TUGAS AKHIR – KI141502

RANCANG BANGUN *SYNCHRONOUS* PVP *MULTIPLAYER ONLINE* DALAM *GAME* SOSIAL RANGERS COMPANION DENGAN MENGGUNAKAN UNITY DAN *FRAMEWORK* PHOTON UNITY NETWORKING PADA PERANGKAT ANDROID

ASKARY MUHAMMAD
NRP 5111100121

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Ridho Rahman H., S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

FINAL PROJECT- KI141502

DESIGN AND IMPLEMENTATION SYNCHRONOUS PVP MULTIPLAYER ONLINE IN RANGERS COMPANION OF SOCIAL GAME USING UNITY AND PHOTON UNITY NETWORKING FRAMEWORK ON ANDROID DEVICE

ASKARY MUHAMMAD
NRP 5111100121

Advisor
Imam Kuswardayan, S.Kom., M.T.
Ridho Rahman H., S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2015

LEMBAR PENGESAHAN

RANCANG BANGUN *SYNCHRONOUS* PVP *MULTIPLAYER ONLINE* DALAM *GAME* SOSIAL RANGERS COMPANION DENGAN MENGGUNAKAN UNITY DAN FRAMEWORK PHOTON UNITY NETWORKING PADA PERANGKAT ANDROID

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:
ASKARY MUHAMMAD
NRP. 5111 100 121

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

Ridho Rahman H., S.Kom., M.S. ..... (pembimbing 2)
NIP: 19870213 201404 1 001

SURABAYA
JUNI, 2015

RANCANG BANGUN *SYNCHRONOUS* PVP *MULTIPLAYER ONLINE* DALAM *GAME* SOSIAL RANGERS COMPANION DENGAN MENGGUNAKAN UNITY DAN FRAMEWORK PHOTON UNITY NETWORKING PADA PERANGKAT ANDROID

Nama Mahasiswa : Askary Muhammad
NRP : 51111 100 121
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing II : Ridho Rahman H., S.Kom., M.Sc.

ABSTRAK

Kemajuan perkembangan teknologi game yang tiap tahunnya berkembang pada perangkat lunak dan perangkat keras khususnya pada perangkat mobile yang mendukung sistem operasi Android, kini telah mumpuni menjalankan aplikasi game berskala 3D dan pengintegrasian antar perangkat lunak dan perangkat keras yang menghasilkan performa yang optimal.

Pengalaman bermain pengguna sangat bervariasi dari segi aspek yang berbeda-beda cara untuk meningkatkan pengalaman bermain pada pengguna. Bermain dalam satu waktu yang sama dalam sebuah permainan dengan pemain lainnya merupakan salah satu caranya untuk menciptakan pengalaman bermain game menjadi lebih menyenangkan.

Dalam Tugas Akhir ini dibangun sebuah permainan yang bergenre action yang memiliki gameplay yang interaktif untuk mengendalikan karakter dan menerapkan tipe permainan synchronous multiplayer untuk mode pvp dalam sebuah permainan. Antar pemain dapat langsung berinteraksi ketika bermain karena dipertemukan secara realtime. Metode penerapan untuk merealisasikannya tersebut akan menggunakan perpaduan dari unity engine dengan framework photon unity network untuk mengintegrasikan unity engine dengan photon cloud. Metode tersebut akan diuji kemampuannya, dari proses

awal bermain sampai akhir permainan serta integrasi aturan main terhadap mode offline mode practice maupun synchronous multiplayer. Dengan fungsionalitas aplikasi permainan tersebut diketahui bahwa cara bermain yang mempertemukan antar pemain secara realtime akan menciptakan suasana menyenangkan dalam bermain.
Kata kunci**: **PVP Multiplayer, Action Game, Unity, Photon Cloud, Photon Unity Networking, Synchronous

DESIGN AND IMPLEMENTATION SYNCHRONOUS PVP MULTIPLAYER ONLINE IN RANGERS COMPANION OF SOCIAL GAME USING UNITY AND PHOTON UNITY NETWORKING FRAMEWORK ON ANDROID DEVICE

Student Name	: Askary Muhammad
NRP	: 51111 100 121
Major	: Teknik Informatika FTIf-ITS
Advisor I	: Imam Kuswardayan, S.Kom., M.T.
Advisor II	: Ridho Rahman H., S.Kom., M.Sc.

ABSTRACT

The growth of game development technology in every year, especially of hardware and software on mobile device that support Android platform, now, they can run some 3D video games and integrated between hardware and software that give you optimal performance.

User gaming experience is greatly vary in every differente aspects of how to enhance gaming experience to users. Playing game at realtime with each other person is one way of create gaming experience more enjoyable.

In this final project built an action game which has interactive gameplay to control the character and implement synchronous multiplayer game for pvp mode in the game. Among the players can directly interact when playing because they current in realtime. The method to realize it, will use the combination of Unity Engine with Photon Unity Network Framework to integrate Unity Engine with the Photon Cloud. The method will be tested from start process of the game until the end of the process as well as integrated with the game rules to offline mode and synchronous mode. The game functionality in mind that the way to bring together players playing in realtime, will create a pleasant atmosphere while playing.

.

***Keywords**: PVP Multiplayer, Action Game, Unity, Photon Cloud, Photon Unity Networking, Synchronous*

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 "Rancang Bangun *Synchronous* PVP *Multiplayer Online* dalam *Game* Sosial Rangers Companion dengan Menggunakan Unity dan Framework Photon Unity Networking".

Pengerjaan tugas akhir ini adalah momen bagi penulis untuk mengeluarkan seluruh kemampuan, hasrat, dan keinginan yang terpendam di dalam hati mulai dari masuk kuliah hingga lulus sekarang ini, lebih tepatnya di kampus Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak. Melalui lembar ini, penulis ingin secara khusus menyampaikan ucapan terima kasih kepada:

1. Allah Subhanahu wa ta'ala yang telah melimpahkan rahmat, hidayah, dan inayah-Nya sehingga penulis, Alhamdulillah mampu menyelesaikan tugas akhir dengan baik.
2. Junjungan kita Nabi Muhammad Shallahu Alayhi Wasallam, Sahabat dan Keluarganya yang telah menjadi inspirasi, contoh yang baik bagi penulis sehingga tetap termotivasi dalam mengerjakan tugas akhir.
3. Kedua orang tua penulis, ibu tercinta dan almarhum ayah yang telah mencurahkan kasih sayang, perhatian, dan doa kepada penulis.
4. Bapak Imam Kuswardayan, S.Kom., M.T. dan Bapak Ridho Rahman H., S.Kom., M.Sc. selaku dosen pembimbing yang telah memberikan bimbingan, motivasi, dan meluangkan waktu untuk membantu pengerjaan tugas akhir ini.

5. Prof. Ir. Joko Lianto Buliali, M.Sc.,Ph.D. selaku dosen wali yang telah memberikan perhatian dan motivasi kepada penulis selama menjadi mahasiswa di lingkungan Teknik Informatika ITS.
6. Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D selaku dosen yang pertama kali telah memberikan pengarahan, pengalaman berharga dan bimbingan sehari-hari selama 2 semester awal.
7. Ibu Nanik Suciati, S.Kom., M.Kom., Dr.Eng. selaku ketua jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom, M.Sc. selaku koordinator TA dan koordinator KP dan segenap Bapak/Ibu dosen Teknik Informatika yang telah memberikan ilmunya kepada penulis.
8. Segenap staff Jurusan Teknik Informatika ITS, terima kasih telah mengayomi penulis selama berada di TC.
9. Kepada Munasir Fams yang telah memberi dukungan sebagai keluarga besar munasir.
10. Kepada Wa Ben yang telah banyak membantu berbagai kebutuhan selama kuliah dan banyak membantu keluarga.
11. Kepada khususnya Rumhadi Fams, keluarga dan 6 saudara kandung yang selalu memberi saran dan dukungan selama kuliah.
12. Kepada Nenek Anis dan Keluarga Tenggilis yang telah banyak membantu, mendukung dan menasihati.
13. Kepada Pina Rosica yang selalu memberi motivasi, bertukar wawasan dan ilmu baru dan sahabat terbaik seperjuangan.
14. Komunitas Science3 25 yang selalu menemani perjuangan suka dan duka.
15. Sahabat IGS (Didik, Punggi, Rustam, Toto, Jordi, Andrie, Risal, Ruslan, dkk) dan sahabat KCV beserta para adminnya (Admin 2011, Admin 2012) lainnya yang menemani penulis di saat suka dan duka.

16. Seluruh teman Teknik Informatika ITS angkatan 2011 yang telah menemani dan memberi pengalaman berharga bagi penulis sejak maba sampai lulus.
17. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya tugas akhir ini, penulis mengucapkan terima kasih.

Penulis telah berusaha sebaik mungkin dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2015
Penulis

Askary Muhammad

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan dan kemajuan teknologi pada industri *game* sangat banyak yang menciptakan inovasi, kreatifitas dan ide-ide baru serta menghimpun banyak pengguna yang terlibat dan saling berinteraksi. Cara pengguna berinteraksi tersebut kini dapat dinikmati secara *realtime* (*synchronous*) maupun *turn based* (*asynchronous*) dengan pemanfaatan teknologi SaaS (*Software as a Service*) yang diakses melalui internet oleh perangkat Android, Iphone, Windows Phone dan sebagainya.

Game saat ini sudah dapat dimainkan secara *massive* dan *online* karena ada *server* yang menangani para pengguna dalam berinteraksi. Sudah umum dan banyak sekali *game* sosial yang menerapkan metode *turn based* (*Asynchronous*) dimana interaksi antar pengguna tidak dipertemukan secara *realtime* atau tidak dibatasi waktu. Tetapi penerapan secara *realtime* (*Synchronous*) sedikit ditemukan di pasar *smartphone*, karena pada umumnya diterapkan pada perangkat PC*/Desktop*. Kebanyakan dari pengguna *smartphone* yang memainkan *game* yang menerapkan metode *realtime* (*Synchronous*) lebih cepat bosan, oleh karena itu *game* yang akan dibuat membutuhkan unsur *gameplay* dari karakter *game* yang populer seperti *Defense Of The Ancient*. Unsur karakter tersebut akan menjadi karakter utama dalam *game* sosial ini yang berjudul Rangers Companion dan *game* ini akan bergenre *action* karena *basic gameplay* pada *game* sosial ini adalah pertarungan di arena dimana karakter yang dimainkan menggunakan senjata busur untuk mempertahankan diri. Aset-aset yang akan dibuat pada *game* ini berupa 3D dan diproyeksikan oleh kamera secara *orthographic* atau bisa disebut juga dengan 3D *Isometric*.

Game ini utamanya akan berjalan pada perangkat Android. Kemudian fitur sosial yang diutamakan yaitu PVP *Multiplayer Online* dengan menggunakan *framework* Photon Unity Networking,

sehingga pengguna dapat berinteraksi dengan sesama pengguna perangkat Android.

1.2 Rumusan Masalah

Berikut rumusan masalah yang diangkat dalam tugas akhir ini adalah :

1. Bagaimana membuat *basic gameplay* untuk *game* sosial Rangers Companion?
2. Bagaimana membuat fitur sinkronisasi PVP *Multiplayer Online* dengan menggunakan Photon Unity Networking dan data apa saja yang perlu disinkronisasi?
3. Bagaimana merancang dan mengimplementasikan integrasi data pemain, karakter, aturan main dan komponennya untuk fitur PVP *Multiplayer*?
4. Bagaimana mekanisme alur kerja PVP *Multiplayer* dari proses bisnis persiapan bermain, mulai bermain sampai menyelesaikan permainan?

1.3 Batasan Masalah

Berikut batasan masalah pada tugas akhir ini adalah:

1. Pembuatan *game* ini akan menggunakan IDE Unity dan bahasa pemrograman C#.
2. *Map* terdiri dari 3 tipe, yaitu untuk *room* berkapasitas sedikit, sedang dan besar untuk menampung pemain.
3. *Offline single player* digunakan pemain untuk berlatih dengan melawan AI/BOT sehingga ketika seseorang yang baru pertama kali memainkan *game* tipe seperti ini tidak kebingungan dalam memainkannya.
4. Total maksimum CCU (*Concurent Users*) yang dapat terhubung ke Photon Cloud yaitu 20 CCU.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah mengimplementasikan fitur sosial PVP *Multiplayer Online* pada

game Rangers Companion dengan pengintegrasian Unity dan Photon Unity Networking.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Sebagai bentuk implementasi Photon Unity Networking pada fitur sosial secara *synchronous* untuk *game* sosial Rangers Companion.
2. Memberikan suasana ide *game* baru dari pengambilan unsur karakter *game* yang populer dari perangkat PC/*Desktop* agar pengguna tidak bosan dengan *game* pada *smartphone* yang fitur sosialnya menggunakan *synchronous*.
3. Memberikan hiburan dan melatih ketangkasan bermain kepada pengguna.

1.6 Metodologi

Pembuatan tugas akhir dilakukan menggunakan metodologi sebagai berikut:

A. Studi literatur

Pada tahap studi literatur ini, akan dilakukan pengumpulan informasi dan referensi yang akan menjadi acuan proses perancangan dan implementasi pada *game* yang akan dibangun, mulai dari proses pembuatan *game* mekanik karakter 3D dan lingkungannya serta *basic gameplay* di Unity dengan fitur sosial yang akan diimplementasikan menggunakan Photon Unity Networking.

B. Analisis dan Perancangan perangkat lunak

Pada tahap ini diawali dengan melakukan analisa, perencanaan dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang akan dihadapi pada tahap implementasi. Kemudian kebutuhan-kebutuhan dalam perancangan sistem ini meliputi data yang digunakan, antarmuka, alur, proses bisnis dan pengintegrasian Unity dengan Photon Unity Networking.

C. Implementasi dan pembuatan sistem
Pada tahap ini dilakukan implementasi metode yang diusulkan dari rancangan yang telah dibuat pada tahap sebelumnya dengan menggunakan Unity Engine sebagai *game engine* dan Mono Develop sebagai *editor*-nya.

D. Uji coba dan evaluasi
Pada tahap ini dilakukan uji coba dengan menggunakan beberapa macam kondisi untuk mencoba aplikasi bisa berjalan atau tidak. Uji fungsionalitas untuk mengetahui apakah aplikasi sudah memenuhi semua kebutuhan fungsional. Selain itu dilakukan pula uji non-fungsionalitas untuk mengetahui apakah unsur fleksibilitas dan performa aplikasi mampu mendukung fungsionalitas utama sistem.

E. Penyusunan laporan tugas akhir
Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan tugas akhir.

1.7 Sistematika Penulisan

Buku tugas akhir ini terdiri dari beberapa bab, yang dijelaskan 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 pembuatan *Game* Sosial Rangers Companion.

BAB V PENGUJIAN DAN EVALUASI

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

BAB VI PENUTUP

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

BAB II
TINJAUAN PUSTAKA

2.1 PVP *Multiplayer Online*

PVP adalah *Player Versus Player*, dimana merupakan ragam dari permainan multiplayer yang mengedepankan keterampilan atau keunggulan dari seorang pemain atau bisa disebut dengan *gamers*. PVP juga merupakan salah satu fitur *game* sosial dan tipe interaktif *gameplay* dari *multiplayer* dimana dalam satu waktu dan *server* yang sama terjadi aktifitas bermain yang bisa terdiri antara lebih dari 2 pemain. [2]

Perkembangan awal PVP *Multiplayer* bermula dari *game multiplayer* yang menggunakan jaringan koneksi lokal (*Local Area Network*) dan saat ini teknologinya sudah mendukung berbagai perangkat *smartphone* dengan hanya terhubung koneksi internet pada *game* yang memiliki fitur tersebut. [2]

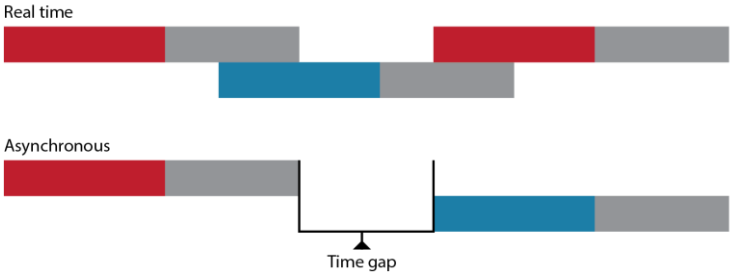
Pada pengembangan jenis permainan *multiplayer* ini, pengembang permainan dapat mengadopsi *gamemode* seperti *Capture The Flag*, *Deathmatch FFA, Deathmatch Team*, *King of The Hill*, *Dueling*. Kemampuan bermain pemain akan diuji pada *gamemode* tersebut dari segi kemampuan taktis dan strategi untuk memenangkan permainan melawan pemain-pemain lainnya.

