# ABSTRAK

Pada *shooting game* diperlukan adanya *bot*, sehingga pemain memiliki lawan bermain. Penerapan AI pada *bot* terdapat masalah pembelajaran untuk cara bergerak dan menembak. Salah satu cara untuk menyelesaikan masalah tersebut digunakan metode *reinforcement learning*. Untuk mendapatkan hasil pergerakan dan menembak yang tepat, diperlukan melakukan pembelajaran data. Dalam melakukan pembelajaran, diberikan beberapa aturan-aturan dalam menentukan keberhasilan. Pengujian dilakukan dengan menggunakan data yang sudah dipelajari dan diolah oleh *bot.* Hasil penelitian menunjukkan bahwa algoritma *reinforcement learning* mampu memberikan keberhasilan *bot* dalam melakukan tembakan yang relatif cukup baik.

Kata Kunci : *reinforcement learning,* AI, *bot*, *shooting game*

# KATA PENGANTAR

Ucapan syukur penulis panjatkan Tuhan Yang Maha Esa karena berkat Rahmat dan Karunia-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul “Pengembangan *Bot* pada *Game Gang Garrison* 2 dengan *Reinforcement Learning*”.

Penulisan Tugas Akhir ini diajukan untuk memenuhi salah satu syarat kelulusan dalam jenjang perkuliahan Strata I Sekolah Tinggi Manajemen Informatika Dan Komputer.

Dalam kesempatan ini penulis ingin mengucapkan terima kasih kepada :

1. Bapak Dr. Pahala Sirait, S.T., M.Kom., selaku Pembimbing I yang telah membimbing penulis mengerjakan Tugas Akhir ini.
2. Bapak Andrew Sagita Jauhari, S.Kom., M.T., selaku Pembimbing II yang telah membimbing penulis mengerjakan Tugas Akhir ini.
3. Bapak Dr. Mimpin Ginting, M.S., selaku Ketua STMIK Mikroskil Medan
4. Bapak Djoni, S.Kom., M.T.I., selaku Wakil Ketua I STMIK Mikroskil Medan
5. Bapak **Sunario Megawan, S.Kom., M.Kom., selaku Ketua Program Studi Teknik Informatika STMIK Mikroskil Medan.**
6. **Bapak dan Ibu Dosen yang telah mendidik dan membimbing penulis dalam mengerjakan Tugas Akhir ini.**
7. **Orang tua dan teman-teman kami yang selama ini telah men-*support* penulis tetap semangat untuk menyelesaikan Tugas Akhir ini.**

**Penulis menyadari bahwa penelitian Tugas Akhir ini masih jauh dari sempurna sehingga penulis membutuhkan kritik dan saran yang bersifat membangun untuk kemajuan pendidikan di masa yang akan datang.**

**Akhir kata, sengaja penelitian Tugas Akhir ini memberikan manfaat, khususnya bagi penulis dan umumnya bagi semua orang dalam rangka menambah wawasan pengetahuan.**

Medan, Ferbruari 2017

Penulis

# DAFTAR ISI

[ABSTRAK i](#_Toc475721789)

[KATA PENGANTAR ii](#_Toc475721790)

[DAFTAR ISI iii](#_Toc475721791)

[DAFTAR GAMBAR iv](#_Toc475721792)

[DAFTAR TABEL vi](#_Toc475721793)

[BAB I PENDAHULUAN 1](#_Toc475721794)

[1.1 Latar Belakang 1](#_Toc475721796)

[1.2 Rumusan Masalah 2](#_Toc475721797)

[1.3 Tujuan dan Manfaat 2](#_Toc475721798)

[1.4 Batasan Masalah 3](#_Toc475721799)

[1.5 Metodologi Pengembangan Sistem 3](#_Toc475721800)

[BAB II TINJAUAN PUSTAKA 5](#_Toc475721801)

[2.1 Pengolahan Citra 5](#_Toc475721803)

[2.2 Grafika Komputer 5](#_Toc475721804)

[2.2.1 Grafik Transformasi Dua Dimensi 6](#_Toc475721805)

[2.3 *Game* 7](#_Toc475721806)

[2.4 Artificial Intelligence 16](#_Toc475721807)

[2.5 Reinforcement Learning 19](#_Toc475721808)

[2.6 Algoritma yang diajukan 20](#_Toc475721809)

[BAB III ANALISIS DAN PERANCANGAN 22](#_Toc475721810)

[3.1 Analisis 22](#_Toc475721812)

[3.1.1 Analisis Proses 22](#_Toc475721813)

[3.1.1.1 Merekam Cara Gerak *Bot* dan Perhitungan *Reward* untuk Pergerakan 22](#_Toc475721814)

[3.1.1.2 Mengolah RL *Movement* 26](#_Toc475721815)

[3.1.1.3 Membaca Data Gerakan pada RL *Movement* dan Belajar Cara Gerak *Bot* 29](#_Toc475721816)

[3.1.1.2 Mengolah RL *Shooting* 33](#_Toc475721817)

[3.1.2 Analisis Kebutuhan 37](#_Toc475721818)

[3.1.2.1 Kebutuhan Fungsional 38](#_Toc475721819)

[3.1.2.2 Kebutuhan Non-Fungsional 38](#_Toc475721820)

[3.2 Pemodelan Sistem dan Perancangan 39](#_Toc475721821)

[BAB IV HASIL DAN PENGUJIAN 45](#_Toc475721822)

[4.1 Tampilan 45](#_Toc475721824)

[4.1.1 Tampilan Menu Utama 45](#_Toc475721825)

[4.1.2 Tampilan Host Game 46](#_Toc475721826)

[4.1.3 Tampilan Pemilihan Tim 47](#_Toc475721827)

[4.1.4 Mode Permainanan *Generator* 48](#_Toc475721828)

[4.2 Hasil 49](#_Toc475721829)

[4.2.1 Hasil Pembelajaran Pergerakan 49](#_Toc475721830)

[4.2.2 Hasil Pembelajaran Menembak 54](#_Toc475721831)

[4.3 Pengujian 58](#_Toc475721832)

[BAB V KESIMPULAN DAN SARAN 69](#_Toc475721833)

[5.1 Kesimpulan 69](#_Toc475721835)

[5.2 Saran 69](#_Toc475721836)

[DAFTAR PUSTAKA 70](#_Toc475721837)

# DAFTAR GAMBAR

[Gambar 2.1 Permainan *Pac-man* merupakan *arcade game.* 8](#_Toc475088618)

[Gambar 2.2 DOTA2 merupakan PC *game*. 9](#_Toc475088619)

[Gambar 2.3 Perangkat Konsol (Nintendo Wii, Xbox 360, dan Playstation 3). 9](#_Toc475088620)

[Gambar 2.4 SONY PSP (PlayStation Portabel) merupakan *platform* untuk *handheld games*. 10](#_Toc475088621)

[Gambar 2.5 Inilah tampilan *game shooting* 3D. 11](#_Toc475088622)

[Gambar 2.6 *Gang Garrison 2* merupakan *game shooting* 2D dengan sudut pandang orang ketiga. 11](#_Toc475088623)

[Gambar 2.7 *Tekken* merupakan *game fighting* yang dikenal banyak orang. 12](#_Toc475088624)

[Gambar 2.8 *Game* aksi-*adventure* salah satunya adalah *Assassin's Creed*. 13](#_Toc475088625)

[Gambar 2.9 Salah satu permainan simulasi yaitu *The Sims.* 13](#_Toc475088626)

[Gambar 2.10 Tampilan status karakter. 14](#_Toc475088627)

[Gambar 2.11 Tampilan pertarungan dalam salah satu *game* RPG. 14](#_Toc475088628)

[Gambar 2.12 Clash of Clans merupakan permainan strategi. 15](#_Toc475088629)

[Gambar 2.13 Salah satu *puzzle* *game* adalah *Cut The Rope*. 16](#_Toc475088630)

20

20

23

27

29

33

40

41

43

45

46

47

48

48

49

51

52

54

56

57

58

59

60

61

62

64

65

66

67

68

68

# DAFTAR TABEL

[Tabel 3.1 Tabel Gerak RL. 30](#_Toc475096685)

[Tabel 3.2 Tabel Gerak RL yang sudah dinormalisai. 34](#_Toc475096686)

[Tabel 3.3 *Shooting* RL. 37](#_Toc475096687)

[Tabel 3. 4 Tabel *Shooting* RL dengan 400 kali pembelajaran yang sudah dinormalisasi. 41](#_Toc475096688)

[Tabel 3. 5 Tabel *Shooting* RL dengan 800 kali pembelajaran yang sudah dinormalisasi. 43](#_Toc475096689)

57

58

53

55

56

57

59

60

61

62

63

64

65

66

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

*Video game* merupakan salah satu media hiburan yang paling popular untuk semua usia. Sejak pertama kali ditemukan sampai sekarang, *game* telah mengalami kemajuan. Hal ini ditandai dengan perkembangan kualitas grafiknya, jenis permainan atau cara bermain, algoritma yang diterapkan, dan cara interaksi *game* (Aji, Irfan Satya., 2014). *Game* dapat diklasifikasikan berdasarkan *platform* dan *genre*. *Game* yang diklasifikasikan berdasarkan *platform* adalah *Arcade games,* PC *games,* *Console games, Handheld games, Mobile games*. Dimana *Platform* merupakan tempat atau media dimana *game* dijalankan. Dan untuk klasifikasi berdasarkan *genre* adalah **Aksi-*Shooting, Fighting* (pertarungan), Aksi-*Adventure,* Simulasi, *Role Playing,*** Strategi, dan *Puzzle.* (Iman, Nurul., 2013).

Salah satu game yang cukup terkenal dan diminati banyak orang adalah ber-*genre Thrid Person Shooting* (TPS) *Game* 2D, yang berarti pemain melawan pemain laindengan sudut pandang orang ketiga dan pemain bisa melihat seluruh jalan permainan. Salah satu permainan yang ber-*genre* TPS adalah *Gang Garrison* 2. Permainan ini bersifat *open source*, dimana para penggemar ataupun pengembang dapat mengubah *source code* yang ada pada permainan untuk menambahkan fitur. Ada fitur *bot* yang ditambah oleh pengembang yang ber­-*username* Orpheon pada forum *game*. Ada 6 (enam) model permainan pada *game* ini yaitu *capture the flag, control point, arena, generator, king of the hill,* dan *dual king of the hill*. Akan dipilih model permainan yaitu *generator*. Pada model *generator*, pemain harus menghancurkan mesin generator lawan sebelum mesin generator pemain dihancurkan oleh lawan. Pada *bot* sebelumnya sudah menggunakan kemampuan kecerdasan buatan yang sangat terbatas, sehingga kurang cerdas dan tidak menarik. Untuk itu akan ditambahkan kemampuan, sehingga permainan menjadi lebih dinamis dengan pengambilan keputusan berbasis pengetahuan (*learning)*. *Bot* ini nantinya dapat menambah ketertarikan dan tantangan bagi pemain, dimana *bot* dapat melakukan *learning* pada permainan.

*Bot* ini akan menggunakan metode *reinforcement learning* (RL) yang merupakan turunan dari AI (*Artificial Intelligence*), dimana metode ini dapat belajar dari pengetahuan yang didapat dari interaksi antara sistem dengan lingkungan (Wiering et al, 2007). Metode ini dapat merekam langkah yang pemain gunakan, kemudian disimpan dalam RL *memory* sebagai *state* dan *action*-nya. *State* dan *action* tersebut yang akan menentukan *reward* yang akan diterima oleh *bot* tersebut. *Bot* akan menyimpan dan memakai *state* dan *action* dengan *reward* tertinggi. *Bot* ini akan diuji terhadap *bot* yang sudah ada menggunakan metode SARSA (**S**tate-**A**ction-**R**eward-**S**tate-**A**ction), dimana *bot* akan melakukan *state* dan *action* secara berulang hingga mendapat hasil (*reward*) akhir tertinggi.

Berdasarkan uraian diatas, maka diajukan "Pengembangan *Bot* pada *Game Gang Garrison 2* dengan *Reinforcement Learning*" sebagai judul tugas akhir dengan harapan dapat mengembangkan permainan ini menjadi lebih baik.

## 1.2 Rumusan Masalah

Pada permainan sebelumnya, sistem permainan bersifat statis karena *bot* tersebut memiliki jalur dan gerakan yang sama. Untuk ini perlu menambahkan dinamisasi permainan yaitu berupa penambahan arah gerakan dan penambahan keputusan oleh sistem, sehingga diperlukan *reinforcement learning* untuk mengatasinya.

## 1.3 Tujuan dan Manfaat

Adapun tujuan dari tugas akhir ini adalah menambahkan kemampuan terhadap *bot* dengan menambahkan *learning* dalam permainan *Gang Garrison 2* sehingga lebih dinamis dan cerdas. *Bot* dapat memilih *action* terbaik dari *state* yang tersimpan pada RL *memory*.

Adapun manfaat dari tugas akhir ini adalah:

* Membuat permainan *gang garrison 2* ini menjadi lebih menarik bagi pemain karena pemain melawan *bot* yang lebih cerdas.
* Menjadi bahan pembelajaran lebih lanjut mengenai *reinforcement learning* serta menjadi bahan referensi untuk pengembangan selanjutnya.

## 1.4 Batasan Masalah

Adapun batasan masalah pada tugas akhir ini adalah:

* Menggunakan 1 (satu) model permainan dari 6 (enam) model permainan yang tersedia.
* Menggunakan 3 (tiga) jenis karakter dari 9 (sembilan) jenis karakter yang tersedia.
* Penggunaan *bot* maksimal 3 (tiga) *bot* dan 3 (tiga) pemain.
* Menggunakan algoritma *reinforcement learning*.
* Durasi permainan berlangsung selama 5 (lima) menit per 1 kali main.

