RANCANG BANGUN *GAME* SURVIVAL FROM ISLAND MENGGUNAKAN METODE *FINITE STATE MACHINE* (FSM)

SKRIPSI

Disusun Oleh :

Lalu Sanostra Aulia Fahmi

10.18.137

PROGRAM STUDI TEKNIK INFORMATIKA S-1

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2014

LEMBAR PERSETUJUAN DAN PENGESAHAN

RANCANG BANGUN *GAME SURVIVAL FROM ISLAND* MENGGUNAKAN METODE *FINITE STATE MACHINE* (FSM)

SKRIPSI

Disusun dan Diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai Gelar Sarjana Teknik Informatika Strata Satu (S-1)

Disusun Oleh :
Lalu Sanostra Aulia Fahmi
10.18.137

Diperiksa dan Disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Joseph Dedy Irawan,ST, MT	**Suryo Adi Wibowo, ST., MT**
NIP. 197404162005031002	**NIP.P. 1031000438**

Ketua Program Studi Teknik Informatika S-1

Joseph Dedy Irawan,ST, MT
NIP. 197404162005031002

PROGRAM STUDI TEKNIK INFORMATIKA S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2014

LEMBAR KEASLIAN

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini:

Nama	:	Lalu Sanostra Aulia Fahmi
NIM	:	10.18.137
Program Studi	:	Teknik Informatika S-1
Fakultas	:	Teknologi Industri

Menyatakan dengan sesungguhnya bahwa Skripsi saya yang berjudul:

"RANCANG BANGUN *GAME* SURVIVAL FROM ISLAND MENGGUNAKAN METODE *FINITE STATE MACHINE* (FSM)"

Adalah Skripsi saya sendiri bukan duplikat serta mengutip atau menyadur seluruhnya karya orang lain kecuali dari sumber aslinya.

Malang, 14 Agustus 2014

Yang membuat pernyataan

Lalu Sanostra Aulia Fahmi

RANCANG BANGUN *GAME SURVIVAL FROM ISLAND* MENGGUNAKAN METODE *FINITE STATE MACHINE* (FSM)

Lalu Sanostra Aulia Fahmi

Program Studi Teknik Informatika S-1
Fakultas Teknologi Industri
Institut Teknologi Nasional Malang
Jl. Raya Karanglo Km. 2 Tasikmadu-Malang
Email: lalusanostra@gmail.com

Dosen Pembimbing: 1. Joseph Dedy Irawan, ST, MT
2. Suryo Adi Wibowo, ST, MT

Abstraksi

Adventure Game adalah game berjenis petualangan dengan alur cerita yang berkesinambungan disertai dengan perpindahan atau pergantian tempat dari satu wilayah ke wilayah lainnya. Kecerdasan buatan didefinisikan sebagai kecerdasan yang ditujukan untuk suatu entitas buatan. Sebuah kecerdasan yang diciptakan dan dimasukkan ke dalam mesin atau objek computer agar dapat melakukan tindakan alami layaknya manusia. Pada penelitian ini akan dibangun sebuah game sederhana yang berjenis adventure, yaitu game Surival from Island.

Algoritma Finite State Machine (FSM) diterapkan kepada musuh, dengan memberikan sensor sebagai indicator respon terhadap gerakan pemain. Game adventure ini dibangun menggunakan aplikasi Unity dan bahasa pemrograman JavaScript dan CSharp, sehingga proses perancangan karakter, animasi, dan algoritma FSM dapat dicapai. Dalam game ini terdapat dua jenis musuh, yaitu hewan buas dan yang terakhir boss berbentuk monster. Perilaku yang dilakukan oleh musuh meliput melihat pemain, mengejar pemain,dan menyerang pemain.

Dari hasil pengujian pada pengguna bahwa dari tingkat ketertarikan disimpulkan bahwa 45% orang mengatakan tampilan game ini menarik. Sedangkan dari tingkat kesulitan, 40% mengatakan game ini sangat sulit. dan dari ketertarikan alur cerita dari game 50% mengatakan alur cerita cukup baik.

Kata Kunci: *Adventure Game, Finite State Machine, JavaScript, CSharp dan Unity.*

KATA PENGANTAR

Puji syukur kehadirat Allah yang maha kuasa, karena telah memberikan rahmat dan hidayah-Nya sehingga penyusun dapat menyelesaikan skripsi dengan judul RANCANG BANGUN *GAME SURVIVAL FROM ISLAND* MENGGUNAKAN METODE *FINITE STATE MACHINE* (FSM) sesuai dengan waktu yang ditentukan.

Penyusunan skripsi ini merupakan salah satu persyaratan untuk menyelesaikan program pendidikan Strata Satu (S-1) Teknik Informatika, Fakultas Teknologi Industri di Institut Teknologi Nasional Malang.

Pada penyusunan skripsi ini penyusun mengucapkan banyak terima kasih sebesar-besarnya kepada:

1. Bapak Moh. Saleh dan Ibu Fawziah, yang merupakan kedua orang tua dan pendukung utama dari segi moril maupun materil.
2. Ir.Soeparno Djiwo, MT, selaku Rektor Institut Teknologi Nasional Malang.
3. Ir. Anang Subardi, MT, selaku Dekan Fakultas Teknologi Industri, Institut Teknologi Nasional Malang.
4. Joseph Dedy Irawan, ST, MT, selaku Ketua Program Studi Teknik Informatika, Institut Teknologi Nasional Malang.
5. Sonny Prasetio, ST, MT, selaku Sekertaris Program Studi Teknik Informatika, Institut Teknologi Nasional Malang.
6. Joseph Dedy Irawan, ST, MT, selaku Dosen Pembimbing I, yang selalu memberikan masukkan.
7. Suryo Adi Wibowo, ST, MT, selaku Dosen Pembimbing II dan sekaligus Kepala Laboratorium Robotika
8. Semua dosen Program Studi Teknik Informatika yang telah membantu dalam penulisan dan masukan.
9. Semua teman seperjuangan yang telah membantu dalam terselesaikannya skripsi ini.

Penyusun menyadari bahwa skripsi masih jauh dari sempurna, oleh karena itu penyusun mengharapkan kritik dan saran dari pembaca, Semoga skripsi ini bisa bermanfaat bagi pembaca.

Malang, Agustus 2014

Penyusun

Lembar Persembahan

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Terima Kasih kepada *Allah SWT.* Yang telah melancarkan Tugas Akhir yang saya buat ini, sehingga saya dapat menyelesaikan Skripsi tepat waktu.

Terima Kasih kepada kedua orang tua saya, Bapak *H.L.Moh. Saleh* dan Ibu *Baiq Fawziah* yang telah membimbing dan membesarkan saya hingga menyelesaikan kuliah ini dan tidak lupa juga telah memberi semangat dan dukungan selama ini.

Terima kasih kepada Kakak-Kakak ku, *kak iyen, kak don, dan Kak Suk* yang telah menjadi saudara yang selalu melindungi dan mensupport saya sebagai adik.

Dan tidak lupa juga Terima kasih kepada Teman-Teman seperjuangan Teknik Informatika S-1 ITN malang; *Faiz, Snapp, Yosi, Zico, Abraar, Azam, Yessy, Citra, Putri, Kristin, Sudar, (alm) davi* dan seluruh teman jurusan informatika. Dan tidak lupa juga kepada semua Aslab Robotika yang telah memberikan kenangan selama saya masih menjabat sebagai aslab disana ☺.

Terima kasih kepada *Yoga, Kokoh* atas bantuan, maupun hiburan yang telah kalian berikan selama ini dan telah menjadi teman sekontrakan dari semester 3 sampai kita lulus ☺. Terima kasih juga kepada tante puji (ibu dari Kristin) atas Bantuan Do'a, nasihat, semangat, serta kenangan yang telah diberikan.

Ku Persembahkan karya ini kepada kalian, semoga *Allah SWT* membalas kebaikan dan dipermudah dalam segala hal. Amin.

DAFTAR ISI

Halaman

HALAMAN JUDUL ..... **i**
LEMBAR PERSETUJUAN ..... **ii**
LEMBAR KEASLIAN ..... **iii**
ABSTRAK ..... **iv**
KATA PENGANTAR ..... **v**
DAFTAR ISI ..... **vii**
DAFTAR TABEL ..... **viii**
DAFTAR GAMBAR ..... **ix**

BAB I PENDAHULUAN
1.1. Latar Belakang 1
1.2. Rumusan Masalah 2
1.3. Tujuan Penelitian 2
1.4. Batasan Masalah 2
1.5. Metodologi 3
1.6. Sistematika Penulisan Laporan 3

BAB II DASAR TEORI
2.1. *Game* 5
2.2. Sejarah *Game* 5
2.3. Syarat Perancangan *Game* 10
2.4. Jenis *Game* 11
2.5. *Finite State Machine* 13
2.6. Diagram Alir 14
2.7. *Unity* 15
2.8. *Blender* 16

BAB III ANALISA DAN PERANCANGAN
3.1. Analisa Sistem 17
3.1.1. Analisis Masalah 17
3.1.2. Pengenalan Game Survival From Island 18
3.1.3. *Storyline* 18
3.1.4. *Storyboard* 19
3.1.5. *GamePlay* 21
3.1.6. Analisa Kebutuhan Non Fungsional 22
3.1.7. Analisa Kebutuhan Fungsional 23
3.2. Perancangan Sistem 24
3.2.1. Pengenalan Karakter 24
3.2.2. Pengenalan Level *Environtment* 25
3.2.3. Perancangan Struktur Menu 26
3.2.4. Alur *Game* 28
3.2.5. Alur FSM 29
3.2.6. Perancangan Aksi Karakter 31
3.2.7. Perancangan Antar Muka 32

3.2.8. *Cheat Code* 39

BAB IV IMPLEMENTASI DAN PENGUJIAN
4.1. Implementasi 40
4.1.1 Implementasi Perangkat Keras 40
4.1.2 Implementasi Perangkat Lunak 40
4.1.3 Implementasi Aplikasi 41
4.1.4 Implementasi Gerakan Animasi Objek 3D 41
4.1.5 Implementasi *Stage / Area* di Unity 52
4.1.6 Implementasi *Main Menu* 64
4.1.7 Implementasi *Menu Play Game* 70
4.1.8 Implementasi Karakter Lawan 78
4.2. Pengujian 83
4.2.1. Pengujian Fungsional 83
4.2.2. Pengujian Non Fungsional 87

BAB V PENUTUP
5.1 Kesimpulan 90
5.2 Saran 90

DAFTAR PUSTAKA **92**
LAMPIRAN **93**

DAFTAR TABEL

Halaman

Tabel 2.1. Tabel Simbol Diagram Alir 15
Tabel 3.1. Tabel *StoryBoard Game* 20
Tabel 3.2. Tabel Karakter *Game* 24
Tabel 3.3. Tabel Pengenalan Level Enviroment 26
Tabel 3.4. Tabel *Cheat Code* 39
Tabel 4.1. Tabel Implementasi Perangkat Keras 40
Tabel 4.2. Tabel *Shortcut button editing bone* 45
Tabel 4.3. Tabel *Key Control Player* 72
Tabel 4.4. Tabel Pengujian Fungsional 84
Tabel 4.5. Tabel Pengujian *Finite State Machine* 84
Tabel 4.6. Tabel Pengujian Sistem *Health* Karakter 86
Tabel 4.7. Tabel Pengujian *Cheat* 86
Tabel 4.8. Tabel pengujian *Control Player* 87
Tabel 4.9. Tabel Pengujian Jenis *OS* 87
Tabel 4.10. Tabel Pengujian *Performance Hardware* 88
Tabel 4.11. Tabel Pengujian Ketertarikan Tampilan *Game* 88
Tabel 4.12. Tabel Pengujian Tingkat Kesulitan *Level* 88
Tabel 4.13. Tabel Ketertarikan Alur Cerita *Game* 89

DAFTAR GAMBAR

Halaman

Gambar 2.1. Contoh Alur diagram *Finite State Machine* Sederhana 13
Gambar 2.2. Tampilan *WorkSpace Unity* 16
Gambar 2.3. Tampilan *WorkSpace Blender.* 16
Gambar 3.1. Struktur Menu 27
Gambar 3.2. *Flowchart* Alur Game 28
Gambar 3.3 Perancangan Aksi Karakter Anjing, Harimau, dan Serigala 29
Gambar 3.4 *Flowchart* Aksi Karakter Musuh Anjing dan Harimau 30
Gambar 3.5 Perancangan Aksi Karakter Boss 30
Gambar 3.6 *Flowchart* Aksi Karakter Boss 31
Gambar 3.7 Perancangan Aksi Karakter pada *Player* 32
Gambar 3.8 Perancangan Antar Muka Menu Utama 33
Gambar 3.9 Perancangan Antar Muka Menu *How to Play* 33
Gambar 3.10 Perancangan Antar Muka *Options* 34
Gambar 3.11 Perancangan Antar Muka *About* 34
Gambar 3.12 Perancangan Antar Muka Level 1 35
Gambar 3.13 Perancangan Antar Muka Level 2 36
Gambar 3.14 Perancangan Antar Muka Level 3 37
Gambar 3.15 Perancangan Antar Muka Level 4 38
Gambar 3.16 Rancangan Antar Muka *Respawn* 38
Gambar 4.1 *Shortcut Blender* 41
Gambar 4.2 *Start Menu Blender.* 42
Gambar 4.3 Halaman Depan *Blender* 42
Gambar 4.4 Meng-*Import* File Objek 3D (i) 43
Gambar 4.5 Meng-*Import File* Objek 3D (ii) 43
Gambar 4.6 Objek 3D pada Workspace Blender 44
Gambar 4.7 *Menu Add Single bone* 44
Gambar 4.8 Tampilan Objek *bone* 45
Gambar 4.9 *Toolbar Edit mode* 46
Gambar 4.10 *Toolbar Armature* 46
Gambar 4.11 *Bone* dan Objek 3D Terlihat 47
Gambar 4.12 Desain *Bone* Objek 3D 47
Gambar 4.13 Setting Workspace "Pose Mode" 48
Gambar 4.14 *Toolbar Animation* 48
Gambar 4.15 *Insert Keyframe* 49
Gambar 4.16 Perubahan Gerak antar *frame* 49
Gambar 4.17 *Setting Preview Animation* 50
Gambar 4.18 *Setting Dope Sheet* 50
Gambar 4.19 Tampilan *Dope Sheet* 51
Gambar 4.20 *Setting Action Editor* 51
Gambar 4.21 Menyimpan *file* Format *.fbx* 52
Gambar 4.22 *Shortcut Unity* 52
Gambar 4.23 *Menu Start Unity* 53
Gambar 4.24 *Window Project Unity* 53

Gambar 4.25 *WorkSpace Unity* 54
Gambar 4.26 *WorkSpace scene*..... 54
Gambar 4.27 *Setting Terrain.* 55
Gambar 4.28 Contoh *Texture* Rumput Lapangan 56
Gambar 4.29 *Setting Inspector Paint Texture* 56
Gambar 4.30 *Setting Add Terrain Texture* 57
Gambar 4.31 Tampilan *Terrain* dengan *Texture* 57
Gambar 4.32 *Setting Rise/Lower Terrain*..... 58
Gambar 4.33 *Contoh Rise/Lower Terrain*..... 58
Gambar 4.34 *Import Package Terrain*..... 59
Gambar 4.35 *window Import Package*..... 59
Gambar 4.36 *Setting Paint Details* 60
Gambar 4.37 *Add Grass Texture Window*..... 60
Gambar 4.38 Contoh menambahkan rumput pada *stage* 61
Gambar 4.39 *Setting Edit Trees*..... 61
Gambar 4.40 *Window Add Tree*..... 62
Gambar 4.41 Hasil *Add Tree* 62
Gambar 4.42 *Desain Terrain*..... 63
Gambar 4.43 *Directional Light*..... 63
Gambar 4.44 objek *plane* 64
Gambar 4.45 Membuat *Material* 65
Gambar 4.46 *Settingan Material*..... 65
Gambar 4.47 *Settingan Material*..... 66
Gambar 4.48 *Text 3D Hello World*..... 66
Gambar 4.49 *Setting Text 3D*..... 67
Gambar 4.50 *Box Collider* pada teks 67
Gambar 4.51 *File JavaScript*..... 68
Gambar 4.52 *Script Click Button*..... 68
Gambar 4.53 *Script Call GUI*..... 69
Gambar 4.54 *Script* Memanggil Stage Baru..... 69
Gambar 4.55 *Drag Script ke Button Text* 70
Gambar 4.56 *Import Character Controller*..... 71
Gambar 4.57 *Character Controller*..... 71
Gambar 4.58 Tombol *Play* 72
Gambar 4.59 Setting Tangan Player 73
Gambar 4.60 Menyatukan tangan dengan *Player*..... 74
Gambar 4.61 Setting Animasi Tangan 74
Gambar 4.62 *Tab Animator & file Animator*..... 75
Gambar 4.63 *New State Condition*..... 75
Gambar 4.64 *Setting State* 76
Gambar 4.65 *State Idle*..... 76
Gambar 4.66 *Parameter State* 76
Gambar 4.67 *Transision State* 77
Gambar 4.68 *Setting Transision* 77
Gambar 4.69 *Animator controller dengan Transision*..... 78
Gambar 4.70 *Import Karakter lawan*..... 79
Gambar 4.71 *Animator controller lawan* 79
Gambar 4.72 *Script Health lawan*..... 80
Gambar 4.73 *Script FSM*..... 81
Gambar 4.74 *Script function LookAt*..... 82

Gambar 4.75 *Script function Attack* 82
Gambar 4.76 *Script function Chase* 83

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan *Game* di dunia semakin pesat, tidak terkecuali di Indonesia. *Game* saat ini sudah menjadi alternatif hiburan bagi semua kalangan. Industri dan bisnis pengembangan *Game* juga sudah menjadi suatu hal yang menjanjikan, terbukti dengan banyaknya perusahaan pengembang *Game* di Amerika, Eropa dan Asia. Negara Indonesia masih terhitung sebagai konsumen *Game*, dilihat dari tingkat konsumsi *game* yang sangat tinggi. Banyak perusahaan-perusahaan yang membawa *game-game* bagus dari luar negeri untuk dimainkan di Indonesia. Di balik semua itu, tersirat keinginan dari putra putri Indonesia untuk membuat *Game* mereka sendiri, tetapi masih saja terbentur dengan masalah ilmu, biaya dan tingkat kesulitan pembuatan *game* yang memang cukup tinggi, padahal kreativitas, inovasi dan imajinasi mereka tidak kalah dengan pengembang *game* luar negeri. Untuk itu penulis berusaha membuat *Game Survival from Island* ini dari *Unity* dan *Blender* menggunakan metode *Finite State Machines* (FSM).

Dalam *Game* jenis petualangan (*adventure Game*) pemain dituntut kemampuan berfikirnya untuk menganalisa situasi secara visual, memecahkan rangkaian peristiwa, dan menggunakan senjata sebagai alat pertarungan. Dengan ini secara tidak langsung *game* jenis petualangan bermanfaat karena sebenarnya membantu untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. Oleh karena itu penulis membuat *game* jenis petualangan.

Jenis metode *Finite State Machine* (FSM) sangat sesuai digunakan dalam pembuatan jenis petualangan karena FSM sendiri merupakan metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal yaitu *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. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relatif komplek.

1.2. Rumusan Masalah

Adapun rumusan masalah yang akan dibahas dalam skripsi ini adalah bagaimana mengimplementasikan *Finite State Machine* (FSM) sebagai metode untuk menentukan reaksi lawan dalam *Game Survival from Island.*

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah mengimplementasikan *Finite State Machine* (FSM) sebagai perilaku lawan terhadap pemain. Dengan memberikan sensor pada musuh, kemudian dari sensor tersebut mampu memberi reaksi pemain.

1.4. Batasan Masalah

Batasan masalah dalam membangun *game Survival from Island* Menggunakan *Metode Finite State Machine (FSM)* antara lain:

1. *Platform* yang digunakan dalam membangun *Game Survival from Island* adalah *Unity* dengan bahasa pemrograman *JavaScript* dan *CSharp (C#.)*
2. Metode yang digunakan adalah *Metode Finite State Machine* (FSM) untuk menentukan perlaku lawan kepada pemain.
3. *Game* yang dibangun adalah jenis *Adventure Game.*
4. Adanya lawan bertarung pemain yaitu binatang Anjing, Harimau, Serigala dan musuh *boss* yaitu *Monster* misterius yang harus dikalahkan oleh pemain.

1.5. Metodologi

Tahapan-tahapan pada metode penelitian yang digunakan dalam pelaksanaan penelitian ini, meliputi :

1. Studi *literature.*

   Teknik pengumpulan data dengan mencari bahan-bahan referensi dari berbagai sumber sebagai landasan teori yang berhubungan dengan permasalahan yang akan dijadikan objek penelitian.

2. Analisa kebutuhan.

   Dengan informasi yang telah diperoleh akan dianalisa agar diperoleh suatu kerangka sebagai sumber acuan perancangan *game*.

3. Perancangan.

   Setelah pengumpulan data selesai, berlanjut pada tahap perancangan *game*.

4. Implementasi

   Data-data yang telah terkumpul, kemudian diimplementasikan ke dalam *game* bersama dengan *Finite State Machine.*

5. Pengujian

   Jika aplikasi telah selesai selanjutnya ke tahap pengujian, hal ini dilakukan untuk mengetahui apakah *game* yang dirancang sudah benar atau mendapat kesalahan.

1.6. Sistematika Penulisan

Untuk mempermudah dan memahami pembahasan pada penulisan skripsi ini, maka sistematika penulisan yang diperoleh sebagai berikut:

BAB I : **Pendahuluan**

Berisi latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, metode penelitian dan sistematika penulisan.

BAB II : **Dasar Teori**

Berisi tinjauan pustaka mengenai permasalahan yang berhubungan dengan penelitian ini.

BAB III : **Analisa dan Perancangan**

Berisi mengenai perancangan antar muka aplikasi *game Survival from Island.*

BAB IV : **Implementasi dan Pengujian**

Berisi implementasi terhadap proses pembuatan komponen-komponen dari *game Survival from Island*, serta melakukan pengujian terhadap aplikasi *game Survival from Island.*

BAB V : **Kesimpulan dan Saran**

Berisi kesimpulan dari hasil penelitian yang dilakukan dan saran yang dapat digunakan sebagai bahan pertimbangan untuk pengembangan penelitian selanjutnya.

BAB II
LANDASAN TEORI

2.1 *Game*

Game merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami penggunaa animasi haruslah memahami pembuatan *Game.* Atau jika ingin membuat *Game,* maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.

Dalam bahasa Indonesia "*Game*" dapat diartikan sebagai permainan. Permainan adalah suatu kegiatan yang di dalamnya terdapat peraturan-peraturan dan sebuah sistem yang mana pemain terlibat dalam suatu konflik buatan.Agar tidak terjadi kecurangan di dalam sebuah permainan, makan peraturan dibuat membatasi perilaku pemain dan menentukan jalannya permainan. Sehingga setiap pemain memiliki peluang yang sama untuk memenangkan permainan.

Game merupakan permainan yang merujuk pada kelincahan intelektual atau *intellectual capability.* Kelincahan intelektual pada tingkatan tertentu merupakan ukuran sejauh mana g*ame* tersebut menarik untuk dimainkan. Tidak hanya pada kelincahan intelektualnya saja, *game* juga mampu melatih kemampuan seseorang berfikir dan bertindak dalam memecahkan suatu masalah dengan cepat dan tepat. Karena di dalam sebuah *Game* terdapat berbagai konflik atau masalah yang menuntut sang pemain untuk dapat menyelesaikannya dengan hasil maksimal.

2.2 Sejarah *Game*

Dalam perjalanannya, *Game* juga memiliki sejarah yang cukup panjang. Berikut ini adalah sejarah *Game* mulai dari generasi pertama hingga masa kini:

1. *Game* Generasi Pertama

Pada tahun 1972, saat itu banyak orang yang masih belum mengenal *concole Game* dan *computer Game*. Namun sebuah perusahaan bernama *Magnavox* meluncurkan sebuah *video Game* pertama yaitu *Odyssey*. *Magnavox Odyssey,*

console *Game* pertama di dunia yang mengoperasikan *Pong*. Tidak lama setelah itu sebuah *Game arcade* legendaris *Atari* berjudul "*Pong*" muncul. *Pong* merupakan sebuah *Game* sederhana yang mengambil konsep permainan tenis, dengan 1 bola dan 2 papan di kiri dan di kanan.

Pada tahun 1975 akhirnya *Magnavox* menyerah dan menghentikan produksi *Odissey*. Sebagai gantinya mereka mengikuti jejak *Atari* dengan memproduksi mesin dingdong bernama *Odyssey* 100, yang khusus menyajikan *Game Pong*.

2. *Game* Generasi Kedua

Pada tahun 1976, *Fairchild* mencoba menghidupkan kembalidunia video *Game* dengan menciptakan VES (*Video Entertainment System*). VES adalah mesin pertma yang disebut *console. Concole* ini menggunakan kaset *magnetic* yang disebut *cartridge*. Konsep ini kemudian diikuti oleh beberapa produsen lain termasuk *Atari*, *Magnavox*, dan RCA, ketiga persahaan tersebut juga merilis *console* serupa.

Pada tahun 1977, dunia *Game console* menjadi tidak popular, *Game* yang tidak berhasil menarik minat. *Fairchild* dan RCA mengalami kebangkrutan. Praktis, hanya ada *Atari* dan *Magnavox* yang masih bertahan di dunia *video Game*. Kemudian pada tahun 1978, *Magnavox* meluncurkan *Odyssey* 2. Seperti halnya *Odyssey* pertama, *console* ini pun gagal di pasaran. Tak lama berselang *Atari* memunculkan *console* legendaris, *Atari* 2600, yang terkenal dengan *Game Space Invaders*. Pada tahun 1980, berbagai produsen *console* muncul dan mereka mengambil *Atari* 2600 menjadi konsep dan perkembangan dunia *Game* pun semakin pesat.

Pada tahun 1983, dunia *video Game* kembali ambruk. *Game* yang kurang kreatif membuat *conclose* kembali mendapat sambutan dingin, apalagi *personal computer* saat itu semakin canggih. Masyarakat lebih memilih membeli PC, karena selain untuk bermain PC juga produktif untuk bekerja.

3. *Game* Generasi Ketiga

Pada tahun 1983, perusahaan bernama *Famicom* Jepang menciptakan gebrakan baru, sebuah *console* bernama *Nitendo Entertaintment System* (NES) dirilis

di akhir 1983. *Console* ini menampilkan gambar dan animasi resolusi tinggi untuk pertama kalinya. Setelah mendapat sambutan hangat di Jepang, *Famicom* memperluas pemasarannya di America, yang dikenal dengan NES. *Nitendo* memiliki *chip* pengaman pada *cartridge Game* mereka, dengan demikian seluruh *Game* yang diliris haruslah seijin *Nitendo*. Kemudian, muncul sebuah *Game* yang sangat legendaris, yaitu *Super Mario Brothes* (*Mario Bross*), yang mana *Game* ini masih tetap eksis hingga sekarang. Akhirnya *Nitendo* dan *Famicom* berhasi menguasai pasar *video Game* di era generasi ketiga.

4. *Game* Generasi Keempat

Pada tahun 1988, NES mendapat sambutan hangat di seluruh dunia. Sampai akhirnya sebuah perusahaan bernama *Sega* mencoba menyaingi produk *Nitendo*, *Sega* merilis *console next-generation* mereka, yaitu *Sega Mega Drive* (yang juga dikenal dengan *Sega Genesis*). *Console* ini menyajikan gambar yang lebih tajam dan animasi yang lebih halus dibandingkan dengan NES. *Console* ini berhasil memberi tekanan, tetapi NES tetap bertahan dengan angka penjualan tinggi.

Selanjutnya pada tahun 1990, *Nitendo* kembali menggebrak dengan *console next-generation* merekan dengan produk *NESES* (*Super Nitendo Entertaintment System*). Selama empat tahun *Nitendo* dan *Sega* menjadi musuh bebuyutan, meskipun ada beberapa produsen seperti *SNK*, dengan *NeoGeo*-nya, *NEC* dengan *TurboGrafx-16* dan *Philips CD-i*.

Rifalitas yang legendaris, *Super NES* dan *Mario Bross* sebagai ikonnya melawan *Sega Mega Drive* dengan *Sonic The Hedgehog* sebagai ikonnya.

