusan otonom berdasarkan status internal (energi dan kelaparan) serta informasi lingkungan, sementara konflik sumber daya diselesaikan secara terpusat oleh *CropManager* untuk mencegah persaingan antar *agent*.



Gambar 2. Arsitektur lingkungan dan alur koordinasi dalam MAS

Gambar 2 menunjukkan lingkungan eksperimental, yang terdiri dari tiga *agent* yang beroperasi di dalam satu area yang berisi tiga tanaman dan objek pendukung seperti tempat penyimpanan, tempat tidur, dan sumber air. Setiap *agent* beroperasi secara paralel dan berinteraksi dengan lingkungan dinamis. Proses ini dimulai dengan fase *world state sensoring* di mana setiap *agent* mengamati kondisi lingkungan, seperti tahap pertumbuhan tanaman, kebutuhan perawatan, dan ketersediaan sumber daya. Setelah *agent* memperoleh informasi lingkungan, mereka kemudian memasuki fase *auction*. Pada fase ini, setiap *agent* melakukan *utility bidding*, artinya setiap *agent* menghitung dan memilih *goal* terbaik untuk dieksekusi pada waktu tersebut. Penerapan mekanisme penawaran utilitas terbukti efektif dalam menyelaraskan target otonom masing-masing *agent* dengan efisiensi penyelesaian tugas secara keseluruhan [21].

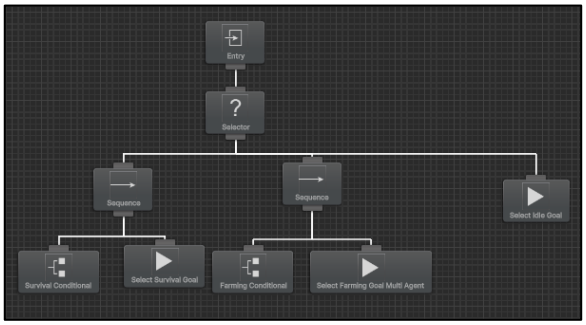
Setelah itu, tujuan terbaik beserta nilai utilitasnya dikirim ke komponen pusat, yaitu *CropManager*. *Goal* mewakili sasaran yang ingin dicapai oleh setiap *agent*, sedangkan *action* adalah urutan langkah optimal yang diambil untuk mencapai *goal*.

Tabel 1. Parameter awal agent dan bobot utilitas

Agent	**Status**	**Weight Energy (WE)**	**Weight Hungry (WH)**
Agent A	Energy=20 Hunger=80	0,2	0,1
Agent B	Energy=33 Hunger=12	0,1	0,6
Agent C	Energy=21 Hunger=70	0,3	0,2

Untuk menghasilkan perilaku yang beragam dan menghindari keputusan yang seragam, setiap *agent* diinisialisasi dengan parameter internal dan bobot utilitas yang berbeda-beda. Tabel 1 menunjukkan parameter awal yang digunakan dalam eksperimen ini, termasuk nilai bobot dan keadaan internal masing-masing *agent*. Bobot energi (WE) dan bobot rasa lapar (WH) menentukan sejauh mana biaya energi dan tingkat rasa lapar memengaruhi keputusan *agent*. Karena bobot-bobot yang berbeda ini, setiap *agent* dapat merespon situasi yang sama dengan cara yang berbeda.

