IMPLEMENTASI *GAME TAG* DENGAN *MACHINE LEARNING* MENGGUNAKAN *UNITY*

SKRIPSI

MICHAEL ALESSANDRO

191401113

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

IMPLEMENTASI *GAME TAG* DENGAN *MACHINE LEARNING* MENGGUNAKAN *UNITY*

SKRIPSI

MICHAEL ALESSANDRO

191401113

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024

PENGESAHAN

Judul	: Implementasi *Game Tag* Dengan *Machine Learning* Menggunakan *Unity*
Kategori	: SKRIPSI
Nama	: Michael Alessandro
Nomor Induk Mahasiswa	: 191401113
Program Studi	: SARJANA (S-1) ILMU KOMPUTER
Departemen	: ILMU KOMPUTER
FAKULTAS	: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA
Tanggal Sidang	: Senin, 29 April 2024

Komisi Pembimbing :

Pembimbing I	Pembimbing II
Dr. Jos Timanta Tarigan, S.Kom., M.Sc.	Sri Melvani Hardi, S.Kom., M.Kom.
NIP. 198501262015041001	NIP. 198805012015042006

Diketahui / disetujui oleh
Program Studi S-1 Ilmu Komputer
Ketua

Dr. Amalia, ST, MT.
NIP. 197812212014042001

PENGAKUAN

IMPLEMENTASI GAME TAG DENGAN MACHINE LEARNING MENGGUNAKAN UNITY

SKRIPSI

Saya mengakui, dengan pengecualian beberapa kutipan dan ringkasan yang telah dikutip sumbernya, skripsi ini adalah hasil penelitian saya sendiri.

Medan, April 2024

Michael Alessandro
191401113

PENGHORMATAN

Puji syukur penulis ucapkan kehadirat Tuhan Yang Maha Esa atas rahmat dan penyertaannya-Nya penulis dapat menyelesaikan skripsi ini dengan baik guna memenuhi syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi S-1 Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara.

Pada kesempatan ini, penulis ingin menyampaikan rasa terimakasih kepada semua pihak yang telah membantu penulis dalam Implementasi skripsi ini dalam bentuk doa, arahan, bimbingan, ilmu, dan dukungan, khususnya :

1. Bapak Prof. Dr. Muryanto Amin, S.Sos., M.Si.selaku Rektor Universitas Sumatera Utara
2. Ibu Dr. Maya Silvi Lidya, B.Sc, M.Sc selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi
3. Ibu Dr. Amalia, S.T, M.T selaku Ketua Prodi S-1 Ilmu Komputer Universitas Sumatera Utara
4. Fauzan Nurahmadi, S.Kom., M.Cs selaku Dosen Pembimbing Akademik
5. Dosen Pembimbing I, Dr. Jos Timanta Tarigan, S.Kom., M.Sc. yang telah membimbing dan memberikan saran kepada penulis dalam pembuatan skripsi
6. Dosen Pembimbing II, Sri Melvani Hardi, S.Kom., M.Kom. yang telah membimbing dan memberikan saran kepada penulis dalam penyusunan skripsi
7. Seluruh staff pengajar dan pegawai Fakultas Ilmu Komputer dan Teknologi Informasi yang telah membantu dan mempermudah proses penyusunan skripsi ini.
8. Ibunda Sulastry dan Ayahanda Zainuddin yang selalu memberikan kasih sayang, semangat, dan doa sehingga penulis dapat menyelesaikan skripsi ini.
9. Keluarga penulis yang telah memberikan kasih sayang, doa, serta dukungan kepada penulis dalam penyusunan skripsi ini.
10. Teman penulis, Andika yang telah memberikan support, doa, semangat dan dukungan kepada penulis dalam penyusunan skripsi ini.
11. Seluruh teman seperjuangan, Kevin, Kirey, Hafiz Alfiandi, Putra, Christian, Victory, Filbert, Christine, Roshan, Irsyad, Ivan, Niko, Ricky.

12. Keluarga besar IMILKOM yang telah memberikan pengalaman dan wawasan yang berharga kepada penulis.
13. Serta semua pihak yang terlibat, baik secara langspung atau tidak langsung yang tidak dapat penulis sebutkan satu persatu.

Medan, April 2024

Michael Alessandro

ABSTRAK

Implementasi *game* dengan menggunakan *machine learning* sangat jarang dibahas dan ditemukan, namun perkembangannya sangat pesat karena berbasis *digital*, oleh karena itu diperlukan penelitian yang membahas tentang kombinasi *machine learning* dan *game*. *Game* ini akan dibuat menggunakan *unity* yang sudah satu program dengan *unity machine learning agent.* Pada penelitian ini dibangun *game tag* dengan menggunakan *unity machine learning agent* yang sangat *extensible. Unity machine learning agent* menggunakan algoritma *proximal policy optimization (PPO)* yang diklasifikasikan sebagai metode gradien kebijakan untuk melatih jaringan kebijakan *agent.* Pada implementasi *game tag* ini adalah *game* yang memiliki minimal 2 orang player atau lebih dimana 1 akan menjadi *catcher* dan satu lagi pemain akan menjadi *runner.* Pada *game* ini dilihat seberapa efisien *enemy agent* dapat dilatih dengan variabel rata-rata waktu tersisa sehingga dapat menciptakan strategi untuk mencapai tujuan yang telah ditentukan secepat mungkin serta mendapatkan generasi yang paling baik. Pada *game* ini dilakukan pengujian untuk melihat kehandalan dari *unity machine learning agent* dalam menangkap *player* didapatkan dari generasi ke-1 sampai generasi ke-5. Hasil yang didapatkan sampai pengujian generasi ke-5 adalah generasi ke 3 memiliki waktu terburuk yaitu 70 detik rata-rata waktu tersisa dan yang terbaik yaitu generasi ke-5 dengan waktu tersisa *enemy agent* rata-rata sekitar 252 detik dari waktu permainan 300 detik. Kemudian generasi ke-5 memiliki nilai penangakapan yang gagal terhadap *user* adalah 0 sehingga penelitian dibuat sampai generasi ke 5 dikarenakan sudah dianggap handal dalam bermain bersama *player*.

Kata kunci: *machine learning*, *game*, *unity machine learning agent*, akurasi.

IMPLEMENTATION OF TAG GAMES WITH MACHINE LEARNING USING UNITY

ABSTRACT

The implementation of games using machine learning is very rarely discussed and discovered, but its development is very rapid because it is digital-based, therefore research is needed that discusses the combination of machine learning and games. This game will be created using Unity which is already a program with the Unity machine learning agent. In this research, a tag game was built using a highly extensible Unity machine learning agent. Unity machine learning agent uses the proximal policy optimization (PPO) algorithm which is classified as a policy gradient method to train the agent policy network. In the implementation of this tag game, it is a game that has a minimum of 2 or more players, where 1 will be the catcher and the other player will be the runner. In this game, we see how efficiently the enemy agent can be trained with the variable average remaining time so that it can create a strategy to achieve the specified goals as quickly as possible and get the best generation. In this game, testing was carried out to see the reliability of the Unity Machine Learning Agent in capturing players from the 1st generation to the 5th generation. The results obtained until testing the 5th generation are that the 3rd generation has the worst time, namely 70 seconds with an average remaining time and the best is the 5th generation with an average remaining enemy agent time of around 252 seconds from a game time of 300 seconds. Then the 5th generation had a value of 0 failed arrests for users, so research was carried out until the 5th generation because it was considered reliable in playing with players.

Keywords: machine learning, game, unity machine learning agent, accuracy.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sekarang ini, dimana *machine learning* berkembang pesat dan sudah banyak *game* berbasis digital, sehingga banyak pengujian game yang dapat menggunakan *machine learning* untuk melakukan simulasi antara interaksi program atau interaksi program dengan pengguna. *Machine learning* dapat mempercepat pemrosesan data dan pengujian *game* dalam hal melihat *bug* dan mengambil sejumlah besar data pengalaman pengguna dalam *game* (Dr. Yaya, 2020).

Proses pengujian menggunakan pohon generasi yang dibuat dalam *machine learning* dapat dilakukan secara massal dan mudah. Namun, karena kurangnya penelitian tentang menggabungkan *game* dan *machine learning*, sehingga ada kekurangan data yang cukup untuk menghasilkan *game* dengan *machine learning*. Ini disebabkan oleh penelitian akademis yang lebih sedikit dibandingkan penggunaan komersial yang membahas tentang *machine learning* dalam *game*, sehingga mengakibatkan kurangnya minat untuk mengangkat topik ini (Fontes & Gay, 2021).

Menurut (Taye, 2023) *machine learning* adalah tentang bagaimana membuat komputer yang dapat menjawab situasi atau keadaan secara otomatis melalui pengalaman. Menurut (Heryadi et al., 2020) *machine learning* merupakan salah satu cabang ilmu Kecerdasan Buatan (*Artificial Intelligence*) yang berkembang sangat cepat dan telah menyebabkan masalah klasifikasi, regresi, *clustering*, dan *anomaly detection* pada berbagai bidang dapat diatasi lebih efisien.

Ada dua cara penggunaan *machine learning* dalam *game* yang sering digunakan ,yaitu: *neural network* with *backpropagation* yang biasanya dianggap sebagai metode pembelajaran terawasi dengan metode klasik yang memiliki basis pengetahuan yang besar. Serta *neural network with evolutionary algorithms* yang merupakan *neural network* tipikal yang menggunakan algoritma evolusioner untuk mengoptimalkan dirinya sendiri dan menawarkan penyetelan berbagai parameter yang lebih efisien dalam *neural network* (Nork et al., 2018).