5. *Game* Generasi Kelima

Pada tahun 1990-1994, *Sega* dan *Nitendo* kembali bersaing. Berbagai *Game* fenomenal dirilis. SNES menyertakan *chip Super FX* pada *cartridge* mereka, dan Sega menggunakan *Sega Virtual Processor*. Keduanya bertujuan untuk meningkatkan kualitas grafis dari *Game*. Alhasil, SNES dan *Sega* saling beradu dengan *Game*-nya seperti *Donkey Kong Country* (SNES) dan *Vectorman* (*Sega*).

Tetapi, pada tahun 1993 sebuah perusahaan bernama *Panasonic* merilis *console*-nya yaitu *Panasonic 3DO*. Ini adalah *console* pertama yang menggunakan *CD* sebagai pengganti *cartridge*. Harganya sangat mahal membuat *console* ini tidak popular. *3DO* tidak bertahan lama dan harus segera menghentikan produksinya. *3DO* adalah *console* pertama yang menggunakan media *CD*.

Akhirnya, pada tahun 1944 *Atari* lembali memunculkan *console* barunya untuk menandingi *Nitendo* dan *Sega*. *Atari Jaguar* jelas jauh lebih canggih ketimbang *NES* maupun *Sega Drive*, tetapi penggunaannya yang sulit menjadi batu sandungan. Pada tahun yang sama *Sony* merilis *console super* canggih, yaitu *PlayStation*. *Atari* pun bangkrut dan akhirnya melakukan *merger*. *Console* Jepang ini segera mendapat sambutan hangat hingga saat ini.

6. *Game* Generasi Keenam

Pada tahun 1998 setelah jatuhnya *Nitendo* dan *Sega*, kini dunia *console* menjadi milik *Sony*. *PlayStation* manjadi raja dan bisa dibilang tidak memiliki pesaingan, *Sega* mencoba meluncurkan *Sega Dreamcast* untuk mematahkan dominasi *Sony PlayStation*, tetapi kembali gagal dan akhirnya tahun itu juga *Sega* mengundurkan diri dari dunia produsen *console*.

Di tahun 2000, *Sony* semakin merajalela ketika mereka berhasil merilis *console* barunya, bernama *PlayStation2* yang telah berbasis *DVD*. *Nitendo* mencoba bertahan di dunia *console* dengan merilis *GameCube*. *Console* ini tidak menggunakan *DVD* 12cm biasa, melainkan *DVD* yang berukuran lebih kecil, yaitu 8cm. Ukuran medianya yang tidak umum menjadi *GameCube* kurang popular. Satu-satunya pesaing serius *PlayStation2* adalah *Xbox*. Sebuah *console* keluaran *Microsoft* yang menggebrak dengan tampilan *visual* yang sangat tajam dan berkuaitas. Sayangnya *Game Xbox* tidak sepopular *PlayStation2*.

7. *Game* Generasi Ketujuh

Tahun 2005, *Xbox* teramabat meluncur ke pasaran dibanding *PlayStation2*, dan *support Game* tenar juga sangat minim. Tetapi, *Microsoft* seolah beajar dari

kesalahan. Pada saat *Sony* masih melakukan riset untuk *console PlayStation3* yang menggunakan *Blu-Ray*, *Microsoft* kali ini telah mengambil langkah inifiatif dengan mengeluarkan *Xbox 360*, sebuah *console* generasi terkini yang memanfaatkan media *HD-DVD*.

Xbox 360 hadir dengan segudang *fitur* istimewa, mulai dari grafis hingga kecerdasan yang terdapat dalam permainan (*Artificial Intelligence*). Yang akhirnya memperoleh *Best Game Of Years 2006*, selang seminggu sebelum *Nitendo* meluncurkan terobosannya, yaitu *Nitendo Wii*. Posisi *PlayStation3* kurang menuntungkan. Selain harganya mahal, *console* ini juga harus didukung dengan kualitas *device* yang canggih.

8. *Game* Generasi *Handleheld*

Merebaknya popularitas *Game* membuat berbagai perusahaan elektronik berusaha membuat trobosan baru. Di antaranya adalah membuat sebuah mesin *Game* berukuran kecil, yang bisa dibawah kemanapun. Sekitar tahun 1976-1979, sejarah *video Game* saku ini bermula, beberapa piranti dan *mattel* dirilis ke pasaran, tetapi tidak begitu popular. Demikian dengan *handheld* buatan *Milton Bradley* yang dilempar ke pasaran.

Pada tahun 1980-1984 perusahaan-perusahaan Jepang mulai merambah pasar Handleld, tetapi tetap saja hasilnya. Berlanjut hingga 1984, hingga *Game*boy pun muncul. Handleld buatan Nitendo ini begitu diminati dan dinobatkan sebagai Hendleld pertama di dunia yang angka penjualannya dikatakan sukses.

Pada tahun 1989, *Atarai* mengakhiri era *Handleld* hitam putih. Produk andalannya adalah Atari *Lynx* yang membawa dimensi baru. Ini adalah *Handleld* pertama yang mampu menampilakan warna, sekaligus animasi 3 dimensi (3D) yang sederhana. Pada awal 1990 dunia *Handleld* semakin menggila, *NEC*, perusahaan elektronik terkemuka di Jepang membuat *Handleld* yang mampu membuat animasi 3D lebih kompleks, karena menerapkan konsep grafis 3D untuk PC.

Kemudian pada tahun 1998-2000, *Sony* merilis *Pocket Station* dan mmemberikan kejutan besar di dunia *console*. *Handleld* ini memiliki kualitas visual

yang juah lebih baik dibandingkan *handleld* lain yang ada di pasaran. Salah satunya adalah *Nitendo* dan *Gumpei Yokoi*, memutuskan untuk keluar dan bergabung dengan *Bandai*, kemudian merilis *WonderSwan* dan *WonderSwan Color*.

Akhirnya pada tahun 2000-2006, *Sony* merilis *handleld* pertama yang menggunakan cakram bernama *PSP (PlayStation Potable)* dan dibarengi dengan hadirnya *Nitendo DS*, yang menggunakan konsep *dual screen*. Lalu disusul oleh *Game Boy Micro* dan *Game Park XGP*. *Nitendo DS Lite* dan *Pelican VG Pocket Caplet* menjadi *handleld* terbaru yang dilempar ke pasaran.

2.3. Syarat Perancangan *Game*

Berikut adalah syarat utama dalam merancang sebuah *game*, yaitu:

1. *Game are not linear*

Pada umumnya di dalam sebuah *game* terdapat tingkat kesulitan yang berbeda. Tetapi dalam menyelesaikan setiap kasus di dalam *game*, pemain tidak harus menyelesaikannya secara *linier* ataupun berurutan. Melainkan pemain dapat menyelesaikannya dengan mendahulukan cara yang dianggap mudah. Berdasarkan alur tersebut, maka *Game* dikatakan tidak *linear* (lurus).

2. *Game have a goal*

Di dalam merancang sebuah *Game*, *Game* harus memiliki tujuan dan hasil akhir permainan. Seperti halnya pada permainan catur, pemain harus mengalahkan lawan dengan cara "*checkmate*". Atau seperti pada *Game RPG*, pemain harus bertemu dengan salah satu tokoh di *game* tersebut untuk memperoleh *item* yang berguna pada perjalanan selanjutnya.

3. *Game must be winnable*

Di dalam merancang sebuah *Game, Game* tersebut juga harus memiliki cara untuk dapat dimenangkan oleh pemain. Jadi, di dalam permainan tersebut antara pemain dengan musuh atau antara pemain dengan pemain memiliki kesempatan menang yang sama besar.

4. *Start, Middle and Ending Of The Game*

Di dalam sebuah *game, game* wajib memiliki posisi awal (*starting point*) yang mana pada posisi ini pemain akan mengawali permainannya. Sedangkan untuk *Middle*, merupakan alur perjalanan atau proses yang terjadi selama permainan sehingga sampai pada tujuan permainan atau akhir dari permainan tersebut (*Ending*).

2.4. Jenis *Game*

Berikut ini adalah jenis *game* yang ada hingga saat ini:

1. *Action Game*

Action Game merupakan jenis *Game* yang menampilkan sebuah medan pertempuran atau sebuah kasus yang harus diselesaikan dengan cara bertempur dengan dilengkapi persenjataan yag komplit, disertai perpindahan tempat, memiliki tensi yang tinggi, dan umumnya diambil dari sudut pandang orang pertama. Contoh: *Sierra, Conter Strike, Resident Evil, Batman*, dan sebagainya.

2. *Adventure Game*

Adventure Game adalah *game* berjenis petualangan dengan alur cerita yang berkesinambungan disertai perpindahan atau pergantian tempat dari suatu wilayah ke wilayah yang lain dan biasanya disertai dengan teka-teki untuk mengakhiri sebuah misi. Contoh : *Super Mario Bross, Tiny Toon, Donkey Kong*, dan sebagainya.

3. *Casual Game*

Casual merupakan *game* hasil implementasi dari permainan tradisional maupun permainan sehari-hari dan biasanya menggunakan papan sebagai alasnya. Contoh: *Sudoku Cafe, Chessmaster 9000, Poker Texas, Sanghai Mahjong*, dan sebagainya.

4. *Educational Game*

Merupaka *game* menggunakan media pembelajaran interaktif sebagai intinya. Yang mana *game* ini untuk memberikan pembelajaran, berkaitan dengan penerapan ilmu pengetahuan, ataupun memberi stimulus pada kinerja saraf dan otak pemain.

Contoh: *Barbie Secret Agent, Blue's Clues Learning Time, Zuma, Angry Bird*, dan sebagainya.

5. *Role Playing Game*

Role Playing Game (RPG) merupakan jenis *game* yang menerapkan sebuah tokoh (*avatar*) untuk menjalani sebuah cerita layaknya kehidupan sehari-hari yang nyata. *game* ini tidak cukup diselesaikan dalam waktu satu hari atau sekali main, karena umunya *game* jenis ini berkaitan dengan *level avatar*, menjalankan misi yang berbeda pada setiap tempat, arena permainan yang luas, dan meningkatkan *level* setiap musuh. Pada saat ini *game* RPG umumnya disajikan dengan gafis ysng menarik, adanya transaksi jual beli antara pemain dan pelaku *game* lainnya, aturan pertarungan dilakukan dengan *turn-based*, dan adanya aksesoris untuk menambah kinerja *avatar*. Contoh: *GTA, Raknarok, Zoid, Dragon Nest*, dan sebagainya.

6. *Simulation Game*

Adalah jenis *game* yang mengkondisikan pemain untuk menirukan kegiatan layaknya di dunia nyata secara alami. *game* jenis ini mengacu dan bersumber sesuai kenyataan dan dapat diterapkan pula di dunia nyata. Contoh : *Grand Turismo, Playboy, Flight Simulator 2002*, dan sebagainya.

7. *Sport Game*

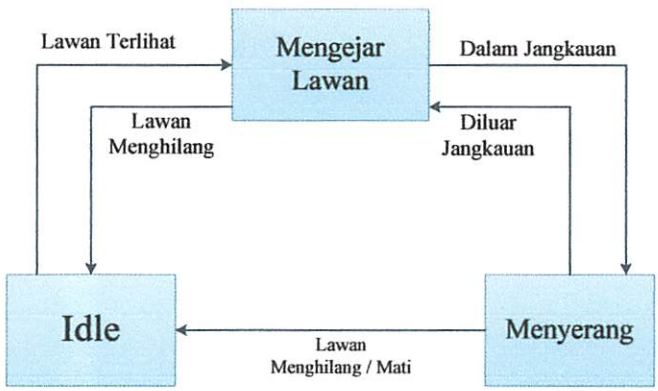
Adalah jenis *game* dengan mengadopsi pada kegiatan olah raga di kehidupan nyata serta dilengkapi dengan aturan mainnya. *Game* jenis ini merupakan *game* yang melibatkan kecerdasan yang diterapkan pada musuh, kecerdasan lainnya juga diterapkan pada wasit, juri, dan peraturanpertandingan (penentuan pelanggaran). Contoh: *Pro Evolutin Soccer (PES), Winning Eleven, Street Ball, WWE*, dan sebagainya.

8. *Strategi Game*

Merupakan jenis *game* yang pada umumnya identik dengan situasi peperangan, biasanya menampilka pemandangan dari atas (*angel from top*), arena tempur digambarkan sebagai peta yang bersegmen, pemain dimulai dengan membangun sendiri markas dan perlengkapan pemain. *Game* ini tidak memainkan tensi yang tinggi, tapi menuntun pemain untuk perfikir logis, cepat dan tepat dalam memenangkan pertempuran. Contoh: *Red Alert, Vietnam Squad Battles, Resign Of Chaos, Soccer Manager*, dan sebagainya.

2.5 *Finite State Machine*

Finite State Machine merupakan salah satu logika penalaran yang memperlihatkan perilaku *system* dengan berdasarkan tiga hal, yaitu *state* (keadaan), *event* (kejadian), dan *action* (aksi). Pada suatu saat, *system* akan berada pada salah satu *state* yang aktif. Sistem dapat beralih atau bertransisi menuju *state* lain jika mendapatkan masukan atau *event* tertentu. Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh *system* ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relative kompleks.



Gambar 2.1 Contoh Alur diagram *Finite State Machine* Sederhana

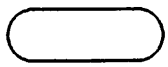
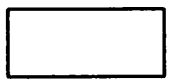
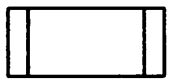
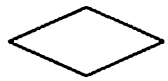
Pada gambar 2.1, diagram tersebut memperlihatkan FSM dengan tiga *action* dan tiga *event* serta lima *state* yang berbeda: ketika *system* mulai dihidupkan, *system*

akan bertransisi menuju *action Idle*, pada *event* ini *system* akan menghasilkan *action* mengejar musuh ketika terjadi *state* musuh tampak, sedangkan jika terjadi *state* musuh dalam jangkauan makan akan menghasilkan *action* menyerang. Dan ketika terjadi *state* musuh menghilang makan akan menghasilkan *action Idle* dan seterusnya.

2.6 Diagram Alir

Diagram Alir adalah diagram yang menggambarkan bagaimana jalannya program maupun system kerja mulai dari awal hingga akhir. Setiap diagram alir harus memiliki titik awal dan titik akhir (*start and stop*). Diagram alir dibentuk dengan memanfaatkan simbol-simbol tertentu. Perancangan sebuah diagram alir umumnya sebagai bahan mentah sebelum proses coding sesungguhnya. Pada table 2.1 terlihat simbol-simbol yang digunakan pada diagram alir.

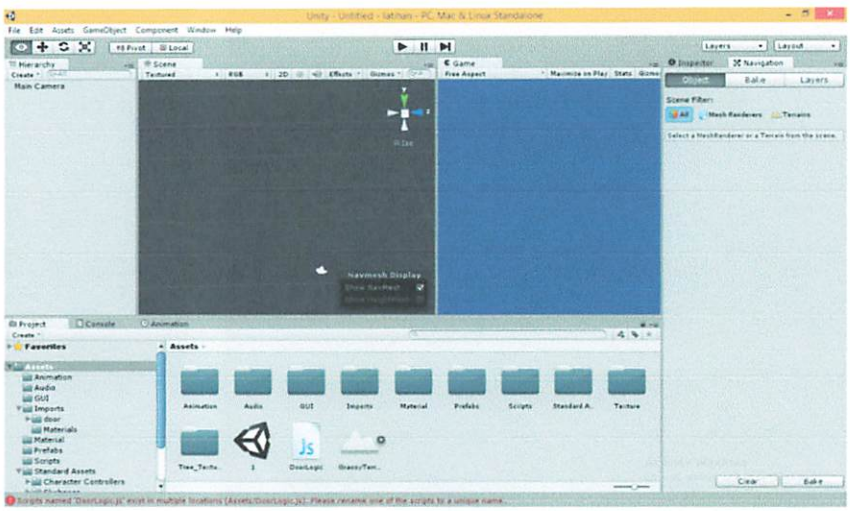
Tabel 2.1 Simbol Diagram Alir

SIMBOL	NAMA	FUNGSI
	**TERMINATOR**	Permulaan/akhir program
	GARIS ALIR (FLOW LINE)	Arah aliran program
	**PREPARATION**	Proses inisialisasi/pemberian harga awal
	**PROCESS**	Proses perhitungan/proses pengolahan data
	**INPUT/OUTPUT DATA**	Proses input/output data, parameter, informasi
	**PREDEFINED PROCESS (SUB PROGRAM)**	Permulaan sub program/proses menjalankan sub program
	**DECISION**	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
	**ON PAGE CONNECTOR**	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	**OFF PAGE CONNECTOR**	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

2.7 Unity

Unity adalah sebuah sebuah *tool* yang terintegrasi untuk membuat *game*, arsitektur bangunan dan simulasi. *Unity* bisa untuk *games* PC dan *games Online*. Untuk *games Online* diperlukan sebuah plugin, yaitu *Unity Web* Player, sama halnya dengan *Flash Player* pada *Browser*.

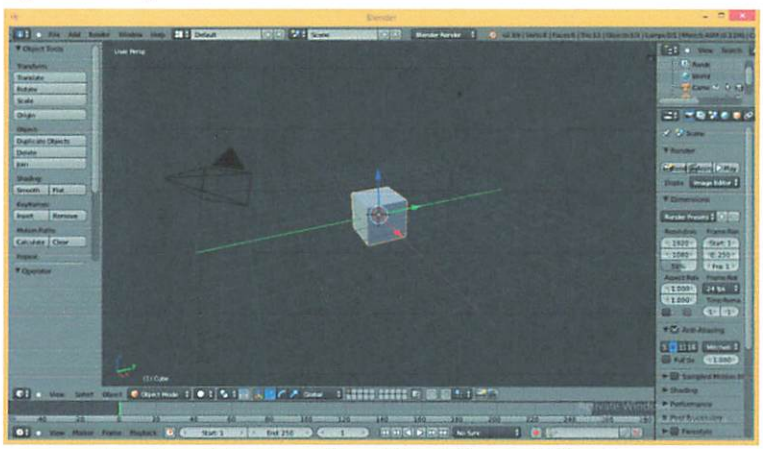
Unity tidak dirancang untuk proses desain atau modelling, dikarenakan *unity* bukan *tool* untuk mendesain. Jika ingin mendesain, pergunakan *3D editor* lain seperti *3dsmax* atau *Blender*. Banyak hal yang bisa dilakukan dengan *unity*, ada fitur *audio reverb zone*, *particle effect*, dan *Sky Box* untuk menambahkan langit.



Gambar 2.2 Tampilan WorkSpace Unity.

2.7 Blender

Blender adalah program 3D dan animasi yang bersifat *opensource*, bebas untuk dikembangkan oleh penggunanya dan dapat didistribusikan kembali dan bersifat Legal. *Blender* memiliki video *compositor* dan *intergrated game engine*, Karya yang dihasilkan tidak ada sifat royalty kepada *developer*, dan dapat dipublikasikan baik *free* maupun untuk dikomersilkan.



Gambar 2.3 Tampilan WorkSpace Blender.

BAB III
ANALISA DAN PERANCANGAN

3.1 Analisis Sistem

Analisis sistem merupakan penguraian dari suatu informasi yang utuh ke dalam bagian komponen-komponennya dengan maksud untuk mengidentifikasi permasalahan, baik analisis kebutuhan non-fungsional maupun analisis kebutuhan fungsional.

3.1.1 Analisis Masalah

Analisi masalah merupakan proses identifikasi serta evaluasi terhadap *game* sejenis dan *game* yang akan dibangun oleh penulis.

Dalam *game* petualangan, kita dituntut untuk menyelesaikan beberapa misi yang telah tersedia. Dalam menyelesaikan misi tersebut tentunya kita butuh strategi untuk memecahkan setiap misi. Dengan memikirkan strategi jalan tentunya secara tidak langsung otak kita diperas agar dapat berhasil menjalankan misi. Tentunya kita akan berfikir dua kali yang secara tidak langsung akan membuat pola pikir kita berubah yakni harus berpikir dua kali sebelum melakukan sesuatu.

.*Game* dengan *genre Adventure Game* dengan *First Person Controller* (sudut Pandang orang pertama) yaitu tampilan *game* yang terlihat seperti penglihatan manusia pada umumnya dan sudah tidak asing lagi dalam dunia *game* khususnya di Indonesia. Dalam perkembangannya *genre game* seperti ini telah berkembang pesat karena jenis permainan ini terbilang menantang dan menyenangkan untuk dimainkan oleh masyarakat terutama remaja-remaja. Beberapa *game* yang memakai *genre* ini di antaranya *Battlefield*, *FarCry,* dan *Call of Dutty*.

3.1.2 Pengenalan *Game Survival from Island*

Game yang akan dibangun adalah *game Survival from Island* dengan *genre Adventure Game* dengan *First Person Game*. *Game* ini dibangun dengan mengaplikasikan teknologi sebagai sarana untuk meningkatkan kemampuan, serta ketelitian para pemain. Selain itu kemampuan koordinasi antara mata dan tangan bisa diasah melalui media *game* ini Fitur-fitur yang digunakan pada *game* ini adalah:

1. Grafik yang digunakan adlah 3 Dimensi (3D).
2. *Game* ini menerapkan *genre Adventure Game* dengan *First person Controller.*
3. Metode yang digunakan adalah *Finite State Machine.*
4. FSM terletak pada perilaku musuh ketika sensor mengetahui keberadaan pemain, melihat pemain, mendekati pemain, dan menyerang pemain.

3.1.3 *Storyline*

Pada *game Survival from Island* ini terdapat 4 *level* yang harus dilalui oleh pemain. Apabila ingin melewati suatu *level*, Pemain diharuskan untuk menyelesaikan *Quest* / Misi yang terdapat pada setiap *level*. Sebelum melewati *level* pertama pemain akan menjumpai cerita perjalanan seseorang mantan *veteran* Perang yang melakukan perjalanan liburan dengan kapal pesiar, namun diperjalanan terjadi kecelakaan yang mengakibatkan pemain terdampar di sebuah pulau. Pada *level* pertama pemain hanya melewati satu misi yang harus diselesaikan, yaitu mencari sesuatu alat / senjata untuk dapat bertahan hidup dipulau tersebut, yang kemudian dapat melaju ke *level* berikutnya.

Kemudian pada *level* berikutnya yaitu *level* dua, Pemain akan dihadapi pada suasana tengah hutan dimana terdapat banyak sekali hewan-hewan liar yang siap menerkam pemain. Di *level* dua ini pemain akan mendapatkan misi dimana diharuskan mencari jalan keluar dari hutan, dan bertahan hidup dari serangan-serangan musuh. Jika pemain berhasil menemukan jalan keluar maka akan melanjutkan ke *level* berikutnya yaitu *level* tiga.

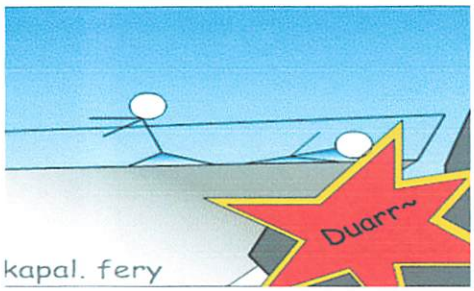
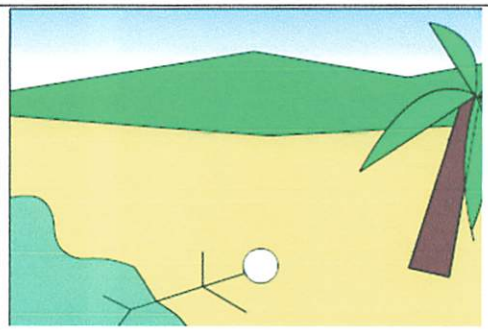
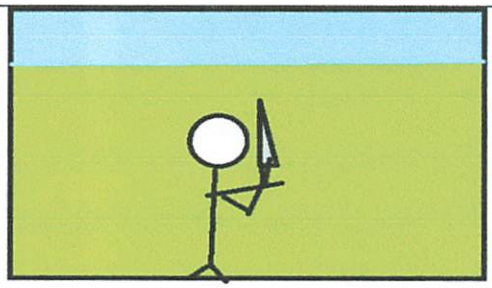
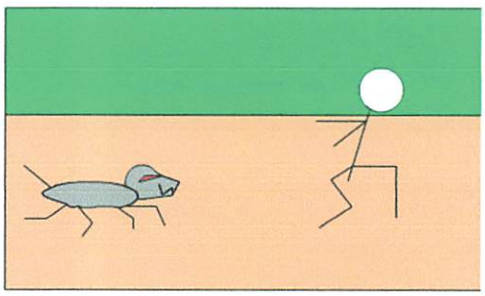
Setelah berhasil melewati *level* 2 pemain akan sampai pada *level* 3. Disini pemain ditengah perjalannya akan menemukan sebuah rumah / gubuk tua dimana disana dia akan menemukan alat untuk meminta bantuan kepada orang lain di luar pulau, tidak hanya itu disana juga terdapat senjata yang cukup kuat untuk pemain gunakan. Didalam *level* ini juga pemain akan dihadang mahluk buas lainnya, yaitu Serigala hutan, dimana kemampuan dari serigala ini jauh lebih ganas. Di *level* ini setelah menemukan alat di gubuk tua itu selanjutnya pemain diharuskan mencari jalan keluar dan menuju ke *level* selanjutnya yaitu *level* 4 (*Final*).

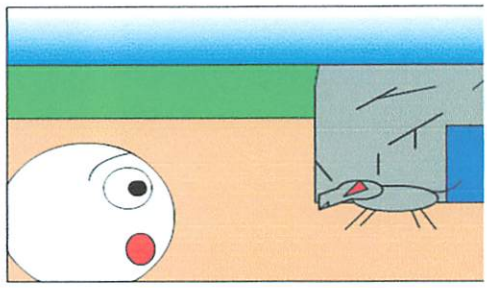
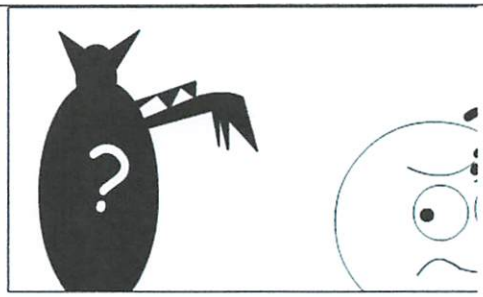
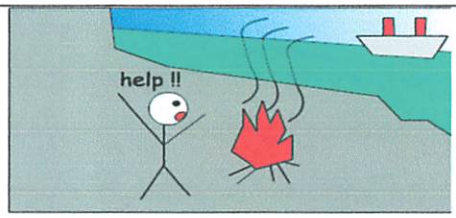
Level 4 ini adalah level terakhir dalam game ini. pemain akan berada pada sebuah perjalanan mendaki gunung yang terjang, dimana untuk mengaktifkan alat yang ditemukan di *level* 3 hanya bisa diaktifkan dengan *tower* yang cukup tinggi. Maka dari itu disini pemain memiliki misi untuk menuju *tower* yang ada diatas pegunungan, namun tidak sampai disitu saja, disana pemain akan berhadapan dengan musuh terakhir (*boss*) yaitu seekor *Monster* misterius yang akan siap menerkam pemain. Setelah *Monster* tersebut telah dikalahkan maka pemain dapat mengaktifkan alat yang didapatkan sebelumnya di dekat tower tersebut dan pemain pun dapat menghubungi orang diluar pulau untuk menyelamatkan.

3.1.4 *Storyboard*

Adalah serangkaian sketsa dibuat berbentuk persegi panjang yang menggambarkan suatu urutan (alur cerita) elemen-elemen yang diusulkan untuk aplikasi multimedia. Gambar 3.1 berikut adalah penggalan *storyboard.*

Tabel 3.1 *Storyboard Game*

Stage	Scene	Keterangan
Intro		Saat liburan dengan menggunakan kapal pesiar, tiba-tiba terjadi kecelakaan yang mengakibatkan kapal dan penumpang tenggelam.
Intro		Si *Player* terdampar di sebuah pulau misterius
Level 1		Mencari benda-benda yang dibutuhkan untuk bertahan hidup di dalam pulau
Level 2		Melawan beberapa Hewan buas dalam perjalanan menyusuri pulau.

Stage	Scene	Keterangan
Level 3		Menemukan sebuah gudang tua, dimana gudang tersebut dijaga oleh beberapa hewan penghuni pulau tersebut.
Level Final		Melawan mahluk misterius yang ada di pulau. Dimana *player* akan di kejar oleh mahluk tersebut.
Last Scene		*Player* menemukan kapal pesiar yang lewat, dan menyelamatkan diri.