2.2 *Game* Sosial

Game sosial merupakan *game* yang dapat beraktifitas dan berinteraksi secara *virtual* antar sesama penggunanya. Cara mempertemukan penggunanya secara umum ada dua metode, yaitu dengan metode *synchronous* dan *asynchronous*. Pada metode *synchronous*, antar pengguna dipertemukan secara *realtime* ketika bermain sedangkan pada metode *asynchronous*, antar pengguna dapat berinteraksi tanpa dibatasi oleh waktu. [6]

2.3 *Synchronous Multiplayer vs Asynchronous Multiplayer*

Merupakan konsep tipe permainan yang diterapkan dalam keadaan *realtime* dan memiliki batasan waktu (harus bersamaan). Tipe permainain seperti ini membutuhkan koneksi yang stabil dimana keadaan stabil tersebut akan optimal jika *ping latency* dibawah 100ms. Tidak seperti *asynchronous*, konsep *synchronous* harus selalu berada dalam *state* yang sama antar klien lainnya. Jika salah satu klien berada dalam *state* yang berbeda, maka konsep *synchronous* akan hilang. Berikut penjelasan perbedaan konsep antara *synchronous* dan *asynchronous* pada Gambar 2.1. [6]



Gambar 2.1 Konsep *Time Gap*

2.4 Windrunner (*Character of* Defense Of The Ancient)

Windrunner adalah karakter dari *game* Defense Of The Ancient yang memiliki kemampuan yang kuat dan ahli dalam menggunakan busurnya. Karakter ini merupakan karakter yang memiliki karakteristik gesit seperti angin dan berpengalaman menghadapi hidup di alam liar seperti hutan, gua dan sebagainya. Hidupnya di alam liar dengan ancaman binatang buas sehingga membuat dirinya seperti pemburu. [3]

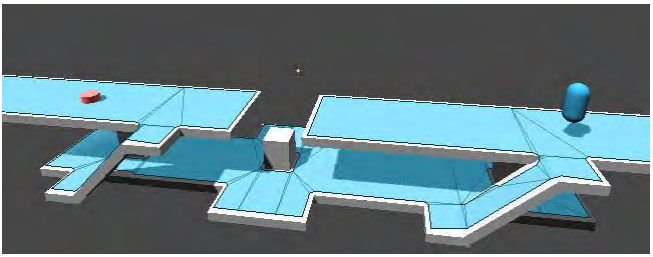
2.5 Dungeon Hunter

Dungeon Hunter merupakan sebuah *game multiplatform* yang dirilis oleh Gameloft dimana *game* ini memiliki *genre RPG Action* yang disajikan dengan grafis 3D dan merupakan *game*

terlaris pada platform iOS, Android, Windows Phone, PC/*Desktop* dengan jutaan pengguna yang aktif tiap harinya. *Game* ini merupakan *game* sosial mendukung mode *offline* dan *online multiplayer*. *Game* ini menceritakan seorang pejuang yang harus menyelamatkan bangsanya dari ratu jahat yang telah berbuat kekacauan di negerinya. *Game* ini memiliki latar *gothic*, karena musuh-musuh yang dihadapi oleh pemain adalah monster-monster yang berasal dari neraka. Saat ini versi *game* ini sudah sampai versi 5. [4]

2.6 Navmesh Agent Unity

Navmesh Agent merupakan komponen pada unity yang memudahkan pengembang untuk mengimplementasikan karakter pemain dalam pembuatan AI agar memiliki sifat responsif terhadap *terrain* pada sebuah *scene game* dan mendeteksi adanya tumbukkan terhadap objek sekitarnya. Sehingga pembuatan AI dapat lebih cepat dan efisien untuk melacak keberadaan musuh serta memudahkan dalam pengembangan aplikasi permainan. Mengacu pada Gambar 2.2, pelacakan sebuah objek memanfaatkan hasil *bake* pada warna biru *terrain* untuk mengetahui posisi sasaran objek. [1]

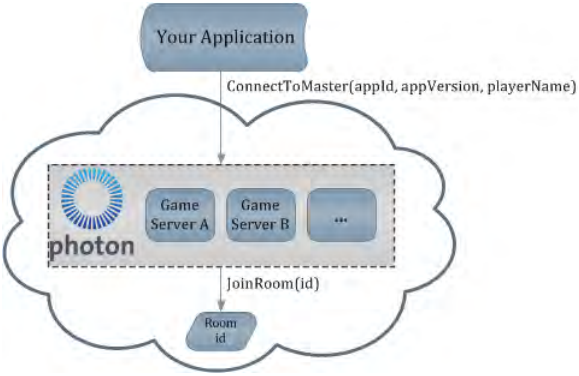


Gambar 2.2 Navmesh Agent Unity

2.7 Photon Unity Networking

PUN (Photon Unity Networking) adalah sebuah *framework Software Development Kit* untuk IDE Unity yang menyediakan *class library* untuk pembuatan *game multiplayer*. Model

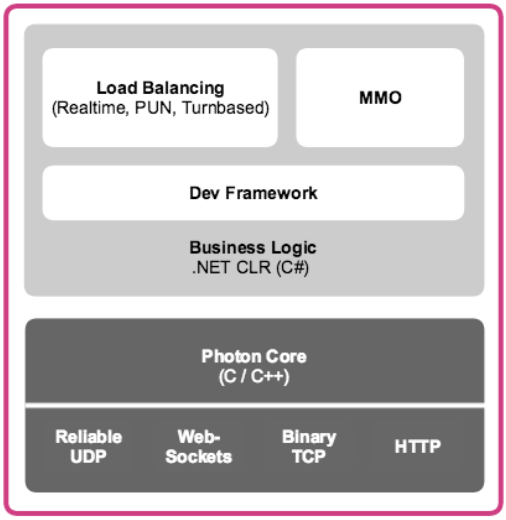
arsitektur dalam pembuatan *game multiplayer* salah satunya adalah SaaS (*Software as a Service*). Kemampuan *web service* pada *server*-nya memiliki stabilitas yang sangat direkomendasikan untuk para pengembang yang menggunakan Unity dalam pengembangan permainan. Berikut gambaran integrasi aplikasi dengan Photon Unity Networking pada Gambar 2.3. [5]



Gambar 2.3 Fitur Photon Unity Networking

2.8 Photon Cloud

Photon Cloud merupakan *server* photon untuk menjalankan *multiplayer game backend*. Aplikasi yang berjalan pada bahasa pemrograman C#. Pada *backend* tersebut sudah diatur dalam optimasi performa oleh Photon sehingga pengembang tidak perlu sulit untuk mengatur *load balancing server* dalam menjaga keseimbangan transmisi pertukaran data. Berikut arsitektur yang diterapkan seperti pada Gambar 2.4. [5]



Gambar 2.4 Arsitektur Photon Cloud

2.9 Photon Data *Serialization*

Photon Data *Serialization* merupakan cara photon untuk sinkronisasi pertukaran data. Photon memodifikasi Network View pada fitur *native multiplayer* pada Unity Engine menjadi Photon View yang ditanamkan protokol untuk berkomunikasi antar klien. Photon hanya mendukung beberapa tipe data yang dapat dilakukan *serialization* atau sinkronisasi antar klien. Berikut daftar tipe data yang didukung oleh Photon pada Gambar 2.5. Tidak hanya Network View saja yang dimodifikasi tetapi pemanggilan fungsi secara *remote* mengalami modifikasi untuk mengarahkan target *serialization* menggunakan Photon RPC. [5]

type (C#)	size [bytes]	description
byte	2	8 bit unsigned
boolean	2	true or false
short	3	16 bit
int	5	32 bit
long	9	64 bit
float	5	32 bit
double	9	64 bit
string	3 + size(UTF8.GetBytes(string))	< short.MaxValue length
byte-array	5 + 1 * length	< int.MaxValue length
int-array	5 + 4 * length	< int.MaxValue length
array of <type>	4 + size(entries) - count(entries)	< short.MaxValue length
hashtable	3 + size(keys) + size(values)	< short.MaxValue pairs
dictionary	3 + size(keys) + size(values)	< short.MaxValue pairs additional size if K- or V-type is object

Gambar 2.5 Photon *Primary Supported Types*

Photon mendukung data *custom* Unity tambahan yang menampung data *primary* yang bertujuan untuk meningkatkan performa. Berikut data tipe tambahan yang didukung oleh Photon pada Gambar 2.6. [5]

type (C#)	size [bytes]	description
Vector2	12	2 floats
Vector3	16	3 floats
Quaternion	20	4 floats
PhotonPlayer	8	integer PhotonPlayer.ID

Gambar 2.6 Photon *Additional Supported Type*

BAB III
ANALISIS DAN PERANCANGAN

Bab ini menjelaskan tentang analisis dan perancangan aplikasi *game* sosial Rangers Companion dengan menggunakan Unity dan Photon Unity Networking. Pembahasan yang akan dilakukan meliputi analisis fitur yang dibutuhkan dan perancangan aplikasi permainan.

3.1 Analisis Perancangan Permainan

Permainan video pada perangkat *mobile* dimana salah satu platform yang penggunanya sangat banyak dan didominasi oleh Platform Android, saat ini perkembangannya sangat pesat seiring dengan berkembangnya teknologi dari segi perangkat lunak maupun perangkat keras. Perangkat Android yang dilengkapi dengan perangkat keras serta memiliki spesifikasi tinggi telah banyak dipasaran dan mudah didapat sehingga sangat mumpuni untuk menjalankan banyak aplikasi, menjalankan aplikasi yang berat sekalipun seperti permainan video 3D. Pengalaman dalam bermain *game* menjadi tantangan terhadap pengembang untuk berinovasi, menciptakan sebuah permainan. Pasar aplikasi permainan pada Google Play rata-rata sudah menerapkan konsep permainan yang dapat terhubung *online* dengan *cloud* sehingga antar pemain dapat berinteraksi dan pengalaman permainan juga akan menyenangkan.

Saat ini sangat populer dan banyak disegani oleh banyak pemain terhadap konsep permainan yang secara *synchronous* terhubung ke *cloud* tetapi konsep permainan tersebut biasanya diterapkan pada platform PC/*Desktop*. Aplikasi permainan ini dibangun dengan tujuan untuk mengimplementasikan konsep *synchronous* PVP *Multiplayer Online* pada perangkat *mobile* di platform Android. Pemain akan langsung berinteraksi terhadap pemain lainnya dalam waktu yang bersamaan. Perangkat menghubungkan antar pemain melalui perantara internet yang

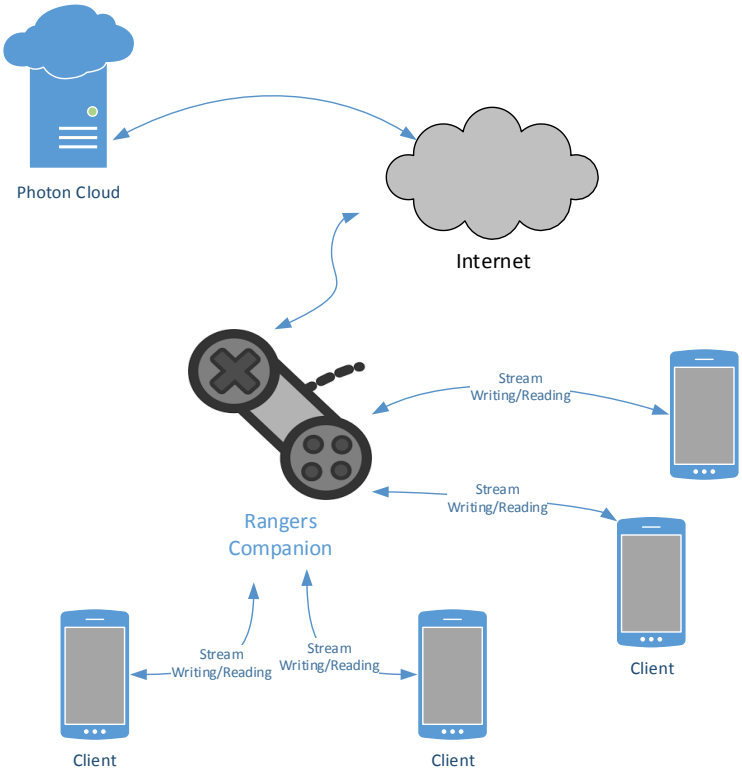
terhubung ke *cloud* yang secara sinkron mengirimkan dan menerima data dari antar pemain.

Penulis menggunakan teknologi *game engine* Unity dengan *Framework* Photon Unity Networking untuk memfasilitasi pengembangan permainan. Photon Unity Networking telah terkonfigurasi dengan Photon Cloud sehingga pengembang tidak perlu lagi mendesain pengintegrasian Photon Unity Networking dengan Photon Cloud.

3.1.1 Deskripsi Umum Aplikasi Permainan

Tugas akhir yang akan dikembangkan adalah sebuah permainan 3D bergenre *action* dengan konsep *synchronous* PVP. Untuk mensinkronisasikan antar pemain, aplikasi permainan harus dapat mengatur pertukaran data antar klien. Setiap klien akan mengirimkan datanya dan menerima data dari *cloud server* dan data yang dapat dikirimkan ke Photon Cloud dan data yang diterima hanya berupa data primitif saja. *Framework* Photon Unity Networking mengatur batasan tersebut agar performa permainan tetap stabil.

Elemen-elemen dalam permainan yang akan difokuskan dalam melakukan sinkronisasi yaitu pergerakkan pemain, penembakkan anak panah (*projectile*) dan *score leaderboard* yang dimana data-data tersebut akan ditransmisikan melalui *server* dan kemudian akan di*broadcast* ke semua klien yang terdapat pada *room* yang sama. Berikut gambaran transmisi data pertukaran pada Gambar 3.1.



Gambar 3.1 Konsep Pertukaran Data *Streaming* Photon

Setiap pemain menggunakan karakter pemain yang sama dan penentuan menang atau kalah ditentukan oleh tiap pemainnya dengan mengandalkan ketangkasan bermain. Aplikasi permainan ini menyediakan dua mode bermain yakni *Practice* dan *Find Match*. Pada mode *Practice*, pemain bermain melawan AI untuk berlatih. Pada mode *Find Match*, pemain bermain melawan pemain lainnya dengan secara otomatis aplikasi permainan akan mencarikan *room* permainan dan apabila tidak ada *room* permainan yang tersedia, maka aplikasi permainan akan mengalokasikan untuknya. Permainan PVP dapat dimainkan

ketika pemain yang tergabung telah memasuki batas maksimum sebuah *room* permainan.

Dalam sistem permainan ini, pemain mengontrol karakter dengan menggunakan *virtual analog* untuk melakukan pergerakan dan *virtual button* untuk menembakkan busur dan melihat *Scoreboard*.

3.1.2 Spesifikasi Kebutuhan Fungsional

Berdasarkan deskripsi umum aplikasi permainan, terdapat 2 kebutuhan fungsional dari aplikasi permainan ini yaitu bermain *multiplayer* secara sinkron dengan pemain lainnya dan bermain *offline mode practice*.

3.1.3 Spesifikasi Kebutuhan Non-Fungsional

Terdapat beberapa kebutuh non-fungsional yang jika dipenuhi, maka dapat meningkatkan kualitas dari aplikasi permainan. Berikut daftar kebutuhan non-fungsional:

1. **Performa Grafis 3D**

Permainan ini harus dapat berjalan dan dimainkan secara lancar pada *device* Android. Kelancaran *framerate* dipengaruhi oleh jumlah verteks, titik, garis yang terhitung pada sebuah objek 3D serta partikel objek. Pengaturan cahaya juga sangat berpengaruh dalam memproyeksikan bayangan. Faktor-faktor tersebut juga tergantung oleh *GPU unit* yang dimiliki oleh *device* Android

2. **Kenyamanan dan Kehandalan**

Permainan ini harus dapat berjalan bermacam-macam layar Android dalam orientasi *landscape* dan dapat menangani *crash* dalam situasi yang memungkinkan menjadi penyebab atau sebuah bug, sehingga harus dapat menekan bug sekecil mungkin kemungkinan.

3.1.4 Identifikasi Pengguna

Berdasarkan deskripsi umum diatas, maka dapat diketahui bahwa pengguna yang akan menggunakan aplikasi ini hanya satu yaitu pemain yang menjalankan permainan dan berinteraksi antar pemain. Penjelasan tersebut terdapat pada Tabel 3.1.

Tabel 3.1 Identifikasi Pengguna

Nama Aktor	Definisi
Pemain	Orang yang menjalankan permainan, memainkan karakter dan berinteraksi dengan pemain lainnya

3.2 Perancangan Permainan

Tahap perancangan dalam subbab ini dibagi menjadi beberapa bagian yaitu perancangan algoritma, perancangan arsitektur, perancangan proses permainan, perancangan kontrol permainan, perancangan antarmuka permainan, aturan main.

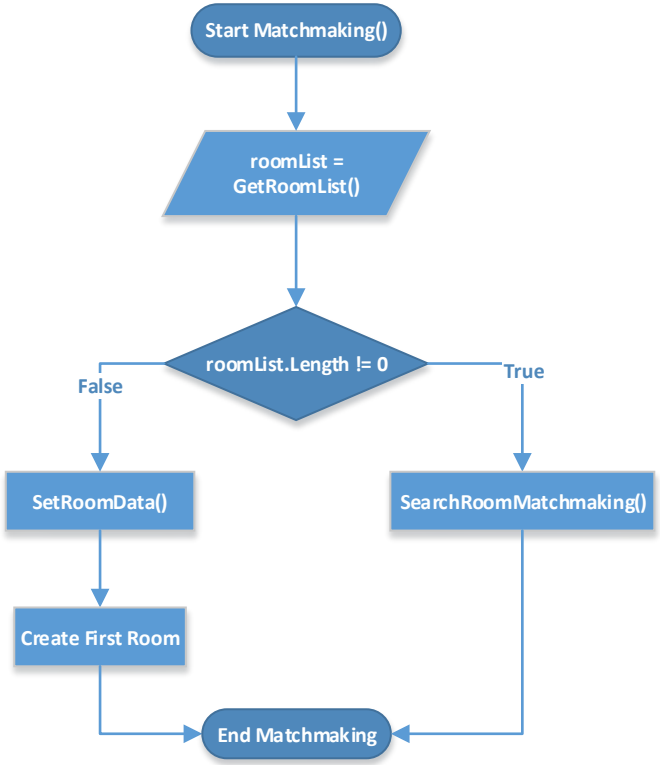
3.2.1 Perancangan Algoritma

Pada subbab ini akan dijelaskan rancangan algoritma yang akan digunakan untuk memenuhi kebutuhan fungsionalitas perangkat lunak yang dibangun. Rancangan algoritma akan digambarkan dalam bentuk *flowchart*. Berikut ini adalah penjelasan dari rancangan algoritma yang digunakan dalam perancangan permainan ini dan algoritma ini akan memanfaatkan *event-event* yang ada di Photon dan Photon Properties. Photon Properties itu sendiri adalah tempat menyimpan nilai pada *cloud server* agar memudahkan klien untuk mengakses *shared value*.

