TUGAS AKHIR - KI141502

3D VIRTUAL PET GAME MULTIPLAYER DENGAN MENGGUNAKAN GOOGLE CARDBOARD

DIMAS RISKAHADI
NRP 5112100134

Dosen Pembimbing
Darlis Herumurti, S.Kom., M.Kom.
Imam Kuswardayan, S.Kom., M.T.

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

TUGAS AKHIR – KI141502

3D VIRTUAL PET GAME MULTIPLAYER DENGAN MENGGUNAKAN GOOGLE CARDBOARD

DIMAS RISKAHADI
NRP 5112100134

Dosen Pembimbing
Darlis Herumurti, S.Kom., M.Kom.
Imam Kuswardayan, S.Kom., M.T.

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

(Halaman ini sengaja dikosongkan)

FINAL PROJECT- KI141502

3D VIRTUAL PET MULTIPLAYER GAME WITH USING GOOGLE CARDBOARD

DIMAS RISKAHADI
NRP 5112100134

Advisor
Darlis Herumurti, S.Kom., M.Kom.
Imam Kuswardayan, S.Kom., M.T.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

3D VIRTUAL PET GAME MULTIPLAYER DENGAN 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 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
DIMAS RISKAHADI
NRP. 5112 100 134

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Darlis Herumurti, S.Kom., M.Kom ..... (pembimbing 1)
NIP: 19771217 200312 1 001

Imam Kuswardayan, S.Kom., M.T. ..... (pembimbing 2)
NIP: 19761215 200312 1 001

SURABAYA
JULI, 2016

(Halaman ini sengaja dikosongkan)

3D VIRTUAL PET MULTIPLAYER GAME DENGAN MENGGUNAKAN GOOGLE CARDBOARD

Nama Mahasiswa : Dimas Riskahadi
NRP : 5112 100 134
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Darlis Herumurti, S.Kom., M.Kom.
Dosen Pembimbing II : Imam Kuswardayan, S.Kom., M.T.

ABSTRAK

Memelihara hewan adalah aktivitas yang menyenangkan bagi anak-anak dan dewasa. Akan tetapi, terdapat kondisi dimana masyarakat tidak bisa memelihara hewan tersebut. Sehingga beralih menuju hewan peliharaan virtual yang lebih praktis. Peliharaan virtual ini dapat ditemukan pada game simulasi peliharaan. Dengan game ini, sisi menyenangkan dan manfaat memelihara peliharaan di dunia nyata tetap dirasakan game ini.

Saat ini, perkembangan game sangatlah pesat di sisi teknologi permainan dan inovasinya. Game simulasi perlu ada perubahan agar pemain menjadi lebih terhibur. Salah satunya pengembangan fitur game yang populer saat ini yaitu multiplayer. Dimana game dapat dimainkan lebih dari satu orang Selain itu, untuk menambah suasana dalam bermain game dapat menggunakan teknologi realitas virtual.

Tujuan utama dari tugas akhir ini adalah membuat permainan virtual pet game dengan menggunakan fitur multiplayer dan virtual reality yang menggunakan google cardboard. Metode membangun game simulasi peliharaan ini, dengan membuat rancangan menggunakan diagram finite state machine untuk menggambarkan aturan permainan peliharaan. Game ini akan ditambahkan pertarungan yang memiliki fitur multiplayer dengan menggunakan plugin photon dan dapat ditampilkan dalam realitas virtual menggunakan alat google

cardboard. Dari evaluasi game oleh pengguna, diharapkan inovasi dari game ini layak untuk dikembangkan.

Kata kunci**: **Cardboard, Game, Multiplayer, Simulasi, Virtual Reality (VR).

3D VIRTUAL PET MULTIPLAYER WITH USING GOOGLE CARDBOARD

Student Name	: Dimas Riskahadi
NRP	: 5112 100 134
Major	: Teknik Informatika FTIf-ITS
Advisor I	: Darlis Herumurti, S.Kom., M.Kom.
Advisor II	: Imam Kuswardayan, S.Kom., M.T.

ABSTRACT

Raising animals is a fun activity for kids and adults. However, there is a condition where people could not care for the animal. So, switching to the virtual pet more practical. This virtual pet can be found on pets simulation game. With this game, the fun and benefits of raising a pet in the real world still felt same.

Currently, game development is very rapid in the technology game and innovation. Game simulations need to improve so that the player becomes more entertained. One's popular feature game today is multiplayer. Where the game can be played more than one person. In addition, to increase circumstance in playing games can use virtual reality technology.

The main objective of this final project is to create a virtual pet game features multiplayer game using virtual reality and using google cardboard. Methods of building this pet simulation game, making the design using finite state machine diagram to describe the rules of the game allowed. The game will be added to feature multiplayer battles using photon plugin and can be displayed in virtual reality using google tool cardboard. From the evaluation of the game by the user, the expected innovations of this game to develop.

Keywords**: **Cardboard, Game, Multiplayer, Simulation, Virtual Reality (VR).

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur kehadirat Allah Subhanahu wa Ta'ala yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "3D VIRTUAL PET GAME MULTIPLAYER DENGAN GOOGLE CARDBOARD ".

Pengerjaan Tugas Akhir ini adalah momen bagi penulis untuk mengeluarkan seluruh kemampuan, hasrat, dan keinginan penulis sejak pertama kali memasuki dunia kuliah di kampus Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini, penulis mendapatkan banyak bantuan dari berbagai pihak. Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih secara khusus kepada:

1. Allah SWT atas limpahan rahmat dan rezeki-Nya sehingga penulis dapat menyelesaikan Tugas Akhir.
2. Ibu penulis, Iskandar Ningsih, yang selalu memberikan dukungan, doa, perhatian, kasih sayang, serta modal finansial dalam menjalani kuliah.
3. Kakak penulis, Nana Riskahadi, Wulan Riskahadi dan Doby Hendriawan, yang telah membantu dalam hal akomodasi dan memberi semangat untuk menyelesaikan studi penulis.
4. Bapak Darlis Herumurti selaku dosen pembimbing Tugas Akhir pertama dan yang telah memberikan arahan dalam pengerjaan Tugas Akhir ini.
5. Bapak Imam Kuswardaryan selaku dosen pembimbing Tugas Akhir kedua yang dengan sabar membimbing penulis dalam pengerjaan Tugas Akhir ini.
6. Ibu Wijayanti Nurul Khotimah selaku dosen wali yang telah berkenan memberikan saran dan arahan kepada penulis selama menjalani studi.

7. Bapak Radityo Anggoro selaku dosen koordinator Tugas Akhir yang telah membantu penulis atas segala sesuatu yang berkenaan dengan syarat-syarat dan terlaksananya sidang Tugas Akhir.
8. Bapak Darlis Herumurti selaku Ketua Jurusan Teknik Informatika ITS yang selama ini memberikan bantuan kepada penulis baik dalam hal kuliah maupun kegiatan-kegiatan lainnya.
9. Dosen-dosen Teknik Informatika yang dengan sabar mendidik dan memberikan pengalaman baru kepada penulis selama berkuliah di Teknik Informatika.
10. Staf TU Teknik Informatika ITS yang senantiasa memudahkan segala urusan penulis di jurusan.
11. Rekan-rekan dan pengelola Laboratorium Interaksi, Grafika, dan Seni yang telah memberikan fasilitas serta kesempatan melakukan riset atas Tugas Akhir yang dikerjakan penulis.
12. Rekan-rekan dan sahabat-sahabat penulis angkatan 2012 yang telah memberikan dorongan motivasi dan bantuan kepada penulis.
13. Rekan-rekan penulis pada Rumpun Mata Kuliah Interaksi, Grafika, dan Seni yang selalu bahu membahu dalam hal kesuksesan bersama.
14. Pihak-pihak lain yang tidak sengaja terlewat dan tidak dapat penulis sebutkan satu per satu.

Penulis telah berusaha sebaik mungkin dalam menyusun Tugas Akhir ini, namun penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian dalam pengerjaannya. Kritik dan saran yang membangun dapat disampaikan kepada penulis sebagai bahan perbaikan selanjutnya.

Surabaya, 13 Juli 2016
Penulis

Dimas Riska Hadi

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

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

1.1 Latar Belakang

Game adalah perangkat lunak yang bertujuan memberikan hiburan bagi penggunanya. Pengguna *game* saat ini sudah mencapai seluruh kalangan masyarakat dari anak-anak hingga dewasa, baik dari kalangan masyarakat ekonomi bawah hingga atas. *Game* favorit masyarakat seperti *Counter Strike Online* untuk versi desktop dan *Clash of Clans* (CoC) untuk versi mobile adalah beberapa *game* tersebut yang memiliki fitur *multiplayer* [1]. *Multiplayer* adalah jenis *game* yang dapat dimainkan oleh banyak orang. Dengan fitu ini dapat menambah nuansa kompetisi antar pemain dan dapat menjadi sarana komunikasi. Sehingga *game multiplayer* lebih unggul daripada *game singleplayer*.

Game simulasi adalah *game* yang permainannya menyerupai aktivitas seperti di dunia nyata. *Game* ini memiliki *gameplay* yang mudah dimengerti karena terdapat instruksi yang terstruktur. Salah satu jenis *game* simulasi adalah simulasi peliharaan. Simulasi peliharaan menampilkan karakter peliharaan seperti hewan atau alien yang lucu. *Game* ini dibentuk disebabkan oleh permalahan dari masyarakat yang sulit untuk memelihara peliharaan, baik masalah finansial, maupun lingkungan tinggal. Dengan *game* ini, pengguna dapat mendapatkan pengalaman memelihara hewan peliharaan yang sama seperti dunia nyata hanya dengan *smartphone*. Salah satu *game* yaitu Pou, *game* memelihara alien [2]. Pemain akan diberikan intruksi untuk merawat peliharaan agar tetap hidup dan tumbuh dengan memberi makan dan aktivitas memelihara peliharaan lainnya.

Dunia teknologi game saat ini semakin berkembang, yang meningkatkan interaksi antar penggunanya. Dimana dahulu

diperkenalkan teknologi 2D, lalu 3D dan sekarang *virtual reality* (VR) yang sedang muncul. Kelebihan dari virtual reality ini adalah pengguna dapat merasakan sensasi dunia nyata dalam dunia maya [3]. Seperti melihat bintang, melihat panorama alam dan sebagainya sehingga membuat pengguna benar berada pada dunia tersebut secara kasat mata. Teknologi yang digunakan untuk *virtual reality* ini diantaranya adalah *oculust Rift*, *Hololens* dan *google cardboard*. Pengguna akan menikmati hasil dunia virtual ini melalui teknologi ini, jika *oculus* menggunakan komputer, maka *google cardboard* dengan *handphone* dengan spesifikasi tertentu.

Dari beberapa referensi yang telah disampaikan, maka dapat dimunculkan inovasi baru berupa *game* yang *multiplayer* dengan menggunakan *virtual reality*. Pengguna dapat melakukan pemeliharaan dengan hanya menggunakan *smartphone* dan melakukan pertarungan ditambah dengan *google cardboard*. Tugas akhir ini dinamakan *3D virtual pet game multiplayer* dengan menggunakan *google cardboard*. Diharapkan dari hasil akhir ini dapat menjadi tahap awal perkembangan *3D game* dengan menggunakan *virtual reality* yang dapat dimainkan secara bersama-sama.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimanakah cara untuk membuat simulasi *virtual pet*?
2. Bagaimanakah membuat *virtual pet game* menjadi *game multiplayer*?
3. Bagaimanakah cara untuk menerapkan *virtual pet* dengan realitas virtual?
4. Bagaimanakah aturan main yang digunakan dalam *virtual pet* dan bertarung?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Penggunaan *device google cardboard* dalam penggunaan *virtual reality*.
2. Menggunakan sistem operasi android pada *smartphone* yang digunakan.
3. Arsitektur jaringan yang digunakan pada permainan ini adalah client-server.
4. *Multiplayer* dapat digunakan oleh 2-4 pemain dalam satu kelompok/*room* dan jumlah pemain dalam satu server maksimal adalah 25 orang.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah terciptanya sebuah permainan mobile bernama 3D Virtual Pet Game Multiplayer dengan menggunakan Google Cardboard.

1.5 Manfaat

Manfaat dari hasil pembuatan Tugas Akhir ini antara lain:

1. Menggantikan *real pet* dengan *virtual pet* yang jauh lebih mudah dan menyenangkan.
2. Membuat permainan yang menghibur bagi seluruh kalangan masyarakat.
3. Sebagai bentuk kontribusi dalam implementasi *virtual* Reality pada dunia *game*.

1.6 Metodologi

Pembuatan Tugas Akhir dilakukan menggunakan metodologi sebagai berikut:

A. Studi literatur
   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan perangkat lunak yaitu sebagai berikut:
   1. Unity3D;
   2. *Virtual Reality* (VR);
   3. Photon Unity Networking;

4. Bahasa Pemrograman C#;

B. Perancangan perangkat lunak

Pada tahap ini akan diawali dengan melakukan analisis awal terhadap permasalahan utama yang memunculkan konsep pembuatan tugas akhir ini. Kemudian melakukan perancangan pengerjaan pembuatan tugas akhir ini dan proses-proses yang perlu dilalui untuk meyelesaikan tugas akhir ini. Langkah yang pada tahap ini diantaranya:

C. Implementasi dan pembuatan sistem

Tugas akhir ini akan menggunakan Unity *Game Engine* sebagai alat pengembangan dengan bahasa C#. Dan *Google Cardboard SDK* dan device *google cardboard* sebagai media dan alat pengembangan *virtual reality* pada unity. Dan Unity Photon Networking sebagai layanan client-server pada unity untuk *game multiplayer*.

D. Uji coba dan evaluasi

Pada tahap ini, akan dilakukan pengujian terhadap perangkat lunak menggunakan data atau skenario yang telah dipersiapkan sebelumnya yakni sebagai berikut:

1. Pengujian *black-box*

   Pengujian *black-box* adalah pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. Pengujian ini dilakukan untuk menguji apakah proses kinerja perangkat lunak *game* ini sudah sesuai dengan kebutuhan pengguna atau tidak.
2. Pengujian usabilitas

   Pengujian usabilitas dilakukan dengan cara melakukan survei ke pengguna Android di sekitar lingkungan Teknik Informatika ITS. Survei dilakukan untuk mengukur tingkat kepuasan dalam hal hiburan dari *game* yang dibuat.

E. Penyusunan laporan tugas akhir

Pada tahap ini, dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi perangkat lunak perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab, antara lain sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan tugas akhir, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II TINJAUAN PUSTAKA

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir ini.

BAB III ANALISIS DAN PERANCANGAN

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi pembangkitan area permainan, dan antarmuka permainan.

BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari perangkat lunak yang dibuat dengan melihat keluaran yang dihasilkan oleh perangkat lunak dan evaluasi untuk mengetahui kemampuan perangkat lunak.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan perangkat lunak selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan Tugas Akhir ini. Teori-teori tersebut *adalah Unity3D Game Engine*, *Virtual Reality* (VR), *Unity Multiplayer*, *Google Cardboard, dan Simulasi Peliharaan*.

2.1 Unity3D *Game Engine*

Unity adalah sebuah *game engine* yang sangat *user friendly* dan mendukung cross platform. Tampilan unity sangatlah mudah, sehingga menjadi salah satu *game engine* yang memiliki peringkat teratas. Beberapa platform yang didukung oleh *game engine* ini seperti android, MacOS, Windows, Xbox 360, Wii, Iphone dan Ipad. Unity dibagun oleh Davil Helgason Nicholas pada tahun 2004. *Game engine* ini dibangun dengan mengekapsulasi beberapa fungsi standar yang umum digunakan dalam pembuatan *game*. Misalnya fungsi rendering, pemanggilan suara, network, atau pembuatan partikel *special effect*. Tidak hanya itu, *game engine* ini menyediakan *asset store* yang merupakan penawaran dari pengembang unity di dunia untuk menambah fitur baru dalam *game*. Secara default, unity telah diatur untuk pembuatan *first person shooting* (FPS) terbukti dengan banyaknya fitur dari FPS yang dimilikinya, namun Unity juga bisa digunakan untuk membuat game bergenre *Role Playing Game* (RPG) dan *Real Time Strategy* (RTS) [4]. Selain itu, unity juga dapat digunakan untuk membuat game dengan genre lain

Unity 3D mendukung *game* berbentuk 3D dan 2D. Fitur animasi juga dimiliki pada *game engine* ini, sehingga dapat digunakan tanpa perlu mengimport dari perangkat lunak lain. Unity juga mendukung blender sebagai perangkat lunak desain dan animasi. Blender dapat digunakan untuk meningkatkan kemampuan desain dan animasi yang masih kurang lengkap pada unity. Bahasa pemrograman yang digunakan adalah C#, JavaScript

dan Boo. Pengguna dapat menggunakan script tersebut secara bersamaan atau memilih salah satunya.

2.2 *Virtual Reality (VR)*

Virtual reality atau realitas virtual adalah sebuah teknologi yang membuat pengguna atau *user* dapat berinteraksi dengan lingkungan yang ada didalam dunia maya yang disimulasikan oleh komputer, sehingga pengguna merasa berada pada lingkungan tersebut. Dalam bahasa indonesia *virtual reality* dikenal sebagai realitas maya [3].

Dengan menggunakan teknologi virtual reality, pengguna tidak seperti melihat layar monitor didepan mereka, melainkan berada di dalam dan dapat berinteraksi dengan dunia 3D. Dengan menggunakan pasca indera sebanyak mungkin, seperti penglihatan, pendengaran, sentuhan, bahkan bau, komputer menjadi sebuah alat penghubung untuk dunia buatan ini. Realitas virtual menggunakan alat bernama *Head Mounted Device* (HMD) sebagai komponen untuk menampilkan lingkungan visual. Salah satu HMD yang terkenal yaitu Oculust Rift dan Microsoft Hololens [5].

2.3 Unity *MultiPlayer*

Multiplayer adalah model cara bermain untuk bermain dengan menggunakan komputer dan video *game* dimana terdapat dua pemain atau lebih pemain yang bermain pada saat yang bersamaan [6]. *Multiplayer* menggunakan layar terpisah atau dua layar atau juga bisa dalam satu layar pemain bermain dengan sistem terpisah yang terhubung ke sistem dengan menggunakan LAN atau server *game* internet. *Game multiplayer* dapat dimainkan dengan internet dan dapat dimainkan oleh pemain yang terpisah tempat.

Pada Unity ada berbagai macam cara untuk mengimplementasikan *multiplayer* seperti menggunakan Photon Unity Networking (PUN), nuggeta ataupun socket programming. Photon adalah plugin yang digunakan dan berkolaborasi dengan unity mengembangkan dan meluncurkan perangkat lunak *games multiplayer* secara *real time* [7]. Arsitektur yang digunakan pada

photon adalah client-server. Yang sudah tidak perlu memperhatikan pengiriman data TCP dan UDP, karena sudah diolah langsung oleh server di photon.

2.4 Google Cardboard

Google cardboard ditujukan untuk dijadikan alat *virtual reality* yang dapat membuat penggunanya dapat menikmati *virtual Reality* (VR) dengan simple, mudah [8]. Alat ini menggunakan bahan kertas kardus sebagai komponen utama. Alat ini dirangkai sedemikian rupa sehingga menyerupai Oculus Rift yang murah dan mudah digunakan. Didalam komponen tersebut terdapat bagian 2 lensa sebagai lensa virtual dan tempat untuk meletakkan *smartphone*. Dimana lensa ini akan melihatkan hasil *virtual reality* yang ada di smartphone tersebut. Untuk *controller* menggunakan magnet yang berada disamping *google cardboard* yang mendeteksi tombol *click*.

Untuk dapat menggunakan *google cardboard*, dibutuhkan smartphone yang bertipe android. Dan untuk pengembangannya terdapat *google cardboard* SDK yang menfasilitasi peletakan kamera pada obyek virtual. Pengembang dapat menggunakan unity sebagai sarana pengembangannya dengan menggunakan bahasa pemrograman C#.

Gambar 2.1 Cardboard[8]

2.5 Pou

Gambar 2.2 Permainan Simulasi Pou[2]

Pou adalah aplikasi permainan mobile untuk Blackberry, iOS dan Android, dikembangkan dan diterbitkan oleh Paul Salameh. *Game* ini memiliki karakter utama sesosok alien yang menyerupai kentang berbentuk segitiga [2]. Pou ini adalah *game* yang termasuk dalam permainan simulasi peliharaan. Dia bisa diberi makan, dibersihkan/dimandikan, diajak bermain, dan ditidurkan. Ada juga mini *game* dalam aplikasi dan koin yang digunakan untuk membeli kostum, dekorasi, dan memodifikasi lingkungan permainan. Secara singkat, Pou bisa digambarkan sebagai sebuah *game* yang mirip dengan Tamagotchi.

2.6 Simulasi Peliharaan Augmented Reality

Simulasi peliharaan adalah sebuah *game* yang dapat menggambarkan mekanisme kehidupan nyata. *Game* ini dibuat sedemikian rupa untuk mendekati kehidupan nyata. Salah satu *game* yang bertipe simulasi seperti simulasi memasak, simulasi kehidupan seperti The SIMS dan sebagainya.

Pada konsep penelitian sebelumnya dibuat sebuah *game* simulasi yaitu Simulasi Peliharaan virtual dengan menggunakan

augmented reality [9]. Permainan ini berusaha untuk menumbuhkan hewan peliharaan dari bayi menjadi dewasa seperti yang ditunjukkan pada Gambar 2.1 dan terdapat fitur bertarung dengan melawan musuh *artificial intelligent* dengan *augmented reality* seperti pada Gambar 2.2.

