

# **Pengembangan Game *Tower Defense* Berbasis *Phaser Engine* dengan Implementasi *Q-Learning* untuk Musuh Adaptif**

**Muhammad Agil Perdana<sup>1</sup>, Rasim<sup>2\*</sup>, Rizky Rahman J.<sup>3</sup>**

<sup>1,2,3</sup>Fakultas Pendidikan Ilmu Pengetahuan Alam, Ilmu Komputer, Universitas Pendidikan Indonesia, Bandung, Indonesia

Email: <sup>1</sup>agilperdana1st@upi.edu, <sup>2\*</sup>rasim@upi.edu, <sup>3</sup>rizky\_rjp@upi.edu

(\*Email Corresponding Author: rasim@upi.edu)

Received: 12 Juli 2026 | Revision: 5 September 2026 | Accepted: 21 September 2026

## **Abstrak**

Kualitas perilaku musuh menjadi faktor penentu keterlibatan pemain pada game bergenre *tower defense*, namun algoritma *pathfinding* klasik seperti A\* tidak mempertimbangkan tingkat ancaman di sepanjang rute, sehingga musuh selalu menempuh rute yang sama meskipun terdapat pertahanan sangat kuat. Penelitian ini bertujuan mengimplementasikan algoritma *Q-Learning* pada unit musuh dalam game *tower defense* berbasis *phaser engine* untuk menghasilkan agen yang adaptif terhadap perubahan ancaman. Agen dirancang dengan state berupa posisi grid dan level ancaman diskrit pada empat arah, ruang aksi empat arah pergerakan, serta fungsi *reward* berjenjang. Training dilakukan selama 5.000 episode dengan skema *epsilon-greedy*, dan kemampuan adaptasi diuji pada lima skenario konfigurasi menara, dibandingkan dengan agen non-adaptif (*random walk* dan A\* statis). Hasil menunjukkan agen mencapai pembelajaran konvergen, dengan *reward* meningkat 96% dan *success rate* 99,84% selama training. Agen terbukti beradaptasi secara proporsional terhadap konfigurasi ancaman, ditunjukkan melalui perubahan pola jalur yang signifikan serta reaksi *real-time* terhadap menara yang muncul mendadak. Pada evaluasi akhir, agen mencapai *success rate* 100% dengan efisiensi jalur 1,15–1,47 kali jarak optimal. Dibandingkan *random walk*, efisiensi jalur agen 20–30 kali lebih baik, sedangkan terhadap A\* statis terdapat *trade-off* wajar (selisih 15–47% langkah) demi mengurangi eksposur ancaman. Penelitian ini membuktikan bahwa *Q-Learning* tabular efektif menghasilkan perilaku musuh adaptif dengan komputasi ringan untuk platform *browser*.

**Kata Kunci:** *Reinforcement Learning, Q-Learning, Musuh Adaptif, Tower Defense, Phaser Engine, Game Development Life Cycle, Kecerdasan Buatan*

## **Abstract**

Enemy behavior quality is a key factor determining player engagement in Tower Defense games, yet classical pathfinding algorithms such as A\* do not account for threat levels along a route, causing enemies to always take the same path even when players have placed strong defenses. This study implements the Q-Learning algorithm on enemy units in a Phaser Engine-based Tower Defense game to produce an agent capable of adapting to changing threat configurations. The agent's state combines grid position with discrete threat levels in four directions, with an action space of four movement directions and a tiered reward function. Training was conducted over 5,000 episodes using an epsilon-greedy scheme, and adaptability was tested across five tower configuration scenarios, compared against non-adaptive agents (random walk and static A\*). Results show the agent achieved convergent learning, with average reward increasing by 96% and a 99.84% success rate during training. The agent adapted proportionally to threat configurations, evidenced by significant path pattern changes and real-time reactions to suddenly placed towers. In final evaluation, the agent achieved a 100% success rate with a path efficiency ratio of 1.15–1.47 relative to the optimal distance. Compared to random walk, path efficiency was 20–30 times better, while against static A\* there was a reasonable trade-off (a 15–47% step difference) to reduce threat exposure. This study demonstrates that tabular Q-Learning effectively produces adaptive enemy behavior with lightweight computation suitable for browser-based platforms.

**Keywords:** *Reinforcement Learning, Q-Learning, Adaptive Enemy, Tower Defense, Phaser Engine, Game Development Life Cycle, Artificial Intelligence*

## **1. PENDAHULUAN**

*Tower defense* merupakan salah satu *genre game* strategi yang konsisten diminati, ditandai dengan keberadaan sejumlah judul mapan yang tetap aktif dimainkan dalam jangka waktu panjang, seperti *Plants vs. Zombies*, *Kingdom Rush*, dan *Bloons Tower Defense* [20]. Daya tarik utama *genre* ini terletak pada keseimbangan antara perencanaan strategis penempatan menara dan pengambilan keputusan taktis terhadap gelombang musuh yang datang, yang keduanya dipengaruhi langsung oleh kualitas perilaku unit musuh yang dihadapi pemain.

Perilaku *non-playable character* (NPC) dalam game, termasuk musuh pada *genre tower defense*, umumnya diimplementasikan menggunakan struktur logika eksplisit seperti *finite state machine* atau pohon keputusan, di mana pengembang secara manual mendefinisikan aturan respons untuk tiap kondisi yang diantisipasi [8], [11]. Pendekatan ini efektif untuk skenario sederhana, namun menghadapi keterbatasan mendasar seiring meningkatnya kompleksitas interaksi antara pemain-lingkungan. Pada *genre tower defense* secara khusus, musuh yang dikendalikan algoritma *pathfinding* klasik seperti A\* akan selalu menempuh rute dengan jarak tempuh tersingkat, tanpa mekanisme apa pun untuk mempertimbangkan tingkat bahaya di sepanjang rute tersebut.

*Reinforcement learning* (RL) menawarkan pendekatan alternatif, memungkinkan agen mempelajari kebijakan perilakunya sendiri melalui interaksi langsung dengan lingkungan, dipandu sinyal *reward*, dan telah menunjukkan keberhasilan pada berbagai domain seperti robotika dan sistem kendali otonom [25]. Namun, penerapan RL pada game *tower defense* berbasis *browser* menghadapi kendala praktis yaitu, platform *client-side* berbasis *JavaScript* umumnya tidak memiliki akses akselerasi perangkat keras (GPU) yang dibutuhkan pendekatan *Deep Reinforcement Learning* seperti *Deep Q-Network* (DQN) atau *Proximal Policy Optimization* (PPO) [15].

