JURNAL INFOTEL
Informatics - Telecommunication - Electronics

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

PENERAPAN *REINFORCEMENT LEARNING* PADA PERILAKU *AGENT* DALAM PERMAINAN *TAG*

Brilly Jalu Kumara Biseka [1], Adhi Prahara [2]
[1,2] Universitas Ahmad Dahlan
1,2 Jl. Ringroad Selatan, Daerah Istimewa Yogyakarta 55191, Indonesia
[1] brilly1900018294@webmail.uad.ac.id , [2] adhi.prahara@tif.uad.ac.id

Received, Revised, Accepted

Saat ini tidak banyak dari game yang memiliki NPC dengan perilaku yang sulit ditebak sehingga ketika pemain bermain akan merasa cepat bosan dengan game yang dimainkan dikarenakan NPC mudah dikalahkan dan game dapat mudah diselesaikan. Kebanyakan dari game masih menggunakan metode rule-based dalam pengembangan NPC-nya. Maka dari itu penelitian ini bertujuan untuk membuat agent NPC yang perilakunya sulit ditebak dengan menggunakan metode Reinforcement Learning dan menggunakan algoritma Proximal Policy Optimization (PPO). Dalam pelatihanya terdapat 2 agent yang akan dilatih yaitu agent pengejar dan agent pemain dan 1 objek target koin. Proses pelatihan ini ditentukan oleh total percobaan percobaan agent yang disebut MaxSteps dan memiliki waktu 60 detik setiap sesi pelatihan. Hasil dari pelatihan ini akan ditunjukan dalam bentuk grafik Cumulative Reward, grafik Episode Length, grafik Policy Loss, dan grafik Value Lost. Pada hasil penelitian ini menunjukan agent-agent permainan tag ini mampu bersaing untuk mengejar target dan menghindari rintangan dengan baik. Hasil pada pengujian agent pemain menunjukan rata-rata koin yang terambil meningkat dari 61% menjadi 79% sedangkan pada agent pengejar menunjukan rata-rata menangkap agent pemain sedikit menurun dari 66% menjadi 63%. Dapat disimpulkan bahwa kemampuan agent pemain mengalami peningkatan sedangkan kemampuan agent pengejar mengalami sedikit penurunan

Keywords – *Unity*, *Ml-Agents*, *Proximal Policy Optimization*, *Agent*, *Tag*

I. INTRODUCTION

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. Salah satu jenis game yang berkembang saat ini adalah game yang dimainkan dengan adanya lawan / pemain dalam video game tersebut [1]. Permainan tag adalah salah satunya karena dalam permainan ini terdapat 2 karakter atau lebih yaitu pengejar dan pemain yang dikejar [2]. 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) [3]. NPC sendiri memiliki tindakan yang memungkinkan mereka untuk bernalar tentang apa yang dapat mereka lakukan dalam permainan dan bagaimana mereka dapat melakukannya [4]. Begitu juga untuk membuat NPC yang menggunakan desain rule base static pergerakanya dapat diprediksi dapat mempengaruhi nilai hiburan game [5]. Beberapa video game zaman dahulu masih menggunakan pendekatan rule base yang prosesnya dikontrol secara manual oleh skrip yang telah dibuat. Sehingga NPC tidak memiliki kemampuan untuk beradaptasi dan berkembang seiring waktu dengan lingkungan mereka [6].

Seiring meningkatnya kompleksitas permainan dan visualisasi virtual yang lebih dinamik, skrip ini tidak dapat lagi memenuhi semua situasi yang ada dan menyebabkan perilaku yang kurang dinamis dan kurang adaptif [7]. Video game saat sudah dianggap sebagai simulasi dari kehidupan nyata sehingga, pemodelan dari sikap sebuah agent dalam video game

dapat meningkatkan pengalaman pemain dalam berbagai cara, misalnya, sebagai musuh ataupun sekutu [8]. Peneliti mengusulkan menggunakan metode machine learning untuk mengatasi permasalahan tersebut.

Unity Juga telah menyediakan open source API bernama Unity ML-Agents yang dapat digunakan untuk menyimulasikan learning environment yang dalam melatih agent NPC [9]. Salah satu metode yang dapat digunakan untuk melatih agent NPC dengan Unity ML-Agents adalah reinforcement learning yang bertujuan melatih agent NPC agar memutuskan cara terbaik untuk melakukan tugas yang diberikan dan membuat perilaku sebaik mungkin [10]. Reinforcement Learning telah menunjukkan kinerja yang mengesankan di berbagai bidang termasuk game, penempatan chip, energi, rekomendasi, robotika dan banyak lainnya [11] serta menjadi langkah untuk menuju pembangunan sistem otonom dengan pemahaman tingkat yang lebih tinggi tentang dunia visual [12].

ML-Agents SDK terdiri dari tiga entitas inti: Sensor, Agen, dan Akademi [13]. Terdapat beberapa algoritma yang tersedia pada metode Deep Reinforcement Learning, salah satunya adalah algoritma Proximal Policy Optimization (PPO) [14]. Algoritma ini dapat mengoptimasi kebijakan pada setiap aksi yang dilakukan oleh agent agar menghasilkan agent yang efisien dan stabil. Algoritma PPO bekerja mengatur pembaruan kebijakan dengan rasio probabilitas yang terpotong, dan membuat parameter kebijakan[15].