3.2. Arsitektur *Goal-Oriented Behavior Tree*



Gambar 3. Arsitektur GOBT

Pada Gambar 3, arsitektur GOBT menggabungkan pemilihan prioritas umum (makro) dengan perencanaan tindakan terperinci (mikro). Pada tingkat makro, BT terlebih dahulu memeriksa kebutuhan dasar seperti rasa lapar dan energi. Jika kondisi ini mendesak, *agent* akan menjalankan *survival planner* dengan memilih tujuan makan atau tidur. Jika tidak, *agent* beralih ke *farming planner* dengan memilih tujuan berdasarkan perhitungan utilitas dari beberapa tanaman. Jika tidak ada tujuan yang sesuai, *agent* akan memasuki *idle*. Pada tingkat mikro, GOAP menerima tujuan dari *planner* dan mengatur urutan tindakan berdasarkan *precondition*, *effects*, dan *cost*. Misalnya, untuk *planting*, *agent* dapat memilih tindakan *PlantFast* jika memiliki *shovel*, atau tindakan *PlantSlow* jika tidak memiliki *shovel*. Untuk *watering*, tanaman harus berada dalam kondisi yang membutuhkan air. Untuk *harvesting*, tanaman harus sudah berada pada tahap panen. Dengan demikian, BT berfokus pada penentuan tujuan, Utilitas berada di dalam perencana sebagai pemilih tujuan, dan GOAP menangani langkah-langkah untuk mencapainya.

Tabel 2. *GOAP actions*

Action	cost	Preconditions	Effects
Get Seed	1	-	HasSeed ↑
Get Shovel	1	Shovel Available	Has Shovel ↑
Plant Fast	1	CropStage ≤ 0, HasSeed ≥ 1, HasShovel ≥ 1	CropStage ↑
Plant Slow	3	Crop Stage = 0, Has Seed	CropStage ↑
Get Watering Can	2	-	HasWateringCan ↑
Water Crop	2	Crop Needs Water, Has Watering Can	CropStage ↑, CropNeedsWater ↓, HasWateringCan ↓
Harvest Crop	3	Crop Stage ≥ 3	CropStage = 0, HasFood ↑
Eat	1	HasFood ≥ 1	Hunger ↓
Sleep	1	-	Energy ↑

Tabel 2 merangkum daftar tindakan yang tersedia dalam sistem, termasuk *precondition*, *effects*, dan *cost* masing-masing tindakan. Informasi ini berfungsi sebagai dasar bagi proses perencanaan pada tingkat mikro. Pada tingkat ini, GOAP akan mengevaluasi tindakan yang dapat dilakukan oleh *agent* dengan mempertimbangkan *precondition* yang harus dipenuhi dan *effects* yang dihasilkan. Proses perencanaan menggunakan algoritma rantai mundur, yang dimulai dari tujuan yang diinginkan dan bekerja mundur melalui urutan tindakan yang diperlukan hingga mencapai keadaan awal yang valid.

Setiap tindakan menimbulkan biaya yang secara dinamis memengaruhi keadaan internal *agent*, terutama tingkat energi dan rasa lapar. Perubahan ini memengaruhi keputusan *agent* pada langkah berikutnya, karena nilai utilitas menyesuaikan dengan kondisi terbaru. Dengan demikian, perencanaan tidak hanya menghasilkan urutan tindakan yang valid, tetapi juga mempertimbangkan efisiensi dan kondisi agent.

3.3. Koordinasi Utilitas dan Mekanisme *Auction*

Koordinasi antar *agent* dikelola oleh *CropManager* melalui dua tahap yaitu *reservation* dan *auction*. Alokasi tugas berbasis *auction* secara luas diakui sebagai metode yang sangat efisien dan dapat diskalakan untuk menyelesaikan penugasan dinamis dalam lingkungan *multi-agent*, karena memungkinkan *agent* untuk memaksimalkan utilitas sambil secara efektif meminimalkan konflik area target dan sumber daya [22].

Dalam proses *auction*, evaluasi nilai utilitas yang diajukan oleh *agent* dihitung menggunakan persamaan dasar berikut:

$$U(goal) = (W_{Goal} \times GB) - \left(W_E \frac{E_{Cost}}{E_{Max}}\right) - \left(W_H \frac{H_{Cost}}{H_{Max}}\right) \quad (1)$$

Berdasarkan Persamaan 1, nilai utilitas $U(goal)$ berbanding lurus dengan nilai manfaat dasar dari sebuah *goal* yang dikalikan dengan bobot prioritas tugas (W_{Goal}). Sebaliknya, skor utilitas ini akan direduksi oleh pengeluaran energi (E_{Cost}) dan peningkatan kelaparan (H_{Cost}) yang dinormalisasi terhadap batas maksimumnya (E_{Max} dan H_{Max}), kemudian dikalikan dengan bobot preferensi *agent* (W_E dan W_H). Dengan demikian, *agent* dapat menyesuaikan sensitivitasnya terhadap konsumsi energi dan kelaparan dalam proses pengambilan keputusan.

