

**IMPLEMENTASI ALGORITMA FUZZY SUGENO UNTUK
MENGATUR PERGERAKAN QUADCOPTER DALAM
MENGHINDARI OBSTACLE PADA GAME
SIMULASI X-DRONE**

SKRIPSI

Oleh :
ASMARANI PRATAMA Y.HADAD
NIM. 14650071



**JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2018**

**IMPLEMENTASI ALGORITMA FUZZY SUGENO UNTUK MENGATUR
PERGERAKAN QUADCOPTER DALAM MENGHINDARI OBSTACLE
PADA GAME SIMULASI X-DRONE**

SKRIPSI

Diajukan kepada:

**Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN)
Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)**

Oleh :

**ASMARANI PRATAMA Y.HADAD
NIM.14650071**

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2018**

LEMBAR PERSETUJUAN

**IMPLEMENTASI ALGORITMA FUZZY SUGENO UNTUK MENGATUR
PERGERAKAN QUADCOPTER DALAM MENGHINDARI OBSTACLE
PADA GAME SIMULASI X-DRONE**

SKRIPSI

Oleh :

**ASMARANI PRATAMA Y.HADAD
NIM. 14650071**

Telah Diperiksa dan Disetujui untuk Diuji

Tanggal : 03 Mei 2018

Pembimbing I

Yunifa Miftachul Arif, M.T
NIP. 19830616 2001101 1 004

Pembimbing II

Roro Inda Melani, S.Kom., M.Sc
NIP. 19780825 200501 2 008

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Olyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

IMPLEMENTASI ALGORITMA FUZZY SUGENO UNTUK MENGATUR PERGERAKAN QUADCOPTER DALAM MENGHINDARI OBSTACLE PADA GAME SIMULASI X-DRONE

SKRIPSI

Oleh :

ASMARANI PRATAMA Y.HADAD
NIM. 14650071

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal : 31 Mei 2018

Susunan Dewan Penguji

Penguji Utama : Fresy Nugroho, M.T
NIP. 19710722 201101 1 001

Ketua Penguji : Hani Nurhayati, M.T
NIP. 19780625 200801 2 006

Sekretaris Penguji : Yunifa Miftachul Arif, M.T
NIP. 19830616 2001101 1 004

Anggota Penguji : Roro Inda Melani, S.Kom., M.Sc
NIP. 19780825 200501 2 008

Tanda Tangan

()

()

()

()

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang



()
Anhyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini :

Nama : Asmarani Pratama Y.Hadad

NIM : 14650071

Fakultas / Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Skripsi : **IMPLEMENTASI ALGORITMA FUZZY SUGENO
UNTUK MENGATUR PERGERAKAN QUADCOPTER DALAM
MENGHINDARI OBSTACLE PADA GAME SIMULASI X-DRONE**

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-nenar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 2018

Yang membuat pernyataan

METERAI
TEMPEL

86323AFF124854359

6000
ENAM RIBU RUPIAH

Asmarani Pratama Y.Hadad

NIM. 14650071

MOTTO

If you stay, stay forever.

If you change, change for the better,

And if you talk, make sure what you say.

Ingin segera selesai, tapi segitu aja perjuanganmu ?

Segitu aja belajarmu ?

Segitu aja dirimu menghormati guru ?

Segitu aja dirimu melawan kemalasan ?

Segitu aja perjuanganmu dan membiarkan orang lain mengerjakan tugas akhirmu sebagai seorang mahasiswa ?

Segitu aja perjuanganmu ? Hanya karena ingin cepat, jalan pintas kau lalui.

Kadang yang menjadi ujian kita di tugas akhir adalah karma dari hasil ketidakjujuran selama perkuliahan.

Tugas asal buat, belum jadi *copy paste* punya teman. Akhirnya apa ? kita nggak bisa melakukan apa-apa saat menghadapi tugas akhir, berharap ada yang datang mengerjakan tugas ini saja.

Tapi masih ada waktu untuk bertaubat yaitu dengan kembali dengan giat belajar.

Lulus jangan asal lulus, luluslah dengan ridho kedua pembimbingmu dan orang tua mu.

Lulus jangan hanya asal lulus, jangan berakhir hanya bisa jadi tukang ketik di masa yang akan datang, walau tak tau apa yang akan terjadi di masa depan.

“Dari Mahasiswa yang telah menghadapi Tugas Akhir”

HALAMAN PERSEMBAHAN

Alhamdulillah rabbil'alam, segala puja dan puji syukur selalu tercurahkan ke hadirat Allah SWT yang telah menguatkan hati ini dalam menghadapi berbagai problematika skripsi sehingga akhirnya saya dapat menyelesaikan pendidikan S1 di Universitas Islam Negeri Maulana Malik Ibrahim Malang ini. Sholawat serta salam kepada junjungan kita Nabi Muhammad SAW yang telah membawa risalah kebenaran sebagai petunjuk hidup kita di dunia ini.

Terima kasih kepada malaikat tanpa sayap, Mama dan Papa yang tak henti-hentinya selalu memberikan motivasi kalau kamu pasti bisa dengan keterbatasan ilmu yang dimiliki. Selalu berusaha memenuhi kebutuhan yang dibutuhkan, baik dalam bentuk bertukar pikiran, materil, kekuatan, motivasi, dan perasaan yang mereka berikan, kepada tim hore Asharizat terima kasih dukungan kalian.

Terima Kasih kepada teman-teman Biner, Mahasiswa Teladan, Geng Pojokan, Ontaki, Komunitas Teknik Informatika, Ma'had Sunan Ampel Al-A'li, Musyrif-Musyrifah, dll. Yang telah memberikan warna suka maupun duka dalam empat tahun ini dan saling merasakan asam manisnya lingkungan perkuliahan dan kepada yang sering menanyakan “Kapan Sidang, Sempro,dll ?” terima kasih karena selalu mengajak dan mengingatkan untuk segera menyelesaikan skripsi ini. Semoga kita bisa mengamalkan ilmu yang kita dapat selama perkuliahan ini, Terima Kasih.

KATA PENGANTAR

Segala puji bagi Allah SWT atas segala rahmat dan karunia-Nya yang telah diberikan kepada penulis sehingga dapat menyelesaikan skripsi dengan judul “Implementasi Algoritma *Fuzzy Sugeno* untuk mengatur pergerakan *Quadcopter* dalam menghadapi *Obstacle* pada *Game* simulasi X-Drone” dengan baik. Sholawat serta salam selalu tercurahkan kepada junjungan Nabi Muhammada SAW yang membimbing umatnya dari zaman kebodohan menuju Islam yang *rahmatan lilalamin*.

Dalam menyelesaikan skripsi ini, penulis melalui banyak hambatan dengan keterbatasan ilmu yang di miliki, tapi semangat tetap melangkah untuk menyelesaikan skripsi ini, sesuai dengan firman Allah yang artinya “Sesungguhnya Allah tidak akan mengubah keadaan suatu kaum sehingga mereka mengubah keadaan yang ada pada diri mereka sendiri” (Q.S. Ar-Rad : 11). Dalam menyelesaikan skripsi ini, banyak pihak yang telah memberikan bantuan secara motivasi, semangat, maupun meteril, penulis ingin menyampaikan doa dan ucapan terima kasih yang sedalam-dalamnya kepada :

1. Bapak Dr. Cahyo Crysdiyan, selaku ketua jurusan Teknik Informatika yang selalu memotivasi untuk selalu berjuang dan bisa lulus tepat waktu.
2. Bapak Yunifa Miftachul Arif, M.T, selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, mengarahkan dan memberikan masukan kepada penulis dalam pengerjaan ini hingga akhir.
3. Ibu Roro Inda Melani, S.Kom., M.Sc, selaku dosen pembimbing II yang senantiasa memberikan masukan dan nasihat serta petunjuk skripsi ini.

4. Bapak Juniardi Nur Fadhilah, M.T, selaku dosen pembimbing dan penasehat yang selalu memberikan pengarahan serta saran yang benar dalam pengerjaan skripsi ini.
5. Segenap dosen Teknik Informatika yang telah memberikan bimbingan keilmuan kepada penulis selama masa studi.
6. Teman-teman seperjuang Teknik Informatika angkatan 2014 (Biner).
7. Teman-teman komunitas Otomasi dan Robotika UIN Maliki Malang (ONTAKI).
8. Semua pihak yang tidak mungkin penulis sebutkan satu per satu, atas segala yang telah diberikan, penulis ucapkan terima kasih yang sebesar-besarnya.

Berbagai kekurangan dan kesalahan yang mungkin pembaca temukan dalam penulisan skripsi ini. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi kesempurnaan laporan dan kemajuan penulis dikemudian hari. Semoga apa yang menjadi kekurangan bisa disempurnakan oleh peneliti selanjutnya dan semoga karya ini senantiasanya memberi manfaat, Amiiin.

Malang, 08 Juni 2018

Penulis

DAFTAR ISI

HALAMAN JUDUL.....	i
LEMBARAN PERSETUJUAN.....	ii
LEMBARAN PENGESAHAN.....	iii
PERNYATAAN KEASLIAN TULISAN.....	iv
MOTTO.....	v
HALAMAN PERSEMBAHAN.....	vi
KATA PENGANTAR.....	vii
DAFTAR ISI.....	ix
DAFTAR GAMBAR.....	xii
DAFTAR TABEL.....	xv
ABSTRAK.....	xvi
ABSTRACT.....	xvii
ملخص.....	xviii
BAB I. PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Identifikasi Masalah.....	4
1.3 Rumusan Masalah.....	5
1.4 Tujuan.....	5
1.5 Batasan Masalah.....	5
1.6 Manfaat Penelitian.....	6
1.7 Metode Penelitian.....	6
1.8 Sistematika Penelitian.....	8
BAB II. TINJAUAN PUSTAKA.....	10
2.1 Penelitian Terkait.....	10
2.2 <i>Quadcopter</i>	11
2.2.1 Konfigurasi dan Sistem gerak <i>Quadcopter</i>	13
2.2.2 Konfigurasi <i>Remote Control</i> terhadap <i>Quadcopter</i>	15
2.2.3 Sensor Ultrasonik.....	19
2.3 <i>Game</i>	19
2.3.1 Sejarah <i>Game</i>	21
2.3.2 <i>Genre Game</i>	22

2.3.3 Platform <i>Game</i>	24
2.3.4 Elemen <i>Game</i>	24
2.4 <i>Game</i> Simulasi.....	26
2.4.1 Sejarah <i>Game</i> Simulasi.....	27
2.5 <i>Game Engine</i>	28
2.6 Unity 3D.....	30
2.6.1 Sejarah Unity 3D.....	30
2.6.2 Fitur-fitur Unity.....	31
2.7 <i>Fuzzy Logic</i>	35
2.7.1 Fungsi Keanggotaan.....	37
2.7.1 Sistem Inferensi Fuzzy.....	39
BAB III. ANALISIS DAN PERANCANGAN.....	43
3.1 Desain Sistem.....	43
3.1.1 Deskripsi Aplikasi.....	43
3.1.2 <i>Game Play</i>	44
3.1.3 <i>Design Interface</i>	45
3.1.4 Desain Karakter.....	50
3.1.5 <i>Finite State Machine</i>	50
3.1.7 Alur Sistem Otomasi.....	52
3.2 Perancangan Fuzzy.....	53
3.2.1 Variabel Fuzzy.....	54
3.2.2 Nilai Linguistik.....	55
3.2.3 Fuzzyfikasi.....	55
3.2.4 Fuzzy Rule.....	60
3.2.5 Implikasi dan Contoh Perhitungan.....	62
3.2.6 Defuzzyfikasi.....	65
BAB IV. HASIL DAN PEMBAHASAN.....	67
4.1 Implementasi Desain Sistem.....	67
4.1.1 Kebutuhan Perangkat Keras.....	67
4.1.2 Kebutuhan Perangkat Lunak.....	68
4.2 Tampilan Karakter.....	68
4.3 Tampilan <i>Game</i>	69
4.3.1 Tampilan <i>Splash Screen</i>	69

4.3.2	Tampilan <i>Main Menu</i>	70
4.3.3	Tampilan <i>Loading Bar Screen</i>	70
4.3.4	Tampilan <i>Canvas Over Game</i>	71
4.3.5	Tampilan <i>Canvas Win Game</i>	72
4.3.6	Tampilan <i>Awal Game</i>	72
4.3.7	Tampilan <i>Map Game</i>	73
4.4	Implementasi Algoritma <i>Fuzzy Sugeno</i>	74
4.5	Uji Coba.....	80
4.5.1	Uji Coba Sensor.....	81
4.5.2	Uji Coba <i>X-Drone</i> Bergerak.....	82
4.5.3	Uji Coba <i>Rule X-Drone</i>	83
4.5.4	Uji Coba <i>X-Drone</i> Menghindari Halangan.....	97
4.5.5	Uji Coba <i>X-Drone</i> dalam Keadaan Mengurangi Kecepatan.....	100
4.5.6	Uji Coba <i>X-Drone</i> dengan Keadaan <i>Obstacle</i>	103
4.5.7	Uji Coba <i>X-Drone</i> dengan berbagai Arah <i>Obstacle</i>	104
4.5.8	Uji Coba <i>X-Drone</i> dengan jumlah <i>Obstacle</i>	105
4.6	Analisa Hasil Uji Coba.....	107
4.7	Integrasi Sains dengan Islam.....	110
BAB V.	PENUTUP.....	113
5.1	Kesimpulan.....	113
5.2	Saran.....	114
DAFTAR PUSTAKA		115

DAFTAR GAMBAR

Gambar 1.1	Metode Penelitian.....	6
Gambar 2.1	Arah Putaran Baling-baling <i>Quadcopter</i>	14
Gambar 2.2	Perbedaan <i>transmitter</i> mode 1 dan mode 2.....	15
Gambar 2.3	Fungsi <i>throttle</i> untuk Naik ke Atas.....	16
Gambar 2.4	Fungsi <i>throttle</i> untuk Turun ke Bawah.....	16
Gambar 2.5	Fungsi <i>rudder</i> untuk Memutar 360° ke Kanan.....	17
Gambar 2.6	Fungsi <i>rudder</i> untuk Memutar 360° ke Kiri.....	17
Gambar 2.7	Fungsi <i>pitch elevator</i> untuk Maju ke Depan.....	18
Gambar 2.8	Fungsi <i>pitch elevator</i> untuk Mundur ke Belakang.....	18
Gambar 2.9	Fungsi <i>roll</i> ke arah Samping Kanan.....	18
Gambar 2.10	Fungsi <i>roll</i> ke arah Samping Kiri.....	19
Gambar 2.11	Fungsi Kenggotaan Segitiga.....	38
Gambar 2.12	Fungsi Keanggotaan Trapesium.....	38
Gambar 2.13	Fungsi Keanggotaan Kurva Bahu.....	39
Gambar 2.14	Diagram Blok Sistem Inferensi <i>Fuzzy</i>	40
Gambar 2.15	Inferensi dengan menggunakan Metode Tsukamoto.....	41
Gambar 3.1	Diagram Blok.....	44
Gambar 3.2	<i>Remote Control</i> untuk Mode 2.....	45
Gambar 3.3	Rancangan <i>Splash Screen</i>	46
Gambar 3.4	Rancangan <i>Loading Bar Screen</i>	46
Gambar 3.5	Rancangan <i>Main Menu</i>	47
Gambar 3.6	Rancangan <i>Game</i>	47
Gambar 3.7	Rancangan <i>Map Game</i>	48
Gambar 3.8	Rancangan <i>Win Screen</i>	49
Gambar 3.9	Rancangan <i>Game Over Screen</i>	49
Gambar 3.10	Karakter <i>X-Drone</i>	50
Gambar 3.11	<i>FSM Drone off Remote Control</i>	51
Gambar 3.12	<i>FSM Drone on Remote Control</i>	51
Gambar 3.13	Diagram Alur Sistem Otomasi.....	52
Gambar 3.14	Alur Proses <i>Fuzzy</i>	54
Gambar 3.15	Area <i>Box Collider</i> pada <i>Quadcopter</i>	54

Gambar 3.16	<i>Fuzzy Interface System</i>	56
Gambar 3.17	Derajat Himpunan Keanggotaan Sensor Kanan.....	56
Gambar 3.18	Derajat Himpunan Keanggotaan Sensor Tengah.....	57
Gambar 3.19	Derajat Himpunan Keanggotaan Sensor Kiri.....	58
Gambar 4.1	Tampilan <i>X-Drone</i> dari samping.....	68
Gambar 4.2	Tampilan <i>X-Drone</i> dari Atas.....	69
Gambar 4.3	Tampilan <i>Splash Screen</i>	69
Gambar 4.4	Tampilan <i>Main Menu</i>	70
Gambar 4.5	Tampilan <i>Loading Bar Screen</i>	71
Gambar 4.6	Tampilan <i>Game Over</i>	72
Gambar 4.7	Tampilan <i>Win Game</i>	72
Gambar 4.8	Tampilan <i>Awal Game</i>	73
Gambar 4.9	Tampilan <i>Map Game</i>	73
Gambar 4.10	Pengambilan data sensor jarak.....	74
Gambar 4.11	Gambar <i>Fuzzyfikasi dan Implikasi</i>	75
Gambar 4.12	Implementasi <i>Fuzzy Rule</i>	77
Gambar 4.13	<i>Source Code</i> Bergerak <i>X-Drone</i>	77
Gambar 4.14	<i>Source Code</i> Mengurangi Kecepatan pada <i>X-Drone</i>	78
Gambar 4.15	<i>Source Code</i> Geser Kanan.....	79
Gambar 4.16	<i>Source Code</i> Geser Kiri.....	79
Gambar 4.17	<i>Source Code</i> <i>X-Drone</i> Diam.....	80
Gambar 4.18	Tampilan Deteksi Sensor <i>X-Drone</i> ke <i>Obstacle</i>	81
Gambar 4.19	Hasil Uji coba Sensor Jarak.....	82
Gambar 4.20	Gambar <i>X-Drone</i> saat Diam.....	83
Gambar 4.21	Gambar <i>X-Drone</i> saat bergerak kesamping kanan.....	83
Gambar 4.22	Gambar Percobaan <i>Rule</i> Jauh – Jauh – Jauh	84
Gambar 4.23	Gambar Percobaan <i>Rule</i> Jauh – Jauh – Sedang.....	84
Gambar 4.24	Gambar Percobaan <i>Rule</i> Jauh – Jauh – Dekat.....	85
Gambar 4.25	Gambar Percobaan <i>Rule</i> Jauh – Sedang – Jauh.....	85
Gambar 4.26	Gambar Percobaan <i>Rule</i> Jauh – Sedang – Sedang.....	86
Gambar 4.27	Gambar Percobaan <i>Rule</i> Jauh – Sedang – Dekat.....	86
Gambar 4.28	Gambar Percobaan <i>Rule</i> Jauh – Dekat – Jauh.....	87
Gambar 4.29	Gambar Percobaan <i>Rule</i> Jauh – Dekat – Sedang.....	87

Gambar 4.30	Gambar Percobaan <i>Rule</i> Jauh – Dekat – Dekat.....	88
Gambar 4.31	Gambar Percobaan <i>Rule</i> Sedang – Jauh – Jauh.....	88
Gambar 4.32	Gambar Percobaan <i>Rule</i> Sedang – Jauh – Sedang.....	89
Gambar 4.33	Gambar Percobaan <i>Rule</i> Sedang – Jauh – Dekat.....	89
Gambar 4.34	Gambar Percobaan <i>Rule</i> Sedang – Sedang – Jauh.....	90
Gambar 4.35	Gambar Percobaan <i>Rule</i> Sedang – Sedang – Sedang.....	90
Gambar 4.36	Gambar Percobaan <i>Rule</i> Sedang – Sedang – Dekat.....	91
Gambar 4.37	Gambar Percobaan <i>Rule</i> Sedang – Dekat – Jauh.....	91
Gambar 4.38	Gambar Percobaan <i>Rule</i> Sedang – Dekat – Sedang.....	92
Gambar 4.39	Gambar Percobaan <i>Rule</i> Sedang – Dekat – Dekat.....	92
Gambar 4.40	Gambar Percobaan <i>Rule</i> Dekat – Jauh – Jauh.....	93
Gambar 4.41	Gambar Percobaan <i>Rule</i> Dekat – Jauh – Sedang.....	93
Gambar 4.42	Gambar Percobaan <i>Rule</i> Dekat – Jauh – Dekat.....	94
Gambar 4.43	Gambar Percobaan <i>Rule</i> Dekat – Sedang – Jauh.....	94
Gambar 4.44	Gambar Percobaan <i>Rule</i> Dekat – Sedang – Sedang.....	95
Gambar 4.45	Gambar Percobaan <i>Rule</i> Dekat – Sedang – Dekat.....	95
Gambar 4.46	Gambar Percobaan <i>Rule</i> Dekat – Dekat – Jauh.....	96
Gambar 4.47	Gambar Percobaan <i>Rule</i> Dekat – Dekat – Sedang.....	96
Gambar 4.48	Gambar Percobaan <i>Rule</i> Dekat – Dekat – Dekat.....	97
Gambar 4.49	Tampilan Menara kembar dari sisi <i>X-Drone</i>	98
Gambar 4.50	Tampilan <i>X-Drone</i> saat menghindari <i>Obstacle</i>	98
Gambar 4.51	Tampilan jarak dekat antara <i>X-Drone</i> dengan <i>Obstacle</i>	99
Gambar 4.52	Nilai Sensor Jarak dan Aksi <i>Fuzzy</i>	99
Gambar 4.53	Tampilan jarak sedang antara <i>X-Drone</i> dengan <i>Obstacle</i>	101
Gambar 4.54	Aksi dan Nilai pengurangan Kecepatan <i>X-Drone</i>	101
Gambar 4.55	Tampilan jarak dekat antara <i>X-Drone</i> dengan <i>Obstacle</i>	102
Gambar 4.56	Aksi diam pada <i>X-Drone</i>	103
Gambar 4.57	Arah-arrah keberadaan <i>Obstacle</i>	104
Gambar 4.58	Tampilan peningkatan nilai fps pada <i>fuzzy rule</i>	109

DAFTAR TABEL

Tabel 2.1	Formulasi Fungsi Keanggotaan Trapesium.....	39
Tabel 3.1	Derajat Himpunan Keanggotaan Sensor Kanan.....	57
Tabel 3.2	Derajat Himpunan Keanggotaan Sensor Tengah.....	58
Tabel 3.3	Derajat Himpunan Keanggotaan Sensor Kiri.....	59
Tabel 4.1	Kebutuhan Perangkat Keras.....	67
Tabel 4.2	Kebutuhan Perangkat Lunak.....	68
Tabel 4.3	Uji Coba Pergerakan <i>X-Drone</i> dengan Keadaan <i>Obstacle</i>	103
Tabel 4.4	Uji Coba Pergerakan <i>X-Drone</i> dengan berbagai arah <i>Obstacle</i> ..	104
Tabel 4.5	Uji Coba <i>X-Drone</i> dengan Jumlah <i>Obstacle</i>	105
Tabel 4.6	Hasil Uji Coba Pergerakan <i>X-Drone</i>	107

ABSTRAK

Pratama Y.Hadad, Asmarani. 2016. **Implementasi Algoritma Fuzzy Sugeno untuk Mengatur Pergerakan Quadcopter dalam Menghindari Obstacle pada Game Simulasi X-Drone**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing : (I) Yunifa Miftachul Arif, M.T, (II) Roro Inda Melani, S.Kom., M.Sc

Kata Kunci : *Game Simulasi, Quadcopter, Obstacle, Fuzzy Sugeno*

Quadcopter atau yang sering disebut *drone*, memiliki banyak keistimewaan dalam dunia transportasi misalnya digunakan untuk pos dalam mengantarkan paket ataupun surat. *Quadcopter* menyerupai pesawat yang dikontrol melalui kendali jarak jauh atau *autopilot* ini mulai populer digunakan secara umum dan luas seperti fotografi atau videografi. Akan tetapi sering terjadinya kecelakaan yang disebabkan oleh *quadcopter* karena tidak terdeteksinya *obstacle* atau halangan. *Quadcopter* di simulasikan pada *Game Simulasi X-Drone* yang menggunakan algoritma *fuzzy sugeno* yang menentukan aksi pergerakan *quadcopter* saat terdeteksinya *obstacle*.

Aksi yang diberikan pada *X-Drone* selaku *Quadcopter* dalam *game* simulasi adalah bergerak, geser kanan, geser kiri, kurangi kecepatan, dan diam dengan menggunakan metode *fuzzy sugeno*. Dari hasil percobaan yang dilakukan menunjukkan akurasi metode *fuzzy sugeno* dalam menentukan pergerakan *quadcopter* dalam menghindari *obstacle* adalah 96.3%, dari hasil tersebut menunjukkan bahwa metode tersebut bagus dan tepat untuk digunakan pada pergerakan *X-Drone* dalam menghindari *obstacle*.

ABSTRACT

Pratama Y.Hadad, Asmarani. 2016. **Implementation of Fuzzy Sugeno Algorithm to Manage Quadcopter Movement in Avoiding Obstacle on X-Drone Simulation Game.** Undergraduate Thesis. Informatics Engineering Department, Faculty of Science and Technology, State Islamic University of Maulana Malik Ibrahim Malang.

Advisers : (I) Yunifa Miftachul Arif, M.T, (II) Roro Inda Melani, S.Kom., M.Sc

Keywords : Game Simulation, Quadcopter, Obstacle, Fuzzy Sugeno

The Quadcopter or often called drone has many privileges in the world of transportation for example used for the post in delivering the package or letter. The Quadcopter resembles a plane that is controlled via remote control or autopilot is becoming popularly used in general and broad such as photography or videography. However, frequent occurrences of accidents caused by the Quadcopter because of no detection of obstacles. The Quadcopter is simulated on the X-Drone Simulation Game which uses the Fuzzy Sugeno algorithm that determines the action of quadcopter movement when detected obstacle.

Action given on X-Drone as Quadcopter in simulation game is moved, right slide, left shear, reduce speed, and silence or can't move to forward by using the Fuzzy Sugeno method. From the experimental results, the accuracy of The Fuzzy Sugeno method in determining quadcopter movement in avoiding the obstacle is 96.3%, the result shows that the method is good and proper for use on X-Drone movement in avoiding an Obstacle.

ملخص

فيراتما، أسماراني. التنفيذ خوارزميه ضبابية سوجنو لتنظيم حركه المروحية الرباعية في تجنب العقبات علي لعبه محاكاة-X بدون طيار. بحث العلم . التقنية المعلوماتية كلية العلوم والتكنولوجيا في جامعه مولانا مالك إبراهيم الاسلاميه الحكومية مالانغ.

المشرفة : (1) يونيفا مفتاح العارف، الماجستير. (2) رورو عند ميلاني، الماجستير.

الكلمة : لعبه المحاكاة (game simulasi)، المروحية التربيعية (Quadcopter)، عقبة (obstacle)، ضبابي سوجنو (fuzzy sugeno).

المروحية التربيعية أو غالبا ما تسمى باسم طيار، والعديد من الامتيازات في عالم النقل علي سبيل المثال تستخدم لأخر في تقديم حزم أو الرسائل. المروحية التربيعية تشبه الطائرة التي تسيطر عليها عن طريق جهاز التحكم عن بعد أو الطيار الألى بدأت تكون شعبية وفسحة مثل التصوير الفوتوغرافي أو الفيديو. ولكن في كثير من الأحيان الحوادث الناجمة عن عقبه بسبب وجود المروحية التربيعية لأن عدم الاكتشاف العقبة أو عرقله. المروحية الرباعية في محاكاة ألعاب التي تستخدم الاشعه السينية طائرات بدون طيار-X- (Drone) التي خوارزميه ضبابية سوجنو يحدد العمل عندما الحركة عقبه الكشف العقبة.

العمل يعطي على طائرات بدون طيار-X كما المروحية التربيعية في اللعبة المحاكاة هو الانتقال ، انزلاق ، انزلاق اليسار اليمين ، والحد من السرعة ، والصمت باستخدام طريقه ضبابية سوجنو. من نتائج التجارب التي أجريت يدل علي دقه طريقه ضبابي سوجنو في تحديد الحركة الرباعية في تجنب العقبة هو 96.3 % ، من تلك النتائج تبين أن طريقة جيده ومناسبه لاستخدامها علي حركه الطائرة بدون طيار في تجنب العقبات.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam perkembangan transportasi pada zaman sekarang telah mengalami perkembangan pesat. Transportasi digunakan untuk memudahkan manusia dalam melakukan aktivitas sehari-hari, seperti mengantar barang yang kuantitasnya banyak dan berat sehingga tidak memungkinkan manusia untuk mengangkatnya. Berkembangnya transportasi telah dijelaskan dalam Al-Qur'an yang berbunyi :

وَحْمِلُ أَثْقَالِكُمْ إِلَىٰ بَلَدٍ لَّمْ تَكُونُوا بَالِغِيهِ إِلَّا بِشِقِّ الْأَنْفُسِ ۚ إِنَّ رَبَّكُمْ لَرَّءُوفٌ
رَّحِيمٌ ﴿٧﴾ وَالْخَيْلَ وَالْبِغَالَ وَالْحَمِيرَ لِتَرْكَبُوهَا وَزِينَةً وَيَخْلُقُ مَا لَا تَعْلَمُونَ ﴿٨﴾

Artinya :

“7. Dan ia memikul beban-bebanmu ke suatu negeri yang kamu tidak sanggup sampai kepadanya, melainkan dengan kesukaran-kesukaran (yang memayahkan) diri. Sesungguhnya Tuhanmu benar-benar Maha Pengasih lagi Maha Penyayang, 8. dan (dia telah menciptakan) kuda, bagal dan keledai, agar kamu menungganginya dan (menjadikannya) perhiasan. dan Allah menciptakan apa yang kamu tidak mengetahuinya.” (Q.S. An-Nahl: 7-8).

