PENERAPAN *MAXIMUM ELEMENT ALGORITHM* DAN *FINITE STATE MACHINE* PADA GAME "STRIVE"

SKRIPSI

Disusun Oleh :

MUHAMMAD KEVIN DESMAIZAL
12.18.255

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

2016

LEMBAR PERSETUJUAN

PENERAPAN *MAXIMUM ELEMENT ALGORITHM* DAN *FINITE STATE MACHINE* PADA GAME "STRIVE"

SKRIPSI

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

Disusun Oleh :

MUHAMMMAD KEVIN DESMAIZAL

NIM : 12.18.255

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Suryo Adi Wibowo, ST, MT	**Febriana Santi Wahyuni, S.kom, M.kom**
NIP.P 1031000438	**NIP.P 1031600425**

Mengetahui
Ketua Program Studi Teknik Informatika S-1

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

PROGRAM STUDI TEKNIK INFORMATIKA S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2016

LEMBAR KEASLIAN

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama : MUHAMMAD KEVIN DESMAIZAL

NIM : 12.18.255

Program Studi : Teknik Informatika S-1

Fakultas : Fakultas Teknologi Industri

Menyatakan dengan sesungguhnya bahwa Skripsi saya yang berjudul :

"Penerapan *Maximum Element Algorithm* Dan *Finite State Machine* Pada Game "Strive""

Adalah skripsi sendiri bukan duplikasi serta mengutip atau menyadur seluruhnya karya orang lain kecuali dari sumber aslinya.

Malang, Januari 2015

Yang membuat pernyataan

Muhammad Kevin Desmaizal

Abstrak

Game saat ini sudah menjadi alternatif hiburan bagi tua, muda, pria maupun wanita. Industri dan bisnis pengembangan game juga sudah menjadi suatu hal yang menjanjikan. Platform game adalah salah satu genre yang sudah mendunia, yang juga biasa disebut platformer adalah game yang dalam gameplay nya melibatkan perjalanan antar platform dengan cara meloncat (biasanya juga berayun dan memantul).

Metode Maximum Element Algorithm adalah suatu algoritma yang digunakan untuk mencari elemen/nilai terbesar dari suatu data array.Kemampuan kecerdasan buatan dalam mendukung sebuah game tidak dapat dipungkiri lagi. Salah satunya adalah penggunaan finite state machine untuk mendukung interaksi dengan pemain.

Dari hasil pengujian pada pengguna 11 dari 16 orang menyatakan gameplay game "Strive" menarik dengan presentase 68.75% dan sisa 5 dari 16 lainya menyatakan kurang menarik dengan presentase 31.25%. Untuk genre platformer 12 dari 16 orang menyatakan menarik dengan presentase 75%, 3 dari 16 menyatakan kurang menarik dengan presentase 18.75%, dan 1 dari 16 menyatakan tidak menarik dengan presentase 6.25%, dan untuk desain karakter 11 dari 16 menyatakan menarik dengan presentase 68.75%, 2 dari 16 menyatakan kurang menarik dengan presentase 18.75%, dan 2 dari 16 menyatakan tidak menarik dengan presentasi 12.5%

Kata Kunci : *Platform game, Maximum element algorithm, finite state machine, Unity*

KATA PENGANTAR

Dengan memanjatkan puji syukur kehadirat Allah Yang Maha Kuasa. Karena atas berkah rahmat dan karunia-Nya sehingga penyusun dapat menyelesaikan skripsi dengan judul PENERAPAN MAXIMUM ELEMENT ALGORITHM DAN FINITE STATE MACHINE PADA GAME "STRIVE", sesuai dengan waktu yang ditentukan.

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

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

1. Dr. Ir. Lalu Mulyadi, MT, selaku Rektor Institut Teknologi Nasional Malang.
2. Ir. Anang Subardi, MT, selaku Dekan Fakultas Teknik Industri, Institut Teknologi Nasional Malang.
3. Bapak Joseph Dedy Irawan, ST, MT, Ketua Program Studi Teknik Informatika S-1 Institut Teknologi Nasional Malang.
4. Bapak Sonny Prasetio, ST, MT, selaku sekertaris Program Studi Teknik Informatika S-1 Institut Teknologi Nasional Malang.
5. Suryo Adi Wibowo, ST, MT, selaku Dosen Pembimbing I, yang selalu memberi masukan, kritikan, dan saran.
6. Febriana Santi Wahyuni, S.kom, M.kom, selaku Dosen Pembimbing II, yang selalu memberi masukan, kritikan dan saran.

Penyusun menyadari bahwa skirpsi ini masih belum sempurna karena keterbatasan waktu dan pengetahuan yang penyusun miliki. Semoga skripsi ini bermanfaat bagi pembaca.

Malang, 2016

Penulis,

(Muhammad Kevin Desmaizal)

DAFTAR ISI

DAFTAR GAMBAR

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 tua, muda, pria maupun wanita. 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*, ini dilihat dari tingkat konsumsi *game* yang sangat tinggi, terutama *game* konsol, *Local Area Network (LAN)* dan *online*. Banyak perusahaan-perusahaan yang membawa *game-game* bagus dari luar negeri untuk dimainkan di Indonesia.

Platform game adalah salah satu *genre* yang sudah mendunia, yang juga biasa disebut *platformer* adalah *game* yang dalam *game*play nya melibatkan perjalanan antar platform dengan cara meloncat (biasanya juga berayun dan memantul). *Genre* ini biasanya dihubungkan dengan tokoh-okoh kartun seperti Sonic the hedgehog, Mario, dan rayman, walaupun mungkin mempunyai tema yang lainnya. Elemen-elemen tradisional dari *game* ini termasuk berlari, lompat, dan memanjat tangga atau pijakan. *Genre* ini seringkali meminjam elemen dari genre lain seperti *fighting* atau perkelahian dan *shooting* atau tembak-menembak.

Maka tidak dipungkiri bahwa dibutuhkan sesuatu yang berbeda pada rule permainan dalam suatu *game*. Hal ini sangat berkaitan dengan kecerdasan buatan (*artificial intelligence*) yang diterapkan pada *game*. Sebelumnya, sebuah sistem *game*, jika sudah dimainkan sampai tuntas oleh seorang , maka ketika player yang sama memulai lagi permainan dari awal, maka rule permainannya akan sama.

Namun, untuk saat ini sesuai dengan perkembangan game dan kecerdasan buatan yang diterapkan, sistem dalam game sudah dapat belajar mengenali pola permainan dari player dan ketika player tersebut memulai permainan kembali, maka sistem ini akan menggunakan rule yang berbeda untuk pemain yang sama ini, sehingga game menjadi lebih menarik dan menantang untuk dimainkan.

Metode Maximum Element Algorithm adalah suatu algoritma yang digunakan untuk mencari elemen/nilai terbesar dari suatu data *array*. Algoritma ini bekerja dengan cara membandingkan masing-masing elemen *array*. (Hidayat, 2009)

Kemampuan kecerdasan buatan dalam mendukung sebuah *game* tidak dapat dipungkiri lagi. Salah satunya adalah penggunaan *finite state machine* untuk mendukung interaksi dengan pemain. (Rotianingsih 2013)

Berdasarkan latar belakang diatas penulis membuat sebuah penerapan *Maximum Element Algorithm* dan *finite state machine* pada game "Strive".

1.2 Rumusan Masalah

Rumusan masalah yang dapat dikemukakan oleh penulis adalah sebagai berikut:

1. Bagaimana merancang *game platformer* ?
2. Bagaimana implementasi metode *Maximum Element Algorithm* pada karakter utama dalam game platformer ?
3. Bagaimana implementasi kecerdasan buatan *Finite State Machine* pada musuh dalam game platformer ?

1.3 Tujuan

Tujuan dari penyusunan skripsi berikut adalah:

a) Membangun sebuah *game platformer* menggunakan perangkat lunak Unity.
b) Mengimplementasikan metode *Maximum Element Algorithm* pada karakter utama dalam *game platformer*.
c) Mengimplementasikan kecerdasan buatan *Finite State Machine* musuh dalam *game platformer*.

1.4 Batasan Masalah

Dalam penyusunan skripsi agar menjadi sistematis dan mudah dimengerti, maka akan diterapkan beberapa batasan masalah. Batasan-batasan masalah itu antara lain :

BAB II
LANDASAN TEORI

2.1 *Video Game*

Permainan *video* atau *video game* adalah permainan yang menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh piranti *video*. Permainan *video* umumnya menyediakan sistem penghargaan – misalnya skor – yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan. Kata "*video*" pada "permainan *video*" pada awalnya merujuk pada piranti tampilan raster. Namun dengan semakin dipakainya istilah "*video game*", kini kata permainan *video* dapat digunakan untuk menyebut permainan pada piranti tampilan apapun. Sistem elektronik yang digunakan untuk menjalankan permainan *video* disebut platform, contohnya adalah *PC* atau *Console*. (Apperley, 2012)

Genre atau ragam permainan *video* digunakan untuk menggolongkan permainan *video* berdasarkan interaksi bidang permainannya, bukan hanya perbedaan visual maupun naratif. Permainan *video* diklasifikasikan independen berdasarkan pengaturan atau isi dari dunia permainan tersebut, tidak seperti film ataupun buku. (Apperley, 2012)

Berikut ini adalah daftar umum yang digunakan untuk menentukan *genre* dari sebuah permainan *video*. Daftar ini tidak berarti lengkap karena pada dasarnya permainan *video* berubah dengan cepat. Sebuah permainan biasanya juga memiliki lebih dari satu *genre* (*Hybrid-genre*). *Genre* sebenarnya dari sebuah permainan terbuka bagi para setiap individual untuk menentukan. (Apperley, 2012)

Contoh *genre-genre* dari game adalah (Apperley, 2012) :

a. Racing

Dimana pemain menggunakan kendaraan untuk memenangkan balapan. Tujuan dalam game pada genre ini biasanya adalah untuk mendapatkan kendaraan baru atau mendapatkan *trophy* /piala dalam kejuaraan kejuaraan.