Game merupakan suatu hasil berdasarkan situasi konflik dimana kepentingan dua pihak atau lebih dihadapkan yang didefinisikan oleh beberapa set aturan atau parameter (Asmiatun & Putri, 2017) *Machine learning* adalah kemampuan *machine* untuk belajar atau beradaptasi terhadap suatu situasi atau keadaan yang baru (Dinata & Hasdyna, 2020). Pada *machine*

learning Implementasi parameter dilakukan dengan *pathfinder* yaitu jarak dari titik A ke titik B, kemudian *finite machine state* yang menentukan keadaan(*state*) *machine* dan bergerak dengan informasi yang diperoleh (Sharma, 2019).

Video game yang akan diterapkan dalam penelitian ini adalah *game tag* atau dalam bahasa Indonesia yaitu *game* kejar-kejaran dimana pemainnya akan dikejar oleh *enemy agent*. Tujuan dalam permainan adalah agar pemain tidak tertangkap oleh *enemy agent* dalam jangka waktu tertentu dan agar *enemy agent* mengejar dan menangkap pemain. Dalam permainan ini agar tidak monoton menggunakan beberapa kendala yaitu tembok yang bisa berbentuk seperti *maze* atau biasa disebut juga Labirin yang merupakan sebuah sistem jalur yang rumit, berliku-liku, serta memiliki banyak jalan buntu(Lawson, 2019). Sehingga dengan rintangan, pemain dan *agent* dapat melakukan strategi yang berbeda serta *enemy agent* dalam mengejar pemain menggunakan radius tangkap yang berbentuk lingkaran dimana jika pemain berada di daerah lingkaran dalam jangka waktu tertentu pemain akan kalah.

Program yang akan dibuat pada penelitian ini menggunakan *unity* yaitu aplikasi pengolah gambar, grafik, suara, input, dan lain-lain yang ditujukan untuk membuat suatu *game* (Nugroho & Pramono, 2017), serta memiliki fitur *cross platform* yang dapat membuat *game* 3D, 2D, VR (*Virtual Reality*), dan AR (*Augmented Reality*) (Hanggoro et al., n.d.) serta menggunakan *machine learning* dalam membuat *agent* menggunakan *unity machine learning Agent*, yaitu perangkat lunak yang dapat mengembangkan karakter menggunakan *machine learning*. *machine learning* akan diterapkan pada *agent* dalam beberapa generasi agar mempercepat *machine learning* secara bertahap dengan membuat parameter atau *rules* yang dapat menjadi panduan *agent* menggunakan *reward* dan *punishment* dalam pembelajaran sehingga *agent* dapat berjalan secara efisien.

Penelitian ini diharapkan menghasilkan *enemy agent* yang dapat bermain dengan pemain dan satu *player agent* yang menggantikan pemain selama simulasi dengan melatih *agent* tersebut menggunakan *unity machine learning agent* agar *agent* dapat mengetahui saat *game* dimulai, melakukan tujuan, dapat bergerak, dapat mengikuti aturan, dan dapat membuat strategi.

Dalam permainan, tujuan *enemy agent* adalah untuk menangkap pemain dalam waktu sesingkat mungkin dan *agent* juga perlu mengetahui kapan permainan dimulai untuk menghemat waktu serta *agent* dilatih menggunakan *unity machine learning agent* dengan menggunakan metode *reward* dan *punishment*. Dimana jika *agent* melakukan hal yang dapat

memangkas waktu dalam mencapai tujuan maka akan diberikan poin tambahan namun jika *agent* melakukan hal yang menambah waktu untuk mencapai tujuan maka akan mengurangi poin yang didapat. Sehingga dapat menghasilkan strategi untuk menangkap *player* setelah latihan sesuai aturan yang telah ditentukan dengan menggunakan *unity machine learning agent*.

Gerakan dalam *game* dibatasi karena *game* dibuat dalam 2D. Jadi ada 4 gerakan yang bisa dilakukan yaitu: bergerak ke depan, bergerak ke belakang, bergerak ke kiri, dan bergerak ke kanan. Serta mengatur waktu atau waktu permainan agar pemain bisa memenangkan permainan. Waktu yang ditentukan adalah 5 menit dimana jika waktu mencapai menit ke-5 maka enemy *agent* diberikan penalti (pengurangan poin) maksimal hingga 0 tetapi poin terbanyak diberikan pada awal permainan dan dikurangi sesuai dengan penambahan waktu. Namun untuk *player agent* yang paling banyak mengganti poin pemain di menit ke-5 dan nilai 0 di awal permainan.

Penggunaan *unity machine learning agent* dilakukan dengan menentukan parameter yang diperlukan kemudian menjalankan banyak simulasi. Setelah simulasi dijalankan, generasi terbaik (poin terbanyak) akan dipilih dan simulasi akan diulangi dengan *agent* yang dipilih. Ini akan diulang sampai *agent* telah memenuhi standar yang ditetapkan oleh penulis.

Standar pada *agent* dilakukan dengan beberapa pengujian yang dimainkan langsung oleh penulis jika rata-rata waktu yang dibutuhkan *enemy agent* untuk menangkap pemain tidak banyak berubah maka *agent* melakukan simulasi selanjutnya dan dilakukan pengujian ulang.

1.3 Rumusan Masalah

Seberapa efisien *enemy agent* dapat dilatih dengan variabel rata-rata waktu tersisa sehingga dapat menciptakan strategi untuk mencapai tujuan yang telah ditentukan secepat mungkin serta mendapatkan generasi yang paling baik.

1.2 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut :

1. Program menggunakan *unity* dalam pembuatan *game*.
2. *Agent* yang digunakan adalah *unity machine learning agent*.
3. Parameter pengukuran yang digunakan untuk menganalisis kinerja *agent* adalah waktu.
4. *Agent* yang dibuat berjumlah 2 buah, 1 machine untuk dilatih sebagai lawan dan 1 *agent* sebagai pengganti pemain saat simulasi.
5. Bahasa Pemrograman menggunakan C#.

1.4 Tujuan Penelitian

Untuk Mengimplementasikan *game tag* dengan *machine learning* untuk menghasilkan *agent* yang dapat bermain bersama *player* serta, melakukan *goal*(tujuan) yaitu menangkap player sebelum tenggang waktu yang ditentukan dan Mengetahui waktu rata-rata *agent* dalam menangkap *player*.

1.5 Manfaat Penelitian

Diharapkan *agent* yang dihasilkan mampu memberikan kemudahan terkait pengembangan penelitian yang lebih lanjut yang berhubungan dengan *machine learning* pada *game* yang menggunakan *unity machine learning agent*.

1.6 Metodologi Penelitian

Metode penelitian yang dilakukan dalam penelitian ini adalah:

1. Studi Pustaka

   Pada tahap ini merupakan tahapan peninjauan pustaka berupa buku, artikel, *paper*, jurnal, makalah, penelitian dan situs-situs *internet* lainnya yang didokumentasikan dalam bentuk jurnal untuk memperoleh informasi dan data yang terkait dengan penggunaan *machine learning, game*, dan penggunaan *machine learning* dalam *game*.

2. Analisa dan Perancangan

   Pada tahap ini merupakan tahapan analisa terhadap apa yang diperlukan dalam penelitian *game* yang menggunakan *machine learning* dan melakukan perancangan dalam bentuk *flowchart*.

3. Implementasi

   Pada tahap ini merupakan tahapan dimana rancangan general arsitektur sebelumnya akan diimplementasikan dengan *unity engine* untuk menghasilkan *game tag* yang menggunakan *unity machine learning agent*.

4. Pengujian

   Pada tahap ini merupakan tahapan dimana aplikasi akan diuji apakah aplikasi sudah memenuhi standar yang sudah ditetapkan sebelumnya serta perbaikan pada *error* yang ada pada aplikasi

5. Dokumentasi

   Pada tahap ini merupakan tahapan dimana penelitian di dokumentasi dan penulisan laporan dan kesimpulan akhir dalam bentuk skripsi untuk memperlihat hasil dari penelitian yang telah dibuat.

1.7 Penelitian Relevan

Berikut adalah beberapa penelitian yang berkaitan dengan Implementasi *game tag* dengan *machine learning* menggunakan *unity*:

1. Penelitian yang berjudul "*Tic-Tac-Toe Learning Using Artificial Neural Networks*" oleh (Rajab et al., n.d.) menggunakan *machine learning* untuk memprediksi gerakan dalam *tic-tac-toe* menggunakan program *python*.
2. Penelitian yang berjudul "Implementasi *Machine Learning* pada *Game* Balap 2D dengan Rintangan Berbasis Unity" oleh (Amdanni, 2022) menggunakan *machine learning* untuk melatih *agent* untuk balapan dengan *unity* menggunakan *library MLAgents*.
3. Penelitian yang berjudul "*Chess Results Analysis Using Elo Measure with Machine Learning*" oleh (Thabtah et al., 2020) menggunakan *machine learning* untuk memprediksi pengaruh *elo*(kumpulan performa permainan yang dahulu) untuk menentukan persentase kemenangan sebelum *game* dimulai yang datanya didapat dari *server chess online*.
4. Penelitian yang berjudul "Pembuatan *Game RPG Adventure of The Dungeon*" oleh (Azmi et al., 2020) menggunakan *finite state machine*(kecerdasan buatan) untuk mengatur pertahanan musuh pada *game RPG(role-playing game)*.
5. Penelitian yang berjudul "Penerapan Model Pembelajaran dengan Metode *Reinforcement Learning* Menggunakan Simulator Carla" oleh (Dharma & Tambunan, 2021) menggunakan CNN(*Convolutional Neural Network*) untuk menentukan gerakan mobil carla dengan memberi nilai positif atau negatif dalam gerakan.
6. Penelitian yang berjudul "Pembuatan *Game The Legend of Timun Mas* Dengan Menggunakan *Unity*" oleh (Septian, 2020) membuat suatu *game mobile* menggunakan bahasa C# dengan menggunakan metode Pengembangan *system Multimedia Development Life Cycle* (MDLC) pada *unity engine*.
7. Penelitian yang berjudul "Pengembangan Aplikasi Pengetahuan Bahasa Pemograman Dasar Dan Lanjut Berbasis Android Menggunakan Metode *Game Development Life Cycle*" oleh (Alfasyam, 2023) membuat suatu *game* yang menyediakan pengalaman belajar yang menyenangkan dan kompetitif dalam mempelajari bahasa pemrograman dasar dan lanjut.

