TUGAS AKHIR – KI141502

Game Puzzle Multiplayer Berbasis Oculus Rift *Virtual Reality* dan Kinect (Implementasi Oculus Rift dan Kinect)

SADDHANA ARTA DANISWARA
NRP 5114100191

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

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

TUGAS AKHIR – KI141502

Game Puzzle Multiplayer Berbasis Oculus Rift *Virtual Reality* dan Kinect (Implementasi Oculus Rift dan Kinect)

SADDHANA ARTA DANISWARA
NRP 5114100191

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

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

MULTIPLAYER PUZZLE GAME BASED ON OCULUS RIFT VIRTUAL REALITY AND KINECT (Implementation of Oculus Rift and Kinect)

SADDHANA ARTA DANISWARA
NRP 5114100191

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

INFORMATICS DEPARTEMENT
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

***Game Puzzle Multiplayer* Berbasis Oculus Rift *Virtual Reality* dan Kinect (Implementasi Oculus Rift dan Kinect)**

TUGAS AKHIR

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

Oleh :
SADDHANA ARTA DANISWARA
NRP : 5114100191

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr.Eng Darlis Herumurti, S.Kom, M.Kom ..... (pembimbing 1)
NIP: 197712172003121001

Imam Kuswardayan, S.Kom, M.T ..... (pembimbing 2)
NIP: 197612152003121001

SURABAYA
JUNI 2018

[Halaman ini sengaja dikosongkan]

***Game Puzzle Multiplayer* Berbasis Oculus Rift *Virtual Reality* dan Kinect (Implementasi Oculus Rift dan Kinect)**

Nama Mahasiswa	: SADDHANA ARTA DANISWARA
NRP	: 5114100191
Departemen	: Informatika FTIK-ITS
Dosen Pembimbing I	: Dr.Eng Darlis Herumurti, S.Kom, M.Kom.
Dosen Pembimbing II	: Imam Kuswardayan, S.Kom, M.T.

Abstrak

Kemajuan teknologi membuat banyak perubahan terutama dalam perkembangan aplikasi permainan, salah satunya adalah pengaplikasian *virtual reality* dalam permainan. *Virtual reality* (VR) adalah teknologi yang memungkinkan penggunanya untuk masuk kedalam dunia virtual dan berinteraksi dengan lingkungan sekitar. Pengaplikasian VR selain dalam permainan juga digunakan untuk medis, militer, dan edukasi.

Kinect merupakan teknologi yang memungkinkan komputer untuk mendeteksi seluruh tubuh *user* untuk digunakan sebagai *input*. Kinect dapat dimanfaatkan dalam berbagai bidang seperti *game*, pendidikan, kesehatan, dan *fashion*.

Puzzle merupakan salah satu genre permainan yang sangat populer untuk edukasi, karena permainan dengan genre *puzzle* mengharuskan pemainnya untuk berpikir jika ingin menyelesaikan permainan.

Penulis memilih untuk menggabungkan teknologi VR dengan Kinect karena penulis menyadari kekurangan dari masing masing perangkat. Penulis berharap dengan menggabungkan kedua teknologi, dapat melengkapi kekurangan dari kedua perangkat. Permainan yang dibuat mengambil genre *puzzle*, karena genre ini memungkinkan untuk digunakannya perangkat Kinect dan Oculus Rift secara bersamaan. Permainan *puzzle* ini berjudul 'Unknown'. Permainan ini dimainkan oleh dua orang pemain. Salah satu pemain menggunakan Oculus Rift dan yang yang satunya menggunakan Kinect. Pemain saling berkoordinasi untuk menyelesaikan teka teki yang diberikan.

Uji coba dilakukan dengan menjalankan semua kasus penggunaan yang telah dirancang sebelumnya. Setiap fungsi harus bisa dijalankan tanpa adanya kendala. Uji coba Oculus Rift dan Kinect dilakukan secara bersamaan.

Dari hasil uji coba dapat disimpulkan bahwa permainan Unknown telah berjalan sesuai dengan yang telah dirancang. Namun deteksi dari Kinect masih dapat disempurnakan.

Kata kunci: Oculus Rift, Kinect, dungeon, puzzle

MULTIPLAYER PUZZLE GAME BASED ON OCULUS RIFT VIRTUAL REALITY AND KINECT
(IMPLEMENTATION OF OCULUS RIFT AND KINECT)

Name	: SADDHANA ARTA DANISWARA
NRP	: 5114100191
Major	: Informatics– FTIK ITS
Supervisor I	: Dr.Eng Darlis Herumurti, S.Kom, M.Kom.
Supervisor II	: Imam Kuswardayan, S.Kom, M.T.

Abstract

Technological advances make the solution in the application process, one of which is the application of virtual reality in the game. Virtual reality (VR) is a technology that allows users to enter the virtual world and use the environment. VR application in addition to in-game also for medical, military, and education.

Kinect is a technology that allows computers to detect the entire body of the user to serve as input. Kinect can be used in various fields such as games, education, health, and fashion.

Puzzles is one of the most popular game genres for education, because games with the puzzle genre require players to think if they are happy to play.

The author chose to use VR technology with Kinect because the authors found each device. Software with good technology, can be broadcast from the second device. The games that are made take the puzzle genre, as this genre allows for Android devices Kinect and Oculus Rift simultaneously. The game puzzle entitled 'Unknown'. This game is played by two players. One player uses Oculus Rift and the one who uses Kinect. Players coordinate each other to cultivate a given puzzle.

Trials are performed using all pre-designed use cases. Each function must be able to run in the absence of inventory. Oculus Rift and Kinect trials are conducted simultaneously.

From the test results it can be concluded that the unknown game has been running in accordance with the one that has been designed. But detection from Kinect can still be refined.

Keywords: Oculus Rift, Kinect, dungeon, puzzle

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيم

Alhamdulillahirabbil'alamin, segala puji penulis panjatkan kepada Allah SWT, yang selalu memberikan kemudahan dibalik kesulitan pada setiap hambanya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul

"*GAME PUZZLE MULTIPLAYER* BERBASIS OCULUS RIFT *VIRTUAL REALITY* DAN KINECT (IMPLEMENTASI OCULUS RIFT DAN KINECT)".

Pengerjaan Tugas Akhir in telah menjadi salah satu pengalaman yang paling berharga sepanjang hayat penulis. Pada pengerjaan Tugas Akhir ini penulis dapat mengimplementasikan berbagai macam ilmu yang telah didapatkan penulis selama menempuh perkuliahan di Informatika ITS.

Tugas Akhir ini dapat selesai karena bantuan dan dukungan dari berbagai pihak. Pada kesempatan ini penulis mengucapkan rasa syukur dan terima kasih kepada:

1. Allah SWT, karena atas izin-Nya dan kemudahan yang diberikannya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Adik serta Saudara-saudara yang selalu mendoakan dan mendukung penulis.
3. Dr.Eng Darlis Herumurti, S.Kom, M.Kom. selaku pembimbing I yang selalu memberikan arahan, motivasi dan bantuan sekaligus bimbingan kepada penulis selama pengerjaan Tugas Akhir.
4. Pak Imam Kuswardayan, S.Kom, M.T.selaku pembimbing II yang juga telah sangat membantu, dan membimbing saat pengerjaan Tugas Akhir ini.

5. Romi Yehezkiel Purba yang tergabung bersama penulis dalam tim Tugas Akhir , yang telah membantu penulis selama pengerjaan Tugas Akhir.
6. Rekan-rekan Soritia Rizal, Romi, Evan, Rian, Adiwinoto, Dwiyandono, Anggit, Ucup, Agun, Andre, Fito, Angga, Rochman, OjanGG, Bagas, Iqbal, Digoz yang sangat unik telah menemani penulis baik susah maupun senang selama perkuliahan di Informatika ITS dan terus memberikan semangat pada penulis untuk menyelesaikan Tugas Akhir ini.
7. Bapak dan Ibu Dosen Karyawan Teknik Informatika FTIf ITS yang telah memberikan ilmunya.
8. Teman-teman TC khususnya Satria, Rijal, dan Panji yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, kakak-kakak angkatan 2013, 2012, serta adik-adik angkatan 2015 dan 2016 yang membuat penulis untuk selalu belajar.
9. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis memohon maaf apabila terdapat kekurangan dan kesalahan dalam penulisan Tugas Akhir ini. Kritik dan saran penulis harapkan untuk perbaikan kedepannya. Semoga Tugas Akhir ini dapat memberikan Manfaat yang sebesar besarnya.

Surabaya, Juni 2018

Saddhana Arta Daniswara

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR LAMPIRAN

[Halaman sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Diharapkan dari penjelasan dalam bab ini gambaran tugas akhir secara umum dapat dipahami.

1.1. Latar Belakang

Aplikasi permainan menjadi salah satu hiburan untuk berbagai macam kalangan. Perkembangannya juga sangat cepat dan hal tersebut membuat aplikasi terus berkembang mengikuti zaman. Berbagai perkembangan teknologi juga diterapkan di aplikasi permainan dengan tujuan untuk menambah pengalaman bermain pemain. Salah satunya adalah *virtual reality* dan Kinect [1].

Virtual reality (VR) merupakan teknologi yang memungkinkan *user* melakukan simulasi terhadap objek nyata yang mampu membangkitkan suasana tiga dimensi (3D). VR ini dapat dimanfaatkan dalam berbagai bidang seperti *game*, arsitek, pendidikan, kesehatan, simulasi berkendara, bahkan digunakan dalam bidang militer. Namun teknologi VR memiliki kekurangan yaitu tidak dapat mendeteksi pergerakan kaki dari pemain. Pengguna VR juga tidak dapat berinteraksi langsung dengan dunia nyata karena pengguna VR menggunakan perangkat yang menutup bagian mata pengguna VR.

Kinect merupakan teknologi yang memungkinkan komputer untuk mendeteksi seluruh tubuh *user* untuk digunakan sebagai *input*. Kinect dapat dimanfaatkan dalam berbagai bidang seperti *game*, pendidikan, kesehatan, dan *fashion*. Kinect mempunyai kekurangan dalam hal interaksi dengan objek di dalam game. Interaksi pemain dalam Kinect sangat terbatas. Pengguna Kinect tidak bisa memegang dan membawa objek yang

berada di dalam permainan. Namun pengguna Kinect mempunyai kelebihan yaitu dapat berinteraksi dengan dunia diluar permainan.

Berdasarkan kekurangan dari masing masing perangkat, penulis memutuskan untuk menggunakan kedua alat bersamaan untuk saling melengkapi kekurangan dari alat lain. Oleh karena itu Tugas Akhir yang akan dibuat adalah *game* yang menggabungkan VR Oculus Rift dengan Kinect, dengan harapan pemain mendapatkan sensasi yang berbeda dari game pada umumnya karena pemain diharuskan untuk saling bekerja sama dan berkomunikasi. Tujuan dari pengerjaan Tugas Akhir ini adalah menghasilkan aplikasi permainan yang menggabungkan teknologi Kinect dengan Oculus Rift.

1.2. Rumusan Masalah

Berdasarkan latar belakang, terdapat beberapa rumusan masalah yang terdapat pada tugas akhir ini, antara lain adalah sebagai berikut:

1. Bagaimana cara untuk mengimplementasi Kinect pada permainan?
2. Bagaimana cara untuk mengimplementasi Oculus Rift pada permainan?
3. Bagaimana cara menggabungkan Kinect dan Oculus Rift pada permainan?

