BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Saat ini teknologi telah mengalami perkembangan dengan pesat dan membawa banyak perubahan dalam pembuatan aplikasi, salah satu aplikasi yang berkembang saat ini adalah *video game*. *Video game* sendiri merupakan permainan elektronik yang melibatkan interaksi antar pemain / pengguna melalui perangkat masukan seperti *keyboard, joystick*. Terdapat beberapa genre yang saat ini sangat popular diantaranya adalah FPS *(First Person Shooter),* RPG *(Role Playing Game), Adventure, Platfomer, Action - Adventure, Strategy, Racing, Simulation*, dan banyak genre lainnya. Salah satu jenis game yang berkembang saat ini adalah game yang dimainkan dengan adanya lawan / pemain dalam video game tersebut. Permainan *tag* adalah salah satunya karena dalam permainan ini terdapat 2 karakter atau lebih yang bermain yaitu pengejar dan pemain yang dikejar. Dalam permainannya pemain akan akan menghindar dari pengejar agar lawan tidak ditandai dan pengejar akan mencoba mengejar pemain agar dapat ditangkap. Jika lawan sudah ditandai pengejar maka pemain akan kalah atau tidak bergerak.

Seringkali dalam video game kita dapat bertemu dengan lawan yang tidak dikendalikan oleh pemain di dunia nyata atau bisa disebut NPC *(Non-Player Character).* NPC */ Agent* sendiri biasanya dapat dilatih agar dapat melakukan tindakannya tanpa perlu diperintah. Begitu juga untuk membuat NPC yang kompetitif dibutuhkan desain *rule base* yang kompleks. Beberapa *video game* zaman dahulu masih menggunakan pendekatan *rule base* yang prosesnya dikontrol secara manual oleh skrip yang telah dibuat. Sehingga pemain dapat melawan musuh dengan sangat cepat dan membuat pemain kurang bersemangat dalam bermain. Dalam permainan *tag*

*agen*t NPC dan pemain dapat berinteraksi dalam permainan *tag*. Pemain dapat mencoba untuk berkomunikasi atau bekerja sama dengan *agent* NPC untuk mencapai tujuan bersama untuk memenangkan permainan.

Seiring meningkatnya kompleksitas permainan dan visualisasi *virtual* yang lebih dinamik, skrip ini tidak dapat lagi memenuhi semua situasi yang ada dan jauh kemiripannya dengan perilaku manusia. Video game saat sudah dianggap sebagai simulasi dari kehidupan nyata sehingga, pemodelan dari sikap sebuah *agent* dalam *video game* diambil dari sikap di dunia sebenarnya [1]. Peneliti mengusulkan menggunakan metode *machine learning* untuk mengatasi permasalahan tersebut. Metode *machine learning* memungkinkan *agent* NPC untuk meningkatkan performa dengan cara belajar dari kesalahan dan keberhasilan atau dengan meniru taktik dari pemain [2].

Unity Juga telah menyediakan *open source* API bernama *Unity ML-Agents* yang dapat digunakan untuk menyimulasikan *learning environment* yang dalam melatih *agent* NPC. Salah satu metode yang dapat digunakan untuk melatih *agent* NPC dengan *Unity ML-Agents* adalah *reinforcement learning* yang bertujuan melatih *agent* NPC dengan memberikan *reward* agar mencapai hasil yang optimal. Kelebihan dalam menggunakan *reinforcement learning* adalah *agent* dapat melakukan pembelajaran dengan sendirinya melalui proses *trial and error* [2]. *Reinforcement Learning* telah menunjukkan kinerja yang mengesankan di berbagai bidang termasuk *game*, penempatan *chip*, energi, rekomendasi, robotika dan banyak lainnya. Oleh karena itu, *Reinforcement Learning* telah aktif diteliti baik di bidang akademik maupun industri akhir-akhir ini. Mengikuti tren ini, beberapa kerangka kerja RL telah dirilis baru-baru ini. [3, p. 1]

Terdapat beberapa algoritma yang tersedia pada metode *Deep Reinforcement Learning*, salah satunya adalah algoritma *Proximal Policy Optimization* (PPO). Algoritma ini dapat mengoptimasi kebijakan pada setiap aksi yang dilakukan oleh *agent* agar menghasilkan agent

yang efisien dan stabil. Algoritma PPO bekerja dengan memperbaiki kebijakan secara iteratif melalui *trial and error*. Algoritma ini dimulai dari kebijakan awal, kemudian mengumpulkan data dengan berinteraksi dengan lingkungan serta mengambil tindakan berdasarkan kebijakan tersebut.