Dalam Al-Qur'an surah An-Nahl ayat 7-8 telah dijelaskan betapa Allah mempermudah manusia dalam membawa beban-beban yang berat dengan menggunakan kendaraan, dan pada ayat “Allah menciptakan apa yang kamu tidak mengetahuinya” inilah Allah menunjukkan kebesaran-Nya yang jauh dari nalar seorang manusia, siapa yang tahu pada perkembangan transportasi saat ini yang

dulunya hanyalah menggunakan hewan seperti unta dan lainnya, sekarang bisa menerbangkan burung besi di udara dan bisa melayarkan kapal yang begitu besar untuk berlayar menuju ke penjuru dunia.

Dalam mempermudah manusia dalam berbagai bidang diciptakanlah *quadcopter* atau yang sering disebut *drone* yang memiliki banyak keistimewaan bahkan dalam dunia transportasi misalnya digunakan untuk pos dalam mengantarkan paket ataupun surat. *Quadcopter* merupakan mesin terbang tanpa awak menyerupai pesawat yang dikontrol melalui kendali jarak jauh atau *autopilot* saat ini mulai populer untuk pengguna secara umum dan luas. Teknologi *multi-tasking* yang awalnya digunakan hanya untuk kepentingan militer, kini mulai dilirik untuk fungsi-fungsi lain. Fungsi istimewa yang dimiliki *drone* menyebabkan penggunaannya mulai melebar ke dunia bidang fotografi udara, jurnalisme, hingga riset. Untuk kelas fotografi udara sudah sangat beragam jenisnya, mulai pengambilan video dan foto, *monitoring*, hingga penggunaan untuk Sistem Informasi Geografis atau GIS (Anshori, 2016).

Dengan segala keistimewaan *quadcopter* tersebut mengakibatkan banyaknya peminat yang tertarik untuk menjadi *pilot quadcopter*, akan tetapi banyak dari para pengemudi pemula mengurungkan niatnya untuk mempelajari untuk mengemudi drone, karena terhalang biaya dan semakin kompleks komponen yang dimiliki oleh *quadcopter* maka semakin mahal harganya. Bahkan jika telah memiliki *drone* juga terkadang, pengemudi masih takut untuk mengendalikannya karena takut jatuh akibat kehilangan frekuensi pada ketinggian tertentu ataupun menabrak benda-benda sekeliling.

Menurut data DJI, salah satu produser terbesar *quadcopter* mengatakan bahwa dalam lima tahun terakhir angka rata-rata penjualan *drone* terus meningkat di kisaran 68 persen. Hingga perkiraan tahun 2019, tercatat pengiriman barang sebanyak 3 juta unit. Jumlah ini meningkat drastis dibanding tahun 2015, dengan jumlah pengiriman sebanyak 40 ribu unit. Lebih dari 120 ribu unit *drone* telah terdaftar di Cina. Sementara di Amerika Serikat, jumlah *drone* yang telah didaftarkan baru mencapai 77 ribu unit.

Quadcopter merupakan pesawat tanpa awak bukanlah mainan yang bisa diterbangkan dengan bebas, untuk itu pengemudi yang akan menerbangkan *quadcopter* terikat peraturan undang-undang yang dikeluarkan oleh Menteri Perhubungan Republik Indonesia yang dimana berdasarkan peraturan Menteri Perhubungan Republik Indonesia NO PM 90 Tahun 2015 tentang pengendalian pengoperasian pesawat udara tanpa awak di ruang udara yang dilayani Indonesia, ada beberapa peraturan yang semestinya tidak boleh dilanggar oleh pengemudi pesawat udara tanpa awak atau *quadcopter*. Misalnya pengemudi tidak boleh menerbangkan pesawat tanpa awak melebihi dari 500 ft atau 150 meter, dan tidak boleh menerbangkan pesawat udara tanpa awak ke kawasan udara yang terbatas ataupun kawasan udara yang terlarang.

Sering terjadinya kecelakaan yang disebabkan oleh *quadcopter* tidak mendeteksinya *obstacle* atau halangan, salah satu kasus kecelakaan yang terjadi di bandara Heatrow London yang melibatkan pesawat komersial jenis Airbus A320 ditabrak oleh *quadcopter* dan menyebabkan salah satu sayap pesawat patah. Kemudian ketidakpandaian pengemudi dalam mengendalikan *quadcopter* juga

salah satu penyebab kecelakaan *quadcopter*, hal ini disebabkan pengemudi tidak memahami fungsi-fungsi dari *remote control* terhadap *quadcopter* dan tidak terdeteksi *obstacle* dari *quadcopter*.

Dari permasalahan tersebut, akan diselesaikan menggunakan simulasi yang akan diterapkan pada sebuah proses mengemudi *quadcopter* yang dinyatakan dalam sebuah *game*, dimana *game* simulasi akan meniru proses mengemudi *quadcopter* sehingga pemain juga tertantang untuk memainkan *game* simulasi sekaligus belajar mengemudikan *quadcopter* berdasarkan fungsi-fungsi *remote control* yang digunakan oleh pemain untuk mengendalikan *quadcopter*.

Untuk itu, dalam pembuatan *game* simulasi *quadcopter* ini diperlukan metode agar dapat mendeteksi jarak dari *quadcopter* ke *obstacle* sehingga memberikan aksi sesuai dengan jarak yang terdeteksi, dan metode yang dipilih adalah metode *fuzzy sugeno*, hal ini karena metode *fuzzy sugeno* yang merupakan cabang dari *fuzzy logic* memberikan nilai output yang lebih konstan dibandingkan cabang lain, sehingga nilai ini menentukan pergerakan *quadcopter* secara konstan sesuai dengan jaraknya (Ahmad Ainun Najib, 2017).

1.2 Identifikasi Masalah

Berdasarkan latar belakang di atas, didapatkan bahwa penelitian ini dilaksanakan karena ketidakpandaian pengemudi atau *pilot* pemula *drone* dalam mengendalikan *drone* sehingga penelitian dilakukan untuk memberikan simulasi mengendalikan *drone* dalam bentuk *game* pada *pilot drone* pemula sehingga dapat mempelajari fungsi-fungsi pada *remote control*, dan dalam menentukan jarak dari *quadcopter* ke *obstacle* dengan mensimulasikan sensor ultrasonik terdapat

memberikan aksi yang berbeda terhadap *quadcopter* berdasarkan jarak dengan menggunakan metode *fuzzy sugeno*.

1.3 Rumusan Masalah

Berdasarkan identifikasi masalah, maka dapat dirumuskan dengan pertanyaan berikut :

1. Seberapa akurat penggunaan metode *fuzzy sugeno* dalam mengatur pergerakan *quadcopter* berdasarkan jarak pada *game* simulasi *X-Drone*?

1.4 Tujuan

Berdasarkan rumusan masalah, maka tujuan dari penelitian ini adalah :

1. Untuk membangun *game* simulasi *X-Drone*.
2. Untuk mengukur keakuratan penggunaan algoritma *fuzzy sugeno* dalam mengatur pergerakan *quadcopter* berdasarkan jarak pada *game* simulasi *X-Drone*.

1.5 Batasan Masalah

Untuk menghindari meluasnya permasalahan yang ada, serta keterbatasan ilmu dan kemampuan yang dimiliki oleh penulis maka:

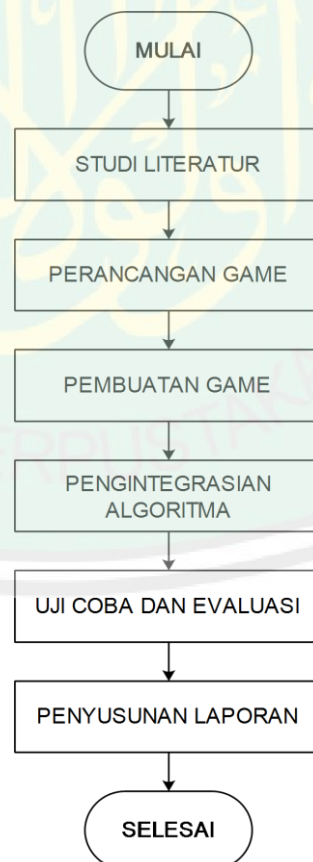
1. *Game* ini dimainkan oleh *Single Player*.
2. *Game* berbasis 3D untuk *PC*.
3. *Platform* yang digunakan adalah *Windows*.
4. Implementasi *Fuzzy Sugeno* pada pergerakan *quadcopter*.
5. Sensor hanya bisa mendeteksi *obstacle* yang berada di depan atau yang berada diradius sudut 30° dengan 13 *raycast* dengan perbedaan *angle* 2.5° .

1.6 Manfaat Penelitian

Adapun manfaat yang dapat diperoleh pada penelitian ini adalah membantu para pengemudi *quadcopter* untuk mengendalikan *quadcopter* dengan baik dan benar. Dengan adanya *game* ini berharap dapat membantu mengurangi kecelakaan *quadcopter* karena ketidakpandaian pengemudi dalam mengendalikan *quadcopter*.

1.7 Metode Penelitian

Menurut Sugiyono (2013:2) metode penelitian merupakan cara ilmiah untuk mendapatkan data dengan tujuan dan kegunaan tertentu. Peneliti membagi pengerjaan penelitian ini dalam beberapa tahap, antara lain :



Gambar 1.1. Metode Penelitian

Pada Gambar 1.1 merupakan metode penelitian yang akan digunakan penulis dalam mengerjakan penelitian, dibawah ini merupakan penjelasan masing-masing tahap.

1. Studi Literatur

Pada tahap ini akan dilakukan pengumpulan data dari literatur-literatur yang terkait penelitian ini. Literatur didapatkan dari buku, jurnal, atau skripsi yang sudah ada. Literatur berisi informasi pembuatan *game* tentang penerapan algoritma *fuzzy sugeno* yang diterapkan pada *game*.

2. Perancangan *Game*

Pada tahap ini dilakukan analisis dari hasil pengumpulan data dari literature-literatur terkait dengan penelitian ini. Setelah itu akan dilakukan perancangan skenario *game*, seperti desain *game*, desain *interface*, dan juga perancangan proses-proses utama dari *game* itu sendiri.

3. Pembuatan *Game*

Pada tahap ini, dilakukan proses pembangunan *game* dengan desain *interface*, desain karakter, integrasi dengan penerapan algoritma *fuzzy sugeno* dan juga menyusun proses-proses utama dalam *game* sehingga menghasilkan *game* yang sesuai dengan hasil yang memenuhi kriteria.

4. Pengintegrasian Algoritma

Pada tahap ini dilakukan pengintegrasian antara *game* simulasi *quadcopter* dengan algoritma *fuzzy sugeno* yang digunakan untuk memberikan aksi pada *quadcopter* berdasarkan input *remote control* melalui *keyboard*.

5. Uji Coba dan Evaluasi

Pada tahap ini dilakukan proses uji coba untuk mengetahui hasil dari pembuatan *game* simulasi *quadcopter* yang telah dibuat dan mengetahui kelancaran saat memainkan *game* tersebut.

6. Penyusunan Laporan

Pada tahap ini merupakan tahap terakhir dimana dalam pembuatan laporan ini bisa bermanfaat bagi penelitian-penelitian selanjutnya yang mana penelitian ini berisi mengenai hasil seluruh dokumentasi dari keseluruhan pelaksanaan penelitian

1.8 Sistematika Penulisan

Penulisan skripsi ini tersusun dalam 5 (lima) bab dengan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN

Bab ini berisi tentang penjelasan penulis menyusun laporan penelitian ini berserta manfaat dan tujuan penelitian.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan teori dasar dan data-data yang berhubungan dengan pembuatan *game* dan penerapan algoritma

BAB III ANALISIS DAN PERANCANGAN

Bab ini berisi desain rancangan *game* dan rancangan aplikasi maupun implementasi algoritma *fuzzy sugeno* pada pemberian pergerakan pada *quadcopter*.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil implementasi algoritma *fuzzy sugeno* terhadap *game* simulasi *quadcopter* beserta uji coba pada *game*.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan secara keseluruhan pada penelitian pembuatan *game* simulasi *quadcopter* dan penerapan metode *fuzzy sugeno*.



BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Pada *paper* yang berkaitan dengan *game* simulasi yakni penelitian yang berjudul “Pembuatan Aplikasi Simulasi Ujian Praktik Pengambilan Surat Izin Mengemudi Kendaraan Roda Empat” yang diteliti oleh (Kurniadi, Yohan, dkk) yang menjelaskan bahwa pada aplikasi ini terbuat dari 3D yang dapat mensimulasikan ujian praktik untuk kendaraan roda empat. *Game* yang berbasis *immersive tools* ini yang menggunakan control dari *Logitech Steering Wheel G27* yang dimana mengeluarkan hasil output sesuai dengan control kemudi pada mobil, selain berbasis *immersive tools*, *game* ini juga berbasis *virtual reality* yang *real time*.

Kemudian juga terdapat skripsi oleh (Sukma, Ridhoi Isham, 2017) pada Universitas Islam Negeri Maulana Malik Ibrahim Malang yang berjudul tentang “Identifikasi Rute Kebakaran Hutan dengan Menggunakan Algoritma A* (A Star) berbasis *Game Simulasi Drone (Quadcopter)* yang menjelaskan dalam menentukan dan mensimulasikan bagaimana kerusakan hutan akibat kebakaran bisa ditindak lanjuti dengan menggunakan *quadcopter* sehingga *quadcopter* dapat menemukan koordinat rute terdekat untuk ke titik koordinat kebakaran.

Penelitian dalam menentukan jarak sensor ultrasonik yang dipaparkan oleh (Rachman, Fathur Zaini, dkk) dalam penelitiannya yang berjudul “Robot Penjejak Ruang dan Kendali Ganda melalui *Bluetooth*” menjelaskan bahwa penggunaan sensor ultrasonik untuk menghindari halangan, maka robot akan bergerak secara

otomatis ke arah lain tanpa menabrak penghalang tersebut. Dan dalam pengujian keakuratannya pada sensor ultrasonik maka didapatkan pada jarak 2-300 cm jika sensor mendeteksi kurang dan lebih dari jarak tersebut maka sensor ultrasonik tidak dapat mendeteksi.

Dalam menghindari *obstacle* atau halangan yang berada didepan *quadcopter*, maka terdapat referensi yang diteliti oleh (Najib, Ahmad Ainun, 2017) yang berjudul “Robot Beroda untuk Menghindari Hambatan Menggunakan *Fuzzy Logic*” yang menjelaskan bahwa *fuzzy logic* yang digunakan untuk menentukan tingkah laku robot beroda. Dari hasil percobaan yang dilakukan akurasi metode *fuzzy logic* untuk menentukan tingkah laku Robot Beroda adalah 92.5%, dan didapatkan hasil bawah metode tersebut cukup bagus dan tepat untuk digunakan pada robot beroda.

Dalam perhitungan *fuzzy sugeno* maka terdapat referensi skripsi yang ditulis oleh (Fauzan, Muhammad. 2016) yang berjudul “Implementasi Algoritma *Fuzzy State Machine* untuk Perilaku *Non Playable Character* pada Game *First Person Shooter ‘Patriosm Young’*”, yang menggunakan batasan masalah terhadap *fuzzy sugeno* dimana metode ini akan mengatur tindakan *NPC*, dengan menggunakan metode *fuzzy segeno* sehingga *NPC* dapat merespon tindakan dari pemain.

2.2 Quadcopter

Quadcopter atau yang sering dikenal dengan nama *drone* adalah salah satu jenis wahana tanpa awak yang memiliki empat motor yang dilengkapi dengan empat *propeller* pada masing-masing motornya yang digunakan untuk terbang

dan bermanuver. Sedangkan pengertian *Quadcopter* menurut para ahli yakni, menurut (Hansson, 2010) adalah pesawat tanpa awak yang memiliki empat buah baling-baling sebagai penggerakannya, dan menurut (Brescieni, 2008) *quadcopter* adalah salah satu *platform unmanned aerial vehicle* (UAV) yang saat ini banyak diriset karena kemampuannya melakukan *take-off* dan *landing* secara vertikal dengan menggunakan empat baling-baling.

Quadcopter memiliki kelebihan mampu terbang ke segala arah, mengudara tanpa landasan panjang, serta bergerak secara 3 sumbu derajat kebebasan. Beberapa orang menggunakan *quadcopter* sendiri ditujukan untuk berbagai fungsi misalnya fotografi udara, pantauan video dari atas yang biasa digunakan saat pengevakasian bencana alam, dan juga untuk pemantauan. *Quadcopter* dapat dioperasikan tanpa menggunakan awak atau pilotnya, pengemudi *quadcopter* tetap berada di daratan dan mengendalikan *quadcopter* lewat fasilitas *remote control*. Hal inilah yang menyebabkan *drone* sering digunakan oleh pihak militer terutama untuk misi-misi yang memiliki resiko besar bagi pesawat yang dioperasikan oleh *pilot* (Luukonen, 2011).

Quadcopter juga dapat digunakan untuk membawa barang maupun mengirim paket dalam jumlah beban yang relative ringan, penggunaan *quadcopter* ini telah diterapkan di beberapa perusahaan di Amerika Serikat. Selain digunakan untuk mengirim barang, *quadcopter* juga sering digunakan dan dilibatkan dalam proses pembuatan film untuk mengambil gambar maupun video dengan menggunakan kamera dari ketinggian yang dipasang pada *body quadcopter*.

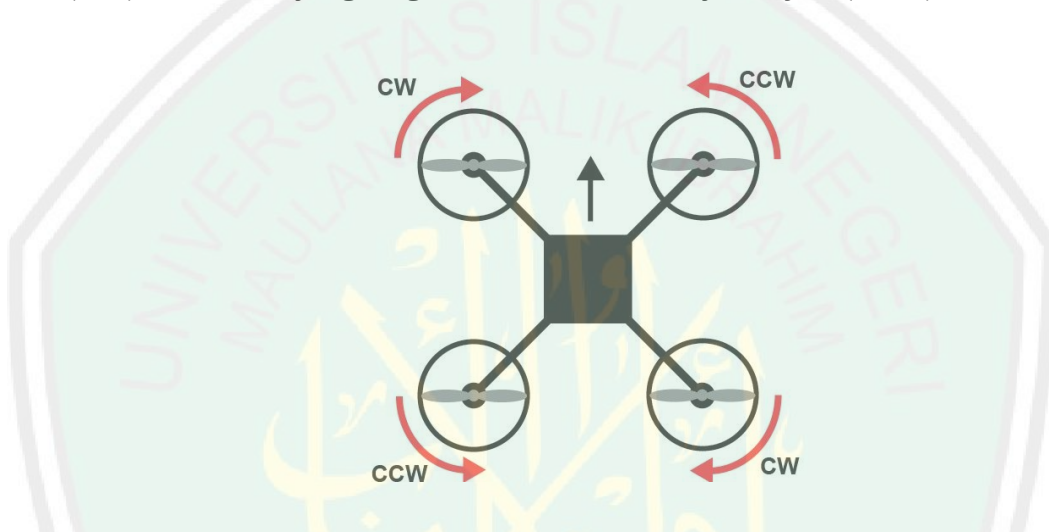
2.2.1 Konfigurasi dan Sistem Gerak *Quadcopter*

Quadcopter memiliki empat baling-baling penggerak yang diposisikan tegak lurus terhadap bidang datar. Masing-masing rotor (baling-baling dan motor penggeraknya) menghasilkan daya angkat dan memiliki jarak yang sama terhadap pusat massa pesawat. Dengan daya angkat masing-masing rotor sebesar lebih dari seperempat berat keseluruhan, memungkinkan *quadcopter* untuk terbang (Daniel, 2012).

Komponen utama dari *quadcopter* yaitu *fuselage* yang merupakan *body quadcopter*, dimana bagian ini adalah bagian yang paling banyak kegunaannya pada *quadcopter*. Baling-baling adalah penghasil gaya angkat pada *quadcopter*. Penempatan dan penyesuaian motor untuk memberikan stabilitas pada *quadcopter* selama melakukan penerbangan. *Driver* motor merupakan sarana yang mendukung untuk melakukan pergerakan motor dengan memberikan daya. *Elevator* adalah control permukaan yang mengatur gerak naik-turun *quadcopter*, ketika *elevator* (motor) depan turun kebawah maka gaya angkat pada motor belakang akan bertambah dan menyebabkan motor belakang akan tertarik untuk naik sementara motor depan *quadcopter* akan turun ke bawah. Misalnya ketika motor sebelah kiri turun ke bawah sedangkan motor kanan ke atas, maka gaya angkat akan bertambah pada motor sebelah kanan, sedangkan motor sebelah kiri gaya angkatnya akan berkurang yang akan menyebabkan *quadcopter* akan bergerak ke arah kiri (Blakelock, 1965).

Komponen *quadcopter* memiliki rotor, yang dimana masing-masing rotor (baling-baling dan motor penggeraknya) menghasilkan daya angkat dan memiliki

jarak yang sama terhadap pusat massa wahana. Dengan daya angkat masing-masing rotor sebesar lebih dari seperempat berat *quadcopter* keseluruhan, memungkinkan *quadcopter* untuk terbang. Kecepatan *quadrotor* tergantung pada kekuatan motor dan berat *quadrotor* itu sendiri. Untuk menghindari terjadinya momen putar pada *body*, arah putaran baling-baling pada setiap rotornya berbeda seperti yang terlihat Gambar 2.1. Terdapat 2 rotor yang bergerak searah jarum jam (CW) dan 2 rotor yang bergerak berlawanan arah jarum jam (CCW).



Gambar 2.1. Arah Putaran Baling-baling *Quadcopter*.

Konfigurasi yang paling sering digunakan adalah X. Ketika *quadcopter* sedang terbang dan melayang di udara maka kecepatan putar pada setiap rotornya adalah sama. Saat *quadcopter* melakukan gerakan maju, 2 buah baling-baling atau *propeller* yang berada dibelakang akan berputar lebih cepat sehingga *body quadcopter* akan miring ke depan. Gaya dorong yang dihasilkan keempat *propeller* akan mempunyai komponen gaya ke atas dan ke depan sehingga *quadcopter* akan terdorong ke arah depan sambil mempertahankan ketinggiannya (Risha, 2016).

Dalam membuat atau merakit *quadcopter* diperlukan beberapa komponen yang diperlukan, yaitu diantaranya *frame*, *remote control*, *brushless*, *propeller* atau baling-baling yang menyesuaikan dengan *body quadcopter*, *battery Li-Po*, dan mikrokontroler ATmega,

2.2.2 Konfigurasi Remote Control terhadap Quadcopter

Dalam menerbangkan ada beberapa dasar yang harus diketahui terutama pada konfigurasi *remote Control* terhadap *quadcopter*. *Remote Control* atau *transmitter* pada umumnya terdapat 2 mode, yaitu mode 1 dan mode 2, perbedaan *transmitter* mode 1 dan mode 2 tergantung pada orientasi kebiasaan tangan penggunaannya, kebalikan antara kiri dan kanan. Ibarat mobil Indonesia dengan mobil di luar negeri, jika mobil di Indonesia kemudi stir terdapat di sisi kanan sedangkan mobil di luar negeri kemudi terdapat di sisi kiri.



Gambar 2.2. Perbedaan *transmitter* mode 1 dan mode 2

Pada gambar 2.2 merupakan perbedaan *transmitter* mode 1 dan mode 2, kebanyakan *quadcopter* di Indonesia menggunakan *transmitter remote control* mode 2 yang sesuai dengan orientasi kebiasaan orang Indonesia setir kemudi yang berada di sisi kanan. Jadi dipastikan sebelum membeli ataupun membuat

quadcopter, pastikan *transmitter remote control* nya menggunakan *transmitter mode 2*. Berikut adalah fungsi-fungsi yang terdapat pada *transmitter remote control*.

a. Fungsi *Throttle*

Fungsi *throttle* atau gas pada *transmitter quadcopter* adalah untuk menaikkan laju dan menurunkan *quadcopter*. Seperti jika kita naik sepeda motor, jika kita memutar gas semakin dalam maka sepeda motor akan semakin kencang sama halnya pada *quadcopter* semakin kita menekan fungsi *throttle* maka naik maupun turun *quadcopter* akan semakin laju.



Gambar 2.3. Fungsi *throttle* untuk naik ke atas.



Gambar 2.4. Fungsi *throttle* untuk turun ke bawah.

b. Fungsi *Rudder/Yaw*

Fungsi yaw atau *rudder* pada *transmitter remote control quadcopter* adalah untuk memutar 360° ke kiri maupun ke kanan.



Gambar 2.5. Fungsi *rudder* untuk memutar 360° ke kanan.



Gambar 2.6. Fungsi *rudder* untuk memutar 360° ke kiri.

c. Fungsi *Pitch Elevator*

Fungsi *Pitch Elevator* pada *transmitter remote control quadcopter* adalah untuk menggerakkan maju atau mundur *quadcopter*.



Gambar 2.7. Fungsi *Pitch Elevator* untuk maju ke depan.



Gambar 2.8. Fungsi *Pitch Elevator* untuk mundur ke belakang.

d. Fungsi *Aileron/Roll*

Fungsi *Aileron/Roll* pada *transmitter remote control quadcopter* adalah untuk membuat *quadcopter* bergerak ke arah samping kanan atau kiri.



Gambar 2.9. Fungsi *Roll* ke arah samping kanan.



Gambar 2.10. Fungsi *Roll* ke arah samping kiri.

2.2.3 Sensor Ultrasonik

Menurut (Hani, 2010) sensor ultrasonik adalah sensor yang bekerja berdasarkan prinsip pantulan gelombang suara, dimana sensor menghasilkan gelombang suara yang kemudian menangkapnya kembali dengan perbedaan waktu sebagai dasar pengindraannya. Prinsip kerja sebuah sensor ultrasonik yaitu dengan mendeteksi objek dengan cara mengirim gelombang ultrasonik dan kemudian menerima pantulan gelombang tersebut.

2.3 Game

Game atau permainan merupakan sebuah aktivitas interaktif yang melibatkan satu pemain atau lebih yang didalamnya terdapat peraturan, tujuan, dan tantangan. Pengertian *game* menurut para ahli, yakni: Menurut Joseph Saulter dalam buku *Introduction to Video Game Design and Development* (2007:4), sebuah *game* memiliki beberapa karakteristik utama yaitu memiliki satu atau lebih dari satu pemain, terdapat peraturan dalam *game*, cara bermain yang terorganisir, dan memiliki tujuan serta akibat atau hasil. Menurut (Asrori Imam, 2009) *game* atau permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu

sehingga ada yang menang dan ada yang kalah, biasanya konteks tidak serius dengan tujuan penyegaran.

Menurut Schell (2008) *game* adalah kegiatan penyelesaian masalah, didekati dengan sikap yang menyenangkan, *game* juga sesuatu yang saat membuat pemain menemukan kesenangan dalam dalam memainkannya. *Game* yang bagus adalah *game* yang dapat membuat pengguna berpartisipasi secara aktif dan mempunyai jumlah tantangan yang tepat, tidak terlalu sedikit atau terlalu banyak. Sikap orang ketika sedang bermain *game*, bisa saja berbeda ketika orang itu sedang tidak bermain *game* maka dia akan merasa sedang berada di “dunia” yang *game* tersebut ciptakan.

Schell menyebutkan bahwa kebanyakan orang biasanya suka memecahkan masalah atau menghadapi tantangan. *Game* pasti menyediakan masalah dan tantangan untuk dihadapi oleh pengguna, jika *game* tidak menyediakan tantangan maka *game* tersebut akan kurang menyenangkan untuk dimainkan. *Game* juga menyediakan *goals* (tujuan) untuk pengguna, sehingga pengguna mempunyai tujuan dalam memainkan *game* tersebut. Jika *game* tidak mempunyai *goals* mungkin pengguna akan menemukan bahwa *game* tersebut membosankan.

Terori permainan (*game*) pertama kali ditemukan oleh sekelompok ahli matematika pada tahun 1944. Teori itu dikemukakan oleh John Von Neumann and Oskar Morgenstern yang berisi: “Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan

menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi.

2.3.1 Sejarah *Game*

Manusia telah mengenal dan memainkan *game* sejak zaman dahulu. Di Sahara ditemukan sebuah papan permainan terbuat dari batu yang berusia ± 5000 tahun. Menurut David Fox dan Roman Verhosek (2002), permainan *Go*, yang populer di negara-negara oriental, telah ada sejak 200 B.C. Bahkan permainan mirip *Backgamon* (*Tabula*, *Nard*) dicatat pada *script* romawi kuno. Memasuki zaman modern, *Game* yang pertama di dunia diciptakan pada tahun 1963 oleh Steve Russel seorang ahli computer yang berasal dari Amerika. *Game* yang pertama kali dibuat adalah *Apace war* yang kemudian dikembangkan oleh sebuah tim Martin Graetz, Pete Simson dan Dan Edwards. Mereka juga mengubah persepsi masyarakat pada waktu itu yang menganggap *computer* hanya untuk kerja yang serius.

Konsol Game yang pertama di dunia dibuat oleh Ralph H. Baer, lahir 98 Maret 1922. Seorang Jerman yang tinggal di Amerika sejak kecil. Ralph menciptakan sebuah permainan di televisi yang saat itu sedang ia kerjakan sekitar tahun 1966, di perusahaan yang bernama Sanders. Penemuan ini dikembangkan hingga menjadi *prototype konsol game* pertama yang dinamakan *Brown Box* dan dipatenkan pada tahun 1986. Ralph juga menemukan kontrol pistol untuk *video games* yang bisa dimainkan di televisi, dan juga merupakan yang pertama di dunia.