1.8. Sistematika Penulisan

Sistematika penulisan skripsi ini dibagi menjadi 5 bagian yang terdiri dari :

BAB 1 : Pendahuluan

Pada bagian pertama skripsi ini akan dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB 2 : Tinjauan Pustaka

Mengenai beberapa teori tentang *video game* dan *machine learning* serta kombinasi *game* dengan *machine learning*.

BAB 3 : Analisis dan Perancangan Sistem

Pada bagian ketiga skripsi ini akan diuraikan mengenai rumusan masalah, analisis proses dan Implementasi *game* berbasis *unity*.

BAB 4 : Implementasi dan Pengujian

Bab ini berisi tentang penerapan *unity machine learning agent* di aplikasi *unity* untuk membuat *game tag* dan mendapatkan hasil pengujian *game*.

BAB 5 : Kesimpulan dan Saran

Bab ini memuat kesimpulan yang diperoleh dari penelitian dan saran dari peneliti yang dapat membantu penelitian selanjutnya.

BAB 2

TINJAUAN PUSTAKA

2.1 Video Game

Permainan atau *game* adalah suatu sistem buatan dimana *player* dibatasi oleh aturan-aturan yang telah ditentukan sebelumnya dan juga terlibat dalam konflik yang sedang berlangsung. *Game* juga dapat didefinisikan sebagai sistem yang memiliki aturan dasar dan juga memiliki nilai akhir yang bersifat kuantitatif. *Player* yang bermain *game* mencoba untuk mendapatkan hasil yang mereka inginkan dan dapat berhubungan secara emosional dengan hasil tersebut.

Video game ditemukan sebagai "permainan elektronik di mana pemain mengontrol gambar di layar video" (Merriam-Webster), "permainan dimana pemain menggunakan kontrol elektronik untuk memindahkan gambar di *layer* atau permainan di mana Anda menekan tombol untuk mengontrol dan memindahkan gambar pada layar"(*Oxford Advanced Learner's Dictionary*).

2.2 Machine Learning

Machine Learning adalah salah satu cabang kecerdasan buatan (*Artificial Intelligence*) dimana *machine* dikembangkan untuk dapat belajar sendiri tanpa arahan dari penggunanya dan berdasarkan disiplin ilmu lain seperti statistik, matematika dan data mining sehingga *machine* dapat belajar dengan menganalisis data tanpa perlu diprogram ulang atau diinstruksikan. .

Dalam hal ini *machine learning* memiliki kemampuan untuk mendapatkan data yang ada dengan perintahnya sendiri dan juga dapat mempelajari data yang ada dan data yang diperolehnya sehingga dapat melakukan tugas tertentu. Tugas *machine learning* dapat dilakukan dengan sangat bervariasi, tergantung pada apa yang mereka pelajari.

Machine learning adalah salah satu jenis kecerdasan buatan di mana komputer menggunakan data dalam jumlah besar untuk mempelajari cara melakukan tugasnya (*Oxford Advanced Learner's Dictionary)* serta belajar mengembangkan *model* dan aturan dengan diberi data dalam jumlah besar sehingga mampu meningkatkan kinerjanya sendiri dengan terus memasukkan data baru ke dalam *model* statistik yang ada (*Merriam-Webster Dictionary*).

2.3 Unity Engine

Unity engine adalah aplikasi pengolah gambar, grafik, suara, input, dan lain-lain yang ditujukan untuk membuat suatu *game* (Nugroho & Pramono, 2017), serta memiliki fitur *cross platform* seperti: *Windows*, *Android*, *Mac*, *iOS*, PS3 dan lainnya yang dapat membuat *game* 3D, 2D, VR (*Virtual Reality*), dan AR (*Augmented Reality*). (Hanggoro et al., n.d.) *Engine Unity* dapat digunakan secara gratis dan mudah digunakan dengan *library* yang disediakan oleh *development team* dan pengembang lainnya, sehingga memudahkan dalam mempelajari cara membuat *game*.

2.4 Unity Machine Learning Agent

Unity machine learning agents adalah *toolkit* gratis yang dikembangkan oleh *unity* untuk meningkatkan *game* kecerdasan buatan dengan *machine learning*. Biasanya saat mengembangkan kecerdasan buatan untuk sebuah *game*, Anda akan memeriksa untuk melihat apakah kondisi tertentu benar lalu menjalankan tindakan tertentu. Bentuk kecerdasan buatan ini berfungsi, tetapi pada intinya hal itu dapat diprediksi dan dibatasi. (Haq et al., 2022)

Unity machine learning agent menggunakan algoritma proximal policy optimization (PPO) yang diklasifikasikan sebagai metode gradien kebijakan untuk melatih jaringan kebijakan *agent*. Kebijakan jaringan adalah fungsi yang digunakan *agent* untuk membuat keputusan. Intinya, untuk melatih jaringan kebijakan yang tepat, PPO melakukan pembaruan kebijakan kecil (ukuran langkah), sehingga *agent* dapat mencapai solusi optimal dengan handal. Sebuah langkah yang terlalu besar dapat mengarahkan kebijakan ke arah yang salah, sehingga kecil kemungkinannya untuk pulih; langkah yang terlalu kecil akan menurunkan efisiensi secara keseluruhan. Akibatnya, PPO mengimplementasikan fungsi klip yang membatasi pembaruan kebijakan agen agar tidak terlalu besar atau terlalu kecil.

Kemajuan terkini dalam kecerdasan buatan didorong oleh hadirnya lingkungan simulasi yang semakin realistis dan kompleks. Namun, banyak lingkungan yang ada memberikan visual yang tidak realistis, fisika yang tidak akurat, kompleksitas tugas yang rendah, perspektif agen yang terbatas, atau kapasitas interaksi yang terbatas antara *agent* buatan. Selain itu, banyak *platform* tidak memiliki kemampuan untuk mengonfigurasi simulasi secara fleksibel, sehingga menjadikan lingkungan simulasi sebagai kotak hitam dari perspektif sistem pembelajaran.

Dalam karya ini, kami mengusulkan taksonomi baru dari platform simulasi yang ada

dan mendiskusikan platform umum kelas tingkat tinggi yang memungkinkan pengembangan lingkungan pembelajaran yang kaya akan kompleksitas *visual*, fisik, tugas, dan sosial. Kami berpendapat bahwa *agent game modern* secara unik cocok untuk bertindak sebagai platform umum dan sebagai studi kasus yang menguji *agent Unity* dan *Unity ML-Agents Toolkit open source*.

Kami kemudian mensurvei penelitian yang dimungkinkan oleh *Unity* dan *Unity ML-Agents Toolkit*, mendiskusikan jenis penelitian yang dapat difasilitasi oleh platform umum yang fleksibel, interaktif, dan mudah dikonfigurasi.(Juliani et al., 2018)

2.5 Game Tag

Game tag dalam permainan dimana terdapat 2 pemain atau lebih dimana 1 *player* akan menjadi pengejar(*catcher*) dan sisa *player* sebagai yang dikejar. Tujuan utama permainan ini adalah pengejar harus menyentuh *player* lainnya sehingga pemain tersebut menjadi pengejar.(Britannica.2018)

Game tag memiliki area yang di batasi untuk bermain dimana biasanya memiliki rintangan untuk membuat *game* lebih seru dan menarik. Pada *game tag*, *Catcher* harus mendekati dan menandai *player*, kemudian *player* yang sudah di tandai akan menjadi *catcher* dan *catcher* yang menandai *player* akan menjadi *runner* kembali.

BAB 3

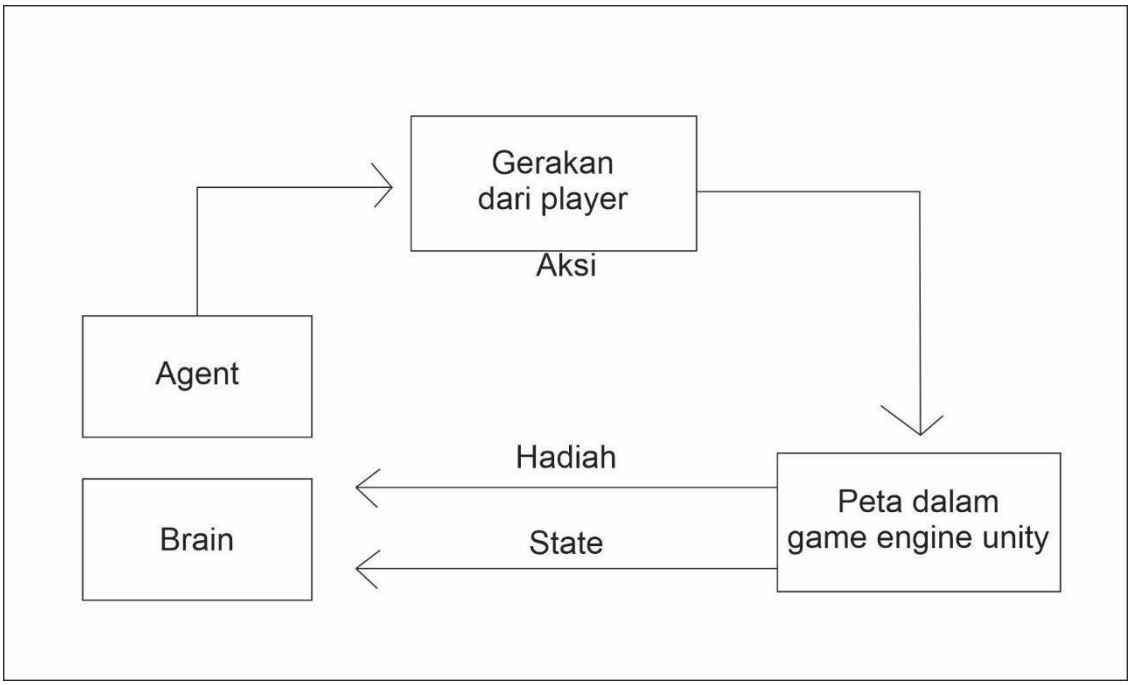
ANALISIS DAN PERANCANGAN SISTEM

3. 1. Analisis Sistem

Analisis sistem penelitian ini menggunakan *unity machine learning agents* untuk membuat *game tag* dengan membuat sistem dimana musuh akan mengejar *player* yang melibatkan langkah-langkah menggunakan sistem *reinforcement learning*.