Alih-alih menetapkan W_{Goal} yang seragam untuk semua *agent*, penelitian ini mengkonfigurasi setiap *agent* dengan serangkaian bobot prioritas tujuan yang berbeda, sebagaimana disajikan pada Tabel 3.

Tabel *3. Goal priority weights setiap agent*

Goal	*Farmer A*	*Farmer B*	*Farmer C*
Planting	0,4	0,7	0,5
Watering	0,3	0,2	0,3
Harvesting	0,8	0,1	0,6

Konfigurasi ini melakukan diferensiasi untuk setiap *agent*. Misalnya, *Agent* B menunjukkan preferensi terkuat untuk *planting* (W_{Goal} =0,7), sementara *Agent* A dan *Agent* C lebih cenderung *harversting* (W_{Goal} =0,8 dan 0,6). Akibatnya, *agent* yang beroperasi dalam kondisi lingkungan yang identik akan tetap menghasilkan nilai utilitas yang berbeda dan mengurangi kemungkinan penawaran pada target yang sama.

Untuk mencegah tabrakan penawaran ketika beberapa *agent* memilih tanaman yang sama, sistem menyertakan *distance bonus* agar *agent* yang paling dekat secara lokasi mendapatkan prioritas lebih tinggi. Perhitungannya menggunakan jarak Euclidean (d) dengan formula sebagai berikut:

$$distanceBonus = 0.15 \times \left(1 - clamp_{[0,1]}\left(\frac{d}{50}\right)\right) \quad (2)$$

Variabel d merepresentasikan jarak dalam satuan koordinat dunia (*world units*), di mana satu unit setara dengan lebar satu petak lahan simulasi. Nilai ini kemudian dinormalisasi ke dalam interval [0,1] menggunakan fungsi clamp. Bonus mencapai nilai maksimal sebesar 0,15 saat *agent* berada tepat di lokasi target (d=0), dan berkurang secara linier hingga mencapai nilai 0 pada ambang batas jarak 50 unit. Oleh karena itu, nilai utilitas akhir yang diajukan ke lelang adalah:

$$U_{final} = U(goal) + distanceBonus \quad (3)$$

Mekanisme ini akan meminimalkan biaya perjalanan sekaligus mengurangi probabilitas tabrakan penawaran antar *agent* yang berdekatan. Selain itu, sistem menerapkan mekanisme penalti adaptif untuk memastikan kelangsungan hidup *agent*. Ketika tingkat energi mendekati ambang kritis, nilai utilitas semua *farming goals* dikurangi sebesar 50%, sehingga mendorong *agent* untuk memprioritaskan *survival goals* daripada melanjutkan pekerjaan yang mengonsumsi energi. Dalam hal koordinasi, jika *agent*

memenangkan proses lelang, target yang dipilih memasuki status *reserved* dan tidak dapat diakses oleh *agent* lain. Target ini akan dilepaskan kembali setelah *goal* selesai atau jika terjadi gangguan. Mekanisme ini dirancang untuk mencegah perebutan sumber daya sambil menghindari kondisi kelaparan, di mana *agent* lain tidak mendapatkan kesempatan untuk mengakses sumber daya yang sama.

4. HASIL DAN PEMBAHASAN

Bagian ini mendemonstrasikan implementasi arsitektur GOBT dalam skenario tiga *agent* dan tiga tanaman. Tanaman diset pada kondisi awal berbeda (tahap 0, 3, dan 2).