1.3. Batasan Masalah

Batasan masalah yang terdapat pada tugas akhir ini adalah sebagai berikut:

1. Permainan hanya bisa dimainkan oleh 2 pemain.
2. Aplikasi yang dibangun hanya berjalan pada perangkat computer dengan Operation System Windows.
3. Lingkungan pengembangan yang digunakan menggunakan aplikasi Unity 3D Free License dengan bahasa pemrograman C# dengan menggunakan Oculus Utilities for Unity, dan Kinect SDK.

1.4. Tujuan

Tujuan pembuatan tugas akhir ini adalah:

1. Membuat aplikasi permainan berbasis *virtual reality* yang diintegrasikan dengan Kinect.
2. Membuat aplikasi permainan yang dapat dimainkan dengan *multiplayer*.

1.5. Manfaat

Manfaat dari penelitian tugas akhir ini adalah menciptakan aplikasi permainan khusunya *puzzle* yang menggabungkan teknologi VR Oculus Rift dengan Kinect. Penggabungan kedua perangkat ini diharapkan dapat mengatasi kekurangan dari masing masing perangkat, dan dapat memberikan solusi terhadap penerapan Oculus Rift dan Kinect pada aplikasi lain yang lebih kompleks.

1.6. Metodologi Pembuatan Tugas Akhir

Pembuatan tugas akhir dilakukan dengan tahapan-tahapan sebagai berikut:

1. Penyusunan proposal

   Langkah awal dalam mengerjakan tugas akhir adalah dengan menyusun proposal tugas akhir. Pada proposal penulis melakukan studi mengenai cara kerja Oculus Rift dan Kinect.
2. Studi literatur

   Pada tahap ini, akan dicari studi literatur yang relevan untuk dijadikan referensi dalam pengerjaan tugas akhir. Literatur-literatur yang dimaksud disebutkan sebagai berikut:
   1. Unity
   2. Bahasa pemrograman C#
   3. Oculus Rift
   4. Kinect
3. Perancangan permainan

   Pada tahap ini akan dilakukan perancangan kasus penggunaan Oculus Rift dan Kinect.

4. Implementasi dan pembuatan sistem
   Pada tahap ini dilakukan implementasi dari Oculus Rift dan Kinect. Aplikasi ini diimplementasikan dengan menggunakan Game Engine Unity Personal Edition, dengan bahasa pemrograman C#.

5. Pengujian dan Evaluasi
   Pada tahap ini dilakukan pengujian dari setiap kasus penggunaan yang sudah didesain sebelumnya. Pengujian terhadap integrasi Oculus Rift dan Kinect dilakukan dengan memastikan apakah seluruh fungsi sudah dapat berfungsi tanpa ada kendala.
6. Penyusunan laporan tugas akhir
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini.

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini terdiri atas beberapa bab yang tersusun secara sistematis, yaitu sebagai berikut.

1. BAB I, Pendahuluan, bab ini berisi penjelasan mengenai latar belakang masalah, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu rumusan permasalahan, batasan masalah, dan sistematika penulisan juga merupakan bagian dari bab ini.
2. BAB II, Tinjauan Pustaka, Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan yang menjadi dasar dari pembuatan Tugas Akhir ini.
3. BAB III, Analisa dan Perancangan Sistem, Bab ini membahas tentang analisis permasalahan, deskripsi umum sistem, dan perancangan aturan main.
4. BAB IV, Implementasi Sistem, Bab ini berisi implementasi dari perancangan sistem dan implementasi fitur-fitur penunjang aplikasi termasuk implementasi pembuatan

aplikasi permain dengan menerapkan integrasi Kinect dan Oculus Rift.

5. BAB V, Pengujian dan Evaluasi, Bab ini membahas pengujian dengan metode kotak hitam (*black box testing*) untuk mengetahui aspek nilai fungsionalitas dari perangkat lunak dan nilai kegunaan yang dibuat.
6. BAB VI, Kesimpulan dan Saran, Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan pada Tugas Akhir ini. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Pada Bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir ini. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap perangkat lunak yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1. Rancang Bangun Perangkat Lunak

Rancang bangun perangkat lunak adalah suatu ilmu yang mempelajari proses pembuatan aplikasi yang melingkupi analisis permasalahan dan kebutuhan, perencanaan, analisis sistem, implementasi, serta pemeliharaan perangkat lunak termasuk perbaikan jika ditemukannya *bug* [2].

2.2. Human Computer Interaction

Human Computer Interaction, atau dalam bahasa Indonesia yaitu "Interaksi Manusia dan Komputer" merupakan ilmu yang mempelajari tentang interaksi antara manusia dan komputer. Merancang bagaimana sistem komputer agar efektif, efisien, mudah dan menyenangkan agar masyarakat dapat menyadari manfaat perangkat komputasional [3].

2.3. Unity (Game Engine)

Unity merupakan sebuah *game engine* yang dikembangkan oleh Unity Technologies. Unity dapat menciptakan *game* ke dalam beberapa sistem operasi sekaligus. Antara lain: Windows Phone, Android, iOS, Windows 8, OSX, Tizen OS, Blackberry 10, Playstation 3, Playstation 4, XBOX, Oculus Rift dan sebagainya. *Game* yang dapat dibuat dengan Unity ini bisa dalam bentuk 3D atau 2D [4].

2.4. Bahasa Pemrograman C#

C# (dibaca: c sharp) merupakan sebuah bahasa pemrograman modern yang bersifat *general-purpose*, berorientasi objek yang dapat digunakan untuk membuat program di atas arsitektur Microsoft .Net framework. Bahasa pemrograman ini memiliki kemiripan dengan bahasa Java, C dan C++.

2.5. Puzzle

Puzzle adalah salah satu genre permainan yang memiliki beberapa turunan contohnya *maze* [5]. *Game* dengan genre *puzzle* sendiri memiliki banyak model permainan yang terus berkembang mengikuti zaman. *Game puzzle* memiliki penyelesaian yang tak pasti dan tentunya menggunakan sejumlah waktu untuk menyelesaikannya [6].

2.6. *Game* Yang Mirip

Terdapat *game* yang mirip dengan *game* yang kami buat. *Game* tersebut berjudul We Were Here. Logo We Were Here dapat dilihat pada Gambar 2.1. We Were Here memiliki kemiripan dalam segi tema dan genre.

Gambar 2.1 Logo We Were Here

Sumber:Google Images

2.7. Kinect

Kinect for Xbox atau biasanya Kinect (Dulunya di ketahui dengan sebutan Project Natal), adalah "*controller-free*

gaming dan pengalaman hiburan" oleh Microsoft dan Xbox *video game platform*, dan dapat digunakan untuk Windows 8. Kinect bersaing dengan Wii Remote dengan Wii MotionPlus dan PlayStation Move dengan PlayStation Eye. Kinect memiliki kekurangan yaitu sangat sedikit interaksi yang dapat dilakukan pemain terhadap objek yang ada di dalam permainan. Pengguna Kinect tidak dapat memegang dan membawa objek yang berada di dalam permainan [7].

2.8. Oculus Rift

Oculus Rift adalah output display seperti juga halnya layar komputer yang menampilkan dunia virtual 3 dimensi, hanya saja alat tersebut diletakkan sangat dekat dengan mata (seperti di gambar) disebut juga sebagai VR HMD (Virtual Reality – Head Mounted Display). Dengan desain optik yang tepat dan berkualitas mata kita bisa fokus pada display yang sangat dekat tersebut. Oculus Rift memiliki kekurangan yaitu tidak dapat mendeteksi pergerakan kaki pemain dan pemain tidak dapat berinteraksi dengan objek di luar permainan [8].

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai metodologi pemecahan masalah yang digunakan sebagai dasar solusi dari pembuatan Tugas Akhir.

3.1. Analisis

3.1.1. Deskripsi Game Unknown

Unknown yang dalam artian bahasa indonesia adalah tidak diketahui adalah sebuah game bergenre *puzzle* dengan tema eksplorasi dan misteri. *Game* ini dimainkan oleh dua orang pemain. Pada *game* ini pemain dihadapkan pada sebuah *puzzle* yang berbentuk *dungeon* besar yang terdiri dari 6 ruangan yang berbeda ukuran. Dalam game ini, kedua pemain harus bekerja sama dalam menyelesaikan tantangan pada setiap ruangan. Kedua pemain akan mendapatkan peran masing-masing akan tetapi tetap saling berkaitan dengan satu sama lain. Pemain pertama akan menggunakan perangkat Oculus Rift dan pemain kedua akan menggunakan perangkat Kinect.

3.1.2. Analisis *Gameplay*

Unknown memiliki aturan yang cukup sederhana. *Puzzle* dalam permainan ini dapat diselesaikan oleh dua pemain dengan metode berbeda, yaitu dengan cara melihat langsung di dalam dunia virtual melalui perangkat Oculus Rift oleh pemain pertama dan pemain kedua yang akan mengikuti arahan dari pemain pertama dengan menggunakan perangkat Kinect. Pemain akan diberikan berbagai macam petunjuk di dalam dunia virtual dan pada dunia nyata dalam bentuk teka teki. Kedua pemain harus dapat memecahkan teka teki agar bisa melanjutakn ke tahap berikutnya.

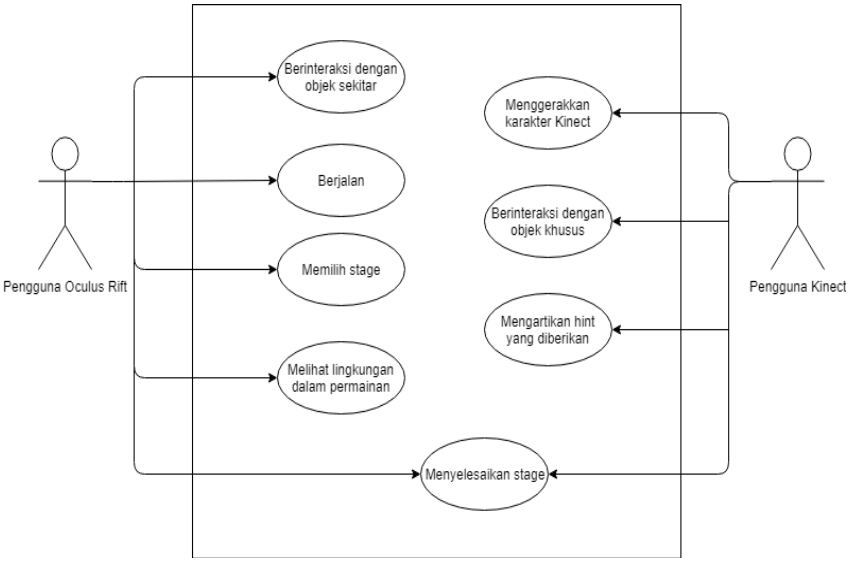
Kedua pemain memiliki sebuah karakter yang dapat digerakkan. Pemain pertama akan menggunakan perangkat Oculus Rift dan melihat melalui orang pertama (*first-person*) dan

pemain kedua dengan perangkat kinect akan diwakilkan oleh sebuah *avatar* di dalam game. *Avatar* pemain kedua akan dapat dilihat oleh pemain pertama. Pergerakan akan sesuai dengan gesture pemain dalam dunia nyata. Pemain dengan Oculus Rift dapat bergerak dengan *controller* yang disediakan oleh perangkat Oculus Rift. Beberapa objek yang terletak pada *dungeon* merupakan alat untuk menyelesaikan teka teki yang diberikan pada pemain.