## 1.5 Metodologi Pengembangan Sistem

Dalam pelaksanaan tugas akhir ini, langkah-langkah yang dilakukan adalah sebagai berikut:

1. Mengumpulkan referensi yang berhubungan dengan algoritma *reinforcement learning* dan referensi-referensi lain yang berhubungan dengan tugas akhir.
2. Pengembangan sistem yang menggunakan metode *waterfall*. Adapun tahapan-tahapan dalam metode *waterfall* adalah :
   1. Analisis

Pada tahap ini mencakup analisis proses dan pemodelan sistem identifikasi persyaratan fungsional dari sistem yang akan dikembangkan. Analisis fungsional sistem akan digambarkan dalam bentuk *activity diagram*.

* 1. Desain

Pada tahap ini akan menterjemahkan syarat kebutuhan pengembangan perangkat lunak. Tahapan proses ini berfokus pada struktur data.

* 1. *Coding* (pengkodean)

Pada tahap ini akan dilakukan implementasi dari analisis dan desain dengan menggunakan *coding* GML (*Game Maker Language*).

* 1. Pengujian

Proses pengujian dilakukan dalam 4 tahap yang berkelanjutan tanpa menghapus data yang disimpan dalam RL *memory*. Data yang akan tetap disimpan dalam RL *memory* adalah data pengujian yang memiliki *reward* paling besar. Dalam 4 tahap tersebut, setiap tahap memiliki jumlah pembelajaran yang berbeda dimulai dari 200, 300, 400 dan 800 kali pembelajaran yang dilakukan oleh pemain terhadap *bot* A. Pada saat sudah mencapai akhir jumlah pembelajaran dalam setiap tahap pengujian, maka *bot* A akan diuji dengan melawan *bot* B. Hal tersebut dilakukan untuk mengetahui reward tertinggi yang dapat dicapai oleh *bot*. Pengujian tersebut juga untuk mengetahui perbandingan antara *bot* A dan *bot* B serta mengetahui perbedaan reward dari 4 tahap pengujian.

1. Menarik kesimpulan berdasarkan hasil pengujian terhadap program dan menyusun laporan tugas akhir

# BAB II

# TINJAUAN PUSTAKA

## 2.1 Pengolahan Citra

Pengolahan citra merupakan salah satu cara untuk memperbaiki kualitas citra sehingga mudah diinterpretasi oleh manusia maupun mesin. Beberapa field research terkait dengan pengolahan data citra, antara lain grafika komputer (*computer grafik*), pengolahan citra (*image processing*), dan pengenalan pola (*pattern recognition* atau *image interpretation*). Pengolahan citra dilakukan dengan mentransformasikan citra menjadi citra lain yang mempunyai kualitas lebih baik yang operasinya meliputi perbaikan kualitas citra (image enhancement), pemugaran citra (*image restoration*), pemampatan citra (*image compression*), segmentasi citra (*image segmentation*), analisis citra (*image analysis*), dan rekontruksi citra (Hermawati, 2013) (Gonzalez R.C. & Woods R.E, 2002) (Aniati, 1992).

## 2.2 Grafika Komputer

Grafika komputer adalah teknik-teknik dalam ilmu komputer dan matematika untuk merepresentasikan dan memanipulasi data gambar menggunakan komputer. Dengan bahasa lain, istilah grafika komputer juga dapat diartikan segala sesuatu selain teks atau suara. Seiring dengan perkembangan teknologi dewasa ini, gambar-gambar yang dihasilkan dan ditampilkan pada komputer menjadi bagian kehidupan sehari-hari yang dapat ditemui misalnya pada televisi, koran dan majalah yang fungsinya untuk menampilkan hasil yang lebih komunikatif dan realistis. Selain itu juga grafika komputer ditemukan pada bidang-bidang kedokteran, geologi dan tak terkecuali dalam bidang pendidikan untuk pengajaran dan penulisan karya-karya ilmiah (John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D.Foley, Steven K. Feiner, Kurt Akeley, 2014).

Grafika komputer pada dasarnya adalah suatu bidang komputer yang mempelajari cara meningkatkan dan memudahkan komunikasi antar manusia dengan mesin dengan cara membangkitkan, menyimpan dan memanipulasi gambar atau model suatu objek menggunakan komputer. Grafika komputer memungkinkan manusia untuk berkomunikasi lewat gambar, bagan dan diagram. Grafika komputer merupakan studi tentang bagaimana membuat gambar dan bagaimana memanipulasinya dengan menggunakan komputer (Juan Nicky Aristo Pattymahu & Oktoverano Lengkong, 2016).

## 2.2.1 Grafik Transformasi Dua Dimensi

Ada dua cara untuk melakukan transformasi, transformasi objek dan transformasi koordinat. Pada transformasi objek,semua titik di sembarang objek akan di ubah sesuai aturan tertentu, sedangkan system koordinatnya tetap. Objek pada transformasi sistem koordinat tetap. Namun karena system koordinatnya yang diubah, maka kedudukan objek harus disesuaikan dengan kedudukan system koordinat yang baru.

Transformasi pada dasarnya adalah mengubah posisi setiap titik, misalnya Py dari sembarang objek ke posisi yang lain. Sebagai contoh adalah Q yang menggunakan persamaan atau algoritma. Hal itu berarti terdapat suatu fungsi T yang memetakan koordinat P menjadi koordinat Q dan bisa Dituliskan Sebagai berikut :

T (Px, Py) = (Qx, Qy)

Atau bisa juga ditulis : Q = T (P)

Dengan P = (Px, Py) dan Q = (Qx, Qy).

Dalam transformasi 2D, Fungsi transformasi T akan memetakan P = (Px, Py) menjadi Q = (Qx, Qy). Qx dan Qy berhubungan dengan Px dan Py berdasarkan persamaan berikut :

Qx = aPy + cPy + TRx

Qy = bPx + dPy + Try

Adanya a, b, c, d menjadikan trx dan try sebagai sembarang konstanta (Sianturi, Edy Victor Haryanto).

## 2.3 *Game*

*Video game* diartikan sebagai: *a game played by electronically manipulating images produce by a computer program on a monitor or other display*. *Video game* adalah suatu permainan yang dimainkan melaluimanipulasi gambar elektronik yang diproduksioleh program komputer dalam monitor atautampilan layar lainnya. Jenis permainan ini menekankan pada permainan menggunakan tombol yang terhubung pada layar (Wahyudhi, 2014).

*Game* berasal dari kata bahasaInggris yang memiliki arti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual (*intellectual* *playability*)”. Game juga bisa diartikan sebagai arena keputusan dan aksi pemainnya, ada target-target yang ingin dicapai pemainnya. Kelincahan intelektual pada tingkat tertentu merupakan ukuran sejauh mana *game* itu menarik untuk dimainkan secara maksimal. Game juga secara nyata mempertajam daya analisis para penggunanya untuk mengolah informasi dan mengambil keputusan cepat yang jitu (Novaliendri, 2013).

Jeannie Novak dan Luis Levy membagi jenis perangkat *video game* dalam 4 jenis, yaitu: *arcade*, *console*, *handheld* dan *computer* (Novak, Jeannie & Luis Levy, 2008).

Berikut penjelasannya:

1. *Arcade games*

*Arcade* merujuk pada permainan *video game* menggunakan layar khusus disertai tombol-tombol di bawahnya (Novak, Jeannie & Luis Levy, 2008).

Memiliki *box* atau mesin yang memang didesain khusus untuk jenis *video game* tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa "masuk" dan "menikmati", seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya) (Iman, 2013).



Gambar 2.1 Permainan *Pac-man* merupakan *arcade game.*

**http://www.greatneckgames.com/images/games/arcadegames/large/ Ms-pacman.jpg**

1. PC *games*

*Video game* yang dimainkan secara personal layaknyapada konsol. *Video game* dalam perangkat inibiasanya terinstalisasi dalam format aplikasi,sama bentuknya dengan aplikasi pupulerkomputer lainnya semacam dan *Windows Media Player*. Sebelum memainkannya, *gamer* diharuskan menginstal *video game* ini dariCD (*Compact Disc*). *Video game* format komputersengaja dibuat untuk memberikan hiburan dalamsatu perangkat yang biasanya digunakan orang sebagai instrumen penting dalam pekerjaannya (Novak, Jeannie & Luis Levy, 2008).



Gambar 2.2 DOTA2 merupakan PC *game*.

**http://static5.gamespot.com/uploads/scale\_super/mig/6/7/0/9/2046709-610657\_20130718\_011.jpg**

1. *Console games*

* Console* atau konsol merupakan perangkat *video game* yang membutuhkan perankat elektronik lain sebagai penunjangnya. Beberapa jenis konsol yang sudah akrab di telingaadala *Playstation* (PS) seri 1 sampai 4, *Sega,* *X-Box, Dreamcast* dan lain-lain. *Konsol* terdiri dari satu perangkat pemutar *video game* dan dua alat penunjang sebagai kontrol permainan yang lazim disebut *console controller*, *gamepad*, *joystick* atau lebih lumrah di negeri ini disebut *stick* (Novak, Jeannie & Luis Levy, 2008)(Iman, 2013).

Gambar 2.3 Perangkat Konsol (Nintendo Wii, Xbox 360, dan Playstation 3).

**http://www.nopomobilebikeshop.com/wpcontent/uploads/2016/10/Last-Gen-Gaming-Consoles.jpg**

1. *Handheld games*

Mesin untuk memainkan *game*, bentuknya tidak lebih besar dari genggaman kedua tangan, mudah dibawa dalam genggaman, dan ringan (Iman, 2013).

Handheld merujuk pada perangkat game yang bersifat mobile, bisa dibawa kema-mana, oleh karena bentuknya yang hampir seukuran dengan hand phone. PS menjadi satu yang populer mengeluarkan terobosan video game berbasis handheld, dengan produk unggulannya seperti PS 2 Portable dan PS Vita (Novak, Jeannie & Luis Levy, 2008).



Gambar 2.4 SONY PSP (PlayStation Portabel) merupakan *platform* untuk *handheld games*.

**http://www.technologizer.com/wpcontent/uploads/2009/04/pspslim.jpg**

*Game* juga dapat diklasifikasikan berdasarkan *genre*. *genre* merupakan jenis atau tipe dari permainan, berikut *game* yang diklasifikasikan berdasarkan *genre* (Iman, 2013):

1. Aksi-*Shooting*

*Video game* jenis ini sangat memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari *game* jenis ini adalah tembak-menembak.



Gambar 2.5 Inilah tampilan *game shooting* 3D.

**https://lh3.ggpht.com/D2\_G7b8jJ3w4BV9YCHw7zzxkRaq1yy4hD6z1WMJ2st04onssyyPtYPTdFEYcmF72g=h900**



Gambar 2.6 *Gang Garrison 2* merupakan *game shooting* 2D dengan sudut pandang orang ketiga.

**https://lh3.ggpht.com/D2\_G7b8jJ3w4BV9YCHw7zzxkRaq1yy4hD6z1WMJ2st04onssyyPtYPTdFEYcmF72g=h900**

1. *Fighting* (pertarungan)

Jenis permainan ini memang memerlukan kecepatan refleks dan koordinasi mata-tangan, tetapi inti dari *game* ini adalah penguasaan jurus, pengenalan karakter dan timing sangatlah penting, *combo-*pun menjadi esensial untuk mengalahkan lawan secepat mungkin. Dan berbeda seperti *game* aksi pada umumnya yang hanya melawan komputer saja, pemain jenis *fighting game* ini baru teruji kemampuan sesungguhnya dengan melawan pemain lainnya.



Gambar 2.7 *Tekken* merupakan *game fighting* yang dikenal banyak orang.

**http://i.kinjaimg.com/gawkermedia/image/upload/t\_original/tnw3aacqqjcjjcmmn3zp.jpg**

1. Aksi-*Adventure*

Memasuki gua bawah tanah, melompati bebatuan di antara lahar panas, bergelayutan dari pohon satu ke pohon lain, bergulat dengan ular sambil mencari kunci untuk membuka pintu kuil legendaris, atau sekedar mencari telepon umum untuk mendapatkan misi berikutnya, itulah beberapa dari banyak hal yang karakter pemain harus lakukan dan lalui dalam *video games* jenis ini.



Gambar 2.8 *Game* aksi-*adventure* salah satunya adalah *Assassin's Creed*.

**https://megadownloadcore.files.wordpress.com/2014/06/a1.jpeg**

1. Simulasi

*Video game* jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor.



Gambar 2.9 Salah satu permainan simulasi yaitu *The Sims.*

**http://www.inlovewithandroid.com/images/best-simulation-games-for-android-the-sims-free-play3.jpg**

1. *Role Playing*

*Video game* jenis ini sesuai dengan terjemahannya, bermain peran, memiliki penekanan pada tokoh atau peran perwakilan pemain di dalam permainan, yang biasanya adalah tokoh utamanya. Dimana seiring kita memainkannya, karakter tersebut dapat berubah dan berkembang ke arah yang diinginkan pemain (biasanya menjadi semakin hebat, semakin kuat, semakin berpengaruh, dan lain-lain) dalam berbagai parameter yang biasanya ditentukan dengan naiknya *level*, baik dari status kepintaran, kecepatan, kekuatan karakter, senjata yang semakin hebat, ataupun jumlah teman maupun makhluk peliharaan (*pet*).



Gambar 2.10 Tampilan status karakter.

**https://i.ytimg.com/vi/PHR3yHgXRrk/maxresdefault.jpg**



Gambar 2.11 Tampilan pertarungan dalam salah satu *game* RPG.

**http://static.gamespot.com/uploads/original/416/4161502/2431351-0002.jpg**

1. Strategi

Kebalikan dari *video game* jenis *action* yang berjalan cepat dan perlu refleks secepat kilat, *video game* jenis strategi, layaknya bermain catur, justru lebih memerlukan keahlian berpikir dan memutuskan setiap gerakan secara hati-hati dan terencana.



Gambar 2.12 Clash of Clans merupakan permainan strategi.