3.1.1 Analisis Masalah

Dalam menganalisis masalah diidentifikasi permasalahan dan risiko dari sistem yang akan dibangun. Tujuannya agar sistem yang dibangun dapat berjalan sesuai dengan yang diharapkan. Permasalahan yang akan diidentifikasi adalah bagaimana menciptakan agent yang dapat bergerak dan mengambil keputusan layaknya manusia. Dengan menggunakan *unity machine learning agent* dalam Implementasi *agent*, *agent* dapat memprediksi pergerakan lawan dan mengambil pergerakan yang paling optimal untuk memenangkan permainan, sehingga menjadi solusi dari permasalahan tersebut.



Gambar 3.1 Analisis Masalah

3.1.2 Analisis Kebutuhan

Dalam menganalisis kebutuhan akan dilakukan identifikasi kebutuhan dalam perancangan suatu sistem. Analisis kebutuhan dibedakan menjadi dua yaitu kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan fungsional mencakup berbagai proses yang akan dikerjakan oleh sistem. Selain itu, kebutuhan fungsional mencakup informasi-informasi yang diperlukan oleh sistem dan hasil dari sistem tersebut.

1. Kebutuhan fungsional sistem yaitu sebagai berikut:

    1. *Neural Network*

        Sistem dapat memberikan otak untuk dilawan oleh *enemy agent* dan *player agent* yang ada. *Neural network* otak sebagai berikut::

        1. *Enemy Agent* : merupakan musuh pemain dan bertugas mengejar pemain yang mempunyai otak *ChasingEnemyAgent*.
        2. *Player Agent* : Merupakan agen yang dilatih untuk menggantikan posisi pemain pada saat Latihan yang mempunyai otak *PlayerAgent*.

    2. Menerima input posisi player dan musuh

    Sistem dapat menerima posisi *player* dan musuh untuk dijadikan posisi *Agent*.

    3. Menerima input posisi unit musuh

    Sistem dapat menerima posisi *unit* musuh untuk memperhitungkan langkah terbaik

    4. Memeriksa semua posisi yang paling tepat

    Sistem dapat memeriksa seluruh kemungkinan posisi pergerakan *unit* untuk mendapatkan pergerakan yang paling optimal.

    5. Mengolah nilai untuk *reward* dan *punishment*

    Sistem dapat melakukan perhitungan nilai *reward* dan *punishment* untuk semua posisi yang dilakukan oleh *agent* sehingga dapat mengetahui posisi yang paling optimal.

    6. Memberikan posisi ideal

    Sistem dapat memberikan posisi paling ideal untuk memenangkan permainan.

2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan persyaratan yang diperlukan untuk menentukan elemen apa saja yang dibutuhkan sistem. Hal ini juga menentukan bagaimana sistem akan bekerja di masa depan.

Persyaratan sistem non-fungsional adalah sebagai berikut:

1. Tampilan

Tampilan system memiliki tampilan yang lebih bagus dan mudah dipahami.

2. Efisiensi

Sistem yang dibangun dapat digunakan oleh computer lebih efisien.

3. Hemat Biaya

Sistem ini tidak memerlukan dikarenakan gratis.

4. Kontrol

Sistem dapat dikontrol dengan Gerakan yang mudah dan reaktif.

5. Dokumentasi

Sistem dapat menyimpan hasil perhitungan untuk waktu tangkap tersisa dari permainan.

3.1.3 Analisis Proses

Dalam penelitian ini sistem yang dibangun akan mengimplementasikan *reward* dan *punishment* menggunakan *unity machine learning agent*. Adapun *reward* dan *punishment* dalam penelitian ini adalah:

Tabel 3.1 reward and punishment enemy agent

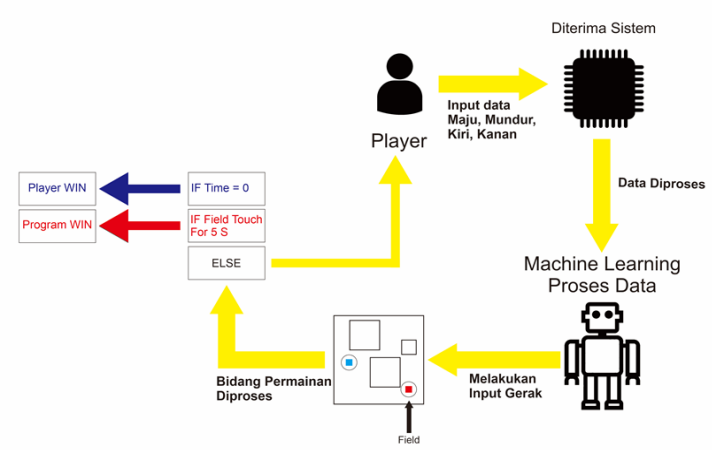
Reward Enemy Agent	Punishment Enemy Agent
Distance Reward: Agent menerima hadiah karena semakin dekat dengan *target* dan bergerak ke arah itu. Imbalannya sebanding dengan jarak dari target, mendorong *agent* untuk bergerak lebih dekat. float distanceReward = 0.2f * (proximityThreshold - distanceToTarget);	*Penalty for Staying Too Far: Agent* menerima penalti karena berada terlalu jauh dari *player*, mendorongnya untuk tetap dekat dengan *target*. AddReward (-0,01f);
Proximity Reward: Agent menerima hadiah atas kedekatannya dengan *player*. Hadiahnya berkurang seiring bertambahnya jarak ke *target.* AddReward(0.1f / (jarakToTarget + 1.0f));	*Penalty for Staying in Place: Agent* menerima penalti karena tetap di satu tempat. Ini mendorong *agent* untuk terus bergerak. AddReward (-0,01f);
Movement Reward: Agent menerima hadiah kecil karena tidak tinggal di satu tempat. Ini menghukum *agent* karena diam terlalu lama. AddReward(penaltyForStayingInPlace);	*Jump Height Penalty: Agent* menerima penalti karena melompat terlalu tinggi. Ini mencegah lompatan berlebihan. AddReward(jumpHeightPenalti);
Obstacle Avoidance Reward: Agent menerima hadiah karena tidak bertabrakan dengan rintangan. Ini mendorong *agent* untuk menghindari rintangan di jalurnya. AddReward (0,01f);	*Collision Penalty: Agent* menerima penalti karena bertabrakan dengan rintangan. Ini membuat *agent* enggan untuk menabrak rintangan. AddReward (-0.1f);
Boundary Reward: Agent menerima hadiah karena tetap berada dalam batas. AddReward (0,01f);	

Tabel 3.2 tabel reward dan punishment player agent

Reward Player Agent	Punishment Player Agent
Time Reward: Agent menerima hadiah seiring waktu. Hal ini mendorong *agent* untuk bertahan lebih lama. AddReward(timeRewardMultiplier);	*Captured Penalty:* Ada pengganti penalti jika *player* ditangkap. Namun implementasinya masih belum ada. AddReward (-10.0f);
Center Reward: Agent menerima hadiah karena lebih dekat dengan pusat lingkungan. Hal ini mendorong *agent* untuk tetap berada dalam area yang ditentukan. AddReward(centerReward);	

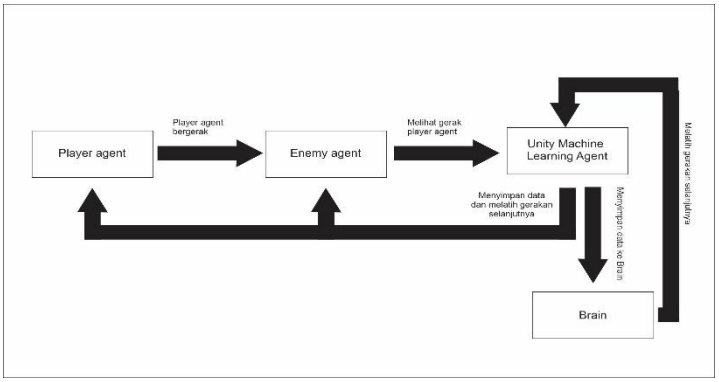
3. 2. Arsitektur Umum Sistem

Arsitektur sistem adalah gambaran umum proses kerja sistem secara menyeluruh. Berikut merupakan gambar arsitektur umum pada penelitian ini.



Gambar 3.2 Arsitektur Umum Sistem Enemy Agent Dengan User

Player akan memulai permainan kemudian waktu akan dimulai dan diberikan 5 menit dan rintangan akan dihasilkan secara acak. *Agent* kemudian mengetahui bahwa permainan telah dimulai dan mulai memproses data lokasi *player* untuk menangkap *player* dan *player* juga dapat .pada setiap detik pergerakan *player*, kedua *agent* akan mengolah data lokasi *player* dan menghitung rute terdekat tanpa hambatan serta membuat strategi untuk menangkap *player*.