3.2.1.1 Rancangan Algoritma *Matchmaking*

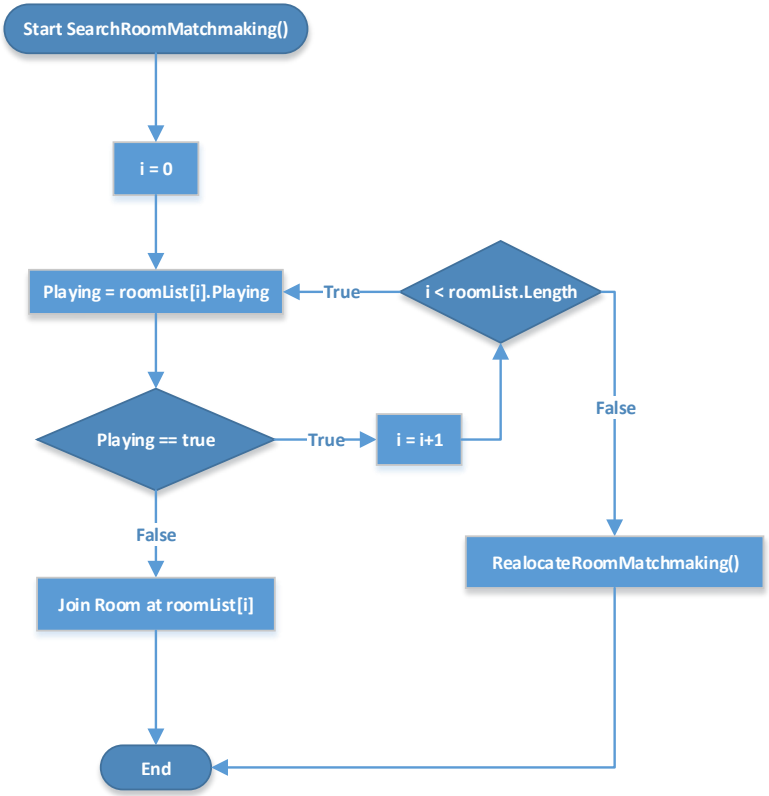
Matcmaking dalam fitur permainan ini berperan sangat penting dan merupakan peran utama dalam aplikasi permainan ini. Dalam peran tersebut, *matchmaking* mempertemukan antar

pemain yang sedang aktif melakukan pencarian pertarungan PVP. Berikut penjelasan algoritma dalam bentuk *flowchart* pada Gambar 3.2.



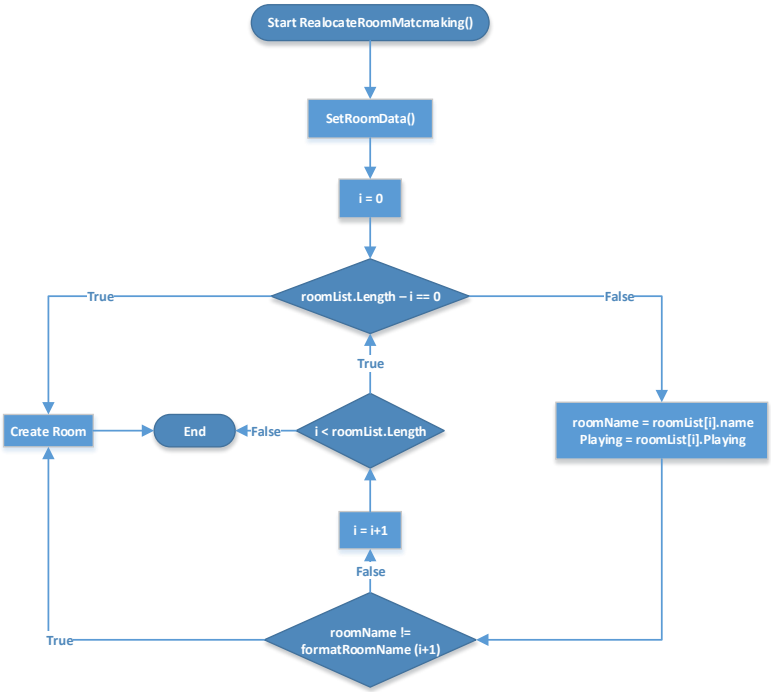
Gambar 3.2 Konteks Algoritma *Matchmaking*

Mengacu pada Gambar 3.2, algoritma membutuhkan daftar *room* yang didapatkan dari *interface* Photon *Behaviour*. Jika belum ada *room* yang tersedia pada daftar *room*, maka dilakukan fungsi detil dari algoritma matchmaking yaitu SearchRoomMatchmaking(). Berikut detil penjelasan *flowchart* pada Gambar 3.3 dan Gambar 3.4.



Gambar 3.3 Detil Algoritma *Matchmaking* (1)

Mengacu pada Gambar 3.3, ketika fungsi ini dipanggil, maka ia akan memulai perulangan sebanyak daftar *room* yang tersedia. Pencarian bertujuan untuk mencari *room* yang memiliki status sedang tidak bermain (*waiting*). Jika tidak menemukan *room* yang tersedia dan ber-status tidak sedang bermain (*waiting*) maka akan dilakukan pemanggilan fungsi RealocateRoomMatchmaking() yang dijelaskan lebih detil pada Gambar 3.4. Jika sebaliknya maka pemain akan bergabung ke dalam *room*.



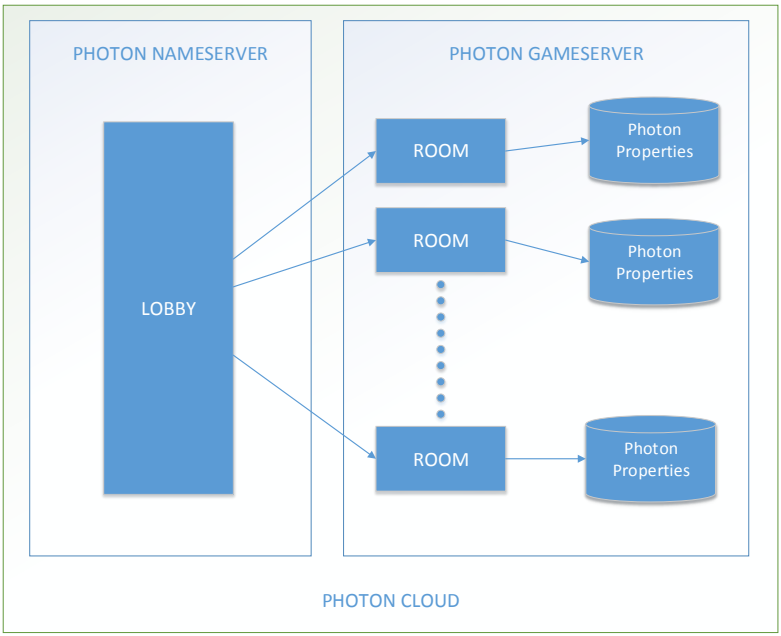
Gambar 3.4 Detil Algoritma *Matchmaking* (2)

Mengacu pada Gambar 3.4 ketika fungsi ini dipanggil, maka ia akan memulai untuk menyimpan komponen *custom properties* pada Photon Room Properties yaitu nama *room*, waktu mulai bermain, status bermain dan tipe *room*. Kemudian akan dilakukan perulangan sebanyak daftar *room* yang tersedia. Kemudian nama pada setiap *room* akan dicek agar *room* yang akan dibuat tidak akan sama namanya dengan *room* lain.

3.2.2 Perancangan Arsitektur Umum Matchmaking

Photon memiliki fitur dan metode sendiri dalam mengelola sebuah *game* multiplayer yang menggunakan *framework*-nya. Photon Unity Networking merupakan class library yang dibungkus menjadi *framework* untuk mengelola

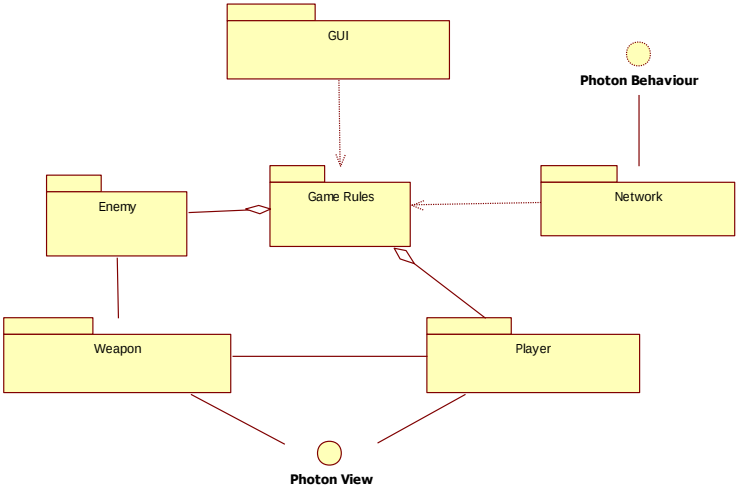
pertukaran data sinkronisasi antar klien melalui Photon Cloud. Arsitektur aplikasi permainan Rangers Companion ini pada umumnya langsung mempertemukan dengan pemain lainnya dengan memanfaatkan *event* dan *state* yang ada pada Photon Unity Networking, komponen data pemain dan *room* tersimpan dalam *Custom Properties*. Berikut adalah arsitektur sistem dari aplikasi permainan pada Gambar 3.5.



Gambar 3.5 Perancangan Arsitektur Umum Matchmaking

3.2.3 Perancangan *Package Diagram*

Untuk merancang aplikasi permainan ini, dibutuhkan model perencanaan rancangan yang terdiri dari *package-package* dengan penggambaran konteks *package* secara keseluruhan yang akan diimplementasikan seperti pada Gambar 3.6.



Gambar 3.6 Konteks *Package Diagram*

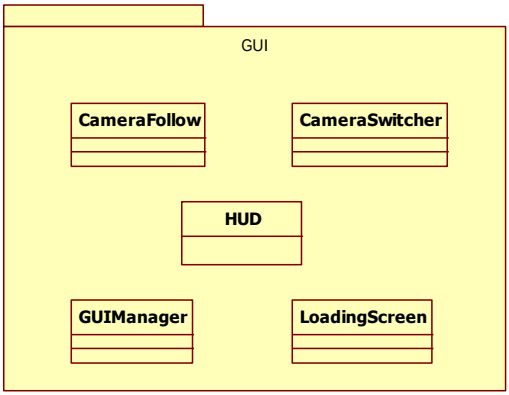
Elemen-elemen pada Gambar 3.6 terdiri dari *package* GUI, *package Enemy*, *package Game Rules*, *package Network*, *package Weapon* dan *package Weapon*. Penjelasan tiap *package* akan dijabarkan menjadi subbab 3.2.3.1, subbab 3.2.3.2, subbab 3.2.3.3, subbab 3.2.3.4, subbab 3.2.3.5, subbab 3.2.3.6.

Pembuatan aplikasi permainan akan mengintegrasikan *basic gameplay* dengan Photon Cloud sehingga aturan main pada *package Game Rules* menjadi sebuah *dependency* dari *package Network*. GUI akan selalu menampilkan kegiatan dalam bentuk HUD (*Head Under Display*) sehingga pada *package* GUI harus ber*dependency* terhadap *package Game Rules* karena semua aturan main diatur pada *package* tersebut. Pemain dan AI/BOT terhubung terhadap aturan-aturan main sehingga kedua objek tersebut pada *package Player* dan *Enemy* harus teragregasi oleh *package Game Rules*. Kedua package dari *Player* dan *Enemy* memiliki *behaviour* terhadap *package* yang sama (*package Weapon*) yang berfungsi untuk melakukan penyerangan terhadap lawannya.

Photon memiliki kelas berbagai *interface* pendukung Unity Engine. Pada perancangan ini, *interface* yang digunakan yaitu *Photon Behaviour* untuk menurunkan fungsi-fungsi utama pada Photon Unity Networking. Photon View memiliki fungsi untuk mensinkronisasikan data pada sebuah objek dengan melakukan *state serialization*.

3.2.3.1 *Package* GUI

Pada *package* ini, seperti gambaran pada Gambar 3.7 memiliki peranan untuk mengatur jalannya tampilan aplikasi permainan. Penjelasan tiap komponen kelas akan dijelaskan pada Tabel 3.2.



Gambar 3.7 *Package* GUI

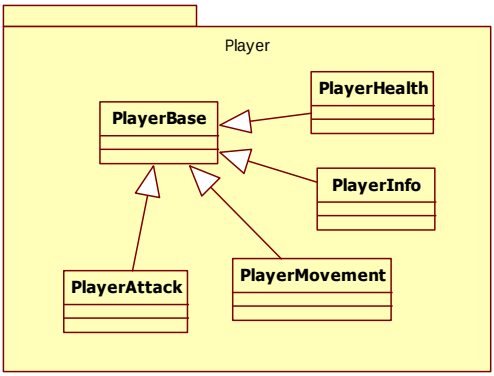
Tabel 3.2 Deskripsi *Package* GUI

Nama Kelas	Deskripsi
CameraFollow	mengatur kamera agar mengikuti pergerakkan pemain
CameraSwitcher	Mengatur penggunaan antar kamera *standby* dan kamera aktif bermain

HUD	Mengatur komponen seperti *healthbar* dan *virtual controller* yang selalu berada pada layar ketika aktif bermain
GUIManager	Mengatur kegiatan semua komponen yang ada pada tampilan
LoadingScreen	Mengatur perpindahan *scene* agar performa aplikasi tetap terjaga

3.2.3.2 *Package Player*

Pada *package* ini, seperti gambaran pada Gambar 3.8 memiliki peranan untuk mengendalikan kegiatan karakter pemain dengan integrasi terhadap *basic gameplay*. Penjelasan tiap komponen kelas akan dijelaskan pada Tabel 3.3.



Gambar 3.8 *Package Player*

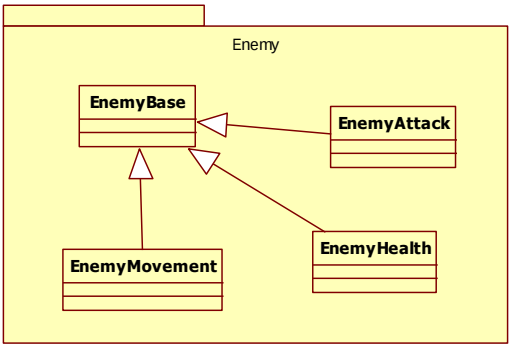
Tabel 3.3 Deskripsi *Package Player*

Nama Kelas	Deskripsi
PlayerBase	Sebagai komponen generalisasi untuk objek karakter

PlayerMovement	Sebagai komponen untuk menggerakkan karakter pemain
PlayerAttack	Sebagai komponen untuk melakukan serangan dari karakter pemain
PlayerHealth	Sebagai komponen untuk mengatur daur hidup dan mati karakter pemain
PlayerInfo	Sebagai komponen untuk memberikan informasi mengenai karakter pemain

3.2.3.3 *Package Enemy*

Pada *package* ini, seperti gambaran pada Gambar 3.9 memiliki peranan untuk mengendalikan kegiatan AI/BOT pada *Offline Mode Practice*. Penjelasan tiap komponen akan dijelaskan pada Tabel 3.4.



Gambar 3.9 *Package Enemy*

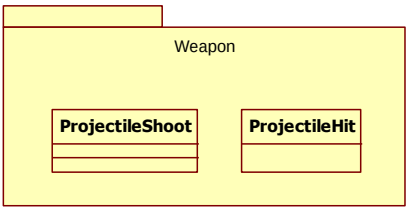
Tabel 3.4 Deskripsi *Package Enemy*

Nama Kelas	Deskripsi
EnemyBase	Sebagai komponen generalisasi

	dari objek AI/BOT
EnemyMovement	Sebagai komponen untuk menggerakkan karakter AI/BOT yang didalam fungsinya terdapat komponen dari Unity Engine yaitu *NavMeshAgent* untuk melakukan *pathfinding*
EnemyAttack	Sebagai komponen untuk melakukan serangan dari AI/BOT
EnemyHealth	Sebagai komponen untuk mengatur daur hidup dan mati karakter AI/BOT

3.2.3.4 *Package Weapon*

Pada *package* ini, seperti gambaran pada Gambar 3.10 memiliki peranan untuk menginstansiasikan objek yang akan menyerang lawan untuk memberikan *damage* terhadap lawan. Penjelasan tiap komponen akan dijelaskan pada Tabel 3.5.



Gambar 3.10 *Package Weapon*

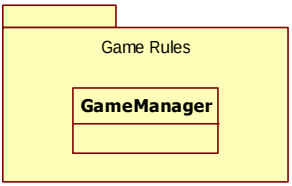
Tabel 3.5 Deskripsi *Package Weapon*

Nama Kelas	Deskripsi
ProjectileShoot	Sebagai komponen untuk

	menginstansiasikan objek yang bertipe serangan jarak jauh
ProjectileHit	Sebagai komponen untuk menginstansiasikan objek yang bertipe serangan jarak dekat

3.2.3.5 *Package Game Rules*

Pada package ini, seperti gambaran pada Gambar 3.11 memiliki peranan untuk mengendalikan kegiatan yang berisi aturan main pada tipe permainan *offline* maupun *online*. Penjelasan tiap komponen akan dijelaskan pada Tabel 3.5.



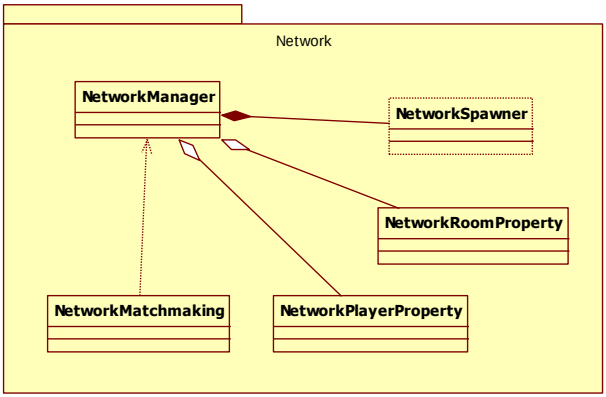
Gambar 3.11 *Package Game Rules*

Tabel 3.6 Deskripsi *Package Game Rules*

Nama Kelas	Deskripsi
GameManager	Sebagai komponen global yang berperan untuk mengatur jalannya permainan. Kelas ini berisi aturan main pada tipe permainan *offline* maupun *online*

3.2.3.6 *Package Network*

Pada *package* ini, seperti gambaran pada Gambar 3.12 memiliki peranan untuk mengintegrasikan *basic gameplay* dengan Photon Cloud menggunakan Photon Unity Networking. Penjelasan tiap komponen akan dijelaskan pada Tabel 3.7.



