TUGAS AKHIR – KI1502

Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game Menggunakan Google Cardboard

Aliya Rahma Najihati
NRP. 051113 40000 011

Dosen Pembimbing 1
Dr.Eng Darlis Herumurti, S.Kom., M.Kom.

Dosen Pembimbing 2
Imam Kuswardayan, S.Kom., MT.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI1502

Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer *Survival Game* Menggunakan Google Cardboard

Aliya Rahma Najihati
NRP. 051113 40000 011

Dosen Pembimbing 1
Dr.Eng Darlis Herumurti, S.Kom., M.Kom.

Dosen Pembimbing 2
Imam Kuswardayan, S.Kom., MT.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI1502

Implementation of Immersive Technology in Developing Multiplayer Virtual Reality *Survival Game* Using Google Cardboard

Aliya Rahma Najihati
NRP. 051113 40000 011

Supervisor 1
Dr.Eng Darlis Herumurti, S.Kom., M.Kom.

Supervisor 2
Imam Kuswardayan, S.Kom., MT.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game Menggunakan Google Cardboard

TUGAS AKHIR

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

Oleh:
Aliya Rahma Najihati
NRP. 051113 40000 011

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. .....
(NIP. *197712172003121002*) (Pembimbing 1)

2. Imam Kuswardayan, S.Kom., M.T. .....
(NIP. *197612152003121001*) (Pembimbing 2)

SURABAYA
Juni, 2018

(Halaman ini sengaja dikosongkan)

Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game Menggunakan Google Cardboard

Nama : **Aliya Rahma Najihati**
NRP : **051113 40000 011**
Departemen : **Informatika**
Fakultas Teknologi Informasi dan Komunikasi ITS
Dosen Pembimbing I : **Dr.Eng Darlis Herumurti, S.Kom., M.Kom.**
Dosen Pembimbing II : **Imam Kuswardayan, S.Kom., MT.**

ABSTRAK

Survival game (single-player atau multiplayer), adalah sub-genre dari action video game dengan latar belakang lingkungan open-world (terbuka) dan bersifat intens dimana pemain umumnya memulai permainan dengan peralatan yang terbatas dan diharuskan mengumpulkan resources untuk bertahan hidup selama mungkin.

Proses imersi dalam realitas virtual merupakan sebuah persepsi bahwa pengguna seakan-akan secara fisik berada dalam dunia non-fisik melalui gambar, suara atau bentuk rangsangan lain. Tujuan dari pengerjaan Tugas Akhir ini adalah dapat menghasilkan permainan Survival Game yang didesain menggunakan konsep realitas virtual sehingga pemain dapat merasakan pengalaman terjun ke dalam permainan secara imersif, baik secara single-player maupun multi-player.

Hasil uji coba Tugas Akhir menunjukkan bahwa aturan permainan dan fungsionalitas Survival Game berhasil dirancang

dan diimplementasikan baik untuk single-player maupun multiplayer.

Kata kunci: Survival Game, Unity, Multiplayer, Realitas Virtual.

Implementation of Immersive Technology in Developing Multiplayer Virtual Reality Survival Game Using Google Cardboard

Name	**: Aliya Rahma Najihati**
NRP	**: 051113 40000 011**
Department	**: Department of Informatics Faculty of Information and Communication Technology ITS**
Supervisor I	**: Dr.Eng Darlis Herumurti, S.Kom., M.Kom.**
Supervisor II	**: Imam Kuswardayan, S.Kom., MT.**

ABSTRACT

Survival game is a sub-genre of action video game with intense open-world environment where player generally starts the game with limited resources. As the game progresses, Player is usually also required to collect resources to survive as long as possible.

Immersion into virtual reality consists as a perception where player is physically present in non-physical world by surrounding the user of virtual reality system with various pictures, sounds, or other kinds of stimuli. The purpose of this Final Project is being able to develop a survival game which is designed using the concept of virtual reality so that player is able to experience the game immersively, be it playing it alone (single-player) or playing with another person (multiplayer).

Testing result of this Final Project shows that both gameplay and functionalities of Survival Game have been succesfully

designed and implemented for both single-player and multiplayer mode.

Key words: Survival Game, RPG, Unity, Multiplayer, Virtual Reality.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul "Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer *Survival Game* Menggunakan Google Cardboard".

Pengerjaan tugas akhir ini penulis lakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Program Studi S-1 Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Allah SWT atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Keluarga penulis, Bapak Tubagus Atmiko, Ibu Hidayati, saudari kembar Aliya Fathma Najihati, dan juga keluarga yang tidak dapat penulis sebutkan satu per satu yang telah memberi dukungan moral dan material serta doa untuk penulis.
3. Bapak Dr. Eng Darlis Herumurti, S.Kom., M.Kom. selaku dosen pembimbing I yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
4. Bapak Imam Kuswardayan S.Kom., MT. selaku dosen pembimbing II yang telah memberikan bimbingan dan arahan dalam pengerjaan tugas akhir ini.
5. Teman-teman angkatan 2013 yang memiliki dosen pembimbing yang sama yang telah memberikan dukungan selama saya menyelesaikan Tugas Akhir ini.

6. Seluruh pihak yang tidak bisa saya sebutkan satu persatu yang telah memberikan dukungan selama saya menyelesaikan Tugas Akhir ini.

Penulis menyadarai bahwa masih terdapat banyak kekurangan dalam tugas akhir ini. Oleh karena itu, penulis menerima dengan rendah hati kritik dan saran untuk pembelajaran dan perbaikan ke depannya. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Juni 2018

Penulis

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Bagian ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan dan manfaat, metodologi dan sistematika penulisan yang digunakan dalam pembuatan tugas akhir ini.

1.1 Latar Belakang

Video Game berbasis realitas virtual akhir-akhir ini berkembang pesat sejak adanya teknologi terbaru yang memaksimalkan sensasi bermain di dunia dalam game. Teknologi ini sudah diimplementasikan ke dalam game sejak pertengahan tahun 1990 yang diperkenalkan oleh perusahaan Nintendo. Pada saat itu teknologi yang digunakan masih berupa tampilan monochromatic dan sudut pandang yang sangat terbatas. Seiring perkembangan teknologi, game berbasis realitas virtual mulai dikembangkan lagi dengan teknologi yang lebih canggih dan fitur yang lebih maksimal. Tidak hanya untuk home video game console, tetapi juga handheld console untuk merasakan sensasi realitas virtual dalam game dalam bentuk portabel.

Walaupun teknologi tentang realitas virtual semakin canggih, tidak banyak game yang dibuat dalam bentuk berbasis realitas virtual, khususnya di device portabel seperti smartphone, dari website informasi vrgamesfor.com, dari 1134 game berbasis virtual reality yang dirilis, hanya sekitar 194 yang di smartphone, itu juga tidak semua device realitas virtual bisa digunakan [1]. Game yang dibuat kebanyakan mensimulasikan dalam keadaan tertentu, beberapa contohnya seperti simulasi naik Roller Coaster, terjebak di dalam suatu ruangan, berada di suasana perang, dan di ruang angkasa. Permainan yang akan dibuat merupakan permainan simulasi terdampar di suatu tempat terpencil.

Permainan non-VR yang dijadikan referensi dalam membuat permainan in adalah The Sims 2 : Cast Away, dimana pemain memainkan tokoh karakter yang terdampar di suatu pulau antah berantah. Pemain kemudian diharuskan menyelesaikan misi-misi untuk bertahan hidup, yangmana pemain bisa membunuh hewan untuk memanfaatkan dagingnya, mengumpulkan buah, item dan lain-lain. Permainan ini akan diimplementasikan dalam bentuk realitas virtual dengan menggunakan *head-up display* Google Cardboard untuk mendapatkan sensasi sudut pandang orang pertama dari tokoh yang terdampar.

Dari pembuatan game ini, diharapkan pengguna dapat mengerti dan merasakan sensasi bagaimana cara bertahan hidup di antah berantah dengan lebih nyata. *Asset* yang digunakan di dalam permainan ini pun dipilih berdasarkan tingkat kenyataannya.

1.2 Rumusan Masalah

Perumusan masalah yang terdapat pada tugas akhir ini, antara lain:

1. Bagaimana merancang aturan main dan skenario dari permainan simulasi orang terdampar?
2. Bagaimana merancang perilaku karakter musuh dengan metode *Finite State Machine* (FSM)?
3. Bagaimana implementasi dari rancangan permainan diselesaikan dengan *Game Engine Unity*?
4. Bagaimana implementasi dari rancangan permainan diselesaikan secara *single-player* dan *multiplayer*?

1.3 Batasan Masalah

Batasan masalah yang terdapat pada tugas akhir ini, yaitu sebagai berikut:

1. Dikembangkan untuk *smartphone* bersistem operasi Android.
2. Penggunaan Cardboard sebagai media permainan.
3. Aplikasi dibangun dengan game engine Unity3D dengan bantuan Software Development Kit (SDK) GoogleVR

1.4 Tujuan

Tujuan dalam pembuatan tugas akhir ini adalah terciptanya game simulasi dengan cara mensimulasikan berbagai aktifitas bertahan hidup dari sudut pandang orang yang terdampar.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini, antara lain:

1. Mengimplementasikan prinsip Virtual Reality Multiplayer dalam beberapa aspek pada permainan *Survival Game*,
2. Memberikan media hiburan bagi pemain.

1.6 Metodologi

1. Penyusunan proposal tugas akhir
   Tahap pertama dalam proses pengerjaan tugas akhir ini adalah menyusun proposal tugas akhir. Pada proposal tugas akhir ini diajukan implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer *Survival Game* Menggunakan Google Cardboard.

2. Studi literatur

Pada tahap ini, akan dicari studi literatur yang relevan untuk dijadikan referensi dalam pengerjaan tugas akhir. Studi literatur ini didapatkan dari buku, internet, dan materi-materi kuliah yang berhubungan dengan metode yang akan digunakan.

3. Analisis dan desain perangkat lunak
   Perangkat lunak yang akan dibangun merupakan aplikasi untuk perangkat desktop dan Android. Akan ada tiga macam tampilan dari aplikasi ini. Tampilan untuk menu, tampilan memilih mode permainan, dan tampilan permainan.

4. Pengembangan perangkat lunak
   Pembangunan *game* akan dilakukan dengan menggunakan bahasa pemrograman C# dan Game Engine Unity 5.5.6f1.

5. Pengujian dan evaluasi
   Pengujian yang akan dilakukan adalah pengujian fungsionalitas. Pengujian ini bertujuan untuk memeriksa apakah *game* sudah terimplementasikan atau belum. Kemudian akan dilakukan evaluasi untuk memeriksa hasil implementasi *game*.

6. Penyusunan buku tugas akhir
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini. Pada tahap ini juga disertakan hasil dari implementasi metode dan algoritma yang telah dibuat. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:
   1. Pendahuluan

a. Latar Belakang
b. Rumusan Masalah
c. Batasan Tugas Akhir
d. Tujuan
e. manfaat
f. Metodologi
g. Sistematika Penulisan

2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7 Sistematika Penulisan

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

BAB III. ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis

permasalahan membahas permasalahan yang yang diangkat dalam pengerjaan tugas akhir.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Bab ini berisi proses implementasi dari setiap kelas pada semua modul.

BAB V. PENGUJIAN DAN EVALUASI

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

BAB VI. KESIMPULAN DAN SARAN

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan metode yang diajukan pada pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Virtual Reality (VR)

Virtual Reality atau Realitas Virtual adalah teknologi komputer yang menggunakan perangkat lunak untuk menghasilkan gambar yang realistis, suara dan sensasi lain yang mereplika lingkungan nyata (atau membuat lingkungan dari imajinasi). Teknologi ini mensimulasikan kehadiran fisik pengguna dalam lingkungan dengan memungkinkan pengguna untuk berinteraksi dalam ruangan ini dan benda di dalamnya dengan cara menggunakan tampilan layar atau proyektor khusus dan perangkat lainnya. Teknologi ini ditunjukkan pada Gambar 2.1.

VR dibagi dan secara obyektif hadir seperti dunia fisik, ditampilkan seperti sebuah karya seni, dan memiliki imajinasi yang tidak terbatas dan tidak berbahaya. Ketika VR tersebar di penjuru dunia, VR tidak akan dilihat sebagai media penengah yang digunakan dalam dunia nyata, melainkan sebagai dunia tambahan [2].

Gambar 2.1 Cara Kerja Virtual Reality

Dengan menggunakan Teknologi Realitas Virtual, pengguna dapat memanfaatkan inderanya untuk berinteraksi dengan lingkungan. Alat yang diperlukan untuk menampilkan lingkungan realitas virtual disebut Head-mounted display, sedangkan alat yang dibutuhkan untuk menyesuaikan posisi dan orientasi pengguna yaitu tracker.

2.2 Google Cardboard

Google Cardboard adalah alat virtual reality 3D yang dibuat dari karton yang diperkenalkan tahun 2015. Didesain oleh Google dan dibuat oleh pihak ketiga, alat ini digunakan untuk menampilkan tampilan stereoscopic. Alat in dikendalikan oleh gerakan kepala dan fitur accelerometer yang ada di smartphone [3]. dijual dengan harga yang murah untuk merasakan sensasi realitas virtual. Bentuk Cardboard dapat dilihat lebih lanjut pada Gambar 2.2

Gambar 2.2 Model dan Penggunaan Google Cardboard

2.3 Unity3D

Unity3D adalah mesin pengembangan game yang user-friendly dan powerful yang dapat dibuat di berbagai platform. Unity3D dapat digunakan dengan mudah untuk para pemula dan dapat digunakan secara maksimal untuk para ahli. Aplikasi ini dapat menarik perhatian siapa saja bagi yang ingin membuat game 3D untuk mobile, desktop, web, dan console [4]. Unity dibuat oleh Davil Helgason Nicholas pada taun 2004. Aplikasi ini dibangun dengan fungsi standar umum yang digunakan dalam pembuatan game. Game engine ini juga menyediakan asset store untuk menambahkan asset dan fitur yang diinginkan pengguna. Contoh tampilan project Unity3D dapat dilihat pada Gambar 2.3.

Gambar 2.3. Tampilan Antarmuka Aplikasi Unity3D

Unity 3D mendukung game berbentuk 3D dan 2D. Unity juga mendukung aplikasi pihak ketiga seperti Blender sebagai peragkat lunak desain dan animasi. Blender dapat digunakan untuk desain dan animasi objek yang dibutuhkan dalam game dikarenakan Unity tidak menyediakan fitur desain objek. Bahasa pemograman yang digunakan untuk Unity adalah C#, Javascript, dan Boo. Pengguna dapat menggunakan bahasa pemograman tersebut secara bersamaan dalam satu proyek atau memilih salah satu bahasa pemograman tersebut.

2.4 Permainan Serupa (The Sims 2 : Castaway)

The Sims 2 : Castaway adalah konsol spin-off ketiga dari video game The Sims 2 yang dikembangkan oleh The Sims Division. Permainan ini dirilis di Amerika Serikat oleh Electronic Arts pada 22 Oktober 2007 dan 26 Oktober 2007 di Inggris Raya untuk Nintendo DS, Wii, PlayStation 2 dan

PlayStation Portable. The Sims 2: Castaway juga tersedia untuk platform mobile. Nokia menyediakan The Sims 2: Castaway pada Ovi Store. Game ini digunakan sebagai referensi untuk game simulasi orang terdampar yang akan dibuat pada tugas akhir ini. Gameplay dari game ini akan dimodifikasi sesuai dengan kebutuhan game simulasi untuk tugas akhir. Aturan bermain dari game ini adalah pemain mengontrol karakter yang terdampar.

Tujuan game ini adalah menemukan cara kembali ke peradaban. Pemain ditugaskan melakukan berbagai aktifitas mulai dari mencari makanan, minuman dan lain-lain. Jika pemain tidak berhati-hati, karakter yang dikendalikan akan menjadi stress dan akhirnya akan mengalami breakdown dimana karakter tersebut akan malas melakukan apapun termasuk makan dan minum, sehingga mengakibatkan kematian. Tampilan *screenshot* permainan *The Sims 2 : Castaway* dapat dilihat pada Gambar 2.4.

Gambar 2.4 Tampilan *screenshot* The Sims 2: Castaway

(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini membahas mengenai perancangan dan pembuatan perangkat lunak. Perangkat lunak yang dibuat pada tugas akhir ini adalah survival game dengan konsep realitas virtual yang imersif dalam kedua mode, baik single-player maupun multiplayer.

3.1 Perancangan Permainan

Pada subbab ini akan dibahas skenario permainan dan aturan main pada *game* ini.

3.1.1 Skenario Permainan

Game ini adalah *game* 3D dengan tema petualangan, action dan survival (ekstensi dari action). Game ini dimainkan oleh satu orang (single-player) maupun lebih dari satu (multiplayer). Target dari game ini adalah menjelajahi open world dan pada akhirnya mencapai transporter terakhir (transporter final) yang terletak di Area 3 sebelum seluruh health point mencapai 0 yang berarti game over. Pemain dapat melewati musuh tanpa bertarung dengan musuh tersebut untuk menghindari berkurangnya health point yang tidak perlu. Musuh dapat berjalan-jalan sendiri dan akan mengejar pemain bila pemain berada dalam radar. Pemain dapat bertarung dengan serangan melee seperti pedang. Pemain juga dapat mengambil dan menggunakan item yang sudah diambil.

Pengaturan musuh juga diatur pada tahap ini. Ada 2 jenis musuh, yaitu musuh lemah dan musuh kuat. Rancangan musuh dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Rancangan level musuh

Area	Musuh Lemah	Musuh Kuat
1	0	0
2	15	0
3	10	15

3.1.2 Aturan Permainan

Permainan ini memiliki beberapa aturan main, yaitu:

1. Pada area 2 dan 3 akan ada musuh yang muncul.
2. Pemain dapat menyerang musuh dengan menggunakan serangan melee.
3. Pemain dapat memilih untuk mengabaikan musuh.
4. Musuh akan melakukan serangan secara otomatis sesuai dengan atribut *attack* dari masing-masing karakter.
5. Atribut attack akan mempengaruhi berapakah damage yang akan dihasilkan, minimal nilai damage adalah 20.
6. Apabila musuh kehilangan seluruh *health point*-nya maka musuh akan hancur.
7. Apabila *Experience Point* (EXP) mencapai nilai maksimum level tersebut, pemain akan naik level.
8. Pemain dapat mengambil item *potion* dan *weapon* yang tergeletak di beberapa tempat.
9. Pemain dapat menggunakan item *potion* dan *weapon* yang sudah tersimpan dalam inventori
10. Musuh akan ter-*spawn* pada lokasi random setiap kali musuh jenis tersebut dikalahkan.
11. Pemain akan GameOver bila HealthPoint mencapai 0
12. Pemain akan Menang bilamana mencapai transporter terakhir (Transporter Final) yang terletak di Area 3.

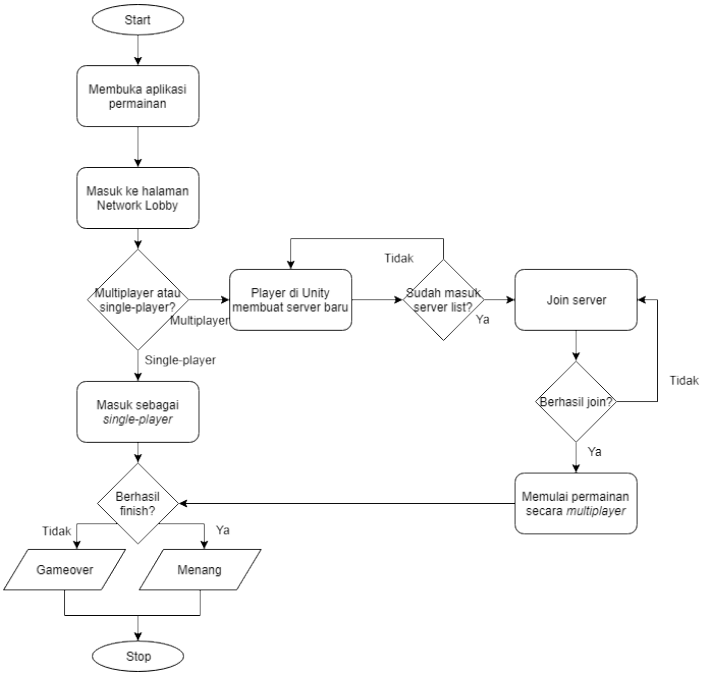
3.2 Perancangan Perangkat Lunak

Tahap perancangan pada bab ini dibagi menjadi beberapa bagian yaitu perancangan proses, perancangan alur permainan, perancangan diagram kasus penggunaan, spesifikasi skenario kasus penggunaan, perancangan antarmuka aplikasi, dan perancangan data.

