TUGAS AKHIR – KI141502

IMPLEMENTASI AI MENGGUNAKAN LIBRARY A* PATHFINDING PROJECT OLEH ARON GRANBERG DAN TINGKAT KEBERANIAN LAWAN PADA GAME PLATFORMER ASHES OF RESURGENCE

FAIS ZHARFAN AZIF
NRP 5112100098

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Wijayanti Nurul K., S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

FINAL PROJECT– KI141502

AI IMPLEMENTATION USING A* PATHFINDING PROJECT LIBRARY BY ARON GRANBERG AND BRAVERY LEVEL APPLICATION ON PLATFORMER GAME ASHES OF RESURGENCE

FAIS ZHARFAN AZIF
NRP 5112100098

Advisor
Imam Kuswardayan, S.Kom., M.T.
Wijayanti Nurul K., S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI AI MENGGUNAKAN LIBRARY A* PATHFINDING PROJECT OLEH ARON GRANBERG DAN TINGKAT KEBERANIAN LAWAN PADA GAME PLATFORMER ASHES OF RESURGENCE

Tugas Akhir

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi, Grafika, dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
FAIS ZHARFAN AZIF
NRP. 5112 100 098

Disetujui oleh Dosen Pembimbing Tugas Akhir:
Imam Kuswardayan, S.Kom., M.T. .....
NIP: 19761215 200312 1 001 (pembimbing 1)

Wijayanti Nurul K., S.Kom., M.Sc. .....
NIP: 19860312 201212 2 004 (pembimbing 2)

SURABAYA
JULI, 2016

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI AI MENGGUNAKAN LIBRARY A* PATHFINDING PROJECT OLEH ARON GRANBERG DAN TINGKAT KEBERANIAN LAWAN PADA GAME PLATFORMER ASHES OF RESURGENCE

Nama Mahasiswa : Fais Zharfan Azif
NRP : 5112 100 098
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing II : Wijayanti Nurul K., S.Kom., M.Sc.

ABSTRAK

Semakin berkembangnya zaman, game semakin dapat diterima oleh berbagai kalangan. Tua maupun muda, semua tidak asing dengan game. Berbagai jenis game pun dapat ditemukan di era global ini, dari game edukasi yang ditujukan untuk anak balita hingga game khusus untuk orang dewasa. Genre game yang dapat dimainkan pun semakin bervariasi, seperti first-person shooter, open world, hingga MMORPG.

Dari sekian banyak game yang tersedia, beberapa genre game yang dulunya berjaya di era 90an mulai tergeser dengan game modern, salah satunya adalah game bergenre platformer. Walaupun beberapa game platformer seperti super mario masih dapat bertahan hingga saat ini, kebanyakan game platformer kurang memiliki inovasi terutama di bidang artificial intelligence (AI). Pemain selalu ingin tantangan lebih jika bermain game, dan tanpa AI, pemain dapat menang dengan mudah dan pada akhirnya pemain tidak merasa tertantang dan bosan.

Untuk itu, dalam Tugas Akhir ini, dibangun sebuah game bergenre platformer dengan menggunakan AI. AI diimplementasikan dengan menggunakan library A pathfinding project yang dibuat oleh Aron Granberg untuk penerapan tingkat keberanian lawan. Game dibangun dengan menggunakan Unity dan dapat dijalankan pada desktop dengan sistem operasi*

windows atau perangkat mobile dengan sistem operasi android. Tujuan dari pembuatan game ini adalah untuk menerapkan AI pada musuh sehingga musuh akan memiliki perilaku yang disesuaikan dengan tingkat keberanian.

Kata kunci**: **Unity, AI, A* Pathfinding Project, Platformer, Tingkat keberanian.

AI IMPLEMENTATION USING A* PATHFINDING PROJECT LIBRARY BY ARON GRANBERG AND BRAVERY LEVEL APPLICATION ON PLATFORMER GAME ASHES OF RESURGENCE

Student Name	: Fais Zharfan Azif
NRP	: 5112 100 098
Major	: Informatics Engineering FTIf-ITS
Advisor I	: Imam Kuswardayan, S.Kom., M.T.
Advisor II	: Wijayanti Nurul K., S.Kom., M.Sc.

ABSTRACT

As the global age continues, video game is more accepted by many. Old and young are all used to gaming. Even various kinds of games can be found nowadays, ranging from educational games aimed at young children to games specifically designed for adults. The genre of games that can be played are also increasingly varied, such as first-person shooter, open world, MMORPG, and many more.

Among the many games available, several genre of games that were once triumphed in the 90s era began to be replaced with modern games, one of which is the platformer genre. Although several platformer games such as Super Mario can still survive until today, most platformer games lack of innovation, especially in the field of artificial intelligence (AI). Yet, players always want more challenge when playing games, and without AI, players can win easily and therefore the players will quickly feel less challenged and eventually bored.

*Therefore, in this final project, a platformer game using AI is built. The implemented AI is using the A * Pathfinding Project library by Aron Granberg nd the bravery level on enemies. The game will be built using Unity and can be run on a desktop with windows operating system or mobile device with the*

android operating system. The objective of building this game is to apply AI to the enemy so the enemy will have a behavior adapted to the bravery level.

Keywords**: **Unity, AI, A* Pathfinding Project, Platformer, Bravery Level.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

KODE SUMBER

BAB I
PENDAHULUAN

Bab ini memaparkan garis besar Tugas Akhir yang meliputi latar belakang, tujuan dan manfaat pembuatan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Game bergenre platformer bukanlah sesuatu yang baru dalam dunia *game*. Sejak dikenalkan pada tahun 1980, game platformer terus mengalami perkembangan, termasuk penerapan kecerdasan buatan (*artificial intelligence*). Pada *game* platformer, kecerdasan buatan diaplikasikan pada lingkungan (*environment*) maupun pada karakter yang dikontrol komputer atau NPC (*Non-Player Character*). Kecerdasan buatan memiliki fungsi yang bermacam-macam, mulai dari berbagai pola penyerangan dan bertahan yang dimiliki oleh NPC hingga pencarian jalur yang tercepat. Akan tetapi, kecerdasan buatan pada NPC sebagian besar memiliki manuver atau pola pergerakan yang sama, tidak peduli apakah musuhnya terlalu kuat maupun terlalu lemah.

Walaupun kecerdasan buatan dalam *game* adalah hal yang hampir selalu ada, sebagian besar platformer *game* yang dijual secara komersial menggunakan kecerdasan buatan yang sedikit atau bahkan tidak sama sekali. NPC yang seolah-olah menggunakan kecerdasan buatan hanya melakukan pergerakan sesuai dengan pola yang telah ditentukan [1]. Kalaupun menggunakan kecerdasan buatan yang dinamis, kecerdasan buatan pada *game* bergenre platformer biasanya berupa pencarian jalan tercepat (*pathfinding*), bukan untuk menentukan *behaviour* NPC. Dengan menggunakan konsep-konsep pendekatan AI dalam penerapan sistem tingkat keberanian pada game, NPC dapat menentukan pilihan *behaviour* terbaik dalam usaha yang tidak hanya untuk mengalahkan musuhnya, tetapi juga usaha untuk bertahan hidup.

Melalui pendekatan ini, diharapkan NPC pada *game* platformer tetap akan melakukan serangan kepada lawannya, tetapi tidak akan lagi terobsesi untuk mengalahkan musuhnya. Jika musuh terlalu kuat, NPC akan melarikan diri atau melakukan *behaviour* bertahan. Sebaliknya, jika musuh terlalu lemah, NPC akan melakukan penyerangan pada musuhnya secara lebih agresif.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang dikemukakan, beberapa permasalahan yang akan diselesaikan dalam tugas akhir adalah sebagai berikut:

1. Bagaimana rancangan aturan main *game* Ashes of Resurgence.
2. Bagaimana rancangan skenario atau level dalam *game* Ashes of Resurgence.
3. Bagaimana rancangan *AI* dalam sistem tingkat keberanian musuh.
4. Bagaimana pengimplementasian sistem tingkat keberanian kedalam *game* Ashes of Resurgence.

1.3 Batasan Masalah

Beberapa batasan masalah yang menjadi batas pada tugas akhir ini adalah sebagai berikut:

1. Pendekatan *AI* hanya diimplementasikan dalam sistem tingkat keberanian.
2. Sistem tingkat keberanian digunakan untuk menentukan frekuensi serangan, kecepatan pergerakan, dan jarak musuh dengan pemain.
3. *AI* tidak digunakan untuk proses *learning* kebiasaan main pemain.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini diantaranya:

1. Membuat sebuah game yang mengimplementasikan *AI*

2. Menerapkan algoritma A* dalam pencarian jalan tercepat bagi musuh
3. Menerapkan sistem tingkat keberanian musuh sebagai bentuk AI.

1.5 Manfaat

Manfaat dari pembuatan Tugas Akhir ini antara lain:

1. Sebagai bentuk penerapan *AI* pada game bergenre platformer.
2. Memberikan sebuah konsep baru pada game bergenre platformer sehingga kedepannya dapat digunakan sebagai referensi.

1.6 Metodologi

Metodologi dalam pembuatan Tugas Akhir ini antara lain sebagai berikut:

A. Studi literatur
   Pada tahap studi literatur, akan dipelajari sejumlah *textbook* dan referensi yang berkaitan dalam pembuatan aplikasi untuk tugas akhir. Referensi yang akan dipelajari akan membahas beberapa hal berikut:
   1. Algoritma A*;
   2. A* Pathfinding Project;
   3. Unity;
   4. Pathfinding untuk 2D game platformer;

B. Perancangan perangkat lunak
   Pada tahap ini, akan dilakukan perancangan dan desain dari aplikasi. Perancangan akan didasari oleh kebutuhan yang sudah dianalisa sebelumnya. Langkah detail dari tahap ini adalah sebagai berikut:
   1. Perancangan mekanik dasar dari game.
   2. Desain tampilan dalam game utamanya.
   3. Perancangan mekanik pelengkap beserta desainnya.
   4. Analisis pathfinding dalam game.
   5. Perancangan *AI* dalam bentuk sistem tingkat keberanian.

C. Implementasi dan pembuatan sistem
Aplikasi akan dibuat dengan kakas bantu Unity2D. Urutan pengerjaan dalam tahap ini adalah sebagai berikut:
1. Pencarian dan pendataan aset yang digunakan dalam game akan dilakukan secara berulang ketika aset baru diperlukan.
2. Membuat elemen – elemen dasar yang diperlukan untuk player, seperti stats, equip, weapon, dan sebagainya.
3. Membuat elemen – elemen dasar yang diperlukan untuk musuh.
4. Pembuatan tampilan untuk tiap stage.
5. Pembuatan jalan dan lokasi musuh untuk tiap stage.
6. Pembuatan *scene* World Map untuk meletakkan seluruh stage.
7. Implementasi pathfinding menggunakan A* pada musuh.
8. Implementasi sistem tingkat keberanian yang berbeda untuk tiap musuh.

D. Uji coba dan evaluasi
Pada tahap ini, pengujian dan evaluasi akan dilakukan secara langsung oleh pengguna untuk mendapatkan hasil evaluasi dan *feedback* langsung dari pengguna. Beberapa skenario yang juga akan diuji antara lain:
1. Pengujian skenario khusus
   Pengujian skenario khusus adalah pengujian aplikasi dengan skenario yang tidak umum / ekstrim untuk menguji apakah aplikasi masih dapat berjalan semestinya atau tidak.
2. Pengujian usabilitas
   Pengujian usabilitas adalah pengujian yang dilakukan langsung oleh pengguna. Setiap pengguna yang melakukan pengujian akan diberi survey untuk menilai usabilitas aplikasi.

E. Penyusunan laporan tugas akhir
Pada tahap terakhir ini, penyusunan laporan tugas akhir dilakukan. Laporan tugas akhir akan menjelaskan tentang

aplikasi yang dibangun berdasarkan teori dan perancangan yang dilakukan.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab yang membahas detail dari Tugas Akhir. Bab-bab ini antara lain sebagai berikut.

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang masalah yang diangkat, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi pembuatan, beserta sistematika penulisan tugas akhir.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang dasar teori dan materi-materi yang digunakan selama menyusun tugas akhir untuk membantu dalam pengerjaan.

BAB III ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang analisis yang dilakukan untuk menyusun aplikasi tugas akhir beserta perancangan apa saja yang dilakukan sebelum melakukan implementasi pembuatan aplikasi tugas akhir.

BAB IV IMPLEMENTASI

Bab ini menjelaskan tentang proses implementasi dari pembuatan aplikasi tugas akhir berdasarkan analisis dan perancangan yang telah dilakukan pada bab sebelumnya.

BAB V PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tentang keseluruhan proses pengujian beserta evaluasi yang dilakukan untuk mendapatkan data mengenai kemampuan aplikasi yang dibuat dengan melihat hasil dari pengujian dan evaluasi.

BAB VI PENUTUP

Bab ini membahas tentang kesimpulan dari pengujian dan evaluasi yang berhubungan dengan topik yang dibahas beserta saran pengembangan aplikasi kedepannya.

BAB II
TINJAUAN PUSTAKA

Bab ini akan membahas mengenai teori, materi, dan *tools* yang digunakan dalam keseluruhan pembuatan tugas akhir.

2.1 Unity *Game Engine*

Unity adalah sebuah *game engine* atau aplikasi perantara atau *middleware* dalam pembuatan *game* yang dibuat oleh Unity Technologies. Unity mempermudah proses pembuatan dan pengembangan *game* dengan memiliki berbagai jenis macam *library* dan fungsi yang dapat langsung digunakan sehingga pengembang *game* tidak perlu membuat game dari *scratch*. Unity adalah *middleware* yang *powerful* dan dapat membuat *game* di berbagai macam *platform* sehingga disebut sebagai *cross-platform game engine*. Unity juga mendukung pengembangan *game* dalam dua dimensi (2D) maupun tiga dimensi (3D). Hingga saat ini, Unity mendukung berbagai macam platform, diantaranya windows, android, iOS, windows phone, Play Station 3, Play Station 4, Xbox one, Xbox 360, Ps Vita, Wii U, dan yang terbaru adalah Oculus Rift [2].

2.2 Algoritma A*

Algoritma A Star atau A* adalah algoritma yang digunakan untuk pencarian jalan tercepat (*pathfinding*) atau *graph traversal*, yaitu proses pencarian jalan tercepat melalui titik-titik atau node yang telah didefinisikan sebelumnya. Walaupun A* bukanlah algoritma yang selalu menemukan jalan tercepat, A* memiliki waktu pencarian jalan yang cepat dan hasil yang cukup akurat karena A* menggabungkan algoritma Djikstra dan *Greedy Best-First-Search* [3]. Metode pencarian jalur A* mirip dengan *Greedy Best-First-Search* dalam pemilihan *node* secara heuristik.

Dalam pemilihan *node* nya, A* akan memilih *node* dengan *vertex* yang dekat dengan titik awal, seperti Djikstra dan juga memilih *node* dengan *vertex* yang mendekati *goal*. Umumnya,

algoritma A* menggunakan dua *array* yang merepresentasikan biaya atau *cost* dari titik awal menuju titik manapun dan *array* yang merepresentasikan estimasi biaya dari *node* n menuju *goal*. Hubungan kedua array dirumuskan pada formula nomor 1.

$$f(n) = g(n) + h(n) \quad (1)$$

Keterangan:
f(n) = estimasi total biaya *node* ke n
g(n) = biaya dari *node* ke n ke *node* manapun
h(n) = estimasi biaya *node* n ke *goal*

Dalam setiap *loop*, A* akan mencari *vertex* yang memiliki estimasi total biaya terkecil dari *node* yang tersedia. Dalam pencarian jalurnya, A* memiliki dua *list* yang sering disebut *open list* dan *closed list*. *Open list* berisi mengenai *node* selanjutnya yang akan diamati, sedangkan *closed list* berisi data mengenai *node* yang telah diamati/dievaluasi sehingga *node* tersebut tidak perlu dievaluasi kembali.

2.3 A* Pathfinding Project

A* Pathfinding Project merupakan sebuah *project* yang dibuat oleh Aron Granberg yang bertujuan untuk membuat fungsi yang tidak tersedia pada Unity, yaitu fitur *pathfinding. Project* ini menggunakan algoritma A* sebagai dasar pencarian jalan yang digunakan. Dengan menggunakan A* Pathfinding Project, pengembang *game* Unity dapat menggunakan fungsi-fungsi umum yang biasa terdapat pada *pathfinding*, yaitu membuat *path* berdasarkan *grid* atau membuat *node* secara manual, dan memberikan fungsi *pathfinding agent* pada *GameObject* yang diinginkan [4].

2.4 Platformer

Platformer adalah salah satu *genre game* dimana mekanik permainan utamanya adalah adanya sebuah pijakan atau *platform*

dan gaya gravitasi. *Game* dengan genre ini biasanya memiliki karakter yang diarahkan oleh pemain untuk melewati berbagai rintangan seperti jurang dengan melompat pada *platform-platform* yang tersedia. Platformer umumnya menampilkan elemen-elemen permainan dan karakternya dari sisi samping. Elemen utama dalam permainan platformer adalah adanya aksi melompat baik itu karakter utama maupun karakter musuh. Contoh *game* dengan *genre platformer* yaitu: Super mario, MegaMan, Castlevania, dan sebagainya.

2.5 Tingkat Keberanian

Tingkat Keberanian atau *bravery level* adalah fitur *AI* yang diimplementasikan. Tingkat Keberanian digambarkan dengan sebuah variabel angka bernilai positif atau negatif untuk menentukan perilaku agennya. Semakin tinggi tingkatnya, semakin berani agen untuk mendekat dan menyerang musuhnya. Sebaliknya, jika nilai variabel ini negatif, agen tidak akan menyerang dan berusaha menjauhi pemain hingga batas tertentu.

2.6 Ashes of Resurgnce

Ashes of Resurgence adalah sebuah game 2D bergenre platformer yang dibangun dengan menggunakan Unity. Game ini memiliki elemen – elemen *Role-Playing Game* (RPG) seperti HP, SP, *Attack*, dan sebagainya. Tujuan permainan ini adalah untuk mengarahkan karakter menuju titik terakhir. Dalam perjalanan menuju titik akhir, pemain akan dihadapkan dengan berbagai macam rintangan dan musuh yang semakin kuat seiring dengan kekuatan pemain. Semakin banyak pemain mengalahkan musuh, *level* pemain akan semakin sehingga musuh – musuh yang lebih kuat dapat dikalahkan. Permainan ini dapat dimainkan pada perangkat PC dengan sistem operasi Windows atau *mobile* dengan sistem operasi Android.

(Halaman ini sengaja dikosongkan)

BAB III
ANALISIS DAN PERANCANGAN

Bab ini akan membahas tentang analisis dan perancangan dari aplikasi yang digunakan untuk tugas akhir.

3.1 Analisis Sistem

Perangkat lunak akan dibangun dengan menggunakan *game engine* Unity karena Unity dapat mengimplementasikan berbagai macam fitur yang tidak terdapat pada *game engine* lain. Selain itu, *scripting* pada Unity memiliki cakupan yang sangat luas sehingga jenis modifikasi apapun yang dapat dilakukan melalui *scripting* dapat diimplementasikan oleh Unity. Kelebihan ini akan sangat membantu dalam implementasi sebuah konsep yang belum pernah dicoba sebelumnya.

Untuk menerapkan fitur *Pathfinding* yang akan digunakan musuh untuk mencari jalan, dibutuhkan algoritma yang dapat melakukan kalkulasi dengan cepat dengan tingkat akurasi yang cukup tinggi. Akurasi yang selalu sempurna tidak terlalu dibutuhkan, karena agen diharapkan menyerupai makhluk hidup, dimana terkadang dapat melakukan kesalahan. Untuk itu, digunakan algoritma A*.