Pada akhir 90-an dan awal 2000-an, dengan berkembangnya *game* seperti : *Ultima Online*, *Ragnarok Online* dan *Everquest* menyadarkan kita bahwa *game* tidak hanya menarik dengan grafik yang bagus, namun juga dengan kehadiran pemain lain (*multiplayer*). Sementara *multiplayer game* terus berkembang, muncul paradigma baru dalam dunia informasi. Dengan banyaknya penggunaan *handphone*, *game* turut berkembang dan mendukung pesatnya pemasaran *micro device* tersebut. Walaupun grafik yang dapat didukung tidak sebanding dengan *console*, *handphone* menawarkan kepraktisan yang tidak bisa diberikan oleh *console* (David Fox dan Roman Verhosek, 2002)

2.3.2 Genre Game

Menurut (Sibero, 2009), *genre game* adalah klasifikasi *game* yang didasari interaksi pemainnya. *Game* dapat dikelompokkan ke dalam beberapa jenis yaitu sebagai berikut :

a. Game Simulasi

Contoh permainan yang termasuk dalam jenis *game* simulasi adalah simulasi konstruksi dan manajemen, simulasi kendaraan seperti yang diterapkan pada permainan balapan, perang, dan luar angkasa.

b. Game Edukasi

Game dengan jenis edukasi contohnya adalah *edugames* yang dibuat dengan tujuan spesifik yaitu sebagai alat pendidikan, misalnya *game* untuk belajar mengenal warna, mengenal huruf dan angka, belajar matematika, dan masih banyak yang lainnya.

Pengembang yang membuat *game* jenis ini harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan, dan meningkatkan ketrampilan pemain yang memainkannya.

c. *Entertainment*

Game entertainment ini berfungsi sebagai hiburan bagi orang yang memainkannya. Jenis *game* ini dibagi lagi menjadi beberapa bagian yaitu sebagai berikut :

1. Aksi / *Shooting*, (tembak-tembakan, peraturan, tergantung cerita dan tokoh di dalamnya). *Game* jenis ini memerlukan kecepatan refleks, koordinasi mata dan tangan, serta waktu.
2. *Game* pertualangan, *game* murni pertualangan lebih menekankan pada jalan cerita dan kemampuan berpikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan karakter hingga penggunaan benda-benda tepat pada tempat yang tepat
3. *Role playing*, *game* ini sesuai dengan terjemahannya yakni bermain peran. *Game* jenis ini memiliki penekanan pada tokoh perwakilan pemain dalam permainan dan biasanya tokoh atau katakter utama yang dimainkan dapat berubah menjadi semakin hebat dan kuat
4. *Endless game*, *game* jenis ini adalah jenis *game* yang tidak terdapat akhir cerita dari *game* tersebut. Pemain tidak pernah bisa untuk menyelesaikan *game* kecuali jika pemain tersebut mengalami kematian dalam *game* sehingga *game* akan berakhir. Jenis permainan ini

biasanya memiliki cerita yang sederhana, dan pemain hanya akan mendapatkan skor dan jarak terjauh saat bermain.

2.3.3 Platform Game

Untuk memainkan *game* terdapat berbagai cara untuk memainkannya, yaitu dengan menggunakan berbagai platform berikut:

- a. *Arcade games*, yaitu yang sering disebut ding-dong di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki *box* atau mesin yang memang khusus di design untuk jenis *video games* 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 tertentu).
- b. *PC games*, yaitu *video game* yang dimainkan menggunakan *Personal Computer*.
- c. *Console game*, yaitu *video games* yang dimainkan menggunakan *console* tertentu, seperti *Playstation 2*, *Playstation 3*, *XBOX 360*, dan *Nintendo Wii*.
- d. *Handheld games*, yaitu yang dimainkan di *console* khusus *video games* yang dapat dibawa kemana-mana, contoh *Nintendo DS* dan *Sony PSP*.
- e. *Mobile games*, yaitu yang dapat dimainkan atau khusus *mobile phone* atau *PDA*.

2.3.4 Elemen Game

Menurut Teresa Dillon (2012) komponen dasar sebuah *game* adalah sebagai berikut:

a. Plot

Plot biasanya berisi informasi tentang hal-hal yang akan dilakukan oleh *player* dalam *game* dan secara detail perintah tentang hal yang harus dicapai dalam *game*.

b. Tema

Didalam biasanya ada pesan moral yang akan disampaikan karakter atau pemain sebagai karakter utama maupun karakter utama maupun karakter yang lain memiliki ciri dan sifat tertentu.

c. *User Interface*

Merupakan fitur-fitur yang mengkomunikasikan *user* dengan *game*. *Interface* merupakan semua tampilan yang ada dalam suatu *game*. Sebuah *interface* yang baik adalah *interface* yang tidak membosankan dan memudahkan pemain *game*.

d. Aturan/*rules*

Game rule merupakan aturan perintah, cara menjalankan, fungsi objek dan karakter di didunia permainan dunia *game* bisa berupa pulau, dunia khayalan, dan tempat-tempat lain yang sejenis dipakai sebagai *setting* tempat dalam permainan *game*.

e. Animasi

Animasi ini selalu melekat pada dunia *game*, khususnya untuk gerakan karakter-karakter yang ada dalam *game*, seperti properti dari objek.

f. *Object*

Merupakan sebuah hal yang penting dan biasanya digunakan untuk pemain dalam memecahkan masalah, adakalanya pemain harus punya keahlian dan pengetahuan untuk bisa memainkannya.

g. Teks, Grafik, dan *Sound*

Game biasanya merupakan kombinasi dari media teks, grafik maupun suara, walaupun tidak harus semuanya ada dalam permainan *game*.

2.4 *Game Simulasi*

Menurut (Novak, 2012:76) *simulation game* adalah genre *game* yang mencoba untuk mereplikasi sistem, mesin, dan pengalaman dengan menggunakan peraturan sebenarnya yang ada di dunia. Mayoritas *game* simulasi diciptakan untuk tujuan hiburan, namun ada juga institusi pemerintah dan militer yang mengembangkan *game* simulasi untuk keperluan *training* dan *recruitment*. Ada beberapa tipe *game* simulasi seperti kendaraan, *participatory*, dan *process sims*. Peraturan yang berlaku dalam *game* simulasi adalah berdasarkan objek dan situasi dunia sesungguhnya. Menurut Bob Bates dalam buku *Game Design* (2004: 58-60), terdapat beberapa elemen yang harus ada dalam sebuah *game* simulasi yaitu, *wish fulfillment*, *hardcore versus casual gamer*, *simple interface*, *keep it fun*.

Tujuan *game* simulasi adalah agar pemain mendapatkan ilmu yang berasal dari *game* tentang objek atau kegiatan asli dunia. *Game* simulasi memberikan kesempatan kepada pemain untuk mendapatkan *exposure* yang tidak dapat dijangkau dalam dunia nyata. Pemain dapat mengunjungi lingkungan yang unik untuk mendapatkan pengalaman yang tidak bisa didapatkan dalam dunia nyata.

Simulasi dapat berupa *digital game* adalah *game flight simulator* atau *pinball* yang meniru proses nyata dari sudut yang diluncurkan.

Ada beberapa jenis permainan simulasi, diantaranya *life-simulation games*, *construction, and management simulation games*, dan *vehicle simulation*. Pada *life-simulation games*, pemain bertanggung jawab atas sebuah tokoh atau karakter dan memenuhi kebutuhan tokoh selayaknya kehidupan nyata, akan tetapi di ranah *virtual*. Karakter memiliki kebutuhan dan kehidupan layaknya manusia, seperti kegiatan bekerja, bersosialisasi, makan, belanja, dan sebagainya. Biasanya karakter ini hidup dalam sebuah dunia virtual yang dipenuhi karakter-karakter yang dimainkan pemain lainnya. Contoh permainannya adalah *Second Life*.

2.4.1 Sejarah Game Simulasi

Runtutan sejarah dan perkembangan tentang simulasi ternyata sangat panjang dan menarik untuk dikaji oleh salah satu seorang blogger yang bernama Godhams Road Safety dalam postingannya Simulasi dalam Perkembangan *Digital Game* pada tanggal 03 Juli 2012, mencoba untuk mengelompokkannya dari berbagai sumber dan mengurutkannya berdasarkan urutan waktu. Ketika perang dunia I berkecamuk (1914 – 1918), militer memanfaatkan *simulator* kuda mekanik yang terbuat dari kayu untuk melatih para relawan perang berlatih menaiki kuda. Dimaksudkan supaya prajurit lebih pintar mengendarai kuda yang dapat digunakan untuk menyerang musuh atau gerilya.

Perkembangan simulasi pun berlanjut ke era digital *game* elektronik. Awalnya *video game* merupakan pengembangan dari tabung sinar katoda yang terdapat dalam sistem peluru pertahanan pada akhir perang dunia II, program-

program ini kemudian diadaptasi ke dalam bentuk permainan simulator rudal *Cathode Ray Tube Device Amusement*. Permainan ini terinspirasi dari kecanggihan radar Perang Dunia II. Inilah *game* elektronik pertama yang tercatat dalam sejarah. Meskipun memiliki unsur permainan, *game* ini tidaklah populer. Penggunaannya terbatas untuk kepentingan simulasi latihan militer belaka. Tampilannya pun masih sederhana, belum berwarna dan hanya mengeluarkan suara tat-tit-tut. Tahun 1952, AS Douglas membuat *game* tic tac toe yang ditampilkan dalam sebuah tabung vakum computer. Kemudian tahun 1958, Willy Higginbotham membuat *game Tennis for Two* yang berjalan di osiloskop yang tergabung ke analog Donner Computer. Tahun 1961 – 1962, *game Space War* dikembangkan di MIT menggunakan grafik vektor di PDP-1.

Pada media 1970-an pula, *game* elektronik dapat dinikmati di rumah-rumah. Adalah Ralph Baer, seorang jerman berdarah Yahudi, yang mendesain *video game* rumahan pertama dengan protipe bernama *The Brown Box*. Pada akhirnya, perkembangan *video game* mencapai puncaknya, baik yang berjenis *arcade* maupun *portable*, dengan genre yang beraneka ragam, dari RPG sampai simulasi, dan hingga kini banyak digemari oleh beberapa kalangan, tidak hanya anak-anak tetapi juga orang dewasa.

2.5 Game Engine

Menurut Lewis, Michael & Jacobson, Jeffrey (2002) dalam penelitiannya *Game Engines in Scientific Research* menyebutkan bahwa *game engine* merujuk pada kumpulan modul kode simulasi yang tidak secara langsung menentukan perilaku permainan (*game logic*) atau lingkungan permainan (*level data*). *Game*

engine mencakup modul untuk menangani *input*, *output* (3D *Rendering*, gambar 2D, suara) dan *generic physics* atau dinamika untuk dunia *game*.

Menurut Rickman Roedavan (2016) dalam bukunya *Unity Tutorial Game Engine* menyebutkan bahwa *Game Engine* adalah sebuah perangkat lunak yang dirancang untuk membuat *game*. Sebuah *game engine* biasanya dibangun dengan mengenkapsulasi beberapa fungsi standar yang umum digunakan dalam pembuatan *game*. Misalnya, fungsi *rendering*, pemanggilan suara, *network*, atau pembuatan partikel untuk *special effect*. Sebagian besar *game engine* umumnya merupakan berupa *library* atau sekumpulan fungsi-fungsi yang penggunaannya dipadukan dengan bahasa pemrograman.

Diawal tahun 2000-an, *game engine* mengalami perkembangan yang cukup signifikan. Beberapa *game engine* mulai dilengkapi dengan *world editor*, sehingga dalam menggunakan perangkat lunak 3D seperti 3DMax atau Blender, pembuatan level atau dunia *game* dapat dibuat melalui sebuah perangkat lunak tersendiri yang telah dirancang khusus untuk *game engine* tersebut.

Ada banyak *game engine* yang dirancang untuk bekerja pada konsol *video game* dan sistem operasi desktop seperti Microsoft Windows, Linux, dan Mac OS X. Fungsi utama yang disediakan oleh *game engine* biasanya adalah *render* untuk grafis 2D atau 3D, suara, *script*, animasi, kecerdasan buatan, jaringan, *streaming*, manajemen memori, *threading*, dukungan lokalisasi, dan adegan grafik. *Game Engine* biasanya menyediakan *platform* abstraksi, yang memungkinkan permainan yang sama untuk dijalankan pada berbagai *platform* termasuk konsol *game* dan komputer pribadi dengan berbagai perbedaan yang tidak besar. *Game Engine*

yang membantu dalam membuat keputusan untuk menentukan *frame* sampai menentukan *artwork* yang ada di dalam *scene*.

2.6 Unity 3D

Menurut (Sari P.Z, 2013) Unity adalah salah satu *game engine* yang mudah digunakan, hanya membuat objek dan diberikan fungsi untuk menjalankan objek tersebut. Dalam setiap objek mempunyai variabel, variabel inilah yang harus dimengerti supaya dapat membuat *game* yang berkualitas. Berikut ini adalah bagian-bagian dalam *Unity* .

Menurut buku yang diterbitkan Informatika menyebutkan bahwa Unity 3D merupakan sebuah *game engine*, yaitu *software* pengolah gambar, grafik, suara, *input*, dan lain-lain yang ditunjukkan untuk *game*. Unity 3D merupakan *game engine multiform* yang mampu di-publish secara *standalone* (.exe), berbasis web, android, iOS, XBOX, maupun PS4, dengan catatan mendapatkan lisensi. Unity merupakan sebuah *game engine* yang dibuat oleh Unity Technology. Kelebihan unity dibandingkan dengan *game engine* lainnya adalah kemampuan *game cross platform*. Dengan unity 3D, *game* yang dibuat dapat dimainkan diberbagai perangkat, *smartphone* dan *game console*.

2.6.1 Sejarah Unity 3D

Unity Technologies dibangun pada tahun 2004 oleh David Helgason (CEO), Nicholas Francis (CCO), dan Joachim Ante (CTO) di Copenhagen, Denmark setelah *game* pertama GooBall, gagal lagi dalam meraih sukses. Ketiganya menyadari sebuah *engine* dan *tool* dalam sebuah pengembangan *game* dan berencana untuk membuat sebuah *engine* yang dapat digunakan oleh semua

dengan harga yang terjangkau. Unity Technologies mendapat bantuan dana dari Sequoia Capital, WestSummit Capital, dan iGlobe Partners. Kesuksesan *unity* terletak pada fokus mereka untuk memenuhi kebutuhan *indie developer* yang tidak dapat membangun *game engine* mereka sendiri atau membeli *lisensi game engine* yang terlalu mahal. Fokus perusahaan ini adalah “*Democratize game development*” atau diterjemahkan sebagai “Demokrasi pembangunan *game*” dan membuat sebuah pembangunan *game* baik 2D maupun 3D bisa dicapai oleh banyak orang.

Unity 3D dibagi menjadi dua versi, yaitu berbayar dan versi gratis. Pada versi gratis terdapat beberapa fitur yang tidak dapat digunakan seperti tidak dapat melakukan konversi ke console.

2.6.2 Fitur-fitur Unity

a. Rendering

Unity telah mendukung penggunaan *graphic engine*, seperti Direct3D (Windows, Xbox 360), Open GL (Mac, Windows, Linux, PS3), OpenGL ES (Android iOS), dan APIs (Wii). Selain itu, unity 3D juga mendukung penggunaan *bump mapping*, *reflection mapping*, *parallax*, *screen space ambient occlusion* (SSAO), *dynamic shadows* menggunakan *shadow maps*, *render-to-texture* dan *full-screen post-processing effects*.

Unity dapat mengambil format desain dari *3D Max*, *Maya*, *SoftImage*, *Blender*, *Modo*, *ZBrush*, *Cinema 4D*, *Cheetah3D*, *Adobe Photoshop*, *Adobe Fireworks*, dan *Allegorithmic Substance*. Asset tersebut tersebut dapat

ditambahkan ke *game project* dan diatur melalui *graphical user interface unity*.

Shaderlab adalah bahasa yang digunakan untuk *shaders*, dimana mampu memberikan deklaratif “*programming*” dari *fixes-function pipeline* dan *program shader* ditulis dalam *GLSL* atau *Cg*. Sebuah *shader* dapat menyertakan banyak varian dan sebuah spesifikasi *fallback decralative*, dimana membuat Unity dapat mendeteksi berbagai macam *video card* terbaik saat ini, dan jika tidak ada yang kompatibel, maka akan dilempar menggunakan *shader* alternatif yang mungkin dapat menurunkan fitur atau performa.

b. *Scripting*

Script game engine dibuat dengan *Mono 2.6*, sebuah implementasi *open source* dari *.NET Framework* dapat menggunakan *Unity Script* (bahasa terkustomisasi yang terinspirasi dari *sintax ECMAScript*, dalam bentuk *Javascript*), *C#*, atau *Boo* (terinspirasi dari *sintax bahasa pemrograman phyton*). Dimulai dengan dirilisnya versi 3.0, *unity* menyertakan versi *MonoDevelop* yang terkustomisasi untuk *debug script*.

c. *Asset Tracking*

Unity juga menyertakan *Server Unity Asset* yang merupakan sebuah solusi terkontrol untuk *developer game asset* dan *script*. *Server* tersebut menggunakan *PostgreSQL* sebagai *backend*, *sistem audio* dibuat menggunakan *FMOD library* (dengan kemampuan untuk memutar *Ogg Vorbis compressed audio*), *video playback* menggunakan *Theora codec*,

engine daratan dan *vegetasi* (dimana mensupport *tree billboard*, *Occlusion Culling* dengan *Umbra*), *built-in loghtmapping* dan *global illumination* dengan *beast*, *multiplayer networking* menggunakan *RakNet*, dan *navigasi mesh* pencari jalur *built-in*.

d. *Platforms*

Unity support pengembangan ke berbagai *platform*. Di dalam *project*, *developer* memiliki control untuk mengirim ke berbagai perangkat seperti *mobile*, *web browser*, *desktop*, dan *console*. *Unity* juga mengijinkan spesifikasi kompresi *texture* dan pengaturan resolusi di setiap *platform* yang didukung.

Saat ini *platform* yang didukung adalah *Balckberry 10*, *Windows 8*, *Windows Phone 8*, *Windows*, *Mac*, *Linux*, *Android*, *iOS*, *Unity Web Player*, *Adobe Flash*, *PlayStation 3*, *Xbox 360*, *Wii U*, dan *Wii*. Meskipun tidak semua terkonfirmasi secara resmi, *Unity* juga mendukung *PlayStation Vita* yang dapat dilihat pada game *Escape Plan* dan *Oddworld: New 'n' Tasty*.

e. *Asset Store*

Pada *unity* terdapat *Asset Store* yang menjadi tempat untuk mendapatkan *asset*, dimana *asset* merupakan *asset* permainan yang akan direferensikan oleh beberapa komponen, *asset* itu sendiri, atau kelengkapan penunjang pembuatan *game*. *Asset* yang ada pada *Unity 3D* dibagi menjadi dua, yaitu eksternal dan internal. *Asset* eksternal merupakan *asset* yang ditambahkan dan bersumber di luar *Unity 3D*, seperti *3D Model*, *Texture*, *Sound Effect*. *Asset* internal

merupakan asset yang sudah ada didalam Unity 3D seperti *Materials*, *Shaders*, *Cube Maps*, *Physics Material*, dan *Prefabs*.

Pada *asset store* di *download* akan menjadi *packages* yang dimana *package* merupakan kumpulan asset yang sudah dijadikan satu. Beragam *packages* dapat didownload di situs unity <http://assetsstore.unity3d.com>. *Packages* ada yang berbayar dan juga terdapat gratis, pada *packages* terdapat *game object* dan *prefabs*, yang dimana *prefabs* merupakan sebuah container atau sebagai salah satu cara membuat grup asset sehingga dapat digunakan berkali-kali didalam sebuah *project*, *prefabs* juga dianggap sebagai symbol (flash), tetapi gabungan script juga dapat dijadikan *prefab*.

Pembuatan *prefab* dapat diambil dari gabungan materi di dalam Hierarchy, biasanya diawali menggunakan *tools* yang dinamakan *Game Object*. Untuk membuat *prefabs* dengan menggunakan *game object* yang empty atau kosong. Ciri-ciri *game object*, yaitu:

1. Dapat dipindahkan (Move), diputar (Rotated), dan mengubah ukuran (Scale).
2. Dapat diberi nama atau di *rename*.
3. Mempunyai sifat hierarchy atau link.
4. Dapatkan didefinisikan melalui *Component*. *Component* adalah grup dari suatu fungsi yang berisikan parameter-parameter yang mendefinisikan seperti apa bentuk atau sifat dari *game object*.

f. *Physics*

Unity juga memiliki *support built-in* untuk *PhysX physics engine* (Sejak *Unity* versi 3.0) dari *NVidia* (sebelumnya *Ageia*) dengan penambahan kemampuan untuk *simulasi real-time cloth* pada *arbitrary* dan *skinned meshes*, *thick ray cast*, dan *collision layers* (Rickman Roedavan, 2014).

g. *Raycast*

Raycasting merupakan cara untuk melakukan pengecekan dengan menggunakan suatu garis lurus yang tidak terlihat untuk mendapatkan object apa saja yang ada di depannya dan dimanakah titik *collider* yang mengenai Ray (sinar).

2.7 *Fuzzy Logic*

Logika *fuzzy* pertama dikenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang cocok di implementasikan pada sistem mulai dari sistem yang sederhana, sistem kecil, jaringan PC dan lain-lain. Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner yang artinya hanya mempunyai dua kemungkinan, “ya atau tidak”, “benar atau salah”, “baik atau buruk”, dan lain-lain. Oleh sebab itu, semua dapat mempunyai nilai keanggotaan 0 atau 1. Namun, dalam logika *fuzzy* memungkinkan untuk nilai keanggotaan berada antar 0 dan 1. Artinya, bisa saja suatu keadaan mempunyai dua nilai “ya atau tidak”, “benar dan salah” secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya.

Jika dibandingkan dengan logika konvensional, kelebihan logika *fuzzy* adalah kemampuannya dalam proses penalaran secara bahasa sehingga dalam

perancangannya tidak memerlukan penamaan matematik yang rumit. Untuk memahami logika *fuzzy*, sebelumnya perhatikan tentang konsep himpunan *fuzzy*, himpunan *fuzzy* memiliki 2 atribut, yaitu :

1. *Lingustik*, yaitu suatu kelompok yang mewakili suatu keadaan tertentu dengan menggunakan bahasa alami, misalnya DINGIN, SEJUK, PANAS mewakili variable temperature.
2. *Numeris*, yaitu suatu nilai yang menunjukkan ukuran dari suatu variabel, misalnya 10, 30 dan sebagainya. (T.Sutejo, 2011).

Definisi formal logika *fuzzy*, logika *fuzzy* merupakan sebuah logika yang dipresentasikan oleh ekspresi *fuzzy* (rumus) yang memenuhi kriteria sebagai berikut:

1. Nilai kebenaran, 0 dan 1, dan variabel x_i ($\in [0,1]$, $i = 1,2,\dots,n$) merupakan ekspresi *fuzzy*.
2. Jika f merupakan ekspresi *fuzzy*, $\sim f$ juga merupakan ekspresi *fuzzy*.
3. Jika f dan g merupakan ekspresi *fuzzy*, $f \wedge g$ dan $f \vee g$ juga merupakan ekspresi *fuzzy* (Lee, 2005:201).

Terdapat banyak mode aturan *fuzzy* yang bisa digunakan dalam proses *inference* akan tetapi ada dua model aturan yang paling sering digunakan yaitu :

1. Model Mamdani

Bentuk aturan yang digunakan pada model Mamdani adalah sebagai berikut :

IF x_1 is A_1 AND ... AND x_n is A_n THEN y is B Dimana $A_1, \dots, A_n, B$ adalah nilai-nilai linguistik, sedangkan “ x_1 is A_1 “ menyatakan bahwa nilai dari variable x_1 adalah anggota himpunan fuzzy A .

2. Model Sugeno

Model Sugeno merupakan varian dari model Mamdani dan memiliki bentuk aturan sebagai berikut : IF x_1 is A_1 AND AND x_n is A_n THEN $y=f(x_1, \dots, x_n)$ Dimana f bisa berupa sembarang fungsi dari variabel-variabel masukan yang nilainya berada dalam interval variabel keluaran. Dari penjelasan tentang logika fuzzy dapat diketahui bahwa suatu sistem yang menggunakan logika fuzzy mampu menangani suatu masalah ketidakpastian dimana masukan yang diperoleh merupakan suatu nilai yang kebenarannya bersifat sebagian (Dewi, 2003).

Selanjutnya dalam menentukan perubahannya digunakan model fuzzy sugeno, dimana setiap outputnya telah ditetapkan pada konstanta yang sudah ditentukan sebelumnya.

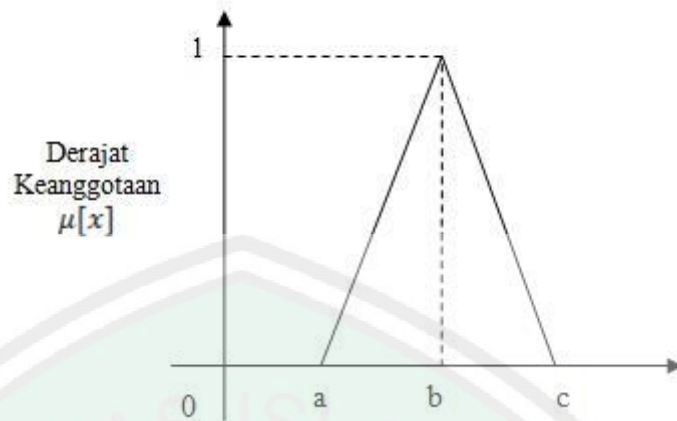
2.7.1 Fungsi Keanggotaan

Fungsi keanggotaan (*member function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval 0 sampai 1.

1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah menggunakan pendekatan fungsi (Kusumadewi & Purnomo, 2010).

1. Fungsi Keanggotaan Segitiga

Kurva Segitiga pada dasarnya merupakan gabungan antara 2 garis (linear) seperti terlihat pada gambar 2.13 :



Gambar 2.11. Fungsi Keanggotaan Segitiga.

Fungsi Keanggotaan yang mempunyai parameter a, b, c dengan formulasi segitiga :

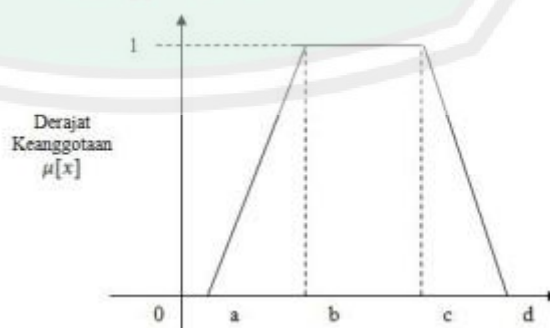
$$\mu(x; a, b, c) = 0; x \leq a \text{ atau } x \geq c$$

$$\mu[x] = (x-a)/(b-a); a \leq x \leq b$$

$$(b-x)/(c-b); b \leq x \leq c$$

2. Fungsi Keanggotaan Trapesium

Kurva Trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1 seperti terlihat pada gambar 2.12.



Gambar 2.12. Fungsi Keanggotaan Trapesium.

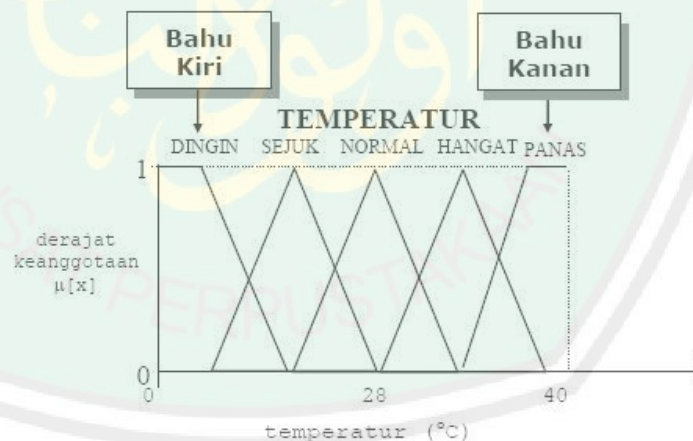
Fungsi keanggotaan yang mempunyai parameter a, b, c, d dengan formulasi Trapezium $(x; a, b, c, d) = 0; x \leq a$ atau $x \geq d$.

Tabel 2.1. Formulasi Fungsi Keanggotaan Trapezium

$(x-a)/(b-a);$	$a \leq x \leq b$
$\mu[x] = 1;$	$b \leq x \leq c$
$(d-x)/(d-c);$	$x \geq d$

3. Fungsi Keanggotaan Bahu

Daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun. Himpunan *fuzzy* bahu, bukan segitiga, digunakan untuk mengakiri variabel suatu daerah *fuzzy*. Bahu kiri bergerak dari benar ke salah, demikian juga bahu kanan bergerak dari salah ke benar.



Gambar 2.13. Fungsi keanggotaan Kurva Bahu.

2.7.2 Sistem Inferensi Fuzzy

Fuzzy Inferense System (FIS) atau *Fuzzy Inferense Engine* adalah sistem yang dapat melakukan penalaran dengan prinsip serupa seperti manusia

melakukan penalaran dengan nalurnya (Alavi, et al., 2010). Langkah pertama dari FIS adalah untuk menetapkan nilai keanggotaan untuk data *input* dan output.(Alidoosti, et al., 2012).