3.2.1 Perancangan Proses

Pada subbab ini akan dijelaskan mengenai rancangan proses yang akan dilakukan untuk mendukung kebutuhan fungsionalitas perangkat lunak yang sudah dirancang. Proses ini penting agar perangkat lunak dapat berjalan secara baik dan lancar.

Secara umum alur proses pada *game* ini hanya satu, yaitu pengguna akan membuka permainan dan memasuki menu awal, lalu akan terjadi percabangan apakah pengguna akan memulai permainan atau keluar permainan. Jika pengguna memilih untuk memulai permainan, maka permainan akan dimulai. Jika pengguna memilih keluar saat berada di menu utama, maka permainan akan ditutup/keluar. Selengkapnya, *flowchart* alur proses ini dapat dilihat pada Gambar 3.1.



Gambar 3.1. Flowchart alur proses permainan

3.2.2 Perancangan Skenario Aturan Main

Aaplikasi ini menerapkan aturan main dengan cara bertahan hidup setelah terdampar di suatu pulau dan mengalahkan musuh. Selain itu, pemain juga diharuskan untuk mengeksplor area agar dapat menemukan *transporter* untuk bisa lanjut ke level selanjutnya.

Rincian lebih lanjut, permainan akan dibuat menjadi tiga (3) level, dengan level pertama memiliki tingkat kesulitan paling mudah. Pada level ini, pemain berada di suatu pulau kecil. Pemain harus mengambil senjata pedang dan mengalahkan boneka *dummy* untuk mengaktifkan suatu *transporter* yang akan

mengantarkan pemain ke level selanjutnya. Pada level ini, tidak ada musuh yang berkeliaran. Level selanjutnya berlatar belakang pulau lain lagi, dengan area permainan yang lebih luas dan tingkat kesulitan yang lebih dari level sebelumnya. Disini, sudah terdapat beberapa musuh yang berkeliaran di sekitar pemain dan siap menyerang pemain saat jarak sudah dekat.

Langkah pertama yang harus dilakukan untuk memulai permainan ini adalah memilih mode permainan : apakah ingin memainkan permainan sebagai *single-player* maupun *multiplayer*. Setelah memilih mode permainan, sistem akan langsung masuk ke permainan. Dalam permainan, kontrol yang digunakan adalah dengan mengeklik tombol pada Kontroller untuk bergerak maju atau berjalan, lalu mengeklik lagi untuk berhenti. Pengguna harus bertahan hidup dan melawan musuh tertentu untuk menyelesaikan permainan ini dengan menelusuri area, menyembuhkan diri, melawan atau menghindari musuh lainnya. Cara melawan musuh adalah dengan memakai senjata dan mengarahkan kamera ke arah musuh lalu mengeklik tombol pada Kontroller untuk mengayunkan senjata, Saat menelusuri area pengguna diminta untuk tetap bertahan hidup sampai menemukan jalan keluar. Dalam perjalanannya menelusuri area, pengguna dapat menemukan barang-barang bantuan agar pengguna dapat bertahan hidup lebih lama. Barang-barang yang dimaksud adalah senjata lain dan kotak obat. Senjata lain berguna untuk menambah jumlah *damage* yang pada musuh dan kotak obat berguna untuk menambah nilai dari nyawa pengguna.

Kondisi kalah atau menang dalam permainan ini yaitu, pengguna akan menang jika mampu bertahan hidup dan mencapai berhasil mengalahkan musuh tertentu. Kemudian pengguna akan kalah dan layer akan langsung berganti ke layer Gameover bila

healthbar pemain habis. Skenerio permainan dapat dilihat pada Tabel 3.2

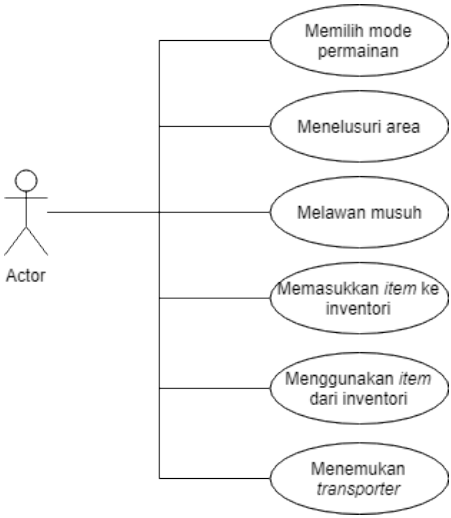
Tabel 3.2. Skenario Permainan

Spesifikasi	**Keterangan**	
Nama Skenario	Aturan Permainan	
Deskripsi	Skenerio ini berfungsi agar aktor dapat menyelesaikan aplikasi permainan	
Kondisi Awal	Objek sudah tampil pada aplikasi	
Kondisi Akhir	Pindah ke *scene level* selanjutnya	
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Mengarahkan perangkat pada objek tertentu	Menampilkan objek sesuai arah sudut *smartphone*
	Mengeklik tombol pada Kontroller (kondisi berhenti)	Menggerakkan kamera ke depan
	Mengeklik tombol pada Kontroller (kondisi berjalan)	Menghentikan gerak kamera ke depan
	Menggerakkan kamera ke arah bawah pada sudut tertentu	Menampilkan UI inventori
	Mengarahkan kamera ke *item* lalu mengeklik tombol pada Kontroller	Menyimpan *item* ke dalam inventory
	Mengeklik tombol pada Kontroller (kondisi inventory UI sedang muncul)	Menggunakan *item* pada inventory
	Mengarahkan kamera ke musuh dan	

	mengeklik tombol pada Kontroller (kondisi senjata telah terpasang)	Menyerang musuh
	Menabrak objek transporter	Mengirim pemain ke level selanjutnya
		Menampilkan informasi permainan selesai

3.2.3 Perancangan Diagram Kasus Penggunaan

Dalam aplikasi tugas akhir ini, terdapat enam kasus penggunaan yang ada yaitu memilih mode permainan (*multiplayer* atau *single-player)*, menelusuri area, melawan musuh, mengambil *item*, dan menemukan transporter. Rancangan kasus penggunaan dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram kasus penggunaan

3.2.4 Spesifikasi Skenario Kasus Penggunaan

Setelah diketahui bagaimana perancangan kasus penggunaan, maka spesifikasi kasus penggunaan akan dijelaskan pada subbab berikut.

3.2.4.1 Memilih Mode Permainan

Spesifikasi kasus penggunaan Memilih Mode Permainan dapat dijabarkan pada Tabel 3.3.

Tabel 3.3 Spesifikasi Kasus Penggunaan UC-001

Spesifikasi	**Keterangan**	
Kode Kasus Penggunaan	UC-001	
Nama Kasus Penggunaan	Memilih Mode Permainan	
Deskripsi	Kasus pengguna ini berfungsi supaya aktor dapat memilih mode permainan yang diinginkan	
Kondisi Awal	Aplikasi sudah berjalan dan menampilkan pilihan mode permainan	
Kondisi Akhir	Permainan dimulai dengan mode permainan yang dipilih terbuka	
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Menekan layar Android pada kotak pilihan mode permainan	*Load* dan *render* objek

3.2.4.2 Menelusuri Area

Spesifikasi kasus penggunaan Menelusuri Area dapat dijabarkan pada Tabel 3.4.

Tabel 3.4 Spesifikasi Kasus Penggunaan UC-002

Spesifikasi	**Keterangan**	
Kode Kasus Penggunaan	UC-002	
Nama Kasus Penggunaan	Menelusuri Area	
Deskripsi	Kasus pengguna ini berfungsi supaya aktor dapat bergerak dan berjalan dengan bebas mengelilingi lingkungan virtual dalam permainan	
Kondisi Awal	Objek sudah tampil pada aplikasi	
Kondisi Akhir	-	
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Mengarahkan target/sasaran pada objek tertentu	Menampilkan objek sesuai arah sudut *smartphone*
	Mengeklik tombol pada Kontroller (kondisi berhenti)	Menggerakkan gerak kamera ke depan
	Mengeklik tombol pada Kontroller (kondisi berjalan)	Menghentikan gerak kamera ke depan
Alur Alternatif	Aktor	Sistem
	Berhenti di titik tertentu sampai nyawa habis	*Load* scene “GAME OVER”

3.2.4.3 Melawan Musuh

Spesifikasi kasus penggunaan Melawan Musuh dapat dijabarkan pada Tabel 3.5.

Tabel 3.5 Spesifikasi Kasus Penggunaan UC-003

Spesifikasi	**Keterangan**	
Kode Kasus Penggunaan	UC-003	
Nama Kasus Penggunaan	Melawan musuh	
Deskripsi	Kasus pengguna ini berfungsi supaya aktor dapat melawan musuh yang ada	
Kondisi Awal	Layar permainan sudah muncul	
Kondisi Akhir	*Healthbar* musuh berkurang	
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Menyerang musuh saat musuh sudah berada dalam jarak serang senjata	*Healthbar* musuh berkurang
Alur Alternatif	Menyerang musuh saat musuh berada di luar jarak serang senjata	*Healthbar* musuh tidak berkurang

3.2.4.4 Mengambil *Item*

Spesifikasi kasus penggunaan Mengambil *Item* dapat dijabarkan pada Tabel 3.6.

Tabel 3.6 Spesifikasi Kasus Penggunaan UC-004

Spesifikasi	**Keterangan**
Kode Kasus	UC-004

Penggunaan		
Nama Kasus Penggunaan	Mengambil *Item*	
Deskripsi	Kasus pengguna ini berfungsi agar aktor dapat mengambil *item* yang tergeletak di area	
Kondisi Awal	*Item* belum tersimpan ke inventory	
Kondisi Akhir	*Item* tersimpan ke inventory	
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Mengarahkan *reticle* ke *item* untuk memasukkan *item* ke inventory (kondisi pemain dalam jarak ambil dengan *item*)	- *Item* melayang ke arah pemain - *Item* masuk ke dalam inventori
Alur Alternatif	Aktor	Sistem
	Mengarahkan *reticle* ke *item* untuk memasukkan *item* ke inventory (kondisi di luar jarak ambil dengan *item*)	- Pemain berhenti - *Item* tidak masuk ke dalam inventori

3.2.4.5 Menggunakan *Item*

Spesifikasi kasus penggunaan Menggunakan *Item* dijabarkan pada Tabel 3.7.

Tabel 3.7 Spesifikasi Kasus Penggunaan UC-005

Spesifikasi	**Keterangan**
Kode Kasus Penggunaan	UC-005
Nama Kasus	Menggunakan *Item*

Penggunaan		
Deskripsi	Kasus pengguna ini berfungsi supaya aktor dapat menggunakan *item* pada inventory	
Kondisi Awal	*Item* sudah tersimpan ke inventory	
Kondisi Akhir	*Item* pada inventory berkurang	
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Mengarahkan *reticle* ke arah bawah pada sudut tertentu untuk memunculkan InventoryUI dan mengarahkan *reticle* ke slot *item* yang diinginkan untuk menggunakan *item*	- Menampilkan inventory UI - Mengurangi jumlah *item* tersebut pada inventory - Memberi efek pada pengguna sesuai dengan atribut pada *item* tersebut

3.2.4.6 Menemukan Transporter

Spesifikasi kasus penggunaan Menemukan Transporter dijabarkan pada Tabel 3.8.

Tabel 3.8 Spesifikasi Kasus Penggunaan UC-006

Spesifikasi	Keterangan
Kode Kasus Penggunaan	UC-006
Nama Kasus Penggunaan	Menemukan transporter
Deskripsi	Kasus pengguna ini berfungsi supaya aktor dapat melanjutkan ke level berikutnya atau menyelesaikan permainan
Kondisi Awal	*Transporter* tidak terlihat dan belum aktif
Kondisi Akhir	*Transporter* muncul dan aktif

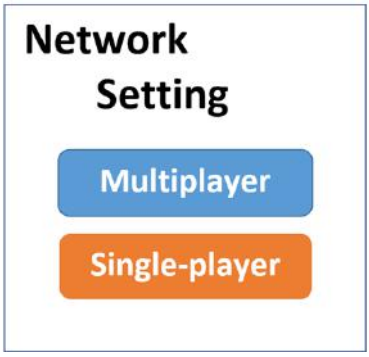
Aktor	Pengguna	
Alur Utama	Aktor	Sistem
	Mengalahkan musuh tertentu	Menampilkan transporter
	Menyentuh transporter	Mengirim pemain ke level selanjutnya

3.2.5 Perancangan Antarmuka Pengguna

Subbab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk Tugas Akhir. Dalam *game* ini terdapat beberapa tampilan, yaitu tampilan awal (menu utama), tampilan *network settings*, tampilan permainan, tampilan saat menang, dan tampilan saat *gameover*.

3.2.5.1 Tampilan Mode Permainan

Halaman ini berisikan tentang tampilan jaringan. Rancangan tampilan jaringan dapat dilihat pada Gambar 3.3. Pemain dapat memilih ingin menjalankan permainan secara *multiplayer* ataupun *single-player*.

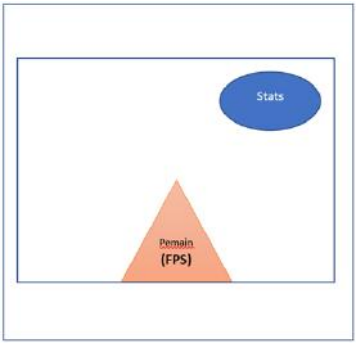


Gambar 3.3 Rancangan tampilan mode permainan

3.2.5.2 Tampilan Permainan

Tampilan permainan ialah halaman saat pemain telah menekan tombol *start*. Rancangan Tampilan Permainan dapat dilihat pada Gambar 3.4. Pada layar terdapat beberapa hal yaitu:

1. Stats pemain (dalam bentuk text) yang menunjukkan jumlah *health point*, *health bar,* dan level pemain saat itu.



Gambar 3.4 Rancangan tampilan permainan

3.2.5.3 Tampilan Pesan Menang dan Kalah

Tampilan pesan menang ialah pesan yang muncul ketika pemain telah berhasil mengalahkan satu musuh besar di *dungeon* kedua. Tampilan pesan kalah ialah pesan yang muncul ketika pemain *game over* atau kehilangan seluruh *health point*. Tampilan pesan menang dan kalah tidak ada tombol karena secara otomatis akan memulai level selanjutnya setelah beberapa saat. Rancangan tampilan pesan menang dan kalah dapat dilihat pada Gambar 3.5 dan Gambar 3.6.

Gambar 3.5 Rancangan tampilan pesan menang

Gambar 3.6 Rancangan tampilan pesan kalah

3.2.6 Proses Perancangan Data

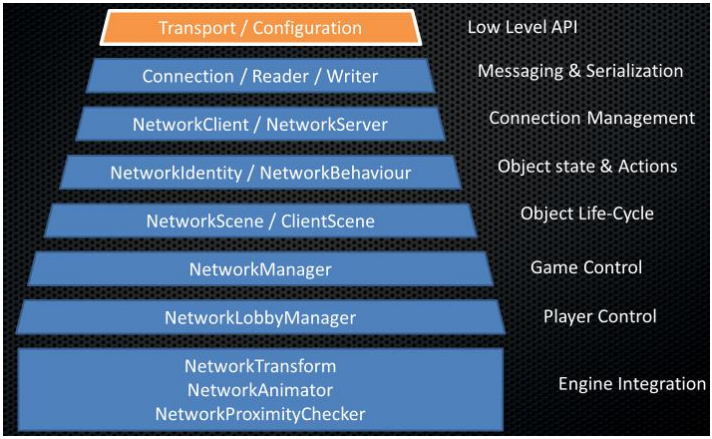
3.2.6.1 Proses Perancangan Jaringan Multiplayer

Multiplayer High-Level API (HLAPI) Unity adalah suatu sistem untuk membangun kemampuan multiplayer pada permainan-permainan Unity. Sistem ini dibangun diatas lapisan *real-time* komunikasi level transport yang lebih lebih rendah, dan mengatasi banyak tugas-tugas yang dibutuhkan pada permainan multiplayer. Selama lapisan transport mendukung segala jenis dari topologi jaringan, HLAPI berperan sebagai suatu *server authoritative system;* Meskipun begitu, sistem ini memperbolehkan satu dari partisipan untuk menjadi client dan server dalam waktu yang sama, sehingga tidak ada proses *dedicated server* yang dibutuhkan. Bekerja pada konjungsi dengan servis internet memungkinkan permainan multiplayer untuk dimainkan lewat internet dengan sedikit tambahan dari

pengembang. HLAPI adalah suatu set perintah *networking* baru yang dibangun di dalam Unity menggunakan *namespace* baru : UnityEngine.Networking. Sistem ini berfokus pada kemudahan dalam penggunaan dan pengembangan secara iterative yang menyediakan layanan yang berguna untuk permainan *multiplayer* seperti :

- Penanganan-penganan pesan *(message handlers)*
- *Serialization* performa tinggi
- Obyek manajemen terdistribusi
- Sinkronisasi *state*
- Kelas-kelas jaringan : Server, Client, Connection, dan sebagainya.

HLAPI dibangun dari sekumpulan lapisan yang menambah fungsionalitas, ditunjukkan pada Gambar 3.7.



Gambar 3.7 Lapisan-lapisan jaringan

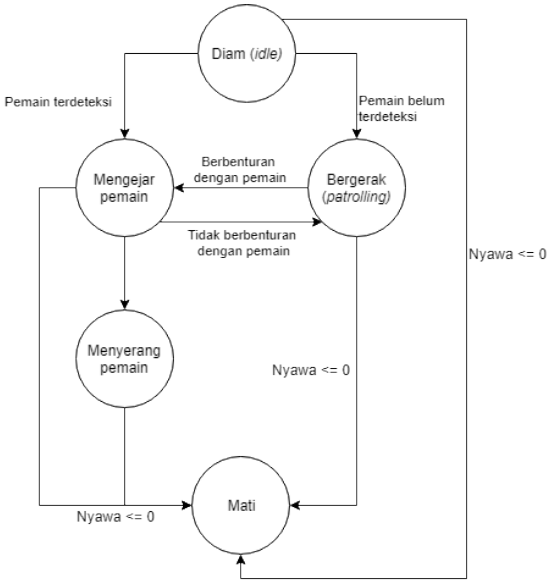
3.2.6.2 Proses Perancangan FSM

Perancangan perilaku perlu diterapkan pada *Non Playable Character* (NPC), dalam hal ini musuh, sehingga nantinya dapat memberi kesan yang lebih nyata, menghibur, serta dapat memberikan tantangan dalam aplikasi permainan ini. *Finite State Machine (FSM)* adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: State (Keadaan), Event (kejadian) dan action (aksi). Perilaku untuk musuh ini dirancang dengan metode *finite state machine* yaitu bekerjanya beberapa *state* (keadaan) tertentu berdasarkan parameter-parameter tertentu pula. Proses Perancangan FSM akan dijelaskan dalam FSM perilaku musuh.

3.2.6.2.1 FSM Perilaku NPC (Musuh)

Perilaku untuk musuh dibagi menjadi lima, yaitu kondisi diam, kondisi bergerak (*patrolling*), kondisi mengejar pemain, kondisi menyerang pemain, serta kondisi mati. Kondisi utama musuh adalah kondisi bergerak *(patrolling)*. Musuh akan selalu melakukan patrol atau *roaming* di area tersebut. Ketika pemain terdeteksi (jarak pemain berada dalam radius *trigger* milik musuh), maka kondisi musuh akan berganti menjadi mengejar pemain. Setelah itu, ketika pemain berbenturan dengan musuh (pemain berada dalam radius OnCollisionEnter milik musuh), musuh berganti kondisi menjadi menyerang pemain. Jika nilai nyawa musuh mencapai 0 maka musuh akan masuk kondisi mati.

Berikut Gambar 3.8 adalah rancangan FSM untuk perilaku musuh dalam permainan dengan lima kondisi utama tersebut.