3.2 Perancangan Perangkat Lunak

Subbab ini akan menjelaskan tentang rancangan – rancangan umum perangkat lunak yang dibangun pada tugas akhir. Rancangan ini meliputi deskripsi umum perangkat lunak, spesifikasi kebutuhan fungsional, spesifikasi kebutuhan non-fungsional, dan karakteristik pengguna perangkat lunak.

3.2.1 Deskripsi Umum Perangkat Lunak

Perangkat lunak yang dibangun adalah sebuah *game* 2D bergenre platformer. Elemen – elemen RPG seperti *Health Points* (HP), *Special Points* (SP), *Attack*, *Defense*, dan sebagainya

digunakan dalam *game* yang dibangun. Selain itu, permainan ini juga menambahkan kecerdasan buatan pada musuh yang akan mempengaruhi perilaku musuh tersebut. Tujuan dari permainan ini adalah mengarahkan pemain ke titik akhir dan mengalahkan *boss.* Pemain dinyatakan kalah ketika HP telah mencapai 0 atau pemain terjatuh ke jurang. Pemain dapat melakukan lompatan untuk mencapai tujuan dan serangkaian serangan yang dapat digunakan untuk mengalahkan musuh. Beberapa serangan pemain membutuhkan sejumlah SP untuk digunakan. Apabila SP tidak mencukupi, maka serangan tersebut tidak bisa digunakan. Permainan hanya dapat dimainkan oleh satu orang dan dapat dimainkan pada desktop dengan sistem operasi Windows dan pada perangkat *mobile* dengan sistem operasi android.

Untuk memulai permainan, pemain harus memilih tombol "new game" atau "load game" pada antarmuka **Main Menu**. Tombol "load game" akan muncul ketika permainan sudah pernah dimainkan sebelumnya. Jika tidak, tombol "new game" yang akan muncul. Setelah memilih *new game*, pemain harus memilih *class* dari tokoh yang akan dimainkan pada antarmuka **Class Select**. Tiap *class* memiliki *stats* dan jenis serangan masing-masing. Setelah itu, pemain harus menyelesaikan *stage* pertama. Kemudian, pemain akan dibawa ke antarmuka **World Map** dimana *stage* berikutnya dapat dipilih. Tiap *stage* harus diselesaikan jika ingin menyelesaikan seluruh permainan.

3.2.2 Spesifikasi Kebutuhan Fungsional

Kebutuhan fungsional dari permainan ini adalah sebagai berikut:

1. Terdapat karakter/tokoh yang dikendalikan oleh pemain.
2. Terdapat musuh yang dikendalikan sistem.
3. *Stage* dimana karakter pemain dan musuh ditempatkan.
4. Rintangan yang dapat diselesaikan dalam *stage.*
5. *Power up* untuk membantu pemain mencapai tujuan.

3.2.3 Spesifikasi Kebutuhan Non-Fungsional

Selain kebutuhan fungsional, terdapat juga beberapa kebutuhan non-fungsional yang dapat menunjang kualitas dari permainan, yaitu:

1. Performa sistem, dapat diukur berdasarkan *frame rate*. Semakin tinggi *frame rate* nya, maka permainan akan semakin mulus.
2. Waktu respon atau *response time*. Permainan harus dengan cepat memproses masukan dari pemain. Keterlambatan waktu respon akan berpengaruh secara signifikan kepada kepuasan pemain.
3. Kalkulasi AI. Perhitungan untuk agen AI membutuhkan waktu yang cepat agar musuh cepat merespon sesuai dengan keadaan.
4. Tidak sering mengalami *crash* atau *error.*
5. Animasi termasuk salah satu kebutuhan non-fungsional yang penting dalam sebuah permainan. Animasi yang buruk akan membuat pemain cepat bosan.

3.2.4 Karakteristik Pengguna

Dilihat dari deskripsi umum perangkat lunak yang dijelaskan sebelumnya, dapat disimpulkan bahwa pengguna yang akan memainkan permainan ini ada dua jenis, seperti yang tercantum pada Tabel 3.1.

Tabel 3.1 Karakteristik Pengguna

Nama Aktor	Tugas	Hak Akses Aplikasi	Kemampuan yang harus dimiliki
Pemain	Pihak yang menggunakan	Menggunakan aplikasi	Tidak ada
Pengemban	Pihak yang	Membuat,	Pemrograman

g	mengembangkan	mengubah, dan menggunakan aplikasi	dan pengembangan *game*

3.3 Perancangan Sistem

Tahap perancangan dalam subbab ini dibagi menjadi beberapa bagian yaitu perancangan mekanik permainan, perancangan antarmuka, dan perancangan kecerdasan buatan.

3.3.1 Perancangan Mekanik Permainan

Dalam aplikasi permainan yang dibangun dalam Tugas Akhir ini, mekanik permainan dibagi menjadi dua jenis, yaitu:

3.3.1.1 Mekanik Utama

Mekanik dasar atau mekanik utama dari permainan ini adalah adanya beberapa elemen berikut:

1. Karakter/Tokoh yang dikendalikan oleh pemain. Karakter ini dapat melakukan beberapa aksi, yaitu berjalan ke kiri dan ke kanan, melompat, dan menyerang.
2. Musuh yang dikendalikan sistem. Musuh memiliki satu aksi utama, yaitu menyerang.
3. *Stage* dimana karakter pemain dan musuh ditempatkan. *Stage* terdiri dari beberapa *platform* sebagai pijakan karakter pemain dan musuh.
4. Gravitasi. Karakter pemain dan musuh mengikuti gaya gravitasi ke sumbu-y negatif. Jika karakter atau musuh tidak menginjak sebuah *platform*, maka karakter/musuh tersebut akan jatuh.

5. Terdapat jarak antar *platform* pada *Stage* dimana karakter pemain dan musuh harus melompat jika ingin mencapai *platform* berikutnya.
6. Kondisi kalah, jika karakter pemain jatuh kedalam jurang atau terkena terlalu banyak serangan dari musuh.
7. Kondisi menang, jika karakter pemain telah berhasil mencapai titik akhir pada *Stage*.

3.3.1.2 Mekanik Tambahan

Untuk membuat aplikasi semakin menarik, ditambahkan mekanik lain untuk melengkapi mekanik utama dari aplikasi. Mekanik tersebut adalah:

3.3.1.2.1 Mekanik Karakter Pemain

1. Tipe serangan dalam permainan dibagi menjadi dua tipe, yaitu *physical* dan *magical*. Rumus perhitungan *damage* ditunjukkan pada formula nomor 2.

$$Damage = \left(\frac{enPAttack}{plPDefense}\right) + \left(\frac{enMAttack}{plMDefense}\right) \qquad (2)$$

keterangan:
Damage = Nilai total yang digunakan untuk mengurangi *health points.*
enPAttack = kekuatan serangan *physical* dari musuh.
plPDefense = kekuatan pertahanan *physical* pemain.
enMAttack = kekuatan serangan *magical* dari musuh.
plMDefense = kekuatan pertahanan *magical* pemain.

2. *Stats* untuk menentukan kekuatan tiap karakter pada permainan. *Stats* yang digunakan adalah:
    - *Level* yang menunjukkan tingkat kekuatan karakter;

- *Health Points* (HP) yang menunjukkan nyawa karakter;
- *Special Points* (SP) yang menunjukkan sisa energi yang dimiliki untuk melakukan serangan spesial;
- *Attack* yang menunjukkan kekuatan serangan bertipe *physical;*
- *Defense* yang menunjukkan resistansi serangan bertipe *physical;*
- *Magic Attack* yang menunjukkan kekuatan serangan bertipe *magical;*
- *Magic Defense* yang menunjukkan resistansi serangan bertipe *magical;*
- *Experience Points* (Exp) yang menunjukkan berapa banyak Exp yang dibutuhkan untuk menaikkan *Level;*
- *Weapon Skills* yang menunjukkan tingkat kemahiran karakter dalam menggunakan jenis senjata tertentu. *Skill* dapat ditingkatkan dengan menggunakan senjata atau *weapon* yang sesuai.
- Ketika *Level* naik, *stats* yang lain akan bertambah sesuai dengan *level-up* formula nomor 3.

$$NewStat = CurStat + \lfloor StatGrow \rfloor \quad \text{atau} \quad NewStat = CurStat + \lceil StatGrow \rceil \tag{3}$$

keterangan:
NewStat = Nilai *stat* setelah *level up.*
CurStat = Nilai *stat* sebelum *level up.*
StatGrow = variabel bernilai *float* yang ditentukan sebelumnya.

Setiap *level up*, akan digunakan salah satu dari dua formula tersebut yang dipilih secara acak. Selain *stats*, jumlah *exp* yang harus didapatkan untuk naik *level* juga bertambah, sesuai dengan formula nomor 4.

$$NewExp = level * 4 \quad (4)$$

dengan *NewExp* merupakan nilai *Exp* yang harus didapatkan untuk *level up*.

3. *Retries* yang menunjukkan berapa banyak kekalahan yang dapat ditoleransi sebelum mencapai *game over*.
4. *Burst.* Sebuah mekanisme dimana setiap kali karakter pemain terkena serangan, *burst points* akan bertambah. Apabila *burst points* telah mencapai maksimal, pemain dapat mengaktifkan *burst mode* dimana karakter pemain akan menjadi lebih kuat dalam periode waktu tertentu. Formula untuk penambahan *burst points* ditunjukkan pada formula nomor 5.

$$BP = CurBP + \left(\left(\frac{damage}{maxHP}\right) \times maxBurst\right) \quad (5)$$

keterangan:
BP = Nilai *Burst points* baru
CurBP = Nilai *Burst point* sebelum terkena *damage*
damage = Nilai *damage* yang diterima pemain
maxHP = Jumlah HP maksimal pemain
maxBurst = Konstanta untuk nilai *burst* maksimal. Bernilai 2000.

5. *Money.* Sebuah sistem keuangan yang didapatkan dari mengalahkan musuh. Jumlah uang yang

dikeluarkan bergantung pada jumlah maksimal HP dari musuh.

6. *Player Class.* Terdapat empat jenis *class* yang dapat dipilih, yaitu *mage, warrior, knight,* dan *ninja*. Tiap *class* memiliki perkembangan *stats* dan serangkaian serangan awal yang berbeda.
7. *Skills. Skills* adalah serangan spesial yang membutuhkan sejumlah SP. SP yang dibutuhkan untuk tiap *skill* disebut *Sp Cost*. Tiap *class* memiliki *Skill* awal yang berbeda, walaupun semua *Skill* dapat didapatkan dalam permainan. *Skill* didapatkan dengan cara meningkatkan *Weapon Skills* tertentu.
8. *Equipment.* Pemain dapat mengubah *equipment* atau *gear* dari karakter pemain. *Equipment* terdiri dari *head, body, leg, left weapon* dan *right weapon*. Tiap *equipment* memberikan tambahan pada *Stats* karakter.
9. *Abilities*, yaitu sekumpulan efek yang bisa didapatkan dengan menggunakan *Ability Points*. *Ability Points* didapatkan melalui kenaikan *Level* atau *Level Up*. *Ability* yang bisa didapatkan antara lain:
    - *Power+, ability* untuk menambah kekuatan serangan tertentu;
    - *Rapid+, ability* untuk meningkatkan *combo* atau menambah frekuensi serangan;
    - *Special,* berbagai macam *ability* yang tidak termasuk dua *ability* diatas, diantaranya adalah mempercepat penambahan *burst points,* mengurangi *Sp Cost* yang dibutuhkan, menambah HP dan SP, dan menambahkan *side effects* pada serangan.
10. *Items.* Terdapat beberapa *items* yang bisa didapat dari musuh atau dari toko. *Items* memiliki efek tertentu diantaranya:
    - Mengembalikan HP atau SP yang berkurang dengan jumlah tertentu;

- Menambah *Stats* secara permanen;
- Menambah jumlah *Retries*;
- Menambah *Burst Points.*

11. *World Map* dimana *stage* selanjutnya dapat dipilih. Pemain juga dapat melakukan perubahan pada karakter seperti mengganti *equip* atau *skill* dan membeli barang.
12. *Shop*. Terdapat toko yang menjual *weapons* dan *items* yang dapat dibeli dengan menggunakan *money.*

3.3.1.2.2 Mekanik Musuh

1. Musuh juga memiliki *Stats* dan *Level* yang berbeda-beda yang akan dijelaskan lebih lanjut pada bagian perancangan musuh.
2. Terdapat tiga *state* untuk tiap musuh selain *boss*, diantaranya:
    - Normal, dimana musuh tidak mengetahui keberadaan karakter pemain dan sedang melihat sekitarnya;
    - Waspada, dimana musuh mengetahui adanya karakter pemain di dekatnya, tetapi masih belum dapat memastikan lokasinya;
    - Beraksi, dimana musuh telah mengetahui lokasi karakter.
3. Kecerdasan buatan. Tiap musuh memiliki perilaku yang berbeda sesuai dengan tingkat keberanian masing-masing. Akan dijelaskan lebih lanjut pada bagian perancangan kecerdasan buatan.

3.3.1.2.3 Mekanik *Stage*

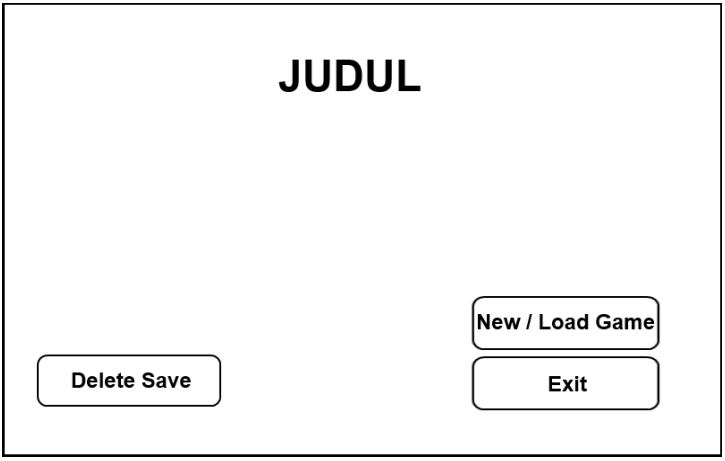
1. Terdapat *platform* yang bisa bergerak dengan sendirinya ketika karakter pemain menyentuh *platform* tersebut.
2. Terdapat *effect* pada *stage* tertentu dimana pergerakan pemain menjadi licin atau gravitasi menjadi berkurang.
3. Terdapat *obstacle* pada *stage* tertentu, diantaranya:
    - Bola salju, jika karakter pemain terkena bola salju, HP akan berkurang;
    - Magma, jika karakter pemain terkena magma, HP akan berkurang dan pemain terkena *effect burn*.
4. Terdapat beberapa *side effect* yang dapat mempengaruhi karakter, diantaranya:
    - *Stun*, *effect* yang membuat karakter tidak dapat melakukan apapun selama jangka waktu tertentu;
    - *Poison*, *effect* yang membuat HP dari karakter musuh terus berkurang dalam jangka waktu tertentu;
    - *Burn*, *effect* yang membuat HP dari karakter pemain terus berkurang dalam jangka waktu tertentu.

3.3.2 Perancangan Antarmuka

Pada subbab ini akan dijelaskan mengenai beberapa antarmuka pengguna yang dirancang untuk digunakan dalam perangkat lunak yang akan dibangun. Antarmuka pengguna tersebut antara lain adalah antarmuka *main menu*, antarmuka *class select*, antarmuka *inStage*, dan antarmuka *world map*. Detail tiap antarmuka yang disebutkan adalah sebagai berikut.

3.3.2.1 Antarmuka *Main Menu*

Rancangan antarmuka **Main Menu** bertujuan untuk menampilkan judul dari perangkat lunak beserta tombol untuk navigasi *main menu*, yaitu tiga tombol navigasi utama, **new game/load game** untuk memulai permainan, **exit** untuk keluar dari aplikasi, dan **delete save** untuk menghapus *save game* dan memulai permainan baru. **New game** akan muncul ketika pemain baru saja memainkan perangkat lunak untuk pertama kali atau setelah pemain menghapus *save game*. **Load game** akan menggantikan bagian tombol new game ketika pemain sudah pernah memainkan perangkat lunak dan *save game* tersedia pada aplikasi. Rancangan antarmuka **Main Menu** dapat dilihat pada gambar 3.1.

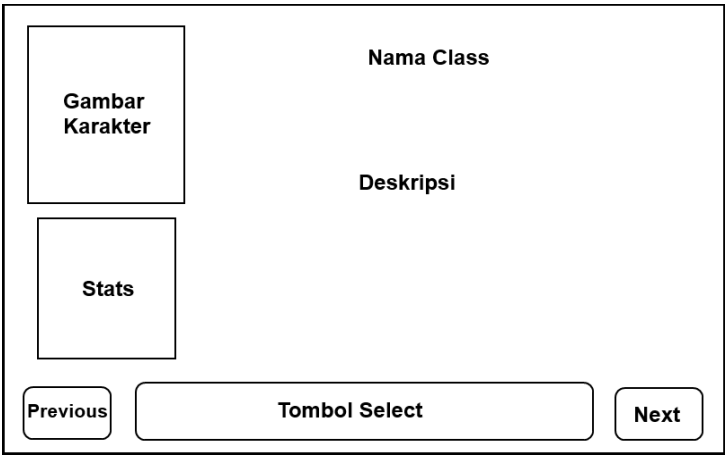


Gambar 3.1 Rancangan Antarmuka *Main Menu*

3.3.2.2 Antarmuka Class Select

Antarmuka **Class Select** adalah antarmuka dimana pemain memilih *class* dari tokoh yang akan dimainkan. *Class* hanya dapat dipilih pada antarmuka ini ketika pemain baru saja memainkan permainan atau setelah menekan tombol *new game*

pada antarmuka *main menu*. Elemen yang ditunjukkan pada antarmuka ini adalah penampilan tokoh dalam *class* tertentu, *stats,* beserta deskripsi tokoh. Tombol navigasi yang disediakan adalah tombol kiri dan kanan serta tombol *select* untuk memilih *class.* Rancangan antarmuka **Class Select** dapat dilihat pada Gambar 3.2.