4.1. Hasil Eksekusi Skenario MAS

Tabel 4. Log 0-20 Detik

Time	*Agent*	*System Category*	*Decision and Execution Action*	*Status & Utility Parameters*
T=00:00	*Farmer A*	*GOAL & ACTION*	*Select SleepGoal -> Start the sleep process.*	*Energy = 20*
T=00:00	*Farmer C*	*GOAL & ACTION*	*Select SleepGoal -> Start the sleep process.*	*Energy = 21*
T=00:00	*Farmer B*	*AUCTION & GOAL*	*Winning Crop A auction -> Executing PlantingGoal.*	*U = 0.800, Stage = 0*
T=00:02	*Farmer A*	*AUCTION & GOAL*	*Winning Crop B auction (while sleeping) -> HarvestingGoal.*	*U = 0.903, Stage = 3*
T=00:04	*Farmer A*	*ACTION*	*Finished sleeping -> Start HarvestCrop action on target Crop B.*	*Energy = 80 (Recover)*
T=00:05	*Farmer B*	*RESOURCE & ACTION*	*Get seeds -> Start PlantSlow action on target Crop A.*	*Seed = 1*
T=00:06	*Farmer C*	*AUCTION & GOAL*	*Winning the Crop C auction (while sleeping) -> WateringGoal.*	*U = 0.395, Stage = 2*
T=00:08	*Farmer C*	*ACTION*	*Finished sleeping.*	*Energy = 80 (Recover)*
T=00:09	*Farmer B*	*GOAL*	*Active interrupt evaluation -> Select emergency SleepGoal.*	*Energy = 15 (Critical)*
T=00:10	*Farmer B*	*ACTION & AUCTION*	*PlantSlow canceled -> Target Crop A released -> Start sleeping.*	*Target Status: Free*
T=00:13	*Farmer C*	*RESOURCE & ACTION*	*Get watering tool -> Start WaterCrop action on target Crop C.*	*WateringCan = 1*
T=00:13	*Farmer A*	*ACTION & CROP*	*Crop B harvest is complete (plant resets to Stage 0) -> Target Crop B is released.*	*+1 Food (Shared = 1)*
T=00:13	*Farmer A*	*GOAL*	*Internal status change -> Select EatGoal.*	*Hunger = 90*
T=00:14	*Farmer B*	*AUCTION*	*Reschedule the Crop A auction (agent is still sleeping).*	*U = 0.803*
T=00:16	*Farmer B*	*ACTION*	*Finish sleeping -> Continue the PlantSlow action on target Crop A.*	*Energy = 80 (Recover)*
T=00:18	*Farmer A*	*ACTION & GOAL*	*Finish eating -> Win Crop B auction -> Execute PlantingGoal.*	*Hunger = 40, U = 0.518*
T=00:19	*Farmer A*	*RESOURCE & ACTION*	*Get seeds -> Start PlantSlow action on target Crop B.*	*Seed = 1*

Berdasarkan catatan log, pada T=00:00, *Farmer* A dan *Farmer* C memilih untuk tidur karena tingkat energi awal mereka rendah (20 dan 21), sementara *Farmer* B langsung memenangkan tugas penanaman Tanaman A dengan nilai utilitas 0,800. Pada T=00:02, *Farmer* A mengajukan permintaan untuk memanen Tanaman B dan berhasil dengan utilitas 0,903. Selanjutnya, pada T=00:06, *Farmer* C mengambil tugas menyiram Tanaman C dengan utilitas 0,395. Pola ini menunjukkan bahwa ketiga jenis tugas pertanian dapat dibagi dan dieksekusi secara paralel sejak awal tanpa konflik antar *agent*.

Selama fase eksekusi, diamati bahwa penyesuaian rencana berjalan lancar. *Farmer* B sempat mulai menanam Tanaman A secara perlahan, tetapi pada T=00:10 tindakan tersebut berhenti karena ia beralih ke tujuan tidur ketika energinya turun menjadi 15, dan reservasi pada Tanaman A dilepaskan. Pada T=00:13, *Farmer* A berhasil memanen Tanaman B, mengembalikannya ke tahap 0 dan meningkatkan makanan bersama sebesar +1, kemudian beralih ke tujuan makan. Setelah selesai makan pada T=00:18, *Farmer* A kembali ke aktivitas pertanian dan mengirimkan tugas penanaman untuk Tanaman B dengan utilitas 0,518. Rangkaian peristiwa ini menunjukkan bahwa sistem mampu beradaptasi antara kebutuhan dasar dan pekerjaan pertanian, sambil mempertahankan konsistensi dalam manajemen reservasi.