b. RPG(Role-Playing Game)

Salah satu game yg mengandung unsur experience atau leveling dalam gameplay nya. Biasanya dalam game ini kita memiliki kebebasan untuk menjelajah dunia game tersebut, dan kadang kala dalam beberapa game, kita dapat menentukan ending dari game tersebut.

c. Platformer

Sebuah permainan platform (atau platformer) adalah permainan video yang ditandai dengan mengharuskan pemain untuk melompat ke dan dari platform ditangguhkan atau hambatan atas (melompat teka-teki).

d. Strategi

Dimana pemain menjalankan berbagai unit-unit yang unik untuk memenangkan permainan tersebut. Gameplay nya biasanya kita mengatur unit atau pasukan untuk bertahan, menahan, bahkan mengalahkan musuh yang ada di game tersebut. Pemain juga membangun pertahanan, bangunan, dan pasukan dengan biaya tertentu yang dapat di dapatkan dari sumber sumber yang ada di dalam map pertempuran. Game strategy ada yang berbasis Real Time Strategy dan ada yang Turn Based Strategy.

e. Simulation

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

f. Puzzle

Berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kota masuk ke tempat yang scharusnya, itu semua termasuk dalam jenis ini.

g. FPS(First Person Shooter)

First Person Shooter merupakan genre game yang dalam permainannya mengusung tema tembak-tembakan atau perang.

h. Fighting

Genre game bertarung. Seperti dalam arcade, pemain dapat mengeluarkan jurus-jurus ampuh dalam pertarungannya. Genre fighting biasanya one on one dalam sebuah arena yang sempit.

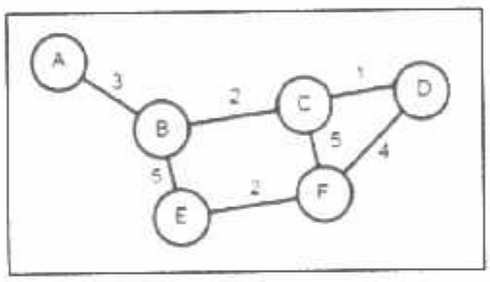
2.2 *AI(Artificial Intelligence)* Pada *Game*

Kecerdasan buatan (*Artificial Intelligence*) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu *entitas* buatan. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. (Martiana 2005)

Berikut adalah contoh AI yang ada pada game :

a. *Pathfinding*

Algoritma A* adalah jenis algoritma pencarian rute yang dapat mencari rute terpendek antara 2 titik pada *weighted graph*. Pada *weighted graph*,rute terpendek sama dengan rute dengan nilai (*cost*) terkecil. (Riwinoto, 2014)

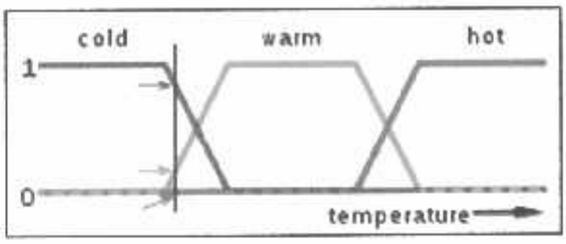


Gambar 2.1 Weighted graph (Riwinoto, 2014)

Algoritma A* menggunakan metode heuristic untuk mengurangi area pencarian dimana hanya area yang menjanjikan saja yang akan diuji. Secara umum *pathfinding* dapat dibedakan menjadi pathfinding statik dan dinamik. (Riwinoto, 2014)

b. *Fuzzy Logic*

Logika *Fuzzy* adalah peningkatan dari logika Boolean yang berhadapan dengan konsep *kebenaran sebagian*. Saat logika klasikmenyatakan bahwa segala hal dapat diekspresikan dalam istilah biner (0 atau 1, hitam atau putih, ya atau tidak). (Sugiarto 2012)



Gambar 2.2 Fuzzy logic (Sugiarto 2012)

Logika *Fuzzy* memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan", dan "sangat". Logika ini berhubungan dengan set *fuzzy* dan teori kemungkinan. (Sugiarto 2012)

c. *Minimax*

Pada algoritma *minimax*, pengecekan akan seluruh kemungkinan yang ada sampai akhir permainan dilakukan. Pengecekan tersebut akan menghasilkan pohon permainan yang berisi semua kemungkinan tersebut. (Setiadi 2012)

Komputer akan memilih langkah yang paling membuat lawan mendapatkan keuntungan minimum, dan yang paling membuat komputer itu sendiri mendapatkan keuntungan maksimum. (Setiadi 2012)

Berikut garis besar algoritma *minimax* secara umum :

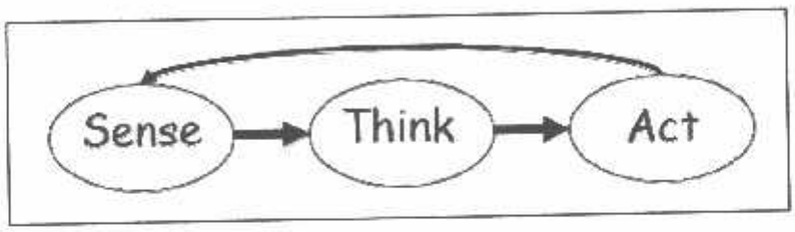
```
Cari langkah yang dengan nilai maksimum
IF langkah tersebut merupakan langkah kemenangan
   THEN pilih lagkah tersebut.
ELSE
   FOR EACH kemungkinan langkah yang ada
        Cari langkah lawan yang bernilai minimum.
        RETURN nilai dari langkah tersebut.
        Pilih langkah yang bernilai maksimum darilangkah-langkah tersebut
```

Algoritma 2.1 Minimax (Setiadi 2012)

d. Game Agent

Kecerdasan buatan ini berfokus pada *agent* dalam game, yang dimana *agent* ini bisa disebut NPC(*non-playable character*), teman, musuh, atau netral bagi pemain. (Claypool, 2007)

Akan bergerak sesuai perulangan *sense*(merasakan)-*think*(berpikir)-*act*(bergerak), saat terjadi kejadian yang spesifik maka akan menjalakan *sense* atau mengambil informasi yang didapat, lalu *think* atau mengolah informasi yang didapat, kemudian *act* atau bergerak sesuai informasi yang telah diolah. (Claypool, 2007)



Gambar 2.3 Game agents (Claypool, 2007)

e. *Filtered randomness*

Kecerdasan buatan yang akan mengeluarkan nilai *random* atau acak yang tidak bisa diprediksi. Pada algoritma nya akan mengambil hasil nilai-nilai sebelumnya dan akan dibandingkan dengan nilai yang didapat, jika nilai sama sampai batas yang ditentukan maka nilai akan diambil lagi secara acak. (Claypool, 2007)

Misalkan sebuah koin dilempar sebanyak 10 kali dan tidak boleh ada yang sama sebanyak 5 kali, maka akan menghasilkan masing-masing sisi sebanyak 5 kali. (Claypool, 2007)

f. N-Gram statistical prediction

Algoritma yang memprediksi *value* atau nilai yang akan dikeluarkan oleh pemain dengan membaca nilai sebelum-sebelumnya yang dikeluarkan oleh pemain. Misalkan nilai yang dikeluarkan pemain secara berkala adalah 7-3-8-7-3-8-7-3-...., maka yang diprediksi akan keluar selanjutnya adalah 8. (Claypool, 2007)

BAB III
PERANCANGAN GAME

3.1. Analisa

Analisa atau analisis itu sendiri adalah suatu usaha untuk mengamati secara detail sesuatu hal atau benda dengam cara menguraikan komponen-komponen pembentukannya atau penyusunnya untuk dikaji lebih lanjut. Pada bagian ini akan menjelaskan analisa kebutuhan game, dan kebutuhan non-fungsional yang ada pada skripsi penerapan *Maximum Element Algorithm* dan *Finite State Machine* pada game "Strive".

3.1.1. Analisa Kebutuhan Game

Dalam pembuatan game ini dilakukan analisa apa saja fitur yang ada pada game, *resource* yang diperlukan, *AI yang* diterapkan, perangkat lunak dan perangkat keras yang digunakan.

3.1.2. Kebutuhan Non-Fungsional

Merupakan kebutuhan diluar kebutuhan fungsinal sistem yang meliputi analisa kebutuhan perangkat keras, dan perangkat lunak.

1. **Analisa Kebutuhuhan Perangkat Keras**

Perangkat keras yang dibutuhkan untuk menjalakan game ini antara lain :

Tabel 3.1 Analisa kebutuhan perangkat keras

Perangkat keras	Spesifikasi
Prosesor	Pentium 4 2.40 GHz
RAM	512mb
VGA	Intel HD Graphic 3000
Monitor	
Mouse & Keyboard	

2. Analisa Kebutuhan Perangkat Lunak

Perangkat keras tidak berarti tanpa perangkat lunak begitu juga sebaliknya Jadi perangkat lunak dan perangkat keras saling mendukung satu sama lain Perangkat keras hanya berfungsi jika diberikan instruksi-intruksi kepadanya. Untuk menjalankan game ini digunakan beberapa perangkat lunak, antara lain :

a) Sistem Operasi Windows, Machintosh, atau Linux

3.2. Perancangan

Definisi perancangan adalah penggambaran, perancanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi. Berikut adalah perancangan apa saja yang ada pada skripsi penerapan *Maximum Element Algorithm* dan *Finite State Machine* pada game "Strive".

