IMPLEMENTASI DINAMIKA PENERBANGAN QUADCOPTER PADA LINGKUNGAN MAYA 3D

SKRIPSI

Diajukan untuk memenuhi persyaratan
Memperoleh gelar Sarjana Komputer

Disusun Oleh :
NARENDRA SULAKSONO
NIM 105060801111019

TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI DINAMIKA PENERBANGAN QUADCOPTER
PADA LINGKUNGAN MAYA 3D

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Narendra Sulaksono
NIM:105060801111019

Skripsi ini telah diuji dan dinyatakan lulus pada
8 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eriq M. Adams J., S.T., M.Kom.	Gembong Edhi Setyawan., S.T., M.T.
NIP:19850410 201212 1 001	NIK: 2012087612011001

Mengetahui
Ketua Program Studi Teknik Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 11 Januari 2015

Narendra Sulaksono

NIM.105060801111019

KATA PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi Allah SWT karena atas rahmat dan hidayahNya-lah penulis dapat menyelesaikan Tugas Akhir yang berjudul "Implementasi Dinamika Penerbangan Quadcopter Pada Lingkungan Maya 3D". Shalawat serta salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian. Tugas Akhir ini disusun untuk memenuhi sebagai persyaratan memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer Universitas Brawijaya Malang. Melalui kesempatan ini, penulis menyampaikan rasa hormat dan terima kasih penulis yang sebesar-besarnya kepada pihak yang telah memberikan bantuan lahir maupun batin selama penulisan tugas akhir ini. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Kepada orang tua penulis dan seluruh keluarga yang senantiasa tiada henti hentinya memberikan do'a demi terselesainya tugas akhir ini.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, MT dan Bapak Issa Arwani, S.Kom., M.Sc selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak / Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Bapak Eriq Muh. Adams Jonemaro, ST., M.Kom dan Bapak Gembong Edhi Setyawan, S.T., M.T selaku dosen pembimbing tugas akhir penulis.
5. Seluruh dosen Program Studi Informatika atas kesediaan membagi ilmunya kepada penulis.
6. Semua Asisten Ka. Lab serta Laboran dari Laboratorium Program Studi Informatika yang telah memberikan banyak bantuan dan dukungan dalam menyelesaikan tugas akhir ini.
7. Nur Intan Savitri, Dhio Alfanda Fanani, Bambang Ega , Anindita Setyamurti, Renno Andhika Syawaludin, Vicky Lazuardi Kautsar atas dukungan dan semangat yang di berikan.
8. Seluruh pihak yang telah membantu kelancaran penulisan tugas akhir yang tidak dapat penulisan sebutkan satu per satu.

Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun, sangat penulis harapkan. Semoga tugas akhir ini membawa manfaat bagi penyusun maupun pihak lain yang menggunakannya.

Malang, 11 Januari 2015

Penulis

ABSTRAK

Narendra Sulaksono. 2015. : IMPLEMENTASI DINAMIKA PENERBANGAN QUADCOPTER PADA LINGKUNGAN MAYA 3D. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Eriq M. Adams J., S.T., M.Kom. dan Gembong Edhi Setyawan., S.T., M.T.

Quadcopter adalah jenis helikopter yang memiliki 4 rotor. Pengembangan spesifikasi baru sebuah quadcopter memerlukan banyak waktu dan uang dikarenakan kerusakan saat melakukan uji coba penerbangan. Untuk meminimalisasi biaya uji coba ini dapat dibuat simulasi virtual tiga dimensi yang meniru pergerakan quadcopter setelah desain spesifikasi quadcopter didapatkan. Simulasi dibuat menggunakan *game engine* Unity3D dan menggunakan *physic engine* bawaan dari Unity untuk simulasi pergerakan secara fisika. Rumus mekanika klasik newton dapat digunakan untuk mensimulasikan efek dorongan dan putaran pada baling-baling virtual yang digunakan untuk menggerakan quadcopter. Hasil implementasi ini adalah sebuah simulasi virtual tiga dimensi yang meniru pergerakan quadcopter. Spesifikasi yang dapat dirubah pada simulasi ini hanya massa, kecepatan putaran maksimal dan tipe baling-baling yang dipilih. Simulasi ini diuji menggunakan teknik whitebox untuk pengujian source code yang menghasilkan hasil valid untuk perhitungan dorongan dan putaran. Selain itu diuji performa aplikasi menggunakan teknik *FPS Benchmarking* dan menghasilkan simulasi yang berjalan rata-rata dalam kecepatan 59 FPS.

Kata Kunci : Unity Game Engine, Quadcoper, Simulasi

ABSTRACT

Narendra Sulaksono. 2015. : IMPLEMENTATION OF QUADCOPTER FLIGHT DYNAMICS IN VIRTUAL 3D ENVIRONMENT. Undergraduate Thesis of Informatic Study Program, Information technology and computer science Program, Brawijaya University. Advisors : Eriq M. Adams J., S.T., M.Kom. andGembong Edhi Setyawan., S.T., M.T.

Quadcopter is a type of helicopter that have 4 rotors. Development of new specifications takes a lot of time and monetary cost because of failure in test flight. A virtual 3D simulation can be made to help reduce these costs by testing the flight virtually after the design specification has been made. This simulation is made with Unity3D game engine and using Unity physic engine to simulate the movement physics. Classical newton formula is used to simulate the thrust and torque of the spinning propellers which is used to move the quadcopter. The quadcopter specification that can be changed are total mass, maximum propellers velocity and propeller types. This simulation is tested via whitebox testing to test the validity of functions which returns all valid for test of thrust and torque functions. FPS benchmarking is used to test the performance of simulation which output 59 average FPS while the simulation is being used.

Keyword :Simulation, Unity Game Engine, Quadcopter

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL ix
DAFTAR GAMBAR x
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Ruang Lingkup Masalah 2
1.4 Tujuan 2
1.5 Manfaat 2
1.6 Sistematika Penulisan 2
BAB 2 LANDASAN KEPUSTAKAAN 4
2.1 Kajian Pustaka 4
2.1.1 Quadcopter Dynamic Modelling and Simulation (Quad-Sim) 4
2.1.2 Virtual Model of Unmanned Aerial Vehicle 7
2.2 Dasar Teori 9
2.2.1 Simulasi Komputer 9
2.2.2 Quadcopter 9
2.2.3 Game Engine Unity 17
BAB 3 METODE PENELITIAN DAN PERANCANGAN 20
3.1 Metode Penelitian 20
3.1.1 Studi Literatur 20
3.1.2 Perancangan 20
3.1.3 Implementasi 21
3.1.4 Evaluasi 21
3.2 Perancangan 22
3.2.1 Deskripsi Aplikasi 22
3.2.2 Perancangan Aplikasi 22
3.2.3 Perancangan *Quadcopter* dan Sistem Kendali 24

3.2.4 Perancangan Dinamika Quadcopter 25
BAB 4 IMPLEMENTASI 27
4.1 Pemilihan Teknologi 27
4.2 Implementasi *Scene* 28
4.2.1 *Scene* Customize 28
4.2.2 *Scene* In-Game 28
4.3 Implementasi Quadcopter 29
4.3.1 Game Object Quadcopter 29
4.3.2 Implementasi Pergerakan *Quadcopter* 30
BAB 5 PENGUJIAN DAN EVALUASI 35
5.1.1 Pengujian *Whitebox* 35
5.1.2 Pengujian Performa 41
5.1.3 Evaluasi Hasil 42
BAB 6 PENUTUP 43
6.1 Kesimpulan 43
6.2 Saran 43
DAFTAR PUSTAKA 44

DAFTAR TABEL

Tabel 3.1 Implementasi Pembuatan Aplikasi 21
Tabel 4.1 Spesifikasi perangkat keras laptop. 27
Tabel 4.2 Spesifikasi perangkat pengembang aplikasi. 27
Tabel 4.3 Source code kalkulasi dorongan *quadcopter.* 30
Tabel 4.4 *Source code method* CalcTorque. 30
Tabel 4.5 *Source code method* FixedUpdate. 31
Tabel 4.6 Source code metode Update 32
Tabel 5.1 Source Code Method CalcThrust 35
Tabel 5.2 Kasus uji untuk metode CalcThrust 36
Tabel 5.3 Source Code Method CalcTorque 36
Tabel 5.4 Kasus Uji metode CalcTorque 37
Tabel 5.5 Source Code Metode Update 38
Tabel 5.6 Kasus Uji metode update. 41
Tabel 5.7 Statistik Hasil FPS Benchmarking 42

DAFTAR GAMBAR

Gambar 2.1 Simulasi Quad-Sim saat pertama dibuka (Hartman, 2014). 5
Gambar 2.2 Isi dari block Attitude Controller (Hartman, 2014). 5
Gambar 2.3 Isi dari block Quadcopter Controller Mixing (Hartman, 2014). 6
Gambar 2.4 Representasi virtual quadcopter (Gvoth, 2014). 7
Gambar 2.5 Ruangan yang dipakai untuk menerbangkan quadcopter (Gvoth, 2014). 8
Gambar 2.6 Tata letak kendali manual pada gamepad (Gvoth, 2014). 8
Gambar 2.7 *Quadcopter* dan alat kendalinya 10
Gambar 2.8 Frame kosong sebuah *quadcopter* (Liang, 2013) 10
Gambar 2.9 *Brushless* motor tanpa menggunakan baling-baling (Liang, 2013) 11
Gambar 2.10 Gambaran potongan melintang*brushless motor* (Liang, 2013) 12
Gambar 2.11 Dua pasang baling-baling *CW dan CCW* (Liang, 2013) 12
Gambar 2.12 *Electronic Speed Controller*(Liang, 2013) 13
Gambar 2.13 *Flight Control board*(Liang, 2013) 14
Gambar 2.14 Baterai LiPo *ZIPPY* 4000 (Liang, 2013) 14
Gambar 2.15 Tiga jenis pergerakan *quadcopter*pada sumbu x,y,z (Alex, 2012). 15
Gambar 2.16 Hubungan perubahan kecepatan rotor pada pergerakan *quadcopter*. 15
Gambar 2.17 Kendali umum pada *quadcopter* 16
Gambar 2.18 Tampilan Editor Unity (Unity, 2014) 18
Gambar 2.19 Komponen *Box Collider* dalam panel *Inspector* (Unity, 2014). 19
Gambar 2.20 Komponen *Rigidbody* dalam panel *inspector* (Unity, 2014). 19
Gambar 3.1 Diagram alir metodologi pengembangan simulasi. 20
Gambar 3.2 Diagram screen-flow aplikasi 22
Gambar 3.3 Konsep tampilan *scene* customisasi 23
Gambar 3.4 Konsep tampilan *HUD* dalam simulasi 23
Gambar 3.5 Rancangan susunan input pada keyboard 24
Gambar 3.6 Flowchart dinamika pengendalian quadcopter. 25
Gambar 4.1 *Screenshot* tampilan *scenecustomize.* 28
Gambar 4.2 *Screenshot scene in-game* 28
Gambar 4.3 Susunan *hierarchygame object quadcopter*. 29
Gambar 5.1 Flow graph dari metode CalcThrust. 35
Gambar 5.2 Flow graph dari metode CalcTorque 37
Gambar 5.3 Hasil *FPS Benchmarking* 42

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Quadcopter merupakan salah satu jenis helikopter yang memiliki empat buah ruotor. Berbeda dengan helikopter biasa, *quadcopter* memiliki empat buah rotor dimana satu pasang berputar searah jarum jam dan pasang lainnya berputar berlawan dengan arah jarum jam. Gerakan yang dilakukan oleh *quadcopter* dilakukan dengan merubah kecepatan putaran satu atau lebih rotor (Stafford, 2014).

Dahulu *quadcopter* memiliki ukuran yang besar, serta sistem kendali dan stabilitas yang rumit untuk seorang atau lebih pilot membuat penelitian dalam bidang quadcopter dihentikan, tetapi Melalui perkembangan teknologi mikrokontroler, desain dari *quadcopter* menjadi lebih kecil dan lincah dibandingankan pendahulu-pendahulunya. Penambahkan mikrokontroler sebagai sistem kendali elektrik dan sensor untuk menjaga stabilitas serta dengan ukuran dan kemampuan bermanuver yang tinggi membuat *quadcopter* dapat digunakan bukan hanya didalam tetapi juga diluar ruangan (Hoffman, 2004)

Quadcopter mulai digunakan oleh berbagai kalangan mulai dari peminat model *RC (Remote Controlled)*, peneliti, komersil sebagai alat observasi, bahkan pihak militer dalam operasi-operasi militernya. Melalui perkembangan teknologi kecerdasan buatan, *Quadcopter* mulai ditambahkan sistem kendali otomatis yang membuat pemanfaatan *quadcopter* semakin luas.

Perancangan dan pengembangan model *quadcopter* baru membutuhkan waktu yang lama serta biaya yang besar. Hal ini dikarenakan banyaknya percobaan, perhitungan dan pengkondisian kembali yang perlu dilakukan ketika model yang dicoba gagal untuk terbang.

Simulasi merupakan metode pelatihan yang sering digunakan dengan cara membuat keadaan atau alat tiruan yang memiliki sifat seperti sesungguhnya. Pembuatan simulasi memerlukan model yang memiliki sifat yang sama seperti alat, sistem atau proses yang akan disimulasikan (Banks, 2001).