4.2. Hasil Eksekusi Skenario Non MAS

Evaluasi skenario Non-MAS dilakukan tanpa CropManager sebagai koordinator terpusat.

Tabel 5. Log 0-20 Detik

Time	*Agent*	*System Category*	*Decision and Execution Action*	*Status & Utility Parameters*
T=00:00	*Farmer A*	*GOAL & ACTION*	*Select SleepGoal → Start the sleep process.*	*Energy = 20*
T=00:00	*Farmer C*	*GOAL & ACTION*	*Select SleepGoal → Start the sleep process.*	*Energy = 21*
T=00:00	*Farmer B*	*GOAL*	*Select PlantingGoal → Crop A (utility intent).*	*U = 0.800, Stage = 0*
T=00:01	*Farmer B*	*GOAL*	*Target switch: Re-evaluate → Select PlantingGoal → Crop C.*	*U = 0.797, Stage = 0*
T=00:02	*Farmer A*	*GOAL*	*Select HarvestingGoal → Crop B (while sleeping).*	*U = 0.903, Stage = 3*
T=00:04	*Farmer A*	*ACTION*	*Finished sleeping → Start HarvestCrop action on target Crop B.*	*Energy = 80 (Recover)*
T=00:05	*Farmer B*	*RESOURCE*	*Get seeds.*	*Seed = 1*
T=00:06	*Farmer B*	*RESOURCE & ACTION*	*Get shovel → Start PlantFast action on target Crop C.*	*Shovel = 1*
T=00:06	*Farmer C*	*GOAL*	*CONFLICT INTENT: Select HarvestingGoal → Crop B (while sleeping) — same target as Farmer A.*	*U = 0.685, Stage = 3*
T=00:08	*Farmer C*	*ACTION*	*Finished sleeping → Start HarvestCrop on target Crop B (no reservation check).*	*Energy = 80 (Recover)*
T=00:09	*Farmer B*	*GOAL*	*Select SleepGoal (energy critical).*	*Energy = 15 (Critical)*
T=00:10	*Farmer B*	*ACTION*	*PlantFast COMPLETE → Crop C planted (2s) → Start sleeping.*	*Seed = 0, Shovel = 0*
T=00:13	*Farmer A*	*ACTION & CROP*	*HarvestCrop COMPLETE → Crop B harvested (reset to Stage 0) → +1 Food.*	*SharedTotal = 1*
T=00:13	*Farmer C*	*ACTION*	*HarvestCrop INTERRUPTED — Crop B already harvested by Farmer A. Wasted effort: 5 detik.*	—
T=00:13	*Farmer A*	*GOAL*	*Internal status change → Select EatGoal.*	*Hunger = 90, Food = 1*
T=00:13	*Farmer C*	*GOAL*	*Select PlantingGoal → Crop B (Stage 0 setelah reset).*	*U = 0.613, Stage = 0*
T=00:18	*Farmer A*	*ACTION & GOAL*	*Finish eating → Select PlantingGoal → Crop B.*	*Hunger = 40, U = 0.518*
T=00:18	*Farmer C*	*RESOURCE & ACTION*	*Get seeds → Start PlantSlow on target Crop B (utility intent).*	*Seed = 1*
T=00:19	*Farmer A*	*RESOURCE & ACTION*	*CONFLICT: Get seeds → Start PlantSlow on target Crop B — same target as Farmer C.*	*Seed = 1*

Berdasarkan log (Tabel 5), pada T=00:00 *Farmer* A dan C memilih tidur (energi rendah), sementara *Farmer* B memulai tugas penanaman (U=0,800). Perilaku pertama yang tidak stabil terjadi pada T=00:01, di mana *Farmer* B berganti target dari Tanaman A ke Tanaman C. Ketidakstabilan ini (perbedaan utilitas hanya 0,003) disebabkan oleh tidak adanya mekanisme reservasi; *agent* terus menghitung ulang utilitas secara lokal pada setiap siklus keputusan, menghasilkan seleksi target yang terus berubah-ubah.

