INOTEK, Vol. 9
ISSN: 2580-3336 (Print) / 2549-7952 (Online)
Url: https://proceeding.unpkediri.ac.id/index.php/inotek/

Integrasi Deep Reinforcement Learning untuk Pengendalian Musuh yang Dinamis pada Game Action RPG

[1*]Muhammad Ainul Yaqin, [2]Danang Wahyu Widodo, [3]Ardi Sanjaya
[1] Teknik Informatika, Universitas Nusantara PGRI Kediri
E-mail: *[1]aanainul110303@gmail.com, [2]danayudo@yahoo.com, [3]dersky@gmail.com
Penulis Korespondens : Muhammad Ainul Yaqin

Abstrak—Perilaku musuh dalam game sering kali dibangun menggunakan pola statis yang sulit beradaptasi dengan pemain, sehingga menurunkan tantangan dan daya tarik permainan. Penelitian ini mengembangkan sistem kecerdasan buatan (AI) musuh pada game 3D menggunakan pendekatan Deep Reinforcement Learning (DRL) yang terintegrasi dengan Godot Engine. Sistem ini memungkinkan musuh mengambil keputusan secara adaptif berdasarkan situasi permainan. Jika sistem DRL tidak tersedia, musuh tetap dapat beraksi menggunakan logika konvensional yang telah diprogram. Evaluasi menunjukkan bahwa musuh berbasis DRL memperoleh cumulative reward lebih tinggi dan perilaku yang lebih variatif dibandingkan AI berbasis aturan tetap.
Kata Kunci— Deep Reinforcement Learning, Kecerdasan Buatan, Permainan Aksi, RPG

Abstract— *Enemy behavior in games is often built using static patterns that are difficult to adapt to the player, thus lowering the challenge and appeal of the game. This research develops an enemy artificial intelligence (AI) system in 3D games using the Deep Reinforcement Learning (DRL) approach integrated with the Godot Engine. This system allows the enemy to make adaptive decisions based on the game situation. If the DRL system is not available, the adversary can still act using conventional logic that has been programmed. The evaluation shows that the DRL-based adversary achieves higher cumulative rewards and more varied behaviors than the fixed rule-based AI.*
Keywords— *Artificial Intelligence, Deep Reinforcement Learning, Game Action, RPG*

I. PENDAHULUAN

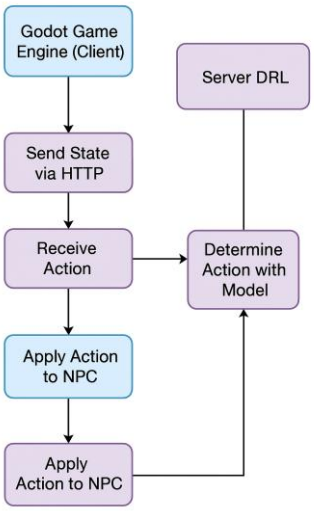
Dalam pengembangan *game* modern, keberadaan musuh yang cerdas dan adaptif memainkan peran penting dalam meningkatkan kualitas interaksi dan tantangan yang dirasakan pemain. Meskipun pendekatan berbasis aturan tetap (*rule-based*) telah lama digunakan, pendekatan ini sering kali menghasilkan perilaku musuh yang statis dan mudah ditebak. *Deep Reinforcement Learning* (DRL) muncul sebagai solusi yang menjanjikan karena kemampuannya untuk membentuk strategi berdasarkan pengalaman interaktif agen dalam lingkungan yang dinamis [1].

Salah satu tantangan utama adalah integrasi DRL langsung ke dalam *game engine* seperti Godot, yang meskipun bersifat *open-source* dan fleksibel, masih jarang digunakan secara luas untuk pelatihan dan *deployment* agen DRL. Upaya mengintegrasikan DRL dalam Godot telah dilakukan oleh beberapa peneliti, seperti Ranaweera dan Mahmoud yang mengembangkan pendekatan langsung dengan menggunakan *Python bindings* [2], serta Beeching dkk. yang memperkenalkan *framework Godot Reinforcement Learning Agents* untuk menjembatani algoritma DRL seperti PPO dan DQN dengan lingkungan *game* Godot [3].