Pada simulasi peliharaan ini, terdapat berbagai macam perawatan yang dilakukan seperti memberi makan, obat, tidur dan melihat statistik dari hewan peliharaan. Pada simulasi ini juga menampilkan berbagai macam animasi yang digunakan untuk melihat aksi apa yang dilakukan oleh pemain.

Gambar 2.3 Pertumbuhan peliharaan

Gambar 2.4 Pertarungan Peliharaan

2.7 *First Person Shooter* (FPS)

FPS merupakan genre action dalam video game, dimana sudut pandangnya berada pada sisi protagonis. FPS merupakan tipe game yang menampilkan apa yang orang sebenarnya lihat dan lakukan pada sebuah game. Pada umumnya FPS, menampilkan pandangan mata pemain kedepan dan hanya memperlihatkan tangan karakter yang membawa senjata dan perlengkapan di bawah layar. Permainan ini menggunakan pergerakan kedepan, kebelakan dan kesamping dengan menggunakan game controller. Untuk merubah sudut pandangan dapat dilakukan dengan menggerakkan mouse atau game controller [10].

2.8 *Finite State Machine* (FSM)

Finite State Machines (FSM) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan 3 komponen yaitu *State* (Keadaan)*, Event* (Kejadian)*,* dan *Action* (Aksi) [11]. Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada pada salah satu state yang aktif. Sistem dapat beralih atau bertransisi menuju state lain jika mendapatkan masukan atau event tertentu, baik yang berasal dari perangkat luar atau komponen dalam sistemnya itu sendiri (misal interupsi timer). Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relative kompleks

Dalam menggambarkan *state machine*, sebuah lingkaran mewakili *state* di mana dalam lingkaran tersebut dituliskan nama *state* yang dapat berupa huruf, angka, kata, dan. Di antara *state* terdapat garis yang memiliki arah sebagai penghubung antara satu *state* dengan *state* lainnya. Arah dari garis menunjukkan ke arah mana transisi serta label pada garis juga menunjukkan *event trigger* yang akan mengubah *state*. Setiap *state machine* memiliki *start state* yang pertama diinisiasi sebagai langkah awal dalam proses dan di akhiri dengan *stop state* jika proses berakhir.

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas analisis dan perancangan pembuatan *game* 3D simulasi hewan peliharaan dengan menggunakan *google cardboard*. *Game* ini bergenre *First Person Shooter (FPS) Multiplayer* yang berbasis 3D dengan *Virtual Reality*.

3.1 Analisis Sistem

Berdasarkan latar belakang pembuatan dari aplikasi ini, dapat diketahui bahwa *game* yang akan dibuat berbentuk 3D dan diimplementasikan pada platform mobile dengan sistem operasi android. *Game* ini ditujukan kepada perangkat mobile karena lebih praktis. Perangkat mobile untuk game ini di syaratkan dapat mendukung *game* 3D dan *google cardboard* SDK seperti memiliki sensor gyroscope dan sensor magnet.

Game realitas virtual adalah *game* yang dapat diunggulkan dalam memberikan suasana yang nyata dan lebih menarik. Aplikasi *game* ini menggunakan teknologi *google cardboard* sebagai media realitas virtualnya. Dimana teknologi ini menggunakan smartphone sebagai media penampil realitas virtual, sedangkan untuk penerima input digunakan magnet di google carboard yang didukung sensor gyroscope sebagai trigger.

Game ini dibuat dengan Unity3D *game engine*. Game ini mudah mengimplementasikan *game* 3D, mendukung realitas virtual dan cross-platform. Unity3D memiliki plugin *multiplayer* yaitu Photon Unity Networking. Dimana photon yang dipilih berjenis *Real Time* secara client-server, sehingga terjadi sinkronisasi antara client dalam setiap waktu

3.1.1 Deskripsi Umum Perangkat lunak

Perangkat lunak yang akan dibangun adalah *3D virtual pet game multiplayer* dengan menggunakan *google cardboard*. *Game* ini direncanakan berjalan pada perangkat *mobile* dengan sistem

operasi Android. Spesifikasi perangkat mobile memiliki sensor yang mendukung *google cardboard*, seperti gyroscope dan accelerometer.

Game ini direncanakan menggunakan karakter peliharan dalam bentuk 3D, dan memiliki pilihan perawatan seperti memberi makan, obat dan sebagainya. Karakter peliharaan dapat melakukan pertarungan multiplayer. Saat pertarungan digunakan perangkat tambahan yaitu *google cardboard*. Pemain dapat menggunakan magnet yang ada disamping *google cardboard* untuk melakukan intruksi terhadap karakter untuk menyerang.

Saat aplikasi ini pertama dijalankan, pemain akan memilih karakter peliharaan terlebih dahulu. Pada awal permainan, karakter berada pada pertumbuhan yang paling kecil. Karakter dapat berubah menjadi besar, jika levelnya bertambah. Saat dalam permainan simulasi, pengguna menggunakan uang untuk membelikan makanan, obat dan berbagai macam skill untuk meningkatkan kemampuan dari peliharaan tersebut. Pemain dapat menambah uang dengan melakukan pertarungan. Karakter peliharaan bisa sakit dan mati, jika pengguna tidak merawatnya dengan baik.

Peliharaan bisa tumbuh menjadi besar dan mendapatkan skill yang lebih baik, jika sering melakukan pertarungan dengan lawan dan menambah level. Untuk melakukan pertarungan, pengguna perlu membuat *room* yang digunakan sebagai tempat bertarung. Selanjutnya, pemain lain dapat menerima tawaran pertarungan dengan *join room* yang tersedia. Sistem akan menunggu hingga pemain terkumpul pada batas waktu yang ditentukan. Pengguna dapat menggunakan *google cardboard* untuk masuk ke dunia virtual, akan tetapi jika tanpa *google cardboard* dapat melakukan pertarungan dengan kontrol sentuh. Pemain dapat menggerakkan karakter dengan sensor accelerometer dan dapat mengubah sudut pandang dengan merotasi *google cardboard*. Permainan berakhir jika waktu bermain selesai atau salah satu karakter player mati.

3.1.2 Spesifikasi Kebutuhan Fungsional

Berdasarkan deskripsi umum sistem yang telah disampaikan sebelumnya, maka kebutuhan fungsional yang perlu dipenuhi adalah:

1. Pembuatan peliharaan virtual.
2. Fitur *multiplayer* pada pertarungan peliharaan.
3. Penerapan realitas virtual dengan *google cardboard* pada permainan.

3.1.3 Spesifikasi Kebutuhan Non-Fungsional

Terdapat beberapa kebutuhan non-fungsional yang apabila dipenuhi, dapat meningkatkan kualitas dari perangkat lunak ini. Berikut daftar kebutuhan non-fungsional:

1. *Frame Rate*

Game perlu mengatasi nilai *frame rate* rendah atau *lag*. *Game* 3D lebih banyak merender obyek 3D, sehingga membuat game ini *lag* dan dapat menganggu permainan. Untuk meningkatkan *frame rate* perlu mengoptimalkan dan mempertimbangkan jumlah obyek 3D yang dibuat.

2. Grafis

Kenyamanan mata di permainan realitas virtual ini sangat penting. Game ini perlu mengurangi tingkat kecahayaan tinggi yang menyebabkan mata rusak. Untuk grafis yang berada pada simulasi digunakan grafis yang standar dan memudahkan pengguna memainkan permainan ini.

3. Koneksi Internet

Permainan ini mengakomodasi *multiplayer* yang menggunakan internet sebagai media penghubung antar pemain. Sehingga perangkat lunak yang dibangun perlu mempertimbangkan pengiriman data pada koneksi internet yang tidak stabil/rendah.

3.1.4 Karakteristik Pengguna

Berdasarkan deskripsi umum di atas, maka dapat diketahui bahwa pengguna yang menggunakan perangkat lunak ini yaitu pemain. Karakteristik pengguna tercantum dalam Tabel 3.1.

Tabel 3.1 Tabel Karakteristik Pengguna

Nama Aktor	Tugas	Hak Akses Perangkat lunak	Kemampuan yang harus dimiliki
Pemain	Pihak yang menggunakan perangkat lunak	Menggunakan perangkat lunak	Tidak ada

3.2 Perancangan Sistem

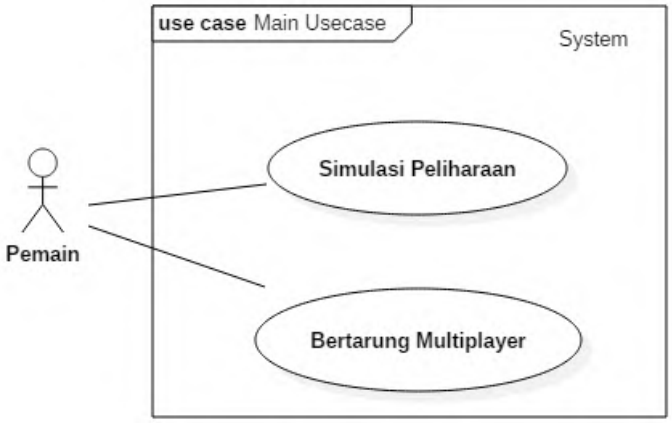
Tahap perancangan dalam subbab ini dibagi menjadi beberapa bagian yaitu perancangan diagram kasus penggunaan, perancangan skenario kasus penggunaan, perancangan antarmuka permainan, perancangan karakter, perancangan kontrol permainan, perancangan alur permainan dan perancangan fitur realitas virtual permainan.

3.2.1 Perancangan Diagram Kasus Penggunaan

Dalam perangkat lunak ini, terdapat dua kasus utama penggunaan yaitu bermain simulasi memelihara peliharaan dan bermain *multiplayer*. Pengguna atau entitas luar dari sistem adalah pemain.

3.2.2 Perancangan Skenario Kasus Penggunaan

Kasus penggunaan untuk merancang perangkat lunak dapat ditunjukkan pada Gambar 3.1.



Gambar 3.1 Usecase utama permainan

Penjelasan dari masing-masing kasus penggunaan dicantumkan pada Tabel 3.2. Tabel tersebut berisi penjelasan skenario yang akan dilakukan ketika pengujian.

Tabel 3.2 Skenario Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-001	Simulasi Peliharaan	Pemain bermain simulasi peliharaan dengan merawat karakter peliharaan yang telah dipilih
2	UC-002	Bertarung *Multiplayer*	Pemain memainkan karakter peliharaan untuk bertarung dengan pemain lain di internet

3.2.2.1 Kasus Penggunaan Permainan

Penjelasan kasus penggunaan permainan untuk skenario UC-001 yakni simulasi peliharaan ditunjukkan pada Tabel 3.3.

Tabel 3.3 Skenario Simulasi Peliharaan

Nama Kasus Penggunaan	Simulasi Peliharaan
Kode	UC-001
Deskripsi	Aktor memilih karakter yang diinginkan dan melakukan perawatan
Aktor	Pemain
Kondisi Awal	Pemain sudah masuk Perangkat lunak
Alur Normal	1. Pemain membuka permainan simulasi. 2. Sistem menampilkan karakter yang diinginkan dengan kondisi awal. 3. Pemain melakukan perawatan 4. Sistem melakukan perubahan terhadap kondisi karakter

Selanjutnya, penjelasan kasus penggunaan permainan untuk skenario UC-002 yakni bertarung *multiplayer* ditunjukkan pada Tabel 3.4.

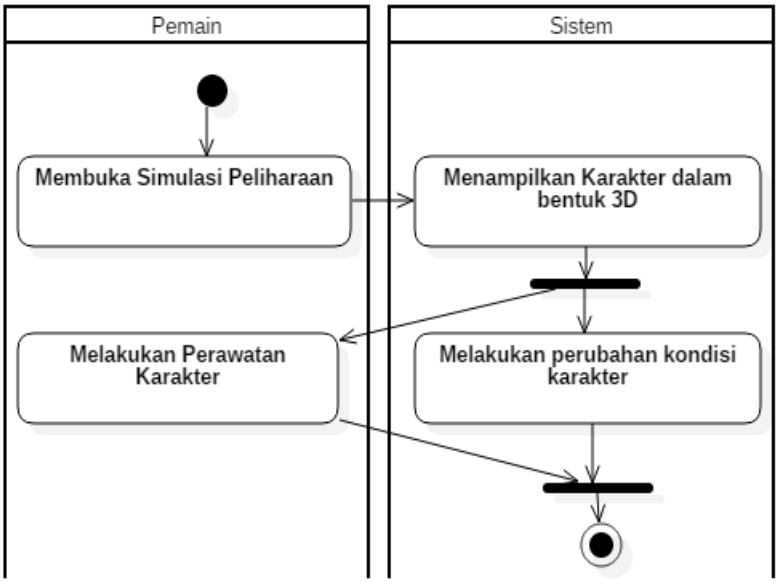
Tabel 3.4 Skenario Bertarung Multiplayer

Nama Kasus Penggunaan	Bertarung *Multiplayer*
Kode	UC-002
Deskripsi	Aktor bermain dengan karakter yang telah terpilih.
Aktor	Pemain
Kondisi Awal	Pemain sudah masuk ke perangkat lunak dan telah memilih karakter.
Alur Normal	1. Pemain memilih mode permainan 2. Pemain membuat tantangan. 3. Sistem menerima tantangan 4. Sistem mengirimkan ke pemain lain. 5. Permainan Dimulai
Alur Alternatif	2. Pemain mencari tantangan 3. Sistem menampilkan tantangan 4. Pemain memilih tantangan 5. Permainan dimulai

3.2.2.2 Diagram Aktivitas

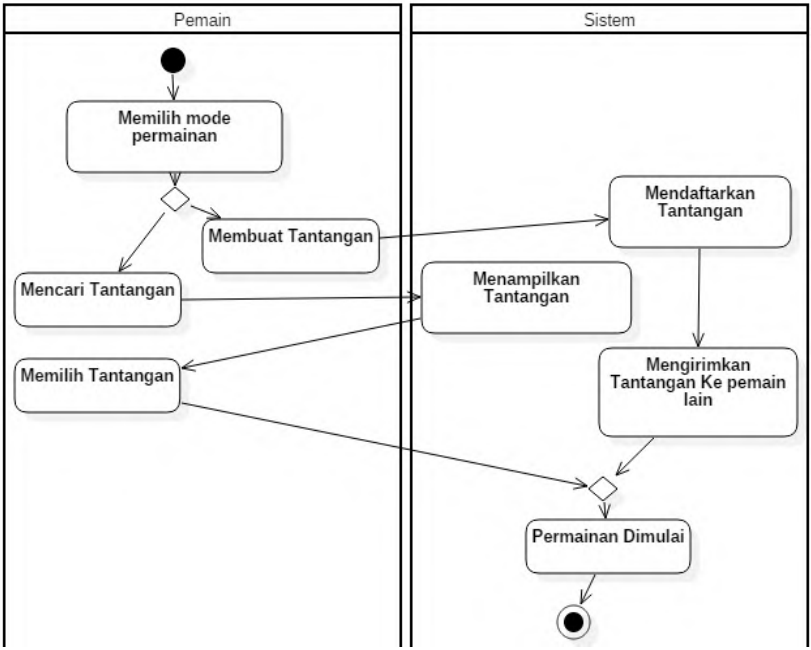
Diagram aktivitas adalah pengambaran langkah-langkah dalam melakukan suatu aktivitas tertentu. Diagran ini menampilkan langkah-langkah normal yang dilakukan pengguna yaitu pemain. Diagram aktivitasi bertujuan untuk memberikan gambaran pemain menjalankan studi kasus tertentu di permainan ini dari awal permainan hingga kondisi permainan berakhir.

Diagram aktivitas dari kasus penggunaan UC-001, yaitu simulasi peliharaan ditunjukkan pada Gambar 3.2.



Gambar 3.2 Diagram aktivitas simulasi peliharaan

Kemudian, diagram aktivitas dari kasus penggunaan UC-002, yaitu bertarung *multiplayer* ditunjukkan pada Gambar 3.3.



Gambar 3.3 Diagram Bertarung Multiplayer

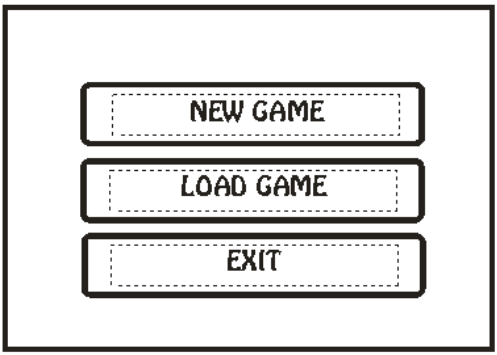
3.2.3 Perancangan Antarmuka Permainan

Subbab ini membahas rancangan antarmuka yang digunakan pada permainan ini. Rancangan antarmuka yang dibahas meliputi desain dan fungsi antarmuka. Dalam perangkat lunak ini terdapat beberapa antarmuka, yaitu **Antamuka Menu Utama, Antarmuka Simulasi Peliharaan, Antarmuka Bertarung** dan **Antarmuka Cardboard**.

3.2.3.1 Antarmuka Menu Utama

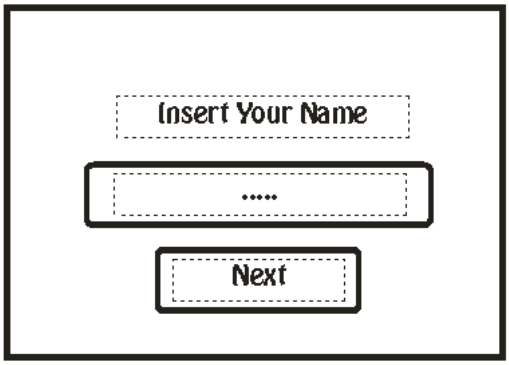
Antarmuka menu utama merupakan antarmuka yang pertama kali muncul ketika perangkat lunak pertama kali dijalankan. Antarmuka pertama akan ini berisi 3 tombol, yaitu tombol "**New Game**" untuk menuju ke antarmuka pemilihan karakter, "**Load Game**" untuk menuju ke antarmuka simulasi saat

terakhir dimainkan dan "**Exit**" untuk keluar dari perangkat lunak. Desain rancangan dari menu utama *game* ini ditunjukkan oleh Gambar 3.4.



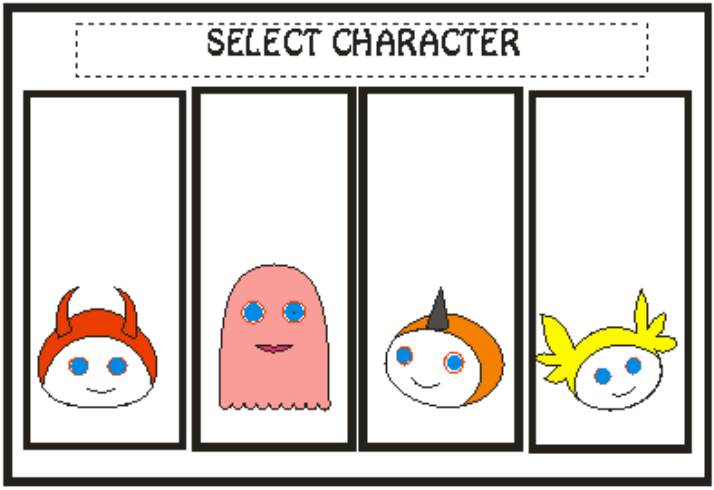
Gambar 3.4 Rancangan Menu Utama

Setelah tombol "**New Game**" di tekan, maka sistem menampilkan antarmuka *form* penulisan nama pemain yang ditunjukkan pada Gambar 3.5. Di *form* ini, pemain memasukkan namanya. Sistem akan menyimpan informasi nama pemain dan digunakan sebagai identitas pemain pada saat simulasi peliharaan dan bertarung.



Gambar 3.5. *Form* pengisian nama

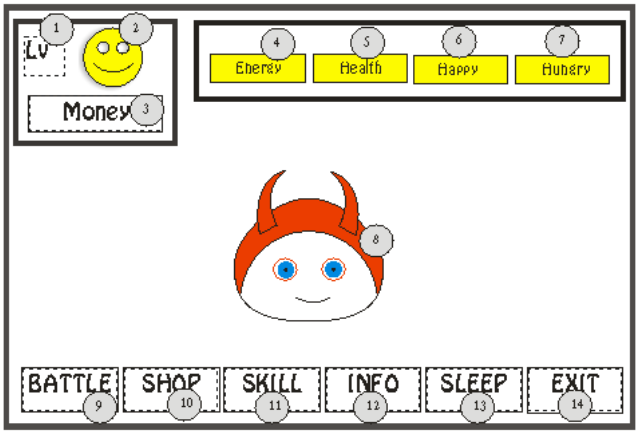
Selanjutnya, pemain memilih tombol "**Next**" dan menuju ke antarmuka pemilihan karakter peliharaan yang ditunjukkan pada Gambar 3.6. Pemain dapat memilih 4 karakter peliharaan yang telah disediakan. Pemain memilih karakter dengan cara menekan gambar karakter yang diinginkan. Setelah selesai memilih karakter yang diinginkan, sistem akan menampilkan notifikasi hasil data yang telah dimasukkan pemain ditampilkan, jika sudah yakin benar maka pemain memilih tombol "**Confirm**" dan jika ingin memperbaiki data sebelumnya, maka memilih tombol "**Cancel**".