Konflik pertama terjadi pada T=00:06 ketika *Farmer* C memilih *HarvestingGoal* pada Tanaman B, target yang sama yang sedang dikerjakan oleh *Farmer* A sejak T=00:04. Karena absennya deteksi reservasi, *Farmer* C memulai *HarvestCrop* pada T=00:08, menyebabkan dua *agent* mengeksekusi aksi panen secara bersamaan pada target tunggal dari T=00:08 hingga T=00:13. Ini merupakan pemborosan sumber daya komputasional. Pada T=00:13, setelah *Farmer* A menyelesaikan panen, eksekusi *Farmer* C otomatis terhenti (*INTERRUPTED*), menyebabkan pemborosan 5 detik waktu eksekusi. *Farmer* C kemudian segera memilih tujuan berikutnya: menanam Tanaman B yang telah direset ke tahap 0.

Konflik kedua terjadi secara paralel pada T=00:18 dan T=00:19 ketika *Farmer* C dan *Farmer* A secara bersamaan memilih *PlantingGoal* pada Tanaman B. Tanpa mekanisme reservasi dari *CropManager*, keduanya menginisiasi aksi *PlantSlow* secara tumpang tindih pada target fisik yang sama. Karena batas pengamatan berakhir di detik ke-20, *Farmer* A tercatat membuang waktu eksekusi selama 1 detik pada konflik susulan ini. Kondisi akhir pada batas waktu pengamatan 20 detik ini menunjukkan kegagalan sistem Non-MAS dalam mendistribusikan tugas secara berbasis lokasi, yang memicu pemborosan sumber daya komputasi.

4.3. Implementasi Interface

Pada Gambar 4, Antarmuka simulasi dalam Unity dilengkapi dengan panel inspeksi yang memungkinkan pemantauan *real-time* terhadap perubahan status *agent*, kondisi lingkungan, serta perhitungan utilitas di setiap keputusan. Fitur

perekaman alur proses ini berfungsi krusial untuk mendeteksi kesalahan teknis dan memastikan sistem AI tetap beroperasi secara responsif dan stabil selama interaksi kompleks berlangsung.

Gambar 4. Testing Scene

4.4. Analisis Perbandingan Kinerja

Untuk mengukur efektivitas mekanisme koordinasi berbasis *auction* dan *reservation* yang diimplementasikan dalam *CropManager*, perbandingan langsung antara skenario MAS dan Non-MAS dilakukan pada kondisi awal identik dalam rentang waktu 20 detik.

Tabel 6. Perbandingan Metrik MAS dan Non-MAS

Metrik	MAS	Non-MAS
Jumlah kejadian konflik target (*Conflict event*)	0 kejadian	2 kejadian (T=00:08, T=00:19)
Total waktu eksekusi yang terbuang (*Wasted time*)	0 detik	6 detik (5 detik konflik ke-1 + 1 detik konflik ke-2)
Jumlah maksimal *agent* yang tumpang tindih	1 *agent* per target	2 *agent* pada target yang sama
Ketidakstabilan pemilihan target (*Re-evaluate*)	0 kejadian	1 kejadian (Farmer B, T=00:01)

Berdasarkan data pada Tabel 6 dan catatan log sebelumnya, keunggulan arsitektur MAS terlihat jelas dari efisiensi distribusi tugasnya. Keberhasilan ini sangat dipengaruhi oleh mekanisme reservasi pada *CropManager* yang langsung mengunci target sesaat setelah pemenang lelang ditentukan. Hasilnya, seluruh tugas pertanian dapat dieksekusi secara paralel sejak awal tanpa adanya tumpang tindih. Berbeda halnya dengan skenario Non-MAS, ketiadaan sistem penguncian membiarkan dua *agent* mengeksekusi target fisik yang sama secara bersamaan. Kondisi ini mengakibatkan interupsi pada aksi *Farmer* C di T=00:13 yang membuang waktu 5 detik, serta munculnya konflik kedua di T=00:19. Hingga batas pengamatan 20 detik, insiden tersebut secara keseluruhan menyebabkan pemborosan waktu komputasi selama 6 detik tanpa menghasilkan *output* apa pun.