Gambar 3.3 Arsitektur Umum Sistem Enemy Agent Dengan Player Agent

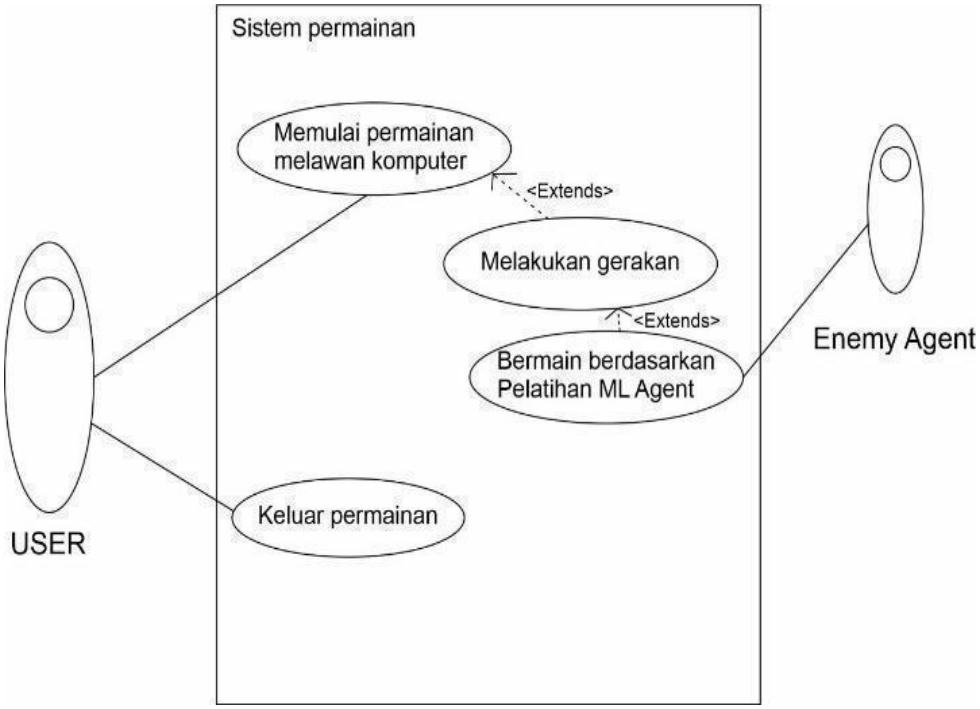
Player agent akan melakukan gerakan lalu *enemy agent* akan melihat ke arah mana *player agent* akan bergerak kemudian akan disimpan ke dalam *unity machine learning agent* yang membuat suatu *brain*(otak) lalu otak memberikan instruksi kepada *unity machine learning agent* untuk memberikan gerakan kepada *player agent* dan *enemy agent*.

3.3. Pemodelan Sistem

Pemodelan sistem adalah proses pengembangan model abstrak suatu sistem, dimana masing-masing model menyajikan pandangan atau perspektif yang berbeda dari sistem tersebut. Pemodelan sistem dapat merepresentasikan suatu sistem dengan menggunakan notasi grafis.

3.2.1. Diagram *Usecase*

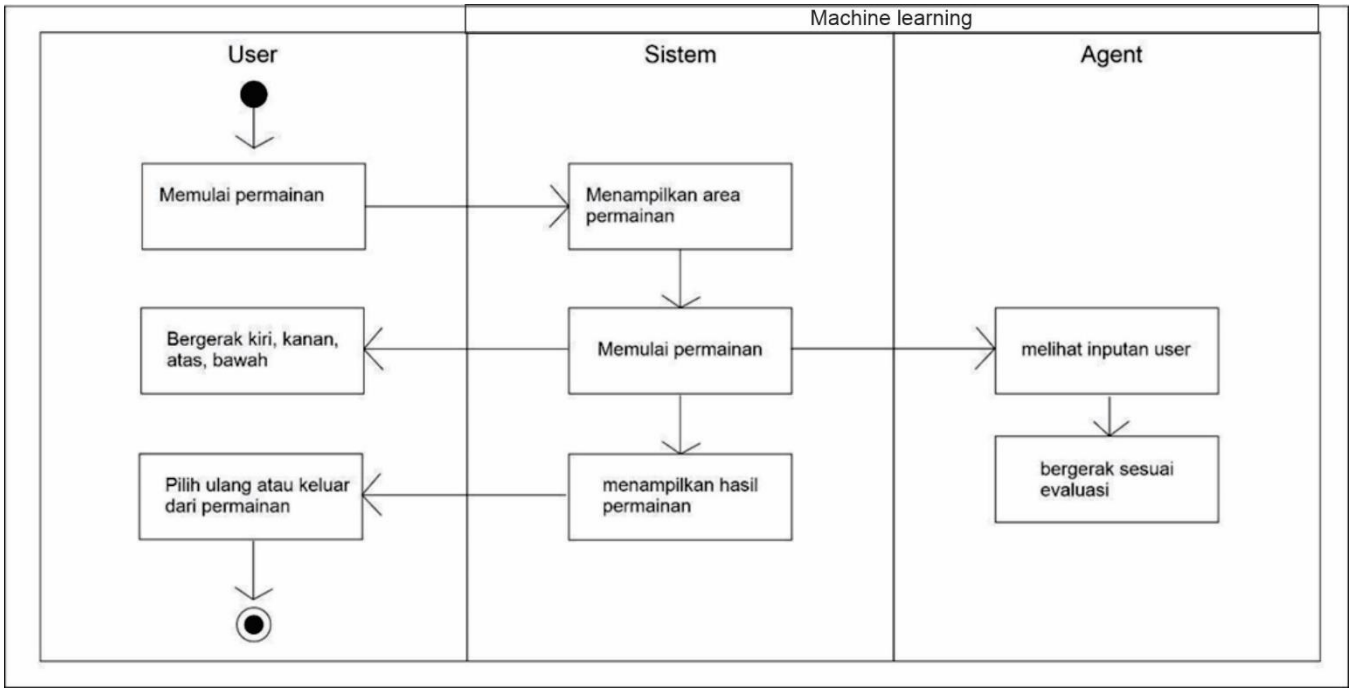
Use Case Diagram merupakan gambaran atau representasi fungsi-fungsi sistem untuk memetakan kebutuhan sistem dan interaksi antara sistem dan pengguna. Gambar 3.2 merupakan *use case diagram* dari *game tag* yang akan digunakan dalam penelitian ini.



Gambar 3.4 Diagram Usecase

3.2.2. Diagram *Activity*

Pada langkah ini, serangkaian alur kerja atau aktivitas akan ditampilkan dalam cara *agent* mengejar pengguna dan masukan pengguna. Diagram aktivitas dalam penelitian ini dapat dilihat pada Gambar 3.3 di bawah ini.



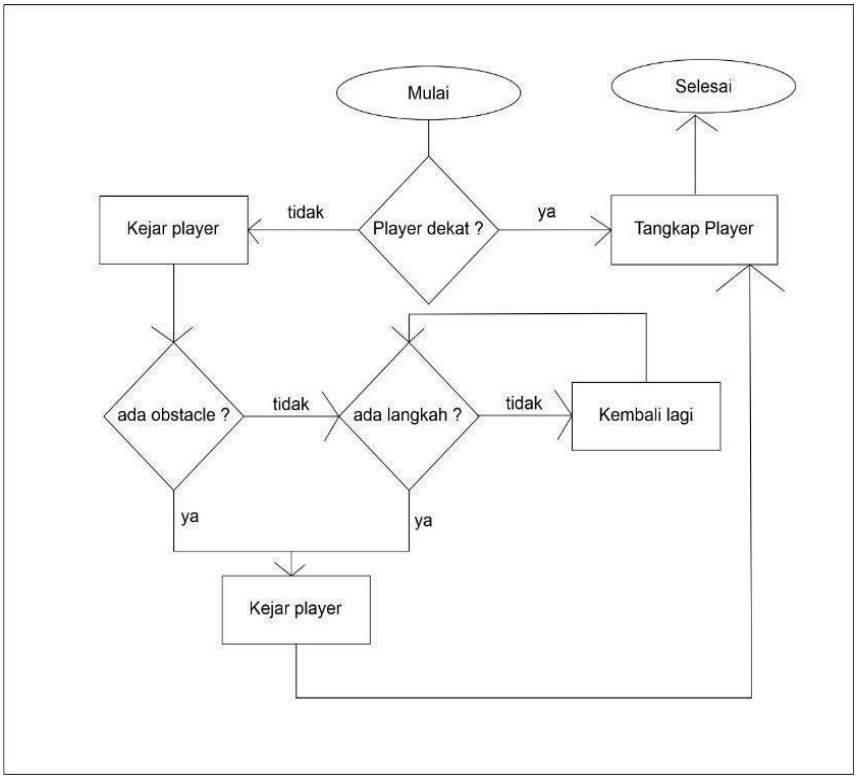
Gambar 3. 5 Diagram *Activity*

3.3 Flowchart

Flowchart merupakan salah satu jenis diagram yang bertujuan untuk menggambarkan langkah-langkah penyelesaian suatu permasalahan dengan menggunakan simbol-simbol tertentu yang berkaitan. Terdapat beberapa *flowchart* dalam penelitian ini, yaitu *flowchart* evaluasi langkah *Agent* dan *flowchart* sistem.