Gambar 3.6 Antarmuka pemilihan karakter peliharaan

3.2.3.2 Antarmuka Simulasi Peliharaan

Tampilan **Simulasi Peliharaan** adalah antarmuka yang tampil setelah antarmuka menu utama selesai. Antarmuka ini digunakan untuk memelihara karakter peliharaan. Pemain dapat membeli makanan dan obat pada toko. Selain itu pemain dapat meningkatkan kemampuan dengan latihan ataupun menggunakan skill. Rancangan antarmuka simulasi peliharaan ditunjukkan pada Gambar 3.7.



Gambar 3.7 Antarmuka Simulasi Peliharaan

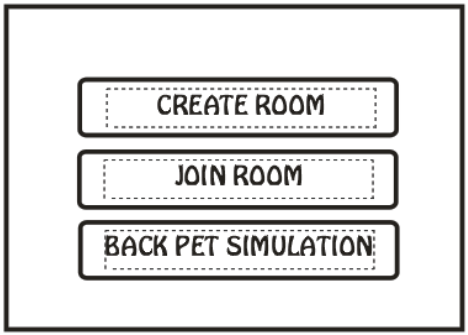
Penjelasan dari antarmuka simulasi peliharaan, yang ditunjukkan sesuai dengan nomornya, adalah sebagai berikut :

1. **Level Info**, merupakan antarmuka yang menunjukkan level karakter saat ini.
2. **Karakter Info**, merupakan antarmuka yang menunjukkan karakter apa yang dipilih dan nama dari pemain yang telah dimasukkan.
3. **Money**, merupakan antarmuka yang menunjukkan jumlah uang yang dimiliki oleh pemain saat ini
4. **Energy**, merupakan antarmuka yang menunjukkan kondisi energi yang dimiliki oleh karakter peliharaan.
5. **Health**, merupakan antarmuka yang menunjukkan kondisi kesehatan yang dimiliki oleh karakter peliharaan.
6. **Happy**, merupakan antarmuka yang menunjukkan kondisi kebahagiaan yang dimiliki oleh karakter peliharaan.
7. **Hungry**, merupakan antarmuka yang menunjukkan kondisi lapar yang dimiliki oleh karakter peliharaan.

8. **Karakter peliharaan**, bentuk 3D karakter peliharaan yang dibuat oleh sistem.
9. **Tombol "Battle"**, merupakan tombol untuk menuju ke antarmuka bertarung.
10. **Tombol "Shop"**, merupakan tombol untuk menuju ke antarmuka toko penjualan perlengkapan peliharaan.
11. **Tombol "Skill"**, merupakan tombol untuk menuju ke antarmuka pemilihan skill dari karakter peliharaan.
12. **Tombol "Info"**, merupakan tombol untuk mengetahui tingkat kemampuan karakter peliharaan dan informasi pertarungan.
13. **Tombol "Sleep"**, merupakan tombol untuk membuat karakter peliharaan tertidur atau bangun.
14. **Tombol "Exit"**, merupakan tombol untuk keluar dari permainan simulasi.

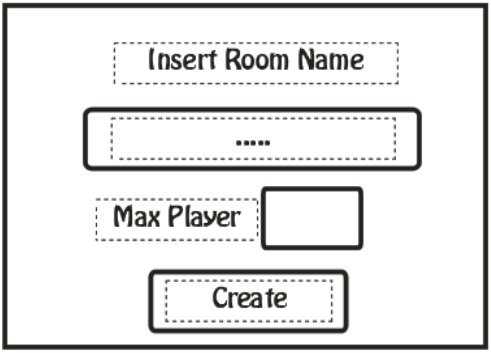
3.2.3.3 Antarmuka Bertarung

Antarmuka **Bertarung** merupakan antarmuka yang tampil setelah tombol "**Battle**" pada Antarmuka **Simulasi Peliharaan** ditekan. Pada antarmuka ini akan ditampilkan pilihan "**Create Room**", "**Join Room**", "**Back Pet Simulation**". Pemain dapat membuat *room* untuk bermain dengan kalangan tertentu dengan pemain yang terbatas. Sistem *room* ini didefinisikan oleh plugin photon unity networking. Ketika pemain memilih "**Create Room**", maka sistem menampilkan antarmuka membuat *room*. Ketika pemain memilih "**Join Room**", maka sistem menampilkan antarmuka bergabung *room*. Dan ketika pemain memilih "**Back Pet Simulation**", maka sistem akan mengembalikan pemain ke antarmuka simulasi peliharaan. Rancangan antarmuka bertarung ditunjukkan pada Gambar 3.8.



Gambar 3.8 Antarmuka utama bertarung

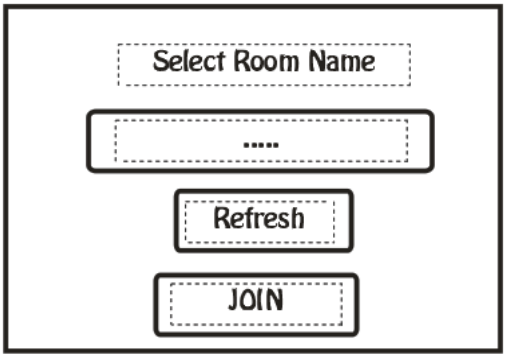
Saat pemain berada diantarmuka membuat room, pemain perlu menuliskan nama room yang diinginkan. Nama room yang diberikan haruslah unik untuk mempermudah mengenali room oleh pemain lawan. Selanjutnya, pemain dapat memilih jumlah maksimal dari pemain pada kolom *max player*. Selanjutnya pemain dapat memilih tombol "**Create**" untuk membuat room. Sistem akan membawa pemain menuju ke arena pertandingan. Rancangan dari antarmuka membuat room ditunjukkan pada Gambar 3.9.



Gambar 3.9 Antarmuka "Create Room"

Saat pemain bergabung room, pemain dapat memilih room yang sudah dibuat. Jika room belum ditemukan atau masih kosong, pemain dapat menekan tombol "**refresh**", sehingga sistem akan

menampilkan kembali room yang sudah terdaftar. Selanjutnya pemain dapat memilih tombol "**Join**" untuk bergabung room dan menuju ke arena pertarungan. Rancangan antarmuka bergabung room ditunjukkan pada Gambar 3.10.

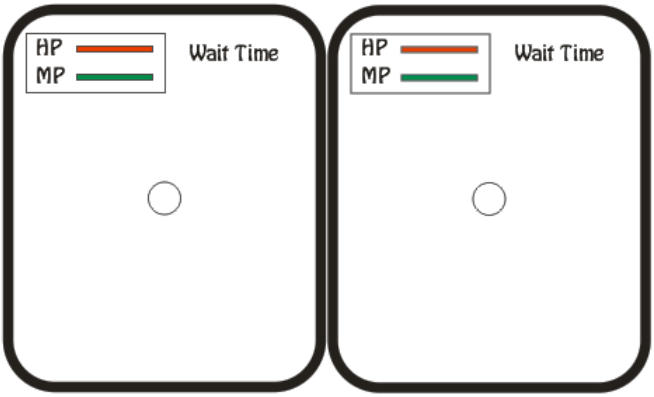


Gambar 3.10 Antarmuka "Join Room"

3.2.3.4 Antarmuka Realitas Virtual Pemain

Antarmuka realitas rirtual (VR) dalam *game* ini menggunakan *package* atau *library google cardboard* SDK. Antarmuka VR digunakan pada saat melakukan bertarung *multiplayer*. Dalam *game* ini, tampilan dari realitas virtual berbentuk First Person Shooter (FPS) terhadap karakter peliharaan.

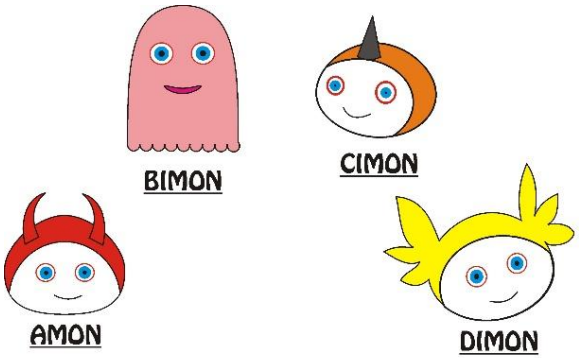
Pada perancangan antarmuka ini, akan didesain tampilan yang dilihat pemain saat melakukan pertarungann dengan menggunakan *google cardboard*. Karakteristik dari tampilan *google cardboard* adalah layar yang terbentuk menjadi 2 bagian dengan 2 sudut pandang sesuai dengan mata. Pada antarmuka ini ditampilkan nilai *health point(HP)* yang merupakan nilai nyawa karakter yang dimiliki dan *magic point(MP)* yang merupakan nilai yang digunakan untuk mengeluarkan serangaan *skill* dalam bentuk *bar*. Selain itu ditampilkan kalkulasi waktu bermain (*wait time*) dan sasaran target untuk menembak. Rancangan tampilan dari antarmuka *google cardboard* pemain ditunjukkan pada Gambar 3.11.



Gambar 3.11 Antarmuka Cardboard

3.2.4 Perancangan Karakter Peliharaan

Pada subbab ini akan membahas rancangan karakter yang ada dalam *game*. Karakter dibuat dalam bentuk 3D. Dalam *game* ini, terdapat 4 macam karakter yaitu Amon, Bimon, Cimon dan Dimon. Karakter tersebut memiliki perkembangan dari kecil karakter menjadi karakter dewasa. Berikut konsep desain awal karakter dalam bentuk 2D ditunjukkan pada Gambar 3.12.

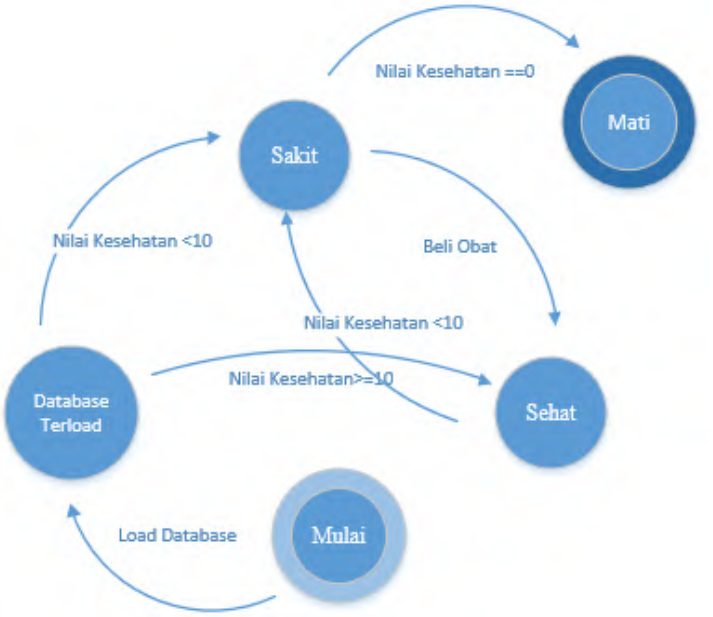


Gambar 3.12 Rancangan karakter peliharaan

3.2.5 Perancangan Permainan Simulasi Peliharaan

Pada subbab ini akan dibahas mengenai rancangan simulasi yang dilakukan pada hewan peliharaan oleh sistem dan pemain. Perancangan ini menggunakan *finite state machine* (FSM). Beberapa parameter yang dimiliki oleh hewan peliharaan yaitu kesehatan (*health*), Kelaparan (*hungry*), kebahagiaan (*happy*), dan tenaga (*energy*).

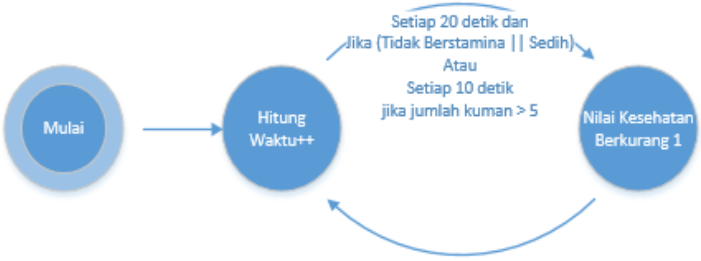
1. **Kesehatan**



Gambar 3.13 FSM Kesehatan Peliharaan

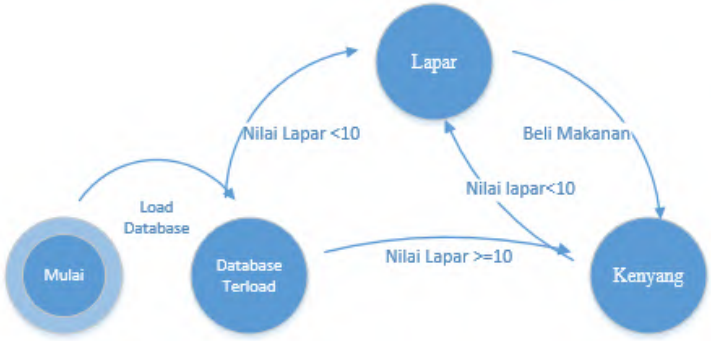
Pada model FSM kesehatan yang ditunjukkan pada Gambar 3.13 ini, terdapat dua kondisi kesehatan dari peliharaan yaitu sakit dan sehat. Kondisi awal didapatkan dari kondisi terakhir yang tersimpan. Jika pemain baru memulai permainan, maka nilai kesehatan akan di pasang sebesar 100. Nilai kesehatan ini dipengaruhi oleh kondisi energi dan kebahagiaan yang ditandai

dengan tidak berstamina dan sedih. Jika peliharaan berada pada kondisi tidak berstamina atau mengantuk ataupun keadaan sedih, maka nilai kesehatan akan dikurangi tiap 20 detik. Nilai kesehatan juga dipengaruhi dengan keberadaan kuman. Kuman akan dimunculkan secara acak oleh sistem pada waktu tertentu. Ketika kuman mencapai jumlah 5 atau lebih, maka nilai kesehatan akan dikurangi 1 setiap 10 detik. Proses pengurangan nilai kesehatan ini ditunjukkan pada Gambar 3.14. Nilai kesehatan yang berada kurang dari 10, menjadikan peliharaan sakit atau dalam keadaan lemah. Pemain dapat menaikkan nilai kesehatan menjadi lebih tinggi dengan membeli obat yang berada pada toko. Jika nilai kesehatan berada pada nilai 0, maka peliharaan akan mati.



Gambar 3.14 FSM kontrol waktu kesehatan

2. Kelaparan

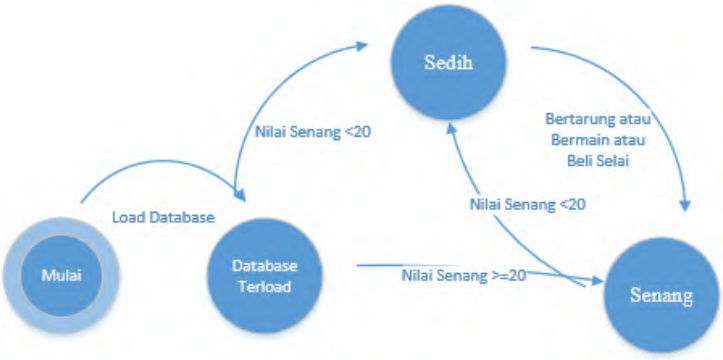


Gambar 3.15 FSM Kelaparan Peliharaan

Pada model FSM kelaparan yang ditunjukkan pada Gambar 3.15, terdapat dua kondisi perliharaan terhadap makanan yaitu lapar dan kenyang. Kondisi awal didapatkan dari kondisi terakhir yang tersimpan. Jika pemain baru memulai permainan, maka nilai lapar akan dipasang sebesar 100. Nilai lapar ini dipengaruhi oleh waktu. Waktu dijalankan oleh sistem dan setiap 20 detik akan mengurangi nilai lapar. Proses pengurangan nilai kelaparan ini ditunjukkan pada Gambar 3.16. Jika nilai lapar kurang dari 10, maka peliharaan akan berada pada kondisi kelaparan. Untuk menaikan nilai kelaparan, pemain membeli makanan pada toko.

Gambar 3.16 FSM kontrol waktu lapar

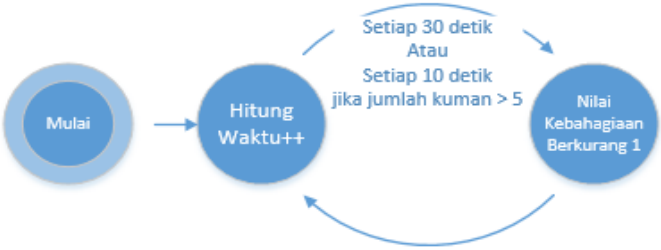
3. Kebahagiaan



Gambar 3.17 FSM Kebahagiaan

Pada model FSM kebahagiaan yang ditunjukkan pada Gambar 3.17 terdapat dua kondisi kebahagian yaitu senang dan sedih.

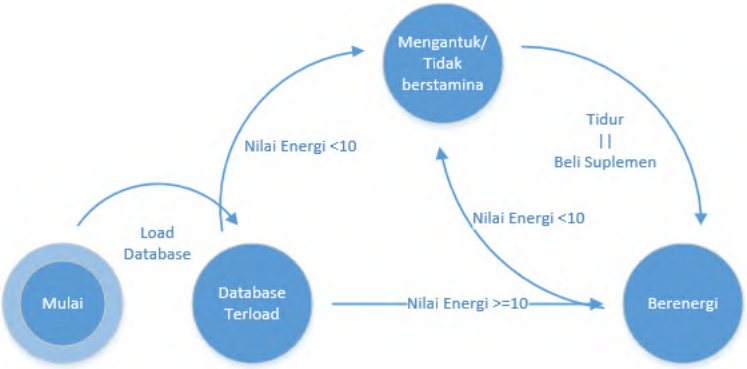
Kondisi awal didapatkan dari kondisi terakhir yang tersimpan. Jika pemain baru memulai permainan, maka nilai senang akan dipasang sebesar 100. Nilai senang ini dipengaruhi oleh waktu dan lapar. Waktu dijalankan oleh sistem dan setiap 30 detik nilai senang akan dikurangi. Jika peliharaan dalam kondisi lapar, maka sistem akan mengurangi nilai senang kembali, seperti yang ditunjukkan pada Gambar 3.19. Sama halnya dengan nilai kesehatan, nilai kebahagiaan juga dipengaruhi dengan keberadaan kuman. Kuman akan dimunculkan secara acak oleh sistem pada waktu tertentu. Ketika kuman mencapai jumlah 5 atau lebih, maka nilai kebahagiaan akan dikurangi 1 setiap 10 detik. Proses pengurangan nilai kebahagiaan dapat ditunjukkan pada Gambar 3.18. Jika nilai senang kurang dari 20, maka peliharaan akan sedih. Untuk menaikan nilai senang dapat dilakukan dengan bermain, bertarung atau membeli selai di toko.



Gambar 3.18 FSM kontrol waktu kebahagiaan

Gambar 3.19 FSM kontrol waktu kebahagiaan 2

4. Tenaga



Gambar 3.20 FSM Tenaga

Pada model FSM tenaga yang ditunjukkan pada Gambar 3.19, terdapat dua kondisi perliharaan terhadap tenaga yaitu berenergi dan mengantuk/tidak berstamina. Kondisi ini awal didapatkan dari kondisi terakhir yang tersimpan. Jika pemain baru memulai permainan, maka nilai energi akan dipasang 100. Nilai energi ini dipengaruhi oleh waktu dan lapar. Waktu dijalankan oleh sistem dan setiap 25 detik akan mengurangi nilai energi. Proses pengulangan energi ini ditunjukkan pada Gambar 3.21. Jika kondisi lapar, nilai energi juga dikurangi, seperti yang ditunjukkan pada Gambar 3.22. Jika nilai energi kurang dari 10, maka peliharaan akan berada pada kondisi mengantuk. Untuk menaikan nilai energi dapat dilakukan dengan tidur atau membeli suplemen pada toko.

Gambar 3.21 FSM kontrol energi

Gambar 3.22 FSM kontrol energi 2

3.2.6 Perancangan Permainan Bertarung

Rancangan permainan menggunakan sudut pandang *first person shooter* (FPS) saat pemain mengendalikan karakter peliharaan 3D. Pemain akan bertarung dengan karakter milik pemain lain dalam duel *multiplayer*. Setiap karakter peliharaan dapat diubah kemampuannya.

Cara memenangkan pertandingan adalah dengan menghabiskan HP karakter lawan terlebih dahulu sebelum HP pemain habis. Permainan dimulai dengan sistem men-set nilai *waiting time*, untuk menunggu pemain bergabung dalam satu *room*. Pada waktu ini, karakter pemain akan diam dan tidak bisa berjalan dan menyerang hingga waktu menunggu selesai. Lalu sistem akan menampilkan nilai *playtime* yang menunjukkan pertarungan dimulai. Selama permainan, jika ada HP pemain lawan sudah mencapai 0, maka permainan selesai. Dan jika waktu bermain sudah selesai dan nilai HP belum ada yang 0, maka akan dilakukan perhitungan HP antar pemain. Pemain dengan HP yang tertinggilah yang akan menang. Dan jika nilai HP sama, maka permainan akan dianggap sebagai seri.