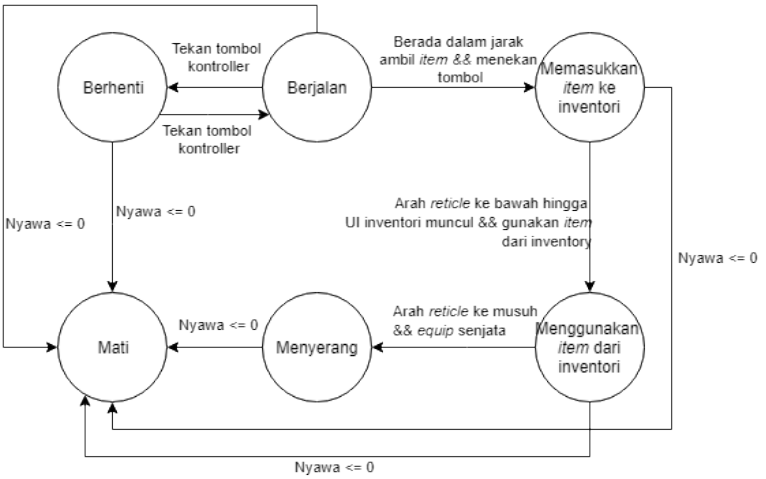


Gambar 3.8 FSM perilaku NPC (musuh)

3.2.6.2.2 FSM Perilaku *Player*

Perilaku untuk *Player* dibagi menjadi enam, yaitu kondisi berhenti, kondisi berjalan, kondisi memasukkan *item* ke inventori, kondisi menggunakan *item,* kondisi menyerang, dan kondisi mati. Kondisi pertama *Player* ketika memulai permainan adalah diam. Ketika pemain menekan tombol pada Kontroller, maka *Player* akan berganti kondisi menjadi berjalan dan berhenti kembali ketika pemain menekan lagi tombol pada Kontroller. Ketika pemain memasang senjata dan pemain berada dalam jarak serang yang dimiliki musuh, *Player* akan berganti kondisi menjadi menyerang NPC. Ketika *Player* menggunakan *item* dari inventori, maka *Player* berada dalam kondisi menambah *item* ke dalam inventori. Jika nilai nyawa *Player* mencapai 0 maka *Player*

akan masuk kondisi mati. Diagram dapat dilihat pada Gambar 3.9.



Gambar 3.9 Rancangan FSM Perilaku *Player*

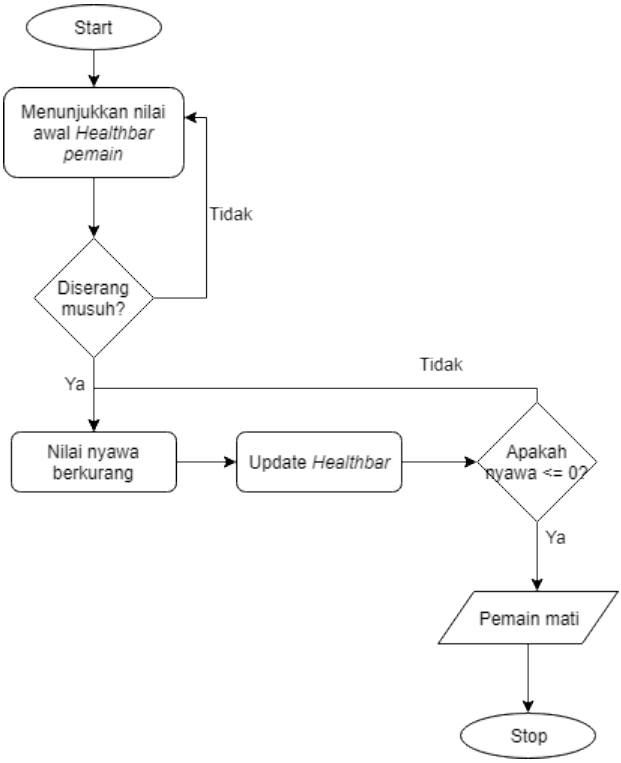
3.2.6.3 Proses Tampilkan *Stats* Pemain

Pemain dapat melihat informasi *stats* seperti level dan *healthpoint* (HP) yang dimiliki dalam bentuk teks dan *healthbar*. Untuk bisa naik level, pemain perlu mengumpulkan nilai *experience point* (EXP) hingga *exp* mencapai jumlah yang diperlukan. *Exp* didapatkan setiap pemain berhasil mengalahkan musuh. Tiap jenis musuh memiliki nilai *exp* yang berbeda-beda sesuai dengan tingkat kesulitan dalam mengalahkan musuh tersebut. Setiap kali pemain naik level, *attack power* pemain terhadap musuh dan nilai maksimum *healthbar* yang dimiliki pemain akan naik. Tidak hanya itu, nyawa pemain saat ini juga akan bertambah.

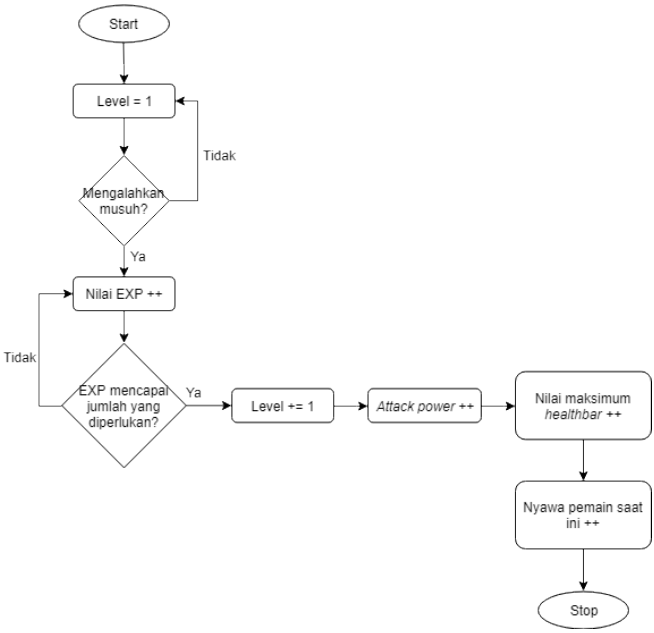
Healthbar sendiri akan terus berkurang setiap kali serangan musuh mengenai pemain. Bila nilai *healthbar* mencapai

angka 0, maka pemain akan kalah dan pesan *gameover* pun akan muncul.

Gambar 3.10 dan Gambar 3.11 adalah *flowchart* proses menampilkan informasi *healthbar* pemain dan *flowchart* proses menampilkan informasi level pemain.



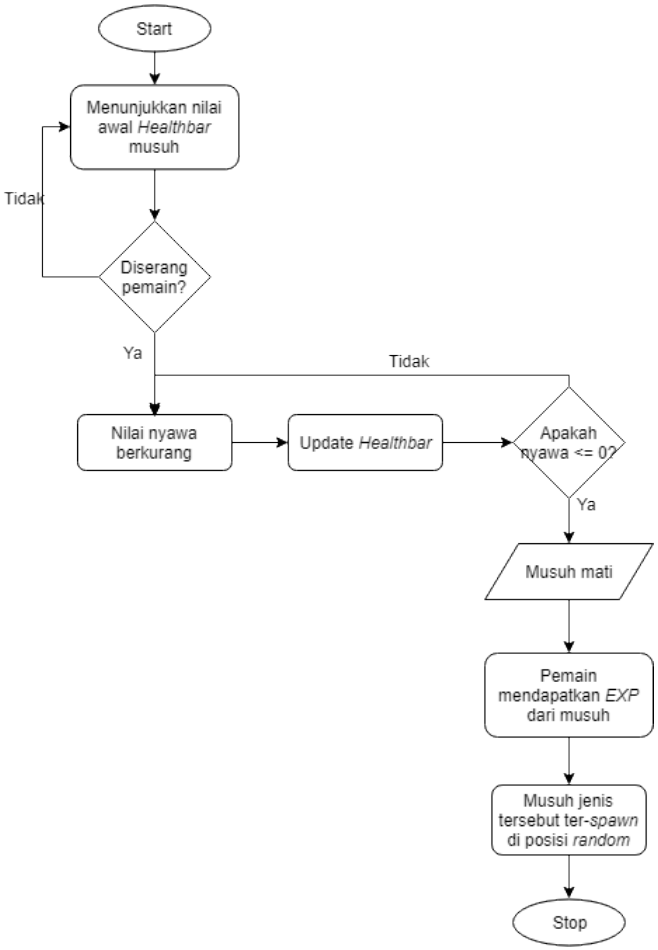
Gambar 3.10 Flowchart Menampilkan Informasi *Healthbar* Pemain



Gambar 3.11 Flowchart Menampilkan Informasi Level Pemain

3.2.6.4 Proses Tampilkan *Healthbar* Musuh

Sama seperti pemain, musuh juga memiliki *healthbar*. *Healthbar* musuh akan berkurang pula bila serangan pemain mengenai musuh. Bila *healthbar* musuh mencapai 0, maka musuh akan mati. Pemain lalu mendapatkan EXP dari musuh yang dikalahkan tersebut. Musuh dengan jenis yang sama pun akan ter-*spawn* pada posisi yang *random* 5 detik setelah musuh tadi mati. Gambar 3.12 menunjukkan *flowchart* proses menampilkan informasi nyawa *(healthbar)* musuh.

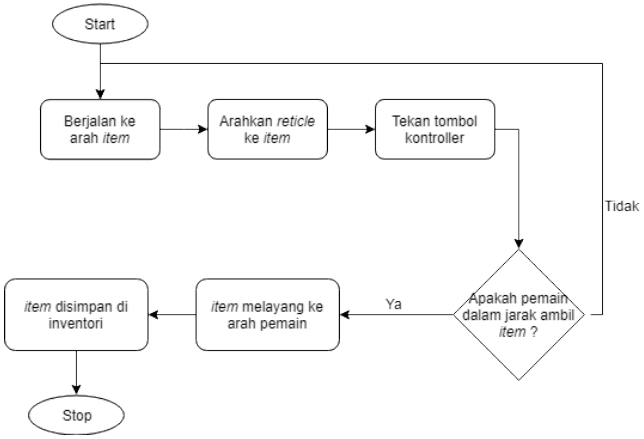


Gambar 3.12 Flowchart Menampilkan Informasi *Healthbar* Musuh

3.2.6.5 Proses Memasukkan *Item* ke Inventori

Pemain dapat mengambil *item* yang tergeletak di jalan dan memasukkannya ke inventori dengan cara mengarahkan *reticle* ke *item* dan menekan tombol pada controller (*gamepad)*.

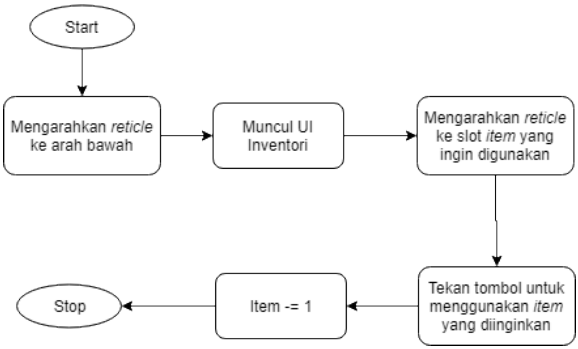
Bila pemain memang sudah berada dalam jarak ambil *item* tersebut, maka *item* akan melayang ke arah pemain sebelum menghilang. Gambar 3.13 menampilkan *flowchart* proses memasukkan *item* ke dalam inventori.



Gambar 3.13 Flowchart Proses Memasukkan *Item* ke Inventori

3.2.6.6 Proses Penggunaan *Item* dari Inventori

Pemain dapat menggunakan *item* yang ada di dalam inventori dengan cara mengarahkan *reticle* ke arah bawah hingga UI inventori muncul. Pada UI inventori tersebut, akan muncul beberapa slot, ada yang sudah berisi *item* dan ada yang kosong pula. Pemain lalu mengarahkan *reticle* ke arah slot *item* yang ingin digunakan dan menekan tombol pada *controller*. Jumlah *item* tersebut akan berkurang 1, dan efek yang sesuai dengan atribut *item* tersebut akan terjadi. Gambar 3.14 menunjukkan *flowchart* proses penggunaan *item* dari inventori.

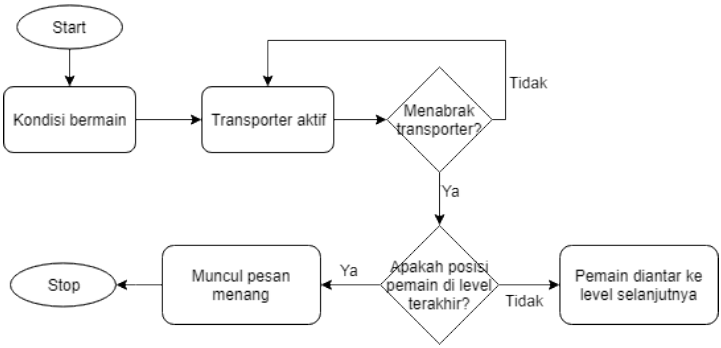


Gambar 3.14 Flowchart Proses Penggunaan *Item* dari Inventori

3.2.6.7 Proses Tampilkan Info Menang

Ketika pengguna telah menemukan *transporter* di area terakhir maka sistem akan menampilkan informasi bahwa pemain telah menang. *Transporter* selain di area terakhir akan mengantarkan pemain ke area selanjutnya. Pembuatan informasi pada garis *finish* membutuhkan area untuk menampilkan informasi dan *Trigger* untuk menampilkan informasi. *Trigger* akan aktif jika kamera/pemain menabraknya, sehingga informasi Pemain Menang akan tampil.

Beberapa hal yang perlu diperhatikan dalam perancangan ini adalah membuat objek *transporter* memberi *collider* pada objek tersebut. Tergantung pada area pemain saat itu, pemain menabrak objek tersebut dapat menyebabkan pemain dikirim ke area selanjutnya atau memunculkan pesan menang. Untuk lebih jelasnya, berikut Gambar 3.15 adalah *flowchart* proses menampilkan informasi Pemain Menang.



Gambar 3.15 Flowchart Menampilkan Informasi Pemain Menang

3.2.6.8 Proses Perancangan Kontrol Aplikasi

Pada aplikasi ini terdapat dua jenis perangkat input, yaitu *Smartphone* android. *Smartphone* akan membantu fungsi pergerakan arah tampilan aplikasi yang dilihat oleh pengguna. Apabila *smartphone* dimiringkan ke atas atau ke bawah, maka pergerakan kamera juga akan bergerak ke atas dan ke bawah. Tidak hanya arah atas dan dan bawah, arah kamera juga akan mengikuti posisi *smartphone* sehingga dapat melihat ke arah 360 derajat.

Tombol pada *controller (gamepad)*, untuk input pada permainan, mengatur interaksi dengan tombol atau objek. Pengguna dapat mengarahkan pointer atau yang dapat disebut *reticle* pada tengah layar ke arah tombol atau objek yang dituju lalu menekan tombol pada *controller* tersebut untuk melakukan suatu interaksi dengan objek yang diinginkan.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi merupakan lingkungan dimana *game* akan dibangun. Lingkungan implementasi dibagi menjadi dua, yaitu lingkungan implementasi berupa perangkat keras dan lingkungan implementasi berupa perangkat lunak, dapat dilihat lebih jelas pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi

Jenis Perangkat	Keterangan
Perangkat Keras	**Pengembangan Aplikasi** Prosesor : Intel® Core(TM) i7-7500U CPU @ 2.70GHz (8 CPUs) ~3.5 GHz Memori : 8 GB **Implementasi Aplikasi** Prosesor: Qualcomm Snapdragon 820 Quad core (2x2.15 GHz Kryo dan 2x1.6 GHz Kryo) Memori : 4 GB

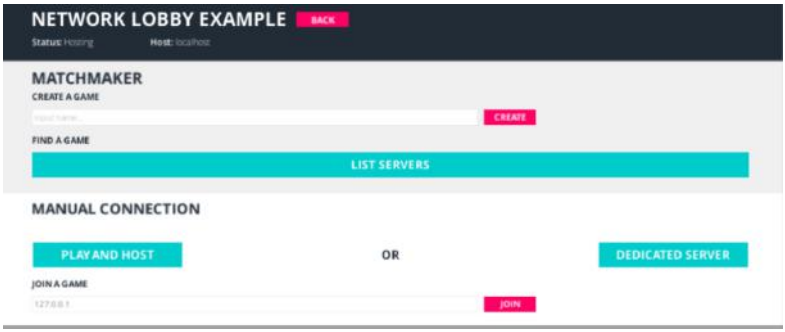
Perangkat Lunak	Sistem Operasi : - Microsoft Windows 10 64 bit (Pengembangan) - Android 7.1.1 (Implementasi) Perangkat Pengembang : - Unity 5.5.6f1 - Mono Studio

4.2 Implementasi Pembuatan Aplikasi

Pada tahap implementasi pembuatan aplikasi ini, akan dibagi ke dalam sembilan subbab. Kesembilan subbab tersebut yaitu meliputi Implementasi Realitas Virtual dengan *Cardboard*, Implementasi Pembuatan *Scene*, Implementasi Import *Assets*, Implementasi *Load* Objek ke *Scene*, Implementasi Penambahan Audio, Implementasi Pembuatan *Script*, Implementasi Kontrol Aplikasi, dan Implementasi Mendapatkan Informasi.

4.2.1 Implementasi Antarmuka Pemilihan Mode Permainan

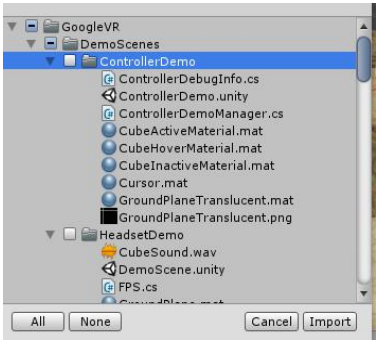
Pemain harus memilih mode permainan terlebih dahulu : apakah ingin bermain sendiri atau bermain bersama pemain lain. Untuk bermain sendiri, cukup tekan tombol “Play and Host”, sedangkan untuk agar bisa bermain dengan pemain lain, pemain sebagai Host membuat nama server terlebih dahulu. Selanjutnya, pemain Client menekan tombol “List Servers”. Di dalamnya, pemain Client akan melihat nama server yang tadi dibuat oleh pemain Host. Pemain Client harus menekan tombol “Join” yang ada di sebelah kanan nama server. Setelah itu, semua pemain akan diantarkan ke dalam permainan.



Gambar 4.1 Antarmuka Pemilihan Mode Permainan

4.2.2 Implementasi Realitas Virtual dengan Cardboard

Untuk pembuatan projek virtual reality pada perangkat bergerak, diperlukan SDK Google Carboard yang dapat diunduh pada https://developers.google.com/vr. Karena pengembangan aplikasi ini dilakukan pada Unity maka pilih Google VR SDK for Unity. Setelah SDK sudah diunduh maka import package kedalam projek Unity. Caranya dengan memilih menu "Assets" lalu pilih "Import Package" dan "Custom Package". Pilih SDK yang diunduh dan tunggu hingga proses load assets selesai. Jika sudah pilih tombol import. Perhatikan Gambar 4.2.



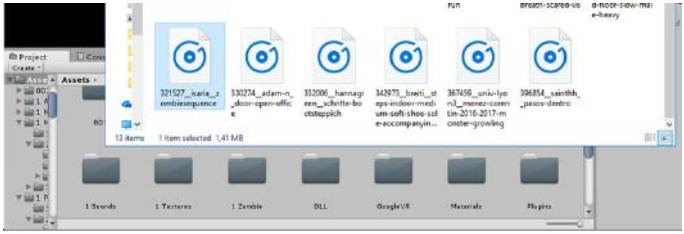
Gambar 4.2 Import Unity Package

4.2.3 Implementasi Pembuatan *Scene*

Untuk menambahkan scene, hal yang perlu dilakukan adalah menuju menu “File” lalu pilih “New Scene”. Setelah scene baru terbuat, kita dapat menambah game objek kedalam scene dan mengatur sesuai rancangan yang dibuat.

4.2.4 Implementasi Import Assets

Implementasi import assets dapat dilakukan dengan cara *drag and drop* dari file explore ke tab “Project” didalam folder “Assets” pada unity. Untuk lebih jelasnya dapat dilihat pada Gambar 4.3. Kita juga dapat mengatur pengelompokan asset dengan membuat folder-folder supaya lebih rapi dan memudahkan dalam pencarian asset.