3.3.1 Flowchart evaluasi langkah agent

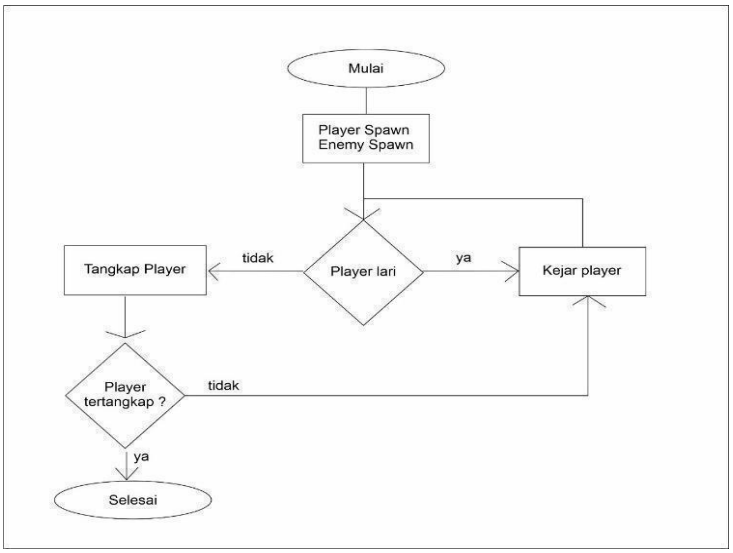
Sebelum mengambil langkah, *agent* akan mengevaluasi terlebih dahulu setiap kemungkinan posisi. Dalam mengevaluasi langkah-langkahnya, *agent* akan menggunakan *unity machine learning agents* dengan membandingkan jalur tercepat untuk menangkap *player*. Gambar 3.4 akan menampilkan diagram alur untuk mengevaluasi langkah-langkah *agent*.



Gambar 3. 6 *Flowchart Evaluasi Langkah Agent*

3.3.2 *Flowchart Sistem*

Flowchart sistem menggambarkan pemecahan masalah untuk seluruh sistem. Gambar 3.5 menunjukkan *Flowchart* sistem.



Gambar 3. 7 Flowchart Sistem

3.4 Perancangan Sistem Interface

Perancangan antarmuka merupakan proses Implementasi desain tampilan yang akan digunakan sebagai media komunikasi antara pengguna dan sistem. Pada sistem ini terdapat beberapa tampilan antarmuka yaitu tampilan menu utama, tampilan cara bermain, tampilan permainan, dan tampilan akhir permainan.

3.4.1 Rancangan Tampilan Menu Utama

Halaman menu utama merupakan tampilan awal sistem. Pada halaman utama terdapat *welcome*, tombol *play*, tombol cara bermain.

Gambar 3. 8 Halaman Menu Utama

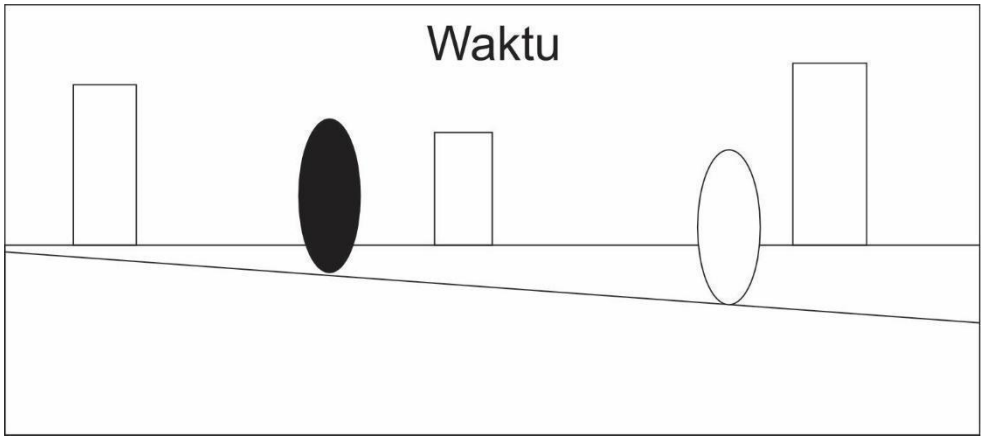
3.4.2 Rancangan Tampilan Cara Bermain

Tampilan rencana cara bermain akan berisi informasi yang menjelaskan aturan dalam permainan. Tampilan sederhana dan sedikit informasi namun informatif akan ada pada halaman cara bermain ini sehingga pengguna hanya membutuhkan waktu singkat untuk memahami aturan mainnya. Selain itu akan terdapat tombol kembali yang akan membawa pengguna kembali ke halaman menu utama.

Gambar 3. 9 Halaman Cara Bermain

3.4.3 Rancangan Tampilan Permainan

Halaman permainan akan muncul ketika pengguna menekan tombol *play* pada halaman *menu* utama. Akan ada area permainan 100 x 100. *Player* satu akan bermain dengan musuh berwarna merah (di gambar bewarna hitam) dan *Player* satu akan memainkan karakter berwarna biru (di gambar berwarna putih). Di sisi atas terdapat teks yang menunjukkan waktu pemutaran.



Gambar 3. 10 Halaman Awal Bermain

3.4.4 Rancangan Tampilan Permainan Berakhir

Jika pemain kalah maka sistem akan menampilkan tampilan bahwa permainan telah selesai. Sistem akan menampilkan waktu terakhir permainan dan terdapat tombol ulangi permainan.

Gambar 3. 11 Halaman Permainan Berakhir

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Hardware dan Software

Untuk membangun implementasi sistem *unity machine learning agent* pada *game tag* tentunya memerlukan *software* dan *hardware* yang mendukung dalam membangun sistem tersebut. Dalam penelitian ini penulis menggunakan perangkat lunak dan perangkat keras dengan spesifikasi sebagai berikut:

4.1.1 Spesifikasi Hardware

Berikut adalah spesifikasi *hardware* yang digunakan penulis untuk membangun sistem dalam penelitian ini :

1. 13th Gen Intel(R) Core(TM) i3-1305U 1.60 GHz
2. Intel UHD Graphics
3. RAM DDR5 8 GB
4. SSD 478 GB

4.1.2 Spesifikasi Software

Berikut adalah spesifikasi *software* yang digunakan penulis untuk membangun sistem dalam penelitian ini :

1. Windows 11 pro
2. Unity 2022.3.17f1
3. Visual Studio 2022

4.2 Penerapan Rancangan Interface

Tampilan interface sistem dibuat berdasarkan rancangan interface yang sudah dijelaskan pada bab sebelumnya. Berikut tampilan *interface* untuk Implementasi *Game Tag* Dengan *Machine Learning* Menggunakan *Unity*.

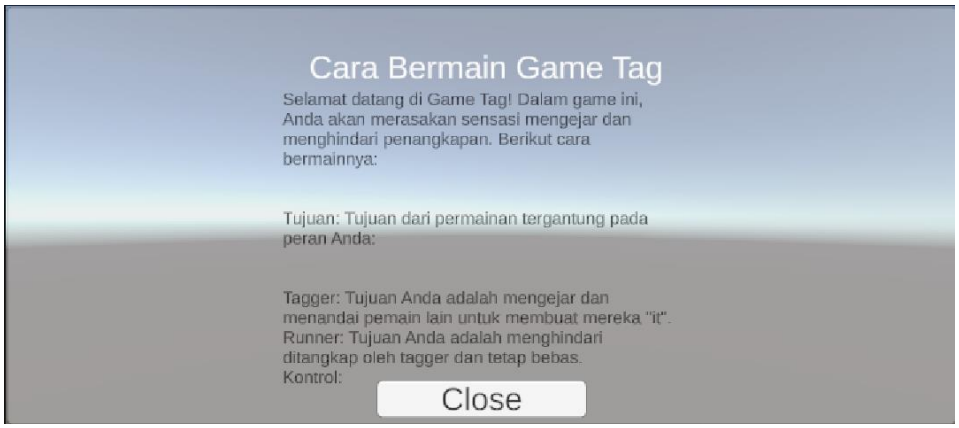
4.2.1 Desain Menu Home

Menu Home merupakan tampilan awal saat aplikasi dijalankan. Terdapat *text welcome* pada *game* dan juga dua tombol yaitu tombol dan tombol keluar aplikasi cara bermain.

Gambar 4.1 Tampilan Awal

4.2.2 Desain Menu Cara Bermain

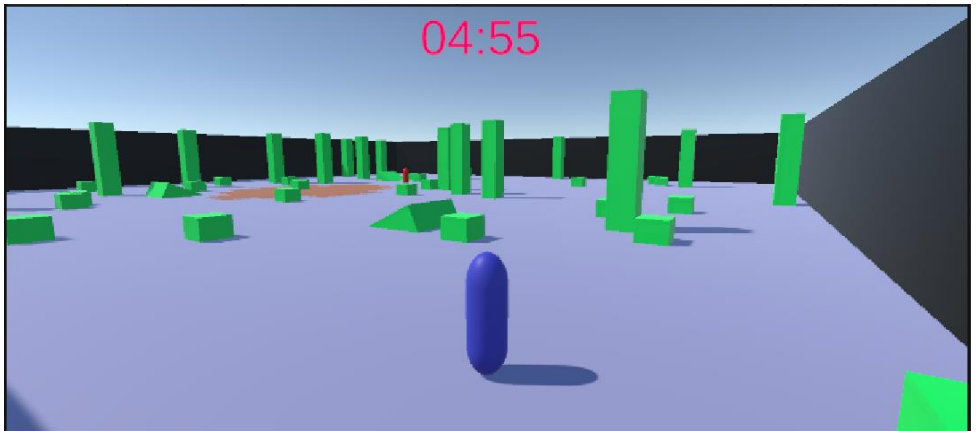
Pada halaman cara bermain, pemain dapat membaca peraturan yang berlaku pada permainan ini. *Player* juga akan diberitahu *tagger* dan *runner* yang ada di dalam *game*. Terdapat tombol tutup di bawah yang akan membawa pemain kembali ke *menu* utama.