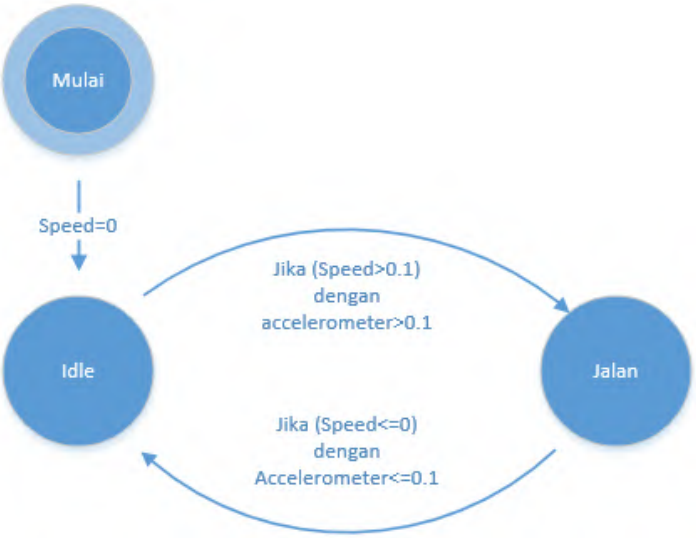
3.2.7 Perancangan Kontrol Karakter Bertarung

Untuk mendukung kontrol karakter bertarung dan realitas virtual diperlukan smartphone yang memiliki sensor gyroscope dan accelerometer. Gyroscope berfungsi sebagai rotasi karakter saat melakukan pergerakan kepala terhadap sumbu y dari *google*

cardboard. Dan accelerometer sebagai kontrol untuk mengatur kecepatan berlari karakter. Dan trigger pada *google cardboard* dapat digunakan sebagai alat pemicu serangan.

Karakter memiliki 3 rancangan FSM pada saat pertarungan yaitu FSM karakter berjalan, FSM karakter menyerang dan FSM *health Point* karakter:

1. **FSM karakter berjalan**

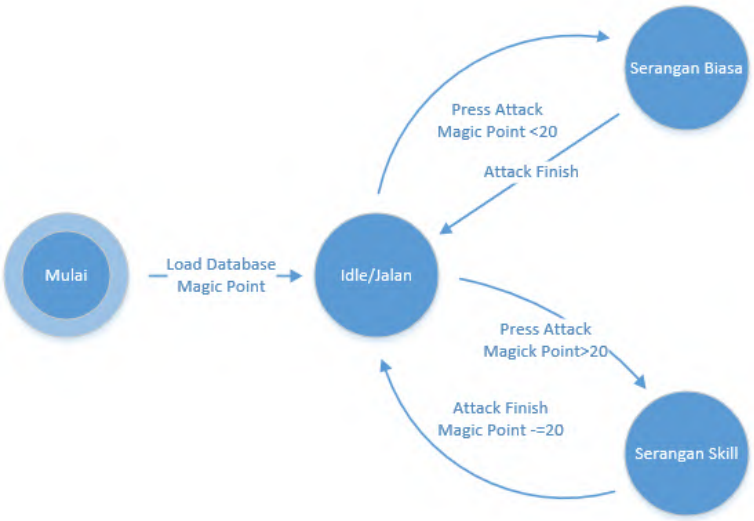


Gambar 3.23 FSM Karakter Berjalan

Pada model FSM yang ditunjukkan pada Gambar 3.23 ini memperlihatkan bagaimanakah karakter berjalan. Dengan kontrol menggunakan accelerometer untuk menambah kecepatan / *speed*, terdapat 2 kondisi yaitu *idle* dan jalan. Jika accelerometer berada diatas 0.1 atau merubah *speed* 0.1 maka karakter akan berjalan dan sedangan berada dibawah atau sama, maka karakter akan diam dan tidak bergerak. Accelerometer menggunakan sumbu z, sehingga jika pemain menunduk saat dalam menggunakan *google cardboard* maka

akan menambah nilai accelerometer. Sedangkan jika pemain menghadap ke depan, maka nilai accelerometer akan mendekati 0 atau karakter tidak bergerak. Sedangkan untuk rotasi sudut pandang, pemain dapat melakukan gerakan kepala searah sumbu y.

2. FSM karakter menyerang



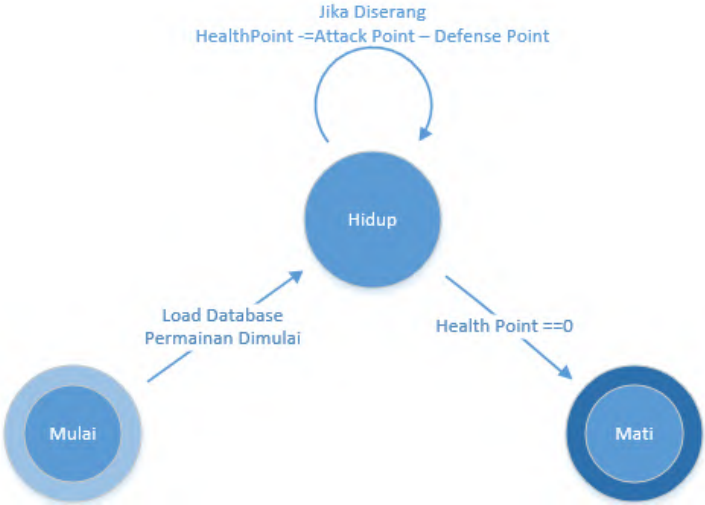
Gambar 3.24 FSM Karakter Menyerang

Pada model FSM yang ditunjukkan pada Gambar 3.24 ini, memperlihatkan bagaimanakah karakter melakukan penyerangan. Penyerangan karakter ini bergantung dari nilai *magic point* yaitu serangan dengan skill atau serangan biasa. Ketika pemain menekan trigger pada *google cardboard* atau serang, maka akan terdapat 2 kondisi serangan. Jika *magic point* berada diatas 20 poin atau sama dengan, maka serangan yang dimunculkan adalah serangan skill yang tingkat pemberian luka adalah maksimal sesuai dengan kekuatan yang dimiliki. Akibat dari penyerangan jenis ini adalah mengurangi nilai dari *magic point* sebesar 20 poin. Jika *magic point*

berada pada kurang dari 20 poin maka serangan yang dilakukan adalah serangan biasa yang nilai pemberian luka sebesar level dari karakter. *Magic point* akan bertambah sebesar 5 poin dalam selang waktu 2 detik seperti yang ditunjukkan Gambar 3.25.

Gambar 3.25 FSM penambahan *magic point*

3. FSM *health point* karakter



Gambar 3.26 FSM *Health Point* Karakter

Pada model FSM yang ditunjukkan pada Gambar 3.26 ini, memperlihatkan model alur kehidupan karakter saat bertarung dengan dinyatakan pada *health point*. Nilai *health point* disesuaikan dengan tingkat kemampuan karakter dan bernilai penuh pada awal pertandingan. Nilai *health point* bisa diubah dengan meningkatkan kemampuan dari karakter saat simulasi peliharaan. Nilai *health point* akan berkurang, jika diserang lawan. Perancangan pengurangan nilai *health point* yaitu nilai *health point* sekarang dikurangi besar nilainya serangan yang telah dikurangi oleh *defense point* dari pemain sendiri. Karakter akan mati jika nilai *health point*-nya 0.

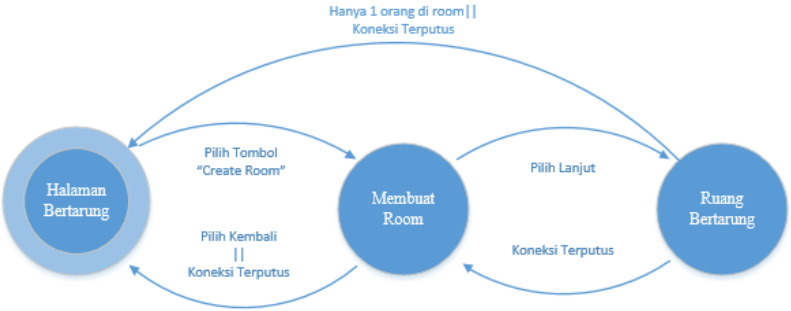
3.2.8 Perancangan Sistem *Multiplayer*

Perancangan *multiplayer* pada *game* ini menerapkan plugin *Photon Unity Networking*. *Photon Unity Networking* menggunakan sistem *room* pada pengaturannya. *Photon Unity Networking* menyediakan *server* untuk menginstansiasi karakter pada jaringan sehingga semua pemain yang tersambung pada satu *room* dapat bertemu. Beberapa tahap koneksi yang ada di photon ini adalah digunakan untuk membuat *room* dan bergabung dengan *room*:

1. Photon membuat *room*

Rancangan alur membuat *room* ditunjukkan pada Gambar 3.27. Pada halaman awal bertarung akan didesain dimana terdapat pilihan **Create Room**. *Room* ini digunakan sebagai tempat koneksi antar pemain yang bermain dalam ruang tertentu agar terjadi pemisahan pengaturan koneksi dengan pemain global. Untuk membuat *room* dibutuhkan koneksi terhadap server untuk mengetahui apakah pemain dalam keadaan online. Jika tidak, pemain akan kembali ke halaman awal. Ketika memilih **Create Room** Sistem akan melakukan koneksi terhadap server di photon. Jika pemain berhasil terhubung, pemain akan berpindah ke halaman **Create Room**. Di halaman membuat *room*, pemain akan menuliskan nama *room* dan jumlah maksimal pemain yang diinginkan dalam satu *room*. Lalu pemain memilih pilihan lanjut

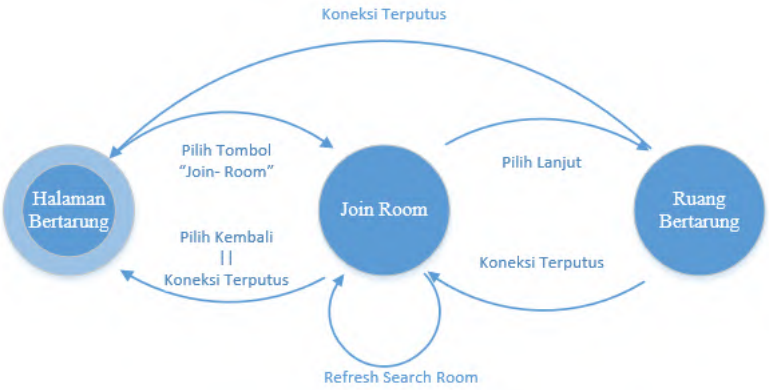
dan sistem akan melakukan pengecekan kembali pada koneksi server photon. Jika berhasil pemain akan menuju halaman bermain. Dan jika koneksi pemain terputus, maka akan kembali ke halaman membuat *room*. Jika pada saat permainan dimulai pemain kehilangan koneksi, maka pemain akan kembali ke halaman awal.



Gambar 3.27 FSM *Create Room*

2. Photon bergabung *room*

Rancangan alur bergabung *room* ditunjukkan pada Gambar 3.28. Pada halaman awal bertarung terdapat pilihan bergabung dengan *room* yang sudah ada. Saat pemain memilih pilihan ini, sistem akan melakukan koneksi terhadap server photon dan mengambil *room* mana saja yang telah dibuat. Jika koneksi berhasil, pemain akan dibawa menuju halaman **Join Room**. Jika koneksi pemain terputus, maka akan kembali ke halaman awal. Saat di halaman **Join Room**, sistem akan menampilkan *room* yang sedang online dan dipakai. Jika *room* yang diinginkan belum tampil, pemain dapat melakukan *refresh*, sehingga sistem akan memperbarui daftar *room* yang sedang online. Lalu pemain memilih *room* yang diinginkan dan memilih pilihan lanjut untuk menuju ruangan bertarung. Jika koneksi pemain terputus, maka kembali ke halaman **Join Room**. Dan jika saat permainan dimulai koneksi pemain terputus, pemain kembali ke halaman awal.



Gambar 3.28 FSM *Join Room*

3. Sinkronisasi Karakter

Untuk sinkronisasi karakter, sistem perlu menyamakan gerakan yang dilakukan oleh pemain ke *device* pemain lain, sehingga terbentuk sinkronisasi gerakan. Beberapa data yang perlu dikirimkan ke server photon untuk sinkronisasi ini adalah gerakan, *health point* yang dimiliki pemain dan waktu permainan. Salah satu pemain akan menjadi kontrol untuk mengatur waktu permainan seperti server local dalam satu *room*.

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan perangkat lunak. Di dalamnya mencakup proses penerapan dan pengimplementasian algoritma, dan antar muka yang mengacu pada rancangan yang telah dibahas sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi dari Tugas Akhir ini ditunjukkan pada Tabel 4.1 sebagai lingkungan pembangunan aplikasi dan Tabel 4.2 sebagai lingkungan percobaan aplikasi.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak (1)

Perangkat Keras	Prosesor: Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz Memori: 4096 MB RAM
Perangkat lunak	Sistem Operasi: Microsoft Windows 8 64-bit Perangkat Pengembang: Unity3D

Tabel 4.2 Lingkungan Implementasi Perangkat Lunak (2)

Perangkat Keras	Prosesor: 4x ARM Cortex-A7 @ 1.18GHz Memori: 921MB RAM
Perangkat lunak	Sistem Operasi: Android Versi 4.4.2

4.2 Implementasi Alur Permainan

Pada subbab ini, akan dibahas implementasi fungsi utama permainan berdasarkan perancangan dari FSM dan kode program yang digunakan.

4.2.1 Implementasi Antarmuka Menu Utama

Antarmuka **Menu Utama** adalah tampilan pertama saat perangkat lunak pertama kali dibuka. Tampilan ini memiliki tiga

utama tombol yaitu "**New Game**", "**Load Game**" dan "**EXIT**". Tampilan dari antarmuka ini ditunjukkan pada Gambar 4.1.

Gambar 4.1 Tampilan antarmuka main menu

Tombol "**New Game**" digunakan untuk masuk ke antarmuka selanjutnya yaitu antarmuka pengisian data pemain dan pemilihan karakter peliharaan. Dengan menekan tombol ini, sistem akan melakukan pengecekan terhadap data *game* yang tersimpan. Jika ada, sistem akan memunculkan notifikasi untuk menghapus data atau tidak. Jika iya, sistem akan menampilkan antarmuka selanjutnya. Jika tidak, kembali ke antarmuka main menu. Berikut adalah implementasi dari tombol "**New Game**" yang ditunjukkan pada Kode Sumber 4.1.

```
1.  public void ActivatePanel2(bool isForce){
2.      DeactiveAllPanel ();
3.      if (isForce) {
4.          secondPanel.SetActive(true);
5.      } else {
6.          if (!playerManager.checkFile()) {
7.              secondPanel.SetActive (true);
8.          } else {
9.              ActivatePanel5(1);
```

```
        }
    }
}
```

Kode Sumber 4.1 Tombol "New Game"

Kode sumber diatas menunjukkan bahwa sistem menampilkan notifikasi bahwa terdapat data yang tersimpan dan langsung menuju scene selanjutnya dengan syarat menghapus data. Pada Gambar 4.2 ditunjukkan notifikasi yang muncul saat ditemukan data *game* yang tersimpan, dan sistem memerlukan persetujuan pemain untuk menghapus data sebelumnya jika ingin bermain dari awal. Jika memilih tidak, maka akan kembali ke *main menu*.

Gambar 4.2 Notifikasi "New Game"

Tombol "**Load Game**" digunakan untuk menuju ke *game* secara langsung dari kondisi terakhir data disimpan. Sistem akan mencari dulu data yang disimpan. Jika ada, maka langsung menuju antarmuka *game* simulasi peliharaan. Jika tidak ada, maka akan menampilkan notifikasi peringatan. Berikut ini adah implementasi dari tombol "**Load Game**" yang ditunjukkan pada Kode Sumber 4.2.

```
public void LoadPanel(){
    if (playerManager.LoadCharacter ()) {
        GoToLoadScene();
    } else {
        ActivatePanel5(2);
    }
```

```
7.  }
```

Kode Sumber 4.2 Tombol Load Game

Pada Gambar 3.4 adalah notifikasi yang muncul jika pemain memilih tombol "**Load Game**", tetapi tidak ada data *game* yang ditemukan oleh sistem.

Gambar 4.3. Notifikasi "Load Game"

Tombil "**Exit**" digunakan untuk keluar dari perangkat lunak ini. Implementasi dari tombol "**Exit**" ditunjukkan pada Kode Sumber 4.3.

```
1.  public void Quit(){
2.      Application.Quit ();
3.  }
```

Kode Sumber 4.3 Tombol "Quit Game"

Setelah tombol "**New Game**" pada antarmuka **Main Menu** ditekan, terdapat antarmuka pemilihan karakter pemain. Antarmuka ini memiliki 2 fungsi yaitu untuk mendapatkan informasi pemain dan memilih karakter peliharaan yang hendak digunakan pemain pada simulasi peliharaan dan bertarung. Untuk informasi pemain, pemain mengisikan nama yang diinginkan dan sistem akan menyimpan nama pemain sebagai identitas saat permainan simulasi dan saat pertarungan. Tampilan dari formulir permintaaan informasi pemain ditunjukkan pada Gambar 4.4.

Gambar 4.4 Antarmuka Formulir Informasi Pemain

Setelah selesai mengisi nama, pemain dapat menekan tombol "**NEXT**". Sistem akan menampilkan antarmuka pemilihan karakter peliharaan yang ditunjukkan pada Gambar 4.5. Untuk memilih karakter peliharaan, pemain harus menekan salah satu gambar karakter peliharaan yang diinginkan. Dalam *game* ini terdapat 4 pilihan karakter yaitu Amon, Bimon, Cimon dan Dimon. Masing–masing karakter ini adalah bentuk 2 dimensi dari karakter peliharaan 3 dimensi. Bentuk karakter peliharaan yang dewasa tidak ditunjukkan untuk menambah rasa penasaran dari pemain. Berikut ini adalah gambar dari pemilihan peliharaan.

Gambar 4.5 Antarmuka Pemilihan Karakter Peliharaan

Implementasi untuk memilih karakter peliharaan ditunjukkan pada Kode Sumber 4.4.

```
1.  public void CharacterSelection1(){
2.      pm.idCharacter="Character1";
3.  }
4.  public void CharacterSelection2(){
5.      pm.idCharacter="Character2";
6.  }
7.  public void CharacterSelection3(){
8.      pm.idCharacter="Character3";
9.  }
10. public void CharacterSelection4(){
11.     pm.idCharacter="Character4";
12. }
```

Kode Sumber 4.4 Pemilihan Karakter Peliharaan

Setelah memilih karakter yang diinginkan, sistem akan menampilkan notifikasi dari hasil data yang telah dimasukkan yaitu nama pemain dan karakter terpilih. Tampilan dari hasil notifikasi informasi pemain ditunjukkan pada Gambar 4.6.

Gambar 4.6 Notifikasi Hasil Data Pemain

Jika pemain menekan tombol "**Yes**", sistem menuju permainan simulasi peliharaan. Dan jika memilih tombol "**Cancel**" maka akan kembali ke antarmuka formulir pengambilan nama. Implementasi notifikasi ditunjukkan pada Kode Sumber 4.5.

```
public void ShowCharacter(){

1.      string name="";
2.      if (pm.idCharacter == "Character1") {
3.          name = "Amon";
4.      } else if (pm.idCharacter == "Character2") {
5.          name = "Bimon";
6.      } else if (pm.idCharacter == "Character3") {
7.          name = "Cimon";
8.      } else if (pm.idCharacter == "Character2") {
9.          name = "Dimon";
10.     }
11.
12.     notificationText.text = "Player: " + pm.playerName
    + '\n' + "Character: " + name;
13.
14. }
```

Kode Sumber 4.5 Notifikasi hasil data pemain

4.2.2 Implementasi Antarmuka Simulasi Peliharaan

Antarmuka Simulasi Peliharaan adalah antarmuka yang tampil setelah pemain selesai memilih karakter atau menekan tombol "**Load Game**". Antarmuka ini digunakan untuk merealisasikan kasus penggunaan UC-001. Tampilan utama dari simulasi peliharaan ditunjukkan pada Gambar 4.7.

Gambar 4.7 Tampilan Utama Simulasi Peliharaan

Pada antarmuka simulasi peliharaan terdapat komponen *User Interface* (UI) informasi dari peliharan yang dimiliki sesuai dengan rancangan dari permainan simulasi. Komponen yang ditampilan adalah level peliharaan saat ini, karakter peliharaan yang dipilih, nama pemain, jumlah uang, kondisi tenaga, kondisi kesehatan, kondisi kebahagiaan dan kondisi kelaparan.

Pada deretan bawah antarmuka simulasi peliharaan terdapat beberapa tombol yang digunakan untuk melakukan aksi-aksi yang sudah dijelaskan pada tahap perancangan. Berikut ini adalah penjelasan lengkapnya dan implementasinya

1. **Tombol "Battle"**

Tombol "**Battle**" difungsikan untuk menuju ke halaman bertarung. Ada persyaratan berupa nilai kesehatan dan nilai kebahagiaan, jika untuk menuju ke halaman bertarung. Jika kondisi persyaratan tidak terpenuhi, maka tombol ini akan non-aktif. Implementasi dari tombol ini jika ditekan ditunjukkan pada Kode Sumber 4.6.

```
1.  public void BattleScene(){
```

```
2.      if (healthControl.GetHealth () > 10 && happyControl.Ge
    tHappy()>20 && sleepControl.GetSleep()>20)
3.          Application.LoadLevel("IntroBattleScene");
4.
5.  }
```

Kode Sumber 4.6 Tombol "Battle"

2. Tombol “Shop”

Tombol “**Shop**” difungsikan untuk menampilkan tampilan persediaan kebutuhan dari peliharaan. Pemain dapat membeli perlengkapan kehidupan dari peliharaan dan juga latihan untuk meningkatkan kemampuan peliharaan. Tampilan dari antarmuka awal menu shop ditunjukkan pada Gambar 4.8.