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. Fungsi reward merupakan umpan balik objektif dari lingkungan [16]. Reinforcement learning dapat digunakan sebagai metode untuk dapat melatih agent dengan menggunakan tools Unity ML-Agents dan menggunakan algoritma Proximal Policy Optimization (PPO) untuk melakukan simulasi agent [17].

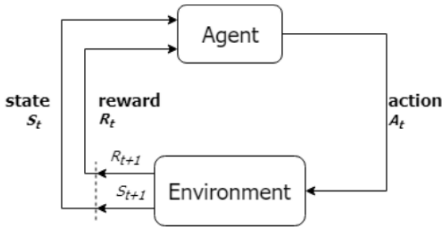
II. RESEARCH METHOD

A. Reinforcement Learning

Reinforcement learning merupakan adalah paradigma belajar yang berkaitan dengan pembelajaran untuk mengendalikan suatu sistem sehingga dapat memaksimalkan ukuran kinerja numerik yang mengekspresikan tujuan jangka panjang [18]. Metode Reinforcement learning menggunakan metode yang berbasis model. Model ini dapat diberikan atau dipelajari dari pengalaman [19]. Dalam proses pembelajaran, data yang digunakan adalah agent yang melakukan pembelajaran berdasarkan trial and error [20]. Reinforcement learning bertujuan untuk menghasilkan kebijakan yang optimal di lingkungan tertentu [21]. Reinforcement learning berkembang pesat disemua bidang dan digunakan dalam banyak contoh dunia nyata [22].

Agent yang melakukan pembelajaran melalui trial and error akan diberi imbalan atas tindakan yang benar dan mendapat hukuman atas tindakan negatif yang seharusnya tidak dilakukannya [23] di mana satu-satunya input dari lingkungan adalah reward skalar yang tertunda dan tugas masing-masing agen adalah memaksimalkan hadiah jangka panjang yang dihitung setiap tindakan [24].

Secara khusus, agent dan lingkungan berinteraksi di masing-masing waktu t = 0,1,2,3,..., dan setiap kali t, agent mendapatkan keadaan (State) St, di mana S adalah himpunan keadaaan yang mungkin terjadi. Selanjutnya, agent akan memilih tindakan At, di mana A adalah serangkaian tindakan pada keadaan St. Selanjutnya, agent mendapat hadiah Rt kemudian mendapat keadaan baru (St + 1)[25]. Berikut lampiran gambar 2.1 mengenai proses interaksi antara agent dan lingkungan.



Gambar 2. 1 Interaksi Agent Dengan Environment[23]

B. Proximal Policy Optimization

Proximal Policy Optimization merupakan algoritma yang mengajarkan agen untuk memperbarui kebijakan yang sama yang saat ini digunakan agen untuk memilih tindakan [26]. Algoritma PPO ini beberapa manfaat dari dari trust region policy optimiza-tion (TRPO) yang relatif rumit, dan tidak kompatibel dengan arsitektur yang mencakup berbagi parameter antara kebijakan dan fungsi nilai, atau dengan tugas tambahan, sedangkan algoritma PPO ini jauh lebih sederhana untuk diterapkan, lebih umum, dan memiliki kompleksitas sampel yang lebih baik (secara empiris) [14].

$r_t(\theta)$ menunjukkan rasio probabilitas $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta\ old}(a_t \mid s_t)}$, jadi $r_t(\theta_{old}) = 1$. TRPO memaksimalkan tujuan "surrogate" yang ditunjukan pada rumus 2.1

$$L^{CPI}(\theta) = \hat{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta\ old}(a_t \mid s_t)}\ \hat{A}t\right] = \hat{E}_t\left[r_t(\theta)\ \hat{A}t\right] \quad (2.1)$$

CPI mengacu pada iterasi kebijakan konservatif, di mana tujuan ini diusulkan tanpa kendala, maksimalisasi LCPI akan mengarah pada pembaruan kebijakan yang terlalu besar [14]. Oleh karena itu,

perlu mempertimbangkan bagaimana memodifikasi tujuan, untuk menghukum perubahan pada kebijakan yang memindahkan $r_t(\theta)$ dari 1 [14]. Tujuan utama yang diusulkan ditunjukkan dalam rumus 2.2

$$L^{CPI}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\,\hat{A}t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}t\right)\right] \quad (2.2)$$

Agen dilatih menggunakan Proximal Policy Optimization (PPO) menggunakan clipped surrogate objective. Karena penggunaan lapisan berulang (seperti LSTM), tindakan yang akan dipilih a_t pada waktu t dan s_t merupakan keadaan pada waktu t [14]. $\pi_\theta(a_t \mid s_t)$ menunjukan probabilitas kebijakan untuk mengambil tindakan a_t dalam keadaan s_t dengan parameter θ. $\hat{A}t$ menunjukkan perkiraan keuntungan berdasarkan Generalized Advantage Estimation (GAE), θ merupakan parameter jaring saraf dan rentang klip [14].