Sejumlah penelitian terdahulu telah mengeksplorasi penerapan RL pada game, namun menunjukkan sejumlah celah (*gap*) yang menjadi posisi penelitian ini. Dias, Foleiss, & Lopes [3] mengusulkan alur kerja RL untuk melatih agen yang memainkan *tower defense* sebagai pemain, bukan mengendalikan perilaku musuh. Bergdahl, Sestini, & Gisslén [2] dari EA SEED menerapkan agen hibrida berbasis PPO dan *heuristic* berskrip untuk pengujian otomatis level pada *Plants vs. Zombies*, mencapai *success rate* 57,12% dibanding 47,95% pada AI skrip; penelitian ini juga melaporkan bahwa RL sulit digeneralisasi lintas level pada game *tower defense* yang bersifat sangat spesifik-puzzle, dan tetap berfokus pada sisi pemain, bukan musuh. Kajian [17] terhadap penerapan RL untuk perilaku NPC secara umum menemukan bahwa RL menghasilkan NPC yang lebih adaptif dan taktis dibanding pendekatan *rule-based*, namun implementasinya pada game nyata masih terbatas pada studi kasus sederhana. Studi komparatif [18] terhadap skalabilitas algoritma navigasi grid (*Q-Learning*, PPO, *Monte Carlo*, *A\**) menemukan bahwa *Q-Learning* tabular tidak *scalable* pada grid berukuran besar, namun tetap kompetitif pada grid berukuran kecil-menengah. Sementara itu, Zhang et al. [28] menunjukkan *Q-Learning* tabular tetap efektif memunculkan perilaku strategis pada game spasial sederhana tanpa memerlukan aproksimasi fungsi neural.

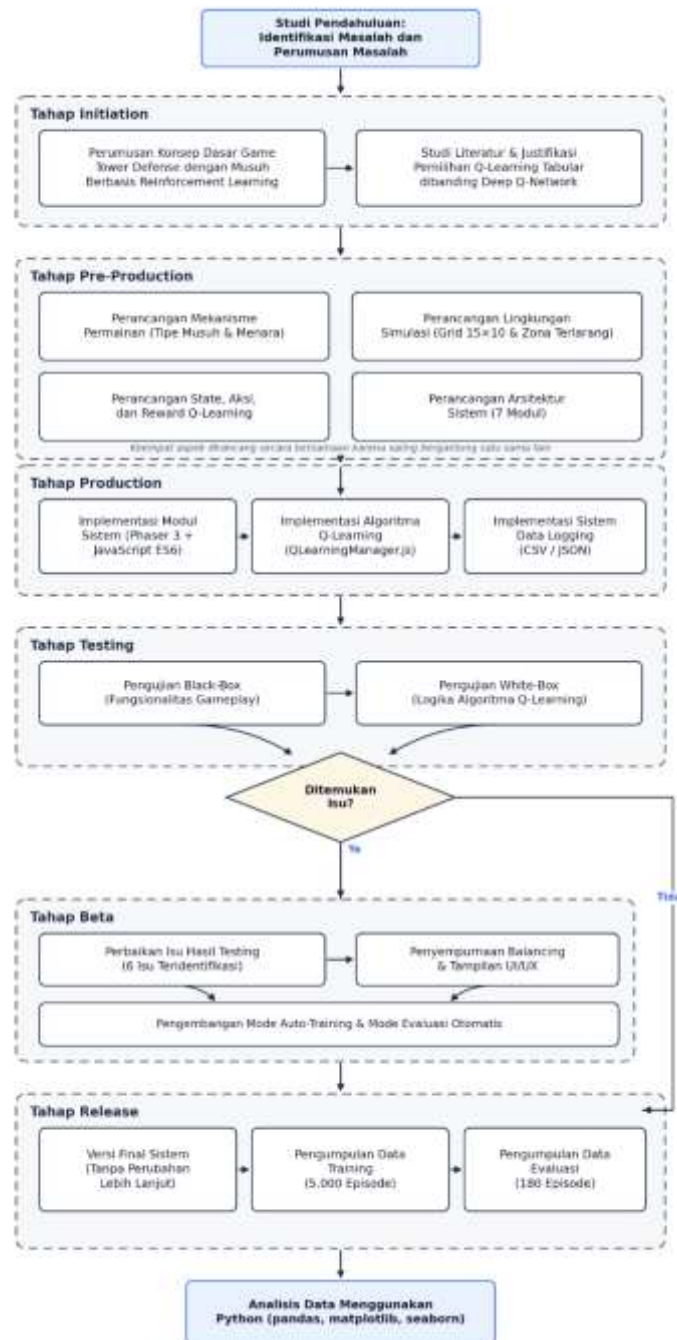
Berdasarkan penelusuran tersebut, ditemukan tiga celah utama. Pertama, mayoritas penelitian RL pada *tower defense* mengendalikan sisi pemain [2], [3], bukan sisi musuh sebagaimana difokuskan penelitian ini. Kedua, pendekatan *Deep RL* seperti PPO [2] membawa kebutuhan komputasi yang tidak sejalan dengan batasan platform *browser* yang ringan, sementara *Q-Learning* tabular telah terbukti tetap kompetitif pada *state space* kecil-menengah [18], sebagaimana karakteristik lingkungan grid 15×10 pada penelitian ini. Ketiga, belum ditemukan penelitian yang secara eksplisit merancang representasi *state* berbasis kesadaran ancaman lokal sekaligus menguji kemampuan adaptasi *real-time* terhadap perubahan konfigurasi ancaman yang dibuat pemain di tengah episode permainan [17], [18], aspek yang umumnya hanya diuji pada konfigurasi level yang tetap/statis.