Pengujian ini membuktikan bahwa perbedaan preferensi pribadi *agent* belum cukup kuat untuk mencegah terjadinya bentrokan. Ketika jumlah target yang tersedia lebih sedikit daripada jumlah *agent* yang bekerja, para *agent* secara otomatis akan mengincar target terbaik yang sama. Kondisi ini bahkan memicu konflik beruntun yang memperburuk efisiensi sistem. Sebagai contoh, saat konflik pertama terjadi pada skenario Non-MAS (T=00:08), kondisi ini memaksa *Farmer* C untuk melepaskan targetnya dan mencari tujuan baru. Namun, karena situasi di sekitar belum banyak berubah, proses perhitungan ulang tersebut justru membuat *Farmer* C kembali memilih target yang sama, sehingga memicu bentrokan berulang.

Temuan ini menunjukkan bahwa diferensiasi berbasis bobot utilitas hanya efektif jika didukung mekanisme koordinasi seperti *auction* dan *reservation*. Integrasi antara GOBT dan *CropManager* terbukti mampu mengatasi kelemahan tersebut secara komprehensif. Arsitektur MAS ini tidak hanya berhasil mempertahankan otonomi masing-masing *agent*, tetapi juga mengeliminasi seluruh pemborosan waktu eksekusi, dengan mencapai nol kejadian konflik dan nol detik *wasted execution time*.

5. KESIMPULAN DAN SARAN

Implementasi GOBT dalam penelitian ini berhasil mengintegrasikan prioritas *agent* dan koordinasi sumber daya MAS ke dalam satu sistem di lingkungan RPG dinamis. Hasil pengujian menunjukkan bahwa pembagian peran antara BT (penentu konteks), modul Utilitas (pemilihan tujuan), dan GOAP (perencana aksi) mampu mempertahankan respon adaptif terhadap perubahan. Mekanisme reservasi dan *auction utility* yang diterapkan pada *CropManager* terbukti efektif mengurangi persaingan target, mempertahankan eksklusivitas pengolahan, dan mencegah konflik. Pendekatan ini menunjukkan potensi sebagai dasar AI koordinatif dalam permainan yang membutuhkan *agent* otonom, kooperatif, dan mudah dianalisis kuantitatif. Hasil perbandingan MAS dan Non-MAS menegaskan efektivitas mekanisme koordinasi yang diusulkan. Pengembangan selanjutnya dapat berfokus pada integrasi komunikasi yang lebih kompleks (misalnya, negosiasi *real-time*) atau penerapan pada genre RPG dengan skala lingkungan yang lebih besar dan dinamis.

DAFTAR PUSTAKA

[1] P. R. Setiawan *et al.*, “Systematic Literature Review of HCI Principles in Role – Playing Game Design : Towards a Comprehensive Framework for Enhancing Programming Skills,” vol. 15, pp. 33–48, 2026.

[2] Y. Karaca, D. Derias, and G. Sarsar, “AI-Powered Procedural Content Generation: Enhancing NPC Behaviour for an Immersive Gaming Experience,” *SSRN Electron. J.*, no. December, 2024, doi: 10.2139/ssrn.4663382.

[3] R. Ploug, E. Rimer, A. K. Skov Petersen, and M.

Scirea, "Open-Ended NPC Dialogue Favors Casual Players: A Pilot Comparison of Three LLM-Driven Dialogue Systems," *IEEE Conf. Comput. Intell. Games, CIG*, 2025, doi: 10.1109/CoG64752.2025.11114150.

[4] L. Hermawan, M. Malla, and M. B. Ismiati, "Penerapan Algoritma PRNG Dalam Permainan Snack and Ladders Berbasis Digital," *J. Inform. Upgris*, vol. 8, no. 2, pp. 174–177, 2023, doi: 10.26877/jiu.v8i2.13512.