**https://i.ytimg.com/vi/ck7GFdGAeKg/maxresdefault.jpg**

1. *Puzzle*

*Video game* jenis ini sesuai dengan namanya berintikan menganai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kotak masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini.



Gambar 2.13 Salah satu *puzzle* *game* adalah *Cut The Rope*.

**http://cdn3.macworld.co.uk/cmsdata/reviews/3497454/cut\_the\_rope\_800x450\_thumb800.jpg**

*Genre game* yang ada pada tugas akhir ini adalah Aksi-*Shooting* dan juga dalam 2D (dua dimensi). Nama *game* yang digunakan pada tugas akhir ini adalah *Gang Garrison 2*.

## 2.4 Artificial Intelligence

Kecerdasan buatan memang kerap diidentikkan dengan kemampuan robot yang dapat berperilaku seperti manusia (Desiani A & Arhami M, 2006). Berbagai definisi diungkapkan oleh para ahli untuk dapat memberi gambaran mengenai kecerdasan buatan beberapa diantaranya :

a. Kecerdasan Buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas.

b. Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia.

c. Kecerdasan Buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan bentuk simbol-simbol daripada bilangan, dan memproses informasi berdasarkan metode heuristik atau dengan berdasarkan sejumlah aturan.

*Artificiall intelligence* adalah agen-agen yang ada untuk belajar melalui lingkungan dan persepsi serta tindakan (Russel, Stuart J. and Norvig, Peter, 1995)

Ada beberapa tipe agen *artificial intelligence*:

1. *Reasoning*

*Reasoning* adalah serangkaian proses yang memungkinkan untuk menyediakan dasar untuk penilaian, membuat keputusan, dan prediksi.

1. *Learning*

*Learning* adalah kegiatan memperoleh pengetahuan atau keterampilan dengan mempelajari, berlatih, diajarkan, atau mengalami sesuatu. *Learning* meningkatkan pengetahuan ke agen yang diteliti.

*Learning* dapat dikategorikan sebagai berikut :

* *Auditory Learning*

*Auditory learning* belajar dengan cara mendengarkan dan pendengaran

* *Episodic Learning*

*Episodic learning* belajar dengan cara mengingat urutan peristiwa yang telah disaksikan atau dialami. Hal ini sejalan dan teratur.

* *Motor Learning*

*Motor learning* belajar dari keakuratan gerakan dari suatu motor.

* *Observational Learning*

*Observational learning* belajar dengan mengamati dan meniru orang lain.

* *Perceptual Learning*

*Perceptual learning* belajar untuk mengenali sesuatu yang telah terlihat atau terjadi sebelumnya.

* *Relational Learning*

*Relational learning* belajar untuk membedakan antara berbagai input berdasarkan sifat relasional, bukan sifat mutlak.

* *Spatial Learning*

*Spatial learning* belajar melalui input visual seperti gambar, warna, peta, dan lainnya untuk diolah.

* *Stimulus-Response Learning*

*Stimulus-Response learning* belajar untuk melakukan perilaku tertentu ketika pemicu tertentu hadir.

1. *Problem Solving*

*Problem solving* adalah proses melakukan percobaan untuk mencapai solusi yang diinginkan dari situasi sekarang dengan cara mengambil beberapa jalur, baik rintangan yang diketahui atau tidak diketahui.

1. *Perception*

*Perception* adalah proses memperoleh, menafsirkan, memilih, dan mengatur informasi sensorik.

1. *Linguistic Intelligence*

*Linguistic intelligence* merupakan kemampuan menyerupai seseorang dalam menggunakan, memahami, berbicara, dan menulis bahasa verbal dan tertulis. Hal ini penting dalam komunikasi interpersonal.

Kemudian *machine learning* juga termasuk dalam *artificial intelligence* yang merupakan bagian dari disiplin ilmu matematika lanjutan yang mencakup pemodelan yang canggih, optimasi, dan pembelajar untuk penelitian. Algoritma *machine learning* dibagi menjadi beberapa subkategori, yakni *supervised learning, unsupervised, semi-supervised learning, transduction, reinforcement learning,* dan *developmental learning* (Hall, Patrick., Dean, Jared., Kabul, Iiknur Kaynar., & Silva, Jorge, 2014) .

## 2.5 Reinforcement Learning

Dalam hal banyaknya masalah yang kompleks, *reinforcement learning* merupakan satu-satunya cara yang layak untuk melatih sistem agar memiliki performa yang tinggi. Misalnya, dalam bermain *game*, sangat sulit bagi manusia untuk memberikan evaluasi yang akurat dan konsisten dari sejumlah besar posisi yang diperlukan untuk melatih fungsi evaluasi dari contoh. Sebaliknya, sistem ini dapat mengetahui ketika itu menang atau kalah, dan dapat menggunakan informasi tersebut untuk mempelajari fungsi evaluasi yang dapat memberikan perkiraan yang cukup akurat dari probabilitas menang dari posisi apapun. Disatu sisi, *reinforcement learning* adalah penyajian kembali seluruh masalah AI. *Bot* di lingkungan mendapatkan persepsi, memetakannya menjadi utilitas positif atau negatif, dan kemudian digunakan untuk memutuskan tindakan apa yang harus diambil (Russel, Stuart J. and Norvig, Peter, 1995).



Gambar 2.14 Model umum *reinforcement learning*.



Gambar 2.15 Masalah sederhana dalam *reinforment learning*.

Dengan 7 (tujuh) *state* dan 3 (tiga) *action* yang berbeda; kanan, kiri, dan diam. Tujuannya adalah agar kereta dorong di atas pindah ke *state* nomor 4 dan diam di sana.

## 2.6 Algoritma yang diajukan

*Bot* dapat belajar untuk bermain dengan diawasi cara belajarnya, dengan diberikan contoh-contoh situasi permainan bersama dengan langkah terbaik untuk situasi tertentu. Jika tidak diberikan contoh maka *bot* mencoba langkah acak, setelah itu *bot* dapat membangun model prediksi lingkungannya. Setelah memiliki model prediksi lingkungan, bot akan mencoba langkah baru dan mendapat umpan balik yang baru dari lingkungan. Tanpa adanya umpan balik, *bot* tidak memiliki alasan untuk memutuskan langkah yang diambil. Jika *bot* menerima umpan balik, *bot* dapat memutuskan menang atau kalah, umpan balik disebut *reward,* atau *reinforcement*.

*Reinforcement learning* dapat belajar dari pengetahuan yang didapatkan dari interaksi anatara sistem dengan lingkungan (Wiering, 2007). Secara umum algoritma *reinforcement learning* menggunakan *Markov Decision Prosess* (MDP) (Nathan Sprague, Dana Ballard, 2003).

Ada beberapa aturan algoritma *markov decision process* yang dapat digunakan dalam *reinforcement learning*  yakni *Q-Learning* dan *State-Action-Reward-State-Action* (SARSA). *Q-Learning* merupakan algoritma *off-policy* dimana agen belajar nilai pasangan *State-Action* yang tidak perlu berdasarkan dari tindakan yang dilakukan, karena *update* mereka dilakukan terlepas dari tindakan saat ini, melainkan berkaitan dengan tindakan yang dapat memaksimalkan nilai dari pasangan *State-Action* berikutnya. Aturan pada *Q-Learning* :

𝑄𝑘+1(𝑠𝑡 , 𝑎𝑡 ) = 𝑄𝑘 (𝑠𝑡 , 𝑎𝑡 ) + 𝛼 [𝑟𝑡+1 + 𝛾 max 𝑄𝑘 (𝑠𝑡+1, 𝑎𝑡+1) − 𝑄𝑘 (𝑠𝑡 , 𝑎𝑡 )]

𝑎

Dimana α ∈ (0,1) adalah yang disebut *learning rate*. Hal ini menentukan tingkat *update*.

*State-Action-Reward-State-Action* (SARSA) merupakan algoritma *on-policy* dimana agen belajar nilai *State-Action* berdasarkan tindakan yang dilakukan. Dan akan dilakukan evaluasi pada kebijakan sekarang. Tidak seperti *Q-Learning* yang melakukan satu kebijakan dan mengevaluasi satu sama lain (Wender, Stefan., & Watson, Ian., 2012).

SARSA memperbarui aturan menjadi sebagai berikut:

𝑄𝑘+1(𝑠𝑡 , 𝑎𝑡 ) = 𝑄𝑘 (𝑠𝑡 , 𝑎𝑡 ) + 𝛼[𝑟𝑡+1 + 𝛾𝑄𝑘 (𝑠𝑡+1, 𝑎𝑡+1) − 𝑄𝑘 (𝑠𝑡 , 𝑎𝑡 )]

Dimana aksi 𝑎𝑡 + 1 adalah *action* yang dilakukan pada *state* 𝑠𝑡 + 1 berdasarkan pada kebijakan saat ini (Corazza, Marco & Sangalli, Andrea, 2015).

Aturan algoritma *reinforcement learning* yang akan diterapkan pada *bot* adalah SARSA.

# BAB III

# ANALISIS DAN PERANCANGAN

Pengembangan *bot* pada permainan ini menggunakan metode *waterfall*, yang memiliki tahap analisis, perancangan, dan pengkodean. Pada bab ini akan dijelaskan secara rinci tahap analisis dan perancangan.

## 3.1 Analisis

Pada subbab analisis akan dijelaskan 2 (dua) tahapan analisis, yaitu analisis proses dan analisis kebutuhan. Berikut penjelasan masing-masing analisis.

## 3.1.1 Analisis Proses

Analisis proses menjelaskan tentang proses pembelajaran *bot* menggunakan *Reinforcement Learning* (RL). Pada penelitian ini, dipilih menggunakan pendekatan algoritma SARSA, proses pembelajaran akan dijelaskan secara bertahap dimulai dengan merekam cara gerak *bot* dan perhitungan reward untuk pergerakan, mengolah RL, membaca data gerakan pada RL *memory* dan merekam cara gerak *bot*, merekam cara *bot* menembak dan perhitungan reward untuk menembak serta membaca rekaman cara *bot* menembak.

## 3.1.1.1 Merekam Cara Gerak *Bot* dan Perhitungan *Reward* untuk Pergerakan

*Bot* akan diberikan instruksi untuk bergerak dengan berjalan ke kiri (L) , kanan (R), melompat (J) dan berhenti (S). Semua koordinat posisi pergerakan dari *bot* (*State*) dan langkah-langkah yang diambil oleh *bot* (*Action*) untuk mencapai koordinat tertentu (*State'*) akan direkam ke RL *memory*. Diagram rekam dan penyimpanan data dapat dilihat pada Gambar 3.1.

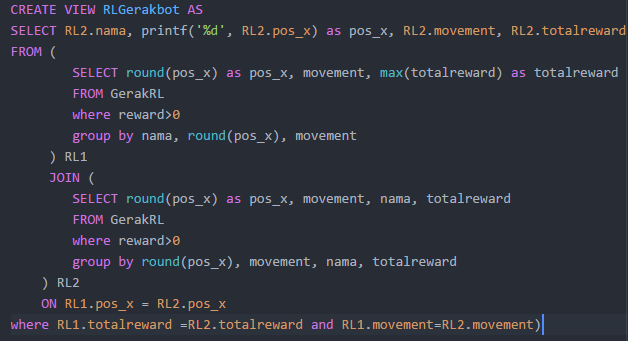
****

Gambar 3.1 Diagram rekam dan penyimpanan data serta Diagram Membaca Data Gerakan pada RL *Memory* dan Merekam Cara Gerak *Bot*.

Agar dapat memahami proses kerja rekam dan penyimpanan data, diberikan contoh kasus berikut. Pada tahap pertama, jalankan *bot* untuk mengeksplorasi lingkungan dalam batas waktu tertentu, setelah itu semua titik koordinat yang dicapai oleh *bot* akan disimpan dalam tabel, setelah itu akan dilakukan perhitungan reward untuk mengetahui nilai yang didapat oleh *bot* untuk setiap pergerakan yang dilakukan. Misalkan diambil pergerakan *bot* selama 99 kali permainan, hasilnya dapat dilihat pada Tabel 3.1.

## 3.1.1.2 Mengolah RL *Movement*

RL *Movement* dinormalisasi agar pembacaan data akan lebih efisien. Pada penormalisasi gerakan RL *Movement* dapat dilakukan dengan cara membatasi nilai ambang dari *state* (S/posisi), dan *action* yang berbeda (A) dengan total *reward* masing masing yang *maximum (*Rt) serta membuang semua *state* dan *action* yang tidak sesuai dengan *policy*. Dapat dilihat pada Gambar 3.2, *syntax* yang mengolah data gerak.



Gambar 3.2 *Syntax* dalam SQL untuk mengolah data gerak.

## 3.1.1.3 Membaca Data Gerakan pada RL *Movement* dan Belajar Cara Gerak *Bot*

Data Gerakan pada RL *Memory* yang sudah dinormalisasi akan dibaca untuk pergerakan *bot*. Semua koordinat posisi pergerakan dari *bot* dan langkah-langkah yang diambil oleh *bot* untuk mencapai koordinat tertentu akan kembali direkam lagi ke RL *Memory*. Perekaman data tersebut berfungsi sebagai bahan pertimbangan untuk pembelajaran berikutnya dan akan kembali diolah untuk mendapatkan peningkatan dalam pergerakan. Diagram dapat dilihat pada Gambar 3.1.

*Bot* akan diberikan instruksi untuk menembak dengan sudut tembak, jenis tembakan dan musuh yang ditargetkan. Pergerakan *bot* akan membaca Data Gerakan pada RL *Memory* dan hasil pergerakan yang terjadi tidak akan direkam. Perekaman data hanya akan terjadi apabila *bot* sudah mendekati musuh. Koordinat posisi pergerakan dari *bot* (*State*) dan aksi yang diambil oleh *bot* (*Action*) untuk menjatuhkan musuh akan direkam ke RL *Memory*. Diagram rekam dan penyimpanan data dapat dilihat pada Gambar 3.3.