Menurut Kusumadewi & Hartati (2010), sistem inferensi *fuzzy* merupakan suatu kerangka komputasi yang didasarkan pada teori himpunan *fuzzy*, aturan *fuzzy* yang berbentuk IF-THEN, dan penalaran *fuzzy*. Secara garis besar, diagram blok proses inferensi *fuzzy* terlihat pada gambar 2.11.



Gambar 2.14. Diagram Blok Sistem Inferensi *Fuzzy*.

Sistem inferensi *fuzzy* menerima *input crisp*. *Input* ini kemudian dikirim ke basis pengetahuan yang berisi *n* aturan *fuzzy* dalam bentuk IF-THEN. *Fire strength* akan dicari pada setiap aturan. Selanjutnya pada hasil agregasi akan dilanjutkan dengan *defuzzy* untuk mendapatkan nilai *crisp* sebagai *output* sistem. Sistem inferensi *fuzzy* didasarkan pada konsep penalaran monoton. Pada metode penalaran secara monoton, nilai *crisp* pada daerah konsekuen dapat diperoleh secara langsung berdasarkan *fire strength* pada antesedennya. Salah satu syarat yang harus dipenuhi pada metode penalaran ini adalah himpunan *fuzzy* pada konsekunnya harus bersifat monoton (baik monoton naik maupun monoton turun). Salah satu metode FIS yang dapat digunakan untuk pengambilan keputusan adalah metode *Tsukamoto*.

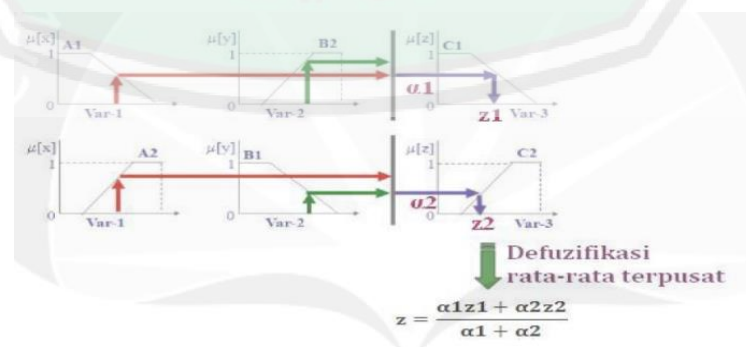
Pada metode *Tsukamoto*, implikasi setiap aturan berbentuk implikasi ”sebab-akibat”/ implikasi “*Input-Output*” dimana antara antesden dan konsekuen harus ada hubungannya. Setiap aturan direpresentasikan menggunakan himpunan-himpunan *fuzzy*, dengan fungsi keanggotaan yang monoton. Kemudian untuk menentukan hasil tegas (*Crisp Solution*) digunakan rumus penegasan (defuzifikasi) yang disebut metode rata-rata terpusat atau metode defuzifikasi rata-rata terpusat (*Center Average Defuzzifier*) (Setiadji,2009). Untuk lebih memahami metode *Tsukamoto*, perhatikan contoh di bawah ini.

Misalkan ada 2 variabel *input*, Var-1 (x) dan Var-2 (y), serta variable output, Var-3 (z), dimana Var-1 terbagi atas 2 himpunan yaitu A1 dan A2. Var-2 terbagi atas 2 himpunan B1 dan B2, Var-3 terbagi juga atas 2 himpunan yaitu C1 dan C2 (C1 dan C2 harus monoton). Ada 2 aturan yang digunakan, yaitu :

[R1] IF (x is A1) and (y is B2) THEN (z is C1)

[R2] IF (x is A2) and (y is B1) THEN (z is C2)

Pertama-tama dicari fungsi keanggotaan dari masing-masing himpunan *fuzzy* dari setiap aturan, yaitu himpunan A1, B2 dan C1 dari aturan *fuzzy* [R1], dan himpunan A2, B1 dan C2 dari aturan *fuzzy* [R2]. Aturan *fuzzy* R1 dan R2 dapat direpresentasikan dalam gambar 2.12 untuk mendapatkan suatu nilai crips z.



Gambar 2.15. Inferensi dengan menggunakan Metode Tsukamoto.

Karena pada metode *Tsukamoto* operasi himpunan yang digunakan adalah konjungsi (AND), maka nilai keanggotaan antesden dari *fuzzy* [R1] adalah irisan dari nilai keanggotaan A1 dari Var-1 dengan nilai keanggotaan B1 dari Var-2. Nilai keanggotaan antesden dari operasi konjungsi (*And*) dari aturan *fuzzy* [R1] adalah nilai minimum antara nilai keanggotaan A1 dari Var-1 nilai keanggotaan B2 dari Var-2. Demikian pula nilai keanggotaan antesden dari aturan *fuzzy* [R2] adalah nilai minimum antara nilai keanggotaan A2 dari Var-1 dengan nilai keanggotaan B1 dari Var-2. Selanjutnya, nilai keanggotaan antesden dari aturan *fuzzy* [R1] dan [R2] masing-masing disebut dengan a1 dan a2. Nilai a1 dan a2 kemudian disubsitusikan pada fungsi keanggotaan himpunan C1 dan C2 sesuai dengan aturan *fuzzy* [R1] dan [R2].

Untuk memperoleh nilai output *crisp* nilai tegas z, dicari dengan cara mengubah *input* (berupa himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*) menjadi suatu bilangan pada domain himpunan *fuzzy* tersebut. Cara ini disebut dengan metode defuzifikasi (penegasan). Metode defuzifikasi yang digunakan dalam metode *Tsukamoto* adalah metode defuzifikasi rata-rata terpusat (*Center Average Defuzzifier*) yang dirumuskan sebagai berikut :

$$Z = \frac{(\alpha - \text{predikat}_1 * Z_1) + (\alpha - \text{predikat}_2 * Z_2) + \dots + (\alpha - \text{predikat}_n * Z_n)}{(\alpha - \text{predikat}_1) + (\alpha - \text{predikat}_2) + \dots + (\alpha - \text{predikat}_n)}$$

(Defuzifikasi rata-rata terpusat)

BAB III

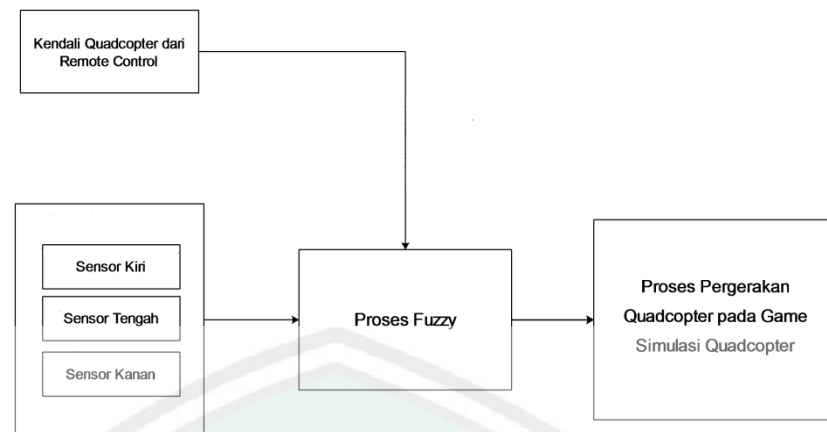
ANALISIS DAN PERANCANGAN

3.1 Desain Sistem

Desain sistem merupakan perancangan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah dari satu kesatuan yang utuh dan berfungsi. Sistem ini dibangun dari *input* yang merupakan proses awal berjalannya sistem hingga *output* akhir yang akan didapatkan melalui sistem yang menghubungkan *remote control* yang menggunakan *keyboard* dan terhubung ke *game* simulasi *X-Drone*.

3.1.1 Deskripsi Aplikasi

Konsep pada *game* ini adalah *game* simulasi dengan menggunakan sudut pandang orang ketiga atau *third person* dalam memainkannya. *Game* ini mensimulasikan bagaimana mengendarai *quadcopter* bagi pemain dimana pemain menerbangkan dan mengarahkan *quadcopter* berdasarkan *inputan* dari *remote control*, *game* ini dimainkan pada platform *personal computer* dengan menggunakan sistem operasi *windows*. Untuk lebih jelasnya, dijelaskan menggunakan diagram blok pada Gambar 3.1.



Gambar 3.1. Diagram Blok

Pada gambar 3.1, dapat dijelaskan bahwa pada sistem ini mulanya di kendalikan menggunakan *remote control* untuk pengendalian utama, kemudian akan diproses menggunakan oleh logika *fuzzy* yang juga mendapatkan *input* jarak dari sensor-sensor pada *quadcopter* pada *game* simulasi. Hasil dari logika *fuzzy* tersebut di interpresentasikan terhadap proses pergerakan *quadcopter* pada *game* simulasi *x-drone*.

3.1.2 Gameplay

Untuk memainkan *game* simulasi *quadcopter*, pemain harus memahami fungsi-fungsi *input* yang diberikan melalui *remote control*, yakni terdapat fungsi *throttle*, *rudder*, *pitch*, dan *Roll*. Dari fungsi-fungsi tersebut maka pemain dapat menggerakkan *quadcopter*. Dibawah ini adalah Gambar 3.2 dengan penjelasannya :



Gambar 3.2. Remote control untuk mode 2.

- a. Fungsi *Throttle Increase/Decrease* : untuk menerbangkan *quadcopter* ke atas atau kebawah, menggunakan tombol *keyboard* W dan S.
- b. Fungsi *Rudder Yaw Left/Right* : untuk memutar rotasi *quadcopter* ke arah kiri atau ke kanan menggunakan tombol *keyboard* A dan D.
- c. Fungsi *Cyclic Pitch Forward/Back* : untuk menggerakkan *quadcopter* maju kedepan atau mundur ke belakang menggunakan tombol *keyboard* up arrow dan down arrow.
- d. Fungsi *Cyclic Roll Left/Right* : untuk menggerakkan *quadcopter* ke samping kanan atau kiri menggunakan tombol *keyboard* left arrow dan right arrow

Dengan menggerakkan *quadcopter* ini maka pemain mencoba untuk menggerakkan *quadcopter* kepada titik pendaratan terakhir.

3.1.3 Design Interface

Design interface merupakan perancangan rancangan dari *game* simulasi *x-drone* yang dirangkum sebagai berikut:

1. Rancangan *Splash Screen*



Gambar 3.3. Rancangan *Splash Screen*.

Pada Gambar 3.3 merupakan rancangan dari *splash screen* yang berisikan nama karakter dari *game* tersebut yaitu *X-Drone*. Rancangan *splash screen* akan muncul pertama kali ketika pemain menjalankan *game X-Drone*.

2. Rancangan *Loading Bar Screen*



Gambar 3.4. Rancangan *Loading Bar Screen*.

Pada Gambar 3.4 merupakan rancangan *loading bar screen* yang menampilkan nama dari *game x-drone simulation* dan *loading*

bar, yang dimana pada rancangan ini *loading bar* akan mengextract seluruh *asset* ke dalam *game* tersebut agar dapat berjalan dengan baik.

3. Rancangan Main Menu



Gambar 3.5. Rancangan Main Menu.

Pada Gambar 3.5 merupakan rancangan *main menu* pada *game simulasi x-drone* yang berisikan dua tombol yaitu *Main Menu* dan *Exit*. Tombol *main menu* ketika di klik akan menuju ke *scene game*, sedangkan tombol *exit* ketika di klik akan keluar dari *game*.

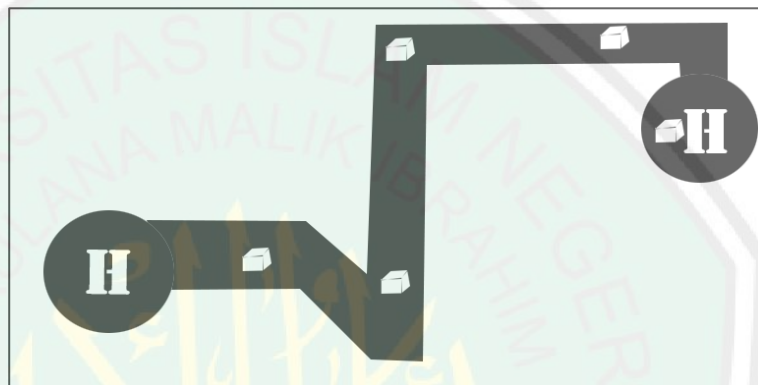
4. Rancangan Game



Gambar 3.6. Rancangan Game.

Pada Gambar 3.6 merupakan rancangan *game* ketika pemain telah menekan tombol *main menu* pada rancangan *main menu* sebelumnya. Pada rancangan *game* ini terdapat karakter *x-drone* yang berada pada area *game* yang dikelilingi oleh hutan, dan berbagai macam *asset* lainnya

5. Rancangan *Map Game*



Gambar 3.7. Rancangan *Map Game*.

Pada Gambar 3.7 yang merupakan peta *game* yang dimana rancangan pada *game* akan dirancang memiliki titik penerbangan dan titik pendaratan, selama menuju ke titik pendaratan *x-drone*. Area yang berwarna putih nantinya akan di isi oleh beberapa *environment* seperti gunung, rumput atau tumbuhan beserta karakter-karakter animasi lainnya.

6. Rancangan *Win Screen*



Gambar 3.8. Rancangan *Win Screen*.

Pada Gambar 3.8 yang merupakan rancangan *win screen* yang dimana rancangan ini akan muncul jika pemain berhasil mengontrol *x-drone* mendarat di titik pendaratan.

7. Rancangan *Game Over Screen*



Gambar 3.9. Rancangan *Game Over Screen*.

Pada Gambar 3.9 merupakan rancangan *game over* yang dimana akan tampil jika pemain tidak dapat mengarahkan *x-drone* mendarat pada titik terakhir.

3.1.4 Desain Karakter

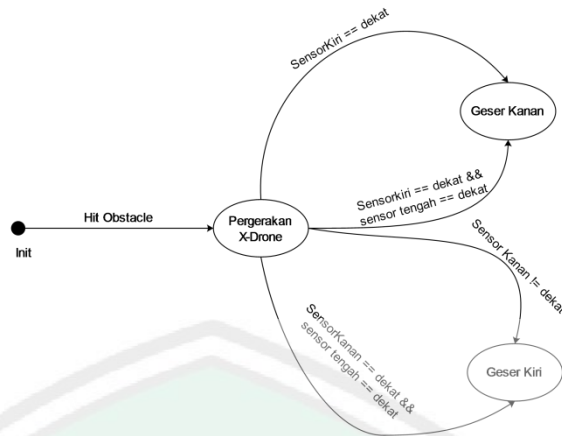
Desain karakter merupakan perancangan karakter yang diperlukan dalam *game* simulasi *x-drone* berikut adalah rancangan karakter pada *game* simulasi *x-drone* :



Gambar 3.10. Karakter X-Drone.

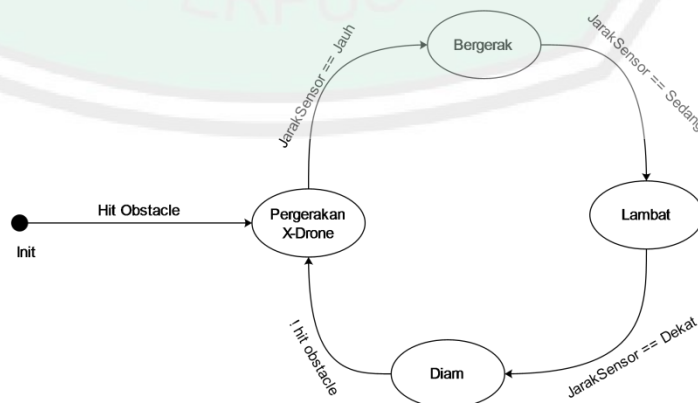
3.1.5 Finite State Machine

FSM adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: *State* (Keadaan), *Event* (kejadian) dan *action* (aksi). Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada pada salah satu *state* yang aktif. Sistem dapat beralih atau bertransisi menuju *state* lain jika mendapatkan masukan atau *event* tertentu, baik yang berasal dari perangkat luar atau komponen dalam sistemnya itu sendiri. Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relatif kompleks (Setiawan : 2006).



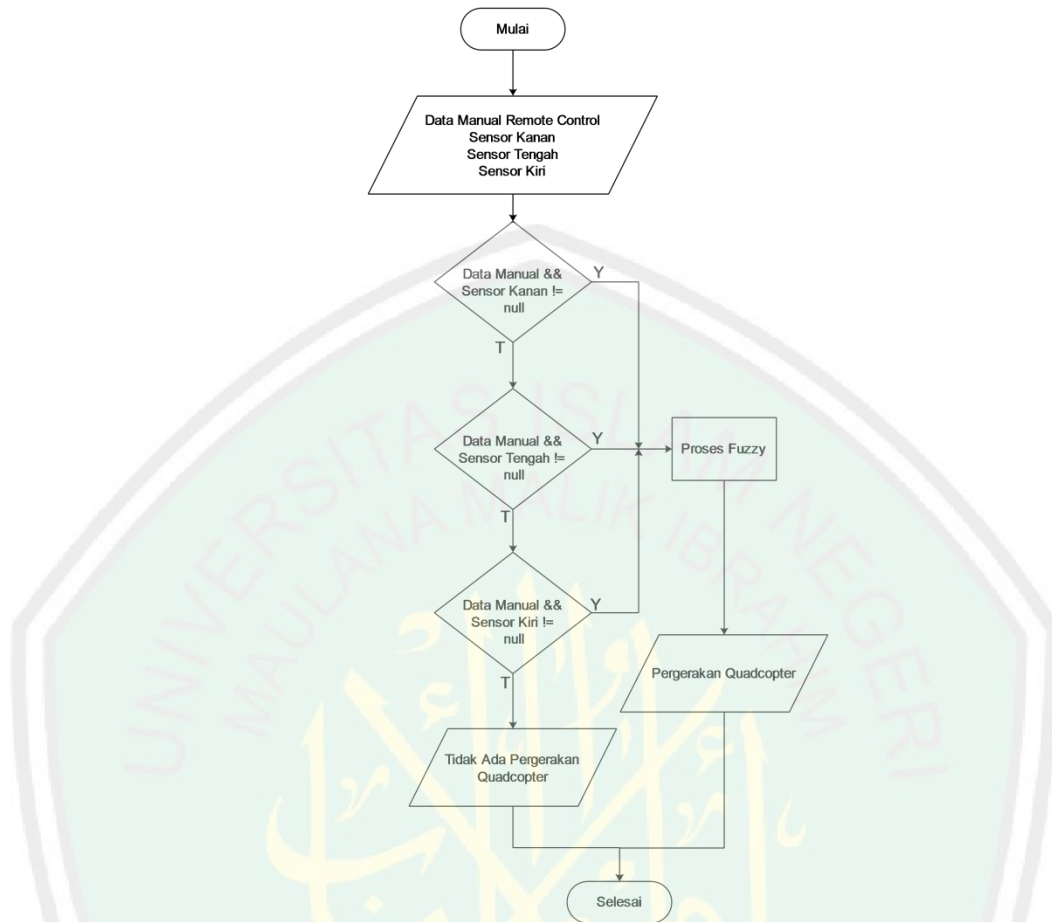
Gambar 3.11. FSM *Drone off remote control*.

Pada Gambar 3.11 menjelaskan FSM ini dilaksanakan jika terdapat aksi geser kanan dan geser kiri, sistem secara otomatis mematikan fungsi-fungsi pada *remote control* dan bergerak secara otomatis. Saat sensor kanan sama dengan dekat atau sensor kanan dan tengah sama dengan dekat maka akan melakukan aksi geser kiri, sedangkan saat sensor kiri sama dengan dekat atau sensor kiri dan tengah sama dengan dekat maka akan melakukan aksi geser kanan. Berbeda halnya dengan Gambar 3.12 ini yang dimana masih dapat menerima *input* dari *remote control*, seperti pada aksi lambat fungsi *throttle increase* masih dapat berfungsi akan tetapi dikurangi kecepatannya, sedangkan pada aksi diam fungsi *throttle increase* tidak berfungsi, akan tetapi fungsi-fungsi *remote control* yang lain masih berfungsi.



Gambar 3.12. FSM *Drone On Remote Control*

3.1.6 Alur Sistem Otomasi



Gambar 3.13. Diagram Alur Sistem Otomasi.

Sistem otomatis ini merupakan sistem kendali *quadcopter* yang mampu menghindari *obstacle* atau halangan yang berada di depan *quadcopter*. Pada diagram alur sistem otomasi yang di tunjukkan pada Gambar 3.13. Data Manual dari *remote Control* akan dibaca dan didapatkan hasil *inputnya* melalui *keyboard* sedangkan data-data sensor berasal pada data *collider* dan *raycast hit* yang mensimulasikan sensor pada *quadcopter*.

Sehingga didapatkan *input*, maka proses kedua adalah melakukan proses *inference*, dimana dalam proses *inference* ini terdapat aturan dasar atau

biasa disebut *fuzzy rule*, pada proses ini hasil dari *inputan fuzzy* akan dicari nilai implikasi berdasarkan *fuzzy rule* yang telah dibuat. Kemudian selanjutnya dilakukan proses pengolahan yaitu *defuzzyfication* yang akan menghasilkan *output*. Dimana hasil *output* tersebut dengan tujuan untuk menghindari halangan jika ditemukan halangan didepan *quadcopter*.

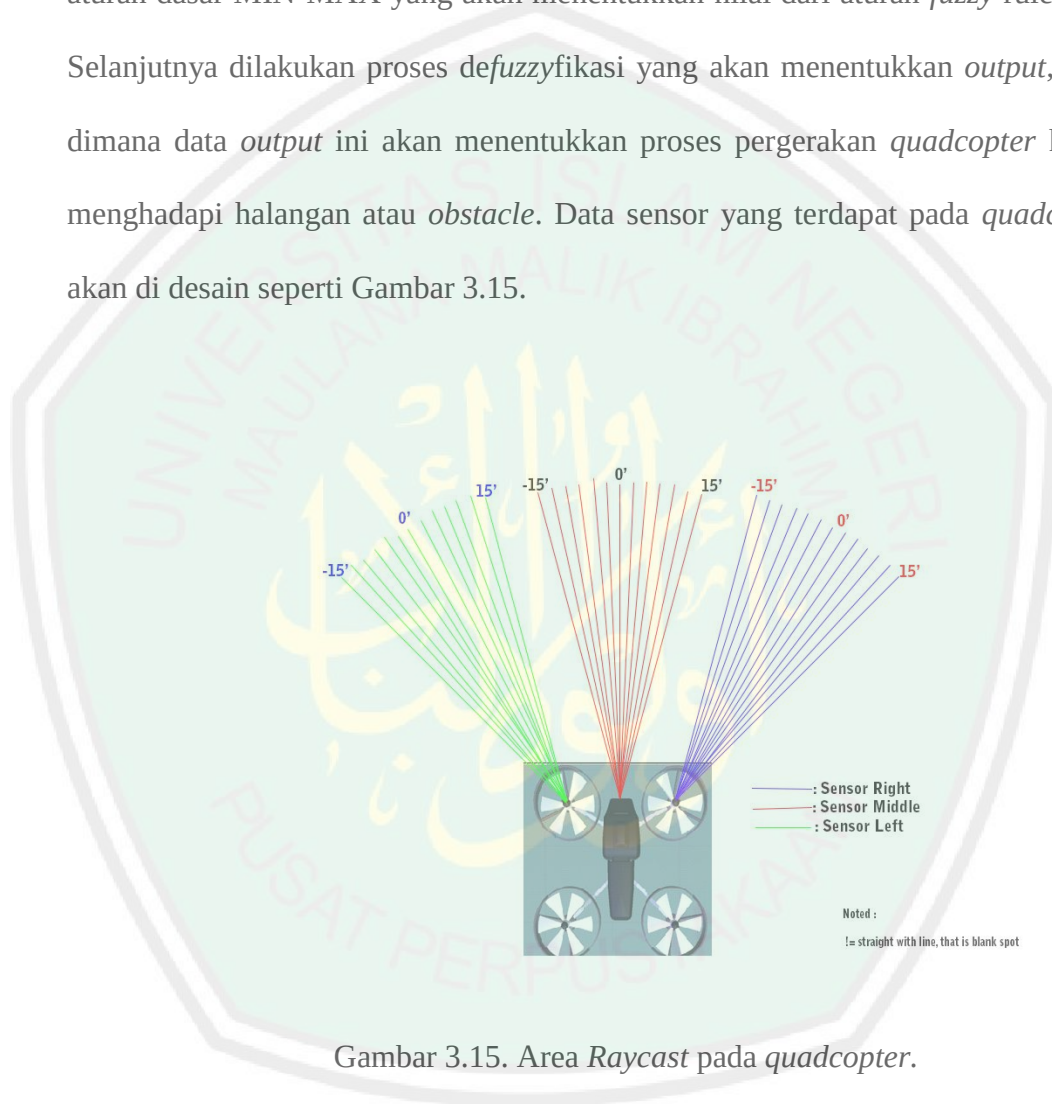
3.2 Perancangan Fuzzy

Logika *fuzzy* akan digunakan untuk mensimulasikan sensor *ultrasonic* dan *obstacle* yang ada pada *quadcopter* pada *game* simulasi sehingga sensor akan mendeteksi keberadaan objek dihadapannya sehingga dapat memberikan keadaan pergerakan *quadcopter*, sensor-sensor ini menggunakan pada *unity* sehingga dapat mendeteksi objek didepannya. Metode *fuzzy* yang digunakan adalah *fuzzy sugeno*.



Gambar 3.14. Alur Proses Fuzzy.

Pada Gambar 3.14 menjelaskan bahwa masukan data berasal dari data sensor yang terdiri dari sensor kiri, sensor tengah, dan sensor kanan. Berdasarkan *inputan* data tersebut kemudian di olah menjadi himpunan anggota *fuzzy*, selanjutnya akan dilakukan proses *inference* yang dimana pada proses ini terdapat aturan dasar MIN-MAX yang akan menentukan nilai dari aturan *fuzzy rule base*. Selanjutnya dilakukan proses defuzzyfikasi yang akan menentukan *output*, yang dimana data *output* ini akan menentukan proses pergerakan *quadcopter* ketika menghadapi halangan atau *obstacle*. Data sensor yang terdapat pada *quadcopter* akan di desain seperti Gambar 3.15.



Gambar 3.15. Area Raycast pada *quadcopter*.

Pada Gambar 3.15 dapat di ketahui bahwa sensor kanan berada pada posisi kemiringan ke kanan 30° dari depan, sensor tengah berada pada posisi 0° ke depan, sensor kiri berada pada posisi kemiringan ke kiri 30° dari depan. Masing-masing sensor memiliki radius deviasi pemancaran *raycast* -15° ke kiri dan 15° ke kanan

yang membentuk sudut 30° , dan memiliki 13 *raycast* dengan perbedaan *angle* 2.5° .

3.2.1 Variabel *Fuzzy*

Variabel yang menjadi himpunan keanggotaan *fuzzy* pada *game* simulasi *quadcopter* akan dibagi menjadi tiga variabel yaitu “Sensor Kiri”, “Sensor Tengah”, “Sensor Kanan”, dan memiliki variabel *output* yang memberikan aksi pergerakan *quadcopter* apabila menemukan objek dihadapannya.

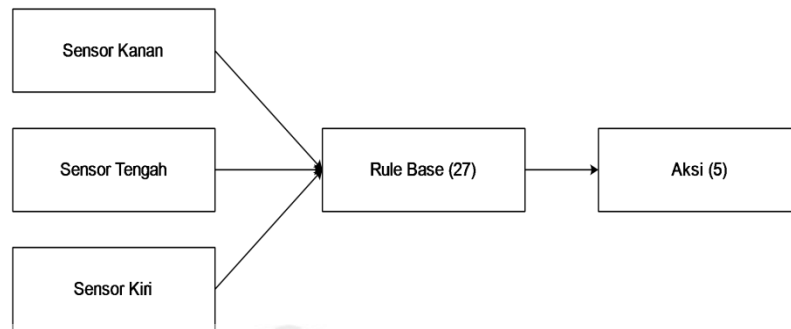
3.2.2 Nilai Linguistik

Dari empat variabel yang akan digunakan maka dapat diketahui nilai linguistiknya dan kehimpunan anggota masing-masing, sebagai berikut:

1. Sensor Kiri : “Jauh”, “Sedang”, “Dekat”.
2. Sensor Tengah : “Jauh”, “Sedang”, “Dekat”.
3. Sensor Kanan : “Jauh”, “Sedang”, “Dekat”.
4. *Output* Aksi : “Bergerak”, “Geser Kanan”, “Kurangi Kecepatan”, “Geser Kiri”, “Diam”.

3.2.3 *Fuzzyfikasi*

Fuzzyfikasi adalah proses pemetaan nilai *crisp* ke dalam himpunan *fuzzy*, dan menentukan derajat keanggotaannya. Secara garis besar akan dijelaskan pada gambar di bawah ini :



Gambar 3.16. *Fuzzy Interface System*.

Berdasarkan Gambar 3.16 di atas maka dapat dilakukan pemetaan sebagai berikut :

1. Sensor Kanan

Variabel kondisi yang terdapat pada jarak sensor kanan adalah dekat, sedang, jauh.