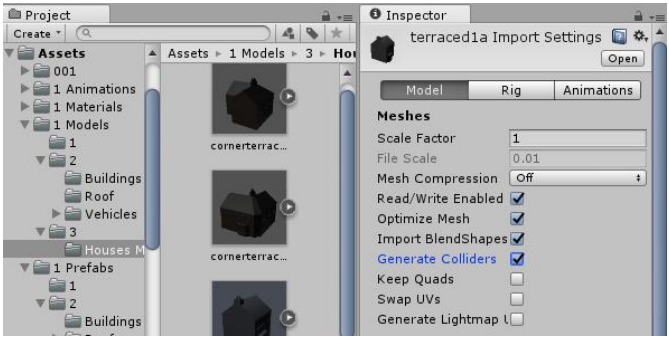
Gambar 4.3 Import *Asset* ke Proyek Unity

4.2.5 Implementasi Load Objek ke Scene

Untuk memasukan objek-objek ke dalam scene dapat dimasukkan dengan cara *drag and drop* berkas dari tab “Project” ke dalam *Scene.* Untuk melihat properti objek mengenai posisi atau yang lainnya, pilih objek yang akan dilihat propertinya. Informasi properti terdapat pada tab “Inspector”. Pada tab “Inspector” dapat ditambahkan beberapa komponen yang mempengaruhi objek pada *scene*.

Untuk objek 3D yang berasal dari file ekspor aplikasi blender dapat langsung ditambahkan *collider* agar dapat

mendeteksi tumbukan antar objek. Hal ini diperlukan untuk mengatur pergerakan kamera agar tidak menembus objek. Langkah-langkahnya yaitu pada aset berekstensi .fbx aktifkan *Generate Collider,* kemudian klik tombol *Apply* (Gambar 4.4).



Gambar 4.4 Generate *Collider* pada Model 3D

Sedangkan untuk animasi yang berasal dari aplikasi blender dapat dilihat pada bagian "Animations". Nama aplikasi dan pengaturan lainnya dapat juga diubah melalui bagian "Animations".

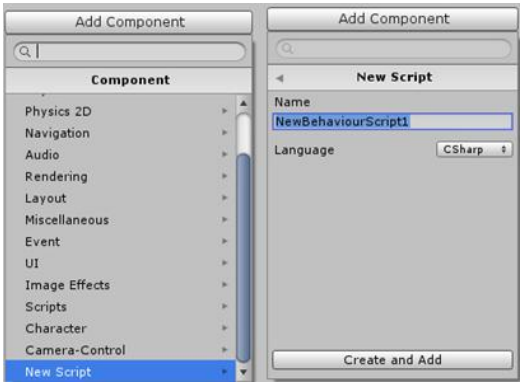
4.2.6 Implementasi Penambahan Audio

Untuk menambah audio langkah pertama yang harus dilakukan adalah pilih gameobject yang akan menyimpan audio tersebut. Kemudian pada tab "Inspector" tambahkan komponen *Audio Source*, jika berhasil maka akan muncul komponen *Audio Source* pada tab "Inspector" seperti pada Gambar 4.5. Pada komponen *Audio Source* terdapat *Audio Clip* yang berfungsi mengambil media music (.mp3 atau .wav) dari tab "Project". Hilangkan centang pada *Play on awake* jika tidak ingin memutar audio saat *Load Scene* dan centang *Loop* jika ingin memutar kembali audio setelah mencapai akhir.

Gambar 4.5 Inspector Komponen Audio Source

4.2.7 Implementasi Penambahan Script

Untuk membuat *script* yang mempengaruhi objek adalah dengan memilih terlebih dahulu objek yang akan diberi *script*. Pada tab "Inspector" tambahkan komponen *script*. Jika tidak ada *script* yang sesuai, kita akan membuat file *script* baru dan mengubah isi file sesuai yang kita inginkan (Gambar 4.6).



Gambar 4.6 Pembuatan *Script* Baru

4.2.8 Implementasi Kontrol Aplikasi

Kontrol aplikasi adalah suatu cara untuk interaksi pengguna dengan sistem. Input yang digunakan adalah tombol yang ada pada kontroller.

Untuk berinteraksi dengan segala object, pengguna mengarahkan *reticle* ke obyek tersebut dan menekan tombol pada Kontroller. Untuk lebih jelasnya dapat dilihat pada Kode Sumber 4.1.

```
// Otherwise if the Google VR button, or the Gear VR touchpad is pressed
        else if (Input.GetButtonDown("Fire1")) {
            GameObject currentGazeObject = gazeInputModule.GetCurrentGameOb
ject();
            if (currentGazeObject != null) {
                if (currentGazeObject.layer == 5){
                    return;
                }
            }
            // Change the state of moveForward
            moveForward = !moveForward;
            if (moveForward == false) { myCC.SimpleMove(Vector3.zero); }
        }
```

Kode Sumber 4.1 Fungsi untuk Berjalan

4.2.8.1 Implementasi Jalan Otomatis *(Autowalk)*

Pengguna dapat menelusuri tempat dengan cara berjalan otomatis. Pengguna hanya perlu menekan tombol pada Cardboard untuk menggerakkan kamera ke depan dan menekan kembali tombol tersebut untuk menghentikan gerak kamea ke depan. Trigger tersebut akan mengubah nilai dari variable bertipe boolean “walking” (lihat Kode Sumber 4.2). Untuk itu diperlukan sebuah *script* yang mengatur gerakan kamera supaya dapat berjalan otomatis.

```
// Use this for initialization
    void Start () {
        // Get CharacterController
        myCC = GetComponent<CharacterController>();
        // Find GazeInputModule
        gazeInputModule = GameObject.FindObjectOfType<GazeInputMod
uleInventory>();
    }

    // Update is called once per frame
    void Update ()
    {
        if (!isLocalPlayer)
        {
            return;
        }

 // Kalau ada obyek interaktif di pandangan kita dan berada dalam
jarak ambil
        if (allowMovement == false || GameObject.Find("InventoryUI
(Clone)") != null) {
            // nggak gerak
            if (moveForward == true) { moveForward = false; }
        }
        // Otherwise if the Google VR button, or the Gear VR touch
pad is pressed
        else if (Input.GetButtonDown("Fire1")) {
            GameObject currentGazeObject = gazeInputModule.GetCurr
entGameObject();
            if (currentGazeObject != null) {
                if (currentGazeObject.layer == 5){

                    return;
                }
            }
            // Change the state of moveForward
            moveForward = !moveForward;
            if (moveForward == false) { myCC.SimpleMove(Vector3.ze
ro); }
        }
        // Check to see if I should move
        if (moveForward) {
            // Find the forward direction
            Vector3 forward = Camera.main.transform.TransformDirec
tion(Vector3.forward);
            // Tell CharacterController to move forward
            myCC.SimpleMove(forward * moveSpeed);
        }
```

```
    }

    public void AllowMovement(bool status) {
        allowMovement = status;
        if (moveForward == true) {
            moveForward = false;
        }
    }
}
```

Kode Sumber 4.2 Fungsi *Autowalk*

Pada Kode Sumber 4.3 terdapat variable "head" yang bertipe *GvrHead* (kelas dari package SDK Google VR). Variable digunakan untuk mengambil arah dari kamera. Terdapat juga variabel "speed" yang mengatur kecepatan jalan kamera/pemain.

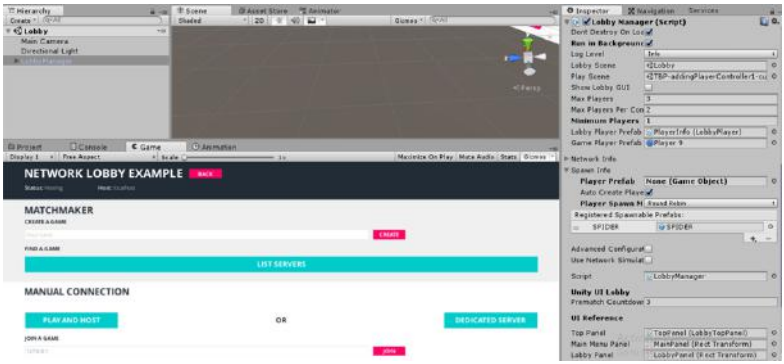
```
using UnityEngine;

[AddComponentMenu("GoogleVR/GvrHead")]
public class GvrHead : MonoBehaviour {
  public bool trackRotation = true;
  public bool trackPosition = true;
  public Transform target;
  public bool updateEarly = false;

  public Ray Gaze {
    get {
      UpdateHead();
      return new Ray(transform.position, transform.forward);
    }
  }

  public delegate void HeadUpdatedDelegate(GameObject head);

  public event HeadUpdatedDelegate OnHeadUpdated;

  void Awake() {
    GvrViewer.Create();
  }

  private bool updated;

  void Update() {
    updated = false;  // OK to recompute head pose.
```

```
    if (updateEarly) {
      UpdateHead();
    }
  }

  // Normally, update head pose now.
  void LateUpdate() {
    UpdateHead();
  }

  // Compute new head pose.
  private void UpdateHead() {
    if (updated) {  // Only one update per frame, please.
      return;
    }
    updated = true;
    GvrViewer.Instance.UpdateState();

    if (trackRotation) {
      var rot = GvrViewer.Instance.HeadPose.Orientation;
      if (target == null) {
        transform.localRotation = rot;
      } else {
        transform.rotation = target.rotation * rot;
      }
    }

    if (trackPosition) {
      Vector3 pos = GvrViewer.Instance.HeadPose.Position;
      if (target == null) {
        transform.localPosition = pos;
      } else {
        transform.position = target.position + target.rotation * p
os;
      }
    }

    if (OnHeadUpdated != null) {
      OnHeadUpdated(gameObject);
    }
  }
}
```

Kode Sumber 4.3 GvrHead

4.2.8.2 Implementasi Pilih Mode Permainan

Pada pemilihan mode permainan, pengguna perlu memberi *trigger on click button* pada tombol yang dilihat (dalam hal ini dibantu sebuah target titik pada tengah kamera (*reticle*) untuk konfirmasi pilihan. Untuk me3mbuat kode program berjalan maka diperlukan untuk menambah komponen *Collider* pada masing-masing menu tombol (Gambar 4.7).



Gambar 4.7 Pengaturan Komponen Mode Permainan

4.2.9 Implementasi Mendapatkan Informasi

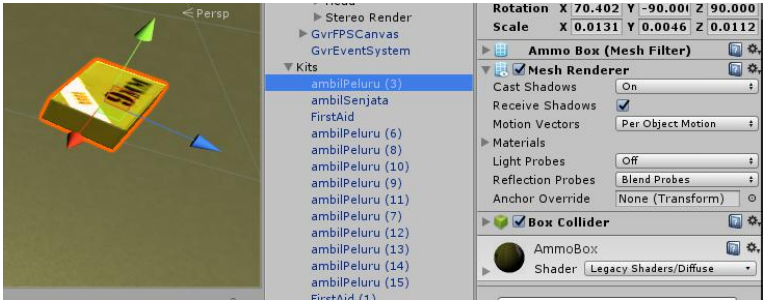
Untuk menampilkan sebuah informasi maka diperlukan sebuah *trigger* yang akan menampilkan informasi pada *screen* android jika *trigger* tersebut aktif. Hal yang diperlukan adalah sebuah canvas yang menampung isi informasi dan kotak *trigger* yang akan menampilkan informasi. Langkah-langkahnya adalah sebagai berikut:

1. Buat objek untuk interaksi, pastikan sudah memiliki collider, selanjutnya membuat fungsi OnCollisionEnter() untuk *player*. Dalam permainan ini terdapat tiga jenis barang bantuan, yaitu

senjata, kotak obat dan apel untuk *player*. Untuk lebih jelasnya lihat contoh Gambar 4.8 dan fungsinya pada Kode Sumber 4.6

2. Tambahkan canvas pada hirarki projek yang di dalamnya memuat latar informasi dan teks yang berisi informasi (Gambar 4.19). Atur mode "Render Canvas" sebagai *world space* dan *event camera* merujuk pada Main Camera di hirarki projek.
3. Pada latar informasi, tambahkan kode program yang mengatur informasi supaya selalu tampil mengikuti arah kamera. Perhatikan Gambar 4.10.

Jika menginginkan canvas mengikuti suatu objek berjalan, masukkan canvas ke dalam *child* objek tersebut, terutama jika objek memiliki kepala. Untuk lebih jelasnya dapat dilihat pada Gambar 4.11.

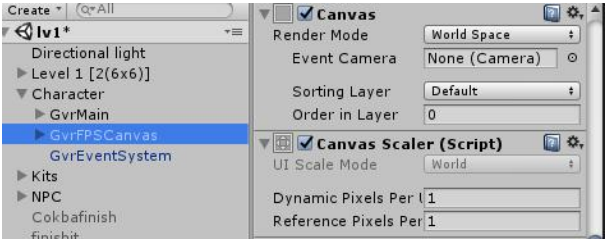


Gambar 4.8 Membuat gameobject *item*

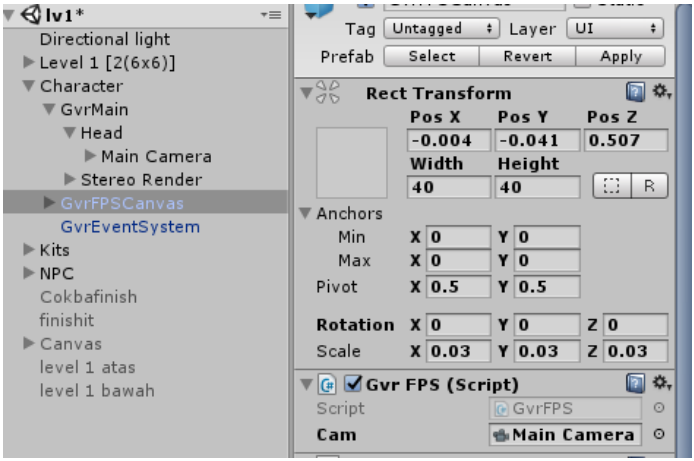
```
CreateEvent(this.gameObject, EventTriggerType.PointerDown, PickItemUp);
CreateEvent(this.gameObject, EventTriggerType.PointerEnter,OnPointerEnter);
CreateEvent(this.gameObject, EventTriggerType.PointerExit, OnPointerExit);


// Create an Event on an Event Trigger
private void CreateEvent(GameObject o, EventTriggerType type, UnityAction a
ction) {
   EventTrigger.Entry entry = new EventTrigger.Entry();
   entry.eventID = type;
   entry.callback.AddListener((eventData) => { action.Invoke(); });
   _eventTrigger.triggers.Add(entry);}

   private void PickItemUp() {
     if (_pickedUp) return;
```

```
     // Check how far away the player is from the item
     var distance = Vector3.Distance(this.transform.position, Camera.main.t
ransform.position);

    // if we're too far away, don't allow the user to pick the item up
    if (distance > itemData.maxPickupDistance) {return;}
     var result = _vrInventory.AddItem(itemData.name, quantity);
     if (result != VRInventory.ePickUpResult.Success) {
       // if we weren't able to pick up the item, don't continue
       return;}

     // prevent this event from being fired again
     _pickedUp = true;

    // Disable item collision (it can get in the way of the camera)
    var collider = this.GetComponent<Collider>();
    if(collider != null) collider.enabled = false;

    // Move the item to the player as a 'pick up' animation
   var position = Camera.main.transform.position + new Vector3(0, -
0.5f, 0);
   iTween.MoveTo(this.gameObject,
   iTween.Hash("position", position,
               "time", 0.35f,
               "oncomplete", "ItemPickupAnimationComplete",
               "oncompletetarget", this.gameObject));

   var sound = itemData.pickupSound;
   if (quantity > 1 && itemData.pickupMultipleSound != null) {
     sound = itemData.pickupMultipleSound;}

   if (sound != null) {
     _audioSource.clip = sound;
     _audioSource.Play();
    }
 }
```

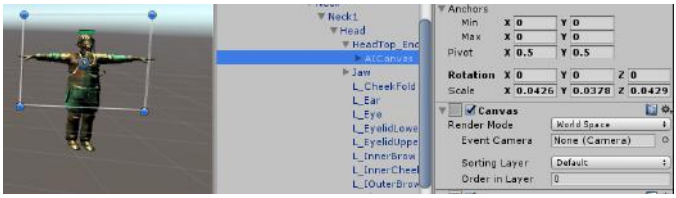
Kode Sumber 4.4 Menekan tombol pada *controller* saat *reticle* focus pada *item*



Gambar 4.9 Membuat Area Informasi



Gambar 4.10 Mengatur posisi teks informasi terhadap kamera



Gambar 4.11 Mengatur posisi teks informasi musuh bergerak terhadap kamera

4.3 Implementasi Aplikasi Permainan

4.3.1 Implementasi Multiplayer

Pada saat pengguna memilih mode permainan *Multiplayer*, pengguna sebagai *Player 1* diharuskan untuk membuat koneksi server baru dari aplikasi permainan yang dijalankan di Unity dengan cara memilih tombol "Create" dan memasukkan suatu nama untuk koneksi server tersebut. Selanjutnya, pengguna sebagai *Player 2* memilih tombol "List Servers" yang akan menampilkan koneksi server yang dibuat *Player 1* tadi. Tekan

button “Join” yang ada di sisi kanan koneksi server tersebut. Sistem akan menunggu sejenak, sebelum memberi response dengan cara menghilangkan *button* “Join” dan mengarahkan *Player 2* ke tampilan lain dimana pengguna dapat melihat bahwa daftar pemain yang terhubung pada koneksi server tersebut berjumlah 2.

Selanjutnya, *Player 1* memulai permainan dengan cara memilih tombol “Join” yang berada di sisi kanan. *Player 2* lalu melakukan hal yang sama. Sistem akan menunggu sejenak, sebelum *scene* permainan tampil di layer masing-masing.

Dalam pengembangan permainan *multiplayer*, pengembang biasanya membuat suatu GameObject yang secara konseptual mewakili *player* di dalam permainan tersebut. Jadikan GameObject ini sebagai suatu Prefab, dan tempelkan semua *script* ke dalamnya, yang mana akan mengendalikan apa saja yang dapat pengguna lakukan saat memainkan permainan ini. Lalu, pada *script* LobbyManager, *assign* Prefab tersebut ke dalam field “Game Player Prefab”. Pengaturan ini dapat dilihat pada Gambar 4.12.

Gambar 4.12 Pengaturan Lobby Manager

4.3.2 Implementasi Finite State Machine Karakter Musuh

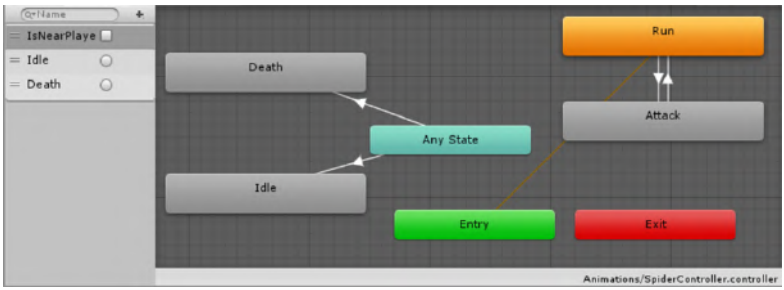
Finite State Machine (FSM) adalah konseptual mesin yang hanya bisa mejalankan satu dari sekian *finite state* dalam satu waktu. FSM biasanya digunakan dalam bidang kecerdasan buatan untuk menentukan perilaku dari *computer-controlled character*, dalam hal ini adalah NPC/musuh.

Unity menyediakan *mechanim system* yang menggunakan FSM untuk mengontrol perilaku animasi. Maka dari itu, penulis menggunakan fitur dalam Unity, yaitu Animator Controller untuk memberikan pengaturan perilaku NPC/musuh yang telah diberi animasi sebelumnya. Dalam animator controller. Keadaan-keadaan utama untuk musuh dibagi menjadi empat, yaitu keadaan ketika diam (*Idle*), keadaan ketika bergerak (*Run*), keadaan menyerang pemain *(Attack)*, serta keadaan ketika mati *(Death)*. Setiap keadaan tersebut memiliki animasi dan penulis juga mnyertakan tambahan animasi untuk membuat perpindahan animasi lebih halus.

Kondisi utama musuh adalah diam (*idle)* untuk sesaat sebelum menjalankan kondisi berjalan *(run)* melakukan patrol/*roaming* dengan arah yang *random*. Syaratnya adalah musuh masih hidup. Ketika posisi *player* sudah masuk dalam radius OnTriggerEnter () milik musuh, maka kondisi 'isNearPlayer' menjadi *true*.