3.1.5 *Game Play*

Pada permainan *game Survival from Island* terdapat empat *level* yang harus dilewati pemain agar dapat menyelesaikan *game Survival from Island* ini. Langkah-langkah bermain pada *game Survival from Island* antara lain:

a. Pada *level* satu pemain harus mencari sebuat senjata yang berada di stage untuk menyelesaikan *level.*
b. Dengan mengalahkan para musuh seperti hewan buas, pemain dapat menyelesaikan misi.
c. Pemain diberikan kemampuan untuk meyerang musuh dengan berbagai senjata, serangan akan mengakibatkan kematian pada musuh.
d. Jika pemain terkena serangan musuh maka *health* pemain akan berkurang.

e. Jika *health* pada nilai nol pemain pamain akan mati.
f. Permainan akan berakhir jika pemain mati atau pemain telah menyelesaikan misi.
g. Jika pemain mati bisa kembali bermain dengan menekan tombol *Respawn* dan status *health* akan kembali normal.

3.1.6 Analisa Kebutuhan Non-Fungsional

Pada analisis kebutuhan non-fungsional ini dijelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis pengguna.

3.1.6.1 Analisi Kebutuhan Perangkat Lunak

1. Spesifikasi perangkat lunak bagi pengembang yang digunakan dalam membangun aplikasi *game Survival from Island* adalah:
   a. Sistem Operasi Windows XP
   b. Unity 4.3.4
   c. Blender
2. Perangkat lunak yang digunakan oleh pemain untuk menjalankan *game Survival from Island* adalah sistem operasi Windows (Windows XP, Windows Vista, Windows 7).

3.1.6.2 Analisis Kebutuhan Perangkat Keras

1. Spesifikasi minimum perangkat keras yang dibutuhkan oleh pengembang adalah:
   a. Prosesor dengan kecepatan 1.6 Ghz.
   b. RAM 1 Gb.
   c. Hardisk 20 Gb.
   d. Monitor.
   e. VGA Card 256 Mb.
   f. *Mouse dan Keyboard*
   g. Speaker

2. Spesifikasi *minimum* perangkat keras yang dibutuhkan oleh pemain adalah:
    a. Prosesor dengan kecepatan 1.6 Ghz ke atas.
    b. RAM 1 Gb.
    c. Hardisk 700 Mb.
    d. Monitor.
    e. VGA Card 128 Mb.
    f. Mouse dan Keyboard
    g. Speaker

3.1.6.3 Analisi Pengguna

Selain dibutuhkannya perangkat keras dan perangkat lunak, user juga sangat dibutuhkan dalam menggunakan aplikasi *game* ini. Adapun spesifikasi *user* yang dibutuhkan antara lain:

1. Prioritas *game* ini dibuat untuk anak-anak antara umur 15 sampai 25 tahun.
2. Pengguna umum yang mengerti dalam penggunaan komputer.

3.1.7 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem berjalan dengan baik sesuai kebutuhan.

1. *Game* harus ada keterangan yang menjelaskan peraturan permainan yaitu bagaimana kondisi yang harus dicapai pemain agar dapat menyelesaikan setiap *level*, dengan begitu akan memudahkan pengguna untuk memainkannya.
2. Pada menu utama pemain bisa memilih untuk memulai bermain, mempelajari petunjuk permainan atau menuju menu pengaturan.
3. Pada setiap *level* terdapat *Quest /* misi , misi harus diselesaikan pemain, selama melewati *level* akan ditampilkan misi dari pemain.

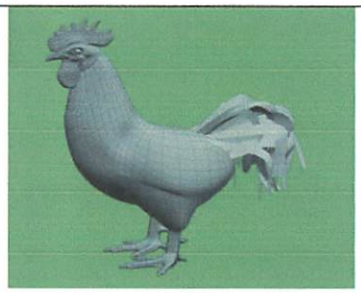
3.2 Perancangan Sistem

Perancangan sistem adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan untuk memberikan gambaran secara terperinci. Pada *game Survival from Island* , metode FSM diterapkan dalam setiap musuh dalam *game*. Terdapat dua susunan aksi yang akan dilakukan oleh musuh dalam *game Survival from Island* yaitu pemain berada dalam jarak jangkauan – melihat pemain pemain dan dalam jangkauan serang – mendekati pemain – menyerang pemain.

3.2.1 Karakter

Terdapat beberapa karakter dalam *game Survival From Island* ini. dikarenakan game ini bersifat *First person Controller* (Sudut pandang orang pertama) maka pemain/*player* tidak memiliki tampilan secara menyeluruh. Beberapa karakter yang akan muncul di *game* diantaraya :

Tabel 3.2 *Karakter Game*

Karakter	Tampilan	Keterangan
Ayam		Karakter ini akan menjadi item untuk menambah *health* dari pemain didalam game.
Anjing Hutan		Karakter *Enemy/* lawan dari *player* dimana anjing ini akan menyerang pemain yang ada didekatnya.

Karakter	Tampilan	Keterangan
Serigala		Karakter ini bersifat agresif dalam menyerang *Player* . memiliki daya serang yang cukup kuat.
Harimau		Karakter ini cukup kuat dan memiliki sensor yang cukup peka untuk melihat pemain.
Mysterious Monster		Karakter ini adalah musuh terakhir yang akan dihadapi oleh *player*.

3.2.2 Pengenalan *Level Environtment*

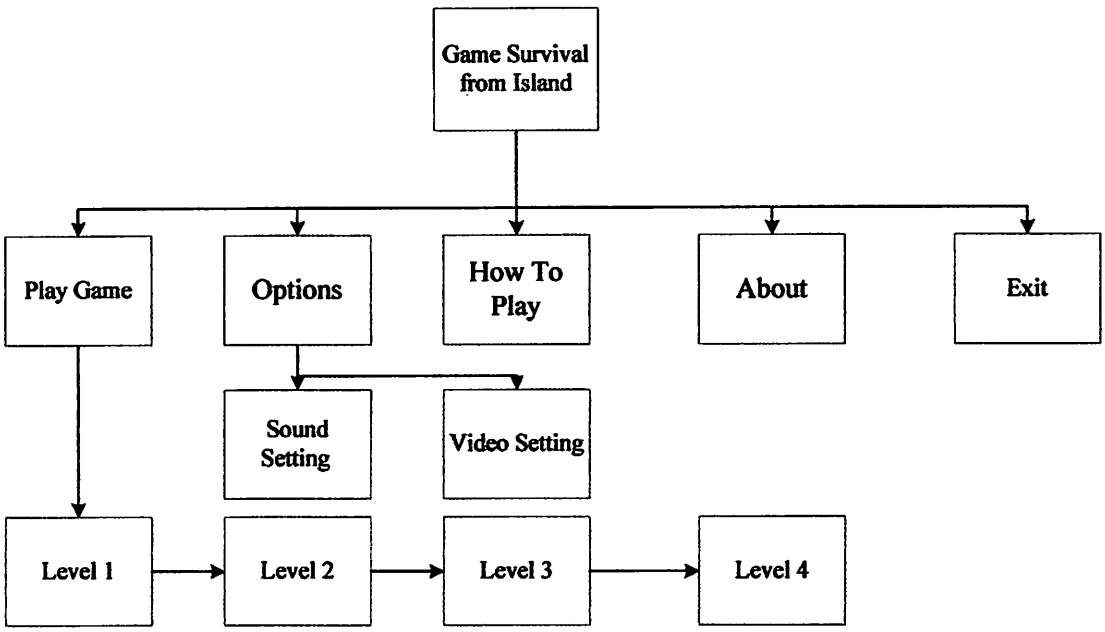
Pengenalan *level environtment* adalah pembahasan mengenai lingkungan yang terdapat dalam *game Survival from Island*. Nama *level* yang terdapat dalam *game Survival from Island* beserta keterangannya dapat dilihat pada table 3.3.

Tabel 3.3 Pengenalan *Level Environtment*

No	Nama	Keterangan
1.	*Level* 1	Pada *level* satu pemain diberikan misi untuk mencari sebuah alat sebagai senjata. *Environtment* dari *level* satu berada di tepi Pantai.
2.	*Level* 2	Pada *level* dua pemain diberikan misi untuk mengalahkan musuh yang menghadang dan mencari jalan keluar. *Environtment* dari *level* dua berada di tengah hutan.
3.	*Level* 3	Pada *level* tiga pemain diberikan misi untuk mengalahkan serigala yang menghadang dan mencari peralatan di gubuk tua. *Environtment* dari *level* tiga berada di tengah hutan berkabut.
4.	*Level* 4	Pada *level* empat pemain diberikan misi untuk mendaki gunung yang terjang untuk menuju tower yang ada di atas gunung, dan di puncak gunung pemain diharuskan berhadapkan dengan musuh terakhir. *Environtment* dari *level* empat berada di Gunung.

3.2.3 Perancangan Struktur Menu

Perancangan struktur menu adalah perancangan tata urutan menu dari *game Survival from Island.* Perancangan struktur menu pada *game Survival from Island* dapat dilihat pada gambar 3.1



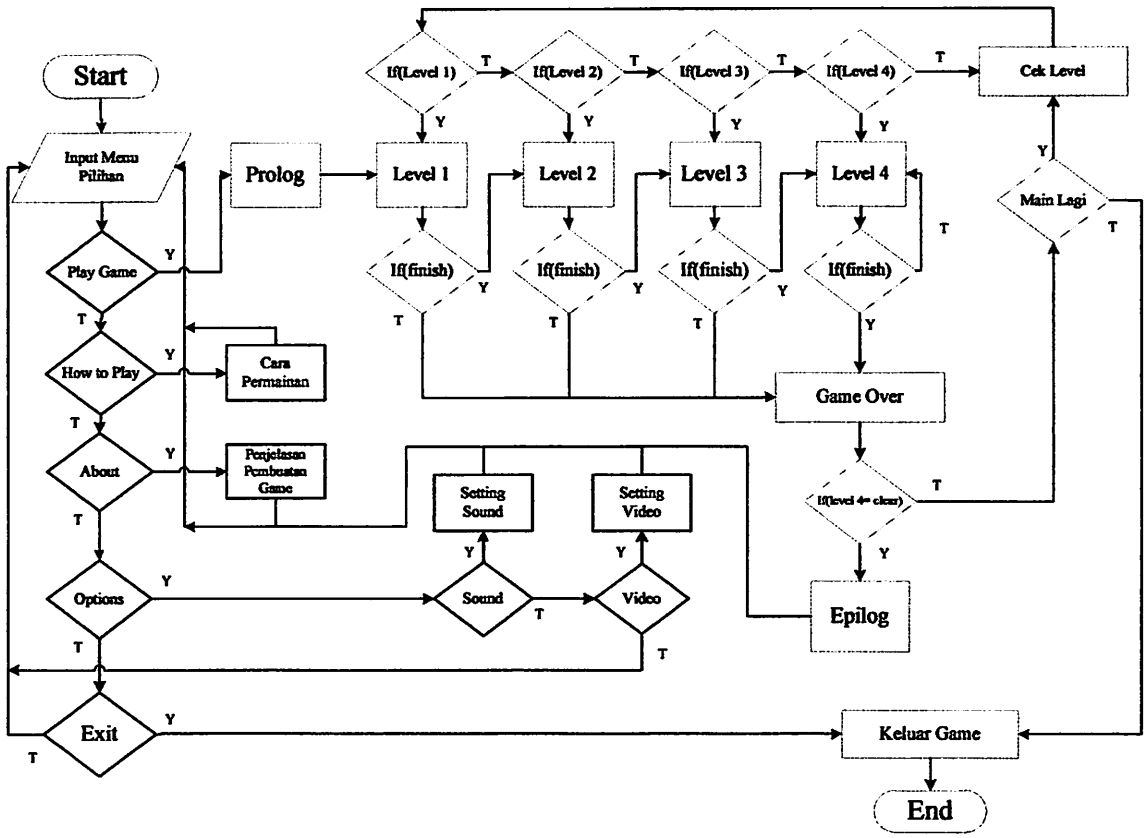
Gambar.3.1 Struktur Menu

Pada gambar 3.1, *game* dimulai dari menu utama dimana pada tampilan menu utama terdapat beberapa pilihan menu yaitu *Play Game*, *Options*, *How to Play, About,* serta menu untuk keluar dari *game*. Pilihan pertama pada menu utama adalah *Play Game*, menu ini digunakan untuk memulai permainan dari *game Survival from Island*, apabila pemain telah memilih menu mulai main pemain akan dihadapkan pada menu tampilan dari *level* satu yang kemudian dapat dilanjutkan ke *level* selanjutnya sampai *level* empat apabila pemain dapat menyelesaikan *level*.

Selanjutnya adalah pilihan menu kedua pada menu utama yaitu *Option*, yang memiliki fungsi untuk mengatur *sound* dan *video*. Berikutnya adalah menu *How to Play* yang berisi penjelasan tentang cara untuk memainkan game. Selanjutnya adalah menu *About* yang berisi tentang *game* yang dibuat. Dan yang terakhir menu keluar digunakan apabila pemain ingin keluar dari permainan, menu pilihan keluar akan mengeluarkan pemain dari permainan serta menutup program.

3.2.4 Alur *Game*

Perancangan Alur berfungsi untuk mengetahui alur proses dari program dimulai dari *start* program sampai pada *end* program. Alur dari *game Survival from Island* ditunjukkan dengan *flowchart* pada gambar 3.2



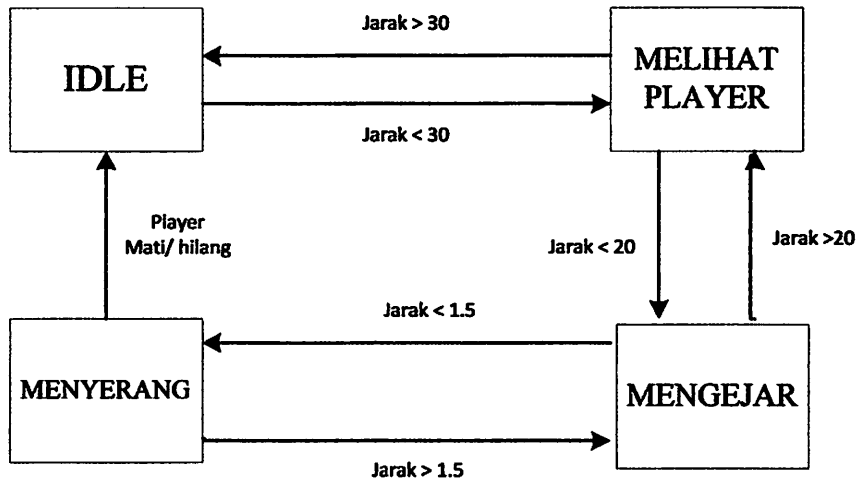
Gambar.3.2 *Flowchart* Alur Game

Seperti yang terlihat pada gambar 3.2, program dimulai dari *start* program yang kemudian dilanjutkan pada menu utama yaitu menu tampilan pilihan untuk memulai permainan, pengaturan, petunjuk, tentang game, serta terdapat *opsi* untuk keluar. Pada *opsi* pengaturan akan terdapat beberapa pengaturan sebelum permainan dimulai seperti pengaturan audio dan video grafik. Menu *How To Play* untuk mengetahui cara permainan dan menu *About* untuk penjelasan game Kemudian *opsi* keluar digunakan apabila pemain tidak ingin untuk memulai permainan, *opsi* ini berfungsi untuk keluar dari *game* dan mengakhiri program.

Apabila pemain memilih untuk mulai main, pemain akan dihadapkan pada cerita prolog yang kemudian dilanjutkan ke *level* satu sampai *level* empat, di sini pemain akan memulai permainan dan melewati rintangan pada setiap *level* selanjutnya. Akan tetapi jika pemain gagal untuk melewati rintangan atau kehabisan *helath* maka akan ada pilihan *respawn* (main lagi) atau *main menu.* Apabila pemain memilih *respawn* maka pemain akan memainkan level yang sama dan jika milih *main menu* game akan berakhir atau *Game Over* dan dikembalikan ke tampilan awal.

3.2.5 Alur FSM

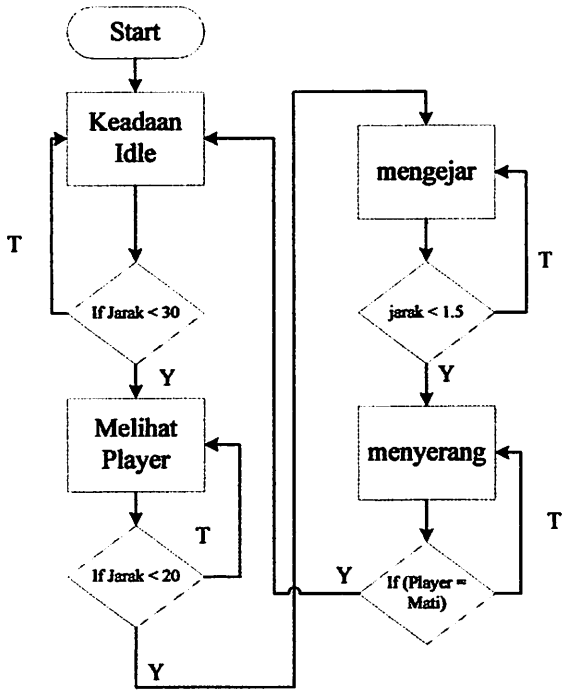
Alur metode *Finite State Machine* yang terdapat pada musuh-musuh seperti anjing, harimau, dan serigala pada *level 2,* dan *level 3* dalam *game Survival from Island* ditunjukan pada gambar 3.3.



Gambar 3.3 Perancangan Aksi Karakter Anjing, Harimau, dan Serigala.

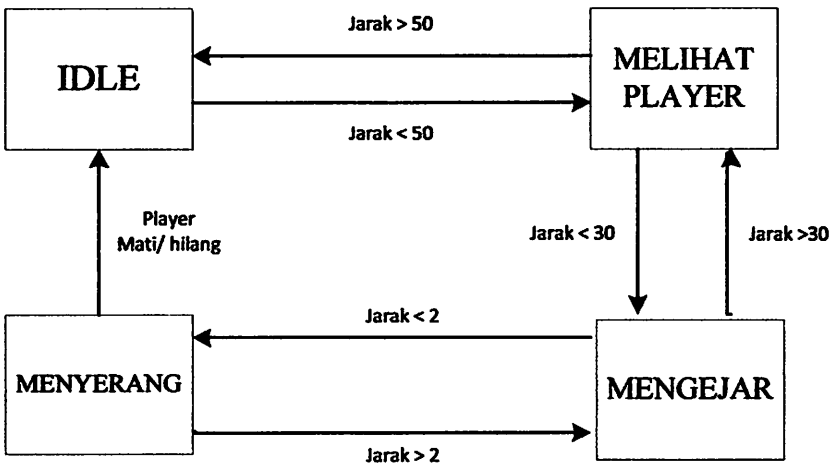
Pada gambar 3.3, metode ini ditanamkan pada musuh dalam *game Survival from Island.* Aksi awal yang dilakukan musuh adalah *Idle*, jika jarak musuh dengan *player* lebih dari 30 *unit* maka musuh akan tetap diam di tempat awal, jika jarak dengan *player* kurang dari 30 *unit* maka musuh akan melihat ke *player* dan jika kurang dari 20 *unit* dari *player* maka musuh akan bergerak mendekati *player* dan jika jarak antara musuh dan *player* sudah lebih kecil 1.5 *unit*, maka musuh akan melakukan aksi penyerangan kepada *player.*

Alur aksi musuh troll dan bos musuh juga dijelaskan dengan *flowchart* dalam gambar 3.4



Gambar 3.4 *Flowchart* Aksi Karakter Musuh Anjing dan Harimau.

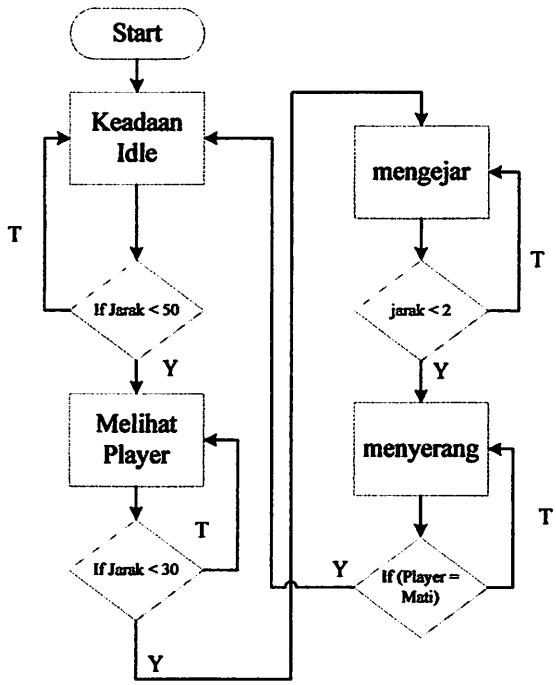
Alur metode *Finite State Machine* yang terdapat pada musuh terakhir yaitu *Boss Monster* pada level 4 ditunjukan pada gambar 3.5.



Gambar 3.5 Perancangan Aksi Karakter Boss

Pada gambar 3.5, metode ini ditanamkan pada musuh terakhir *(Boss)* dalam *game Survival from Island.* Aksi awal yang dilakukan musuh adalah *Idle*, jika jarak musuh dengan *player* lebih dari 50 *unit* maka musuh akan tetap diam di tempat awal, jika jarak dengan *player* kurang dari 50 *unit* maka musuh akan melihat ke *player* dan jika kurang dari 30 *unit* dari *player* maka musuh akan bergerak mengejar *player* dan jika jarak antara musuh dan *player* sudah lebih kecil 2 *unit*, maka musuh akan melakukan aksi penyerangan kepada *player.*

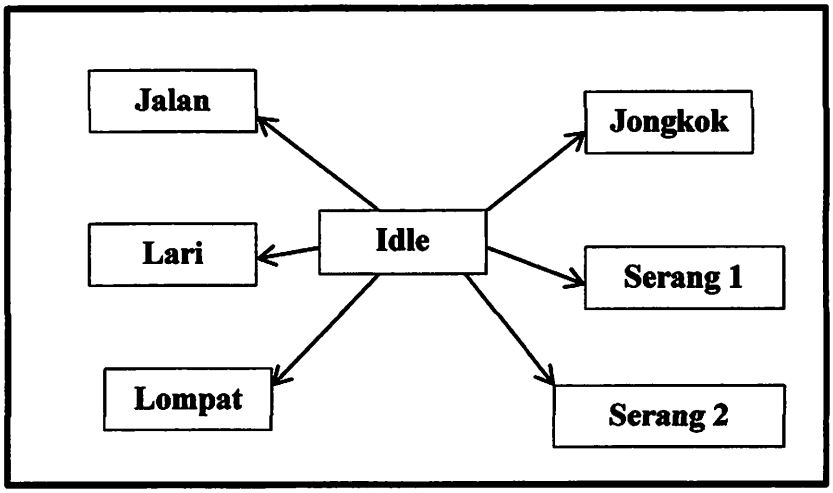
Alur aksi musuh troll dan bos musuh juga dijelaskan dengan *Block Diagram* dalam gambar 3.6



Gambar 3.6 *Flowchart* Aksi Karakter Boss

3.2.6 Perancangan Aksi Karakter

Perancangan Aksi karakter adalah perancangan reaksi tindakan karakter terhadap suatu kondisi tertentu, karakter pemain memiliki beberapa aksi yaitu *Idle*, berjalan, berlari, merunduk, melompat, dan menyerang. Perancangan aksi karakter ditunjukkan pada gambar 3.7



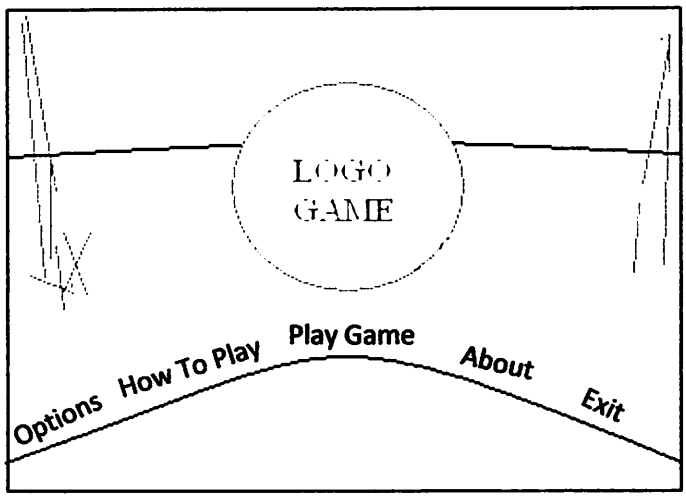
Gambar 3.7 Perancangan Aksi Karakter pada Player

3.2.7 Perancangan Antar Muka

Perancangan antar muka bertujuan untuk memberikan gambaran tentang aplikasi *game* yang akan dibangun.

1. **Perancangan Antar Muka Menu Utama**

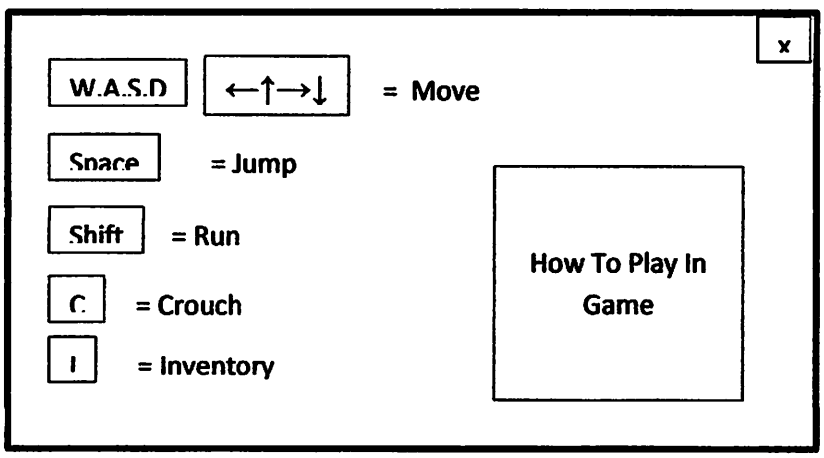
   Perancangan antar muka menu utama adalah perancangan tampilan menu utama dari *game Survival from Island* dimana pada menu utama terdapat tombol *Play Game*, *How to Play*, *Options*, *About* serta tombol *Exit*. Tombol *Play Game* berfungsi untuk memulai permainan, jika pemain memilih tombol mulai main maka pemain akan dihadapkan pada antar muka dari *level* 1. Tombol *How to Play* digunakan apabila pemain tidak mengetahui cara bermain dari *game Survival from Island*, tombol *About* akan mengarahkan pemain pada antar muka keterangan pembuat *game*. Kemudian tombol *Options* digunakan apabila ingin mengganti *setting* awal dari *game* sebelum memulai permainan dan selanjutnya adalah tombol *Exit* digunakan untuk keluar dari permainan. Rancangan antar muka menu utama dari *game Survival from Island* ditunjukkan dalam gambar 3.8



Gambar 3.8 Perancangan Antar Muka Menu Utama.

2. Perancangan Antar Muka *How to Play*

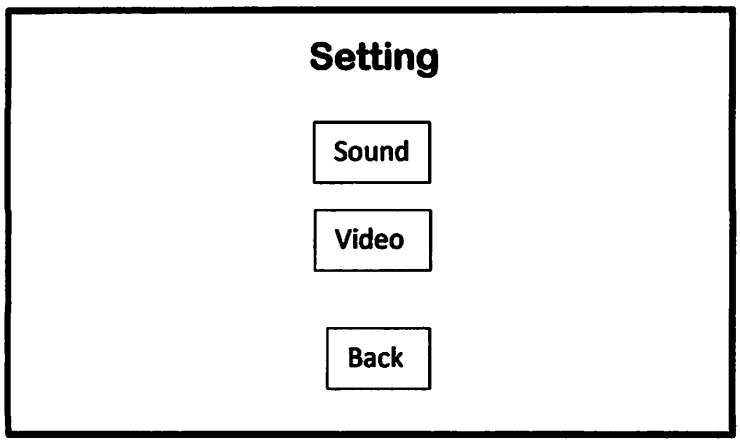
Perancangan antar muka *How to Play* adalah perancangan tampilan menu petunjuk dari *game Survival from Island* dimana pada menu *How to Play* terdapat petunjuk serta cara untuk memainkan *game*. Di antaranya adalah tombol W,A,S,D atau Arah pada *keyboard* untuk melakukan aksi bergerak sesuai arah yang diinginkan, tombol *space* untuk melakukan aksi melompat, tombol C pada *keyboard* untuk melakukan gerakan jongkok, tombol *Shift* pada *keyboard* untuk berlari, tombol I pada *keyboard* untuk mengecek *Inventory* / tas *player* dan juga terdapat gambar *Mouse* dimana klik kanan untuk menyerang. Rancangan antar muka petunjuk dari *game Survival from Island* ditunjukkan dalam gambar 3.9.