Gambar 4.8 Antarmuka "Shop"

Tombol “**Food**” difungsikan untuk menampilan tampilan pembelian konsumsi untuk peliharaan. Tampilan dari antarmuka penjualan konsumsi peliharaan ditunjukkan pada gambar 4.9.

Gambar 4.9 Antarmuka "Food Shop"

Tombol "**Medicine**" difungsikan menampilkan antarmuka pembelian obat-obatan dan minuman energi untuk peliharaan. Tampilan dari antarmuka *medicine* ditunjukkan pada Gambar 4.10.

Gambar 4.10 Antarmuka "Medicine Shop"

Tombol "**Training**" difungsikan untuk menampilkan antarmuka menaikkan kemampuan peliharaan. Tampilan dari antarmuka *training* ditunjukkan pada Gambar 4.11.

Gambar 4.11 Antarmuka "Training"

Untuk melakukan pembelian terdapat beberapa syarat yaitu uang yang dimiliki dan *item* yang didapatkan. Untuk membeli pemain perlu memilih *item* terlebih dulu. Karena item yang ada memiliki kelas yang sama, maka implementasinya juga sama. Implementasi untuk mendapatkan *item* yang dipilih diperlihatkan di Kode Sumber 4.7.

```
1.  public void ItemChoice(){
2.      myItem.typeItem = itemType;
3.      myItem.costMoney = price;
4.      myItem.impv = ability;
5.  }
```

Kode Sumber 4.7 Pemilihan "*Item*" di Shop

Setelah memilih *item* dengan menekan tombol “Buy”, lalu sistem akan melakukan pengecekan uang yang dimiliki. Implementasi dari pembelian *item* ditunjukkan pada Kode Sumber 4.8.

```
1.  public void Buy(bool flag){
2.      if (!flag) {
```

```
3.          if (moneyControl.GetCurrentMoney () - myItem.costM
    oney >= 0) {
4.           ActivatePanel5 (1);
5.          } else {
6.           ActivatePanel5 (2);
7.          }
8.      } else {
9.      moneyControl.SetCurrentMoney(moneyControl.GetCurrentMo
    ney () - myItem.costMoney);
10.     pManager.BuyItem(myItem.typeItem,myItem.impv);
11.     }
12. }
```

Kode Sumber 4.8 Pembelian *Item*

Saat pembelian dilakukan, sistem akan menampikan hasil notifikasi konfirmasi dari barang yang dipilih. Notifikasi yang muncul adalah setelah sistem mengonfirmasi bahwa terdapat cukup uang yang dimiliki pemain dan menanyakan untuk melanjutkan pembelian. Antarmuka notifikasi pembelian ditunjukkan pada Gambar 4.12.

Gambar 4.12 Notifikasi Pembelian Kondisi Berhasil

Jika sistem mengkonfirmasi uang yang dimiliki tidak cukup, maka sistem menampilkan notifikasi. Sehingga kembali ke tampilan shop sebelumnya. Antarmuka notifikasi dapat ditunjukkan pada Gambar 4.13.

Gambar 4.13 Notifikasi Pembelian Kondisi Gagal

Saat sistem menampilkan konfirmasi pembayaran, jika pemain memilih tombol "**Yes**", maka barang akan diolah untuk diterapkan kepada komponen-komponen yang bersangkutan sesuai dengan deskripsi dari *item*. Jika pemain memilih tombol "**No**", maka akan kembali ke antarmuka pembelian kembali. Implementasi dari penerapan pembelian item ditunjukkan pada Kode Sumber 4.9.

```
1.  public void BuyItem(string name, int value){
2.      if (name == "health") {
3.          healthControl.SetHealth (value);
4.      } else if (name == "level") {
5.          levelControl.SetLevel (value);
6.      } else if (name == "happy") {
7.          happyControl.SetHappy (value);
8.      } else if (name == "hungry") {
9.          hungryControl.SetHungry (value);
10.     } else if (name == "sleep") {
11.         sleepControl.SetSleep (value);
12.     } else if (name == "attack") {
13.         strengthControl.AddStrength(1);
14.     } else if (name == "defense") {
15.         defenseControl.AddDefense(1);
16.     } else if (name == "speed") {
17.         speedControl.AddSpeed(1);
18.     } else if (name == "hp") {
19.         hpControl.AddHP(1);
20.     }
21. }
```

Kode Sumber 4.9 Penerapan pembelian item

3. **Tombol "Skill"**

Tombol "**Skill"** difungsikan untuk menampilkan antarmuka pemilihan *skill* tambahan pada karakter. *Skill* dapat digunakan secara langsung pada saat bertarung dan memiliki dapat menambah kemampuan peliharaan. Perbedaaan antara penambahan skill dan training hanya dari cara mendapatkannya. Skill didapat dengan level yang diraih, sedangkan training menggunakan uang,. Antarmuka skill ditunjukkan pada Gambar 4.14.

Gambar 4.14 Antarmuka Skill

Jika ingin memilih skill yang diinginkan, maka pemain perlu menekan tombol "**USE**" sehingga skill akan digunakan secara langsung oleh karakter peliharaan. Skill yang dapat dipilih akan tersedia sesuai dengan level dari pemain. Untuk melihat skill yang digunakan dapat memilih menu info. Implementasi dari pemilihan skill yang telah dipilih ditunjukkan pada Kode Sumber 4.10.

```
1.  public void SetSkillName(){
2.      if(!locked)
3.          sChoice.setSkillName (skillName);
```

```
4.  }
```

Kode Sumber 4.10 Pemilihan Skill

4. Tombol "Info"

Tombol "**Info**" difungsikan untuk menampilkan data kemampuan dari karakter yang pemain miliki. Data kemampuan yang ditampulkan adalah yang memiliki kaitan dengan informasi pertarungan. Seperti jumlah kemenangan yang dimiliki (*Winner Point*), jumlah pertarungan yang telah dilakukan (*Experience Point*), kemampuan pertarungan peliharaan dan *skill* yang dimiliki. Antarmuka informasi karakter peliharaan ditunjukkan pada Gambar 4.15.

Gambar 4.15 Antarmuka Info Peliharaan

5. Tombol "Setting"

Tombol "**Setting**" difungsikan untuk mengatur lingkungan dari simulasi peliharaan. Pengaturan dapat berupa menambah atau mengurangi pencahayaan yang ada pada lingkungan ataupun menghidupkan/mematikan suara background musik. Tampilan dari antarmuka dari *setting* ditunjukkan pada Gambar 4.16.

Gambar 4.16 Antarmuka Setting

6. Tombol "Sleep"

Tombol "**Sleep**" difungsikan untuk memerintah peliharaan untuk tidur. Tampilan dari perubahan hewan peliharaan yang tidur ditunjukkan pada Gambar 4.17.

Gambar 4.17 Kondisi pemeliharaan tertidur

Dan implementasi dari perintah membuat peliharaan tertidur ditunjukkan pada Kode Sumber 4.11.

```
1.  public void ChangeSleep(){
2.      isSleep = !isSleep;
3.      if (isSleep) {
4.          brightControl.SleepMode ();
5.          sleepButton.color=new Color(0.5f,0.5f,0.5f,0.7f);

6.      }else{
7.          brightControl.LightBackMode();
8.          sleepButton.color=new Color(1f,1f,1f,1f);
9.      }
10.     player.GetComponent<SpriteRenderer> ().enabled = isSle
    ep;
11.     eye.SetActive (isSleep);
12. }
```

Kode Sumber 4.11 Perintah membuat tidur peliharaan

7. Tombol "Quit"

Tombol "**Quit**" difungsikan untuk menutup perangkat lunak sekaligus menyimpan data peliharaan pada saat ini. Implementasi dari menutup perangkat lunak diperlihatkan pada Kode Sumber 4.11.

```
1.  public void ExitGames(){
2.      Application.Quit ();
3.  }
4.  public void SaveCurrentAbility(){
5.      pInfo.time = timeControl.GetTime ();
6.      pInfo.level = levelControl.GetLevel ();
7.      pInfo.hungry = hungryControl.GetHungry ();
8.      pInfo.health = healthControl.GetHealth ();
9.      pInfo.happy = happyControl.GetHappy ();
10.     pInfo.money = moneyControl.GetCurrentMoney ();
11.     pInfo.battlePoint = expControl.GetExperience ();
12.     pInfo.winPoint = winPointControl.GetWinPoint ();
13.     pInfo.attackPoint = strengthControl.GetAttackPoint ();
14.     pInfo.healthPoint = hpControl.GetHP ();
15.     pInfo.speedPoint = speedControl.GetSpeed ();
16.     pInfo.evadePoint = defenseControl.GetDefensePoint ();
17.     pInfo.isSleeping = sleepControl.GetisSleep ();
18.     pInfo.sleep = sleepControl.GetSleep ();
```

```
19.     pInfo.attackPoint = strengthControl.GetAttackPoint ();
20.     pInfo.healthPoint = hpControl.GetHP ();
21.     pInfo.speedPoint = speedControl.GetSpeed ();
22.     pInfo.jumlahKuman = jmlKuman;
23.     saveLoadM.SavePlayerInformation ();
24. }
```

Kode Sumber 4.12 Menutup Game dan Menyimpan Data

4.2.3 Implementasi Permainan Simulasi Peliharaan

Pada bab perencanaan telah disebutkan beberapa komponen yang mempengaruhi jalannya kehidupan peliharaan yaitu kesehatan, tenaga, kelaparan dan kebahagiaan. Pada kontrol simulasi ini, akan menjalankan waktu yang ditentukan sistem untuk mempengaruhi komponen yang dimiliki oleh karakter peliharaan. Implementasi dari komponen kesehatan ditunjukkan pada Kode sumber 4.13, kelaparan pada Kode sumber 4.14, kebahagiaan pada Kode sumber 4.15 dan energy pada kode sumber 4.16.

```
1.  void FixedUpdate(){
2.      if (timeControl != null) {
3.          time = Mathf.Floor(timeControl.GetTime ());

4.          if(time!=timeDelta){
5.              if(healthControl.GetHealth()==0){
6.                  ignorePanel.SetActive(true);
7.                  // activate DeadPanel
8.                  menuControl.ActivatePanel3();
9.              }
10.
11.             if(time%10==1){
12.                 if(jmlKuman>5){
13.                     // healthy -1
14.                     healthControl.SubHealth();
15.                 }
16.             }
17.             if(time%20==1){
18.                 if(happyControl.GetHappy()<20||sleep
    Control.GetSleep()<10)
19.                         //healthy -1
```

```
                            healthControl.SubHealth();
            }

            if (time%30==1){
                //Spawn kuman
                Instantiate(moneySprite,transform.po
sition, transform.rotation);
                jmlKuman++;
            }
            timeDelta=time;
        }
    }
}
```

Kode Sumber 4.13 Kontrol Nilai Kesehatan

```
void FixedUpdate(){
    if (timeControl != null) {
        time = Mathf.Floor(timeControl.GetTime ());

        if(time!=timeDelta){
            if(time%20==1){
                    //hungry -1
                    hungryControl.SubHungry();
            }
            timeDelta=time;
        }
    }
}
```

Kode Sumber 4.14 Kontrol Nilai Kelaparan

```
void FixedUpdate(){
    if (timeControl != null) {
        time = Mathf.Floor(timeControl.GetTime ());

        if(time!=timeDelta){

            if(time%10==1){
                if(jmlKuman>5){
                    // happy -1
                    happyControl.SubHappy();
                }
            }
```

```
12.
13.             if (time%30==1){
14.                 // happy -1
15.                 happyControl.SubHappy();
16.                 if(hungryControl.GetHungry()<10)
17.                         //happy -1
18.                         happyControl.SubHappy();
19.             }
20.             timeDelta=time;
21.         }
22.     }
23. }
```

Kode Sumber 4.15 Kontrol Nilai Kebahagiaan

```
1.  void FixedUpdate(){
2.      if (timeControl != null) {
3.          time = Mathf.Floor(timeControl.GetTime ());

4.          if(time!=timeDelta){
5.              if(time%22==1 && sleepControl.GetisSleep
    ()){
6.                      // sleep +1
7.                      sleepControl.SetSleep(1);
8.              }
9.
10.             if(time%25==1 && !sleepControl.GetisSlee
    p() ){
11.                 // sleep -1
12.                 sleepControl.SubSleep();
13.                 if(hungryControl.GetHungry()<10)
14.                     // sleep -1
15.                     sleepControl.SubSleep();
16.             }
17.             timeDelta=time;
18.         }
19.     }
20. }
```

Kode Sumber 4.16 Kontrol Nilai Energi

Fungsi pada kontrol simulasi ini bergantung pada waktu yang dihitung, sehingga kontrol ini bisa berjalan dengan baik.

Implementasi pengaturan waktu yang digunakan pada kontrol simulasi ditunjukkan pada Kode sumber 4.17.

```
1.  public float GetTime(){
2.      time = (time+Time.deltaTime)% maxTime;
3.      return time;
4.  }
```

Kode Sumber 4.17 Kontrol waktu

4.2.4 Implementasi Karakter Peliharaan

Karakter peliharaan pada permainan ini berbasis 3D. Dalam *game* ini, terdapat 4 macam karakter yaitu Amon, Bimon, Cimon dan Dimon. Karakter tersebut memiliki perkembangan dari bayi karakter menjadi karakter dewasa. Berikut ini adalah implementasi dari karakter peliharaan.

1. Karakter "Amon"

Implementasi dari amon kecil (kiri) dan amon dewasa (kanan) ditunjukkan pada Gambar 4.18.

Gambar 4.18 Implementasi karakter "Amon"

2. Karakter "Bimon"

Implementasi dari bimon kecil (kiri) dan bimon dewasa (kanan) ditunjukkan pada Gambar 4.19.

Gambar 4.19 Implementasi karakter "Bimon"

3. Karakter "Cimon"

Implementasi dari cimon kecil (kiri) dan cimon dewasa (kanan) ditunjukkan pada Gambar 4.20.

Gambar 4.20 Implementasi karakter "Cimon"

4. Karakter "Dimon"

Implementasi dari dimon kecil (kiri) dan dimon dewasa (kanan) ditunjukkan pada Gambar 4.21.

Gambar 4.21 Implementasi karakter "Dimon"

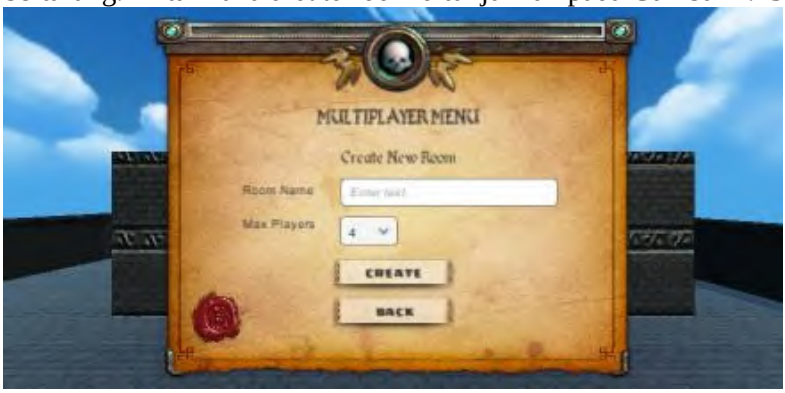
4.2.5 Implementasi Antarmuka Bertarung

Antarmuka **Bertarung** adalah antarmuka yang tampil setelah tombol "**Battle**" di antarmuka **Simulasi Peliharaan** ditekan. Antarmuka ini digunakan untuk melakukan kasus penggunan UC-002 yaitu bermain *multiplayer*. Terdapat beberapa pilihan pada pembuatan antarmuka bermain *multiplayer* ini yaitu *adventure* untuk melakukan pertarungan sendiri melawan *artificial intelligent*, *create room* dan *join room* untuk melakukan pertarungan secara online. Tampilan awal antarmuka bertarung ditunjukkan pada Gambar 4.22.

Gambar 4.22 Antarmuka awal bertarung

Pada pilihan *adventure* pengguna akan langsung menuju ke dalam lingkungan bertarung tanpa menggunakan koneksi internet, dimana permainan ini didesain untuk tutorial bagi pemain yang baru pertama kali memainkan *game* ini dan dimainkan secara *offline*. Sedangkan untuk *create room* dan *join room*, perlu ada tambahan informasi terlebih dahulu yang dijelaskan selanjutnya dan harus ada koneksi internet yang stabil.

Pada *create room*, pengguna perlu mengisikan nama unik dari *room* yang digunakan. Setelah selesai mengisi nama *room*, maka pemain dapat memilih tombol "**Create**" dan menuju ke halaman bertarung. Antarmuka *create room* ditunjukkan pada Gambar 4.23.



Gambar 4.23 Antarmuka *Create Room*

Berikut ini adalah implementasi dari membuat *room* beserta dari pengambilan informasi *room* yang akan dibuat ditunjukkan pada Kode Sumber 4.18.

```
1.  public void BattleStartOption1(bool flag){
2.      if (flag) {
3.          DeactiveAllPanel ();
4.          backgroundPanel.SetActive (false);
5.      } else {
6.          matchManager.JoinRoom (roomNameText.text, maxPlaye
    rs);
7.          ActivateLoadPanel();
8.          isWaiting = true;
```

```
9.          joinTime = 10f;
10.         stateWait = 21;
11.     }
12. }
13. void OnJoinedLobby(){
14.     if (roomName != null && PhotonNetwork.room==null) {
15.             RoomOptions roomOptions=new RoomOptions(){isVi
    sible=true,maxPlayers=Byte.Parse(maxPlayer.ToString())};
16.             PhotonNetwork.JoinOrCreateRoom(roomName,roomOp
    tions,TypedLobby.Default);
17.     }
18. }
```

Kode Sumber 4.18 Membuat *room*

Pada *join room* pemain perlu mencari nama dari *room* yang ingin dimasuki. Jika *room* yang diinginkan belum muncul pada dropdown, maka pengguna dapat melakukan menekan tombol *refresh*. Jika *room* dalam kondisi penuh, maka pengguna tidak dapat memasuki room tersebut. Tampilan antarmuka dari *join room* ditunjukkan pada Gambar 4.24.

Gambar 4.24 Antarmuka *join room*

Untuk kode program dari *join room*, sama dengan konsepnya dengan *create room*. Hal yang membedakan adalah

implementasi dari *refresh*. Berikut ini implementasi dari tombol " *refresh"* ditunjukkan pada Kode Sumber 4.19.

```
1.  public void ShowRoomList(){
2.      roomList = new string[100];
3.      roomList = matchManager.GetListRoom ();
4.      selectRoomDropdown.options.Clear ();
5.      foreach (string roomName in roomList) {
6.          selectRoomDropdown.options.Add(
7.                  new Dropdown.OptionData(roomName)
8.              );
9.      }
10.     selectRoomDropdown.captionText.text = roomList [0];
11. }
```

Kode Sumber 4.19 Memperbarui Room

Dan berikut ini adalah implementasi mendapatkan *room* yang terdaftar pada server photon ditunjukkan pada Kode Sumber 4.20.

```
1.  public string[] GetListRoom(){
2.      string[] roomListName=new string[100];
3.      int count = 0;
4.      foreach (RoomInfo game in PhotonNetwork.GetRoomList())
    {
5.          roomListName[count]=game.name;
6.          count++;
7.      }
8.      return roomListName;
9.  }
```

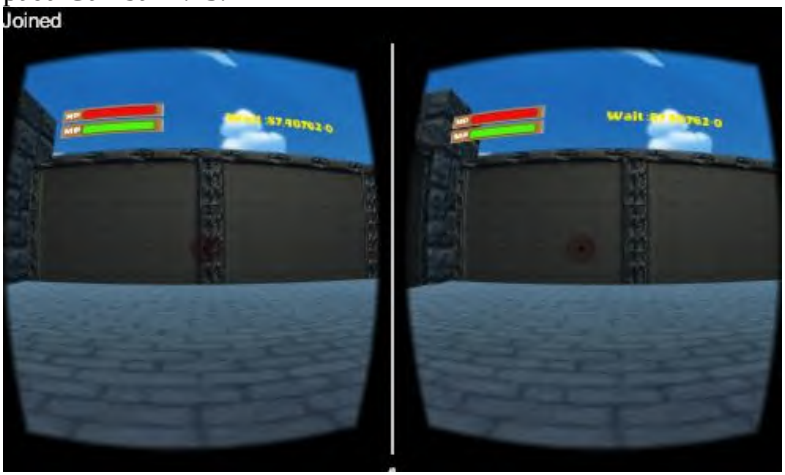
Kode Sumber 4.20 Mendapatkan Room

Pada saat pemain membuat *room*, sistem akan memberi waktu maksimal 30 detik untuk melakukan koneksi terhadap server photon. Jika gagal terhubung, maka akan kembali ke menu create *room*. Hal ini berlaku juga untuk *join room*. Pada waktu menunggu

tersebut, akan ditampilkan antarmuka *loading* agar pengguna tidak merasa jenuh menunggu.