[5] R. Danil Fajri, M. A. Syahputra, T. Raja, M. Zaki, A. Saifudin, and I. Kusyadi, "Perancangan Kecerdasan Buatan pada NPC Menggunakan UNITY 2D dan Perilakunya Terhadap Player," *J. Inform. ...*, vol. 6, no. 3, pp. 2622–4615, 2021, [Online]. Available: http://openjournal.unpam.ac.id/index.php/informatika575

[6] T. M. Siswoko, H. Haryanto, K. Hastuti, and R. Kadiasti, "Non-Playable Character (NPC) based on Behaviour Tree for Enhanced Immersive Experience in the Serious Game 'Warik's Adventure,'" *J. Appl. Informatics Comput.*, vol. 7, no. 2, pp. 284–290, 2023, doi: 10.30871/jaic.v7i2.6753.

[7] M. Ç. Uludağlı and K. Oğuz, *Non-player character decision-making in computer games*, vol. 56, no. 12. Springer Netherlands, 2023. doi: 10.1007/s10462-023-10491-7.

[8] A. Boldachev, "Executable Ontologies in Game Development : From Algorithmic Control to Semantic World Modeling," *arXiv*, pp. 1–25, 2026.

[9] Y. Hong, T. Yan, and J. Seo, "GOBT: A Synergistic Approach to Game AI Using Goal-Oriented and Utility-Based Planning in Behavior Trees," *J. Multimed. Inf. Syst.*, vol. 10, no. 4, pp. 321–332, 2023, doi: 10.33851/jmis.2023.10.4.321.

[10] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of Behavior Trees in robotics and AI," *Rob. Auton. Syst.*, vol. 154, p. 104096, 2022, doi: 10.1016/j.robot.2022.104096.

[11] J. S. Minji Kwon, "Game AI Agents Using Deliberative Behavior Tree Based on Utility Theory," 2022.

[12] J. Barambones, J. Cano-Benito, I. Sanchez-Rivero, R. Imbert, and F. Richoux, "Multiagent Systems on Virtual Games: A Systematic Mapping Study," *IEEE Trans. Games*, vol. 15, no. 2, pp. 134–147, 2022, doi: 10.1109/TG.2022.3214154.

[13] S. Wen *et al.*, "Auction-Based Behavior Tree Evolution for Heterogeneous Multi-Agent Systems," *Appl. Sci.*, vol. 14, no. 17, 2024, doi: 10.3390/app14177896.

[14] Y. Zhou *et al.*, "Multi-agent task allocation method based on the cost-effectiveness maximization multi-round auction algorithm," *Front. Phys.*, vol. 13, no. February, pp. 1–12, 2026, doi: 10.3389/fphy.2025.1617607.

[15] Z. Fang, T. Ma, J. Huang, Z. Niu, and F. Yang, "Efficient Task Allocation in Multi-Agent Systems Using Reinforcement Learning and Genetic Algorithm," *Appl. Sci.*, vol. 15, no. 4, 2025, doi: 10.3390/app15041905.

[16] C. Stephane and J. Eric, "Binary GPU-Planning for Thousands of NPCs," *IEEE Conf. Comput. Intell. Games, CIG*, vol. 2020-Augus, pp. 678–681, 2020, doi: 10.1109/CoG47356.2020.9231696.

[17] T. Liu, A. Cann, I. Colbert, and M. Saeedi, "Combining Reinforcement Learning and Behavior Trees for NPCs in Video Games with AMD Schola," 2025, [Online]. Available: http://arxiv.org/abs/2510.14154

[18] Ian Millington, *Artificial Intelligence for Games*. Morgan Kaufmann Publishers, 2006.

[19] S. J. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*. Pearson Series, 2021.

[20] E. Adams, *Fundamentals of game design*, vol. 47, no. 08. 2014. doi: 10.5860/choice.47-4462.

[21] J. Tan, R. Khalili, and H. Karl, "Learning to Bid Long-Term: Multi-Agent Reinforcement Learning with Long-Term and Sparse Reward in Repeated Auction Games," 2022, [Online]. Available: http://arxiv.org/abs/2204.02268

[22] M. Braquet and E. Bakolas, "Greedy Decentralized Auction-based Task Allocation for Multi-Agent Systems," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 675–680, 2021, doi: 10.1016/j.ifacol.2021.11.249.