3.2. Perancangan

3.2.1. Kasus Penggunaan

Game Unknown memiliki kasus penggunaan seperti yang terlihat pada gambar 3.1.



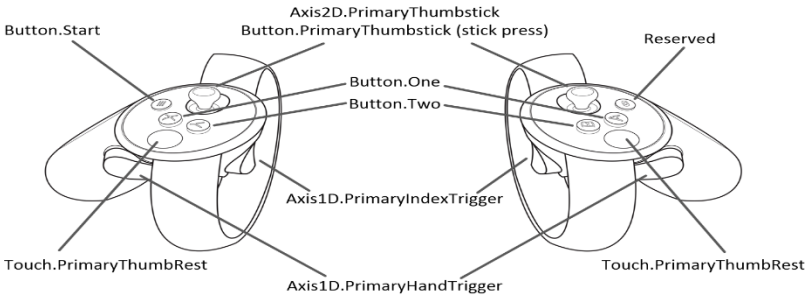
Gambar 3.1 Kasus Peggunaan

Tabel 3.1 Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-001	Berinteraksi dengan objek sekitar	Pengguna Oculus Rift dapat menyentuh, membawa benda yang ada di dalam *dungeon*
2	UC-002	Berjalan	Pengguna Oculus Rift dapat berpindah tempat menggunakan controller Oculus Rift
3	UC-003	Memilih *stage*	Pengguna Oculus Rift dapat memilih stage yang ingin dimainkan pada main menu
4	UC-004	Melihat lingkungan dalam permainan	Pengguna Oculus Rift dapat melihat lingkungan sekitar
5	UC-005	Menggerakkan karakter Kinect	Pengguna Kinect dapat menggerakkan *avatar* kinect yang ada di dalam permainan
6	UC-006	Berinteraksi dengan objek khusus	Pengguna Kinect dapat menggunakan avatar kinect untuk berinteraksi dengan objek sekitar
7	UC-007	Mengartikan hint yang diberikan	Pengguna kinect dapat mengartikan hint yang diberikan
8	UC-008	Menyelesaikan stage	Kedua pengguna dapat menyelesaikan stage dalam permainan

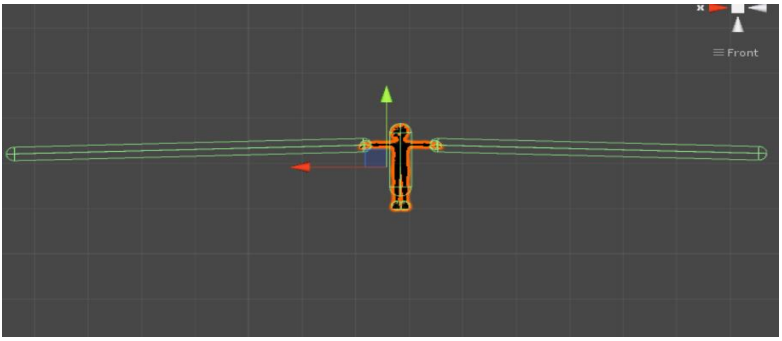
3.2.2. Perancangan Kontrol Pemain

Dalam implementasi pergerakan, pengguna Oculus Rift menggunakan *controller* yang telah disediakan oleh Oculus Rift. Pemain yang menggunakan Oculus Rift dapat memegang dan membawa objek yang ada di dalam permain dengan cara menekan tombol tertentu seperti yang ada pada gambar 3.2.



Gambar 3.2 *Controller* Oculus Rift

Untuk pengguna Kinect, akan dideteksi oleh sensor Kinect. Kinect akan mendeteksi pergerakan tangan dan kaki dari pemain. Untuk dapat berinteraksi dengan objek, karakter Kinect akan diberikan *collider* khusus dibagian tangan dari karakter seperti yang dapat dilihat pada gambar 3.3. *Collider* ini akan berfungsi sebagai *trigger* bila bersentuhan dengan objek yang telah ditentukan, dan akan menyebabkan *event* tertentu.



Gambar 3.3 Karakter Kinect

3.2.3. Perancangan *Stage*

Game Unknown memiliki 6 *stage* atau ruangan yang terletak pada 1 *dungeon* besar. Terdapat *stage* awal sebagai *stage tutorial* dan 5 *stage* yang menerapkan integrasi Kinect dan Oculus Rift. Tujuan akhir dan jumlah kunci yang dibutuhkan untuk menyelesaikan tiap *stage* juga berbeda. Berikut pembagian tentang penugasan dan format *stage* pada tiap *stage*:

***Stage* 1**: *Stage* 1 merupakan tahap *tutorial* untuk membiasakan diri dengan kendali Kinect. Di *stage* ini, pengguna Kinect akan diminta untuk menyentuh pilar yang ada di depan, atas, kiri, dan kanan seperti yang disediakan untuk menghidupkan 4 obor yang akan membuka jalan bagi pengguna Oculus Rift. Stage dapat dilihat pada lampiran 7.13.

***Stage* 2**: *Stage* 2 adalah *stage* pertama yang tidak membutuhkan bantuan dari pengguna Kinect. Di stage ini pengguna Oculus Rift akan diminta untuk mengumpulkan 6 buah tengkorak yang tersembunyi di dalam ruangan. Setelah mengumpulkan semua tengkorak, pemain harus mencari letak dimana tengkorak harus disatukan. Tempat pengumpulan tengkorak terletak di pojok ruangan dimana ada meja yang yang terletak di bawah warna langit langit yang berbeda. Penyelesaian *stage* 2 akan membuka pintu pada stage 3. Stage dapat dilihat pada lampiran 7.14.

***Stage* 3:** Pada *stage* 3, pengguna Kinect akan bertugas kembali. Pemain kinect akan diminta untuk menggerakkan patung yang ada di dalam ruangan. Pemain Oculus harus mengarahkan pengguna Kinect menuju letak patung yang seharusnya. Di tempat dimana obor kosong, disanalah letak patung yang seharusnya. Stage dapat dilihat pada lampiran 7.15.

***Stage* 4**: *Stage* 4 Penyelesaian stage ini ada pada *stage* sebelumnya, yaitu *stage* 2. Pada *stage* 2, terletak sebuah lilin di

meja. Lilin tersebut harus dibawa ke patung yang menunggu untuk membuka jalan ke ruangan berikutnya. Stage dapat dilihat pada lampiran 7.16.

***Stage* 5**: Rintangan pada *stage* 5 adalah pengenalan simbol. Setelah mengetahui simbol yang harus dipakai, maka pemain akan dapat membuka peti harta karun di ujung kanan sisi ruangan stage 5 menggunakan kunci yang berada diawal ruangan. Peti harta karun tersebut akan berisi kunci untuk melanjutkan ke *stage* berikutnya. Stage dapat dilihat pada lampiran 7.17.

***Stage* 6**: *Stage* 6 adalah tahap terakhir permainan. Disini, pemain Kinect akan terkurung di dalam ruangan penjara. Di dalam ruangan tersebut akan ada simbol baru. Simbol tersebut berarti hanya bisa disentuh pemain Kinect. Setelah disentuh, beberapa pintu penjara akan terbuka. Pemain Oculus Rift harus mencari penjara dengan simbol yang sama seperti di *stage* 5 sebelumnya dan masuk. Untuk bisa membuka penjara lain disediakan kunci di dalam ruangan di penjara dengan simbol. Gunakan pedang yang ada di *stage* 5 untuk menghancurkan tong yang menghalangi. Setelah mendapatkan kunci, cari buku sihir di antara ruangan penjara dan menuju ke ujung lorong untuk memenangkan permainan. Stage dapat dilihat pada lampiran 7.18.

BAB IV
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari rancangan yang dibahas pada bab sebelumnya. Pada bab ini akan dijelaskan mengenai integrasi dari Oculus Rift dan Kinect ke dalam permainan. *Game engine* yang digunakan adalah Unity menggunakan bahasa C#.

4.1. Lingkungan Pengembangan Sistem

Lingkungan pengembangan sistem merupakan lingkungan dimana sistem ini dibangun, dimana akan dijelaskan mengenai kebutuhan perangkat yang diperlukan untuk membangun sistem ini. Berikut adalah lingkungan dari pengembangan sistem:

1. Sistem operasi Windows 10 Home Edition 64 bit.
2. Unity Game Engine 2017
3. Visual Studio 2017 sebagai *editor*

4.2. Implementasi *Gameplay*

Implementasi dari permainan dilakukan dengan bahasa pemrograman C#.

4.2.1. Implementasi Pergerakan Pemain

Di dalam permainan, pemain yang menggunakan perangkat Oculus Rift dapat bergerak maju, mundur, kiri, kanan, dan menggenggam objek di dalam *dungeon* menggunakan *controller* Oculus Rift. Sedangkan pemain dengan Kinect dapat menggerakkan karakter sesuai dengan pergerakan tubuh pemain. Implementasi pergerakan karakter oleh pemain sebagaimana yang telah dijelaskan dapat dilihat pada Lampiran 7.1 dan Lampiran 7.2.

4.2.2. Implementasi Stage

Pemain dapat menyelesaikan *stage* setelah mendapatkan kunci dan meletakkannya di tempat yang sesuai. Berikut implementasi *stage* 1 pada Kode Sumber 4.1 dan implementasi

Kinect *stage* 1 pada Kode Sumber 4.2. Tampilan dari *stage* 1 dapat dilihat pada gambar 4.1.