Gambar 3.12 *Package Network*

Tabel 3.7 Deskripsi *Package Network*

Nama Kelas	Deskripsi
NetworkManager	Sebagai komponen global untuk menangani fungsional *method* dari koneksi *server* dan aturan protokol permainan *online*
NetworkMatchmaking	Sebagai komponen untuk melakukan pencarian *room*, alokasi *room* dimana komponen ini ditujukan untuk mempertemukan antar pemain yang melakukan *Find Match*
NetworkSpawner	Sebagai komponen untuk menginstansiasikan objek *network*, agar setiap objek yang berada dalam *scene* selalu tersinkronisasi
NetworkRoomProperty	Sebagai komponen untuk menyimpan data string *unique*

	dari data *room* permainan
NetworkPlayerProperty	Sebagai komponen untuk menyimpan data string *unique* dari data pemain

3.2.4 Perancangan *Class Diagram*

Sebelum memasuki tahapan implementasi kode, dibutuhkan gambaran keterkaitan antar kelas menggunakan diagram kelas. Penjelasan tiap diagram kelas akan dijelaskan pada subbab 3.2.4.1, subbab 3.2.4.2 dan subbab 3.2.4.3.

3.2.4.1 *Class Diagram* Pemain

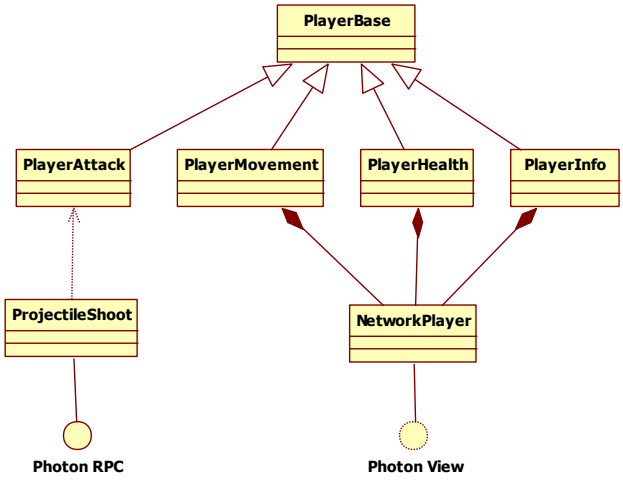
Pada objek karakter pemain, akan diimplementasikan sesuai dengan rancangan diagram kelas seperti pada Gambar 3.13.

Dari generalisasi PlayerBase, terdapat kelas yang memiliki fungsional sebagai *behaviour* terhadap objek karakter pemain.

Pada bagian kelas PlayerMovement, PlayerHealth dan PlayerInfo mengomposisikan kelas NetworkPlayer untuk pemanggilan *method* pada Photon dalam peranan sinkronisasi menggunakan data *stream write* dan *read* (*state serialization*) agar terjadi pertukaran data atribut pada kelas PlayerMovement, PlayerHealth dan PlayerInfo.

Photon View akan ditambahkan ke dalam objek aset sebagai syarat *state serialization*. Kemudian pada tiap kelas akan menggunakan komponen tersebut melakukan streaming data ke *server* dimana data-data yang dikirim akan diteruskan pada klien yang terhubung dalam satu *room* yang sama.

Photon *RPC* akan dipanggil oleh kelas ProjectileShoot untuk sinkronisasi dalam *remote access* sebuah *method* dalam kelas. Jika *method RPC* dipanggil, maka Photon akan melakukan sinkronisasi sesuai dengan target klien yang ditentukan dari *interface* Photon Targets.



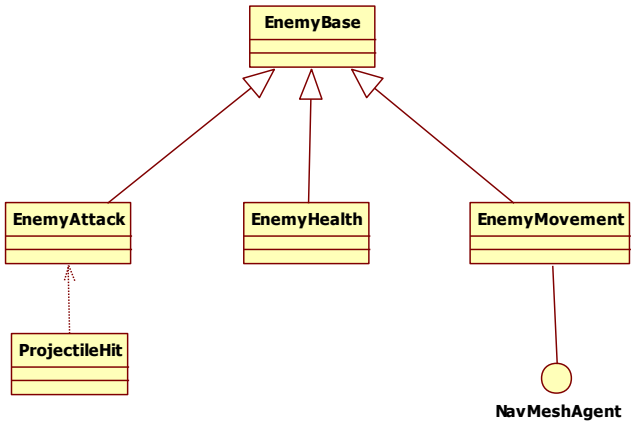
Gambar 3.13 *Class Diagram* Pemain

3.2.4.2 *Class Diagram* AI/BOT

Pada objek AI/BOT, akan diimplementasikan sesuai dengan rancangan diagram kelas seperti pada Gambar 3.14.

Objek AI/BOT akan menggunakan *NavMeshAgent* untuk melacak karakter pemain yang berada dalam jangkauan dan sedang berpijak pada *terrain* yang telah di *bake* oleh *Navigation Mesh Renderer*. Pelacakan lokasi pemain akan diimplementasikan pada kelas EnemyMovement.

Pada kelas EnemyAttack, akan melakukan pengecekkan jarak objek AI/BOT terhadap karakter pemain, jika target sesuai dengan jarak sisa yang ditentukan oleh *NavMeshAgent*, maka objek AI/BOT akan memanggil kelas ProjectileHit untuk melakukan serangan.



Gambar 3.14 *Class Diagram* AI/BOT

3.2.4.3 *Class Diagram Network*

Rancangan kelas diagram ini merupakan bagian penting dari fitur fungsionalitas *Synchronous Multiplayer* dan akan diimplementasikan sesuai dengan rancangan diagram kelas seperti pada Gambar 3.15.

Pada kelas NetworkManager, menggunakan metode pola perancangan *singleton* dikarenakan komponen ini akan berjalan terus-menerus dan tidak akan berhenti selama mode *Synchronous Multiplayer* sedang berlangsung.

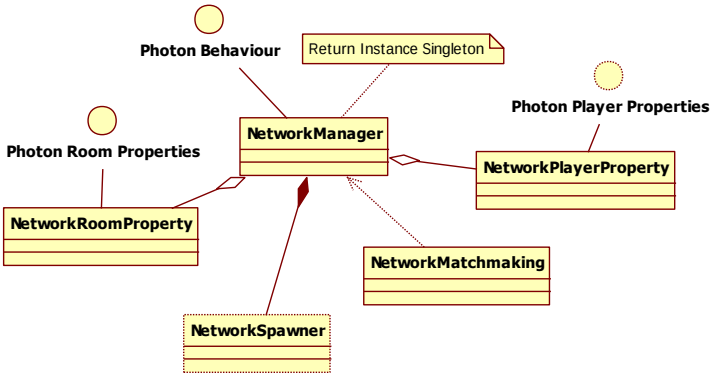
Pada kelas NetworkMatchmaking, kelas ini akan menunggu *event* dari kelas NetworkManager. Ketika event dipanggil oleh NetworkManager. NetworkMatchmaking tidak dapat berjalan tanpa dikendalikan oleh NetworkManager.

Pada kelas NetworkSpawner berfungsi untuk melakukan instansiasi objek karakter pemain dan penghidupan kembali jika karakter pemain dalam keadaan *state* mati.

Photon memiliki fitur penyimpanan *state* sementara pada *server*-nya (Photon Cloud), fitur tersebut beguna untuk menyimpan informasi *room*, skor *leaderboard* tiap pemain,

aktifitas di tiap klien. Fitur tersebut yaitu Photon Room Properties dan Photon Player Properties. Perbedaannya adalah, penyimpanan dalam Photon Room Properties dapat diketahui nilainya oleh klien dalam satu *room* yang sama, tetapi jika ingin mendapatkan nilai informasi dalam sebuah *room* melalui Photon Room Properties, maka *masterclient* di *room* yang bersangkutan harus melakukan pengaturan pada *method* CustomPropertiesForLobby yang terdapat pada Photon. Sedangkan Photon Player Properties hanya dapat diakses ketika dalam satu *room* yang sama.

NetworkPlayerProperty dan NetworkRoomPropery bertugas untuk menyimpan nilai *unique* yang terdapat pada Photon Player Properties dan Photon Room Properties.



Gambar 3.15 *Class Diagram Network*

3.2.5 Perancangan AI/BOT

Perancangan ini mengacu pada tinjauan pustaka pada bagian 2.6. pembuatan AI/BOT memanfaatkan fitur unity Navmesh Agent untuk melakukan *pathfinding* dengan melakukan *generate baking terrain* yang akan berfungsi untuk pendeteksian titik pijak karakter agar AI/BOT dapat melakukan pencarian karakter. Berikut lingkup perancangan.

Kondisi Nilai Jarak	Keterangan
Jarak >= 1.5 (dalam satuan tipe data float)	AI/BOT melakukan pengejaran terhadap karakter pemain
Jarak < 1.5 (dalam satuan tipe data float	AI/BOT melakukan penyerangan terhadap pemain

3.2.6 Perancangan Masukkan Data Sinkronisasi

Perancangan ini adalah mekanisme sinkronisasi dari masukkan data ketika sebuah klien melakukan *streaming* data ke Photon Cloud. Berdasarkan dokumentasi pada Photon (dijelaskan pada tinjauan pustaka pada bagian Photon Data *Serialization*), tidak semua tipe didukung oleh Photon. Pada aplikasi permainan ini akan dijabarkan tipe data apa saja yang dijadikan input untuk *streaming* data untuk setiap tujuan sinkronisasi. Berikut detil penjabaran pada Tabel 3.8.

Tabel 3.8 Detil *Stream Serialization*

Nama Sinkronisasi	Detil	Data *Stream*
Pergerakkan Karakter	Posisi	Vector3 (float, float, float)
	Rotasi	Vector3 (float, float, float)
	Animasi	Bool
***Projectile* Anak Panah**	Posisi	Vector3 (float, float, float)
	Rotasi	Vector3 (float, float, float)
	Animasi	Bool
Healthbar	Kapasitas Health	Float

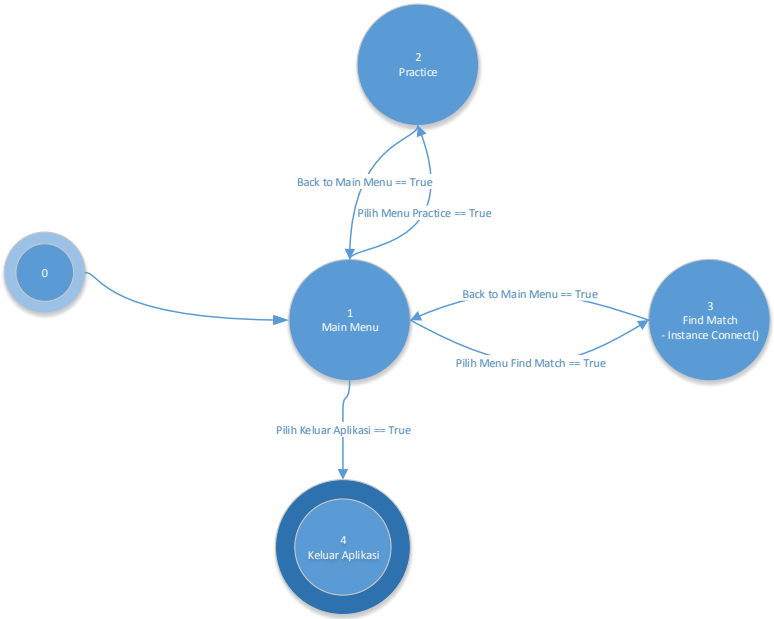
Ada beberapa kasus data tidak diperlukan untuk melakukan *data serialization* seperti pemanggilan fungsi secara *remote* (Penggunaan Photon RPC), klien tidak melakukan *data serialization*, melainkan secara lokal. Dikarenakan fungsi seperti instansiasi dilakukan secara *remote* sehingga pada setiap klien akan secara lokal memanggil fungsi dan setiap klien akan terlihat sinkron walaupun tidak dilakukan *data serialization*. Hal itu bertujuan untuk menghemat dan menjaga performa dari jalannya permainan yang *synchronous*. Untuk rancangan sinkronisasi pada *leaderboard*, digunakan pemanfaatan Photon Room Properties dan Photon Player Properties dalam penerapan sinkronisasi.

3.2.7 Perancangan Proses Permainan

Pada rancangan proses permainan ini akan dijelaskan mengenai proses yang terjadi dalam permainan. Proses terdiri dari alur *state* pada *gameplay* yang terintegrasi pada *state synchronous* maupun pada *offline mode*.

3.2.7.1 Perancangan Proses Navigasi *Scene*

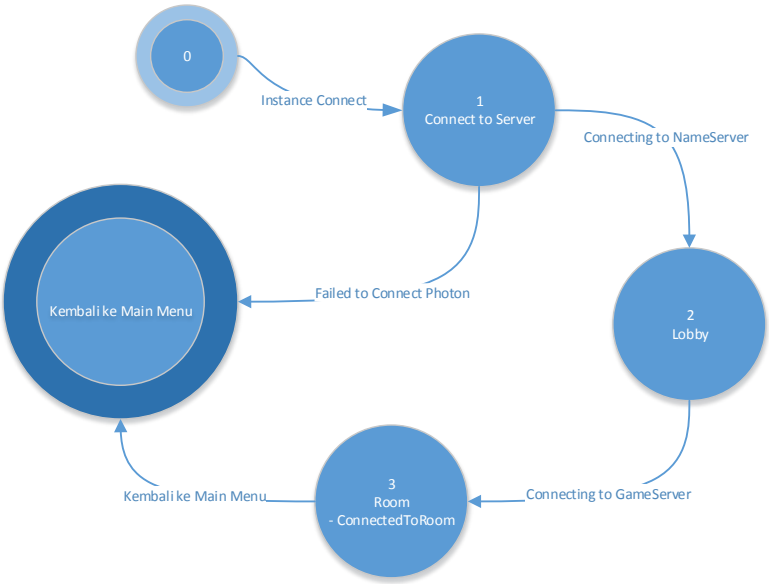
Proses ini merupakan proses perpindahan sistem menu aplikasi yang terdiri dari menu utama, *practice*, *find match* dan untuk keluar dari permainan. Selama aplikasi berlangsung, pemain dapat berinteraksi dengan menu sesuai dengan fungsional di tiap *scene*. Aplikasi permainan akan mengatur kondisi yang dipisahkan oleh *loading screen* pada tiap *scene*. Berikut penjelasan alur proses pada Gambar 3.16.



Gambar 3.16 Perancangan Proses Navigasi *Scene*

3.2.7.2 Perancangan Proses Koneksi ke Photon Cloud

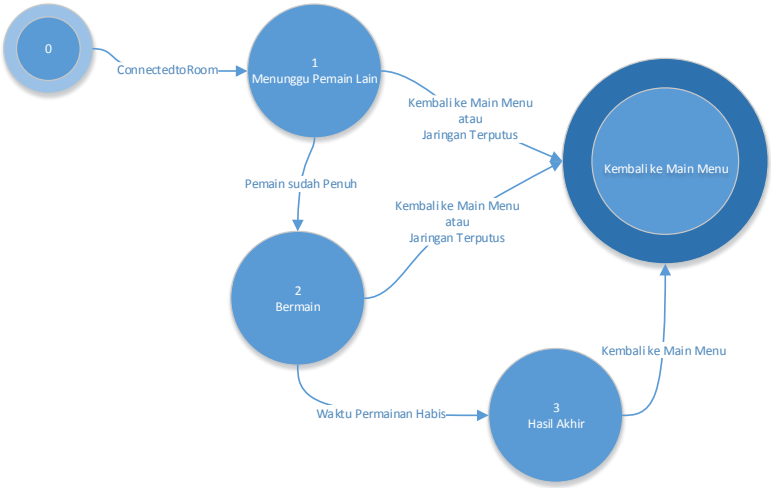
Proses ini merupakan proses terhubungnya aplikasi permainan ke *cloud server*. Proses ini terdiri dari *state-state* yang terdapat pada *cloud server* dan penulis memanfaatkan *state* yang ada untuk mempertemukan antar pemain (*matchmaking*). Berikut penjelasan alur proses pada Gambar 3.17.



Gambar 3.17 Perancangan Proses Koneksi ke Photon Cloud

3.2.7.3 Perancangan Proses *Gameplay*

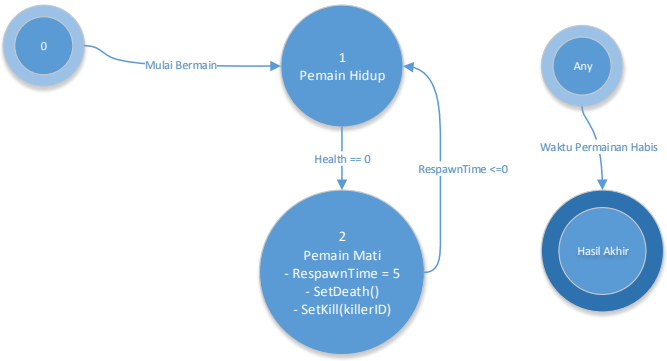
Proses ini merupakan proses berjalannya permainan ketika menunggu pemain yang akan terhubung ke dalam *room* yang sama dan ketika sebuah *room* sudah diisi dengan pemain dengan ketentuan maksimal, maka permainan akan dimulai. Pemain akan bermain sampai waktu yang disediakan habis dan hasil permainan akan di *broadcast* ke semua klien ketika waktu permainan habis dan akan di arahkan kembali ke menu utama permainan. Berikut alur proses pada Gambar 3.18.