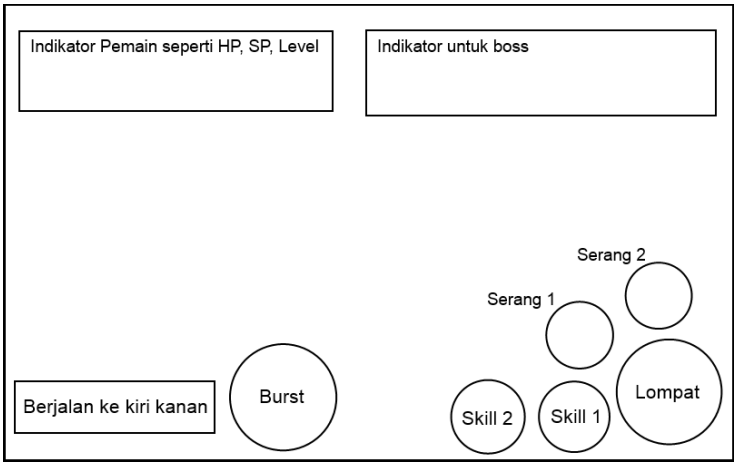


Gambar 3.2 Rancangan Antarmuka *Class Select*

3.3.2.3 Antarmuka *InStage*

Antarmuka *InStage* adalah antarmuka yang muncul ketika pemain berada dalam sebuah *stage*. Untuk menampilkan antarmuka ini, pemain yang baru melakukan permainan baru atau *new game* akan dibawa ke antarmuka ini setelah memilih *class* yang diinginkan pada antarmuka *class select.* Apabila pemain telah bermain sebelumnya dan memilih *load game*, maka antarmuka ini dapat diakses ketika pemain telah memilih *stage* yang tersedia pada antarmuka *world map*. Antarmuka ini menampilkan indikator untuk pemain dan *boss* dan juga berbagai macam tombol kontrol. Tombol kontrol terdiri dari tombol **berjalan, melompat, menyerang,** tombol ***skill,*** dan tombol

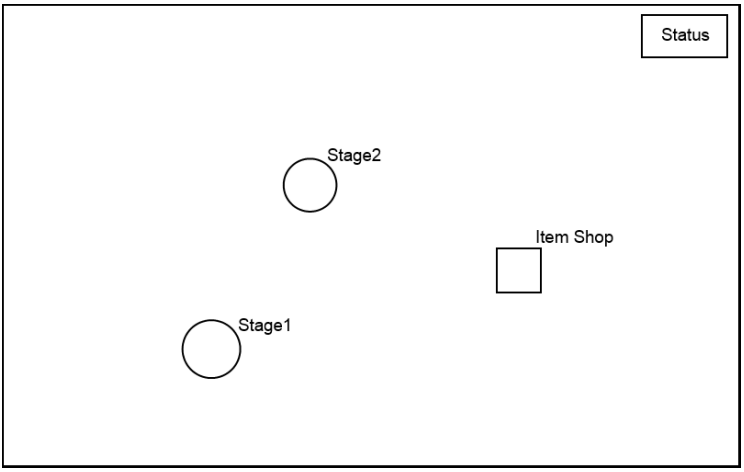
burst. Rancangan antarmuka *InStage* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Rancangan Antarmuka *InStage*

3.3.2.4 Antarmuka *World Map*

Antarmuka ini menampilkan keseluruhan *stage* yang dapat dimainkan oleh pemain beserta *shop* yang dapat dimasuki. Ikon *Stage* akan dibedakan berdasarkan jenis *stage* didalamnya. Ikon tersebut diantaranya adalah kaktus yang menggambarkan *stage* 1, rerumputan yang menggambarkan *stage* 2, pepohonan yang menggambarkan *stage* 3, pegunungan es yang menggambarkan *stage* 4, gunung berapi yang menggambarkan *stage* 5, dan roket luar angkasa yang menggambarkan *stage* 6. Ikon *stage* juga memiliki warna yang berbeda tergantung apakah *stage* tersebut sudah diselesaikan atau belum. *Stage* yang belum terselesaikan ikonnya akan berwarna merah dan *stage* yang telah terselesaikan berwarna hitam. *Stage* yang dapat dipilih tergantung pada *progress* pemain. Rancangan antarmuka *world map* dapat dilihat pada Gambar 3.4.



Gambar 3.4 Antarmuka *World Map*

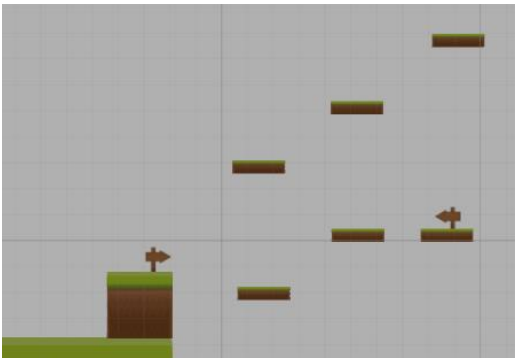
3.3.3 Perancangan *Pathfinding*

Fitur *Pathfinding* yang digunakan untuk musuh pada aplikasi ini dibuat dengan menggunakan A* *Pathfinding Project* yang dibuat oleh Aron Granberg. Fitur *Pathfinding* digunakan ketika aplikasi berada pada antarmuka **InStage**. Fitur ini diaplikasikan pada jenis musuh yang memiliki kemampuan untuk berjalan.

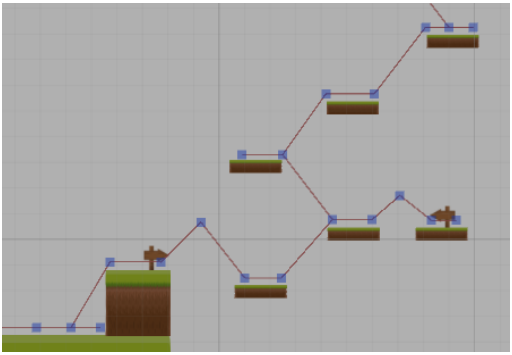
A* *Pathfinding Project* didesain untuk melakukan traversal pada medan dua dimensi (2D) maupun tiga dimensi (3D). Pada aplikasi ini, medan yang digunakan adalah 2D dan memiliki aturan yang berbeda dari *pathfinding* pada pengaturan *default* dari *Pathfinding Project* dikarenakan adanya unsur gravitasi, sehingga dilakukan modifikasi algoritma. Modifikasi tersebut antara lain:

1. Mengubah *path* yang sebelumnya berbentuk grid menjadi graf. Bentuk graf menyesuaikan dengan tatanan *platform*, ditunjukkan pada Gambar 3.5, di setiap *stage* [5]. *Path* diubah dengan cara menghapus

komponen *path* berbentuk graf dan membuat beberapa objek yang diletakkan pada *stage*. Objek – objek ini akan dimanfaatkan sebagai *node* pada graf. Setelah itu, dibuat peraturan bahwa jarak antar *node* memiliki batas maksimal, sehingga *node* yang berjarak terlalu jauh tidak terkoneksi satu sama lain. Hal ini dilakukan agar agen dapat melakukan navigasi pada graf yang tidak menyalahi aturan permainan. Hasil graf pada *stage* ditunjukkan pada Gambar 3.6.



Gambar 3.5 Tatanan *platform* dalam *stage*



Gambar 3.6 Tatanan graf pada *platform*

2. Navigasi pada graf untuk perbedaan koordinat y dibedakan menjadi dua aturan, yaitu:
    a. Jika koordinat y untuk *node* selanjutnya berada pada titik y yang lebih kecil daripada *node* sebelumnya, maka pergerakan hanya dilakukan pada sumbu x, dan agen akan "jatuh" dengan sendirinya ke titik y yang lebih rendah.
    b. Jika koordinat y untuk *node* selanjutnya lebih besar, maka agen akan melakukan lompatan dengan kecepatan yang sesuai dengan jarak yang harus dilompati. Perhitungan kecepatan menggunakan formula nomor 6.

$$y_{max} = {v_0}^2 sin^2(\theta)/(2g) \quad (6)$$

Y_{max} = tinggi maksimal lompatan
V_0 = kecepatan awal
g = percepatan gravitasi

3.3.4 Perancangan Stage

Dalam aplikasi, terdapat enam *level* atau *stage* yang dapat dimainkan secara berurutan. *Stage* tersebut diantaranya adalah *stage 1 Hoot Desert* yang bertema padang pasir, *stage 2 Syn's Garden* yang bertema padang rerumputan, *stage 3 Hollow Forest* yang bertema hutan seram, *stage 4 Crystal Mountain* yang bertema gunung es, *stage 5 The Volcano* yang bertema gunung berapi, dan *stage 6 Outer Space* yang bertema luar angkasa.

Skenario untuk tiap level akan dibuat secara manual tetapi akan mengikuti aturan tertentu agar tingkat kesulitan berawal dari mudah yang berangsur-angsur menjadi lebih susah. Aturan skenario level ditunjukkan pada Tabel 3.2.

Tabel 3.2 Aturan Skenario Level

Level/ Stage	Model Platform	Jenis Rintangan	Jenis Musuh
1	Jarak antar platform pendek	Tidak ada	Kecepatan rendah, serangan simpel
2	Jarak antar platform normal	Tidak ada	Kecepatan normal, serangan simpel
3	Jarak antar platform normal	Tidak ada	Kecepatan cepat, serangan beragam
4	Jarak antar platform normal, terdapat platform bergerak	Bola salju, platform licin	Serangan musuh susah dihindari
5	Jarak antar platform tinggi, mudah terjatuh	Magma, api	Serangan musuh susah dihindari, musuh banyak
6	Jarak antar platform normal, terdapat platform bergerak	Setengah gravitasi	Boss Terakhir

3.3.5 Perancangan Musuh

Karena terdapat beberapa *Stage*, musuh juga dibuat beragam dan memiliki berbagai macam metode serangan yang juga menyesuaikan dengan tingkat kesulitan desain dan rancangan tiap *stage*. Daftar musuh dapat dilihat pada Tabel 3.3. dan rancangan *stats* untuk tiap musuh dapat dilihat pada Tabel 3.4.

Tabel 3.3 Daftar Musuh

Nama Musuh	Pergerakan	Metode Serangan
Robo Warrior	Mendekati pemain	Menembakkan proyektil
Cannibos	Mendekati pemain	-
Necromancer	Menjauhi pemain	Membangkitkan Mangkorak
Mangkorak	Mendekati pemain	Melompat
Dragon Rider	Mendarat dan Terbang	Menembakkan api
Snowman	-	Mengeluarkan bola salju
EyeBeast	Mendekati pemain	Meledakkan diri
Daemabora	-	Mengeluarkan gelombang panas
Robo General (boss)	-	Menembakkan proyektil
Cannibos Chief (boss)	Melompat mendekati pemain	Memanggil Cannibos
Syn the Devourer (boss)	-	Menyemburkan api, Menggigit, Menembakkan bom api
Galros (boss)	Mendekati pemain, melompat	Menembakkan proyektil, melempar pedang
Arodam (boss)	-	Melempar batu, menembakkan sejumlah batu
The Catastrophe (boss)	-	Menembakkan proyektil, Mengeluarkan serangan listrik, api, dan udara, Hujan meteor

Tabel 3.4 Distribusi Stats Musuh

Nama Musuh	HP	Atk	Def	Magic Atk	Magic Def
Robo Warrior	200	10	7	5	5
Cannibos	150	20	10	7	12
Necromancer	120	1	10	1	20
Mangkorak	200	40	3	1	7
Dragon Rider	300	20	15	25	20
Snowman	500	1	14	1	14
EyeBeast	200	20	13	20	13
Daemabora	500	1	10	1	30
Robo General (boss)	900	15	14	10	12
Cannibos Chief (boss)	3500	40	20	14	14
Syn the Devourer (boss)	7000	60	24	50	24
Galros (boss)	12000	70	30	60	40

Arodam (boss)	25000	70	50	60	55
The Catastrophe (boss)	15000	40	25	40	25
The Catastrophe 2nd Form (boss)	30000	80	25	80	25
The Catastrophe 3rd Form (boss)	60000	120	50	160	50

3.3.6 Perancangan Tingkat Keberanian Musuh

Fitur tingkat keberanian adalah fitur yang akan mempengaruhi perilaku musuh ketika bertemu dengan karakter pemain. Dalam sistem tingkat keberanian, terdapat variabel yang disebut *Bravery Value* (BV) atau nilai keberanian yang merupakan hasil kalkulasi dari tingkat keberanian tiap musuh. Berdasarkan hasil eksplorasi dengan cara mencoba langsung dalam permainan, formula perhitungan BV ditunjukkan pada formula nomor 7.

$$BV = BL + (\frac{(EnemyLv - PlayerLv)^2}{2}) \text{ ;EnemyLv > PlayerLv}$$
$$BV = BL - (\frac{(EnemyLv - PlayerLv)^2}{2}) \text{ ;EnemyLv < PlayerLv} \quad (7)$$

Keterangan:
BV = *Bravery Value;*
BL = *Bravery Level* tiap musuh;
EnemyLv = Level musuh;
PlayerLv = Level pemain.

BV mengatur frekuensi serangan musuh dan kecepatan pergerakan musuh. Semakin besar BV, semakin cepat pergerakan musuh dan semakin tinggi frekuensi serangan, begitu pula sebaliknya. Apabila BV bernilai negatif, maka musuh tidak menyerang dan akan mencoba kabur dari jangkauan karakter pemain. Setiap musuh memiliki perilaku tersendiri untuk merespon tingkat keberanian. Perilaku untuk tiap musuh berdasarkan nilai BV dapat dilihat pada Tabel 3.5.

Tabel 3.5 Perilaku Musuh

Nama Musuh	Perilaku	
	BV positif	BV negatif
Robo Warrior	Mendekati pemain, menyerang	Menjauhi pemain
Cannibos	Mendekati pemain	Menjauhi pemain
Necromancer	Menjauhi pemain, menyerang	Menjauhi pemain
Mangkorak	Mendekati pemain	Mendekati pemain
Dragon Rider	Mendarat, menyerang	Terbang menjauh
Snowman	Menyerang	Menyerang
EyeBeast	Mendekati pemain, menyerang	Mendekati pemain, menyerang
Daemabora	Menyerang	Menyerang

3.3.7 Daftar Aset

Subbab ini akan menjelaskan tentang aset yang akan digunakan dalam perangkat lunak yang dibangun. Aset meliputi aset gambar dan suara. Daftar aset dapat dilihat pada Tabel 3.6.

Tabel 3.6 Daftar Aset

No.	Fungsi	Sumber
1.	Gambar Main Menu	https://www.nps.gov/features/yell/slidefile/fire/wildfire88/groundfire/page-3.htm
2.	Gambar Stage 1	http://opengameart.org/content/free-desert-platformer-tileset
3.	Gambar Stage 2	http://www.gameart2d.com/free-platformer-game-tileset.html
4.	Gambar Stage 3	http://rinnaki.deviantart.com/art/Haunted-Forest-303660090
5.	Gambar Stage 4	http://opengameart.org/content/winter-platformer-game-tileset
6.	Gambar Stage 5	http://opengameart.org/content/free-volcano-platform-tileset
7.	Gambar Stage 6	http://opengameart.org/content/space-backgrounds-7
8.	Gambar Stage 6	http://opengameart.org/content/asteroids
9.	Gambar Karakter Utama dan musuh Galros	http://happywithgame.com/2d-game-character-design.html
10.	Gambar Ability	http://opengameart.org/content/markeus-b-ui-buttons
11.	Gambar Shop	http://sweetmoon.deviantart.com/art/The-Armor-Shop-366287596
12.	Gambar musuh robot	http://www.gameart2d.com/character-spritesheet-10.html
13.	Gambar Arodam	http://opengameart.org/content/golem-animations
14.	Gambar Catastrophe	http://opengameart.org/content/3-form-rpg-boss-harlequin-epicycle
15.	Animasi efek	http://opengameart.org/content/animate

		d-particle-effects-2
16.	Animasi efek	http://www.gamedev.net/topic/586876-pow-studios-free-sprite-animations/
17.	Animasi ledakan	http://opengameart.org/content/wgstudio-explosion-animation
18.	Gambar efek	http://opengameart.org/content/shield-aura-effect
19.	Gambar musuh	http://opengameart.org/content/2d-rpg-enemy-set
20.	Gambar musuh	https://id.pinterest.com/tinyringh/2d-sprite/
21.	Gambar musuh	http://opengameart.org/content/dragon-animated-sprite
22.	Gambar musuh	http://opengameart.org/content/floating-eye-beast
23.	Efek suara	http://opengameart.org/content/fantasy-sound-effects-tinysized-sfx
24.	Efek suara	http://opengameart.org/content/coins-sound-effects-library
25.	Efek suara	http://opengameart.org/content/punches-hits-swords-and-squishes
26.	Efek suara	http://opengameart.org/content/rage-mode
27.	Efek suara	http://opengameart.org/content/death-sounds
28.	Efek suara	http://opengameart.org/content/male-gruntyelling-sounds
29	Gambar map	http://pintify.net/photo/503066220848583782

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini akan membahas mengenai implementasi pembuatan perangkat lunak sesuai dengan analisis dan perancangan perangkat lunak pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi dari Tugas Akhir ini ditunjukkan pada Tabel 4.1, Tabel 4.2, dan Tabel 4.3.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak (1)

Perangkat Keras	Prosesor: Intel(R) Core(TM) i5 CPU M 460 @ 2.53GHz Memori: 2048MB RAM
Perangkat Lunak	Sistem Operasi: Microsoft Windows 8.1 32-bit Perangkat Pengembang: Unity3D

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak (2)

Perangkat Keras	Prosesor: Intel(R) Dual-Core CPU @ 1GHz Memori: 1024MB RAM
Perangkat Lunak	Sistem Operasi: Android Versi 4.2.2 Jelly Bean

Tabel 4.3 Lingkungan Implementasi Perangkat Lunak (3)

Perangkat Keras	Prosesor: Intel(R) Quad-Core CPU @ 1.5GHz Cortex-A53 & Quad-Core CPU @ 2.0 GHz Cortex-A57 Memori: 2048MB RAM
Perangkat Lunak	Sistem Operasi: Android Versi 6.0 Marshmallow

4.2 Implementasi Alur Proses Aplikasi

Di subbab ini akan dijelaskan mengenai proses implementasi berdasarkan alur dari perancangan yang telah dibahas pada bab sebelumnya.

4.2.1 Implementasi Mekanik Permainan

4.2.1.1 Implementasi Mekanik Utama

Mekanik yang pertama adalah mekanik untuk karakter utama. Karakter utama memiliki kemampuan untuk berjalan ke kiri dan ke kanan, melompat, serta melakukan serangan. Untuk dapat mengimplementasikannya, digunakan *physics2D* dari *unity* khususnya komponen *RigidBody2D* dimana *RigidBody2D* menerapkan hukum gaya fisika pada permainan sehingga dapat diterapkannya aturan kecepatan, percepatan, gravitasi, dan sebagainya.

Gambar 4.1 Tampilan Karakter Utama
Sumber: Tabel 3.6. no. 9

Pada karakter utama, ditunjukkan pada Gambar 4.1, pergerakan ke kanan dan ke kiri dilakukan dengan memberikan kecepatan/*velocity* sebesar x atau -x kepada *gameobject* karakter kode sumber yang mengatur pergerakan kanan kiri pemain dapat dilihat pada kode sumber 4.1. Sedangkan untuk melompat, pada *gameobject* pemain akan ditambahkan sebuah gaya keatas sebesar y yang ditunjukkan pada kode sumber 4.2. Untuk aksi menyerang, serangan dibedakan menjadi 5 fungsi yang dibedakan berdasarkan

jenis senjata. Kode sumber menyerang ditunjukkan pada kode sumber 4.3.

```
if(recoil == 0 && (Input.GetKey(KeyCode.RightArrow) || PlayerContr
ollerButtons.rightHold) && isPaused == false)
   {
       GetComponent<Rigidbody2D>().velocity = new Vector2(walkSpee
d, GetComponent<Rigidbody2D>().velocity.y);
       h = 1;
       if (grounded && onWall == false)
           bodyAnimation.SetBool ("Walking", true);
   }
else if(recoil == 0 && (Input.GetKey(KeyCode.LeftArrow) || PlayerC
ontrollerButtons.LeftHold)  && isPaused == false)
   {
       GetComponent<Rigidbody2D>().velocity = new Vector2(-
1*walkSpeed, GetComponent<Rigidbody2D>().velocity.y);
       h = -1;
       if(onWall == false && grounded)
           bodyAnimation.SetBool ("Walking", true);
   }
```

Kode Sumber 4.1 Pergerakan Kiri Kanan Player

```
SoundEffect.isJump = true;
bodyAnimation.SetBool ("Jumping", true);
jumpTimes++;
GetComponent<Rigidbody2D>().velocity = new Vector2(GetComponent<Ri
gidbody2D>().velocity.x, 0);
GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpForce));
```

Kode Sumber 4.2 Lompatan Player

```
void WeaponAttack(int weaponIndex)
{
        if (weaponIndex == 1)
            ProjectileShoot ();
        else if (weaponIndex == 2)
            SwordSlash ();
        else if (weaponIndex == 3)
            SpearStab ();
        else if (weaponIndex == 4)
            ShurikenThrow ();
        else if (weaponIndex == 5)
            MagicRod ();
}
```

Kode Sumber 4.3 Fungsi Serangan Player

Selain tokoh pemain, musuh juga menggunakan komponen *Rigidbody2D* agar hukum fisika juga dapat diimplementasikan. Implementasi musuh akan dijelaskan secara lebih detail pada subbab 4.2.5.