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class Room1Script : MonoBehaviour {
6.
7.      [SerializeField]
8.      private GameObject[] lamp, fire, kinect;
9.      [SerializeField]
10.     private GameObject door;
11.     [SerializeField]
12.     private bool[] flag;
13.     // Use this for initialization
14.     void Start () {
15.
16.         for (int i = 0; i < 4; i++)
17.         {
18.             kinect[i].active = true;
19.             lamp[i].active = false;
20.             fire[i].active = false;
21.         }
22.     }
23.
24.     // Update is called once per frame
25.     void Update () {
26.
27.     }
28.
29.     private void Check()
30.     {
31.         if (flag[0] == true && flag[1] == true && flag[2] ==
    true && flag[3] == true)
32.         {
33.             door.active = false;
34.             kinect[0].active = false;
35.             kinect[1].active = false;
36.             kinect[2].active = false;
37.             kinect[3].active = false;
38.             if (PlayerPrefs.GetInt("Level") < 1)
```

```
            {
                PlayerPrefs.SetInt("Level", 1);
            }
        }
    }

    public void Lamp(int direction)
    {
        Check();

        if(flag[direction] == true)
        {
            lamp[direction].active = false;
            fire[direction].active = false;
            flag[direction] = false;
        }
        else
        {
            lamp[direction].active = true;
            fire[direction].active = true;
            flag[direction] = true;
        }
    }
}
```

Kode Sumber 4.1 *Stage* 1

```
private void OnTriggerEnter(Collider collision)
    {
        if (collision.gameObject.tag == "Tangan")
        {
            room1Script.Lamp(direction);
        }

    }
```

Kode Sumber 4.2 Kinect *button stage* 1

Gambar 4.1 *Stage* 1

Stage 3 dibagi menjadi 3 bagian, yaitu implementasi tombol bagi pengguna Kinect, implementasi cara menggerakkan patung, dan implementasi saat patung sudah dalam posisi yang tepat atau kondisi *clear stage*. Berikut implementasinya dalam Lampiran 7.3, Lampiran 7.4, dan Lampiran 7.5. Desain dari *Stage* 3 dapat dilihat pada gambar 4.2.

Gambar 4.2 *Stage* 2

Stage 6 terdiri dari 3 tahap implementasi yaitu, ruangan tempat pemain Kinect, saat pemain mendapatkan buku sihir, dan saat buku sihir digunakan di ujung lorong ruangan. Implementasi akan ditampilkan dalam Lampiran 7.6, 7.7, 7.8. Desain dari *Stage* 6 dapat dilihat pada gambar 4.3.

Gambar 4.3 *Stage* 6

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas hasil dan pembahasan pada aplikasi yang dikembangkan. Pada bab ini akan dijelaskan tentang data yang digunakan, hasil yang didapatkan dari penggunaan perangkat lunak dan uji coba yang dilakukan pada perangkat lunak yang telah dikerjakan untuk menguji apakah fungsionalitas aplikasi telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya

5.1. Lingkungan Pengujian

Spesifikasi dari perangkat keras dan perangkat lunak yang digunakan untuk pengujian aplikasi Unknown tertera pada Tabel 5.1.

Tabel 5.1 Lingkungan pengujian

Perangkat Keras 1	msi GE 63VR 7RE CPU: i7 7700HQ RAM : 16 GB VGA : NVIDIA GTX1060
Perangkat Keras 2	Oculus Rift
Perangkat Keras 3	Kinect 2.0
Perangkat Lunak	Windows 10 Home Edition

5.2. Pengujian Aplikasi

Pengujian dilakukan untuk mengetahui apakah aplikasi yang dirancang sudah diimplementasikan dan berjalan seperti yang telah dirancang sebelumnya. Pengujian dilakukan dengan metode *black-box*.

5.2.1. Skenario Pengujian

Skenario pengujian fungsionalitas pada Tabel 5.2.

Tabel 5.2 Pengujian aplikasi permainan

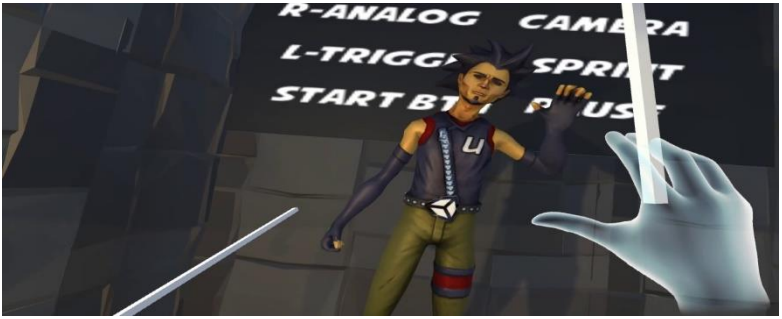
Kondisi Awal	Pengguna berada pada layar *Main Menu*
Prosedur Pengujian	Pengguna masuk ke *Stage Selection*, memilih *Stage* yang akan dimainkan, lalu memainkan permainan hingga selesai
Hasil yang diharapkan	Pengguna berhasil menyelesaikan permainan.
Hasil yang diperoleh	Pengguna berhasil menyelesaikan permainan.
Kesimpulan	Pengujian berhasil

5.2.1.1. Pengujian Main Menu dan Stage Selection

Pengujian pertama adalah pemain yang menggunakan Oculus Rift masuk ke dalam tampilan *Main Menu* seperti yang terlihat pada Gambar 5.1. Pengguna Oculus Rift dapat bergerak di dalam ruangan dan melihat layar *Stage Selection*. Dalam pengujian ini, dapat dilihat bahwa UC-004 telah berfungsi dengan baik.

Gambar 5.1 Tampilan *Main Menu*

Pengguna Oculus Rift dapat melihat pergerakan dari pengguna Kinect seperti yang terlihat pada Gambar 5.2 dari gambar dibawah, dapat disimpulkan bahwa UC-005 telah berhasil diimplementasikan.



Gambar 5.2 Tampilan pergerakan Kinect

Setelah masuk ke dalam ruangan, pemain Oculus Rift dapat melihat layar *Stage Selection* seperti yang terlihat pada Gambar 5.3.

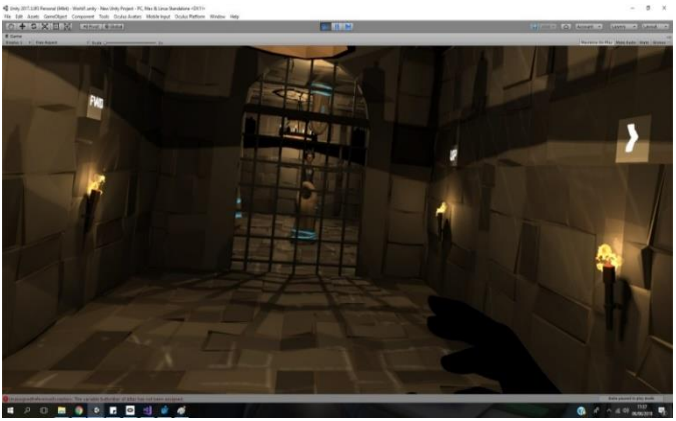
Gambar 5.3 Tampilan *Stage Selection*

Pemain dapat memilih *stage* yang. Setelah pemain memilih *stage* yang diinginkan dan menekan tombol *play*. Pemain akan melihat cahaya dan pemain akan berpindah ke dalam *stage* yang dipilih.

Setelah melakukan pengujian, sistem berhasil masuk ke *Game Menu*, Pengguna Oculus dapat melihat pergerakan pengguna Kinect dan melakukan *load* sesuai *stage* yang dipilih. Sehingga dapat disimpulkan bahwa pengujian untuk layar *Main Menu* dan *Stage Selection* berhasil. Dari gambar 5.3 dapat disimpulkan bahwa UC-003 telah berfungsi.

5.2.1.2. Pengujian Pergerakan Karakter Pemain

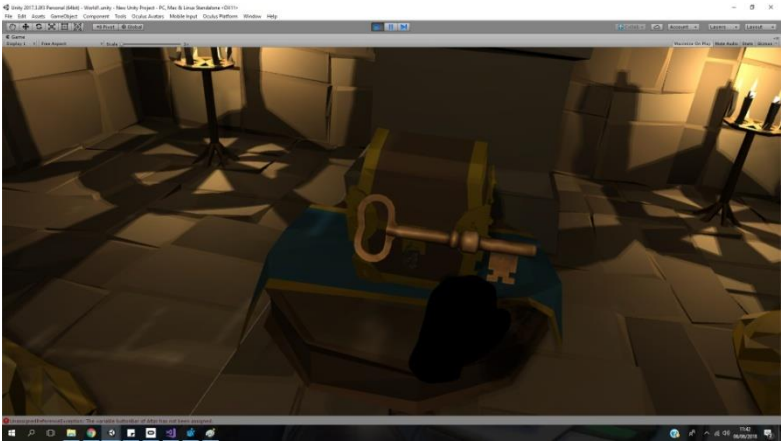
Dalam pengujian ini pemain diminta untuk menggerakkan karakter, dan berinteraksi dengan objek sekitar. Pemain dapat menggerakkan karakter dengan cara menggunakan joystick yang disediakan oleh Oculus Rift. Setelah pengujian, karakter pemain dapat bergerak dengan baik. Berikut hasil pengujian ini dapat dilihat pada Gambar 5.4. Dalam pengujian ini membuktikan bahwa UC-001, dan UC-002 telah berfungsi.



Gambar 5.4 Tampilan pergerakakan karakter

5.2.1.3. Pengujian Stage Clear

Pengujian ini dilakukan untuk mengetahui apakah seluruh stage dapat diselesaikan. Pemain akan mencoba untuk menyelesaikan setiap *stage* dari awal sampai terakhir. Tampilan pemain menyelesaikan suatu *stage* dapat dilihat pada Gambar 5.5. Dari pengujian ini dapat diambil kesimpulan bahwa UC-006, UC-007, dan UC-008 telah bekerja sebagaimana mestinya.



Gambar 5.5 Tampilan pemain menyelesaikan *stage*

5.2.2. Hasil Pengujian

Hasil pengujian yang dilakukan berdasarkan skenario sebelumnya, menunjukkan bahwa seluruh fungsi pada permainan telah berjalan dengan baik dan sesuai dengan yang telah dibuat pada tahap perancangan. Hasil pengujian dapat dilihat pada Tabel 5.3.

Tabel 5.3 Hasil pengujian

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan	Hasil
1	UC-001	Berinteraksi dengan objek sekitar	Pengguna Oculus Rift dapat menyentuh, membawa benda yang ada di dalam *dungeon*	Berhasil
2	UC-002	Berjalan	Pengguna Oculus Rift dapat berpindah tempat menggunakan controller Oculus Rift	Berhasil
3	UC-003	Memilih *stage*	Pengguna Oculus Rift dapat memilih stage yang ingin dimainkan pada main menu	Berhasil
4	UC-004	Melihat lingkungan dalam permainan	Pengguna Oculus Rift dapat melihat lingkungan sekitar	Berhasil
5	UC-005	Menggerakkan karakter Kinect	Pengguna Kinect dapat menggerakkan *avatar* kinect yang ada di dalam permainan	Berhasil
6	UC-006	Berinteraksi dengan objek khusus	Pengguna Kinect dapat menggunakan avatar kinect untuk berinteraksi dengan objek sekitar	Berhasil
7	UC-007	Mengartikan hint yang diberikan	Pengguna kinect dapat mengartikan hint yang diberikan	Berhasil
8	UC-008	Menyelesaikan stage	Kedua pengguna dapat menyelesaikan stage dalam permainan	Berhasil

5.3. Pengujian Pengguna

Pengujian pengguna adalah untuk mendapatkan informasi mengenai pendapat pengguna mengenai game ‘Unknown’. Pengujian dilakukan ke 16 pengguna. Para pengguna ini terdiri dari 15 mahasiswa ITS, dan 1 mahasiswa Unair yang sebagian besar belum pernah menggunakan Kinect.

5.3.1. Skenario Pengujian

Pengujian dilakukan oleh pengguna (pemain) dengan memainkan game ‘Unknown’. Pemain memainkan seluruh *level* mulai dari awal hingga terakhir. Pemain akan bertukar peran setelah berhasil menyelesikan seluruh *level*. Pemain yang menggunakan Kinect akan menggunakan Oculus Rift begitu juga sebaliknya. Hal ini dilakukan agar kedua penguji mendapatkan pengalaman yang sama. Kemudian pemain diberikan *form* berisi beberapa pertanyaan. Pertanyaan yang diberikan meliputi pendapatnya mengenai game ‘Unknown’, dan masukan untuk pengembangan aplikasi permainan lebih lanjut.

Form yang diberikan kepada pengguna adalah seperti yang terlihat pada Tabel 5.4.

Tabel 5.4 Form Penilaian Pengguna

No	Keterangan	Nilai
1	Sangat Tidak Setuju (STS)	1
2	Tidak Setuju (TS)	2
3	Kurang Setuju (KS)	3
4	Cukup Setuju (CS)	4
5	Setuju (S)	5
6	Sangat Setuju (SS)	6

No	Pertanyaan
1	Pernahkah anda menggunakan Kinect?
2	Pernahkah anda menggunakan Oculus Rift?

3	Apakah anda terbiasa dengan control Oculus Rift?						
4	Apakah anda terbiasa dengan control Kinect?						
5	Apakah anda suka dengan game puzzle?						
Parameter Antar Muka		STS	TS	KS	CS	S	SS
1	Tampilan menu mudah dipahami						
2	Tampilan menu memiliki warna, dan desain yang menarik						
3	Hint dapat dibaca dengan mudah						
Parameter Immersive		STS	TS	KS	CS	S	SS
4	Objek di dalam game terasa nyata						
5	Pergerakan karakter Kinect terasa nyata						
6	Objek yang ditampilkan sesuai dengan tema game						
7	Desain dari dungeon mendukung tema misterius yang diusulkan						
8	Desain dari dungeon mendukung teka teki yang ada dalam level						
9	Hint yang diberikan menantang pemain						
10	Gameplay mendukung tema misterius yang diusulkan						
11	Pergerakan Oculus Rift terasa nyata						
Parameter Kenyamanan		STS	TS	KS	CS	S	SS
12	Saat menggunakan Oculus Rift tidak terasa mual atau						

	pusing						
13	Kontrol pergerakan player tidak membingungkan						

5.3.2. Hasil Pengujian

Setelah dilakukan pengujian oleh pengguna, didapatkan hesil seperti yang tertera pada Tabel 5.5.

Tabel 5.5 Hasil Pengujian Pengguna

Parameter Antar Muka		STS	TS	KS	CS	S	SS	Rata-Rata
1	Tampilan menu mudah dipahami	0	0	3	3	8	2	4.56
2	Tampilan menu memiliki warna, dan desain yang menarik	0	0	0	6	6	4	4.87
3	Hint dapat dibaca dengan mudah	0	1	2	3	2	8	4.87
Parameter Immersive		STS	TS	KS	CS	S	SS	Rata-Rata
4	Objek di dalam game terasa nyata	0	0	0	3	7	6	5.18
5	Pergerakan karakter Kinect terasa nyata	0	0	2	4	6	4	4.75
6	Objek yang ditampilkan sesuai dengan tema game	0	0	0	6	8	2	4.75

7	Desain dari dungeon mendukung tema misterius yang diusulkan	0	0	0	6	7	3	4.81
8	Desain dari dungeon mendukung teka teki yang ada dalam level	0	0	0	4	12	0	4.94
9	Hint yang diberikan menantang pemain	0	0	0	9	4	3	4.35
10	Gameplay mendukung tema misterius yang diusulkan	0	0	0	4	9	3	4.94
11	Pergerakan Oculus Rift terasa nyata	0	0	0	2	6	8	5.37
Parameter Kenyamanan		STS	TS	KS	CS	S	SS	Rata-Rata
12	Saat menggunakan Oculus Rift tidak terasa mual atau pusing	1	1	1	3	3	7	4.68
13	Kontrol pergerakan player tidak membingungkan.	0	0	0	8	6	2	4.62

Pengguna menjelaskan bahwa permainan ‘Unknown’ sudah baik di bidang tampilan dan desain namun dibutuhkan perhatian lebih pada bidang kenyamanan saat bermain dan bagi pemain yang menggunakan perangkat kinect masih terdapat bug. Bug yang terjadi disebabkan oleh beberapa hal.

1. Tidak terdapat daya yang cukup oleh USB. Cara untuk menyelesaikan masalah ini adalah dengan menambahkan USB 3.0 kedalam perangkat komputer.
2. Kinect tidak dapat mendeteksi pergerakan pengguna dengan baik bila ada manusia lain yang berada di lingkungan deteksi Kinect. Cara untuk menyelesaikan masalah ini ada dengan mensterilkan area pengguna Kinect dari manusia selain pengguna.

Untuk kenyamanan pengguna Oculus Rift dapat dilakukan pembiasaan, karena masalah dalam pengguna Oculus Rift adalah mual ketika bermain. Hal tersebut adalah *Motion Sickness. Motion Sickness* adalah perasaan yang tidak enak pada tubuh karena gerakan yang berulang. Hal ini disebabkan oleh gerakan yang tidak disadari atau tidak disebabkan oleh kesadaran kita. Oleh karena itu makin sering pengguna menggunakan Oculus Rift makin kecil kemungkinan pengguna terkena *Motion Sickness*.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Penerapan perangkat Kinect dalam perancangan 'Unknown' diterapkan dalam bentuk script pada game engine, dan permainan dapat berjalan sebagaimana mestinya. Namun masih terjadi masalah ketika terdapat banyak orang dibelakang pengguna Kinect. Akan terjadi kesalahan ketika Kinect mencoba melakukan deteksi pergerakan dari pemain.
2. Penerapan perangkat Oculus Rift dalam perancangan 'Unknown' diterapkan dalam bentuk script pada game engine, dan permainan dapat berjalan sebagaimana mestinya. Dalam penggunaan Oculus Rift, terjadi kendala dikarenakan faktor manusia. Kendala tersebut adalah *Motion Sickness*. *Motion Sickness* tersebut terjadi karena pengguna tidak terbiasa menggunakan perangkat Oculus Rift. Dapat diatasi dengan membiasakan menggunakan Oculus Rift.
3. Oculus Rift dan Kinect telah terimplementasi kedalam *game* 'Unknown'. Namun masih ada beberapa masalah ketika game dijalankan. Diantaranya, ketika dimainkan di komputer dengan USB yang tidak mempunyai daya yang cukup akan membuat Kinect tidak responsive. Dapat diatasi dengan cara memperbanyak USB 3.0 pada komputer.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan:

1. Menambah kenyamanan pemain saat di dalam game.
2. Menambah interaksi pemain dengan objek di dalam permainan.
3. Membuat animasi pergerakan pemain kinect lebih halus.

DAFTAR PUSTAKA

[1] Ijsselsteijn, Wijanand et al. (2008). Measuring the Experience of Digital Game Enjoyment. Proceedings of Measuring Behavior 2008, 6th International Conference on Methods and Techniques in Behavioral Research. Maastricht, Netherlands, August 26-29, 2008.

[2] Rucker, Rudy. (2003). Software Engineering and Computer Games. Britain: Dorset Press.

[3] Dix, Alan et al. (2004). Human-Computer Interaction 3rd. England: Pearson Education Limited.

[4] Unity. "CREATE GAMES, CONNECT WITH YOUR AUDIENCE, AND ACHIEVE SUCCESS".Unity Technologies, 2017. [Online]. Available: http://unity3d.com/unity. [Diakses 2September 2017].

[5] Buck, Jamis. (2015). *Maze* for Programmers. United States of America: The Pragmatic Programmers.

[6] Kemppainen, Jaakko. (2014). Designing a Knowledge Based Puzzle Game. Finlandia: Aalto University.

[7] https://developer.microsoft.com/en-us/windows/kinect

[8] https://www.oculus.com/rift

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Saddhana Arta Daniswara, mahasiswa asal Surabaya yang lahir di Kediri pada tanggal 21 Maret tahun 1998. Akrab disapa dengan panggilan Danis, penulis juga mempunyai hobi bermain game. Danis tertarik dengan dunia IT semenjak duduk di bangku SMA. Danisberbagai organisasi yang ada di ITS. Diantaranya adalah UKM IFLS dan staff di event jurusan informatika yaitu Schematics.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