Penelitian ini memposisikan diri pada celah tersebut dengan mengimplementasikan algoritma *Q-Learning* tabular pada unit musuh dalam game *tower defense* berbasis *phaser engine*. *State* agen dirancang sebagai kombinasi posisi grid dan level ancaman diskrit pada empat arah yang bersebelahan langsung, dengan fungsi *reward* berjenjang untuk mendorong perilaku yang proporsional terhadap besarnya ancaman, bukan sekadar respons biner. Penelitian ini bertujuan membuktikan bahwa agen yang dihasilkan mengalami proses pembelajaran yang konvergen, mampu beradaptasi terhadap perubahan konfigurasi ancaman baik statis maupun dinamis, serta menunjukkan kinerja yang secara terukur lebih baik dibandingkan pendekatan non-adaptif konvensional (*random walk* dan *A\** statis). Kontribusi penelitian ini diharapkan menjadi referensi teknis bagi pengembang game dalam merancang mekanisme AI musuh yang lebih adaptif dan responsif terhadap strategi pemain, dengan kebutuhan komputasi yang tetap ringan untuk platform berbasis *browser*.

## 2. METODOLOGI PENELITIAN

### 2.1 Tahapan Penelitian

Penelitian ini menggunakan desain penelitian pengembangan (*Research and Development/R&D*) [21], dengan produk berupa sistem game *tower defense* berbasis *phaser engine* yang unit musuhnya dikendalikan oleh agen *Q-Learning*. Pengembangan sistem dilakukan melalui pendekatan *Game Development Life Cycle* (GDLC) [22], sebuah varian model R&D yang disesuaikan untuk konteks pengembangan game, terdiri atas enam tahap berurutan sebagaimana ditunjukkan pada Gambar 1.



**Gambar 1.** Alur tahapan penelitian *Game Development Life Cycle* (GDLC)

Tahap *Initiation* berfokus pada perumusan konsep dasar game serta justifikasi pemilihan *Q-Learning* tabular dibandingkan *Deep Q-Network* (DQN), mempertimbangkan keterbatasan sumber daya komputasi pada platform *browser* tanpa akselerasi GPU [15]. Tahap *Pre-Production* mencakup perancangan mekanisme permainan, lingkungan simulasi berupa grid 15×10 satuan, representasi *state-aksi-reward* untuk algoritma *Q-Learning*, serta arsitektur sistem modular yang memisahkan logika *Q-Learning* dari logika visual/*gameplay* [9]. Tahap *Production* merupakan implementasi rancangan ke dalam kode program menggunakan *JavaScript* (ES6 Modules) dengan *Phaser 3*, mencakup mekanisme pergerakan agen berbasis *tween*, penembakan menara berbasis deteksi jarak, dan sistem pencatatan data (*data logging*) yang mengeksport hasil ke format CSV dan JSON. Tahap *Testing* dilakukan secara internal untuk mengidentifikasi kesalahan implementasi, baik pada aspek *gameplay* maupun pada logika algoritma *Q-Learning* itu sendiri, melalui kombinasi pengujian *black-box* dan *white-box*. Tahap *Beta* menindaklanjuti seluruh isu yang ditemukan pada tahap *Testing*, termasuk perbaikan pembaruan nilai *Q* pada kondisi terminal dan penambahan penalti *anti-oscillation* untuk mengatasi osilasi pergerakan agen. Tahap *Release* menghasilkan versi final sistem yang digunakan tanpa perubahan lebih lanjut selama seluruh proses pengumpulan data penelitian.

Data dikumpulkan melalui modul pencatatan yang terintegrasi ke dalam sistem game, mencatat data pada dua granularitas yaitu catatan tingkat-langkah (*step-level*) dan catatan tingkat-episode (*episode-level*), melalui dua skema pengumpulan yaitu sesi training sepanjang 5.000 episode, dan sesi evaluasi terkontrol pada lima skenario konfigurasi ancaman. Data yang terkumpul dianalisis secara kuantitatif-deskriptif menggunakan Python dengan pustaka pandas untuk pengolahan data tabular serta matplotlib dan seaborn untuk visualisasi.

## 2.2 Desain Algoritma Q-Learning

Inti metode penelitian ini terletak pada perancangan tiga komponen *Markov Decision Process* (MDP) [23]: *state*, aksi, dan fungsi *reward*, yang secara langsung menentukan apa yang dapat diamati dan dipelajari agen. *State* dirancang sebagai kombinasi posisi grid agen (x, y) beserta level ancaman pada empat arah grid yang bersebelahan langsung (atas, bawah, kiri, kanan). Level ancaman dihitung berdasarkan estimasi *damage per second* (DPS) gabungan seluruh menara yang jangkauannya mencakup suatu sel, kemudian didiskritisasi menjadi empat level [1], sebagaimana dirangkum pada Tabel 1. Diskritisasi ini diperlukan agar ruang *state* tetap terenumerasi secara terbatas, sebuah prasyarat mendasar bagi pendekatan Q-Learning tabular.

**Tabel 1.** Pemetaan DPS ke level ancaman diskrit

| Level Ancaman | DPS                      | Keterangan       |
|---------------|--------------------------|------------------|
| 0             | 0                        | Aman             |
| 1             | $0 < \text{DPS} \leq 20$ | Ancaman rendah   |
| 2             | $> 20$                   | Ancaman tinggi   |
| 3             | -                        | Sel di luar grid |

Ruang aksi dirancang sesederhana mungkin, terdiri atas empat aksi pergerakan diskrit (atas, bawah, kiri, kanan) tanpa opsi pergerakan diagonal. Fungsi *reward* dirancang melalui pendekatan *reward shaping* [7], memberikan sinyal parsial pada setiap langkah untuk mempercepat pembelajaran, dirangkum pada Tabel 2.