3.2.1. Gameplay

a) Pada *level* 0, 1, dan 2 pemain hanya bisa menghindari diri dari monster agar bisa bertahan hidup dan akan ada *platform* atau pijakan dimana pemain bisa untuk melompat atau menuruni platform untuk mencari *NPC*(*non-playable character*) yang akan memberi pemain pertanyaan agar *portal* atau gerbang bisa digunakan untuk mengantar pemain ke *level* berikutnya.

b) Di *level* 3, dan 4 pemain akan mendapatkan senjata sesuai dengan pertanyaan yang dijawab oleh pemain pada *level- level* sebelumnya yang bisa digunakan untuk menghancurkan *crystal* sesuai senjata untuk membuka *portal* agar bisa melanjutkan ke *level* berikutnya.

c) Di *level* 5 pemain akan di hadapkan oleh *boss*, dimana serangan dan ketahanannya lebih kuat dibanding monster biasa. Jika *boss* telah dikalahkan maka pemain akan mendapatkan bonus score.

d) *Potion* dan *Live Potion* akan ada disetiap *level* untuk membantu pemain mendapat tambahan *HP*(*hitpoint*), *Lives*, dan skor, tapi beberapa akan ditempatkan di *platform* yang ada sedikit berbeda dari platform utama.

3.2.2. Karakter

Berikut adalah karakter atau objek yang ada di dalam game beserta nama, gambar, dan keterangan dari karakter atau objek tersebut. Agar mempermudah dalam pemahaman secara mendetail. Pada tabel 3.2.

Tabel 3.2 Karakter/objek pada game

No	Karakter/Objek	Gambar	Keterangan
1	Pemain		Karakter utama yang dijalankan
2	Monster - Glitch		Monster akan mengejar pemain jika jarak antar pemain dan monster cukup dekat. Jika pemain ada kontak dengan monster, maka *hitpoint* pemain berkurang. memiliki 3 *hitpoint*.
3	Monster – Glitch Eye		Monster melayang akan mengejar pemain jika jarak antar pemain dan monster dekat. Jika pemain ada kontak dengan monster maka *hitpoint pemain berkurang*. memilki 4 *hitpoint*.
4	Boss Monster - Kabuto		ketika aktif bergerak kekiri dan kekanan, menembakkan bola api, kebal dari serangan pemain, jika pemain ada kontak dengan monster maka *hitpoint* pemain berkurang. Hanya bisa diserang ketika sedang tidak aktif. memilki 50 *hitpoint*.

No	Karakter/Objek	Gambar	Keterangan
5	Trap		Jebakan yang akan naik ketika jarak pemain mulai dekat dan hanya turun ketika turun ketika jarak pemain mulai jauh.
6	Potion		Menambahkan *hitpoint* pemain sebanyak +10
7	Lives Potion		Menambahkan *lives* pemain sebanyak +1 dan menambahkan.
8	Orb		NPC(*non-playable character*) yang bertugas untuk menanyakan pertanyaan kepada pemain untuk mengaktifkan *portal* hanya pada *level* 0, 1, 2
9	Red Crystal		*Crystal* yang hanya bisa dihancurkan dengan senjata pedang untuk membuka *portal* pada *level* 3, 4, 5
10	Green Crystal		*Crystal* yang hanya bisa dihancurkan dengan senjata panah untuk membuka *portal* pada *level* 3, 4, 5
11	Portal		Berfungsi untuk memindahkan pemain ke *level* selanjutnya

3.2.3 Terrain

Berikut adalah medan atau *terrain* apa saja ada pada game termasuk nama dan keterangan :

Tabel 3.3 Terrain pada game

No	Terrain	Gambar	Keterangan
1	Ground		*Ground* atau dasar lantai yang dapat dilewati oleh pemain
2	Platform		*Platform* atau pijakan melayang yang bisa dilewati oleh pemain
3	Red Platform		*Platform* atau pijakan melayang yang bisa dilewati oleh pemain namun jika bersentuhan dengan pemain maka pijakan ini akan mengilang dalam beberapa saat
4	Background level 0		Latar belakang level 0 pada game
5	Background level 1		Latar belakang level 1 pada game
6	Background level 2		Latar belakang level 2 pada game

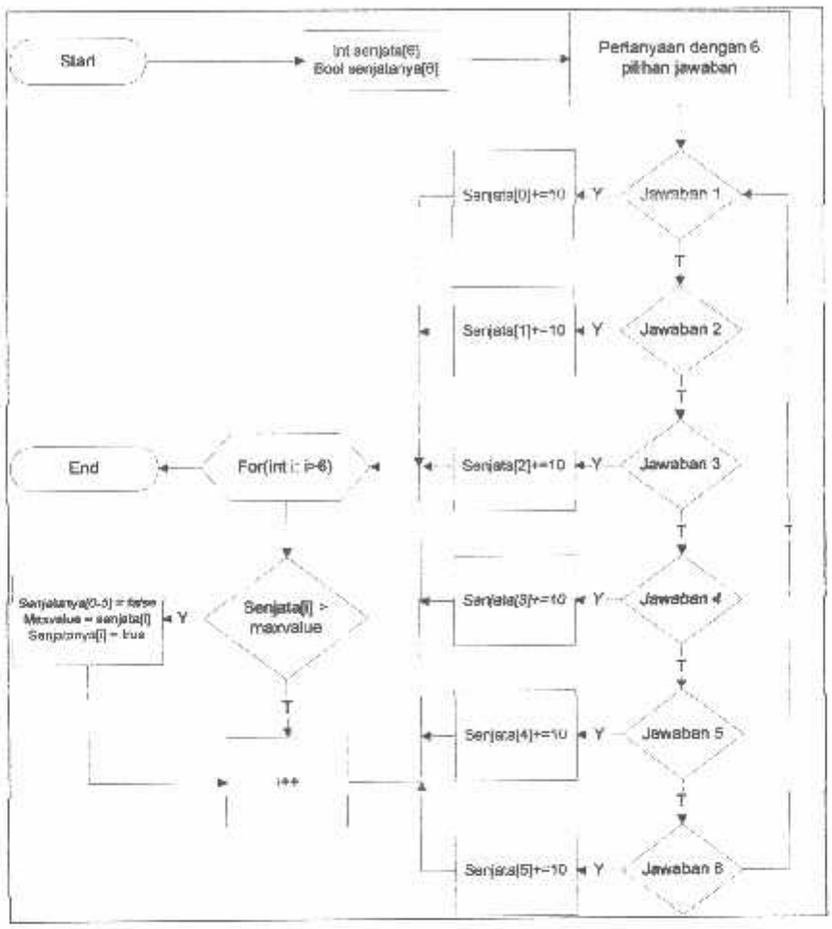
No	Terrain	Gambar	Keterangan
7	Background level 3		Latar belakang level 3 pada game
8	Background level 4		Latar belakang level 4 pada game
9	Background level 5		Latar belakang level 5 pada game
10	Background ending		Latar belakang ending pada game

3.2.4. Perancangan AI(Artificial Intelligence)

a) Perancangan Flowchart *Maximum Element Algorithm* untuk Karakter Utama

Perancangan *AI(Artficial Intelligence) Maximum Element Algortihm* dengan pertanyaan yang akan dijawab oleh pemain, memiliki nilai yang berbeda lalu akan dimasukan dalam sebuah *array* di masing-masing index dan dilakukan pencarian nilai index tertinggi dari *array* tersebut.

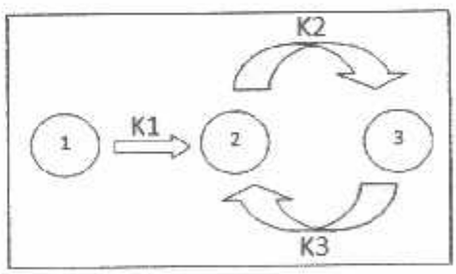
Melalui nilai index tertinggi itu pemain akan diberikan senjata yang berbeda sesuai dengan pertanyaan yang dijawab. Flowchart dapat dilihat pada gambar 3.1.



Gambar 3.1 Flowchart maximum elements algorithm pada karakter utama

b) Perancangan Finite State Machine Pada Monster Glitch

Perancangan *AI(Artificial Intelligence)* pada *monster glitch* menggunakan metode Finite State Machine, yang akan bergerak sesuai dengan pergerakan yang dilakukan oleh karakter utama.



Gambar 3.2 Penerapan Finite State Machine pada monster glitch

Berikut adalah *state* dan kondisi yang ada pada percangan *FSM* pada *monster glitch* :

State 1 : Monster akan bergerak kekiri menambah posisi X = 3.28 kemudian berhenti 2 detik lalu bergerak kekanan posisi X sebelumnya.

State 2 : Monster akan mendekati posisi pemain.

State 3 : Monster akan menyerang pemain sampai *hitpoint* pemain atau monster mencapai 0.

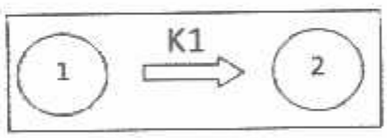
Kondisi K1 : Jika pemain mendekati posisi X = 1.79 dan Y = 1.12 dari monster glitch.

Kondisi K2 : Jika pemain mendekati posisi X =1.09 dan Y = 1.11 dari monster glitch.

Kondisi K3 : Jika pemain menjauh dari monster glitch.

c) Perancangan Finite State Machine Pada Monster Glitch Eye

Perancangan *AI(Artificial Intelligence)* pada *monster glitch eye* menggunakan metode FSM atau *Finite State Machine*, dimana pergerakan *monster* akan berubah sesuai *state* yang dilihat juga dari pergerakan yang dilakukan oleh karakter utama.



Gambar 3.3 Penerapan Finite State Machine pada monster glitch eye.

Berikut adalah *state* dan kondisi yang ada pada percangan *FSM* pada *monster glitch eye* :

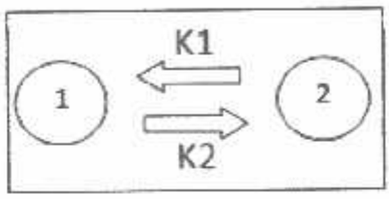
State 1 : Monster diam disatu tempat.