```
1. public virtual void UpdateMovement()
2.     {
3.         if (HaltUpdateMovement)
4.             return;
5.
6.         bool moveForward = Input.GetKey(KeyCode.W) || Input.Ge
   tKey(KeyCode.UpArrow);
7.         bool moveLeft = Input.GetKey(KeyCode.A) || Input.GetKe
   y(KeyCode.LeftArrow);
8.         bool moveRight = Input.GetKey(KeyCode.D) || Input.GetK
   ey(KeyCode.RightArrow);
9.         bool moveBack = Input.GetKey(KeyCode.S) || Input.GetKe
   y(KeyCode.DownArrow);
10.
11.        bool dpad_move = false;
12.
13.        if (OVRInput.Get(OVRInput.Button.DpadUp))
14.        {
15.            moveForward = true;
16.            dpad_move   = true;
17.
18.        }
19.
20.        if (OVRInput.Get(OVRInput.Button.DpadDown))
21.        {
22.            moveBack  = true;
23.            dpad_move = true;
24.        }
25.
26.        MoveScale = 1.0f;
27.
28.        if ( (moveForward && moveLeft) || (moveForward && move
   Right) ||
29.             (moveBack && moveLeft)    || (moveBack && moveRig
   ht) )
30.            MoveScale = 0.70710678f;
31.
```

```
32.         // No positional movement if we are in the air
33.         if (!Controller.isGrounded)
34.             MoveScale = 0.0f;
35.
36.         MoveScale *= SimulationRate * Time.deltaTime;
37.
38.         // Compute this for key movement
39.         float moveInfluence = Acceleration * 0.1f * MoveScale
    * MoveScaleMultiplier;
40.
41.         // Run!
42.         if (dpad_move || Input.GetKey(KeyCode.LeftShift) || In
    put.GetKey(KeyCode.RightShift))
43.             moveInfluence *= 2.0f;
44.
45.         Quaternion ort = transform.rotation;
46.         Vector3 ortEuler = ort.eulerAngles;
47.         ortEuler.z = ortEuler.x = 0f;
48.         ort = Quaternion.Euler(ortEuler);
49.
50.         if (moveForward)
51.             MoveThrottle += ort * (transform.lossyScale.z * mo
    veInfluence * Vector3.forward);
52.         if (moveBack)
53.             MoveThrottle += ort * (transform.lossyScale.z * mo
    veInfluence * BackAndSideDampen * Vector3.back);
54.         if (moveLeft)
55.             MoveThrottle += ort * (transform.lossyScale.x * mo
    veInfluence * BackAndSideDampen * Vector3.left);
56.         if (moveRight)
57.             MoveThrottle += ort * (transform.lossyScale.x * mo
    veInfluence * BackAndSideDampen * Vector3.right);
58.
59.         Vector3 euler = transform.rotation.eulerAngles;
60.
61.         bool curHatLeft = OVRInput.Get(OVRInput.Button.Primary
    Shoulder);
62.
63.         if (curHatLeft && !prevHatLeft)
64.             euler.y -= RotationRatchet;
65.
66.         prevHatLeft = curHatLeft;
67.
```