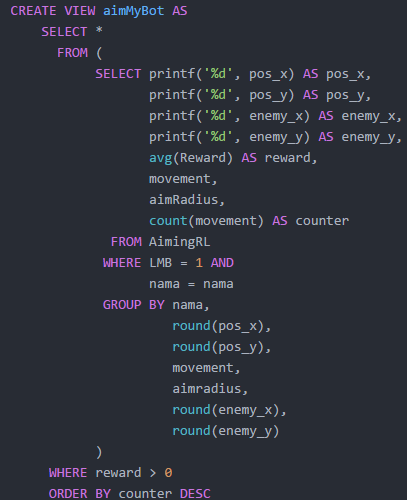


Gambar 3.3 Diagram Merekam Cara Menembak *Bot*.

Berikut diambil data pembelajaran *bot* selama 100 kali permainan, hasilnya dapat dilihat pada Tabel 3.3.

## 3.1.1.2 Mengolah RL *Shooting*

RL *Shooting* dinormalisasi agar pembacaan data lebih efisien dan kemudian ditambah pembelajaran sampai 800 kali. Pada penormalisasi aksi RL *Shooting* dapat dilakukan dengan merata-ratakan nilai semua *reward* tembakan pada *state* dan *action* yang memiliki posisi dan sudut yang sama. Untuk *syntax* pada SQL dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Syntax* dalam SQL untuk mengolah data menembak.

Berikut diambil data pembelajaran *bot* yang sudah dinormalisasi setelah 400 kali pembelajaran, hasilnya dapat dilihat pada Tabel 3.4.

Setelah melakukan normalisasi pada 400 kali pembelajaran, *bot* akan diberikan pembelajaran melawan bot lain sampai mencapai 800 kali pembelajaran dan RL *Shooting* akan dilakukan normalisasi.

Berikut diambil data pembelajaran *bot* yang sudah dinormalisasi setelah 400 kali pembelajaran, hasilnya dapat dilihat pada Tabel 3.5.

## 3.1.2 Analisis Kebutuhan

Analisis kebutuhan merupakan tahapan penting dalam pengembangan sistem untuk mengetahui secara detail sistem yang ingin dikembangkan. Kebutuhan sistem akan ditentukan oleh pemilik dan pengguna sistem. Analisis kebutuhan dibagi menjadi dua, yaitu analisis kebutuhan fungsional yang merupakan spesifikasi inti mengenai hal-hal yang bisa dilakukan oleh sistem dan kebutuhan non-fungsional yang merupakan komponen pendukung pada sistem.

## 3.1.2.1 Kebutuhan Fungsional

1. **Kebutuhan Fungsional untuk Pembelajaran**

Adapun beberapa persyaratan fungsional yang harus dipenuhi oleh perangkat lunak adalah sebagai berikut:

1. Perangkat Lunak Game Maker Ver 8.0 untuk menjalankan permainan.
2. Perangkat Lunak SQL Lite untuk mengolah data pembelajaran.
3. Mengunakan Script GML untuk melakukan *coding*.
4. Mengunakan *source code* Gang Garrison 2.
5. Text Files untuk menulis dan membaca data.
6. Pengunaan Version Control Git untuk melakukan pengaturan versi.
7. Instruksi dasar pergerakan bot sebagai pengetahuan awal dari *bot*.
8. Nilai ambang untuk mengoptimalkan data yang telah ditulis.
9. **Kebutuhan Fungsional untuk Pengujian**

Adapun beberapa persyaratan fungsional yang harus dipenuhi untuk melakukan pengujian adalah sebagai berikut:

1. Data pembelajaran yang disimpan sudah diolah.
2. Pengujian dilakukan dengan cara membaca data yang sudah diolah.
3. Proses pengujian dilakukan dengan memulai permainan antara *bot* RL dengan *bot* dari pengembang lain.
4. Proses pengujian dimulai dengan pembacaan data dari database, kemudian bot akan memilih rangkaian gerakan yang memiliki reward yang optimal. Hasil yang diperoleh dari satu set permainan menentukan apakah *bot* sudah mengetahui cara untuk memperoleh kemenangan.

## 3.1.2.2 Kebutuhan Non-Fungsional

Untuk merumuskan persyaratan non-fungsional dari sistem, maka harus dilakukan analisis terhadap kinerja, informasi, ekonomi, keamanan aplikasi, efisiensi, dan pelayanan customer. Panduan ini dikenal dengan analisis PIECES (*performance*, *information*, *economic*, *control*, *efficiency*, dan *services*). Berikut penjabaran dari analisis yang digunakan :

1. **Kebutuhan Non-Fungsional untuk Pembelajaran**
2. *Performance*

Penulisan data dilakukan pada RAM terlebih dahulu untuk mengurangi beban pada *disk* (memori utama).

1. *Information*

Data yang telah ditulis disimpan dalam Text Files.

1. *Efficiency*

Data yang sudah ditulis diolah terlebih dahulu sebelum dibaca.

1. **Kebutuhan Non-Fungsional untuk Pengujian**
2. *Performance*

Pembacaan data dilakukan pada RAM terlebih dahulu untuk mengurangi beban pada *disk* (memori utama).

1. *Information*

Data dari *bot* dapat ditampilkan ketika bot melakukan pergerakan.

1. *Efficiency*

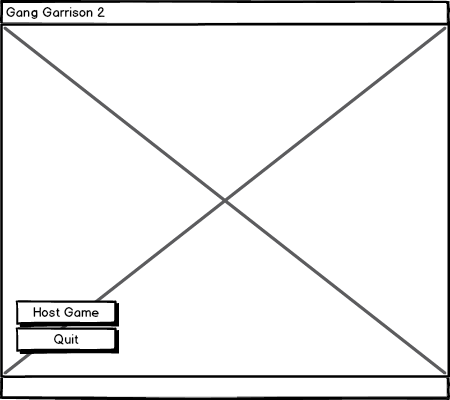
Data yang dibaca sudah diolah terlebih dahulu untuk mendapatkan hasil yang optimal.

## 3.2 Pemodelan Sistem dan Perancangan

Pada tahap ini, dilakukan pemodelan untuk tampilan yang sudah dibangun.

**a. Pemodelan Halaman Utama**

Tampilan ini merupakan tampilan pembuka dari aplikasi dan merupakan tampilan utama yang berfungsi untuk menghubungkan ke tampilan berikutnya. Pemodelan tampilan utama dapat dilihat pada Gambar 3.5.

****

Gambar 3.5 Mockup tampilan awal.

Pada tampilan awal terdapat beberapa menu yaitu :

1. Host Game

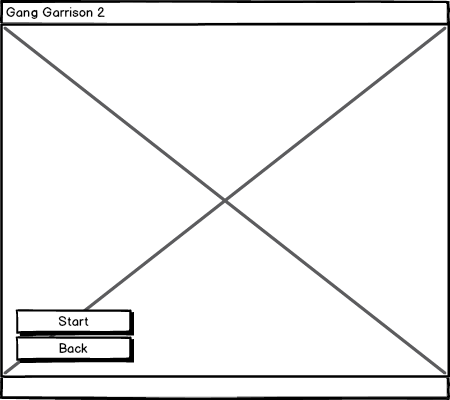
Membuat server dengan settingan yang diatur oleh pemain sendiri.

1. Quit

Keluar dari aplikasi permainan.

**b. Pemodelan Tampilan Host Game**

Pemodelan ini adalah kelanjutan dari pemodelan tampilan utama. Dimana pemodelan ini adalah tampilan untuk mengatur *server* untuk melakukan permainan. Pemodelan dapat dilihat pada Gambar 3.6.



Gambar 3.6 Mockup menu Host Game.

Pada tampilan awal terdapat beberapa menu yaitu

1. Start

Memulai permainan.

1. Back

Kembali ke tampilan sebelumnya.

**b. Pemodelan Tampilan Permainan**

Pemodelan ini menampilkan kita dapat memilih bermain sebagai tim merah atau tim biru. Juga dapat menggunakan *autoselect* dalam pemilihan tim.



Gambar 3.6 Mockup saat Start Game.

Keterangan mockup :

1. 1 : Memilih tim secara acak ( tidak digunakan )
2. 2 : Memilih sebagai penonton atau pengamat dalam permainan.
3. 3 : Memilih bermain sebagai tim merah ( tidak digunakan )
4. 4 : Memilih bermain sebagai tim biru ( tidak digunakan )
5. Keterangan : Menampilkan keterangan pada menu yang di pilih *pointer*.
6. Peta Game : Menampilkan peta permainan.

**c. Perancangan sistem menggunakan activity diagram.**

Pada perancangan ini dijelaskan bagaimana cara kerja *bot* merekam dan membaca rekaman data. Activity Diagram dapat dilihat pada Gambar 3.7.



Gambar 3.7 Activity Diagram.

1. Run Game

Memulai permainan, bot akan diberikan instruksi untuk bergerak dengan berjalan ke kiri (L) , kanan (R), melompat (J) dan berhenti (S).

1. Learn Move

Semua koordinat posisi pergerakan dari bot (State) dan langkah-langkah yang diambil oleh *bot* (Action) untuk mencapai koordinat tertentu (State') akan direkam. Kemudian dilakukan pengecekan apakah *bot* mempelajari gerakan baru, jika ada maka akan dilakukan perhitungan *reward* (*Calculate Reward*) dan data pembelajaran akan disimpan pada RL *Memory*.

1. Optimized Data & Run Game

RL *Memory* dinormalisasi agar pembacaan data menjadi lebih efisien. Penormalisasian gerakan pada RL *Memory* dapat dilakukan dengan cara membatasi nilai ambang dari *state* (S/posisi), dan *action* yang bebeda (A) dengan total reward masing masing yang *maximum* (Rt) serta membuang semua *state* dan *action* yang dilakukan dengan *reward* <= 0. Kemudian dilakukan permainan baru untuk belajar menembak.

1. Learn Shooting

*Bot* akan diberikan instruksi untuk menembak dengan sudut tembakan, jenis tembakan dan musuh yang ditargetkan. Pergerakan *bot* akan membaca Data Gerakan pada RL *Memory* dan hasil pergerakan yang terjadi tidak akan direkam. Perekaman data hanya akan terjadi apabila *bot* sudah mendekati musuh. Koordinat posisi pergerakan dari *bot* (*State*) dan aksi yang diambil oleh *bot* (*Action*) untuk menjatuhkan musuh akan direkam ke RL *Memory.*

# BAB IV

# HASIL DAN PENGUJIAN

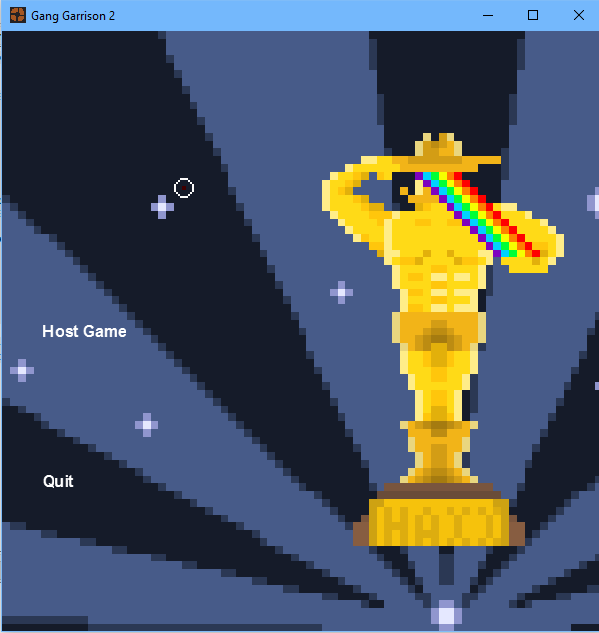
Pada bab ini menjelaskan tentang hasil dari pengujian-pengujian yang telah dilakukan terhadap *bot*.

## 4.1 Tampilan

Pada subbab ini akan ditampilkan *screenshot* dari aplikasi yang dijalankan.

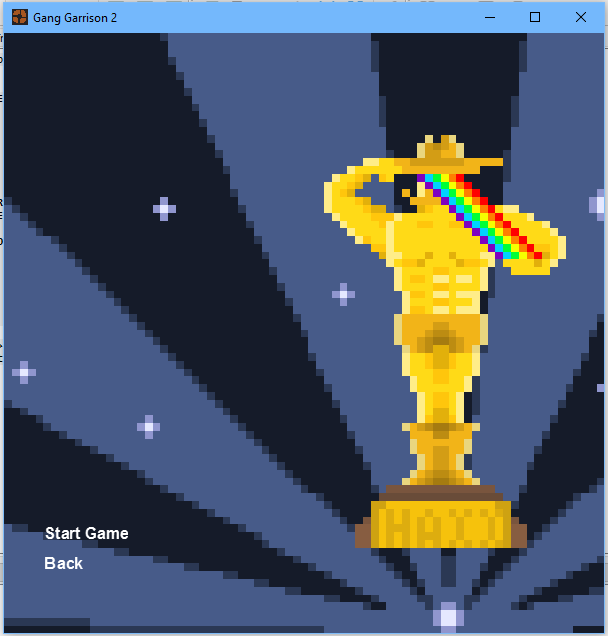
## 4.1.1 Tampilan Menu Utama

Tampilan halaman menu utama seperti terlihat pada Gambar 4.1.



Gambar 4.1 Tampilan Awal.

## 4.1.2 Tampilan Host Game



Gambar 4.2 Tampilan *host game*.

Tampilan halaman *host game* seperti yang terlihat pada gambar 4.2 diatas berfungsi untuk memulai permainan dengan memilih menu *start.*

## 4.1.3 Tampilan Pemilihan Tim

****

Gambar 4.3 Tampilan mulai permainan.

Tampilan halaman Pemilihan Tim seperti yang terlihat pada gambar 4.3 diatas berfungsi sebagai tempat *user* untuk memilih untuk bermain di tim merah atau biru atau sebagai penonton.