C. Identifikasi Pelatihan Agent

Pada tahapan pelatihan agent ini adalah tahapan agent yang melakukan pelatihan menggunakan Toolkit Unity ML-Agents dengan metode Reinforcement Learning. Agen memutuskan tindakan dari serangkaian opsi yang tersedia untuk mengoptimalkan hadiah yang dikumpulkan [27]. Beberapa hal yang dipersiapkan untuk melakukan pelatihan agent.

Untuk lingkungan yang digunakan akan berbentuk seperti labirin tertutup dimana akan diletakan 2 agent pelatihan dan satu target. Agent ini diantaranya agent pemain dan agent pengejar dimana agent pemain memiliki aturan pelatihan untuk berusaha mendapatkan target dan menghindari agent pengejar serta menghindari dinding pembatas dan gangguan. Sedangkan untuk agent pengejar memiliki aturan pelatihan untuk berusaha mengejar agent pemain dan menghindari dinding pembatas dan gangguan. Berikut lampiran tabel 3.1 yang menampilkan daftar identifikasi action yang dimiliki oleh agent pengejar dan agent pemain pada pelatihan ini.

Tabel 2. 1 Identifikasi Action Agent

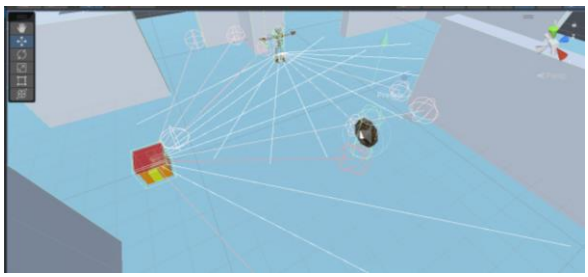
No	Agent	Action
1	Agent Pemain	a) Berjalan b) Menabrak Pemain c) Rotasi d) Menghindari dinding pembatas dan gangguan
2	Agent Pengejar	a) Berjalan b) Menghindari Pengejar c) Mencari Koin d) Rotasi e) Menghindari dinding pembatas dan gangguan

III. RESULT

A. Pelatihan Agent

Dalam pelatihan agent ini akan mengimplementasikan 2 agent yaitu agent pemain dan agent pengejar untuk dilatih menjadi agent yang cerdas. Pada tahapan ini dilakukan satu kali pelatihan dimana kedua agent akan diletakan pada satu lingkungan yang sama. Dalam pelatihan agent memiliki aturan tersendiri pada setiap agent, dimana agent akan berinteraksi dengan lingkungan permainan dan mendapatkan reward sesuai agent masing-masing.

Proses pelatihan ini ditentukan oleh total jumlah percobaan agent yang disebut MaxSteps. MaxSetps digunakan untuk memotivasi agen untuk mempelajari strategi yang efisien dan menghindari penundaan yang tidak perlu [28]. Dalam pelatihanya terdapat 2 objek karakter sebagai agent pemain dan agent pengejar serta satu objek target koin. Area pelatihan pada penelitian ini menggunakan 3D Object yang ada pada Unity. Alas yang digunakan pada area ini menggunakan kubik dan untuk membuat pembatas dinding tiap sisi juga menggunakan Object kubik serta gangguan pada area berupa dinding juga menggunakan object kubik. Berikut lampiran gambar 3.1 menampilkan karakter dan object serta lingkungan yang digunakan dalam pelatihan.



Gambar 3. 1 Area Pelatihan Agent

B. Implementasi Pelatihan Agent

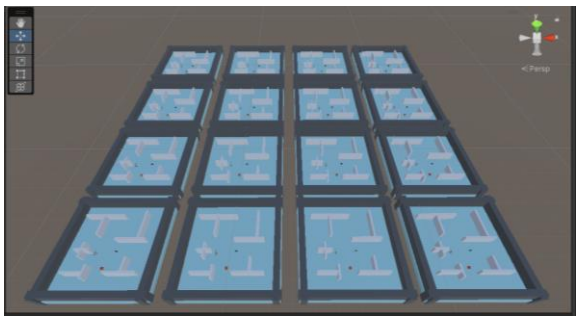
Untuk memulai pelatihan ini agent akan diberikan beberapa konfigurasi action agar dapat berinteraksi dengan lingkungan pelatihan. Agent pemain, agent pengejar, dan target koin akan diletakkan secara acak diarea lingkungan akan mengejar target koin yang diletakkan secara acak diarea lingkungan permainan. Dalam pelatihan setiap episode yang dijalankan diberikan waktu 60 detik untuk melakukan pelatihan. Setiap agent diberi konfigurasi reward yang berbeda-beda.

a) Agent pemain berfokus pada pencarian koin pada lingkungan karena memiliki reward yang besar yaitu 10 poin Ketika agent pemain mendapatkan reward dari koin sebesar 10 point maka agent pengejar akan mendapatkan reward negative sebesar -2

b) Agent pengejar akan berfokus mencari agent pemain karena memiliki reward yang besar yaitu 10 point. Ketika agent pengejar mendapatkan reward dengan menangkap agent pemain sebesar 10 point maka agent pemain akan mendapatkan reward negative sebesar -2.