**Tabel 2.** Komponen fungsi reward

| Komponen                   | Nilai | Kondisi Pemicu              |
|----------------------------|-------|-----------------------------|
| Step penalty               | -1    | Melangkah                   |
| Penalty ancaman rendah     | -10   | Menuju sel level ancaman 1  |
| Penalty ancaman tinggi     | -50   | Menuju sel level ancaman 2  |
| Shaping jarak (mendekat)   | +2    | Jarak ke markas berkurang   |
| Shaping jarak (menjauh)    | -2    | Jarak ke markas bertambah   |
| Penalti aksi tidak valid   | -100  | Aksi ke luar grid/terhalang |
| Penalti anti-oscillation   | -500  | Kembali ke sel sebelumnya   |
| Reward terminal (berhasil) | +1000 | Mencapai markas             |
| Reward terminal (gagal)    | -1000 | Dihancurkan menara          |

Perancangan bertingkat pada penalti ancaman (-10 untuk level rendah, -50 untuk level tinggi) dimaksudkan agar agen mempelajari respons yang proporsional terhadap besarnya ancaman, bukan sekadar respons biner menghindari atau tidak. Nilai Q diperbarui mengikuti persamaan Q-Learning standar [14]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

Dengan  $\alpha$  sebagai *learning rate* dan  $\gamma$  sebagai *discount factor*. Pada kondisi terminal, komponen  $\gamma \cdot \max_{a'} Q(s', a')$  dihilangkan sepenuhnya karena tidak terdapat *next state* yang bermakna, sebagaimana ditemukan dan diperbaiki pada tahap *Testing* penelitian ini. Parameter final yang digunakan pada seluruh proses *training* ditampilkan pada Tabel 3.

**Tabel 3.** Parameter Q-Learning yang digunakan

| Parameter                    | Nilai                      |
|------------------------------|----------------------------|
| Learning rate ( $\alpha$ )   | 0,1                        |
| Discount factor ( $\gamma$ ) | 0,9                        |
| Epsilon awal ( $\epsilon$ )  | 1,0                        |
| Epsilon minimum              | 0,05                       |
| Skema epsilon decay          | Per-episode, faktor 0,9994 |
| Jumlah episode training      | 5.000                      |

Strategi *epsilon-greedy* digunakan untuk mengelola *trade-off* eksplorasi-eksploitasi [19], [27], dengan probabilitas eksplorasi menurun secara bertahap dari dominan eksplorasi menuju dominan eksploitasi sepanjang *training*. Untuk menguji kemampuan adaptasi, disusun empat skenario konfigurasi menara (*baseline* tanpa menara, blokir jalur atas, blokir jalur bawah, blokir kedua jalur) yang diterapkan secara bergantian sebagai kurikulum pembelajaran selama *training*, sekaligus digunakan sebagai skenario evaluasi akhir pada kebijakan yang telah dipelajari, ditambah satu skenario dinamis di mana menara dimunculkan mendadak di tengah episode untuk menguji kemampuan *re-planning* agen secara *real-time*. Kinerja agen kemudian dibandingkan dengan dua *baseline* non-adaptif, yaitu agen bergerak acak (*random walk*) dan agen  $A^*$  statis yang disederhanakan menjadi *breadth-first search* karena seluruh langkah pada grid berbobot seragam [12].

### 3. HASIL DAN PEMBAHASAN

#### 3.1 Lingkungan dan Skenario Pengujian

Lingkungan simulasi direpresentasikan sebagai grid dua dimensi berukuran  $15 \times 10$  satuan, dengan titik *spawn* musuh pada koordinat (1,2) dan titik markas pemain pada koordinat (13,8), menghasilkan jarak Manhattan minimum 18 langkah yang digunakan sebagai pembanding rasio efisiensi jalur. Dua zona terlarang ditetapkan di sekitar titik *spawn* dan markas untuk mencegah strategi penempatan menara yang mengunci pergerakan agen secara tidak representatif. Empat skenario konfigurasi menara disusun untuk menguji adaptasi agen yaitu Skenario A (*baseline*, tanpa menara), Skenario B (blokir jalur atas, 4 menara), Skenario C (blokir jalur bawah, 4 menara), dan Skenario D (blokir kedua jalur, 8 menara), dengan satu kolom celah ( $x=8$ ) yang selalu dibiarkan tidak terisi menara pada tiap konfigurasi untuk menjamin ketersediaan jalur alternatif. Proses *training* dilakukan sebanyak 5.000 episode dengan keempat skenario diterapkan bergantian sebagai kurikulum pada interval 1.250 episode, sedangkan evaluasi akhir dilakukan pada 180 episode independen ( $40 \text{ episode} \times 4 \text{ skenario statis}$ , ditambah 20 episode skenario dinamis) dengan epsilon dikunci pada nilai minimum agar perilaku yang teramati murni mencerminkan kebijakan yang telah dipelajari.

#### 3.2 Proses Pembelajaran (*Training Convergence*)

Bukti keberhasilan proses pembelajaran ditelusuri melalui empat indikator yang saling melengkapi. Pertama, kurva *reward* per episode menunjukkan tren peningkatan yang konsisten, rata-rata *reward* pada 1.000 episode pertama yang awalnya sebesar -12.245, meningkat menjadi -378 pada 1.000 episode terakhir, peningkatannya lebih dari 96%. *Reward* tidak mencapai nilai positif secara rata-rata karena struktur *step penalty* -1 pada setiap langkah membuat akumulasi penalti tetap mendominasi meskipun agen berhasil mencapai markas, nilai yang relevan diinterpretasikan adalah tren menjadi semakin kurang negatif, mengindikasikan lintasan yang semakin efisien. Sebanyak 65 episode (1,3%) pada rentang awal *training* tercatat memiliki *reward* ekstrem (serendah -166.308), seluruhnya terjadi ketika epsilon masih mendekati 1,0 sehingga agen banyak bergerak secara eksploratif murni, fenomena yang lazim ditemui pada tahap awal *reinforcement learning* dengan strategi *epsilon-greedy* [10] dan bukan indikasi kegagalan sistem.

Kedua, kurva epsilon menunjukkan penurunan mulus dan monoton dari 1,0 menuju 0,05 tepat pada episode ke-5.000, sesuai skema *decay* per-episode dengan faktor 0,9994. Percepatan peningkatan *reward* paling signifikan terjadi pada rentang episode 0–1.000, ketika epsilon berada pada kisaran 0,55–1,0, sementara peningkatan menjadi lebih landai setelah episode 3.000 ketika kebijakan sudah relatif stabil.