```
68.         bool curHatRight = OVRInput.Get(OVRInput.Button.Second
    aryShoulder);
69.
70.         if(curHatRight && !prevHatRight)
71.             euler.y += RotationRatchet;
72.
73.         prevHatRight = curHatRight;
74.
75.         euler.y += buttonRotation;
76.         buttonRotation = 0f;
77.
78.         float rotateInfluence = SimulationRate * Time.deltaTim
    e * RotationAmount * RotationScaleMultiplier;
79.
80. #if !UNITY_ANDROID || UNITY_EDITOR
81.         if (!SkipMouseRotation)
82.             euler.y += Input.GetAxis("Mouse X") * rotateInflue
    nce * 3.25f;
83. #endif
84.
85.         moveInfluence = Acceleration * 0.1f * MoveScale * Move
    ScaleMultiplier;
86.
87. #if !UNITY_ANDROID // LeftTrigger not avail on Android game pa
    d
88.         moveInfluence *= 1.0f + OVRInput.Get(OVRInput.Axis1D.P
    rimaryIndexTrigger);
89. #endif
90.
91.         Vector2 primaryAxis = OVRInput.Get(OVRInput.Axis2D.Pri
    maryThumbstick);
92.
93.         if(primaryAxis.y > 0.0f)
94.             MoveThrottle += ort * (primaryAxis.y * transform.l
    ossyScale.z * moveInfluence * Vector3.forward);
95.
96.         if(primaryAxis.y < 0.0f)
97.             MoveThrottle += ort * (Mathf.Abs(primaryAxis.y) *
    transform.lossyScale.z * moveInfluence * BackAndSideDampen * V
    ector3.back);
98.
99.         if(primaryAxis.x < 0.0f)
```

```
100.             MoveThrottle += ort * (Mathf.Abs(primaryAxis.x)
    * transform.lossyScale.x * moveInfluence * BackAndSideDampen *
     Vector3.left);
101.
102.         if(primaryAxis.x > 0.0f)
103.             MoveThrottle += ort * (primaryAxis.x * transform
    .lossyScale.x * moveInfluence * BackAndSideDampen * Vector3.ri
    ght);
104.
105.         Vector2 secondaryAxis = OVRInput.Get(OVRInput.Axis2D
    .SecondaryThumbstick);
106.
107.         euler.y += secondaryAxis.x * rotateInfluence;
108.
109.         transform.rotation = Quaternion.Euler(euler);
110.     }
111.  /// <summary>
112.  /// Gets the halt update movement.
113.  /// </summary>
114.  /// <param name="haltUpdateMovement">Halt update movement.</
    param>
115.  public void GetHaltUpdateMovement(ref bool haltUpdateMovemen
    t)
116.  {
117.      haltUpdateMovement = HaltUpdateMovement;
118.  }
119.
120.  /// <summary>
121.  /// Sets the halt update movement.
122.  /// </summary>
123.  /// <param name="haltUpdateMovement">If set to <c>true</c> h
    alt update movement.</param>
124.  public void SetHaltUpdateMovement(bool haltUpdateMovement)
125.  {
126.      HaltUpdateMovement = haltUpdateMovement;
127.  }
```

Lampiran 7.1 Implementasi pergerakan karakter Oculus Rift

```
protected void MoveAvatar(Int64 UserID)
    {
        if((moveRate == 0f) || !kinectManager ||
           !kinectManager.IsJointTracked(UserID, (int)KinectInterop.JointType.SpineBase))
        {
            return;
        }

        // get the position of user's spine base
        Vector3 trans = kinectManager.GetUserPosition(UserID);

        if(flipLeftRight)
            trans.x = -trans.x;

        // use the color overlay position if needed
        if(posRelativeToCamera && posRelOverlayColor)
        {
            if(backgroundPlane && planeRectSet)
            {
                // get the plane overlay position
                trans = kinectManager.GetJointPosColorOverlay(UserID, (int)KinectInterop.JointType.SpineBase, planeRect);
                trans.z = backgroundPlane.position.z - posRelativeToCamera.transform.position.z - 0.1f;  // 10cm offset
            }
            else
            {
                Rect backgroundRect = posRelativeToCamera.pixelRect;
                PortraitBackground portraitBack = PortraitBackground.Instance;

                if(portraitBack && portraitBack.enabled)
                {
                    backgroundRect = portraitBack.GetBackgroundRect();
                }

                trans = kinectManager.GetJointPosColorOverlay(UserID, (int)KinectInterop.JointType.SpineBase, posRelativeToCamera, backgroundRect);
```

```
34.             }
35.
36.             if(flipLeftRight)
37.                 trans.x = -trans.x;
38.         }
39.
40.         // invert the z-coordinate, if needed
41.         if(posRelativeToCamera && posRelInvertedZ)
42.         {
43.             trans.z = -trans.z;
44.         }
45.
46.         if(!offsetCalibrated)
47.         {
48.             offsetCalibrated = true;
49.
50.             offsetPos.x = trans.x;  // !mirroredMovement ? tra
    ns.x * moveRate : -trans.x * moveRate;
51.             offsetPos.y = trans.y;  // trans.y * moveRate;
52.             offsetPos.z = !mirroredMovement && !posRelativeToC
    amera ? -trans.z : trans.z;  // -trans.z * moveRate;
53.
54.             if(posRelativeToCamera)
55.             {
56.                 Vector3 cameraPos = posRelativeToCamera.transf
    orm.position;
57.                 Vector3 bodyRootPos = bodyRoot != null ? bodyR
    oot.position : transform.position;
58.                 Vector3 hipCenterPos = bodyRoot != null ? body
    Root.position : (bones != null && bones.Length > 0 && bones[0]
     != null ? bones[0].position : Vector3.zero);
59.
60.                 float yRelToAvatar = 0f;
61.                 if(verticalMovement)
62.                 {
63.                     yRelToAvatar = (trans.y - cameraPos.y) -
    (hipCenterPos - bodyRootPos).magnitude;
64.                 }
65.                 else
66.                 {
67.                     yRelToAvatar = bodyRootPos.y -
    cameraPos.y;
68.                 }
```

```
69.
70.                 Vector3 relativePos = new Vector3(trans.x, yRe
    lToAvatar, trans.z);
71.                 Vector3 newBodyRootPos = cameraPos + relativeP
    os;
72.
73. //              if(offsetNode != null)
74. //              {
75. //                  newBodyRootPos += offsetNode.transform.pos
    ition;
76. //              }
77.
78.                 if(bodyRoot != null)
79.                 {
80.                     bodyRoot.position = newBodyRootPos;
81.                 }
82.                 else
83.                 {
84.                     transform.position = newBodyRootPos;
85.                 }
86.
87.                 bodyRootPosition = newBodyRootPos;
88.             }
89.         }
90.
91.         // transition to the new position
92.         Vector3 targetPos = bodyRootPosition + Kinect2AvatarPo
    s(trans, verticalMovement);
93.
94.         if(isRigidBody && !verticalMovement)
95.         {
96.             // workaround for obeying the physics (e.g. gravit
    y falling)
97.             targetPos.y = bodyRoot != null ? bodyRoot.position
    .y : transform.position.y;
98.         }
99.
100.             if (verticalMovement && verticalOffset != 0f &&
101.                 bones[0] != null && bones[3] != null)
102.             {
103.                 Vector3 dirSpine = bones[3].position -
     bones[0].position;
```

```
104.                 targetPos += dirSpine.normalized * verticalOf
    fset;
105.             }
106.
107.             if (forwardOffset != 0f &&
108.                 bones[0] != null && bones[3] != null && bones
    [5] != null && bones[11] != null)
109.             {
110.                 Vector3 dirSpine = (bones[3].position -
     bones[0].position).normalized;
111.                 Vector3 dirShoulders = (bones[11].position -
     bones[5].position).normalized;
112.                 Vector3 dirForward = Vector3.Cross(dirShoulde
    rs, dirSpine).normalized;
113.
114.                 targetPos += dirForward * forwardOffset;
115.             }
116.
117.             if(groundedFeet)
118.             {
119.                 // keep the current correction
120.                 float fLastTgtY = targetPos.y;
121.                 targetPos.y += fFootDistance;
122.
123.                 float fNewDistance = GetDistanceToGround();
124.                 float fNewDistanceTime = Time.time;
125.
126.     //          Debug.Log(string.Format("PosY: {0:F2}, LastY:
     {1:F2},  TgrY: {2:F2}, NewDist: {3:F2}, Corr: {4:F2}, Time: {
    5:F2}", bodyRoot != null ? bodyRoot.position.y : transform.pos
    ition.y,
127.     //              fLastTgtY, targetPos.y, fNewDistance, fFo
    otDistance, fNewDistanceTime));
128.
129.                 if(Mathf.Abs(fNewDistance) >= 0.01f && Mathf.
    Abs(fNewDistance -
     fFootDistanceInitial) >= maxFootDistanceGround)
130.                 {
131.                     if((fNewDistanceTime -
     fFootDistanceTime) >= maxFootDistanceTime)
132.                     {
133.                         fFootDistance += (fNewDistance -
     fFootDistanceInitial);
```

```
134.                         fFootDistanceTime = fNewDistanceTime;
135.
136.                         targetPos.y = fLastTgtY + fFootDistan
    ce;
137.
138.     //                  Debug.Log(string.Format("   >> change
    ({0:F2})! -
     Corr: {1:F2}, LastY: {2:F2},  TgrY: {3:F2} at time {4:F2}",
139.     //                              (fNewDistance -
     fFootDistanceInitial), fFootDistance, fLastTgtY, targetPos.y,
     fFootDistanceTime));
140.                     }
141.                 }
142.                 else
143.                 {
144.                     fFootDistanceTime = fNewDistanceTime;
145.                 }
146.             }
147.
148.             if(bodyRoot != null)
149.             {
150.                 bodyRoot.position = smoothFactor != 0f ?
151.                     Vector3.Lerp(bodyRoot.position, targetPos
    , smoothFactor * Time.deltaTime) : targetPos;
152.             }
153.             else
154.             {
155.                 transform.position = smoothFactor != 0f ?
156.                     Vector3.Lerp(transform.position, targetPo
    s, smoothFactor * Time.deltaTime) : targetPos;
157.             }
158.         }
```

Lampiran 7.2 Implementasi pergerakan karakter Kinect

```
1.      using System.Collections;
2.      using System.Collections.Generic;
3.      using UnityEngine;
4.
5.      public class Room3Controller : MonoBehaviour {
6.
7.          [SerializeField]
8.          private int direction;
9.          [SerializeField]
10.         private Room3StatueController room3StatueController;

11.
12.         // Use this for initialization
13.         void Start () {
14.
15.         }
16.
17.         // Update is called once per frame
18.         void Update () {
19.
20.         }
21.
22.         private void OnTriggerEnter(Collider collision)
23.         {
24.             if (collision.gameObject.tag == "Tangan")
25.             {
26.                 if (direction == 1)
27.                 {
28.                     room3StatueController.Room3Forward();
29.                 }
30.                 else if (direction == 2)
31.                 {
32.                     room3StatueController.Room3Left();
33.                 }
34.                 else if (direction == 3)
35.                 {
36.                     room3StatueController.Room3Right();
37.                 }
38.                 else
39.                 {
40.                     room3StatueController.Room3Backward();
41.                 }
42.             }
```