Simulasi sudah banyak digunakan dalam berbagai kegiatan, diantaranya pengujian dan optimasi performa, pelatihan, edukasi, serta hiburan. Simulasi dapat digunakan untuk menunjukkan efek-efek yang seharusnya timbul, mencoba alternatif-alternatif dan dampaknya dalam sebuah sistem atau operasi, mencoba sebuah sistem yang tidak dapat dicoba secara langsung pada lingkungan yang seharusnya ataupun mencoba alat yang masih dalam tahap perancangan (Sokolowski, 2009).

Masalah dalam pengembangan quadcopter yaitu waktu yang lama untuk merakit quadcopter dan mencobanya serta biaya besar yang dikeluarkan selama proses percobaan tersebut dapat dihindari jika quadcopter dapat diujicoba dahulu pada lingkungan virtual sebelum rancangannya dirakit. Oleh karena itu teknologi simulasi dapat dimanfaatkan selain untuk mempelajari cara

quadcopter terbang, juga dapat menguji spesifikasi rancangan *quadcopter* sebelum memasuki tahapan pembuatannya.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas maka dapat diambil rumusan masalah sebagai berikut:

1. Bagaimana implementasi dinamika penerbangan quadcopter pada lingkungan maya 3D?
2. Bagaimana fungsionalitas hasil implementasi dinamika pergerakan *quadcopter*?
3. Bagaimana mengetahui performa hasil aplikasi implementasi dinamika pergerakan quadcopter?

1.3 Ruang Lingkup Masalah

1. Faktor yang mempengaruhi pergerakan *quadcopter* adalah massa, motor dan baling-baling dan gaya gravitasi.
2. Faktor selain yang disebutkan diatas (tekanan udara, angin) tidak diperhitungan / dihiraukan.

1.4 Tujuan

Tujian dari penelitian ini adalah untuk merancang, mengimplimentasi dan menguji aplikasi simulasi pergerakan quadcopter 3D dalam menggunakan *game engine* Unity.

1.5 Manfaat

Penulisan skripsi ini diharapkan dapat bermanfaat untuk berbagai pihak.

1. Bagi Penulis
    a. Mengaplikasikan ilmu yang didapat selama mengikuti perkuliahan di Teknik Informatika Universitas Brawijaya
    b. Mendapatkan pemahaman tentang perancangan dan pengembangan simulasi pada *platform* PC menggunakan bantuan *game engine Unity.*
2. Bagi Pengguna
    a. Memberi sarana hiburan dan pendidikan secara bersamaan pada *platform* PC.
    b. Menyediakan simulasi yang memberi gambaran cara kerja *quadcopter*.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

1. Pendahuluan

Dalam bab ini diuraikan mengenai latar belakang, perumusan masalah, ruang lingkup masalah, tujuan, manfaat hasil penelitian, dan juga menjelaskan sistematika penulisan.

2. Landasan Kepustakaan

Bab ini menguraikan tentang teori yang berhubungan dengan judul kerja praktik terkait dalam penyusunan aplikasi simulasi dan proses pengembangannya.

3. Metode Penelitian dan Perancangan

Bab ini berisi tentang metodologi yang akan digunakan dalam penelitian ini secara terperinci, dan juga hal yang terkait dengan perancangan aplikasi dari penentuan judul hingga sebelum proses implementasi. Tersusun secara struktural.

4. Implementasi

Bab ini berisikan tentang proses implementasi / penerapan yang akan dilakukan terhadap aplikasi yang dikembangkan secara terperinci dan berurutan. Selain itu akan dibahas evaluasi aplikasi serta proses pengujian aplikasi secara terperinci yang akan akan dilakukan setelah proses implementasi selesai dilakukan.

5. Pengujian dan Evaluasi

Bab ini Berisikan pengujian yang dilakukan kepada aplikasi untuk mengetahui apakah aplikasi telah berjalan sesuai dengan seharusnya. pengujian ini terdiri atas pengujian yang menggunakan *test suite* dan pengujian yang menggunakan *FPS Benchmarking*.

BAB 2 LANDASAN KEPUSTAKAAN

Bab 2 menjelaskan tentang apa saja dasar-dasar yang mendukung pengembangan simulasi ini.Pada bab inidijelaskanistilah-istilah yang terkait pada penelitian.

2.1 Kajian Pustaka

Pada Bagian ini dibahas secara singkat jurnal penelitian sebelumnya yang digunakan sebagai bahan acuan pengembangan simulasi. Jurnal ini digunakan sebagai referensi fitur dan mekanisme simulasi.

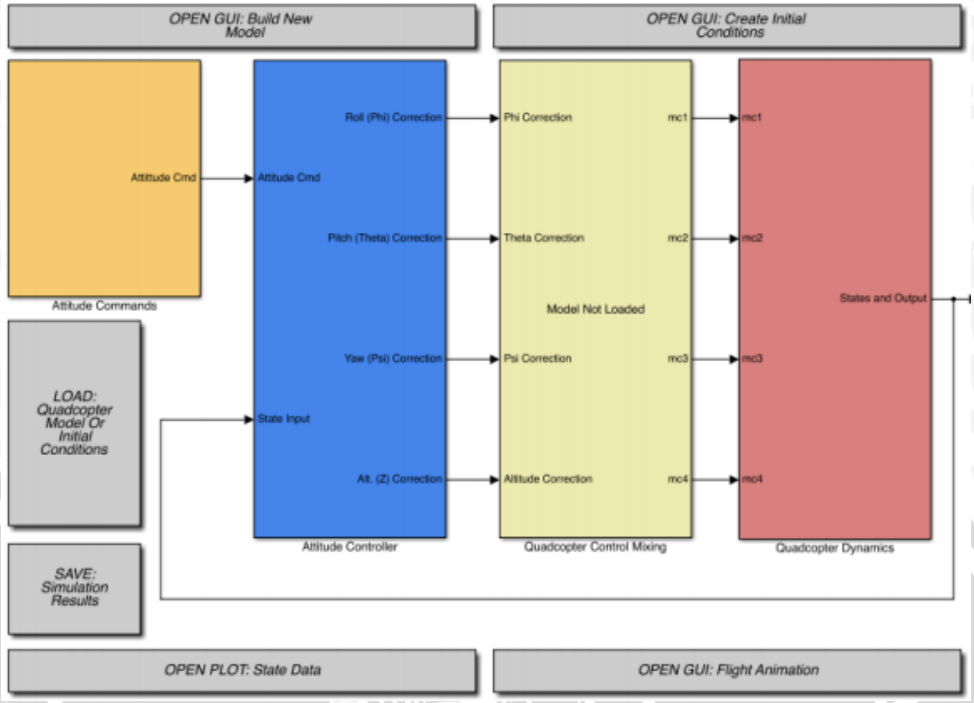
2.1.1 Quadcopter Dynamic Modelling and Simulation (Quad-Sim)

Quad-Sim merupakan sebuah simulasi tentang pemodelan pergerakan *quadcopter* yang berbasis MATLAB dan Simulink. Selain pemodelan pergerakan, Quad-Sim juga menyediakan simulasi controller PID dan PD yang dapat digunakan untuk simulasi pengendali keseimbangan quadcopter.Simulasi ini dikembangkan oleh tim yang berasal dari Drexel University (Hartman, 2014).

Simulasi ini terbuat dari beragam pendekatan *modelling* sifat *quadcopter*. Banyak efek – efek penting yang tidak disimulasikan ataupun disederhanakan (contohnya efek aerodinamis dan efek *blade flapping* dari baling-baling) (Hartman, 2014).

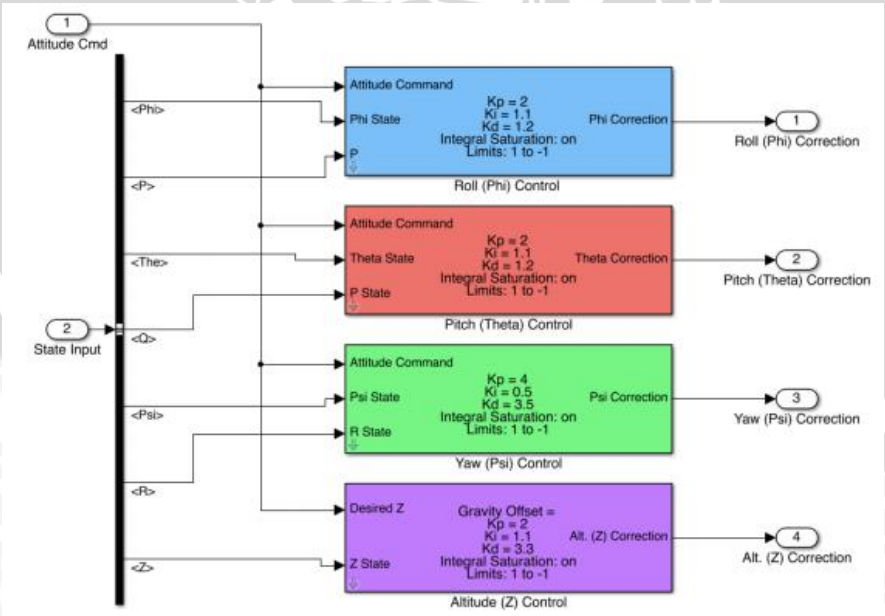
Langkah – langkah menggunakan simulasi quadsim adalah sebagai berikut:

1. Buka file AC_Quadcopter_Simulation.slx
2. Dalam simulasi ini posisi quadcopter tidak diperhatikan, simulasi ini hanya memperhatikan attitude dan altitude quadcopter menggunakan PID controller. Ketika model pertama kali dibuka akan muncul gambar seperti Gambar 2.1.
3. Kotak berwarna abu-abu adalah tombol yang ketika di *double-click*akan dieksekusi sesuai tulisan didalamnya.
4. Double-click tombol LOAD dan pilih sebuah kondisi awal (IC Structure) contohnya “Hover.mat”. double-click tombol load lagi dan pilih model quadcopternya (Quadmodel Structure) contohnya “quadModel_+”.
5. Setelah di load kedua file tersebut simulasi dapat dijalankan.



Gambar 2.1 Simulasi Quad-Sim saat pertama dibuka (Hartman, 2014).

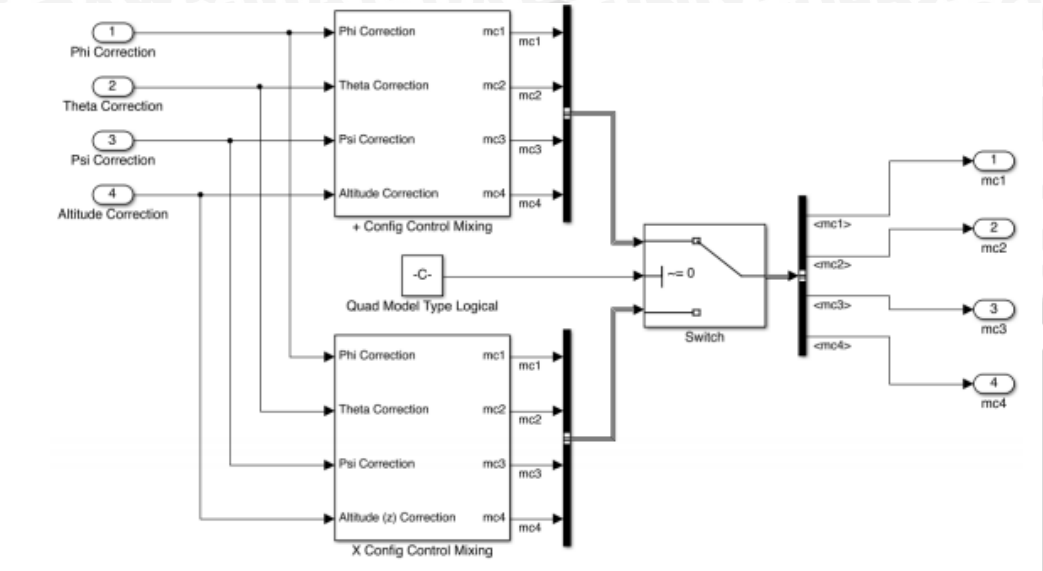
Beberapa hal yang dapat dilakukan sebelum aplikasi dijalankan adalah melihat isi dari block-block yang ada pada multisim. Contohnya ketika block attitude controller di double click akan keluar tampilan seperti gambar 2.2.



Gambar 2.2 Isi dari block Attitude Controller (Hartman, 2014).

Angka yang berada didalam block dapat dirubah dengan cara melakukan double-click block yang diinginkan untuk menampilkan interface untuk mengganti nilai yang ada.

Block Quadcopter Controller Mixing berfungsi untuk merubah koreksi dari phi, theta, psi dan z dari Attitude Controller dan merubahnya menjadi koreksi untuk tiap motor. Dalam block ini terdapat 2 block terpisah untuk konfigurasi Quadcopter + dan X. Block yang aktif ditentukan oleh sebuah switch sesuai Gambar 2.3



Gambar 2.3 Isi dari block Quadcopter Controller Mixing (Hartman, 2014).

Block Quadcopter Dynamics terdiri dari 2 block utama yaitu Motor Dynamics yang mengatur throttle motor dan menghasilkan output RPM untuk tiap motor dan State Equations yang merupakan inti dari simulasi.