Ketiga, rata-rata perubahan nilai  $Q$  ( $\Delta Q$ ) menunjukkan pola naik-kemudian-turun yang khas pada algoritma *temporal difference learning*,  $\Delta Q$  meningkat dari 11,44 pada episode awal menuju puncak 15,34 pada sekitar episode ke-464, sebelum menurun bertahap hingga 3,58 pada episode ke-4.000. Pola ini terjadi karena informasi *reward* terminal baru mulai menyebar mundur (*backpropagate*) ke *state-state* sebelumnya melalui mekanisme *bootstrapping* setelah agen memperoleh cukup pengalaman. Dua momen kenaikan  $\Delta Q$  di tengah tren penurunan (episode ke-2.500 dan ke-3.750) bertepatan presisi dengan jadwal pergantian skenario kurikulum, mengindikasikan agen secara aktif memperbarui kebijakannya ketika dihadapkan pada konfigurasi ancaman baru.

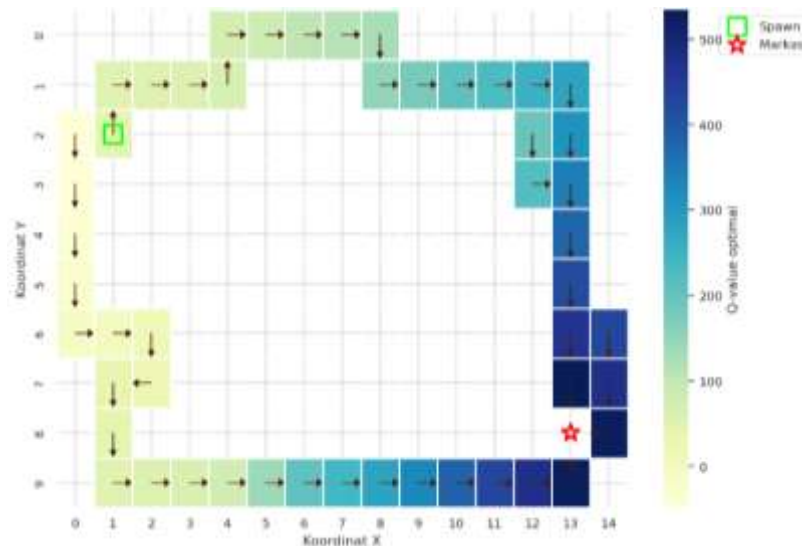
Keempat, *success rate* selama training mencapai 99,84% dari 4.992 episode, dengan 244 dari 250 kelompok 20-episode mencatatkan *success rate* sempurna 100%. Enam kelompok dengan penurunan ringan (85–95%) seluruhnya terjadi pada rentang episode 3.340–4.000, tepat di sekitar transisi ke Skenario D, konsisten dengan lonjakan  $\Delta Q$  pada periode yang sama dan memperkuat interpretasi bahwa periode tersebut merupakan fase penyesuaian singkat terhadap kondisi lingkungan baru.

#### 3.3 Representasi Pengetahuan Agen (*Q-Table*)

*Q-table* hasil *training* mencatat 283 *state* unik dari 1.132 pasangan *state-action* ( $283 \text{ state} \times 4 \text{ aksi}$ ), jauh lebih kecil dari batas atas teoritis 38.400 kombinasi ( $150 \text{ posisi grid} \times 256 \text{ kombinasi level ancaman}$ ), karena sebagian besar kombinasi tersebut secara fisik mustahil terbentuk pada lingkungan permainan ini. Nilai  $Q$  berkisar antara -605,76 hingga 534,42, dengan rata-rata -73,92 dan simpangan baku 207,47; sebanyak 366 pasangan (32,3%) bernilai positif

dan 765 pasangan (67,6%) bernilai negatif, sedangkan hanya 1 pasangan yang masih bernilai nol, menunjukkan hampir seluruh *state-action* telah mengalami pembaruan.

Visualisasi kebijakan optimal (Gambar 2) menunjukkan pola yang koheren. Pada mayoritas koordinat, arah panah mengarah menuju markas melalui jalur atas maupun bawah grid, dengan nilai Q tertinggi (500–534) pada koordinat yang bertetangga langsung dengan markas, sesuai prinsip dasar *Q-Learning* bahwa *state* yang secara temporal lebih dekat dengan *reward* terminal memiliki estimasi nilai lebih tinggi.



**Gambar 2.** Visualisasi kebijakan optimal (*policy heatmap*)

Untuk membuktikan kesadaran agen terhadap ancaman secara konkret, ditemukan 118 dari 283 *state* (41,7%) yang memiliki ancaman menara pada minimal satu arah. Tabel 4 merangkum tiga studi kasus representatif.

**Tabel 4.** Studi kasus nilai Q pada *state* dengan ancaman menara

| Koordinat (x,y) | Ancaman(atas, bawah,kiri, kanan) | Aksi Terpilih | Q(aksi terpilih) |
|-----------------|----------------------------------|---------------|------------------|
| (3,4)           | (1,0,0,2)                        | Kiri          | -43,2            |
| (5,6)           | (2,0,0,0)                        | Kanan         | +177,5           |
| (4,1)           | (0,2,0,2)                        | Atas          | +65,5            |

Ketiga studi kasus secara konsisten menunjukkan agen memilih aksi yang menjauhi arah dengan level ancaman tertinggi, dan pada dua kasus tetap memperoleh nilai Q positif meskipun berada di dekat menara, mengindikasikan agen menemukan rute yang secara neto tetap menguntungkan karena tetap mengarah mendekati markas.

### 3.4 Kemampuan Adaptasi terhadap Perubahan Lingkungan

Pada Skenario A (*baseline*), agen mencapai *success rate* 100% dengan rata-rata 21,1 langkah (mendekati optimal 18 langkah) dan tanpa *damage*, dengan 97,5% episode konsisten menempuh jalur atas. Hasil ini menjadi acuan bagi seluruh perbandingan skenario berikutnya.

Pada skenario satu jalur diblokir, ditemukan hasil yang tidak simetris. Skenario B (blokir atas) meningkatkan rata-rata langkah sebesar 23,7% (menjadi 26,1) dan memunculkan rata-rata *damage* 2,2, namun agen tetap mempertahankan jalur atas sebagai rute dominan (95%) dengan menyelip melalui kolom celah. Sebaliknya, Skenario C (blokir bawah) nyaris tidak berdampak (rata-rata langkah 20,8, *damage* tetap nol), karena jalur bawah memang bukan rute yang secara alami dipilih agen sejak kondisi *baseline*. Temuan ini menunjukkan kemampuan adaptasi agen bersifat proporsional terhadap tingkat gangguan yang benar-benar dialami pada rute yang sesungguhnya dilalui, bukan penyesuaian menyeluruh terhadap seluruh kemungkinan konfigurasi lingkungan.