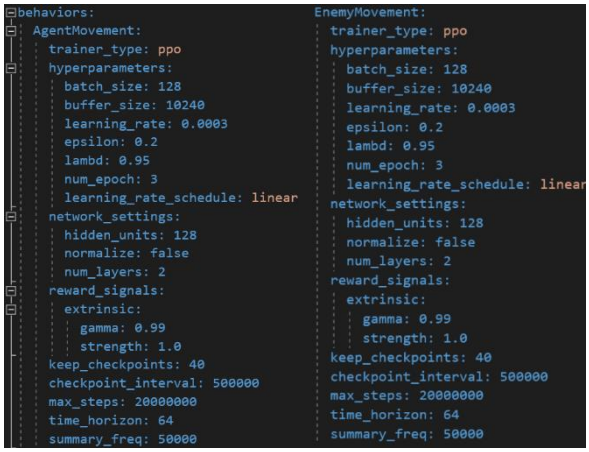
Ketika agent-agent pelatihan memulai pelatihanya agent-agent akan mulai bergerak secara tidak beraturan dan menabrakan diri ataupun menghindari pada setiap

object yang ada pada area pelatihan dengan melihat menggunakan sensor penglihatan ray perception. Ketika sekiranya agent sudah mengetahui seluruh jenis object yang dilihat menggunakan sensor penglihatan ray perception maka agent mulai melakukan tindakan yang pasti seperti menghindari kontak dengan dinding pembatas, dan harus dapat menyelesaikan pelatihan kurang dari waktu yang ditentukan karena akan mendapatkan reward negative sebesar (-1/MaxStep). Dalam pelatihan ini saya menggunakan 16 area pelatihan agar mempercepat proses pelatihan agent yang ditunjukan pada gambar 3.2.



Gambar 3. 2 Area Pelatihan yang Digunakan Pada Pelatihan

Pada penelitian ini menggunakan algoritma PPO dalam pelatihanya dan jumlah total percobaan yang dilakukan oleh agent adalah 20000000 MaxSteps. Dikarenakan terdapat 2 agent maka konfigurasi ini dibuat untuk agent pemain dan agent pengejar. Berikut lampiran gambar 3.3 menampilkan konfigurasi pelatihan agent.



```
behaviors:
  AgentMovement:
    trainer_type: ppo
    hyperparameters:
      batch_size: 128
      buffer_size: 10240
      learning_rate: 0.0003
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      hidden_units: 128
      normalize: false
      num_layers: 2
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 40
    checkpoint_interval: 500000
    max_steps: 20000000
    time_horizon: 64
    summary_freq: 50000
  EnemyMovement:
    trainer_type: ppo
    hyperparameters:
      batch_size: 128
      buffer_size: 10240
      learning_rate: 0.0003
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      hidden_units: 128
      normalize: false
      num_layers: 2
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 40
    checkpoint_interval: 500000
    max_steps: 20000000
    time_horizon: 64
    summary_freq: 50000
```

Gambar 3. 3 Komponen Konfigurasi Pelatihan Agent

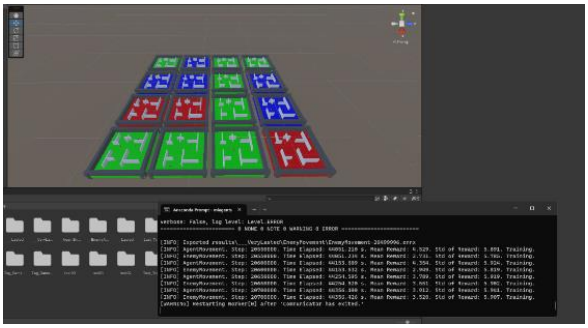
Ketika melakukan pelatihan terdapat penanda jika episode tersebut telah selesai atau belum dan kondisi mana yang menjadi selesainya pelatihan tersebut apakah agent pemain yang menang atau agent pengejar yang menang atau bahkan tidak ada yang menang atau waktu habis. Sehingga kondisi ini perlu diberi penanda yaitu lantai pelatihan akan berubah warna sesuai dengan kondisi ketika episode selesai. Berikut beberapa kondisi dimana berakhirnya setiap episode pelatihan:

a) Jika episode berakhir ketika agent pemain menang maka lantai akan berwarna hijau.

b) Jika episode berakhir ketika agent pengejar menang maka lantai akan berwarna biru.

c) Jika episode berakhir ketika waktu habis maka lantai akan berwarna merah.

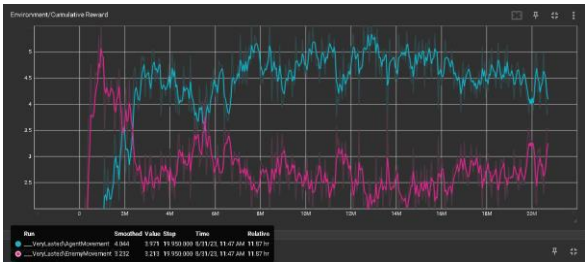
Hasil dari pelatihan ini akan ditunjukan dalam bentuk grafik Cumulative Reward, grafik Episode Length, grafik Policy Loss, dan garfik Value Lost. Berikut lampiran gambar 3.4 proses pelatihan yang telah dilakukan.