Keterangan :

1. 1 : Memilih tim secara acak antara tim biru atau tim merah.
2. 2 : Memilih sebagai penonton atau pengamat dalam permainan.
3. 3 : Memilih bermain sebagai tim merah.
4. 4 : Memilih bermain sebagai tim biru.

## 4.1.4 Mode Permainanan *Generator*



Gambar 4.4 Peta permainan *generator*.

Tampilan pada Gambar 4.4 diatas merupakan peta dari permainan, dimana pada sisi kiri terdapat tim merah yang merupakan bot dari pengembang lain (*Tempest Bot*) dan pada sisi sebelah kanan merupakan *bot* yang menggunakan Reinforcement Learning.



Gambar 4.5 Tampilan markas dan generator dari tim merah.

****

Gambar 4.6 Tampilan markas dan generator dari tim biru.

Pada gambar 4.5 dapat dilihat Markas dan Generator dari Tim Merah serta sisa waktu permainan. Pada gambar 4.6 dapat dilihat markas dan generator dari tim biru dan keadaan atau status generator dari kedua Tim.

## 4.2 Hasil

Pada subbab ini ditampilkan data-data pembelajaran yang telah direkam, kemudian menggunakan data pembelajaran untuk menguji *bot*. Setelah diuji, hasil dari pengujian akan dibandingkan antara data yang belum dinormalisasi dengan data yang sudah dinormalisasi.

## 4.2.1 Hasil Pembelajaran Pergerakan

Pada tahap ini ditampilkan data pembelajaran dari My Bot 1 sampai My Bot 3 berupa tabel serta grafik pergerakan bot. Data pembelajaran ini belum diolah, karena data ini menjadi bahan untuk perbandingan dengan data yang sudah dinormalisasi.

Gambar 4.7 Grafik Nilai Gerak My bot 1.

Pada Tabel 4.1 dan Gambar 4.7 dapat dilihat perolehan total reward MyBot1 selama 99 kali permainan, bot mencoba berbagai pergerakan untuk mencapai titik terjauh.

Gambar 4.8 Grafik Nilai Gerak My bot 2.

Pada Tabel 4.2 dan Gambar 4.8 dapat dilihat perolehan total reward MyBot2 selama 99 kali permainan, bot mencoba berbagai pergerakan untuk mencapai titik terjauh.

Gambar 4.9 Grafik Nilai Gerak My bot 3.

Pada Tabel 4.3 dan Gambar 4.9 dapat dilihat perolehan total reward MyBot3 selama 99 kali permainan, bot mencoba berbagai pergerakan untuk mencapai titik terjauh.

## 4.2.2 Hasil Pembelajaran Menembak

Pada subbab ini ditampilkan data pembelajaran dari My Bot 1 sampai My Bot 3 berupa tabel serta grafik menembak bot. Data pembelajaran ini belum diolah, karena data ini menjadi bahan untuk perbandingan dengan data yang sudah dinormalisasi.

Gambar 4.10 Grafik Nilai Menembak Mybot 1.

Gambar 4.11 Grafik Nilai Menembak Mybot 2.

Gambar 4.12 Grafik Nilai Menembak Mybot 3.

## 4.3 Pengujian

Berikut adalah pengujian data dengan melakukan pembacaan terhadap RL *Movement* dan RL *Shooting*. Dalam melakukan pembelajaran, bot melakukan 800 (delapan ratus) kali pembelajaran terlebih dahulu, terdiri dari 100 kali pembelajaran pergerakan, 300 (tiga ratus) kali pembelajaran menembak dan 400 (empat ratus) kali pembelajaran melawan *bot* dari pengembang lain.

Pengujian pertama dilakukan ketika *bot* sudah melakukan pembelajaran sebanyak 200 (dua ratus) kali yaitu belajar bergerak dan menembak masing-masing 100 (seratus), kemudian akan dilakukan pengujian sebanyak 10 (sepuluh) kali tanpa melawan *bot* untuk mengetahui hasil yang didapat oleh *bot*.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata akurasi dengan menggunakan hasil pengujian pertama dengan total pembelajaran 200 kali. Tabel 4.7 merupakan hasil total rata-rata akurasi dan reward dari 200 kali pembelajaran. Gambar 4.13 merupakan hasil dari 10 kali pengujian setelah 200 kali pembelajaran.

Gambar 4.13 Grafik Nilai Pengujian Akurasi Pertama My Bot 1, 2, dan 3.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata akurasi dengan menggunakan hasil pengujian kedua dengan total pembelajaran 300 kali. Tabel 4.8 merupakan hasil total rata-rata akurasi dari 300 kali pembelajaran. Gambar 4.14 merupakan hasil dari 10 kali pengujian setelah 300 kali pembelajaran.

Gambar 4.14 Grafik Hasil Pengujian Akurasi Kedua My Bot 1, 2, dan 3.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata akurasi dengan menggunakan hasil pengujian ketiga dengan total pembelajaran 400 kali. Tabel 4.9 merupakan hasil total rata-rata akurasi dari 400 kali pembelajaran. Gambar 4.15 merupakan hasil dari 10 kali pengujian setelah 400 kali pembelajaran.

Gambar 4.15 Grafik Nilai Pengujian Akurasi Ketiga My Bot 1, 2, dan 3.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata akurasi dengan menggunakan hasil pengujian keempat dengan total pembelajaran 800 kali. Tabel 4.10 merupakan hasil total rata-rata akurasi dari 800 kali pembelajaran. Gambar 4.16 merupakan hasil dari 10 kali pengujian setelah 800 kali pembelajaran.

Gambar 4.16 Grafik Nilai Pengujian Akurasi Keempat My Bot 1, 2, dan 3.

Perbedaan pada akurasi disebabkan karena perbedaan jenis senjata yang digunakan oleh masing-masing bot. Hal tersebut menyebabkan perbedaan kecepatan menembak peluru dan jumlah peluru yang dikeluarkan per tembakan. Bot 1 memiliki senjata dengan tembakan tercepat dan mengeluarkan 1 peluru per tembakan. Bot 2 memiliki kecepatan menembak yang lambat dan menembakkan 1 peluru per tembakan. Bot 3 memiliki memiliki kecepatan menembak lebih baik dari bot 2 tetapi tidak lebih baik dari bot 1 dan menembakkan 6 peluru dalam satu tembakan.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata reward dengan menggunakan hasil pengujian pertama dengan total pembelajaran 200 kali. Tabel 4.11 merupakan hasil total rata-rata reward dari 200 kali pembelajaran. Gambar 4.17 merupakan hasil dari 10 kali pengujian setelah 200 kali pembelajaran.

Gambar 4.17 Grafik Hasil Pengujian Reward Pertama My Bot 1, 2, dan 3.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata reward dengan menggunakan hasil pengujian kedua dengan total pembelajaran 300 kali. Tabel 4.12 merupakan hasil total rata-rata reward dari 300 kali pembelajaran. Gambar 4.18 merupakan hasil dari 10 kali pengujian setelah 300 kali pembelajaran.

Gambar 4.18 Grafik Hasil Pengujian Reward Kedua My Bot 1, 2, dan 3.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata reward dengan menggunakan hasil pengujian ketiga dengan total pembelajaran 400 kali. Tabel 4.13 merupakan hasil total rata-rata reward dari 400 kali pembelajaran. Gambar 4.19 merupakan hasil dari 10 kali pengujian setelah 400 kali pembelajaran.

Gambar 4.19 Grafik Hasil Pengujian Reward Ketiga My Bot 1, 2, dan 3.

Berikut ini adalah hasil pengujian dengan menghitung rata-rata reward dengan menggunakan hasil pengujian ketiga dengan total pembelajaran 800 kali. Tabel 4.14 merupakan hasil total rata-rata reward dari 800 kali pembelajaran. Gambar 4.20 merupakan hasil dari 10 kali pengujian setelah 800 kali pembelajaran.

Gambar 4.20 Grafik Hasil Pengujian Reward Keempat My Bot 1, 2, dan 3.

Perbedaan pada reward disebabkan karena 2 hal yaitu jenis senjata dan kecepatan gerak. Bot 1 memiliki kekuatan tembakan lemah dan kecepatan gerak paling lambat. Bot 2 memiliki kekuatan tembakan terkuat dan kecepatan gerak stabil. Bot 3 memiliki kekuatan tembak setara dengan Bot 1 dan memiliki kecepatan gerak tercepat.

Berikut ditampilkan grafik perbandingan rata-rata akurasi dan reward dari semua pengujian. Gambar 4.21 merupakan perbandingan rata-rata akurasi. Gambar 4.22 merupakan perbandingan rata-rata reward.

Gambar 4.21 Perbandingan rata-rata akurasi My Bot 1, 2, dan 3.

Gambar 4.22 Grafik Perbandingan rata-rata reward My Bot 1, 2, dan 3.

# BAB V

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil penelitian terhadap pengembangan bot pada game Gang Garrison 2 dengan *Reinforcement Learning*, dapat disimpulkan bahwa :

1. Perbandingan akurasi dan reward dapat berbeda karena ada faktor yang mempengaruhi seperti jumlah tembakan.
2. Semakin banyak pembelajaran tidak menjamin hasil dapat menjadi lebih baik karena belum tentu pembelajaran ke depan dapat menghasilkan reward yang maksimal.

## 5.2 Saran

Adapun saran dari hasil penelitian ini adalah menggunakan pendekatan lain untuk memaksimalkan akurasi dan reward serta diharapkan dapat dilakukan uji coba dengan semua class.

# DAFTAR PUSTAKA

Aniati, M. (1992). *Pengantar Pengolahan Citra.* Jakarta: P.T. Elex Media Komputindo & Pusat Antar Universitas Ilmu Komputer Universitas Indonesia.

Corazza, Marco & Sangalli, Andrea. (2015). *Q-Learning and SARSA : a comparison between two intelligent stochastic control approaches financial trading* .

Desiani A & Arhami M. (2006). *Konsep Kecerdasan Buatan.* Yogyakarta: Penerbit ANDI.

Gonzalez R.C. & Woods R.E. (2002). *Digital Image Processing Second Edition.* New Jersey: Prentice Hall Inc.

Hall, Patrick., Dean, Jared., Kabul, Iiknur Kaynar., & Silva, Jorge. (2014). *, An Overview of Machine Learning with SAS¬¬¬® Enterprise Miner™.*

Hermawati. (2013). *Pengolahan Citra Digital.* Yogyakarta: Penerbit ANDI.

Iman, N. (2013). *Pembangunan Aplikasi Game Hybird Shooter Side-scrolling Destroyer Garuda Berbasis Desktop.* http://elib.unikom.ac.id/files/disk1/621/jbptunikompp-gdl-nurulimann-31004-10-13.unik-i.pdf.

John F. Hughes, Andries Van Dam, Morgan Mcguire, David F. Sklar, James D.Foley, Steven K. Feiner, Kurt Akeley. (2014). *Computer Graphics : Principles and Practice (3rd edition).* Addison-Wesley.

Juan Nicky Aristo Pattymahu & Oktoverano Lengkong. (2016). *APLIKASI VIRTUAL EKSPLORASI RSUP PROF DR. R. D. KANDOU MANADO* , 2.6-32.

Nathan Sprague, Dana Ballard. (2003). Multiple-Goal Reinforcement Learning with Modular Sarsa (0). http://library.mpib-berlin.mpg.de/toc/ze\_2006\_1479.pdf.

Novak, Jeannie & Luis Levy. (2008). *Play The Game; The Parent’s Guide to Video Game.* Boston: Thomson Course Technology.

Novaliendri, D. (2013). Aplikasi Game Geografi Berbasis Multimedia Interaktif (Studi Kasus Siswa Kelas IX SMPN 1 RAO). *Jurnal Teknologi Informasi & Pendidikan 6.2* , 1-100.

Russel, Stuart J. and Norvig, Peter. (1995). *Artificial Intelligence A Modern Approach.* http://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf.

Sianturi, E. V., *SISTEM TRANSFORMASI LUKISAN OBJEK DUA DIMENSI DAN TIGA DIMENSI PADA GRAFIKA KOMPUTER DENGAN MENGGUNAKAN MATRIKS TRANSFORMASI* , 133.

Wahyudhi, J. (2014). *Video game sebagai media pembelajaran sejarah (suatu alternatif dalam menyelenggarakan pembelajaran sejarah* , SOSIO-DIDAKTIKA: Social Science Education Journal 1.2 (2014): 199-210.

Wender, Stefan., & Watson, Ian. (2012). *Applying Reinforcement Learning to Small Scale Combat in the Real-Time Strategy Game StarCraft:Broodwar* , http://geneura.ugr.es/cig2012/papers/paper44.pdf.

Wiering, M. A. (2007). *Learning to Play.* http://pad.twiki.di.uniroma1.it/pub/ApprAuto/WebHome/lerningchessgames.pdf.