Gambar 3.9 Perancangan Antar Muka Menu *How to Play*

3. Perancangan Antar Muka *Options*

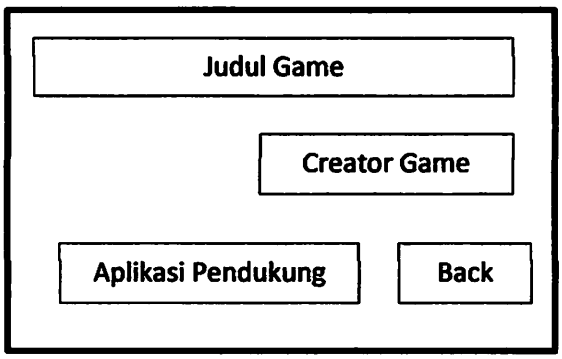
Perancangan Antar muka *Options* adalah perancangan tampilan menu pengaturan dari *game Survival from Island* dimana pada menu pengaturan terdapat pilihan pengaturan sebelum memainkan *game*. Pilihan menu pada pengaturan yaitu pengaturan *Sound* dan *Video* di dalam *game*. Terdapat juga tombol *BACK* yang berfungsi untuk kembali ke menu utama.



Gambar 3.10 Perancangan Antar Muka *Options*

4. Perancangan Antar Muka *About*

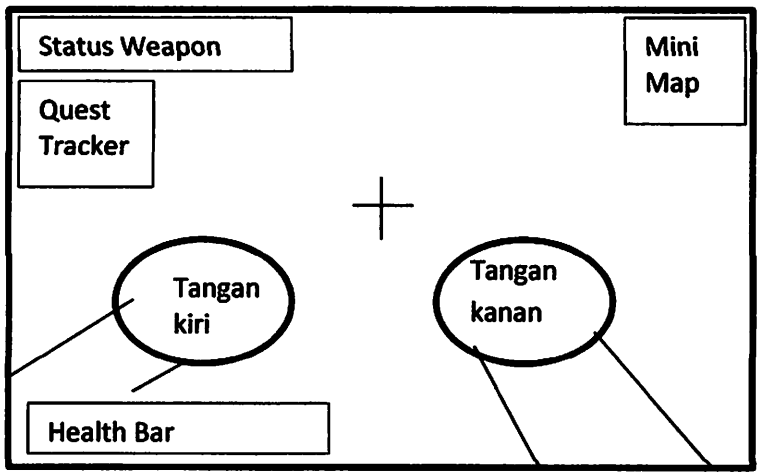
Perancangan Antar muka *About* adalah perancangan tampilan menu tentang *game Survival from Island* dimana pada antar muka ini berisi informasi seperty pembuat dari game dan software yang digunakan. Terdapat juga tombol *BACK* yang berfungsi untuk kembali ke menu utama.



Gambar 3.11 Perancangan Antar Muka *About*

5. Perancangan Antar Muka *Level* 1

Perancangan antar muka *level* satu adalah perancangan tampilan pada *level* satu dari *game Survival from Island* dimana pada *level* 1 pemain akan memulai permainan pada *environtment level* 1. Dalam antar muka *level* 1 akan terdapat *Health Bar* yang menunjukkan status darah atau kehidupan dari karakter *Player*. Pada gambar 3.12 terdapat *Health Bar*, kemudian *Quest Tracker* sebagai petunjuk *Player* untuk melakukan misi yang diperintahkan, *Status Weapon* yang menunjukan senjata yang sedang digunakan, kemudian *Mini Map* sebagai peta kecil untuk melihat sekitar *Player*. Pada *level* satu karakter dapat digerakkan oleh pemain untuk berjalan, berlari, melompat, merunduk dengan menekan tombol pada *keyboard* setra memukul dengan meng-*click* kanan *Mouse* seperti yang ditunjukkan dalam antar muka *How to Play*. Rancangan antar muka *level* 1 dari *game Survival from Island* ditunjukkan dalam gambar 3.12.

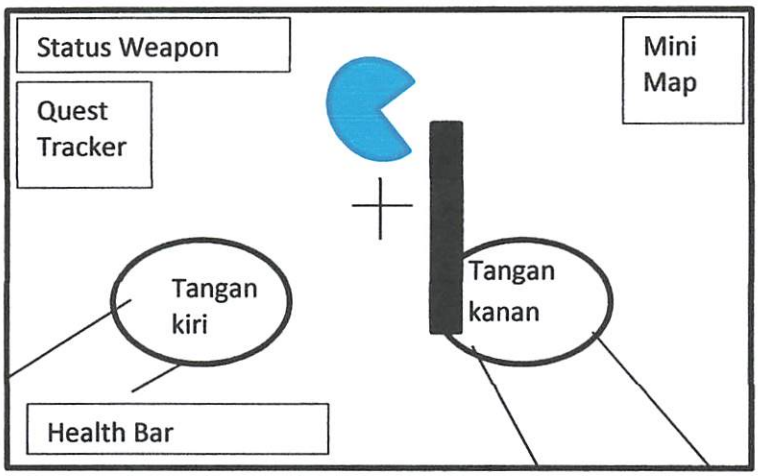


Gambar 3.12 Rancang Antar Muka *Level 1*

6. Perancangan Antar Muka *Level* 2

Perancangan antar muka *level* dua adalah perancangan tampilan pada *level* dua dari *game Survival from Island* dimana pada *level* 2 pemain akan memulai permainan pada *environtment level* 2. Dalam antar muka *level* 2 akan terdapat *Health Bar* yang menunjukkan status darah atau kehidupan dari karakter *Player*. Pada gambar 3.13 terdapat *Health Bar*, kemudian *Quest Tracker* sebagai petunjuk *Player* untuk melakukan misi yang diperintahkan, *Status Weapon* yang

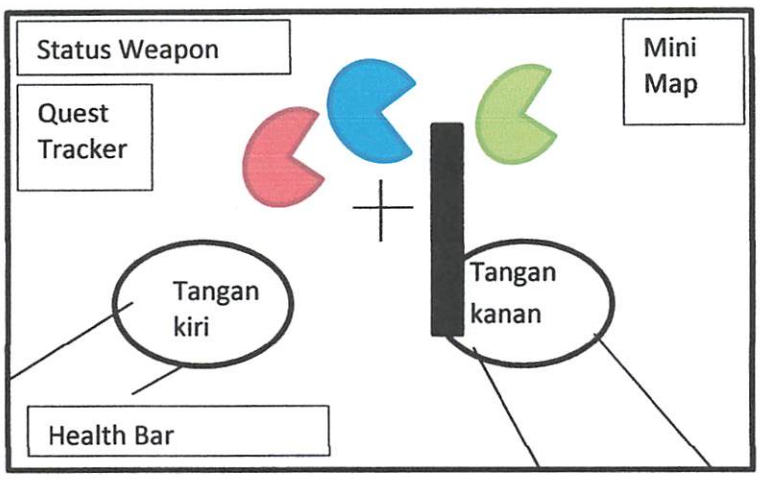
menunjukan senjata yang sedang digunakan, kemudian *Mini Map* sebagai peta kecil untuk melihat sekitar *Player*. Pada *level* dua karakter dapat digerakkan oleh pemain untuk berjalan, berlari, melompat, merunduk dengan menekan tombol pada *keyboard* setra memukul dengan meng-*click* kanan *Mouse* seperti yang ditunjukkan dalam antar muka *How to Play*. Rancangan antar muka *level* 2 dari *game Survival from Island* ditunjukkan dalam gambar 3.13.



Gambar 3.13 Rancang Antar Muka *Level 2*

7. Perancangan Antar Muka *Level* 3

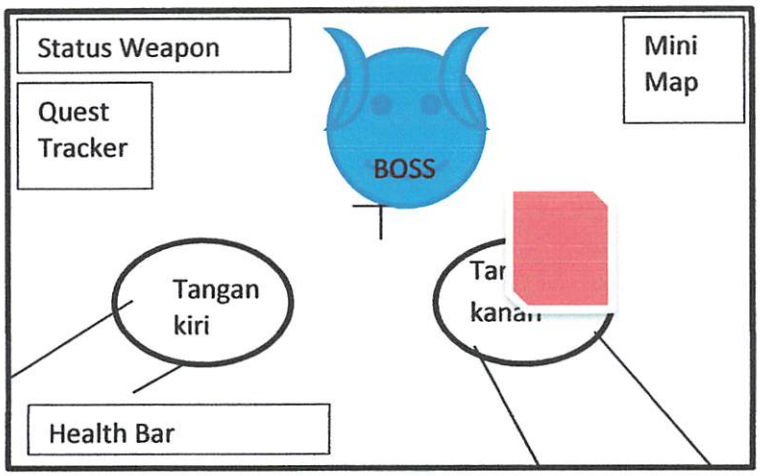
Perancangan antar muka *level* tiga adalah perancangan tampilan pada *level* tiga dari *game Survival from Island* dimana pada *level* 3 pemain akan memulai permainan pada *environtment level* 3. Dalam antar muka *level* 3 akan terdapat *Health Bar* yang menunjukkan status darah atau kehidupan dari karakter *Player*. Pada gambar 3.14 terdapat *Health Bar*, kemudian *Quest Tracker* sebagai petunjuk *Player* untuk melakukan misi yang diperintahkan, *Status Weapon* yang menunjukan senjata yang sedang digunakan, kemudian *Mini Map* sebagai peta kecil untuk melihat sekitar *Player*. Pada *level* tiga karakter dapat digerakkan oleh pemain untuk berjalan, berlari, melompat, merunduk dengan menekan tombol pada *keyboard* setra memukul dengan meng-*click* kanan *Mouse* seperti yang ditunjukkan dalam antar muka *How to Play*. Rancangan antar muka *level* 3 dari *game Survival from Island* ditunjukkan dalam gambar 3.14.



Gambar 3.14 Rancang Antar Muka *Level 3*

8. Perancangan Antar Muka *Level* 4

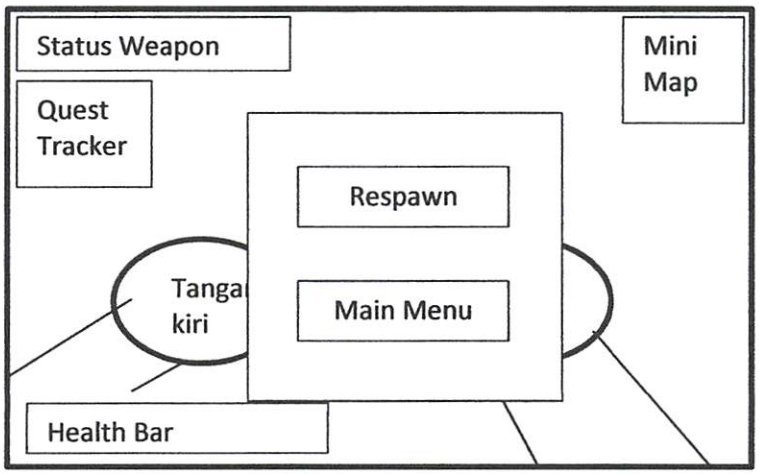
Perancangan antar muka *level* Empat adalah perancangan tampilan pada *level* empat dari *game Survival from Island* dimana pada *level* 4 pemain akan memulai permainan pada *environtment level* 4. Dalam antar muka *level* 4 akan terdapat *Health Bar* yang menunjukkan status darah atau kehidupan dari karakter *Player*. Pada gambar 3.15 terdapat *Health Bar*, kemudian *Quest Tracker* sebagai petunjuk *Player* untuk melakukan misi yang diperintahkan, *Status Weapon* yang menunjukan senjata yang sedang digunakan, kemudian *Mini Map* sebagai peta kecil untuk melihat sekitar *Player*. Pada *level* empat karakter dapat digerakkan oleh pemain untuk berjalan, berlari, melompat, merunduk dengan menekan tombol pada *keyboard* setra memukul dengan meng-*click* kanan *Mouse* seperti yang ditunjukkan dalam antar muka *How to Play*. Rancangan antar muka *level* 4 dari *game Survival from Island* ditunjukkan dalam gambar 3.15.



Gambar 3.15 Rancang Antar Muka *Level 4*

9. Perancangan Antar Muka *Respawn*

Perancangan Antar muka *Respawn* adalah perancangan tampilan saat kondisi *player* mati, yang kemudian muncul menu untuk melanjutkan permainan atau tidak. Jika pemain memilih *Respawn* maka karakter *Player* akan dihidupkan kembali ketempat semula, dan apabila tidak maka pemain akan dilemparkan ke menu utama. Rancangan antar muka *respawn* dari game *Survival from Island* ditunjukan pada gambar 3.16.



Gambar 3.16 Rancangan Antar Muka Respawn

3.2.8 Cheat Code

Cheat code merupakan kode rahasia yang berfungsi untuk memberikan suatu nilai property tertentu dalam sebuah *game* sehingga dapat mempengaruhi permainan. Fungsi dari setiap kode berbeda-beda, nilai yang dapat berubah ketika kode dimasukkan adalah *Health Point*, *Power*, dan *Level*. Cheat code dapat dimasukkan ketika memainkan game, kemudian menekan tombol *Escape* dan menekan tombol *Input Code*. Kode rahasia yang terdapat dalam *game Survival from Island* ditunjukkan pada tabel 3.4.

Tabel 3.4 *Cheat Code*

No.	Cheat Code	Fungsi
1.	*tooeasy*	Menuju *level* 2
2.	*luckyday*	Menuju *level* 3
3.	*callmeyourboss*	Menuju *level* 4
4.	*imtheironman*	*Health* menjadi Banyak
5.	*hulkmodeon*	*Power* dari serangan *Player* Meningkat
6.	*iwannabenormal*	Kembali ke kondisi normal

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Tujuan implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem sehingga *user* dapat memberi masukan demi berkembangnya sistem yang telah dibangun sebagai simulasi dari *game* Petualngan *Surviival from Island.* Perangkat keras dan perangkat lunak yang digunakan untuk impelementasi dan pengujian aplikasi yaitu sebagai berikut:

4.1.1 Implementasi Perangakat Keras

Perangakat keras yang akan digunakan untuk menjalankan aplikasi *game* Petualangan *Surviival from Island* ini tidak harus computer yang berspesifikasi tinggi. Hal ini disebabkan aplikasi *game* ini dapat diatur sesuai spesifikasi user agar dapat dimainkan. Sebagai ilustrasi sistem minimum yang dapat menjalanka aplikasi dapat dilihat pada table 4.1 di bawah ini.

Tabel 4.1 Implementasi Perangkat Keras

Processor	AMD Athlon neo MV-10 1.6Ghz
RAM	1GB
HDD	250GB
Resolusi Layar	1280 x 800
Keyboard	1
Mouse	1
Speaker	YA

4.1.2 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk menjalankan aplikasi *game Surviival from Island* ini adalah sebagai berikut:

1. Sistem Operasi: *Microsoft Windows XP, 7 dan 8.*
2. *Framenetwork 3.5*

4.1.3 Implementasi Aplikasi

Aplikasi *game Surviival from Island* ini merupakan sebuah *game* petualangan yang menceritakan seseorang yang terdampar di dalam sebuah pulau tak berpenghuni. Aplikasi ini tidak perlu dilakukan instalasi, cukup dengan melakukan *double click* pada aplikasi ini (*Executable*).

4.1.4 Implementasi Gerakan Animasi Objek 3D

Untuk menggerakkan atau menganimasikan Objek 3D dapat menggunakan aplikasi yang berbasis 3D seperti *Blender*. Menganimasikan objek disini bertujuan untuk membuat game tampak lebih hidup, dan terasa nyata.

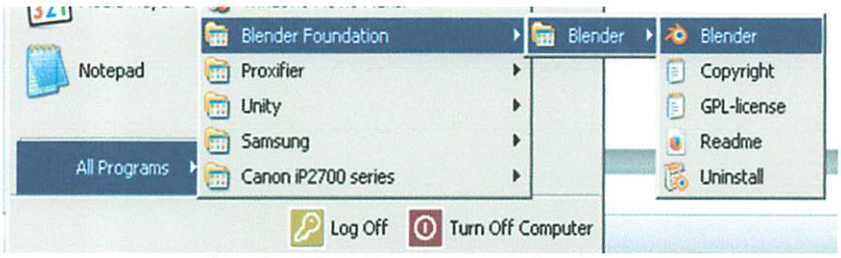
Dalam pembuatan animasi 3D terdapat beberapa tahap yang harus dilakukan. berikut beberapa tahap pembuatan animasi terhadap objek 3D :

1. Buka Program *Blender.*

   Untuk membuka program *Blender* terdapat beberapa cara. Cara pertama adalah membuka program melalui shortcut program yang terdapat pada desktop Windows pada komputer anda

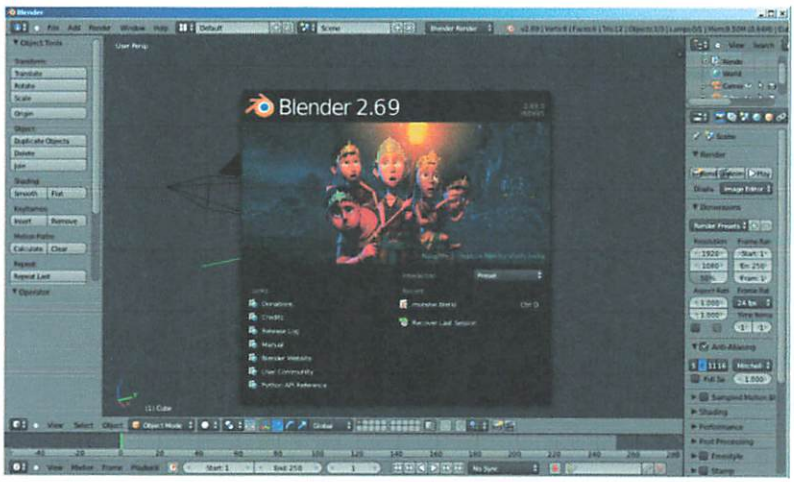
Gambar 4.1 *Shortcut Blender*

Cara kedua adalah membuka program *Blender* melalui menu *start* pada W*indows*. Untuk membuka program *Blender* memlaui *start* menu lakukan *click* pada tombol *start* pada *Windows* di pojok kiri layar, setelah itu akan muncul *Start* menu *Windows*, *click "All Program"* kemudian "*Blender Foundation*" dilanjutkan "*Blender*" dan *Click* icon *Blender*.



Gambar 4.2 *Start Menu Blender.*

Setelah *Short Cut* program *Blender* diklik, maka program akan terbuka. Tunggu beberapa saat sampai program berhasil dibuka dan menampilkan halaman depan dari *Blender*.



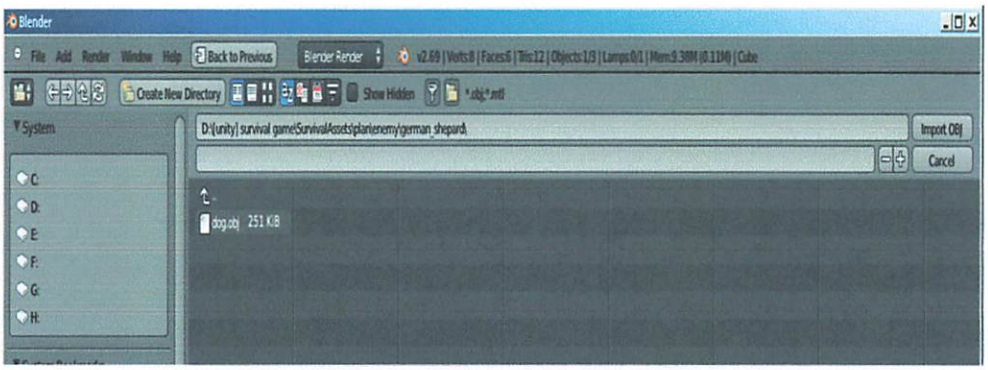
Gambar 4.3 Halaman Depan *Blender*

2. Meng-*Import* objek 3D ke aplikasi

Setelah Aplikasi blender terbuka selanjutnya adalah meng-*import* Objek 3D yang akan dianimasikan. dengan cara *Click "file"* kemudian pilih "*Import*" kemudian pilih objek 3D sesuai dengan Format Objek yang dimiliki.

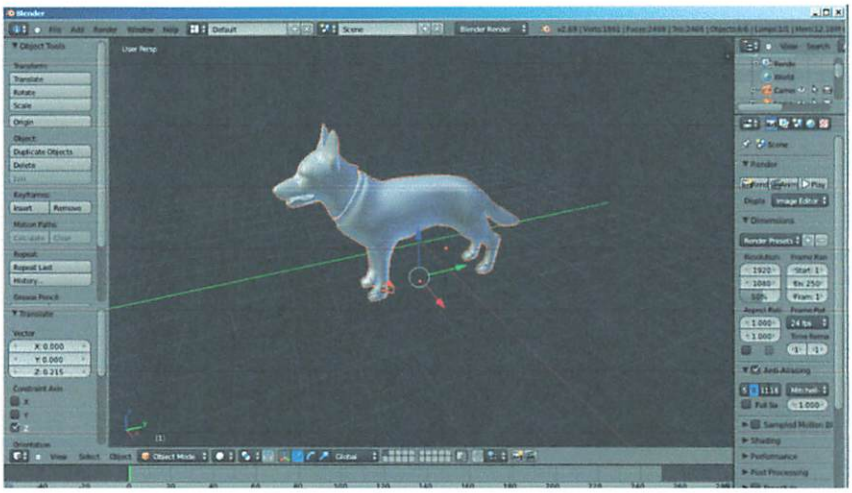
Gambar 4.4 Meng-*Import* File Objek 3D (i)

Kemudian akan muncul tampilan untuk memilih *file* objek 3D yang akan dipakai. dibagian *System* pilih *Drive* mana *file* objek tersimpan, setelah selesai menemukannya kemudian menekan tombol *Import*.



Gambar 4.5 Meng-*Import File* Objek 3D (ii)

Setelah berhasil meng-*Import file* objek / model 3D akan berada pada *WorkSpace Blender* dan siap untuk di animasikan. Seperti pada Gambar 4.6 berikut.

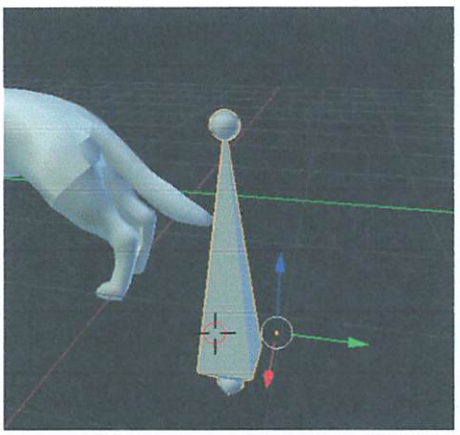


Gambar 4.6 Objek 3D pada *Workspace Blender*

3. Memberi *Bone* pada Objek 3D

Setelah berhasil meng-*import,* selanjutnya adalah tahap untuk memberikan *Bone* / tulang untuk Objek 3D tersebut. *Bone* disini digunakan untuk memberikan efek gerak terhadap objek 3D yang diam. Pertama Klik *"add"* kemudian *"armature"* dan kemudian *"single bone"* sehingga muncul sebuah objek yang mana digunakan sebagai *bone*.

Gambar 4.7 *Menu Add Single bone*



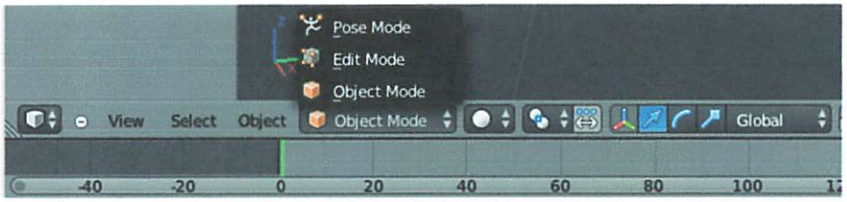
Gambar 4.8 Tampilan Objek *bone*

Setelah meng-klik *single bone* akan muncul objek yang digunakan sebagai inti *bone* (Tulang inti) dimana *bone* ini dapat diubah sesuai keinginan kita. dengan menggunakan beberapa *short-cut button* untuk mempermudah dan memanipulasi objek *bone,* seperti yang di tunjukan pada table 4.2 berikut

Tabel 4.2 *Shortcut button editing bone*

Tombol	Fungsi
E	Berfungsi untuk meng-*Extrude* atau menambahkan sendi *bone* sesuai target bone yang ditentukan
S	Mengatur Skala / ukuran *bone* sesuai yang dibutuhkan.
R	Mengubah posisi Rotasi sesuai dengan arah x,y,z
W	Membagi *bone* menjadi 2 bagian
A	Menyeleksi semua objek *bone*

perlu diperhatikan untuk meng-*edit bone* pada *workspace blender* harus berada pada keadaan "*edit mode*". Untuk mengubah keadaan tersebut dengan cara meng-klik tombol "*object mode*" kemudian pilih "*edit mode*".

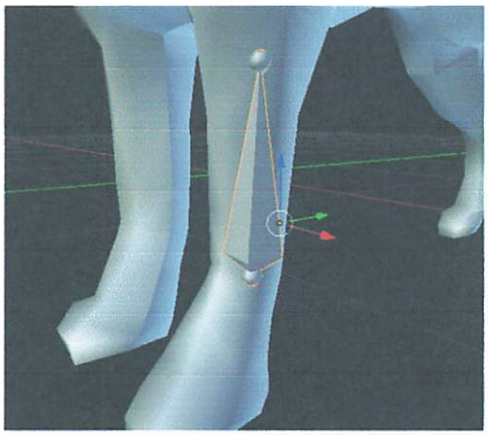


Gambar 4.9 *Toolbar Edit mode*

Selanjutnya untuk mempermudah mengatur pembentukan *bone* terhadap objek yang digunakan dapat menggunakan setting sebagai berikut. pertama-tama klik *icon "Armature"* yang berada pada kanan layar. dan pada *options* "*Display*" centang "*X-Ray*".

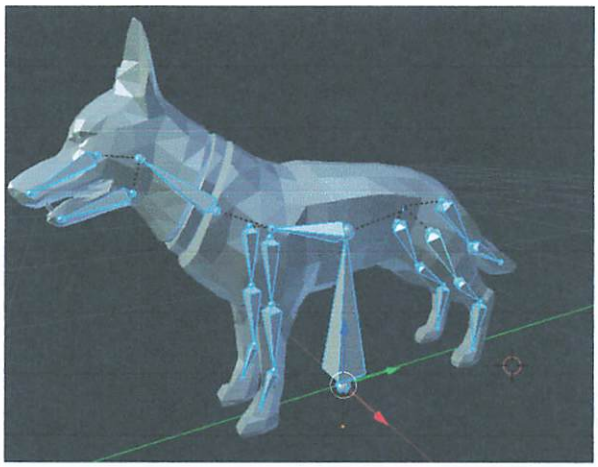
Gambar 4.10 *Toolbar Armature*

Setelah itu maka objek *bone* dengan objek 3D yang telah di-*import* apabila disatukan maka objek *bone* akan terlihat jelas sehingga memudahkan dalam meng-*edit bone* sesuai bentuk tubuh objek 3D.



Gambar 4.11 *Bone* dan Objek 3D Terlihat

Selanjutnya menggunakan tombol *shortcut* yang ada, kemudian edit *bone* sehingga tampak seperti gambar 4.12 berikut.



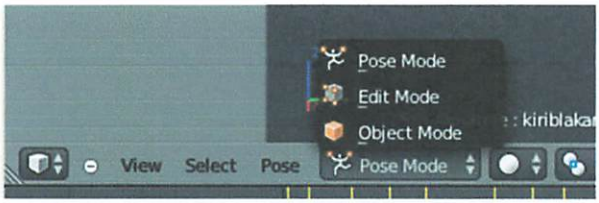
Gambar 4.12 Desain *Bone* Objek 3D

Perlu diperhatikan bahwa penggunaan *bone* hanya pada bagian yang akan digerakkan / dianimasikan saja agar menghemat penggunaan *memory* komputer yang berlebihan.