State 2 : Monster akan mendekati posisi pemain.

Kondisi K1 : Jika pemain mendekati posisi X = 0.74 dan Y = 5.42 dari *monster glitch eye*.

d) Perancangan Finite State Machine Pada *Trap*

Perancangan *AI(Artificial Intelligence)* pada *trap* menggunakan metode *Finite State Machine*, dimana trap akan naik sesuai dengan pergerakan yang dilakukan oleh karakter utama.



Gambar 3.4 Penerapan Finite State Machine pada trap

Berikut adalah *state* dan kondisi yang ada pada percangan *FSM* pada *trap* :

State 1 : *Trap* turun dengan mengurangi posisi Y = 0.78.

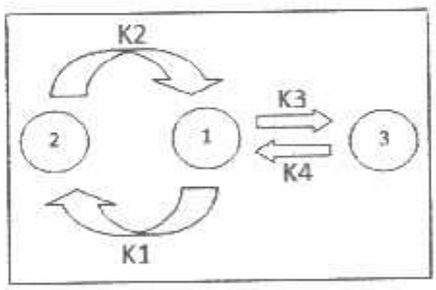
State 2 : *Trap* naik dengan menambah posisi Y = 0.78.

Kondisi K1 : Jarak pemain mendekati posisi X = 1.46 dan Y = 0.88 dari *trap*.

Kondisi K2 : Jarak pemain menjauh posisi X = 1.46 dan Y = 0.88 dari *trap*.

e) Perancangan Finite State Machine Pada Boss Monster Kabuto

Perancangan *AI(Artificial Intelligence)* pada boss monster kabuto menggunakan metode *Finite State Machine*, yang akan bergerak sesuai dengan pergerakan yang dilakukan oleh karakter utama.



Gambar 3.5 Penerapan Finite State Machine pada boss monster kabuto

Berikut adalah *state* dan kondisi yang ada pada percangan *FSM* pada *boss monster kabuto* :

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Hasil

Berikut tahapan hasil implementasi *game* "Strive" menggunakan *engine* Unity. Tampilan yang terdapat pada game yaitu tampilan menu utama, tentang, petunjuk permainan, *level* 0, *level* 1, *level* 2, *level* 3, *level* 4, dan *level* 5(*Boss*). Tampilan-tampilan tersebut adalah sebagai berikut :

4.1.1 Tampilan Menu Utama

Tampilan menu utama adalah tampilan yang muncul pada saat membuka *game* "Strive". Pada saat tampilan terdapat pilihan Mulai, Tentang, dan Keluar. Seperti pada gambar 4.1.

Gambar 4.1 Tampilan menu utama

4.1.2 Tampilan Tentang

Tampilan tentang adalah tampilan berisikan tentang informasi dari *game* "Strive" secara tertulis, dan jika pilihan ke menu utama dipilih maka akan berpindah ke menu utama. Seperti pada gambar 4.2.

Gambar 4.2 Tampilan tentang

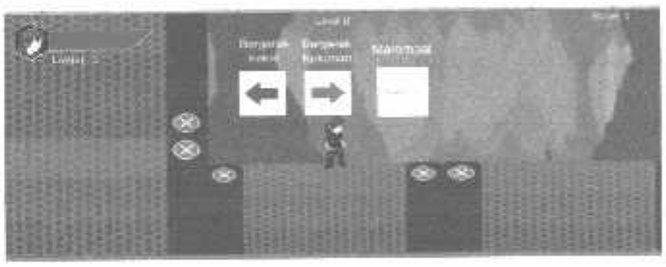
4.1.3 Tampian Petunjuk Permainan

Tampilan petunjuk permainan adalah tampilan berisikan tentang petunjuk dan kontrol pada dari game "Strive" secara tertulis, dan jika pilihan ke menu utama dipilih maka akan berpindah ke menu utama. Seperti pada gambar 4.3.

Gambar 4.3 Tampilan petunjuk permainan

4.1.4 Tampilan *Start Game*

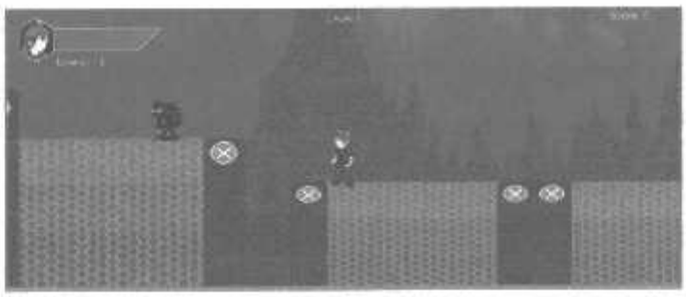
Tampilan *start game* dimulai pada *level* 0 akan ditampilkan ketika tombol mulai dipilih dari menu utama. Dari level ini pemain akan diberitahu dasar-dasar kontrol dan tujuan dari permainan. Setelah menjawab pertanyaan dari *NPC(non-playable character)* maka *portal* akan terbuka dan pemain bisa melanjutkan ke level selanjutnya. Tampilan terlihat pada gambar 4.4.



Gambar 4.4 Tampilan level 0

4.1.5 Tampilan *Level* 1

Tampilan level 1 akan ditampilkan ketika pemain telah melewati *level* sebelumnya. Skor dan *lives* yang dimiliki akan tetap sama dari level sebelumnya. Pemain harus mencari dan menjawab pertanyaan dari *NPC(non-playable character)* maka *portal* akan terbuka untuk melanjutkan ke *level* berikutnya. Tampilan terlihat pada gambar 4.5.



Gambar 4.5 Tampilan level 1

4.1.6 Tampilan *Level* 2

Tampilan *level* 2 akan ditampilkan ketika pemain telah melewati *level* sebelumnya. Skor dan *lives* yang dimiliki akan tetap sama dari level sebelum nya. Pemain harus mencari dan menjawab pertanyaan dari *NPC(non-playable character)* maka *portal* akan terbuka untuk melanjutkan ke *level* berikutnya. Tampilan terlihat pada gambar 4.6.

Gambar 4.6 Tampilan level 2

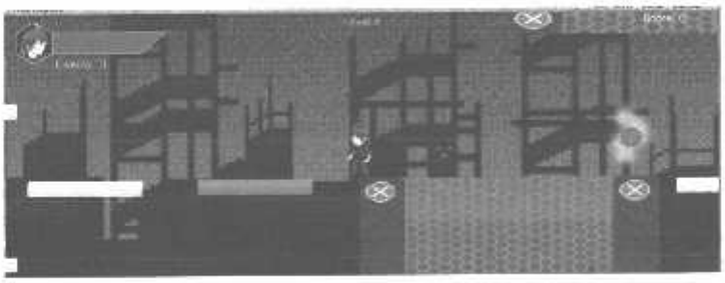
4.1.7 Tampilan *Level* 3

Tampilan level 3 akan ditampilkan ketika pemain telah melewati *level* sebelumnya. Skor dan *lives* yang dimiliki akan tetap sama dari *level* sebelumnya namun pada *level* ini monster bertambah satu jenis dan jumlahnya semakin banyak dibanding *level* sebelumnya. Pemain mendapatkan senjata sesuai dengan banyaknya nilai dari pertanyaan yang dijawab di *level-level* sebelumnya yang bisa digunakan untuk melawan *monster* untuk mendapat skor tambahan dan menghancurkan *crystal* sesuai senjata yang didapat agar *portal* bisa terbuka schingga pemain bisa melanjutkan ke *level* berikutnya. Tampilan terlihat pada gambar 4.7.

Gambar 4.7 Tampilan level 3

4.1.8 Tampilan *Level* 4

Tampilan *level* 4 akan ditampilkan ketika pemain telah melewati *level* sebelumnya. Skor dan *lives* yang dimiliki akan tetap sama dari *level* sebelum nya. Pemain bisa menggunakan senjata yang bisa digunakan untuk melawan monster dan menghancurkan *crystal* sesuai senjata yang didapat agar *portal* bisa terbuka sehingga bisa melanjutkan ke *level* berikutnya. Tampilan terlihat pada gambar 4.8.



Gambar 4.8 Tampilan level 4

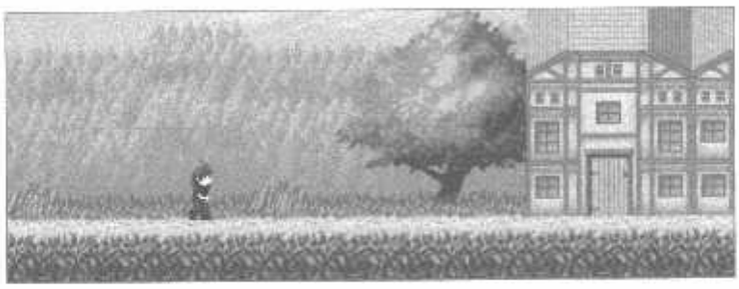
4.1.9 Tampilan *Level* 5(*Boss*)

Tampilan *level* 5(*Boss*) akan ditampilkan ketika pemain telah melewati *level* sebelumnya. Skor dan *lives* yang dimiliki akan tetap sama dari *level* sebelumnya namun pada *level* ini *monster* hanya ada *monster boss kabuto* yang memiliki ketahanan yang lebih kuat dan pergerakan yang berbeda dengan *monster* atau *trap* yang ada di*level-level* sebelumnya. Pemain bisa menggunakan senjata yang didapat dari *level-level* sebelumnya bisa untuk melawan *boss monster* hingga kalah dan melewati *portal* menyelesaikan *game*. Tampilan terlihat pada gambar 4.9.