**LISTING PROGRAM**

**Objects->InGameElements->MyPlayer**

**Create Event:**

execute code:

{

object = -1;

team = TEAM\_SPECTATOR;

class = CLASS\_SOLDIER;

name = "My Bot";

destroy = 0// Gets set to one when you need to destroy a bot

reward =0//moving reward

totalreward=0 //just for movement finding optimal moving

shootreward=0//shooting reward

oldshootreward=0

domove=false

sql="" //sql syntax

sqlvalue[0]=0 //sqlvalue length

sqlvalue2[0]=0 //sqlvalue length for shooting sql

dowrite=true

prefermove="" //read from file

nomoving=0 //count stuck

shootingfps=0//shooting frame count

triggershoot=false //trigger was fight? if triggershoot fps counter +1

oldHP=0

enemyoldHP[0]=0

noshooting=0

inlist=false;

//\*\*\*\*\*\* I'm making statistic arrays

stats[KILLS] = 0;

stats[DEATHS] = 0;

stats[CAPS] = 0;

stats[ASSISTS] = 0;

stats[DESTRUCTION] = 0;

stats[STABS] = 0;

stats[HEALING] = 0;

stats[DEFENSES] = 0;

stats[INVULNS] = 0;

stats[BONUS] = 0;

stats[DOMINATIONS] = 0;

stats[REVENGE] = 0;

stats[POINTS] = 0;

//statistic array for single life/arena

roundStats[KILLS] = 0;

roundStats[DEATHS] = 0;

roundStats[CAPS] = 0;

roundStats[ASSISTS] = 0;

roundStats[DESTRUCTION] = 0;

roundStats[STABS] = 0;

roundStats[HEALING] = 0;

roundStats[DEFENSES] = 0;

roundStats[INVULNS] = 0;

roundStats[BONUS] = 0;

roundStats[DOMINATIONS] = 0;

roundStats[REVENGE] = 0;

roundStats[POINTS] = 0;

timesChangedCapLimit = 0;

lastKnownx=0;

lastKnowny=0;

humiliated=0;

//Arena mode - used by server to check if the player can spawn

canSpawn = 1;

if instance\_exists(ArenaHUD) {

if ArenaHUD.roundStart == 0 canSpawn = 0;

}

//Sentries for Engies

//sentryBuilt = 0;

sentry=-1;

isHaxxyWinner = false;

}

**Destroy Event:**

execute code:

if(object != -1) with(object) instance\_destroy();

with(Rocket) if(ownerPlayer == other.id) instance\_destroy();

with(Flame) if(ownerPlayer == other.id) instance\_destroy();

with(Shot) if(ownerPlayer == other.id) instance\_destroy();

with(Needle) if(ownerPlayer == other.id) instance\_destroy();

with(Sentry) if(ownerPlayer == other.id) instance\_destroy();

with(DeathCam) if(killedby == other.id) instance\_destroy();

**Alarm Event for alarm 0:**

execute code:

canSpawn = 0;

**Alarm Event for alarm 5:**

execute code:

if(global.isHost && global.mapchanging == 0 && team != TEAM\_SPECTATOR && canSpawn == 1) {

var group, spawnpointID, numSpawnPoints;

group = selectSpawnGroup(team);

if (group==-1) {

show\_message("This map does not contain valid spawn points");

}

if(team == TEAM\_RED) {

numSpawnPoints = ds\_list\_size(global.spawnPointsRed[0,group]);

} else {

numSpawnPoints = ds\_list\_size(global.spawnPointsBlue[0,group]);

}

spawnpointID = floor(random(numSpawnPoints));

sendEventSpawn(id, spawnpointID, group);

doEventSpawn(id, spawnpointID, group);

}

**Other Event: User Defined 12:**

execute code:

var subobjects;

if(global.updateType == FULL\_UPDATE) {

write\_ubyte(global.serializeBuffer, stats[KILLS]);

write\_ubyte(global.serializeBuffer, stats[DEATHS]);

write\_ubyte(global.serializeBuffer, stats[CAPS]);

write\_ubyte(global.serializeBuffer, stats[ASSISTS]);

write\_ubyte(global.serializeBuffer, stats[DESTRUCTION]);

write\_ubyte(global.serializeBuffer, stats[STABS]);

write\_ushort(global.serializeBuffer, stats[HEALING]);

write\_ubyte(global.serializeBuffer, stats[DEFENSES]);

write\_ubyte(global.serializeBuffer, stats[INVULNS]);

write\_ubyte(global.serializeBuffer, stats[BONUS]);

write\_ubyte(global.serializeBuffer, stats[POINTS]);

}

subobjects=0;

if(object != -1) subobjects |= $01;

if(sentry != -1) subobjects |= $02;

if(isHaxxyWinner) subobjects |= $04;

write\_ubyte(global.serializeBuffer, subobjects);

if(object != -1) with(object) event\_user(12);

if(sentry != -1) with(sentry) event\_user(12);

**Other Event: User Defined 13:**

execute code:

if(global.updateType == FULL\_UPDATE) {

receiveCompleteMessage(global.serverSocket,12,global.deserializeBuffer);

stats[KILLS] = read\_ubyte(global.deserializeBuffer);

stats[DEATHS] = read\_ubyte(global.deserializeBuffer);

stats[CAPS] = read\_ubyte(global.deserializeBuffer);

stats[ASSISTS] = read\_ubyte(global.deserializeBuffer);

stats[DESTRUCTION] = read\_ubyte(global.deserializeBuffer);

stats[STABS] = read\_ubyte(global.deserializeBuffer);

stats[HEALING] = read\_ushort(global.deserializeBuffer);

stats[DEFENSES] = read\_ubyte(global.deserializeBuffer);

stats[INVULNS] = read\_ubyte(global.deserializeBuffer);

stats[BONUS] = read\_ubyte(global.deserializeBuffer);

stats[POINTS] = read\_ubyte(global.deserializeBuffer);

}

var charObj, subobjects;

receiveCompleteMessage(global.serverSocket,1,global.deserializeBuffer);

subobjects = read\_ubyte(global.deserializeBuffer);

// If the player has a character object on the server

if(subobjects & $01 != 0) {

if(object == -1) {

charObj = getCharacterObject(team, class);

if(charObj != -1) {

object = instance\_create(0,0,charObj);

object.player = id;

object.team = team;

with(object) {

event\_user(0);

}

} else {

show\_message("Invalid player object while deserializing");

}

}

with(object) event\_user(13);

} else {

if(object != -1) with(object) instance\_destroy();

object = -1;

}

// If the player has a sentry object on the server

if(subobjects & $02 != 0) {

if(sentry == -1) {

sentry = instance\_create(0,0,Sentry);

sentry.ownerPlayer=id;

sentry.team=team;

}

with(sentry) event\_user(13);

} else {

if sentry != -1 with sentry instance\_destroy();

sentry = -1;

}

isHaxxyWinner = (subobjects & $04 != 0)

**Script->MyBot->AvoidObstacle**

if(nomoving-1>30)

{

left=0;

jump=0;

right=1;

if(dir==1)

{

if(ds\_map\_exists(movedata2,pos))

{

list=ds\_map\_find\_value(movedata2,pos);

jlhmove=ds\_list\_size(list);

var i;

maxreward=0;

index=-1

for(i=0;i<jlhmove;i+=1)

{

if(real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1))>maxreward)

{

maxreward=real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1));

index=i;

}

}

ds\_list\_replace(ds\_list\_find\_value(list,index),1,0);

}

}

nomoving-=1;

dir=-1;

}

else

{

nomoving=0;

}

**Script->MyBot->Checkfailure**

if(shootreward<=0)

{

if(ds\_map\_exists(aimdata,lastKnownx))

{

tmppx=ds\_map\_find\_value(aimdata,lastKnownx);

if(ds\_map\_exists(tmppx,lastKnowny))

{

tmppy=ds\_map\_find\_value(tmppx,lastKnowny);

if(ds\_map\_exists(tmppy,nearestEnemy.x))

{

tmppex=ds\_map\_find\_value(tmppy,nearestEnemy.x);

if(ds\_map\_exists(tmppex,nearestEnemy.y))

{

tmppey=ds\_map\_find\_value(tmppex,nearestEnemy.y);

var tmpi;

tmpindex=-1;

for(tmpi=0;tmpi<ds\_list\_size(tmppey);tmpi+=1)

{

tmppr=ds\_list\_find\_value(tmppey,tmpi);

if(ds\_list\_find\_value(tmppr,0)>dr)

{

dr=ds\_list\_find\_value(tmppr,0);

tmpindex=tmpi;

}

}

tmppr=ds\_list\_find\_value(tmppey,tmpindex);

dr=ds\_list\_find\_value(tmppr,0);

ds\_list\_replace(tmppr,0,dr-5);

}

}

}

}

if(inlist)

{

show\_debug\_message(name);

show\_debug\_message('failed');

shootreward-=1;

inlist=false;

}

}

**Script->MyBot->CreateMyBot**

mybot = instance\_create(0, 0, MyPlayer)

ds\_list\_add(global.players, mybot)

with mybot

{

MyBotInit()

}

mybot.team = GetMyBotTeam(mybot)

//mybot.class = GetMyBotClass(mybot)

if global.mybotNamePrefix == ""

{

mybot.name = "My Bot "+string(global.mybotNameCounter)

}

else

{

mybot.name = global.mybotNamePrefix+string(global.mybotNameCounter)

}

if(global.mybotNameCounter==1)

{

mybot.class= CLASS\_HEAVY

}

else if(global.mybotNameCounter==2)

{

mybot.class= CLASS\_SOLDIER

}

else

{

mybot.class= CLASS\_SCOUT

}

global.mybotNameCounter += 1

mybot.alarm[5] = 1

ServerPlayerJoin(mybot, global.sendBuffer)

ServerPlayerChangeteam(ds\_list\_size(global.players)-1, mybot.team, global.sendBuffer)

ServerPlayerChangeclass(ds\_list\_size(global.players)-1, mybot.class, global.sendBuffer)

**Script->MyBot->** **GetMyBotInput**

// If you want to make your own bot off this, you need to define these 6 values:

left = 0

right = 0

jump = 0

LMB = 0

RMB = 0

aimDirection = 0

bubbleHP = 0

//BotMain();

//readfile();

//movementleftRL();//Kiri Learn

//movementleftjumpRL();//Kiri Jump Learn

//movementReactiveShoot();//Learn Shooting

//movement();

movementreadplay();

if pressed

{

jump = 0

pressed = 0

}

else if jump

{

pressed = 1

}

keybyte = 0 // keybyte converter.

if left == 1

{

keybyte |= $40

}

if right == 1

{

keybyte |= $20

}

if jump == 1

{

keybyte |= $80

}

if LMB == 1 and humiliated=0

{

keybyte |= $10

}

if RMB == 1 and humiliated=0

{

keybyte |= $08

}

if bubbleHP

{

keybyte |= $04

}

object.keyState = keybyte

object.aimDirection = aimDirection

object.netAimDirection = aimDirection\*65536/360

**Script->MyBot->** **GetMyBotTeam**

// argument0=the bot object

if global.mybotMode == 0

{

argument0.team = TEAM\_RED

}

else

{

argument0.team = TEAM\_BLUE

}

//return argument0.team

**Script->MyBot->Movement**

with GeneratorHUD

{

global.mytime=timer;

}

pos=real(round(lastKnownx));

if(ds\_map\_exists(movedata2,pos))

{

show\_debug\_message(name);

list=ds\_map\_find\_value(movedata2,pos);

jlhmove=ds\_list\_size(list);

var i;

maxreward=0;

for(i=0;i<jlhmove;i+=1)

{

if(real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1))>maxreward)

{

maxreward=real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1));

prefermove=ds\_list\_find\_value(ds\_list\_find\_value(list,i),0);

}

}

show\_debug\_message(prefermove);

}

if(global.mytime>0)

{ if(lastKnownx>2500)

{

left=1;

}

if(prefermove=="L")

{

left=1;

}

if(prefermove=="LJ")

{

left=1;

jump=1;

}

mov=prefermove;

reward=reward-lastKnownx;

if(sqlvalue[0]==0)

{

reward=0;

}

if(reward<-50.0)

{

reward=-10;

}

if(reward>50.0)

{

reward=10;

}

if(reward<=0&& reward>-1)

{

nomoving+=1

}

if (nomoving=15)

{

nomoving+=random(30);

}

if(nomoving<15 && reward >0)

{

dir=1;

nomoving=0

}

if(nomoving>15)

{

AvoidObstacle();

mov="R"

}

totalreward+=reward;

if(sqlvalue[0]!=0)

{

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//reward

}

global.playerx=0;

global.playery=0;

with Player

{

global.playerx=lastKnownx;

global.playery=lastKnowny;

}

sqlvalue[0]+=1;

sqlvalue[sqlvalue[0]]=ds\_list\_create();

ds\_list\_add(sqlvalue[sqlvalue[0]],string(name));//Nama

ds\_list\_add(sqlvalue[sqlvalue[0]],string(class));//Class

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnownx));//Pos\_x

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnowny));//Pos\_y

ds\_list\_add(sqlvalue[sqlvalue[0]],string(object.hp));//HP

ds\_list\_add(sqlvalue[sqlvalue[0]],string(mov));//movement

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//previous reward

reward=lastKnownx;

}

else if(global.mytime==0&&dowrite)

{

var i;

for(i=1;i<=sqlvalue[0];i+=1)

{

ds\_list\_add(sqlvalue[i],string(totalreward));//total reward

currentreward=ds\_list\_find\_value(sqlvalue[i],ds\_list\_size(sqlvalue[i])-2);

totalreward=real(totalreward)-real(currentreward);

}

sql='INSERT INTO `GerakRL`(`Nama`,`Class`,`Pos\_x`,`Pos\_y`,`HP`,`Movement`,`PreviousReward`,`Reward`,`TotalReward`,`GamePlay`) VALUES (';

var j;

for(i=1;i<sqlvalue[0];i+=1)

{

sqlcmd=sql;

for(j=0;j<ds\_list\_size(sqlvalue[i]);j+=1)

{

if(j>0)

{

sqlcmd+=",";

}

sqlcmd+='"'+ds\_list\_find\_value(sqlvalue[i],j)+'"';

}

sqlcmd+=',"'+string(global.mybotgameplay-1)+'"';

sqlcmd+=');';

file\_text\_write\_string(global.sqldump, sqlcmd);

file\_text\_writeln(global.sqldump);

}

dowrite=false;

}

**Script->MyBot->MovementleftRL**

with GeneratorHUD

{

global.mytime=timer;

}

if(global.mytime>0)

{

left=1

reward=reward-lastKnownx;

mov='L';

if(sqlvalue[0]==0)

{

reward=0;

}

if(reward<-50.0)

{

reward=-10;

}

if(reward>50.0)

{

reward=10;

}

totalreward+=reward;

if(sqlvalue[0]!=0)

{

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//reward

}

global.playerx=0;

global.playery=0;

with Player

{

global.playerx=lastKnownx;

global.playery=lastKnowny;