Saat kondisi 'isNearPlayer' menjadi *true*, musuh akan otomatis mengejar pemain (*run*). Selanjutnya, bila pemain sudah masuk ke radius fungsi OnCollisionEnter () milik musuh, musuh akan menjalankan kondisi 'attack' dan animasi 'attack' yang sebelumnya sudah di*assign.* Bila pemain berhasil melarikan diri dari musuh (di luar radius fungsi OnTriggerEnter () musuh), maka kondisi 'isNearPlayer' menjadi *false* dan musuh akan secara otomatis mulai melakukan patroli *(run)* lagi. Bagian

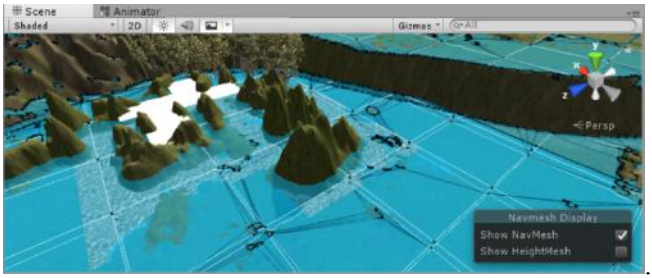
terakhir adalah kondisi saat musuh mati. Kondisi tersebut dapat terjadi saat *Health Point* habis dan kondisi 'Death' terpanggil. Untuk lebih jelasnya dapat melihat Gambar 4.13.



Gambar 4.13 Pengaturan *Animator Controller*

Ketika sudah selesai membuat FSM dengan Animator Controller, yang selanjutnya harus dilakukan adalah menentukan jalan yang dapat dilalui dan tidak dapat dilalui oleh NPC dengan Navigation. Navigation dalam Unity dapat mengatur area mana yang dapat dilalui dan tidak dapat dilalui oleh suatu objek, terutama objek dengan atribut NavMesh Agent.

NavMesh Agent digunakan untuk membuat karakter yang diberi atribut dapat mencapai tujuannya (*goal position*), dalam hal ini musuh, dengan tetap melewati navigation area, sehingga musuh tidak menembus dinding dan sebagainya. Untuk lebih jelasnya dapat dilihat pada Gambar 4.14. Kode program untuk musuh *spider* saat menyerang dapat dilihat pada Kode Sumber 4.5, sedangkan kode program untuk musuh melakukan *patrolling* dapat dilihat pada Kode Sumber 4.6.



Gambar 4.14 Navigation Area

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using MobileVRInventory;

public class SpiderAttack : MonoBehaviour
{
    private Animator _animator;
    private GameObject _player;
    private bool _collidedWithPlayer;

    // TAMBAHAN BUAT ROAMING
    private NavMeshAgent nav;
    private Transform player;

    private VRInventoryExampleSceneController pdamage;

    void Start ()
    {
        _player = GameObject.FindGameObjectWithTag ("Player");
        _animator = GetComponent<Animator> ();

        // TAMBAHAN BUAT ROAMING
        nav = GetComponent<NavMeshAgent>();
        player = GameObject.FindGameObjectWithTag("Player").transform;

        pdamage = FindObjectOfType<VRInventoryExampleSceneController> ();
    }

    void OnTriggerEnter (Collider other)
    {
        if (other.gameObject == _player)
        {
            _animator.SetBool ("IsNearPlayer", true);

            // TAMBAHAN BUAT ROAMING
```

```
            print("SPIDER SEES YOU!");
            nav.SetDestination (player.position);
            //---------- selesai ----------//

            print("enter trigger with _player");
        }
    }

    void OnCollisionEnter(Collision other)
    {
        if (other.gameObject == _player)
        {
            _collidedWithPlayer = true;
            print("enter collided with _player");
        }
    }

    void OnCollisionExit(Collision other)
    {
        if (other.gameObject == _player)
        {
            _collidedWithPlayer = false;
            print("exit collided with _player");
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.gameObject == _player)
        {
            _animator.SetBool ("IsNearPlayer", false);
            print("exit trigger with _player");
        }
    }
    void Attack()
    {
        if (_collidedWithPlayer)
        {
            print("player has been hit");
            pdamage.PlayerTakeDamage (20);
            //pdamage.health -= 20;
            pdamage.healthSlider.value = pdamage.health;
            pdamage.UpdateBars ();
            print ("PLAYER'S HEALTH : " + pdamage.health);
            //pdamage.UpdateBars();
        }
    }

    void Update()
    {
        pdamage.UpdateBars ();
    }
}
```

Kode Sumber 4.5 Musuh menyerang

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
using MobileVRInventory;

public class SpiderMovement : MonoBehaviour
{
    // Use this for initialization
    private NavMeshAgent _nav;
    private Transform _player;

    private Enemy _health;
    private Animator _animator;

    // TAMBAHAN BUAT PATROLLING / ROAMING
    public float wanderRadius;
    public float wanderTimer;
    private float timer;

    void Start ()
    {
        _nav = GetComponent<NavMeshAgent>();
        _player = GameObject.FindGameObjectWithTag("Player").trans
form;
        _health = GetComponent<Enemy> ();

        // ANIMASI
        _animator = GetComponent<Animator> ();

        // TAMBAHAN BUAT ROAMING / PATROLLING
        timer = wanderTimer;
    }

    // ANIMASI
    void Death()
    {
        _animator.SetTrigger ("Death");
    }

    // Update is called once per frame
    void Update ()
    {
        if (_health.Health > 0)
        {
            // TAMBAHAN BUAT ROAMING / PATROLLING
```

```
            timer += Time.deltaTime;
            //print (timer);
            if (timer >= wanderTimer) {
                Vector3 newPos = RandomNavSphere (transform.positi
on, wanderRadius, -1);
                _nav.SetDestination (newPos);
                timer = 0;
            }
        }
        else
        {
            _nav.enabled = false;
            Death ();                // ANIMASI
        }
    }

    // TAMBAHAN BUAT ROAMING / PATROLLING
    public static Vector3 RandomNavSphere(Vector3 origin, float di
st, int layermask) {
        Vector3 randDirection = Random.insideUnitSphere * dist;
        randDirection += origin;
        NavMeshHit navHit;
        NavMesh.SamplePosition (randDirection, out navHit, dist, l
ayermask);
        return navHit.position;}
}
```

Kode Sumber 4.6 Musuh *patrolling*

4.3.3 Implementasi Menelusuri Area *(Autowalk)*

Dalam permainan ini, pemain dapat bergerak maju secara otomatis dengan cara menekan tombol pada Cardboard *(autowalk)*. Selengkapnya dapat dilihat pada Kode Sumber 4.7.

```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;

[RequireComponent(typeof(CharacterController))]
public class DemoAutowalkInventory : NetworkBehaviour {
    // Move speed
    public float moveSpeed = 3f;
    // If the player is allowed to move (don't allow movement when
 looking at UI)
```

```
    public bool allowMovement = true;
    // My character controller
    private CharacterController myCC;
    // Should I move forward or not
    private bool moveForward;
    // Gaze Input Module, to know what object user is looking at
    private GazeInputModuleInventory gazeInputModule;

    // Use this for initialization
    void Start () {
        // Get CharacterController
        myCC = GetComponent<CharacterController>();
        // Find GazeInputModule
        gazeInputModule = GameObject.FindObjectOfType<GazeInputMod
uleInventory>();
    }

    // Update is called once per frame
    void Update ()
    {
        if (!isLocalPlayer)
        {
            return;
        }

        // If there is an interactive object at gaze and it's with
in distance
        if (allowMovement == false || GameObject.Find("InventoryUI
(Clone)") != null) {
            // do not allow movement
            if (moveForward == true) { moveForward = false; }
        }
        // Otherwise if the Google VR button, or the Gear VR touch
pad is pressed
        else if (Input.GetButtonDown("Fire1")) {
            GameObject currentGazeObject = gazeInputModule.GetCurr
entGameObject();
            if (currentGazeObject != null) {
                if (currentGazeObject.layer == 5){

                    return;
                }
            }
            // Change the state of moveForward
            moveForward = !moveForward;
            if (moveForward == false) { myCC.SimpleMove(Vector3.ze
ro); }
        }
```

```
        // Check to see if I should move
        if (moveForward) {
            // Find the forward direction
            Vector3 forward = Camera.main.transform.TransformDirec
tion(Vector3.forward);
            // Tell CharacterController to move forward
            myCC.SimpleMove(forward * moveSpeed);
        }
    }

    public void AllowMovement(bool status) {
        allowMovement = status;
        if (moveForward == true) {
            moveForward = false;
        }
    }
}
```

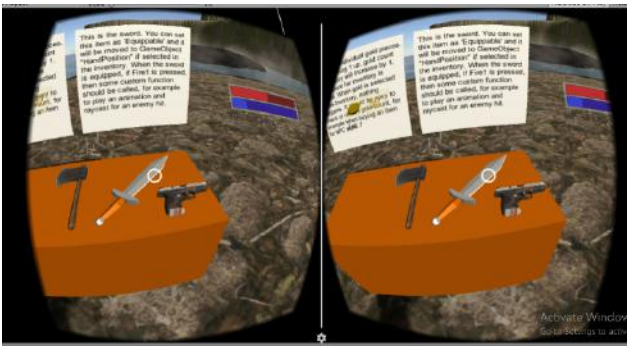
Kode Sumber 4.7 Pemain berjalan

4.3.4 Implementasi Memasukkan *Item* ke Inventori dan Menggunakan *Item* dari Inventori

Dalam implementasi aturan permainan pada aplikasi ini pengguna diminta untuk tetap bertahan hidup dengan mengumpulkan *item* yang ditentukan di jalan ketika mencari jalan keluar dari suatu area. Untuk memasukkan *item* dalam permainan ini, pengguna harus mendekati *item* dan mengeklik tombol pada Cardboard. Bila pemain sudah berada dalam jarak ambil *item* tersebut, maka *item* akan tertarik masuk ke arak pemain. Bila ingin menggunakan *item*, arahkan kamera ke arah bawah hingga UI inventori muncul. Pilih slot *item* yang diinginkan dan klik tombol pada Cardboard lagi untuk menggunakan *item* tersebut.

Implementasi memasukkan *item* ke inventori dan penggunaan *item* dari inventori dapat dilihat pada Gambar 4.15 dan Gambar 4.16. Untuk pembuatan proses memasukkan *item* yang tergeletak ke inventori, diperlukan kode program sesuai Kode Sumber 4.8. Sedangkan untuk proses menggunakan *item*

dari inventori, dapat dilihat pada kode program sesuai Kode Sumber 4.9.

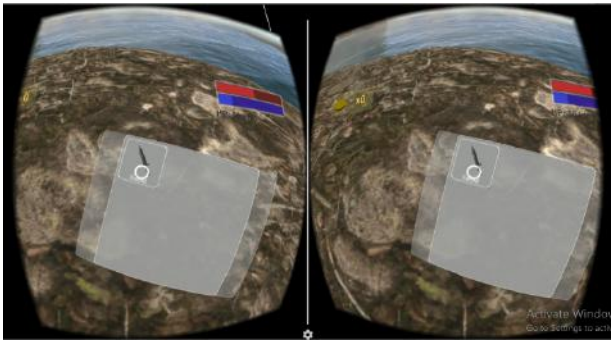


Gambar 4.15 Pemain hendak mengambil *item*

```
CreateEvent(this.gameObject, EventTriggerType.PointerDown, PickItemUp);
CreateEvent(this.gameObject, EventTriggerType.PointerEnter,OnPointerEnter);
CreateEvent(this.gameObject, EventTriggerType.PointerExit, OnPointerExit);


// Create an Event on an Event Trigger
private void CreateEvent(GameObject o, EventTriggerType type, UnityAction a
ction) {
   EventTrigger.Entry entry = new EventTrigger.Entry();
   entry.eventID = type;
   entry.callback.AddListener((eventData) => { action.Invoke(); });
   _eventTrigger.triggers.Add(entry);}

   private void PickItemUp() {
     if (_pickedUp) return;

     // Check how far away the player is from the item
     var distance = Vector3.Distance(this.transform.position, Camera.main.t
ransform.position);

    // if we're too far away, don't allow the user to pick the item up
    if (distance > itemData.maxPickupDistance) {return;}
     var result = _vrInventory.AddItem(itemData.name, quantity);
     if (result != VRInventory.ePickUpResult.Success) {
       // if we weren't able to pick up the item, don't continue
       return;}

     // prevent this event from being fired again
     _pickedUp = true;
```

```
    // Disable item collision (it can get in the way of the camera)
    var collider = this.GetComponent<Collider>();
    if(collider != null) collider.enabled = false;

    // Move the item to the player as a 'pick up' animation
   var position = Camera.main.transform.position + new Vector3(0, -
0.5f, 0);
   iTween.MoveTo(this.gameObject,
   iTween.Hash("position", position,
               "time", 0.35f,
               "oncomplete", "ItemPickupAnimationComplete",
               "oncompletetarget", this.gameObject));

   var sound = itemData.pickupSound;
   if (quantity > 1 && itemData.pickupMultipleSound != null) {
     sound = itemData.pickupMultipleSound;}

   if (sound != null) {
     _audioSource.clip = sound;
     _audioSource.Play();
    }
 }
```

Kode Sumber 4.8 Memasukkan *item* ke inventori



Gambar 4.16 Pemain menggunakan *item* pedang dari inventori

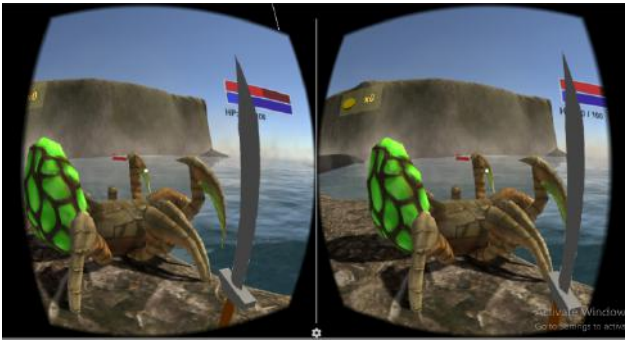
```
//When we're using sword
public override void OnItemUsed() {
  if (swingInProgress) return;
```

```
  var itemGazedAt = gazeInputModuleInventory.GetCurrentGameObject(
);

  if(itemGazedAt != null) {
  // Don't swing at inventory items, just pick them up
       var inventoryItem = itemGazedAt.GetComponent<InventoryItem>
();
       if (inventoryItem != null) return; }

       SwingBegin();

       if (itemGazedAt != null) {
        var enemy = itemGazedAt.GetComponent<Enemy>();

        if (enemy != null) {
          var distance = Vector3.Distance(itemGazedAt.transform.po
sition, Camera.main.transform.position);
          if (distance <= SwingDistance) {
  // TakeDamage returns true if the target has been killed
            if (enemy.TakeDamage(35)) {
            // notify the controller that we have won
                    exampleController.Victory();
                   }
               }
            }
            }

            // When the reticle focuses on Dummy
         if (itemGazedAt != null)
         {
             var dummy = itemGazedAt.GetComponent<Dummy>();
             if (dummy != null) {
                 var distance = Vector3.Distance(itemGazedAt.trans
form.position, Camera.main.transform.position);
                 if (distance <= SwingDistance) {
  // TakeDamage returns true if the target has been killed
                     if (dummy.TakeDamage(35)) {
                  // notify the controller that we have won
                      exampleController.transporterNotif();
                     }
                 }
             }
         }
```

Kode Sumber 4.9 Menggunakan *item* dari inventori

4.3.5 Implementasi Menyerang Musuh

Pemain dapat menyerang musuh dengan cara memilih senjata dari inventory, memfokuskan *reticle* ke arah musuh, lalu menekan tombol yang ada pada Cardboard. Bila pemain melakukannya saat sedang berada dalam radius jarak serang yang dimiliki musuh, *health bar* yang dimiliki musuh akan berkurang. Implementasi pemain menyerang musuh dapat dilihat lebih jelas pada Gambar 4.17, sedangkan implementasi dari kode programnya dapat dilihat pada Kode Sumber 4.10.



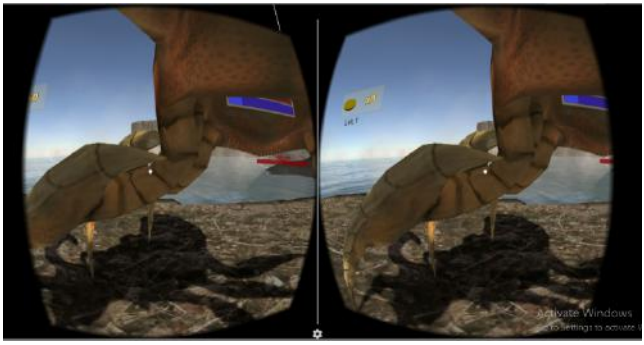
Gambar 4.17 Pemain menyerang musuh

```
//kalo ada object yang diliat reticle
if (itemGazedAt != null) {
    var enemy = itemGazedAt.GetComponent<Enemy>();

    if (enemy != null) {    //kalo yang diliat object Enemy
     var distance = Vector3.Distance(itemGazedAt.transform.position, Camera
                    .main.transform.position);
     if (distance <= SwingDistance) {
       // TakeDamage returns true if the target has been killed
       if (enemy.TakeDamage(35)) {
         notify the controller that we win
         exampleController.Victory(); }
      }
     }
 }
```

Kode Sumber 4.10 Pemain menyerang musuh

4.3.6 Implementasi Diserang Musuh

Selain bias menyerang musuh, tentunya pemain dapat diserang musuh yang berkeliaran pula. Saat serangan dari musuh mengenai pemain, *healthbar* pemain akan berkurang. Pemain diharuskan untuk berpikir cepat dan menentukan tindakan apa yang dapat dilakukan oleh pemain saat *healthbar* kian menipis. Bila *health point* yang dimiliki pemain mencapai 0, system akan langsung menampilkan scene *GameOver* dan permainan pun selesai. Implementasi permainan saat pemain diserang oleh musuh dan implementasi tampilan GameOver dapat dilihat pada Gambar 4.18 dan Gambar 4.19, sedangkan kode programnya dapat dilihat pada Kode Sumber 4.11 dan Kode Sumber 4.12 secara berurutan.



Gambar 4.18 Pemain diserang musuh

```
public void PlayerTakeDamage (int amount)
{
    health -= amount;
    UpdateBars ();
    if (health < 0)
    {
        print ("PLAYER IS DEFEATED!");
        SceneManager.LoadScene ("GameOver");
    }
}
```

Kode Sumber 4.11 Pemain diserang musuh

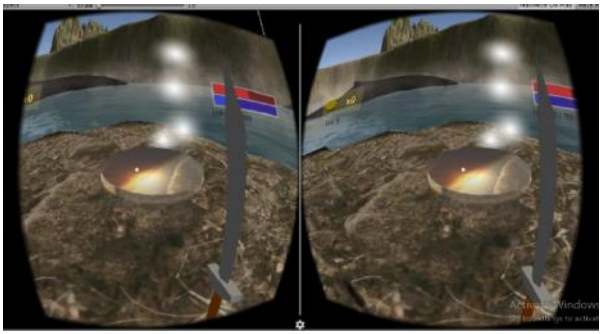
Gambar 4.19 Pemain kalah (GameOver)

```
if (health < 0)
{
    print ("PLAYER IS DEFEATED!");
    SceneManager.LoadScene ("GameOver");
}
```

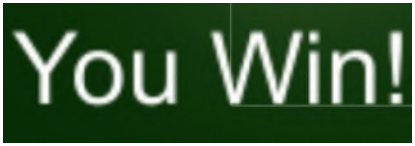
Kode Sumber 4.12 Pemain kalah

4.3.7 Implementasi Menemukan *Transporter*

Pemain diharuskan untuk mencari *transporter* untuk bisa menyelesaikan misi di area tersebut dan memenangkan permainan. *Transporter* akan aktif secara sendirinya bila pemain dapat mengalahkan musuh-musuh tertentu. Setelah transporter berhasil aktif, pemain menabrakkan diri dengan transporter tersebut. Sistem akan mengantarkan pemain ke area selanjutnya. Di area terakhir, saat pemain mengaktifkan *transporter* terakhir, pemain berhasil memenangkan permainan. Implementasi aktifasi *transporter* dan implementasi pesan menang dapat dilihat pada Gambar 4.20 dan Gambar 4.21, sedangkan kode program aktifasi *transporter,* proses pengantaran pemain ke area selanjutnya, dan layar menang dapat dilihat lebih lanjut pada Kode Sumber 4.13, Kode Sumber 4.14 dan Kode Sumber 4.15.