Untuk implementasi *stage*, pada satu *scene* akan diletakkan sejumlah *platform* secara manual. Setiap *platform* memiliki komponen *Collider* dimana komponen ini membuat karakter-karakter yang ada pada permain berpijak pada setiap *platform.* Detail implementasi setiap *stage* akan dijelaskan pada subbab 4.2.4.

Kemudian, kondisi menang dalam permainan adalah apabila pemain berhasil mencapai *platform* terakhir yang ditentukan, dan mengalahkan *boss.* Pemain dianggap kalah apabila pemain gagal mencapai *platform* terakhir.

4.2.1.2 Implementasi Mekanik Tambahan

Subbab ini akan menjelaskan tentang proses implementasi mekanik tambahan yang telah direncanakan pada tahap perancangan. Mekanik tambahan dibagi menjadi tiga jenis, yaitu mekanik karakter pemain, mekanik musuh, dan mekanik *stage*.

4.2.1.2.1 Mekanik Karakter Pemain

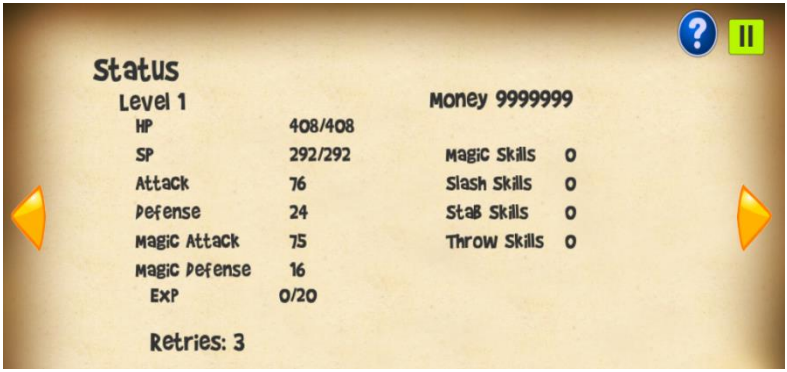
Dalam permainan, setiap serangan memiliki *damage* yang dibedakan menjadi dua, yaitu *physical* dan *magical*. Pembagian cukup dengan mendefinisikan tiap senjata dan serangan menjadi dua jenis, dan perhitungan *damage* juga dibedakan menjadi dua. Pembagian *Damage* senjata dapat dilihat pada kode sumber 4.4 dan perhitungan *damage* dapat dilihat pada kode sumber 4.5.

```
public static void Equip(bool handLocation, int weaponType, int
index)
{
        if (weaponType == 1)
        {
                //default fireball
                if (index == 1)
                {
                        //if right/primary
                        if (handLocation)
                        {
                                SpriteChange.pWeaponSprite =
magic[index-1];
                                pWeaponP = 0;
                                pWeaponM = 16;
                                Attack.rightHand = weaponType;
                        }
```

Kode Sumber 4.4 Pembagian Damage pada Senjata

```
public static void Damaged(int damage, int mDamage)
{
        int damagePoints = damage/Stats.def;
        int mDamagePoints = mDamage / Stats.mDef;
        int damageTot = damagePoints + mDamagePoints;
}
```

Kode Sumber 4.5 Perhitungan Damage



Gambar 4.2 Status

Mekanik lain yang ditambahkan adalah *stats* atau dapat disebut sebagai status, ditunjukkan pada Gambar 4.2, yaitu angka-angka yang menunjukkan nilai kekuatan dari pemain atau karakter utama yang dimainkan dan juga nilai kekuatan musuh. *Stats* mempengaruhi berbagai macam aspek dalam permainan, terutama aspek *damage*. *Stats* mempengaruhi jumlah *damage* yang didapat ketika pemain ataupun musuh menyerang. Seperti dilihat pada kode sumber 4.5, *damage* yang didapatkan akan semakin kecil apabila *stat def* atau *magic defense (mDef)* semakin besar. Kode sumber untuk perhitungan nilai *stats* ketika *level* upidapat dilihat pada kode sumber 4.6.

```
public static void LevelUp()
{
        Abilities.points += 2;
        HeadOverDisplay.levelUpNotif = true;
        level++;
        hp +=
Random.Range(PlayerClass.hpGrow/10,Mathf.CeilToInt(PlayerClass.hpG
row*1f/10)+1);
        Stats.hp +=
Random.Range(PlayerClass.hpGrow/10,Mathf.CeilToInt(PlayerClass.hpG
row*1f/10)+1);
        sp += Random.Range(PlayerClass.spGrow/10,
Mathf.CeilToInt(PlayerClass.spGrow*1f/10)+1);
        Stats.sp += Random.Range(PlayerClass.spGrow/10,
Mathf.CeilToInt(PlayerClass.spGrow*1f/10)+1);
        atk += Random.Range(PlayerClass.atkGrow/20,
Mathf.CeilToInt (PlayerClass.atkGrow*1f/20)+1);
        def += Random.Range(PlayerClass.defGrow/20,
Mathf.CeilToInt (PlayerClass.defGrow*1f/20)+1);
        mAtk += Random.Range(PlayerClass.mAtkGrow/20,
Mathf.CeilToInt (PlayerClass.mAtkGrow*1f/20)+1);
        mDef += Random.Range(PlayerClass.mDefGrow/20,
Mathf.CeilToInt (PlayerClass.mDefGrow*1f/20)+1);
        spd += Random.Range(1, 3);
        exp -= maxExp;
        maxExp += (level*4);
}
```

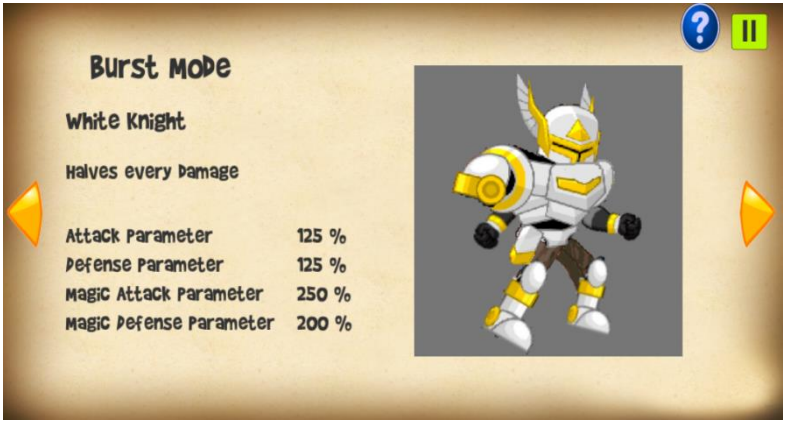
Kode Sumber 4.6 Perhitungan Stats

Selain *stats*, terdapat pula *retries* yang memiliki fungsi seperti *lives* pada permainan lain. *Retries* ditampilkan dalam

permainan, dan pemain akan *game over* ketika *retries* bernilai 0 dan pemain kalah jika *retries* masih diatas 0, maka pemain diberi kesempatan untuk bermain kembali. Kode sumber yang mengecek jumlah *retries* ketika kalah dapat dilihat pada kode sumber 4.7.

```
if(retries>0)
{
        Application.LoadLevel(Application.loadedLevel);
        retries--;
}
else
{
        StageProgress.justStarted = true;
        LoadingScene.SetNextLevel("Death");
        Application.LoadLevel("LoadingScene");
}
```

Kode Sumber 4.7 Pengecekan Retries



Gambar 4.3 Burst

Mekanik selanjutnya adalah mekanik *burst* dimana pemain akan mendapatkan kekuatan tambahan dalam jangka waktu tertentu. *Burst* akan bertambah jika pemain terkena *damage*. Gambar 4.3 Menunjukkan tampilan info *burst.* Kode

sumber 4.8 menunjukkan mekanisme penambahan *burst*, dan kode sumber 4.9. menunjukkan mekanisme aktivasi *burst.*

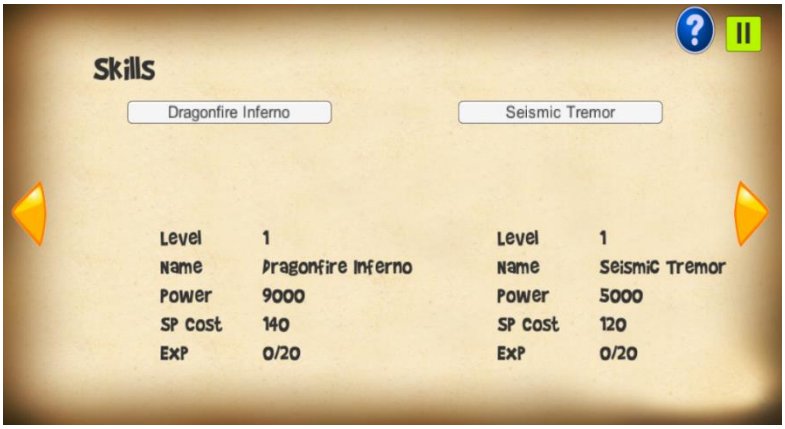
```
public static void GainBurst(int value)
{
        if(Burst.isBurst == false)
        {
                burst += Mathf.CeilToInt
(value*MultiplierVariables.hyperCharge*1.0f / 100*1.0f);
                if (burst>=2000)
                {
                        HeadOverDisplay.burstAvailable = true;
                        burst = 2000;
                }
        }
}
```

Kode Sumber 4.8 Mekanisme Penambahan Burst

```
public static void BurstOn (int burstIndex)
{
        MultiplierVariables.recoilPoints = 0;
        HeadOverDisplay.burstAvailable = false;
        Player.timeToChangeToBurstArmor = true;
        if(isBurst == false)
                justActivated = true;
        isBurst = true;
        Stats.maxHp *= 2;
        Stats.hp += Stats.maxHp/2;
        Stats.maxSp *= 2;
        Stats.sp += Stats.maxSp/2;
        Stats.atk = (Stats.atk*atk)/100;
        Stats.def = (Stats.def*def)/100;
        Stats.mAtk = (Stats.mAtk*mAtk)/100;
        Stats.mDef = (Stats.mDef*mDef)/100;
        if (burstIndex == 1)
                MagicalSurge ();
        else if (burstIndex == 2)
                Berserk ();
        else if (burstIndex == 3)
                ColossalArmor ();
        else if (burstIndex == 4)
                Trickster ();
        else if (burstIndex == 5)
```

```
                AutoDieEnemy ();
        Else
                Debug.Log("Burst not found.");
}
```

Kode Sumber 4.9 Mekanisme Aktivasi Burst

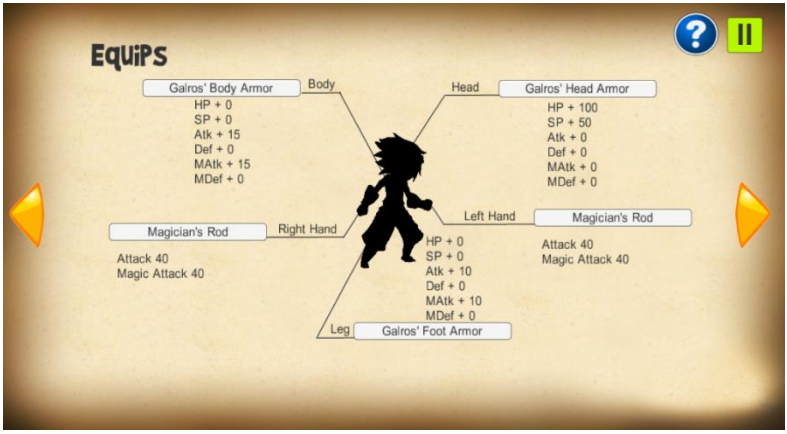


Gambar 4.4 Skills

Selain metode menyerang dengan senjata, pemain dapat melakukan kemampuan tertentu atau *skill*. Setiap *skill* memiliki kekuatan sendiri yang ikut bertambah seiring dengan *level* pemain. Tampilan info *skill* ditunjukkan pada Gambar 4.4. Perhitungan kekuatan *skill* dapat dilihat pada kode sumber 4.10.

```
power [1] = 10 * Stats.mAtk * (2 + level [1]);
power [2] = 10 * Stats.atk * (4 + level [2]);
power [3] = 10 * Stats.def * (6 + level [3]);
power [4] = 10 * (Stats.atk * (level [4]));
power [5] = 10 * Stats.mAtk * (6 + level [5]);
power [6] = 10 * Stats.atk * (6 + level [6]);
power [7] = 10 * Stats.atk * (2 + level [7]);
power [8] = 10 * Stats.atk * (5 + level [8]);
power [9] = 10 * Stats.mAtk * (4 + level [9]);
power [10] = 10 * Stats.atk * (6 + (2*level [10]));
```

Kode Sumber 4.10 Perhitungan Kekuatan Skill



Gambar 4.5 Equipment

Equipment yang dapat dimanfaatkan oleh pemain adalah *head*, *body*, *leg*, serta senjata untuk tangan kiri dan kanan. Setiap *equipment* memiliki *boost* tersendiri untuk tiap *stat.* Kemudian, keseluruhan *boost* dari *equipment* akan disimpan dalam variabel yang akan ditambahkan pada *stats.* Gambar 4.5 menunjukkan tampilan info *equipment*. Kode sumbernya dapat dilihat pada kode sumber 4.11 dan 4.12.

```
public static void FinalCalculate()
{
        hp = hHp + bHp + lHp;
        sp = hSp + bSp + lSp;
        atk = hAtk + bAtk + lAtk;
        def = hDef + bDef + lDef;
        mAtk = hMAtk + bMAtk + lMAtk;
        mDef = hMDef + bMDef + lMDef;
}
```

Kode Sumber 4.11 Equipment Boost

```
public static void SetStats()
{
        damageDurationMax = 100;
        maxHp = (MultiplierVariables.hp * Player.hp)/100 +
Equipment.hp;
        maxSp = (MultiplierVariables.sp * Player.sp)/100 +
Equipment.sp;
```

```
        atk = (MultiplierVariables.atk * Player.atk)/100 +
Equipment.atk;
        def = (MultiplierVariables.def * Player.def)/100 +
Equipment.def;
        mAtk = (MultiplierVariables.mAtk * Player.mAtk)/100 +
Equipment.mAtk;
        mDef = (MultiplierVariables.mDef * Player.mDef)/100 +
Equipment.mDef;
        weight = Mathf.RoundToInt(Equipment.weight*1f/3);
        charisma = Equipment.charisma;
        if(Options.isInStage == false)
        {
                hp = maxHp;
                sp = maxSp;
        }
}
```

Kode Sumber 4.12 Perhitungan Total Stats

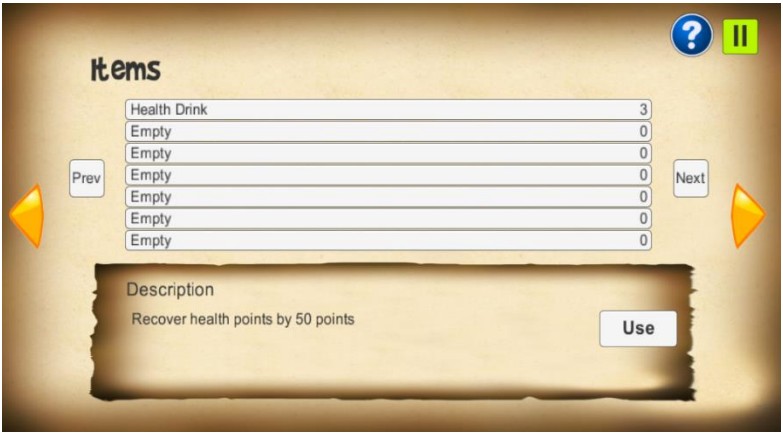
Gambar 4.6 Abilities
Sumber: Tabel 3.6. no. 10

Mekanik lain yang diimplementasikan adalah *Abilities*, ditunjukkan pada Gambar 4.6. Mekanisme *Abilities* adalah mengubah variabel-variabel yang mempengaruhi berbagai aspek dalam permainan yang disebut *MultiplierVariables*. Variabel-variabel ini memiliki nilai *default* dan nilainya akan berubah

ketika sebuah *ability* didapatkan. Kode sumber nilai *default MultiplierVariables* dapat dilihat pada kode sumber 4.13.

```
//abilities
public static int multiplyManaRegen = 1;
public static int revenge = 0;
public static int damageConverter = 0;
public static int hyperCharge = 100;
public static int spAdapter = 0;
public static int hpAdapter = 0;
public static int expAdapter = 100;
public static int energySaver = 100;
public static int hyperTension = 0;
public static int bodyRegeneration = 0;
public static int balancer = 0;
public static bool strongBulk = false;
public static int  stunningTrick = 0;
public static int venomAttack = 0;
```

Kode Sumber 4.13 Nilai Default MultiplierVariables



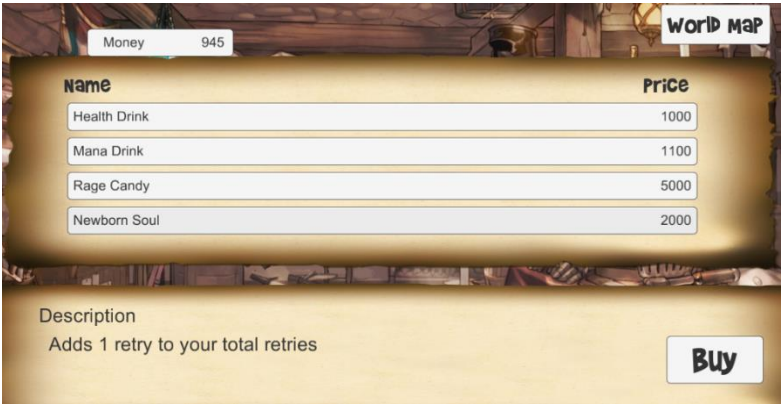
Gambar 4.7 Items

Items juga terdapat dalam permainan. Terdapat empat jenis *item*, yaitu HP *Recovery*, *SP Recovery*, *Stat Boost*, dan *Special*. HP *Recovery* mengembalikan sejumlah HP yang hilang, SP *Recovery* mengembalikan sejumlah SP, *Stat Boost* menambahkan nilai *stats* tertentu secara permanen, dan *special*

memiliki berbagai fungsi yaitu menambah *level*, *retries*, dan *burst*. Gambar 4.7 Menunjukkan tampilan info *item*. Kode sumber untuk *item* kategori *special* dapat dilihat pada kode sumber 4.14.

```
else if(itemType == 4)
{
        if(itemIndex == 1)
                Stats.GainExp(Player.maxExp-Player.exp);
        else if(itemIndex == 2)
        {
                if(Stats.burst >= 2000)
                        return false;
                Stats.GainBurst(400);
        }
        else if(itemIndex == 3)
        {
                Stats.retries++;
                retriesUsed = true;
        }
        else
                return false;
        return true;
}
```

Kode Sumber 4.14 Penggunaan Special Item



Gambar 4.8 Item Shop
Sumber: Tabel 3.6. no. 11

Toko atau *Shop* juga diimplementasikan dalam permainan yang ditunjukkan pada Gambar 4.8. Terdapat dua jenis toko, yaitu

Item Shop dan *Weapon Shop*. Kode sumber pembelian *item* dapat dilihat pada kode sumber 4.15 dan kode sumber penambahan *item* kedalam *inventory* dapat dilihat pada kode sumber 4.16.

```
public void Buy(int itemType, int itemIndex, int price)
{
        if(Player.money >= price)
        {
                if(Items.GetItem(itemType, itemIndex))
                        Player.money -= price;
        }
}
```

Kode Sumber 4.15 Pembelian Item

```
public static bool GetItem(int itemType, int itemIndex)
{
        PlayerCollections dummy = new PlayerCollections ();
        foreach(PlayerCollections col in Player.playerItems)
        {
                if (col.type == itemType)
                {
                        if (col.index == itemIndex)
                        {
                                if (col.quantity < 99)
                                {
                                        col.quantity++;
                                        return true;
                                }
                                else
                                {
                                        return false;
                                }
                        }
                }
        }
        Debug.Log("Dont have item");
        dummy = new PlayerCollections();
        dummy.type = itemType;
        dummy.index = itemIndex;
        dummy.quantity = 1;
        Player.playerItems.AddLast (dummy);
        return true;
}
```

Kode Sumber 4.16 Penambahan Item kedalam Inventory

4.2.1.2.2 Mekanik Musuh

Selain pemain, musuh juga memiliki *level* dan *stats*. Perbedaannya, musuh tidak memiliki *exp* sendiri sehingga *level* musuh akan ditentukan secara manual. *Exp* yang dimiliki musuh adalah *exp reward* untuk pemain ketika berhasil mengalahkan musuh. Kode sumber perhitungan *stats* musuh dapat dilihat pada kode sumber 4.17.

```
void SetStats()
{
        maxHp += level * Mathf.CeilToInt(baseHp*1f/20);
        atk += level * Mathf.CeilToInt(baseAtk*1f/20);
        def += level * Mathf.CeilToInt(baseDef*1f/20);
        mAtk += level * Mathf.CeilToInt(baseMAtk*1f/20);
        mDef += level * Mathf.CeilToInt(baseMDef*1f/20);
        exp +=  level * Mathf.CeilToInt(exp*1f/6);
        this.hp = this.maxHp;
}
```

Kode Sumber 4.17 Perhitungan Stats Musuh

Mekanik lain yang ditambahkan pada musuh adalah *alert system*. Musuh memiliki tiga kondisi *alert system* yang ditampilkan dengan tanda seru "!" yaitu normal, waspada, dan beraksi. Sistem ini diimplementasikan dengan cara membuat sebuah *Collider* pada musuh yang berfungsi sebagai jarak pandang musuh. Jika pemain masuk kedalam jarak pandang musuh, maka musuh akan segera beraksi. Gambar 4.9 menunjukkan tentang tampilan *alert system.* Kode sumber jarak pandang musuh dapat dilihat pada kode sumber 4.18.