4. Menganimasikan Objek 3D

Setelah selesai dengan memasang *bone* pada objek 3D selanjutnya adalah proses animasi. proses ini adalah menggerakkan objek 3D sesuai dengan keinginan. Pertama-tama ubah keadaan

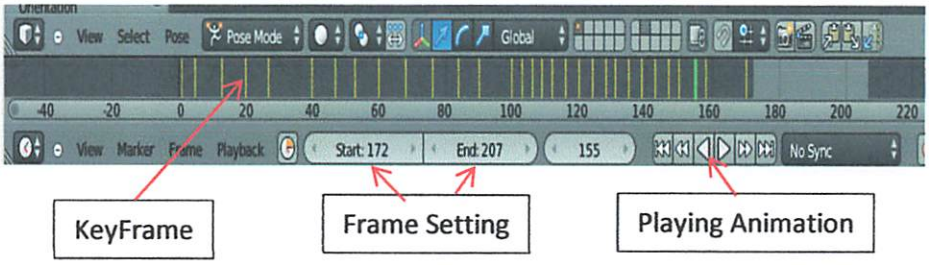
settingan mode menjadi "*Pose Mode*" dengan cara klik pada bagian "*Edit Mode*" dan klik "*Pose Mode*".



Gambar 4.13 *Setting Workspace "Pose Mode"*

Setelah menjadi *Pose mode, bone* yang digerakkan secara otomatis objek yang terdapat *bone* tersebut akan ikut bergerak mengikuti arah dari *bone* tersebut.

Untuk memulai membuat animasi, pertama mengenali *toolbar* untuk membuat animasi

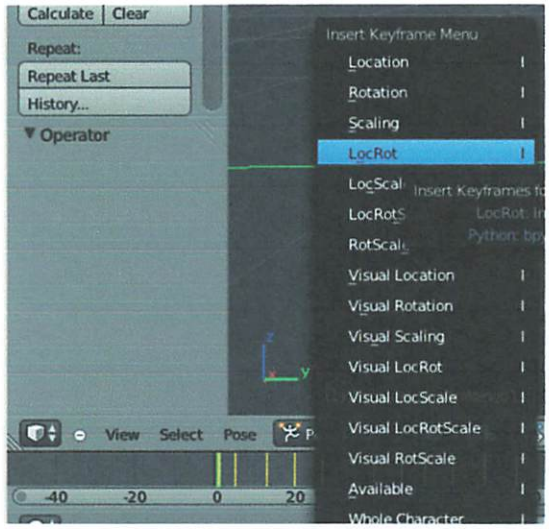


Gambar 4.14 *Toolbar Animation*

Pada Gambar 4.14 terdapat beberapa *tool,* pertama *Keyframe* digunakan sebagai perekam gerakan objek, jadi *keyframe* satu dengan yang lain akan saling membentuk gerakan sesuai dengan *pose* yang diberikan. kedua *Frame Setting* adalah untuk mengatur animasi *frame* yang digunakan. Dan tombol *Playing Animation* untuk melihat gerakan objek yang telah digerakkan.

Selanjutnya beberapa tahap dalam pembuatan gerakan animasi pada objek 3D. Pertama-tama membuat *keyframe* pada pose awal dengan cara meng-klik *Frame* keberapa animasi dimulai, pertama kita klik pada *frame* ke-0 untuk memulai kemudian tekan tombol "I" untuk membuat *keyframe*, dan sebelumnya *Cursor mouse* harus berada pada

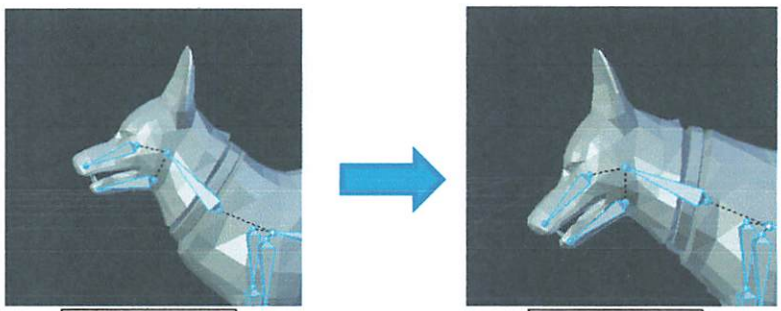
Workspace Blender sehingga muncul settingan untuk meng-*insert keyframe.*



Gambar 4.15 *Insert Keyframe*

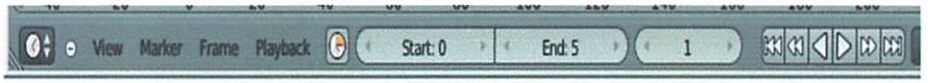
Saat meng-*insert keyframe* akan muncul beberapa pilihan seperti pada gambar 4.15. pilihan *insert* tersebut digunakan sesuai gerakan yang akan kita buat untuk objek 3D tersebut. karena penulis hanya menggerakkan animasi berdasarkan lokasi Rotasi maka *insert* yang dipilih adalah "*LocRot*" yang berarti Lokasi rotasi akan dimasukkan ke *frame* ke – 0.

Selanjutnya pada *frame* ke 5 gerakkan *bone* yang di inginkan untuk digerakkan.



Gambar 4.16 Perubahan Gerak antar *frame*

Dan selanjutnya ulangi proses awal dengan menekan tombol "I" lalu pilih "*LocRot*" dikarenakan hanya menggerakkan sesuai gerakan rotasi. Untuk melihat hasil animasi atur "*frame Setting*" dengan *Start = 0 dan End =5.* kemudian klik tombol *play.*



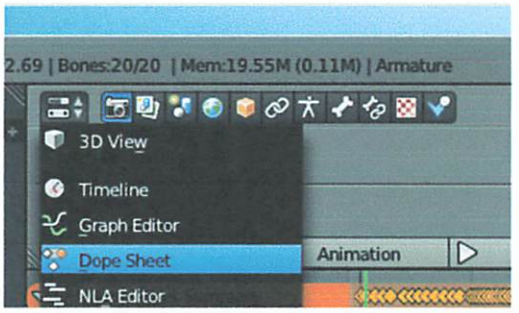
Gambar 4.17 *Setting Preview Animation*

Lakukan proses tersebut secara berulang-ulang untuk menghasilkan animasi yang sesuai dengan keinginan.

5. Menyimpan Objek Animasi

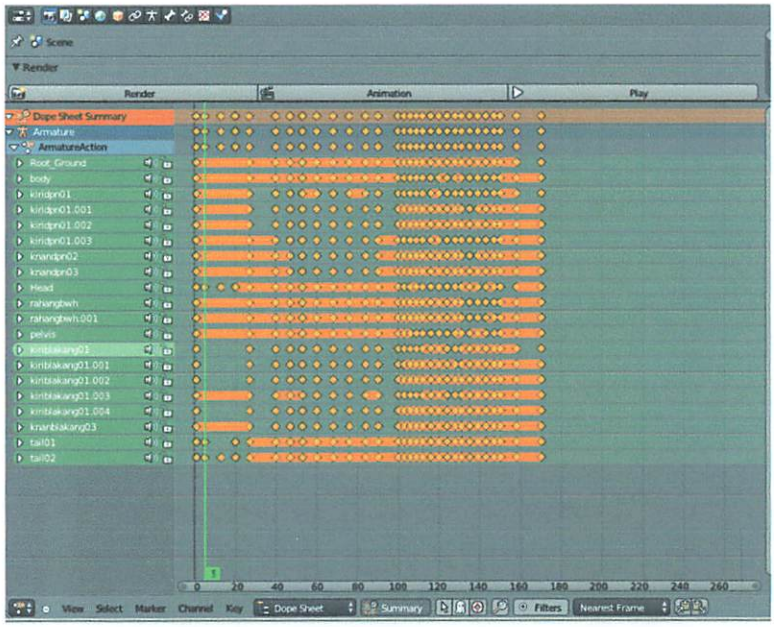
Setelah Animasi pada *Frame-frame* telah selesai selanjutnya adalah menyimpan file objek 3D yang telah dianimasikan agar dapat masuk kedalam *game* nantinya.

Pertama-tama menekan tombol *properties* yang ada disebelah kanan. dan mengubahnya menjadi mode "*Dope Sheet*".



Gambar 4.18 *Setting Dope Sheet*

Setelah menekan tombol tersebut. *Properties* akan tampak seperti gambar 4.19.



Gambar 4.19 Tampilan *Dope Sheet*

Di *Dope Sheet* sendiri animasi yang telah kita buat dapat diubah sesuai keinginan kita. seperti mengubah kecepatan dengan menabahkan panjang *frame* ataupun sebaliknya.

Setelah semua selesai selanjutnya adalah memberi nama Animasi yang telah dibuat dengan meng-klik *Dope Sheet* yang berada dibawah dan mengubahnya menjadi "*Action Editor*" dan pada kolom *Amature* dapat diganti sesuai keinginan sebagai nama dari animasi nantinya.

Gambar 4.20 *Setting Action Editor*

Setelah Memberi nama Animasi, tahap terakhir adalah menyimpan *file* Objek 3D yang telah dianimasikan. karena penulis menggunakan program aplikasi *Unity*, penulis menyarankan untuk menggunakan format *.fbx* agar mudah digunakan nanti. cara menyimpan dengan format *.fbx* adalah dengan menekan "*file*" kemudian "*Export*" kemudian pilih "*Autodesk FBX*". dan ganti nama *file* sesuai keinginan untuk meyimpannya.

Gambar 4.21 Menyimpan *file* Format *.fbx*

4.1.5 Implementasi *Stage / Area* di *Unity*

Untuk Membuat sebuah *Game* hal yang dibutuhkan pertama adalah sebuah *stage/area. Stage* merupakan sesuatu hal yang penting dimana *stage* adalah tempat dimana *Player* akan melakukan permainan, dan juga dapat digunakan berbagai hal yang lain, seperti digunakan untuk *Main Menu 3D, Credit* di *game* dan lain-lain.

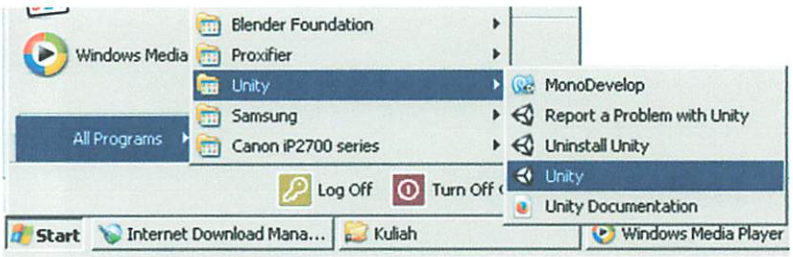
Dalam membuat *Stage* langkah-langkah yang harus dilakukan adalah sebagai berikut :

1. Buka Program *Unity*

   Untuk membuka program *Unity* terdapat 2 cara. Cara pertama adalah membuka program melalui shortcut program yang terdapat pada desktop Windows pada komputer anda

Gambar 4.22 *Shortcut Unity*

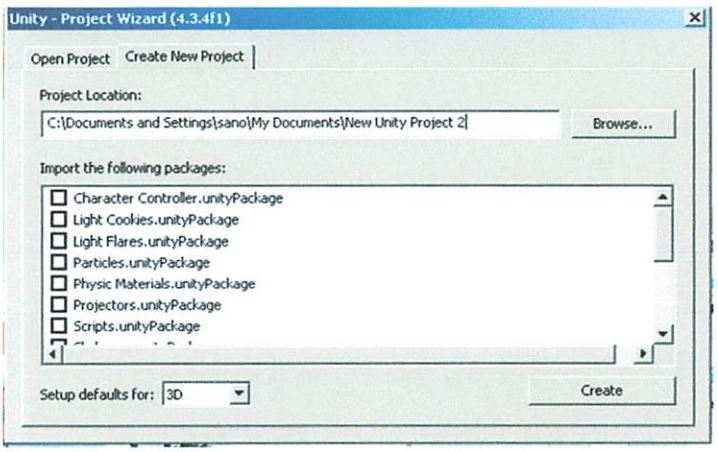
Cara kedua adalah membuka program *Unity* melalui menu *start* pada W*indows*. Untuk membuka program *Blender* memlaui *start* menu lakukan *click* pada tombol *start* pada *Windows* di pojok kiri layar, setelah itu akan muncul *Start* menu *Windows*, *click "All Program"* kemudian "*Unity*" dilanjutkan meng-*Click* icon *Unity*.



Gambar 4.23 *Menu Start Unity*

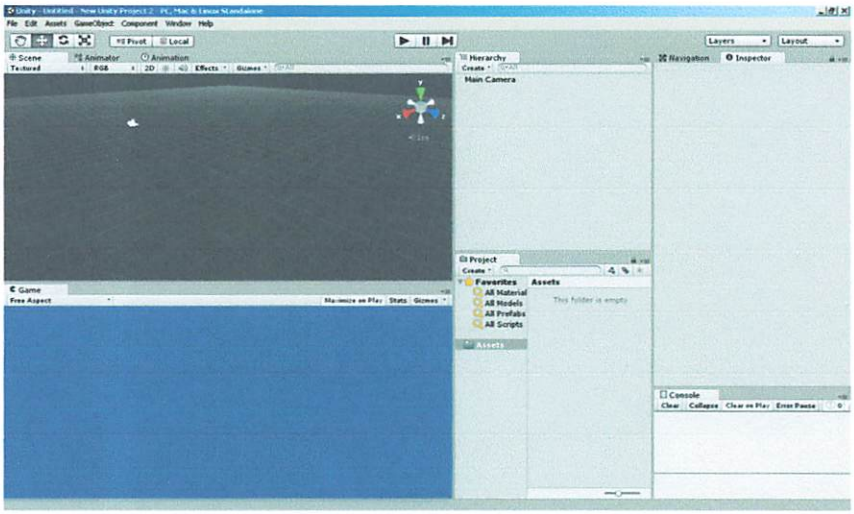
2. Membuat *Project* Baru

Setelah Membuka aplikasi *Unity* akan muncul *window New Project.* Di *window* ini terdapat tab *Open Project* dan juga *Create New Project*, jika sudah memilki project Unity yang sudah siap gunakan *Open Project,* dan apabila akan membuat baru / dari awal, gunakan tab *Create New Project.* Untuk bagian *Import Package* nya *uncheck* semua, atau bisa juga memilih *package* yang dibutuhkan nantinya, dan "*Setup Details*"nya gunakan "3D" . lalu selanjutnya klik tombol "*Create*".



Gambar 4.24 *Window Project Unity*

Setelah menekan tombol "*Create*" maka akan muncul tampilan awal dari aplikasi *Unity*.

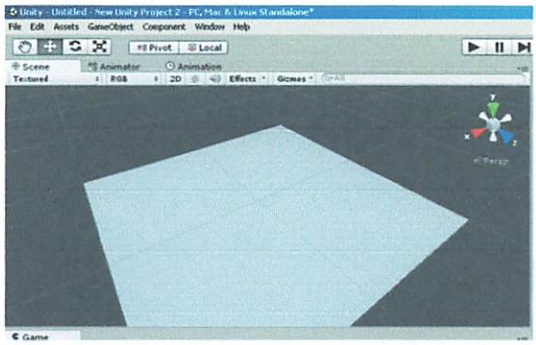


Gambar 4.25 *WorkSpace Unity*

3. Menyiapkan *Stage*

Selanjutnya setelah aplikasi *Unity* telah terbuka, hal selanjutnnya yang harus dilakukan untuk memulai membuat *game* adalah membuat *stage*.

Langkah pertama adalah membuat *Terrain*. *Terrain* disini adalah sebuah *Plate / area stage* yang dapat dimodifikasi secara menyeluruh. Untuk Membuat *Terrain*, klik menu *GameObject* > *Create Other* > *Terrain*. setelah itu akan muncul sebuah *Plate* di *workspace scene*.



Gambar 4.26 *WorkSpace scene*

Selanjutnya adalah mengatur Ukuran *Stage*. dengan cara meng-klik "*Terrain*" yang berada pada kolom "*Hierarchy*". kemudian pada kolom "*inspector*" > "*Terrain(Script)*", klik tombol berbentuk "*Gear*" yang kemudian akan muncul *setting* untuk *stage*. Lalu lihat settingan *Resolution,* di sana aka nada beberapa settingan untuk mengatur *width, height,* dan *length.* Resolusi dapat diatur sesuai keinginan. disarankan untuk menggunakan settingan *Terrain Width = 500, Terrain Length = 500, Terrain Height = 600.*

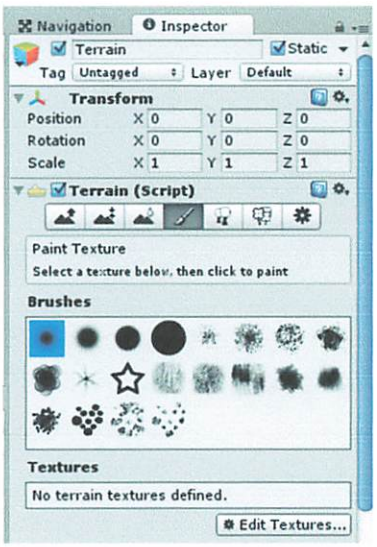
Gambar 4.27 *Setting Terrain*

4. Mengedit *Stage*

Setelah menyetting *terrain,* selanjutnya adalah mengedit *terrain* sehingga tampak bagus dan indah. Pertama-tama adalah memberikan *Texture* ke *plate Terrain* sehingga tampak nyata. sebelumnya siapkan gambar *texture*, contohnya seperti gambar pasir, bebatuan, ataupun lapangan rumput yang memiliki resolusi yang sama antara *width* dengan *Height* . untuk memasukkan *texture* hanya dengan men-*Drag* gambar ke tab *Project.*

Gambar 4.28 Contoh *Texture* Rumput Lapangan

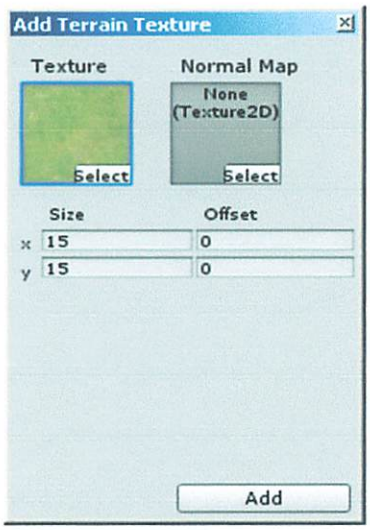
Selanjutnya adalah memasukkan *texture* kedalam *terrain* dengan cara : masuk ke tab *inspector* lalu klik tombol berbentuk kuas *(Paint Texture)* di "*Terrain (Script)*".



Gambar 4.29 *Setting Inspector Paint Texture*

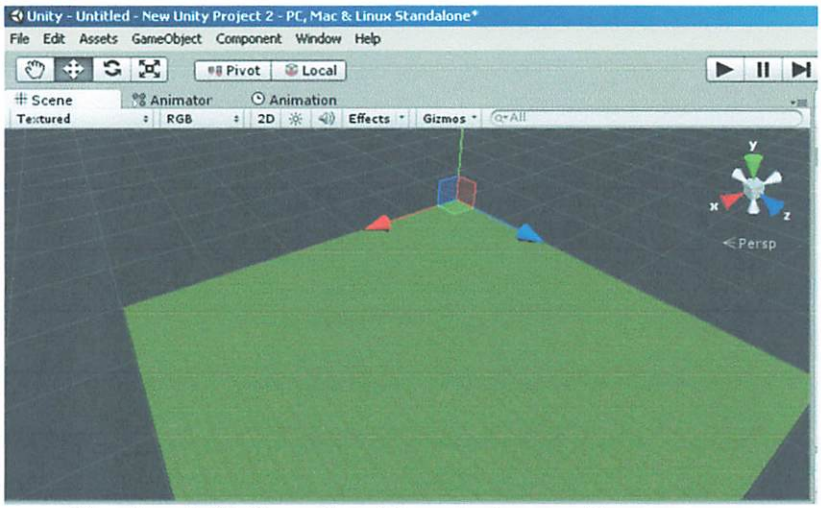
Kemudian klik "*Edit Textures...*" untuk memilih *texture* yang akan digunakan. sebagai contoh gunakan *texture* rumput lapangan. dan setelah mengklik tombol "*Edit Textures...*" akan muncul *window* baru

untuk memilih *texture.* untuk size bisa diatur sesuai keinginan. untuk *default* gunakan size X = 15 dan Y =15. lalu klik *add.*



Gambar 4.30 *Setting Add Terrain Texture*

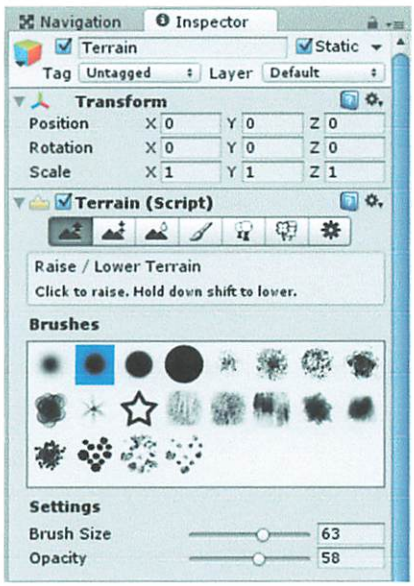
Setelah menekan tombol *add Terrain* secara otomatis akan berubah warna sesuai dengan *texture* yang telah dimasukkan.



Gambar 4.31 Tampilan *Terrain* dengan *Texture*

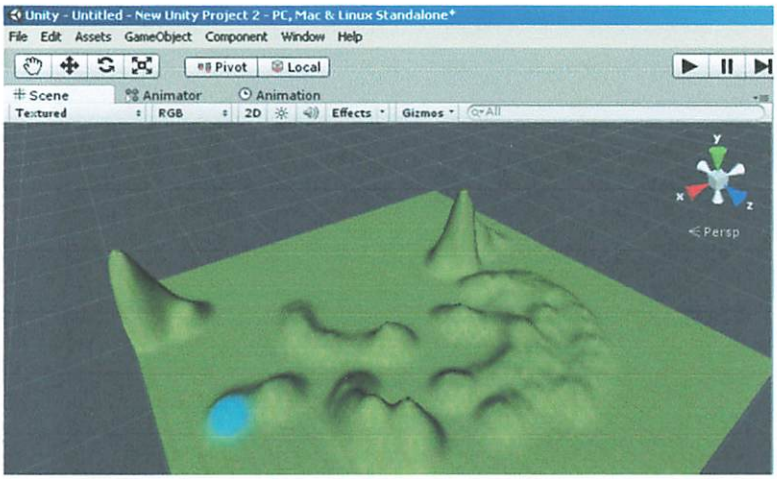
Selanjutnya adalah Mengedit *terrain* dengan mengatur ketinggian dari *plate terrain* agar terlihat seperti adanya lekukan tanah. caranya: pertama-tama ke tab *Inspector,* kemudian pada *Terrain*

(Script) tekan tombol berbentuk gunung "*Raise/Lower Terrain*". kemudian pilih *Brush* sesuai keinginan dan ubah setting ukuran *brush* dan *Opacity* dari *brush*.



Gambar 4.32 *Setting Rise/Lower Terrain*

Setelah itu klik pada *workspace Scene* untuk mengubah bentuk dari *terrain* sehingga tampak seperti tumpukan tanah ataupun gunung yang menonjol.

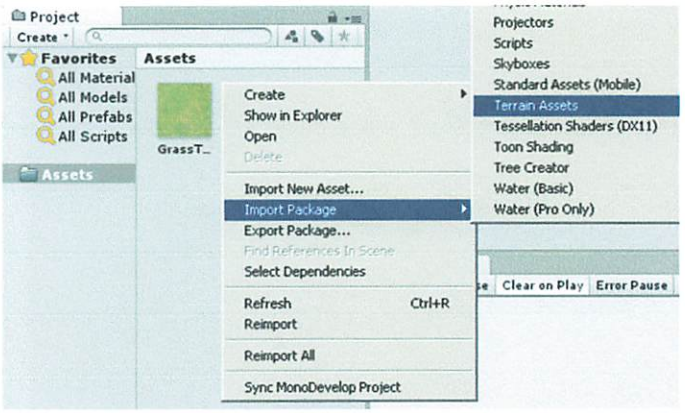


Gambar 4.33 *Contoh Rise/Lower Terrain*

5. Menambahkan Objek di *terrain*

Setelah cara untuk mengubah bentuk dari *Terrain* selanjutnya adalah untuk menambahkan beberapa object di terrain agar tampat nyata, seperti menambahkan Objek pohon, rumput dan lain-lain.

Untuk menambahkan Objek-objek tersebut pertama yang dilakukan adalah meng-*import package default* yang telah disediakan oleh *Unity.* caranya adalah klik kanan pada tab *project,* kemudian pilih "*Import Package*" dan pilih "*Terrain Asset*".

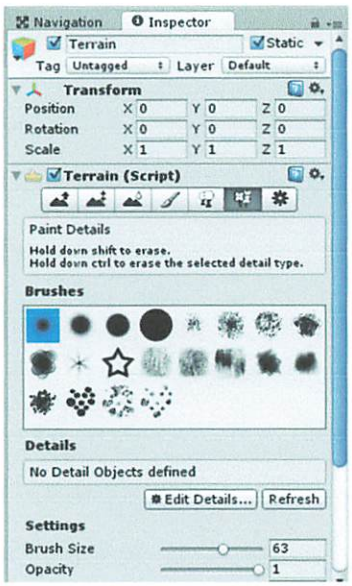


Gambar 4.34 *Import Package Terrain*

Setelah itu akan muncul *window* baru kemudian klik "*Import*". setelah klik *import*, tunggu sampai proses selesai, kemudian akan muncul sebuah folder "*Standart Assets*" yang didalamnya terdapat "*Terrain Assets*" yang berisi berbagai macam model 3d, seperi pohon, rumput dan lain-lain.

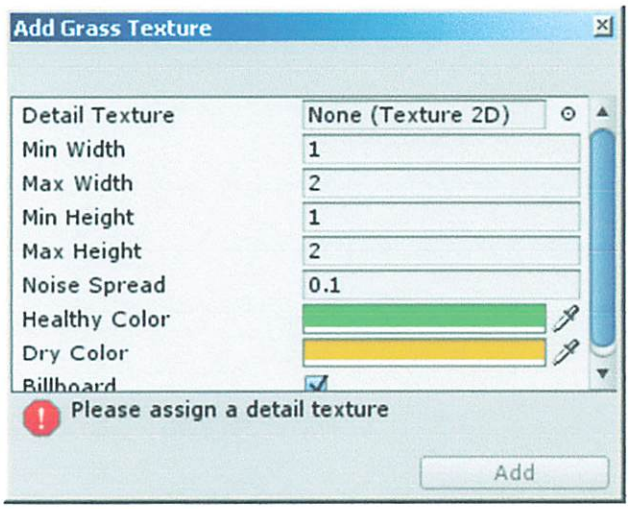
Gambar 4.35 *window Import Package*

Selanjutnya adalah menambahkan objek seperti rumput, pohon dan lain-lain, dengan cara klik *"terrain"* pada tab *"hierarchy"* kemudian pada tab *"Inspector"* > *"terrain (Script)"* pilih tombol *"Paint Details"* kemudian klik tombol *"edit Details"*.



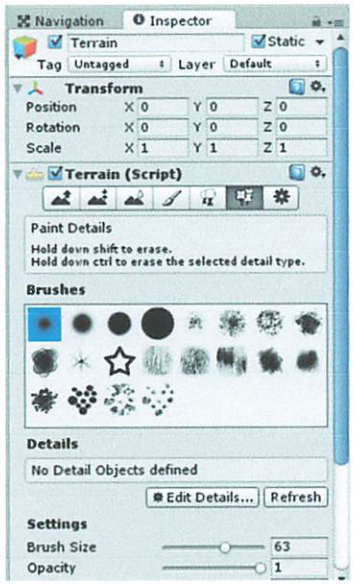
Gambar 4.36 *Setting Paint Details*

Setelah Klik *"Edit Details"* selanjutnya pilih *"Add Grass Texture"* maka akan muncul *window* baru untuk memasukkan objek rumput.