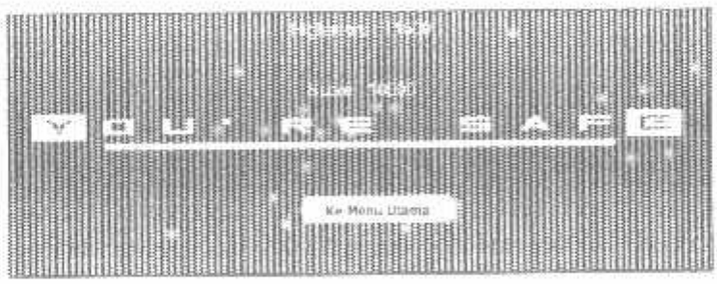
Gambar 4.9 Tampilan level 5(boss)

4.1.10 Tampilan Ending

Tampilan *Ending* akan ditampilkan ketika pemain berhasil mengalahkan *boss* dan melewati portal. akan kembali ke menu utama ketika pilihan ke menu utama dipilih. Tampilan terlihat pada gambar 4.10 dan gambar 4.11.



Gambar 4.10 Tampilan ending



Gambar 4.11 Tampilan setelah ending

4.1.11 Tampilan *Game Over*

Tampilan *Game Over* akan ditampilkan ketika pemain kehabisan *lives* karena kehabisan *hitpoint* akibat serangangan monster, trap, ataupun jatuh ke lubang. Dan akan kembali ke menu utama ketika pilihan ke menu utama dipilih. Tampilan terlihat pada gambar 4.12.

Gambar 4.12 Tampilan game over

4.2 Pengujian Sistem

Pengujian sistem dilakukan apakah game berjalan tanpa error didalam game. Saat menjalankan game yang terdiri dari fungsional, pengujian pada pengguna pada pengguna fungsional, pengujian *AI*, pengujian performa, pengujian control player, dan pengujian pengguna.

4.2.1 Pengujian Fungsional

Pengujian fungsional adalah pengujian mengenai proses fungsional yang ada dalam *game*. Hasil dari pengujian dapat dilihat pada Tabel 4.1.

Tabel 4.1 Pengujian fungsional

No.	Fungsi	Output	Hasil
1	Pilihan mulai pada menu utama	Link yang dituju sesuai	Sesuai
2	Pilihan tentang pada menu utama	Link yang dituju sesuai	Sesuai
3	AI Finite State Machine pada monster glitch	Monster glitch akan mengejar pemain jika memasuki radius dan akan menyerang jika radius semakin dekat	Sesuai
4	AI Finite State Machine pada monster glitch eye	Monster glitch eye akan mengejar pemain jika memasuki radius	Sesuai

4.2.2 Pengujian AI(Artificial Intelligence)

Pengujian AI(Artificial Intelligence) adalah pengujian mengenai fungsi yang berkaitan dengan AI(Artificial Intelligence) yang ada dalam game. Hasil dari pengujian AI maximum elemenet algorithm dapat dilihat pada tabel 4.2 dan hasil pengujian dari AI finite state machine dapat dilihat pada tabel 4.3.

Tabel 4.2 pengujian AI maximum element algorithm

No.	Fungsi	Output	Hasil
1	AI Maximum Element Algorithm Senjata 1 pada karakter utama	Senjata akan diberikan untuk pemain sesuai nilai index array tertinggi. jika nilai index [0] daripada index lainnya maka senjata pedang akan diberikan.	Sesuai
2	AI Maximum Element Algorithm Senjata 2 pada karakter utama	Senjata akan diberikan untuk pemain sesuai nilai index array tertinggi. jika nilai index [1] daripada index lainnya maka senjata panahan akan diberikan.	Sesuai
3	AI Maximum Element Algorithm Senjata 3 pada karakter utama	Senjata akan diberikan untuk pemain sesuai nilai index array tertinggi. jika nilai index [2] daripada index lainnya maka senjata palu akan diberikan.	Sesuai
4	AI Maximum Element Algorithm Senjata 4 pada karakter utama	Senjata akan diberikan untuk pemain sesuai nilai index array tertinggi. jika nilai index [3] daripada index lainnya maka senjata pistol akan diberikan.	Sesuai
5	AI Maximum Element Algorithm Senjata 5 pada karakter utama	Senjata akan diberikan untuk pemain sesuai nilai index array tertinggi. jika nilai index [4] daripada index lainnya maka senjata tombak akan diberikan.	Sesuai

No.	Fungsi	Output	Hasil
6	AI Maximum Element Algorithm Senjata 6 pada karakter utama	Senjata akan diberikan untuk pemain sesuai nilai index array tertinggi. jika nilai index [5] daripada index lainnya maka senjata keris akan diberikan.	Sesuai

Tabel 4.3 pengujian AI finite state machine

No.	Fungsi	Output	Hasil
1	AI Finite State Machine pada monster glitch	Monster glitch akan mengejar pemain dengan jika memasuki radius X = 1.79, Y = 1.12 dan akan menyerang jika radius X = 1.10, Y = 1.11 dari sekitar monster glitch	Sesuai
2	AI Finite State Machine pada monster glitch eye	Monster glitch eye akan mengejar pemain jika memasuki radius X = 0.75, Y = 5.42 dari sekitar monster glitch eye	Sesuai
3	AI Finite State Machine pada boss monster glitch kabuto	Ketika aktif monster glitch kabuto akan bergerak kekiri dan kekanan sambil mengeluarkan bola api yang mengejar pemain dan tidak bisa diserang dengan senjata selama 30 detik. Ketika tidak aktif monster glitch kabuto akan berhenti bergerak di tengah dan bisa diserang selama 15 detik.	Sesuai
4	AI Finite State Machine pada trap	Jika pemain mendekati radius X = 6, Y = 1.29 maka trap akan naik dan hanya akan turun jika pemain menjauh dari radius tersebut	Sesuai

No.	Tombol	Fungsi	Output
5	Z	Melanjutkan dialog	Text pada dialog box dilanjutkan(Sesuai)
6	A	*Menanyakan* objek	*Pemain menanyakan/memulai* dialog dengan objek(Sesuai)
7	X	Menyerang	Menggunkan senjata untuk menyerang(pada level tertentu)

Tabel diatas menunjukan bahwa semua tombol dari control player berjalan sesuai dengan fungsi yang diinginkan.

4.2.5 Pengujian Pada Pengguna

Pengujian pada pengguna ditujukan kepada pengguna/*user* yang memainkan *game* "Strive".terdapat kriteria dan penilaian pada pengujian terdapat pada Tabel 4.6.

Tabel 4.6 Pengujian pada pengguna

No	Kriteria	Penilaian		
		Menarik	Kurang menarik	Tidak menarik
1	Gameplay	68.75%	31.25%	0%
2	Genre platformer pada game	75%	18.75%	6.25%
3	Desain karakter dan background	68.75%	18.75%	12.5%

Dari tabel 4.5 dapat disimpulkan bahwa 11 dari 16 orang menyatakan gameplay game "Strive" menarik dengan presentase 68.75% dan sisa 5 dari 16 lainya menyatakan kurang menarik dengan presentase 31.25%. Untuk genre platformer 12 dari 16 orang menyatakan menarik dengan presentase 75%, 3 dari 16 menyatakan kurang menarik dengan presentase 18.75%, dan 1 dari 16 menyatakan tidak menarik dengan presentase 6.25%, dan untuk desain karakter 11 dari 16 menyatakan menarik dengan presentase 68.75%, 2 dari 16 menyatakan kurang menarik dengan presentase 18.75%, dan 2 dari 16 menyatakan tidak menarik dengan presentasi 12.5%.

BAB V
PENUTUP

5.1. Kesimpulan

Dari hasil penelitian penerapan *maximum element algorithm* dan *finite state machine* pada *game* "Strive" kesimpulan yang dapat diuraikan oleh penulis adalah sebagai berikut :

1. Dari hasil pengujian fungsional pada game "Strive", pada masing-masing fungsi berjalan sesuai dengan output, tidak ada error ataupun salah pada fungsi.
2. Dari hasil pengujian *AI(Artificial Intelligence)* pada game "Strive", dinyatakan sesuai dengan output, tidak ada error ataupun salah pada fungsi.
3. Dari hasil pengujian performa dinyatakan game dijalankan dikomputer dengan spesifikasi RAM minimal 1GB, VGA minimal 512MB, dengan OS Windows XP, 7, dan 8.1 berjalan dengan lancar. Dan dapat disimpulkan bahwa game "Strive" dapat dimainkan dengan processor minimal celeron, ram minimal 1 GB, VGA minimal 512MB, dan OS minimal XP.
4. Dari hasil pengujian *control* gerak dari player, bahwa semua tombol dari *control player* berjalan sesuai dengan fungsi yang diinginkan.
5. Dari hasil pengujian pada pengguna 11 dari 16 orang menyatakan gameplay game "Strive" menarik dengan presentase 68.75% dan sisa 5 dari 16 lainya menyatakan kurang menarik dengan presentase 31.25%. Untuk genre platformer 12 dari 16 orang menyatakan menarik dengan presentase 75%, 3 dari 16 menyatakan kurang menarik dengan presentase 18.75%, dan 1 dari 16 menyatakan tidak menarik dengan presentase 6.25%, dan untuk desain karakter 11 dari 16 menyatakan menarik dengan presentase 68.75%, 2 dari 16 menyatakan kurang menarik dengan presentase 18.75%, dan 2 dari 16 menyatakan tidak menarik dengan presentasi 12.5%.

5.2. Saran

Dalam skripsi penerapan *Maximum Element Algorithm* dan *Finite State Machine* pada *game* "Strive" ini masih memiliki kekurangan-kekurangan. berikut adalah saran yang dapat dikemukakan oleh penulis :

1. Game dikembangkan untuk versi Mobile dengan OS Android dan iOS.
2. Menambahkan AI(Artificial Intelligence) dengan metode *A*/pathfinding* yang dapat diterapkan pada musuh sehingga musuh bisa mengejar dengan menghitung jarak terdekat
3. Menambahkan AI(Artificial Intelligence) dengan metode *Filtered randomness* yang dapat diterapkan pada *Trap* sehingga penempatannya acak dan diposisi yang selalu berbeda untuk menambah tingkat kesulitan game.