Gambar 3. 4 Proses Pelatihan Agent

C. Grafik Cumulative Reward

Grafik cumulative reward ini merupakan grafik yang digunakan untuk menunjukkan berapa banyak rata-rata hadiah yang diperoleh agen [29]. Pada grafik ini agent-agent memiliki grafik reward yang berbeda-beda ini menandakan agent-agent sedang bersaing untuk mendapatkan reward yang bagus. Pada MaxStep 6000000 agent pengejar mulai mengejar nilai reward tapi belum melampaui nilai reward agent pemain. Pada saat pelatihan dengan nilai MaxStep 6000000 agent-agent sudah mulai memiliki niali yang stabil sehingga nilai maxstep ini akan digunakan untuk menyematkan kecerdasan pada agent. Berikut grafik Cumulative Reward yang ditunjukan pada gambar 3.5.

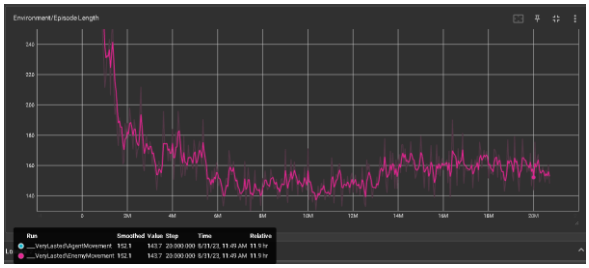


Gambar 3. 5 Grafik Cumulative Reward

D. Grafik Episode Length

Grafik episode length ini merupakan grafik yang digunakan untuk menunjukan perubahan panjang waktu episode atau juga bisa diartikan jumlah langkah atau tindakan yang diambil oleh agent untuk menuju tujuan tertentu [30]. Grafik ini menunjukan bagaimana percobaaan agent berinteraksi dengan menabrak target yang dituju dari episode ke episode. Dalam grafik ini ditunjukan nilai yang tinggi diawal pelatihan yang dapat diartikan agent memulai dengan percobaan yang banyak diawal untuk mencapai target. Seiring berjalanya pelatihan grafik mulai turun pada MaxStep

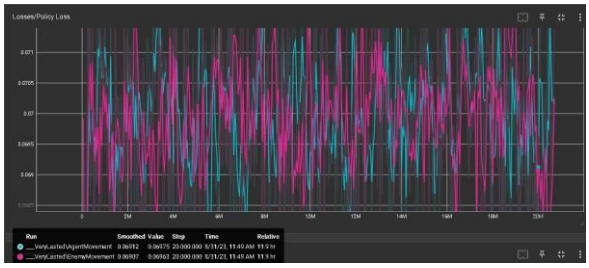
1000000 dan grafik mulai stabil pada saat MaxStep 6000000 sehingga dapat disimpulkan agent mulai menemukan cara yang efektif untuk menemukan target dengan rata-rata waktu percobaan lebih sedikit dari awal pelatihan. Jadi dapat disimpulkan agent-agent melakukan rata-rata waktu percobaan yang semakin membaik pada proses pelatihanya untuk mencoba menabrakan target reward positif dan menghindari reward negative. Berikut grafik episode length yang ditunjukan pada gambar 3.6.



Gambar 3. 6 Grafik Episode Length

E. Grafik Policy Loss

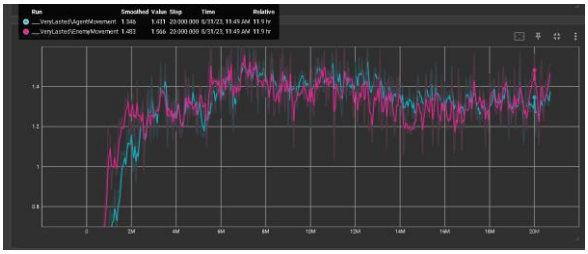
Grafik policy loss ini merupakan grafik yang digunakan untuk untuk memperbarui dan meningkatkan kebijakannya [28]. Grafik ini akan naik jika salah satu agent telah berhasil menyesuaikan kebijakan efektif sedangkan jika salah satu agent mengalami penurunan grafik maka agent tersebut belum dapat dapat menyesuaikan kebijakan secara efektif. Pada nilai policy ini ditentukan pada penelitian ini mengalami naik turun dari awal pelatihan karena agent-agent ini melakukan persaingan dan bukanya melakukan Kerjasama, jika salah satu agent mendapatkan reward positif maka nilai policy akan turun sedangkan agent pengejar akan mendapatkan nilai reward negative dan membuat nilai policynya menjadi naik begitu juga sebaliknya. Jadi dapat disimpulkan Berikut grafik policy loss yang ditunjukan pada gambar 3.7.