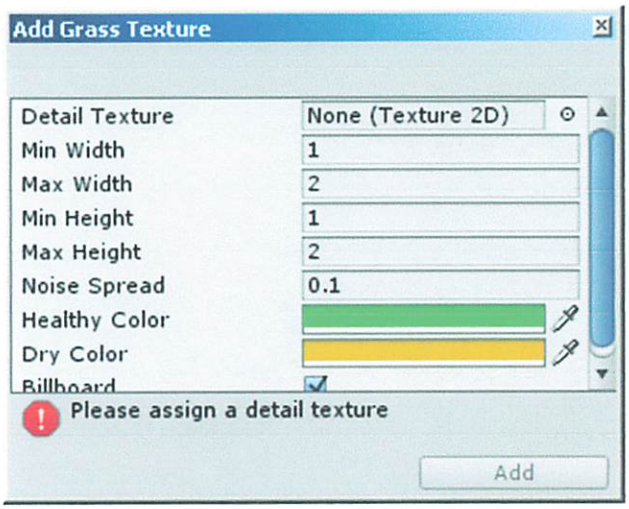
Gambar 4.37 *Add Grass Texture Window*

Selanjutnya adalah menambahkan objek seperti rumput, pohon dan lain-lain, dengan cara klik *"terrain"* pada tab *"hierarchy"* kemudian pada tab *"Inspector"* > *"terrain (Script)"* pilih tombol *"Paint Details"* kemudian klik tombol *"edit Details"*.



Gambar 4.36 *Setting Paint Details*

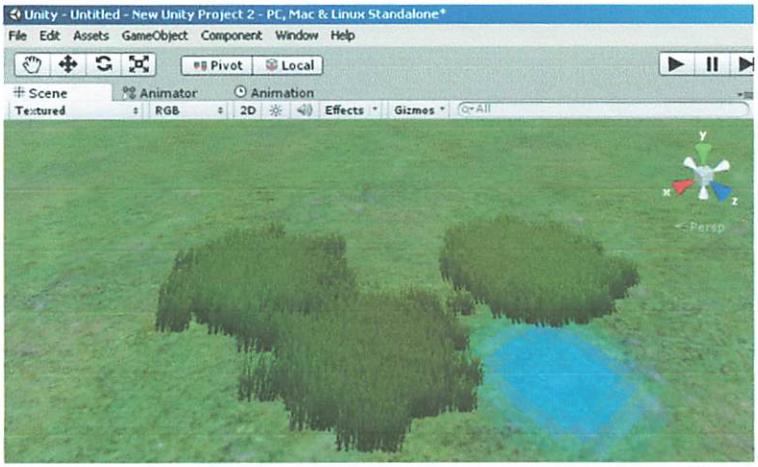
Setelah Klik *"Edit Details"* selanjutnya pilih *"Add Grass Texture"* maka akan muncul *window* baru untuk memasukkan objek rumput.



Gambar 4.37 *Add Grass Texture Window*

Pada *detail texture* klik lingkaran kecil sebelah kanan kemudian akan muncul lagi *window* baru kemudian pilih texture rumput sesuai keinginan. untuk settingan lainnya gunakan settingan *default.* kemudian klik "*add*" untuk menambahkan *texture.*

Selanjutnya untuk menambahkan objek rumput tersebut hanya dengan meng-klik pada terrain yang berada di *workspace Scene*.

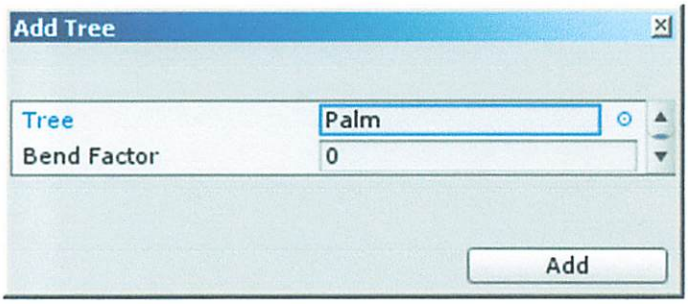


Gambar 4.38 Contoh menambahkan rumput pada stage

Untuk Menambahkan objek pohon menggunakan cara yang hampir sama, pada tab "*inspector*" > "*Terrain (script)*" pilih tombol "*Place trees*" kemudian pada option "*Tree*" klik tombol "*Edit Trees*" kemudian "*add tree*".

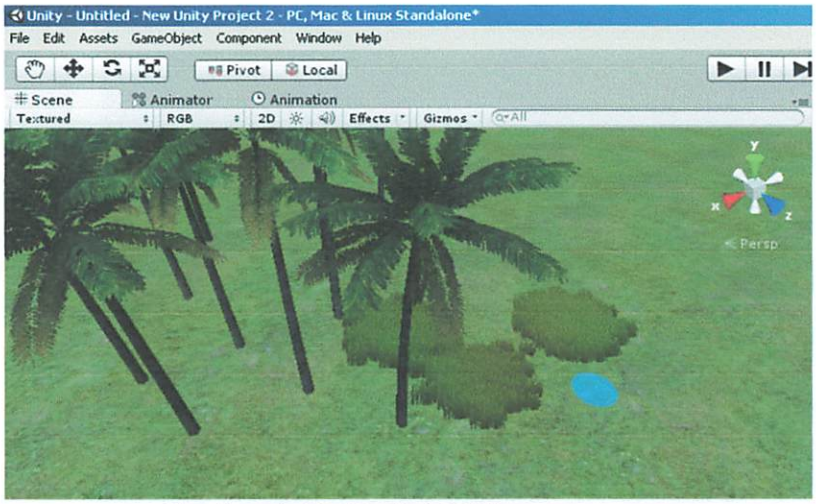
Gambar 4.39 *Setting Edit Trees*

Setelah meng-klik "*add tree*" akan muncul *window* baru untuk memilih model tree yang diinginkan.



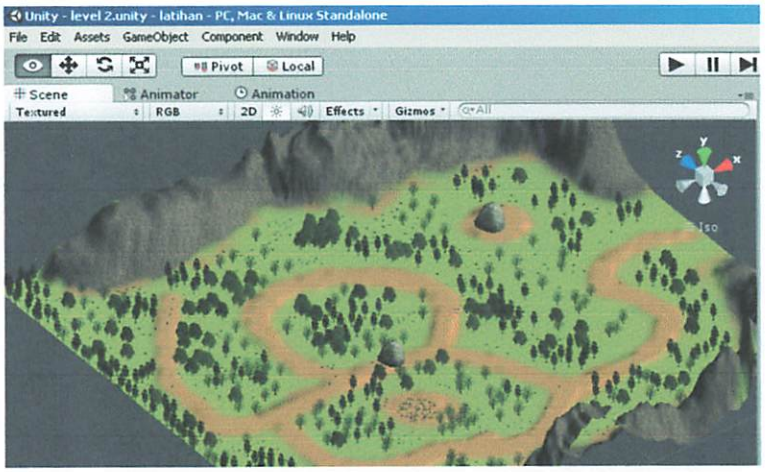
Gambar 4.40 *Window Add Tree*

Kemudian klik tombol "*add*" setelah itu setting *Tree* yang ada di *inspector* untuk mengatur jumlah pohon yang akan dimasukkan ke stage. kemudian klik pada terrain untuk memunculkan pohon tersebut.



Gambar 4.41 *Hasil Add Tree*

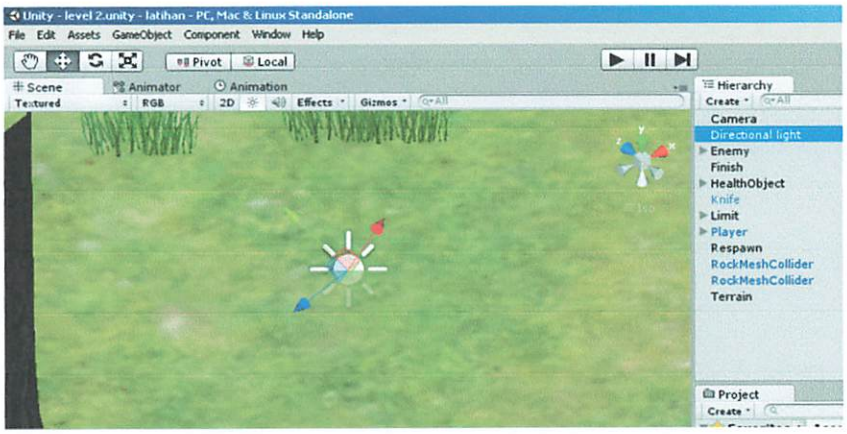
Selanjutnya adalah mendesain tata letak dari objek dari *terrain* sehingga menjadi tampak nyata dan indah nantinya di dalam *game*. seperti desain Gambar 4.42.



Gambar 4.42 *Desain Terrain*

Saat Mengeksekusi Permainan pada *Terrain,* Arena akan tampak gelap. maka dari itu dibutuhkan *Lighting* (Pencahayaan) agar tampak terang pada saat memainkan *game.*

Langkah pertama adalah meng-klik "*GameObject*" > "*Create Other*" > "*Directional Light*" maka akan muncul sebuah objek yang berfungsi sebagai pencahayaan di dalam area. Sudut dan arah cahaya bisa diubah sesuai keinginan.



Gambar 4.43 *Directional Light*

4.1.6 Implementasi *Main Menu*

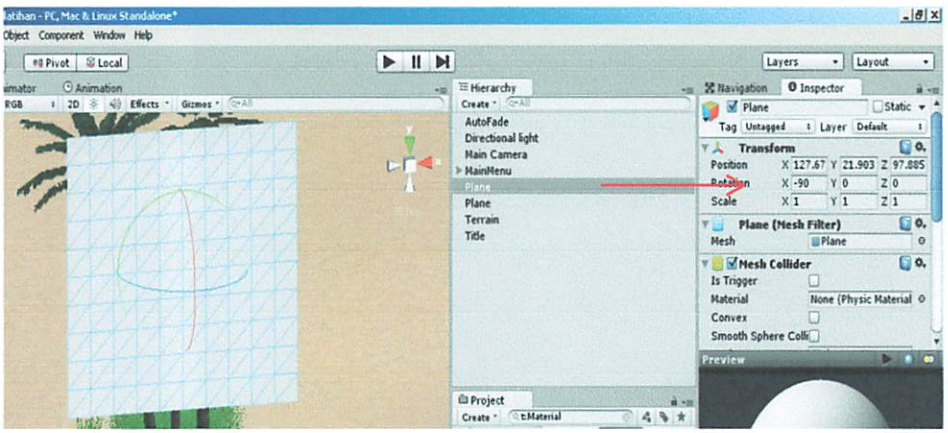
Main Menu merupakan Tampilan awal dari *game Survival from Island* yang berisi tombol-tombol untuk memulai permainan, petunjuk, setting dan lain-lain. Desain main menu dibuat sesederhana mungkin agar pemain dapat dengan mudah mengerti permainan tersebut.

Berikut adalah langkah-langkah pembuatan main menu 3D dalam game *Survival From Island* :

1. Memasukkan Logo *Game* di *Main Menu*

   Setelah *Terrain* telah disiapkan untuk *Main Menu* selanjutnya adalah menambahkan logo *game* di dalam *game.* sebelumnya setting *Main Camera* pada arah yang akan disorot oleh kamera pada saat memilih main menu nanti.

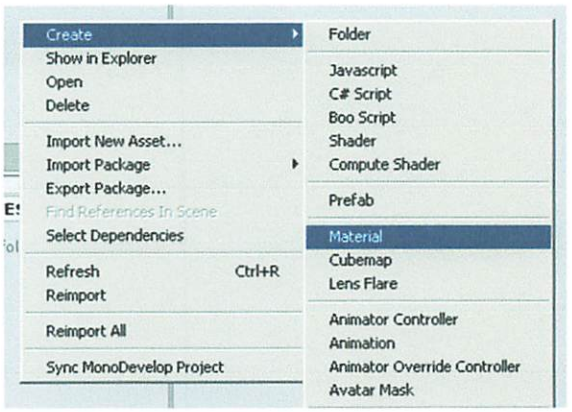
   Langkah Pertama adalah membuat objek *plane*, dengan cara meng-klik "*GameObject*" > "*Create Other*" > "*Plane*". maka akan muncul objek berupa persegi berwarna putih di *stage*. kemudian ubah rotasi sumbu X menjadi -90.



Gambar 4.44 objek *plane*

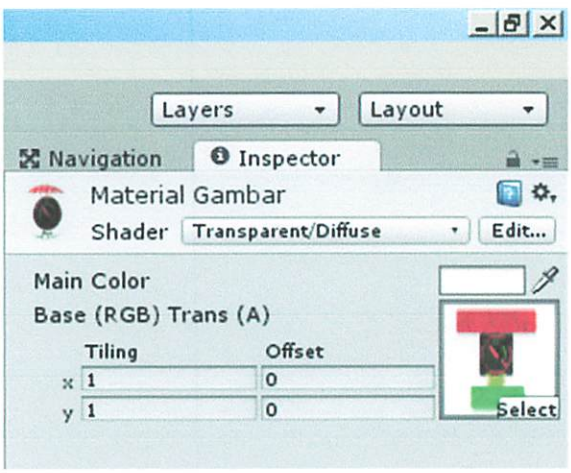
Selanjutnya Menyiapkan logo untuk game. bisa di kreasikan sesuai kebutuhan, dan gambar harus berformat PNG atau GIF karena gambar dengan format tersebut dapat menggunakan *background transparent.* Kemudian membuat sebuah material sebagai *Element* dari

plane. dengan cara klik kanan pada tab *Assets* kemudian pilih "*Create*" kemudian "*Material*".



Gambar 4.45 Membuat *Material*

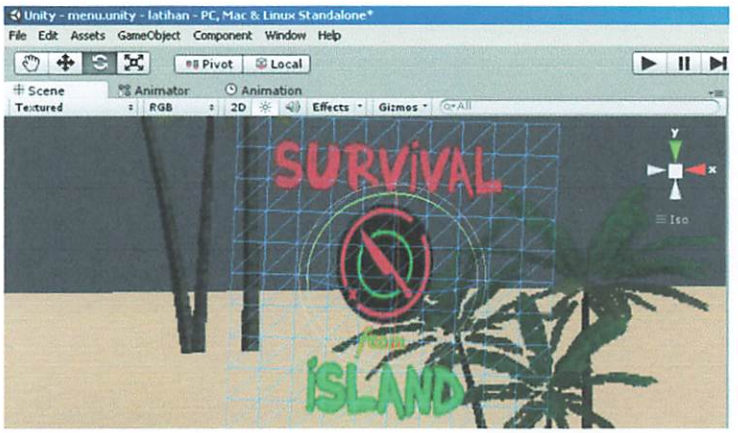
Maka Akan muncul sebuah *Material* di dalam tab *Asset*. selanjutnya men-setting *material*. setelah meng-klik *Material* pada tab *Inspector* selanjutnya pada "*Shader*" ubah menjadi "*Transparant /Diffuse*". dan pada kolom *image. Drag* gambar yang digunakan sebagai logo.



Gambar 4.46 *Settingan Material*

Setelah settingan *material* selesai. *Drag material* yang sudah di setting tadi. ke dalam *plane* pada *stage*, maka secara otomatis *plane* akan berubah menjadi logo. setelah dimasukkan, gambar akan terlihat

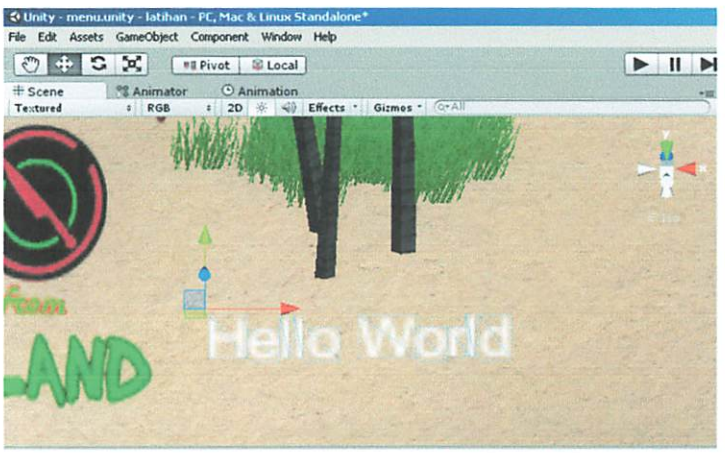
terbalik. maka dari itu ubah settingan Rotasi menjadi X = 90 dan Y = 180. sehingga akan tampak seperti gambar 4.47 berikut.



Gambar 4.47 *Settingan Material*

2. Membuat *Text Button*

Selanjutnya untuk membuat button 3D text adalah dengan cara meng-klik *"GameObject"* > *"Create Other"* > *"3D Text"*.maka secara otomatis akan muncul tulisan "*Hello World*" pada *stage.*



Gambar 4.48 *Text 3D Hello World*

Kemudian Untuk Mengubah Tulisan, *Font*, Ukuran dan lain-lain. setting di tab *"Inspector"* > *"Text Mesh"*. untuk mengubah teks, ganti kolom *"Text"* yang berisi *"Hello world"*, untuk mengubah

ukuran berada pada "*Font size*", mengubah *font*, pada kolom "*Font*" namun sebelumnya harus meng-*import* font yang akan digunakan, dan "*Color*" untuk mengubah warna teks.

Gambar 4.49 *Setting Text 3D*

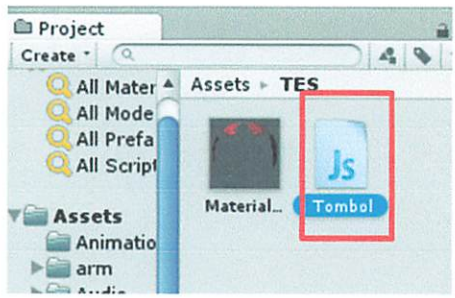
Selanjutnya agar Tulisan Tersebut dapat di-klik, harus dibuat objek Collider didalam teks tersebut, dengan cara "*add Component*" pada tab "*Inspector*" Kemudian pilih "*Physic*" lalu "*Box Collider*". maka secara otomatis akan muncul garis hijau mengelilingi pada teks.

Gambar 4.50 *Box Collider* pada teks

3. Menambahkan *Script*

Setelah Tombol *text 3D* telah selesai selanjutnya adalah proses *Scripting*. yang memiliki beberapa fungsi seperti membuka *GUI*

options, ataupun memulai permainan. langkah pertama yang dilakukan untuk melakukan *Scripting* adalah membuat *assets* berupa *JavaScript* dengan cara, klik kanan pada tab *assets* kemudian pilih "*Create*" kemudian " *JavaScript*". maka pada tab asset akan muncul sebuah *file* "*JavaScript*" ubah nama *file* tersebut sesuai keinginan.



Gambar 4.51 *File JavaScript*

Selanjutnya klik dua kali *file* tersebut sehingga muncul Editor *MonoDevelop.* kemudian masukkan beberapa *script* berikut:

```
10 function OnMouseUp(){
11 audio.PlayOneShot(beep);
12 yield new WaitForSeconds(0.35);
13
14 if(QuitButton){
15 Application.Quit();
16 }
17 else{
18 Howtogame = true;
19 }
20 }
```

Gambar 4. 52 *Script Click Button*

Script di atas adalah fungsi yang digunakan pada saat *player* meng-Klik *Button* yang ada.. Pada *script* tersebut yang di kotak merah adalah keadaan dimana akan memanggil fungsi *GUI.* Isi dari fungsi *GUI* dapat dimodifikasi sesuai dengan keinginan.

```
function OnGUI () {
//  GUI.skin.box = myskin;
    if (Howtogame == true)
        GUI.Box (Rect (Screen.width*0.5-260,Screen.height*0.5-260,520,520), "How To Play");
        GUI.DrawTexture(Rect( Screen.width*0.5-250,Screen.height*0.5-250,500,500),tutorial);
        if(GUI.Button(Rect(Screen.width*0.5+205,Screen.height*0.5-255,50,25), "close"))
        {
            Howtogame = false;
        }
    }
```

Gambar 4.53 *Script Call GUI*

Script dapat diubah sesuai fungsi yang dibutuhkan. apabila ingin memanggil *Stage* permainan setelah menekan tombol, contohnya saja "*Level 1*" maka hanya mengubah *Script* menjadi seperti Gambar 4.54 berikut.

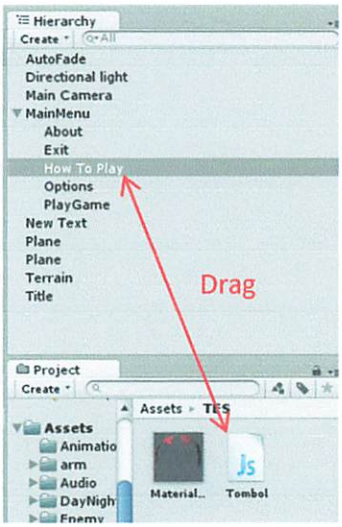
```
function OnMouseUp(){
audio.PlayOneShot(beep);
yield new WaitForSeconds(0.35);

if(QuitButton){
Application.Quit();
}
else{
Application.LoadLevel("Level 1");
}
}
```

Gambar 4.54 *Script* Memanggil Stage Baru

Pada *Script* di atas pada bagian "*Level 1*" adalah nama dari *Scene* yang telah dibuat. jadi setelah menekan tombol dengan *script* tersebut makan akan meng-*load Scene* "*Level 1*".

Setelah *Script* Selesai *save script* tersebut. lalu kembali ke aplikasi *Unity.* pada tab "*Hierarchy*" klik nama dari *Text* yang sebelumnya telah dibuat. lalu *Drag Script* yang telah dibuat kedalam text tersebut. sehingga tombol tersebut dapat berfungsi.



Gambar 4.55 *Drag Script* ke *Button Text*

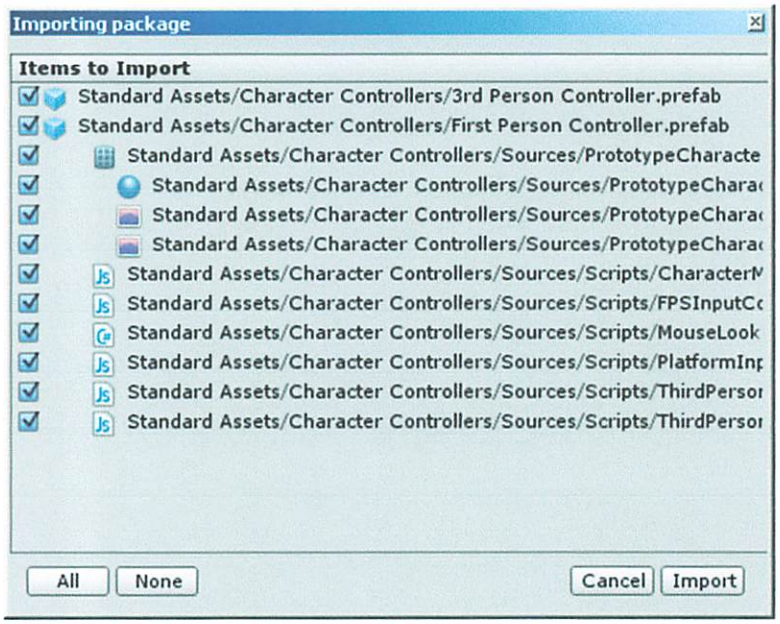
4.1.7 Implementasi *Menu Play Game*

Play Game merupakan sistem yang dipergunakan oleh pengguna agar dapat memulai permainan. Di saat pengguna menggunakan atau memilih menu untuk permainan, sistem akan melakukan proses yang terdapat pada *Scene* yg dituju.

Saat memulai permainan, dibutuhkan sebuah objek yang digunakan sebagai *player* yang akan dikendalikan oleh pemain. berikut adalah langkah-langkah pembuatan *player* dimana Kontrol *player* ini berjenis *FPC (First Person Controller).*

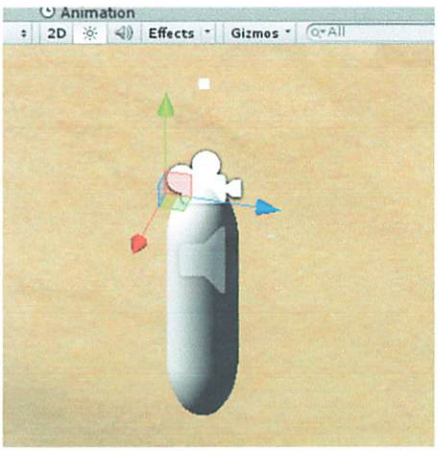
1. Meng-*Import Player Controller*

   Hal utama yang dibutuhkan dalam sebuah game adalah seorang player yang dapat dikontrol sesuka hati. untuk membuat sebuah karakter player yang berjenis *FPC (First Person Controller)* pertama-tama adalah meng-*import* assets yang sudah disediakan oleh unity. dengan cara klik kanan pada tab *project* di bagian *assets*. lalu pilih " *import package*" kemudian pilih " *Character Controller*", sehingga muncul *window* baru yang berisi *assets* dari karakter, kemudian klik tombol *import*.



Gambar 4.56 *Import Character Controller*

Setelah selesai proses *import* selanjutnya memasukkan karakter tersebut dengan cara; buka *folder Standart Assets* > *Character Controller* yang berada pada tab *Project*, lalu pilih *file "First Person Controller"*. selanjutnya *drag* file tersebut ke stage sehingga akan muncul sebuah objek berbentuk *capsule* yang akan digunakan sebagai *player*.



Gambar 4.57 *Character Controller*

Setelah karakter berhasil di *import* kedalam *game*, bisa dilakukan percobaan untuk mencoba melakukan gerakan terhadap *Player* namun sebelum itu, harus menghapus *Main Camera* yang sebelumnya sudah berada di *stage*, *dikarenakan Character Controller* ini memiliki *Main Camera* juga, karena karakter ini bersifat *First Person Look* (Sudut pandang orang pertama).

Terdapat beberapa Kontrol *default* yang sudah disediakan di *assets* tersebut. beberapa Kontrol *keyboard* bisa dilihat di tabel berikut:

Tabel 4.3 *Key Control Player*

Tombol	**Fungsi**
W atau ↑	Maju
A atau ←	Ke arah Kiri
S atau ↓	Mundur
D atau →	Ke arah Kanan
Space	Lompat

Dan untuk melihat arah dari *Player* menggunakan Kontrol *Mouse.* untuk mencoba geraka-gerakan tersebut dapat dilakukan dengan menekan tombol *Play* yang berada pada bagian atas tengah.

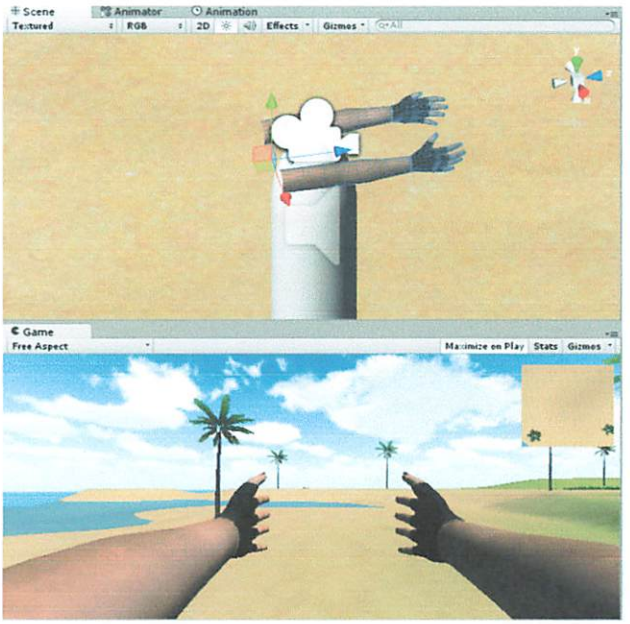
Gambar 4.58 Tombol *Play*

2. Memberikan Tangan ke *Player*

Setelah selesai membuat karakter *player* selanjutnya adalah memberikan tangan ke *Player* agar tampak nyata, karena karakter ini bersifat sudut pandang orang pertama (*First Person*).

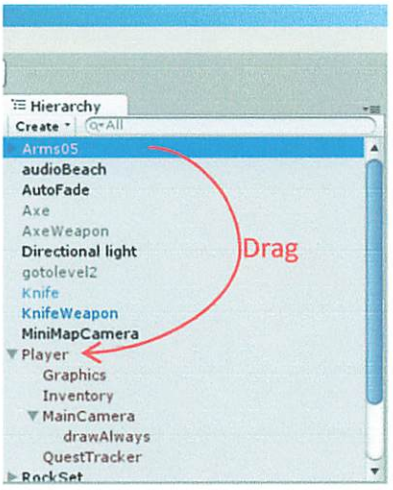
Pertama yang harus disiapkan adalah Sebuah objek tangan. bisa didapatkan dengan men-*download* di internet atau membuat sendiri melalui aplikasi 3D seperti *3dMax, Blender, Maya* dan lain-lain. Setelah objek tangan telah siap, drag file tangan tersebut kedalam *Assets* kemudian tunggu hingga proses selesai.