Hasil dari simulasi dapat dilihat sebagai plot penerbangandengan cara double-click tombol OPEN PLOT atau melalui gambaran animasi penerbangan melalui tombol OPEN GUI.

Kelebihan

- Perhitungan kendali dapat dikustomisasi pengguna.
- Pengguna bebas melakukan kustomisasi aplikasi.
- Aplikasi diarahkan untuk penelitian kestabilan atau otomatisasi.

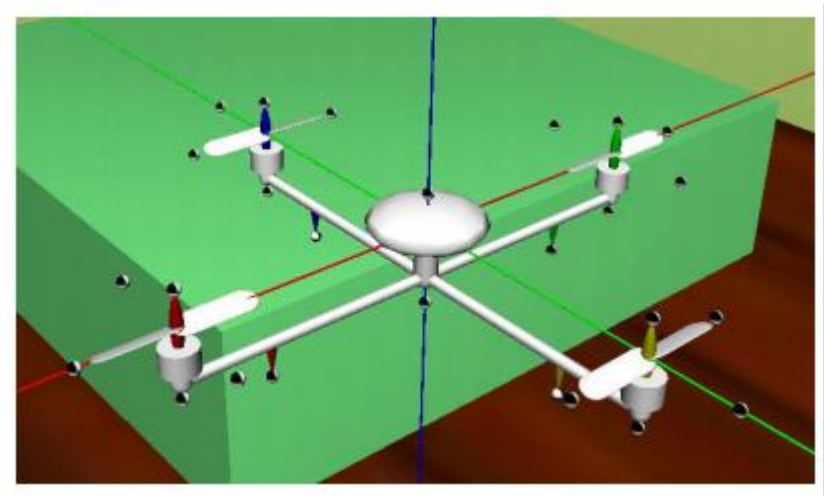
Kekurangan

- Gambaran penerbangan hanya dalam bentuk *chart* dan *graph*.
- Quadcopter tidak dapat dikendalikansecara manual oleh pengguna.
- Aplikasi terlalu rumit untuk pengguna awam quadcopter.

2.1.2 Virtual Model of Unmanned Aerial Vehicle

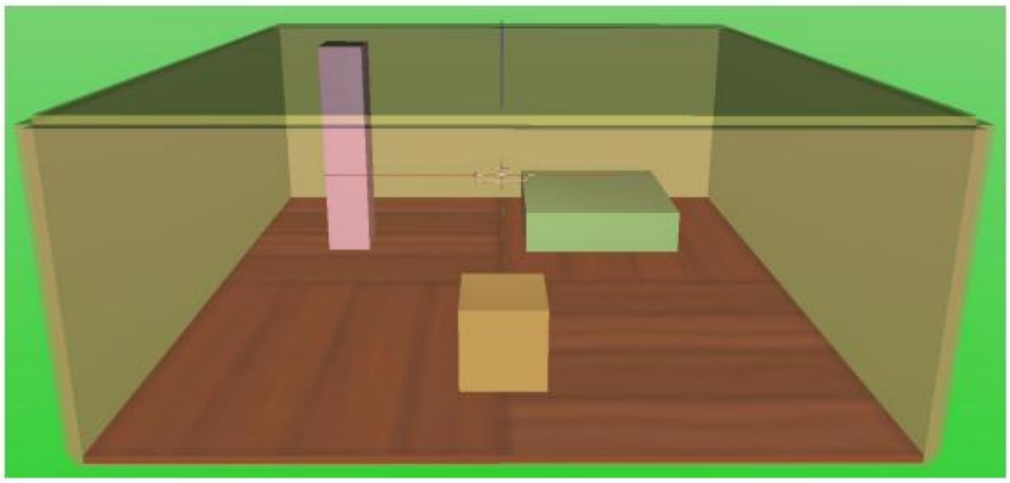
Banyak model simulasi quadcopter yang dapat digunakan untuk pembelajaran, tetapi kebanyakan menghasilkan output hanya berupa graph. Penggunaan bahasa VRML (Virtual Reality Modelling Language) memungkinkan penggambaran objek menggunakan lingkungan berbasis virtual reality 3D.Perkembangan alat simulasi MATLAB Simulink memungkinkan penggunaan bahasa VRML pada scene environmentnya. Simulink 3D Animation Extension dapat menyambung block pada Simulink dengan gambaran 3D. Paper ini membuat desain virtual model dari quadcopter. Persamaan matematis digunakan untuk mengembangkan model simulasi quadcopter. Quadcopter akan digambarkan menggunakan bahasa VRML dan kendali manual akan menggunakan sebuah *joystick*. Penggunaan model ini dapat membantu dalam proses pelatihan kendali dan percobaan algoritma kendali (Gvoth, 2014).

Desain fungsional dari virtual model dibuat di VRML scene dengan parent object bernama “core” dan memiliki 3 fungsional child object “rot_x”, “rot_y” dan “rot_z” yang berfungsi untuk membantu perubahan arah objek. Representasi virtual quadcopter dibuat seperti Gambar 2.4.



Gambar 2.4 Representasi virtual quadcopter (Gvoth, 2014).

Ruang untuk menerbangkan quadcopter juga dibuat. Ruang untuk menerbangkan quadcopter memiliki dimensi 5x5x2 m. titik tengah ruangan adalah koordinat [0,0,0]. Tiga halangan yang memiliki dimensi berbeda ditempatkan didalam ruangan. Quadcopter dapat menabrak ketiga objek rintangan, tembok, lantai dan langit-langit. Tembok dan langit-langit diberikan transparansi sebanyak 50% agar memudahkan pengguna melihat objek quadcopter.



Gambar 2.5 Ruangan yang dipakai untuk menerbangkan quadcopter (Gvoth, 2014).

Quadcopter dapat dikendalikan secara manual menggunakan sebuah *gamepad* Genius G-12X.*Gamepad* ini memiliki *trackpad* empat arah dan dua belas tombol yang dapat digunakan sebagai kendali. Tata letak tombol kendali dapat dilihat pada Gambar 2.6.

Gambar 2.6 Tata letak kendali manual pada gamepad (Gvoth, 2014).

Kelebihan

- Penerbangan quadcopter dapat dilihat secara 3D.
- Quadcopter dapat dikendalikan secara manual

Kekurangan

- Ruang pengoperasian quadcopter sangat terbatas.
- Kualitas 3D yang dihasilkan masih sederhana.

2.2 Dasar Teori

Pada Bagian ini dituliskan teori yang digunakan sebagai dasar pengembangan simulasi.

2.2.1 Simulasi Komputer

Simulasiadalah peniruan operasi sebuah proses atau sistem. Simulasi memerlukan sebuah model yang dapat berperilaku atau memiliki ciri khas dari proses atau sistem yang akan ditiru (Banks, 01).

Simulasi komputer adalah simulasi yang dijalankan pada sebuah komputer atau pada sebuah jaringan komputer untuk meniru tingkah laku sebuah sistem.Simulasi ini menggunakan model abstrak (*Computer* atau *Computational Model*) untuk menyimulasikan sistem. Simulasi komputer berguna dalam pemodelan matematis sistem – sistem alam dalam ilmu fisika, astrofisika, klimatologi, kimia, biologi dan lainnya (Strogatz, 2007).

2.2.2 Quadcopter

Quadcopter atau disebut juga *Quadrotor*, adalah sebuah helikopter yang memiliki empat rotor (Hoffman, 2004). Berbeda dengan helikopter biasa, Quadrotor menggunakan dua set baling-baling *fixed-pitch* kembar dimana dua berputar searah jarum jam dan dua lainnya berlawanan jarum jam. Pergantian RPM dari rotor digunakan untuk mengendalikan dorongan dan torsi. Pengendaliannya dapat diatur dari pergantian kecepatan satu atau lebih rotornya (Stafford, 2014).

Keempat rotormenghadap keatas dan ditempatkan dalam formasi persegi empat sama sisi.Desain *quadcopter* yang unik menyebabkan penerbangangan yang lebih stabil dengan ukuran yang lebih kecil jika dibandingkan dengan helikopter biasa, hal ini menyababkan *quadcopter* ideal untuk kegiatan *aerial photography* dan *surveillance*.*Quadcopter*juga merupakan desain yang paling banyak digunakan dalam pembuatan pesawat tanpa awak (*UAV, Unmanned Aerial Vehicle)*.

Perkembangan teknologi *Quadcopter* sempat terhenti dikarenakan rumitnya pengendalian empat rotor yang bekerja secara independen tanpa menggunakan bantuan elektronik. Perkembangan teknologi dan turunnya harga dari mikroprosesor membuat pengendalian elektronik ataupun pengendalian otomatis *Quadcopter* dapat dilakukan oleh militer, komersil, bahkan hanya sebagai hobi.

Gambar 2.7 *Quadcopter* dan alat kendalinya (sumber : http://webstartforum.com/)

Quadcopter terdiri dari beberapa komponen antara lain *frame,* motor*,* baling-baling, baterai dan *flight controller*, penjelasan dari masing-masing komponen adalah sebagai berikut (Liang, 2013).

a. Frame

Frame merupakan sebuah kerangka dimana perangkat-perangkat lain *quadcopter* disambungkan. *Framequadcopter* terdiri dari *arms*, *center plate* dandapat memiliki*landing skid*.*Arms* merupakan lengan *quadcopter* dimana *ESC*, baling-baling dan motor akan dipasang. Semua *arms* tersambung pada titik tengah *center plate* dimana akan diletakkan *flight controller, receiver,* sensor dan baterai. Dalam pemilihan *frame* harus diperhatikan berat, ukuran dan bahan dari *frame*.*Landing Skid* adalah bagian dari frame untuk meringankan beban saat pendaratan. Penggunaan *landing skid* bersifat opsional.

Gambar 2.8 Frame kosong sebuah *quadcopter* (Liang, 2013)

Framequadcopter kebanyakan terbuat dari carbon fiber, aluminium dan kayu seperti *plywood* atau *MDF (Medium-Density Fibreboard)*. Frame berbahan aluminium merupakan yang paling popular karena beratnya yang relative ringan, rigid dan memiliki harga yang terjangkau. Tetapi bahan aluminium memiliki kekurangan yaitu kurangnya redaman akan menyababkan banyaknya getaran yang berasal dari motor. Getaran inidapatmengacaukan perhitungan dari sensor.

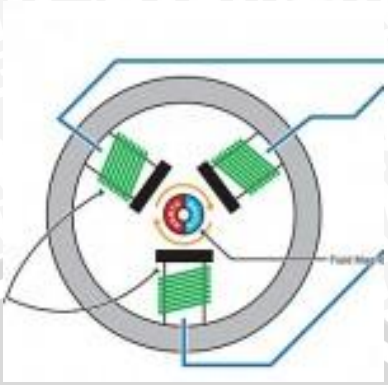
Frame berbahan kayu seperti *MDF* memiliki redaman yang lebih bagus daripada aluminium. Tetapi kayu bukan merupakan bahan yang keras dan dapat patah dengan mudah jika *quadcopter* jatuh. Carbon fiber merupakan bahan paling keras dan memiliki redaman bagus jika dibandingkan dengan kayu dan aluminium, satu-satunya kekurangan dari carbon fiber adalah harganya yang lebih mahal disbanding bahan yang lainnya.

b. Motor

Motor atau memiliki nama lain rotor merupakan alat pemutar baling-baling. Motor yang biasa digunakan untuk quadcopter adalah *brushless motor* yang memiliki sedikit gesekan.

Gambar 2.9 *Brushless* motor tanpa menggunakan baling-baling (Liang, 2013)

Brushless motor tampak seperti motor DC biasa yaitu menggunakan kumparan dan magnet untuk memutar tangkainya. Perbedaannya yaitu pada brushless motor tidak memiliki *brush* pada tangkainya yang biasanya digunakan untuk mengubah arah energi pada kumparan. Untuk menggantikan *brush* pada *brushless motor*digunakan tiga kumparan didalam motor. Umumnya motor *brushless* berputar lebih cepat dan menggunakan energy lebih sedikit jika dibandingkan dengan motor DC, selain itu dalam motor *brushless* tidak terjadi kehilangan energy yang disebabkan oleh *brush* pada motor DC.



Gambar 2.10 Gambaran potongan melintang*brushless motor* (Liang, 2013)

Notasi terpenting yang harus diperhatikan dalam pemilihan brushless motor adalah besarnya nilai Kv. Kv merupakan notasi yang menunjukkan banyaknya putaran per menit (rpm) tanpa beban ketika diberikan perbedaan tegangan sebanyak 1v.

c. Baling-baling

Pada quadcopter baling-baling terpasang pada masing-masing motor. Baling-baling yang digunakan dalam satu frame biasanya terdiri dari dua jenis yaitu dua buah baling-baling yang berputar searah jarum jam (*CW*) dan dua buah yang berputar melawan jarum jam (*CCW*).