Pada Skenario D (kedua jalur diblokir), *success rate* tetap 100% dengan biaya adaptasi terbesar yaitu rata-rata 26,4 langkah (+25% dari *baseline*) dan *damage* 2,73. Perbedaan paling signifikan terletak pada distribusi pola jalur, di mana proporsi episode yang berpindah ke jalur bawah meningkat menjadi 7,5% (dibanding 2,5–5% pada skenario lain),

menunjukkan agen mulai benar-benar memanfaatkan rute alternatif ketika tekanan ancaman pada rute favorit meningkat signifikan.

Pengujian skenario dinamis, di mana menara dimunculkan mendadak pada langkah ke-6 pergerakan agen, menunjukkan *reward* anjlok tajam dari +1 menjadi -53 tepat pada momen menara muncul, membuktikan perubahan *state* akibat menara baru langsung terdeteksi dan memengaruhi keputusan agen pada langkah berikutnya tanpa jeda adaptasi. Agen tetap melanjutkan pergerakan menuju markas dengan *success rate* 100% dan rata-rata 23,4 langkah, lebih efisien dibanding Skenario B statis (26,1 langkah), kemungkinan karena posisi agen yang masih relatif dekat dengan *spawn* saat menara muncul.

### 3.5 Evaluasi Kinerja Kuantitatif dan Perbandingan *Baseline*

Rekapitulasi seluruh skenario evaluasi ditampilkan pada Tabel 5. Seluruh 180 episode evaluasi mencatatkan *success rate* 100%, tidak satu pun berakhir dengan agen dihancurkan menara.

**Tabel 5.** Rekapitulasi hasil evaluasi seluruh skenario

| Skenario            | Success rate | Langkah rata-rata | Efisiensi jalur | Rata-rata kerusakan diterima |
|---------------------|--------------|-------------------|-----------------|------------------------------|
| A — Baseline        | 100%         | 21,1              | -43,2           | 0,00                         |
| B — Blokir Atas     | 100%         | 26,1              | +177,5          | 2,20                         |
| C — Blokir Bawah    | 100%         | 20,8              | +65,5           | 0,00                         |
| D — Blokir Keduanya | 100%         | 26,4              |                 | 2,73                         |
| Dinamis             | 100%         | 23,4              | 1300            | 0,50                         |

Rasio efisiensi jalur (langkah aktual dibagi jarak Manhattan minimum) berkisar 1,15–1,47 di seluruh skenario, tergolong wajar untuk grid dengan pergerakan terbatas pada empat arah ortogonal. Simpangan baku waktu tempuh pada Skenario B (17,4) dan D (13,9) jauh lebih besar dibanding skenario lain (1,4–1,9), dengan nilai maksimum ekstrem (133 langkah pada B, 102 langkah pada D) mengindikasikan sebagian kecil episode di mana agen sempat tertahan sementara di sekitar zona ancaman sebelum akhirnya menemukan jalan keluar.

Perbandingan dengan dua agen *baseline* non-adaptif ditampilkan pada Tabel 6. Agen *random walk* disimulasikan dengan 200 episode independen per skenario, mencapai *success rate* 94,0–97,0% namun membutuhkan 604,9–673,1 langkah rata-rata, lebih dari 20 kali lipat dibanding agen *Q-Learning*. Agen A\* statis (disederhanakan menjadi *breadth-first search* karena bobot langkah seragam) selalu menemukan jalur optimal 18 langkah pada keempat skenario karena tidak memiliki mekanisme untuk mempertimbangkan risiko.

**Tabel 6.** Perbandingan rata-rata langkah: Q-Learning vs baseline

| Skenario            | A* Statis | Q-Learning | Random Walk |
|---------------------|-----------|------------|-------------|
| A — Baseline        | 18        | 21,1       | 604,9       |
| B — Blokir Atas     | 18        | 26,1       | 642,2       |
| C — Blokir Bawah    | 18        | 20,8       | 673,1       |
| D — Blokir Keduanya | 18        | 26,4       | 635,2       |

Ketiga agen menunjukkan karakteristik fundamental berbeda. *Random walk*, meskipun mencapai *success rate* tinggi berkat ukuran grid terbatas dan tanpa batas waktu ketat, memerlukan langkah yang sangat tidak efisien sehingga pada implementasi nyata dengan mekanisme pertempuran aktif kemungkinan besar akan dihancurkan menara jauh sebelum mencapai markas. A\* statis unggul mutlak dalam efisiensi jalur namun sama sekali tidak memiliki mekanisme menghindari zona berisiko tinggi. Agen *Q-Learning* menempati posisi kualitatif berbeda dari keduanya: jumlah langkahnya lebih panjang dibanding A\* statis (selisih 15–47%), namun selisih ini merupakan *trade-off* yang disengaja untuk mengurangi kontak terhadap ancaman menara, sementara efisiensinya tetap 20–30 kali lebih baik dibanding *random walk*.

### 3.6 Implementasi

Sistem diimplementasikan menggunakan *Phaser 3* (versi 3.55.2) yang dimuat melalui CDN, dengan bahasa pemrograman *JavaScript (ES6 Modules)* berjalan sepenuhnya di sisi klien tanpa server *backend*. Arsitektur sistem dipisahkan ke dalam tujuh modul dengan tanggung jawab terisolasi [9], *MainScene* sebagai orkestrator utama,

Pathfinding.js untuk representasi grid dan validasi pergerakan, QLearningManager.js untuk logika inti algoritma *Q-Learning*, DataLogger.js untuk pencatatan data, serta Enemy.js, Tower.js, dan Bullet.js pada lapisan objek permainan. Pemisahan ini memungkinkan logika *Q-Learning* terisolasi sepenuhnya dari logika visual/*gameplay*, sehingga perubahan mekanisme permainan tidak memerlukan modifikasi pada algoritma pembelajaran itu sendiri. Antarmuka sistem mencakup panel *dashboard* samping yang menampilkan kesehatan markas, skor, dan gelombang secara *real-time* (Gambar 3), tampilan *gameplay* aktif dengan menara dan musuh bergerak (Gambar 4), serta indikator visual validasi penempatan menara berupa kotak merah untuk zona terlarang dan kotak hijau untuk area valid.