}

sqlvalue[0]+=1;

sqlvalue[sqlvalue[0]]=ds\_list\_create();

ds\_list\_add(sqlvalue[sqlvalue[0]],string(name));//Nama

ds\_list\_add(sqlvalue[sqlvalue[0]],string(class));//Class

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnownx));//Pos\_x

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnowny));//Pos\_y

ds\_list\_add(sqlvalue[sqlvalue[0]],string(object.hp));//HP

ds\_list\_add(sqlvalue[sqlvalue[0]],string(mov));//movement

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//previous reward

reward=lastKnownx;

}

else if(global.mytime==0&&dowrite)

{

var i;

for(i=1;i<=sqlvalue[0];i+=1)

{

ds\_list\_add(sqlvalue[i],string(totalreward));//total reward

currentreward=ds\_list\_find\_value(sqlvalue[i],ds\_list\_size(sqlvalue[i])-2);

totalreward=real(totalreward)-real(currentreward);

}

sql='INSERT INTO `GerakRL`(`Nama`,`Class`,`Pos\_x`,`Pos\_y`,`HP`,`Movement`,`PreviousReward`,`Reward`,`TotalReward`,`GamePlay`) VALUES (';

var j;

for(i=1;i<sqlvalue[0];i+=1)

{

sqlcmd=sql;

for(j=0;j<ds\_list\_size(sqlvalue[i]);j+=1)

{

if(j>0)

{

sqlcmd+=",";

}

sqlcmd+='"'+ds\_list\_find\_value(sqlvalue[i],j)+'"';

}

sqlcmd+=',"'+string(global.mybotgameplay-1)+'"';

sqlcmd+=');';

file\_text\_write\_string(global.sqldump, sqlcmd);

file\_text\_writeln(global.sqldump);

}

dowrite=false;

}

**Script->MyBot->MovementleftjumpRL**

with GeneratorHUD

{

global.mytime=timer;

}

if(global.mytime>0)

{

left=1

reward=reward-lastKnownx;

mov='L';

if(random(40)>20)

{

jump=1

mov+='J'

}

if(sqlvalue[0]==0)

{

reward=0;

}

if(reward<-50.0)

{

reward=-10;

}

if(reward>50.0)

{

reward=10;

}

totalreward+=reward;

if(sqlvalue[0]!=0)

{

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//reward

}

global.playerx=0;

global.playery=0;

with Player

{

global.playerx=lastKnownx;

global.playery=lastKnowny;

}

sqlvalue[0]+=1;

sqlvalue[sqlvalue[0]]=ds\_list\_create();

ds\_list\_add(sqlvalue[sqlvalue[0]],string(name));//Nama

ds\_list\_add(sqlvalue[sqlvalue[0]],string(class));//Class

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnownx));//Pos\_x

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnowny));//Pos\_y

ds\_list\_add(sqlvalue[sqlvalue[0]],string(object.hp));//HP

ds\_list\_add(sqlvalue[sqlvalue[0]],string(mov));//movement

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//previous reward

reward=lastKnownx;

}

else if(global.mytime==0&&dowrite)

{

var i;

for(i=1;i<=sqlvalue[0];i+=1)

{

ds\_list\_add(sqlvalue[i],string(totalreward));//total reward

currentreward=ds\_list\_find\_value(sqlvalue[i],ds\_list\_size(sqlvalue[i])-2);

totalreward=real(totalreward)-real(currentreward);

}

sql='INSERT INTO `GerakRL`(`Nama`,`Class`,`Pos\_x`,`Pos\_y`,`HP`,`Movement`,`PreviousReward`,`Reward`,`TotalReward`,`GamePlay`) VALUES (';

var j;

for(i=1;i<sqlvalue[0];i+=1)

{

sqlcmd=sql;

for(j=0;j<ds\_list\_size(sqlvalue[i]);j+=1)

{

if(j>0)

{

sqlcmd+=",";

}

sqlcmd+='"'+ds\_list\_find\_value(sqlvalue[i],j)+'"';

}

sqlcmd+=',"'+string(global.mybotgameplay-1)+'"';

sqlcmd+=');';

file\_text\_write\_string(global.sqldump, sqlcmd);

file\_text\_writeln(global.sqldump);

}

dowrite=false;

}

**Script->MyBot->MovementReactiveShoot**

BotGetNearestEnemy();

with GeneratorHUD

{

global.mytime=timer;

}

target = Generator

// Aiming and Fighting

if nearestEnemy != -1

{

if(!triggershoot)

{

var p;

p=1;

rwd=0;

Generator.oldhp=Generator.hp;

Generator.oldshield=Generator.shieldHp;

with(BotPlayer)

{

if global.botNamePrefix == ""

{

if(name == "Tempest Bot "+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

}

else

{

other.rwd=oldHP;

oldHP=0;

//other.eoldHP[other.p]=0;

}

}

}

else

{

if(name == global.botNamePrefix+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

//show\_debug\_message(object.hp);

}

}

}

p+=1;

}

}

triggershoot=true

BotAim()

if point\_distance(object.x, object.y, nearestEnemy.x, nearestEnemy.y) < 350

{

if !wasFighting

{

wasFighting = 1

}

if task != 'objective' and target != nearestEnemy and !target\_in\_sight

{

ds\_list\_clear(directionList)

target = nearestEnemy

}

mov="S"

//some random left

if random(50)<5

{

left=1

mov="L"

}

// Some random jumping...

if random(40)<4

{

jump = 1

if(mov=="S")

{

mov="J"

}

else

{

mov+="J"

}

}

switch (class)

{

case CLASS\_SCOUT:

ScoutFight()

break

case CLASS\_SOLDIER:

SoldierFight()

break

case CLASS\_HEAVY:

HeavyFight()

break

case CLASS\_ENGINEER:

EngineerFight()

break

case CLASS\_PYRO:

PyroFight()

break

case CLASS\_SNIPER:

SniperFight()

break

}

}

else if wasFighting

{

if task != 'objective'

{

ds\_list\_clear(directionList)

target = -1

}

wasFighting = 0

}

}

else if wasFighting

{

if task != 'objective'

{

ds\_list\_clear(directionList)

target = -1

}

wasFighting = 0

}

if(triggershoot)

{

shootingfps+=1//shooting frame count

if(rwd>0)

{

shootreward+=rwd;

}

var p;

p=1;

rwd=0;

with(BotPlayer)

{

if(object!=-1)

{

if(object.lastDamageDealer!=-1)

{

//show\_debug\_message(other.name);

if(object.lastDamageDealer.name==other.name)

{

if global.botNamePrefix == ""

{

if(name == "Tempest Bot "+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

//rwd=eoldHP[p]-object.hp;

}

else

{

other.rwd=oldHP;

oldHP=0;

//eoldHP[p]=0;

//rwd=eoldHP[p];

}

}

}

else

{

if(name == global.botNamePrefix+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

//eoldHP[p]=object.hp;

//rwd=eoldHP[p]-object.hp;

}

else

{

other.rwd=oldHP;

oldHP=0;

//eoldHP[p]=0;

//rwd=eoldHP[p];

}

}

}

}

}

}

p+=1;

}

if(rwd>0)

{

shootreward+=rwd;

}

if(Generator.lastDamageDealer!=-1)

{

if(Generator.lastDamageDealer.name==name)

{

if((Generator.oldhp-Generator.hp)>0)

{

shootreward+=2\*(Generator.oldhp-Generator.hp)

}

if((Generator.oldshield-Generator.shieldHp)>0)

{

shootreward+=(Generator.oldshield-Generator.shieldHp);

}

Generator.oldhp=Generator.hp;

Generator.oldshield=Generator.shieldHp;

}

}

if(object.hp-oldHP<0)

{

shootreward+=object.hp-oldHP

}

sqlvalue2[0]+=1;

sqlvalue2[sqlvalue2[0]]=ds\_list\_create();

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(name));//Nama

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(class));//Class

if(object!=-1)

{

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(object.hp));//HP

}

else

{

ds\_list\_add(sqlvalue2[sqlvalue2[0]],"0");//HP

}

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(lastKnownx));//Pos\_x

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(lastKnowny));//Pos\_y

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(mov));//movement

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(aimDirection));//aimradius

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(LMB));//shoot left?

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(RMB));//shoot right?

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(nearestEnemy.x));//enemy pos\_x

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(nearestEnemy.y));//enemy pos\_y

if(shootingfps>15)

{

sql='INSERT INTO `AimingRL`(`Nama`,`Class`,`HP`,`Pos\_x`,`Pos\_y`,`Movement`,`AimRadius`,`LMB`,`RMB`,`enemy\_x`,`enemy\_y`,`Reward`,`GamePlay`) VALUES (';

var i;

var j;

for(i=1;i<sqlvalue2[0];i+=1)

{

sqlcmd=sql;

for(j=0;j<ds\_list\_size(sqlvalue2[i]);j+=1)

{

if(j>0)

{

sqlcmd+=",";

}

sqlcmd+='"'+ds\_list\_find\_value(sqlvalue2[i],j)+'"';

}

sqlcmd+=',"'+string(shootreward)+'"';

sqlcmd+=',"'+string(global.mybotgameplay-1)+'"';

sqlcmd+=');';

file\_text\_write\_string(global.sqldump, sqlcmd);

file\_text\_writeln(global.sqldump);

}

sqlvalue2[0]=0;

shootreward=0;

shootingfps=0;

triggershoot=false;

}

}

oldHP=object.hp

if(wasFighting!=1)

{

pos=real(round(lastKnownx));

if(ds\_map\_exists(movedata2,pos))

{

list=ds\_map\_find\_value(movedata2,pos);

jlhmove=ds\_list\_size(list);

var i;

maxreward=0;

for(i=0;i<jlhmove;i+=1)

{

if(real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1))>maxreward)

{

maxreward=real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1));

prefermove=ds\_list\_find\_value(ds\_list\_find\_value(list,i),0);

}

}

}

if(global.mytime>0)

{

if(prefermove=="L")

{

left=1;

}

if(prefermove=="LJ")

{

left=1;

jump=1;

}

mov=prefermove;

reward=reward-lastKnownx;

if(sqlvalue[0]==0)

{

reward=0;

}

if(dir==-1)

{

reward\*=-1;

reward/=2;

}

if(reward<-50.0)

{

reward=-10;

}

if(reward>50.0)

{

reward=10;

}

if(reward<=0&& reward>-1)

{

nomoving+=1

}

if (nomoving=15)

{

nomoving+=random(30);

}

if(nomoving<15 && reward >0)

{

dir=1;

nomoving=0

}

if(nomoving>15)

{

AvoidObstacle();

mov="R"

}

reward=lastKnownx;

}

}

if reloadCounter > 0

{

reloadCounter -= 1

}

**Script->MyBot->Movementreadplay**

BotGetNearestEnemy();

with GeneratorHUD

{

global.mytime=timer;

}

target = Generator

// Aiming and Fighting

if nearestEnemy != -1

{

if(!triggershoot)

{

var p;

p=1;

rwd=0;

Generator.oldhp=Generator.hp;

Generator.oldshield=Generator.shieldHp;

with(BotPlayer)

{

if global.botNamePrefix == ""

{

if(name == "Tempest Bot "+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

}

else

{

other.rwd=oldHP;

oldHP=0;

}

}

}

else

{

if(name == global.botNamePrefix+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

}

}

}

p+=1;

}

}

triggershoot=true

BotAim()

if point\_distance(object.x, object.y, nearestEnemy.x, nearestEnemy.y) < 350

{

if !wasFighting

{

wasFighting = 1

}

if task != 'objective' and target != nearestEnemy and !target\_in\_sight

{

ds\_list\_clear(directionList)

target = nearestEnemy

}

mov="S"

//some random left

if random(50)<5

{

left=1

mov="L"

}

// Some random jumping...

if random(40)<4

{

jump = 1

if(mov=="S")

{

mov="J"

}

else

{

mov+="J"

}

}

switch (class)

{

case CLASS\_SCOUT:

ScoutFight()

break

case CLASS\_SOLDIER:

SoldierFight()

break

case CLASS\_HEAVY:

HeavyFight()

break

case CLASS\_ENGINEER:

EngineerFight()

break

case CLASS\_PYRO:

PyroFight()

break

case CLASS\_SNIPER:

SniperFight()

break

}

}

else if wasFighting

{

if task != 'objective'

{

ds\_list\_clear(directionList)

target = -1

}

wasFighting = 0

}

}

else if wasFighting

{

if task != 'objective'

{

ds\_list\_clear(directionList)

target = -1

}

wasFighting = 0

}

if(triggershoot)

{

if(LMB==1)

{

noshooting=0;

dx=real(round(lastKnownx));

dy=real(round(lastKnowny));

dex=real(round(nearestEnemy.x));

dey=real(round(nearestEnemy.y));

dr=-1;

dm="";

da=-1;

if(ds\_map\_exists(aimdata,dx))

{

tmppx=ds\_map\_find\_value(aimdata,dx);

if(ds\_map\_exists(tmppx,dy))

{

tmppy=ds\_map\_find\_value(tmppx,dy);

if(ds\_map\_exists(tmppy,dex))

{

tmppex=ds\_map\_find\_value(tmppy,dex);

if(ds\_map\_exists(tmppex,dey))

{

tmppey=ds\_map\_find\_value(tmppex,dey);

var tmpi;

tmpindex=-1;

for(tmpi=0;tmpi<ds\_list\_size(tmppey);tmpi+=1)

{

tmppr=ds\_list\_find\_value(tmppey,tmpi);

if(ds\_list\_find\_value(tmppr,0)>dr)

{

dr=ds\_list\_find\_value(tmppr,0);

tmpindex=tmpi;

}

}

if(shootingfps>15&& shootreward<0)

{

dr2=ds\_list\_find\_value(tmppr,0);

ds\_list\_replace(tmppr,0,dr2-5);

}

tmppr=ds\_list\_find\_value(tmppey,tmpindex);

dm=ds\_list\_find\_value(tmppr,1);

da=ds\_list\_find\_value(tmppr,2);