Gambar 4.20 Aktifasi *transporter*



Gambar 4.21 Pesan menang

```
// ACTIVATE TRANSPORTER
activateMe = true;
myobject.SetActive (true);
```

Kode Sumber 4.13 Aktifasi *transporter*

```
void OnTriggerEnter(Collider collider)
    {
        if(collider.gameObject.tag =="Player" && disableTimer<=0)
        {
            foreach(TeleportPad tp in FindObjectsOfType<TeleportPad>())
            {
                if(tp.code==code && tp!=this){
                    tp.disableTimer = 3;
                    Vector3 position = tp.gameObject.transform.position;
                    position.y += 2;
                    collider.gameObject.transform.position = position;
                }
            }
        }
    }
}
```

Kode Sumber 4.14 Pengantaran pemain ke area selanjutnya

```
void OnTriggerEnter(Collider collider)
    {
        if(collider.gameObject.tag =="Player" && disableTimer<=0)
        {
            SceneManager.LoadScene(“Win”);
        }
    }
}
```

Kode Sumber 4.15 Pesan menang

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang telah dilakukan.

5.1 Lingkungan Pengujian

Pada proses uji coba ini, lingkungan dibedakan menjadi lingkungan perangkat keras dan perangkat lunak. Berikut ini akan dijelaskan mengenai tiap-tiap lingkungan uji coba aplikasi.

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini dicantumkan pada Tabel 5.1. Uji coba ini dilakukan menggunakan kacamata VR Box dengan *earphone* sebagai alat bantu dengar untuk pengguna.

Tabel 5.1 Lingkungan Perangkat Keras

No.	Deskripsi
1	ASUS A4556U Prosessor : Intel® Core(TM) i7-7500U CPU @ 2.70GHz (8 CPUs) ~3.5 GHz Memori : 8.00 GB Sistem Operasi : Windows 10
2	Model Perangkat ASUS Zenfone 4 Selfie Chipset : Qualcomm Snapdragon 820 CPU : Quad-core (2x2.18 GHz Kryo & 2x1.36 GHz Kryo) Memori : 4.00 GB Sistem Operasi : Android 7.1.1 API level : 24

5.2 Skenario dan Hasil Uji Coba

Pada subbab ini dijelaskan mengenai skenario yang dilakukan untuk mendapatkan hasil uji coba aplikasi. Skenario uji coba aplikasi dilakukan untuk mengetahui ketercapaian aplikasi dalam memenuhi kebutuhan fungsionalitas dan kebutuhan non-fungsionalitas.

5.2.1 Pengujian Fungsionalitas

Pengujian fungsionalitas aplikasi ini dapat dilakukan secara mandiri. Pengujian ini bertujuan untuk mengetahui kesesuaian keluaran dari tiap tahap dan langkah penggunaan fitur terhadap skenario yang dipersiapkan. Skenario yang dibuat mengacu pada kasus penggunaan yang telah dijelaskan pada subbab 3.2.4. Skenario uji coba fungsionalitas yang dilakukan terhadap aplikasi yang dibangun dijelaskan pada Tabel 5.2.

Tabel 5.2 Skenario Uji Coba Fungsionalitas

Kode Uji Coba	Nama Uji Coba
UJ-UC-001	Uji Memilih Mode Permainan
UJ-UC-002	Uji Menelusuri Area
UJ-UC-003	Uji Melawan Musuh
UJ-UC-004	Uji Mengambil *Item*
UJ-UC-005	Uji Menggunakan *Item*
UJ-UC-006	Uji Menemukan *Transporter*

Setiap skenario akan dijelaskan mengenai kondisi awal, masukkan, dan keluaran yang diharapkan, kondisi akhir, dan hasil uji coba. Berikut ini merupakan penjabaran hasil setiap uji coba yang dilakukan.

5.2.1.1 Uji Memilih Mode Permainan

Uji coba ini bertujuan untuk membuat pengguna bermain dengan mode permainan yang diinginkan (*multiplayer* atau *single-player*). Skenario dan hasil pengujian secara lengkap dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil Uji Coba Memilih Mode Permainan

Kode Uji Coba	UJ-UC-001
Kondisi Awal	Aplikasi menampilkan pilihan mode permainan : *Multiplayer* dan *Single-Player*
Skenario 1 *(Pengguna menekan layer Android ke pilihan mode permainan yang ingin dipilih)*	
Masukan	Menekan layar Android pada kotak pilihan mode permainan
Keluaran yang diharapkan	Kotak pilihan dapat di-*klik*
Kondisi Akhir	Permainan dimulai dengan mode permainan yang dipilih terbuka
Hasil Uji Coba	Berhasil

5.2.1.2 Uji Menelusuri Area

Uji coba ini bertujuan untuk mengarahkan pengguna berkeliling dalam realitas virtual area. Skenario dan hasil pengujian secara lengkap dapat dilihat pada Tabel 5.4.

Tabel 5.4 Hasil Uji Coba Menelusuri Area

Kode Uji Coba	UJ-UC-002
Kondisi Awal	Objek sudah tampil pada aplikasi
Skenario 1 *(Pengguna mengarahkan smartphone ke atas, bawah, kanan, kiri, dan belakang)*	
Masukan	Mengarahkan *smartphone* ke sembarang arah

Keluaran yang diharapkan	Layar *smartphone* menampilkan objek 3D sesuai arah *smartphone*
Kondisi Akhir	Layar *smartphone* menampilkan objek 3D sesuai arah *smartphone*
Hasil Uji Coba	Berhasil
Skenario 2 *(Pengguna mengeklik tombol pada controller dalam kondisi berhenti)*	
Masukan	Mengeklik tombol pada *controller* (kondisi berhenti)
Keluaran yang diharapkan	Kamera bergerak ke depan
Kondisi Akhir	Kamera bergerak ke depan
Hasil Uji Coba	Berhasil
Skenario 3 *(Pengguna mengeklik tombol pada controller dalam kondisi berjalan)*	
Masukan	Mengeklik tombol pada *controller* (kondisi berjalan)
Keluaran yang diharapkan	Kamera berhenti
Kondisi Akhir	Kamera berhenti
Hasil Uji Coba	Berhasil
Skenario 4 *(Pengguna berhenti hingga nyawa habis)*	
Masukan	Mengarahkan kamera selain ke arah lantai sehingga berhenti
Keluaran yang diharapkan	Nyawa habis, menampilkan *game over scene*
Kondisi Akhir	*Game Over scene* tampil
Hasil Uji Coba	Berhasil

5.2.1.3 Uji Melawan Musuh

Uji coba ini bertujuan untuk memberi pengguna informasi mengenai musuh saat pemain menyerang. Skenario dan hasil pengujian secara lengkap dapat dilihat pada Tabel 5.5.

Tabel 5.5 Hasil Uji Coba Melawan Musuh

Kode Uji Coba	UJ-UC-003
Kondisi Awal	*Health Point* musuh utuh
Skenario 1 *(Pengguna menyerang musuh saat musuh sudah berada dalam jarak serang senjata)*	
Masukan	Menyerang musuh saat musuh sudah berada dalam jarak serang senjata
Keluaran yang diharapkan	*Health Point* musuh berkurang dan nilai *healthbar* mengalami pembaruan
Kondisi Akhir	*Health Point* musuh berkurang dan nilai *healthbar* mengalami pembaruan
Hasil Uji Coba	Berhasil
Skenario 2 *(Pengguna menyerang musuh saat musuh sudah berada dalam jarak serang senjata)*	
Masukan	Menyerang musuh saat musuh berada di luar jarak serang senjata
Keluaran yang diharapkan	Tidak terjadi perubahan pada *Health Point* musuh sehingga *Healthbar* tidak mengalami perubahan
Kondisi Akhir	Tidak terjadi perubahan pada Health Point musuh sehingga *Healthbar* tidak mengalami perubahan
Hasil Uji Coba	Berhasil

5.2.1.4 Uji Memasukkan *Item* ke Inventori

Uji coba ini bertujuan Kasus pengguna ini berfungsi agar aktor dapat memasukkan *item* di area ke inventori. Skenario dan hasil pengujian secara lengkap dapat dilihat pada Tabel 5.6.

Tabel 5.6 Hasil Uji Coba Memasukkan *Item* ke Inventori

Kode Uji Coba	UJ-UC-004
Kondisi Awal	*Item* belum masuk ke dalam inventori
Skenario 1 *(Pengguna mengarahkan reticle ke item untuk mengambil item (kondisi pemain dalam jarak ambil dengan item))*	
Masukan	Mengarahkan *reticle* ke *item* untuk mengambil *item* (kondisi pemain dalam jarak ambil dengan *item*)
Keluaran yang diharapkan	*Item* masuk ke dalam inventori
Kondisi Akhir	- *Item* melayang ke arah pemain - *Item* berhasil masuk ke dalam inventori
Hasil Uji Coba	Berhasil
Skenario 2 *(Pengguna mengarahkan reticle ke item untukmengambil item (kondisi pemain di luar jarak ambil dengan item))*	
Masukan	Mengarahkan reticle ke item dan mengambil *item* (kondisi pemain di luar jarak ambil dengan *item*)
Keluaran yang diharapkan	*Item* tidak masuk ke dalam inventori
Kondisi Akhir	- Pemain berhenti - Item tidak masuk ke dalam inventori
Hasil Uji Coba	Berhasil

5.2.1.5 Uji Menggunakan *Item* dari Inventori

Uji coba ini bertujuan agar aktor dapat menggunakan *item* dari inventori. Skenario dan hasil pengujian secara lengkap dapat dilihat pada Tabel 5.7.

Tabel 5.7 Hasil Uji Coba Menggunakan *Item* dari Inventori

Kode Uji Coba	UJ-UC-005
Kondisi Awal	*Item* sudah tersimpan ke inventori
Skenario 1 *(Pengguna mengarahkan reticle ke arah bawah pada sudut tertentu untuk memunculkan Inventory UI, mengarahkan reticle ke slot item yang diinginkan untuk mengambil item*	
Masukan	Mengarahkan *reticle* ke arah bawah pada sudut tertentu untuk memunculkan Inventory UI, mengarahkan *reticle* ke *item* yang diinginkan untuk mengambil *item*
Keluaran yang diharapkan	- Item pada inventori berkurang - Terdapat perubahan sesuai dengan atribut *item* tersebut
Kondisi Akhir	- Menampilkan inventory UI - Mengurangi jumlah *item* tersebut pada inventori - Memberi efek pada pengguna sesuai dengan atribut pada *item* tersebut
Hasil Uji Coba	Berhasil

5.2.1.6 Uji Menemukan *Transporter*

Uji coba ini bertujuan agar aktor dapat menggunakan *item* dari inventori. Skenario dan hasil pengujian secara lengkap dapat dilihat pada Tabel 5.8.

Tabel 5.8 Hasil Uji Coba Menemukan *Transporter*

Kode Uji Coba	UJ-UC-006
Kondisi Awal	*Transporter* tidak terlihat dan belum aktif
Skenario 1 *(Pengguna mengalahkan musuh tertentu)*	
Masukan	Mengalahkan musuh tertentu
Keluaran yang diharapkan	*Transporter* muncul dan aktif
Kondisi Akhir	*Transporter* muncul dan aktif
Hasil Uji Coba	Berhasil
Skenario 2 *(Pengguna bersentuhan dengan transporter)*	
Masukan	Menyentuh *transporter*
Keluaran yang diharapkan	Pemain dikirim ke level selanjutnya
Kondisi Akhir	Pemain dikirim ke level selanjutnya
Hasil Uji Coba	Berhasil

5.2.2 Pengujian Pengguna

Pengujian pada permainan yang dibangun tidak hanya dilakukan pada fungsionalitas yang dimiliki, tetapi juga pada pengguna untuk melakukan percobaan secara langsung. Pengujian ini berfungsi sebagai pengujian subjektif yang bertujuan untuk mengetahui tingkat keberhasilan game yang dibangun dari sisi pengguna. Hal ini dapat dicapai dengan meminta penilaian dan tanggapan dari pengguna terhadap sejumlah aspek perangkat lunak yang ada.

5.2.2.1 Skenario Uji Coba Pengguna

Dalam melakukan pengujian aplikasi ini, penguji diminta untuk mencoba menggunakan aplikasi yang bersangkutan dan mencoba semua fungsionalitas serta fitur yang tersedia. Selain itu, pengguna juga diminta memberikan saran untuk pengembangan aplikasi selanjutnya.

Pengujian aplikasi oleh pengguna dilakukan dengan sebelumnya memberi informasi seputar aplikasi, kegunaan, dan fitur-fitur yang dimiliki. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba aplikasi dengan spesifikasi lingkungan yang sama persis dengan yang telah diuraikan pada uji coba fungsionalitas.

Jumlah pengguna yang terlibat dalam pengujian aplikasi ini sebanyak 6 orang. Dalam melakukan pengujian, pengguna melakukan percobaan sebanyak satu kali penggunaan. Ketika memberikan penilaian dan tanggapan terhadap aplikasi, penguji diberikan formulir pengujian aplikasi. Formulir pengujian aplikasi ini memiliki beberapa aspek penilaian dan pada bagian akhir terdapat permintaan saran serta kritik untuk perbaikan fitur aplikasi.

5.2.2.2 Daftar Penguji Aplikasi

Pada subbab ini ditunjukkan daftar pengguna yang bertindak sebagai penguji aplikasi yang telah dibangun. Terdapat tiga orang penguji. Daftar nama penguji perangkat lunak ini ditunjukkan pada Tabel 5.9.

Tabel 5.9 Daftar Nama Penguji

No.	Nama	Pekerjaan
1.	Admiral Budi Arviansyah	Mahasiswa
2.	Anugrah DP	Mahasiswa
3.	Aliya Fathma Najihati	Non-mahasiswa
4.	Rozana Lutfa	Pelajar
5.	Imaduddin Ashshiddiqi	Mahasiswa
6.	Nabil Bastomy	Mahasiswa

5.3 Evaluasi Pengujian

Pada subbab evaluasi pengujian ini akan ditunjukan data rekapitulasi dari hasil pengujian fungsionalitas maupun hasil pengujian pengguna yang telah dilakukan sebelumnya pada subbab 5.2. Rekapitulasi masing-masing pengujian akan dijabarkan pada subbab berikut.

5.3.1 Evaluasi Pengujian Fungsionalitas

Evaluasi pengujian fungsionalitas dilakukan dengan menampilkan data rekapitulasi aplikasi yang telah dipaparkan pada subbab 5.2. Dalam hal ini, rekapitulasi disusun dalam bentuk tabel yang dapat dilihat pada Tabel 5.10. Dari data yang terdapat pada tabel tersebut, diketahui bahwa aplikasi yang dibuat telah memenuhi kasus penggunaan yang telah ditentukan.

Tabel 5.10 Rekapitulasi Hasil Uji Fungsionalitas

Kode	Uji Coba		Hasil
UJ-UC-001	Uji Memilih Mode Permainan	Skenario 1	Berhasil
UJ-UC-002	Uji Menelusuri	Skenario 1	Berhasil

	Area	Skenario 2	Berhasil
		Skenario 3	Berhasil
		Skenario 4	Berhasil
UJ-UC-003	Uji Melawan Musuh	Skenario 1	Berhasil
		Skenario 2	Berhasil
UJ-UC-004	Uji Mengambil *Item*	Skenario 1	Berhasil
		Skenario 2	Berhasil
UJ-UC-005	Uji Menggunakan *Item*	Skenario 1	Berhasil
UJ-UC-006	Uji Menemukan Transporter	Skenario 1	Berhasil
		Skenario 2	Berhasil

5.3.2 Evaluasi Pengujian Pengguna

Evaluasi pengujian pengguna dapat dilihat pada Tabel 5.11. Hasil yang terlihat merupakan hasil yang didapat dari sistem penilaian rata-rata berdasarkan tiap poin indikator. Tiap poin indikator dapat bernilai mulai dari 1 hingga 5. Penjeasan dari nilai-nilai pada kuesioner tersebut dapat dilihat pada Tabel 5.11.

Tabel 5.11 Penjelasan Nilai pada Kuesioner

Nilai	Keterangan
1	Sangat Tidak Setuju
2	Tidak Setuju
3	Kurang Setuju
4	Cukup Setuju
5	Setuju
6	Sangat Setuju

Jawaban kuesioner dari tiap pengguna dapat dilihat pada lampiran. Rangkuman hasil uji coba pengguna sendiri dapat dilihat pada tabel 5.12.

Tabel 5.12 Rangkuman Hasil Formulir Pengujian Aplikasi

Penilaian	**Indikator**	**Penilaian Mayorias**
Kenyamanan Antarmuka dan Lingkungan	Kenyamanan antarmuka	(5+5+2+4+4+5) / 6 = 4,17
	Kenyamanan lingkungan	(4+6+2+4+3+5) / 6 = 4
	Kenyamaan *cardboard* dan controller	(4+6+3+4+4+5) / 6 = 4.33
Total Penilaian		(4.17 + 4 + 4.33) / 3 = 4.17
Kemudahan Permainan	Kemudahan memahami permainan	(4+6+3+4+3+5) / 6 = 4.17
	Kemudahan menjalankan tugas permainan	(3+6+4+5+5+5) / 6 = 4.67
Total Penilaian		(4,17 + 4,67) / 2 = 4.42
Suasana Permainan	Suasana karakter permainan	(5+6+4+5+6+5) / 6 = 5.17
Total Penilaian		5.17
Performa Permainan	Performa permainan	(4+5+2+4+4+5) / 6 = 4
	Performa kontrol permainan	(4+6+4+5+4+5) / 6 = 4.67
Total Penilaian		(4 + 4,67) / 2 = 4.335

Tabel 5.12 menunjukkan penilaian yang ada memiliki tingkat kepuasan cukup setuju dan setuju. Mengenai aspek

kenyamanan antarmuka dan lingkungan, pengguna merasa cukup setuju dengan tingkat kenyamanan saat ini (nilai rata-rata pengguna : 4,17). Untuk aspek kemudahan permaian, pengguna juga merasa cukup setuju dalam memahami permainan dan menjalankan tugas permaianan (nilai rata-rata pengguna : 4,42). Mengenai aspek suasana permainan sendiri, pengguna merasa setuju sudah merasa menjadi karakter utama dalam permainan (nilai rata-rata pengguna : 5,17). Terakhir mengenai aspek performa permainan, pengguna merasa cukup setuju dengan tingkat performa permainan dan control permainan saat ini (nilai rata-rata pengguna : 4,335).

Kesimpulan dari evaluasi ini adalah pengguna merasa cukup setuju dan setuju dalam hal kenyamanan antarmuka dan lingkungan, kemudahan permainan, suasana permainan, serta performa permainan. Meskipun begitu, pengguna merasa masih terdapat fitur yang perlu ditingkatkan untuk kemudahan dan performa permainan. Untuk rangkuman kritik dan saran pengguna dapat dilihat pada Tabel 5.13.

Tabel 5.13 Rangkuman Kritik dan Saran Pengguna

Nama	Kritik dan Saran
Admiral Budi Arviansyah	Permainan sudah bagus dan menarik, tampilan musuh laba-laba seperti kalajengking
Anugerah DP	- Penjelasan dan tampilan lobby mungkin bisa diperbaiki agar lebih mudah dipahami - Awal permainan pemain bisa jatuh ke laut, area mungkin bisa dibatasi - Area mungkin bisa dibuat lebih ramai
Aliya Fathma N	- Asset mungkin bisa lebih bisa ditambah - Instruksi lebih sederhana - Penggunaan asset mungkin bisa lebih dirapikan

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dijelaskan mengenai kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga diberi saran untuk pengembangan aplikasi kedepannya.

6.1 Kesimpulan

Dari hasil pengamatan selama proses pengerjaan mulai dari proses perancangan, implementasi, dan pengujian yang telah dilakukan terhadap aplikasi, diambil kesimpulan sebagai berikut :

1. Berdasarkan hasil uji coba fungsionalitas yang ditunjukkan pada table 5.2.1, aturan main dan skenario permainan berhasil diimplementasikan.
2. Berdasarkan sub-bab 4.3.2 (Implementasi Finite State Machine Karakter Musuh), *finite state machine* pada musuh berhasil diimplementasikan.
3. Berdasarkan hasil uji coba fungsionalitas yang ditunjukkan pada tabel 5.2.1, aplikasi berhasil dibangun sesuai rancangan pada *Unity3D engine*.
4. Berdasarkan hasil uji coba fungsionalitas dan non-fungsionalitas terhadap pengguna yang ditunjukkan pada Tabel 5.3, aplikasi berhasil dibangun baik secara *single-player* maupun *multiplayer*.