Gambar 3.17. Derajat Himpunan Keanggotaan Sensor Kanan

Pada Gambar 3.17 diketahui bahwa jarak sensor kanan diberikan rentang nilai 0-6, hal ini sesuai dengan hasil penelitian keakuratan sensor ultrasonik yaitu 0 – 300 yang diterapkan pada skala *unity*. Detail derajat keanggotaan sebuah sensor kanan dapat dilihat pada tabel di bawah ini :

Tabel 3.1. Derajat Himpunan Keanggotaan Sensor Kanan

No	Kondisi	Jarak(skala unity)
1.	Dekat	0 : 0.75 : 2
2.	Sedang	2 : 3.5 : 4.5
3.	Jauh	4.5 : 5.5 : 6

Fungsi keanggotaan untuk variabel sensor kanan adalah :

$$\mu_{\text{Dekat}} = \begin{cases} 1 : 0 \leq x \leq 0.75 \\ \frac{(2-x)}{(2-0.75)} : 0.75 < x < 2 \\ 0 : x \geq 2 \end{cases} \quad (3.1)$$

$$\mu_{\text{Sedang}} = \begin{cases} 0 : x \leq 2 \text{ atau } x \geq 4.5 \\ \frac{(x-2)}{(3.5-2)} : 2 < x \leq 3.5 \\ \frac{(4.5-x)}{(4.5-3.5)} : 3.5 < x < 4.5 \end{cases} \quad (3.2)$$

$$\mu_{\text{Jauh}} = \begin{cases} 0 : x \leq 4.5 \\ \frac{(x-4.5)}{(5.5-4.5)} : 4.5 < x < 5.5 \\ 1 : x \geq 5.5 \end{cases} \quad (3.3)$$

2. Sensor Tengah

Variabel kondisi yang terdapat pada jarak sensor tengah adalah dekat, sedang, jauh.



Gambar 3.18. Derajat Himpunan Keanggotaan Sensor Tengah.

Pada grafik di atas diketahui bahwa jarak sensor tengah diberikan rentang nilai 0-6, sesuai dengan penelitian sensor ultrasonik keakuratan sensor ultrasonik yaitu 0 – 300 yang diterapkan pada skala *unity*. Detail derajat keanggotaan sebuah sensor tengah dapat dilihat pada tabel di bawah ini

Tabel 3.2. Derajat Himpunan Keanggotaan Sensor Tengah

No	Kondisi	Jarak(<i>skala unity</i>)
1.	Dekat	0 : 0.75 : 2
2.	Sedang	2 : 3.5 : 4.5
3.	Jauh	4.5 : 5.5 : 6

Fungsi keanggotaan untuk variabel sensor Tengah adalah :

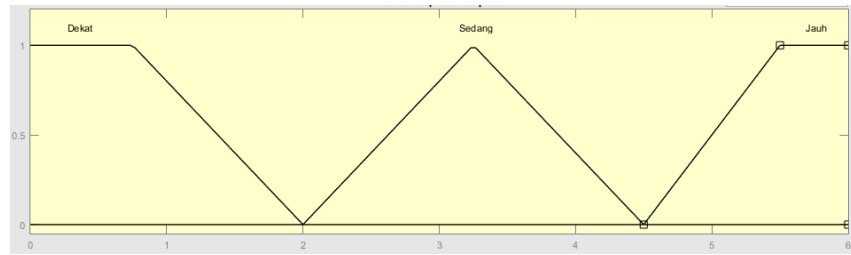
$$\mu_{\text{Dekat}} = \begin{cases} 1 : 0 \leq x \leq 0.75 \\ \frac{(2-x)}{(2-0.75)} : 0.75 < x < 2 \\ 0 : x \geq 2 \end{cases} \quad (3.4)$$

$$\mu_{\text{Sedang}} = \begin{cases} 0 : x \leq 2 \text{ atau } x \geq 4.5 \\ \frac{(x-2)}{(3.5-2)} : 2 < x \leq 3.5 \\ \frac{(4.5-x)}{(4.5-3.5)} : 3.5 < x < 4.5 \end{cases} \quad (3.5)$$

$$\mu_{\text{Jauh}} = \begin{cases} 0 : x \leq 4.5 \\ \frac{(x-4.5)}{(5.5-4.5)} : 4.5 < x < 5.5 \\ 1 : x \geq 5.5 \end{cases} \quad (3.6)$$

3. Sensor Kiri

Variabel kondisi yang terdapat pada jarak sensor kiri adalah dekat, sedang, jauh.



Gambar 3.19. Derajat Himpunan Keanggotaan Sensor Kiri

Pada grafik di atas diketahui bahwa jarak sensor kiri diberikan rentang nilai 0-6, keakuratan sensor ultrasonik yaitu 0 – 300 yang diterapkan pada skala *unity*. Detail derajat keanggotaan sebuah sensor kiri dapat dilihat pada tabel di bawah ini :

Tabel 3.3. Derajat Himpunan Keanggotaan Sensor Kiri

No	Kondisi	Jarak(skala <i>unity</i>)
1.	Dekat	0 : 0.75 : 2
2.	Sedang	2 : 3.5 : 4.5
3.	Jauh	4.5 : 5.5 : 6

Fungsi keanggotaan untuk variabel sensor kiri adalah :

$$\mu_{\text{Dekat}} = \begin{cases} 1 : 0 \leq x \leq 0.75 \\ \frac{(2-x)}{(2-0.75)} : 0.75 < x < 2 \\ 0 : x \geq 2 \end{cases} \quad (3.7)$$

$$\mu_{\text{Sedang}} = \begin{cases} 0 : x \leq 2 \text{ atau } x \geq 4.5 \\ \frac{(x-2)}{(3.5-2)} : 2 < x \leq 3.5 \\ \frac{(4.5-x)}{(4.5-3.5)} : 3.5 < x < 4.5 \end{cases} \quad (3.8)$$

$$\mu_{\text{Jauh}} = \begin{cases} 0 : x \leq 4.5 \\ \frac{(x-4.5)}{(5.5-4.5)} : 4.5 < x < 5.5 \\ 1 : x \geq 5.5 \end{cases} \quad (3.9)$$

4. Output Aksi

Dalam menentukan hasil *output* yang inferensi secara tegas (*crisp*), bagi masing-masing *rule* berdasarkan aturan *fuzzy* yang telah dibentuk, dimana untuk setiap $(Z_1, Z_2, Z_3, \dots, Z_n)$ bernilai konstan dengan menggunakan kenggotaan *singleton*.

- a. Bergerak : $Z_1=1$
- b. Geser Kanan : $Z_2=2$
- c. Kurangi Kecepatan : $Z_3=3$
- d. Geser Kiri : $Z_4=4$
- e. Diam : $Z_5=5$

3.2.4 Fuzzy Rule

Dalam langkah ini, bentuk umum dari aturan yang digunakan dalam fungsi implikasi adalah :

IF x is a AND y is b THEN z is C

Pada perhitungan menggunakan MIN, fungsi ini akan memotoni *output* himpunan *fuzzy*, *rule fuzzy* ini akan diterapkan pada *quadcopter* saat mendeteksi adanya halangan atau *obstacle* didepannya, implikasinya adalah sebagai berikut :

1. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Jauh) and (Sensor Kiri is Jauh) then (Aksi is Bergerak).*
2. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Jauh) and (Sensor Kiri is Sedang) then (Aksi is Lambat).*

3. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Jauh) and (Sensor Kiri is Dekat) then (Aksi is Geser Kanan).*
4. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Sedang) and (Sensor Kiri is Jauh) then (Aksi is Lambat).*
5. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Sedang) and (Sensor Kiri is Sedang) then (Aksi is Lambat).*
6. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Sedang) and (Sensor Kiri is Dekat) then (Aksi is Geser Kanan).*
7. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Dekat) and (Sensor Kiri is Jauh) then (Aksi is Geser Kanan).*
8. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Dekat) and (Sensor Kiri is Sedang) then (Aksi is Geser Kanan).*
9. *If (Sensor Kanan is Jauh) and (Sensor Tengah is Dekat) and (Sensor Kiri is Dekat) then (Aksi is Geser Kanan).*
10. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Jauh) and (Sensor Kiri is Jauh) then (Aksi is Lambat).*
11. *If (Sensor Kanan is Sedang) and (Sensor Tengah is jauh) and (Sensor Kiri is Sedang) then (Aksi is Lambat).*
12. *If (Sensor Kanan is Sedang) and (Sensor Tengah is jauh) and (Sensor Kiri is Dekat) then (Aksi is Geser Kanan).*
13. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Sedang) and (Sensor Kiri is Jauh) then (Aksi is Lambat).*
14. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Sedang) and (Sensor Kiri is Sedang) then (Aksi is Lambat).*

15. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Sedang) and (Sensor Kiri is Dekat) then (Aksi is Geser Kanan).*
16. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Dekat) and (Sensor Kiri is Jauh) then (Aksi is Geser Kanan).*
17. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Dekat) and (Sensor Kiri is Sedang) then (Aksi is Geser Kanan).*
18. *If (Sensor Kanan is Sedang) and (Sensor Tengah is Dekat) and (Sensor Kiri is Dekat) then (Aksi is Geser Kanan).*
19. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Jauh) and (Sensor Kiri is Jauh) then (Aksi is Geser Kiri).*
20. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Jauh) and (Sensor Kiri is Sedang) then (Aksi is Geser Kiri).*
21. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Jauh) and (Sensor Kiri is Dekat) then (Aksi is Geser Kiri).*
22. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Sedang) and (Sensor Kiri is Jauh) then (Aksi is Geser Kiri).*
23. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Sedang) and (Sensor Kiri is Sedang) then (Aksi is Geser Kiri).*
24. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Sedang) and (Sensor Kiri is Dekat) then (Aksi is Geser Kiri).*
25. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Dekat) and (Sensor Kiri is Jauh) then (Aksi is Geser Kiri).*
26. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Dekat) and (Sensor Kiri is Sedang) then (Aksi is Geser Kiri).*

27. *If (Sensor Kanan is Dekat) and (Sensor Tengah is Dekat) and (Sensor Kiri is Dekat) then (Aksi is Diam).*

3.2.5 Implikasi dan Contoh Perhitungan

Berdasarkan *fuzzy rule* di atas, maka dapat dilakukan proses fungsi implikasi dan dimasukkan ke dalam sistem. Sebagai contoh, *input* mendeteksi adanya jarak dari *obstacle*, misal sensor kanan mendeteksi jarak 0.75 skala *unity*, sensor tengah mendeteksi jarak 2.1 skala *unity*, sensor kiri mendeteksi jarak 4.6 skala *unity*. Jika telah diketahui nilai *inputan*, maka langkah selanjutnya adalah menentukan nilai keanggotaan, maka dapat digunakan rumus sebagai berikut:

Jika $x \leq a$ atau $x \geq c$ bernilai 0

Jika $a \leq x \leq b$ maka $(x-a)/(b-a)$

Jika $b \leq x \leq c$ maka $(c-x)/(c-b)$

Dimana a adalah nilai minimum, b adalah nilai sedang, dan c adalah nilai maksimum serta x adalah nilai *inputan*. Maka berdasarkan rumusan tersebut sensor kanan masuk dalam kategori jarak dekat karena memiliki nilai 0.75 skala *unity* sesuai dengan perhitungan 3.10 berikut :

$$\mu_{\text{Dekat}} = \frac{2-0.75}{2-0.75} = \frac{1.25}{1.25} = 1 \quad (3.10)$$

Telah didapatkan hasil di atas maka untuk sensor kanan jarak jauh dan sedang adalah sama dengan 0. Kemudian selanjutnya adalah perhitungan

untuk sensor tengah yang memiliki nilai 2.1 skala *unity* maka termasuk dalam kategori jarak sedang berdasarkan dengan persamaan 3.11 berikut :

$$\mu_{\text{Sedang}} = \frac{4.5-2.1}{4.5-2} = \frac{2.4}{2.5} = 0.9 \quad (3.11)$$

Karena nilainya *input* 2.1 skala *unity* berada pada kategori sedang, maka nilai jauh dan dekat adalah sama dengan 0. Kemudian selanjutnya adalah untuk sensor kiri yang dimana memiliki nilai 4.6 skala *unity* termasuk dalam kategori sedang, sesuai dengan rumus perhitungan di bawah ini.

$$\mu_{\text{Jauh}} = \frac{4.6-4.5}{5.5-4.5} = \frac{0.1}{1} = 0.1 \quad (3.12)$$

Karena nilai *inputannya* adalah 4.6 maka termasuk dalam kategori jauh yang berarti nilai sedang dan dekat adalah sama dengan 0. Setelah mengetahui nilai keanggotaan dari masing-masing sensor maka proses selanjutnya adalah aplikasi fungsi implikasi menggunakan metode min dengan rumus berikut :

$$\alpha_{\text{predikatRn}} = \min (\mu_{sf}(x_i); \mu_{kf}(x_i)) \quad (3.13)$$

1. $\alpha_{\text{predikatR1}} = \min (1, 0, 0) = 0$
2. $\alpha_{\text{predikatR2}} = \min (1, 0, 0.1) = 0$
3. $\alpha_{\text{predikatR3}} = \min (1, 0, 0) = 0$
4. $\alpha_{\text{predikatR4}} = \min (1, 0.9, 0) = 0$
5. $\alpha_{\text{predikatR5}} = \min (1, 0.9, 0) = 0$
6. $\alpha_{\text{predikatR6}} = \min (1, 0.9, 0.1) = 0.1$
7. $\alpha_{\text{predikatR7}} = \min (1, 0, 0) = 0$

8. $\alpha_{predikatR8} = \min (1, 0, 0.1) = 0$
9. $\alpha_{predikatR9} = \min (1, 0, 0) = 0$
10. $\alpha_{predikatR10} = \min (0, 0, 0) = 0$
11. $\alpha_{predikatR11} = \min (0, 0, 0.1) = 0$
12. $\alpha_{predikatR12} = \min (0, 0, 0) = 0$
13. $\alpha_{predikatR13} = \min (0, 0, 0) = 0$
14. $\alpha_{predikatR14} = \min (0, 0.9, 0.1) = 0$
15. $\alpha_{predikatR15} = \min (0, 0.9, 0) = 0$
16. $\alpha_{predikatR16} = \min (0, 0, 0) = 0$
17. $\alpha_{predikatR17} = \min (0, 0, 0.1) = 0$
18. $\alpha_{predikatR18} = \min (0, 0, 0) = 0$
19. $\alpha_{predikatR19} = \min (0, 0, 0) = 0$
20. $\alpha_{predikatR20} = \min (0, 0, 0.1) = 0$
21. $\alpha_{predikatR21} = \min (0, 0, 0) = 0$
22. $\alpha_{predikatR22} = \min (0, 0.9, 0) = 0$
23. $\alpha_{predikatR23} = \min (0, 0.9, 0.1) = 0$
24. $\alpha_{predikatR24} = \min (0, 0.9, 0) = 0$
25. $\alpha_{predikatR25} = \min (0, 0, 0) = 0$
26. $\alpha_{predikatR26} = \min (0, 0, 0.1) = 0$
27. $\alpha_{predikatR27} = \min (0, 0, 0) = 0$

3.2.6 Defuzzyfikasi

Setelah mengetahui nilai keseluruhan dari fungsi implikasi tersebut maka langkah selanjutnya adalah melakukan komposisi aturan dengan menggunakan metode *weighted average* dengan persamaan 3.14 berikut :

$$Z = \frac{(\alpha - \text{predikat}_1 * Z_1) + (\alpha - \text{predikat}_2 * Z_2) + \dots + (\alpha - \text{predikat}_n * Z_n)}{(\alpha - \text{predikat}_1) + (\alpha - \text{predikat}_2) + \dots + (\alpha - \text{predikat}_n)} \quad (3.14)$$

Maka, karena diketahui nilai lainnya adalah sama dengan 0, maka langsung saja mengambil nilai *fuzzy rule* ke 6 dengan nilai 0.1, yang menghasilkan aksi *output* adalah geser kanan dengan nilai *konstanta* adalah 4 maka penyelesaiannya pada persamaan 3.15 berikut.

$$Z = \frac{0.1 \times 4}{0.1} = 4 \quad (3.15)$$

Dari hasil komposisi aturan menghasilkan nilai 4 dengan *output* “Geser Kanan”. Jadi *input* dari variabel sensor kanan yang mendeteksi jarak 0.75 skala *unity*, variabel sensor tengah yang mendeteksi jarak 2.1 skala *unity*, variabel sensor kiri yang mendeteksi jarak 4.6 skala *unity* menghasilkan *output* Z sama dengan 3 berupa perintah “Geser Kanan”.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi Desain Sistem

Pada bab ini membahas tentang implementasi dari perencanaan yang telah dibuat, serta melakukan pengujian terhadap *game* untuk mengetahui apakah *game* tersebut telah berjalan sesuai dengan yang diharapkan.

4.1.1 Kebutuhan Perangkat Keras

Perangkat keras yang diperlukan untuk mengimplementasikan perangkat keras dari aplikasi *game* pada tabel 4.1 berikut.

Tabel 4.1. Kebutuhan Perangkat Keras

No	Perangkat Keras	Spesifikasi
1.	<i>Processor</i>	Intel® Core™ i7-4720HQ CPU @ 2.60 GHz
2.	<i>RAM</i>	8 GB
3.	<i>VGA</i>	NVIDIA GeForce 940M
4.	<i>HDD</i>	1 TB
5.	<i>Monitor</i>	14"
6.	<i>Speaker</i>	On
7.	<i>Mouse & Keyboard</i>	On

Pada Tabel 4.1 menjelaskan bahwa perangkat keras yang digunakan dalam penelitian ini menggunakan *laptop* yang berspesifikasi seperti di atas, yang disambungkan kepada *game* simulasi *x-drone* untuk menerima sinyal-sinyal *inputan* sebagai pergerakan dari *remote control* melalui *keyboard*.

4.1.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang diperlukan untuk mengimplementasikan *game* simulasi *X-Drone* pada tabel 4.2 berikut.

Tabel 4.2. Kebutuhan Perangkat Lunak

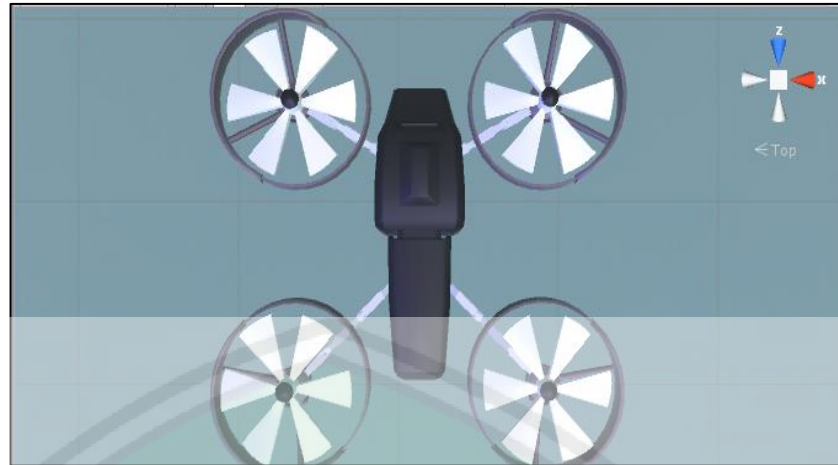
No	Perangkat Lunak	Spesifikasi
1.	Sistem Operasi	<i>Windows 10</i>
2.	<i>Game Engine</i>	<i>Unity 3D</i>
3.	Konsep desain 2D	<i>Adobe Photoshop CS6</i>
4.	Desain 3D	<i>Blender 3D</i>
5.	<i>Script Writer</i>	<i>MonoDevelop</i>

4.2 Tampilan Karakter

Pada Gambar 4.1 dan Gambar 4.2 merupakan tampilan karakter *X-Drone* dari samping dan atas, karakter *X-Drone* ini dibuat sesuai berbeda dengan rancangan awalnya, akan tetapi masih tetap dengan 4 *rotor* untuk sayapnya.



Gambar 4.1. Tampilan *X-Drone* dari samping



Gambar 4.2. Tampilan *X-Drone* dari Atas

4.3 Tampilan *Game*

Pembuatan tampilan *game* ini, melibatkan beberapa komponen *interface* pada *game* simulasi *X-Drone* yaitu :

4.3.1 Tampilan *Splash Screen*

Pada Gambar 4.3 merupakan tampilan dari *splash screen* yang berisikan nama karakter dari *game* tersebut yaitu *X-Drone* dan sesuai pada rancangan awalnya. Tampilan *splash screen* muncul pertama kali ketika pemain menjalankan *game* simulasi *X-Drone*.



Tampilan 4.3. Tampilan *Splash Screen*

4.3.2 Tampilan Main Menu

Pada Gambar 4.4 merupakan tampilan dari *main menu* pada *game simulasi X-Drone* yang berisikan dua tombol yaitu *Start Game* dan *Exit Game*. Tombol *main menu* ketika di klik akan menuju ke *Loading Bar Screen*, sedangkan tombol *exit* ketika di klik akan keluar dari *game*.



Gambar 4.4. Tampilan Main Menu

4.3.3 Tampilan Loading Bar Screen

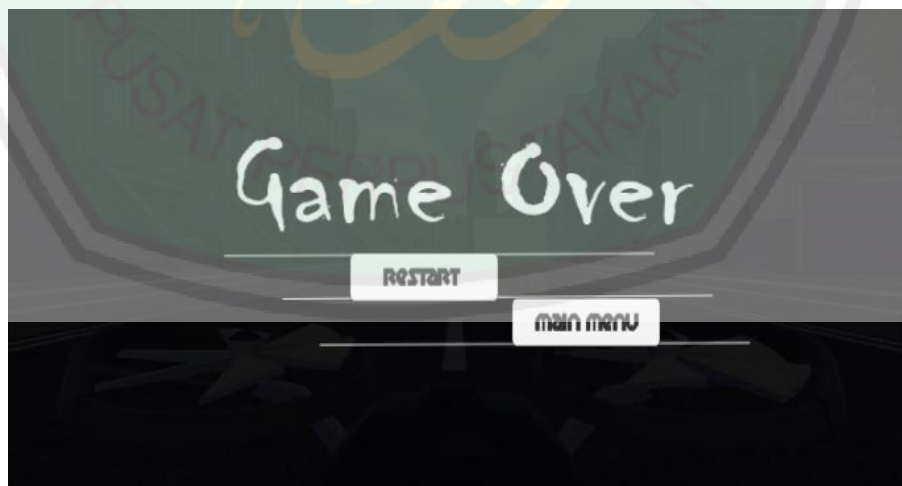
Pada Gambar 4.5 merupakan tampilan *loading bar screen* yang menampilkan nama dari *Game X-Drone simulation* dan *loading bar*, yang dimana pada tampilan ini *loading bar* akan mengextract seluruh *asset* ke dalam *Game* tersebut agar dapat berjalan dengan baik. Tampilan ini muncul setelah pemain menekan *start Game* pada tampilan *main menu*, setelah dari *loading bar screen* maka tampilan awal permainan. Yang dimana tampilan ini memiliki *side bar* yang bergeser sesuai dengan banyak angka yang didapat, di bawah tampilan tersebut terdapat pesan yang dapat berubah-ubah secara acak.



Gambar 4.5. Tampilan *Loading Bar Screen*

4.3.4 Tampilan *Canvas Over Game*

Tampilan Gambar 4.6 merupakan tampilan dari *over game*, tampilan ini muncul ketika pemain keluar terbang melebihi frekuensi dan atau keluar dari zona *map*. Pemain dapat memilih tombol *restart* untuk kembali memainkan *game* dan memilih tombol *main menu* untuk keluar dari *game* menuju ke *Main Menu*.



Gambar 4.6. Tampilan *Game Over*

4.3.5 Tampilan Canvas Win Game

Tampilan Gambar 4.7 merupakan tampilan *win game*, tampilan ini akan muncul ketika pemain berhasil mendaratkan *x-drone* pada titik pendaratan yang berbentuk H. Pemain dapat memilih tombol *restart* untuk kembali memainkan *game* dan memilih tombol *main menu* untuk keluar dari *game* menuju ke *Main Menu*.



Gambar 4.7. Tampilan Win Game

4.3.6 Tampilan Awal Game

Pada awal terlihat pada suasana perkotaan yang dikelilingi berbagai perumahan yang dimana nanti *X-Drone* dimulai pada titik ini, seperti pada Gambar 4.8 berikut.



Gambar 4.8. Tampilan Awal *Game*

4.3.7 Tampilan *Map Game*

Pada Gambar 4.9 yang merupakan peta *game* yang dimana tampilan pada *game* dirancang memiliki titik penerbangan dan titik pendaratan. Area ini terdiri oleh beberapa *environment* seperti gunung, rumput atau tumbuhan beserta karakter-karakter animasi lainnya.



Gambar 4.9. Tampilan *Map Game*

4.4 Implementasi Algoritma *Fuzzy Sugeno*

Berdasarkan rancangan *fuzzy* yang telah dibuat sebelumnya, algoritma *fuzzy sugeno* di implementasikan pada pergerakan *X-Drone* menjauhi *obstacle*. Variabel yang digunakan dalam algoritma ini ada empat, yaitu tiga variabel *input* dan satu variabel *output*, variabel *output* memiliki lima variasi hasil tergantung variasi *input* yang ada. *Output* yang dihasilkan oleh *fuzzy sugeno* dari *defuzzyfikasi* berupa *konstanta* sehingga mewakili aksi *drone* tersebut.

Dalam mengimplementasikan algoritma *fuzzy sugeno*, langkah pertama yang harus dilakukan adalah menentukan dan memodelkan variabel *fuzzy*, variabel yang digunakan yaitu “Sensor Kanan”, “Sensor Tengah”, “Sensor Kiri”,. Berikut implementasi pemodelan variabel *fuzzy* dan pengambilan data :

```
[Header("Sensors")]
Vector3 sensorStartPos = transform.position;
public float sensorLength = 4;
public float frontSensorPosition = 0.5f;
Public float frontSideSensorPosition = 0.5f
public float frontSensorAngle = 30;
public Vector3 targetposition;
public Vector3 dir;
float jaraksensorkanan;
float jaraksensortengah;
float jaraksensorkiri;
```

Gambar 4.10. Pengambilan data sensor jarak

Pada Gambar 4.10 merupakan pengambilan data-data untuk mengambil nilai variabel jarak sehingga bisa ditentukan nilai keanggotaan dan nilai implikasinya. Selanjutnya adalah menentukan nilai derajat keanggotaan terbesar dari masing-masing *variable* yang dimana menggunakan fungsi implikasi dengan metode *metode evarage*.


```

void sensorkanan () {
    if (hit.distance > 0.75 && hit.distance < 2) {
        jaraksensorkanan = 3;
    } else if (hit.distance > 2 && hit.distance < 4.5) {
        jaraksensorkanan = 2;
    } else if (hit.distance > 4) {
        jaraksensorkanan = 1;
    }
}

void sensortengah () {
    if (hit.distance > 0.75 && hit.distance < 2) {
        jaraksensortengah = 3;
    } else if (hit.distance > 2 && hit.distance < 4.5) {
        jaraksensortengah = 2;
    } else if (hit.distance > 4) {
        jaraksensortengah = 1;
    }
}

void sensorkiri() {
    if (hit.distance > 0.75 && hit.distance < 2) {
        jaraksensorkiri = 3;
    } else if (hit.distance > 2 && hit.distance < 4.5) {
        jaraksensorkiri = 2;
    } else if (hit.distance > 4) {
        jaraksensorkiri = 1;
    }
}

```

Gambar 4.11. Gambar *Fuzzyfikasi dan Implikasi*

Pada Gambar 4.11 di atas menjelaskan pada nilai jarak tersebut akan menghasilkan nilai jarak, 1, 2, dan 3. Dimana 1 merupakan *konstanta* dari jauh, 2 merupakan *konstanta* dari sedang, 3 merupakan *konstanta* dari dekat. Kemudian setelah ditemukan nilai-nilai implikasi yang dimasukkan ke dalam *fuzzy rule*, maka akan dilakukan proses *defuzzyfikasi* yang mana akan menentukan aksi pada setiap *fuzzy rule*. proses *fuzzy rule* yaitu proses pencocokan nilai dari hasil sebelumnya akan dicocokkan sehingga akan muncul sebuah *output* berupa aksi menghindari *obstacle* pada *X-Drone*.

```

void fuzzyrule () {
    if (jaraksensorkanan == 1 && jaraksensortengah == 1 && jaraksensorkiri == 1) {
        Move ();
    } else if (jaraksensorkanan == 1 && jaraksensortengah == 1 && jaraksensorkiri == 2) {
        lambat ();
    } else if (jaraksensorkanan == 1 && jaraksensortengah == 1 && jaraksensorkiri == 3)

```

```

{
    geserkanan ();
} else if (jaraksensorkanan == 1 && jaraksensortengah == 2 && jaraksensorkiri == 1)
{
    lambat ();
} else if (jaraksensorkanan == 1 && jaraksensortengah == 2 && jaraksensorkiri == 2)
{
    lambat ();
} else if (jaraksensorkanan == 1 && jaraksensortengah == 2 && jaraksensorkiri == 3)
{
    geserkanan ();
} else if (jaraksensorkanan == 1 && jaraksensortengah == 3 && jaraksensorkiri == 1)
{
    geserkanan ();
} else if (jaraksensorkanan == 1 && jaraksensortengah == 3 && jaraksensorkiri == 2)
{
    geserkanan ();
} else if (jaraksensorkanan == 1 && jaraksensortengah == 3 && jaraksensorkiri == 3)
{
    geserkanan ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 1 && jaraksensorkiri == 1)
{
    lambat ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 1 && jaraksensorkiri == 2)
{
    lambat ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 1 && jaraksensorkiri == 3)
{
    geserkanan ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 2 && jaraksensorkiri == 1)
{
    lambat ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 2 && jaraksensorkiri == 2)
{
    lambat ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 2 && jaraksensorkiri == 3)
{
    geserkanan ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 3 && jaraksensorkiri == 1)
{
    geserkanan ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 3 && jaraksensorkiri == 2)
{
    geserkanan ();
} else if (jaraksensorkanan == 2 && jaraksensortengah == 3 && jaraksensorkiri == 3)
{
    geserkanan ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 1 && jaraksensorkiri == 1)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 1 && jaraksensorkiri == 2)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 1 && jaraksensorkiri == 3)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 2 && jaraksensorkiri == 1)