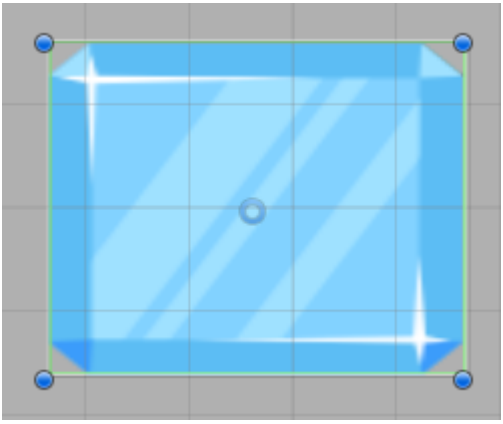
Gambar 4.9 Tiga Kondisi Musuh
Sumber: Tabel 3.6. no. 12

```
void OnTriggerEnter2D (Collider2D col)
{
        if(col.gameObject.tag == "Player" && inRange == false)
        {
        inRange = true;
        lockedTarget = true;
        alerted = true;
        if(hasPathfinding)
                GetComponentInParent<AILerp>().enabled = true;
        }
}
```

Kode Sumber 4.18 Pemain Masuk Jarak Pandang Musuh

4.2.1.2.3 Mekanik Stage

Pada beberapa *stage*, terdapat mekanik tambahan untuk menambah variasi permainan *stage*. Salah satunya adalah *moving platform* atau *platform* bergerak. *Platform* ini dibuat agar ketika pemain menyentuhnya, *platform* akan bergerak menuju titik yang telah ditentukan. Gambar 4.10 menunjukkan contoh *moving platform* dalam permainan. Kode sumber *moving platform* ditunjukkan pada kode sumber 4.19.

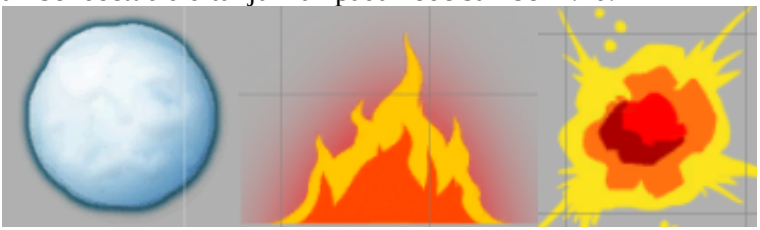


Gambar 4. 10 Platform Bergerak
Sumber: Tabel 3.6. no. 5

```
void OnCollisionEnter2D(Collision2D col)
{
        if (col.gameObject.tag == "Player")
        {
                timeToMove = true;
                InvokeRepeating ("ResetPos", 15f, 15f);
        }
}
void ResetPos()
{
        timeToMove = false;
        this.transform.position = defaultPos;
        this.GetComponent<Rigidbody2D>().velocity = new
Vector2(0,0);
}
```

Kode Sumber 4.19 Platform Bergerak

Selain itu, mekanik lain yang diimplementasikan adalah *obstacle*. *Obstacle* yang telah diimplementasikan berupa bola salju dan magma yang ditampilkan pada Gambar 4.11. Apabila pemain terkena, HP akan berkurang dengan jumlah tertentu. Kode sumber *obstacle* ditunjukkan pada kode sumber 4.20.



Gambar 4.11 Obstacle
Sumber: Tabel 3.6. no. 4, no. 15, no. 16

```
void OnCollisionEnter2D (Collision2D col)
{
        if(col.gameObject.tag == "Player")
        {
        Stats.Damaged(this.GetComponent<Obstacle>().GetDamage(),0)
;
                SoundEffect.isExplosionBig = true;
                Instantiate(explosionBig, transform.position,
```

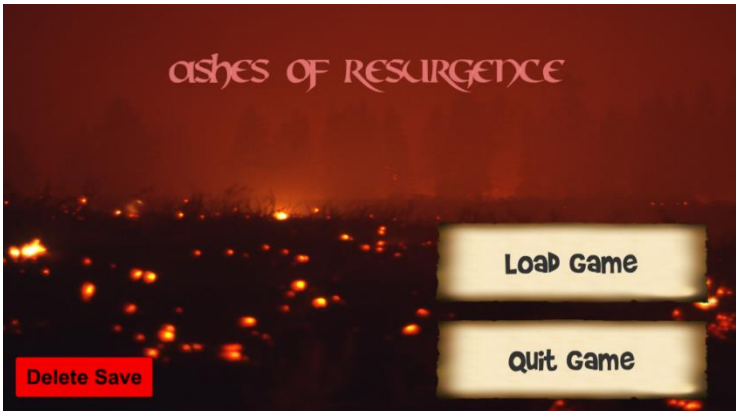
```
Quaternion.Euler(new Vector3(0,0,0)));
        Destroy(gameObject);
        }
        else if(col.gameObject.tag == "Ground" ||
col.gameObject.tag == "Enemy")
        {
                SoundEffect.isExplosionSmall = true;
                Destroy(gameObject);
        }
}
```

Kode Sumber 4.20 Obstacle

4.2.2 Implementasi Antarmuka

Subbab ini menjelaskan mengenai proses implementasi rancangan antarmuka kedalam perangkat lunak.

4.2.2.1 Antarmuka Main Menu



Gambar 4.12 Antarmuka Main Menu
Sumber: Tabel 3.6. no. 1

Antarmuka **Main Menu**, ditunjukkan pada Gambar 4.12, adalah antarmuka pertama yang muncul ketika permainan baru dibuka. Terdapat tiga tombol yang dapat digunakan pada antarmuka ini yaitu New/Load Game dengan kode sumber 4.21,

Quit Game dengan kode sumber 4.22, dan Delete Save dengan kode sumber 4.23.

```
void OnMouseDown()
{
        if(PlayerPrefs.HasKey("lv"))
        {
                SaveLoadData.LoadGame ();
                GameObject loadingScreen = Instantiate
(loadingBar, transform.position, transform.rotation) as
GameObject;

        loadingScreen.GetComponent<LoadingBar>().SetNextLevel("Wor
ldMap");
        }
        else
        {
                GameObject loadingScreen = Instantiate
(loadingBar, transform.position, transform.rotation) as
GameObject;

        loadingScreen.GetComponent<LoadingBar>().SetNextLevel("Cla
ssSelect");
        }
}
```

Kode Sumber 4.21 Load Game

```
void OnMouseDown()
{
        Application.Quit ();
}
```

Kode Sumber 4.22 Quit Game

```
void OnMouseDown()
{
        HeadOverDisplay.burstAvailable = false;
        WorldMap.numberOfStage = 6;
        WorldMap.stageFinished = 0;
        WorldMap.alreadyFromMap = false;
        PlayerPrefs.DeleteAll ();
        LoadGame.hasDeleted = true;
}
```

Kode Sumber 4.23 Delete Save

4.2.2.2 Antarmuka Class Select

Gambar 4.13 Antarmuka Class Select

Antarmuka **Class Select**, ditunjukkan pada Gambar 4.13, adalah antarmuka yang muncul setelah pemain memilih New Game pada antarmuka Main Menu. Terdapat dua tombol utama yaitu tombol navigasi dan tombol pilih atau *select*. Kode sumber untuk tombol navigasi ditunjukkan pada kode sumber 4.24 dan kode sumber untuk *select* ditunjukkan pada kode sumber 4.25.

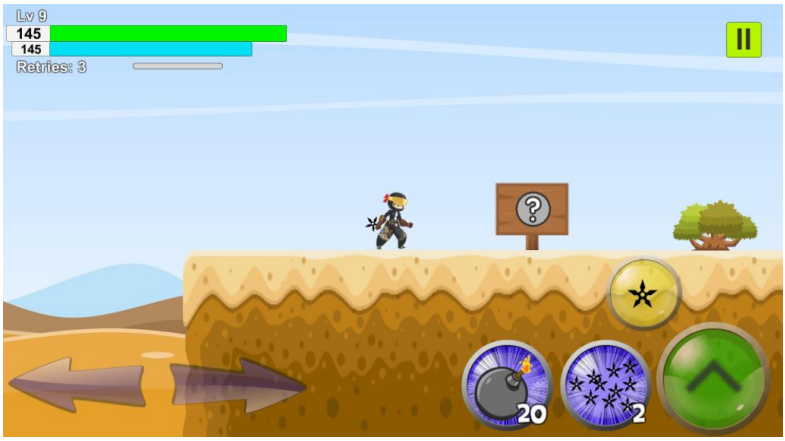
```
if (Input.GetKeyDown (KeyCode.LeftArrow) || LeftButton.leftPress)
{
        LeftButton.leftPress = false;
        classIndex--;
        if(classIndex == 0)
                classIndex = 4+cheat;
        BaseStats (classIndex);
}
else if (Input.GetKeyDown (KeyCode.RightArrow) ||
RightButton.rightPress)
{       RightButton.rightPress = false;
        classIndex++;
        if(classIndex == 5+cheat)
                classIndex = 1;
        BaseStats (classIndex);}
```

Kode Sumber 4.24 Navigasi Class Select

```
else if (hasNotSelected && (Input.GetKeyDown(KeyCode.Space) ||
SelectButton.selPress))
{
        hasNotSelected = false;
        Player.SetWeaponSkills(classIndex);
        GameObject loadingScreen = Instantiate (loadingBar,
transform.position, transform.rotation) as GameObject;
        loadingScreen.GetComponent<LoadingBar>().SetNextLevel("Sta
ge1 - Hoot Desert");
        Skill.skillIndex = classIndex;
        Skill.AddSkill(classIndex);
        Skill.skillIndex2 = 0;
        Stats.burstIndex = classIndex;
        Burst.InitBurstInfo (Stats.burstIndex);
        PlayerCollections dummy = new PlayerCollections ();
        dummy.type = 1;
        dummy.index = 1;
        dummy.quantity = 3;
        Player.playerItems.AddLast (dummy);
```

Kode Sumber 4.25 Select Class

4.2.2.3 Antarmuka InStage



Gambar 4.14 Antarmuka InStage
Sumber: Tabel 3.6. no. 2

Antarmuka **InStage**, ditunjukkan pada Gambar 4.14, adalah antarmuka yang ditampilkan ketika pemain berada dalam

sebuah *stage* yang dipilih sebelumnya. Pada antarmuka inilah permainan utama terjadi. Pemain, musuh, *platform*, dan *obstacle* terletak pada antarmuka ini. Selain itu, antarmuka ini menampilkan sejumlah elemen selain elemen utama. Terdapat beberapa hal yang ditampilkan, diantaranya Gambar 4.15, 4.16, 4.17, 4.18, dan 4.19.

Gambar 4.15 Indikator

Gambar 4.15 menunjukkan tentang tampilan indikator dalam permainan. Indikator menampilkan beberapa informasi penting mengenai karakter yang dimainkan seperti HP bar, SP bar, dan sebagainya. Kode sumber bar indikator ditunjukkan pada kode sumber 4.26.

```
levelIndicator.text = "Lv " + Player.level;
hpIndicator.text = ""+hp;
this.healthBar.value = Mathf.Lerp(this.healthBar.value, (hp *
1.0f)/(maxHp * 1.0f), Time.deltaTime * 8);
if((hp * 1.0f)/(maxHp * 1.0f) > 0.5f)
        this.healthBarFill.color = Color.Lerp(Color.yellow,
Color.green, ((hp-(maxHp/2)) * 1.0f)/(maxHp/2 * 1.0f));
else
        this.healthBarFill.color = Color.Lerp(Color.red,
Color.yellow, (hp * 1.0f)/(maxHp/2 * 1.0f));
spIndicator.text = "" + sp;
this.specialBar.value = Mathf.Lerp(this.specialBar.value, (sp *
1.0f)/(maxSp * 1.0f), Time.deltaTime * 8);
this.burstBar.value = Mathf.Lerp(this.burstBar.value, (burst *
1.0f) / (2000 * 1.0f), Time.deltaTime * 8);
```

Kode Sumber 4.26 Indikator Pemain

Selain indikator, terdapat pula tombol untuk mengendalikan aksi dan pergerakan pemain, yaitu tombol kontrol pemain pada Gambar 4.16, tombol untuk melompat pada Gambar 4.17, tombol menyerang pada Gambar 4.18, dan tombol mengaktifkan *skill* pada Gambar 4.19.

Gambar 4.16 Tombol Pergerakan Pemain
Sumber: Tabel 3.6. no. 29

Gambar 4.17 Tombol Lompat
Sumber: Tabel 3.6. no. 29

Gambar 4.18 Tombol Menyerang
Sumber: Tabel 3.6. no. 29

Gambar 4.19 Tombol "Skill"
Sumber: Tabel 3.6. no. 29

4.2.2.4 Antarmuka World Map

Gambar 4.20 Antarmuka World Map
Sumber: Tabel 3.6. no. 29

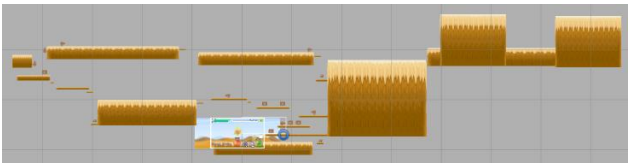
Antarmuka **World Map**, ditunjukkan pada Gambar 4.20, adalah antarmuka yang menampilkan seluruh *stage* yang dapat dipilih beserta *shop* yang tersedia. Pada antarmuka ini, perubahan *skill* dan *equipment* juga dapat dilakukan. Selain itu, *ability* juga dapat di-*upgrade* pada antarmuka ini. Kode sumber untuk masuk kedalam *stage* maupun *shop* ditunjukkan pada kode sumber 4.27.

```
void OnMouseDown()
{
        SoundEffectMenus.isSelectButton = true;
        if (isEnterButton)
                stageInfo.GetComponent<StageInfo> ().AccessScene
(buttonFunction);
        else
                stageInfo.GetComponent<StageInfo> ().Cancel ();
}
```

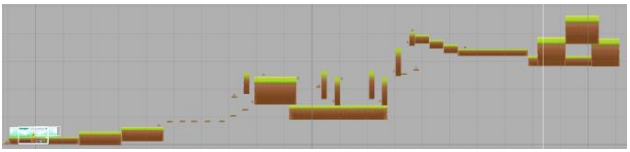
Kode Sumber 4.27 Memilih Stage

4.2.3 Implementasi Stage

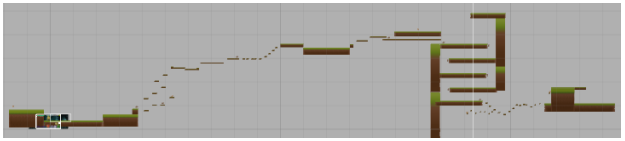
Stage dibuat dengan cara meletakkan *platform* secara manual tetapi masih dalam jangkauan lompatan pemain sehingga suatu *stage* dapat diselesaikan. Terdapat 6 *stage* pada permainan, dengan *stage* 1 bertema padang pasir, *stage 2* bertema rerumputan, *stage 3* bertema hutan, *stage 4* bertema es, *stage 5* bertema gunung berapi, dan *stage 6* bertema luar angkasa. Gambar 4.21 Menampilkan *layout stage* 1, Gambar 4.22 Menampilkan *layout stage* 2, Gambar 4.23 Menampilkan *layout stage* 3, Gambar 4.24 Menampilkan *layout stage* 4, Gambar 4.25 Menampilkan *layout stage* 5, dan Gambar 4.26 Menampilkan *layout stage* 6.



Gambar 4.21 Layout Stage 1
Sumber: Tabel 3.6. no. 2

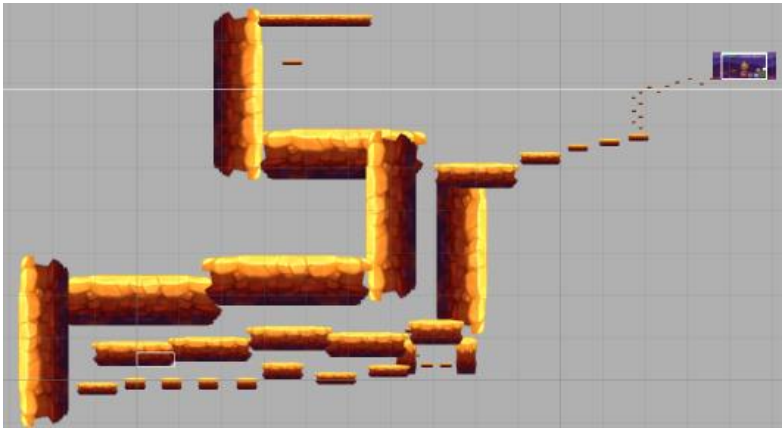


Gambar 4.22 Layout Stage 2
Sumber: Tabel 3.6. no. 3

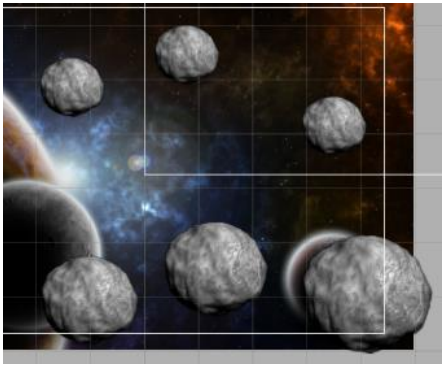


Gambar 4.23 Layout Stage 3
Sumber: Tabel 3.6. no. 4

Gambar 4.24 Layout Stage 4
Sumber: Tabel 3.6. no. 5



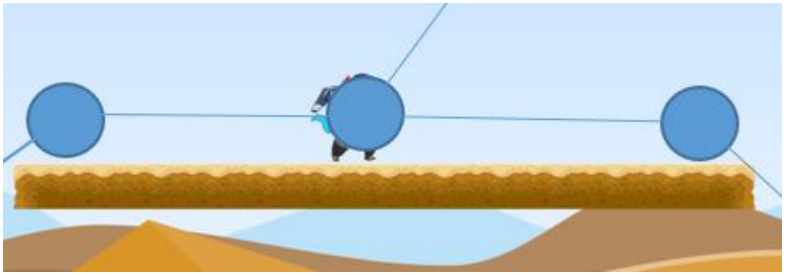
Gambar 4.25 Layout Stage 5
Sumber: Tabel 3.6. no. 6



Gambar 4.26 Layout Stage 6
Sumber: Tabel 3.6. no. 7, no. 8

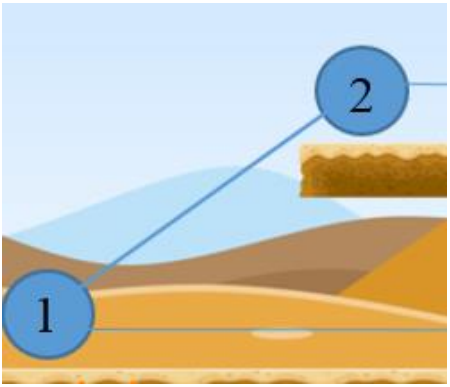
4.2.4 Implementasi Pathfinding

Pathfinding menggunakan algoritma A* dari A* pathfinding project oleh Aron Granberg. Project ini memiliki beberapa pendekatan dalam *pathfinding*. Pada aplikasi ini, fitur *pathfinding* yang digunakan adalah *Lerp*, dimana *lerp* mencari *node* terdekat dari target sehingga *pathfinding* lebih dinamis. Terdapat tiga kondisi yang berbeda dalam algoritma pencarian jalan. Kondisi tersebut adalah:



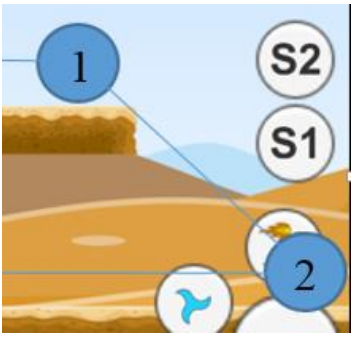
Gambar 4.27 Kondisi Pathfinding 1

Pada Kondisi 1, ditunjukkan pada Gambar 4.27, agen akan menelusuri *platform* dengan kecepatan (x,0). Tidak terdapat gaya pada sumbu y dikarenakan *path* selanjutnya berada pada posisi y yang sama.