Gambar 2.11 Dua pasang baling-baling *CW dan CCW* (Liang, 2013)

Baling-baling memiliki diameter dan *pitch* (kemiringan) yang beragam. Pemilihan baling-baling didasari oleh ukuran *frame* yang digunakan dan pemilihan motor akan dipengaruhi oleh pemilihan baling-baling yang digunakan.Secara umum baling-baling dapat dipilih dengan memperhatikan beberapa hal yaitu:

1. Semakin besar diameter dan *pitch* sebuah baling-baling semakin besar pula dorongan yang dapat dihasilkan. Tenaga yang diperlukan untuk memutar baling-baling tersebut akan bertambah, tetapi akan dapat mengangkat beban yang lebih besar.
2. RPM yang tinggi akan meningkatkan kecepatan dan *maneuverability* tetapi kemampuan mengangkat beban akan sangat terbatas.
3. Dalam memilih baling-baling harus diperhatikan fungsi dari *quadcopter* apakah akan digunakan untuk penerbangan stabil dan mengangkat beban berat seperti kamera biasanya digunakan baling-baling yang memiliki diameter dan *pitch* yang tinggi dan RPM yang tidak terlalu tinggi, sedangkan jika dibuat untuk melakukan manuver-manuver rumit lebih baik menggunakan motor dengan RPM tinggi dan baling-baling dengan diameted dan *pitch* rendah.

d. Electronic Speed Controller

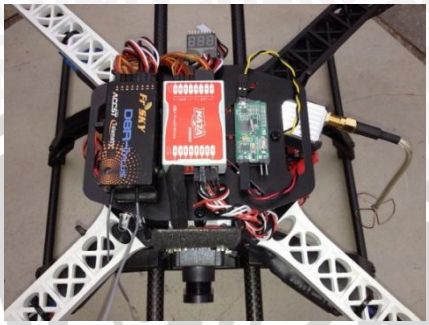
Gambar 2.12 *Electronic Speed Controller*(Liang, 2013)

Electronic Speed Controller atau biasa disingkat sebagai *ESC* adalah sebuah papan pengendali motor yang memiliki input sebuah batere dan output berupa tiga fase untuk mengendalikan motor.*ESC* menghasilkan tiga sinyal berfrekuensi tinggi dengan fase yang berbeda dan terkendali yang akan menyebabkan motor untuk tetap berputar.

Faktor terpenting dalam memilih ESC adalah sumber arus masukannya. Faktor terpenting kedua adalah kemampuan untuk diprogram, sebagian ESC dapat diprogram untuk menggunakan frekuensi sinyal yang tidak diantara 1ms sampai 2ms.

e. Flight Controller

Flight Controller merupakan papan yang menampung *microcontroller* dan beberapa sensor untuk menjaga kestabilan sebuah *quadcopter*.



Gambar 2.13 *Flight Control board*(Liang, 2013)

Beberapa *flight controller* memiliki fitur-fitur tambahan seperti:

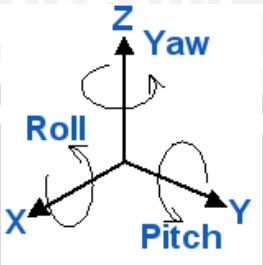
- Kemampuan untuk menstabilkan kembali quadcopter jika *pitch* ataupun *rollstick* dilepas.
- Kemampuan untuk melakukan hover di ketinggian tertentu tanpa perlu mengendalikan *throttle* secara manual.
- Kemampuan untuk melakukan *hover* di lokasi tertentu.
- Kemampuan untuk kembali ke titik dimana *quadcopter* memulai *take-off*.
- Kemampuan untuk mengikuti *waypoint*.

f. Baterai

Gambar 2.14 Baterai LiPo *ZIPPY* 4000 (Liang, 2013)

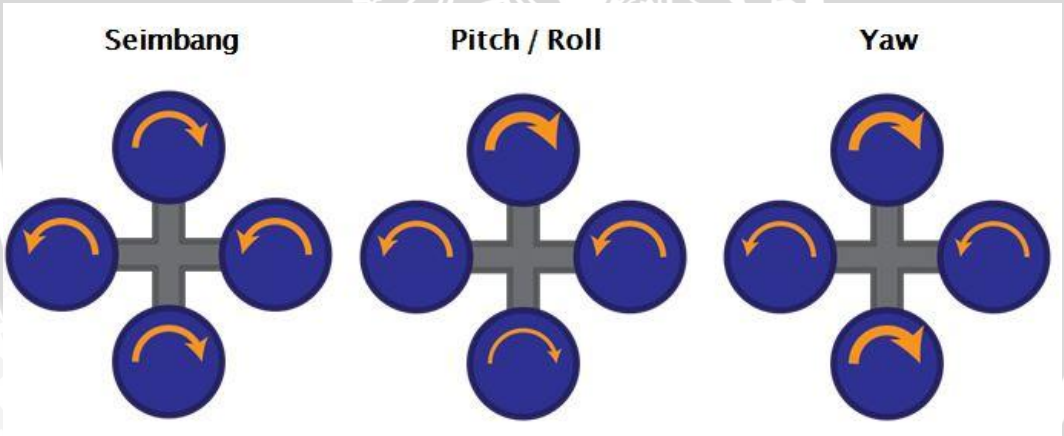
Baterai yang dianjurkan untuk *quadcopter* adalah baterai LiPo karena ringan dan arus yang dihasilkan memenuhi kebutuan untuk memutar motor *brushless*. Dalam memilih baterai LiPo yang beberapa hal yang perlu diperhatikan antara lain Voltase, Kapasitas Baterai dan *discharge rate*.

Quadcopter beroperasi dengan memutarkan keempat rotornya, dimana dua rotor yang berseberangan berputar searah jarum jam dan dua lainnya berputar berlawanan dengan arah jarum jam. Manuver pada quadcopter pada dasarnya terdiri dari 3 pergerakan yaitu *pitch, roll* dan *yaw* (Alex, 2012).



Gambar 2.15 Tiga jenis pergerakan *quadcopter*pada sumbu x,y,z (Alex, 2012).

Manuver *quadcopter* dilakukan dengan cara merubah kecepatan putaran pada satu atau lebih rotor.. Jika keempat rotor berputar dengan kecepatan yang sama maka *quadcopter* seharusnya akan melayang secara stabil. Jika salah satu rotor dipercepat atau diperlambat maka pergerakan *pitch / roll* akan terjadi. Jika 2 rotor yang berputar searah dipercepat atau diperlambat maka akan terjadi pergerakan *yaw.*



Gambar 2.16 Hubungan perubahan kecepatan rotor pada pergerakan *quadcopter (sumber : https://en.wikipedia.org/wiki/Quadcopter)*

Pergerakan *quadcopter* ini dikendalikan pengguna menggunakan kendali jarak jauh yang memiliki 2 *joystick*. *Joystick* kiri jika digerakan secara vertikal akan mengendalikan kecepatan putaran secara keseluruhan yang digunakan untuk mengatur ketinggian terbang, sedangkan gerakan horizontal digunakan untuk mengendalikan gerakan *yaw*. *Joystick* kanan pada gerakan vertikal mengendalikan *pitch* dan gerakan horizontal mengendalikan gerakan *roll*.

Gambar 2.17 Kendali umum pada *quadcopter*

Sebuah *quadcopter* terbang dengan memutarkan baling-baling sehingga putaran tersebut mengeluarkan dorongan. Dorongan ini merupakan sebuah gaya (*force)* yang orthogonal terhadap baling-baling. Gaya ini akan mendorong badan *quadcopter* searah dengan arah gaya. Besarnya gaya tiap baling-balingdapat diketahui melalui persamaan berikut (Khan, 2014).

$$|T_i| = \rho A v_i^2$$

(2.1)

Gaya T untuk baling-baling i dapat diketahui dari parameter v (kecepatan putaran), ρ (kepadatan udara) dan A (luas penampang).Pada saat take-off semua motor akan berputar menghasilkan dorongan yang sama ke satu arah. Jika diketahui semua motor berputar dengan kecepatan putaran yang sama maka jumlah dorongan pada *quadcopter* adalah:

$$netT = \rho A \sum_{i=1}^{4} [v_i]^2$$

(2.2)

Quadcopter akan melayang (*hovering)* saat semua gaya yang bekerja pada badan *quadcopter* adalah 0.Jika diketahui gaya yang bekerja pada *quadcopter* adalah gaya tiap baling-baling yang searah dan gaya gravitasi yang mengarah selalu kearah daratan bumi, dan gaya baling-baling dianggap berlawanan arah dengan arah gaya gravitasi maka *hovering* akan didapat saat jumlah semua dorongan baling-baling yang didapat pada persamaan 2.2 sama besar dengan gaya tarik gravitasi (g).

$$netT = g$$

(2.3)

Manuver *yaw* pada *quadcopter* berbeda dengan manuver lain karena tidak menggunakan dorongan pada baling-baling, tetapi menggunakan torsi yang dihasilkan dari gesekan putaran baling-baling dengan udara. Perhitungan besarnya putaran dalam manuver yaw dapat diketahui melalui persamaan 2.4 berikut.

$$\tau B_z = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

(2.4)

Persamaan 2.4 merupakan persamaan torsi yang terjadi pada badan quadcopter dalam sumbu z. b adalah koefisien gesek. ω merupakan kecepatan putaran baling-baling dalam satuan rad/detik (Gibiansky, 2012).

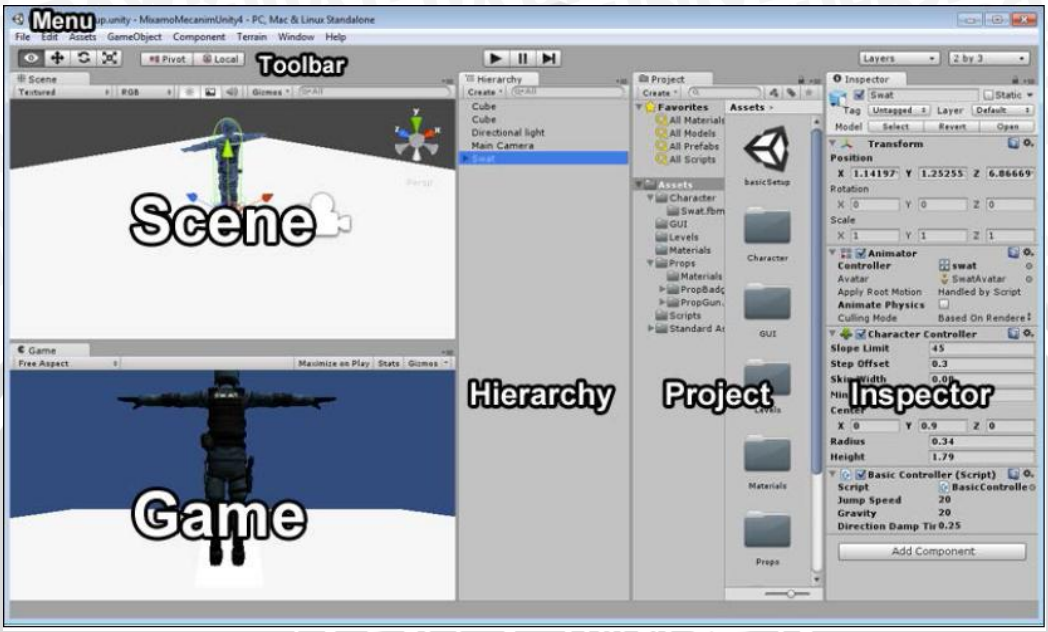
2.2.3 Game Engine Unity

Unity adalah sebuah *game engine* 3D *multi-platform*yang dikembangkan oleh Unity Technologies.Unity dapat digunakan untuk membuat dan mengembangkan *game* dalam berbagai macam platform seperti Windows, Mac, Linux, iOS, Android, Xbox 360, Wii U, PS3 serta *web-based* dengan menggunakan *plugin* Unity. Unity memiliki dua versi yaitu Unity *pro* dan Unity *free*. Unity *free* dapat digunakan secara bebas tanpa biaya untuk individu atau perusahaan yang memiliki pendapatan kotor tahunan dibawah US$100.000. Unity *pro* walaupun memerlukan biaya untuk pemakaiannya menawarkan lebih banyak fitur untuk mempermudah pembuatan dan efisiensi *game*. Unity Technologies membuat gratis semua fitur engine dengan dirisilisnya Unity 5.0 pada tanggal 13 maret 2015.

Fitur yang ditawarkan oleh unity antara lain *portability*, *graphic engine* untuk beberapa API, *Physics Engine* berbasis NVDIA PhysX dan*scripting* berbasis Mono, yaitu implementasi *open-source* dari *framework* .NET. Programmer dapat melakukan *Scripting* dalam Bahasa UnityScript (menyerupai JavaScript), C# atau Boo (syntax menyerupai Python) (Chu, 2010).

Game dalam *unity* terdiri dari satu atau lebih *scene*, dimana dalam setiap *scene* dapat terdiri dari banyak "*game object".* G*ame object* merupakan sebuah penampung untuk satu atau lebih "*component*". *Component* merupakan implementasi fungsional yang menyebabkan tiap *game object* memiliki sifat dan karakteristik. Beberapa contoh *component* dalam unity antara lain *transform* yaitu posisi, rotasi dan skala sebuah objek, *light* yang memberikan efek cahaya pada sebuah objek atau *script* yang telah dibuat oleh programmer.