Penelitian ini mengambil pendekatan berbeda, yaitu melalui arsitektur *client-server* menggunakan protokol HTTP. Dalam sistem ini, Godot berperan sebagai klien yang mengirimkan status lingkungan kepada *eksternal server* berbasis Python, yang menggunakan DRL sebagai pusat pengambilan keputusan. Pendekatan ini sejalan dengan arsitektur modular yang digunakan dalam penelitian sebelumnya untuk mendukung pemisahan antara komponen visual dan agen kecerdasan buatan melalui jaringan terdistribusi [4]. Musuh yang menjadi objek penelitian, bernama "Wraith", akan dilatih menggunakan algoritma *Proximal Policy Optimization* (PPO), yang dikenal karena stabilitas dan efisiensinya dalam lingkungan kompleks [5]. Tujuan dari penelitian ini adalah melihat efektivitas pendekatan DRL dengan integrasi DRL dengan Godot melalui *communication* HTTP.

II. METODE

A. Desain Sistem



Gambar 1. Alur Desain Sistem

Sistem terdiri dari dua komponen utama yang saling terintegrasi yaitu *Game Engine* (Godot) dan Server DRL (Python). Proses integrasi mengikuti pendekatan *Game Development Life Cycle* (GDLC), khususnya tahap produksi dan integrasi komponen sistem [6], yang berfokus pada penyatuan berbagai komponen sistem dan pengujian fungsional. Tahap produksi dalam pengembangan game merupakan proses di mana seluruh komponen individual

digabungkan menjadi satu kesatuan sistem permainan yang dapat diuji dan dijalankan secara lengkap.

Pada penelitian ini, proses integrasi mencakup:

1. Sinkronisasi antara Godot dan *Server* DRL

   Godot berperan sebagai klien yang secara berkala mengirimkan status lingkungan permainan ke server DRL melalui HTTP. Status ini meliputi informasi seperti posisi karakter, status HP/MP, dan jarak musuh. Server kemudian memproses data ini menggunakan model pembelajaran yang telah dilatih untuk menghasilkan aksi yang sesuai. Mekanisme ini menyerupai metode socket-based yang umum dalam integrasi lintas platform DRL [1].
2. Implementasi API FastAPI:

   Server DRL menggunakan FastAPI untuk menyediakan *endpoint*, yang memfasilitasi komunikasi secara real-time antara Godot dan model DRL. Setiap permintaan dari Godot menghasilkan keputusan aksi dari agen DRL, yang kemudian diterapkan ke dalam game secara langsung.
3. Pengujian Integrasi

   Setelah integrasi dilakukan, dilakukan pengujian menyeluruh untuk memastikan:
   1) Konsistensi data antara Godot dan server DRL
   2) Respons adaptif dari Enemy terhadap perubahan kondisi lingkungan
   3) Ketahanan sistem terhadap skenario permainan yang kompleks

B. Logika Konvensional Enemy

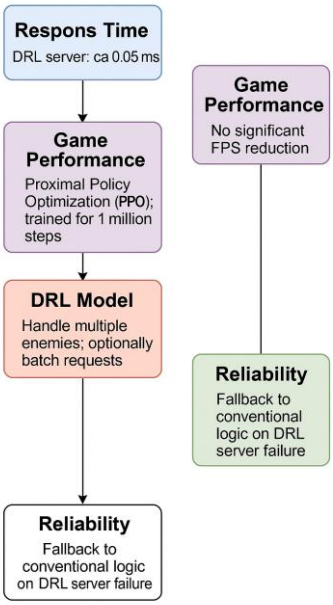
Dalam *Enemy script*, terdapat bagian kode yang mengatur musuh jika tidak menggunakan DRL atau *server* tidak nyala. Berikut penjelasan logika konvensional yang digunakan dalam Tabel 1 :

Tabel 1. *State* Logika Konvensional Enemy

Action	**Penjelasan Logika *state***
Summon	Jika energi musuh cukup dan jumlah *summon* belum maksimal, serta skill tidak sedang *cooldown*, musuh akan memanggil bantuan (*summon*).
Tornado	Jika *skill* "*full tornado*" tidak sedang *cooldown* dan jarak ke pemain sangat dekat, musuh akan menggunakan *skill* ini.
Full Tornado	Jika *skill* "tornado" tidak sedang *cooldown* dan jarak ke pemain cukup dekat, musuh akan menggunakan *skill* ini.
Basic Attack	Jika jarak ke pemain dalam rentang serang, musuh akan menyerang dengan sihir.
Mundur	Jika terlalu dekat dengan pemain, musuh akan mundur.
Mendekat	Jika terlalu jauh, musuh akan mendekat ke pemain.