Gambar 3. 7 Grafik Policy Loss

F. Grafik Value Loss

Grafik value loss merupakan grafik yang digunakan untuk mengukur prediksi yang dilakukan oleh agent dalam pelatihan. Grafik value loss ini ditentukan oleh kesulitan yang dialami oleh agent, jika agent mengalami kesulitan dalam menentukan prediksi pada pelatihan maka grafik ini akan naik dan jika agent sudah mulai memprediksi tindakan yang akan dilakukan maka grafik value loss akan turun. Grafik value loss pada penelitian ini grafik dimulai dengan naiknya nilai value loss yang menandakan agent mengalami kesulitan menentukan prediksi yang terjadi pada lingkungan pelatihan. Kenaikan drastis ini berhenti pada Maxstep 2000000 dan mulai mengalami kestabilan dan menunjukan adanya penurunan sedikit pada maxstep 14000000 walaupun begitu nilai maxstep masih cenderung tinggi dan tidak mengalami penurunan yang drastis maka disimpulkan bahwa para agent masih mengalami kesulitan entuk menentukan prediksi pada pelatihan. Berikut grafik value loss yang ditunjukan pada gambar 3.8.



Gambar 3. 8 Grafik Value Loss

G. Pengujian Simulasi Kecerdasan Agent

Pada simulasi pengujian kecerdasan agent ini dibagi menjadi 2 kali yaitu pengujian pada tingkat kesulitan easy dan hard. Setiap tingkat kesulitan pengujian terdapat 2 agent yang akan dihitung performanya untuk menunjukan siapa yang lebih baik. Pada pengujian ini digunakan untuk menentukan waktu permainan pada setiap tingkatan dengan cara menghitung waktu agent untuk mendapatkan koin pada setiap percobaan. Saya memilih pengujian ini karena apakah hasil dari pengujian ini dapat menentukan kualitas agent menjadi lebih baik atau tidak selama proses pengujian. Berikut komponen pelatihan pada tiap kesulitan:

- Pada level kesulitan easy terdapat 2 agent pemain dan 1 agent pengejar. Dalam pelatihanya waktu yang diberikan adalah 60 detik dan terdapat 10 koin pada arena
- Pada level kesulitan hard terdapat 3 agent pemain dan 2 agent pengejar. Dalam pelatihanya waktu yang diberikan adalah 90 detik dan terdapat 15 koin pada arena

a) Kesulitan Easy

- Agent Pemain

Tabel 3. 1 Hasil Pengujian Tingkat Kesulitan Easy Agent Pemain

Percobaan ke-	**Total Koin Terambil (x) (Max=10 koin)**	**Agent Pemain yang di selamatkan**	**Waktu yang ditempuh (y) (Max=60 dalam detik)**	**Rata-rata koin terambil (x/10) *100%**
1	6	1	29	60%
2	6	2	26	60%

Percobaan ke-	Total Koin Terambil (x) (Max=10 koin)	Agent Pemain yang di selamatkan	Waktu yang ditempuh (y) (Max=60 dalam detik)	Rata-rata koin terambil (x/10) *100%
3	7	4	60	70%
4	6	1	28	60%
5	7	3	50	70%
6	4	1	21	40%
7	6	2	40	60%
8	5	4	44	50%
9	8	0	60	80%
10	7	2	60	70%

- Agent Pengejar

Tabel 3. 2 Hasil Pengujian Tingkat Kesulitan Easy Agent Pengejar

Percobaan ke-	Total Koin Terambil (x) (Max=10 koin)	Agent Pemain yang di selamatkan	Waktu yang ditempuh (y) (Max=60 dalam detik)	Rata-rata koin terambil (x/10) *100%
1	3	1	29	75%
2	4	2	26	67%
3	3	4	60	42%
4	3	1	28	75%
5	5	3	50	62%
6	3	1	21	75%
7	5	2	40	71%
8	6	4	44	60%
9	1	0	60	100%
10	2	2	60	50%

Pada pengujian tingkat kesulitan easy yang telah dilakukan oleh agent pemain pada tabel 3.1 dan agent pengejar pada tabel 3.2 yang dilakukan sebanyak 10 kali hasil pengujianya dipaparkan sebagai berikut:

Untuk pengujian agent pemain didapatkan perhitungan rata-rata dibawah ini:

- Rata-rata waktu yang ditempuh: 41.8 Detik
- Rata-rata koin yang terambil: 61%

Untuk pengujian agent pengejar didapatkan perhitungan rata-rata dibawah ini:

- Rata-rata waktu yang ditempuh: 41.8 Detik
- Rata-rata pengejar menangkap agent pemain: 66%

Kesimpulan yang didapatkan dari pengujian agent pada tingkat kesulitan easy adalah rata-rata koin yang diambil agent pemain 61% sedangkan rata-rata agent pengejar menangkap agent pemain adalah 66%. Hal ini menunjukan agent pengejar lebih unggul dari agent pemain. Untuk rata-rata waktu yang dicapai adalah 41.8 detik.