Dalam antarmuka unity terdapat beberapa mode tata letak yang dapat di pilih. Tata letak *Unity* selalu dimulai dengan mode *Wide*, hal ini berbeda apabila pengguna telah merubahnya. Cara merubah nya pada akses ke *Window-Layouts* dan memilih opsi lain, seperti 4 *Split* atau 2 *by 3*, dan perhatikan bagaimana antarmuka diatur dalam Gambar 2.18menggunakan opsi *layout* "2 *by* 3"

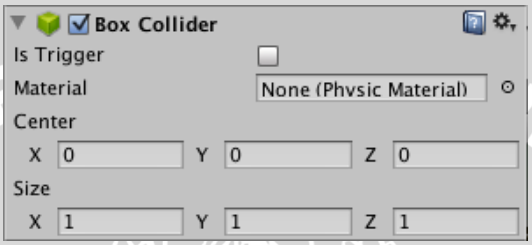


Gambar 2.18 Tampilan Editor Unity (Unity, 2014)

a. **Scene:** Tampilan ini digunakan untuk *position*, *rotate*, *scale*, dan *select* objek pada *game* dan juga mengarahkan level pengguna.
b. **Game:** Tempat untuk memainkan dan menguji *game*. Ini akan menghasilan pengalaman bermain dari *player* seakurat mungkin.
c. **Hierarchy:** Berisi list objek-objek yang ada di *game* (seperti karakter, kamera, level geometri, pencahayaan, dan tekstur GUI)
d. **Project:** Tempat untuk membuat, mengatur, dan mengakses asset-aset *game* dari model 3D dan tekstur 2D sampai *script* C#, seluruh elemen akan dilist disini.
e. **Inspector:** Melakukan konfigurasi berbagai objek *game* (dari tampilan *Hierarchy*) atau asset (dari tampilan *Project*). Termasuk merubah peraturan ***Transform*,** mengkonfigurasi komponen yang telah ada dan membuat komponen baru. Pada *inspector* juga dapat dilakukan penyesuaian preferensi lain untuk *game*.
f. **Toolbar:** Termasuk merubah alat yang digunakan untuk memanipulasi objek permainan dan navigasi adegan, alat control yang digunakan untuk bermain/berhenti dan menghentikan level, dan alat *drop-down* yang digunakan untuk mengelola *layers* dan *layouts*.
g. **Menu:** Memberikan akses ke beragam daftar perintah yang meliputi impor/ekspor asset, pengaturan preferensi, pembuatan objek permainan, medan, tata letak, dan dokumentasi.

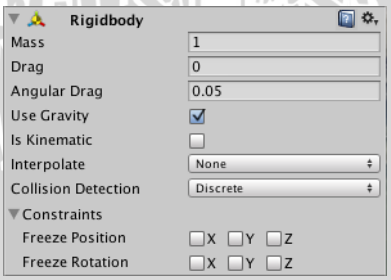
Physics Engine merupakan salah satu fitur yang ditawarkan oleh Unity. *Physics engine* yang digunakan oleh Unitymenggunakan*SDKmiddleware* Nvidia PhysX. Penggunaan *physics engine* ini mempermudah pemakai Unity dalam menerapkan sistem fisika dengan cara menyediakan *components* yang bekerja berdasarkan mekanika klasik (hukum fisika newton). Beberapa *components* yang disediakan Unity antara lain adalah *collider* dan *rigidbody*.

Collider merupakan sebuah *component* yang disediakan Unity untuk menandakan bahwa objek tersebut dapat berinteraksi dengan objek lain yang memiliki *collider.Collider* dalam Unity terbagi menjadi beberapa jenis tergantung pada bentuk primitifnya yaitu *Box, Capsule, Sphere, Terrain dan Mesh collider.*



Gambar 2.19 Komponen *Box Collider* dalam panel *Inspector* (Unity, 2014).

Box, Capsule dan *Sphere collider* merupakan *collider* primitif yang berbentuk kubus, kapsul dan bola, *terrain collider* berfungsi untuk mengimplentasikan tabrakan dimana bentuknya mengikuti objek datar*terrain* yang digunakan.*Mesh collider* adalah *collider* yang bentuknya akan menyerupai *mesh* dari objeknya terpasang, hal ini akan membuat tubrukan yang lebih presisi tetapi akan memiliki pemrosesan yang lebih berat jika dibandingkan penggunaan *collider* primitif.



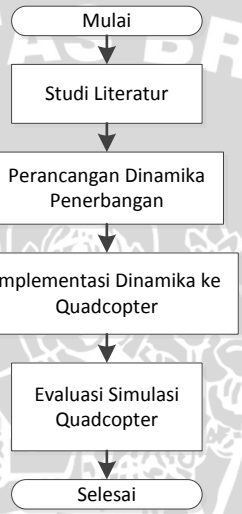
Gambar 2.20 Komponen *Rigidbody* dalam panel *inspector* (Unity, 2014).

Rigidbody merupakan komponen yang disediakan Unity untuk menandakan bahwa sebuah *GameObject* dapat dikendalikan melalui *physics engine.Rigidbody* dapat menerima gaya (*forces*) dan torsi (*torques*) yang akan mempengaruhi gerakan objek tersebut yang dapat berasal dari gravitasi, *scripting* atau tubrukan dengan objek yang memiliki *collider* jika objek *rigidbody* juga memiliki *collider*.

BAB 3 METODE PENELITIAN DAN PERANCANGAN

3.1 Metode Penelitian

Pada bab ini dijelaskan langkah - langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari simulasi yang akan dibuat.Langkah-langkah metodologi dalam tugas akhir ini terdiri dari: studi literatur, perancangan, implementasi dan pengujian. Gambar 3.1 merupakan diagram alir dari metodologi pengembangan simulasi ini.



Gambar 3.1 Diagram alir metodologi pengembangan simulasi.

3.1.1 Studi Literatur

Studi literatur menjelaskan dasar teori dan referensi yang akan digunakan dalam perancangan dan pembuatan simulasi. Studi literatur mengandung data-data yang dikumpulkan dari buku, website, materi pembelajaran, maupun simulasi sejenis yang sudah ada yang menjadi sumber dari teori dan referensi yang akan diimplementasikan lebih lanjut pada simulasi yang dibuat.Dalam penelitian ini studi literatur yang digunakan antara lain :

a. Simulasi
b. Game Engine Unity
c. Quadcopter

3.1.2 Perancangan

Setelah semua analisis kebutuhan dilaksanakan, hal berikutnya merupakan perancangan dari aplikasi simulasi yang akan dikembangkan. Dalam perancangan ini dilakukan semua tahap awal pengembangan aplikasi.

Perancangan berisi tentang keseluruhan konsep aplikasi dan teknis dari aplikasi simulasi. Bagian perancangan dapat berisikan hal berikut.

1. *Flow Chart* Penerbangan.
2. Rancangan Quadcopter.
3. Perhitungan Manual.

3.1.3 Implementasi

Implementasi aplikasi simulasi ini akan dilakukan mengacu kepada pengguna memiliki *personal computer* masing-masing. Implementasi juga dilakukan menggunakan bahasa pemrograman C# dengan *Game Engine*Unity. Hasil dari aplikasi ini adalah sebuah *pack file* berisi program ber ekstensi windows dan *file-file* yang diperlukan oleh aplikasi. Implementasi juga akan membahas antarmuka, dan cara penggunaan aplikasi. Implementasi aplikasi dapat dibagi menjadi beberapa bagian agar mudah dipahami, bagian tersebut antara lain:

1. Pemilihan teknologi dan *platform*.
2. Implementasi Quadcopter.

Tabel berikut menunjukkan program atau aplikasi yang digunakan dalam melakukan implementasi pembuatan Aplikasi Simulasi:

Tabel 3.1 Implementasi Pembuatan Aplikasi

Jenis Implementasi	Nama Program
Desain 3D	Blender 2.68a
Desain 2D	Adobe Photoshop CS6
Game Engine	Unity 4.3.0f4
Bahasa Pemrograman	C#

3.1.4 Evaluasi

Pengujian atau evaluasi aplikasi dilakukan agar dapat menunjukkan bahwa game yang telah dikembangkan dapat bekerja dengan baik sesuai dengan kebutuhan dan perancangan yang telah ditetukan sebelumnya. Pengujian juga bertujuan untuk menemukan kesalahan-kesalahan atau kekurangan-kekurangan pada aplikasi yang diuji.

Evaluasi pengujian aplikasi terdiri atas:

1. Whitebox Testing

2. Performance Testing

Dari pengujian ini akan didapatkan hasil pengujian yang dapat digunakan untuk memperbaiki kesalahan atau kekurangan yang ditemukan dalam aplikasi yang diuji.

3.2 Perancangan

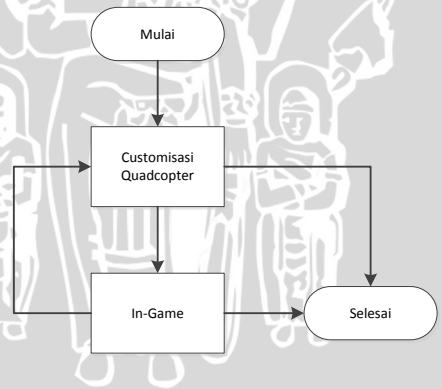
Perancangan berisi konsep dari aplikasi yang akan dikembangan seperti yang telah dijelaskan pada metodologi.

3.2.1 Deskripsi Aplikasi

Aplikasi ini dirancang untuk menyimulasikan sifat atau pergerakan *quadcopter* berdasarkan beberapa faktor yaitu massa total, motor/rotor dan baling-baling yang dapat dipilih pengguna. Pengguna simulasi merupakan satu orang yang dapat mengendalikan *quadcopter. Quadcopter* dalam simulasi dapat dikendalikan seperti menggunakan *quadcopter* sebenarnya dengan *remotecontrol*.

3.2.2 Perancangan Aplikasi

Aplikasi terbagi menjadi 3 *scene* yaitu *scene* main menu, *scene* customisasi *quadcopter* dan *scene* simulasi atau in-game.Hubunganantar*scene* digambarkan dengan diagram screen-flow pada Gambar 3.2.

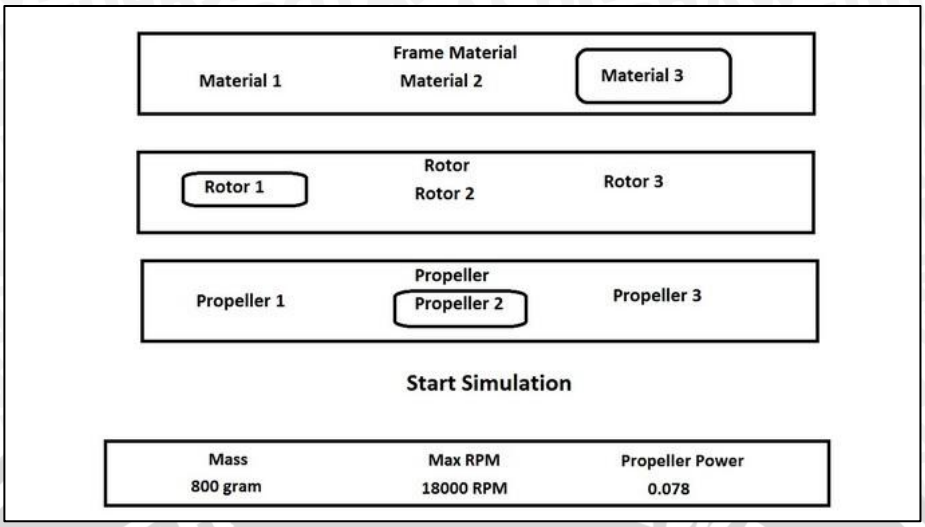


Gambar 3.2 Diagram screen-flow aplikasi

Penjelasan masing-masing *scene* adalah sebagai berikut.

a. Scene customisasi quadcopter

Scene ini adalah tempat dimana pengguna dapat memilih bahan *frame*, rotor, dan baling-baling dari yang sudah disediakan. Pada scene ini juga ditampilkan massa, RPM maksimal dan konstanta kekuatan baling-baling untuk memberikan informasi hasil pemilihan part yang digunakan pengguna. Konsep tampilan dari *scene* inidapat dilihat pada Gambar 3.3dibawah.