Gambar 4.28 Kondisi Pathfinding 2

Pada Kondisi 2, ditunjukkan pada Gambar 4.28, agen akan berjalan dengan sebuah gaya (x,y). Nilai x dan y didapatkan dengan menghitung jarak antara titik 1 dan 2, kemudian melakukan kalkulasi untuk mendapatkan gaya lompat yang sesuai.



Gambar 4.29 Kondisi Pathfinding 3

Pada Kondisi 3, ditunjukkan pada Gambar 4.29, agen akan berjalan dengan sebuah gaya (x,y) seperti pada kondisi 1. Perbedaan tinggi akan menyesuaikan secara otomatis karena adanya gaya gravitasi.

Kode sumber untuk fitur *pathfinding* yang dijelaskan diatas ditunjukkan pada kode sumber 4.27.

```
if(nodeNow.x != path.vectorPath[currentWaypointIndex].x)
{
int dir = 1;
if(nodeNow.x > path.vectorPath[currentWaypointIndex].x)
        dir = -1;
if(nodeNow.y >= path.vectorPath[currentWaypointIndex].y)
{
        GetComponent<Rigidbody2D>().velocity = new
Vector2(dir*speed,GetComponent<Rigidbody2D>().velocity.y);
}
```

```
else{   if(nodeNow.y + 100 <
path.vectorPath[currentWaypointIndex].y)
        jumpforceY = 2200;
                else if(nodeNow.y + 200
<path.vectorPath[currentWaypointIndex].y)
        jumpforceY = 3200;
                else
        jumpforceY = 4200;
                if(nodeNow.x + 100 <
path.vectorPath[currentWaypointIndex].x)
        jumpforceX = 600;
                else
        jumpforceX = 1250;
                float gravity =
this.GetComponent<Rigidbody2D>().gravityScale;
                float maxDistance =
Mathf.Abs(path.vectorPath[currentWaypointIndex].x) -
Mathf.Abs(nodeNow.x);
                maxDistance = Mathf.Abs(maxDistance) * 2;
                float maxHeight =
Mathf.Abs(path.vectorPath[currentWaypointIndex].y) -
Mathf.Abs(nodeNow.y);
                jumpforceY = Mathf.Sqrt(2 * gravity * maxHeight);
                float time = 2 * jumpforceY / gravity;
                jumpforceX = maxDistance / time;
                jumpforceX *= 3;
                jumpforceY *= 5;
        GetComponent<Rigidbody2D>().velocity = new
Vector2(dir*jumpforceX,jumpforceY);
        }
}
```

Kode Sumber 4.28 Pathfinding

4.2.5 Implementasi Musuh

Seperti yang dijelaskan pada bab 3.3.5, terdapat beberapa musuh sebagai tantangan pemain yang memiliki pola pergerakan yang berbeda-beda. Musuh dengan pola yang lebih rumit akan diletakkan pada *stage* yang lebih sulit. Musuh dalam permainan antara lain:

Gambar 4.30 Karakter Musuh Robo Warrior
Sumber: Tabel 3.6. no. 12

Robo Warrior adalah musuh pertama yang muncul dalam permainan. Robo Warrior dapat berjalan dan menembakkan sebuah roket. Gambar 4.30 menunjukkan tampilan Robo Warrior dan kode sumber Robo Warrior ditunjukkan pada kode sumber 4.28.

```
void Update () {
        if(timeToAttack == 0)
        {
                timeToAttackMax = 8 -
(GetComponent<Bravery>().GetBraveryValue()*1f/10);
                if(timeToAttackMax <= 0)
                        timeToAttackMax = 0.5f;
                if (this.GetComponent<Enemy> ().IsInRange())
                {
                        int probability = Random.Range(1,101);
                        if(probability <=
GetComponent<Bravery>().GetBraveryValue())
                                Attack();
        timeToAttack = Time.time + timeToAttackMax;
                }
        }
        if (this.GetComponent<Enemy> ().IsInRange() &&
timeToAttack < Time.time)
        {
                Attack();
                timeToAttack = Time.time + timeToAttackMax;
        }
}
```

Kode Sumber 4.29 Robo Warrior

Gambar 4.31 Karakter Musuh Cannibos
Sumber: Tabel 3.6. no. 20

Cannibos pertama kali muncul pada *stage* 2. Cannibos memiliki kemampuan berjalan dan berusaha untuk menyentuh pemain. Gambar 4.31 menunjukkan tampilan Cannibos dan kode sumber Cannibos dapat dilihat pada kode sumber 4.29.

```
void Update () {
        if(this.GetComponent<Enemy>().IsFacingRight())
                GetComponent<Rigidbody2D>().velocity = new
Vector2(moveSpeed, GetComponent<Rigidbody2D>().velocity.y);
        Else
                GetComponent<Rigidbody2D>().velocity = new
Vector2(-moveSpeed,GetComponent<Rigidbody2D>().velocity.y);
}
```

Kode Sumber 4.30 Cannibos

Gambar 4.32 Karakter Musuh Necromancer
Sumber: Tabel 3.6. no. 20

Necromancer muncul pada *stage 3*. Necromancer berusaha untuk menjauh dari jangkauan serangan pemain dan

membangkitkan mangkorak untuk menyerang pemain. Gambar 4.32 menunjukkan tampilan Necromancer dan kode sumber Necromancer ditunjukkan pada kode sumber 4.30.

```
void Update () {
        if(locked == false && this.GetComponent<Enemy>
().IsInRange())
                locked = true;
        if(waiting == 0)
        {
                nextAttack = 10 -
(GetComponent<Bravery>().GetBraveryValue()*1f/10);
                if(nextAttack <= 0)
                        nextAttack = 0.1f;
                if (this.GetComponent<Enemy> ().IsInRange())
                {
                        int probability = Random.Range(1,101);
                        if(probability <=
GetComponent<Bravery>().GetBraveryValue())
                                Summon();
                        waiting = Time.time + nextAttack;
                }
        }
        if (locked && summons == null)
        {
                if(waiting < Time.time)
                {
                        {
                                Summon ();
                        }
                        waiting = Time.time + nextAttack;
                }
        }
        if (this.GetComponent<Enemy>().IsInRange())
        {
        if(this.GetComponent<Enemy>().IsFacingRight())

        GetComponent<Rigidbody2D>().velocity = new Vector2(-
moveSpeed, GetComponent<Rigidbody2D>().velocity.y);
        else
        GetComponent<Rigidbody2D>().velocity = new
Vector2(moveSpeed, GetComponent<Rigidbody2D>().velocity.y);
        }
}
```

Kode Sumber 4.31 Necromancer

Gambar 4.33 Karakter Musuh Mangkorak
Sumber: Tabel 3.6. no. 20

Mangkorak berusaha mendekati dan melompati pemain. Gambar 4.33 menunjukkan tampilan Mangkorak dan kode sumber mangkorak ditunjukkan pada kode sumber 4.31.

```
void Update () {
        if (this.GetComponent<Enemy>().IsInRange() && grounded)
                Jump();
}
```

Kode Sumber 4.32 Mangkorak

Gambar 4.34 Karakter Musuh Dragon Rider
Sumber: Tabel 3.6. no. 19

Secara *default*, Dragon Rider terbang diatas *platform* dan ketika pemain telah terlihat, Dragon Rider mendarat dan menyerang dengan menembakkan api secara horizontal kearah pemain. Serangan api Dragon Rider dapat meledak dan mengeluarkan percikan api apabila mengenai *platform* atau objek

yang terdapat pada *stage*. Gambar 4.34 menunjukkan tampilan Dragon Rider dan kode sumber Dragon Rider dapat dilihat pada kode sumber 4.32.

```
void Update () {
        if(timeToAttack == 0)
        {
                nextAttack = 8 -
(GetComponent<Bravery>().GetBraveryValue()*1f/15);
                if(nextAttack <= 0)
                        nextAttack = 0.4f;
        }
        if (this.GetComponent<Enemy> ().IsInRange () &&
GetComponent<Bravery>().GetBraveryValue() > -100)
        {
                if (grounded == false)
                {
                        Land ();
                }
                else if (timeToAttack < Time.time)
                {
                        Attack ();
                        timeToAttack = Time.time + nextAttack;
                }
        }
        else if(grounded ||
GetComponent<Bravery>().GetBraveryValue() <= -100)
                Fly ();
}
```

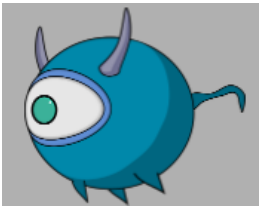
Kode Sumber 4.33 Dragon Rider

Gambar 4.35 Karakter Musuh Snowman
Sumber: Tabel 3.6. no. 5

Snowman merupakan musuh utama pada *stage 4*. Snowman tidak dapat bergerak dan menyerang dengan mengeluarkan bola salju raksasa. Gambar 4.35 menunjukkan tampilan Snowman dan kode sumber Snowman ditunjukkan pada kode sumber 4.33.

```
void Update () {
        if(TimeToAttack<Time.time)
        {
                Attack();
                TimeToAttack = Time.time + 2;
        }
}
```

Kode Sumber 4.34 Snowman



Gambar 4.36 Karakter Musuh EyeBeast
Sumber: Tabel 3.6. no. 22

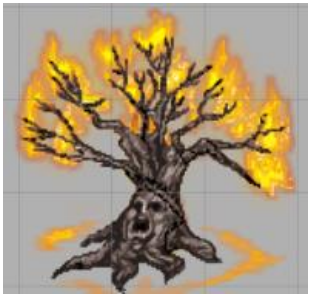
EyeBeast adalah musuh yang muncul pada *Stage 5*. Ketika EyeBeast melihat pemain, EyeBeast akan mendekati pemain dan meledakkan diri ketika menyentuh sesuatu. Gambar 4.36 menunjukkan tampilan EyeBeast dan kode sumber EyeBeast ditunjukkan pada kode sumber 4.34.

```
void Update () {
        if(GetComponent<Enemy>().IsStunned())
                this.GetComponent<Rigidbody2D>().velocity = new
Vector2 (0,0);
        else
        {
        if(fov.GetComponent<EnemyRange>().GetALertStatus() &&
        attackPosition == false)
                {
```

Kode Sumber 4.35 EyeBeast

```
                this.GetComponent<Rigidbody2D>().velocity = new
Vector2 (direction*400,0);
                if(transform.position.x >= patrolMax && direction
== 1)
                        direction = -1;
                else if(transform.position.x <= patrolMin &&
direction == -1)
                        direction = 1;
                }
        if(attackRange.GetComponent<EnemyRange>().GetRange() &&
attackPosition == false)
                {
                        flyingDir = player.position -
transform.position;
                        startLocation = new Vector2
(transform.position.x * direction * (-1), transform.position.y);
        this.GetComponent<Rigidbody2D>().velocity = flyingDir*3;
                        attackPosition = true;
                }
                else if(attackPosition == true)
                {
                this.GetComponent<Rigidbody2D>().velocity =
flyingDir*3;
                }
        }
}
```

Kode Sumber 4.35 EyeBeast (lanjutan)



Gambar 4.37 Karakter Musuh Daemabora
Sumber: Tabel 3.6. no. 19

Daemabora berlokasi di *stage 5*. Ketika pemain berada dalam jangkauannya, Daemabora mengeluarkan *heatwave* secara

berulang-ulang. Gambar 4.37 menunjukkan tampilan Daemabora dan kode sumber Daemabora ditunjukkan pada kode sumber 4.35.

```
void Update () {
        if(hasActive == false &&
range.GetComponent<EnemyRange>().GetALertStatus())
        {
                bodyAnim.Play("DaemaboraActivate");
                hasActive = true;
        }
        if(hasActive && timeToAttack<Time.time)
        {
                heatWaveGenerate();
                timeToAttack = Time.time + 2;
        }
}
```

Kode Sumber 4.36 Daemabora

Gambar 4.38 Karakter Musuh Robo General
Sumber: Tabel 3.6. no. 12

Robo General merupakan *boss* dari *stage 1*. Robo General tidak berjalan dan menembakkan roket besar yang berubah arah. Gambar 4.38 menunjukkan tampilan Robo General dan kode sumber Robo General ditunjukkan pada kode sumber 4.36.

```
void Update () {
        if (timeToAttack < Time.time &&
GetComponent<Enemy>().IsDead() == false)
```

Kode Sumber 4.37 Robo General

```
        {
                Attack();
                timeToAttack = Time.time + timeToAttackMax;
        }
}
```

Kode Sumber 4.37 Robo General (lanjutan)

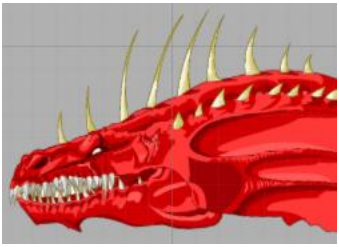
Gambar 4.39 Karakter Musuh Cannibos Chief
Sumber: Tabel 3.6. no. 20

Cannibos Chief merupakan *boss* dari *stage 2*. Cannibos chief melakukan lompatan dan memanggil 5 Cannibos untuk menyerang. Gambar 4.39 menunjukkan tampilan Cannibos Chief dan kode sumber Cannibos Chief ditunjukkan pada kode sumber 4.37.

```
void Update () {
        if(GetComponent<Enemy>().IsDead() == false)
        {
                if(timeSpawn < Time.time)
                {
                        for(int i = 0; i < 5; i++)
                        {
                                if(cannibos[i] == null)
                                {
                                        Attack(i);
                                        timeSpawn = Time.time +
2;
                                        break;
                                }
                                timeSpawn = Time.time + 2;
                        }
                }
```

```
        }
        else
        {
                Stats.isDamaged = true;
                for(int i = 0; i < 5; i++)
                {
                        if(cannibos[i] != null)
                        {
                        Destroy(cannibos[i].gameObject);
                        }
                }
        }
}
```

Kode Sumber 4.38 Cannibos Chief



Gambar 4.40 Karakter Musuh Syn the Devourer
Sumber: Tabel 3.6. no. 21

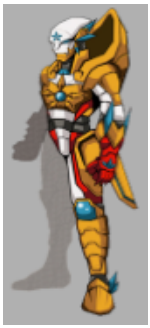
Syn adalah *boss* dari *stage 3.* Syn memiliki tiga metode serangan, yaitu menggigit, menyemburkan api, dan mengeluarkan granat api. Gambar 4.40 menunjukkan tampilan Syn dan kode sumber Syn ditunjukkan pada kode sumber 4.38.

```
void Update () {
        if(Time.timeScale != 0 && GetComponent<Enemy>().IsDead()
== false)
        {
                if(timeToAttack == 0)
                {
                        if(attackArray == 1)
                        {
                                if
```

```
(this.gameObject.transform.rotation.z > 0.1f ||
this.gameObject.transform.rotation.z < -0.1f)
== 3)
                        {
                                        rotateDir *= -1;
        this.gameObject.transform.Rotate (new Vector3 (0, 0,
0.2f*rotateDir));
        this.GetComponent<Animator>().Play("BossSynFireBreath");
                                if (delay == 0)
                                        Attack1 ();
                                else
                                        delay--;
                        }
                        else if(attackArray == 2)
                        {
                                Attack2();
                                attackDuration = 0;
                        }
                        else if(attackArray
                                if(mangap == false)
                                {
        this.gameObject.GetComponent<UnityEngine.UI.Image>
().sprite = attackSprite;
                                        mangap = true;
                                }
        this.gameObject.transform.position = new
Vector2(this.transform.position.x - 20*rotateDir,
this.transform.position.y);
        this.gameObject.transform.Rotate (new Vector3 (0, 0,
0.4f*rotateDir));
                                if
(this.gameObject.transform.position.x < (defaultX - 500))// ||
this.gameObject.transform.rotation.z < -0.1f)
                                {
        this.gameObject.GetComponent<UnityEngine.UI.Image>
().sprite = defaultSprite;
                                        rotateDir *= -1;
                                }
                                else
if(this.gameObject.transform.position.x >= defaultX)
                                {
                                        mangap = false;
                                        attackDuration = 0;
                                }
                        }
                        if(attackDuration == 0)
                        {
                                rotateDir = 1;
```

```
                                attackArray = Random.Range(1,4);
        this.gameObject.GetComponent<UnityEngine.UI.Image>
().sprite = defaultSprite;
        this.gameObject.transform.rotation = Quaternion.Euler( new
Vector3(0,0,5));
                                attackDuration =
attackDurationMax;
                                timeToAttack = timeToAttackMax;
                        }
                        else
                                attackDuration--;
                }
                else
                        timeToAttack--;
        }
}
```

Kode Sumber 4.39 Syn

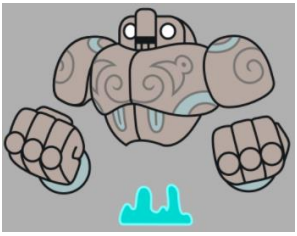


Gambar 4.41 Karakter Musuh Galros
Sumber: Tabel 3.6. no. 9

Galros merupakan *boss* dari *stage 4*. Galros menyerang dengan menembakkan roket yang memantul jika terkena *wall* dan melempar pedang yang juga akan memantul ketika mengenai *wall*. Galros dapat melakukan *dash* menuju kearah pemain dan melompat dengan tinggi yang fleksibel. Gambar 4.41 menunjukkan tampilan Galros dan kode sumber Galros ditunjukkan pada kode sumber 4.39.

```
void Update () {
        if(GetComponent<Enemy>().IsDead() == false)
        {
                if(haveShot == false &&
GetComponent<Rigidbody2D>().velocity.y < 0)
                        ShootBouncingBullet();
                if(waiting == 0)
                {
                        if(attackDeterminate <= 100)
                                HyperDash();
                        else if(attackDeterminate <= 250)
                                JumpAttack();
                        else
                                BackDash();
                        waiting = Random.Range(50,300);
                        attackDeterminate = waiting;
                }
                else
                        waiting--;
        }
}
```

Kode Sumber 4.40 Galros



Gambar 4.42 Karakter Musuh Arodam
Sumber: Tabel 3.6. no. 13

Arodam merupakan *boss* dari *stage 5.* Arodam tidak bergerak dan memiliki tiga metode menyerang, yaitu melempar batu, melempar 3 batu, dan menembakkan batu terarah. Gambar 4.42 menunjukkan tampilan Arodam dan kode sumber Arodam dapat dilihat pada kode sumber 4.40.

```
void Update () {
        if(timeToReadyAttack < Time.time)
        {
                bodyAnim.Play("ArodamReadyingAttack");
        }
        if(timeToAttack < Time.time)
        {
                bodyAnim.Play("ArodamAttack");
                if(next == 1)
                        ThrowScatterRock();
                else if(next == 2)
                        StartCoroutine(HomingRock());
                else
                        ThrowTripleRock();
                timeToReadyAttack = Time.time + Random.Range(1,4);
                NextAttack();
        }
}
```

Kode Sumber 4.41 Arodam

Gambar 4.43 Karakter Musuh The Catastrophe
Sumber: Tabel 3.6. no. 14

The Catastrophe adalah *boss* dari *stage 6* sekaligus merupakan *final boss* permainan. The Catastrophe memiliki 3 wujud dan dapat melakukan 5 metode serangan yang berbeda dengan rincian Wujud pertama dapat melakukan 1 metode menyerang, wujud kedua dapat melakukan 4 metode menyerang, dan wujud ketiga dapat melakukan 5 metode menyerang. Gambar 4.43 menunjukkan tampilan The Catastrophe dan kode sumber The Catastrophe ditunjukkan pada kode sumber 4.41.