Gambar 3.18 Perancangan Proses *Gameplay*

3.2.7.4 Perancangan Proses *Lifecycle* Karakter

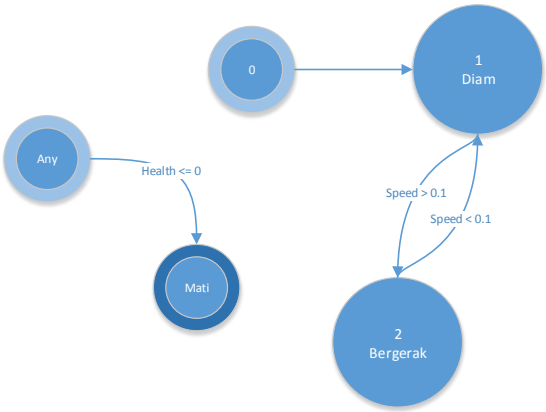
Proses ini merupakan menentukan skor pemain berupa pencapaian berhasilnya membunuh pemain lain dan berapa kali dibunuh. Jika pemain berhasil dibunuh, maka akan ditambahkan nilai *death* pada scoreboard dan mendapatkan *ID* pembunuh untuk menambahkan nilai *kill* bagi yang berhasil membunuh dan ditambahkan nilai *kill* pada *scoreboard*. Berikut penjelasan alur proses pada Gambar 3.19.



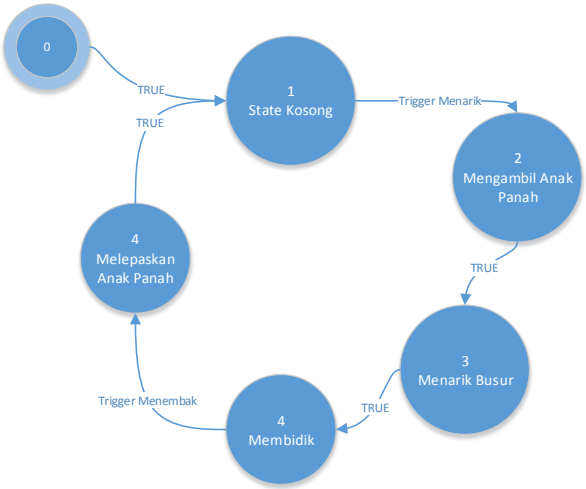
Gambar 3.19 Perancangan Proses *Lifecycle* Karakter

3.2.7.5 Perancangan Proses Mekanik Karakter

Proses ini menjelaskan bagaimana karakter dapat melakukan pergerakan, transisi animasi, sistem pertarungan. Mekanik karakter ini menggunakan dua layer *finite state machine* dan saling terintegrasi. Pada Unity Engine, integrasinya menggunakan teknik *Skeleton Mask*. Berikut penjelasan tentang proses mekanik karakter pada Gambar 3.20 dan Gambar 3.21



Gambar 3.20 Perancangan Proses Mekanik Karakter (Layer 1)



Gambar 3.21 Perancangan Proses Mekanik Karakter (Layer 2)

3.2.8 Perancangan Kontrol Permainan

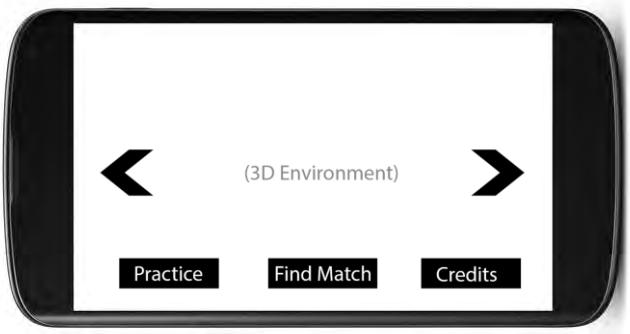
Terdapat beberapa jenis kontrol ketika bermain sebagai HUD (*Head Under Display*) yaitu menggunakan *virtual analog* dengan menyentuhnya dan menggeser arah *axis* untuk mengendalikan karakter. *Virtual button* untuk menembakkan busur dan melihat *scoreboard*.

3.2.9 Perancangan Antarmuka Permainan

Subbab ini membahas bagaimana rancangan antarmuka permainan yang akan digunakan untuk tugas akhir. Rancangan antarmuka yang dibahas berjumlah satu buah dimana terdiri dari 6 tipe yaitu: menu utama, Pertarungan, *Popup* Konfirmasi, *Loading Screen*, Informasi *Scoreboard* dan Hasil Pertarungan.

3.2.9.1 Perancangan Antarmuka Menu Utama

Pada antarmuka ini, aplikasi permainan akan menampilkan menu untuk mengganti arena pertarungan, memilih mode pertarungan *offline practice*, melakukan *matchmaking* pada *find match* dan *credits* untuk menampilkan informasi pengembang. Berikut ilustrasi perancangan antarmuka pada Gambar 3.22.



Gambar 3.22 Perancangan Antarmuka Menu Utama

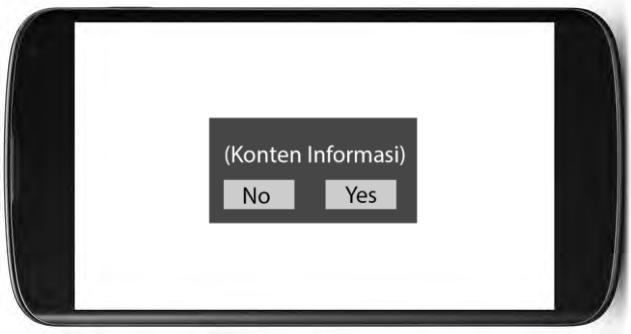
3.2.9.2 Perancangan Antarmuka Pertarungan

Pada antarmuka ini, akan menampilkan HUD (*Head Under Display*) yang terdiri dari *virtual analog* untuk mengontrol karakter, tombol *shoot* untuk menembakan busur dan tombol piala untuk menampilkan *scoreboard*. Berikut ilustrasi perancangan antarmuka pada Gambar 3.23.

Gambar 3.23 Perancangan Antarmuka Pertarungan

3.2.9.3 Perancangan Antarmuka *Popup* Konfirmasi

Pada antarmuka ini, merupakan *template* yang akan ditampilkan untuk meminta konfirmasi kepada pemain seperti konfirmasi untuk kembali ke menu utama sebelum dan keluar dari aplikasi. Berikut ilustrasi perancangan antarmuka pada Gambar 3.24.



Gambar 3.24 Perancangan Antarmuka *Popup* Konfirmasi

3.2.9.4 Perancangan Antarmuka *Scoreboard*

Pada antarmuka ini, menampilkan informasi mengenai nama-nama pemain yang terhubung dalam satu *room*, melihat skor banyaknya membunuh lawan dan berapa banyak pemain

dibunuh oleh pemain lawan. Berikut ilustrasi perancangan antarmuka pada Gambar 3.25.

Gambar 3.25 Perancangan Antarmuka *Scoreboard*

3.2.9.5 Perancangan Antarmuka Hasil Pertarungan

Pada antarmuka ini menampilkan hasil akhir pertarungan ketika waktu pertarungan berakhir. Setiap pemain akan mengetahui siapa yang memenangkan pertarungan, berapa total kill yang diperoleh dan satu tombol untuk kembali ke menu utama. Berikut ilustrasi perancangan antarmuka pada Gambar 3.26.

Gambar 3.26 Perancangan Antarmuka Hasil Pertarungan

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan aplikasi permainan. Di dalamnya mencakup proses penerapan dan pengimplementasian algoritma, init permainan dan antarmuka yang mengacu pada rancangan yang telah dibahas sebelumnya.

4.1 Lingkungan Implementasi

Lingkungan implementasi tugas akhir dijelaskan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat Keras	Prosesor : - Intel(R) Core(TM) i3-2367M CPU @ 1.40GHz Memori : - 4 GB
Perangkat Lunak	Sistem Operasi : - Microsoft Windows 7 Home Premium 64-bit Perangkat Pengembang : - Unity Engine - Mono Develop

4.2 Implementasi *Gameplay* Permainan

Pada subbab ini akan dibahas implementasi mengenai pembuatan mekanik karakter dan aturan main dalam aplikasi permainan .

4.2.1 Implementasi Pergerakkan Pemain

Implementasi pembuatan pergerakkan pemain dituliskan pada Gambar 4.1. Pada pergerakkan pemain ini, membutuhkan variabel untuk menampung nilai x, y dan z pada 3D *world space*. Kemudian kontrol input akan mendapatkan nilai axis yang akan

diproyeksikan untuk mengetahui ke arah mana karakter pemain akan bergerak.

```
64  movement.Set(ETCInput.GetAxis("Horizontal"), 0f,
    ETCInput.GetAxis("Vertical"));
65  movement = movement.normalized * speed *
    Time.deltaTime;
66  playerRigidbody.MovePosition(transform.position +
    movement);
67  if (movement != Vector3.zero)
68  {
69      targetRotation =
    Quaternion.LookRotation(movement);
70      transform.eulerAngles = Vector3.up *
    Mathf.MoveTowardsAngle(transform.eulerAngles.y,targetR
    otation.eulerAngles.y, rotationSpeed * Time.deltaTime
    * 10);
71      anim.SetBool("IsMove", true);
72  }
73  else
74  {
75      anim.SetBool("IsMove", false);
76  }
```

Gambar 4.1 Implementasi Pergerakkan Pemain

4.2.2 Implementasi Pemain Menyerang

Implementasi pembuatan agar pemain dapat melakukan serangan terhadap lawan dituliskan pada Gambar 4.2. Konsep dalam mengimplementasikan pemain menyeran yaitu ketika inputan *virtual button* ditekan, maka objek untuk membidik target akan dimunculkan dan ketika melepaskan tombol yang ditekan, sistem akan memanggil fungsi untuk instansiasi objek panah dan menembakkannya.

```
58  if(ETCInput.GetButtonDown("Attack"))
59  {
60      crosshair.SetActive(true);
61  }
62
```

```
63  if(ETCInput.GetButtonUp("Attack") && canShot)
64  {
65      canShot = false;
66      time = 0;
67      InvokingArrow();
68      crosshair.SetActive(false);
69  }
```

Gambar 4.2 Implementasi Pemain Menyerang

4.2.3 Implementasi *Lifecycle* Pemain dan AI

Implementasi pembuatan daur hidup dan mati karakter pemain dan AI dituliskan pada Gambar 4.3. Ketika objek terkena tumbukkan oleh *projectile*, maka *health* atribut akan dikurangi sebesar *damage*.

```
38  anim.SetTrigger("Hit");
39  health -= damage;
40  if(health <= 0)
41      Die(projectileID);
```

Gambar 4.3 Implementasi *Lifcycle* Pemain dan *AI*

4.2.4 Implementasi AI *Pathfinding*

Implementasi pembuatan pencarian untuk melacak lokasi karakter pemain dan pergerakkan *AI* dituliskan pada Gambar 4.4. Pada implementasi ini, digunakan komponen *NavMeshAgent* untuk melacak sebuah objek yang berada pada *ground* yang telah di *bake* oleh fitur navigasi pada Unity Engine, kemudian NavMeshAgent akan memberi kalkulasi jarak terhadap target.

```
12  void Awake ()
13  {
14      player = GameObject.FindGameObjectWithTag
    ("Player").transform;
15      nav = GetComponent<NavMeshAgent> ();
16      anim = GetComponent<Animator> ();
17  }
18
19  void Update ()
20  {
```

```
21      transform.LookAt(GameObject.Find
("RangerArcherOffline").transform.position);
22      nav.SetDestination (player.position);
23      if (nav.remainingDistance > 1.5f)
24      {
25          anim.SetBool ("IsMove", true);
26      }
27      else
28      {
29          anim.SetBool("IsMove",false);
30      }
31  }
```

Gambar 4.4 Implementasi *AI Pathfinding*

4.2.5 Implementasi *AI* Menyerang

Implementasi pembuatan *AI* agar dapat menyerang karakter pemain dituliskan pada Gambar 4.5. Pada implementasi ini, *NavMeshAgent* akan mencari informasi jarak sisa terhadap karakter pemain, ketika jarak sisa sudah mendekati pemain, maka sistem akan melakukan instansiasi objek *hit* untuk mengirim *damage* ke pemain.

```
30  time += Time.deltaTime;
31
32  if(time > cooldown)
33  {
34      canHit = true;
35  }
36
37  nav.SetDestination
    (GetComponent<EnemyMovement>().player.position);
38  if (nav.remainingDistance <= 1.5f)
39  {
40      anim.SetTrigger("Attack");
41      if(canHit)
42      {
43          time = 0;
44          canHit = false;
45          Invoke("InstanceHit",0.5f);
```

```
46      }
47  }
```

Gambar 4.5 Implementasi AI Menyerang

4.2.6 Implementasi Aturan Main

Pada implementasi aturan main ini akan terbagi lagi menjadi beberapa subbab. Pada aturan main *offline*, pemain bebas bermain dan bebas menentukan ingin mengakhiri permainan atau tidak dan karakter pemain selalu dalam keadaan *immortal* sedangkan aturan main *online*, berikut akan dijelaskan pada subbab berikut.

4.2.6.1 Implementasi Pemain Menunggu

Implementasi agar pemain diberi batas untuk menunggu pemain lain sampai batas maksimum dituliskan pada Gambar 4.6. Klien akan mengecek total pemain yang ada dalam *room*, ketika total pemain sudah mencapai maksimum maka *timer* siap untuk untuk mulai bermain akan dijalankan.

```
195 Text text = statusPlayers.GetComponent<Text>();
196 text.text = GameManager.roomName +"\n\nWaiting for
    Players "+ totalPlayers.ToString() + "/" +
    maxPlayers.ToString();
197 if(totalPlayers == maxPlayers)
198 {
199     GameManager.checkedIn = false;
200     GameManager.readyToPlay = true;
201     NetworkManager.SetRoomProperty
    (NetworkRoomProperty.Playing, true);
202     NetworkManager.SetRoomProperty
    (NetworkRoomProperty.StartTime, PhotonNetwork.time);
203     GameObject.FindObjectOfType
    <GameManager>().waitTimerStart = true;
204 }
```

Gambar 4.6 Implementasi Pemain Menunggu

4.2.6.2 Implementasi Memulai Permainan

Implementasi agar pemain diberi batasan waktu selama bermain dituliskan pada Gambar 4.7. Implementasi ini berada pada *state* ketika *timer* untuk bersiap main dimulai. Kontrol input akan diaktifkan dan berhitung mundur untuk siap memulai permainan.

```
177  Text text = statusPlayers.GetComponent<Text>();
178  text.text = GameManager.roomName +"\n\nGame Will Start
     in "+ timer.ToString();
179
180  if(timer <= 0)
181  {
182      ActivatingController();
183      NetworkSpawner.RelocateSpawnPlayer(playCamera);
184      NetworkManager.SaveState();
185      NetworkManager.SetRoomProperty
     (NetworkRoomProperty.StartTime, PhotonNetwork.time);
186      GameObject.FindObjectOfType
     <GameManager>().waitTimerStart = false;
187      GameObject.FindObjectOfType
     <GameManager>().battleTimerStart = true;
188  }
```

Gambar 4.7 Implementasi Memulai Permainan

4.2.6.3 Implementasi Penghidupan Kembali Karakter

Implementasi agar karakter pemain selalu kembali hidup ketika dibunuh oleh lawannya dituliskan pada Gambar 4.8. Implementasi ini berjalan ketika di panggil oleh komponen *lifecycle* pemain dan *respawn timer* akan berhitung mundur sampai pada *respawn timer* < 0 maka karakter akan hidup kembali.

```
66  if(respawnTimer > 0)
67  {
68      respawnTimer -= Time.deltaTime;
69      GameObject statusPlayer =
     GameObject.FindObjectOfType<GUIManager>
     ().statusPlayers;
```

```
70      statusPlayer.transform.parent.gameObject.
SetActive(true);
71      statusPlayer.GetComponent<Text> ().text = "You're
Dead\n\nRespawning in " +
((int)respawnTimer).ToString() + "s";
72
73      if(respawnTimer <= 0)
74      {
75          GameObject.FindObjectOfType<NetworkSpawner>
().RespawnPlayer();
76          statusPlayer.transform.parent.gameObject.
SetActive(false);
77      }
78  }
```

Gambar 4.8 Implementasi Penghidupan Kembali Karakter

4.2.6.4 Implementasi Mengakhiri Permainan

Implementasi untuk mengakhiri permainan ketika waktu yang disediakan habis dituliskan pada Gambar 4.9. Implementasi ini akan mengecek waktu yang ada pada Photon Cloud akan selalu tersinkronisasi dengan pemain lainnya. Kemudian saat waktu habis, maka permainan akan memanggil fungsi untuk menunjukkan hasil akhir.

```
86  if(battleTimerStart)
87  {
88      double timePassed = PhotonNetwork.time - (double)
PhotonNetwork.room.customProperties
[NetworkRoomProperty.StartTime];
89      timeRemainingBattle = totalBattleTime - (int)
timePassed;
90      GameObject battleTime =
GameObject.FindObjectOfType <GUIManager>().battleTime;
91      int minute = timeRemainingBattle / 60;
92      int seconds = timeRemainingBattle % 60;
93      battleTime.GetComponent <Text>().text = "Battle
Time : " + minute + "m " + seconds + "s";
94
95      if(timeRemainingBattle <=0) battleTimerStart =
```

```
        false;
 96
 97         if(!battleTimerStart)
 98         {
 99             PhotonNetwork.Disconnect();
100         }
101     }
```

Gambar 4.9 Implementasi Mengakhiri Permainan

4.3 Implementasi Jaringan Permainan

Untuk merealisasikan sinkronisasi pada aplikasi permainan, maka dibutuhkan pengatur jaringan untuk mengatur *event-event* yang ada pada Photon dan merealisasikan algoritma *matchmaking* untuk mempertemukan antar pemain. Pemain dan *room* memilii sebuah penyimpanan nilai data pada *custom properties*. *Property* tersebut akan dimanfaatkan untuk menyimpan nilai-nilai untuk sinkronisasi yang tidak dieksekusi secara terus-menerus. Pada setiap pemain terdapat komponen Photon View yang akan bekerja untuk *state serialization* dan objek sinkronisasi secara kontinu.