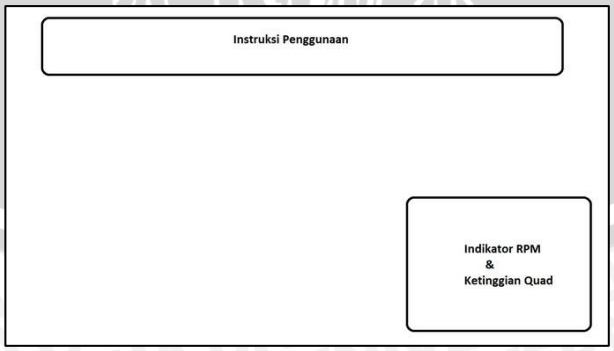


Gambar 3.3 Konsep tampilan *scene* customisasi

Pada Gambar 3.3 pemain dapat memilih material, rotor dan propeller yang diinginkan. Part yang telah dipilih pemain akan ditampilkan berbeda dengan part yang tidak terpilih.Pada tampilan dibawah *scene* terdapat status dari part yang dipilih, status tersebut adalah massa, RPM maksimal dan konstanta kekuatan propeller. Ketika selesai merubah part yang digunakan, pengguna dapat menjalankan simulasi dengan memilih start simulation.

b. Scene simulasi (in-game)

Pada Scene ini akan dijalankan simulasi pergerakan *quadcopter*. Pengguna akan mengedalikan quadcopter menggunakan sudut pandang orang ketiga dari *quadcopter* yang digunakan. Scene ini juga akan menampilkan *HUD (Heads-Up Display)* yang berisi instruksi penggunaan *quadcopter,* informasi kecepatan putaran tiap rotor dan ketinggian *quadcopter.*Konsep tampilan *HUD* scene ini dapat dilihat pada Gambar 3.4

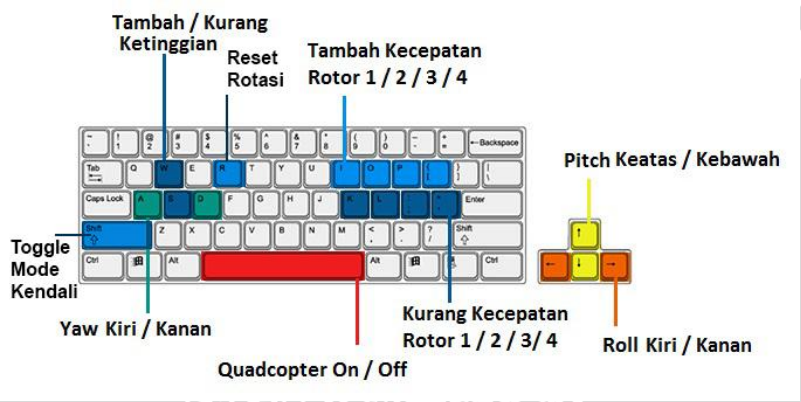


Gambar 3.4 Konsep tampilan *HUD* dalam simulasi

3.2.3 Perancangan *Quadcopter*dan Sistem Kendali

Quadcopter akan terdiri dari tiga bagian yaitu *frame,* motor dan baling-baling. Masing-masing bagian memiliki beberapa model yang dapat dipilih oleh pengguna. Selain model dari masing-masing bagian, pengguna dapat menentukan massa total *quadcopter* dan kecepatan putaran maksimal dari motor. Pemilihan model baling-baling akan menentukan besarnya kekuatan dorongan yang dihasilkan dari baling-baling tersebut.

Keyboard digunakan seagai input pengendalian*quadcopter.* Kendali disamakan seperti meggunakan *controller* aslinya yaitu *throttle, yaw, pitch* dan *roll* akan merubah kecepatan putarandua atau lebih motor. Selain itu pengguna diberikan pilihan tombol untuk menambah atau mengurangi kecepatan putaran salah satu rotor. Pengguna juga dapat mematikan atau menyalakan *quadcopter*, me*reset* rotasi *quadcopter* dan merubah mode kendali *auto-hover*. Rancangan susunan input pada *keyboard* dapat dilihat pada Gambar 3.5.



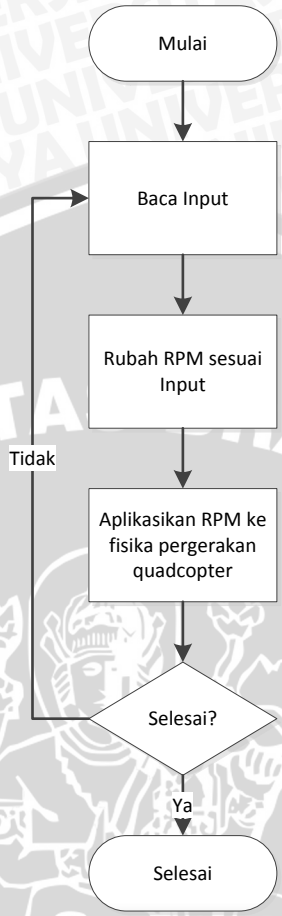
Gambar 3.5 Rancangan susunan input pada keyboard

Pergerakan *quadcopter* dibuat menggunakan fitur-fitur yang telah disediakan dalam *physic engine* Unity. *Quadcopter*diterbangkan dengan memanipulasi jumlah *force* yang dikeluarkan pada lokasi tiap motor dengan arah*force* yang selalu mengarah kebawah*frame quadcopter*.Jumlah *force* yang dikeluarkan satu motor didapat dari perhitungan menggunakan persamaan (2.1). Persamaan

(**2.2**) menggap bahwa jumlah force pada *quadcopter* didapat dari penambahkan *force*setiap motor. Pada mode kendali *auto-hover* digunakan persamaan

(**2.3**) untuk mendapatkan total *force* yang diperlukan untuk melakukan *hovering* kemudian *force* tersebut disebar secara merata ke lokasi tiap motor.

3.2.4 Perancangan Dinamika Quadcopter



Gambar 3.6 Flowchart dinamika pengendalian quadcopter.

Quadcopter digerakkan dengan cara merubah RPM dari satu atau lebih motornya. Perubahan RPM akan menyebabkan perubahan dorongan pada bagian-bagian badan quadcopter yang menyebabkan perubahan ketinggian, perputaran ataupun kemiringan quadcopter. Untuk mengendalikan RPM, pengguna disediakan beberapa tombol keyboard yang dapat dilihat pada gambar 3.6.

Ketika tombol w ditekan maka keempat motor akan menambah kecepatan putarannya.Hal ini akan membuat dorongan meningkat pada semua lokasi baling-baling dan quadcopter akan bergerak keatas. Ketika tombol s ditekan kebalikannya akan terjadi. RPM tiap motor akan berkurang secara bersamaan menyebabkan dorongan semua baling-baling berkurang dan membuat quadcopter turun.

Ketika tombol a ditekan sepasang motor yang berputar searah jarum jam (motor 1 dan 3) menambah kecepatan dan motor yang berputar berlawanan arah jarum jam (motor 2 dan 4) akan mengurangi kecepatan putarannya. Ketidakseimbangan putaran ini akan menyebabkan torsi tidak seimbang dan quadcopter akan berputar berlawanan arah jarum jam. Ketika tombol d ditekan

akan terjadi kebalikannya. Motor 1 dan 3 mengurangi kecepatan dan motor 2 dan 4 akan mempercepat putarannya. Quadcopter akan berputar searah putaran jarum jam.

Ketika tombol panah kiri ditekan maka RPM motor 2 akan bertambah dan RPM motor 4 berkurang. Hal ini menyebabkan dorongan pada sisi kanan quadcopter bertambah dan dorongan pada sisi kiri berkurang. Perubahan keseimbangan dorongan menyebabkan quadcopter akan miring dan menukik kearah kiri. Kebalikannya terjadi jika tombol panah kekanan ditekan. RPM motor 2 berkurang dan RPM motor 4 bertambah, dorongan pada sisi kiri bertambah dan dorongan pada sisi kanan berkurang maka quadcopter akan miring dan menukik kearah kanan.

Tombol panah atas ketika ditekan akan mengurangi RPM motor 1 dan menambah RPM motor 3. Hal ini menyababkan dorongan di depan quadcopter berkurang dan dorongan dibelakang quadcopter bertambah. Karena perubahan dorongan tersebut quadcopter akan menukik kebawah. Kebalikannya terjadi apabila tombol panah bawah ditekan. RPM motor 1 akan bertambah dan RPM motor 3 berkurang menyebabkan quadcopter menukik keatas.

Akan terdapat 2 fungsi yang mengatur pergerakan quadcopter secara fisika yaitu satu fungsi untuk menghitung dorongan yang dihasilkan tiap baling-baling. Dan satu fungsi untuk menghitung besarnya quadcopter melakukan manuver yaw. Fungsi untuk menghitung dorongan akan menggunakan persamaan 2.1, sedangkan untuk menghitung besarnya manuver yaw akan digunakan persamaan 2.4

BAB 4 IMPLEMENTASI

Bab ini akan membahas tentang pengimplementasian pada aplikasi Simulasi yang didasarkan pada perancangan yang telah dibahas pada bab sebelumnya.

4.1 Pemilihan Teknologi

Pengimplementasian aplikasi dibuat dalam lingkungan yang terdiri atas perangkat keras serta perangkat lunak. Perangkat keras yang digunakan dalam pengembangan menggunakan sebuah laptop. Spesifikasi laptop dapat dilihat pada Tabel 4.1

Tabel 4.1 Spesifikasi perangkat keras laptop.

Model	ASUS A43S
Processor	Intel® Core™ i5 CPU @2.40GHz
RAM	4 GB
Harddisk	1 TB HDD
Graphic Card	AMD Radeon HD 6730M

Spesifikasi dari perangkat lunak yang dipakai pada pengembangan aplikasi ini dapat dilihat pada Tabel 4.2.

Tabel 4.2 Spesifikasi perangkat pengembang aplikasi.

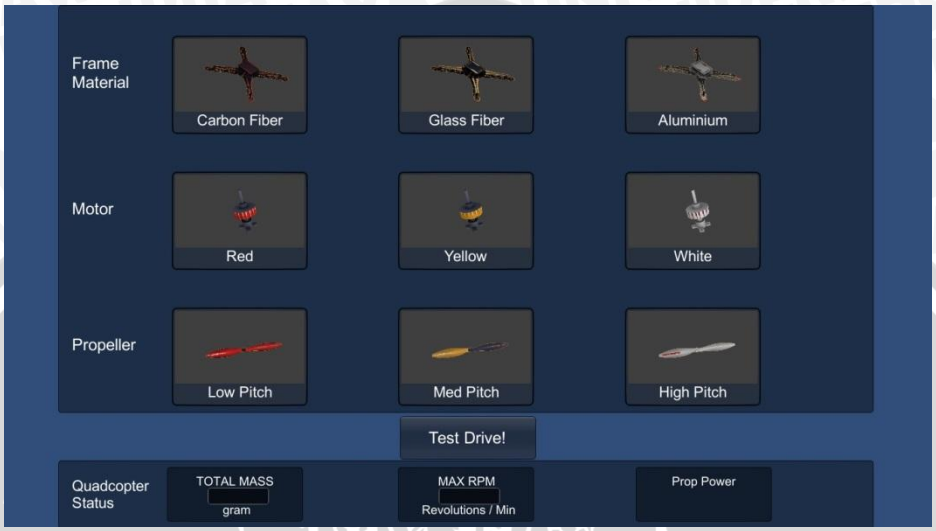
Operating System	Windows 7 Ultimate Service Pack 1 64-bit
DirectX Version	DirectX 11
Game Engine	Unity 4.3.0f4
Integrated Development Environment	Microsoft Visual Studio 2010
Programming Language	C#
2D Graphics Editor	Adobe Photoshop CS6 Windows 7 Paint
3D Graphics Editor	Blender 2.68a

Perangkat keras dan lunak yang telah disebutkan digunakan dalam pengimplementasian aplikasi sesuai kebutuhan dan batasan dari game.

4.2 Implementasi *Scene*

Scene dalam aplikasi ini terdiri dari tiga bagian yaitu *scene* main menu, *scene* customisasi *quadcopter* dan *scene in-game*.

4.2.1 *Scene* Customize



Gambar 4.1 *Screenshot* tampilan *scenecustomize.*

Gambar 4.1 merupakan implementasi untuk scene customisasi. Pada scene customize ditampilakan pilihan frame material, motor dan propeller. Selain itu pengguna dapat memasukkan total massa quadcopter dalam satuan gram dan putaran maksimum motor dalam satuan Revolution / Minute (RPM). Nilai Prop Power akan didapatkan dari pemilihan baling-baling.

4.2.2 *Scene* In-Game

Gambar 4.2 *Screenshot scene in-game*

Gambar 4.2 merupakan implementasi dari scene In-Game. Pada scene In-Game ditampilkan quadcopter ditengah layar. Kamera akan mengikuti pergerakan quadcopter. Pada HUD (Heads Up Display) layar ditampilkan tombol-tombol yang dapat digunakan. Kecepatan putaran motor dan ketinggian terbang quadcopter ditampilkan pada HUD sisi kanan-bawah layar.

4.3 Implementasi Quadcopter

Implementasi *quadcopter* dalam aplikasi akan terbagi menjadi 3 bagian yaitu: Implementasi *Game Object Quadcopter*, Implementasi Pergerakan *Quadcopter* dan *Implementasi* Kendali *Quadcopter.*

4.3.1 Game Object Quadcopter

Game object penyusun *quadcopter* terdiri dari QuadContainer, Collider13, Collider 24, Frame, Motor1, Motor2, Motor3, Motor4 serta Propeller1, Propeller2, Propeller3 dan Propeller4.*Game object* ini tersusun secara hirarki sesuai dengan Gambar 4.3. Penjelasan dari masing-masing game object adalah sebagai berikut:

Gambar 4.3 Susunan *hierarchygame object quadcopter*.

- QuadContainer : merupakan *parent* dari semua *game object quadcopter*. Dalam *game object* ini juga dimasukkan *Component* Rigidbody, *Script* CopterControlForceAt yang berfungsi sebagai penggerak *quadcopter*dan *Script* CopterParts untuk menyimpan bagian-bagian penyusun quadcopter yang sedang digunakan.
- Collider13 dan Collider24 : kedua *game object* ini masing-masing menyimpan sebuah*box collider* yang ketika digabungkan akan membentuk sebuah*frame quadcopter*.
- Frame : *Game Object* ini selain menyimpan model dan tekstur dari *frame quadcopter* yang sedang digunakan juga berfungsi sebagai acuankoordinat lokasi model komponen-komponen *quadcopter*.
- Motor1, Motor2, Motor3, Motor4 : Masing-masing motor menyimpan model dan tekstur dari motor yang digunakan serta digunakan sebagai acuan koordinat lokasi masing-masing *propeller*nya.