Gambar 4.2 Tampilan Cara Bermain

4.2.3 Desain Tampilan Bermain

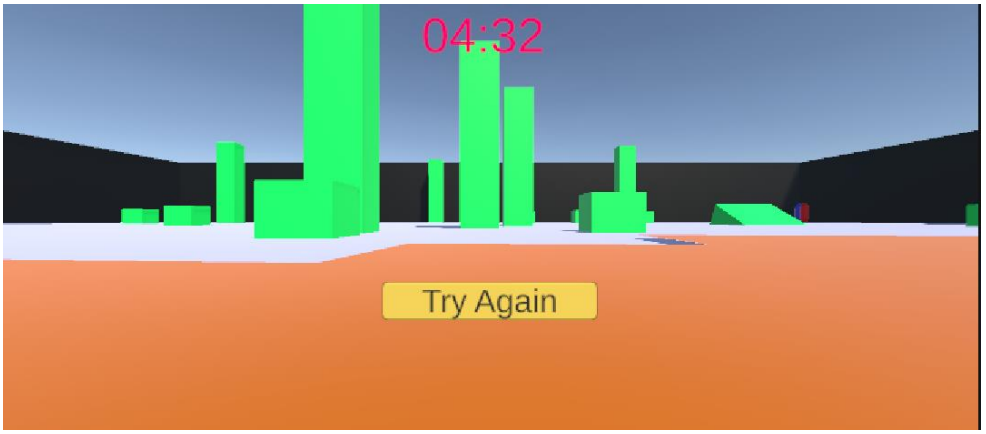
Halaman bermain akan menampilkan lapangan permainan dan musuh serta karakter yang akan dimainkan oleh *player*. *Player* akan memainkan karakter biru dan pemain kedua atau *agent* akan memainkan karakter merah. Karakter musuh bertugas mengejar dan menangkap *player* dalam waktu tertentu dan *player* bertugas melarikan diri dari musuh dalam jangka waktu tertentu.



Gambar 4.3 Tampilan Bermain

4.2.4 Desain Tampilan Permainan Berakhir

Di halaman akhir permainan terdapat hasil sisa waktu permainan. Sistem akan menampilkan tombol putar ulang.



Gambar 4.4 Gambar Tampilan Coba Lagi

4.3 Pengujian

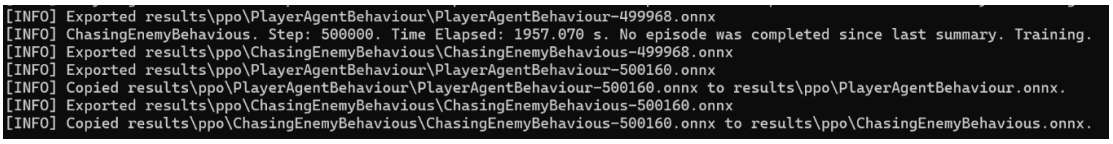
Pada tahap pengujian peneliti akan menjalankan sistem yang telah dibangun dan hasil dari tahap ini akan diketahui apakah sistem yang dibangun sudah berjalan sesuai dengan yang diharapkan.

4.3.1 Pengujian Unity Machine learning agent

Pengujian *agent* dilakukan untuk mengetahui apakah *agent* yang diterapkan dapat berjalan dengan baik. Dalam permainan yang dibangun, *agent* akan mengambil langkah dengan menghitung jarak musuh kemudian memeriksa apakah posisi tersebut paling cepat menuju *player* yang ada. Kesempatan bermain akan terhenti jika musuh dapat mengejar *player* dan sudah berpindah ke posisi optimal. Musuh akan melatih penyintas 5 generasi dengan setiap pelatihan 500 ribu langkah.

Dibawah ini adalah link *training agent* dari generasi 1 sampai generasi ke 5 :

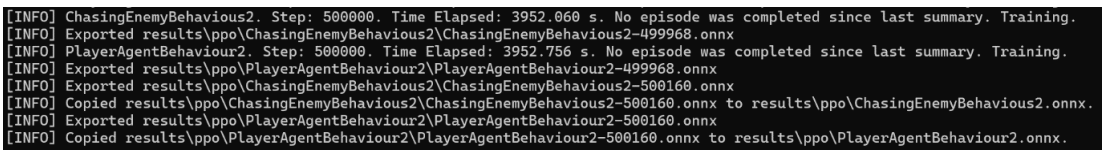
1. Pelatihan generasi 1



Gambar 4.5 Hasil Training Generasi 1

Gambar di atas menggambarkan langkah *training* yang dilakukan sebanyak 500.160 langkah untuk melatih *enemy agent* dalam menangkap *player agent* dengan waktu 1,957.070 detik

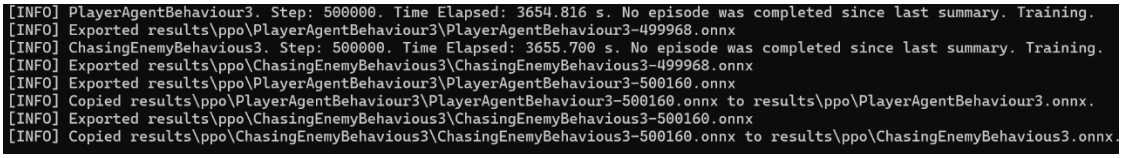
2. Pelatihan generasi 2



Gambar 4.6 Hasil Training Generasi 2

Gambar di atas menggambarkan langkah *training* yang dilakukan sebanyak 500.160 langkah untuk melatih *enemy agent* dalam menangkap *player agent* dengan waktu 3,952.756 detik

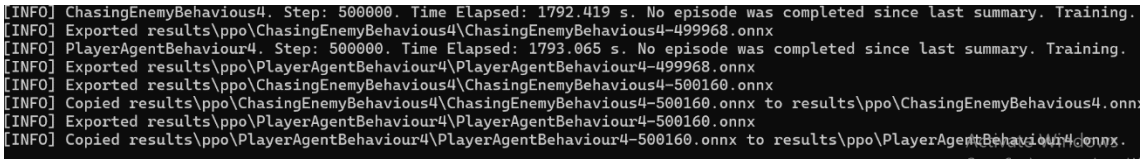
3. Pelatihan generasi 3



Gambar 4.7 Hasil Training Generasi 3

Gambar di atas menggambarkan langkah *training* yang dilakukan sebanyak 500.160 langkah untuk melatih *enemy agent* dalam menangkap *player agent* dengan waktu 3,655.700 detik

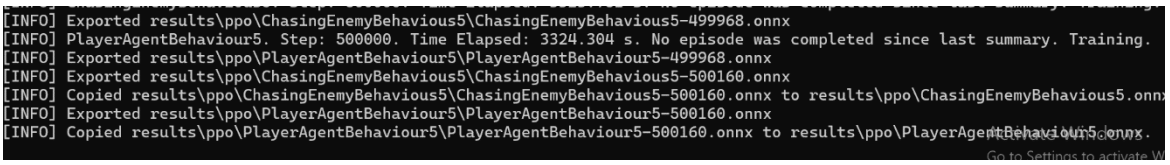
4. Pelatihan generasi 4



Gambar 4.8 Hasil Training Generasi 4

Gambar di atas menggambarkan langkah *training* yang dilakukan sebanyak 500.160 langkah untuk melatih *enemy agent* dalam menangkap *player agent* dengan waktu 1,792.419 detik

5. Pelatihan generasi 5



Gambar 4.9 Hasil Training Generasi 5

Gambar di atas menggambarkan langkah *training* yang dilakukan sebanyak 500.160 langkah untuk melatih *enemy agent* dalam menangkap *player agent* dengan waktu 3,324.304 detik

4.3.2 Pengujian Kehandalan Machine learning agent

Pengujian *agent* dilakukan untuk mengetahui seberapa baik *agent* merespon berbagai pergerakan pengguna untuk menangkap *player*. Pengujian dilakukan oleh 5 generasi dan 5 peserta yang diminta bermain melawan *agent* sebanyak 2 kali yang telah diterapkan oleh *unity machine learning agent* yang totalnya 10 game untuk masing masing generasi. Tabel 4.1 dibawah ini akan menampilkan hasil pengujian *agent* pada *game tag*.

Tabel 4.1 pengujian gen 1 sampai gen 5 terhadap player berdasarkan sisa waktu

Game	Waktu Tersisa (detik)				
	Gen 1	Gen 2	Gen 3	Gen 4	Gen 5
Game 1	96	52	111	144	277
Game 2	122	93	0	170	231
Game 3	160	227	47	203	238
Game 4	196	156	105	92	278
Game 5	93	145	99	281	242
Game 6	225	250	118	57	267
Game 7	24	128	111	146	238
Game 8	197	37	51	151	269
Game 9	83	98	43	272	193
Game 10	17	39	22	121	287
Rata-Rata	121	122	70	163	252

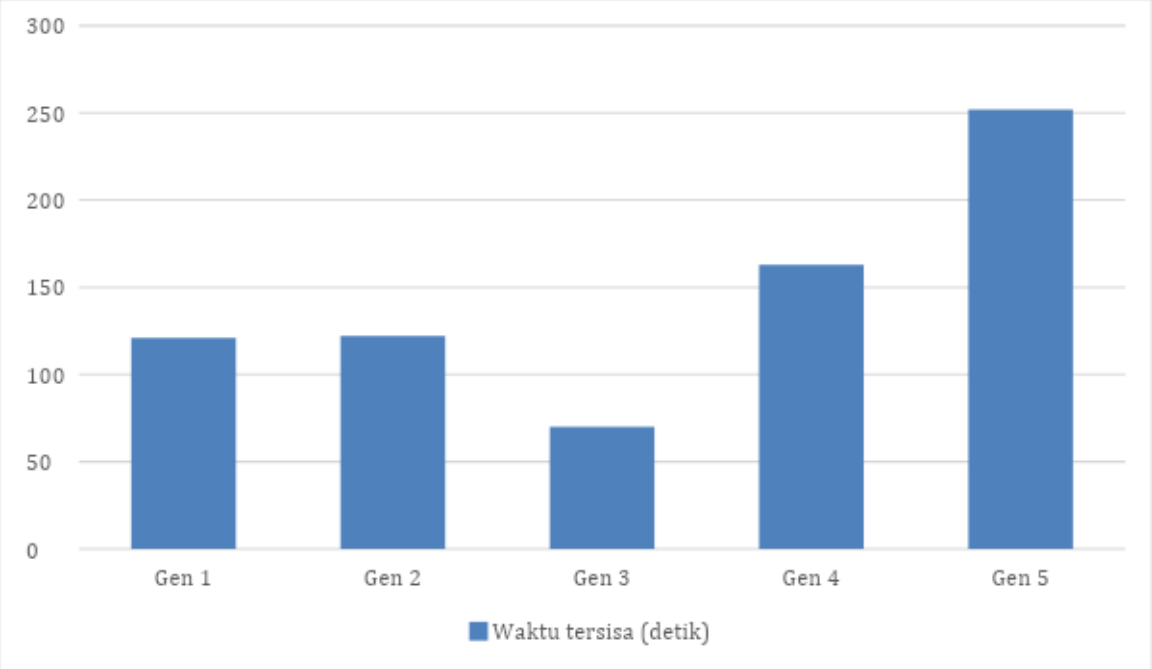
4.3.3 Pengujian Kehandalan Machine learning agent yang tidak berhasil

Pengujian *agent* dalam menangkap *player* dalam waktu 5 menit sehingga dinyatakan tidak berhasil tabel berikut merupakan alasan dan jumlah penangkapan tidak berhasil oleh *agent* dalam menangkap *player*.