}

else

{

show\_debug\_message(name);

show\_debug\_message("Not in list");

}

}

else

{

show\_debug\_message(name);

show\_debug\_message("Not in list");

}

}

else

{

show\_debug\_message(name);

show\_debug\_message("Not in list");

}

}

else

{

show\_debug\_message(name);

show\_debug\_message("Not in list");

}

if(dr!=-1)

{

if(dm=="L")

{

left=1;

jump=0;

right=0;

}

if(dm=="LJ")

{

left=1;

jump=1;

right=0;

}

if(dm=="J")

{

left=0;

jump=1;

right=0;

}

if(dm=="S")

{

left=0;

jump=0;

right=0;

}

mov=dm;

aimDirection=da;

inlist=true;

show\_debug\_message("Read");

show\_debug\_message(name);

show\_debug\_message(mov);

show\_debug\_message(aimDirection);

}

}

shootingfps+=1//shooting frame count

if(rwd>0)

{

shootreward+=rwd;

}

var p;

p=1;

rwd=0;

with(BotPlayer)

{

if(object!=-1)

{

if(object.lastDamageDealer!=-1)

{

if(object.lastDamageDealer.name==other.name)

{

if global.botNamePrefix == ""

{

if(name == "Tempest Bot "+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

}

else

{

other.rwd=oldHP;

oldHP=0;

}

}

}

else

{

if(name == global.botNamePrefix+string(p))

{

if(object!=-1)

{

other.rwd=oldHP-object.hp;

oldHP=object.hp;

}

else

{

other.rwd=oldHP;

oldHP=0;

}

}

}

}

}

}

p+=1;

}

if(rwd>0)

{

shootreward+=rwd;

}

if(Generator.lastDamageDealer!=-1)

{

if(Generator.lastDamageDealer.name==name)

{

if((Generator.oldhp-Generator.hp)>0)

{

shootreward+=2\*(Generator.oldhp-Generator.hp)

}

if((Generator.oldshield-Generator.shieldHp)>0)

{

shootreward+=(Generator.oldshield-Generator.shieldHp);

}

Generator.oldhp=Generator.hp;

Generator.oldshield=Generator.shieldHp;

}

}

if(object.hp-oldHP<0)

{

shootreward+=object.hp-oldHP

}

sqlvalue2[0]+=1;

sqlvalue2[sqlvalue2[0]]=ds\_list\_create();

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(name));//Nama

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(class));//Class

if(object!=-1)

{

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(object.hp));//HP

}

else

{

ds\_list\_add(sqlvalue2[sqlvalue2[0]],"0");//HP

}

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(lastKnownx));//Pos\_x

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(lastKnowny));//Pos\_y

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(mov));//movement

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(aimDirection));//aimradius

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(LMB));//shoot left?

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(RMB));//shoot right?

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(nearestEnemy.x));//enemy pos\_x

ds\_list\_add(sqlvalue2[sqlvalue2[0]],string(nearestEnemy.y));//enemy pos\_y

if(shootingfps>15)

{

checkfailure();

sql='INSERT INTO `AimingRL`(`Nama`,`Class`,`HP`,`Pos\_x`,`Pos\_y`,`Movement`,`AimRadius`,`LMB`,`RMB`,`enemy\_x`,`enemy\_y`,`Reward`,`GamePlay`) VALUES (';

var i;

var j;

for(i=1;i<sqlvalue2[0];i+=1)

{

sqlcmd=sql;

for(j=0;j<ds\_list\_size(sqlvalue2[i]);j+=1)

{

if(j>0)

{

sqlcmd+=",";

}

sqlcmd+='"'+ds\_list\_find\_value(sqlvalue2[i],j)+'"';

}

sqlcmd+=',"'+string(shootreward)+'"';

sqlcmd+=',"'+string(global.mybotgameplay-1)+'"';

sqlcmd+=');';

file\_text\_write\_string(global.sqldump, sqlcmd);

file\_text\_writeln(global.sqldump);

}

sqlvalue2[0]=0;

shootreward=0;

shootingfps=0;

triggershoot=false;

}

}

oldHP=object.hp

if(LMB==0)

{

noshooting+=1;

}

if(wasFighting!=1)

{

pos=real(round(lastKnownx));

if(ds\_map\_exists(movedata2,pos))

{

list=ds\_map\_find\_value(movedata2,pos);

jlhmove=ds\_list\_size(list);

var i;

maxreward=0;

for(i=0;i<jlhmove;i+=1)

{

if(real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1))>maxreward)

{

maxreward=real(ds\_list\_find\_value(ds\_list\_find\_value(list,i),1));

prefermove=ds\_list\_find\_value(ds\_list\_find\_value(list,i),0);

}

}

}

if(global.mytime>0)

{ if(noshooting>60&& lastKnownx>2500)

{

left=1;

}

if(prefermove=="L" && noshooting>60)

{

left=1;

}

if(prefermove=="LJ" && noshooting>60)

{

left=1;

jump=1;

}

mov=prefermove;

reward=reward-lastKnownx;

if(sqlvalue[0]==0)

{

reward=0;

}

if(reward<-50.0)

{

reward=-10;

}

if(reward>50.0)

{

reward=10;

}

if(reward<=0&& reward>-1)

{

nomoving+=1

}

if (nomoving=15)

{

nomoving+=random(30);

}

if(nomoving<15 && noshooting>60)

{

dir=1;

nomoving=0

}

if(nomoving>15 &&triggershoot==false)

{

AvoidObstacle();

mov="R"

}

totalreward+=reward;

if(sqlvalue[0]!=0)

{

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//reward

}

global.playerx=0;

global.playery=0;

with Player

{

global.playerx=lastKnownx;

global.playery=lastKnowny;

}

if(triggershoot)

{

nomoving=0;

}

sqlvalue[0]+=1;

sqlvalue[sqlvalue[0]]=ds\_list\_create();

ds\_list\_add(sqlvalue[sqlvalue[0]],string(name));//Nama

ds\_list\_add(sqlvalue[sqlvalue[0]],string(class));//Class

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnownx));//Pos\_x

ds\_list\_add(sqlvalue[sqlvalue[0]],string(lastKnowny));//Pos\_y

ds\_list\_add(sqlvalue[sqlvalue[0]],string(mov));//movement

ds\_list\_add(sqlvalue[sqlvalue[0]],string(reward));//previous reward

reward=lastKnownx;

}

else if(global.mytime==0&&dowrite==false)

{

var i;

for(i=1;i<=sqlvalue[0];i+=1)

{

ds\_list\_add(sqlvalue[i],string(totalreward));//total reward

currentreward=ds\_list\_find\_value(sqlvalue[i],ds\_list\_size(sqlvalue[i])-2);

totalreward=real(totalreward)-real(currentreward);

}

sql='INSERT INTO `GerakRL`(`Nama`,`Class`,`Pos\_x`,`Pos\_y`,`Movement`,`PreviousReward`,`Reward`,`TotalReward`) VALUES (';

var j;

for(i=1;i<sqlvalue[0];i+=1)

{

sqlcmd=sql;

for(j=0;j<ds\_list\_size(sqlvalue[i]);j+=1)

{

if(j>0)

{

sqlcmd+=",";

}

sqlcmd+='"'+ds\_list\_find\_value(sqlvalue[i],j)+'"';

}

sqlcmd+=');';

file\_text\_write\_string(global.sqldump, sqlcmd);

file\_text\_writeln(global.sqldump);

}

dowrite=false;

}

}

if reloadCounter > 0

{

reloadCounter -= 1

}

**Script->MyBot->MyBotInit**

pressed = 0

target = -1

aimModifier = 0

isHealing = 0

patient = -1

if variable\_local\_exists("directionList")

{

ds\_list\_clear(directionList)

}

else

{

directionList = ds\_list\_create()

}

wasFighting = 0

dir = 1

stuckTimer = 0

oldX = 0

oldY = 0

reloadCounter = 0

// Task selecting

task = choose("roam", "roam", "roam", "objective", "hunt")

if class == CLASS\_MEDIC

{

task = 'roam'

}

**Script->MyBot->readlearn**

with MyPlayer

{

movedata2=ds\_map\_create();

num = 0;

file = file\_text\_open\_read(working\_directory + "\move.csv");

while (!file\_text\_eof(file))

{

text=file\_text\_read\_string(file);

splitBy='|';

slot = 0;

bn='';

var split;

str2 = "";

for (i = 1; i < (string\_length(text)+1); i+=1) {

currStr = string\_copy(text, i, 1);

if (currStr == splitBy) {

if(slot==0)

{

if(str2!=name)

{

break;

}

}

if(slot==1)

{

str2=real(str2);

if(!ds\_map\_exists(movedata2,str2))

{

ds\_map\_add(movedata2,str2,ds\_list\_create());

}

bn=str2;

}

else if(slot==2)

{

lst=ds\_map\_find\_value(movedata2,bn);

ds\_list\_add(lst,ds\_list\_create());

listrw=ds\_list\_find\_value(lst,ds\_list\_size(lst)-1);

ds\_list\_add(listrw,str2);

}

else if(slot==3)

{

lst=ds\_map\_find\_value(movedata2,bn);

listrw=ds\_list\_find\_value(lst,ds\_list\_size(lst)-1);

ds\_list\_add(listrw,str2);

}

slot+=1;

str2 = "";

} else {

str2 = str2 + currStr;

}

}

if(slot==3)

{

lst=ds\_map\_find\_value(movedata2,bn);

listrw=ds\_list\_find\_value(lst,ds\_list\_size(lst)-1);

ds\_list\_add(listrw,str2);

}

file\_text\_readln(file);

num+=1;

}

file\_text\_close(file);

}

**Script->MyBot->readlearnshoot**

with MyPlayer

{

aimdata=ds\_map\_create();

num = 0;

file = file\_text\_open\_read(working\_directory + "\aim"+string(name)+".csv");

show\_debug\_message("aim"+string(name)+".csv");

while (!file\_text\_eof(file))

{

text=file\_text\_read\_string(file);

splitBy='|';

slot = 0;

bn='';

var split;

str2 = "";

var px,py,pex,pey,pr,pm,pa;

for (i = 1; i < (string\_length(text)+1); i+=1) {

currStr = string\_copy(text, i, 1);

if (currStr == splitBy) {

if(slot==-1)

{

}

if(slot==0)

{

px=real(str2);

}

else if(slot==1)

{

py=real(str2);

}

else if(slot==2)

{

pex=real(str2);

}

else if(slot==3)

{

pey=real(str2);

}

else if(slot==4)

{

pr=real(str2);

}

else if(slot==5)

{

pm=str2;

}

else if(slot==6)

{

pa=real(str2);

}

slot+=1;

str2 = "";

}

else {

str2 = str2 + currStr;

}

}

file\_text\_readln(file);

if(!ds\_map\_exists(aimdata,px))

{

ds\_map\_add(aimdata,px,ds\_map\_create());

}

tmppx=ds\_map\_find\_value(aimdata,px);

if(!ds\_map\_exists(tmppx,py))

{

ds\_map\_add(tmppx,py,ds\_map\_create());

}

tmppy=ds\_map\_find\_value(tmppx,py);

if(!ds\_map\_exists(tmppy,pex))

{

ds\_map\_add(tmppy,pex,ds\_map\_create());

}

tmppex=ds\_map\_find\_value(tmppy,pex);

if(!ds\_map\_exists(tmppex,pey))

{

ds\_map\_add(tmppex,pey,ds\_list\_create());

}

tmppey=ds\_map\_find\_value(tmppex,pey);

ds\_list\_add(tmppey,ds\_list\_create());

tmppr=ds\_list\_find\_value(tmppey,ds\_list\_size(tmppey)-1);

ds\_list\_add(tmppr,pr);

ds\_list\_add(tmppr,pm);

ds\_list\_add(tmppr,pa);

num+=1;

}

file\_text\_close(file);

}

**Table Aiming**

CREATE TABLE AimingRL (

Nama TEXT,

Class INT,

HP REAL,

pos\_x REAL,

pos\_y REAL,

Movement TEXT,

AimRadius REAL,

LMB INT,

RMB INT,

enemy\_x REAL,

enemy\_y REAL,

Reward REAL,

GamePlay INTEGER

);

**Table Gerak**

CREATE TABLE GerakRL (

Nama TEXT,

Class INT,

Pos\_x REAL,

Pos\_y REAL,

HP REAL,

Movement TEXT,

PreviousReward REAL,

Reward REAL,

TotalReward REAL,

GamePlay INTEGER

);

**Table Normalize Gerak**

CREATE VIEW RLGerakbot AS

SELECT RL2.nama,

printf('%d', RL2.pos\_x) AS pos\_x,

RL2.movement,

RL2.totalreward

FROM (

SELECT round(pos\_x) AS pos\_x,

movement,

max(totalreward) AS totalreward

FROM GerakRL

WHERE reward > 0

GROUP BY nama,

round(pos\_x),

movement

)

RL1

JOIN

(

SELECT round(pos\_x) AS pos\_x,

movement,

nama,

totalreward

FROM GerakRL

WHERE reward > 0

GROUP BY round(pos\_x),

movement,

nama,

totalreward

)

RL2 ON RL1.pos\_x = RL2.pos\_x

WHERE RL1.totalreward = RL2.totalreward AND

RL1.movement = RL2.movement;

**Table Normalisasi Aiming**

CREATE VIEW aimMyBot1 AS

SELECT \*

FROM (

SELECT printf('%d', pos\_x) AS pos\_x,

printf('%d', pos\_y) AS pos\_y,

printf('%d', enemy\_x) AS enemy\_x,

printf('%d', enemy\_y) AS enemy\_y,

avg(Reward) AS reward,

movement,

aimRadius,

count(movement) AS counter

FROM AimingRL

WHERE LMB = 1 AND

nama = nama

GROUP BY nama,

round(pos\_x),

round(pos\_y),

movement,

aimradius,

round(enemy\_x),

round(enemy\_y)

)

WHERE reward > 0

ORDER BY counter DESC;