- Propeller1, Propeller2, Propeller3, Propeller4 : Setiap propeller menyimpan model dan tekstur baling-baling yang digunakan.

4.3.2 Implementasi Pergerakan *Quadcopter*

Dorongan *quadcopter* didapat daripersamaan(2.1). Tetapi karena tidak disimulasikannya kepadatan udara, maka variabel kepadatan udara dianggap 1 oleh karena itu dapat tidak dimasukkan kedalam perhitungan.*source code* dari persamaan (2.1) dituliskan pada fungsi CalcThrust dalam Tabel 4.3.

Tabel 4.3 Source code kalkulasi dorongan *quadcopter.*

Source code Method**CalcThrust**	

1	`Vector3 CalcThrust(float xSectA, float currentRPM, Vector3 direction)`
	`{`
	`Vector3 resultVector = direction;`
	`resultVector *= xSectA * Mathf.Pow(currentRPM * 0.1f, 2);`
	`return resultVector;`
	`}`

Metode CalcThrust memiliki input berupa variabel xSectA yaitu luas penampang baling-baling, currentRPM yaitu kecepatan putaran motor saat ini dan sebuah variabel direction yaitu arah dari dorongan. Pertama fungsi ini akan menerima arah dorongan berupa vector arah tiga dimensi, setelah itu vector arah tersebut diberi besaran sebesar perhitungan sesuai dengan persamaan (2.1). kecepatan putaran motor dikalikan dengan 0.1 karena besaran kecepatan yang digunakan masih sebagai RPM (Round per Minute) dan perlu dikonversikan menjadi Radian per sekon.

Tabel 4.4 *Source code method* CalcTorque.

Source Code Method **CalcTorque**	
1	`Vector3 CalcTorque(float rpm1, float rpm2, float rpm3, float rpm4, Vector3 direction)`
2	`{`
3	`Vector3 result;`
4	
5	`result = -direction;`

Tabel 4.4 (Lanjutan)

```
6
7               rpm1 *= 0.1f;
8               rpm2 *= 0.1f;
9               rpm3 *= 0.1f;
10              rpm4 *= 0.1f;
11
12              result *= (((rpm1 * rpm1) - (rpm2 * rpm2) + (rpm3 * rpm3) -
    (rpm4 * rpm4)) * Parts.PropA);
13
14              return result;
15          }
```

Metode CalcTorque berfungsi untuk menghitung rotasi quadcopter dalam sumbu z yaitu saat quadcopter melakukan manuver yaw.Metode ini memiliki parameter input keempat kecepatan putaran motor dan sebuah vector arah perputaran.Dalam metode ini digunakan persamaan 2.4 yang digunakan pada baris 12 tabel 4.4 diatas.Pada baris ke-5 variabel direction dibuat minus karena torsi yang terjadi diakibatkan oleh gaya gesek yang berlawanan arah dengan putaran baling-baling, lalu keempat RPM dikalikan 0.1f agar satuannya berubah dari *Revolution per Minute* (putaran per menit) menjadi *Radians per Second* (Radian per sekon).

Tabel 4.5 *Source code method* FixedUpdate.

	Source Code Method **FixedUpdate**
1	`void FixedUpdate()`
2	`    {`
3	`        int i = 0;`
4	`        foreach (Transform prop in PropTranses)`
5	`        {`
6	`            if (prop != null)`
7	`            {`
8	`   rigidbody.AddForceAtPosition(CalcThrust(Parts.PropA, RPMs[i], transform.up), transform.TransformPoint(ThrustPos[i]));`

Tabel 4.5 (Lanjutan)

```
9                }
10               i++;
11           }


12       rigidbody.AddRelativeTorque(CalcTorque(RPMs[0],
RPMs[1], RPMs[2], RPMs[3], Vector3.up));


13     }
```

Dalam metode FixedUpdate dipanggil metode rigidbody.AddForceAtPosition() untuk memberikan dorongan pada lokasi setiap baling-baling. Input pertama metode AddForceAtPosition merupakan jumlah force yang didapat dari pemanggilan metode CalcThrust dengan input besarnya penampang baling-baling pada Parts.PropA, RPM masing-masing motor dan transform.up yaitu arah dorongan yang selalu mengarah keatas *framequadcopter*. Input kedua AddForceAtPosition adalah koordinat lokasi dorongan yaitu pada lokasi baling-baling.

Dalam metode ini juga dipanggil fungsi rigidbody.AddRelativeTorque() untuk memberikan torsi dengan arah relatif parent objeknya. Input metode ini adalah besarnya torsi yang didapat dari pemanggilan metode CalcTorque() dengan parameter keempat RPM motor serta vector arah keatas.

Tabel 4.6 Source code metode Update

```
1       void Update () {
2
3           if (Input.GetKeyDown(KeyCode.R))
4           {
5              transform.rotation = new Quaternion(0, 0, 0, 0);
6              Throttle = 0;
7              Yaw = 0;
8              Pitch = 0;
9              Roll = 0;
10             for (int i = 0; i < 4; i++)
11             {
12                AddRPMs[i] = 0;
13             }
14          }
15
16          else if (Input.GetKeyDown(KeyCode.Space))
17          {
```

Tabel 4.6 (Lanjutan)

```
18              TogglePower();
19          }
20
21          else if (Input.GetKeyDown(KeyCode.LeftShift))
22          {
23              ToggleHover();
24          }
25
26          else if (Input.GetKeyDown(KeyCode.Escape))
27          {
28              Application.LoadLevel("Customize");
29          }
30
31          else if (Power)
32          {
33              Throttle += Input.GetAxis("Throttle");
34              Pitch += Input.GetAxis("Pitch");
35              Roll += Input.GetAxis("Roll");
36              Yaw += Input.GetAxis("Yaw");
37
38              Throttle = Mathf.Clamp(Throttle, -1, 1f);
39              Pitch = Mathf.Clamp(Pitch, -1, 1);
40              Roll = Mathf.Clamp(Roll, -1, 1);
41              Yaw = Mathf.Clamp(Yaw, -1, 1);
42
                AddRPMs[0] += Mathf.Clamp(Input.GetAxis("Motor1"), -1,
43          1f);
                AddRPMs[1] += Mathf.Clamp(Input.GetAxis("Motor2"), -1,
44          1f);
                AddRPMs[2] += Mathf.Clamp(Input.GetAxis("Motor3"), -1,
45          1f);
                AddRPMs[3] += Mathf.Clamp(Input.GetAxis("Motor4"), -1,
46          1f);
47
48              // Balancing
49
50              if (Hover)
51              {
52                  Balance();
53              }
54
                RPMs[0] = RPMreq + maxRPM * Mathf.Clamp((Throttle -
55          Pitch - Yaw + AddRPMs[0]), -1, 1);
                RPMs[1] = RPMreq + maxRPM * Mathf.Clamp((Throttle + Roll
56          + Yaw + AddRPMs[1]), -1, 1);
```

Tabel 4.6 (Lanjutan)

```
57          RPMs[2] = RPMreq + maxRPM * Mathf.Clamp((Throttle + Pitch - Yaw + AddRPMs[2]), -1, 1);
58          RPMs[3] = RPMreq + maxRPM * Mathf.Clamp((Throttle - Roll + Yaw + AddRPMs[3]), -1, 1);
59
60          RotateProps();
61      }
62  }
63
```

Metode update pada Tabel 4.6 berfungsi untuk memeriksa semua input dari pengguna. Pada fungsi ini pertama diperiksa apakah pengguna menekan tombol r yang berfungsi untuk mereset rotasi dari quadcopter, lalu diperiksa apakah pengguna menekan tombol spasi yang berguna untuk menyalakan / menghidupkan quadcopter. Selain itu diperksa penekanan tombol ESC yang berguna untuk kembali ke menu kustomisasi quadcopter.

Setelah semua diperiksa metode ini akan memeriksa apakah quadcopter dalam status hidup atau mati. Jika hidup quadcopter akan dapat dikendalikan melalui input berupa tombol w,a,s,d, 4 arah panah, serta i, o, p, kurung siku buka ([), k, l, titik koma (;) dan tanda petik (‘). Selain tombol tersebut diperiksa apakah quadcopter dalam mode hover. Jika dalam mode hover maka ketika tombol pengendalian quadcopter tidak ditekan RPM quadcopter akan kembali ke RPM yang diperlukan untuk melakukan hovering.

BAB 5 PENGUJIAN DAN EVALUASI

Subab ini akan membahas pengujian aplikasi menggunakan metode pengujian *whitebox* dan metode pengujian *blackbox.*

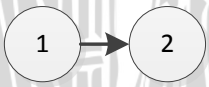
5.1.1 Pengujian *Whitebox*

Pengujian whitebox akan menguji aplikasi berdasarkan detail perancangan serta *source code* dari program serta menggunakan struktur dari desain program secara procedural untuk membagi program kedalam eberapa kasus uji. Pengujian whitebox yang digunakan akan berbasis path. Metode yang perlu diuji adalah metode CalcThrust, CalcTorque, Update.

Tabel 5.1 Source Code Method CalcThrust

Source code Method **CalcThrust**	
1	`Vector3 CalcThrust(float xSectA, float currentRPM, Vector3 direction)`

1	`{`
	`Vector3 resultVector = direction;`
	`resultVector *= xSectA * Mathf.Pow(currentRPM * 0.1f, 2);`
	`return resultVector;`
	`}`



Gambar 5.1 Flow graph dari metode CalcThrust.

Flowgraph metode CalcThrust pada gambar 5.1 menggambarkan path yang terdapat dalam Tabel 5.1. metode CalcThrust tidak memiliki cabang, oleh karena itu flowgraph hanya memiliki 2 node dan 1 edge. nilai cyclometrix complexity dapat ditentukan dari Edge (E) dan Node (N) seperti pada Persamaan dibawah.

$$V(G) = E - N + 2$$

$$V(G) = 1 - 2 + 2$$

$$V(G) = 1 \qquad (5.1)$$

Pada persamaan 5.1 V(G) merupakan nilai cyclometrix complexity, E adalah Edge dan N adalah Node dalam flowgraph. Nilai cyclometrix complexity 1 merupakan nilai dengan kategori kompleksitas rendah.

Tabel 5.2 Kasus uji untuk metode CalcThrust

No.	Parameter Kasus Uji	Hasil yang diharapkan	Hasil yang didapat	Status Valid
1.	xSectA = 0.1 CurrentRPM = 2000 Direction = (0, 1, 0)	4000	4000	Valid

Dari Tabel 5.2 diatas, didapatkan hasil yang valid dari parameter kasus uji yang diberikan. Maka dapat disimpulkan fungsi CalcThrust bekerja sesuai harapan.

Tabel 5.3 Source Code Method CalcTorque

Source Code Method **CalcTorque**	
1	Vector3 CalcTorque(float rpm1, float rpm2, float rpm3, float rpm4, Vector3 direction)
2	{
3	Vector3 result;
4	
5	result = -direction;
6	
7	rpm1 *= 0.1f;
8	rpm2 *= 0.1f;
9	rpm3 *= 0.1f;
10	rpm4 *= 0.1f;
11	
12	result *= (((rpm1 * rpm1) - (rpm2 * rpm2) + (rpm3 * rpm3) - (rpm4 * rpm4)) * Parts.PropA);

Tabel 5.3 (Lanjutan)

13	
14	return result;
15	}

Gambar 5.2 Flow graph dari metode CalcTorque

Sama seperti metode CalcTrust, metode CalcTorque tidak memiliki cabang, karena itu metode CalcTorque hanya memiliki 2 node dan 1 edge. Perhitungan cyclometrix complexity dapat dilihat pada persamaan 5.2 berikut

$$V(G) = E - N + 2$$

$$V(G) = 1 - 2 + 2$$

$$V(G) = 1$$

(5.2)

Nilai cyclometrix complexity yang didapat adalah 1 yang menandakan fungsi CalcTorque ini beresiko rendah.

Tabel 5.4 Kasus Uji metode CalcTorque

No.	Parameter Kasus Uji	Hasil yang diharapkan	Hasil yang didapat	Status Valid
1.	rpm1 = 1956.788 rpm2 = 2005.03 rpm3 = 1956.788 rpm4 = 2005.03 direction = (0, 1 0)	3822.520	3822.5	Valid

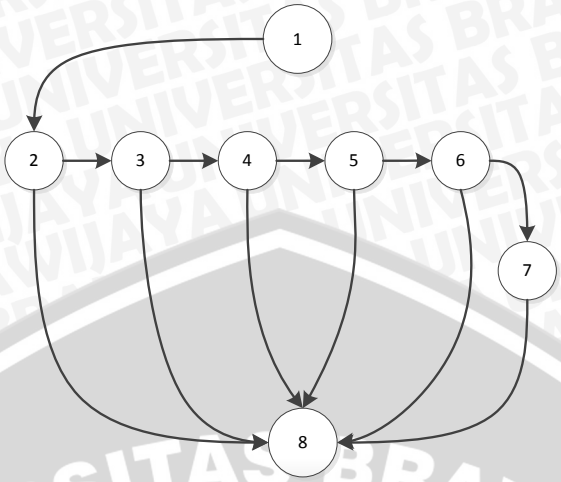
Dapat dilihat pada tabel 5.4. Semua kasus uji menghasilkan hasil valid, sehingga dapat disimpulkan metode telah berfungsi sesuai yang diharapakan.