Tabel 4.2 pengujian gen 1 sampai gen 5 terhadap player yang tidak berhasil

Generasi	Jumlah Penangkapan tidak berhasil	Alasan penangkapan tidak berhasil
Gen 1	3	*Agent* masih kaku dalam pengejaran sehingga pada saat agent ingin berbelok *agent* diam sebentar sebelum mengejar lagi.
Gen 2	4	*Agent* masih kaku dalam mengejar *player* dikarenakan masih meraba dan memprediksi gerak *user* sehingga adanya salah berbelok dalam pengejaran.
Gen 3	6	*Agent* terjebak dengan beberapa *obstacle*(*rintangan)* yang dibuat secara acak sehingga membutuhkan waktu lama sebelum *agent* dapat mengejar *player* Kembali .
Gen 4	2	*Agent* masih terjebak dalam obstacle tetapi membutuhkan waktu lebih sedikit dalam berpindah daripada generasi ke 3.
Gen 5	0	*Agent* sudah cukup dalam pelatihan sehingga bisa memprediksi arah lawan akan berada dimana.

Berikut ini adalah gambaran rata-rata hasil permainan *agent* waktu kejar tersisa pemain yang telah diujikan dalam beberapa generasi dan berjumlah 10 *game* pada setiap generasi kepada peserta dan disusun dalam bentuk bentuk *Bar Chart*.



Gambar 4.10 Bar Chart

Berdasarkan chart pengujian diatas didapatkan bahwa generasi ke 5 lebih efisien dengan waktu tangkap tersisa 252 detik.

BAB 5

KESIMPULAN

5.1. Kesimpulan

Berdasarkan hasil pengujian dari implementasi *unity machine learning agent* pada *game tag* didapat beberapa kesimpulan, yaitu :

1. *Unity machine learning agent* dapat diimplementasikan ke dalam permainan tag sebagai agen yang menjadi lawan pengguna dan dapat menangkap pemain dalam permainan secara efisien. Hal ini dapat dibuktikan dari hasil pengujian dari generasi ke-1 sampai generasi ke-5 yang telah dilakukan dengan rata-rata waktu tersisa 252 detik pada generasi ke-5 dalam mengejar *player* dengan estimasi waktu *game* selama 300 detik.
2. *Game tag* pada penelitian ini dibuat dengan unity machine learning agent versi release 20 dibuat dengan menggunakan Unity Engine 2022.3.17f1

5.2. Saran

Beberapa saran peneliti untuk peneliti selanjutnya, yaitu :

1. Menerapkan optimasi *unity machine learning* dengan mengubah *file training* pada *agent* agar lebih optimal dalam memprediksi pergerakan lawan.
2. Menambahkan mode permainan sehingga pengguna dapat memainkan lebih banyak mode permainan seperti melawan 2 *agent* atau lebih.
3. Menambah suara dan mempercantik tampilan *game* agar pengguna semakin tertarik untuk memainkan *game* tersebut

DAFTAR PUSTAKA

Amdanni, T. R. (2022). Implementasi Machine Learning pada Game Balap 2D dengan Rintangan Berbasis Unity. Universitas Sumatera Utara.

Sharma, Ansh . (2019). Machine learning in game development. https://medium.com/data-science-community-srm/machine-learning-in-game-development-cd6669affe22

Asmiatun, S., & Astrid Novita Putri. (2017). Belajar Membuat Game 2D dan 3D Menggunakan Unity.

Azmi, M. U., Sahertian, J., & Setiawan, A. B. (2020). Implementasi Game RPG Adventure Of The Dungeon. Universitas Nusantara PGRI Kediri.

Alfasyam, B. A. (2023). PENGEMBANGAN APLIKASI PENGETAHUAN BAHASA PEMOGRAMAN DASAR DAN LANJUT BERBASIS ANDROID MENGGUNAKAN METODE GAME DEVELOPMENT LIFE CYCLE. Universitas Malikussaleh.

Hanggoro, A. C., Kridalukmana, R., & Teguh Martono, K. (n.d.). Implementasi Aplikasi Permainan “Jakarta Bersih” Berbasis Unity.

Dharma, A. S., & Tambunan, V. (2021). Penerapan Model Pembelajaran dengan Metode Reinforcement Learning Menggunakan Simulator Carla. JURNAL MEDIA INFORMATIKA BUDIDARMA, 5(4), 1405. https://doi.org/10.30865/mib.v5i4.3169

Fontes, A., & Gay, G. (2021). Using machine learning to generate test oracles: A systematic literature review. Proceedings of the 1st International Workshop on Test Oracles, 1–10.

Haq, M. Y. A., Akbar, M. A., & Afirianto, T. (2022). Pengembangan Non-Player Character (NPC) Menggunakan Unity ML-Agents Pada Karting Microgame. Fountain of Informatics Journal, 7(1), 15–21.

Heryadi, Y., Kristen, U., & Wacana, S. (2020). Machine Learning: Konsep dan Implementasi Teguh Wahyono. https://www.researchgate.net/publication/344419764

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., & others. (2018). Unity: A general platform for intelligent agents. ArXiv Preprint ArXiv:1809.02627.

Dinata, R. K. , & Hasdyna, N. (2020). Machine Learning.

Lawson, V. I. (2019). Implementasi Metode Heuristic dalam Game Eller's Maze. Universitas Sumatera Utara.

Nork, B., Lengert, G. D., Litschel, R. U., Ahmad, N., Lam, G. T., & Logofătu, D. (2018). Machine learning with the pong game: a case study. Engineering Applications of Neural Networks: 19th International Conference, EANN 2018, Bristol, UK, September 3-5, 2018, Proceedings 19, 106–117.

Nugroho, A., & Pramono, B. A. (2017). APLIKASI MOBILE AUGMENTED REALITY BERBASIS VUFORIA DAN UNITY PADA PENGENALAN OBJEK 3D DENGAN STUDI KASUS GEDUNG M UNIVERSITAS SEMARANG (Vol. 14, Issue 2). www.unity3d.com.

Rajab, A. M., Haji, N., Alfarouk, H., Hassan, N. H., & Rajab, A.-R. M. (n.d.). Tic-Tac-Toe Learning Using Artificial Neural Networks International Journal of Engineering and Information Systems (IJEAIS) Designing and Implementation of Tic-Tac-Toe-4X4 based Artificial Intelligence using Python Programming. www.ijeais.org

Septian, B. D. (2020). Implementasi Game The Legend Of Timun Mas Dengan Menggunakan Unity. Skripsi. Teknologi Informasi Dan Komunikasi, Teknik Informasi, Universitas Semarang, Semarang.

Taye, M. M. (2023). Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. Computers, 12(5), 91.

Thabtah, F., Padmavathy, A. J., & Pritchard, A. (2020). Chess results analysis using elo measure with machine learning. Journal of Information & Knowledge Management, 19(02), 2050006.

Merriam-Webster. (n.d.). Machine learning. In Merriam-Webster.com dictionary. Retrieved March 13, 2024, from https://www.merriam-webster.com/dictionary/machine%20learning

machine learning noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com. (n.d.-c). https://www.oxfordlearnersdictionaries.com/definition/english/machine-learning?q=machine+learning yang diakses pada 22 november 2022

video game noun - Definition, pictures, pronunciation and usage notes | Oxford

Advanced Learner’s Dictionary at OxfordLearnersDictionaries.com. (n.d.). https://www.oxfordlearnersdictionaries.com/definition/english/video-game yang diakses pada 22 novermber 2022

video game. (2024). In Merriam-Webster Dictionary. https://www.merriam-webster.com/dictionary/video%20game yang diakses pada 22 november 2022

Britannica, The Editors of Encyclopaedia. "tag". Encyclopedia Britannica, 18 Jul. 2018, https://www.britannica.com/topic/tag-game. diakses pada 8 Maret 2024.

Curiculum vitae

MICHAEL ALESSANDRO

JALAN JENDRAL AHMAD YANI KOMPLEK GWBC No. 2B
20711 Binjai
Indonesia
Mobile No: +62 813 7519 1036

Education Records

2019	Sekolah Menengah Atas SMA Methodist Binjai Indonesia
2019 - present	Bachelor Of Computer Science (Programming) Universitas Sumatera Utara Indonesia

Employement History

2020-2021
(Full time) Graphic Desainer At Percetakan Ria (Printing)
2021-present Management / Graphic Designer at Café House Binjai

Relevant skill

Computer Skill
Microsoft Word, Excel, Powepoint, CorelDraw, Photoshop, Illustrator
Programming Skill
Python, Java, C++, C#, Pascal, Javascript, Html, CSS, PHP
Language Skill
Bahasa Indonesia, English, Hokkien(spoken)

Link Program Game Tag:

https://drive.google.com/drive/folders/1bk_hXrRxfh8ChLsxjNC_W_Vw0L0g4QQi?usp=drive_link