4.3.1 Implementasi *Matchmaking*

Implementasi untuk mempertemukan antar pemain ke dalam sebuah *room* dituliskan pada Gambar 4.10 dan 4.11. Berdasarkan pada rancangan algoritma, pencarian alur pertama adalah mencari *room* yang tersedia, jika *room* belum ada sama sekali maka dilakukan pembuatan *room* pertama, jika *room* tersedia tetapi tidak statusnya sedang bermain maka dilakukan fungsi realokasi *matchmaking* seperti yang ada pada Gambar 4.11.

```
65  if(NetworkManager.roomList.Length !=0)
66  {
67      for(int i=0; i<NetworkManager.roomList.Length; i++)
68      {
69          bool playing = (bool)
    NetworkManager.roomList[i].customProperties
    [NetworkRoomProperty.Playing];
70          if(!playing)
```

```
71          {
72              string roomNum = NetworkManager.roomList[i].name.Replace("RC Room ","");
73              PhotonNetwork.JoinRoom("RC Room "+roomNum);
74              return;
75          }
76      }
77      RealocateRoomMatchmaking();
78  }
79  else
80  {
81      SetRoomData();
82      PhotonNetwork.JoinOrCreateRoom("RC Room 1", roomOptions, TypedLobby.Default);
83  }
```

Gambar 4.10 Implementasi Matchmaking (1)

```
93  public static void RealocateRoomMatchmaking()
94  {
95      SetRoomData();
96      for(int i=0; i<=NetworkManager.roomList.Length; i++)
97      {
98          if(NetworkManager.roomList.Length - i == 0)
99          {
100             PhotonNetwork.CreateRoom("RC Room " + (i+1).ToString(), roomOptions, TypedLobby.Default);
101             return;
102         }
103         if((string) NetworkManager.roomList[i].customProperties [NetworkRoomProperty.RoomName] != ("RC Room"+ (i+1).ToString()))
104         {
105             PhotonNetwork.CreateRoom("RC Room " + (i+1).ToString(), roomOptions, TypedLobby.Default);
106         }
107     }
108 }
```

Gambar 4.11 Implementasi Matchmaking (2)

4.3.2 Implementasi Sinkronisasi Karakter Pemain

Implementasi untuk sinkronisasi data pemain dituliskan pada Gambar 4.12. Implementasi ini akan melakukan *streaming* pertukaran data secara *Reliable Data Compressed* untuk menjamin bahwa data yang akan dikirim pasti sampai.

```
18  public void OnPhotonSerializeView(PhotonStream stream,
    PhotonMessageInfo info)
19  {
20      if (stream.isWriting)
21      {
22          stream.SendNext(transform.position);
23          stream.SendNext(transform.rotation);
24          stream.SendNext(anim.GetBool("IsMove"));
25          stream.SendNext(health);
26      }
27      else
28      {
29          correctPlayerPos =
    (Vector3)stream.ReceiveNext();
30          correctPlayerRot =
    (Quaternion)stream.ReceiveNext();
31          anim.SetBool("IsMove",
    (bool)stream.ReceiveNext());
32          health = (float) stream.ReceiveNext();
33      }
34  }
```

Gambar 4.12 Implementasi Sinkronisasi Karakter Pemain

4.3.3 Implementasi Sinkronisasi *Projectile*

Implementasi untuk melakukan *remote* karakter *network* agar *projectile* tersinkronisasi dituliskan pada Gambar 4.13. Implementasi ini menggunakan *Remote Procedure Call* yang berfungsi untuk memanggil sebuah fungsi pada karakter *network* yang bukan miliknya.

```
32  void InstanceArrow()
33  {
34      projectile = Instantiate(projectilePrefab,
```

```
    projectileSpawn.position, Quaternion.identity)
asGameObject;
35      projectile.transform.position =
projectileSpawn.position;
36      projectile.transform.rotation =
projectileSpawn.rotation;
37
projectile.GetComponent<WeaponShoot>().attackDeliver =
attackDamage;
38      projectile.GetComponent<WeaponShoot>().projectileID
= GetComponent<PhotonView>().viewID;
39  }
40
41  [RPC]
42  void InvokingArrowRPC()
43  {
44      anim.SetTrigger("Attack");
45      Invoke("InstanceArrow", 0.5f);
46  }
```

Gambar 4.13 Implementasi Sinkronisasi *Projectile*

4.3.4 Implementasi Sinkronisasi *Leaderboard*

Implementasi agar papan *leaderboard* selalu terupdate dituliskan pada Gambar 4.14. Implementasi ini memanggil property dari pada tiap pemain lain dan dirinya sendiri yang tersimpan di Photon Cloud.

```
106 for(int i=0; i<PhotonNetwork.playerList.Length; i++)
107 {
108     guiManager.scorePanel[i].SetActive(true);
109     if(PhotonNetwork.playerList[i].isLocal)
110     {
111         guiManager.scorePanel[i].transform.Find
("PlayerName").GetComponent<Text>().color =
Color.blue;
112     }
113     else
114     {
115         guiManager.scorePanel[i].transform.Find
("PlayerName").GetComponent<Text>().color = Color.red;
```

```
116     }
117     guiManager.scorePanel[i].transform.Find
("PlayerName").GetComponent<Text>().text =
PhotonNetwork.playerList[i].name;
118     if(PhotonNetwork.playerList[i].customProperties
[NetworkPlayerProperty.Kills] != null &&
119         PhotonNetwork.playerList[i].customProperties
[NetworkPlayerProperty.Deaths] != null)
120     {
121         guiManager.scorePanel[i].transform.Find
("PlayerKill").GetComponent<Text>().text = ((int)
PhotonNetwork.playerList[i].customProperties
[NetworkPlayerProperty.Kills]).ToString();
122         guiManager.scorePanel[i].transform.Find
("PlayerDeath").GetComponent<Text>().text = ((int)
PhotonNetwork.playerList[i].customProperties
[NetworkPlayerProperty.Deaths]).ToString();
123     }
124 }
```

Gambar 4.14 Implementasi Sinkronisasi *Leaderboard*

4.4 Implementasi Antarmuka

Implementasi antarmuka menggunakan UGUI (Unity GUI) yang ada pada Unity, memanfaatkan anchor posisi dari layar sehingga dapat beradaptasi ke berbagai ukurang yang memiliki aspek rasio sama. Berikut hasil implementasi antarmuka yang telah direncanakan pada bagian perancangan antarmuka.

Gambar 4.15 Implementasi Antarmuka Menu Utama

Pada Gambar 4.15, implementasi antarmuka ini terdiri dari pilihan menu *Practice*, *Find Match* dan *Credits*. Jika pemain memilih menu Practice, maka akan diarahkan pada *scene* permainan *offline* dan bertarung melawan AI/BOT. Jika pemain memilih menu *Find Match*, maka akan diarahkan pada scene permainan *online* dan melakukan matchmaking untuk bertemu dengan pemain-pemain lain dan bertarung melawannya. Jika pemain memilih menu *Credits*, maka akan muncul *popup* informasi mengenai pengembang.

Gambar 4.16 Implementasi Antarmuka *Gameplay*

Pada Gambar 4.16, implementasi antarmuka ini terdiri HUD (*Head Under Display*) perangkat *virtual controller* untuk mengendalikan pemain, *virtual buton* untuk membidik sasaran dan menembakkan *projectile*. Pada *Offline Mode Practice* tidak ada *virtual button* untuk menampilkan *scoreboard*

Pada Gambar 4.17, implementasi ini berfungsi untuk mengatur perpindahan *scene*, metode yang digunakan yaitu dengan menunggu *scene* jika progres aset yang sudah di*load* secara keseluruhan, maka objek *loading screen* akan langsung dihancurkan dan *scene* langsung berubah

Gambar 4.17 Implementasi Antarmuka Loading Screen

Pada Gambar 4.18, implementasi antarmuka ini terjadi ketika player sedang menunggu pemain lain untuk bermain, ketika karakternya menunggu dan ketika menunggu karakter pemain untuk *respawning*. Terdiri dari informasi teks dan *overlay* pada HUD (*Head Under Display*) agar terlihat ada perbedaan sudut pandang kamera yang sedang *standby*.

Gambar 4.18 Implementasi Antarmuka *Waiting Room*

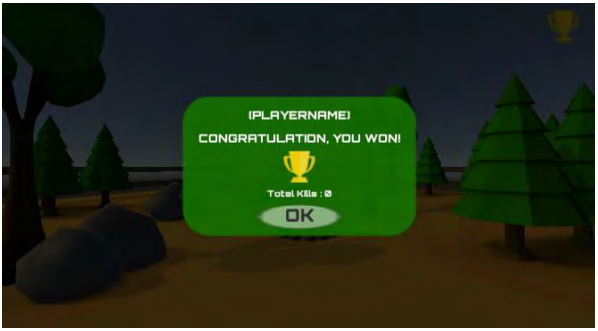
Pada Gambar 4.19, implementasi antarmuka ini cukup sederhana dengan memberikan *toggle* HUD (*Head Under Display*) yang muncul ketika tombol untuk memunculkan scoreboard disentuh.

Gambar 4.19 Implementasi Antarmuka Leaderboard

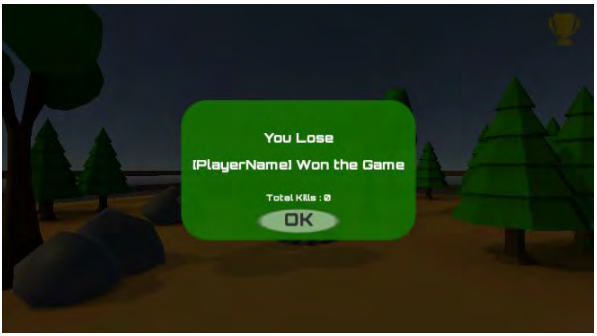
Pada Gambar 4.20, implementasi antarmuka ini merupakan cara untuk memberikan konfirmasi terhadap pemain. Konfirmasi ini akan muncul ketika pemain menekan tombol *back* dari sistem Android.

Gambar 4.20 Implementasi Antarmuka Konfirmasi *Popup*

Pada Gambar 4.21 dan Gambar 4.22, implementasi antarmuka ini merupakan noftifikasi ketika permainan telah berakhir dan memberi tahu siapakah pemenang dari permainan dan memberi tahu apakah ia kalah atau tidak. Terdiri dari sebuah panel, teks informasi dan sebuah tombol. Jika tombol disentuh, maka akan mengarahkan pemain ke menu utama.



Gambar 4.21 Implementasi Antarmuka Hasil Akhir Menang



Gambar 4.22 Implementasi Antarmuka Hasil Akhir Menang

BAB V
PENGUJIAN DAN EVALUASI

Bab ini berisi bahasan mengenai uji coba dan evaluasi aplikasi permainan Rangers Companion menggunakan Unity Engine dengan Photon Unity Networking. Uji coba dilakukan menggunakan metode kotak hitam berdasarkan skenario yang telah ditentukan. Uji coba dilakukan terhadap hasil implementasi perangkat lunak yang telah dipaparkan.

5.1 Lingkungan Uji Coba

Proses uji coba dilakukan pada lingkungan yang telah ditentukan. Pada uji coba ini, lingkungan dibedakan menjadi lingkungan perangkat keras dan lingkungan perangkat lunak. Berikut ini dijelaskan mengenai tiap-tiap lingkungan uji coba aplikasi.

5.1.1 Lingkungan Perangkat Keras

Deskripsi perangkat keras untuk proses uji coba perangkat lunak ini dapat dilihat pada Tabel 5.1.

Tabel 5.1 Lingkungan Perangkat Keras

No	Deskripsi
1	Asus Zenfone 5 T00F/T00J RAM 1/2GB
2	Lenovo T900 RAM 2GB
3	Moto E RAM 1GB
4	Smartfren Andromax i2 RAM 512MB
5	Lenovo A706 RAM 1GB
6	Redmi 1S RAM 1GB
7	Lenovo A526 RAM 1GB
8	Samsung Galaxy Note 3 NEO RAM 2GB
10	HTC One M7 RAM 2GB
11	Sony XPERIA T2 1GB

5.1.2 Lingkungan Perangkat Lunak

Deskripsi perangkat lunak untuk proses uji coba dapat dilihat pada Tabel 5.2.

Tabel 5.2 Lingkungan Perangkat Lunak

No	Deskripsi
1	Sistem Operasi Android - Jelly Bean - KitKat - Lollipop

5.2 Skenario Pengujian Fungsionalitas

Pengujian fungsionalitas dilakukan untuk mengetahui kesesuaian keluaran dari tiap tahap atau langkah penggunaan fungsionalitas terhadap skenario yang dipersiapkan. Pada subbab ini, dijelaskan beberapa skenario uji coba aplikasi permainan secara mandiri berdasarkan metode kotak hitam sebagai dasar tolok ukur keberhasilan. Pengujian fungsionalitas yang terdapat pada aplikasi dijabarkan sebagai berikut.

- Pengujian fungsionalitas *Offline Mode Practice* (PF1)
- Pengujian fungsionalitas *Synchronous Multiplayer* (PF2)

Berdasarkan daftar pengujian yang telah disebutkan, dibuat beberapa skenario yang dilakukan pada setiap pengujian tersebut. Penjelasan mengenai cara dan hasil pengujian fungsionalitas perangkat lunak dibahas pada subbab Pengujian.

5.2.1 Pengujian Fungsionalitas *Offline Mode Practice*

Pengujian fungsionalitas *Offline Mode Practice* bertujuan untuk menguji apakah aplikasi permainan ini dapat berjalan *offline* meliputi *basic gameplay* yang sesuai dengan perancangan alur proses permainan dan pemain dapat bermain bersama

AI/BOT yang diciptakan agar pemain dapat berlatih dan mencoba *gameplay* permainan untuk melatih ketangkasan bermainnya.

Ada beberapa objek penting dan fungsionalnya yang harus berjalan dan sangat diutamakan. Berikut objek dan deskripsi fungsional pada Tabel 5.3.

Tabel 5.3 Deskripsi Fungsional PF1

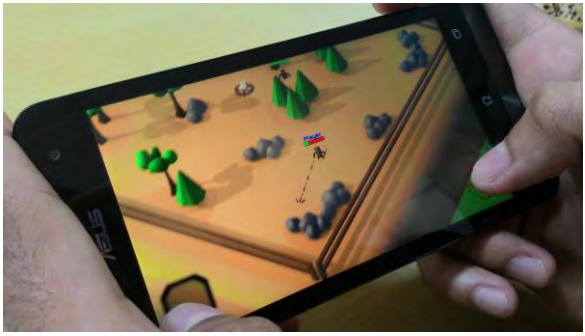
Objek	Deskripsi Fungsional yang harus berjalan
Player	PF1.1 Objek ini harus dapat melakukan pergerakan menggunakan *virtual analog*. Objek ini juga harus dapat melakukan pembidikkan target dan menembakkannya menggunakan *virtual button*. PF1.2 Objek ini harus tetap hidup walaupun komponen *health* sudah habis nilainya dan menyegarkannya kembali komponen *health* sehingga pemain dapat terus bermain dan mencoba.
AI/BOT	PF1.3 Objek ini harus dapat melakukan *pathfinding* untuk mengejar objek pemain. Objek ini harus dapat melakukan serangan jika sudah mendekati objek pemain. PF1.4 Objek ini harus dapat menghidupkan objek baru yang sama (*respawn*) ketika komponen *health* < 0.
Pengatur Antarmuka	PF1.5 Objek ini harus dapat menuntun pemain untuk kembali ke menu utama.

PF1.1 Uji Coba *Virtual Controller*

Fitur ini berlangsung ketika pemain mulai bermain *Practice* dan *Find Match*. Pada umumnya ini merupakan bagian dari *basic gameplay* untuk menggerakkan karakter pemain, membidik dan menembak musuh. Uji coba ini berhasil dalam menggerakkan pemain menggunakan *Virtual Analog* dan *Virtual*

Button seperti pada Gambar 5.1 (menggerakkan karakter pemain), Gambar 5.2 (membidik sasaran), 5.3 (menembakkan projectile).

Gambar 5.1 Uji Coba *Virtual Controller* (1)



Gambar 5.2 Uji Coba *Virtual Controller* (2)

Gambar 5.3 Uji Coba *Virtual Controller* (3)

PF1.2 Uji Coba Penyegaran Kembali Atribut *Health*

Uji coba ini memiliki tingkat keberhasilan 100% karena aplikasi permainan akan selalu meng*update* dan mengecek atribut *health.* Jika *health* < 0 maka akan ter-*update* menjadi 100.

PF1.3 Uji Coba AI *Pathfinding*

Musuh akan selalu mengejar karakter pemain jika jarak dengan karakter pemain jauh. Uji coba ini dilakukan dengan menggerakkan karakter pemain untuk menjauh dari musuh. Musuh akan mengejar dan menyerang ketika jarak dengan karakter pemain sudah dekat. Uji coba ini berhasil, karakter pemain akan diberikan *damage* jika musuh mendekat dan berhasil memukul seperti pada Gambar 5.4.

Gambar 5.4 Uji Coba AI *Pathfinding*

PF1.4 Uji Coba Penghidupan Kembali Musuh

Uji coba ini dilakukan ketika AI/BOT telah kehabisan nilai atribut *health*, kemudian akan dihidupkan kembali ke *spawn point*. Uji coba ini berhasil seperti pada Gambar 5.5.

Gambar 5.5 Uji Coba Penghidupan Kembali Musuh

PF1.5 Uji Coba Kembali ke Menu Utama

Uji coba ini mudah untuk dilakukan, cukup dengan menekan tombol kembali pada device Android, maka akan muncul pilihan popup untuk kembali ke Menu Utama. Uji coba ini berhasil seperti pada transisi Gambar 5.6 ke Gambar 5.7.