Salah satu penelitian terdahulu dengan topik serupa yang telah dilakukan oleh Olli-Pekka Juhola yang menghasilkan perbandingan pelatihan menggunakan *Unity ML-Agent* dengan *Unity Navigation* AI Tradisional. Dalam pengujianya *Unity Navigation* AI akan bersaing dalam mengumpulkan koin di samping *Unity ML-Agent* yang memiliki lima belas tempat latihan berbeda. Hasil dari pelatihan *agent* ini dapat disimpulkan bahwa *Unity ML-Agent* mampu mengumpulkan lebih dari dua kali lipat jumlah koin. *Unity NavMesh Agent* hanya dapat mengumpulkan 153 koin dibandingkan dengan 439 koin yang dikumpulkan oleh *Unity ML-Agent* yang melakukan pembelajaran sendiri tanpa tahu medan yang dilalui.

Maka pada penelitian ini, peneliti ingin melakukan implementasi *Artificial Intelligence* kedalam *agent* NPC pada permainan *tag* dengan tema *action*. Teknik ini akan menciptakan 2 buah *agent* yang pertama adalah *agent* yang mampu melakukan tracking pada arena untuk menemukan musuhnya lalu membuatnya tertangkap dan *agent* yang kedua adalah *agent* yang mampu menghindari pengejar dan membuatnya lolos dari kejaran serta mencari koin untuk mendapatkan *reward* dan menyelesaikan permainan. *Reinforcement learning* dapat digunakan sebagai metode untuk dapat melatih *agent* dan menggunakan tools *Unity ML-Agents* serta menggunakan algoritma *Proximal Policy Optimization* (PPO) untuk melakukan simulasi agent.

1.2 Batasan Masalah Penelitian

Berikut batasan-batasan masalah pada penelitian ini agar menjadi tolak ukur dan dapat terarah:

1. Penelitian ini membahas cara meningkatkan perilaku *agent* NPC agar dapat mengejar dan menghindar target dengan menerapkan *reinforcement learning*.
2. Simulasi ini dilakukan menggunakan *asset* yang sudah tersedia dari *Unity Engine* dan dirancang menggunakan *Unity Machine Learning Agents Toolkit (ML-Agents)* dalam melatih perilaku *agent*.
3. Algoritma yang digunakan pada penelitian ini adalah algoritma *Proximal Policy Optimization* (PPO).
4. Dalam penerapan *agent* disini terdapat 2 *agent* yang bertugas yaitu sebagai *agent* pengejar dan *agent* pemain yang dikejar
5. Game ini memiliki 2 level permainan dan 2 tingkat kesulitan serta menggunakan jenis permainan *Third Person* dengan *platform* PC

1.3 Rumusan Masalah

Berdasarkan latar belakang diatas, maka rumusan masalah pada penelitian ini adalah bagaimana cara menerapkan metode *reinforcement learning* pada *agent NPC* dengan menggunakan algoritma *Proximal Policy Optimization* (PPO) pada *agent* tersebut agar menjadi *agent* yang pintar.

1.4 Tujuan Penelitian

Berdasarkan latar belakang yang telah dijelaskan diatas, tujuan dari penelitian ini adalah menciptakan *agent* NPC yang telah dilatih menggunakan algoritma *Proximal Policy Optimization* (PPO), sehingga menghasilkan *agent* NPC yang memiliki tingkat kecerdasan yang lebih tinggi dan dapat mengambil keputusan yang lebih baik dalam berbagai situasi permainan dan menyematkan *agent* tersebut kedalam permainan.

1.5 Manfaat Penelitian

Berikut beberapa manfaat yang didapatkan dari penelitian ini:

1. Bagi Pemain
    a. Dapat bermain *game* dengan suasana yang berbeda karena lebih seru dan menantang
    b. Pemain dapat mengambil keputusan lebih strategis lagi untuk mengatasi tantangan yang telah diberikan oleh *agent*
2. Bagi Pengembang (Developer)
    a. Dapat meningkatkan efisiensi waktu pembuatan kode progam dalam proses pengembangan *game*
    b. Dapat mengetahui hasil perilaku *agent* pintar yang akan dibuat dalam pengembangan *game* dengan menerapkan kecerdasan buatan