**Gambar 3.** Tampilan dashboard antarmuka sistem



**Gambar 4.** Kondisi gameplay aktif dengan menara dan musuh

Dua mode operasi tambahan dikembangkan untuk mendukung pengumpulan data berskala besar yaitu mode *Auto-Training* yang menerapkan konsep *curriculum learning* [26] dengan empat konfigurasi menara diterapkan bergantian disertai percepatan waktu simulasi, dan mode Evaluasi yang mengotomasi pengujian kelima skenario secara berurutan dengan statistik musuh dijaga tetap untuk menjamin keadilan perbandingan antar skenario.

### 3.7 Diskusi

Perilaku yang teramati sepanjang hasil penelitian ini dapat ditelusuri langsung ke struktur *reward* yang dirancang mulai dari *step penalty* mendorong efisiensi jalur, *shaping* berbasis jarak untuk mengarahkan pergerakan konsisten menuju markas, dan penalti ancaman bertingkat menciptakan insentif proporsional yang menjelaskan mengapa agen bersedia menerima sedikit damage demi menempuh rute yang secara keseluruhan tetap menguntungkan. Sifat *state* yang hanya mempertimbangkan kondisi ancaman pada empat sel bertetangga langsung, bukan peta ancaman global, menjelaskan mengapa agen bereaksi secara lokal dan reaktif terhadap perubahan lingkungan, sebagaimana terlihat jelas pada skenario dinamis, bukan melakukan perencanaan ulang jalur secara menyeluruh.

Penelitian ini memiliki sejumlah keterbatasan yang perlu diungkapkan secara transparan. Representasi *state* berbasis tabel efektif untuk ukuran *state space* penelitian ini (283 *state* teramati dari 38.400 kemungkinan teoritis), namun berpotensi menghadapi *curse of dimensionality* apabila granularitas ancaman atau jangkauan pengamatan diperluas [4], [14]. Perbandingan dengan agen *baseline* pada sub-bab 3.5 disimulasikan secara terpisah tanpa menyertakan mekanisme pertempuran, sehingga hasil perbandingan bersifat estimasi konservatif yang menguntungkan kedua agen *baseline*. Dari sisi implikasi praktis, hasil penelitian ini menunjukkan bahwa musuh berbasis *Q-Learning* mampu menghasilkan perilaku yang terasa responsif terhadap strategi pemain, di mana pemain yang memblokir satu jalur akan melihat musuh menyesuaikan rute secara nyata, yang berpotensi meningkatkan tantangan dan keterlibatan pemain dibandingkan musuh dengan pola pergerakan statis atau murni berbasis *pathfinding* klasik.

## 4. KESIMPULAN

Penelitian ini berhasil mengimplementasikan algoritma *Q-Learning* tabular pada unit musuh dalam game *tower defense* berbasis *phaser engine*, melalui representasi *state* berupa kombinasi posisi grid dan level ancaman diskrit pada empat arah yang bersebelahan langsung, ruang aksi berupa empat pilihan pergerakan ortogonal, serta fungsi *reward* berjenjang yang menggabungkan *step penalty*, penalti ancaman proporsional, *shaping* berbasis jarak, penalti *anti-oscillation*, dan *reward* terminal. Proses pembelajaran agen terbukti mengalami konvergensi yang konsisten selama 5.000 episode *training*, ditunjukkan oleh empat indikator yang saling menguatkan yaitu *reward* rata-rata meningkat lebih dari 96% dari fase awal menuju akhir *training*, epsilon menurun mulus sesuai rancangan dari 1,0 menuju 0,05, perubahan nilai  $Q$  ( $\Delta Q$ ) menunjukkan pola konvergensi khas *temporal difference learning*, serta *success rate* mencapai 99,84% sepanjang *training* dengan periode penyesuaian singkat yang bertepatan tepat dengan pergantian skenario kurikulum. Kemampuan adaptasi agen terhadap perubahan konfigurasi ancaman terbukti melalui pola penghindaran ancaman yang konsisten dan proporsional pada level *state* individual, perubahan pola jalur yang berbanding lurus dengan tingkat ancaman pada rute yang benar-benar dilalui agen, serta reaksi *real-time* terhadap ancaman yang muncul mendadak di tengah episode dengan latensi deteksi hanya satu langkah. Dibandingkan agen *random walk* non-adaptif, agen *Q-Learning* menunjukkan peningkatan efisiensi jalur 20–30 kali lipat pada seluruh skenario yang diuji, sekaligus menunjukkan *trade-off* yang wajar dan dapat dijelaskan dibandingkan agen  $A^*$  statis, dengan selisih 15–47% jumlah langkah yang merepresentasikan biaya yang secara sengaja ditanggung agen demi mengurangi kontak terhadap ancaman menara, sebuah kapabilitas yang sama sekali tidak dimiliki algoritma *pathfinding* klasik. Penelitian lanjutan disarankan mengeksplorasi pendekatan *function approximation* seperti *Deep Q-Network* untuk mengakomodasi representasi *state* yang lebih kaya, mengimplementasikan agen *baseline* secara langsung di dalam game agar aspek *survivability* turut terukur, serta melibatkan pemain manusia sesungguhnya untuk menguji adaptasi agen terhadap strategi penempatan menara yang lebih beragam dan tidak terduga.