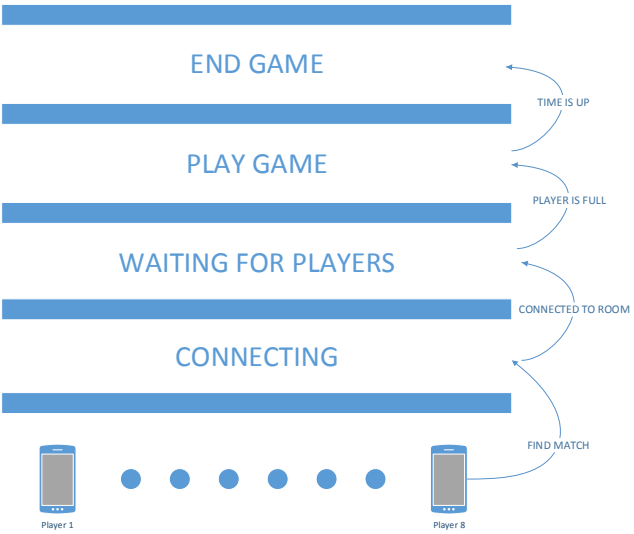
Gambar 5.6 Uji Coba Kembali ke Menu Utama (1)

Gambar 5.7 Uji Coba Kembali ke Menu Utama (2)

5.2.2 Pengujian Fungsionalitas *Synchronous Multiplayer*

Pengujian fungsionalitas *Synchronous Multiplayer* berhubungan dengan pengujian pada subbab 5.2.1 meliputi *basic gameplay* dengan integrasinya pada *state synchronous*. Pengujian fungsionalitas ini bertujuan untuk menguji pengintegrasian Photon Cloud dengan aplikasi permainan yang apabila terintegrasi, maka permainan dapat dimainkan secara *online* dan tersinkronisasi antar klien.

Pada jalannya *gameplay* akan dilakukan uji coba sinkronisasi bagian inti pada pergerakkan, anak panah (*projectile*) dan nilai *health*. Ketika fungsionalitas ini berjalan, aplikasi permainan harus dapat berjalan dengan baik dari *state* awal sampai akhir seperti pada skenario pengujian berikut pada Gambar 5.8.



Gambar 5.8 Tahapan Skenario Pengujian PF2

Gambar menjelaskan tahap-tahap pengujian dalam sebuah konteks bermain Synchronous Multiplayer. Penjabaran tersebut tertera pada Tabel 5.4.

Tabel 5.4 Tahap Pengujian PF2

Tahap	Deskripsi Tahapan Pengujian
PF2.1 Connecting	Pemain memilih tombol *Find Match* dan aplikasi permainan akan menghubungkan ke Photon Cloud. Setelah terhubung, aplikasi permainan harus dapat melakukan matchmaking untuk alokasi *room* untuk mempertemukan antar pemain
PF2.2 Waiting For Players	Ketika pemain terhubung ke sebuah *room* pada *gameserver*, aplikasi permainan harus dapat menerima informasi jumlah total pemain yang telah terhubung ke dalam *room* dan melakukan *trigger*

	memulai permainan ketika pemain sudah penuh.
PF2.3 Play Game	Ketika pemain sudah dinyatakan penuh, aplikasi permainan harus dapat memulai permainan dengan ketentuan sebagai berikut: - Memulai hitung mundur untuk memulai dan tersinkronisasi dengan klien lainnya - Menghitung skor *kill*, *death* dan meng*update leaderboard* yang tersinkronisasi dengan klien lainnya - Menghitung sisa batas waktu permainan dan tersinkronisasi dengan klien lainnya - Melakukan sinkronisasi pada pergerakan tiap pemain, tembakkan *projectile*, *health* pemain dan menghidupkan kembali jika mati.
PF2.4 End Game	Ketika waktu permainan habis, aplikasi permainan harus dapat memberi tahu pemain apakah dia menang atau kalah dan memberikan informasi pemain yang memenangkan permainan.

PF2.1 Uji Coba *Matchmaking*

Pada uji coba ini, menggunakan beberapa klien yang bersamaan memilih menu *Find Match*. Aplikasi permainan harus dapat mengalokasikan *room* permainan dalam mempertemukan antar pemain lainnya secara otomatis. Pada pengujian ini, klien yang mencoba *Find Match*, jika mendapati *room* yang sudah penuh, akan mengalokasikan *room* baru dan pemain tidak bisa

bergabung ke *room* yang statusnya sedang bermain. Uji coba ini berhasil, berikut ilustrasi pada Gambar 5.9 dan 5.10.

Gambar 5.9 Uji Coba *Matchmaking* (1)

Gambar 5.10 Uji Coba *Matchmaking* (2)

PF2.2 Uji Coba Menunggu Pemain

Uji coba ini memastikan ketika pemain berhasil mendapatkan *room*, ia harus menunggu sampai pemain lainnya dapat memenuhi syarat untuk memulai permainan. Uji coba ini berhasil, setiap klien akan menyesuaikan penghitungan total pemain yang ada pada satu *room* dan jika sudah penuh, permainan akan dimulai.

Gambar 5.11 Uji Coba Menunggu Pemain (1)

Gambar 5.12 Uji Coba Menunggu Pemain (2)

PF2.3 Uji Coba Sinkronisasi Permainan

Uji coba ini berlangsung dengan bermain langsung menggunakan berbagai perangkat Android. Ketika pemain menggerakkan karakternya, secara *remote*, karakter tersebut menggerakkannya di klien lainnya, sehingga klien lain dapat melihat pergerakkan yang dilakukan oleh pemain lawannya tersebut. Koneksi internet berpengaruh pada proses sinkronisasi, koneksi yang lambat akan mempengaruhi koneksi yang lancar, maka ketika antar klien saling menembakkan, atribut *health* akan terkalkulasi dengan prediksi oleh Photon, sehingga pemain akan merasakan keanehan ketika dia berhasil menembakkan *projectile* ke lawan, terkadang tidak mengurangi atribut *health* lawan karena mungkin tidak sesuai dengan posisi sebenarnya.

Tabel 5.5 Pengujian Sinkronisasi Pergerakkan

Nama Pengujian	Sinkronisasi Pergerakkan
Tujuan	Melakukan sinkronisasi antar klien pada setiap perubahan posisi objek karakter
Prosedur Pengujian	Salah satu klien pemilik objek melakukan pergerakkan menggunakan virtual *analog*
Hasil yang Diharapkan	Pada setiap klien yang bukan pemilik objek akan melakukan *remote* pergerakkan objek karakter
Total Iterasi Pengujian	10
Jumlah Pengujian Berhasil	10
Jumlah Pengujian Gagal	0
Rata-Rata Keberhasilan	100%

Tabel 5.6 Pengujian Sinkronisasi Anak Panah (*Projectile*)

Nama Pengujian	Sinkronisasi Anak Panah (*Projectile*)
Tujuan	Melakukan sinkronisasi antar klien pada anak panah (*projectile*)
Prosedur Pengujian	Salah satu klien pemilik objek melakukan penembakkan anak panah menggunakan tombol virtual
Hasil yang Diharapkan	Pada setiap klien akan melakukan remote penembakan anak panah (*projectile*)
Total Iterasi Pengujian	10
Jumlah Pengujian Berhasil	9
Jumlah Pengujian Gagal	1
Rata-Rata Keberhasilan	90%

Tabel 5.7 Pengujian Sinkronisasi *Healthbar*

Nama Pengujian	Sinkronisasi *Healthbar*
Tujuan	Melakukan sinkronisasi antar klien pada setiap perubahan nilai dari *healthbar*
Prosedur Pengujian	Salah satu klien pemilik objek melakukan penembakkan anak panah menggunakan tombol virtual, kemudian anak panah (*projectile*) harus bertumbukkan pada karakter lawan
Hasil yang Diharapkan	Pada setiap klien yang bukan pemilik objek akan melakukan *stream* data nilai *health* untuk sinkronisasi *health*
Total Iterasi Pengujian	10
Jumlah Pengujian Berhasil	10
Jumlah Pengujian Gagal	0
Rata-Rata Keberhasilan	100%

Pada uji coba ini aplikasi permainan berhasil melakukan sinkronisasi pergerakkan, *health*, penembakkan *projectile* dan sisa batas waktu permainan pada Gambar 5.13 dan 5.14, serta sinkronisasi *leaderboard* pada Gambar 5.15.

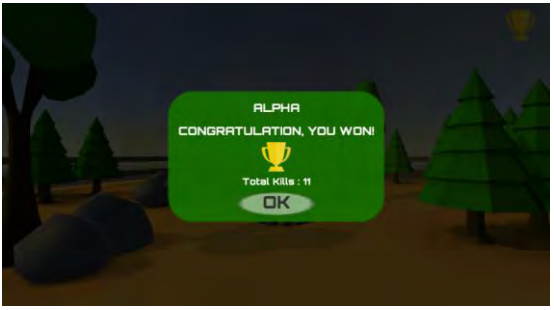
Gambar 5.13 Uji Coba Integrasi Sinkronisasi *Gameplay* (1)

Gambar 5.14 Uji Coba Integrasi Sinkronisasi *Gameplay* (2)

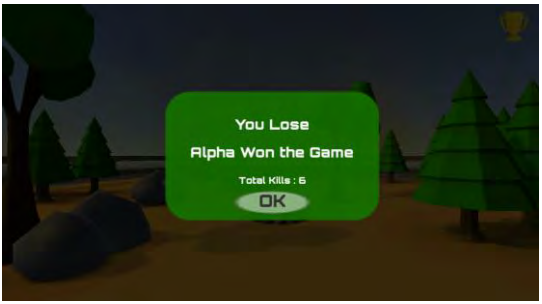
Gambar 5.15 Uji Coba Integrasi Sinkronisasi *Gameplay* (3)

PF2.4 Uji Coba Hasil Akhir Permainan

Uji coba terakhir fungsionalitas dari *Synchronous Multiplayer* yaitu melihat hasil akhir permainan siapa yang memenangkan permainan. Aplikasi permainan ini harus dapat melakukan pemberitahuan pada tiap klien apakah telah memenangkan permainan atau tidak ketika waktu permainan habis. Berikut hasil uji coba pada Gambar 5.16 yang memenangkan permainan dan Gambar 5.17 jika kalah.



Gambar 5.16 Uji Coba Hasil Akhir Permainan (1)



Gambar 5.17 Uji Coba Hasil Akhir Permainan (2)

5.3 Skenario Pengujian Non-Fungsionalitas

Pengujian non-fungsional dilakukan dengan mengacu pada spesifikasi pada bab perancangan yang apabila terpenuhi akan meningkatkan kualitas dari aplikasi permainan ini. Pengujian yang dilakukan yaitu Uji Performa Grafis 3D (PNF1) dan Uji Kehandalan dan Kenyamanan (PNF2).

5.3.1 Uji Coba Pengguna

Dilakukan pengujian terhadap pengguna untuk memberikan timbal balik akurasi penilaian untuk pengujian performa. Berikut Daftar nama penguji pada uji coba pengguna yang dapat dilihat pada

Tabel 5.8 Daftar Pengguna Uji Coba Permainan

No	Nama	Keterangan *Device*
1	Salahudin Agung Wijaya	Lenovo T900
2	Amanda Tiara Averousi	Moto E
3	Hawari Rahman	Asus Zenfone 5 T00F
4	Maranu Toto Negoro	Asus Zenfone 5 T00J
5	Hashfi Alfian Ciyuda	Lenovo A706
6	Sandhi Ading Wasana	Redmi 1S
7	M Iqbal Rustamadji	HTC One M7
8	Muhammad Chaqiqi Mudafi	Lenovo A526
9	Rifi Febrio	Sony XPERIA T2
10	Mahardhika Maulana	Samsung Galaxy Note 3 NEO
11	Punggi Esthi Bawono	Smartfren Andromax i2

Berikut hasil penilaian pengguna terhadap kelancaran permainan, penilaian grafis dan kesesuaian GUI pada Tabel 5.9.

Tabel 5.9 Hasil Pengujian Pengguna

No	Keterangan Pengujian	Penilaian					Rata-Rata
		1	**2**	**3**	**4**	**5**	
1	Kelancaran *FPS* Permainan	0	1	5	0	5	76 %
2	Kualitas Grafis Permainan	0	0	4	4	3	78 %
		Sesuai		Tidak			
3	Kesesuaian GUI dengan *Device*	11		0			100 %
Total Rata-Rata							84,6 %

5.3.2 Uji Performa Grafis 3D

Uji ini berupa perhitungan kecepatan *frame* per detik pada perangkat Android dalam me-*render* objek 3D. Sebuah permainan video akan terasa lancar pada standar *framerate* 24. Hasilnya ditampilkan pada Tabel 5.10.

Tabel 5.10 Uji Performa Grafis 3D

Nama Perangkat	*FPS* Minimum	*FPS* Maksimum	*FPS* Rata-Rata
Lenovo T900	30	30	30
Moto E	17	21	19
Zenfone 5 T00J/T00F	30	30	30
Smartfren Andromax i2	12	16	14
Lenovo A706	20	22	21
Redmi 1S	17	22	19,5
Lenovo A526	30	30	30
Samsung Galaxy Note 3 NEO	18	30	24
HTC One M7	18	24	21
Sony XPERIA T2	18	24	21
Total Rata-Rata			22,95
Persentase Terhadap Standar Kelancaran (24 *FPS*)			95,6%

5.3.3 Uji Kenyamanan dan Kehandalan

Uji ini berupa pengecekkan kompability GUI di layar *landscape orientation* pada perangkat yang berbeda-beda dan mengacu terhadap pengujian pengguna dengan *device* yang dimiliki. Hasilnya dapat dilihat pada Tabel 5.11.

Tabel 5.11 Uji Kenyamanan dan Kehandalan

Nama Perangkat	Kesesuaian GUI pada layar
Lenovo T900	Sesuai
Moto E	Sesuai
Zenfone 5 T00J/T00F	Sesuai
Smartfren Andromax i2	Sesuai
Lenovo A706	Sesuai

Redmi 1S	Sesuai
Lenovo A526	Sesuai
Samsung Galaxy Note 3 NEO	Sesuai
HTC One M7	Sesuai
Sony XPERIA T2	Sesuai

5.4 Evaluasi

Subbab ini membahas mengenai evaluasi terhadap pengujian-pengujian yang telah dilakukan. Dalam hal ini, evaluasi menunjukkan data rekapitulasi dari hasil pengujian fungsionalitas dan pengujian non-fungsionalitas yang telah dilakukan sebelumnya.

5.4.1 Evaluasi Pengujian Fungsionalitas

Evaluasi pengujian fungsionalitas dilakukan dengan menampilkan data rekapitulasi aplikasi permainan yang telah dipaparkan pada subbab 5.2. Dalam hal ini, rekapitulasi disusun dalam bentuk tabel yang dapat dilihat pada Tabel 5.12. Dari data yang terdapat pada tabel tersebut, diketahui bahwa aplikasi permainan yang dibuat telah memenuhi alur perancangan permainan yang telah ditentukan dan dijelaskan pada subbab 3.2.

Tabel 5.12 Rekapitulasi Hasi Uji Coba Fungsionalitas

ID Pengujian	Deskripsi	Hasil
PF1	Pengujian Fungsionalitas *Offline Mode Practice*	Berhasil
PF2	Pengujian Fungsionalitas *Synchonous Multiplayer*	Berhasil

5.4.2 Evaluasi Pengujian Non-Fungsionalitas

Evaluasi pengujian non-fungsionalitas dilakukan dengan menampilkan data rekapitulasi perangkat lunak yang telah dipaparkan pada subbab 5.3. Dalam hal ini, rekapitulasi disusun dalam bentuk tabel yang dapat dilihat pada Tabel 5.13. Dari data diketahui bahwa aplikasi permainan telah memenuhi unsur performa grafis 3D, kenyamanan dan kehandalan untuk meningkatkan kualitas aplikasi permainan.

Tabel 5.13 Rekapitulasi Hasil Uji Coba Non-Fungsionalitas

ID Pengujian	Deskripsi	Hasil
PNF1	Pengujian Non- Fungsionalitas Performa Grafis 3D	Memenuhi
PNF2	Pengujian Non-Fungsionalitas Kenyamanan dan Kehandalan	Memenuhi

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, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dalam proses pengerjaan tugas akhir mulai dari tahap analisis, desain, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. Aplikasi permainan berhasil menjalankan kebutuhan fungsional skenario aturan main yang terintegrasi dengan mode *synchronous* multiplayer maupun *offline*.
2. Integrasi aturan main beserta *gameplay* berhasil berjalan dengan baik dan sesuai dengan skenario pengujian.
3. Berdasarkan pengujian fitur *matchmaking*, aplikasi permainan berhasil dalam mencari dan mempertemukan antar pemain.
4. Berdasarkan pengujian fitur sinkronisasi pemain saat berada pada *state* permainan sedang berlangsung, telah berhasil diimplementasikan dan berjalan dengan baik.
5. Berdasarkan uji performa untuk permainan grafis 3D, setiap perangkat yang memiliki jenis CPU dan GPU akan teroptimisasi jika *terrain* permainan yang dibangun tidak mengaktifkan *rendering* seperti proyeksi bayangan dan *ambient occlusion*. Performa akan meningkat seiring dengan kualitas grafis dari aset. Agar kualitas grafis tampilan *terrain* permainan terjaga, maka harus di *bake* seperti tekstur dan mapping. Penerapan tersebut berhasil dilakukan demi meningkatkan performa dan kualitas.

6. Berdasarkan uji kehandalan dan kenyamanan, UGUI (Unity GUI), dapat menyesuaikan dan mendukung setiap jenis layar perangkat mobile.

6.2. Saran

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

1. Pengembangan kedepannya sebaiknya dapat terintegrasi dengan fitur asynchronous.
2. Pengembangan kedepannya aplikasi permainan dapat dimankan banyak pengguna dan dijadikan produk komersial.