4.2.6 Implementasi Realitas Virtual Pemain

Penggunaan realitas virtual didapatkan saat pemain sudah bergabung room dan sedang bertarung. Pemain perlu menggunakan *google cardboard*, sebab tampilan awal sudah membentuk dunia virtual. Sistem akan menampilkan informasi pengguna seperti *health point* yang berwarna merah, *magic point* yang berwarna hijau, Waktu bermain yang berada di pojok kanan atas dan target yang berwarna hitam ditengah layar. Tampilan informasi tersebut akan selalu mengikuti pengguna, baik dalam keadaan berjalan atau tidak. Jika pemain koneksinya tiba-tiba terputus, maka sistem akan menyelesaikan permainan. Dan pemain harus bermain di awal permainan kembali, Sehingga dapat dipastikan bahwa permainan pada *game* ini sangat memerlukan koneksi internet yang stabil. Tampilan antarmuka dari realitas virtual permainan bertarung pada *google cardboard* ditunjukkan pada Gambar 4.25.



Gambar 4.25 Antarmuka Bertarung Realitas Virtual

Setelah karakter terinstansiasi, sistem akan menampilkan antarmuka pemain, sehingga dapat melihat *bar* status HP, MP dan notifikasi waktu. Nilai status HP dan MP *default* dari tiap karakter yang telah dimiliki. Status ini dapat dilihat di informasi pada antarmuka simulasi peliharaan, sehingga sistem hanya akan mengambil nilai dari database. Implementasi dari sistem untuk menampilkan antarmuka pemain ditunjukkan pada Kode Sumber 4.21.

```
1.  healthSlider.value = healthPlayer.currentHitPoints;
2.  manaSlider.value = playerAttack.manaPoint;
3.  if (gM != null && TimeText != null && !PhotonNetwork.offli
    neMode) {
4.      if (gM.waitingTime > 0) {
5.          TimeText.color = Color.yellow;
6.          TimeText.text = "Wait :" + gM.waitingTime.ToString
    () + "-" + pm.printOut;
7.      } else if (gM.playTime > 0) {
8.          TimeText.color = Color.green;
9.          TimeText.text = "Play :" + gM.playTime.ToString ()
    + "-" + pm.printOut;
10.     } else {
11.         TimeText.color = Color.red;
12.         TimeText.text = "TimeOut";
13.     }
14. } else {
15.     TimeText.color=Color.red;
16.     TimeText.text="Unlimited";
17. }
```

Kode Sumber 4.21 Antarmuka Cardboard Pemain

Untuk setiap pemain lain yang terkoneksi memiliki antarmuka yang sama untuk lokal *device*-nya, akan tetapi antarmuka yang berbeda dengan pemain yang berbeda. Karakter pemain lain dimunculkan dengan menampilkan nilai health point diatasnya.

4.2.7 Implementasi Kontrol Karakter Bertarung

Pengendalian karakter peliharaan pada saat bertarung ada dua jenis, yaitu gerakan dan serangan. Perintah gerakan terdiri dari dua jenis yaitu gerakan kepala dengan *google cardboard* dan *accelerometer*. Untuk menggerakkan karakter peliharaan ke depan, pemain menggunakan sensor accelerometer dengan cara memutar *google cardboard* searah sumbu z. Sedangkan untuk merubah sudut pandang karakter, pemain menggunakan rotasi dengan google cardboard. Penjelasan lebih rinci tentang implementasi perintah gerakan peliharaan adalah sebagai berikut:

1. Accelerometer
   Accelerometer digunakan untuk menggerakkan karakter ke arah depan dengan kecepatan tertentu. Untuk melakukan gerakan ini dapat dilihat dari posisi kepala yang menggunakan *google cardboard*. Posisi kepala untuk kecepatan 0 atau tidak bergerak sama sekali adalah menghadap ke depan. Dan posisi kepala untuk maju adalah menunduk. Semakin menunduk, maka akan semakin cepat gerakan ke depan oleh karakter peliharaan.
2. Gerakan *Cardboard*
   Gerakan *cardboard* digunakan untuk melakukan rotasi karakter peliharaan atau merubah pandangan dari camera. Gerakan *cardboard* ini dilakukan dengan melakukan rotasi terhadap sumbu y. Besar putaran dari yang diperoleh adalah sesuai dengan besaran putaran dari *cardboard.*

Dari kedua gerakan yang dilakukan tersebut di implementasikan oleh sebuah fungsi pergerakan. Khusus pergerakan maju, dapat dilakukan jika waktu waiting time sudah selesai. Berikut ini adalah implementasi dari gerakan karakter ditunjukkan oleh Kode Sumber 4.22.

```
1.  Vector3 rotation1 = head.GetComponent<CardboardHead> ().ro
    tationY.eulerAngles;
2.  float rotationY = Mathf.PI * (rotation1.y / 180);
```

```
3.  float rotationX = Mathf.PI * (rotation1.x);
4.  float koordSide = Mathf.Sin (rotationY);
5.  float koordForward = Mathf.Cos (rotationY);
6.  float speedreduction = Mathf.Abs (Input.acceleration.z);
7.  if (playerController.isGrounded)
8.      vspeed = 0;
9.  vspeed -= gravity * Time.deltaTime;
10. movement = new Vector3 (koordSide, vspeed, koordForward);

11. direction = transform.rotation * movement;
12. if (direction.magnitude > 1) {
13.     direction = direction.normalized;
14. }
15. anim.SetFloat ("Speed", direction.magnitude);
16. playerController.Move (movement * (speed * speedreduction)
    * Time.deltaTime);
```

Kode Sumber 4.22 kontrol gerakan peliharaan

Perintah serangan dibagi menjadi 2 jenis, yaitu serangan normal (*Normal Attack*), dan serangan skill (*Special Attack*). Jika serangan mengenai karakter lawan, karakter lawan akan terkena *damage* yang membuat *Health Point* (HP) berkurang. Jumlah *damage* bergantung pada status kemampuan masing-masing karakter yang digunakan dan jenis serangan yang digunakan,. Penjelasan lebih rinci tentang perintah serangan adalah sebagai berikut:

1. Serangan Skill (*Special Attack*)
   Serangan jenis ini digunakan dengan cara menekan trigger yang ada di *google cardboard*. Serangan jenis ini dapat dilakukan saat MP berada pada nilai yang cukup atau lebih dari sama dengan 20 point. Serangan ini memerlukan 20 MP untuk sekali tembak. Serangan ini dapat mengurangi health point lawan sesuai dengan kemampuan attack yang dimiliki oleh karakter peliharaan.
2. Serangan Normal (*Normal Attack*)

Serangan jenis ini adalah serangan yang digunakan ketika jumlah magic point kita habis atau jumlah serangan skill habis. Kita dapat menggunakan serangan ini dengan cara yang sama dengan serangan skill yaitu dengan menggunakan trigger *google cardboard*. Jumlah efek *damage* yang didapatkan dari penggunaan serangan jenis ini adalah sebesar level karakter peliharaan yang kita miliki.

Untuk menghadapi serangan dari pemain tersebut, terdapat kemampuan karakter peliharaan yaitu *defense point* yang berfungsi untuk menghindari *damage* yang besar dari serangan skill. Nilai bertahan yang dimiliki peliharaan ditunjukkan oleh nilai di *Defense Point*. *Defense point* ini hanya muncul untuk serangan skill dan tidak berpengaruh pada serangan normal. Serangan yang dilakukan karakter akan berfungsi, jika permainan sudah dimulai atau *waiting time* sudah selesai. Implementasi dari kontrol serangan karakter peliharaan ditunjukkan pada Kode Sumber 4.23.

```
1.  if (h != null) {
2.      PhotonView pv = h.GetComponent<PhotonView> ();
3.      if (pv != null) {
4.          if(manaPoint>=20){
5.              int rand=0;
6.              if(p!=null)
7.                   rand= Random.Range(p.defensePoint/2,p.def
    ensePoint);
8.              pv.RPC ("TakeDamage", PhotonTargets.All, (dama
    ge-rand));
9.          }else{
10.             pv.RPC ("TakeDamage", PhotonTargets.All, defau
    ltDamage);
11.         }
12.     }
13. }
14. if (fxManager != null) {
15.     fxManager.GetComponent<PhotonView> ().RPC ("SnipperBul
    letFX", PhotonTargets.All, myAttackPoint.transform.positio
    n, hitPoint);
16.     if(manaPoint>19)
```

```
17.                     fxManager.GetComponent<PhotonView> ().
    RPC ("AttackEffect", PhotonTargets.All, myAttackPoint.tran
    sform.position, hitPoint,playerIdentity.specialAttack);
18.     }else
19.         Debug.Log("FX not FOund");
```

Kode Sumber 4.23 kontrol serangan peliharaan

Karakter peliharaan memiliki nilai health point yang saat melakukan pertarungan. Implementasi kontrol dari health point karakter peliharaan saat pertarungan ditunjukkan pada Kode Sumber 4.24.

```
1.  public void TakeDamage(int amd){
2.      currentHitPoints -= amd;
3.      if (currentHitPoints <= 0) {
4.          Die();
5.      }
6.  }
7.  void Die(){
8.      if (GetComponent<PhotonView> ().instantiationId
    == 0) {
9.          Destroy (gameObject);
10.     }else {
11.         if (GetComponent<PhotonView> ().isMine) {
12.             if (gameObject.tag == "Player") {
13.                 if(PhotonNetwork.offlineMode){
14.                     GameObject.FindObjectOfType<Sin
    gleManager>().LosePlayer();
15.                 }
16.                 else{
17.                     GameObject.FindObjectOfType<Mul
    tiplayerManager> ().LosePlayer ();
18.                 }
19.             }else{
20.                 GameObject.FindObjectOfType<SingleM
    anager>().AddBotDead();
21.             }
22.             PhotonNetwork.Destroy (gameObject);
23.         }
24.     }
```

```
25. }
```

Kode Sumber 4.24 kontrol *health point*

4.2.8 Implementasi Sistem Multiplayer

Untuk mengakomodasi fitur *multiplayer*, *game* ini menggunakan layanan *server* photon secara *online*. Dalam antarmuka *create room*, terjadi koneksi antara client dengan server untuk membuat *room*. Implementasi pembuatan *room* untuk *game* ini pada *server* photon ditunjukkan pada Kode Sumber 4.25.

```
1.  void OnJoinedLobby(){
2.          if (roomName != null && PhotonNetwork.room==null)
    {
3.              RoomOptions roomOptions=new RoomOptions(){isVi
    sible=true,maxPlayers=Byte.Parse(maxPlayer.ToString())};
4.              PhotonNetwork.JoinOrCreateRoom(roomName,roomOp
    tions,TypedLobby.Default);
5.          }
6.      }
```

Kode Sumber 4.25 Membuat Room

Setelah membuat *room* selesai dilakukan, sistem akan melakukan pengecekan terhadap jumlah pemain yang ada di *room*, sehingga pemain dapat mendapatkan nilai *spawn point*. Nilai ini digunakan untuk menginstansi karakter pada tempat terentu. Implementasi instansiasi karakter pada arena ditunjukkan pada Kode Sumber 4.26.

```
1.  void SpawnMyPlayer(){
2.      idSpawn = PhotonNetwork.playerList.Length;
3.      SpawnSpot mySpawn = spawnspot [idSpawn-1];
4.      GameObject player = PhotonNetwork.Instantiate (myMulti
    .namePlayer, mySpawn.transform.position, Quaternion.identi
    ty, 0);
```

```
5.      GameObject cameraPos = player.transform.FindChild("Cam
    eraPosition").gameObject;
6.      GameObject cardboard = Instantiate (CardboardCameraPre
    fab, cameraPos.transform.position, cameraPos.transform.rot
    ation) as GameObject;
7.      cardboard.transform.SetParent (player.transform);
8.  }
```

Kode Sumber 4.26 Instansi karakter peliharaan

Karakter akan disinkronisasi dengan menggunakan photon. Untuk sinkronisasi, yang diperhatikan adalah gerakan dan animasi yang akan dikirim kedalam server photon. Selain itu, pemain juga mengirimkan nilai health point yang dimiliki untuk menyamakan health point dari karakter pemain lain yang terinstansi di masing-masing lingkungan setiap pemain. Implementasi dari pengiriman data pemain dan sinkronisasi pemain ditunjukkan pada Kode Sumber 4.27.

```
1.  void OnPhotonSerializeView(PhotonStream stream, PhotonMess
    ageInfo info){
2.          playerHealth = GetComponent<Health> ();
3.          anim = GetComponent<Animator> ();
4.          if (stream.isWriting) {
5.              stream.SendNext (transform.position);
6.              stream.SendNext (transform.rotation);
7.              stream.SendNext (anim.GetFloat("Speed"));
8.              stream.SendNext (pIdentity.namePlayer);
9.              stream.SendNext (pIdentity.specialAttack);
10.             stream.SendNext (pIdentity.healthPoint);
11.             stream.SendNext (pIdentity.defensePoint);
12.             stream.SendNext (pIdentity.speed);
13.             stream.SendNext (pIdentity.levelPoint);
14.             if(playerHealth!=null)
15.                 stream.SendNext (playerHealth.currentHitPo
    ints);
16.         } else {
17.             correctPlayerPos=(Vector3)stream.ReceiveNext()
    ;
```

```
18.             correctPlayerRot=(Quaternion)stream.ReceiveNex
    t();
19.             anim.SetFloat("Speed",(float)stream.ReceiveNex
    t());
20.             pIdentity.namePlayer = (string)stream.ReceiveN
    ext();
21.             pIdentity.specialAttack =(int)stream.ReceiveNe
    xt();
22.             pIdentity.healthPoint = (int)stream.ReceiveNex
    t();
23.             pIdentity.defensePoint = (int)stream.ReceiveNe
    xt();
24.             pIdentity.speed = (float)stream.ReceiveNext();

25.             pIdentity.levelPoint = (int)stream.ReceiveNext
    ();
26.             if(playerHealth!=null)
27.                 playerHealth.currentHitPoints=(float)strea
    m.ReceiveNext();
28.         }
29.     }
```

Kode Sumber 4.27 Sinkronisasi player

4.2.9 Implementasi Permainan Bertarung

Setelah pemain yang karakternya telah terinstansiasi pada tempat bertarung. Sistem akan memberikan waktu untuk menunggu pemain lain memasuki *room* dengan *waiting time*. Pada saat waktu ini, pemain tidak dapat menyerang dan tidak dapat bergerak maju. Setelah *waiting time* habis, maka sistem akan melakukan pengecheckan terhadap jumlah pemain. Jika pemain sudah mencapai batas minimal, sistem akan menampilkan *play time* yaitu waktu yang digunakan untuk bermain. Implementasi dari waktu bermain ditunjukkan pada Kode Sumber 4.28

```
1.  void Timer(){
2.      if(playTime<=0){
3.          CheckMyPlayerCondition();
4.      }
```

```
5.      else if (playTime>0 && waitingTime < 0) {
6.          CountPlayer();
7.          if(!isPlayerCounter){
8.              isPlayerCounter=true;
9.          }
10.         playTime-=Time.deltaTime;
11.     } else {
12.         waitingTime-=Time.deltaTime;
13.     }
14. }
```

Kode Sumber 4.28 Implementasi waktu bermain

Pada saat permainan, apabila terdapat salah satu karakter yang nilai HP-nya nol atau habis karena terkena serangan dari lawan, maka pemain yang HP karakternya habis tersebut dinyatakan kalah dan pemain yang berhasil menghabiskan HP karakter lawan dinyatakan sebagai pemenang. Sedangkan jika permainan selesai, sistem akan melakukan pengecheckan terlebih dahulu nilai manakah yang Health Point yang terkecil. Nilai HP yang terkecillah yang kalah. Jika nilai HP sama, maka akan dikatakan pertarungan ini seri. Implementasi dari kontrol pertarungan ini untuk mengecek pemain yang menang dan kalah ditunjukkan pada Kode Sumber 4.29.

```
1.  void CheckMyPlayerCondition(){
2.      int countSamehealth = 0;
3.      float higherHealth=0;
4.      players=GameObject.FindGameObjectsWithTag("Player");
5.      GameObject winningPlayer=new GameObject();
6.
7.       foreach(GameObject player in players){
8.          if(player.GetComponent<CameraPlayer>().isActiveAnd
    Enabled){
9.              myPlayer=player;
10.         }
11.         if(player.GetComponent<Health>().currentHitPoints=
    =higherHealth && higherHealth >0){
```

```
12.             countSamehealth++;
13.         }
14.         if(player.GetComponent<Health>().currentHitPoints>
    higherHealth){
15.
16.             winningPlayer=player;
17.             higherHealth=player.GetComponent<Health>().cur
    rentHitPoints;
18.         }
19.     }
20.
21.     if(countSamehealth > 0 && myPlayer.GetComponent<Health
    >().currentHitPoints==higherHealth)
22.         DrawPlayer();
23.     else if (myPlayer.GetComponent<Health>().currentHitPoi
    nts==higherHealth)
24.         WinPlayer ();
25.     else
26.         LosePlayer ();
27. }
```

Kode Sumber 4.29 Kontrol pengecekan kondisi pemain

Setelah permainan selesai, sistem akan menampilkan notifikasi atau *popup* "**GAME OVER**" yang menyatakan bahwa permainan telah selesai dan sambungan kedua pemain dengan *server* Photon akan diputus. Sistem menampilan notifikasi yang berbeda antara pemenang dan kalah. Setelah itu, pemain dapat memilih kembali untuk menuju ke antarmuka "Battle Menu". Setelah masuk ke antarmuka **Battle Menu**, kedua pemain bermain lagi. Tampilan notifikasi "*game over*" ditunjukkan pada Gambar 4.26 dan implementasi dari *game over* ditunjukkan pada Kode Sumber 4.30.

Gambar 4.26 Notifikasi Game Over

```
1.  void DeadPlayer(){
2.      mmManager.Disconnect ();
3.      mmManager.RefreshData ();
4.      isPlaying=false;
5.      setPlay (false);
6.      mmManager.MainCamera.SetActive(true);
7.  }
8.  public void WinPlayer(){
9.      DeadPlayer ();
10.     bmManager.NotifText("You WIN'\n'+25 Coin");
11.     GameObject.FindObjectOfType<PlayerInfo> ().money += 25
    ;
12.     bmManager.ActivatePanel3();
13. }
14.
15. public void DrawPlayer(){
16.     DeadPlayer ();
17.     bmManager.NotifText("You Draw'\n'+10 Coin");
18.     GameObject.FindObjectOfType<PlayerInfo> ().money += 10
    ;
19.     bmManager.ActivatePanel3();
20. }
21.
22. public void LosePlayer(){
23.     DeadPlayer ();
24.     bmManager.NotifText("You WIN'\n'+5 Coin");
```

```
25.     GameObject.FindObjectOfType<PlayerInfo> ().money += 5;

26.     bmManager.NotifText("You Lose");
27.     bmManager.ActivatePanel3();
28. }
```

Kode Sumber 4.30 Menampilkan Notifikasi Game Over

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan menggunakan metode *black-box* berdasarkan skenario yang telah ditentukan dan pengujian dilakukan dengan survei langsung kepada pengguna.

5.1 Lingkungan Uji Coba

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 ditunjukkan pada Tabel 5.1, Tabel 5.2 dan Tabel 5.3.

Tabel 5.1 Perangkat Lunak Ujicoba (1)

Perangkat Keras	Prosesor: 4x ARM Cortex-A7 @ 1.18GHz Memori: 921MB RAM
Perangkat lunak	Sistem Operasi: Android Versi 4.4.2

Tabel 5.2 Perangkat Lunak Ujicoba (2)

Perangkat Keras	Prosesor: Quad Core Cortex-A9 @1.6GHz Memori: 2GB RAM
Perangkat lunak	Sistem Operasi: Android Versi 4.3

Tabel 5.3 Perangkat Lunak Ujicoba (3)

Perangkat Keras	Prosesor: Quad Core Cortex-A53 @1.7GHz && Quad Core Cortex-A53 @1.0GHz Memori: 2GB RAM
Perangkat lunak	Sistem Operasi: Android Versi 5.0.2

5.2 Skenario Pengujian Fungsionalitas

Pengujian ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja semestinya. Selain itu, langkah ini ditujukan untuk mengetahui kesesuaian keluaran dari setiap tahapan atau langkah penggunaan fitur terhadap skenario yang dipersiapkan. Pengujian dilakukan dengan menggunakan metode *black-box.*

5.2.1 Skenario Pengujian Simulasi Peliharaan

Skenario pengujian simulasi peliharaan digunakan untuk mengetahui apakah fungsionalitas simulasi peliharaan sudah berjalan sesuai dengan apa yang diharapkan. Skenario ini ditunjukkan pada Tabel 5.4.

Tabel 5.4 Pengujian Simulasi Peliharaan

Nama Use Case	UC-001
Kondisi Awal	Pemain memasuki main menu dari perangkat lunak dan telah memilih karakter peliharaan
Prosedur Pengujian	Pemain dapat pembelian item di shop (makanan, medicine, training), dapat melihat informasi dari kondisi karakter, menambah skill karakter, melihat informasi dari kemampuan karakter.
Hasil yang diharapkan	Pemain dapat menjalankan simulasi peliharaan dengan fungsionalitas utama aplikasi berjalana dengan baik tanpa adanya bug pada aplikasi
Hasil yang diperoleh	Pemain dapat memelihara dari karakter peliharan dan melihat perkembangannya melalui user interface.