b) Kesulitan Hard

- Agent Pemain

Tabel 3. 3 Hasil Pengujian Tingkat Kesulitan Hard Agent Pemain

Percobaan ke-	Total Koin Terambil (x) (Max=15 koin)	Agent Pemain yang di selamatkan	Waktu yang ditempuh (y) (Max=90 dalam detik)	Rata-rata koin terambil (x/15) *100%
1	9	3	30	60%
2	15	5	85	100%
3	12	3	70	80%
4	15	4	63	100%
5	10	2	56	67%
6	9	3	23	60%
7	10	1	36	67%
8	14	1	60	93%
9	15	2	54	100%
10	11	1	33	73%

- Agent Pengejar

Tabel 3. 4 Hasil Pengujian Tingkat Kesulitan Hard Agent Pemain

Percobaan ke-	Total Koin Terambil (x) (Max=15 koin)	Agent Pemain yang di selamatkan	Waktu yang ditempuh (y) (Max=90 dalam detik)	Rata-rata koin terambil (x/15) *100%
1	6	3	30	67%
2	6	5	85	54%
3	5	3	70	62%
4	4	4	63	50%
5	5	2	56	71%
6	6	3	23	67%
7	4	1	36	80%
8	4	1	60	80%

Percobaan ke-	Total Koin Terambil (x) (Max=15 koin)	Agent Pemain yang di selamatkan	Waktu yang ditempuh (y) (Max=90 dalam detik)	Rata-rata koin terambil (x/15) *100%
9	2	2	54	50%
10	4	1	33	80%

Pada pengujian tingkat kesulitan hard yang telah dilakukan oleh agent pemain pada tabel 3.3 dan agent pengejar pada tabel 3.4 yang dilakukan sebanyak 15 kali hasil pengujianya dipaparkan sebagai berikut:

Untuk pengujian agent pemain didapatkan perhitungan rata-rata dibawah ini:

- Rata-rata waktu yang ditempuh: 51.4 Detik
- Rata-rata koin yang terambil: 79%

Untuk pengujian agent pengejar didapatkan perhitungan rata-rata dibawah ini:

- Rata-rata waktu yang ditempuh: 51.4 Detik
- Rata-rata pengejar menangkap agent pemain: 63%

Kesimpulan yang didapatkan dari pengujian agent pada tingkat kesulitan hard adalah rata-rata koin yang diambil agent pemain 79% sedangkan rata-rata agent pengejar menangkap agent pemain adalah 63%. Hal ini menunjukan agent pemain lebih unggul dari agent pengejar. Untuk rata-rata waktu yang dicapai adalah 51.4 detik.

H. Analisis Data Hasil Pengujian

Dalam tahap pengujian sebelumnya yang dilakukan oleh agent pemain dan agent pengejar yang dilakukan sebanyak 10 kali pada tiap tingkat kesulitan. Berdasarkan hasil pengujian dapat dibandingkan pada tingkat kesulitan easy rata-rata presentase milik agent pemain dan agent pengejar hasilnya jauh lebih besar agent pengejar yaitu 61% < 66%. Pada hasil pengujian tingkat kesulitan hard rata-rata presentase milik agent pemain dan agent pengejar hasilnya jauh lebih besar agent pemain yaitu 79% > 63%. Hal ini menunujkan bahwa agent pemain sudah mulai melakukan persaingan untuk mendapatkan reward seiring banyaknya pelatihan yang dilakukan dan akhirnya mendapatkan rata-rata reward yang lebih tinggi dari agent pengejar, walaupun begitu agent pengejar tetap berusaha mengejar agent pemain pada jumlah MaxSteps 6000000 tapi agent pengejar belum dapat melampaui agent pemain.

IV. CONCLUSSION

Hasil pengujian yang didapatkan pada tingkat kesulitan easy rata-rata koin yang diambil agent pemain 61% sedangkan rata-rata agent pengejar menangkap agent pemain adalah 66%. pada tingkat kesulitan hard rata-rata koin yang diambil agent pemain 79% sedangkan rata-rata agent pengejar menangkap agent pemain adalah 63%. Hal ini dapat disimpulkan dalam pelatihan agent pemain mulai memperbaiki kesalahan yang dilakukan dan dapat naik presentasenya menjadi 79% yang sebelumnya hanya 61%. 1. Dalam melakukan pelatihan 2 agent sekaligus dapat disimpulkan bahwa proses pelatihan cenderung membutuhkan waktu yang lama karena para agent menjadi bersaing satu dengan yang lain untuk memperebutkan nilai reward yang baik.

REFERENCES

[1] L. Caroux, K. Isbister, L. Le Bigot, and N. Vibert, “Player-video game interaction: A systematic review of current concepts,” *Comput Human Behav*, vol. 48, pp. 366–381, 2015, doi: 10.1016/j.chb.2015.01.066ï.

[2] C. R.-P. of the F. I. Workshop and undefined 1994, “Competition, coevolution and the game of tag,” *books.google.com*, Accessed: Nov. 20, 2023. [Online]. Available: https://books.google.com/books?hl=id&lr=&id=8a6uC0BbyjAC&oi=fnd&pg=PA59&dq=tag+game+players&ots=UkWInrhk7i&sig=pwSJ2pdjUMG8vdAAW-AggqsyBSs

[3] G. Grund, P. M. Nilsson, M. Larsson, O. Olsson, T. Foughman, and V. Gustafsson, “Realistic NPCs in Video Games Using Different AI Approaches,” 2016.