Tabel 5.5 Source Code Metode Update

		Source Code Metode Update()
1		void Update () {
2		
3		if (Input.GetKeyDown(KeyCode.R))
4		{
5		transform.rotation = new Quaternion(0, 0, 0, 0);
6		Throttle = 0;
7		Yaw = 0;
8	1	Pitch = 0;
9		Roll = 0;
10		for (int i = 0; i < 4; i++)
11		{
12		AddRPMs[i] = 0;
13		}
14		}
15		
16		else if (Input.GetKeyDown(KeyCode.Space))
17	2	{
18		TogglePower();
19		}
20		
21		else if (Input.GetKeyDown(KeyCode.LeftShift))
22	3	{
23		ToggleHover();
24		}
25		
26		else if (Input.GetKeyDown(KeyCode.Escape))
27	4	{
28		Application.LoadLevel("Customize");
29		}
30		
31		else if (Power)
32		{
33		Throttle += Input.GetAxis("Throttle");
34		Pitch += Input.GetAxis("Pitch");
35		Roll += Input.GetAxis("Roll");
36	5	Yaw += Input.GetAxis("Yaw");
37		
38		Throttle = Mathf.Clamp(Throttle, -1, 1f);
39		Pitch = Mathf.Clamp(Pitch, -1, 1);
40		Roll = Mathf.Clamp(Roll, -1, 1);
41		Yaw = Mathf.Clamp(Yaw, -1, 1);

```
42
                AddRPMs[0] += Mathf.Clamp(Input.GetAxis("Motor1"), -1,
43          1f);
                AddRPMs[1] += Mathf.Clamp(Input.GetAxis("Motor2"), -1,
44          1f);
                AddRPMs[2] += Mathf.Clamp(Input.GetAxis("Motor3"), -1,
45          1f);
                AddRPMs[3] += Mathf.Clamp(Input.GetAxis("Motor4"), -1,
46          1f);
47
48              // Balancing
49
50              if (Hover)
51          6   {
52  5           Balance();
53  (Cont.)     }
54
                RPMs[0] = RPMreq + maxRPM * Mathf.Clamp((Throttle -
55          Pitch - Yaw + AddRPMs[0]), -1, 1);
                RPMs[1] = RPMreq + maxRPM * Mathf.Clamp((Throttle +
56          Roll + Yaw + AddRPMs[1]), -1, 1);
                RPMs[2] = RPMreq + maxRPM * Mathf.Clamp((Throttle +
57          Pitch - Yaw + AddRPMs[2]), -1, 1);
                RPMs[3] = RPMreq + maxRPM * Mathf.Clamp((Throttle -
58          Roll + Yaw + AddRPMs[3]), -1, 1);
59
60              RotateProps();
61          }
62      }
63
```

Source code pada Tabel 5.5 menunjukkan cabang *path* yang ada pada fungsi update. Dalam fungsi ini terdapat 6 *path* yang dapat dilalui aplikasi. *Path* ini dapat digambarkan menggunakan flowgraph seperti pada Gambar 5.3



Gambar 5.3 Flowgraph dari metode update

Flowgraph fungsi update pada Gambar 5.3 menggambarkan 8 *node*. Selain itu terdapat 12 *edge* yang menyambungkan setiap node yang ada. Kemudian nilai dari cyclometrix complexity dapat dicari menggunakan perhitungan (5.3).

$$V(G) = E - N + 2$$

$$V(G) = 12 - 8 + 2$$

$$V(G) = 6$$

(5.3)

Nilai cyclometix complexity fungsi update bernilai 6. Maka dapat diartikan bahwa terdapat 6 *path* independen dalam fungsi update. 6 *path* independen itu adalah sebagai berikut.

1. 1 → 2 → 8
2. 1 → 2 → 3 → 8
3. 1 → 2 → 3 → 4 → 8
4. 1 → 2 → 3 → 4 → 5 → 8
5. 1 → 2 → 3 → 4 → 5 → 6 → 8
6. 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

Setelah diketahui masing-masing *path* independen dapat dibuat kasus uji untuk metode update menggunakan basis path testing.

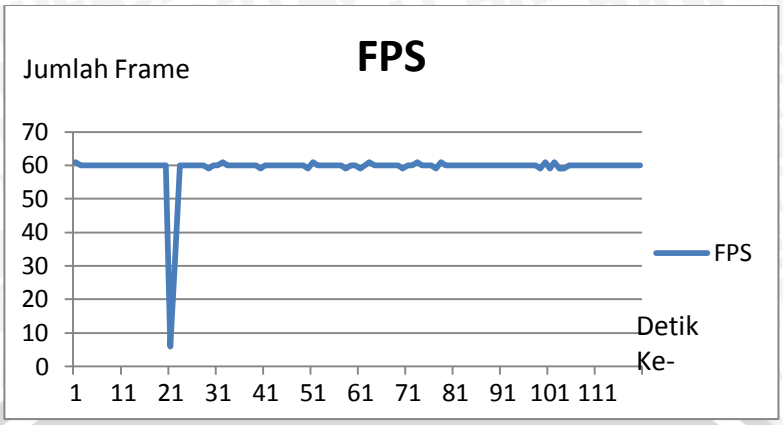
Tabel 5.6 Kasus Uji metode update.

No	Kasus Uji	Hasil yang diharapkan	Hasil yang didapat	Validitas
1	R = true	rotasi quadcopter kembali ke (0, 0, 0)	rotasi quadcopter = (0, 0, 0)	valid
2	R = false, space = true	jika power on -> power off, jika power off -> power on	jika power on -> power off, jika power off -> power on	valid
3	R= false, space = false, LeftShift = true	jika hover on -> hover off, jika hover off -> hover on	jika hover on -> hover off, jika hover off -> hover on	valid
4	R= false, space = false, LeftShift = false, Esc = true	kembali ke scene customisasi	kembali ke scene customisasi	valid
5	R= false, space = false, LeftShift = false, Esc = false, power = true	RPM quadcopter dapat dirubah dengan input w,a,s,d, 4 arah panah dan I,o,p,[,k,l,;,'.	RPM quadcopter dapat dirubah sesuai input	valid
6	R= false, space = false, LeftShift = false, Esc = false, power = true, hover = true	ketika tombol input w,a,s,d, 4 arah panah dan I,o,p,[,k,l,;,' tidak ada yang ditekan, quadcopter akan kembali ke RPM hover	RPM quadcopter kembali ke RPM hover saat tombol input tidak ditekan	valid

Pengujian kasus uji metode update menghasilkan hasil valid untuk semua kasus uji yang ada. Maka dapat ditarik kesimpulan bahwa metode update telah sesuai dengan yang diharapkan.

5.1.2 Pengujian Performa

Pengujian performa aplikasi dilakukan dengan teknik *FPS benchmarking* yaitu memonitor banyaknya *frame* yang digambarkan ke layar dalam satuan detik (*Frame Per Second)*. Hasil *FPS benchmarking* dari aplikasi ini dapat dilihat pada Gambar 5.4 berikut.



Gambar 5.4 Hasil *FPS Benchmarking*

Dapat dilihat pada Gambar 5.4 pada mulanya frame yang digambar stabil pada 60 FPS. Pada detik ke-21 frame turun mendekati 0 karena terjadi perpindahan *scene* dari menu *customize* ke *scene* in-game sehingga tidak ada frame yang digambar. Setelah masuk *scene* in-game, FPS kembali stabil pada tingkat 60 FPS sampai akhir test.

Selain dari hasil FPS Benchmarking diatas statistik hasil *FPS Benchmarking* dapat dilihat pada tabel 5.5 dibawah.

Tabel 5.7 Statistik Hasil FPS Benchmarking

Jumlah Frame	Waktu (ms)	Min	Max	Rata-Rata
7117	120000	0	62	59.308

Statistik benchmarking menunjukkan aplikasi rata-rata berjalan pada kecepatan 59.308 fps. Karena terdapat 1 detik dimana tidak ada frame yang digambar saat perubahan scene terjadi.

Hasil dan statistik hasil *FPS benchmarking*menunjukkan bahwa aplikasi dapat digunakan secara lancar tanpa tersendat – sendat pada lingkungan test.

5.1.3 Evaluasi Hasil

Hasil percobaan whitebox pada Tabel 5.2 dan Tabel 5.4 menghasilkan hasil yang valid untuk metode-metode yang mengatur dinamika pergerakan quadcopter. Hal ini membuktikan bahwa dinamika pergerakan sudah sesuai dengan yang diharapkan. Hasil pengujian blackbox pada **Error! Reference source ot found.** menghasilkan hasil valid untuk setiap input pengguna yang disediakan oleh simulasi.

Percobaan performa yang digambarkan pada graf Gambar 5.4 membuktikan bahwa performa aplikasi tergolong baik dengan permainan rata-rata berjalan pada kecepatan 59 fps.

BAB 6 PENUTUP

6.1 Kesimpulan

1. Implementasi dinamika penerbangan dapat dilakukan dengan perhitungan fisika klasik newton menggunakan persamaan (2.1) untuk menghitung dorongan tiap baling-baling dan persamaan (2.4) untuk menghitung besarnya perputaran gerakan yaw. Hasil dari kedua persamaan ini dapat dimasukkan pada perhitungan physic engine dalam game engine unity.
2. Pengujian fungsionalitas implementasi penerbangan bernilai valid untuk percobaan whitebox sesuai hasil pada Tabel 5.2, Tabel 5.4 serta Tabel 5.6 Kasus Uji metode update.
3. Aplikasi implementasi dapat berjalan lancar sesuai hasil pengujian performa dengan kecepatan rata-rata penggunaan 59 FPS.

6.2 Saran

1. Pengendalian Quadcopter dapat dibuat lebih nyata dengan penggunaan input berupa joystick.
2. Pergerakan quadcopter dapat dibuat lebih akurat menyerupai keadaan sebenarnya jika ditambahkan faktor-faktor eksternal seperti efek dorongan dari angin dan tekanan udara disekitar quadcopter.
3. Simulasi perancangan quadcopter dapat lebih sempurna dengan penambahan komponen pembentuk quadcopter seperti penambahan pilihan penggunaan baterai, *Electronic speed controller* dan hardware tambahan seperti kamera.
4. Perancangan quadcopter dapat dibuat lebih fleksibel dengan pilihan pengguna untuk memindahkan lokasi motor pada frame dan pilihan quadcopter model X ataupun model +.
5. Kendali quadcopter dapat dibuat lebih realistik jika ditambahkan noise pada pengaturan kestabilan quadcopter.
6. Kendali PD ataupun PID dapat digunakan untuk menjaga kestabilan quadcopter jika terdapat noise dalam implementasi.

DAFTAR PUSTAKA

Hartman D., Landis K., Mehrer M., Moreno S., Kim J. 2014. *Quadcopter Dynamic Modelling and Simulation V1.00 Documentation*. https://github.com/dch33/Quad-Sim [6 Agustus 2015]

Gvoth. J, Blaho M, Mudráková T. 2014. *Virtual Model of Unmanned Aerial Vehicle.* Slovak University of Technology. http://www2.humusoft.cz/www/papers/tcb2014/026_gvoth.pdf [20 agustus 2015]

J. Banks, J. Carson, B. Nelson, D. Nicol. 2001.*Discrete-Event System Simulation*. Prentice Hall. p. 3. ISBN 0-13-088702-1.

Sokolowsi, J.A., Banks, C.M. 2009. *Principles of Modelling and Simulation*. Hoboken, NJ: Wiley. P.6.

Gibiansky Andrews. 2012. *Quadcopter Dynamics and Simulation*. http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/ [6 Agustus 2015]

Alex. 2012. *The Quadcopter : control the orientation.* http://theboredengineers.com/2012/05/the-quadcopter-basics/ [6 Agustus 2015]

Strogatz, Steven. 2007. *The End of Insight*. New York: *What is your dangerous idea?.* Harper Perennial. ISBN 9780061214950

Hoffmann, G.M., Rajnarayan, D.G., Waslander, S.L., Dostal, D., Jang, J.S., Tomlin, C.J. 2004. The Stanford Testbed of Autonomous Rotorcraft for Multi Agent Control (STARMAC). Salt Lake City: Proceedings of the 23rd Digital Avionics System Conference.

Stafford, Jesse. 2014. *How a Quadcopter Works | Clay Allen*. http://ffden-2.phys.uaf.edu/webproj/212_spring_2014/Clay_Allen/clay_allen/works.html [6 Agustus 2015]

Liang, Oscar. 2013. *Build a Quadcopter from Scratch – Hardware Overview*. http://blog.oscarliang.net/build-a-quadcopter-beginners-tutorial-1/ [6 Agustus 2015].

Khan, Mohd. 2014. Quadcopter Flight Dynamics : International Journal of Scientific & Technology Research Volume 3, Issue 8.

Chu, Philip. 2010. Game Development with Unity.