6.2 Saran

Adapun saran yang diberikan untuk mengembangkan *game* ini adalah *game* ini masih memerlukan pengembangan lebih lanjut, khususnya dalam hal segi tampilan, optimisasi permainan

dan *multiplayer*. Untuk pengembangan selanjutnya, disarankan untuk menggunakan versi Unity terbaru karena :

- Menyediakan jauh lebih banyak *asset* yang *mobile-friendly*. Dengan menggunakan semua *asset* yang bersifat *mobile-friendly*, penggunaan *asset* dalam jumlah banyak digunakan tidak membuat permainan *lagging*
- Menyediakan alokasi *bandwidth* yang lebih besar. Alokasi *bandwidth* yang lebih besar meminimalisir terjadinya Server Timeout.

DAFTAR PUSTAKA

[1] "VR Game List," VR Games For. [Daring]. Tersedia pada: http://vrgamesfor.com/list/. [Diakses: 13 Desember 2016].

[2] C. Chesher, "Colonizing Virtual Reality," 1992-1984. [Daring]. Tersedia pada : http://cultronix.eserver.org/chesher/?utm_source=friendfeedlikes&utm_medium=twitter. [Diakses: 13 Desember 2016].

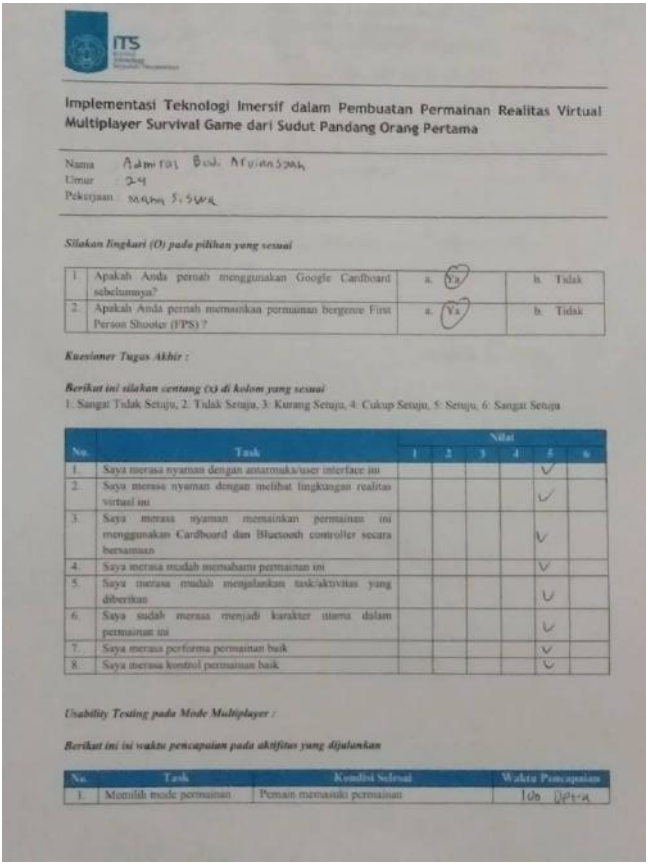
[3] "Google Cardboard Definition from PC Magazine Encyclopedia." [Daring]. Tersedia pada: http://www.pcmag.com/encyclopedia/term/67932/googl e-cardboard. [Diakses: 13 Desember 2016].

[4] "Introduction to Unity3D," Code Envato Tuts+. [Daring]. Tersedia pada https://code.tutsplus.com/tutorials/introduction-to-unity3d--mobile-10752. [Diakses: 13 Desember 2016].

[5] "Finite State Machine," [Daring]. Tersedia pada: http://blog.renatopp.com/2015/03/29/finite-state-machines-in-javascript/machines-in-javascript/. [Diakses 12 Desember 2017]

(Halaman ini sengaja dikosongkan)

LAMPIRAN
HASIL KUESIONER



Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game dari Sudut Pandang Orang Pertama

Nama : Admiral Budi Arviansyah
Umur : 24
Pekerjaan : Mahasiswa

Silakan lingkari (O) pada pilihan yang sesuai

1.	Apakah Anda pernah menggunakan Google Cardboard sebelumnya?	a. Ya	b. Tidak
2.	Apakah Anda pernah memainkan permainan bergenre First Person Shooter (FPS) ?	a. Ya	b. Tidak

Kuesioner Tugas Akhir :

Berikut ini silakan centang (x) di kolom yang sesuai
1. Sangat Tidak Setuju, 2. Tidak Setuju, 3. Kurang Setuju, 4. Cukup Setuju, 5. Setuju, 6. Sangat Setuju

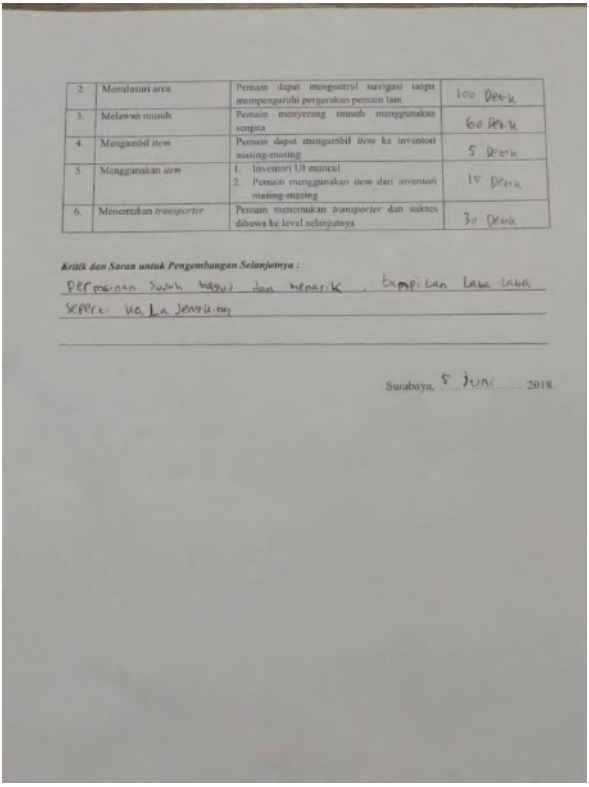
No.	Task	Nilai					
		1	2	3	4	5	6
1.	Saya merasa nyaman dengan antarmuka/user interface ini					✓	
2.	Saya merasa nyaman dengan melibat lingkungan realitas virtual ini					✓	
3.	Saya merasa nyaman memainkan permainan ini menggunakan Cardboard dan Bluetooth controller secara bersamaan					✓	
4.	Saya merasa mudah memahami permainan ini					✓	
5.	Saya merasa mudah mengalankan task/aktivitas yang diberikan					✓	
6.	Saya sudah merasa menjadi karakter utama dalam permainan ini					✓	
7.	Saya merasa performa permainan baik					✓	
8.	Saya merasa kontrol permainan baik					✓	

Usability Testing pada Mode Multiplayer :

Berikut ini isi waktu pencapaian pada aktifitas yang dijalankan

No.	Task	Kondisi Selesai	Waktu Pencapaian
1.	Memilih mode permainan	Pemain memasuki permainan	1ob 10Pts

Gambar 8.1 Lembar Kuesioner Admiral Budi Arviansyah (1)



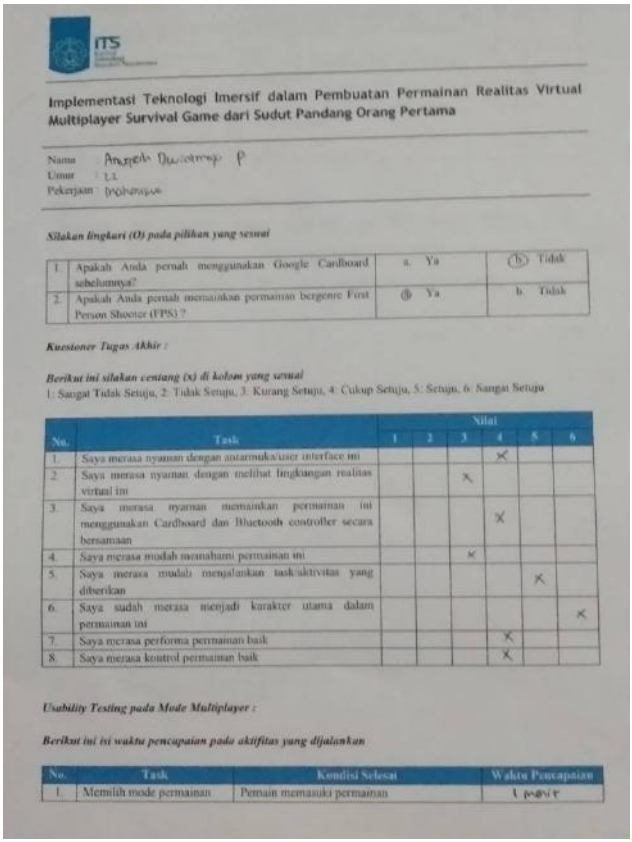
2.	Menalusuri area	Pemain dapat mengontrol navigasi tanpa mempengaruhi pergerakan pemain lain	100 Detik
3.	Melawan musuh	Pemain menyerang musuh menggunakan senjata	60 Detik
4.	Mengambil *item*	Pemain dapat mengambil *item* ke inventori masing-masing	5 Detik
5.	Menggunakan *item*	1. Inventori UI muncul 2. Pemain menggunakan *item* dari inventori masing-masing	15 Detik
6.	Menemukan *transporter*	Pemain menemukan *transporter* dan sukses dibawa ke level selanjutnya	30 Detik

Kritik dan Saran untuk Pengembangan Selanjutnya :

Permainan sudah bagus dan menarik, tampilan lava-lava seperti ada jenglotnya

Surabaya, 5 Juni 2018

Gambar 8.2 Lembar Kuesioner Admiral Budi Arviansyah (2)



Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game dari Sudut Pandang Orang Pertama

Nama : Anugerah Dwiatmojo P
Umur : 23
Pekerjaan : Mahasiswa

Silakan lingkari (O) pada pilihan yang sesuai

1.	Apakah Anda pernah menggunakan Google Cardboard sebelumnya?	a. Ya	(b) Tidak
2.	Apakah Anda pernah memainkan permainan bergenre First Person Shooter (FPS) ?	(a) Ya	b. Tidak

Kuesioner Tugas Akhir :

Berikut ini silakan centang (x) di kolom yang sesuai
1: Sangat Tidak Setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4: Cukup Setuju, 5: Setuju, 6: Sangat Setuju

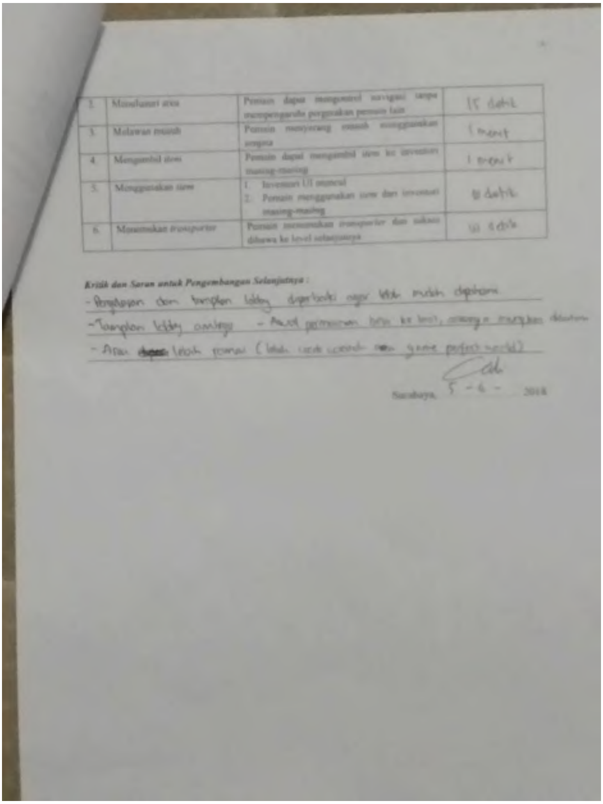
No.	Task	Nilai					
		1	2	3	4	5	6
1.	Saya merasa nyaman dengan antarmuka/user interface ini				X		
2.	Saya merasa nyaman dengan melihat lingkungan realitas virtual ini			X			
3.	Saya merasa nyaman memainkan permainan ini menggunakan Cardboard dan Bluetooth controller secara bersamaan				X		
4.	Saya merasa mudah memahami permainan ini			X			
5.	Saya merasa mudah menjalankan task/aktivitas yang diberikan					X	
6.	Saya sudah merasa menjadi karakter utama dalam permainan ini						X
7.	Saya merasa performa permainan baik				X		
8.	Saya merasa kontrol permainan baik				X		

Usability Testing pada Mode Multiplayer :

Berikut ini isi waktu pencapaian pada aktifitas yang dijalankan

No.	Task	Kondisi Selesai	Waktu Pencapaian
1.	Memilih mode permainan	Pemain memasuki permainan	1 menit

Gambar 8.3 Lembar Kuesioner Anugerah Dwiatmojo P (1)



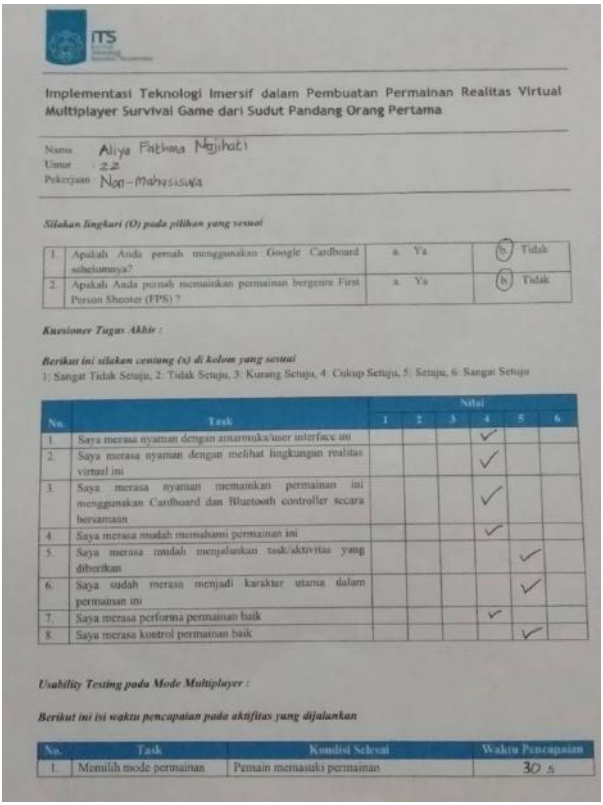
2.	Menelusuri area	Pemain dapat mengontrol navigasi tanpa mempengaruhi pergerakan pemain lain	15 detik
3.	Melawan musuh	Pemain menyerang musuh menggunakan senjata	1 menit
4.	Mengambil item	Pemain dapat mengambil item ke inventori masing-masing	1 menit
5.	Menggunakan item	1. Inventori UI muncul 2. Pemain menggunakan item dari inventori masing-masing	[illegible] detik
6.	Menemukan *transporter*	Pemain menemukan *transporter* dan sukses dibawa ke level selanjutnya	[illegible] detik

Kritik dan Saran untuk Pengembangan Selanjutnya :

- Penjelasan dan tampilan lobby diperbaiki agar lebih mudah dipahami
- Tampilan lobby ambigu - Awal permainan bisa ke level, [illegible]
- Arah lebih [illegible] (lebih cocok contoh game perfect world)

Surabaya, 5 - 6 - 2018

Gambar 8.4 Lembar Kuesioner Anugerah Dwiatmojo P (2)



ITS

Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game dari Sudut Pandang Orang Pertama

Nama : Aliya Fathma Najihati
Umur : 22
Pekerjaan : Non-Mahasiswa

Silahkan lingkari (O) pada pilihan yang sesuai

1.	Apakah Anda pernah menggunakan Google Cardboard sebelumnya?	a. Ya	(b) Tidak
2.	Apakah Anda pernah memainkan permainan bergenre First Person Shooter (FPS) ?	a. Ya	(b) Tidak

Kuesioner Tugas Akhir :

Berikut ini silakan centang (x) di kolom yang sesuai
1: Sangat Tidak Setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4: Cukup Setuju, 5: Setuju, 6: Sangat Setuju

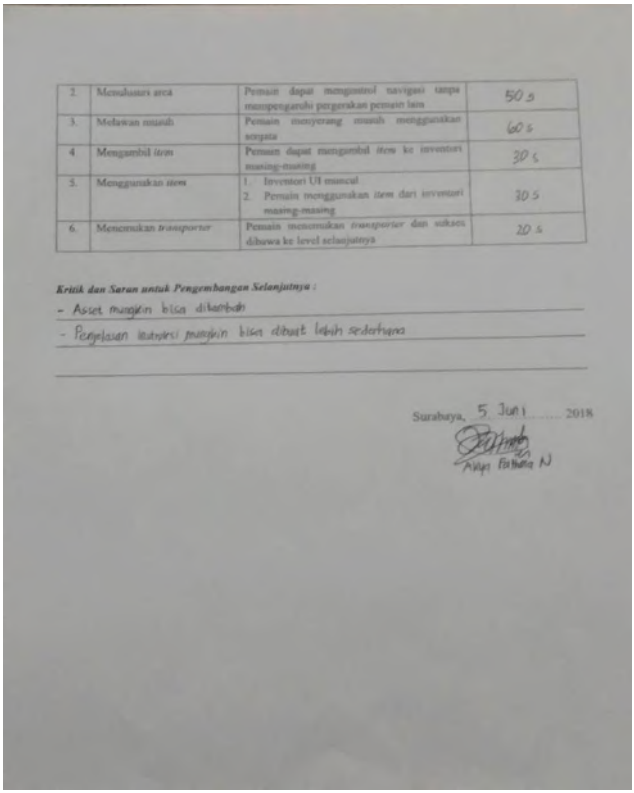
No.	Task	Nilai					
		1	2	3	4	5	6
1.	Saya merasa nyaman dengan antarmuka/user interface ini				✓		
2.	Saya merasa nyaman dengan melihat lingkungan realitas virtual ini				✓		
3.	Saya merasa nyaman memainkan permainan ini menggunakan Cardboard dan Bluetooth controller secara bersamaan				✓		
4.	Saya merasa mudah memahami permainan ini				✓		
5.	Saya merasa mudah menjalankan task/aktivitas yang diberikan					✓	
6.	Saya sudah merasa menjadi karakter utama dalam permainan ini					✓	
7.	Saya merasa performa permainan baik				✓		
8.	Saya merasa kontrol permainan baik					✓	

Usability Testing pada Mode Multiplayer :

Berikut ini isi waktu pencapaian pada aktifitas yang dijalankan

No.	Task	Kondisi Selesai	Waktu Pencapaian
1.	Memilih mode permainan	Pemain memasuki permainan	30 s

Gambar 8.5 Lembar Kuesioner Aliya Fathma Najihati (1)



2.	Menelusuri area	Pemain dapat mengontrol navigasi tanpa mempengaruhi pergerakan pemain lain	50 s
3.	Melawan musuh	Pemain menyerang musuh menggunakan senjata	60 s
4	Mengambil *item*	Pemain dapat mengambil *item* ke inventori masing-masing	30 s
5.	Menggunakan *item*	1. Inventori UI muncul 2. Pemain menggunakan *item* dari inventori masing-masing	30 s
6.	Menemukan *transporter*	Pemain menemukan *transporter* dan sukses dibawa ke level selanjutnya	20 s