```
43.         }
44.
45.         private void OnTriggerExit(Collider collision)
46.         {
47.             if (collision.gameObject.tag == "Tangan")
48.             {
49.                 room3StatueController.Stop();
50.             }
51.
52.         }
53.     }
54.
55. using System.Collections;
56. using System.Collections.Generic;
57. using UnityEngine;
```

Lampiran 7.3 Implementasi tombol Kinect

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. public class Room3StatueController : MonoBehaviour {
6.
7.     [SerializeField]
8.     private float moveSpeed;
9.     private CharacterController controller;
10.    [SerializeField]
11.    private float gravityScale;
12.
13.    private Vector3 moveDirection;
14.
15.    // Use this for initialization
16.    void Start () {
17.        controller = GetComponent<CharacterController>();
18.    }
19.
20.    // Update is called once per frame
21.    void Update () {
22.        //moveDirection = new Vector3(Input.GetAxis("Horizonta
   l") * moveSpeed, 0f, Input.GetAxis("Vertical") * moveSpeed);
23.        moveDirection.y = moveDirection.y + (Physics.gravity.y
    * gravityScale);
24.
25.        controller.Move(moveDirection * Time.deltaTime);
26.    }
27.
28.
29.    public void Room3Forward()
30.    {
31.        moveDirection = new Vector3(0f, 0f, -moveSpeed);
32.        //controller.Move(moveDirection * Time.deltaTime);
33.    }
34.
35.    public void Room3Backward()
36.    {
37.        moveDirection = new Vector3(0f, 0f, moveSpeed);
38.        //controller.Move(moveDirection * Time.deltaTime);
39.    }
40.
41.    public void Room3Left()
```

```
42.     {
43.         moveDirection = new Vector3(moveSpeed, 0f, 0f);
44.         //controller.Move(moveDirection * Time.deltaTime);
45.     }
46.
47.     public void Room3Right()
48.     {
49.         moveDirection = new Vector3(-moveSpeed, 0f, 0f);
50.         //controller.Move(moveDirection * Time.deltaTime);
51.     }
52.
53.     public void Stop()
54.     {
55.         moveDirection = new Vector3(0f, 0f, 0f);
56.         //controller.Move(moveDirection * Time.deltaTime);
57.     }
58. }
```

Lampiran 7.4 Implementasi pergerakan patung

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class StatueInPosition : MonoBehaviour {
6.
7.      [SerializeField]
8.      private GameObject statue, kinect, door, particle;
9.
10.     // Use this for initialization
11.     void Start () {
12.
13.     }
14.
15.     // Update is called once per frame
16.     void Update () {
17.
18.     }
19.
20.     private void OnTriggerEnter(Collider collision)
21.     {
22.         if (collision.gameObject.tag == "Statue")
23.         {
24.             collision.gameObject.active = false;
25.             statue.active = true;
26.             kinect.active = false;
27.             door.active = false;
28.             particle.GetComponent<ParticleSystem>().Play();
29.             if (PlayerPrefs.GetInt("Level") < 3)
30.             {
31.                 PlayerPrefs.SetInt("Level", 3);
32.             }
33.             Invoke("StopEmitter", 3);
34.         }
35.
36.     }
37.
38.     private void StopEmitter()
39.     {
40.         particle.GetComponent<ParticleSystem>().Stop();
41.     }
42. }
43.
```

```
44. moveDirection = new Vector3(0f, 0f, 0f);
45.         //controller.Move(moveDirection * Time.deltaTime);
46.     }
```

Lampiran 7.5 Implementasi *Clear Stage* 3

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class TouchRoom6Kinect : MonoBehaviour {
6.
7.      [SerializeField] private GameObject[] prizon;
8.
9.      // Use this for initialization
10.     void Start () {
11.
12.     }
13.
14.     // Update is called once per frame
15.     void Update () {
16.
17.     }
18.
19.     private void OnTriggerEnter(Collider collision)
20.     {
21.         if (collision.gameObject.tag == "Tangan")
22.         {
23.             //prizon.GetComponent<MeshRenderer>().enabled = f
        alse;
24.             for (int i = 0; i < prizon.Length; i++)
25.             {
26.                 prizon[i].SetActive(false);
27.             }
28.         }
29.
30.     }
31.
32.     private void OnTriggerExit(Collider collision)
33.     {
34.         if (collision.gameObject.tag == "Tangan")
35.         {
36.             //prizon.GetComponent<MeshRenderer>().enabled = t
        rue;
37.             for (int i = 0; i < prizon.Length; i++)
38.             {
39.                 prizon[i].SetActive(true);
40.             }
41.         }
```

```
42.
43.     }
44.
45. }
```

Lampiran 7.6 Implementasi ruangan Kinect

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
4.
5.  public class GrimoireRoom6 : MonoBehaviour {
6.
7.      [SerializeField] private MagicRoom6 magicRoom6;
8.      [SerializeField] private GameObject magicCircle;
9.
10.     // Use this for initialization
11.     void Start () {
12.
13.     }
14.
15.     // Update is called once per frame
16.     void Update () {
17.
18.     }
19.
20.     private void OnTriggerEnter(Collider collision)
21.     {
22.         if (collision.gameObject.tag == "TanganOculus")
23.         {
24.             if (magicRoom6.grimoireOn == false)
25.             {
26.                 magicRoom6.grimoireOn = true;
27.                 magicCircle.GetComponent<ParticleSystem>().Pl
        ay();
28.                 magicCircle.GetComponent<AudioSource>().Play(
        );
29.             }
30.         }
31.
32.     }
33.
34. }
```

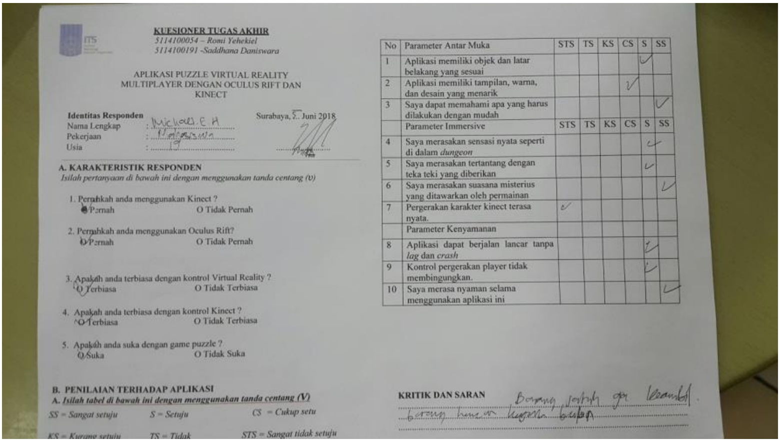
Lampiran 7.7 Implementasi mendapat buku sihir

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4. using UnityEngine.SceneManagement;
5.
6. public class MagicRoom6 : MonoBehaviour {
7.
8.     public bool grimoireOn;
9.     [SerializeField]
10.    private GameObject magicCircle;
11.
12.    // Use this for initialization
13.    void Start () {
14.        grimoireOn = false;
15.    }
16.
17.    // Update is called once per frame
18.    void Update () {
19.
20.    }
21.
22.    private void OnTriggerEnter(Collider collision)
23.    {
24.        if (collision.gameObject.tag == "TanganOculus")
25.        {
26.            if(grimoireOn)
27.            {
28.                if (PlayerPrefs.GetInt("Level") < 6)
29.                {
30.                    PlayerPrefs.SetInt("Level", 6);
31.                }
32.                magicCircle.GetComponent<ParticleSystem>().Pla
   y();
33.                magicCircle.GetComponent<AudioSource>().Play()
   ;
34.                Invoke("GameEnd", 3);
35.            }
36.        }
37.    }
38.
39.    private void GameEnd()
40.    {
41.        SceneManager.LoadScene("StartRoom");
```

```
    }

}
```

Lampiran 7.8 Implementasi menggunakan buku sihir



KUESIONER TUGAS AKHIR
5114100054 – Romi Yehekiel
5114100191 -Saddhana Daniswara

APLIKASI PUZZLE VIRTUAL REALITY MULTIPLAYER DENGAN OCULUS RIFT DAN KINECT

Identitas Responden Surabaya, 5 Juni 2018
Nama Lengkap : Michael E H
Pekerjaan : Mahasiswa
Usia : 19

A. KARAKTERISTIK RESPONDEN
Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (V)

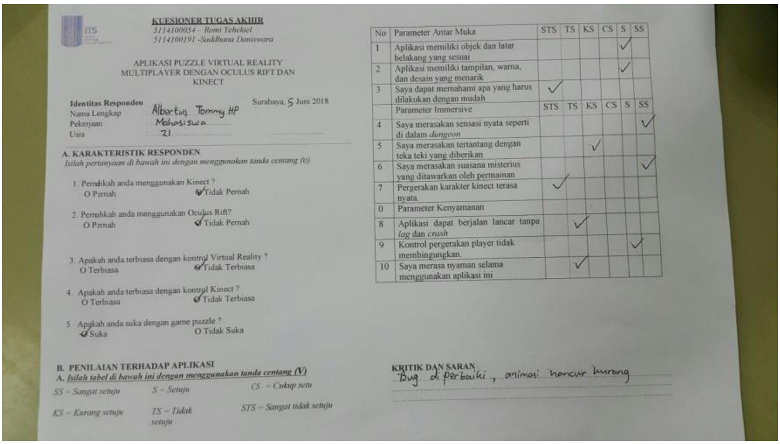
1. Pernahkah anda menggunakan Kinect ? ✓ Pernah O Tidak Pernah
2. Pernahkah anda menggunakan Oculus Rift? ✓ Pernah O Tidak Pernah
3. Apakah anda terbiasa dengan kontrol Virtual Reality ? ✓ Terbiasa O Tidak Terbiasa
4. Apakah anda terbiasa dengan kontrol Kinect ? ✓ Terbiasa O Tidak Terbiasa
5. Apakah anda suka dengan game puzzle ? ✓ Suka O Tidak Suka

B. PENILAIAN TERHADAP APLIKASI
A. *Isilah tabel di bawah ini dengan menggunakan tanda centang (V)*
SS = Sangat setuju *S = Setuju* *CS = Cukup setu*
KS = Kurang setuju *TS = Tidak* *STS = Sangat tidak setuju*

No	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1	Aplikasi memiliki objek dan latar belakang yang sesuai					✓	
2	Aplikasi memiliki tampilan, warna, dan desain yang menarik				✓		
3	Saya dapat memahami apa yang harus dilakukan dengan mudah						✓
	Parameter Immersive	STS	TS	KS	CS	S	SS
4	Saya merasakan sensasi nyata seperti di dalam *dungeon*					✓	
5	Saya merasakan tertantang dengan teka teki yang diberikan					✓	
6	Saya merasakan suasana misterius yang ditawarkan oleh permainan						✓
7	Pergerakan karakter kinect terasa nyata.	✓					
	Parameter Kenyamanan						
8	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*					✓	
9	Kontrol pergerakan player tidak membingungkan.					✓	
10	Saya merasa nyaman selama menggunakan aplikasi ini						✓

KRITIK DAN SARAN [illegible]

Lampiran 7.9 Kuesioner 1



KUESIONER TUGAS AKHIR
5114100054 – Romi Yehekiel
5114100191 -Saddhana Daniswara

APLIKASI PUZZLE VIRTUAL REALITY MULTIPLAYER DENGAN OCULUS RIFT DAN KINECT

Identitas Responden Surabaya, 5 Juni 2018
Nama Lengkap : Albertus Tommy HP
Pekerjaan : Mahasiswa
Usia : 21