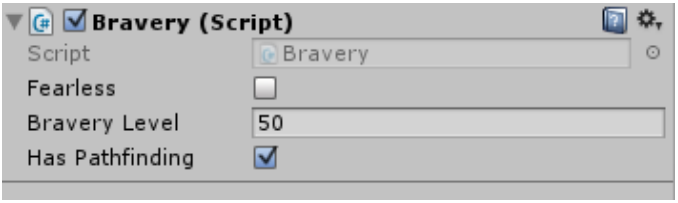
```
void Update(){
        if(Player.playerDead)
                planetaryDevastation = false;
        if(waitTimeForNextAttack < Time.time)
        {       ShootBlueFlame();
                waitTimeForNextAttack = Time.time + waitPlus;
        }
        if(planetaryDevastation == true && planetDVWait <
Time.time)
        {       planetaryDevastation = false;
                darkAura.SetActive(false);
        }
        if(formNow > 1 && waitTimeForNextTB < Time.time)
        {       if(attackType == 1)
                        electricalSurge.SetActive(true);
                else if(attackType == 2)
                        windSurge.SetActive(true);
                else if(attackType == 3)
                        fireSurge.SetActive(true);
                else if(attackType == 4)
                {
                        electricalSurge.SetActive(true);
                        windSurge.SetActive(true);
                        fireSurge.SetActive(true);
                }
        }
        if(formNow > 1 && nextSurge < Time.time)
        {       if(attackType == 1)
                {
                        ShootThunderBolt();
                        waitTimeForNextTB = Time.time +
Random.Range(3,waitPlus2);
                        nextSurge = waitTimeForNextTB + 1.5f;
                }
                else if(attackType == 2)
                {
                        ShootWindBlade();
                        waitTimeForNextTB = Time.time +
Random.Range(3,waitPlus2);
                        nextSurge = waitTimeForNextTB + 1.5f;
                }
                else if(attackType == 3)
                {
                        ShootFlameThrower();
                        waitTimeForNextTB = Time.time +
Random.Range(3,waitPlus2);
                        nextSurge = waitTimeForNextTB + 1.5f;
                }
```

```
                else if(attackType == 4)
                {
                        SummonPlanetaryDevastation();
                        waitTimeForNextTB = Time.time +
planetaryDevastationDuration + Random.Range(3,waitPlus2);
                        nextSurge = waitTimeForNextTB + 1.5f;
                }
                if(formNow == 3)
                {
                        attackType = Random.Range(1,5);
                                if(attackType == 4)
                                nextSurge = Time.time + 3;
                        }
                        else
                        attackType = Random.Range(1,4);
        }
        if(formShouldBe > 1)
        {
                catastrpoheForm.SetBool ("Form", true);
                if(formShouldBe == 3)
                        catastrpoheForm.SetBool ("Form2",
true);
        }
        if(formNow == 1 && formShouldBe == 2)
        {
                catastrpoheForm.SetBool ("Form", true);
                formNow = 2;
                nextSurge = Time.time + 1;
                attackType = Random.Range(1,4);
                if(attackType == 1)
                electricalSurge.SetActive(true);
                else if(attackType == 2)
                        windSurge.SetActive(true);
                else if(attackType == 3)
                        fireSurge.SetActive(true);
        }
        else if(formNow == 2 && formShouldBe == 3)
        {
                waitPlus = 8;
                waitPlus2 = 6;
                catastrpoheForm.SetBool ("Form2", true);
                formNow = 3;
                attackType = 4;
                nextSurge = waitTimeForNextTB +3;
        }
}
```

Kode Sumber 4.42 The Catastrophe

4.2.6 Implementasi Tingkat Keberanian Musuh

Tingkat Keberanian Musuh diimplementasikan dengan rumus *bravery* pada bab 3.3.6. sebagai formula. Setiap musuh memiliki komponen yang disebut *Bravery* dimana komponen ini mengatur sistem tingkat keberanian. Pada *editor*, setiap musuh memiliki nilai awal *bravery* yang disebut *bravery level. Bravery level* merupakan faktor input lain selain jarak antar level musuh dan pemain yang digunakan untuk menghitung nilai akhir dari *bravery.* Semakin tinggi *bravery level*, musuh akan semakin berani melawan pemain dengan level yang lebih tinggi. Gambar 4.44 menunjukkan tampilan komponen *bravery* pada *editor.* dan kode sumber *Bravery* dapat dilihat pada kode sumber 4.43.



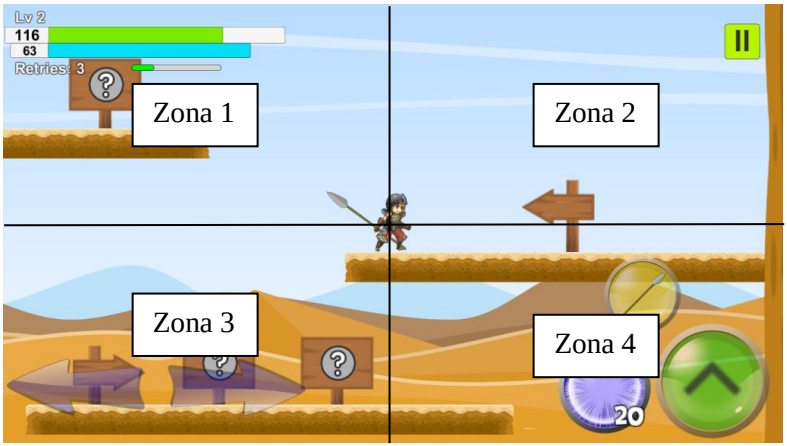
Gambar 4.44 Komponen Bravery pada Musuh

```
public int GetBraveryLevel()
{
        return braveryLevel;
}
void CalculateBraveryValue()
{
        int plusminus = 1;
        if(GetComponent<Enemy>().GetLevel() < Player.level)
                plusminus = -1;
        braveryValue = braveryLevel +
Mathf.CeilToInt((plusminus*(GetComponent<Enemy>().GetLevel() -
Player.level)*(GetComponent<Enemy>().GetLevel() -
Player.level)/2));
}
```

```
public int GetBraveryValue()
{
        return braveryValue;
}
void CalculateAILerpBraveryValue()
{
        float LerpBV = braveryValue*1f / braveryLevel;
        GetComponent<AILerp>().SetBraveryValue(LerpBV);
}
```

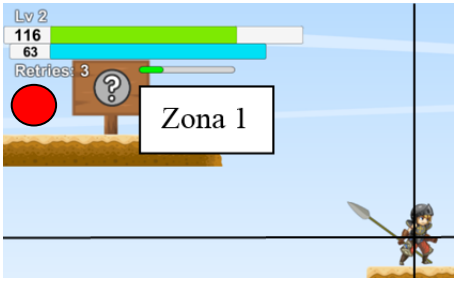
Kode Sumber 4.43 Bravery Level

Apabila musuh memiliki kemampuan untuk memanfaatkan *pathfinding*, maka *Bravery* juga berperan dalam pencarian jalan untuk menjauh dari pemain. Pencarian jalan untuk kabur dibedakan menjadi 4 zona berdasarkan lokasi pemain seperti yang ditampilkan pada Gambar 4.45.



Gambar 4.45 Zona Jalan Kabur Musuh

Apabila musuh berada di zona 1, target untuk *pathfinding* musuh akan berada pada zona 1. Semakin jauh jarak level antara musuh dan pemain, maka target *pathfinding* musuh akan semakin menjauh menuju zona 1. Pada Gambar 4.46, Titik merah adalah contoh target *pathfinding* musuh yang dimaksud pada zona 1.



Gambar 4.46 Target Pathfinding Zona 1

Untuk menentukan target *pathfinding*, zona ditentukan berdasarkan posisi relatif musuh dan pemain. Kemudian, jarak pemain dan target *pathfinding* ditentukan melalui *bravery value.* Setelah itu, akan diinstansiasi sebuah *gameobject* pada lokasi tersebut yang berfungsi sebagai target *pathfinding* musuh. Sehingga, musuh akan menuju *gameobject* tersebut dan seolah-olah berusaha menjauhi pemain. *Gameobject* target ini akan menyesuaikan dengan lokasi pemain, sehingga musuh akan lari jika pemain mendekat. Kode sumber untuk penentuan lokasi *gameobject* target ditunjukkan pada kode sumber 4.44.

```
public void SetHowFarNewPos(float braveryV)
{
        newPathX = braveryV;
        newPathY = braveryV;
}
public void CalculateNewPosRelative()
{
        //finds where should the path positions
        int sideX;
        int sideY;
        if(gameObject.transform.position.x <
player.transform.position.x)
                sideX = -1;
        else
                sideX = 1;
        if(gameObject.transform.position.y+50 <
player.transform.position.y)
```

```
                sideY = -1;
        else
                sideY = 1;
        newPathX *= sideX;
        newPathY *= sideY;
        Vector3 newPathPos = new Vector3(newPathX,newPathY,0);
        newPath.transform.localPosition = newPathPos;
}
```

Kode Sumber 4.44 Penentuan Lokasi Kabur

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan tentang pengujian aplikasi dan hasil dari pengujian. Pengujian dilakukan dengan metode *black-box* menggunakan serangkaian skenario yang disiapkan guna untuk menguji fungsionalitas aplikasi.

5.1 Lingkungan Uji Coba

Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang digunakan dalam masa pengujian dan evaluasi aplikasi. Spesifikasi dari perangkat keras dan perangkat lunak yang digunakan dalam pengujian perangkat lunak ini ditunjukkan pada Tabel 5.1, Tabel 5.2, dan Tabel 5.3.

Tabel 5.1 Lingkungan Uji Coba Perangkat Lunak (1)

Perangkat Keras	Prosesor: Intel(R) Core(TM) i5 CPU M 460 @ 2.53GHz Memori: 2048MB RAM
Perangkat Lunak	Sistem Operasi: Microsoft Windows 8.1 32-bit

Tabel 5.2 Lingkungan Uji Coba Perangkat Lunak (2)

Perangkat Keras	Prosesor: Intel(R) Quad-Core CPU @ 1.5GHz Cortex-A53 & Quad-Core CPU @ 2.0 GHz Cortex-A57 Memori: 2048MB RAM
Perangkat Lunak	Sistem Operasi: Android Versi 6.0 Marshmallow

Tabel 5.3 Lingkungan Uji Coba Perangkat Lunak (3)

Perangkat Keras	Prosesor: Intel(R) Atom(TM) Z2520 CPU @ 1.20GHz Memori: 983MB RAM
Perangkat Lunak	Sistem Operasi: Android Versi 4.4.2

5.2 Skenario Pengujian Fungsionalitas

Skenario pengujian ini bertujuan untuk melakukan uji coba terhadap fungsionalitas perangkat lunak yang dijelaskan pada tahap spesifikasi kebutuhan. Pengujian ini dilakukan untuk mengevaluasi apakah fungsionalitas perangkat lunak sudah sesuai dengan fungsional semestinya.

5.2.1 Skenario Pengujian Kendali Karakter

Skenario pengujian kendali karakter adalah skenario uji coba yang digunakan untuk melakukan evaluasi terhadap tombol kontrol yang digunakan untuk mengendalikan tokoh utama, apakah kontrol sesuai dengan yang diharapkan atau tidak. Skenario ini ditunjukkan pada Tabel 5.4.

Tabel 5.4 Pengujian Kendali Karakter

	UC-001
Kondisi Awal	Pengguna telah memilih karakter dan memasuki antarmuka **InStage.**
Prosedur Pengujian	Pengguna dapat menggunakan tombol kontrol yang ditampilkan di layar untuk menggerakkan karakter. Terdapat tombol untuk mengarahkan ke kiri dan kanan, tombol melompat, menyerang,

	dan tombol *skill.*
Hasil yang diharapkan	Seluruh tombol kontrol yang ditampilkan dapat bekerja sebagaimana mestinya.
Hasil yang diperoleh	Seluruh tombol kontrol yang ditampilkan dapat bekerja sebagaimana mestinya.
Kesimpulan.	Pengujian berhasil

5.2.2 Skenario Pengujian Musuh

Skenario pengujian musuh digunakan untuk menguji apakah musuh sudah dapat bekerja sebagaimana mestinya, yaitu dapat menerima serangan dari pemain, dapat dikalahkan, dan dapat melakukan serangan yang melukai karakter pemain. Skenario ini ditunjukkan pada Tabel 5.5 dan Tabel 5.6.

Tabel 5.5 Pengujian Fungsionalitas Musuh

	UC-002
Kondisi Awal	Pengguna berada pada antarmuka **InStage.**
Prosedur Pengujian	Pengguna menggerakkan karakter pemain untuk terus menyerang musuh hingga musuh dikalahkan.
Hasil yang diharapkan	Musuh menerima sejumlah serangan kemudian kalah dengan meledak.

Hasil yang diperoleh	Musuh menerima sejumlah serangan kemudian kalah dengan meledak.
Kesimpulan.	Pengujian berhasil

Tabel 5.6 Pengujian Fungsionalitas Musuh

	UC-003
Kondisi Awal	Pengguna berada pada antarmuka **InStage.**
Prosedur Pengujian	Pengguna menggerakkan karakter pemain untuk berhenti didepan musuh.
Hasil yang diharapkan	Musuh melakukan serangan yang mengurangi HP dari pemain.
Hasil yang diperoleh	Musuh melakukan serangan yang mengurangi HP dari pemain.
Kesimpulan.	Pengujian berhasil

5.2.3 Skenario Pengujian Stage

Skenario Pengujian Stage dilakukan untuk menguji coba fungsionalitas dari *Stage* beserta rintangannya apakah sudah

sesuai dengan semestinya, yaitu dapat diselesaikan. Skenario ini ditunjukkan pada Tabel 5.7.

Tabel 5.7 Skenario Pengujian Fungsionalitas Stage

	UC-004
Kondisi Awal	Pengguna berada pada antarmuka **World Map.**
Prosedur Pengujian	Pengguna memainkan seluruh *Stage* dari awal hingga akhir.
Hasil yang diharapkan	Pengguna dapat menyelesaikan seluruh *Stage.*
Hasil yang diperoleh	Pengguna dapat menyelesaikan seluruh *Stage.*
Kesimpulan.	Pengujian berhasil

5.2.4 Hasil Pengujian Fungsionalitas

Berdasarkan hasil pengujian dan pengamatan yang dilakukan pada subbab sebelumnya yang ditampilkan pada Tabel 5.8, dapat disimpulkan bahwa pengujian fungsionalitas perangkat lunak berhasil bekerja sebagaimana mestinya.

Tabel 5.8 Hasil Pengujian Fungsionalitas

Kode Uji Coba	**Yang Diuji**	**Hasil**
UC-001	Pengujian Kendali Karakter	Berhasil

UC-002	Pengujian Fungsionalitas Musuh	Berhasil
UC-003	Pengujian Fungsionalitas Musuh	Berhasil
UC-004	Pengujian Fungsionalitas Stage	Berhasil

Berdasarkan hasil uji coba fungsionalitas yang dilakukan dengan metode *black-box*, dapat disimpulkan bahwa keseluruhan uji coba fungsionalitas dapat digunakan dengan baik sesuai dengan kebutuhannya masing-masing.

5.3 Skenario Pengujian Kecerdasan Buatan

Skenario pengujian ini digunakan untuk menguji fungsionalitas fitur kecerdasan buatan pada musuh. Akan dilakukan pengujian terhadap dua fungsionalitas yang berbeda, yaitu pengujian fitur pengejaran pemain dan fitur kabur. Kedua fitur akan diuji coba pada musuh yang dapat melakukan pergerakan selain menyerang. Untuk tiap skenario, uji coba akan dilakukan tiga kali.

5.3.1 Skenario Uji Coba Pengejaran Pemain oleh Musuh

Skenario uji coba ini digunakan untuk mengevaluasi hasil implementasi fitur *pathfinding*. Agen yang dipilih sebagai agen untuk uji coba adalah musuh yang memiliki fitur *pathfinding*. Hal yang diuji coba adalah ketepatan jalan yang dipilih beserta jarak lompatan yang dilakukan agen terhadap berbagai jarak platform yang dapat dilompati. Selain itu, apabila pemain berada diluar jangkauan agen, agen tidak akan berusaha mengejar dan berhenti. Gambar 5.1 menampilkan *screenshot* uji coba dan skenario uji coba ditunjukkan pada Tabel 5.9, 5.10, dan 5.11.

Tabel 5.9 Pengujian Fungsionalitas Pengejaran oleh Musuh

	UC-005
Kondisi Awal	Karakter pemain memiliki level 1 dan memilih *Stage* 1.
Prosedur Pengujian	Pengguna menggerakkan karakter mendekati musuh. Ketika musuh melihat karakter, pengguna mengarahkan karakter untuk menuju lokasi berbeda.
Hasil yang diharapkan	Musuh mengejar pemain dengan cara berjalan dan melakukan lompatan menuju lokasi pemain.
Hasil yang diperoleh	Musuh mengejar pemain dengan cara berjalan dan melakukan lompatan menuju lokasi pemain.
Kesimpulan.	Ketiga pengujian berhasil.

Tabel 5.10 Pengujian Fungsionalitas Pengejaran oleh Musuh

	UC-006
Kondisi Awal	Karakter pemain memiliki level 21 dan memilih *Stage* 2.
Prosedur Pengujian	Pengguna menggerakkan karakter mendekati musuh. Ketika musuh melihat karakter, pengguna mengarahkan karakter untuk menuju lokasi berbeda.

Hasil yang diharapkan	Musuh mengejar pemain dengan cara berjalan dan melakukan lompatan menuju lokasi pemain.
Hasil yang diperoleh	Musuh mengejar pemain dengan cara berjalan dan melakukan lompatan menuju lokasi pemain.
Kesimpulan.	Ketiga pengujian berhasil.

Tabel 5.11 Pengujian Fungsionalitas Pengejaran oleh Musuh

	UC-007
Kondisi Awal	Karakter pemain memiliki level 21 dan memilih Stage 3.
Prosedur Pengujian	Pengguna menggerakkan karakter mendekati musuh. Ketika musuh melihat karakter, pengguna mengarahkan karakter untuk menuju lokasi berbeda.
Hasil yang diharapkan	Musuh mengejar pemain dengan cara berjalan dan melakukan lompatan menuju lokasi pemain.
Hasil yang diperoleh	Musuh mengejar pemain dengan cara berjalan dan melakukan lompatan menuju lokasi pemain.
Kesimpulan.	Ketiga pengujian berhasil.