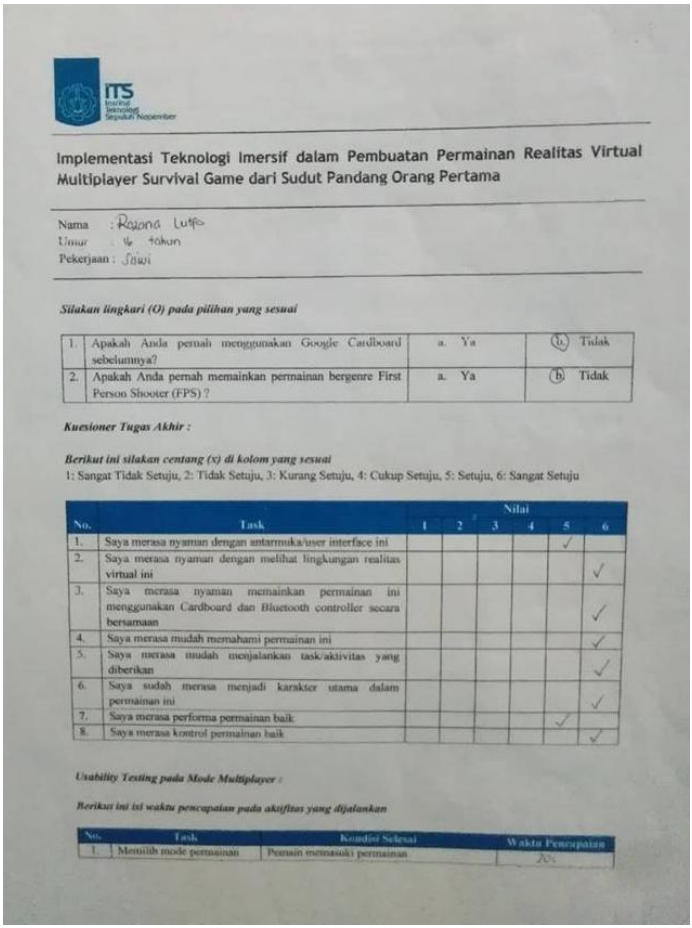
Kritik dan Saran untuk Pengembangan Selanjutnya :

- Asset mungkin bisa ditambah
- Penjelasan instruksi mungkin bisa dibuat lebih sederhana

Surabaya, 5 Juni 2018

Aliya Fathma N

Gambar 8.6 Lembar Kuesioner Aliya Fathma Najihati (2)



Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game dari Sudut Pandang Orang Pertama

Nama : Rozana Lutfa
Umur : 16 tahun
Pekerjaan : Siswi

Silakan lingkari (O) pada pilihan yang sesuai

1.	Apakah Anda pernah menggunakan Google Cardboard sebelumnya?	a. Ya	(b.) Tidak
2.	Apakah Anda pernah memainkan permainan bergenre First Person Shooter (FPS) ?	a. Ya	(b) Tidak

Kuesioner Tugas Akhir :

Berikut ini silakan centang (x) di kolom yang sesuai
1: Sangat Tidak Setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4: Cukup Setuju, 5: Setuju, 6: Sangat Setuju

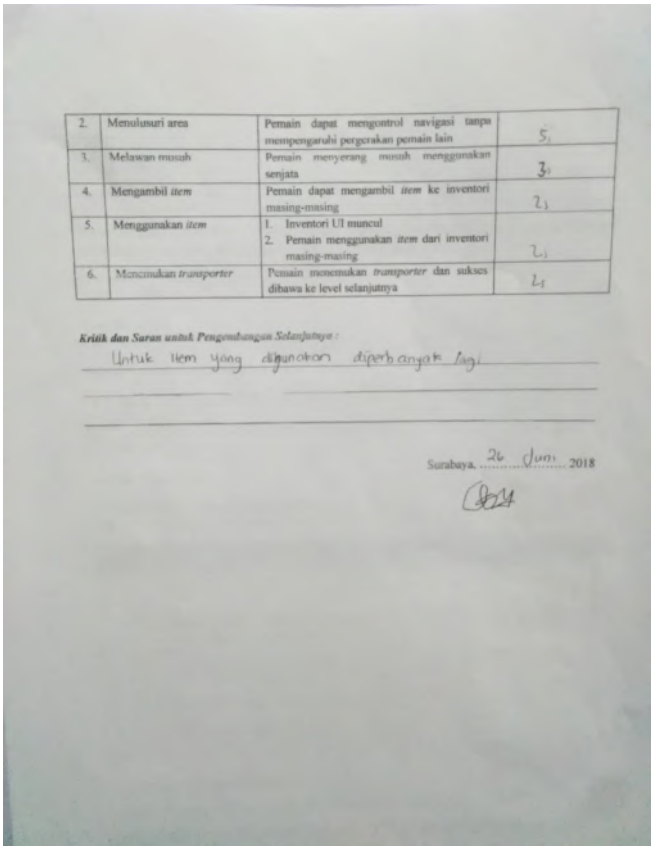
No.	Task	Nilai					
		1	2	3	4	5	6
1.	Saya merasa nyaman dengan antarmuka/user interface ini					✓	
2.	Saya merasa nyaman dengan melihat lingkungan realitas virtual ini						✓
3.	Saya merasa nyaman memainkan permainan ini menggunakan Cardboard dan Bluetooth controller secara bersamaan						✓
4.	Saya merasa mudah memahami permainan ini						✓
5.	Saya merasa mudah menjalankan task/aktivitas yang diberikan						✓
6.	Saya sudah merasa menjadi karakter utama dalam permainan ini						✓
7.	Saya merasa performa permainan baik					✓	
8.	Saya merasa kontrol permainan baik						✓

Usability Testing pada Mode Multiplayer :

Berikut ini isi waktu pencapaian pada aktifitas yang dijalankan

No.	Task	Kondisi Selesai	Waktu Pencapaian
1.	Memilih mode permainan	Pemain memasuki permainan	20s

Gambar 8.7 Lembar Kuesioner Rozana Lutfa (1)



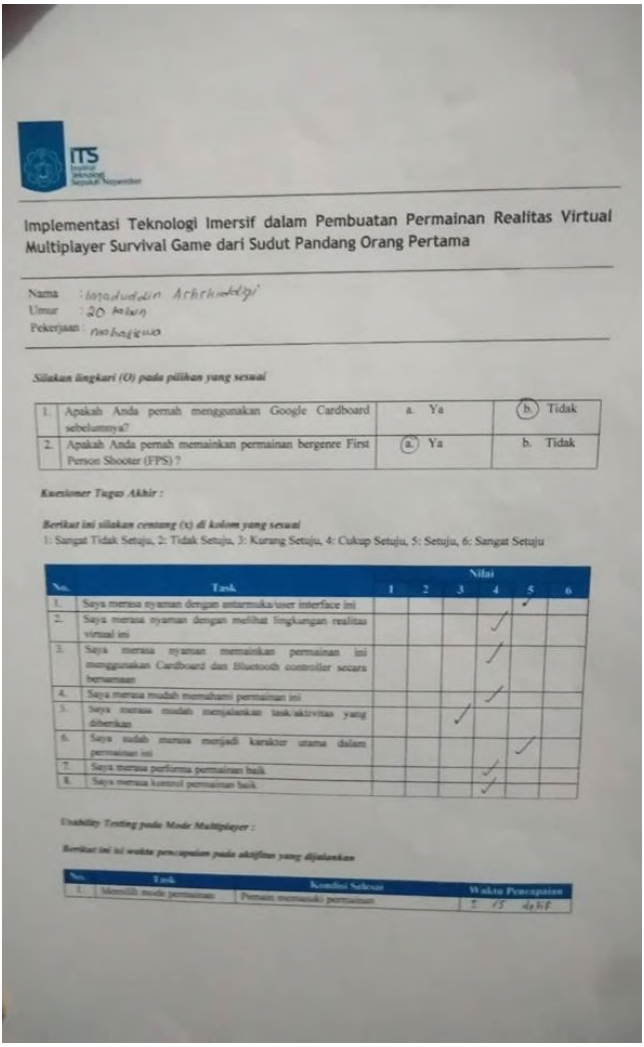
2.	Menulusuri area	Pemain dapat mengontrol navigasi tanpa mempengaruhi pergerakan pemain lain	5
3.	Melawan musuh	Pemain menyerang musuh menggunakan senjata	3
4.	Mengambil *item*	Pemain dapat mengambil *item* ke inventori masing-masing	2
5.	Menggunakan *item*	1. Inventori UI muncul 2. Pemain menggunakan *item* dari inventori masing-masing	2
6.	Menemukan *transporter*	Pemain menemukan *transporter* dan sukses dibawa ke level selanjutnya	4

Kritik dan Saran untuk Pengembangan Selanjutnya :

Untuk item yang digunakan diperbanyak lagi

Surabaya, 26 Juni 2018

Gambar 8.8 Lembar Kuesioner Rozana Lutfa (2)



Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game dari Sudut Pandang Orang Pertama

Nama : Imaduddin Ashshiddiqi
Umur : 20 tahun
Pekerjaan : mahasiswa

Silakan lingkari (O) pada pilihan yang sesuai

1.	Apakah Anda pernah menggunakan Google Cardboard sebelumnya?	a. Ya	(b.) Tidak
2.	Apakah Anda pernah memainkan permainan bergenre First Person Shooter (FPS) ?	(a.) Ya	b. Tidak

Kuesioner Tugas Akhir :

Berikut ini silakan centang (x) di kolom yang sesuai
1: Sangat Tidak Setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4: Cukup Setuju, 5: Setuju, 6: Sangat Setuju

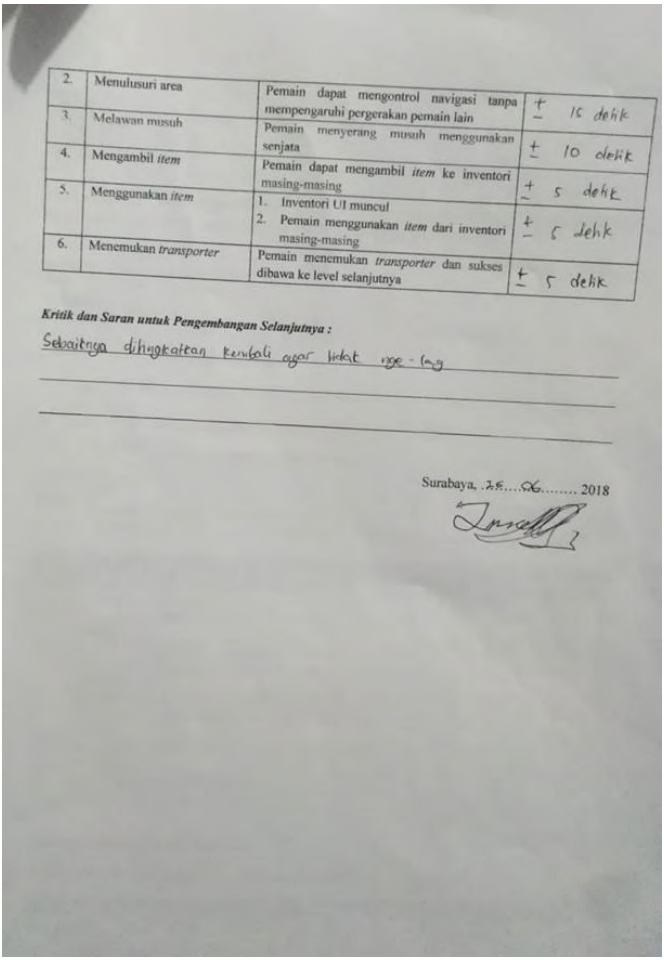
No.	Task	Nilai					
		1	2	3	4	5	6
1.	Saya merasa nyaman dengan antarmuka/user interface ini					✓	
2.	Saya merasa nyaman dengan melihat lingkungan realitas virtual ini				✓		
3.	Saya merasa nyaman memainkan permainan ini menggunakan Cardboard dan Bluetooth controller secara bersamaan				✓		
4.	Saya merasa mudah memahami permainan ini				✓		
5.	Saya merasa mudah menjalankan task aktivitas yang diberikan			✓			
6.	Saya sudah merasa menjadi karakter utama dalam permainan ini					✓	
7.	Saya merasa performa permainan baik				✓		
8.	Saya merasa kontrol permainan baik				✓		

Usability Testing pada Mode Multiplayer :

Berikut ini isi waktu persepsian pada aktifitas yang dijalankan

No.	Task	Kondisi Selesai	Waktu Persepsian
1.	Memilih mode permainan	Pemain memasuki permainan	5 15 detik

Gambar 8.9 Lembar Kuesioner Imaduddin Ashiddiqi (1)



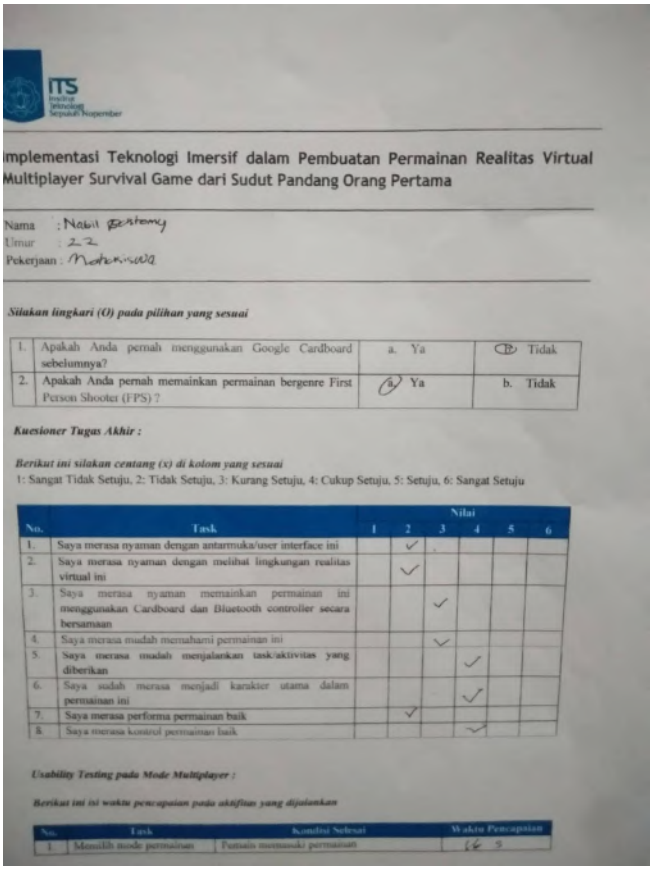
2.	Menulusuri area	Pemain dapat mengontrol navigasi tanpa mempengaruhi pergerakan pemain lain	± 15 detik
3.	Melawan musuh	Pemain menyerang musuh menggunakan senjata	± 10 detik
4.	Mengambil *item*	Pemain dapat mengambil *item* ke inventori masing-masing	± 5 detik
5.	Menggunakan *item*	1. Inventori UI muncul 2. Pemain menggunakan *item* dari inventori masing-masing	± 5 detik
6.	Menemukan *transporter*	Pemain menemukan *transporter* dan sukses dibawa ke level selanjutnya	± 5 detik

Kritik dan Saran untuk Pengembangan Selanjutnya :

Sebaiknya ditingkatkan kembali agar lebih ngga - lag

Surabaya, 26-06-2018

Gambar 8.10 Lembar Kuesioner Imaduddin Ashsiddiqi (2)



ITS
Institut Teknologi Sepuluh Nopember

Implementasi Teknologi Imersif dalam Pembuatan Permainan Realitas Virtual Multiplayer Survival Game dari Sudut Pandang Orang Pertama

Nama : Nabil Bastomy
Umur : 22
Pekerjaan : Mahasiswa

Silakan lingkari (O) pada pilihan yang sesuai

1.	Apakah Anda pernah menggunakan Google Cardboard sebelumnya?	a. Ya	(b) Tidak
2.	Apakah Anda pernah memainkan permainan bergenre First Person Shooter (FPS) ?	(a) Ya	b. Tidak

Kuesioner Tugas Akhir :

Berikut ini silakan centang (x) di kolom yang sesuai
1: Sangat Tidak Setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4: Cukup Setuju, 5: Setuju, 6: Sangat Setuju

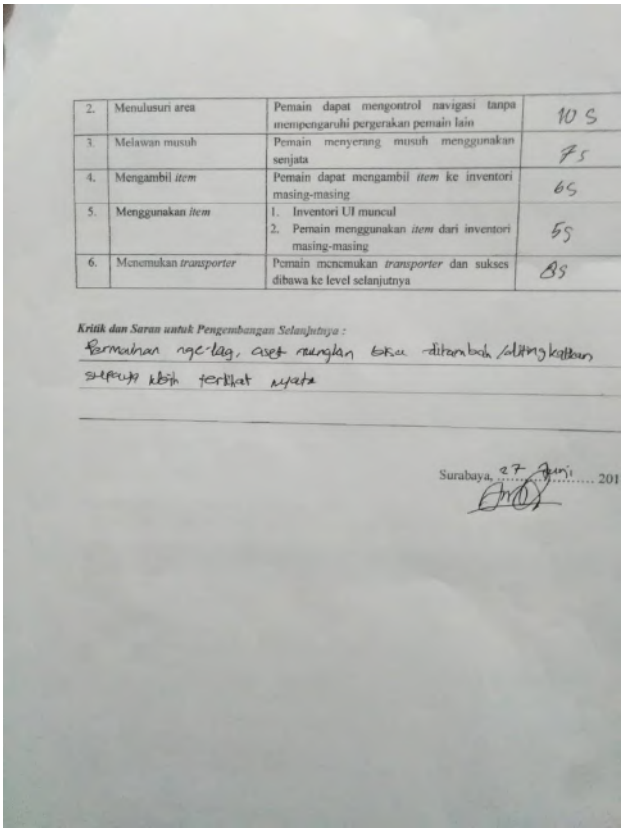
No.	Task	Nilai 1	2	3	4	5	6
1.	Saya merasa nyaman dengan antarmuka/user interface ini		✓				
2.	Saya merasa nyaman dengan melihat lingkungan realitas virtual ini		✓				
3.	Saya merasa nyaman memainkan permainan ini menggunakan Cardboard dan Bluetooth controller secara bersamaan			✓			
4.	Saya merasa mudah memahami permainan ini			✓			
5.	Saya merasa mudah menjalankan task/aktivitas yang diberikan				✓		
6.	Saya sudah merasa menjadi karakter utama dalam permainan ini				✓		
7.	Saya merasa performa permainan baik		✓				
8.	Saya merasa kontrol permainan baik				✓		

Usability Testing pada Mode Multiplayer :

Berikut ini isi waktu pencapaian pada aktifitas yang dijalankan

No.	Task	Kondisi Selesai	Waktu Pencapaian
1.	Memilih mode permainan	Pemain memasuki permainan	56 s

Gambar 8.11 Lembar Kuesioner Nabil Bastomy (1)



2.	Menulusuri area	Pemain dapat mengontrol navigasi tanpa mempengaruhi pergerakan pemain lain	10 S
3.	Melawan musuh	Pemain menyerang musuh menggunakan senjata	7 S
4.	Mengambil *item*	Pemain dapat mengambil *item* ke inventori masing-masing	6 S
5.	Menggunakan *item*	1. Inventori UI muncul 2. Pemain menggunakan *item* dari inventori masing-masing	5 S
6.	Menemukan *transporter*	Pemain menemukan *transporter* dan sukses dibawa ke level selanjutnya	8 S

Kritik dan Saran untuk Pengembangan Selanjutnya :

Permainan ngelag, aset ruangan bisa ditambah/diimprove supaya lebih terlihat nyata

Surabaya, 27 Juni 201

Gambar 8.12 Lembar Kuesioner Nabil Bastomy (2)

BIODATA PENULIS

Aliya Rahma Najihati, lahir pada tanggal 28 Agustus 1995 di Cirebon, Jawa Barat. Hobi yang dimiliki adalah bermain game, membaca, menulis, dan menonton film. Penulis menempuh pendidikan mulai dari SDN 3 Wonogiri (2001-2007), SMPN 1 Wonogiri (2007-2010), SMAN 3 Solo (2010-2013), dan Teknik Informatika ITS (2013-2018). Di jurusan Teknik Informatika ITS, penulis mengambil bidang minat Interaksi Grafika dan Seni (IGS) dan memiliki ketertarikan dalam eksplorasi teknologi di bidang *game*. Selama perkuliahan, penulis aktif dalam organisasi kemahasiswaan, antara lain staff departemen Hubungan Luar Himpunan Mahasiswa Teknik Computer-Informatika 2014, Staff Ahli Himpunan Mahasiswa Teknik Computer-Informatika 2015, Vice-Manager departemen External Affairs Badan Eksekutif Mahasiswa FTIF 2015, International Relation Manager departemen Outgoing Global Internship AIESEC Surabaya, Anggota National Seminar of Technology Schematics 2014, Anggota Sie Hubungan Masyarakat 2015, Penerjemah LPPM ITS. Penulis dapat dihubungi melalui surel : aliya.rahma13@gmail.com.