Kesimpulan.	Pengujian berhasil

Berikut ini adalah hasil dari pengujian untuk melihat perubahan karakter peliharaan. Terlihat bahwa nilai kesehatan, nilai kebahagiaan, nilai energi dan nilai kelaparan terdapat perubahan, seperti pada Gambar 5.1. Perubahan dari nilai kesehatan akan terus menurun, sehingga jika tidak diberi obat,maka akan membuat karakter mati seperti yang ditunjukkan pada Gambar 5.2. Larakter peliharaan yang mati tidak dapat digunakan kembali, sehingga pemain perlu melakukan pemilihan karakter peliharaan kembali. Selanjutnya adalah kondisi peliharaan berubah menjadi dewasa, seperti ditunjukkan pada Gambar 5.3.

Gambar 5.1 Perubahan komponen karakter

Gambar 5.2 Peliharaan Meninggal

Gambar 5.3 Pertumbuhan karakter Peliharaan

5.2.1.1 PF-01: Pengujian Simulasi Peliharaan

Pengujian dimulai ketika pemain memasuki antarmuka **Simulasi Peliharaan**. Pemain dapat menjalankan perawatan peliharaan diantaranya adalah memberi makan, memberi obat, memberi latihan untuk meningkatkan kemampuan karakter, menambah skill yang dimiliki, melihat informasi kemampuan karakter, menidurkan karakter dan sebagainya.

5.2.2 Skenario Pengujian Permainan Multiplayer

Skenario pengujian permainan digunakan untuk mengetahui apakah fungsionalitas bermain sudah berjalan sesuai dengan apa yang diharapkan. Skenario ini ditunjukkan pada Tabel 5.5.

Tabel 5.5 Pengujian Permainan Multiplayer

Nama Use Case	**UC-001**
Kondisi Awal	Pengguna berada pada antarmuka **Multiplayer**
Prosedur Pengujian	Pemain dapat membuat *room* yang diinginkan, selanjutnya pemain memilih tombol *create room* lalu menuju ke lingkungan bertarung. Pemain lain, dapat bergabung dengan menggunakan *join room*. Lalu memilih *room* yang sudah ada.
Hasil yang diharapkan	Pemain dapat menjalankan fungsionalitas *multiplayer* bertarung dengan mudah
Hasil yang diperoleh	Pemain dapat memainkan *multiplayer* dengan prosedur yang telah disebutkan.

Kesimpulan.	Pengujian berhasil

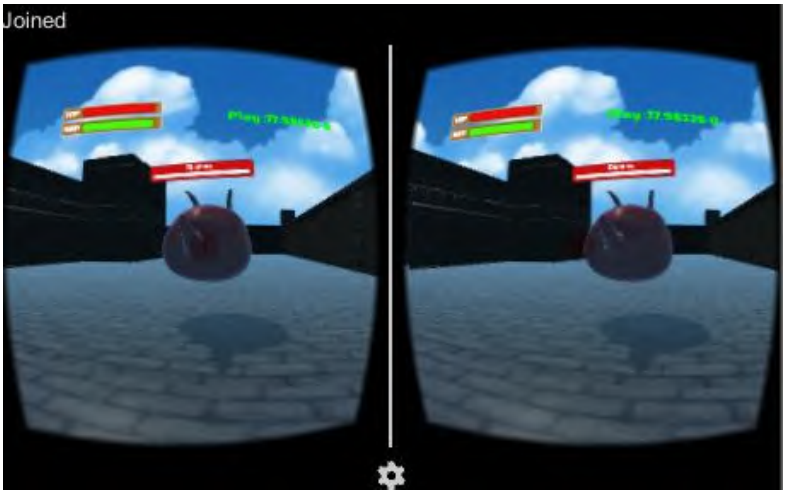
5.2.2.1 PF-02: Pengujian Permainan Multiplayer

Pengujian dimulai ketika pemain memasuki antarmuka **Battle** lalu pemain memilih *create room* atau *join room.* Salah satu pemain dapat memilih *create room* dan pemain lainnya dapat memilih untuk *join room*. Setelah selesai sinkronisasi antar *room*, pemain akan ditampilkan di lingkungan virtual dan dapat melihat karakter peliharaan pemain masing-masing. Hasil dari percobaan dapat dilihat pada Gambar 5.4 . Bahwa 2 karakter sudah berhasil tersinkronisasi dan di instansi di *room* yang sama.

Gambar 5.4 Pengujian *Multiplayer*

5.2.2.2 PF-03: Pengujian Penggunaan Realitas Virtual

Pada pengujian ini, pemain dapat menggunakan google *google cardboard* sebagai media realitas virtual pendukung *game* ini saat dalam melakukan pertarungan. Pemain dapat melihat tampilan realitas virtual saat melakukan pertarungan. Pemain dapat menggunakan trigger sebagai kontrol menembak, gerakan kepala untuk berotasi dan accelerometer/gerakan kepala menunduk dan menengadah untuk mengatur kecepatan karakter peliharaan. Tampilan player dalam kondisi realitas virtual ditunjukkan pada Gambar 5.5.



Gambar 5.5 Pengujian *Cardboard Multiplayer*

5.2.2.3 Hasil Pengujian Fungsional

Subbab ini berisi tentang hasil pengujian fungsionalitas yang sudah dilakukan berdasarkan pada PF01, PF02, dan PF03. Tiga pengujian yang telah dilakukan menunjukkan bahwa semua fungsionalitas permainan berjalan dengan baik dan sesuai dengan apa yang diharapkan serta sesuai dengan skenario dan alur yang telah dibuat pada perancangan. Rekapitulasi hasil pengujian fungsionalitas ditunjukkan pada Tabel 5.6.

Tabel 5.6 Rekapitulasi hasil pengujian fungsional

No	Kode Pengujian	Hasil Pengujian
1.	PF01	Berhasil
2.	PF02	Berhasil
3.	PF03	Berhasil

5.3 Pengujian Pengguna

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

5.3.1 Skenario Uji Coba Pengguna

Dalam melakukan pengujian perangkat lunak, penguji diminta untuk mencoba menggunakan perangkat lunak yang bersangkutan untuk mencoba semua fungsionalitas dan fitur yang tersedia

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

Jumlah pengguna yang terlibat dalam pengujian perangkat lunak sebanyak tiga orang. Dalam melakukan pengujian, pengguna melakukan percobaan lebih dari satu kali penggunaan untuk masing-masing pengguna.

Dalam memberikan penilaian dan tanggapan, penguji diberikan formulir pengujian perangkat lunak. Formulir pengujian perangkat lunak ini memiliki beberapa aspek penilaian dan pada bagian akhir terdapat saran untuk perbaikan fitur.

5.3.2 Daftar Penguji Perangkat lunak

Pada subbab ini ditunjukkan daftar pengguna yang bertindak sebagai penguji coba perangkat lunak yang dibangun.

Daftar nama penguji perangkat lunak ini ditunjukkan pada Tabel 5.7.

Tabel 5.7 Daftar Nama Penguji Coba Perangkat lunak

Nomor	Nama	Pekerjaan
1	Aditya Putra Ferza	Mahasiswa Teknik Informatika ITS
2	Angga Saputra Dwi W.	Mahasiswa Teknik Informatika ITS
3	Dimas Widdy	Mahasiswa Teknik Informatika ITS
4	Hafieludin Yuzuf	Mahasiswa Teknik Informatika ITS
5	Wahyu Widyananda	Mahasiswa Teknik Informatika ITS

5.3.3 Hasil Uji Coba Pengguna

Uji coba yang dilakukan terhadap beberapa pengguna memiliki beberapa aspek yang dipisahkan berdasarkan antarmuka, permainan simulasi, penggunaan realitas virtual, penambahan fitur *multiplayer* dan keseluruhan performa sistem. Sistem penilaian didasarkan pada skala penghitungan satu sampai lima di mana skala satu menunjukkan nilai terendah dan skala lima menunjukkan skala tertinggi. Skala dan katergori penilaian ditunjukkan pada Tabel 5.8. Penilaian akhir dilakukan dengan menghitung berapa banyak penguji yang memilih suatu skala tertentu kemudian dihitung nilai rata-rata dari nilai-nilai tersebut. Hasil uji coba ditunjukkan pada subbab-subbab berikut secara lengkap dengan disertai tabel.

Tabel 5.8 Skala dan kategori penilaian

Nilai	Keterangan
1	Sangat Buruk
2	Buruk
3	Cukup Buruk
4	Cukup Baik
5	Baik
6	Sangat Baik

5.3.3.1 Hasil Penilaian Antarmuka

Penilaian antarmuka difokuskan pada penilaian pengguna terhadap kemudahan antarmuka dan fitur-fitur lain dalam menjalani permainan. Penilaian dari antarmuka ini juga membuktikan kemudahan pemahaman cara bermain game ini. Hasil penilaian pengguna terhadap antarmuka permainan simulasi ditunjukkan pada Tabel 5.9, sedangkan hasil penilaian dari antarmuka permainan bertarung ditunjukkan pada Tabel 5.10.

Tabel 5.9 Penilaian Antarmuka Simulasi

No	Task	Penilaian						Rata-Rata
		1	2	3	4	5	6	
1	Kenyamanan antarmuka	0	0	1	2	0	2	4.6
2	Kemudahan permainan	0	0	0	4	1	0	4.2
							Nilai Akhir	4.4

Tabel 5.10 Penilaian Antarmuka Bertarung

No	Task	Penilaian						Rata-Rata
		1	2	3	4	5	6	
1	Kenyamanan antarmuka	0	0	1	3	0	1	4.2
2	Kemudahan permainan	0	0	0	3	2	0	4.4
Nilai Akhir								4.3

5.3.3.2 Hasil Penilaian Simulasi Peliharaan

Penilaian simulasi peliharaan difokuskan pada penilaian pengguna terhadap kelengkapan dari fungsi simulasi peliharaan dan kemiripan dengan aktivitas di dunia nyata. Hasil penilaian pengguna terhadap simulasi peliharaan ditunjukkan pada Tabel 5.11.

Tabel 5.11 Penilaian Simulasi Peliharaan

No	Task	Penilaian						Rata-Rata
		1	2	3	4	5	6	
1	Kesesuaian simulasi dengan dunia nyata	0	0	0	3	2	0	4.4
2	Kelengkapan Fitur Perawatan	0	0	0	2	3	0	4.6
Nilai Akhir								4.5

5.3.3.3 Hasil Penilaian Realitas Virtual

Penilaian realitas virtual difokuskan pada kemanfaatan yang didapatkan dari inovasi penggunaan realitas virtual dalam permainan pertarungan dan perbandingan tanpa terhadap tanpa penggunaannya. Hasil penilaian pengguna terhadap realitas virtual pertarungan ditunjukkan pada Tabel 5.12.

Tabel 5.12 Penilaian Realitas Virtual

No	Task	Penilaian						Rata-Rata
		1	2	3	4	5	6	
1	Kenyamanan saat bertarung dengan *cardboard*	0	0	0	1	4	0	4.8
2	Kenyamanan saat bertarung tanpa *cardboard* atau *touch*	0	1	0	2	1	1	4.2
3	Suasana permainan virtual	0	0	0	1	2	2	5.2
Nilai Akhir								4.73

5.3.3.4 Hasil Penilaian Multiplayer

Penilaian *multiplayer* difokuskan pada penilaian pengguna terhadap performa *multiplayer* pada saat melakukan pertarungan. Hasil penilaian pengguna terhadap *multiplayer* ditunjukkan pada Tabel 5.13.

Tabel 5.13 Penilaian Multiplayer

No	Task	Penilaian						Rata-Rata
		1	2	3	4	5	6	
1	Performa *Multiplayer*	0	0	1	1	1	2	4.8
Nilai Akhir								4.8

5.3.3.5 Hasil Penilaian Performa Sistem

Penilaian performa sistem ini adalah penilaian yang ditunjukkan untuk melihat kesempuranaan dan diterimanya bentuk tugas akhir dari pengguna, sehingga layak untuk dikembangkan kembali. Hasil penilaian pengguna terhadap penilaian performa sistem ditunjukkan pada Tabel 5.14.

Tabel 5.14 Penilaian Performa Sistem

No	Task	Penilaian						Rata-Rata
		1	2	3	4	5	6	
1	Kombinasi dari simulasi, realitas virtual dan multiplayer	0	0	0	3	1	1	4.6
2	Kombinasi kontrol simulasi dan bertarung	0	0	1	1	3	0	4.4
3	Game ini menghibur	0	0	0	1	2	2	5.2
Nilai Akhir								4.73

5.3.4 Analisis Hasil Pengujian

Dari hasil pengujian dengan rentang nilai 1-6 dapat dianalisa sebagai berikut:

Penilaian antarmuka terdiri dari 2 bagian yaitu antarmuka simulasi dan antarmuka bertarung *multiplayer*. Pada antarmuka simulasi memperoleh hasil cukup baik dengan nilai akhir 4.4 dan dapat disimpulkan bahwa permainan simulasi ini cukup nyaman dan mudah dimengerti. Pada antarmuka bertarung *multiplayer* memperoleh hasil cukup baik dengan nilai 4.3 dan dapat disimpulkan bahwa permainan bertarung *multiplayer* ini cukup nyaman dan mudah dimengerti.

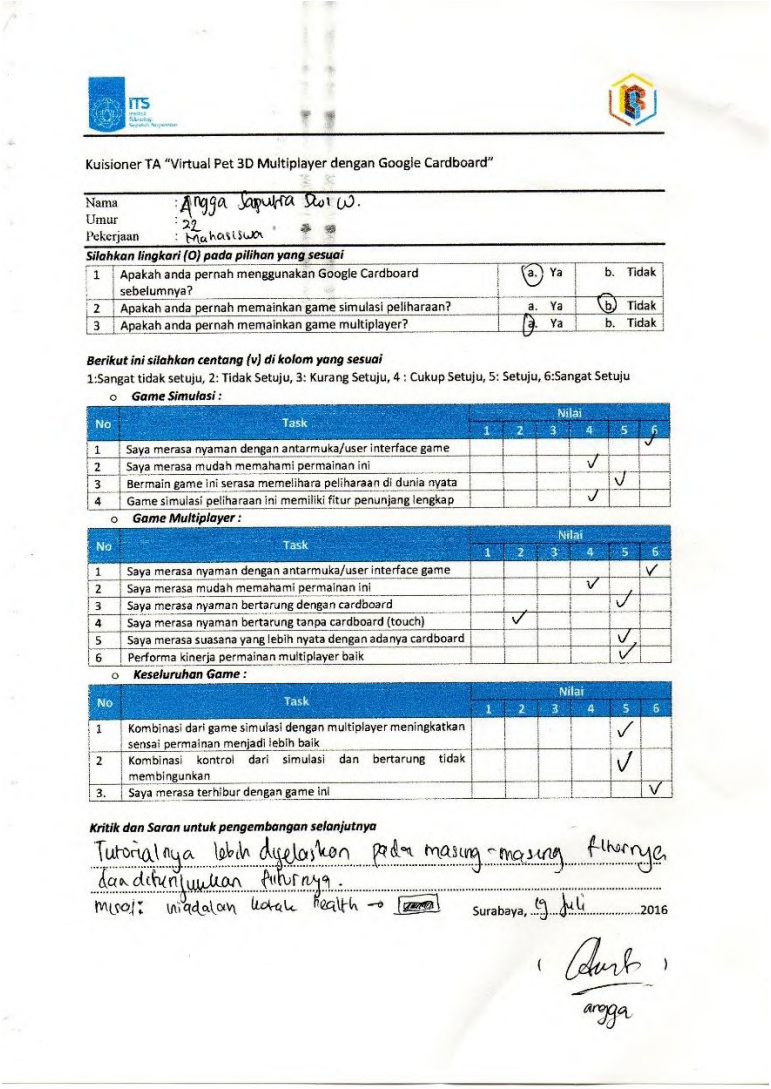
Pada penilaian simulasi peliharaan memperoleh hasil cukup baik dengan nilai akhir 4.5. Hasil ini membuktikan bahwa permainan sudah sesuai dengan perawatan hewan didunia nyata dan memiliki fitur yang lengkap.

Pada penilaian penyajian realitas virtual memperoleh hasil baik dengan nilai akhir 4.73. Hasil ini membuktikan bahwa nuansa realitas virtual didapatkan oleh pengguna dengan menggunakan *google cardboard*. Hasi ini juga membandingkan nilai menggunakan *google cardboard* yang lebih tinggi daripada menggunakan touch dan didapatkan kesimpulan bahwa penggunaan *google cardboard* sangatlah membantu meningkatkan kenyamanan permainan.

Pada penilaian performa *multiplayer* memperoleh hasil baik dengan nilai akhir 4.8. Hasil ini membuktikan bahwa aplikasi sudah berhasil mengelola sistem *multiplayer* dan mengatasi permasalahan koneksi pada permainan.

Pada penilaian performa sistem keseluruhan didapatkan hasil yang baik dengan nilai akhir 4.73. Dari hasil penilaian kombinasi antara penggunaan kontrol dari permainan simulasi dan permainan bertarung didapatkan nilai 4.4 yang berarti cukup baik, sehingga kombinasi kontrol dari 2 permainan tersebut masih bisa dimengerti oleh pengguna dan dapat diterapkan. Dari hasil akhir ini membuktikan bahwa aplikasi ini layak untuk diterapkan pada pasar industri *game*.

LAMPIRAN



Kuisioner TA "Virtual Pet 3D Multiplayer dengan Google Cardboard"

Nama : Angga Saputra Dwi W.
Umur : 22
Pekerjaan : 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 game simulasi peliharaan?	a. Ya	(b.) Tidak
3	Apakah anda pernah memainkan game multiplayer?	(a.) Ya	b. Tidak

Berikut ini silahkan centang (v) di kolom yang sesuai

1:Sangat tidak setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4 : Cukup Setuju, 5: Setuju, 6:Sangat Setuju

- ***Game Simulasi :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game						✓
2	Saya merasa mudah memahami permainan ini				✓		
3	Bermain game ini serasa memelihara peliharaan di dunia nyata					✓	
4	Game simulasi peliharaan ini memiliki fitur penunjang lengkap				✓		

- ***Game Multiplayer :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game						✓
2	Saya merasa mudah memahami permainan ini				✓		
3	Saya merasa nyaman bertarung dengan cardboard					✓	
4	Saya merasa nyaman bertarung tanpa cardboard (touch)		✓				
5	Saya merasa suasana yang lebih nyata dengan adanya cardboard					✓	
6	Performa kinerja permainan multiplayer baik					✓	

- ***Keseluruhan Game :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Kombinasi dari game simulasi dengan multiplayer meningkatkan sensai permainan menjadi lebih baik					✓	
2	Kombinasi kontrol dari simulasi dan bertarung tidak membingunkan					✓	
3.	Saya merasa terhibur dengan game ini						✓

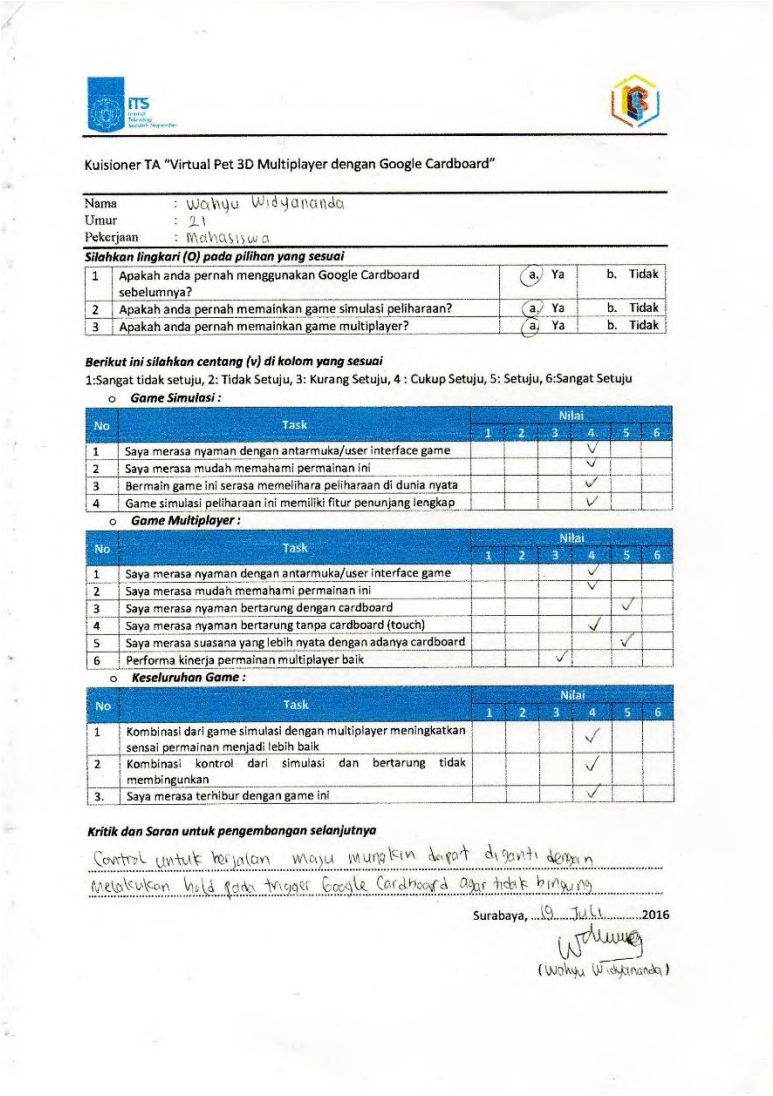
Kritik dan Saran untuk pengembangan selanjutnya