Gambar 5.1 Pengujian Pengejaran Pemain oleh Musuh

5.3.2 Skenario Uji Coba Pelarian Musuh

Skenario uji coba ini digunakan untuk mengevaluasi hasil implementasi fitur tingkat keberanian. Agen yang dipilih sebagai agen untuk uji coba adalah musuh yang memiliki fitur *pathfinding*. Hal yang diuji coba adalah rute pelarian beserta jarak aman yang dilakukan agen terhadap pemain. *Screenshot* uji coba

ditunjukkan pada Gambar 5.2 dan skenario uji coba ditunjukkan pada Tabel 5.12 dan Tabel 5.13.

Tabel 5.12 Pengujian Fungsionalitas Pelarian Musuh

	UC-008
Kondisi Awal	Karakter pemain memiliki level 21 dan memilih Stage 1.
Prosedur Pengujian	Pengguna menggerakkan karakter untuk mendekati musuh.
Hasil yang diharapkan	Musuh berusaha kabur dari jangkauan pemain dengan menggunakan gerakan yang sesuai.
Hasil yang diperoleh	Musuh berusaha kabur dari jangkauan pemain dengan menggunakan gerakan yang sesuai.
Kesimpulan.	Dua dari tiga pengujian berhasil.

Tabel 5.13 Pengujian Fungsionalitas Pelarian Musuh

	UC-009
Kondisi Awal	Karakter pemain memiliki level 30 dan memilih Stage 2.
Prosedur Pengujian	Pengguna menggerakkan karakter untuk mendekati musuh. Ketika musuh melihat karakter pemain, pengguna mengarahkan karakter pemain untuk terus mendekati musuh.
Hasil yang diharapkan	Musuh berusaha kabur dari jangkauan pemain dengan menggunakan gerakan yang sesuai.
Hasil yang diperoleh	Musuh berusaha kabur dari jangkauan pemain dengan menggunakan gerakan

	yang sesuai.
Kesimpulan.	Ketiga pengujian berhasil.

Gambar 5.2 Pengujian Pelarian Musuh

5.3.3 Hasil Pengujian Kecerdasan Buatan

Uji coba yang dilakukan untuk mengevaluasi kinerja kecerdasan buatan dapat disimpulkan bahwa kecerdasan buatan yang dibuat berupa *pathfinding* dan pelarian berhasil. Hasil pengujian kecerdasan buatan ditampilkan pada Tabel 5.14.

Tabel 5.14 Hasil Pengujian Kecerdasan Buatan

Kode Uji Coba	Uji Coba Ke-		
	1	2	3
UC-005	Berhasil	Berhasil	Berhasil
UC-006	Berhasil	Berhasil	Berhasil
UC-007	Berhasil	Berhasil	Berhasil
UC-008	Berhasil	Gagal	Berhasil
UC-009	Berhasil	Berhasil	Berhasil

Terdapat kegagalan uji coba kedua dengan kode UK-001 dikarenakan musuh tetap berada ditempat ketika didekati dengan pemain yang berlevel lebih tinggi. Pada kasus ini, musuh tidak kabur dikarenakan musuh sudah berada pada *node* yang terdekat dengan target pelarian, sehingga musuh tidak pergi kemana-mana walaupun seharusnya ada tempat yang lebih aman yang bisa dicapai. Hal ini bisa dicegah dengan menambahkan lebih banyak *node* sehingga target pelarian bisa tercapai dengan lebih dekat.

5.4 Pengujian Pengguna

Pengujian pengguna adalah pengujian yang dilakukan secara langsung oleh pengguna tanpa campur tangan pengembang

perangkat lunak. Pengujian ini dilakukan dengan cara memberikan perangkat lunak kepada pengguna untuk dimainkan oleh pengguna yang bersangkutan. Pengguna kemudian dimintai pendapat mengenai sejumlah aspek perangkat lunak yang diuji coba. Aspek yang dinilai sesuai dengan pengujian sebelumnya, yaitu fungsionalitas dan fitur kecerdasan buatan perangkat lunak. Pengujian ini bertujuan untuk memberikan tingkat keberhasilan perangkat lunak yang subjektif dari beberapa orang.

5.4.1 Daftar Penguji Perangkat Lunak

Pada subbab ini akan ditunjukkan mengenai daftar pengguna yang akan melakukan proses pengujian dan penilaian perangkat lunak yang dibangun. Daftar penguji ditampilkan pada Tabel 5.15.

Tabel 5.15 Daftar Penguji Perangkat Lunak

Nomor	Nama	Pekerjaan
1	Sekar Kalaksita	Mahasiswa
2	Mohammad Bimoseno	Mahasiswa
3	M. Bagas Preswantoro	Desainer Grafis
4	Pinta Guna Bakti	Mahasiswa

5.4.2 Hasil Pengujian Pengguna

Subbab ini akan menunjukkan keseluruhan hasil pengujian pengguna berdasarkan skala nilai yang ditunjukkan pada Tabel 5.16.

Tabel 5.16 Skala Nilai Hasil Pengujian Pengguna

Nilai	Keterangan
1	Kurang
2	Cukup
3	Baik
4	Sangat Baik

5.4.2.1 Hasil Pengujian Fungsionalitas

Subbab ini akan menjabarkan hasil uji coba yang dilakukan oleh pengguna mengenai fungsionalitas perangkat lunak. Hasil pengujian fungsionalitas ditunjukkan pada Tabel 5.17.

Tabel 5.17 Hasil Pengujian Fungsionalitas

No.	Aspek	Penilaian				Rata-Rata
		1	2	3	4	
1	Kontrol Permainan	0	0	1	3	3,75
2	Fungsionalitas Pemain	0	0	0	4	4
3	Fungsionalitas Musuh	0	0	0	4	4
4	Kesesuaian *Stage*	0	0	2	2	3,5
Nilai Akhir						**3,8125**

Hasil pengujian fungsionalitas oleh pengguna menghasilkan nilai akhir 3,8125 dari total nilai 4 yang membuktikan bahwa fungsionalitas aplikasi telah bekerja dengan baik

5.4.2.2 Hasil Pengujian Kecerdasan Buatan

Subbab ini akan menjelaskan mengenai hasil uji coba perangkat lunak oleh pengguna dalam aspek kecerdasan buatan perangkat lunak yang meliputi fungsionalitas fitur *pathfinding* dan sistem tingkat keberanian. Hasil pengujian ditunjukkan pada Tabel 5.18.

Tabel 5.18 Hasil Pengujian Kecerdasan Buatan

No.	Aspek	Penilaian				Rata-Rata
		1	2	3	4	
1	Kecepatan Perhitungan *Pathfinding*	0	0	0	4	4
2	Ketepatan Jalur *Pathfinding*	0	1	2	1	3
3	Ketepatan Pelarian Musuh	0	1	3	0	2,5
4	Fungsionalitas Tingkat Keberanian	0	0	3	1	3,25
Nilai Akhir						**3,1875**

Untuk pengujian kecerdasan buatan, uji coba nomor 2 terkait dengan ketepatan jalur *pathfinding* mendapat rata-rata nilai 3 dikarenakan terkadang musuh tidak melakukan lompatan dan terjatuh. Hal ini dapat terjadi karena sebelum musuh mencapai *node* yang mengharuskannya melompat, target *pathfinding* berubah sehingga musuh menargetkan *node* lain tanpa melakukan lompatan terlebih dahulu. Untuk uji coba nomor 3, rata-rata bernilai 2,5 dikarenakan pemilihan lokasi kabur yang kurang efisien karena hanya berdasarkan jarak yang ditentukan garis lurus, sehingga walaupun terdapat rute kabur yang lebih baik, musuh akan memilih rute kabur sesuai dengan garis tersebut.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

No. Kuesioner : 4

Sehubungan dengan tugas akhir, dengan ini kami bermaksud melakukan survey mengenai tingkat kepuasan pelanggan terhadap *game Ashes of Resurgence*. Untuk itu kami mohon kesediaan saudara/i untuk memberikan informasi yang kami tanyakan di bawah ini. Semua identitas dan informasi yang saudara/i berikan akan kami jamin kerahasiaanya. Terima kasih atas partisipasinya.

A. Identitas Responden

1. Nama : M. Bimo Seno
2. Jenis Kelamin : (L) / P
3. Usia : 22
4. Pekerjaan : Mahasiswa
5. Nomor HP : 081 931 020 716

B. Pengujian Fungsionalitas

Berilah tanda centang (v) pada nilai yang paling sesuai dengan aspek yang anda coba. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kemudahan Kontrol Permainan; apakah *layout* sudah nyaman, *responsif*, dan berfungsi dengan benar.			✓	
2	Fungsionalitas Pemain; apakah pemain sudah bekerja sebagaimana mestinya.				✓
3	Fungsionalitas Musuh; apakah musuh sudah bekerja sebagaimana mestinya.				✓
4	Kesesuaian *Stage*; apakah *stage* dapat diselesaikan, tingkat kesulitan sudah sesuai dan *obstacle* dapat dihindari.				✓

C. Pengujian Kecerdasan Buatan

Berilah tanda centang (v) pada pernyataan dibawah ini yang paling sesuai dengan diri Anda. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kecepatan Perhitungan Pathfinding; apakah musuh langsung merespon *pemain tanpa penundaan*.				✓
2	Ketepatan Jalur Pathfinding; apakah musuh dapat mengejar pemain, dan ketepatan pergerakan musuh (melompat tepat sasaran, jatuh, dsb).			✓	
3	Ketepatan Pelarian Musuh; apakah musuh dapat kabur dari pemain, dan apakah rute kabur yang dipilih musuh adalah rute yang baik.			✓	
4	Fungsionalitas Tingkat Keberanian; apakah musuh merespon *level* pemain dengan perilaku yang berbeda.				✓

Responden

Bimo

No. Kuesioner : 2

Sehubungan dengan tugas akhir, dengan ini kami bermaksud melakukan survey mengenai tingkat kepuasan pelanggan terhadap *game Ashes of Resurgence*. Untuk itu kami mohon kesediaan saudara/i untuk memberikan informasi yang kami tanyakan di bawah ini. Semua identitas dan informasi yang saudara/i berikan akan kami jamin kerahasiaanya. Terima kasih atas partisipasinya.

A. Identitas Responden

1. Nama : Muhammad Bagas P
2. Jenis Kelamin : (L) / P
3. Usia : 21
4. Pekerjaan : Designer
5. Nomor HP :

B. Pengujian Fungsionalitas

Berilah tanda centang (v) pada nilai yang paling sesuai dengan aspek yang anda coba. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kemudahan Kontrol Permainan; apakah *layout* sudah nyaman, responsif, dan berfungsi dengan benar.				✓
2	Fungsionalitas Pemain; apakah pemain sudah bekerja sebagaimana mestinya.				✓
3	Fungsionalitas Musuh; apakah musuh sudah bekerja sebagaimana mestinya.				✓
4	Kesesuaian *Stage*; apakah *stage* dapat diselesaikan, tingkat kesulitan sudah sesuai dan *obstacle* dapat dihindari.			✓	

C. Pengujian Kecerdasan Buatan

Berilah tanda centang (v) pada pernyataan dibawah ini yang paling sesuai dengan diri Anda. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kecepatan Perhitungan Pathfinding; apakah musuh langsung merespon pemain tanpa penundaan.				✓
2	Ketepatan Jalur Pathfinding; apakah musuh dapat mengejar pemain, dan ketepatan pergerakan musuh (melompat tepat sasaran, jatuh, dsb).				✓
3	Ketepatan Pelarian Musuh; apakah musuh dapat kabur dari pemain, dan apakah rute kabur yang dipilih musuh adalah rute yang baik.			✓	
4	Fungsionalitas Tingkat Keberanian; apakah musuh merespon *level* pemain dengan perilaku yang berbeda.			✓	

Responden

No. Kuesioner : 3

Sehubungan dengan tugas akhir, dengan ini kami bermaksud melakukan survey mengenai tingkat kepuasan pelanggan terhadap *game Ashes of Resurgence*. Untuk itu kami mohon kesediaan saudara/i untuk memberikan informasi yang kami tanyakan di bawah ini. Semua identitas dan informasi yang saudara/i berikan akan kami jamin kerahasiaanya. Terima kasih atas partisipasinya.

A. Identitas Responden

1. Nama : Pinta gunab
2. Jenis Kelamin : (L) P
3. Usia : 21
4. Pekerjaan : Mahasiswa
5. Nomor HP : 085730379342

B. Pengujian Fungsionalitas

Berilah tanda centang (v) pada nilai yang paling sesuai dengan aspek yang anda coba. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kemudahan Kontrol Permainan; apakah *layout* sudah nyaman, responsif, dan berfungsi dengan benar.				√
2	Fungsionalitas Pemain; apakah pemain sudah bekerja sebagaimana mestinya.				√
3	Fungsionalitas Musuh; apakah musuh sudah bekerja sebagaimana mestinya.				√
4	Kesesuaian *Stage*; apakah *stage* dapat diselesaikan, tingkat kesulitan sudah sesuai dan *obstacle* dapat dihindari.				√

C. Pengujian Kecerdasan Buatan

Berilah tanda centang (v) pada pernyataan dibawah ini yang paling sesuai dengan diri Anda. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kecepatan Perhitungan Pathfinding; apakah musuh langsung merespon *pemain tanpa penundaan*.				√
2	Ketepatan Jalur Pathfinding; apakah musuh dapat mengejar pemain, dan ketepatan pergerakan musuh (melompat tepat sasaran, jatuh, dsb).		√		
3	Ketepatan Pelarian Musuh; apakah musuh dapat kabur dari pemain, dan apakah rute kabur yang *dipilih* musuh adalah rute yang baik.		√		
4	Fungsionalitas Tingkat Keberanian; apakah musuh merespon *level* pemain dengan perilaku yang berbeda.			√	

Responden

Pinta

No. Kuesioner :

Sehubungan dengan tugas akhir, dengan ini kami bermaksud melakukan survey mengenai tingkat kepuasan pelanggan terhadap *game Ashes of Resurgence*. Untuk itu kami mohon kesediaan saudara/i untuk memberikan informasi yang kami tanyakan di bawah ini. Semua identitas dan informasi yang saudara/i berikan akan kami jamin kerahasiaanya. Terima kasih atas partisipasinya.

A. Identitas Responden

1. Nama : Br. Sekar K.
2. Jenis Kelamin : L / P
3. Usia : 21 th
4. Pekerjaan : mahasiswa
5. Nomor HP : 085731833155

B. Pengujian Fungsionalitas

Berilah tanda centang (v) pada nilai yang paling sesuai dengan aspek yang anda coba. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kemudahan Kontrol Permainan; apakah *layout* sudah nyaman, responsif, dan berfungsi dengan benar.				√
2	Fungsionalitas Pemain; apakah pemain sudah bekerja sebagaimana mestinya.				√
3	Fungsionalitas Musuh; apakah musuh sudah bekerja sebagaimana mestinya.				√
4	Kesesuaian *Stage*; apakah *stage* dapat diselesaikan, tingkat kesulitan sudah sesuai dan *obstacle* dapat dihindari.			√	

C. Pengujian Kecerdasan Buatan

Berilah tanda centang (v) pada pernyataan dibawah ini yang paling sesuai dengan diri Anda. **Keterangan:**
1:Kurang , 2:Cukup, 3:Baik, 4:Sangat Baik

No	Pertanyaan	1	2	3	4
1	Kecepatan Perhitungan Pathfinding; apakah musuh langsung merespon pemain tanpa penundaan.				√
2	Ketepatan Jalur Pathfinding; apakah musuh dapat mengejar pemain, dan ketepatan pergerakan musuh (melompat tepat sasaran, jatuh, dsb).			√	
3	Ketepatan Pelarian Musuh; apakah musuh dapat kabur dari pemain, dan apakah rute kabur yang dipilih musuh adalah rute yang baik.			√	
4	Fungsionalitas Tingkat Keberanian; apakah musuh merespon *level* pemain dengan perilaku yang berbeda.			√	

Responden

Br. Sekar Kolaksita

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dibahas mengenai kesimpulan yang dapat diambil dari keseluruhan pengerjaan tugas akhir sebagai jawaban dari rumusan masalah yang telah dikemukakan sebelumnya. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak selanjutnya.

6.1. Kesimpulan

Dalam keseluruhan proses pengerjaan tugas akhir ini, didapatkan kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian fungsionalitas, dapat disimpulkan bahwa fungsionalitas perangkat lunak telah diimplementasikan dengan baik dan benar sehingga tidak ditemukan kekurangan dalam uji coba fungsionalitasnya.
2. Berdasarkan hasil pengujian kecerdasan buatan, diketahui bahwa perangkat lunak yang dibangun telah berhasil dalam mengimplementasikan fitur *pathfinding* yang dimodifikasi pada *game* yang bergenre *platformer*.
3. Perangkat lunak juga telah berhasil dalam mengimplementasikan fitur tingkat keberanian pada musuh sehingga musuh memiliki perilaku yang dinamis dan mengikuti perkembangan pemain.

6.2. Saran

Dibawah ini adalah beberapa saran yang ditujukan untuk pengembangan sistem di masa yang akan datang.

1. Peletakan *node* atau jalur bagi fitur *pathfinding* dapat diperbanyak agar gerakan musuh lebih dinamis dan akurat.
2. Pencarian rute kabur dapat ditambahkan dengan jumlah *node* yang bisa dilewati, sehingga pencarian rute adalah kombinasi antara jarak garis lurus terjauh dan *node* terbanyak.

3. Kecepatan pergerakan keseluruhan karakter dapat dikurangi sehingga pergerakan pemain lebih dapat dikontrol dengan mudah.
4. Mengubah bentuk *loading* dari yang sebelumnya menggunakan *loading bar* menjadi menggunakan animasi normal tanpa *bar* karena *loading bar* seringkali tidak memberikan *loading progress.*

DAFTAR PUSTAKA

[1] J. T. Sergey Karakovskiy, "The Mario AI Benchmark and Competitions," *IEEE Transactions on Computational Intelligence and AI in Games (TCIAG), volume 4 issue 1*, pp. 55-67, 2012.

[2] Unity, "What is Unity?," 15 12 2015. [Online]. Available: http://unity3d.com/unity.

[3] A. Patel, "Introduction to A*," 15 12 2015. [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html.

[4] A. Granberg, "A* Pathfinding Project," 10 June 2016. [Online]. Available: http://arongranberg.com/astar/front.

[5] M. R. Dr. Jurgen Eckerle, "Efficient multiple-agent path planning in grid and non-grid worlds," Berner Fachhochschule, Technik und Informatik, Postfach, CH-2501 Biel/Bienne, Seevorstadt, 2010.

BIODATA PENULIS

Penulis dilahirkan di Tulungagung pada tanggal 25 Desember 1993 dan menetap di Surabaya hingga saat ini. Penulis merupakan anak kedua dari dua bersaudara. Penulis telah menempuh pendidikan formal yaitu TK Perwanida Surabaya (1998-2000), SDN Kebonsari II Surabaya (2000-2006), SMP NEGERI 32 Surabaya (2006-2009), SMA NEGERI 15 Surabaya (2009-2012), dan S1 Jurusan Teknik Informatika dengan rumpun mata kuliah Interaksi, Grafika, dan Seni (2012-2016). Selama menjadi mahasiswa, penulis pernah menjadi koordinator berbagai macam acara kemahasiswaan. Penulis juga aktif dalam organisasi kemahasiswaan ITS. Penulis pernah berkecimpung dalam bidang non-akademik, misalnya menjadi vokalis dalam band dan sering mengisi acara di berbagai kegiatan musik. Penulis dapat dihubungi melalui surel fais9999@ymail.com.