Setelah proses meng-*import* tangan selesai, selanjutnya adalah memasukkan tangan tersebut ke-*stage* kemudian pada tab *Hierarchy* akan muncul nama dari objek tangan tersebut. langkah selanjutnya adalah mengatur tangan tersebut sehingga sesuai dengan *player*.



Gambar 4.59 *Setting Tangan Player*

Setelah peletakan tangan selesai, selanjutnya adalah menyatukan tangan tersebut dengan *player*. dengan cara men-*drag* nama tangan yang berada pada *hierarchy* kemudian memasukkannya pada objek *Player* di dalamnya.

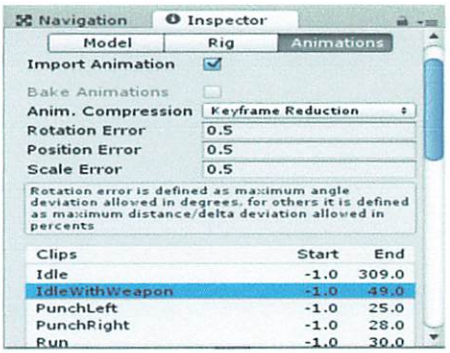


Gambar 4.60 Menyatukan tangan dengan *Player*

Setelah melakukan hal tersebut maka tangan akan mengikuti player kemanapun gerakan yang dilakukan.

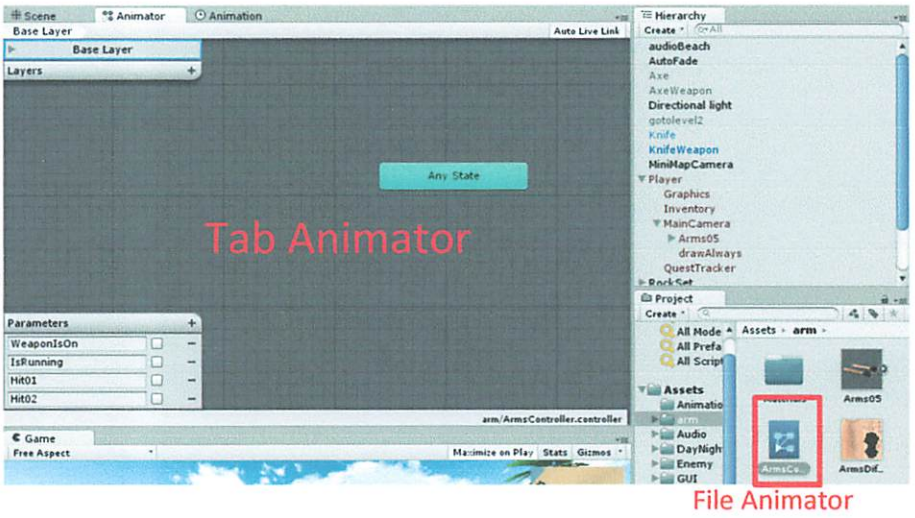
3. Mengatur gerakan animasi tangan

Selanjutnya adalah membuat gerakan animasi pada tangan tersebut sehingga tampak *Real* / nyata. sebelumnya tangan yang telah di*import* tersebut telah memiliki animasi yang sudah dimasukkan. animasi tersebut dapat diatur *timing*-nya atau memberikan nama setiap gerakan dengan cara, klik *file* tangan yang telah di *import,* kemudian pada tab *inspector* tekan tombol "*animations*" disana terdapat beberapa setting untuk animasi yang telah *include* di objek 3D tersebut.



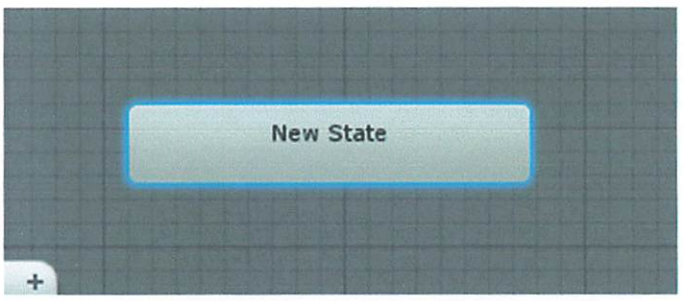
Gambar 4.61 *Setting* Animasi Tangan

Selanjutnya adalah membuat *Animator*. *Animator* disini berfungsi sebagai *Control Animation*, sehingga pergerakan animasi dapat diatur sesuai kebutuhan. untuk membuat *Animator* dapat dilakukan dengan cara klik kanan pada tab *Project* di *Assets*. kemudian pilih "*Create*" > "*Animator Controller*". setelah itu akan muncul sebuah *file* pada *assets*. berikan nama pada *file* tersebut kemudian dobel klik *file* tersebut, sehingga akan muncul tab *Animator*.



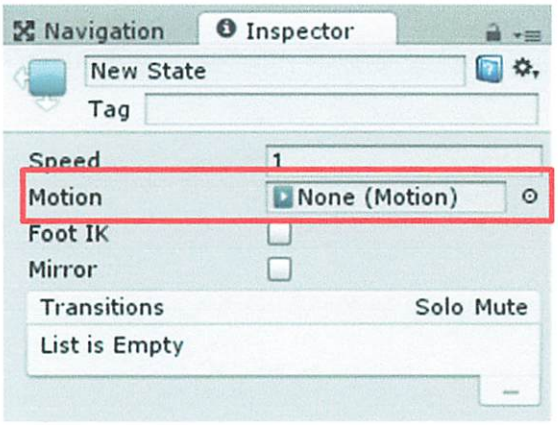
Gambar 4.62 *Tab Animator & file Animator*

Setelah Itu membuat kondisi pada setiap animasi tersebut. dengan cara klik kanan pada *workspace animator* lalu pilih "*Create New State*" > "*Empty*". kemudian akan muncul sebuah *state* / kondisi yang kosong.



Gambar 4.63 *New State Condition*

Selanjutnya Memasukkan animasi pada *State* tersebut, dengan cara klik pada *state* tersebut. kemudian pada tab "*Inspector*". klik pada "*Motion*" kemudian pilih animasi yang akan dipilih.

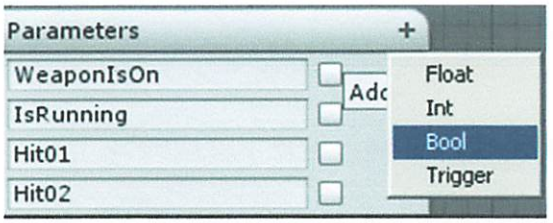


Gambar 4.64 *Setting State*

Sebagai contoh adalah animasi keadaan "*Idle*" dimana keadaan yang akan terjadi saat tidak melakukan apapun. maka disaat *Player* diam, tangan tersebut akan mem-*play* animasi *Idle* dari tangan tersebut.

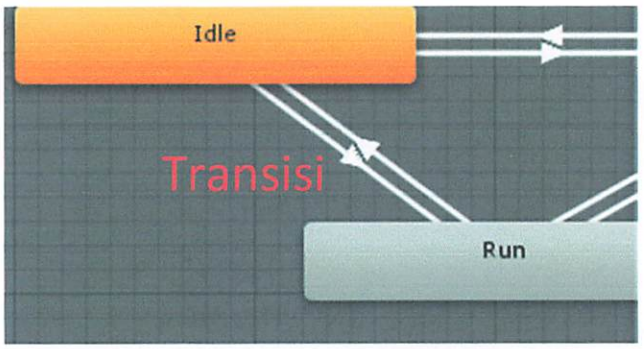
Gambar 4.65 *State Idle*

Untuk mengubah animasi dari satu *State* ke *state* yang lain dibutuhkan sebuah kondisi. untuk membuat kondisi tersebut dengan cara klik tanda "+" pada "*Parameter*" yang berada pada *workspace animator*. kemudian pilih "*Bool*". maka akan muncul kondisi *True/False.*



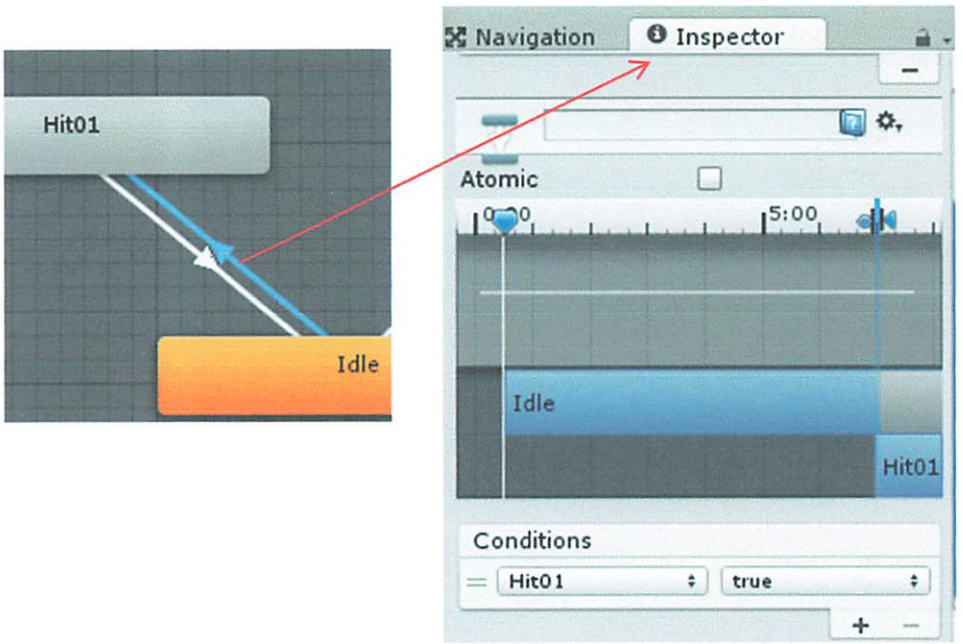
Gambar 4.66 *Parameter State*

Setelah membuat beberapa *state* dengan berbeda-beda animasi dan memberikan beberapa parameter, selanjutnya adalah membuat transisi terhadap animasi satu dengan lainnya, dengan cara klik kanan pada state kemudian pilih "*Make Transision*" lalu akan muncul tanda panah dimana harus tertuju pada state yang lain sebagai transisi/perubahan animasi.



Gambar 4.67 *Transision State*

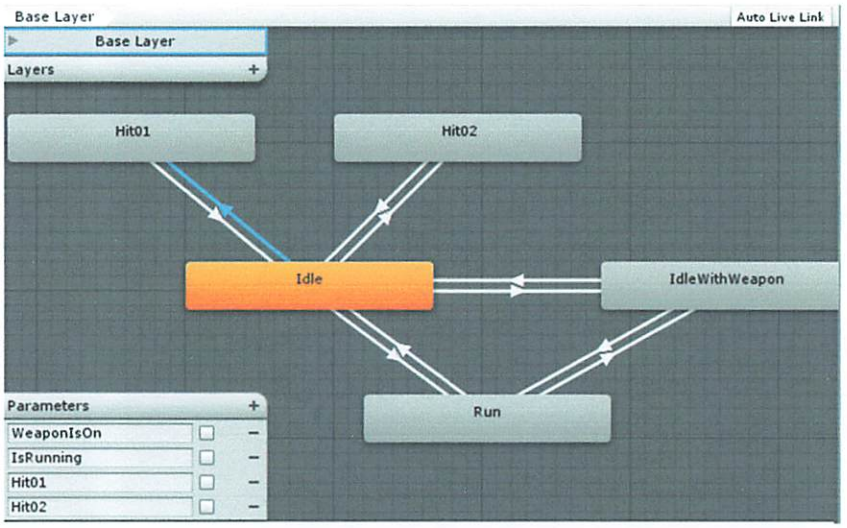
Selanjutnya adalah menyetting transisi tersebut. dengan cara klik pada tanda panah tersebut, sehingga akan muncul settingan pada tab *Inspector.*



Gambar 4.68 *Setting Transision*

Pada Gambar 4.68 di *setting inspector* terdapat setting "*Conditions*" dimana kondisi tersebut berhubungan dengan *Parameters* yang telah dibuat sebelumnya. jadi disaat kondisi tersebut bersifat *true* maka akan menjalankan animasi yang dituju oleh transisi tersebut, dan begitu juga sebaliknya.

Berikut adalah settingan *Animator* Dari tangan dengan beberapa transisi sesuai dengan parameternya.



Gambar 4.69 *Animator controller dengan Transision*

4.1.8 Implementasi Karakter Lawan

Di dalam permainan Karakter Lawan / Musuh adalah karakter yang sangat penting dalam *game* sehingga membuat *game* tersebut tampak begitu menantang dan menarik bagi pemainnya. dengan bentuk ataupun animasi yang menarik karakter musuh bisa menjadi menarik. Dan yang terpenting dalam pembuatan karakter musuh adalah otaknya yaitu *AI (Artificial Intelegence),* dimana berfungsi sebagai perintah otomatis atau pemberian sifat kepada sebuah karakter sehingga dapat menentukan kondisi yang akan dilakukannya dengan sendirinya.

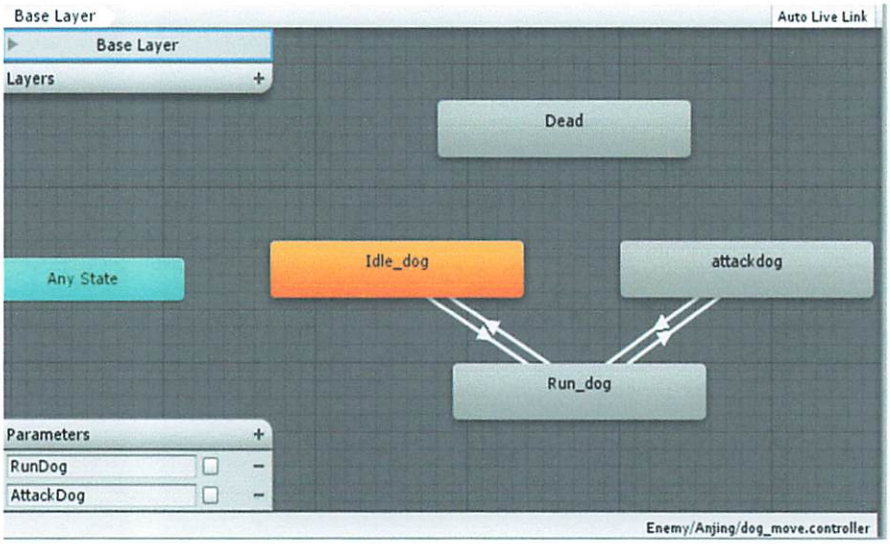
Berikut adalah beberapa tahap / langkah pembuatan karakter lawan yang akan menjadi musuh dari *player* :

1. *Import* Karakter Lawan

Pertama yang harus dilakukan adalah mengimport *file* karakter lawan yg telah terdapat animasi didalamnya. sama halnya dengan mengimport *file* tangan, hanya dengan men-*drag file* tersebut ke *assets,* dan kemudian menaruhnya di *stage*.

Gambar 4.70 *Import Karakter lawan*

Setelah itu menambahkan "*Animator Controller*" pada karakter tersebut. sama hal nya dengan membuat *animator* dari tangan sebelumnya, lakukan transisi, dan kondisi sesuai aksi yang akan dilakukan nantinya. berikut *Animator Controller* dari Karakter lawan tersebut.



Gambar 4.71 *Animator controller lawan*

2. Memberikan kondisi *Health* (darah)

Setiap Lawan dari *player* harus memiliki kondisi *Health* sebagai parameter apakah lawan akan kalah dan mati atau tidak. untuk membuat kondisi *Health* dari karakter lawan dibutuhkan "*JavaScript*". sama seperti sebelumnya pertama membuat *javascript* dengan cara klik kanan dan klik *Create* kemudian "*JavaScript*", dilanjutkan dengan men-doble klik file *javascript* tersebut sehingga akan muncul program "*MonoDevelop*".

selanjutnya adalah memasukkan *script* berikut untuk peng-inisialisasi *Health* dari lawan.

EnemyHealth ▸ Awake

```
1  #pragma strict
2
3  var Darah = 100;
4  var animasiMati : Animator;
5
6
7  function terimaSerangan (Damage : int)
8  {
9      Darah -= Damage ;
10         if (Darah <= 0){
11         //Darah = 1;
12         Mati();
13     }
14 }
15
16 function Mati(){
17
18     AdvanceAIV2.matikah = true;
19     animasiMati.SetBool("AttackDog",false);
20     //yield WaitForSeconds(3);
21     animasiMati.SetBool("RunDog",false);
22     animasiMati.Play("Dead");
23
24     yield WaitForSeconds(3);
25     Destroy (gameObject);
26     AdvanceAIV2.matikah = false;
27     QTLv12.killCount = QTLv12.killCount + 1;
28 }
```

Gambar 4.72 *Script Health lawan*

Untuk penjelasan *script* diatas adalah sebagai berikut: *variable* "Darah" adalah darah / *Health* dari Lawan, dan *variable* "animasiMati" disana adalah sebagai pemanggil *Animator Controller* yang telah dibuat sebelumnya. Untuk Fungsi "terimaSerangan" adalah fungsi dimana darah / *health* dari lawan akan berkurang sesuai dengan serangan yang diberikan oleh *player.* dan yang terakhir adalah fungsi "Mati", dimana akan melakukan aksi dari pemanggilan *variable animator controller* yang telah ditentukan. dan juga menghapus objek lawan saat keadaan mati dengan perintah "*Destroy(gameObject)*;"

3. Menambahkan *Artificial Intelegence*

Sama dengan halnya membuat script *health* dibutuhkan *javaScript* untuk menambahkan kemampuan *AI* pada lawan. untuk permasalahan ini penulis menggunakan Algorithma *AI* yaitu *FSM (Finite State Machine)* sebagai pengaturan sifat dari lawan.

```
function Update () {
    //menentukan jarak antara pemain dengan musuh
    if(RespawnMenuV2.playerIsDead == false)
    {
        Distance = Vector3.Distance(Target.position, transform.position);

        if(Distance < lookAtDistance)
        {
            //TheAnimatorDog.SetBool("RunDog",false);
            if(matikah == false)
            {
            lookAt();
            }
        }
        if(Distance > lookAtDistance)
        {
            TheAnimatorDog.SetBool("RunDog",false);
//          renderer.material.color = Color.green;
        }

        if(Distance < attackRange)
        {
            attack();
        }
        else if(Distance < chaseRange)
        {
            if (matikah == false)
            {
                chase();
            }
```

Gambar 4.73 *Script FSM*

Pada *Script* diatas terdapat beberapa kondisi yang akan dialami oleh lawan, dimana diatur pada *Distance* / Jarak antara player dengan lawan. seperti pada kondisi if yang pertama adalah apabila *player* berada pada jarak jangkauan yang telah ditentukan maka akan menjalakan fungsi "*LookAt*". fungsi tersebut berisi :

```
function lookAt(){
    //renderer.material.color = Color.yellow;

    var rotation = Quaternion.LookRotation(Target.position - transform.position); // melihat sekeli
    transform.rotation = Quaternion.Slerp(transform.rotation, rotation, Time.deltaTime * Damping);

}
```

Gambar 4.74 *Script function LookAt*

Fungsi tersebut adalah untuk membuat objek 3D tersebut melihat kearah *Player* pada kondisi tersebut lawan akan melakukan rotasi ke arah player. Lalu selanjutnya pada kondisi if ke-3 adalah kondisi dimana lawan tidak dapat melihat *player* / di luar jangkauan. kondisi if ke-4 adalah keadaan dimana lawan dapat melakukan serangan kepada pemain.

```
function attack(){
    if(Time.time > attackTime)
    {
        |
        TheAnimatorDog.SetBool("AttackDog", true);
        Target.SendMessage("terimaSerangan", Damage);
        Debug.Log("musuh nyerang");
        attackTime = Time.time + attackRepeatTime;
        yield WaitForSeconds(2);
        TheAnimatorDog.SetBool("AttackDog", false);
    }
}
```

Gambar 4.75 *Script function Attack*

pada fungsi tersebut terdapat beberapa pemanggilan *variable* dari *animator controller* dan juga pengaturan untuk seberapa cepat serangan lawan diluncurkan ke *player.* yang terakhir kondisi *if* ke-5 adalah kondisi dimana *player* dengan lawan berada pada jarak yang

memenuhi sehingga lawan akan mengejar player sampai pada batas jarak yang telah ditentukan.

```
function chase(){
    TheAnimatorDog.SetBool("RunDog", true);
    TheAnimatorDog.SetBool("AttackDog", false);
    //renderer.material.color = Color.red;
    // bergerak mengikuti pemain
    moveDirection = transform.forward;
    moveDirection *= moveSpeed;

    moveDirection.y -= gravity* Time.deltaTime;
    controller.Move(moveDirection* Time.deltaTime);
}
```

Gambar 4.76 *Script function Chase*

fungsi diatas mengaktifkan parameter dari *animator controller* lawan, sehingga lawan akan tampak sedang berlari. dan untuk "*moveDirection*" adalah penentuan arah gerak dan kecepatan dari lawan untuk mengejar *player*.

4.2 Pengujian

Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan.

4.2.1 Pengujian Fungsional

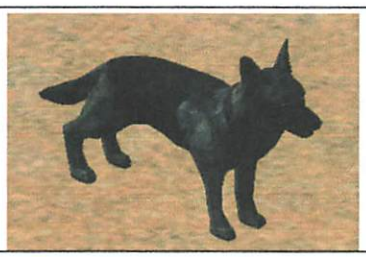
Game yang telah dibuat kemudian diujikan kepada user untuk dimainkan. Selanjutnya *user* diminta untuk menilai kesesuaian dan komentar dalam *game Survival From Island.* Hasil dari pengujian dapat dilihat pada tabel 4.4

Tabel 4.4 Pengujian Fungsional

No.	Fungsi	Hasil
1.	Karakter musuh dapat mengejar karakter *player* Dengan Implementasi *Artificial Intelegence Finite State Machine (FSM).*	[√] Sesuai
2.	Karakter musuh dapat menyerang pemain ketika berada pada jarak serang.	[√] Sesuai
3.	Tombol pada menu utama berjalan sesuai dengan fungsi.	[√] Sesuai
4.	Status bar dan *level* bar pada *game* menunjukkan indikator sesuai dengan fungsi masing-masing.	[√] Sesuai
5.	Pemberian *health* pada karakter berjalan normal dengan indikasi karakter pamian mati ketika *health* poin kurang dari nol.	[√] Sesuai
6.	Karakter *Player* dapat bergerak sesuai dengan fungsi yang diberikan.	[√] Sesuai
7.	*System cheat* yang memungkinkan *player* untuk mengakses *level* berikutnya maupun memberikan kekuatan kepada *player*	[√] Sesuai

Dari tabel 4.4 disimpulkan bahwa semua fungsi berjalan sesuai dengan tujuan yang diharapkan.

Tabel 4.5 Pengujian *Finite State Machine*

No.	Karakter	Action (FSM)	Keterangan
1.		• Pada jarak 25 unit melihat *Player* • Pada jarak 15 unit mengejar *Player* • Pada jarak 1.5 Unit Menyerang *Player*	[√] Sesuai

No.	Karakter	*Action(FSM)*	Keterangan
2.		• Pada jarak 25 unit melihat *Player* • Pada jarak 15 unit mengejar *Player* • Pada jarak 1.5 Unit Menyerang *Player*	[√] Sesuai
3.		• Pada jarak 30 unit melihat *Player* • Pada jarak 20 unit mengejar *Player* • Pada jarak 1.7 Unit Menyerang *Player*	[√] Sesuai
4.		• Pada jarak 30 unit melihat *Player* • Pada jarak 17.5 unit mengejar *Player* • Pada jarak 2.0 Unit Menyerang *Player*	[√] Sesuai

Keterangan :

- Unit : Satuan ukuran / Jarak dalam bidang 3D yang bersumbu pada koordinat x,y dan z.

Dari tabel 4.5 disimpulkan bahwa Pengujian dari *AI Finite State Machine* Sesuai dengan fungsi yang diharapkan.

Tabel 4.6 Tabel Pengujian Sistem *Health* Karakter

No.	Target Uji	Hp Awal	Serangan	*Kekuatan Serang*	*Health* Poin Akhir	Kesimpulan
1.	Player	100	Pukulan,Pisau,Kapak	20,25,30	0	[√] Sesuai
2.	Anjing	100	Gigitan	10	0	[√] Sesuai
3.	Harimau	100	Pukulan, Gigitan	20	0	[√] Sesuai
4.	Serigala	100	Gigitan	15	0	[√] Sesuai
3.	Boss Musuh	100	Pukulan	25	0	[√] Sesuai

Dari tabel 4.6 disimpulkan bahwa pengujian sistem *health* semua berjalan sesuai tujuan yang diharapkan.

Tabel 4.7 Tabel Pengujian *Cheat*

No.	*Cheat Code*	Fungsi	Hasil
1.	*tooeasy*	Menuju *level* 2	[√] Sesuai
2.	*luckyday*	Menuju *level* 3	[√] Sesuai
3.	*callmeyourboss*	Menuju *level* 4	[√] Sesuai
4.	*imtheironman*	Darah tak terbatas	[√] Sesuai
5.	*hulkmodeon*	Pukulan kuat (sekali serang)	[√] Sesuai
6.	*iwannabenormal*	Kembali ke kondisi normal	[√] Sesuai

Dari tabel 4.7 menunjukkan bahwa semua fungsi dari *cheat* berjalan sesuai dengan keinginan *user*.

Tabel 4.8 Tabel pengujian *Control Player*

No.	Tombol	Fungsi	Hasil
1.	W atau ↑	Bergerak Maju	[√] Sesuai
2.	A atau ←	Bergerak Ke kiri	[√] Sesuai
3.	S atau ↓	Bergerak Mundur	[√] Sesuai
4.	D atau →	Bergerak Ke Kanan	[√] Sesuai
5.	*Space*	Melompat	[√] Sesuai
6.	C	Jongkok	[√] Sesuai
7.	*Shift*	Berlari	[√] Sesuai
8.	*Esc*	Menampilkan *Menu* (di dalam *game*)	[√] Sesuai
9.	I	*Inventory* (Tempat barang di *Game*)	[√] Sesuai
10.	*Left Click*	Melakukan Serangan	[√] Sesuai

Dari tabel 4.8 menunjukkan bahwa semua fungsi dari *control player* berjalan sesuai dengan keinginan *user*.

4.2.2 Pengujian Non Fungsional

Pengujian non fungsional dilakukan oleh penulis. Pengujian seputar non fungsional secara *performance* dan secara *interface* dilakukan untuk mendapatkan kesesuaian berdasarkan tahap perancangan.

Tabel 4.9 Pengujian Jenis *OS*

No.	Jenis *Flas Player*	Hasil
1.	Windows XP	[√] Sesuai
2.	Windows 7	[√] Sesuai
3.	Windows 8	[√] Sesuai

Dari tabel 4.9 menunjukkan bahwa untuk 3 jenis Sistem Operasi dapat menjalankan game *Survival From Island* ini.

Tabel 4.10 Tabel Pengujian *Performance Hardware*

No	Processor	RAM	VGA	SO	Keterangan
1.	Intel core i5	4 GB	2737 Mb	Win7	Aplikasi berjalan lancar
2.	Intel core i3	4 GB	2048 Mb	Win8	Aplikasi berjalan lancar
3.	Intel core i3	2 GB	768 Mb	Win7	Aplikasi berjalan lancar
4.	AMD Athlon	1 GB	512 Mb	Win XP	Aplikasi berjalan lancar

Dari tabel 4.10 disimpulkan bahwa *game Survival From Island* dapat dijalankan pada komputer dengan jumlah RAM yang berbeda maupun versi *Windows* yang berbeda.

Tabel 4.11 Tabel Ketertarikan Tampilan *Game*

No.	Jawaban	Responden	Prosentase (%)
1.	Sangat Menarik	7	35%
2.	Menarik	9	45%
3.	Cukup Menarik	4	20%
4.	Kurang Menarik	0	0
5.	Tidak Menarik	0	0

Dari tabel 4.11 disimpulkan bahwa dari 20 *user,* 35% mengatakan *game* ini sangat menarik, 45% mengatakan menarik, dan 20% mengatakan cukup menarik.