## REFERENCES

- [1] S. B. Alaoui and A. Saoud, "How to discretize continuous state-action spaces in Q-learning: A symbolic control approach," in 2024 IEEE 63rd Conference on Decision and Control (CDC), 2024, pp. 8314–8319, doi: <https://doi.org/10.1109/cdc56724.2024.10886490>.
- [2] J. Bergdahl, A. Sestini, and L. Gisslén, "Reinforcement learning for high-level strategic control in tower defense games," in 2024 IEEE Conference on Games (CoG), 2024, pp. 1–8, doi: <https://doi.org/10.1109/cog60054.2024.10645621>.
- [3] A. Dias, J. Foleiss, and R. P. Lopes, "Reinforcement learning in tower defense," in Communications in Computer and Information Science, 2022, doi: [https://doi.org/10.1007/978-3-030-95305-8\\_10](https://doi.org/10.1007/978-3-030-95305-8_10).
- [4] F. Ebrahimbadi, H. Kebriaei, and S. J. Ashtiani, "Enhancing Q-learning via state-space design for active battery balancing," Battery Energy, vol. 5, pp. 1–17, 2026, doi: <https://doi.org/10.1002/bte2.70084>.
- [5] A. Filipović, "The role of artificial intelligence in video game development," Kultura Polisa, 2023, doi: <https://doi.org/10.51738/kpolisa2023.20.3r.50f>.
- [6] N. Golian, V. Golian, and I. Afanasieva, "Black and white-box unit testing for web applications," Bulletin of NTU "KhPI." Series: System Analysis, Control and Information Technologies, 2022, doi: <https://doi.org/10.20998/2079-0023.2022.01.13>.

- [7] S. Ibrahim, M. Mostafa, A. Jnadi, H. Salloum, and P. Osinenko, "Comprehensive overview of reward engineering and shaping in advancing reinforcement learning applications," *IEEE Access*, vol. 12, pp. 175473–175500, 2024, doi: <https://doi.org/10.1109/access.2024.3504735>.
- [8] D. Jagdale, "Finite state machine in game development," *Int. J. Adv. Res. Sci. Commun. Technol.*, 2021, doi: <https://doi.org/10.48175/ijarsct-2062>.
- [9] S. Jarzabek and K. Kumar, "On interplay between separation of concerns and genericity principles: Beyond code weaving," *Comput. Sci. Inf. Syst.*, vol. 13, pp. 731–758, 2016, doi: 10.2298/csis160129028j.
- [10] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Inf. Fusion*, 2022, doi: <https://doi.org/10.1016/j.inffus.2022.03.003>.
- [11] S. Lapeyrate, "Reasoning with ontologies for non-player character's decision-making in games," *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, vol. 18, pp. 303–306, 2022, doi: <https://doi.org/10.1609/aiide.v18i1.21980>.
- [12] H. Liu, J. Cao, and Z. Wang, "Research on path planning of mobile robot in complex environment," *Discov. Appl. Sci.*, vol. 7, 2025, doi: <https://doi.org/10.1007/s42452-025-06713-y>.
- [13] F. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, and Y. Yu, "A survey on model-based reinforcement learning," *Sci. China Inf. Sci.*, vol. 67, 2022, doi: <https://doi.org/10.1007/s11432-022-3696-5>.
- [14] Y. Lyu, A. Côme, Y. Zhang, and M. S. Talebi, "Scaling up Q-learning via exploiting state–action equivalence," *Entropy*, vol. 25, 2023, doi: <https://doi.org/10.3390/e25040584>.
- [15] N. Mehanna and W. Rudametkin, "Caught in the game: On the history and evolution of web browser gaming," in *Companion Proc. ACM Web Conf. 2023*, 2023, doi: <https://doi.org/10.1145/3543873.3585572>.
- [16] N. Midzic and M. Novak, "Performance comparison of WebAssembly and Phaser in procedural maze generation," *Eurasia Proc. Sci. Technol. Eng. Math.*, 2025, doi: <https://doi.org/10.55549/epstem.1753850>.
- [17] S. J, "Improving non-playable-characters (NPC) in games using reinforcement learning," *Int. J. Res. Appl. Sci. Eng. Technol.*, 2025, doi: <https://doi.org/10.22214/ijraset.2025.76473>.
- [18] S. Paul and J. Deshmukh, "Multi agent path finding using evolutionary game theory," *arXiv:2212.02010*, 2022, doi: 10.48550/arxiv.2212.02010.
- [19] H. N and P. A. G, "A brief study of deep reinforcement learning with epsilon-greedy exploration," *Int. J. Comput. Digit. Syst.*, 2022, doi: <https://doi.org/10.12785/ijcds/110144>.
- [20] A. Qaffas, "An operational study of video games' genres," *Int. J. Interact. Mob. Technol.*, vol. 14, pp. 175–194, 2020, doi: <https://doi.org/10.3991/ijim.v14i15.16691>.
- [21] A. Rahayu, "Metode penelitian dan pengembangan (R&D): Pengertian, jenis dan tahapan," *DIAJAR: J. Pendidikan Pembelajaran*, 2025, doi: <https://doi.org/10.54259/diajar.v4i3.5092>.
- [22] R. Ramadan and Y. Widyani, "Game development life cycle guidelines," in *2013 Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, 2013, pp. 95–100, doi: <https://doi.org/10.1109/icacsis.2013.6761558>.
- [23] C. Shi, R. Wan, R. Song, W. Lu, and L. Leng, "Does the Markov decision process fit the data: Testing for the Markov property in sequential decision making," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 8807–8817, doi: <https://doi.org/10.48550/arxiv.2002.01751>.
- [24] J. Stenros, "The game definition game," *Games Cult.*, vol. 12, pp. 499–520, 2016, doi: <https://doi.org/10.1177/1555412016655679>.
- [25] K. Ter, O.-O. Ajayi, and D. Udekwe, "Taxonomy and trends in reinforcement learning for robotics and control systems: A structured review," *arXiv:2510.21758*, 2025, doi: <https://doi.org/10.48550/arxiv.2510.21758>.
- [26] Y. Wen, S. Li, M. Fu, Y. Yang, X. Wang, and P. Liu, "Curriculum reinforcement learning with measurable task representation learning," *Neural Netw.*, vol. 202, p. 109019, 2026, doi: <https://doi.org/10.1016/j.neunet.2026.109019>.
- [27] V. Zangirolami and M. Borrotti, "Dealing with uncertainty: Balancing exploration and exploitation in deep recurrent reinforcement learning," *Knowl.-Based Syst.*, vol. 293, p. 111663, 2023, doi: <https://doi.org/10.1016/j.knosys.2024.111663>.
- [28] J. Zhang, Z. Rong, G. Zheng, J. Zhang, and L. Chen, "The emergence of cooperation via Q-learning in spatial donation game," *J. Phys. Complex.*, vol. 5, 2024, doi: <https://doi.org/10.1088/2632-072x/ad3f65>.