```

```

{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 2 && jaraksensorkiri == 2)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 2 && jaraksensorkiri == 3)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 3 && jaraksensorkiri == 1)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 3 && jaraksensorkiri == 2)
{
    geserkiri ();
} else if (jaraksensorkanan == 3 && jaraksensortengah == 2 && jaraksensorkiri == 2)
{
    diam ();
}
}

```

Gambar 4.12. Implementasi *Fuzzy Rule*

Dari penjelasan bab sebelumnya ada beberapa aksi yang dilakukan oleh *X-Drone* saat menghadapi *obstacle*, berikut adalah *source code* programnya.

a. Berjalan

Aksi berjalan ini diberikan pada *X-Drone* dalam menghadapi *obstacle* dengan melakukan pergerakan berdasarkan *input* dari *remote control*. Aksi ini juga dilakukan saat sensor mendeteksi jarak sensor ke *obstacle* sama dengan jauh, maka tetap melakukan aksi berjalan.

```

void Move(){
    RotatingWing ();
    carihit ();
    MovementUpDown ();
    MovementForward ();
    Rotation ();
    ClampingSpeedValues ();
    Swerwe ();
    DroneSound ();
    ourDrone.AddRelativeForce (Vector3.up * upForce);
    ourDrone.rotation = Quaternion.Euler (
    new Vector3 (tiltAmountForward, currentYRotation, tiltAmountSideWays
    ))}

```

Gambar 4.13. *Source Code* Bergerak *X-Drone*

b. Kurangi Kecepatan

Mengurangi kecepatan dilakukan *X-Drone* ketika terdapat halangan di depannya sehingga sebelum tabrakan terjadi, maka dilakukan pengurangan kecepatan sampai akhirnya jaraknya dinyatakan dekat *X-Drone* akan melakukan aksi diam sebelum menabrak halangan yang ada didepannya, untuk mengatur aksi ini dapat dilihat pada Gambar 4.14.

```
void kurangi kecepatan () {
    if (Input.GetAxis("Vertical") != 0) {
        movementForwardSpeed = 15f;
        tiltAmountForward = 0;
        ourDrone.AddRelativeForce (Vector3.forward * 0.5f * movementForwardSpeed);
        tiltAmountForward = Mathf.SmoothDamp (tiltAmountForward, 20 * nilai_output, ref tiltVelocityForward, 0.1f);
    }
}

void lambat () {
    cariHit ();
    kurangiKecepatan ();
    RotatingWing ();
    MovementUpDown ();
    Rotation ();
    ClampingSpeedValues ();
    Swerwe ();
    DroneSound ();
    ourDrone.AddRelativeForce (Vector3.up * upForce);
    ourDrone.rotation = Quaternion.Euler (
        new Vector3 (tiltAmountForward, currentYRotation, tiltAmountSideWays)
    );
}
```

Gambar 4.14. Source Code Mengurangi Kecepatan pada *X-Drone*

c. Geser Kanan

Geser kanan adalah aksi dimana *X-Drone* memindah posisinya ke arah kanan atau *positive x*, sehingga dapat menghindari *obstacle*, pada jarak sensor kiri atau tengah memiliki nilai sama dengan dekat.

```
private bool geserKanan1 = false;
void bergeserKanan () {
    if (geserKanan1 == true) {
        upForce = 150;
        ourDrone.AddRelativeForce (Vector3.right * 0.5f * sideMovementAmount);
        tiltAmountSideWays = Mathf.SmoothDamp (tiltAmountSideWays, -
```

```

20 * 0.5f, ref tiltAmountVelocity, 0.1f);
    } else {
        tiltAmountSideWays = Mathf.SmoothDamp (tiltAmountSideWays, 0, ref tilt
AmountVelocity, 0.1f);
    }
}
void geserkanan () {
    geserkanan1 = true;
    carihit ();
    bergeserkanan ();
    RotatingWing ();
    MovementUpDown ();
    Rotation ();
    ClampingSpeedValues ();
    Swerwe ();
    DroneSound ();
    ourDrone.AddRelativeForce (Vector3.up * upForce);
    ourDrone.rotation = Quaternion.Euler (
        new Vector3 (tiltAmountForward, currentYRotation, tiltAmountSideWays));
}

```

Gambar 4.15. Source Code Geser Kanan

d. Geser Kiri

Geser kiri adalah aksi dimana *X-Drone* memindah posisinya ke arah kiri atau *negatif x*, sehingga dapat menghindari *obstacle*, pada jarak sensor kanan atau tengah yang memiliki nilai sama dengan dekat.

```

private bool geserkiril = false;
void bergeserkiri () {
    if (geserkiril == true) {
        upForce = 150;
        ourDrone.AddRelativeForce (Vector3.right * (-
0.5f) * sideMovementAmount);
        tiltAmountSideWays = Mathf.SmoothDamp (tiltAmountSideWays, -20 * (-
0.5f), ref tiltAmountVelocity, 0.1f);
    } else {
        tiltAmountSideWays = Mathf.SmoothDamp (tiltAmountSideWays, 0, ref tiltA
mountVelocity, 0.1f);
    }
}
void geserkiri () {
    geserkiril = true;
    carihit ();
    bergeserkiri ();
    RotatingWing ();
    DroneSound ();
    ourDrone.AddRelativeForce (Vector3.up * upForce);
    ourDrone.rotation = Quaternion.Euler (
        new Vector3 (tiltAmountForward, currentYRotation, tiltAmountSideWays);
}

```

Gambar 4.16. Source Code Geser Kiri

e. Diam

Diam merupakan proses ketika *X-Drone* memiliki kecepatan sama dengan 0 untuk maju ke depan. Dengan kecepatan tersebut diakibatkan ketiga sensor mendeteksi jarak sama dengan dekat maka *X-Drone* diam yang artinya tidak mendapatkan *input* untuk maju lagi karena jika maju maka akan langsung menabrak *obstacle*.

```
private bool diam1 = false;
void diam () {
    movementForwardSpeed = 0f;
    tiltAmountForward = 0;
    if (diam1 == true ) {
        tiltAmountForward = Mathf.SmoothDamp (tiltAmountForward, 0 , ref titltVelocityForward, 0.1f);
    }
}
void diam () {
    diam1 = true;
    carihit ();
    diam ();
    RotatingWing ();
    MovementUpDown ();
    Rotation ();
    ClampingSpeedValues ();
    Swerwe ();
    DroneSound ();
    ourDrone.AddRelativeForce (Vector3.up * upForce);
    ourDrone.rotation = Quaternion.Euler (
        new Vector3 (tiltAmountForward, currentYRotation, tiltAmountSideWays));
}
```

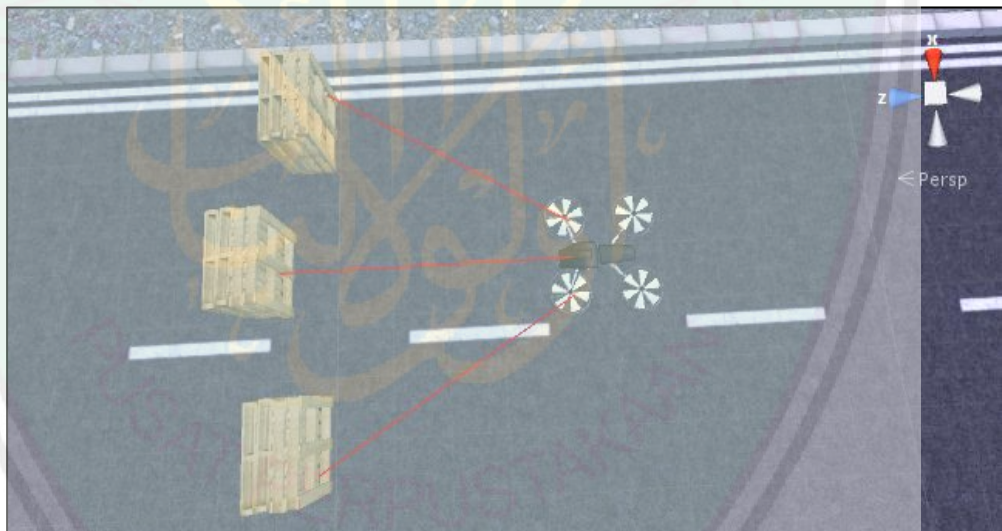
Gambar 4.17. Source Code X-Drone Diam

4.5 Uji Coba

Uji coba dilakukan untuk mengetahui hasil dari implementasi metode dan pergerakan apakah dapat berjalan sesuai yang diharapkan atau sebaliknya. Terdapat uji coba yang dilakukan, yaitu uji coba sensor jarak, uji coba bergerak, uji coba *rule*, uji coba menghindari halangan, dan uji coba mengurangi kecepatan, dll.

4.5.1 Uji coba Sensor

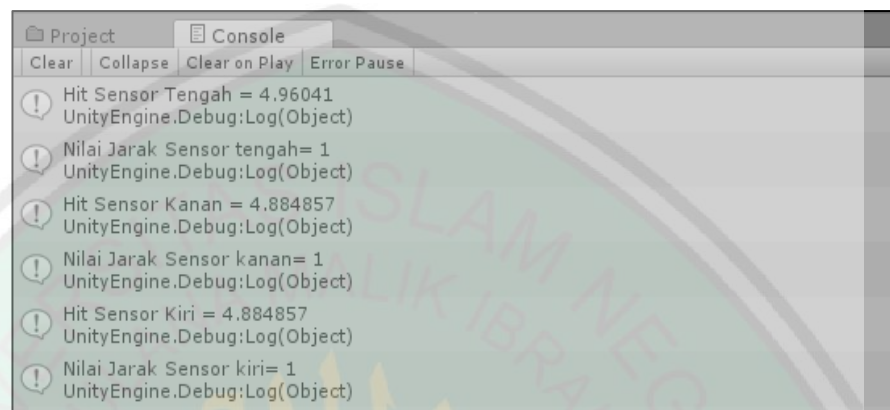
Pengujian terhadap sensor sangat penting untuk dilakukan karena sensor merupakan unsur yang paling penting pada sistem yaitu menjadi *input* dari sistem, tanpa adanya *input* dari sensor maka sensor tidak dapat melakukan pergerakan saat bertemu dengan halangan atau *obstacle*. Pengujian ini dilakukan untuk mengetahui masing-masing sensor apakah dapat bekerja dengan baik atau tidak. Pada penelitian ini menggunakan 3 sensor yaitu sensor kanan, sensor tengah, dan sensor kiri yang di simulasikan menggunakan *raycast* pada *unity* sehingga bisa mendapatkan jarak dari *X-Drone* ke halangan seperti pada Gambar 4.18 berikut ini.



Gambar 4.18. Tampilan Deteksi Sensor *X-Drone* ke *Obstacle*

Dari posisi sensor pada Gambar 4.18, diketahui bahwa masing-masing sensor memiliki radius sensor pemancaran yaitu 30° dengan 13 *raycast* dengan perbedaan *angle* 0.5° , kemudian terdapat perbedaan penempatan sensor kanan, sensor kiri, dan sensor tengah yaitu sensor kanan berada pada posisi kemiringan 30° ke kanan dari depan, sensor tengah berada pada posisi 0° ke

depan, dan sensor kiri berada pada posisi kemiringan 30° ke kiri dari depan. Kemudian di uji apakah sensor dapat menerima *inputan* atau tidak maka dapat di *debug* atau dicetak jarak dari *X-Drone* ke *obstacle* yang muncul pada *console* di *unity*, seperti pada Gambar 4.19 berikut.



Gambar 4.19. Hasil Uji coba Sensor Jarak

Dari hasil pengujian sensor menggunakan *debug* yang dicetak pada *console* dapat diketahui bahwa ketiga sensor yang di simulasikan menggunakan *raycast* dapat bekerja dengan baik sehingga dapat mendeteksi *obstacle* yang berada didepannya. Selanjutnya adalah dilakukan percobaan untuk mengetahui seberapa jauh dapat mendeteksi halangan, untuk pendeteksian jarak ini maka hal tergantung dengan *hit.distance* yang diberikan pada *source code* programnya. Pada *source code* Gambar. 4.10, *sensor length* atau panjang sensor dari *X-Drone* ke *obstacle* sama dengan lima, apabila lebih dari lima maka sensor atau *raycast* tidak mendeteksi *Obstacle* tersebut.

4.5.2 Uji Coba *X-Drone* bergerak

Pengujian ini dilakukan untuk melihat *X-Drone* dapat bergerak sesuai dengan *input* dari *remote control* atau tidak. Dan hasilnya adalah *X-Drone* telah

bergerak sesuai dengan *input*annya dan disesuaikan seperti *quadcopter* pada wujud aslinya, bahkan memiliki derajat kemiringan misalnya saat bergerak ke samping kiri atau kanan maupun maju kedepan dan mundur ke belakang seperti pada Gambar 4.20 yang merupakan *X-Drone* saat tidak menerima *input* apapun dan Gambar 4.21 saat *X-Drone* menerima *input* untuk geser ke kanan.



Gambar 4.20. Gambar *X-Drone* saat Diam



Gambar 4.21. Gambar *X-Drone* saat bergerak ke samping kanan

4.5.3 Uji Coba *Rule X-Drone*

Uji coba dilakukan untuk mengetahui apakah *rule* yang diberikan pada *X-Drone* dapat bekerja dengan baik atau tidak. Untuk masing-masing jarak diberikan *konstanta*, yaitu jarak jauh sama dengan satu, jarak sedang sama

dengan dua, jarak dekat sama dengan tiga. Untuk itu dilakukan uji coba pada masing-masing *rule* yang dimana memiliki sebanyak 27 *rule* sebagai berikut :

1. Jauh – Jauh – Jauh



Gambar 4.22. Gambar Percobaan *Rule* Jauh – Jauh – Jauh

2. Jauh – Jauh – Sedang



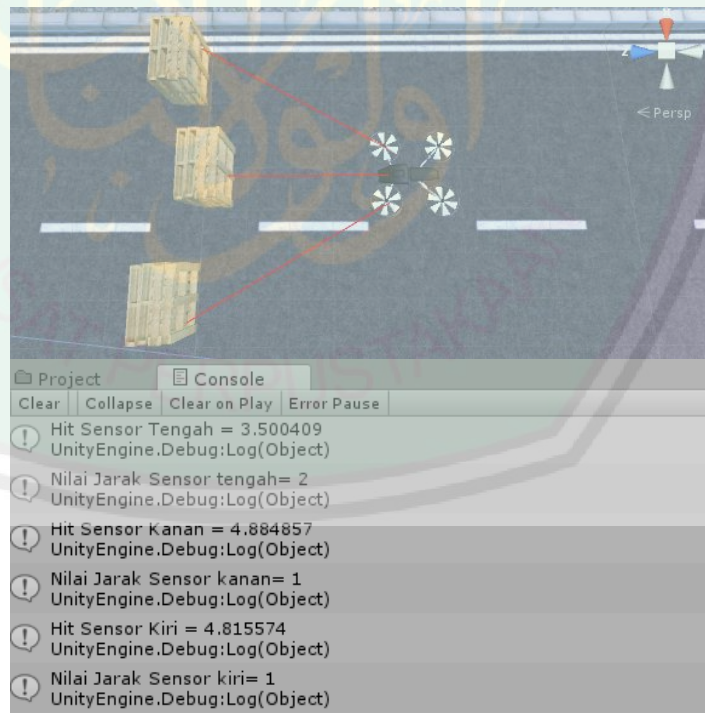
Gambar 4.23. Gambar Percobaan *Rule* Jauh – Jauh – Sedang

3. Jauh – Jauh – Dekat



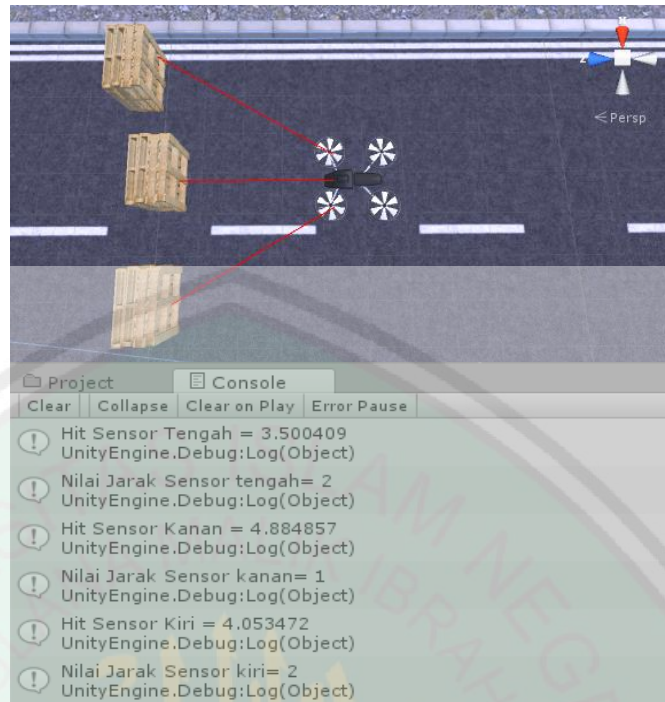
Gambar 4.24. Gambar Percobaan *Rule Jauh – Jauh – Dekat*

4. Jauh – Sedang – Jauh



Gambar 4.25. Gambar Percobaan *Rule Jauh – Sedang – Jauh*

5. Jauh – Sedang – Sedang



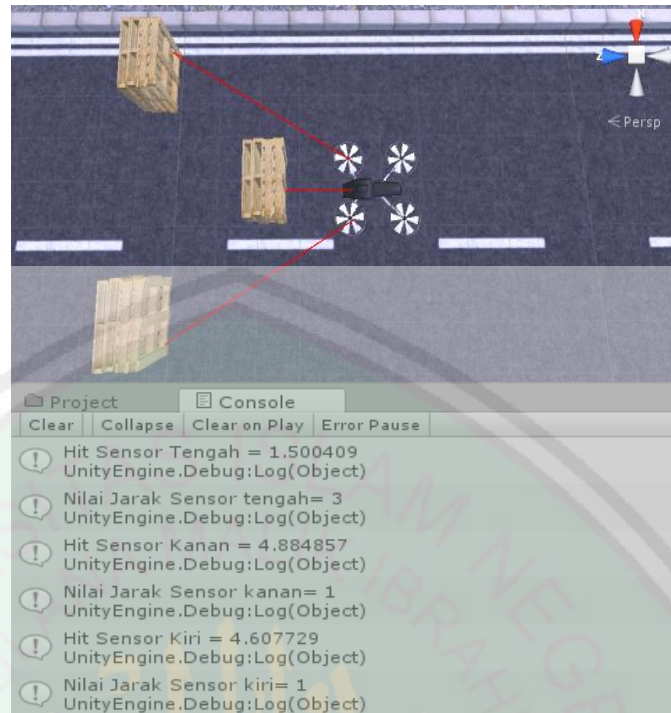
Gambar 4.26. Gambar Percobaan *Rule Jauh – Sedang – Sedang*

6. Jauh – Sedang – Dekat



Gambar 4.27. Gambar Percobaan *Rule Jauh – Sedang – Dekat*

7. Jauh – Dekat – Jauh



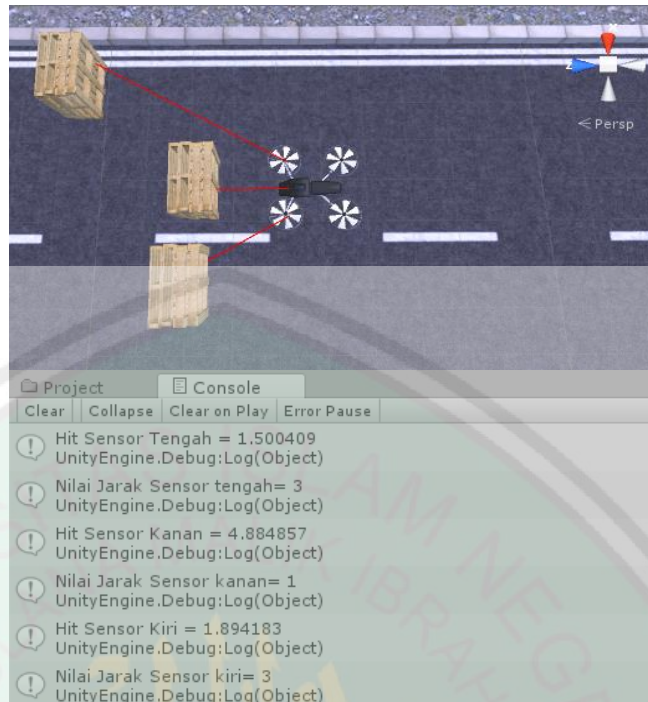
Gambar 4.28. Gambar Percobaan *Rule Jauh – Dekat – Jauh*

8. Jauh – Dekat – Sedang



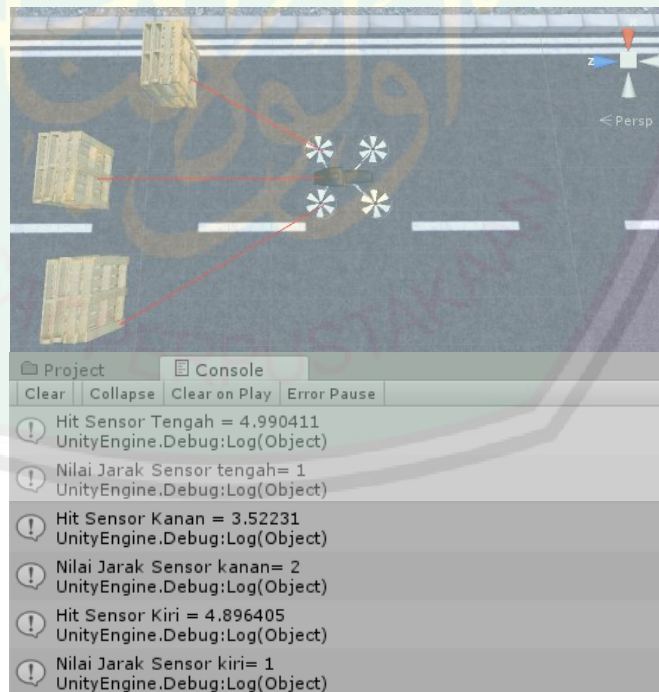
Gambar 4.29. Gambar Percobaan *Rule Jauh – Dekat – Sedang*

9. Jauh – Dekat – Dekat



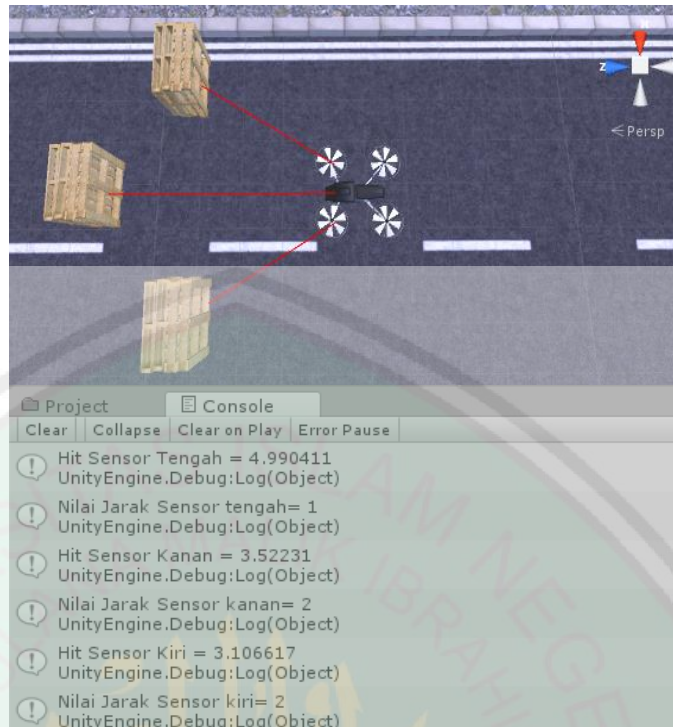
Gambar 4.30. Gambar Percobaan *Rule* Jauh – Dekat – Dekat

10. Sedang – Jauh – Jauh



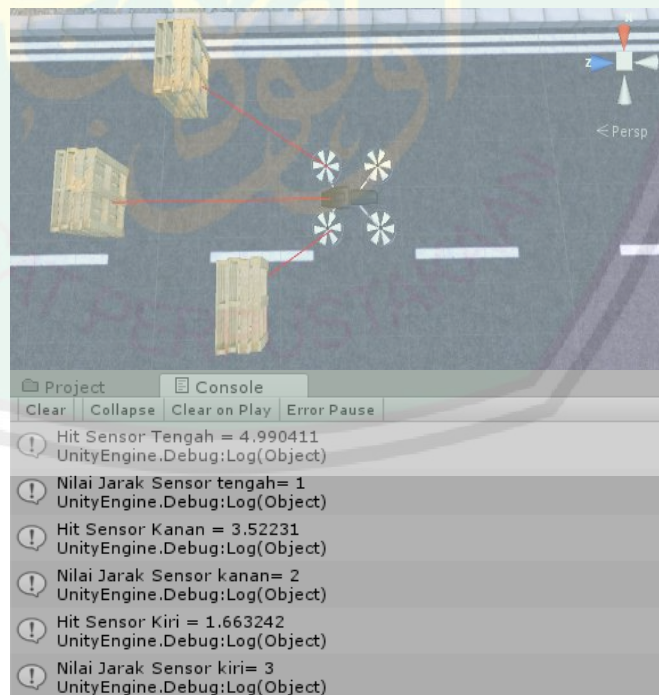
Gambar 4.31. Gambar Percobaan *Rule* Sedang – Jauh – Jauh

11. Sedang – Jauh – Sedang



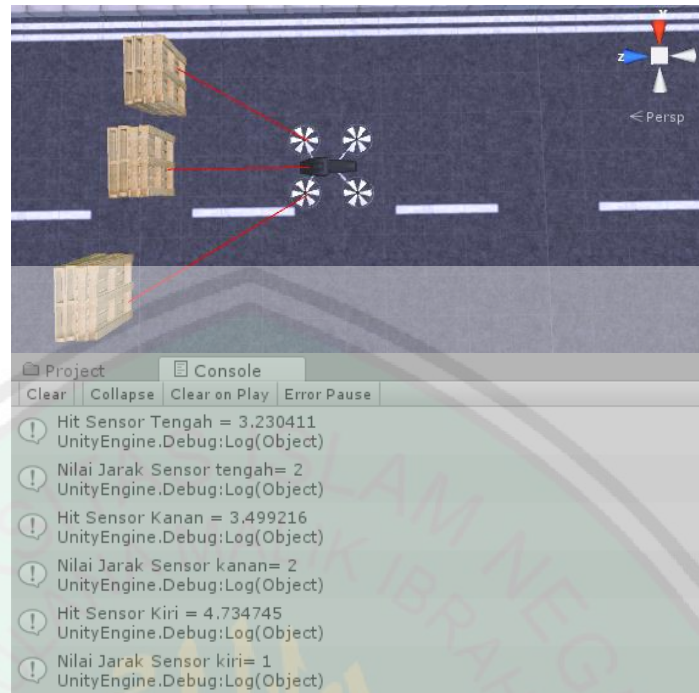
Gambar 4.32. Gambar Percobaan *Rule* Sedang – Jauh – Sedang

12. Sedang – Jauh – Dekat



Gambar 4.33. Gambar Percobaan *Rule* Sedang – Jauh – Dekat

13. Sedang – Sedang – Jauh



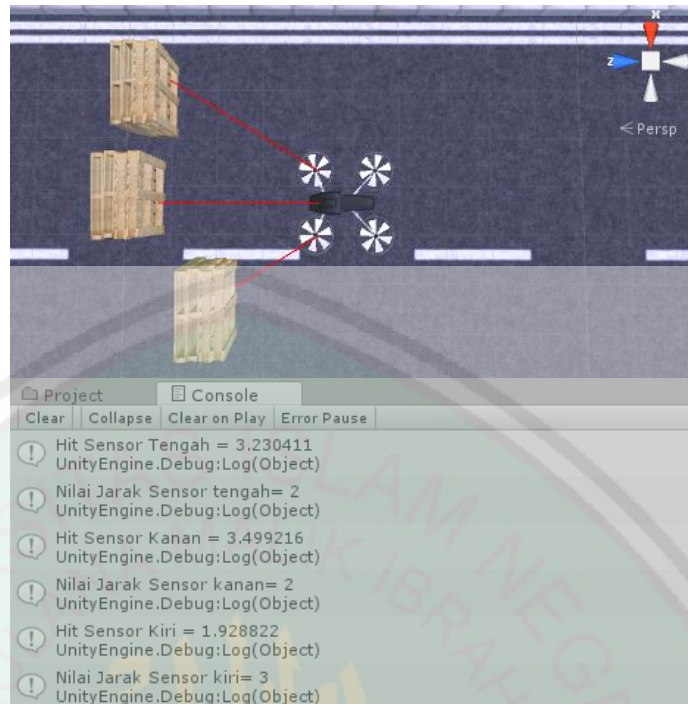
Gambar 4.34. Gambar Percobaan *Rule* Sedang – Sedang – Jauh