Tutorialnya lebih dijelaskan pada masing-masing fiturnya, dan ditunjukkan fiturnya.
misal: imadalam Mode Health → [game]

Surabaya, 19 Juli 2016

(Angga)

Gambar 8.1 Kuisioner Angga



Kuisioner TA "Virtual Pet 3D Multiplayer dengan Google Cardboard"

Nama : Wahyu Widyananda
Umur : 21
Pekerjaan : 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 game simulasi peliharaan?	(a.) Ya	b. Tidak
3	Apakah anda pernah memainkan game multiplayer?	(a.) Ya	b. Tidak

Berikut ini silahkan centang (v) di kolom yang sesuai

1:Sangat tidak setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4 : Cukup Setuju, 5: Setuju, 6:Sangat Setuju

- ***Game Simulasi :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game				✓		
2	Saya merasa mudah memahami permainan ini				✓		
3	Bermain game ini serasa memelihara peliharaan di dunia nyata				✓		
4	Game simulasi peliharaan ini memiliki fitur penunjang lengkap				✓		

- ***Game Multiplayer :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game				✓		
2	Saya merasa mudah memahami permainan ini				✓		
3	Saya merasa nyaman bertarung dengan cardboard					✓	
4	Saya merasa nyaman bertarung tanpa cardboard (touch)				✓		
5	Saya merasa suasana yang lebih nyata dengan adanya cardboard					✓	
6	Performa kinerja permainan multiplayer baik			✓			

- ***Keseluruhan Game :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Kombinasi dari game simulasi dengan multiplayer meningkatkan sensai permainan menjadi lebih baik				✓		
2	Kombinasi kontrol dari simulasi dan bertarung tidak membingunkan				✓		
3.	Saya merasa terhibur dengan game ini				✓		

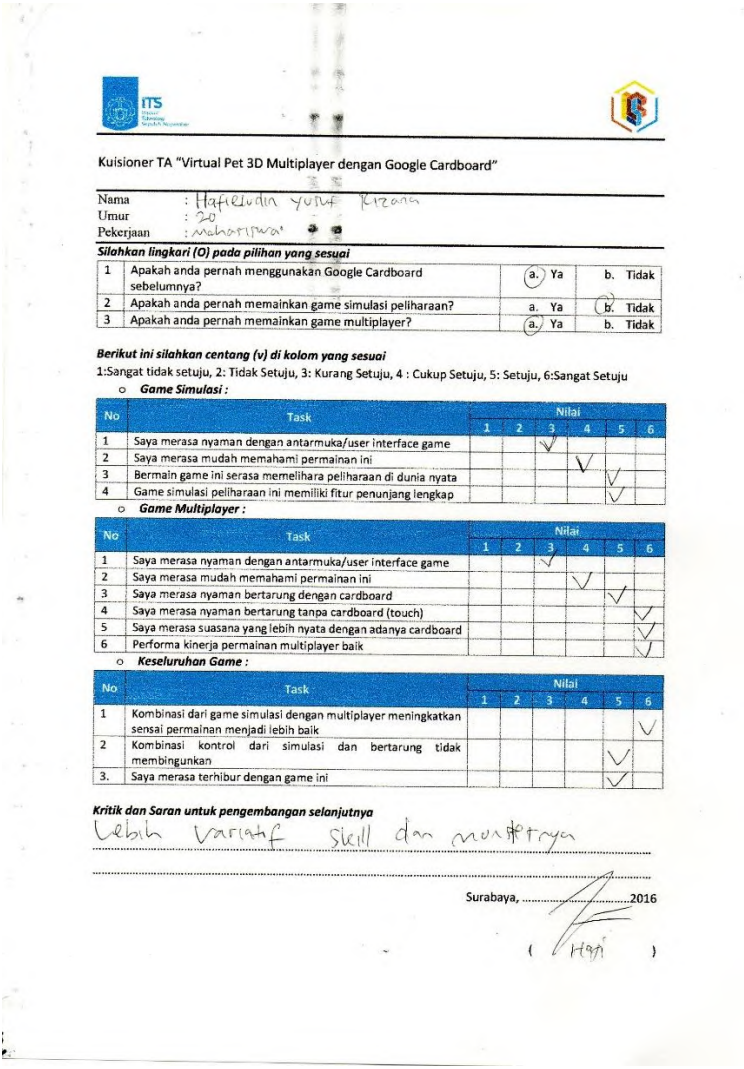
Kritik dan Saran untuk pengembangan selanjutnya

Control untuk berjalan maju mungkin dapat diganti dengan melakukan hold pada trigger Google Cardboard agar tidak bingung

Surabaya, ...19.....Juli.....2016

(Wahyu Widyananda)

Gambar 8.2 Kuisioner Wahyu



Kuisioner TA "Virtual Pet 3D Multiplayer dengan Google Cardboard"

Nama : Hafieludin Yusuf Rizaldi
Umur : 20
Pekerjaan : 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 game simulasi peliharaan?	a. Ya	(b.) Tidak
3	Apakah anda pernah memainkan game multiplayer?	(a.) Ya	b. Tidak

Berikut ini silahkan centang (v) di kolom yang sesuai

1:Sangat tidak setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4 : Cukup Setuju, 5: Setuju, 6:Sangat Setuju

- ***Game Simulasi :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game			V			
2	Saya merasa mudah memahami permainan ini				V		
3	Bermain game ini serasa memelihara peliharaan di dunia nyata					V	
4	Game simulasi peliharaan ini memiliki fitur penunjang lengkap					V	

- ***Game Multiplayer :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game			V			
2	Saya merasa mudah memahami permainan ini				V		
3	Saya merasa nyaman bertarung dengan cardboard					V	
4	Saya merasa nyaman bertarung tanpa cardboard (touch)						V
5	Saya merasa suasana yang lebih nyata dengan adanya cardboard						V
6	Performa kinerja permainan multiplayer baik						V

- ***Keseluruhan Game :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Kombinasi dari game simulasi dengan multiplayer meningkatkan sensai permainan menjadi lebih baik						V
2	Kombinasi kontrol dari simulasi dan bertarung tidak membingunkan					V	
3.	Saya merasa terhibur dengan game ini					V	

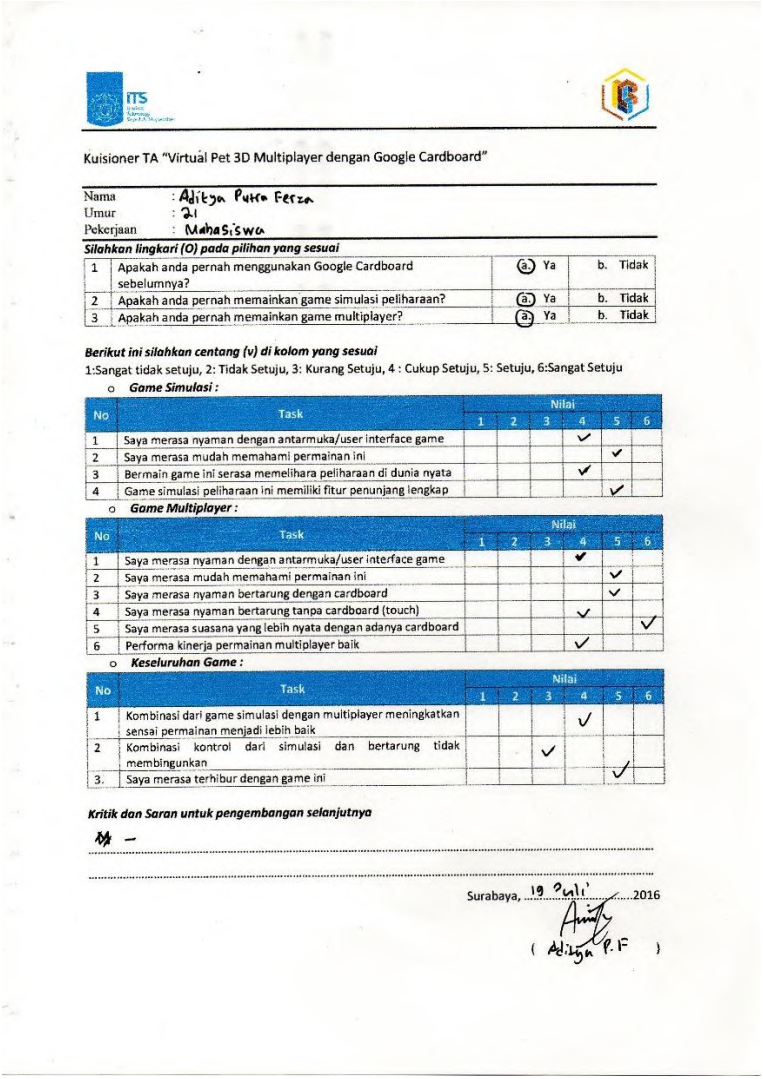
Kritik dan Saran untuk pengembangan selanjutnya

Lebih variatif skill dan monsternya

Surabaya, ...2016

(Hafi)

Gambar 8.3 Kuisioner Hafieludin



Kuisioner TA "Virtual Pet 3D Multiplayer dengan Google Cardboard"

Nama : Aditya Putra Ferza
Umur : 21
Pekerjaan : 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 game simulasi peliharaan?	(a.) Ya	b. Tidak
3	Apakah anda pernah memainkan game multiplayer?	(a.) Ya	b. Tidak

Berikut ini silahkan centang (v) di kolom yang sesuai

1:Sangat tidak setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4 : Cukup Setuju, 5: Setuju, 6:Sangat Setuju

- ***Game Simulasi :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game				✓		
2	Saya merasa mudah memahami permainan ini					✓	
3	Bermain game ini serasa memelihara peliharaan di dunia nyata				✓		
4	Game simulasi peliharaan ini memiliki fitur penunjang lengkap					✓	

- ***Game Multiplayer :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game				✓		
2	Saya merasa mudah memahami permainan ini					✓	
3	Saya merasa nyaman bertarung dengan cardboard					✓	
4	Saya merasa nyaman bertarung tanpa cardboard (touch)				✓		
5	Saya merasa suasana yang lebih nyata dengan adanya cardboard						✓
6	Performa kinerja permainan multiplayer baik				✓		

- ***Keseluruhan Game :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Kombinasi dari game simulasi dengan multiplayer meningkatkan sensai permainan menjadi lebih baik				✓		
2	Kombinasi kontrol dari simulasi dan bertarung tidak membingunkan			✓			
3.	Saya merasa terhibur dengan game ini					✓	

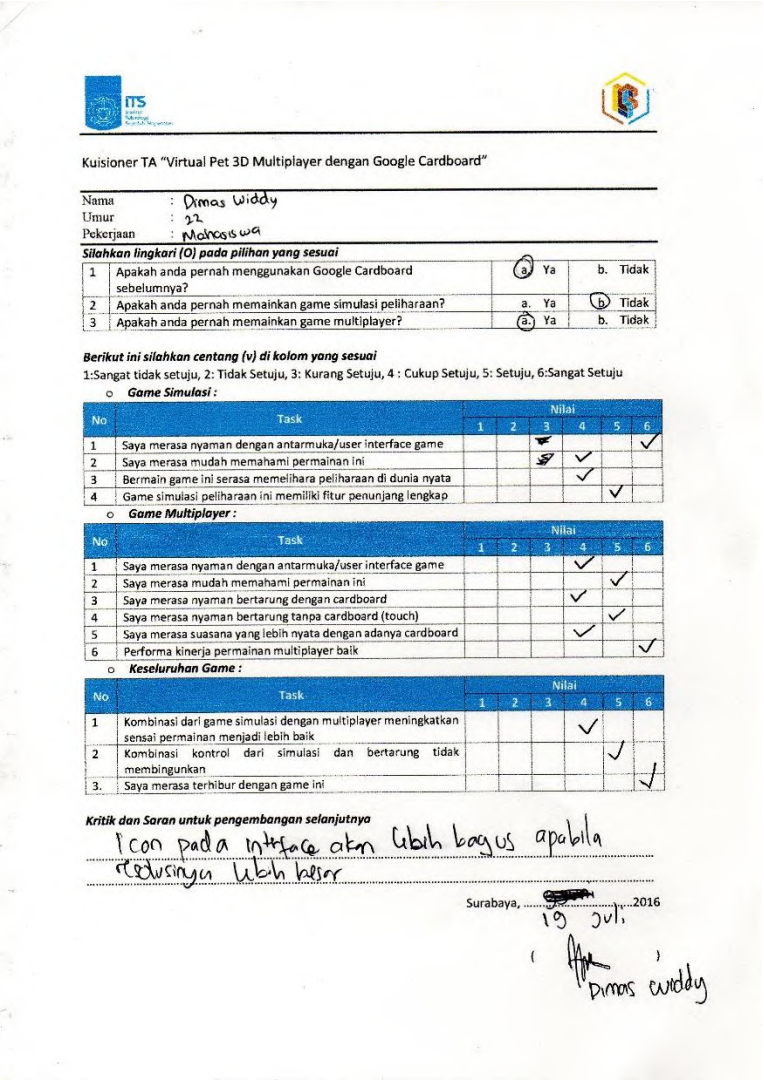
Kritik dan Saran untuk pengembangan selanjutnya

—

Surabaya, 19 Juli 2016

(Aditya P.F)

Gambar 8.4 Kuisioner Aditya



Kuisioner TA "Virtual Pet 3D Multiplayer dengan Google Cardboard"

Nama : Dimas Widdy
Umur : 22
Pekerjaan : 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 game simulasi peliharaan?	a. Ya	(b) Tidak
3	Apakah anda pernah memainkan game multiplayer?	(a) Ya	b. Tidak

Berikut ini silahkan centang (v) di kolom yang sesuai

1:Sangat tidak setuju, 2: Tidak Setuju, 3: Kurang Setuju, 4 : Cukup Setuju, 5: Setuju, 6:Sangat Setuju

- ***Game Simulasi :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game			✓			✓
2	Saya merasa mudah memahami permainan ini			✓	✓		
3	Bermain game ini serasa memelihara peliharaan di dunia nyata				✓		
4	Game simulasi peliharaan ini memiliki fitur penunjang lengkap					✓	

- ***Game Multiplayer :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Saya merasa nyaman dengan antarmuka/user interface game				✓		
2	Saya merasa mudah memahami permainan ini					✓	
3	Saya merasa nyaman bertarung dengan cardboard				✓		
4	Saya merasa nyaman bertarung tanpa cardboard (touch)					✓	
5	Saya merasa suasana yang lebih nyata dengan adanya cardboard				✓		
6	Performa kinerja permainan multiplayer baik						✓

- ***Keseluruhan Game :***

No	Task	Nilai					
		1	2	3	4	5	6
1	Kombinasi dari game simulasi dengan multiplayer meningkatkan sensai permainan menjadi lebih baik				✓		
2	Kombinasi kontrol dari simulasi dan bertarung tidak membingunkan					✓	
3.	Saya merasa terhibur dengan game ini						✓

Kritik dan Saran untuk pengembangan selanjutnya

Icon pada interface akan lebih bagus apabila resolusinya lebih besar

Surabaya, 19 Juli 2016

(Dimas Widdy)

Gambar 8.5 Kuisioner Widdy

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dalam proses pengerjaan Tugas Akhir ini, didapatkan kesimpulan sebagai berikut:

1. Simulasi *virtual pet* dirancang dengan diagram *use case* untuk menggambarkan sistem secara keseluruhan dan *finite state machine* (FSM) untuk memperjelas prinsip kerja sistem. Implementasi mengacu pada rancangan pada diagram tersebut.
2. Implementasi dari fitur *multiplayer* ini menggunakan Photon Unity Networking. Penerapan *multiplayer* ini pada pertarungan karakter peliharaan. Pemain dapat membuat tantangan baru ataupun menerima tantangan yang telah dibuat.
3. Penerapkan realitas virtual menggunakan *google cardboard*. Pertarungan peliharaan menjadi *game* bergenre *first person shooter* dengan *google cardboard* dan *smartphone* sebagai perangkat media virtual dan trigger *google cardboard* sebagai kontrol.
4. Aturan main untuk permainan simulasi peliharaan adalah mempertahankan kondisi peliharaan agar tidak mati. Komponen-komponen yang mempengaruhi kondisi peliharaan adalah kesehatan, kebahagiaan, energi dan kelaparan.
5. Aturan main yang digunakan pada pertarungan adalah mempertahankan *health point* karakter peliharaan dan mengalahkan musuh pada batas waktu yang telah ditentukan.
6. Berdasarkan ujicoba perangkat lunak dengan skala 6, pengujian simulasi peliharaan memiliki nilai 4.5, membuktikan bahwa simulasi peliharaan ini cukup

menyerupai memelihara hewan didunia nyata. Pengujian terhadap realitas virtual mendapat nilai 4.73, membuktikan bahwa permainan realitas virtual bertarung menarik. Pengujian multiplayer mendapat nilai 4.8, membuktikan bahwa permainan multiplayer ini memiliki performa baik. Berdasarkan pengujian keseluruhan sistem mendapat nilai 4.73, membuktikan bahwa permainan ini menghibur partisipan.

6.2. Saran

Berikut adalah beberapa saran untuk pengembangan sistem di masa yang akan datang berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Kontrol karakter yang masih sulit dan membingungkan, sehingga perlu ada perkembangan ide baru kembali untuk mencari kontrol gerakan selain trigger pada *google cardboard.*
2. Koneksi internet yang labil, tidak dapat mendukung permainan ini. Lebih baik ada solusi untuk membuat permainan yang berjalan dengan koneksi internet yang tidak stabil.
3. Lebih variatif skill dan monsternya.
4. Tutorial lebih dijelaskan pada masing-masing fiturnya dan ditunjukkan pada gambar fiturnya.
5. Beberapa ikon masih belum terlihat dijelas, sehingga resolusi perlu ditambah.

DAFTAR PUSTAKA

[1] H. Ulul, "solopos," Game Populer : Clash Of Clans masih peringkat teratas, 14 Januari 2015. [Online]. Available: http://www.solopos.com/2015/01/14/game-populer-clash-of-clans-masih-peringkat-teratas-567676. [Diakses 13 Juli 2016].

[2] W. Pulangsih, "wikugame," 10 Game Merawat Hewan Terbaik, 16 Desember 2013. [Online]. Available: http://www.wikugame.com/2013/12/10-game-merawat-hewan-terbaik.html. [Diakses 13 Juli 2016].

[3] K. G. D. Herlangga, "Codepolitan," Virtual Reality dan Perkembangannya, 7 Maret 2016. [Online]. Available: https://www.codepolitan.com/virtual-reality-dan-perkembangannya/. [Diakses 13 Juli 2016].

[4] R. Roedavan, Unity: Tutorial Game Engine, Bandung: Informatika Bandung, 2014.

[5] B. Jackson, "Marxentlabs," What is Virtual Reality? [Definition and Examples], 2 Juni 2015. [Online]. Available: http://www.marxentlabs.com/what-is-virtual-reality-definition-and-examples/. [Diakses 11 Juni 2016].

[6] Yanna, "Komunikasi.us," sejarah dan perkembangan multiplayer online game, 18 Maret 2015. [Online]. Available: http://komunikasi.us/index.php/course/3291-sejarah-dan-perkembangan-multiplayer-online-game. [Diakses 11 Juni 2016].

[7] Anonim, "Photon Engine," Exit Game Inc, [Online]. Available: https://www.photonengine.com. [Diakses 15 Desember 2015].

[8] Anonim, “Google Cardboard Developer,” Google Inc, [Online]. Available: https://developers.google.com/cardboard/. [Diakses 15 Desember 2015].

[9] C. Allen, J. Pragantha dan D. A. Haris, “3D Virtual Pet Game "Moar" With Augmented Reality to Simulate Pet Raising Scenarion on Mobile Device,” dalam *ICACSIS*, Jakarta, 2014.

[10] Anonim, “technopedia Inc,” First Person Shooter (FPS), [Online]. Available: https://www.techopedia.com/definition/241/first-person-shooter-fps. [Diakses 19 Juli 2016].

[11] Anonim, “flylib,” [Online]. Available: http://flylib.com/books/en/4.70.1.87/1. [Diakses 21 Juni 2016].

BIODATA PENULIS

Penulis dilahirkan di Surabaya pada tanggal 6 Juni 1993, merupakan anak ketiga dari tiga bersaudara. Penulis telah menempuh pendidikan formal yaitu SDN Mojo VII/226 Surabaya (2000-2006), SMP NEGERI 6 Surabaya (2006-2007), SMP IT Abu Bakar Yogyakarta (2007-2009), SMA Negeri 5 Yogyakarta (2009-2011), dan S1 Jurusan Teknik Informatika dengan rumpun mata kuliah Interaksi, Grafika, dan Seni (2012-2016). Selama menjadi mahasiswa, penulis pernah didanai pada PKM-KC 2014-2015. Penulis juga aktif dalam organisasi kemahasiswaan HMTC ITS sebagai anggota di Divisi Kesejahteraan Mahasiswa dan BEMF Teknologi Informasi ITS di Organization Social Responbility (OSR) sebagai anggota dan staff ahli. Penulis memiliki hobi olahraga yaitu futsal dan badminton. Penulis pernah berkecimpung dalam bidang akademik, misalnya menjadi asisten dosen di PIKTI – ITS dan Informatika ITS. Penulis dapat dihubungi melalui surel dimasriskahadi01@gmail.com.