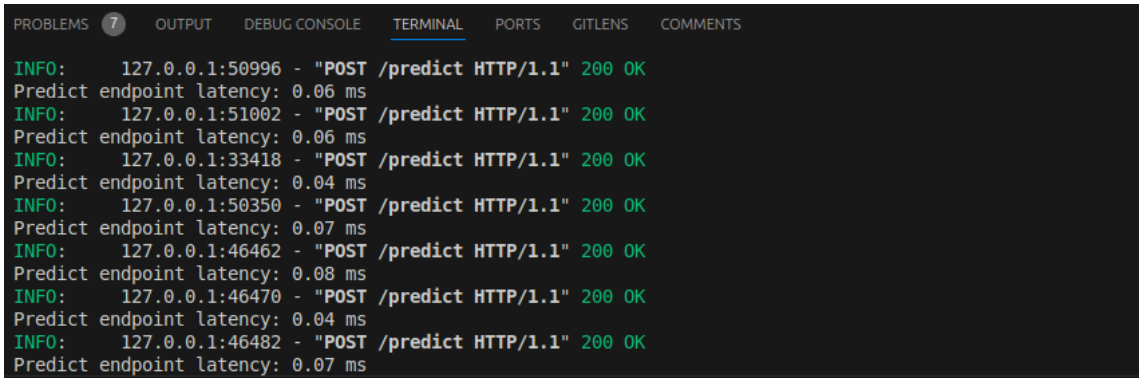
III. HASIL DAN PEMBAHASAN

A. Kinerja Sistem



Gambar 2. Kinerja Sistem

Sistem integrasi antara *Godot Engine* dan server *Deep Reinforcement Learning* (DRL) yang dibangun dengan FastAPI menunjukkan performa yang sangat baik dalam konteks permainan *real-time*. Dengan rata-rata waktu respons server DRL sekitar 0.05 milidetik, hal ini sesuai dengan kebutuhan sistem interaktif yang memerlukan latensi rendah untuk menjaga imersi permainan [7]. Sistem ini memastikan bahwa aksi musuh dalam permainan terasa responsif dan tidak mengganggu pengalaman bermain pengguna, seperti dalam *log screenshot* DRL Server di Gambar 3. Hal ini sejalan dengan kebutuhan aplikasi real-time yang menuntut latensi rendah untuk menjaga interaktivitas dan responsivitas sistem.



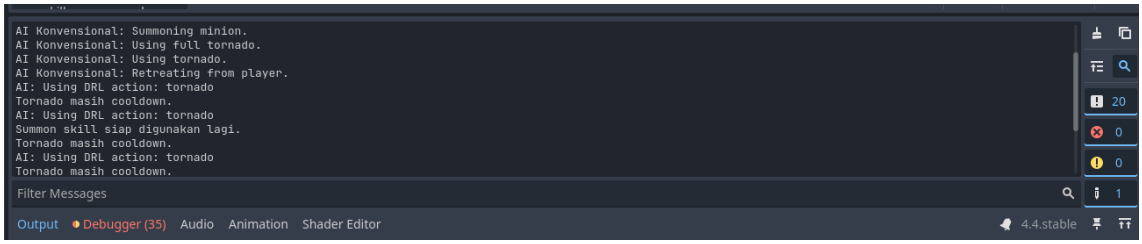
Gambar 3. Log Latensi Integrasi Godot dan DRL Server

Dari sisi performa game, pengujian dilakukan pada komputer dengan spesifikasi menengah (CPU 4-core, RAM 8GB). Hasilnya, integrasi DRL tidak menyebabkan penurunan *frame rate* (FPS) yang signifikan, baik saat model DRL aktif maupun ketika sistem beralih ke mode fallback berbasis aturan. Hal ini dimungkinkan karena script secara otomatis menjalankan logika *rule-based* jika server DRL lambat atau tidak responsif, sehingga *gameplay* tetap berjalan lancar.