A. KARAKTERISTIK RESPONDEN
Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (V)

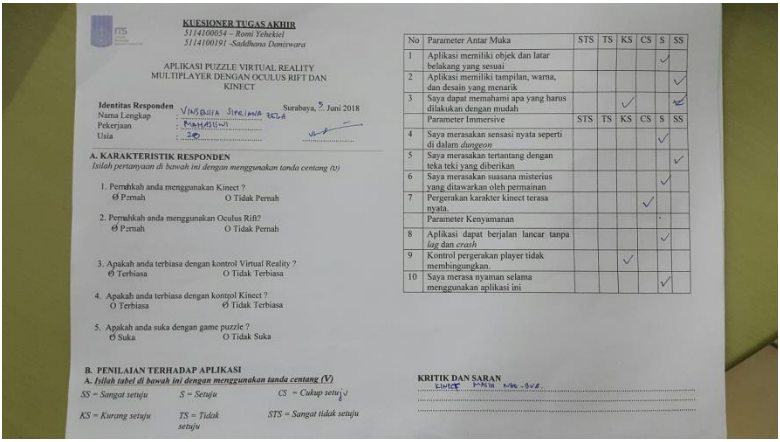
1. Pernahkah anda menggunakan Kinect ? O Pernah ✓ Tidak Pernah
2. Pernahkah anda menggunakan Oculus Rift? O Pernah ✓ Tidak Pernah
3. Apakah anda terbiasa dengan kontrol Virtual Reality ? O Terbiasa ✓ Tidak Terbiasa
4. Apakah anda terbiasa dengan kontrol Kinect ? O Terbiasa ✓ Tidak Terbiasa
5. Apakah anda suka dengan game puzzle ? ✓ Suka O Tidak Suka

B. PENILAIAN TERHADAP APLIKASI
A. *Isilah tabel di bawah ini dengan menggunakan tanda centang (V)*
SS = Sangat setuju *S = Setuju* *CS = Cukup setu*
KS = Kurang setuju *TS = Tidak setuju* *STS = Sangat tidak setuju*

No	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1	Aplikasi memiliki objek dan latar belakang yang sesuai					✓	
2	Aplikasi memiliki tampilan, warna, dan desain yang menarik					✓	
3	Saya dapat memahami apa yang harus dilakukan dengan mudah	✓					
	Parameter Immersive	STS	TS	KS	CS	S	SS
4	Saya merasakan sensasi nyata seperti di dalam *dungeon*						✓
5	Saya merasakan tertantang dengan teka teki yang diberikan			✓			
6	Saya merasakan suasana misterius yang ditawarkan oleh permainan						✓
7	Pergerakan karakter kinect terasa nyata.	✓					
0	Parameter Kenyamanan						
8	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*		✓				
9	Kontrol pergerakan player tidak membingungkan.					✓	
10	Saya merasa nyaman selama menggunakan aplikasi ini		✓				

KRITIK DAN SARAN Bug diperbaiki, animasi hancur kurang

Lampiran 7.10 Kuesioner 2



KUESIONER TUGAS AKHIR
5114100054 – Romi Yehekiel
5114100191 -Saddhana Daniswara

APLIKASI PUZZLE VIRTUAL REALITY MULTIPLAYER DENGAN OCULUS RIFT DAN KINECT

Identitas Responden Surabaya, 5 Juni 2018
Nama Lengkap : VINSENSIA SIPRIANA ZELA
Pekerjaan : MAHASISWI
Usia : 20

A. KARAKTERISTIK RESPONDEN
Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

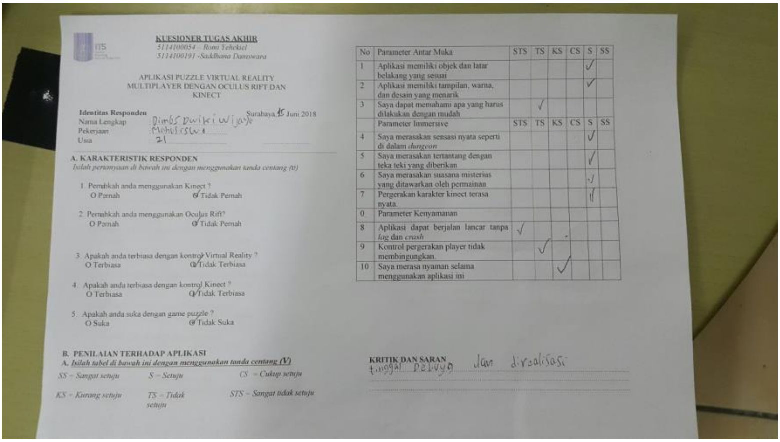
1. Pernahkah anda menggunakan Kinect ?
   Ø Pernah O Tidak Pernah
2. Pernahkah anda menggunakan Oculus Rift?
   Ø Pernah O Tidak Pernah
3. Apakah anda terbiasa dengan kontrol Virtual Reality ?
   Ø Terbiasa O Tidak Terbiasa
4. Apakah anda terbiasa dengan kontrol Kinect ?
   O Terbiasa Ø Tidak Terbiasa
5. Apakah anda suka dengan game puzzle ?
   Ø Suka O Tidak Suka

B. PENILAIAN TERHADAP APLIKASI
A. ***Isilah tabel di bawah ini dengan menggunakan tanda centang (V)***
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak setuju* *STS = Sangat tidak setuju*

No	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1	Aplikasi memiliki objek dan latar belakang yang sesuai					√	
2	Aplikasi memiliki tampilan, warna, dan desain yang menarik						√
3	Saya dapat memahami apa yang harus dilakukan dengan mudah			√			√
	Parameter Immersive	STS	TS	KS	CS	S	SS
4	Saya merasakan sensasi nyata seperti di dalam *dungeon*					√	
5	Saya merasakan tertantang dengan teka teki yang diberikan						√
6	Saya merasakan suasana misterius yang ditawarkan oleh permainan					√	
7	Pergerakan karakter kinect terasa nyata.				√		
	Parameter Kenyamanan						
8	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*					√	
9	Kontrol pergerakan player tidak membingungkan.			√			
10	Saya merasa nyaman selama menggunakan aplikasi ini					√	

KRITIK DAN SARAN
KINECT MASIH NGE-BUG.

Lampiran 7.11 Kuesioner 3



KUESIONER TUGAS AKHIR
5114100054 – Romi Yehekiel
5114100191 -Saddhana Daniswara

APLIKASI PUZZLE VIRTUAL REALITY MULTIPLAYER DENGAN OCULUS RIFT DAN KINECT

Identitas Responden Surabaya, 15 Juni 2018
Nama Lengkap Dimas Dwiki Wijaya
Pekerjaan Mahasiswa
Usia 21

A. KARAKTERISTIK RESPONDEN
Isilah pertanyaan di bawah ini dengan menggunakan tanda centang (v)

1. Pernahkah anda menggunakan Kinect ?
   O Pernah Ø Tidak Pernah
2. Pernahkah anda menggunakan Oculus Rift?
   O Pernah Ø Tidak Pernah
3. Apakah anda terbiasa dengan kontrol Virtual Reality ?
   O Terbiasa Ø Tidak Terbiasa
4. Apakah anda terbiasa dengan kontrol Kinect ?
   O Terbiasa Ø Tidak Terbiasa
5. Apakah anda suka dengan game puzzle ?
   O Suka Ø Tidak Suka

B. PENILAIAN TERHADAP APLIKASI
A. ***Isilah tabel di bawah ini dengan menggunakan tanda centang (V)***
SS = Sangat setuju *S = Setuju* *CS = Cukup setuju*
KS = Kurang setuju *TS = Tidak setuju* *STS = Sangat tidak setuju*

No	Parameter Antar Muka	STS	TS	KS	CS	S	SS
1	Aplikasi memiliki objek dan latar belakang yang sesuai					√	
2	Aplikasi memiliki tampilan, warna, dan desain yang menarik					√	
3	Saya dapat memahami apa yang harus dilakukan dengan mudah		√				
	Parameter Immersive	STS	TS	KS	CS	S	SS
4	Saya merasakan sensasi nyata seperti di dalam *dungeon*					√	
5	Saya merasakan tertantang dengan teka teki yang diberikan					√	
6	Saya merasakan suasana misterius yang ditawarkan oleh permainan					√	
7	Pergerakan karakter kinect terasa nyata.					√	
0	Parameter Kenyamanan						
8	Aplikasi dapat berjalan lancar tanpa *lag* dan *crash*	√					
9	Kontrol pergerakan player tidak membingungkan.		√				
10	Saya merasa nyaman selama menggunakan aplikasi ini			√			

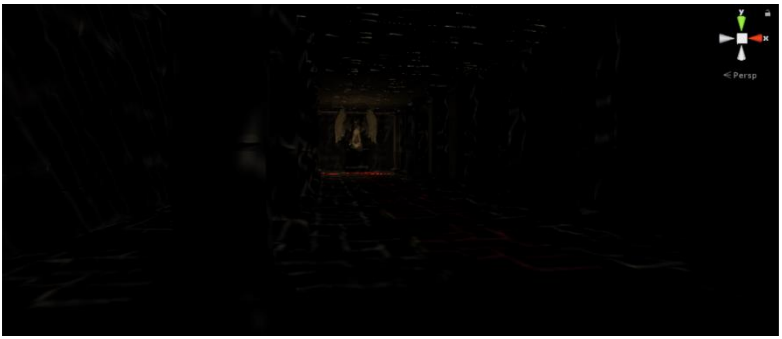
KRITIK DAN SARAN
tinggal peluya dan direalisasi

Lampiran 7.12 Kuesioner 4

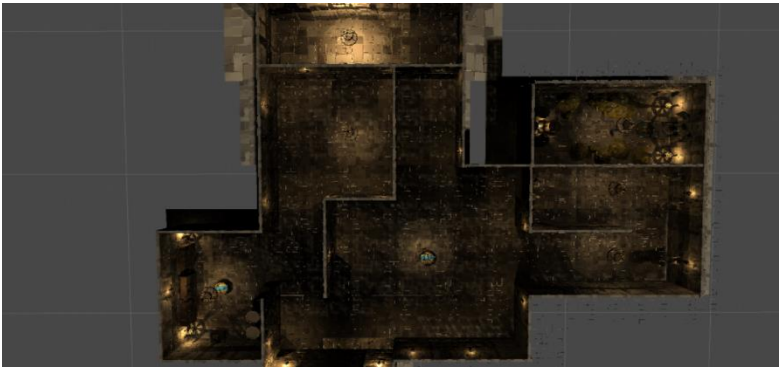
Lampiran 7.13 Level 1

Lampiran 7.14 Level 2

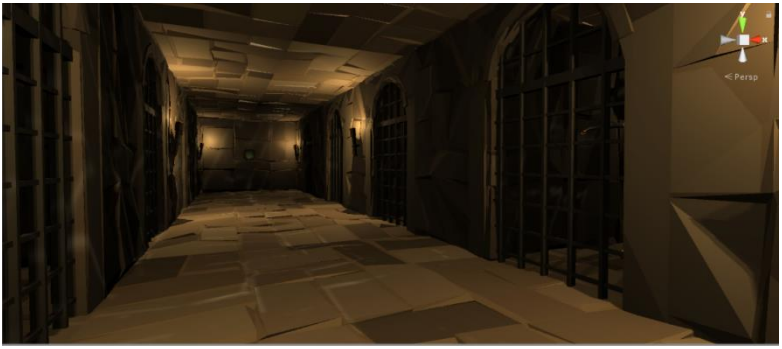
Lampiran 7.15 Level 3



Lampiran 7.16 Level 4



Lampiran 7.17 Level 5



Lampiran 7.18 Level 6