14. Sedang – Sedang – Sedang



Gambar 4.35. Gambar Percobaan *Rule* Sedang – Sedang – Sedang

15. Sedang – Sedang – Dekat



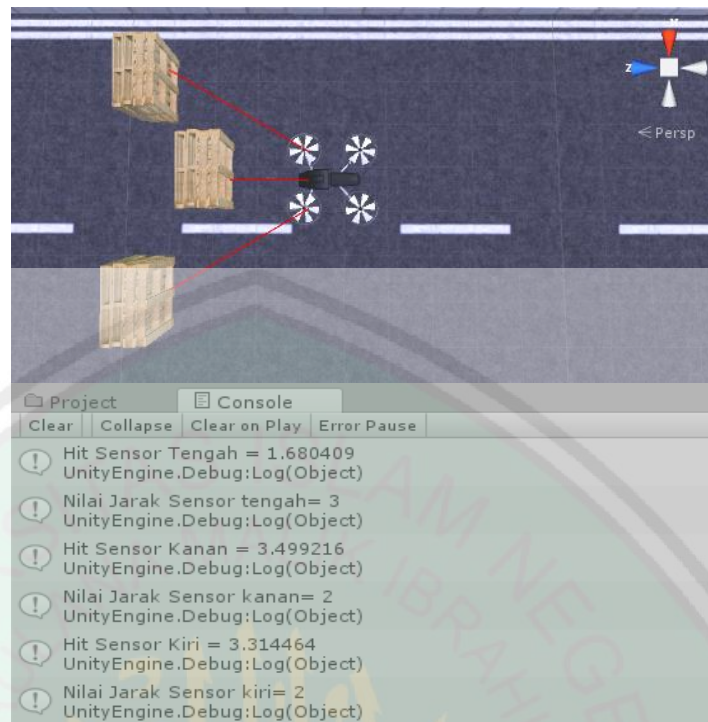
Gambar 4.36. Gambar Percobaan *Rule* Sedang – Sedang – Dekat

16. Sedang – Dekat – Jauh



Gambar 4.37. Gambar Percobaan *Rule* Sedang – Dekat – Jauh

17. Sedang – Dekat – Sedang



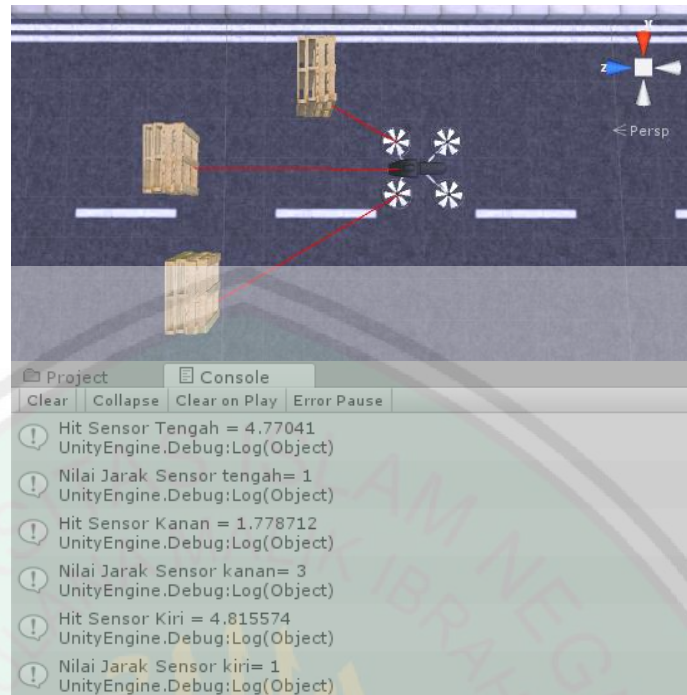
Gambar 4.38. Gambar Percobaan *Rule* Sedang – Dekat – Sedang

18. Sedang – Dekat – Dekat

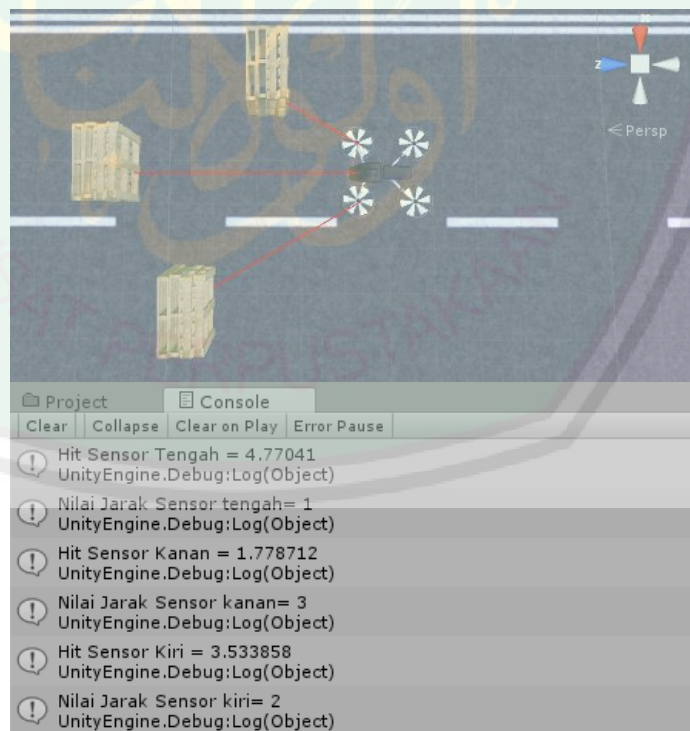


Gambar 4.39. Gambar Percobaan *Rule* Sedang – Dekat – Dekat

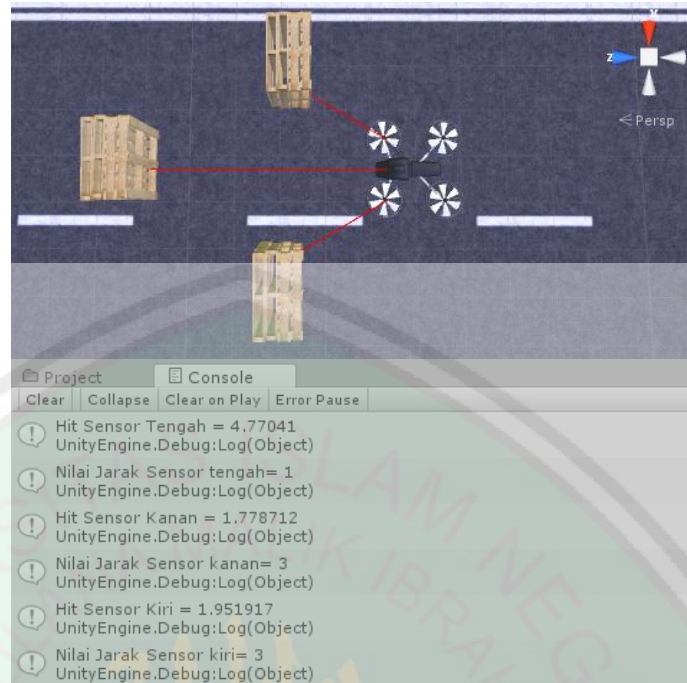
19. Dekat – Jauh – Jauh

Gambar 4.40. Gambar Percobaan *Rule* Dekat – Jauh – Jauh

20. Dekat – Jauh – Sedang

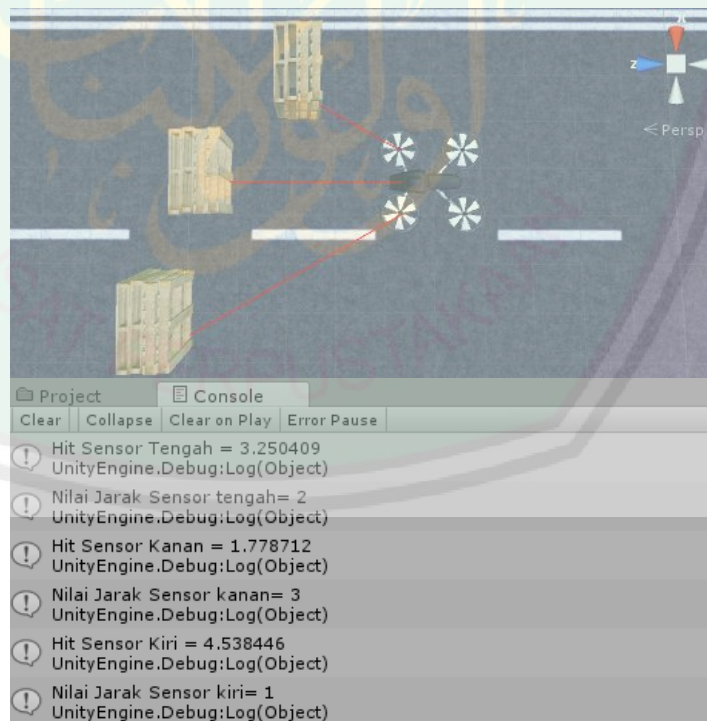
Gambar 4.41. Gambar Percobaan *Rule* Dekat – Jauh – Sedang

21. Dekat – Jauh – Dekat



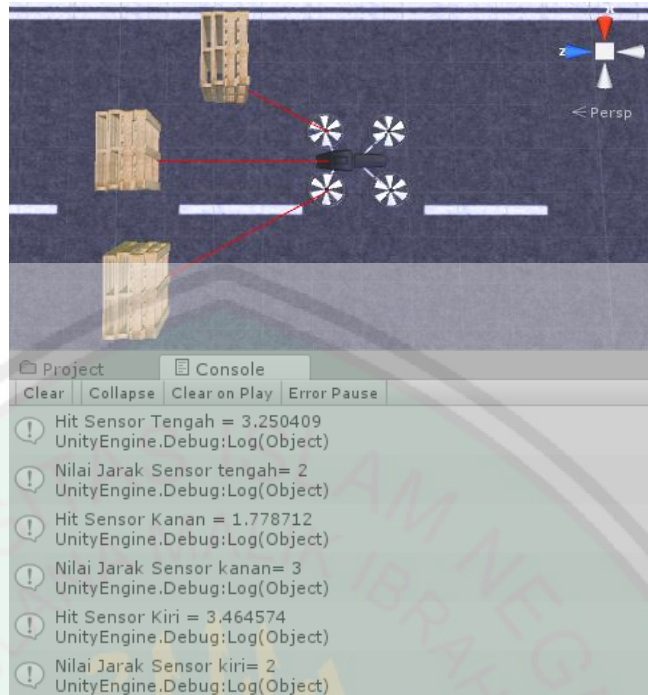
Gambar 4.42. Gambar Percobaan *Rule* Dekat – Jauh – Dekat

22. Dekat – Sedang – Jauh



Gambar 4.43. Gambar Percobaan *Rule* Dekat – Sedang – Jauh

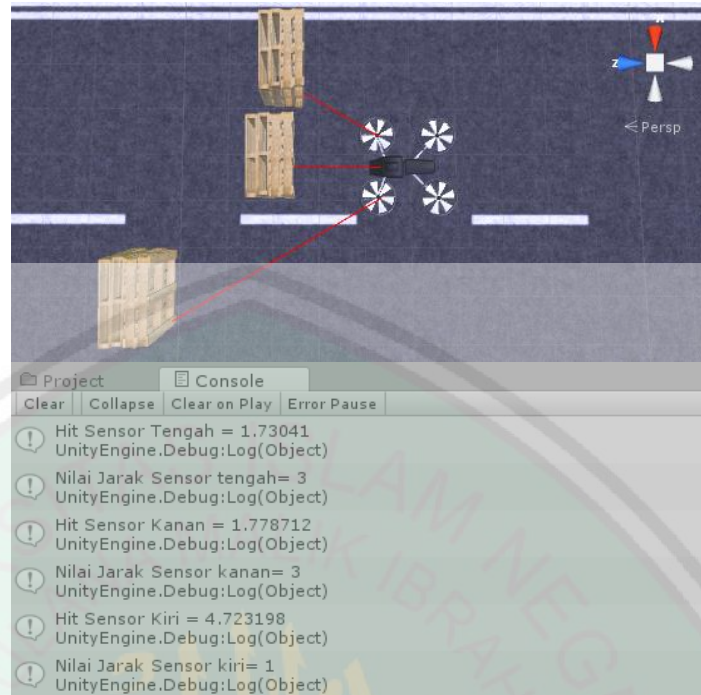
23. Dekat – Sedang – Sedang

Gambar 4.44. Gambar Percobaan *Rule* Dekat – Sedang – Sedang

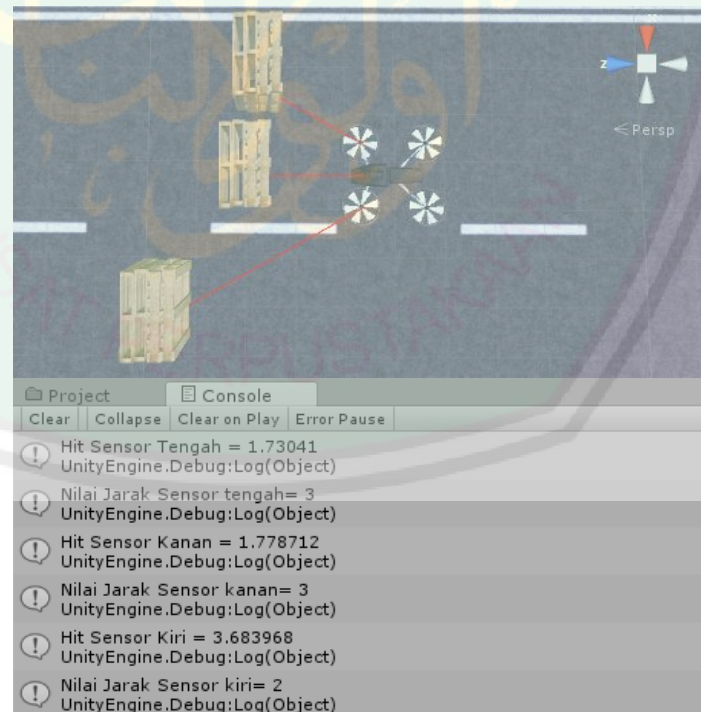
24. Dekat – Sedang – Dekat

Gambar 4.45. Gambar Percobaan *Rule* Dekat – Sedang – Dekat

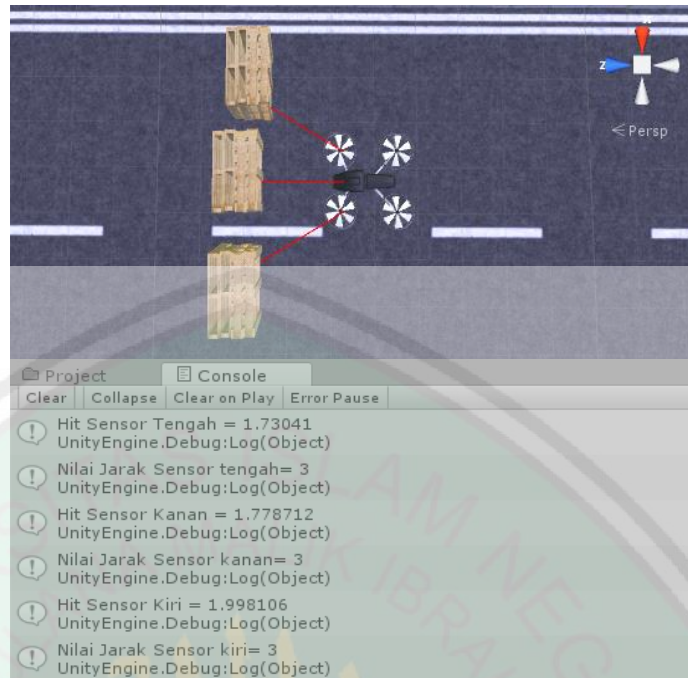
25. Dekat – Dekat – Jauh

Gambar 4.46. Gambar Percobaan *Rule* Dekat – Dekat – Jauh

26. Dekat – Dekat – Sedang

Gambar 4.47. Gambar Percobaan *Rule* Dekat – Dekat – Sedang

27. Dekat – Dekat – Dekat

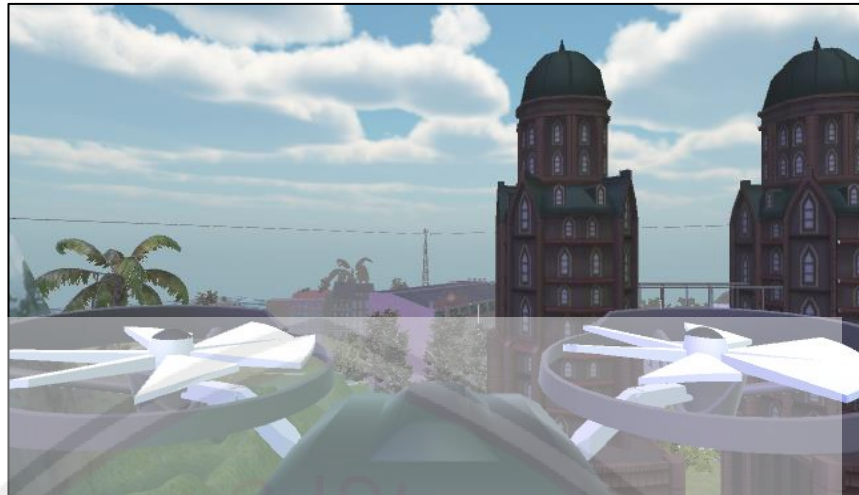


Gambar 4.48. Gambar Percobaan *Rule* Dekat – Dekat – Dekat

4.5.4 Uji Coba *X-Drone* Menghindari Halangan

Untuk menguji coba *X-Drone* dalam menghindari halangan maka dilakukan pengujian langsung antara *X-Drone* dengan *obstacle*. Pada pengujian ini dilakukan untuk mengetahui apakah *X-Drone* akan menghindari halangan yang ada di depannya atau malah beraksi sebaliknya yaitu menabrak halangan yang ada di depannya.

Untuk menghindari halangan terdapat dua aksi yaitu geser kanan dan geser kiri. Aksi ini terjadi jika salah satu atau dua sensor dari ketiga sensor mendeteksi jarak antara *X-Drone* dengan *obstacle* adalah dekat. Dalam percobaan ini dilakukan pada salah satu menara pada *game* simulasi *X-Drone* seperti pada Gambar 4.49 berikut.



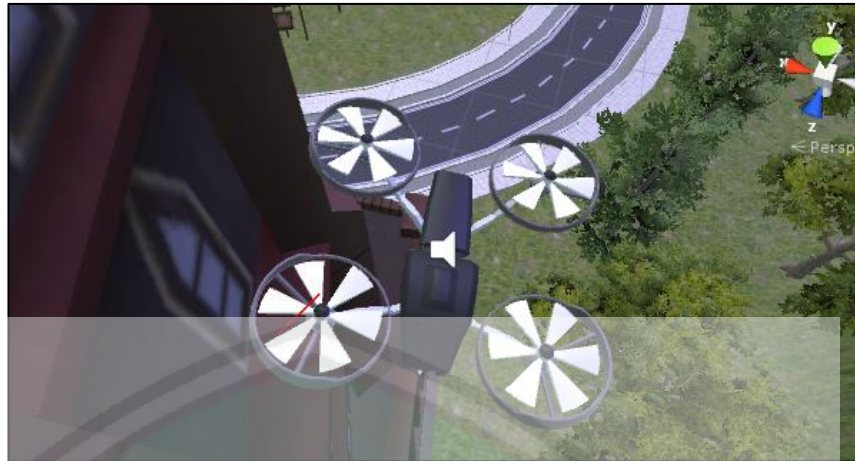
Gambar 4.49. Tampilan Menara kembar dari sisi *X-Drone*

Percobaan ini mendeteksi sensor kiri sama dengan jauh, sensor tengah sama dengan jauh, dan sensor kanan sama dekat. Yang dimana berdasarkan *rule fuzzy* maka diberikan aksi untuk geser kiri pada *X-Drone*, percobaan dilakukan seperti Gambar 4.50 berikut.



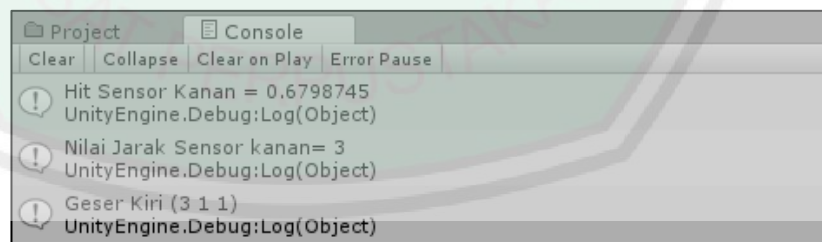
Gambar 4.50. Tampilan *X-Drone* saat menghindari *Obstacle*

Percobaan ini mendeteksi jarak pada *obstacle* sehingga dapat memberi pergerakan secara otomatis melakukan pergerakan sesuai dengan aksi yang diberikan. Jarak antara *X-Drone* dapat dilihat pada Gambar 4.51 yang diberi dipancar menggunakan sinar merah.



Gambar 4.51. Tampilan jarak dekat antara *X-Drone* dengan *obstacle*

Dan nilainya dapat diambil melalui *debug* pada *console* unity, yang dapat di lihat pada Gambar 4.52 bahwa terdeteksi sensor kanan antara *X-Drone* ke *obstacle* sebesar 0.6798745 yang berarti bahwa sensor kanan itu berjarak dekat yang di beri *konstanta* tiga. Berdasarkan *fuzzy rule*, maka aksi yang diberikan pada *X-Drone* adalah geser kiri. Setelah melakukan pergerakan geser ke kanan ataupun ke kiri maka pergerakan ini keluar dari jarak dekat dan masuk ke jarak sedang, yang dimana jika salah satu dan atau kedua dan atau ketiga sensor tersebut terdeteksi maka diberikan aksi lambat.



Gambar 4.52. Nilai Sensor Jarak dan Aksi *Fuzzy*

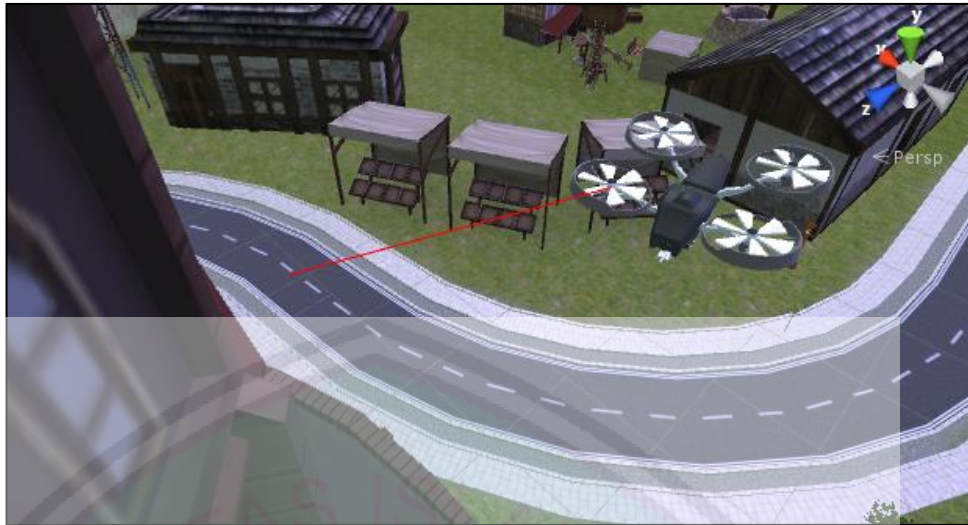
4.5.5 Uji Coba *X-Drone* dalam Keadaan Mengurangi Kecepatan

Pengujian ini dilakukan untuk mengetahui apakah *X-Drone* dapat lambat ketika ada halangan atau *obstacle* yang ada di depannya. Percobaan dilakukan dengan membandingkan *X-Drone* dalam keadaan diam dan lambat apakah ada perbedaan pada kedua aksi tersebut.

Pengujian ini dilakukan pada menara pada *game* simulasi *X-Drone*, aksi lambat diberikan pada *X-Drone* ketika salah satu dan atau kedua dan atau ketiganya mendeteksi halangan atau *Obstacle* pada jarak sama dengan sedang. Sedangkan aksi diam diberikan pada saat ketiga sensor mendapatkan jarak sama dengan dekat.

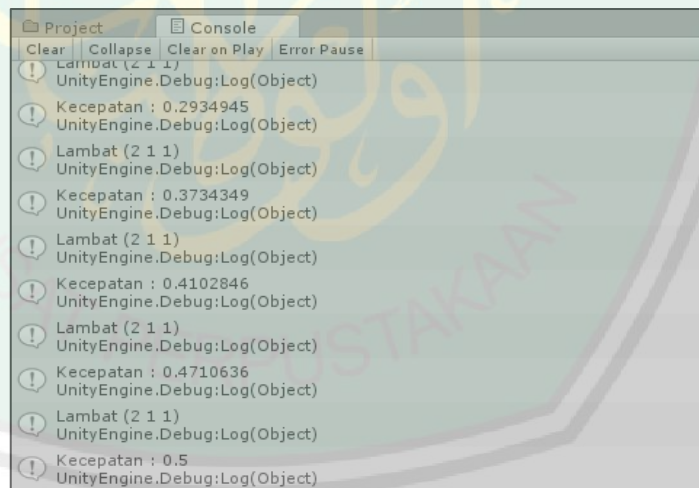
a. Mengurangi Kecepatan

Pengurangan Kecepatan terjadi apabila *fuzzy rule* pada *X-Drone* adalah lambat, pengurangan kecepatan *X-Drone* menjadi $0 - 0.5f$. Yang dimana dalam aksi bergerak kecepatannya adalah $0 - 1f$, hal ini dilakukan agar kecepatan *X-Drone* berkurang dan tidak terlalu cepat saat menghadapi *obstacle*, selain itu dengan adanya pengurangan kecepatan, pengemudi dapat mengetahui bahwa ada *obstacle* yang terdeteksi.



Gambar 4.53. Tampilan jarak sedang antara *X-Drone* dengan *Obstacle*

Pada Gambar 4.53 merupakan jarak sedang yang terdeteksi oleh sensor kanan, seperti penjelasan di atas bahwa pada saat aksi lambat terjadi maka kecepatan yang diberikan kepada *X-Drone* hanya 0 – 0.5f saat pemain menekan tombol untuk maju ke depan, seperti pada Gambar 4.54.

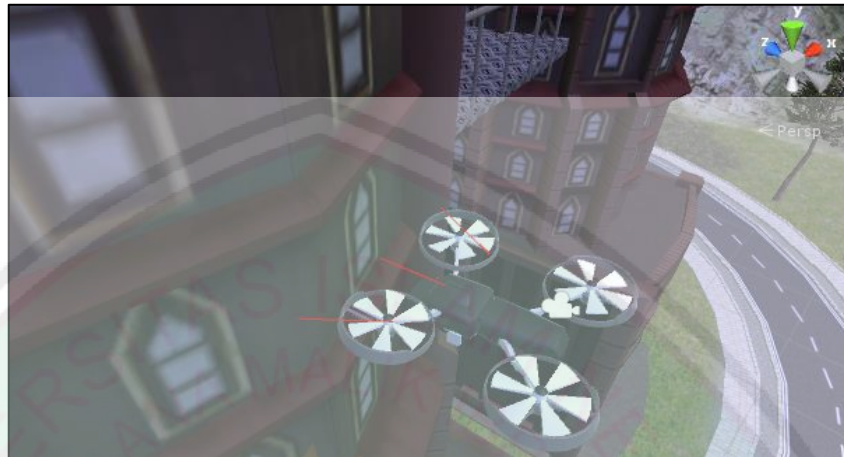


Gambar 4.54. Aksi dan Nilai pengurangan Kecepatan *X-Drone*

b. Diam

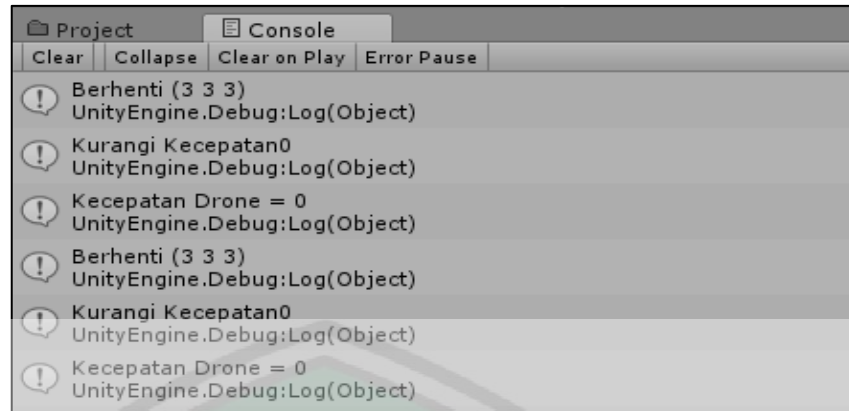
Aksi diam terjadi ketika ketiga sensor mendeteksi jarak antara *X-Drone* dengan *obstacle* adalah dekat, maka kecepatan sama dengan 0 (nol), yang secara otomatis *input* untuk bergerak maju kedepan tidak berfungsi karena

jika *input* maju berfungsi maka dapat terjadi tabrakan. Akan tetapi untuk fungsi yang lain tetap berfungsi bergerak ke samping kanan kiri ataupun naik ke atas maupun ke bawah.