Model DRL yang digunakan adalah *Proximal Policy Optimization* (PPO), yang dilatih selama satu juta langkah. Model yang telah dilatih ini dapat langsung dimuat dan digunakan untuk *inference* tanpa perlu pelatihan ulang, sehingga proses pengambilan keputusan oleh AI musuh berlangsung cepat dan efisien.

Terakhir, sistem ini dirancang dengan tingkat keandalan yang tinggi. Jika terjadi kegagalan pada server DRL, seperti *crash* atau *timeout*, *script* musuh secara otomatis beralih ke logika *rule-based* tanpa menyebabkan *error* pada *game*. Dengan demikian, integrasi DRL pada *Godot Engine* menghasilkan AI musuh yang responsif, *real-time*, dan andal tanpa mengorbankan performa *game* secara keseluruhan.

B. Perilaku Musuh



Gambar 4. Log Prediksi Aksi dari Model ke Agen Musuh

Musuh "Wraith" menunjukkan perilaku adaptif yang dihasilkan dari pelatihan menggunakan PPO. Keputusan seperti mundur saat terdesak, menyerang saat dalam jangkauan optimal, dan penggunaan *skill* pada timing tepat adalah hasil dari proses pelatihan interaktif selama satu juta langkah. PPO dipilih karena kemampuannya menjaga stabilitas pembelajaran pada lingkungan yang kompleks dan memiliki banyak aksi [8][9]. Selain itu, pendekatan ini sejalan dengan konsep *reinforcement learning* yang mempertimbangkan dinamika internal permainan secara lebih mendalam, sebagaimana ditunjukkan dalam penelitian SPRIG oleh Martinez-Lopez dkk yang mengintegrasikan persepsi dan mekanisme *internal game dynamics* untuk menghasilkan AI yang lebih adaptif dan strategis [10].

Jika sistem DRL tidak tersedia atau gagal merespons, musuh secara otomatis menggunakan logika konvensional yang telah diprogram pada *script*. Pada mode ini, perilaku musuh mengikuti aturan tetap, seperti menyerang jika jarak ke pemain sesuai, mundur jika terlalu dekat, atau memanggil bantuan jika energi mencukupi dan skill tidak sedang *cooldown*. Contoh logika ini dapat dilihat pada bagian fallback script, di mana musuh akan memprioritaskan aksi *summon*, *full tornado*, atau tornado sesuai kondisi, lalu menentukan apakah harus menyerang, mundur, atau mendekat ke pemain.

Dengan pendekatan DRL, perilaku musuh menjadi lebih bervariasi dan sulit diprediksi oleh pemain, sehingga tantangan dalam permainan meningkat. Sementara itu, *fallback rule-based* memastikan musuh tetap aktif dan responsif meskipun sistem DRL tidak berjalan, sehingga pengalaman bermain tetap terjaga.

C. Evaluasi *Cumulative Reward*

Gambar 5. Evaluasi *Training Model* Menggunakan *Cumulative Reward*

Cumulative *reward* adalah ukuran seberapa baik musuh menjalankan tugasnya selama satu episode permainan, yaitu dengan menjumlahkan seluruh poin *reward* yang diperoleh dari berbagai aksi, seperti menyerang pemain, bertahan hidup, atau menggunakan skill secara efektif. Pada penelitian ini, model *Deep Reinforcement Learning* (DRL) untuk musuh "Wraith" dilatih menggunakan algoritma *Proximal Policy Optimization* (PPO) selama satu juta langkah.