DAFTAR PUSTAKA

Apperley, Thomas H. "Genre and *Game* Studies: Toward a Critical Approach to *Video Game* Genres", 2012, University of Melbourne.

Blackman, Sue. "Beginning 3D *Game* Development with Unity 4", 2013, Technology in Action.

Claypool, Mark. "Artificial Intelligence For Games", 2007, Worcester Polytechnic Institute. Diambil dari *http://web.cs.wpi.edu/~imgd4000/d07/slides/AI.ppt*.

Hidayat. "Petualangan Organisme", 2009, Institut Teknologi Nasional Malang.

Martiana, Entin. "Modul Ajar Kecerdasan Buatan", 2005, Politeknik Elektronika Negeri Surabaya.

Riwinoto. "Implementasi Pathfinding Dengan Algoritma A* Pada Game Funny English menggunakan Unity 3D Berbasis Graf Navmesh", 2014, Politeknik Negeri Batam.

Rostianingsih, Silvia. "*Game* Simulasi Finite State Machine Untuk Pertanian Dan Peternakan", 2013, Universitas Kristen Perta.

Setiadi, De Rosal Ignatius Moses. "Implementasi Algoritma Minimax Untuk Artificial Intellegence Pada Permainan Catur Sederhana", 2012, Universitas Dian Nuswantoro.

Sugiarto, Yusron. "Logika Fuzzy", 2012, Universitas Brawijaya. Diambil dari *yusronsugiarto.lecture.ub.ac.id/files/2012/04/logika-fuzzy.ppt.*

LAMPIRAN

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : Muhammad Kevin Desmaizal
NIM : 1218255
JURUSAN : Teknik Informatika S-1
JUDUL : Penerapan Maximum Element Algorithm Dan Finite State Machine Pada Game "Strive"

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :
Hari : Sabtu
Tanggal : 16 Januari 2016
Nilai : 84.75 (A)

Panitia Ujian Skripsi :

Ketua Majelis Penguji

Joseph Dedy Irawan, ST, MT
NIP. 197404162005011002

Anggota Penguji :

Dosen Penguji I

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

Dosen Penguji II

Karina Auliasari, ST., M.Eng.
NIP.P. 1031000426

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata 1 Program Studi Teknik Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Muhammad Kevin Desmaizal
NIM : 1218255
JURUSAN : Teknik Informatika S-1
JUDUL : Penerapan Maximum Element Algorithm Dan Finite State Machine Pada Game "Strive"

No	Penguji	Tanggal	Uraian	Paraf
1.	Penguji I	16 Januari 2016	1. Data aktual dilampirkan dan tambah user pengujian	
2.	Penguji II	16 Januari 2016	1. Landasan teori harus diberi sitasi. 2. Kesimpulan diperbaiki 3. Daftar pustaka harus diberi referensi	

Dosen Penguji I

Sonny Prasetio, ST., MT.
NIP. 1031000433

Dosen Penguji II

Karina Auliasari, ST., M.Eng.
NIP. 1031000426

Dosen Pembimbing I

Surya Adi Wibowo, ST., MT.
NIP. 1031000438

Dosen Pembimbing II

Febriana Santi Wahyuni, S.Kom., M.Kom.
NIP. 1031000425

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

FT. 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, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 23 Oktober 2015

Nomor : ITN-593/I.INF/TA/2015
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 : MUHAMMAD KEVIN DESMAIZAL
Nim : 1218255
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 :

23 Oktober 2015 S/D 23 Maret 2016

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 InformatikaS-1
K e t u a,

Joseph Dedy Irawan,ST., MT.
NIP : 19740416200021002

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, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

Malang, 23 Oktober 2015

Nomor : ITN-593/I.INF/TA/2015
Lampiran : ---
Perihal : *Bimbingan Skripsi*

Kepada : Yth. Bpk/Ibu Febriana Santi W, S.Kom, M.Kom
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 : MUHAMMAD KEVIN DESMAIZAL
Nim : 1218255
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 :

23 Oktober 2015 S/D 23 Maret 2016

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 InformatikaS-1
K e t u a,

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

Form S-4a

FORMULIR BIMBINGANG SKRIPSI

Nama : Muhammad Kevin Desmaizal

Nim : 1218255

Masa Bimbingan : 23 Oktober 2015 - 23 Maret 2016

Judul Skripsi : PENERAPAN *MAXIMUM ELEMENT ALGORITHM* DAN *FINITE STATE MACHINE* PADA GAME "STRIVE"

No.	Tanggal	Uraian	Paraf
1	19 - 11 – 2015	Ganti background setiap level	
2	21 - 11 – 2015	Efek partikel pada saat aksi, hitpoint, special weapon	
3	23 - 11 – 2015	Acc Sem. progress	
4	24 - 11 – 2015	Pengujian	
5	12 - 12 – 2015	Revisi paper sem. hasil	
6	14 - 12 – 2015	Acc Sem. hasil	
7	12 - 1 – 2016	Penambahan cheat, revisi bab 4 dan bab 5	
8	14 - 1 – 2016	Acc Kompre	

Malang, 14 - 1 - 2016
Dosen Pembimbing II

Suryo Adi Wibowo, S.T, M.T.
NIP. 1031500480

FORMULIR BIMBINGANG SKRIPSI

Nama : Muhammad Kevin Desmaizal

Nim : 1218255

Masa Bimbingan : 23 Oktober 2015 - 23 Maret 2016

Judul Skripsi : PENERAPAN *MAXIMUM ELEMENT ALGORITHM* DAN *FINITE STATE MACHINE* PADA GAME "STRIVE"

No.	Tanggal	Uraian	Paraf
1	18 - 11 – 2015	Revisi program perbaiki AI *maximum element algorithm*	
2	20 - 11 – 2015	Revisi program menambah AI finite state machine	
3	21 - 11 – 2015	Acc program, konsultasi seminar progress	
4	23 - 11 – 2015	Acc Sem. progress	
5	11 - 12 – 2015	Revisi program, Konsultasi Sem. hasil	
6	12 - 12 – 2015	Acc Sem. hasil	
7	12 - 1 – 2016	Perbaikan Program, perbaikan laporan	
8	13 - 1 – 2016	Acc Kompre	

Malang, 14 - 1 - 2016
Dosen Pembimbing I

Febriana Santi Wahyuni., S.Kom, M.Kom
NIP. P. 1031000425

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : HERIYANTO

Pekerjaan : WELDER

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game	√		
3	Desain karakter dan background		√	

Malang, 2015

Menyetujui,

HERIYANTO

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Wantono

Pekerjaan : Mahasiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay		√	
2	Genre platformer pada game	√		
3	Desain karakter dan background	√		

Malang, 2015

Menyetujui,

Wantono

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Awang Ramadhani Saputra

Pekerjaan : Mahasiswa T. Informatika S-1

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game	√		
3	Desain karakter dan background	√		

Malang, 2015

Menyetujui,

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Muhammad Ghufron

Pekerjaan : Mahasiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay		✓	
2	Genre platformer pada game	✓		
3	Desain karakter dan background			✓

Malang, 2015

Menyetujui,

M. Ghufron

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Arohman Hidayat

Pekerjaan : MAHASiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game			√
3	Desain karakter dan background	√		

Malang, 2015

Menyetujui,

Arohman-H

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Ahmad Mustafa Aziz

Pekerjaan : Mahasiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game	√		
3	Desain karakter dan background			√

Malang, 2015

Menyetujui,

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Ulvi R. Sudariarto

Pekerjaan : Mahasiswa.

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game	√		
3	Desain karakter dan background	√		

Malang, 2015

Menyetujui,

Ulvi. R. Sudariarto

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : MUHAMMAD NUR HADI

Pekerjaan : MAHASISWA

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay		√	
2	Genre platformer pada game		√	
3	Desain karakter dan background	√		

Malang, 2015

Menyetujui,

_____

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Achmad Rozikin

Pekerjaan : Mahasiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game	√		
3	Desain karakter dan background		√	

Malang, 2015

Menyetujui,

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Syahrizal Tsuroya Agusar

Pekerjaan : Mahasiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay		√	
2	Genre platformer pada game	√		
3	Desain karakter dan background	√		

Malang, 2015

Menyetujui,

ANGKET PENGUJIAN

Penerapan Maximum Element Algorithm dan Finite State Machine pada Game "Strive"

Nama : Diman Kristian

Pekerjaan : Mahasiswa

Berilah tanda (√) pada salah satu kolom "Penilaian" dibawah ini :

No.	Kriteria	Penilaian		
		Menarik	Kurang Menarik	Tidak Menarik
1	Gameplay	√		
2	Genre platformer pada game		√	
3	Desain karakter dan background		√	

Malang, 2015

Menyetujui,

Diman Kristian

LAMPIRAN 1. Script Player.cs

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(Controller2d))]
public class Player : MonoBehaviour {
	public float jumpHeight = 4;
	public float timeToJumpApex = .4f;
	float accelerationTimeAirborne = .2f;
	float accelerationTimeGrounded = .1f;
	float moveSpeed = 6;
	float gravity;
	float jumpVelocity;
	Vector3 velocity;
	float velocityXSmoothing;
	public AudioClip sjump;
	float jump=0;
	float groundRadius = 0.1f;
	public LayerMask groundLayer;
	public Transform groundcheck;
	bool isGrounded = true;
	Animator AnimController;
	public bool facingRight = true;
	public Camera MainCamera;
	public Transform Background;
	public Transform HPbar;
	public Transform HPback;
	public bool control;
	Controller2d controller;
```