[4] J. T. Balint, J. M. Allbeck, and R. Bidarra, “Understanding everything NPCs can do: Metrics for action similarity in non-player characters,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Aug. 2018. doi: 10.1145/3235765.3235776.

[5] F. G. Glavin and M. G. Madden, “Skilled Experience Catalogue: A Skill-Balancing Mechanism for Non-Player Characters using Reinforcement Learning,” Jun. 2018, doi: 10.1109/CIG.2018.8490405.

[6] K. Merrick and M. Lou Maher, “Motivated Reinforcement Learning for Non-Player Characters in Persistent Computer Game Worlds.” [Online]. Available: www.secondlife.com

[7] A. Dobrovsky, U. M. Borghoff, and M. Hofmann, “Improving Adaptive Gameplay in Serious Games Through Interactive Deep Reinforcement Learning,” 2019, pp. 411–432. doi: 10.1007/978-3-319-95996-2_19.

[8] E. Alonso, M. Peter, D. Goumard, and J. Romoff, “Deep Reinforcement Learning for Navigation in AAA Video Games,” Nov. 2020, [Online]. Available: http://arxiv.org/abs/2011.04764

[9] A. Juliani *et al.*, "Unity: A General Platform for Intelligent Agents," Sep. 2018, [Online]. Available: http://arxiv.org/abs/1809.02627

[10] L. Universitet and +++ S.------L., "Application of machine learning to construct advanced NPC behaviors in Unity 3D.," 2021, Accessed: Nov. 21, 2023. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2:1604156

[11] K. Min *et al.*, "JORLDY: a fully customizable open source framework for reinforcement learning," Apr. 2022, [Online]. Available: http://arxiv.org/abs/2204.04892

[12] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *ieeexplore.ieee.org*, Accessed: Nov. 22, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8103164/

[13] P. Andersson, "Future-proofing Video Game Agents with Reinforced Learning and Unity ML-Agents," 2021.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Jul. 2017, [Online]. Available: http://arxiv.org/abs/1707.06347

[15] C. C.-Y. Hsu, C. Mendler-Dünner, and M. Hardt, "Revisiting Design Choices in Proximal Policy Optimization," Sep. 2020, Accessed: Nov. 22, 2023. [Online]. Available: http://arxiv.org/abs/2009.10897

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*. MIT Press, 1998.

[17] Y. Savid, R. Mahmoudi, R. Maskeliūnas, and R. Damaševičius, "Simulated Autonomous Driving Using Reinforcement Learning: A Comparative Study on Unity's ML-Agents Framework," *Information (Switzerland)*, vol. 14, no. 5, May 2023, doi: 10.3390/info14050290.

[18] C. Szepesvári, "Algorithms for Reinforcement Learning," 2009.

[19] Y. Li, "Reinforcement Learning in Practice: Opportunities and Challenges," Feb. 2022, Accessed: Dec. 27, 2022. [Online]. Available: http://arxiv.org/abs/2202.11296

[20] I. Uchendu *et al.*, "Jump-Start Reinforcement Learning," 2023.

[21] T. Aris, V. Ustun, and R. Kumar, "Learning to Take Cover on Geo-Specific Terrains via Reinforcement Learning," 2022.

[22] M. Lukas, I. Tomicic, and A. Bernik, "Anticheat System Based on Reinforcement Learning Agents in Unity," *Information (Switzerland)*, vol. 13, no. 4, Apr. 2022, doi: 10.3390/info13040173.

[23] O.-P. Juhola, "Creating Self-Learning AI using Unity Machine Learning," 2019.

[24] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative A g e n ts."

[25] A. Ardiansyah and E. Rainarli, "Implementasi Q-Learning dan Backpropagation pada Agen yang Memainkan Permainan Flappy Bird," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, vol. 6, no. 1, Feb. 2017, doi: 10.22146/JNTETI.V6I1.287.

[26] M. Rubak and F.-H. DI Priv-Doz Mag DI Drtechn Karl Michael Göschka, "Imitation Learning with the Unity Machine Learning Agents Toolkit."

[27] S. A. Murdivien and J. Um, "BoxStacker: Deep Reinforcement Learning for 3D Bin Packing Problem in Virtual Environment of Logistics Systems," *Sensors*, vol. 23, no. 15, Aug. 2023, doi: 10.3390/s23156928.

[28] A. L. Bayona and S. Takahashi Rodriguez, "Comparative Study of SAC and PPO in Multi-Agent Reinforcement Learning Using Unity ML-Agents," 2023.

[29] D. Björkberg, "Comparison of cumulative reward with one, two and three layered artificial neural network in a simple environment when using ml-agents." [Online]. Available: www.bth.se

[30] V. J. Hodge, R. Hawkins, and R. Alexander, "Deep reinforcement learning for drone navigation using sensor data," *Neural Comput Appl*, vol. 33, no. 6, pp. 2015–2033, Mar. 2021, doi: 10.1007/S00521-020-05097-X.