Setelah pelatihan selesai, model diuji dalam simulasi permainan sebanyak 100 episode untuk mengukur performanya. Hasil evaluasi menunjukkan bahwa Grafik hasil evaluasi memperlihatkan tren positif selama pelatihan dan evaluasi. Nilai eval/mean_reward yang mencapai lebih dari 380 menunjukkan bahwa musuh mampu mencapai performa optimal dalam memilih aksi [11], sementara eval/mean_ep_length yang stabil mengindikasikan bahwa musuh dapat bertahan cukup lama dalam medan permainan. Hasil ini mendukung penggunaan PPO sebagai algoritma DRL yang cocok untuk karakter dengan banyak aksi dan dinamika pertempuran yang kompleks seperti Wraith.

IV. KESIMPULAN

Penggunaan FastAPI sebagai *server* dan Godot sebagai *client* dapat diintegrasikan dalam game dengan algoritma DRL sebagai pengendali musuh yang dinamis, yaitu melalui integrasi model PPO berbasis FastAPI dengan latensi rendah serta mekanisme fallback yang menjaga kelancaran permainan. Namun, efektivitas pengambilan keputusan oleh agen DRL di dalam game masih belum optimal, diduga akibat perbedaan antara environment pelatihan dengan kondisi sebenarnya di Godot. Oleh karena itu, diperlukan pengembangan lebih lanjut melalui pelatihan langsung di lingkungan game agar perilaku musuh dapat lebih relevan dan adaptif terhadap situasi nyata permainan.

DAFTAR PUSTAKA

[1] P. Almeida, V. Carvalho, dan A. Simões, "Reinforcement Learning Applied to AI Bots in First-Person Shooters: A Systematic Review," *Algorithms*, vol. 16, no. 7, p. 323, 2023. doi: 10.3390/a16070323.

[2] M. Ranaweera and Q. H. Mahmoud, “Deep Reinforcement Learning with Godot Game Engine,” *Electronics*, vol. 13, no. 5, p. 985, 2024, doi: 10.3390/electronics13050985.

[3] E. Beeching, J. Dibangoye, O. Simonin, and C. Wolf, “Godot Reinforcement Learning Agents,” *arXiv preprint*, arXiv:2112.03636, 2021, doi: 10.48550/arXiv.2112.03636.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” *arXiv preprint*, arXiv:1707.06347, 2017, doi: 10.48550/arXiv.1707.06347.

[5] H. Kim, "An Efficient Load Balancing Scheme for Gaming Server Using Proximal Policy Optimization Algorithm," Journal of Information Processing Systems, vol. 17, no. 2, pp. 297-305, 2021. DOI: 10.3745/JIPS.03.0158.

[6] A. Juliani et al., “Unity: A General Platform for Intelligent Agents,” *arXiv preprint*, arXiv:1809.02627, 2018, doi: 10.48550/arXiv.1809.02627.

[7] L. Wang, B. Li, S. Wang, dan T. Wang, "Enhanced Proximal Policy Optimization for Complex Game AI: Applying Reinforcement Learning to Super Mario," *Academic Journal of Computing & Information Science*, vol. 7, no. 11, pp. 150–154, 2024. doi: 10.25236/AJCIS.2024.071120.

[8] Z. Yang, C. Li, X. Wang, dan Y. Tian, "PPO-ACT: Proximal Policy Optimization with Adversarial Curriculum Transfer for Spatial Public Goods Games," *arXiv preprint arXiv:2505.04302*, 2025. doi: 10.48550/arXiv.2505.04302.

[9] S. Corecco, G. Adorni, dan L. M. Gambardella, "Proximal Policy Optimization-Based Reinforcement Learning and Hybrid Approaches to Explore the Cross Array Task Optimal Solution," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1660–1679, 2023. doi: 10.3390/make5040082.

[10] F. Martinez-Lopez, J. Chen, dan Y. Lu, "SPRIG: Stackelberg Perception-Reinforcement Learning with Internal Game Dynamics," *arXiv preprint arXiv:2502.14264*, 2025. doi: 10.48550/arXiv.2502.14264.

[11] Y. K. Purwanto dan D.-K. Kang, "Multi-Agent Deep Reinforcement Learning for Fighting Game: A Comparative Study of PPO and A2C," *International Journal of Internet, Broadcasting and Communication*, vol. 16, no. 3, pp. 192–198, 2024. doi: 10.7236/IJIBC.2024.16.3.192