DAFTAR PUSTAKA

[1] Unity, "Unity Documentation," 2014. [Online]. Available: http://docs.unity3d.com/Manual/. [Accessed 25 September 2014].

[2] Richard A. Bartle, *Player Versus Player*, *Designing Virtual Words*, Paperback. Ripon, England: New Riders, 2004.

[3] Dotagamepedia, "Windrunner Character," 2014. [Online]. Available: http://dota2.gamepedia.com/Windranger. [Accessed 23 September 2014].

[4] Gameloft, "Dungeon Hunter 4," 2013. [Online]. Available: http://www.gameloft.com/iphone-games/dungeon-hunter-4-free/. [Accessed 20 September 2014].

[5] Photon, "Photon Unity Networking," 2014. [Online]. Available: https://www.exitgames.com/en/PUN. [Accessed 21 September 2014].

[6] Gamasutra, "Designing Multiplayer Games," 2014. [Online] Available: http://gamasutra.com/blogs/DanielCook/20140104/208021/What_Ive_learned_about_designing_multiplayer_games_so_far.php. [Accessed 11 Mei 2015].

LAMPIRAN A
LAMPIRAN PENGUJIAN SINKRONISASI PERGERAKKAN

Tabel 6.1 Pengujian Sinkronisasi Pergerakkan Iterasi 1

<table>
<tr><td colspan="2">ITERASI 1</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Posisi pergerakkan karakter ketika klien A melakukan pergerakkan menggunakan virtual analog yang dimana posisi pergerakkan karakternya pada klien B juga ikut berubah. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 76 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>76 ms</td></tr>
</table>

Hasil Iterasi Pengujian	Berhasil

Tabel 6.2 Pengujian Sinkronisasi Pergerakkan Iterasi 2

ITERASI 2	
Klien A	Klien B
	
Keadaan awal pada Klien A dan B	
	
Posisi pergerakkan karakter ketika klien A melakukan pergerakkan menggunakan virtual *analog* yang dimana posisi pergerakkan karakternya pada klien B juga ikut berubah. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 91 milisekon dan berhasil tersinkronisasi	
Ping Delay	91 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.3 Pengujian Sinkronisasi Pergerakkan Iterasi 3

<table>
<tr><td colspan="2">ITERASI 3</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Posisi pergerakkan karakter ketika klien B melakukan pergerakkan menggunakan virtual analog yang dimana posisi pergerakkan karakternya pada klien A juga ikut berubah. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 88 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>88 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.4 Pengujian Sinkronisasi Pergerakkan Iterasi 4

<table>
<tr><td colspan="2">ITERASI 4</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Posisi pergerakkan karakter ketika klien A melakukan pergerakkan menggunakan virtual analog yang dimana posisi pergerakkan karakternya pada klien B juga ikut berubah. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 48 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>48 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.5 Pengujian Sinkronisasi Pergerakkan Iterasi 5

<table>
<tr><td colspan="2">ITERASI 5</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Posisi pergerakkan karakter ketika klien A melakukan pergerakkan menggunakan virtual analog yang dimana posisi pergerakkan karakternya pada klien B juga ikut berubah. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 51 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>51 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.6 Pengujian Sinkronisasi Pergerakkan Iterasi 6

ITERASI 6	
Klien A	Klien B
	
Keadaan awal pada klien A dan B	
	
Posisi pergerakkan karakter ketika klien B melakukan pergerakkan menggunakan virtual *analog* yang dimana posisi pergerakkan karakternya pada klien A juga ikut berubah. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 72 milisekon dan berhasil tersinkronisasi	
Ping Delay	72 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.7 Pengujian Sinkronisasi Pergerakkan Iterasi 7

<table>
<tr><td colspan="2">ITERASI 7</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Posisi pergerakkan karakter ketika klien A melakukan pergerakkan menggunakan virtual analog yang dimana posisi pergerakkan karakternya pada klien B juga ikut berubah. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 99 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>99 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.8 Pengujian Sinkronisasi Pergerakkan Iterasi 8

ITERASI 8	
Klien A	Klien B
	
Keadaan awal pada klien A dan B	
	
Posisi pergerakkan karakter ketika klien B melakukan pergerakkan menggunakan virtual *analog* yang dimana posisi pergerakkan karakternya pada klien A juga ikut berubah. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 63 milisekon dan berhasil tersinkronisasi	
Ping Delay	63 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.9 Pengujian Sinkronisasi Pergerakkan Iterasi 9

<table>
<tr><td colspan="2">ITERASI 9</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Posisi pergerakkan karakter ketika klien A melakukan pergerakkan menggunakan virtual analog yang dimana posisi pergerakkan karakternya pada klien B juga ikut berubah. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 73 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>73 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.10 Pengujian Sinkronisasi Pergerakkan Iterasi 10

ITERASI 10	
Klien A	Klien B
	
Keadaan awal pada klien A dan B	
	
Posisi pergerakkan karakter ketika klien B melakukan pergerakkan menggunakan virtual *analog* yang dimana posisi pergerakkan karakternya pada klien A juga ikut berubah. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 94 milisekon dan berhasil tersinkronisasi	
Ping Delay	94 ms
Hasil Iterasi Pengujian	Berhasil

LAMPIRAN B
LAMPIRAN PENGUJIAN SINKRONISASI ANAK PANAH (*PROJECTILE*)

Tabel 6.11 Pengujian Sinkronisasi Anak Panah Iterasi 1

<table>
<tr><th colspan="2">ITERASI 1</th></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien A menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Instansiasi dan perubahan posisi anak panah (projectile) ketika klien A melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien B juga ter-instansiasi anak panah (projectile) beserta posisinya. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 84 milisekon dan berhasil tersinkronisasi</td></tr>
</table>

Ping Delay	84 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.12 Pengujian Sinkronisasi Anak Panah Iterasi 2

<table>
<tr><td colspan="2">ITERASI 2</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien B menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Instansiasi dan perubahan posisi anak panah (projectile) ketika klien B melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien A juga ter-instansiasi anak panah (projectile) beserta posisinya. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 109 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>109 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.13 Pengujian Sinkronisasi Anak Panah Iterasi 3

<table>
<tr><td colspan="2">ITERASI 3</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien A menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Instansiasi dan perubahan posisi anak panah (projectile) ketika klien A melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien B juga ter-instansiasi anak panah (projectile) beserta posisinya. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 98 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>98 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.14 Pengujian Sinkronisasi Anak Panah Iterasi 4

ITERASI 4	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B menahan tombol virtual untuk membidik	
	
Instansiasi dan perubahan posisi anak panah (*projectile*) ketika klien B melakukan pelepasan anak panah (*projectile*) dengan melepaskan tombol virtual yang dimana pada klien A juga ter-instansiasi anak panah (*projectile*) beserta posisinya. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 78 milisekon dan berhasil tersinkronisasi	
Ping Delay	78 ms
Hasil Iterasi Pengujian	Berhasil

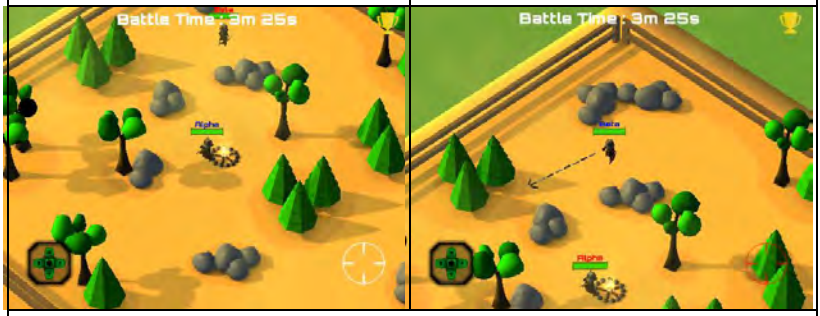
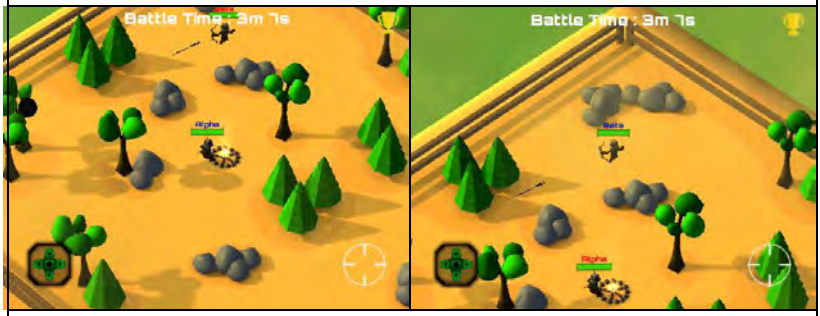
Tabel 6.15 Pengujian Sinkronisasi Anak Panah Iterasi 5

<table>
<tr><td colspan="2">ITERASI 5</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien A menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Instansiasi dan perubahan posisi anak panah (projectile) ketika klien A melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien B juga ter-instansiasi anak panah (projectile) beserta posisinya. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 66 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>66 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.16 Pengujian Sinkronisasi Anak Panah Iterasi 6

ITERASI 6	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B menahan tombol virtual untuk membidik	
	
Instansiasi dan perubahan posisi anak panah (*projectile*) ketika klien B melakukan pelepasan anak panah (*projectile*) dengan melepaskan tombol virtual yang dimana pada klien A juga ter-instansiasi anak panah (*projectile*) beserta posisinya. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 71 milisekon dan berhasil tersinkronisasi	
Ping Delay	71 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.17 Pengujian Sinkronisasi Anak Panah Iterasi 7

<table>
<tr><td colspan="2">ITERASI 7</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien B menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2">  </td></tr>
<tr><td colspan="2">Instansiasi dan perubahan posisi anak panah (projectile) ketika klien B melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien A juga ter-instansiasi anak panah (projectile) beserta posisinya. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 90 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>90 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.18 Pengujian Sinkronisasi Anak Panah Iterasi 8

<table>
<tr><td colspan="2">ITERASI 8</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien A menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Klien A melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien B gagal melakukan instansiasi secara remote dikarenakan pada iterasi ini, delay pertukaran data sangat besar dengan server untuk broadcast ke setiap kliennya, yaitu sebesar 2153 milisekon. Sehingga memungkinan terjadi kegagalan untuk melakukan remote dan hasilnya gagal tersinkronisasi. Dan solusi untuk mengatasi kegagalan tersebut adalah melakukan stream ulang ke server ketika ping delay sudah normal.</td></tr>
<tr><td>Ping Delay</td><td>2153 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Gagal</td></tr>
</table>

Tabel 6.19 Pengujian Sinkronisasi Anak Panah Iterasi 9

<table>
<tr><td colspan="2">ITERASI 9</td></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien A menahan tombol virtual untuk membidik</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Instansiasi dan perubahan posisi anak panah (projectile) ketika klien A melakukan pelepasan anak panah (projectile) dengan melepaskan tombol virtual yang dimana pada klien B juga ter-instansiasi anak panah (projectile) beserta posisinya. Pada iterasi ini, delay pertukaran data dengan server untuk broadcast ke setiap kliennya sebesar 80 milisekon dan berhasil tersinkronisasi</td></tr>
<tr><td>Ping Delay</td><td>80 ms</td></tr>
<tr><td>Hasil Iterasi Pengujian</td><td>Berhasil</td></tr>
</table>

Tabel 6.20 Pengujian Sinkronisasi Anak Panah Iterasi 10

ITERASI 10	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B menahan tombol virtual untuk membidik	
	
Instansiasi dan perubahan posisi anak panah (*projectile*) ketika klien B melakukan pelepasan anak panah (*projectile*) dengan melepaskan tombol virtual yang dimana pada klien A juga ter-instansiasi anak panah (*projectile*) beserta posisinya. Pada iterasi ini, *delay* pertukaran data dengan *server* untuk *broadcast* ke setiap kliennya sebesar 74 milisekon dan berhasil tersinkronisasi	
Ping Delay	74 ms
Hasil Iterasi Pengujian	Berhasil

LAMPIRAN C
LAMPIRAN PENGUJIAN SINKRONISASI *HEALTHBAR*

Tabel 6.21 Pengujian Sinkronisasi *Healthbar* Iterasi 1

<table>
<tr><th colspan="2">ITERASI 1</th></tr>
<tr><td>Klien A</td><td>Klien B</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Keadaan awal pada klien A dan B, klien A melakukan pelepasan anak panah (projectile)</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td colspan="2">Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai health. Anak panah (projectile) bertumbukkan dengan karakter pemain klien B, sehingga karakter pemain B, nilai health akan dikurangi dengan damage dan nilai tersebut akan dilakuan stream ke server dan di broadcast ke setiap klien. Pada iterasi ini, delay pertukaran data dengan server</td></tr>
</table>

untuk *broadcast* ke setiap kliennya sebesar 75 milisekon dan berhasil tersinkronisasi	
Ping Delay	75 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.22 Pengujian Sinkronisasi *Healthbar* Iterasi 2

ITERASI 2	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien A, sehingga karakter pemain A, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

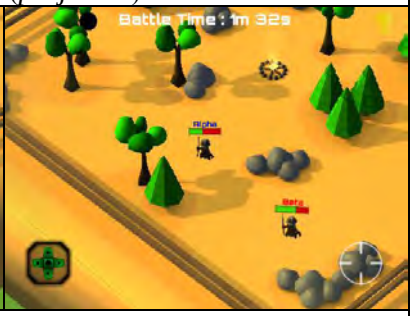
untuk *broadcast* ke setiap kliennya sebesar 123 milisekon dan berhasil tersinkronisasi	
Ping Delay	123 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.23 Pengujian Sinkronisasi *Healthbar* Iterasi 3

ITERASI 3	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien A, sehingga karakter pemain A, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 201 milisekon dan berhasil tersinkronisasi	
Ping Delay	201 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.24 Pengujian Sinkronisasi *Healthbar* Iterasi 4

ITERASI 4	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien A melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien B, sehingga karakter pemain B, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

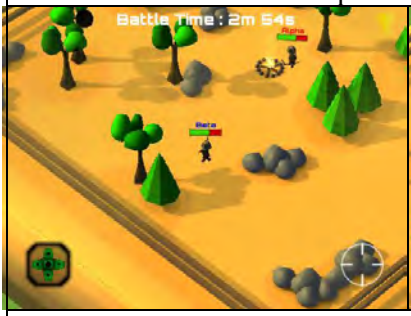
untuk *broadcast* ke setiap kliennya sebesar 92 milisekon dan berhasil tersinkronisasi	
Ping Delay	92 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.25 Pengujian Sinkronisasi *Healthbar* Iterasi 5

ITERASI 5	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien A melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien B, sehingga karakter pemain B, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 62 milisekon dan berhasil tersinkronisasi	
Ping Delay	62 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.26 Pengujian Sinkronisasi *Healthbar* Iterasi 6

ITERASI 6	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien A, sehingga karakter pemain A nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 59 milisekon dan berhasil tersinkronisasi	
Ping Delay	59 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.27 Pengujian Sinkronisasi *Healthbar* Iterasi 7

ITERASI 7	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien A melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien B, sehingga karakter pemain B, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 61 milisekon dan berhasil tersinkronisasi	
Ping Delay	61 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.28 Pengujian Sinkronisasi *Healthbar* Iterasi 8

ITERASI 8	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien A, sehingga karakter pemain A, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 77 milisekon dan berhasil tersinkronisasi	
Ping Delay	77 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.29 Pengujian Sinkronisasi *Healthbar* Iterasi 9

ITERASI 9	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien B melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien A, sehingga karakter pemain A, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 102 milisekon dan berhasil tersinkronisasi	
Ping Delay	102 ms
Hasil Iterasi Pengujian	Berhasil

Tabel 6.30 Pengujian Sinkronisasi *Healthbar* Iterasi 10

ITERASI 10	
Klien A	Klien B
	
Keadaan awal pada klien A dan B, klien A melakukan pelepasan anak panah (*projectile*)	
	
Ketika anak panah betumbukkan dengan karakter pemain, maka akan terjadi pengurangan nilai *health*. Anak panah (*projectile*) bertumbukkan dengan karakter pemain klien B, sehingga karakter pemain B, nilai *health* akan dikurangi dengan *damage* dan nilai tersebut akan dilakuan *stream* ke *server* dan di *broadcast* ke setiap klien. Pada iterasi ini, *delay* pertukaran data dengan *server*	

untuk *broadcast* ke setiap kliennya sebesar 83 milisekon dan berhasil tersinkronisasi	
Ping Delay	83 ms
Hasil Iterasi Pengujian	Berhasil

BIODATA PENULIS

Penulis dilahirkan di Bogor, 1 September 1993, merupakan anak kelima dari 7 bersaudara. Penulis telah menempuh pendidikan formal yaitu TK Gembira Jatibening Jakarta Timur (1997-1999), SD Negeri 03 Cipinang Melayu Jakarta Timur (1999-2005), SMP Negeri 62 Jakarta (2005-2006), SMP Negeri 13 Bandung (2006-2008), SMA Negeri 25 Bandung (2008-2011), dan mahasiswa S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya rumpun mata kuliah Interaksi, Grafika, dan Seni (2011-2015).

Selama menjadi mahasiswa penulis pernah berperan sebagai asisten Basis Data (2013-2014) dan Realitas Virtual (2014-2015). Penulis juga aktif mengembangkan *game* di industri *game* dan mengikuti lomba-lomba di antaranya adalah SNITCH 2013 bidang Lomba Pengembangan Aplikasi (Juara 3) dan dulunya aktif membuat aplikasi dan *games* untuk Developer Level Up Microsoft. Penulis yang memiliki hobi bermain *game* dan olahraga ini merupakan mahasiswa yang aktif dalam organisasi Himpunan Mahasiswa Teknik Computer – Informatika ITS (HTMC). Penulis dapat dihubungi via email askary10993@gmail.com.