```
        void Start() {
                controller = GetComponent<Controller2d> ();
                gravity = -(2 * jumpHeight) / Mathf.Pow (timeToJumpApex, 2);
                jumpVelocity = Mathf.Abs(gravity) * timeToJumpApex;
                AnimController = GetComponent<Animator> ();
        }
        void Update() {
                if (control) {
                float h = Input.GetAxis ("Horizontal");
                if (controller.collisions.above || controller.collisions.below) {
                        velocity.y = 0;
                }
Vector2 input = new Vector2 (Input.GetAxisRaw
("Horizontal"),Input.GetAxisRaw ("Vertical"));
if (Input.GetKeyDown (KeyCode.Space) && controller.collisions.below) {
                velocity.y = jumpVelocity;
                AnimController.SetBool ("IsPlayerOnGround", false);
                AudioSource.PlayClipAtPoint (sjump, this.transform.position);
                }
                float targetVelocityX = input.x * moveSpeed;
                velocity.x = Mathf.SmoothDamp (velocity.x, targetVelocityX, ref
velocityXSmoothing, (controller.collisions.below) ? accelerationTimeGrounded :
accelerationTimeAirborne);
                        velocity.y += gravity * Time.deltaTime;
                        controller.Move (velocity * Time.deltaTime);
                        if (h > 0 && !facingRight)
                                Flip ();
                        else if (h < 0 && facingRight)
                                Flip ();
                }
```

```
    }
    void FixedUpdate(){
isGrounded = Physics2D.OverlapCircle (groundcheck.position, groundRadius,
groundLayer);
            jump = 0;
            float h = Input.GetAxis("Horizontal");
            AnimController.SetBool ("IsPlayerOnGround", isGrounded);
            AnimController.SetFloat("JumpSpeed", velocity.y);
            AnimController.SetFloat("Speed", Mathf.Abs(h));
    }
    void fall(){
            wait ();
            jump = 0;
            AnimController.SetFloat("JumpSpeed", jump);
    }
    IEnumerator wait(){
            yield return new WaitForSeconds (1f);
    }
    void Flip (){
            facingRight = !facingRight;
            Vector3 theScale = transform.localScale;
            theScale.x *= -1;
            transform.localScale = theScale;
    }
    void OnCollisionEnter2D(Collision2D other){
            if (other.transform.tag == "MovingPlatform") {
                    transform.parent = other.transform;
            }}
    void OnCollisionExit2D(Collision2D other){
```

```
            if (other.transform.tag == "MovingPlatform") {
                transform.parent = null;
            }
        }
}
```

LAMPIRAN 2. Script Controller2d.cs

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(BoxCollider2D))]
public class Controller2d : MonoBehaviour {
	public LayerMask collisionMask;
	const float skinWidth = .015f;
	public int horizontalRayCount = 0;
	public int verticalRayCount = 0;
	float horizontalRaySpacing;
	float verticalRaySpacing;
	BoxCollider2D collider;
	RaycastOrigins raycastOrigins;
	public CollisionInfo collisions;
	void Start() {
		collider = GetComponent<BoxCollider2D> ();
		CalculateRaySpacing ();
	}
	public void Move(Vector3 velocity) {
		UpdateRaycastOrigins ();
		collisions.Reset ();
		if (velocity.x != 0) {
			HorizontalCollisions (ref velocity);
		}
		if (velocity.y != 0) {
			VerticalCollisions (ref velocity);
		}
```

```
            transform.Translate (velocity);
        }
        void HorizontalCollisions(ref Vector3 velocity) {
            float directionX = Mathf.Sign (velocity.x);
            float rayLength = Mathf.Abs (velocity.x) + skinWidth;
            for (int i = 0; i < horizontalRayCount; i ++) {
        Vector2 rayOrigin = (directionX == -
1)?raycastOrigins.bottomLeft:raycastOrigins.bottomRight;
rayOrigin += Vector2.up * (horizontalRaySpacing * i);
RaycastHit2D hit = Physics2D.Raycast(rayOrigin, Vector2.right * directionX,
rayLength, collisionMask);
                    if (hit) {
                        velocity.x = (hit.distance - skinWidth) * directionX;
                        rayLength = hit.distance;
                        collisions.left = directionX == -1;
                        collisions.right = directionX == 1;
                    }
            }
        }
        void VerticalCollisions(ref Vector3 velocity) {
            float directionY = Mathf.Sign (velocity.y);
            float rayLength = Mathf.Abs (velocity.y) + skinWidth;
            for (int i = 0; i < verticalRayCount; i ++) {
Vector2 rayOrigin = (directionY == -
1)?raycastOrigins.bottomLeft:raycastOrigins.topLeft;
rayOrigin += Vector2.right * (verticalRaySpacing * i + velocity.x);
RaycastHit2D hit = Physics2D.Raycast(rayOrigin, Vector2.up * directionY,
rayLength, collisionMask);
                    if (hit) {
                        velocity.y = (hit.distance - skinWidth) * directionY;
                        rayLength = hit.distance;
```

```
                        collisions.below = directionY == -1;
                        collisions.above = directionY == 1;
                    }
                }
            }

        void UpdateRaycastOrigins() {
                Bounds bounds = collider.bounds;
                bounds.Expand (skinWidth * -2);
raycastOrigins.bottomLeft = new Vector2 (bounds.min.x, bounds.min.y);
raycastOrigins.bottomRight = new Vector2 (bounds.max.x, bounds.min.y);
raycastOrigins.topLeft = new Vector2 (bounds.min.x, bounds.max.y);
raycastOrigins.topRight = new Vector2 (bounds.max.x, bounds.max.y);
}
        void CalculateRaySpacing() {
Bounds bounds = collider.bounds;bounds.Expand (skinWidth * -2);

horizontalRayCount = Mathf.Clamp (horizontalRayCount, 2, int.MaxValue);
verticalRayCount = Mathf.Clamp (verticalRayCount, 2, int.MaxValue);
horizontalRaySpacing = bounds.size.y / (horizontalRayCount - 1);
verticalRaySpacing = bounds.size.x / (verticalRayCount - 1);
        }

        struct RaycastOrigins {
                public Vector2 topLeft, topRight;
                public Vector2 bottomLeft, bottomRight;
        }

        public struct CollisionInfo {
```

```
        public bool above, below;
        public bool left, right;
        public void Reset() {
            above = below = false;
            left = right = false;
        }
    }
}
```

LAMPIRAN 3. Script Weapon.cs

```
using UnityEngine;
using System.Collections;

public class Weapon : MonoBehaviour {
        privatc bool attacking = false;
        public float attacktime = 1f;
        public float attackcol = 1f;
        public GameObject attackrange;
        public int[] senjata;
        public bool[] senjatanya;
        int max;
        public Animator anim;
        public AudioClip sw1;
        public AudioClip sw2;
        public AudioClip sw3;
        public AudioClip sw4;
        public AudioClip sw5;
        public AudioClip sw6;
        public Transform firepoint;
        public GameObject arrow;
        public GameObject gun;
        public GameObject keris;
        Player player;
        Controller2d control;
        void Awake(){
         anim = GameObject.Find ("Player").GetComponent<Animator> ();
         player = FindObjectOfType<Player> ();
         control = FindObjectOfType<Controller2d> ();
```

```
        senjata [0] = PlayerPrefs.GetInt ("Senjata1");
        senjata [1] = PlayerPrefs.GetInt ("Senjata2");
        senjata [2] = PlayerPrefs.GetInt ("Senjata3");
        senjata [3] = PlayerPrefs.GetInt ("Senjata4");
        senjata [4] = PlayerPrefs.GetInt ("Senjata5");
        senjata [5] = PlayerPrefs.GetInt ("Senjata6");
      }
      void allfalse(){
       for(int j=0; j < 6; j++){
              senjatanya[j] = false;
       }
      }
      void Update () {
       for(int i=0;i<6; i++ ){
              if(max < senjata[i]){
                     max = senjata[i];
                     allfalse();
                     senjatanya[i]=true;
              }
       }
       if (player.control) {
if (Input.GetKeyDown (KeyCode.X) && !attacking && control.collisions.below)
{
                     attacking = true;
                     attacktime = attackcol;
                     if(senjatanya[0]){
                            //Senjata 1 : Pedang
                            anim.SetTrigger ("W1");
              AudioSource.PlayClipAtPoint (sw1, this.transform.position);
```

```
                attackrange.SetActive (true);
        }
        else if(senjatanya[1]){
                //Senjata 2 : Panah
                anim.SetTrigger ("W2");
AudioSource.PlayClipAtPoint (sw2, this.transform.position);
Instantiate (arrow, firepoint.position, firepoint.rotation);
        }
        else if(senjatanya[2]){
                //Senjata 3 : Palu
                anim.SetTrigger ("W3");
AudioSource.PlayClipAtPoint (sw3, this.transform.position);
                attackrange.SetActive (true);
        }
        else if(senjatanya[3]){
                //Senjata 4 : Pistol
                anim.SetTrigger ("W4");
AudioSource.PlayClipAtPoint (sw4, this.transform.position);
Instantiate (gun, firepoint.position, firepoint.rotation);
        }
        else if(senjatanya[4]){
                //Senjata 5 : Tombak
                anim.SetTrigger ("W5");
AudioSource.PlayClipAtPoint (sw5, this.transform.position);
                attackrange.SetActive (true);
        }
        else if(senjatanya[5]){
                //Senjata 1 : Keris
                anim.SetTrigger ("W6");
```

```
                AudioSource.PlayClipAtPoint (sw6, this.transform.position);
                Instantiate (keris, firepoint.position, firepoint.rotation);
                        }
                }
                if (attacking) {
                        if (attacktime > 0) {
                                attacktime -= Time.deltaTime;
                        } else {
                                attacking = false;
                                attackrange.SetActive (false);
                        }
                }
        }
    }
}
```