Tabel 4.12 Tabel Tingkat Kesulitan Level

No.	Jawaban	Responden	Prosentase (%)
1.	Sangat Sulit	8	40%
2.	Sulit	5	25%
3.	Cukup Mudah	5	25%
4.	Mudah	2	10%
5.	Sangat Mudah	0	0

Dari tabel 4.12 disimpulkan bahwa dari 20 *user*, 40% mengatakan *game* ini sangat sulit, 25% mengatakan sulit, 25% cukup mudah, dan 10% mengatakan mudah.

Tabel 4.13 Tabel Ketertarikan Alur Cerita *Game*

No.	Jawaban	Responden	Prosentase (%)
1.	Sangat Baik	3	15%
2.	Baik	10	50%
3.	Cukup Baik	7	35%
4.	Buruk	0	0
5.	Sangat Buruk	0	0

Dari tabel 4.13 disimpulkan bahwa dari 20 *user*, 10% mengatakan *game* ini memiliki cerita yang sangat baik, 50% mengatakan Baik, dan 25% cukup baik untuk seluruh alur cerita.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah Penyelesaian *game Survival from Island* maka penulis dapat menyimpulkan:

1. Implementasi dari *Finite State Machines* dapat diterapkan pada aplikasi *game 3D adventure* dengan indikasi musuh dalam *game* mengejar dan menyerang karakter pemain dalam kondisi tertentu.
2. Semua fungsi dari *cheat* dan *control player* berjalan sesuai dengan keinginan *user* dan *game Survival from Island* dapat dijalankan pada komputer dengan jumlah RAM yang berbeda maupun versi *Windows* yang berbeda.
3. Dari hasil pengujian pada pengguna bahwa dari tingkat ketertarikan disimpulkan bahwa 45% orang mengatakan tampilan *game* ini menarik. Sedangkan dari tingkat kesulitan, 40% mengatakan *game* ini sangat sulit. dan dari ketertarikan alur cerita dari game 50% mengatakan alur cerita cukup baik.
4. Pembuatan media *game* dengan *Unity* memakan banyak *resource*, dibuktikan dengan ukuran file exe yang cukup besar . Dan juga karena memiliki grafik 3D.

5.2 Saran

Setelah dilakukan pengujian terhadap *game Survival from Island* maka masih ada kekurangan sehingga untuk pengembangan lebih lanjut disarankan:

1. Karakter musuh dapat ditambahkan seperti Beruang, Komodo, dan lain-lain sehingga *game* jadi lebih variatif.
2. Menambahkan efek animasi berupa keluarnya darah dari musuh ketika menerima serangan dari *Player*.

3. Dalam *game* ini hanya terdapat 4 *level* sehingga dalam pengembangan lebih lanjut *game Survival from Island* ini dapat ditambahkan level yang lebih banyak hingga 10 level.
4. *Prolog* dan *Epilog* cerita dalam *Survival from Island* berupa narasi sehingga dalam pengembangan lebih lanjut dalam *game Survival from Island* dapat ditambahkan *scene* animasi 3D dengan cerita agar tampilan *scene* lebih menarik.
5. Karakter utama *game Survival from Island* hanya mempunyai 2 jenis senjata sehingga dalam pengambangan lebih lanjut variasi serangan utama dapat ditambahkan sehingga *game* lebih vatiatif seperti penambahan *item-item* baru layaknya helm, baju besi dan lain-lain.

DAFTAR PUSTAKA

Davidson, Andrew.2005. Killer Game Programming in Java. O'Reilly.

Kyaw, Aung Sithu.2013. Unity 4.X Game AI Programming. Packt Publishing.

Michael G strintzis, Nikos Sarris. 2005. 3D Modelling Animation: Syntesis and Analysis Technique for Human Body. IRM Press.

Miller, Tom. 2004. Beginning 3D Game Programming. Sams Publishing.

Norton, Terry. 2013. Learning C# by Developing Games with Unity 3D Bigginer's Guide. Packt Publishing.

Rahardianto, Muhammad Ichsan. 2012. Membuat game 3D berbasis web menggunakan Unity. Interactify Publishing. Jakarta.

Mizanuddin. Unity (game engine), tersedia pada (http://repository.amikom.ac.id/files/Publikasi_07.11.1567.pdf, diakses tanggal 26 Maret pukul 15.06)

Neal Hirsig, 2013. Blender 3D Design Course (http://gryllus.net/Blender/3D.html. Diakses 08 Februari 2014.)

Walker Boys Studio, 2014. Unity Tutorial (Basic-Intermediate) (http://walkerboystudio.com/html/unity_training___free__.html Diakses 18 Januari 2014.)

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. RAYA Karanglo, Km2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 21 April 2014

Nomor : ITN-253/INF/TA/2014
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Joseph Dedy Irawan, ST, MT
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama	:	LALU SANOSTRA AULIA FAHMI
Nim	:	1018137
Prodi	:	Teknik Informatika S-1
Fakultas	:	Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

21 April 2014 S/D 21 September 2014

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP. : 197404162005021002

Form S-4a

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. RAYA Karanglo, Km2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 21 April 2014

Nomor : ITN-253/INF/TA/2014
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Suryo Adi Wibowo, ST.MT
Dosen Pembina Program Studi Teknik Informatika S-1
Institut Teknologi Nasional
Malang

Dengan Hormat,
Sesuai dengan permohonan dan persetujuan dalam proposal skripsi untuk mahasiswa :

Nama : LALU SANOSTRA AULIA FAHMI
Nim : 1018137
Prodi : Teknik Informatika S-1
Fakultas : Teknologi Industri

Maka dengan ini pembimbingan kami serahkan sepenuhnya kepada Saudara/i selama waktu 6 (enam) bulan, terhitung mulai tanggal :

21 April 2014 S/D 21 September 2014

Sebagai satu syarat untuk menempuh Ujian Akhir Sarjana Teknik, Program Studi Teknik Informatika S-1.
Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Program Studi Teknik Informatika S-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP. : 197404162005021002

Form S-4a

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Nasional Malang
Program Studi Teknik Informatika S1

FORMULIR BIMBINGAN SKRIPSI

Nama : Lalu Sanostra Aulia Fahmi
NIM : 1018137
Masa Bimbingan : 19 Maret 2014 –20 Agustus 2014
Judul Skripsi : Rancang Bangun *Game Survival from Island* Menggunakan Metode *Finite State Machine (FSM)*

No.	TANGGAL	URAIAN	PARAF PEMBIMBING
1	14-07-2014	Demo Game	
2	25-07-2014	Revisi Game	
3	23-07-2014	Bimbingan Jurnal Seminar Hasil	
4	28-07-2014	Revisi Jurnal Seminar Hasil	
5	15-08-2014	Bimbingan Laporan Bab 1-5	
6	16-08-2014	Bimbingan Laporan Bab 1-5	

16-8-2014

Malang, 07 Agustus 2014
Dosen Pembimbing

Joseph Dedy Irawan, ST., MT.
NIP. 19740416200501 1002

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Nasional Malang
Program Studi Teknik Informatika S1

FORMULIR BIMBINGAN SKRIPSI

Nama : Lalu Sanostra Aulia Fahmi
NIM : 1018137
Masa Bimbingan : 19 Maret 2014 –20 Agustus 2014
Judul Skripsi : Rancang Bangun *Game Survival from Island* Menggunakan Metode *Finite State Machine (FSM)*

No.	TANGGAL	URAIAN	PARAF PEMBIMBING
1	10-07-2014	Bimbingan Laporan Bab 1-3	
2	12-07-2014	Revisi Laporan Bab 1-3	
3	19-07-2014	Revisi Laporan Bab 1-3	
4	19-07-2014	Demo Game	
5	20-07-2014	Bimbingan Laporan Bab 4-5	
6	21-07-2014	Revisi Laporan Bab 4-5	
5	22-07-2014	Bimbingan Jurnal Seminar Hasil	
6	26-07-2014	Revisi Jurnal Seminar Hasil	
7	15-08-2014	Bimbingan Laporan Bab 1-5	
8	16-08-2014	Bimbingan Laporan Bab 1-5	

Malang, 07 Agustus 2014
Dosen Pembimbing

Suryo Adi Wibowo, ST., MT.
NIP.P. 1031000438

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama : Lalu Sanostra Aulia Fahmi
NIM : 1018137
Jurusan : Teknik Informatika S-1
Judul : Rancang Bangun Game Survival From Island Menggunakan Metode Finite State Machine (FSM).

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Selasa
Tanggal : 19 Agustus 2014
Tempat : Ruang Rapat Dosen Teknik Informatika S-1
Nilai : (A)

Panitia Ujian Skripsi :

Ketua Majelis Penguji

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Anggota Penguji :

Penguji Pertama	Penguji Kedua
Sonny Prasetio, ST.,MT.	**Nurlaily Vendyansyah, ST.**
NIP.P. 1031000433	

INSTITUT TEKNOLOGI NASIONAL MALANG
Fakultas Teknologi Industri
Program Studi Teknik Informatika S1

FORMULIR PERBAIKAN SKRIPSI

Nama : Lalu Sanostra Aulia Fahmi
NIM : 1018137
Jurusan : Teknik Informatika S-1
Judul : Rancang Bangun Game Survival From Island Menggunakan Metode Finite State Machine.

Tanggal	Penguji	Uraian	Paraf
19 Agustus 2014	I	- Tambah Kategori Pengujian - Menambah user pengujian menjadi 20 orang - Revisi Kesimpulan dan Abstrak	
19 Agustus 2014	II	- Revisi Abstrak - Lampirkan Kuisoner Pengujian User	

Anggota Penguji :

Penguji Pertama

Sonny Prasetio, ST.,MT.
NIP.P. 1031000433

Penguji Kedua

15/9/2014

Nurlaily Vendyansyah, ST.

Mengetahui

Dosen Pembimbing I

Joseph Dedy Irawan,ST, MT
NIP. 197404162005031002

Dosen Pembimbing II

Suryo Adi Wibowo, ST., MT
NIP.P. 1031000438

Script Status Health Player

```
#pragma strict
static var MaxDarah = 100;
static var Darah : float;
var texture1: Texture2D;
var texture2: Texture2D;
var texture3: Texture2D;
var Crosshair : Texture2D;
var mat : Material;
var x : float = 0;
var y : float = 0;
var w : float;
var h : float;
var MyClip : AudioClip;
var MyClipDamage : AudioClip;
static var cheatActive : boolean =
false;

function Start()
{
      Darah = MaxDarah;
}

function terimaSerangan (Damage :
int)
{
      audio.PlayOneShot(MyClipDamage)
;
      Darah -= Damage ;

            if (Darah <= 0){
            //Darah = 1;

      audio.PlayOneShot(MyClip);
            Mati();
      }
}

function OnGUI(){
      if(Event.current.type.Equals(Ev
entType.Repaint)){
            var box : Rect = new
Rect(x, y, w, h);
            Graphics.DrawTexture(box,
texture1, mat);
      }
      GUI.DrawTexture(Rect(10,550,200
,40),texture2);
      GUI.DrawTexture(Rect(195,550,51
0,50),texture3);
      GUI.DrawTexture(Rect(Screen.wid
th*0.5-7.5,Screen.height*0.5-
7.5,15,15), Crosshair);
}
```

```
function Update () {
      if(cheatActive == true){
      var healthy2 : float = 1-
(Darah/100000);
            if(healthy2 <= 0){
                  healthy2 = 0.1;
            }
      mat.SetFloat("_Cutoff",
healthy2);
      }
      else{
      var healthy : float = 1-
(Darah/100);
            if(healthy <= 0){
                  healthy = 0.1;
            }
      mat.SetFloat("_Cutoff",
healthy);
      }
}

function Mati(){
      RespawnMenuV2.playerIsDead
= true;
      Debug.Log("i;m die");
}

function RespawnStats()
{
      Darah = MaxDarah;
}
```

Script Status Health Enemy (Lawan)

```
#pragma strict
var Darah = 100;
var animasiMati : Animator;

function terimaSerangan (Damage : int)
{
	Darah -= Damage ;
		if (Darah <= 0){
		//Darah = 1;
		Mati();
	}
}

function Mati(){

	AdvanceAIV2.matikah = true;
	animasiMati.SetBool("AttackDog",false);
	//yield WaitForSeconds(3);
	animasiMati.SetBool("RunDog",false);
	animasiMati.Play("Dead");

	yield WaitForSeconds(3);
	Destroy (gameObject);
	AdvanceAIV2.matikah = false;
	QTLvl2.killCount = QTLvl2.killCount + 1;
}
```

Script Respawn Menu

```
#pragma strict
var lookAround01 : MouseLook;
var lookAround02 : MouseLook;
var charController : CharacterController;
//var charMotor : CharacterMotor;
//var sprintScript : SprintAndCrouch;
var hitStop : MeleeSystem;

static var playerIsDead = false;
```

```
function Start () {
	lookAround01 = gameObject.GetComponent(MouseLook);
	lookAround02 =
GameObject.Find("MainCamera").GetComponent(MouseLook);
	charController = gameObject.GetComponent(CharacterController);
	//charMotor = gameObject.GetComponent(CharacterMotor);
	//sprintScript = gameObject.GetComponent(SprintAndCrouch);
	hitStop = GameObject.Find("Mace").GetComponent(MeleeSystem);
}

function Update () {
	if(playerIsDead == true)
	{
		lookAround01.enabled = false;
		lookAround02.enabled = false;
		charController.enabled =false;
		//sprintScript.enabled = false;
		//charMotor.enabled = false;
		hitStop.enabled = false;

	}
}

function OnGUI()
{
	if(playerIsDead == true)
	{
		if(GUI.Button(Rect (Screen.width*0.5-50, 200-20, 100, 40),
"Respawn"))
		{
			RespawnPlayer();
		}
		if(GUI.Button(Rect (Screen.width*0.5-50, 240, 100, 40),
"Menu"))
		{
			Debug.Log("back to menu");
		}
	}
}

function RespawnPlayer()
{
	Debug.Log("Respawn Player");
}
```

Script Artificial Intelegence Finite State Machine

```
var Distance;
var Target : Transform;
var lookAtDistance = 25.0;
var attackRange = 1.5;
var chaseRange = 15.0;
var moveSpeed = 5.0;
var Damping = 6.0;
var attackRepeatTime = 0.1;
var TheAnimatorDog : Animator;
static var matikah :boolean = false;


var Damage = 40;

private var attackTime : float;

var controller : CharacterController;
var gravity : float= 20.0;
private var MoveDirection : Vector3 = Vector3.zero;

function Start(){
      attackTime = Time.time;
}

function Update () {

      if(RespawnMenuV2.playerIsDead == false)
      {
            Distance = Vector3.Distance(Target.position,
transform.position);

            if(Distance < lookAtDistance)
            {
                  if(matikah == false)
                  {
                  lookAt();
                  }
            }
            if(Distance > lookAtDistance)
            {
                  TheAnimatorDog.SetBool("RunDog",false);
            }

            if(Distance < attackRange)
            {
                  attack();
            }
            else if(Distance < chaseRange)
            {
                  if (matikah == false)
                  {
                        chase();
                  }

            }
      }
```

```
function lookAt(){

      var rotation = Quaternion.LookRotation(Target.position -
transform.position);
      transform.rotation = Quaternion.Slerp(transform.rotation,
rotation, Time.deltaTime * Damping);

}

function chase(){
      TheAnimatorDog.SetBool("RunDog", true);
      TheAnimatorDog.SetBool("AttackDog", false);

      moveDirection = transform.forward;
      moveDirection *= moveSpeed;

      moveDirection.y -= gravity* Time.deltaTime;
      controller.Move(moveDirection* Time.deltaTime);
}

function attack(){
      if(Time.time > attackTime)
      {

            TheAnimatorDog.SetBool("AttackDog", true);
            Target.SendMessage("terimaSerangan", Damage);
            Debug.Log("musuh nyerang");
            attackTime = Time.time + attackRepeatTime;
            yield WaitForSeconds(2);
            TheAnimatorDog.SetBool("AttackDog", false);
      }
}

function terimaSerangan(){
      chaseRange += 30;
      moveSpeed += 2;
      lookAtDistance += 40;
```

Script Senjata (*Weapon Item*)

```
#pragma strict

static var TheDamage: int = 35;
var Distance : float;
var MaxDistance : float = 2.5;
var TheAnimator : Animator;
var DammageDelay : float = 0.6;

static var konfirmasi : boolean = true;
static var dontAttack : boolean = false;

private var Hit01Streak = 0;
private var Hit02Streak = 0;

function Update ()
{
	if(Input.GetButtonDown("Fire1") && konfirmasi == true)
	{
		if (dontAttack == false){
			AttackDamage();
		}
	}
}

function AttackDamage()
{
	if(Random.value >=0.5 && Hit01Streak <=2)
	{
		TheAnimator.SetBool("Hit01", true);
		Hit01Streak += 1;
		Hit02Streak = 0;
	}
	else
	{
		if(Hit02Streak <= 2)
		{
			TheAnimator.SetBool("Hit02", true);
			Hit01Streak = 0;
			Hit02Streak += 1;
		}
		else
		{
			TheAnimator.SetBool("Hit01", true);
			Hit01Streak += 1;
			Hit02Streak = 0;
		}
	}

	yield WaitForSeconds(DammageDelay);
```

```
      var hit: RaycastHit;
      var ray = Camera.main.ScreenPointToRay(Vector3(Screen.width/2,
Screen.height/2, 0 ));
      if(Physics.Raycast(ray, hit))
      {
            Distance = hit.distance;
            if (Distance < MaxDistance)
            {
                  hit.transform.SendMessage("terimaSerangan",
TheDamage, SendMessageOptions.DontRequireReceiver);
            }
      }
      TheAnimator.SetBool("Hit01", false);
      TheAnimator.SetBool("Hit02", false);
}
```

Script Weapon Switching (mengganti senjata)

```
#pragma strict

var currentWeapon = 0;
var maxWeapon = 3;
var theAnimator : Animator;

function Awake ()
{
      SelectWeapon(0);
}

function Update ()
{
      if(Input.GetAxis("Mouse ScrollWheel") > 0)
      {
            if(currentWeapon + 1 <= maxWeapon)
            {
                  currentWeapon++;
            }
            else
            {
                  currentWeapon = 0;
            }
            SelectWeapon(currentWeapon);
      }
      else if(Input.GetAxis("Mouse ScrollWheel") < 0)
      {
            if(currentWeapon - 1 >= 0)
            {
                  currentWeapon--;
            }
            else
            {
                  currentWeapon = maxWeapon;
            }
            SelectWeapon(currentWeapon);
      }
```

```
	if(currentWeapon == maxWeapon + 1)
	{
		currentWeapon = 0;
	}
	if(currentWeapon == -1)
	{
		currentWeapon = maxWeapon;
	}

	if(Input.GetKeyDown(KeyCode.Alpha1))
	{
		currentWeapon = 0;
		SelectWeapon(currentWeapon);
	}
	if(Input.GetKeyDown(KeyCode.Alpha2))
	{
		currentWeapon = 1;
		SelectWeapon(currentWeapon);
	}
	if(Input.GetKeyDown(KeyCode.Alpha3))
	{
		currentWeapon = 2;
		SelectWeapon(currentWeapon);
	}
	if(Input.GetKeyDown(KeyCode.Alpha4))
	{
		currentWeapon = 3;
		SelectWeapon(currentWeapon);
	}
}

function SelectWeapon(index : int)
{
	for(var i = 0; i < transform.childCount;i++)
	{
		if(i == index)
		{
			if(transform.GetChild(i).name == "Fists")
			{
				theAnimator.SetBool("WeaponIsOn", false);
			}
			else
			{
				theAnimator.SetBool("WeaponIsOn", true);
			}
			transform.GetChild(i).gameObject.SetActive(true);
		}
		else
		{
			transform.GetChild(i).gameObject.SetActive(false);
		}
	}
}
```

Script Tombol 3D

```
var levelToLoad : String;
var soundhover : AudioClip;
var beep : AudioClip;
var QuitButton : boolean = false;
function OnMouseEnter(){
audio.PlayOneShot(soundhover);
}
function OnMouseUp(){
audio.PlayOneShot(beep);
yield new WaitForSeconds(0.35);
if(QuitButton){
Application.Quit();
}
else{
Application.LoadLevel(levelToLoad);
}
}
@script RequireComponent(AudioSource)
```

Script FPSWalker (Pergerakan Karakter *Player*)

```
var walkSpeed = 7.0;
var runSpeed = 13.0;
var crouchSpeed = 3.0;
var animasitangan : Animator;
var limitDiagonalSpeed = true;

var enableRun = true;
var enableCrouch = true;

var jumpSpeed = 8.0;
var gravity = 20.0;

var enableFallingDamage = true;
var fallingDamageThreshold = 10.0;
var fallingDamageMultiplier = 2;

var slideWhenOverSlopeLimit = false;
var slideOnTaggedObjects = false;
var slideSpeed = 12.0;

var airControl = false;
 var antiBumpFactor = .75;

var antiBunnyHopFactor = 1;

private var moveDirection = Vector3.zero;
private var grounded = false;
private var controller : CharacterController;
private var myTransform : Transform;
private var speed : float;
private var hit : RaycastHit;
private var fallStartLevel : float;
private var falling = false;
private var slideLimit : float;
```

```
private var rayDistance : float;
private var contactPoint : Vector3;
private var playerControl = false;
private var jumpTimer : int;

private var charHeight : float; //Initial height

function Start () {
      controller = GetComponent(CharacterController);
      myTransform = transform;
      speed = walkSpeed;
      rayDistance = controller.height * .5 + controller.radius;
      slideLimit = controller.slopeLimit - .1;
      jumpTimer = antiBunnyHopFactor;
      oldPos = transform.position;
}

function FixedUpdate() {
      var inputX = Input.GetAxis("Horizontal");
      var inputY = Input.GetAxis("Vertical");
      var inputModifyFactor = (inputX != 0.0 && inputY != 0.0 &&
limitDiagonalSpeed)? .7071 : 1.0;
      animasitangan.SetBool("IsRunning",false);
      if (grounded) {
            var sliding = false;

            if (Physics.Raycast(myTransform.position, -Vector3.up,
hit, rayDistance)) {
                  if (Vector3.Angle(hit.normal, Vector3.up) >
slideLimit)
                        sliding = true;
            }
            else {
                  Physics.Raycast(contactPoint + Vector3.up, -
Vector3.up, hit);
                  if (Vector3.Angle(hit.normal, Vector3.up) >
slideLimit)
                        sliding = true;
            }

            if (falling) {
                  falling = false;
                  if (myTransform.position.y < fallStartLevel -
fallingDamageThreshold && enableFallingDamage == true)
                        ApplyFallingDamage (fallStartLevel -
myTransform.position.y);
            }

            if (Input.GetKey(KeyCode.LeftShift) && enableRun ==
true)
            {
                  animasitangan.SetBool("IsRunning",true);
                  speed = runSpeed;
            }

            else if (Input.GetKey("c") && enableCrouch == true)
            {
```

```
                speed = crouchSpeed;
            }
            else
            {
                speed = walkSpeed;
            }

            if ( (sliding && slideWhenOverSlopeLimit) ||
(slideOnTaggedObjects && hit.collider.tag == "Slide") ) {
                var hitNormal = hit.normal;
                moveDirection = Vector3(hitNormal.x, -hitNormal.y,
hitNormal.z);
                Vector3.OrthoNormalize (hitNormal, moveDirection);
                moveDirection *= slideSpeed;
                playerControl = false;
            }

            else {
                moveDirection = Vector3(inputX *
inputModifyFactor, -antiBumpFactor, inputY * inputModifyFactor);
                moveDirection =
myTransform.TransformDirection(moveDirection) * speed;
                playerControl = true;
            }


            if (!Input.GetButton("Jump"))
                jumpTimer++;
            else if (jumpTimer >= antiBunnyHopFactor) {
                moveDirection.y = jumpSpeed;
                jumpTimer = 0;
            }
        }
        else {

            if (!falling) {
                falling = true;
                fallStartLevel = myTransform.position.y;
            }

            if (airControl && playerControl) {
                moveDirection.x = inputX * speed *
inputModifyFactor;
                moveDirection.z = inputY * speed *
inputModifyFactor;
                moveDirection =
myTransform.TransformDirection(moveDirection);
            }
        }

        moveDirection.y -= gravity * Time.deltaTime;

        grounded = (controller.Move(moveDirection * Time.deltaTime) &
CollisionFlags.Below) != 0;
}
```

```
function Update ()
{
}

function OnControllerColliderHit (hit : ControllerColliderHit) {
      contactPoint = hit.point;
}

function ApplyFallingDamage (fallDistance : float) {
      gameObject.SendMessage("terimaSerangan",
fallDistance*fallingDamageMultiplier);
      Debug.Log ("Ouch! Fell " + fallDistance + " units!");
}

@script RequireComponent(CharacterController)
```

Script Option Menu

```
@script ExecuteInEditMode()
var soundhover : AudioClip;
var beep : AudioClip;
var tutorial :Texture2D;
static var option :boolean = false;
var QuitButton : boolean = false;

var penglihatan : look;
var kamera : GameObject;

var options= false;
var sound = false;
var video = false;

var sfxVol : int = 6;
var musicVol : int = 6;

var fieldOfView : int = 80;

function Start(){
      penglihatan = kamera.GetComponent(look);
}

function OnMouseEnter(){
audio.PlayOneShot(soundhover);
}

function OnMouseUp(){
audio.PlayOneShot(beep);
yield new WaitForSeconds(0.35);

if(QuitButton){
Application.Quit();
```

```
}
else{
option = true;
options = true;
penglihatan.enabled = false;
}
}
@script RequireComponent(AudioSource)

function Update(){
      if(Input.GetKey(KeyCode.Escape)){
            option = false;
            options = false;
            penglihatan.enabled = true;
      }
}

function OnGUI () {
//    GUI.skin.box = myskin;
      if (option == true){
            if(options){
                  GUI.Box(Rect(Screen.width/2-150,Screen.height/2-
150,300,300),"Options");
                  if (GUI.Button(Rect(Screen.width/2 - 50,
Screen.height/2 -80, 100, 30), "Audio Settings")){
                        options = false;
                        sound = true;
                  }


                  if (GUI.Button(Rect(Screen.width/2 - 50,
Screen.height/2 -45, 100, 30), "Video Settings")){
                        options = false;
                        video = true;
                  }

                  if (GUI.Button(Rect(Screen.width/2 - 50,
Screen.height/2 + 10, 100, 30), "close")){
                        options = false;
                        penglihatan.enabled = true;
                        //menu = true;
                  }
            }

            if(sound){
                  GUI.Box(Rect(Screen.width/2-150,Screen.height/2-
150,300,300),"Sound Setting");
                  sfxVol = GUI.HorizontalSlider (Rect
(Screen.width/2 - 50, Screen.height/2, 100, 30), sfxVol, 0.0,
10.0);
                  GUI.Label(Rect(Screen.width/2 - 50 + 110,
Screen.height/2 - 5, 100, 30), "SFX: " + sfxVol);

                  musicVol = GUI.HorizontalSlider (Rect
(Screen.width/2 - 50, Screen.height/2 + 30, 100, 30), musicVol,
0.0, 10.0);
                  GUI.Label(Rect(Screen.width/2 - 50 + 110,
Screen.height/2 + 25, 100, 30), "Music: " + musicVol);
```

```
                  if (GUI.Button(Rect(Screen.width/2 - 50,
Screen.height/2 + 90, 100, 30), "Back")){
                        sound = false;
                        options = true;
                  }
            }


            if(video){
                  GUI.Box(Rect(Screen.width/2-150,Screen.height/2-
200,300,400),"Video Setting");
                  var qualities = QualitySettings.names;
                  GUILayout.BeginVertical ();
                  for (var i = 0; i < qualities.Length; i++){
                        if (GUI.Button(Rect(Screen.width/2 - 50,
Screen.height/2 - 120 + i * 30, 100, 30), qualities[i])){
                              QualitySettings.SetQualityLevel (i,
true);
                        }
                  }

                  GUILayout.EndVertical ();
                  fieldOfView = GUI.HorizontalSlider (Rect
(Screen.width/2 - 50,Screen.height/2 - 150,100,20), fieldOfView,
30, 120);
                  GUI.Label(Rect(Screen.width/2 - 50 + 110,
Screen.height/2 - 155, 100, 30), "FOV: " + fieldOfView);

                  if (GUI.Button(Rect(Screen.width/2 - 50,
Screen.height/2 + 90, 100, 30), "Back")){
                        video = false;
                        options = true;
                  }
            }
      }
}
```