Gambar 4.55. Tampilan jarak dekat antara *X-Drone* dengan *Obstacle*

Pada Gambar 4.55 terlihat bahwa ketiga sensor mendeteksi jarak sama dengan dekat, yang dimana memberikan aksi diam dengan kecepatan sama dengan 0 (nol). Akan tetapi pada aksi ini mengakibatkan perputaran *rotor* sayap *X-Drone* tidak dapat berputar sesuai dengan kecepatan rotasi yang diberikan, hal ini disebabkan gesekan *collider* yang terlalu dekat antara *X-Drone* dengan *obstacle*. Pada posisi *X-Drone* yang seharusnya diam masih terdeteksi bergerak, jadi bisa dikatakan pada aksi diam ini tidak berhasil.

Gambar 4.56. Aksi diam pada *X-Drone*

4.5.6 Uji Coba *X-Drone* dengan Keadaan *Obstacle*

Pengujian ini dilakukan untuk menguji apakah *X-Drone* dapat memberi aksi pergerakan untuk menjauhi *obstacle* saat *obstacle* dalam keadaan diam atau bergerak. Pergerakan *obstacle* yaitu dengan kecepatan antara 0 – 5. Kecepatan sama dengan nol maka *obstacle* dinyatakan diam, jika kecepatan 1-5 maka *obstacle* dinyatakan bergerak.

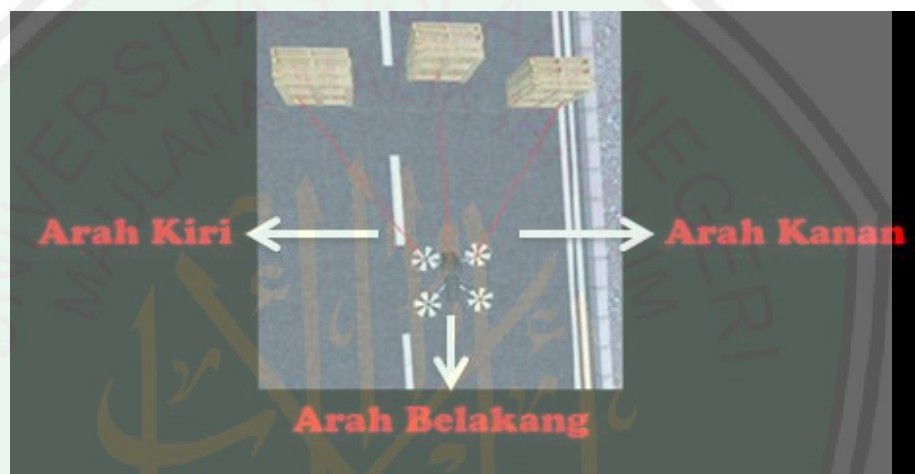
Tabel 4.3. Uji Coba Pergerakan *X-Drone* dengan Keadaan *Obstacle*

No	Keadaan <i>X-Drone</i>	<i>Obstacle</i>	Kecepatan	Keterangan
1	Diam	Bergerak	1	Berhasil
		Bergerak	2	Berhasil
		Bergerak	3	Berhasil
		Bergerak	4	Berhasil
		Bergerak	5	Berhasil
2	Bergerak	Bergerak	1	Berhasil
		Bergerak	2	Berhasil
		Bergerak	3	Berhasil
		Bergerak	4	Berhasil
		Bergerak	5	Berhasil
3	Bergerak	Diam	0	Berhasil

Dari hasil tabel uji coba di atas didapatkan bahwa pergerakan *X-Drone* dalam menghindari *obstacle* berhasil dilakukan pada *obstacle* yang bergerak maupun yang diam.

4.5.7 Uji Coba *X-Drone* dengan berbagai arah *Obstacle*

Uji coba *X-Drone* untuk melihat apa yang terjadi jika *obstacle* berada di arah yang tidak terdeteksi oleh sensor, seperti pada Gambar 4.57 berikut.



Gambar 4.57. Arah-arrah keberadaan *obstacle*

Kemudian dilakukan uji coba dengan mencari keberadaan *obstacle* tersebut, yang bergerak ataupun tidak bergerak dan menghasilkan hasil seperti pada Tabel 4.4 berikut

Tabel 4.4. Uji Coba Pergerakan *X-Drone* dengan berbagai arah *Obstacle*

No	Keadaan <i>X-Drone</i>	<i>Obstacle</i>	Keterangan
1	Diam	Depan	Berhasil
		Kanan	Bertabrakan
		Kiri	Bertabrakan
		Belakang	Bertabrakan
		Atas	Bertabrakan

		Bawah	Bertabrakan
2	Bergerak	Depan	Berhasil
		Kanan	Bertabrakan
		Kiri	Bertabrakan
		Belakang	Bertabrakan
		Atas	Bertabrakan
		Bawah	Bertabrakan

Dari hasil tabel di atas maka dapat disimpulkan bahwa sensor hanya bisa mendeteksi *obstacle* yang bergerak ataupun tidak bergerak yang berada di depan atau yang berada di garis lurus kemiringan 30° sensor dengan radius pemancaran masing-masing sensor sudut deviasi 30° yang memiliki 13 *raycast* dengan perbedaan *angle* 2.5° , selain dari *raycast* tersebut maka disebut *blank spot* yang di mana jika ada *obstacle* yang berada pada posisi tersebut, maka *X-Drone* tidak dapat mendeteksi *obstacle* dan apabila jaraknya dengan *obstacle* terlalu dekat, maka *X-Drone* dengan *obstacle* dapat bertabrakan.

4.6 Uji Coba *X-Drone* dengan Jumlah *Obstacle*

Uji coba ini dilakukan untuk mengetahui apa yang terjadi jika *x-drone* menghadapi *obstacle* lebih dari satu. Hasil uji coba dapat dilihat pada Tabel 4.5 berikut.

Tabel 4.5. Uji Coba *X-Drone* dengan Jumlah *Obstacle*

No	Jumlah Obstacle	Arah Obstacle			Keterangan
		Kanan	Tengah	Kiri	
1.	1 (Satu) Obj	1	0	0	Berhasil
2.	1 (Satu) Obj	0	1	0	Berhasil
3.	1 (Satu) Obj	0	0	1	Berhasil

4.	2 (Dua) Obj	1	1	0	Berhasil
5.	2 (Dua) Obj	0	1	1	Berhasil
6.	2 (Dua) Obj	1	0	1	Gagal
7.	3 (Tiga) Obj	1	1	1	Berhasil

Dari hasil uji coba di atas diketahui bahwa pada keadaan *x-drone* menghadapi dua objek yang terdeteksi di sensor kanan dan sensor kiri tidak berhasil, hal ini disebabkan karena ketidakseimbangan pada output aksi *fuzzy rule*, yang dimana pada posisi dekat yang terdeteksi oleh sensor kanan dan sensor kiri berada pada posisi *fuzzy rule* (Dekat – Jauh – Dekat) yang akan memberikan *output* aksi geser kiri. Akan tetapi setelah memberikan aksi geser kiri, *x-drone* akan bergeser ke kiri dan berada pada posisi *fuzzy rule* (Sedang – Dekat – Dekat) yang dimana akan memberikan aksi *output* geser kanan.

Kemudian setelah memberikan aksi *output* geser kanan, *x-drone* akan bergeser ke kanan dan kembali berada pada posisi *fuzzy rule* (Dekat – Jauh – Dekat) yang dimana akan memberikan aksi *output* geser kiri, dan hal ini berulang seterusnya kecuali pemain menekan tombol dari *remote control* untuk keluar dari aksi tersebut. Ketidakseimbangan *fuzzy rule* pada *x-drone* ketika menghadapi dua *obstacle* yang terdeteksi awal pada sensor kanan dan sensor kiri.

Sehingga dapat disimpulkan bahwa *x-drone* dapat menghindari *obstacle* yang lebih dari dua, akan tetapi jika menghadapi dua *obstacle* yang terdeteksi awal pada sensor kanan dan sensor kiri, maka *x-drone* menjadi tidak seimbang dalam memberikan aksi.

4.7 Analisa Hasil Uji Coba

Analisa dari *X-Drone* dilakukan untuk menguji apakah *rule* yang telah ditetapkan sebelumnya dapat bekerja pada pergerakan *X-Drone* dalam menghadapi *obstacle*. Hasil uji coba dari penelitian dengan ukuran jarak yang dicatat dalam Tabel 4.6 berikut.

Tabel 4.6. Hasil Uji Coba Pergerakan *X-Drone*

No	<i>Rule</i>			<i>Output Fuzzy Rule</i>	Hasil Uji Coba	Keterangan
	Kanan	Tengah	Kiri			
1	Jauh 4.8	Jauh 4.9	Jauh 4.8	Bergerak	Bergerak	Berhasil
2	Jauh 4.8	Jauh 4.9	Sedang 2.9	Lambat	Lambat	Berhasil
3	Jauh 4.8	Jauh 4.9	Dekat 1.8	Geser Kanan	Geser Kanan	Berhasil
4	Jauh 4.8	Sedang 3.5	Jauh 4.8	Lambat	Lambat	Berhasil
5	Jauh 4.8	Sedang 3.5	Sedang 4	Lambat	Lambat	Berhasil
6	Jauh 4.8	Sedang 3.5	Dekat 1.7	Geser Kanan	Geser Kanan	Berhasil
7	Jauh 4.8	Dekat 1.5	Jauh 4.6	Geser Kanan	Geser Kanan	Berhasil
8	Jauh 4.8	Dekat 1.5	Sedang 3.4	Geser Kanan	Geser Kanan	Berhasil
9	Jauh 4.8	Dekat 1.5	Dekat 1.8	Geser Kanan	Geser Kanan	Berhasil
10	Sedang 3.5	Jauh 4.9	Jauh 4.8	Lambat	Lambat	Berhasil
11	Sedang 3.5	Jauh 4.9	Sedang 3.1	Lambat	Lambat	Berhasil
12	Sedang 3.5	Jauh 4.9	Dekat 1.6	Geser Kanan	Geser Kanan	Berhasil
13	Sedang 3.4	Sedang 3.2	Jauh 4.7	Lambat	Lambat	Berhasil
14	Sedang 3.4	Sedang 3.2	Sedang 3.6	Lambat	Lambat	Berhasil
15	Sedang 3.4	Sedang 3.2	Dekat 1.9	Geser Kanan	Geser Kanan	Berhasil

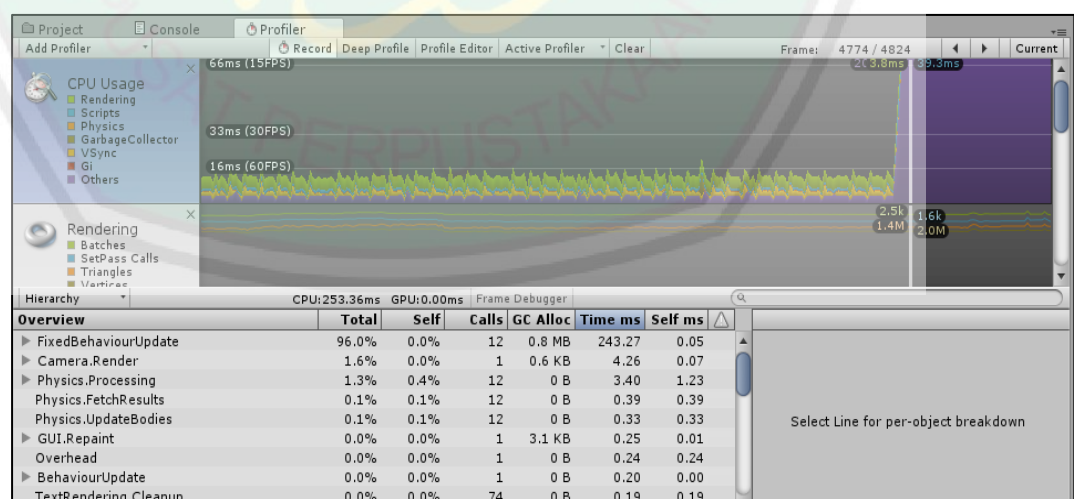
16	Sedang 3.4	Dekat 1.6	Jauh 4.6	Geser Kanan	Geser Kanan	Berhasil
17	Sedang 3.4	Dekat 1.6	Sedang 3.3	Geser Kanan	Geser Kanan	Berhasil
18	Sedang 3.4	Dekat 1.6	Dekat 1.9	Geser Kanan	Geser Kanan	Berhasil
19	Dekat 1.7	Jauh 4.7	Jauh 4.8	Geser Kiri	Geser Kiri	Berhasil
20	Dekat 1.7	Jauh 4.7	Sedang 3.5	Geser Kiri	Geser Kiri	Berhasil
21	Dekat 1.7	Jauh 4.7	Dekat 1.9	Geser Kiri	Geser Kanan	Berhasil
22	Dekat 1.7	Sedang 3.2	Jauh 4.5	Geser Kiri	Geser Kiri	Berhasil
23	Dekat 1.7	Sedang 3.2	Sedang 3.4	Geser Kiri	Geser Kiri	Berhasil
24	Dekat 1.7	Sedang 3.2	Dekat 1.9	Geser Kiri	Geser Kiri	Berhasil
25	Dekat 1.7	Dekat 1.7	Jauh 4.7	Geser Kiri	Geser Kiri	Berhasil
26	Dekat 1.7	Dekat 1.7	Sedang 3.6	Geser Kiri	Geser Kiri	Berhasil
27	Dekat 1.7	Dekat 1.7	Dekat 1.9	Diam	Lambat	Gagal

Dari hasil percobaan pada Tabel 4.6 menunjukkan hasil akurasi sebanyak 96.3%. Hasil tersebut dinyatakan sudah cukup baik karena hanya terjadi tidak sesuai pada satu *rule* saja dari 27 *rule* sehingga mengurangi hasil akurasi pada pergerakan *X-Drone* dalam menghadapi halangan atau *obstacle*.

Pada percobaan tersebut terdapat satu kondisi yang tidak sesuai dengan *rule* yang ditentukan yaitu ketika sensor kanan, sensor tengah, dan sensor kiri mendeteksi adanya halangan dengan jarak DEKAT yang mana *rule* yang harus diberikan pada aksi *X-Drone* adalah DIAM. Akan tetapi aksi yang diberikan adalah diam dengan kecepatan 0 (nol) akan tetapi masih terdapat pergerakan dan perputaran sayap *rotor* pun juga ikut diam. Berdasarkan analisa yang dilakukan

pada kegagalan *rule* tersebut disebabkan karena pergerakan *X-Drone* merupakan pergerakan *relative* yang dibantu dengan *up force* akan tetapi masih dipengaruhi oleh gaya gravitasi yang aktif sehingga masih tetap bergerak walaupun sangat lambat. Hal ini juga disebabkan terlalu dekat *X-Drone* dengan *collider obstacle* sehingga terjadi tabrakan yang membuat *X-Drone* memantul ke belakang, hal ini disebabkan masing-masing *collider x-drone* ataupun *obstacle* sama-sama tidak sama dengan *is trigger* yang artinya tidak bisa dimasuki atau ditembus oleh objek lain.

Kemudian untuk permasalahan perputaran sayap *rotor X-Drone*, walaupun kecepatan *rotor* adalah sama dengan 5 yang meningkat kecepatannya sesuai dengan perubahan *velocity x-drone*. Pada aksi ini, atau lambat, geser kanan, geser kiri, sayap *rotor* terlihat tidak bergerak dan melambat. Dan berdasarkan analisa kegagalan ini disebabkan karena nilai fps (*frame per second*) pada *script fuzzy rule* meningkat 96% daripada aksi bergerak yang hanya 56%, seperti pada Gambar 4.58 berikut.



Gambar 4.58. Tampilan peningkatan nilai fps pada *fuzzy rule*

Kelemahan dalam penelitian ini adalah sensor hanya bisa mendeteksi *obstacle* yang bergerak ataupun tidak bergerak yang berada di depan atau yang berada di garis lurus kemiringan 30° sensor dengan radius pemancaran masing-masing sensor 30° dengan sudut deviasi -15° ke kiri dan 15° ke kanan dan memiliki 13 *raycast* dengan perbedaan *angle* 2.5° , selain dari *raycast* tersebut maka disebut *blank spot* yang dimana jika ada *obstacle* yang berada pada posisi tersebut, maka *X-Drone* tidak dapat mendeteksi *obstacle* dan apabila jaraknya dengan *obstacle* terlalu dekat, maka *X-Drone* dengan *obstacle* dapat bertabrakan.

4.8 Integrasi Sains dengan Islam

Tujuan awal penelitian ini adalah membuat rancangan pengendalian *quadcopter* yang mana dapat menghindari kecelakaan atau kerusakan di muka bumi ini, sebagaimana Allah telah berfirman dalam Surah Ar Rum Ayat 41 yang berbunyi :

ظَهَرَ الْفَسَادُ فِي الْبَرِّ وَالْبَحْرِ بِمَا كَسَبَتْ أَيْدِي النَّاسِ لِيُذِيقَهُمْ بَعْضَ
الَّذِي عَمِلُوا لَعَلَّهُمْ يَرْجِعُونَ ﴿٤١﴾

Artinya :

41. Telah nampak kerusakan di darat dan di laut disebabkan karena perbuatan tangan manusia, supaya Allah merasakan kepada mereka sebahagian dari (akibat) perbuatan mereka, agar mereka kembali (ke jalan yang benar). (Q.S Ar-Rum (30):41)

Karena ketidakpandaian pengemudi dalam mengendalikan *quadcopter* terutama pengemudi pemula yang masih belum paham akan fungsi-fungsi dari *remote control* dan pada *quadcopter* tidak terdapat sensor yang akan mendeteksi adanya halangan atau *obstacle* sehingga terjadinya kecelakaan yang dapat melibatkan apapun yang berada di muka bumi ini, dimana Manusia akan mendapatkan balasan dari perbuatannya, sebagaimana dengan penjelasan hadits berikut :

عَنْ أَبِي عَمْرٍ وَبْنِ جُبَيْرِ بْنِ عَبْدِ اللَّهِ قَالَ: قَالَ رَسُولُ اللَّهِ (ص): مَنْ سَنَّ فِي
الْإِسْلَامِ سُنَّةً حَسَنَةً فَلَهُ أَجْرُهَا وَأَجْرُ مَنْ عَمِلَ بِهَا بَعْدَهُ مِنْ غَيْرِ أَنْ يَنْقُصَ مِنْ
أُجُورِهِمْ شَيْءٌ. وَمَنْ سَنَّ سُنَّةً سَيِّئَةً كَانَ عَلَيْهِ وِزْرُهَا وَوِزْرُ مَنْ عَمِلَ بِهَا بَعْدَهُ
مِنْ غَيْرِ أَنْ يَنْقُصَ مِنْ أَوْزَارِهِمْ شَيْءٌ (رواه مسلم)

Artinya :

Dari Abi Amr Ibnu Jubair Ibnu Abdillah, ia berkata, “Rasulullah bersabda,”Barang siapa yang yang berbuat/prakarsa yang baik dalam Islam, maka ia akan memperoleh pahala dari perbuatan/ prakarsa itu dan pahala dari orang yang melaksanakan atau menirunya. Dan barang siapa berprakarsa yang jelek, maka ia akan mendapatkan dosa dari prakarsanya itu dan dosa dari orang-orang yang mempraktikkan prakarsanya itu tanpa mengurangi dosa yang menirunya. (HR. Muslim)

Drone juga memiliki banyak keistimewaan bahkan dalam dunia transportasi misalnya digunakan untuk pos dalam mengantarkan paket ataupun surat. Fungsi istimewa yang dimiliki *drone* menyebabkan penggunaannya mulai

melebar ke dunia bidang fotografi udara, jurnalisme, hingga riset. Untuk kelas fotografi udara sudah sangat beragam jenisnya, mulai pengambilan video dan foto, *monitoring*, hingga penggunaan untuk Sistem Informasi Geografis atau GIS. Allah telah menciptakan segala sesuatu yang mempermudah manusia dalam berbagai hal terutama dalam hal transportasi sebagaimana Allah berfirman dalam surah An-Nahl ayat 7-8.

وَتَحْمِلُ أَثْقَالَكُمْ إِلَىٰ بَلَدٍ لَّمْ تَكُونُوا بَالِغِيهِ إِلَّا بِشِقِّ الْأَنْفُسِ ۚ إِنَّ رَبَّكُمْ لَرَءُوفٌ رَّحِيمٌ ﴿٧﴾
وَالْخَيْلَ وَالْبِغَالَ وَالْحَمِيرَ لِتَرْكَبُوهَا وَزِينَةً
وَتَخْلُقُ مَا لَا تَعْلَمُونَ ﴿٨﴾

Artinya :

7. Dan ia memikul beban-bebanmu ke suatu negeri yang kamu tidak sanggup sampai kepadanya, melainkan dengan kesukaran-kesukaran (yang memayahkan) diri. Sesungguhnya Tuhanmu benar-benar Maha Pengasih lagi Maha Penyayang,
8. Dan (dia telah menciptakan) kuda, bagal, dan keledai, agar kamu menungganginya dan (menjadikannya) perhiasan. dan Allah menciptakan apa yang kamu tidak mengetahuinya.

Ayat di atas menjelaskan bahwa Allah mempermudah pekerjaan manusia terutama dalam penggunaan transportasi, dan Allah telah menciptakan apa yang tidak kamu ketahui, sebagaimana kita ketahui dengan berkembang zaman, berkembang pula transportasi seperti pesawat yang dulunya hanya menggunakan binatang dalam mempermudah manusia sebagaimana penjelasan ayat di atas.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian tentang pergerakan *quadcopter* dalam menghindari *obstacle* pada *game* simulasi *X-Drone* menggunakan metode *fuzzy sugeno*, setelah melakukan hasil uji coba maka didapatkan bahwa penggunaan metode *fuzzy sugeno* dapat diterapkan untuk menentukan pergerakan *quadcopter* dalam *game* Simulasi *X-Drone* dan memiliki tingkat akurasi sebesar 96.3%. Variabel *Fuzzy Sugeno* terdiri dari tiga variabel yaitu sensor kanan, sensor tengah, sensor kiri, yang dimana masing-masing memiliki nilai linguistik jarak jauh, sedang, dan dekat. Dan terdapat *output* berupa aksi pergerakan pada *quadcopter*, yang memiliki aksi jalan terus, geser kanan, geser kiri, kurangi kecepatan, dan diam.

Kelemahan dalam penelitian ini adalah sensor hanya mendeteksi objek yang berada didepan sensor yaitu tepat 0° dengan sudut deviasi -15° ke kiri dan 15° ke kanan yang terbentuk sudut 30° pada sensor dan memiliki 13 *raycast* dengan perbedaan *angle* 2.5°, selain dari *raycast* tersebut maka disebut *blank spot* yang dimana jika ada *obstacle* yang berada pada posisi tersebut, maka *X-Drone* tidak dapat mendeteksi *obstacle* dan apabila jaraknya dengan *obstacle* terlalu dekat, maka *X-Drone* dengan *obstacle* dapat bertabrakan.

5.2 Saran

Ada beberapa saran yang dapat diajukan pada penelitian selanjutnya.

1. Penggunaan metode *fuzzy sugeno* pada *quadcopter* dapat dilakukan dengan melakukan metode yang berbeda sehingga bisa didapatkan perbedaan dalam mendeteksi *obstacle* dan memberi aksi.
2. Dari sisi *design interface* masih belum maksimal dan belum sesuai dengan dengan perkembangan UI ataupun UX sekarang ini.
3. Pembangunan *game* simulasi bisa ditambahkan dengan pengambilan *item* agar *game* simulasi ini lebih *fun*.

DAFTAR PUSTAKA

- Anshori, Syaifudin. (2016). *Rancang Bangun Quadcopter untuk Pencarian Rute Optimum pada Kebakaran Lahar Gambut menggunakan Metode Particle Swarm Optimazation*. Skripsi Jurusan S1 Teknik Informatika. Malang: Universitas Islam Negeri Maulana Malik Ibrahim Malang.
- Anugerah, Risha. (2016). *Flight Controller pada Sisytem Quadcopter menggunakan sensor IMU (Inertial Measurment Unit) berbasis mikrokontroller Atmega 2560*. Other Thesis. Universitas Shanarta Dharma.
- Arif, Yunifa Miftachul, dkk. (2009). *integrasi hierarchy finite state machine dan logika fuzzy untuk desain strategi NPC Game*, Surabaya : ITS.
- Asrori, Imam. (2008). *Aneka Permainan Penyegaran Bahasa Arab*. Surabaya: Hilal Pustaka.
- Bresciani, T. (2008). *Modelling Identification and Control of a Quadrotor Helicopter*. English, 4(October), p.213.
- Blakelock, John H. (1965). *Automatic Control of Aircraft and Missiles*. John Willey and Sons, Inc. New York.
- Bates, Bob. (2004). *Game Design: Second Edition*. Boston: Thomson Course Technology PTR.
- C. Ivan Sibero, (2009). *Langkah Mudah Membuat Game 3D*. MediaKom: Yogyakarta
- David Fox and Roman Verhosek. 2002. *Micro Java™ Game Development*. Pearson Education, Indianapolis
- Fauzan, Mohammad. (2016). *Implementasi Algoritma Fuzzy State Machine untuk perilaku non playable character pada Game First Person Shooter "Patriosm Young"*. UIN Maulana Malik Ibrahim Malang. Skripsi Jurusan S1 Teknik Informatika. Malang: Universitas Islam Negeri Maulana Malik Ibrahim Malang.
- Gil, A. (2015). *Cara Menerbangkan Drone bagi Pemula*. <http://www.gilangajip.com/cara-menerbangkan-quadcopter-pemula/>. Diakses pada 25 November 2015.
- Hani, S. (2010). *Sensor Ultrasonik SRF05 sebagai Memantau Kecepatan Kendaraan Bermotor*. *Jurnal Teknologi*, 3(2), 120–128.
- Hansson, A. (2010). *Quadrotor UAV Konstruktion och användbarhetsstudie aven UAV i sensornätverk*. Uppsala Universitet, Uppsala.
- Hardiansyah, Ade. (2014). *Rancang Bangun Quadcopter sebagai Pemantau Kampus Politeknik Negeri Sriwijaya*. Other Thesis. Politeknik Negeri Sriwijaya.

- Imansyah, P. A., T, E. S. S., & T, J. H. S. (2016). PERANCANGAN PENGATURAN PENERING TANGAN DENGAN METODE FUZZY LOGIC MENGGUNAKAN RASPBERRY-PI, 3(1), 99–104.
- Kristanto, Daniel. (2012). Rancang Bangun Pesawat Mandiri Tanpa Awak Dengan Empat Baling Baling Penggerak. Fakultas Teknik Elektronika dan Komputer, Universitas Kristen Satya Wacana, Salatiga.
- Kurniadi, Yohan, dkk. *Pembuatan Aplikasi Simulasi Ujian Praktik Pengambilan Surat Izin Mengemudi Kendaraan Roda Empat*
- Lema, Risha Anugerah Nenu. (2016). Flight Controller pada Sistem Quadcopter Menggunakan Sensor IMU (Inertial Measurement Unit) Berbasis Mikrokontroler ATMEGA 2560. Yogyakarta: Universitas Sanata Dharma.
- Luukonen, T., (2011), *Modeling and Control of Quadcopter*, Aalto University, Espoo.
- Lewus, Michael & Jacobson, Jeffrey. (2002). Game Engines in Scientific Research. *Communications of The Acm*, No. 1 Vol.45, hal 27-31
- Najib, Ahmad Ainun. (2017). *Robot Beroda untuk Menghindari Hambatan menggunakan fuzzy logic*. Skripsi Jurusan S1 Teknik Informatika. Malang: Universitas Islam Negeri Maulana Malik Ibrahim Malang.
- Novak, Jeannie. (2012). *Game Development Essentials: An Introduction*. New York: Delinar Cengage Learning.
- Rachman, Fathur Zaini, dkk. Robot Penjelajah Ruangan dengan Sensor ultrasonic dan Kendali Ganda melalui Bluetooth. *Jurnal Teknologi Terpadu No. 2 Vol 4*. Balikpapan : Politeknik Negeri Balikpapan.
- Roedavan, Rickman. (2016). Unity Tutorial Game Edisi Revisi. Bandung: Informatika.
- Saputro, Godham Eko. (2012). Simulasi dan Perkembangan *Digital Game*. godhamroadsafety.blogspot.co.id/, diakses pada tanggal 03 Juli 2012.
- Sari P. Z., Nugroho, H., Jatmiko, A., Agung, A. *Aplikasi Game Action RPG 'RUGEN THE WIGOON MASTERPIECE' Pada Platform Android Dengan Menggunakan Unity*. Skripsi Program S1 Teknik Informatika, Jakarta Barat : Universitas Bina Nusantara 2013.
- Schell, Jesse. (2008). The Art of Game Design – A Book of Lenses. Morgan Kauffman Publisher, Burlington.
- Setiawan, Iwan. (2006). Programmable Logic Controller (PLC) dan teknik Perancangan Sistem Kontrol. Yogyakarta: ANDI.
- Siswojo, Amanda Ekaratih, dkk. (2015). *Perancangan game simulasi sapi kerap sebagai upaya revitalisasi budaya kerap sapi*. Universitas Telkom.
- Sugiyono. (2013). METODE PENELITIAN KUANTITATIF, KUALITATIF DAN R&D. Bandung: Penerbit Alfabeta.

Sukma, Ridhoi Isham. (2017). *Identifikasi Ruter Kebakaran Hutan dengan Menggunakan Algoritma A* berbasis game simulasi drone (quadcopter)*. Skripsi Jurusan S1 Teknik Informatika. Malang:Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Teresa, Dillon. (2004). *Adventure Games for Learning and Storytelling*. A Futurelab prototype context paper: Adventure Author, FutureLab Report.