LAMPIRAN 4 Script Glitch.cs

```
using UnityEngine;
using System.Collections;

public class Glitch : MonoBehaviour {
        public float moveSpeed = 0.6f;
        public float chasespeed;
        float step;
        GlobalScore score;
        public float timestill = 2f;
        public bool ptr=true;
        public bool stillptr = true;
        public bool attacked;
        public bool stillattacked;
        public bool isfacingright;
        public int HP = 2;
        public SpriteRenderer ren;
        private Transform frontCheck;
        public Animator animate;
        public AudioClip sglitch;
        public AudioClip sglitchdamage;
        public AudioClip sglitchdead;
        public bool move=false;
        public GameObject damageparticle;
        public GameObject deadparticle;
        public GameObject glitch;
        public Transform[] point;
        int currentpoint;
```

```
public Transform player;
float hurtForce = 20f;
float repeatDamagePeriod = 1f;
private float lastHitTime;
Player facingplayer;
HP playerhp;
public Transform wallCheck;
public float wallCheckRadius;
public LayerMask whatIsWall;
private bool hittingWall;
void Awake () {
animate = glitch.GetComponent<Animator> ();
player = GameObject.Find("Player").GetComponent<Transform>();
facingplayer = GameObject.Find("Player").GetComponent<Player>();
    score = FindObjectOfType<GlobalScore> ();
    playerhp = FindObjectOfType<HP> ();
    ptr = true;
    chasespeed = moveSpeed+0.02f;
}
void Start(){
    StartCoroutine ("patrol");
}
void Update(){
    chasing ();
hittingWall = Physics2D.OverlapCircle (wallCheck.position, wallCheckRadius,
whatIsWall);
    if (hittingWall) {
        attacked = false;
        move = false;
```

```
                    animate.SetBool ("Move", move);
                    isfacingright = !isfacingright;
                    flip();
                }
            }

            void chasing(){
                if (attacked && Time.timeScale == 1f) {
                    if(!playerhp.immun){
                        glitch.transform.position = Vector2.MoveTowards
(glitch.transform.position, new Vector2 (player.position.x,
glitch.transform.position.y), chasespeed);
                        if(transform.position.x > player.position.x){
                            isfacingright = true;
                            flip ();
                        }
                        else if(transform.position.x < player.position.x)
                        {
                            isfacingright = false;
                            flip();
                        }
                    }
                }
            }
            void damaged(){
                HP--;
                AudioSource.PlayClipAtPoint (sglitch, this.transform.position);
AudioSource.PlayClipAtPoint (sglitchdamage, this.transform.position);
                if (facingplayer.facingRight) {
```

```
                    Vector3 hurtVector = transform.position - player.position +
Vector3.right * 5f;
Instantiate (damageparticle, this.transform.position, this.transform.rotation);
glitch.GetComponent<Rigidbody2D> ().AddForce (hurtVector * hurtForce);
                }
                else if (!facingplayer.facingRight) {
Vector3 hurtVector = transform.position - player.position + Vector3.left * 5f;
Instantiate (damageparticle, this.transform.position, this.transform.rotation);
glitch.GetComponent<Rigidbody2D> ().AddForce (hurtVector * hurtForce);
                }
                attacked = false;
                StartCoroutine ("immune");
                move = false;
                animate.SetBool ("Move", move);
            }

            IEnumerator immune(){
                if (stillptr) {
                        ptr = false;
                }
                if (stillattacked) {
                        attacked =false;
                }
                animate.SetTrigger ("Damaged");
                yield return new WaitForSeconds (1f);
                if (stillptr) {
                        ptr = true;
                        StartCoroutine ("patrol");
                }
```

```
            if (stillattacked) {
                attacked = true;
            }
        }

        void OnTriggerEnter2D(Collider2D col){
            if (col.gameObject.tag == "Player" ) {
                if (!stillattacked) {
                    move = true;
                    animate.SetBool ("Move", move);
            AudioSource.PlayClipAtPoint (sglitch, this.transform.position);
                    ptr = false;
                    stillptr = false;
                    attacked = true;
                    stillattacked = true;
                }
            }
if (col.gameObject.tag == "Slash" || col.gameObject.tag == "Arrow") {
            if (HP > 0) {
                if (Time.time > lastHitTime + repeatDamagePeriod) {
                    damaged ();
                    lastHitTime = Time.time;
                }
                } else {
                    score.score += 500f;
        AudioSource.PlayClipAtPoint (sglitchdead, this.transform.position);
Instantiate (deadparticle, this.transform.position, this.transform.rotation);
                    Destroy (glitch);
                }
```

```
        }
    }
    IEnumerator patrol(){
        while (ptr) {
    if (glitch.transform.position.x == point [currentpoint].position.x) {
                        currentpoint++;
                        animate.SetBool ("Move", false);
                        yield return new WaitForSeconds (timestill);
                        animate.SetBool ("Move", true);
                    }
                    if (currentpoint >= point.Length) {
                        currentpoint = 0;
                    }
    if (glitch.transform.position.x > point [currentpoint].position.x){
                    isfacingright = true;
                    flip();
                }
    else if (glitch.transform.position.x < point [currentpoint].position.x){
                    isfacingright = false;
                    flip ();
                }
                    yield return null;
                }
    }
    void flip(){
        if (isfacingright) {
                glitch.transform.localScale = Vector3.one;}
                else {
glitch.transform.localScale = new Vector3 (-1, 1, 1);}}}
```

LAMPIRAN 5. Script GlitchAttack.cs

```
using UnityEngine;
using System.Collections;

public class GlitchAttack : MonoBehaviour {

        public Animator x;
        public Transform flip;
        public GameObject atk;
        public GameObject glitch;
        public GameObject glitchchar;
        private bool attacking = false;
        public float attacktime = 1f;
        Glitch g;

        // Use this for initialization
        void Start () {
          x = glitch.GetComponent<Animator> ();
          flip = glitch.GetComponent<Transform> ();
          g = glitchchar.GetComponent<Glitch> ();
        }

        void OnTriggerEnter2D(Collider2D col){
          if (col.gameObject.tag == "Player") {
                attacking = true;
                g.stillattacked = false;
          }
        }
```

```
                    void Update(){
                      if(attacking){
                            x.SetTrigger("Attack");
                            g.stillattacked = true;
                            atk.SetActive(true);
                            if(attacktime > 0){
                                  attacktime -= Time.deltaTime;
                            }
                            else{
                                  atk.SetActive(false);
                                  attacking = false;
                                  g.stillattacked = false;
                            }
                      }
                    }
}
```

LAMPIRAN 6. Script EyeGlitch.cs

```
using UnityEngine;
using System.Collections;

public class EyeGlitch : MonoBehaviour {
        Animator anim;
        float hp = 3;
        bool chase;
        public float movespeed;
        public Transform player;
        Player facingplayer;
        HP hitpoint;
        GlobalScore score;

        public AudioClip sglitch;
        public AudioClip sglitchdead;
        public AudioClip sglitchdamage;
        public GameObject damageparticle;
        public GameObject deadparticle;
        public GameObject eye;

        float repeatDamagePeriod = 1f;
        private float lastHitTime;

        void Start(){
        player = GameObject.Find("Player").GetComponent<Transform>();
        facingplayer = GameObject.Find("Player").GetComponent<Player>();
                hitpoint = FindObjectOfType<HP> ();
                anim = eye.GetComponent<Animator> ();;
```

```
            score = FindObjectOfType<GlobalScore> ();
        }
        void Update () {
            if (!hitpoint.immun) {
                if (chase) {
eye.transform.position = Vector3.MoveTowards (eye.transform.position,
player.transform.position, movespeed * Time.deltaTime);
                }
            }
        }
        void OnTriggerEnter2D(Collider2D col){
            if (col.gameObject.tag == "Player") {
            AudioSource.PlayClipAtPoint (sglitch, this.transform.position);
                chase = true;
                anim.SetBool("Chase", chase);
            }
        if (col.gameObject.tag == "Slash" || col.gameObject.tag == "Arrow") {
                if (hp > 0) {
                if (Time.time > lastHitTime + repeatDamagePeriod) {
                        damaged ();
                        lastHitTime = Time.time;
                    }
                } else {
                    score.score += 500f;
        AudioSource.PlayClipAtPoint (sglitchdead, this.transform.position);
        Instantiate (deadparticle, this.transform.position, this.transform.rotation);
                        Destroy(eye);
                        Destroy (gameObject);
                }
```

LAMPIRAN 7. Script Trap

```
using UnityEngine;
using System.Collections;

public class Trap : MonoBehaviour {
                                    float plus;
                                    float minus;
                                    public Transform thetrap;
                                    public AudioClip sup;
                                    public AudioClip sdown;
                                    void Awake(){
                                        plus = transform.position.y + 0.78f;
                                        minus = plus - 0.78f;
                                    }
                                    void OnTriggerEnter2D(Collider2D col){
                                        if (col.gameObject.tag == "Player") {
thetrap.transform.position = new Vector2(transform.position.x, plus);
AudioSource.PlayClipAtPoint(sup, this.transform.position);
                                        }
                                    }
                                    void OnTriggerExit2D(Collider2D a){
                                        if (a.gameObject.tag == "Player") {
thetrap.transform.position = new Vector2(transform.position.x, minus);
AudioSource.PlayClipAtPoint(sdown, this.transform.position);
                                        }
                                    }
}
```

LAMPIRAN 8. Script Blast.cs

```
using UnityEngine;
using System.Collections;

public class Blast : MonoBehaviour {

    public float speed;
    public float homming;
    public float rotationSpeed;
    public GameObject smoke;
    Transform playerpos;
    public float blastTime;
    public AudioClip sblast;
    void Start(){
    playerpos = GameObject.Find ("Player").GetComponent<Transform> ();
        homming = speed * Time.deltaTime;
    }
    // Update is called once per frame
    void Update () {
        if (blastTime > 0) {
            blastTime -= Time.deltaTime;
transform.position = Vector3.MoveTowards (transform.position,
playerpos.transform.position, homming);
        }
        else {
            gone();
        }
        GetComponent<Rigidbody2D>().angularVelocity = rotationSpeed;
    }
```